Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерних наук та інформаційних технологій

ДИПЛОМНА РОБОТА

МАГІСТРА

на тему *Інформаційна технологія автоматизованої генерації тестових завдань до навчальних матеріалів*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Виконав: студент *2* курсу, група *КНм-19-1* Підпис *О.В. Ковальчук* Ініціали, прізвище

Керівник: *ст. викладач кафедри КНІТ* Підпис *О.В. Мазурець* Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КНІТ* Підпис *Р.О. Багрій* Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КНІТ, д.т.н., професор Підпис *О.В. Бармак* Ініціали, прізвище

7 *12* 2020 р.

Хмельницький 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *програмування та комп'ютерних і телекомунікаційних систем*
Кафедра *комп'ютерних наук та інформаційних технологій*
Освітній ступінь *магістр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук та інформаційних технологій*

(підпис)
д.т.н., професор О.В. Бармак
«1» 09 *2020* року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ МАГІСТРА

1. Тема дипломної роботи магістра: *«Інформаційна технологія автоматизованої генерації тестових завдань до навчальних матеріалів»*

2. Завдання видано студенту *Ковальчуку Олексію Володимировичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *ст. викладач Мазурець Олександр Вікторович*
(прізвище, ім'я, по батькові)

4. Затверджені наказом університету від «9» 09 2020 р. № 12

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка інформаційної технології автоматизованої генерації тестових завдань до інформаційних навчальних матеріалів, в основі якої покладено метод автоматизованої генерації тестових завдань за допомогою продукційних правил. Розроблена інформаційна технологія повинна бути реалізована у вигляді програмного продукту для дослідження ефективності інформаційної технології. Також потрібно вдосконалити інформаційну модель семантичної структури навчального курсу для забезпечення можливості з достатньою для генерації тестових завдань інформативністю виконувати формальне подання навчальних матеріалів.

Реферат

Дипломна робота магістра присвячена розробці інформаційної технології для автоматизованої генерації тестових завдань із їх розподілом по семантичній структурі інформаційного навчального матеріалу. Семантична структура інформаційного навчального матеріалу містить систему рубрикації, множини ключових термінів до кожної із рубрик та тестові завдання для перевірки ключових термінів у межах рубрик навчальних матеріалів.

Актуальність теми. Основним завданням навчального процесу є цілеспрямована і планомірна підготовка майбутніх фахівців різного профілю до творчої життєдіяльності у сучасному суспільстві. Основним джерелом інформації в навчальному процесі слугують інформаційні навчальні матеріали. Завдяки інформаційним навчальним матеріалам відбувається передача накопичених теоретичних знань та досвіду студентам. Результат опрацювання такої інформації студентами під час навчальних занять підлягає перевірці на рівень засвоєних знань. Існує ряд різних методів для проведення такої перевірки, але найбільш сучасним та інноваційним є тестування.

Розвиток сучасних інформаційних технологій дозволяє проводити тестування без втручання викладача. Системи тестування самостійно формують перелік тестових завдань на основі банку запитань для проходження тесту, самостійно перевіряють результати тестування та визначають рекомендовану оцінку. На відміну від процесу тестування, створення запитань для наповнення банку питань залишається не автоматизованим процесом і викладач змушений формувати кожне запитання вручну. Завдання, створені в такий спосіб, мають високу якість, цінність та зрозумілість, але вагомим недоліком такого підходу є високі трудові затрати процесу створення тестових завдань. Оскільки на сьогоднійшній день відбувається швидкий розвиток різних сфер людської діяльності часто виникає потреба в оновлені та актуалізації навчальних матеріалів. У зв’язку із цим, існуючу множину тестових завдань потрібно переглядати, редагувати та додавати нові тестові завдання. Оскільки процес

створення тестових завдань не є автоматизованим, викладач змушений завжди це робити в ручному режимі. Також перспективною є можливість використовувати згенеровані тестові завдання для перевірки рівня знань визначених ключових термінів у межах визначених рубрик інформаційного навчально матеріалу. Тому розробка і впровадження інформаційних технологій для автоматизованої генерації тестових завдань, що розподілені за семантичною структурою навчального матеріалу, є актульним завданням.

Мета і задачі роботи. Метою дипломної роботи магістра є розробка інформаційної технології для автоматизованої генерації тестових завдань із їх розподілом по семантичній структурі інформаційного навчального матеріалу. Для досягнення поставленої мети потрібно вирішити наступні задачі:

1) провести аналіз сучасних методів автоматизації генерації тестів для визначення існуючих можливостей генерації тестових завдань із повним покриттям семантичної структури навчального матеріалу;
2) вдосконалити інформаційну модель семантичної структури навчального курсу для забезпечення можливості з достатньою для генерації тестових завдань інформативністю виконувати формальне подання навчальних матеріалів;
3) розробити метод автоматизованої генерації тестових завдань до навчальних матеріалів за допомогою правил продукції;
4) розробити інформаційну технологію автоматизованої генерації тестових завдань за допомогою отриманих моделі та методу;
5) розробити інформаційну систему генерації тестових завдань на основі розробленої інформаційної технології;
6) провести прикладне дослідження ефективності інформаційної технології генерації тестових завдань шляхом дослідження функціональності й ефективності застосування відповідної експериментальної інформаційної системи.

Об’єкт дослідження – процес автоматизованої генерації тестових завдань.

Предмет дослідження – моделі, методи та алгоритми автоматизованої генерації тестових завдань.

Методи дослідження. Для вирішення поставленого завдання генерації тестових завдань до інформаційних навчальних матеріалів використовується метод на основі правил продукції, які належать до продукційної моделі подання знань. В реалізації інформаційної технології використовуються методології проектування інформаційних систем та об'єктно-орієнтований підхід.

Наукова новизна одержаних результатів. Наукова новизна даної роботи полягає в наступному:

1. Вдосконалено інформаційну модель семантичної структури навчального курсу, яка відрізняється тим, що містить подання навчальних матеріалів у вигляді системи рубрикації, множин ключових термінів до кожної із рубрик та тестових завдань для перевірки ключових термінів у визначених місцях їх використання.

2. Розроблено новий метод автоматизованої генерації тестових завдань до навчальних матеріалів, що дозволяє генерувати тестові завдання за контентом навчального матеріалу на основі правил продукції.

3. Розроблено нову інформаційну технологію автоматизованої генерації тестових завдань, що дозволяє за поданням семантичної структури інформаційного навчального матеріалу одержувати множину тестових завдань, призначених для перевірки рівня знань визначених термінів.

4. Розроблено нову інформаційну систему генерації тестових завдань, що за створеною інформаційною технологією дозволяє генерувати та експортувати в середовище тестування множини тестових завдань.

Практичне значення одержаних результатів. Для дослідження ефективності інформаційної технології було реалізовано експериментальну інформаційну систему на основі розробленої інформаційної технології. Розроблена експериментальна інформаційна система виконує наступні функції: завантаження вхідних даних у вигляді семантичної структури інформаційного навчального матеріалу, зчитування продукційних правил, генерація тестових

завдань з однією правильною відповіддю, генерація тестових завдань із декількома правильними відповідями, генерація тестових завдань із можливістю введення відповіді, генерація тестових із варіантами відповіді «Так» або «Ні», редагування тестових завдань, експорт тестових завдань в середовище для тестування.

Використання інформаційної технології дозволяє автоматизовано генерувати тестові завдання, зв'язки між тестовим завданням і ключовими термінами та зв'язки між тестовими завданнями і контентом навчального матеріалу, що дозволяє використовувати згенеровані тестові завдання для адаптивної перевірки знань. Експериментальне тестування розробленої інформаційної системи підтвердило працездатність та ефективність інформаційної технології. За результатом використання експериментальної інформаційної системи вдалося встановити, що розроблена модель навчального курсу, метод та інформаційна технологія дозволяє отримати наступний ефект: покриття семантичної структури інформаційного навчального матеріалу множиною згенерованих тестових завдань складає 94.72% для рубрик, 74.87% для абзаців та 52.23% для речень; можливість використовувати згенеровані тестові завдання для перевірки рівня знань визначених ключових термінів у межах визначених рубрик ІНМ; можливість контролю використання тестових завдань у тестуванні залежно від обсягу наявних у контенті термінів різних рівнів семантичної значущості; можливість використання згенерованих множин тестових завдань для адаптивного тестування.

Проведене дослідження шляхів подальшого використання множини автоматизовано згенерованих тестових завдань встановило, що 24,38% тестових завдань користувачем було включено в кінцеву вибірку тестових завдань без редагування, 26,75% згенерованих тестових завдань було відредаговано та включено в кінцеву вибірку, а 48,87% видалено через їх надлишковість.

Апробація результатів дипломної роботи магістра та публікації.

Основні наукові положення роботи автором висвітлено в 5 наукових публікаціях [47, 48, 49, 50, 51]. Публікація 47 опублікована в фаховому виданні,

включеному в перелік МОН України. За виконання дипломної роботи магістра прийнято участь у науково-практичних конференціях: Міжнародній конференції молодих науковців «Сучасні технології в механіці – 2018» (19-21 квітня 2018 року); Всеукраїнській науково-практичній конференції молодих науковців і студентів «Інтелектуальний потенціал – 2018» (14-16 листопада 2018 року); XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» (14-15 листопада 2019 року); XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (9-10 листопада 2020 року).

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 51 найменувань та 6 додатків. Загальний обсяг дипломної роботи магістра становить 189 сторінок, з них 96 сторінка основного тексту та 93 сторінок додатків. У роботі наведено 36 рисунків та 26 таблиць.

Ключові слова: інформаційний навчальний матеріал, тестовий навчальний матеріал, тестові завдання, правила продукції, ключові терміни, інформаційна система.

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
БД	База даних
ІС	Інформаційна система
ІТ	Інформаційна технологія
ІНМ	Інформаційний навчальний матеріал
ТНМ	Тестовий навчальний матеріал
LMS	Learning Management Systems
MVVM	View-ViewModel-Model (патерн проектування)
WPF	Windows Presentation Foundation

Вступ

Дипломна робота магістра присвячена розробці інформаційної технології для автоматизованої генерації тестових завдань із їх розподілом по семантичній структурі інформаційного навчального матеріалу. Семантична структура інформаційного навчального матеріалу містить систему рубрикації, множини ключових термінів до кожної із рубрик та тестові завдання для перевірки ключових термінів у межах рубрик навчальних матеріалів.

Актуальність теми. Основним завданням навчального процесу є цілеспрямована і планомірна підготовка майбутніх фахівців різного профілю до творчої життєдіяльності у сучасному суспільстві. Основним джерелом інформації в навчальному процесі слугують інформаційні навчальні матеріали. Завдяки інформаційним навчальним матеріалам відбувається передача накопичених теоретичних знань та досвіду студентам. Результат опрацювання такої інформації студентами під час навчальних занять підлягає перевірці на рівень засвоєних знань. Існує ряд різних методів для проведення такої перевірки, але найбільш сучасним та інноваційним є тестування.

Розвиток сучасних інформаційних технологій дозволяє проводити тестування без втручання викладача. Системи тестування самостійно формують перелік тестових завдань на основі банку запитань для проходження тесту, самостійно перевіряють результати тестування та визначають рекомендовану оцінку. На відміну від процесу тестування, створення запитань для наповнення банку питань залишається не автоматизованим процесом і викладач змушений формувати кожне запитання вручну. Завдання, створені в такий спосіб, мають високу якість, цінність та зрозумілість, але вагомим недоліком такого підходу є високі трудові затрати процесу створення тестових завдань. Оскільки на сьогоднійшній день відбувається швидкий розвиток різних сфер людської діяльності часто виникає потреба в оновлені та актуалізації навчальних матеріалів. У зв’язку із цим, існуючу множину тестових завдань потрібно

переглядати, редагувати та додавати нові тестові завдання. Оскільки процес створення тестових завдань не є автоматизованим, викладач змушений завжди це робити в ручному режимі. Також перспективною є можливість використовувати створені тестові завдання для перевірки рівня знань визначених ключових термінів у межах визначених рубрик інформаційного навчально матеріалу. Тому розробка і впровадження інформаційних технологій для автоматизованої генерації тестових завдань, що розподілені за семантичною структурою навчального матеріалу, є актульним завданням.

Мета і задачі роботи. Метою дипломної роботи магістра є розробка інформаційної технології для автоматизованої генерації тестових завдань із їх розподілом по семантичній структурі інформаційного навчального матеріалу. Для досягнення поставленої мети потрібно вирішити наступні задачі:

1) провести аналіз сучасних методів автоматизації генерації тестів для визначення існуючих можливостей генерації тестових завдань із повним покриттям семантичної структури навчального матеріалу;

2) вдосконалити інформаційну модель семантичної структури навчального курсу для забезпечення можливості з достатньою для генерації тестових завдань інформативністю виконувати формальне подання навчальних матеріалів;

3) розробити метод автоматизованої генерації тестових завдань до навчальних матеріалів за допомогою правил продукції;

4) розробити інформаційну технологію автоматизованої генерації тестових завдань за допомогою отриманих моделі та методу;

5) розробити інформаційну систему генерації тестових завдань на основі розробленої інформаційної технології;

6) провести прикладне дослідження ефективності інформаційної технології генерації тестових завдань шляхом дослідження функціональності й ефективності застосування відповідної експериментальної інформаційної системи.

Об’єкт дослідження – процес автоматизованої генерації тестових завдань.

Предмет дослідження – моделі, методи та алгоритми автоматизованої генерації тестових завдань.

Методи дослідження. Для вирішення поставленого завдання генерації тестових завдань до інформаційних навчальних матеріалів використовується метод на основі правил продукції, які належать до продукційної моделі подання знань. В реалізації інформаційної технології використовуються методології проектування інформаційних систем та об’єктно-орієнтований підхід.

Наукова новизна одержаних результатів. Наукова новизна даної роботи полягає в наступному:

1. Вдосконалено інформаційну модель семантичної структури навчального курсу, яка відрізняється тим, що містить подання навчальних матеріалів у вигляді системи рубрикації, множин ключових термінів до кожної із рубрик та тестових завдань для перевірки ключових термінів у визначених місцях їх використання.

2. Розроблено новий метод автоматизованої генерації тестових завдань до навчальних матеріалів, що дозволяє генерувати тестові завдання за контентом навчального матеріалу на основі правил продукції.

3. Розроблено нову інформаційну технологію автоматизованої генерації тестових завдань, що дозволяє за поданням семантичної структури інформаційного навчального матеріалу одержувати множину тестових завдань, призначених для перевірки рівня знань визначених термінів.

4. Розроблено нову інформаційну систему генерації тестових завдань, що за створеною інформаційною технологією дозволяє генерувати та експортувати в середовище тестування множини тестових завдань.

Практичне значення одержаних результатів. Для дослідження ефективності інформаційної технології було реалізовано експериментальну інформаційну систему на основі розробленої інформаційної технології.

Розроблена експериментальна інформаційна система виконує наступні функції: завантаження вхідних даних у вигляді семантичної структури інформаційного навчального матеріалу, зчитування продукційних правил, генерація тестових завдань з однією правильною відповіддю, генерація тестових завдань із декількома правильними відповідями, генерація тестових завдань із можливістю введення відповіді, генерація тестових із варіантами відповіді «Так» або «Ні», редагування тестових завдань, експорт тестових завдань в середовище для тестування.

Використання інформаційної технології дозволяє автоматизовано генерувати тестові завдання, зв’язки між тестовим завданням і ключовими термінами та зв’язки між тестовими завданнями і контентом навчального матеріалу, що дозволяє використовувати згенеровані тестові завдання для адаптивної перевірки знань. Експериментальне тестування розробленої інформаційної системи підтвердило працездатність та ефективність інформаційної технології. За результатом використання експериментальної інформаційної системи вдалося встановити, що розроблена модель навчального курсу, метод та інформаційна технологія дозволяє отримати наступний ефект: покриття семантичної структури інформаційного навчального матеріалу множиною згенерованих тестових завдань складає 94.72% для рубрик, 74.87% для абзаців та 52.23% для речень; можливість використовувати згенеровані тестові завдання для перевірки рівня знань визначених ключових термінів у межах визначених рубрик ІНМ; можливість контролю використання тестових завдань у тестуванні залежно від обсягу наявних у контенті термінів різних рівнів семантичної значущості; можливість використання згенерованих множин тестових завдань для адаптивного тестування.

Проведене дослідження шляхів подальшого використання множини автоматизовано згенерованих тестових завдань встановило, що 24,38% тестових завдань користувачем було включено в кінцеву вибірку тестових завдань без

редагування, 26,75% згенерованих тестових завдань було відредаговано та включено в кінцеву вибірку, а 48,87% видалено через їх надлишковість.

Апробація результатів дипломної роботи магістра та публікації. Основні наукові положення роботи автором висвітлено в 5 наукових публікаціях [47, 48, 49, 50, 51]. Публікація 47 опублікована в фаховому виданні, включеному в перелік МОН України. За виконання дипломної роботи магістра прийнято участь у науково-практичних конференціях: Міжнародній конференції молодих науковців «Сучасні технології в механіці – 2018» (19-21 квітня 2018 року); Всеукраїнській науково-практичній конференції молодих науковців і студентів «Інтелектуальний потенціал – 2018» (14-16 листопада 2018 року); XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» (14-15 листопада 2019 року); XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (9-10 листопада 2020 року).

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 51 найменувань та 6 додатків. Загальний обсяг дипломної роботи магістра становить 189 сторінок, з них 96 сторінка основного тексту та 93 сторінок додатків. У роботі наведено 36 рисунків та 26 таблиць.

Розділ 1

Аналіз сучасного стану області автоматизованої генерації тестових завдань

1.1 Використання подання семантичної структури навчального курсу для вирішення актуальних задач комп’ютерного тестування

Навчальним процесом прийнято вважати систему організації навчально-виховної діяльності, яка за свою основу має взаємозв'язок процесів викладання і вивчення, що спрямовані на досягнення поставлений навчальних цілей для підготовки особистості до професійної діяльності. В роботі [1] процес навчання визначається як цілеспрямована взаємодія вчителя та учнів, у ході якої розв'язуються завдання освіти, виховання і загального розвитку особистості.

Під час навчального процесу відбувається передача накопиченого попередніми поколіннями наукового та соціального досвіду і його перетворення і процесі плину часу. Накопичена інформація вміщується в різноманітних джерелах таких як: підручниках, посібниках, збірниках вправ і задач, довідниках, журналах, довідникових матеріалах, інструкціях, наочності тощо. Наведені джерела узагальнюються одним терміном – «навчальні матеріали». У ряді випадків [2] навчальний матеріал визначається як складна система із специфічними елементами і зв’язками, які їх об’єднують.

Навчальна інформація, яка подається в навчальних матеріалах, буває основною та допоміжною [2]. Основною інформацією вважається така інформація, яка повинна перетворитися в знання або вміння. Допоміжна інформація – інформація, яка відповідає за забезпечення надійності засвоєння основної інформації. В залежності від функцій, які виконують навчальні матеріали, їх можна розділити на групи, як показано на рисунку 1.1.

Таким чином, навчальний курс включає в себе інформаційні, актуалізуючі, стимулюючі, контролюючі, операційні та діагностичні начальні матеріали створені для організації групового або індивідуального навчання. Із

перерахованих елементів навчального курсу до носіїв основної інформації відносять інформаційний, операційний і частково контролюючий навчальний матеріал. Інформаційний вид є найбільш важливим та об’ємним оскільки він відповідає за подання навчальної інформації у вигляді тексту, малюнків, схем, формул тощо. Саме із ІНМ студенти одержують основні теоретичні знання у навчальному курсі.

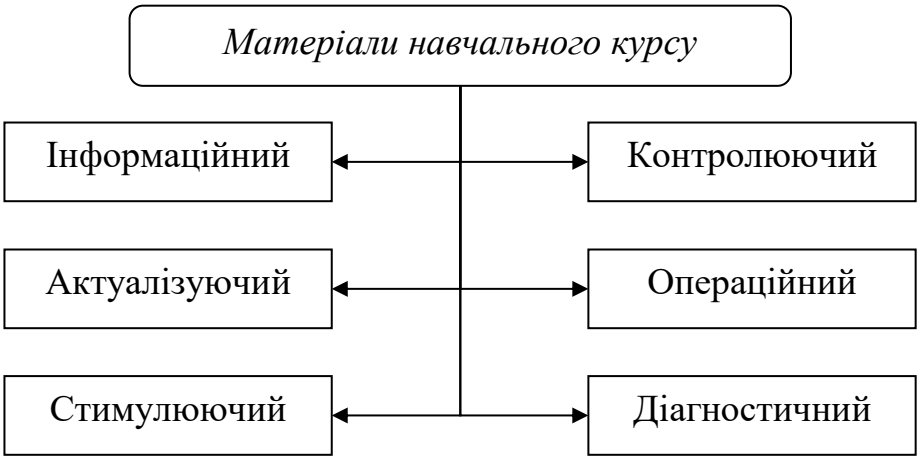


Рисунок 1.1 – Види навчальних матеріалів [2]

Зазвичай ІНМ має ієрархічну структуру подання, що відповідає рубрикації текстового документу. Рубрикацією прийнято вважати систему зв’язків логічних блоків текстового документу, які відображають їх підпорядкування і призначена для структуризації та систематизації контенту [3, 4]. Атомарною одиницею в системі рубрикації є рубрика. Рубрикою є виокремлена заголовком частина тексту, який поєднує спільна проблема, тема, жанр [5]. Семантичний вміст контенту кожної рубрики може бути поданий згорнуто у вигляді реферату, анотації, множини ключових термінів тощо.

Рефератом є короткий переказ великого обсягу первинної інформації (книги, статті, вчення і т.д.) із збереженням його основного смислового змісту. Анотація створюється в результаті аналітично-синтетичного опрацювання великого об’єму інформації, метою якого є отримання короткої інформації, що розкриває логічну структуру і зміст обробленої інформації. В порівняні із

рефератом, анотація наводить лише короткий перелік запитання, що розглядаються в первинній інформації. В той час як рефераті викладається суть питань та найважливіші висновки [6].

ІНМ для пояснення потрібної інформації завжди оперує набором певних термінів. Терміном є слово або словосполучення, що означає чітко окреслене спеціальне поняття якої-небуть галузі, техніки, мистецтва, суспільного життя тощо [7]. Серед усіх термінів, які використовується в ІНМ, є частина таких, що відіграють ключову роль в контексті теми ІНМ або її рубрики і використовуються для вираження аспекту змісту тексту та мають істотне смислове навантаження [8]. Набір ключових термінів в тексті формує його семантичне ядро, що повністю відображає тему та предмет тексту [9].

Отже, важливим елементом подання ІНМ є наявність ключових термінів для кожного розділу (рубрики) документу, які виражають смислове їх навантаження. Тому для перевірки рівня одержаних знань слід керуватися семантичною структурою ІНМ, яка містить систему рубрикації та множини ключових термінів кожної із рубрик.

1.2 Комп’ютерне тестування та його особливості

Операційний, контролюючий та діагностуючий навчальний матеріали виконають допоміжну роль, сприяючи якісному, міцному й більш оперативному засвоєнню основної інформації. Діагностуючий навчальний матеріал має найбільшу відповідальність серед перерахованих видів, оскільки він в більшій мірі робить контроль отриманих знань особи, що навчається.

До основних форм контролю знань відносять [10]:

– письмовий контроль – письмова, розгорнута відповідь на поставлені запитання;

– усний контроль – контроль під час якого відбувається опитування особи, що навчається в усній формі;

– тестування – контроль знань за допомогою тестових завдань;

– графічний – контроль полягає у створенні особою, що навчається, узагальненої наочної моделі, для відображення взаємозв'язків, відношення певних об'єктів або сукупності об’єктів.

– програмований контроль – відбувається за допомогою надання учням, студентам стандартних вимог, що забезпечується за допомогою використанням однакових запитань. Особливістю даного контролю є програмна перевірка відповідей та розрахунок оцінки.

– практична перевірка – здійснюється для перевірки практичних професійних навичок особи.

Серед перерахованих форм контролю найбільш інноваційною є комп’ютерне тестування. Дана форма надзвичайно популярна за кордоном і активно впроваджується Україні. В основі тестування знань покладено тест, як систему формалізованих завдань призначених для встановлення освітнього (кваліфікаційного) рівня особи [7]. Атомарним елементом тесту є тестове завдання. В залежності від типу тестового завдання, воно може мати різну будову, але в більшості випадків складаються із двох частин [11]:

– умови, яка описує проблему та робить постановку завдання для особи яка проходить тест;

– списку варіантів відповідей, серед яких є правильні відповіді та дистрактори (неправильні відповіді).

За рівнем складності тестові завдання поділяють на 4 рівні [13]:

– І рівень включає тестові завдання призначенні для перевірки якості засвоєння навчального матеріалу на «розпізнавальному» рівні. До таких тестових завдань належать завдання розпізнавання чи розрізнення, класифікацію об’єктів, явищ і понять;

– ІІ рівень містить тестові завдання для перевірки якості засвоєння на рівні репродуктивної діяльності, де особа, що проходить тест здатна самостійно відтворювати засвоєні знання та навички;

– III рівень містить завдання для вирішення, яких потрібно застосувати певний перелік дій (алгоритм) для отримання кінцевого результату;

– IV рівень включає тестові завдання високого рівня для вирішення яких потрібно застосування аналізу поставленої проблеми, вміння застосування різних методів розв'язку задач та вміння приймати рішення в проблемних, непередбачуваних ситуаціях.

Тестові завдання мають різні форми [12], перелік яких наведений на рисунку 1.2.

Рисунок 1.2 – Форми тестових завдань [12]

На відміну від інших форм контролю, тестування відрізняється тим, що на його результат не впливає суб'єктивна думка людини, яка перевіряє. Також тестування має наступні переваги:

– упродовж досить обмеженого часу може бути перевірена якість знань, навичок у зазначеної кількості осіб;

– можливий контроль знань, умінь, навичок на необхідному, заздалегідь запланованому рівні;

– увага особи, яка проходить тестування фіксується не на формуванні відповіді, а не осмисленні її суті;

– тестування ставить усіх осіб, які проходять тестування в однакові умови в процесі контролю, оцінювання;

– використання тестів це більш всеохоплюючий інструмент, оскільки в процесі тестування можуть бути перевірені теми з усього курсу, в той час коли на усний екзамен виноситься до 4 запитань, а на письмовий до 5.

Як і будь-яка інша форма контролю, метод тестування також має свої недоліки. До них належать:

– ймовірність випадкового вибору правильної відповіді;

– можливість при застосуванні тестів закритого типу оцінки тільки кінцевий результат (правильно - неправильно), у той час як сам процес, що привів до нього, не розкривається;

– стандартизація мислення без врахування рівня розвитку особистості;

– велика затрата часу на складання необхідного "банку" тестових завдань, їх варіантів, трудомісткість процесу;

– тести не сприяють розвитку мови.

Не зважаючи на перераховані недоліки, популярність перевірки знань за допомогою тестування продовжує зростати. Також у зв’язку із розвитком інформаційних технологій, процес тестування є доволі автоматизованим і ще простішим в проведені. Автоматизовані системи тестування генерують різні варіанти переліків тестових завдань на основі банку тестових завдань, проводять тестування, оцінюють отримані результати тощо. Процес тестування відбувається за допомогою самостійного діалогу особи, яка проходить

тестування, з комп’ютером. Як результат такого діалогу викладач отримує результати тестування і не змушений робити додаткових перевірок.

Не автоматизованим залишається процес формування тестових завдань для формування банку тестів. Ця задача повністю покладається на викладача. Виконання такої роботи є дуже рутинною і затратною по часу роботою. Як результат, не завжди тести створені людиною, покривають весь навчальний матеріал, а такий фактор особливо небезпечний, якщо це сфера охорони здоров’я, безпеки тощо. Тому можна зробити висновок, що задача автоматизованої генерації тестових завдань є актуальною, причому згенерований тест має покривати всю семантичну структуру навчального матеріалу.

1.3 Аналіз сучасних методів автоматизації генерації тестів

На сьогоднішній день існують методи та інформаційні технології автоматизованої генерації тестових завдань. Серед відомих засобів автоматизації генерації тестових завдань необхідно відзначити метод генерації параметризованих задач, метод генерації тестових завдань за понятійно-тезовою моделлю й метод генерації тестових завдань за формалізацією структурованих текстових тверджень. Також даному напрямку присвячено ряд наукових публікації, серед яких варто виділити роботи таких науковців як Титенко С. В. [14], Пасічника Р. М. та Мельника А. М. [15, 16], Попов А. М. [17].

В основі понятійно-тезової моделі [18] покладено подання, за яким навчальний матеріал складається з семантичних одиниць, що знаходяться на різних рівнях. В такому випадку навчальний матеріал розбивається на декілька фрагментів. Кожен фрагмент містить тези, які описуються поняття. Під поняттям розуміється одне-два слова, які описуються предмет розгляду в навчальному матеріалі. Процес формування бази знань являє собою зосереджене, осмислене читання навчального тексту. У випадку коли в тексті зустрічається ключовий термін (поняття) – воно додається в базу знань. В такий самий спосіб додаються

в базу знань тези. Особливо важливим моментом є те, що потрібно вказати для якого поняття обрана теза.

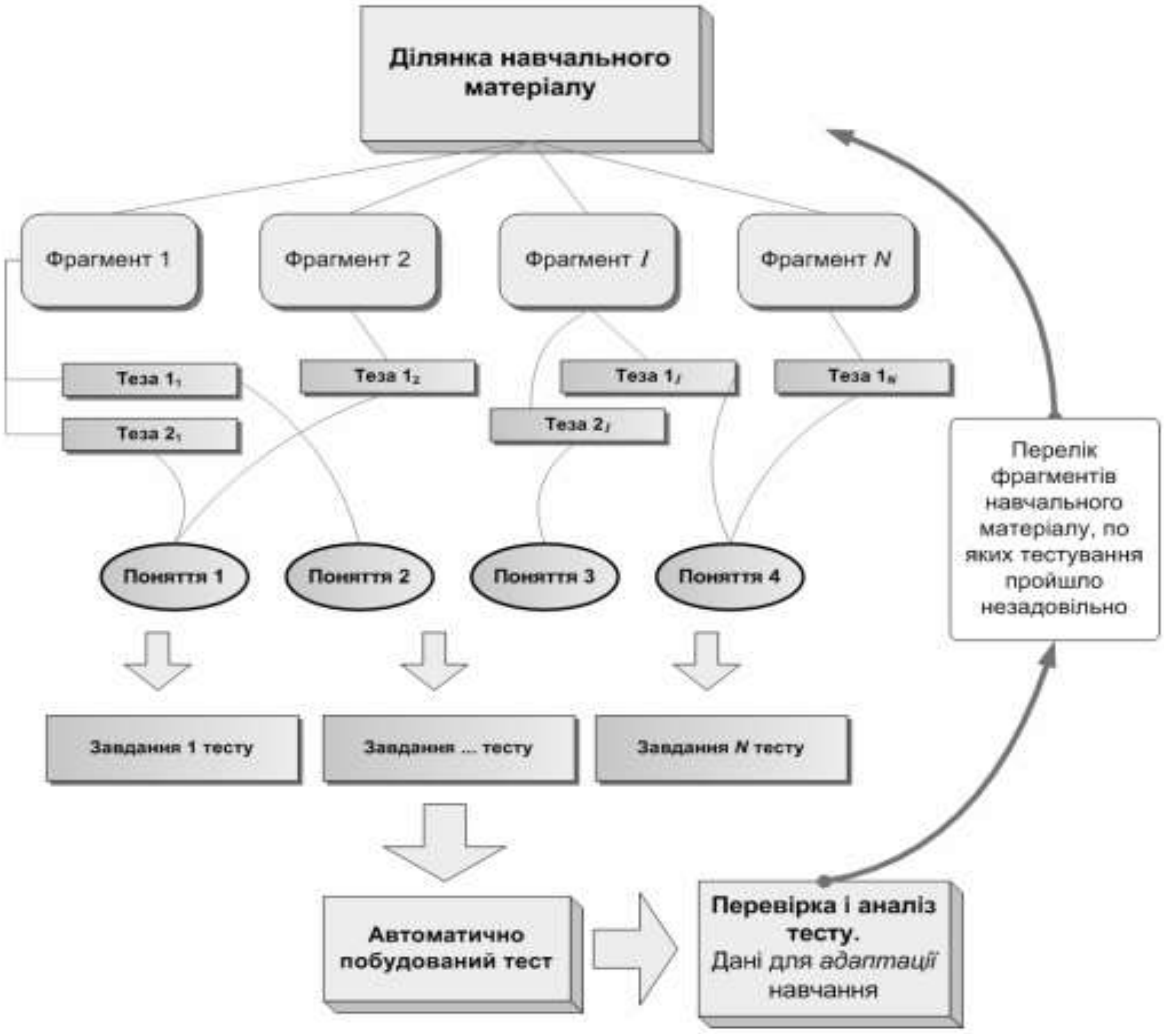


Рисунок 1.3 – Схема використання понятійно-тезисного підходу для формування тестового завдання [19]

Формування тестових завдань відбувається базуючись на семантичних даних понятійно-тезисної бази знань (рис. 1.3). Основним вхідним параметром є вказівка на ділянку навчального матеріалу по якій іде тестування. Першим кроком є визначення кола семантичних даних на основі інформації про вказану навчальну ділянку для побудови на їх основі засобів контролю [18, 19]. Це вдається виконати за допомогою сформованого ланцюга зв’язків «ділянка

навчального матеріалу – кадри, що містяться у цій ділянці – тези кадрів – поняття» при заповнені бази знань.

До недоліків методу можна віднести:

1) надмірним є обмеження структури тесту рамками понять та тезисів, оскільки в практиці формування тестових завдань часто зустрічаються «тезисно-тезисні» конструкції доволі складної структури;

2) неструктурованість подання тез в даному методі обмежує складність тестового завдання, оскільки не проводиться аналіз окремих частин самої тези.

Одним із перспективних і порівняно нескладним у реалізації є метод параметризованих тестів [20]. Даний метод полягає у формуванні набору тестових завдань на основі шаблонів в якому змінюється певний параметр на автоматично згенероване значення за допомогою певної формули або алгоритму. Таким чином може бути згенерована велика кількість однотипних завдань, які при тестуванні будуть індивідуальними для кожної особи, яка проходить тестування. На рисунку 1.4 наведено приклад параметризованого тестового завдання в якому a – змінний параметр.

Яким буде розв’язок квадратного рівняння $ax^2 + 5x + 15 = 0$?

Рисунок 1.4 – Приклад параметризованого тестового завдання

Недоліком параметризованих тестів підходу є його вузька предметна спрямованість, тому що параметризовані тести добре підходять для перевірки практичних навичок у точних науках, але не можуть бути використанні для перевірки теоретичних знань [20]. Відповідно це не позволяє проводити контроль знань в гуманітарних науках.

Відомий метод генерації тестових завдань на основі семантичних мереж [20], який передбачає генерацію тестових завдань на основі бази знань, яку складає експерт. Структурною одиницею бази знань є ланцюг «поняття –

зв’язок – поняття». Експерт з предметної області повинен заповнити базу знань, а створення тестів проходить автоматично шляхом опущення однієї із частин ланцюга. Недоліком даного методу є великі трудові затрати на формування бази знань, потреба в залучені експерта по предметній області та інженера по знанням. Семантичні мережі, так як і інші моделі знань, застосовуються для класичних задач штучного інтелекту, погано підходять для навчальних цілей і у випадку для формування якісних тестових завдань і як результат часто створюються завдання важкі для сприйняття людиною. Семантичні мережі створювались скоріш для «пояснення» знань комп’ютеру, а не людині.

Також для вирішення проблеми генерації тестових завдань використовують технологію автоматизації процесу створення тестових завдань на основі онтології навчального курсу [21]. На вхід технологія приймає онтологію навчального курсу в цілому або окремі його розділи, які зберігаються в базі знань у вигляді формалізованих машинних представлень. Результатом роботи технології є набір тестових завдань різних типів, та різного рівня складності. Для машинного подання онтологічних значень пропонується використовувати уніфіковану фреймову структуру. Перевагою даної технології є те, що можна отримати тестові завдання різного типу і для будь-якої предметної області. Недоліком є те, що технологія вимагає формування бази знань для кожної предметної області, що є досить важкою задачею і вимагає залучення експертів відповідних областей.

В результаті проведеного дослідження (Додаток А) було встановлено, що існуючі на сучасному етапі підходи до генерації тестових завдань мають характерні недоліки, які полягають у відсутності зв’язку створених тестових завдань із елементами семантичної структури ІНМ. Це має наслідком нерівномірність покриттям тестом навчального матеріалу на етапі перевірки рівня знань. Тому є актуальною розробка інформаційної технології генерації тестових завдань, множина яких покриває всю семантичну структуру

навчального матеріалу в вигляді системи рубрикації та множин ключових термінів кожної із рубрик.

1.4 Аналіз сучасних електронних навчальних середовищ

Процес навчання, з кожним роком, стає все більше автоматизованим. Найбільше цьому сприяє розвиток систем управління навчанням, також відомих в англомовних літературних джерелах як LMS. LMS являється прикладні програмні продукти для адміністрування, документування, звітності, тестування та проведення освітніх курсів, навчальних або дослідницьких програм [22]. Найбільш поширеними вимогами до LMS є:

– можливість створення навчальних курсів;

– можливість створення окремих класів, груп;

– наявність форумів для комунікації студентів курсу із викладачем та між собою;

– проведення підсумкового контролю знань (самостійні роботи, тестування);

– масштабованість (здатність системи до розширення і збільшення обсягів оброблюваної інформації);

– мобільність;

– зрозумілий інтерфейс.

Більшість сучасних наявних LMS задовольняють перерахованим вимогам, але не всі з них є доступними для більшості навчальних закладів через високу вартість або складність у розгортанні такої системи. До найбільш використовуваних LMS належать Moodle [23], eLearning Server 3000 [24, 25], ATutor [26], Blackboard [27].

Moodle є безплатною платформою з відкритим кодом, та орієнтована насамперед на організацію взаємодії між викладачем та учнями, хоча підходить і для організації традиційних дистанційних курсів, а також підтримки очного,

стаціонарного навчання. Moodle надає можливість завантажувати контент різних форматів, для подальшого його використання студентами та викладачами [23]. В системі можливий перегляд практично всіх електронних файлів, що є дуже важливим при створенні курсів. Moodle надає інструменти комунікації між викладачами та студентами, а також, безпосередньо, студентам один з одним. Для комунікації доступні форуми, блоги, чати та приватні повідомлення.

Важливою частиною є те, що система Moodle є доволі гнучка у налаштуваннях [23]. Викладач може контролювати доступ до своїх курсів, виставляти оцінки, контролювати виконання завдань. Базовий функціонал можна легко розширити за допомогою плагінів, які можна безкоштовно завантажити із Інтернету або створити самому. Також доступна інтеграція інших сервісів. Наприклад, він легко об’єднується з WordPress або вебінарами Zoom.

Оскільки основною формою контролю знань у дистанційному навчанні є тестування, в Moodle є великий інструментарій для створення тестів і проведення контрольного тестування. Moodle дозволяє створити всіх найбільш поширених типів, а саме:

- множинного вибору;
- одиночного вибору;
- доповнення;
- логічний тест (Так/Ні);
- співвідношення.

Система eLearning Server 3000 має модульну структуру, що дозволяє розширяти та адаптувати її під власні вимоги [24]. Надається можливість створення навчальних курсів і реєстрації на них викладачів та студентів. До навчального курсу можна створювати електронній відомості, журнали, розклади занять. Інтерактивні заняття (лекції, семінари, лабораторні роботи) проходять із використанням таких засобів як відео трансляції, електрона дошка для малювання, форуми, чати. В системі eLearning Server є розвинута підсистема тестування, що дозволяє створити і провести тест будь-якої складності. В

системі доступні такі ж самі типи тестів як і в системі Moodle [25]. Тестова підсистема зберігає всі питання тестових завдань у базі даних.

Система ATutor є простим у встановленні, налаштуванні та підтримці для системних адміністраторів [26]. Викладачі можуть досить легко створювати та переносити навчальні матеріали та запускати свої онлайн-курси. Доступна можливість для викладачів керувати дозволами на запис на курс, переглядати студентів, які записані на курс. Відповідно це дозволяє створювати навчальні групи. Оскільки система є модульна, тобто складається з окремих функціональних одиниць – модулів, то вона відкрита для модернізації і розширення функціональних можливостей. Щодо тестування, то система надає широкі можливості для створення і керування тестами, організовувати бази даних питань курсу, попередній перегляд тестів, перегляд спроб складання тестів студентами, перегляд статистики по тестах. До недоліків системи можна віднести слабо розвинену систему звітності.

Отже, проведений аналіз систем управління навчанням показав, що всі системи здатні ефективно працювати із тестовими завданнями. Модульне об’єктно-орієнтоване середовище Moodle є найбільш поширеним, практичним, доступним та зручним у використання, тому його варто розглядати як засіб використання автоматизовано згенерованих тестових завдань.

1.5 Постановка задачі

Метою дипломної роботи магістра є розробка інформаційної технології для автоматизованої генерації тестових завдань із їх розподілом по семантичній структурі ІНМ. Семантична структура ІНМ містить систему рубрикації, множини ключових термінів до кожної із рубрик та тестові завдання для перевірки ключових термінів у межах рубрик навчальних матеріалів.

Під час дослідження повинні бути вирішенні наступні задачі:

1) вдосконалити інформаційну модель семантичної структури навчального курсу для забезпечення можливості з достатньою для генерації тестових завдань інформативністю виконувати формальне подання навчальних матеріалів;

2) розробити метод автоматизованої генерації тестових завдань до навчальних матеріалів за допомогою правил продукції;

3) розробити інформаційну технологію автоматизованої генерації тестових завдань за допомогою отриманих моделі та методу;

4) розробити інформаційну систему генерації тестових завдань на основі розробленої інформаційної технології;

5) провести прикладне дослідження ефективності інформаційної технології генерації тестових завдань шляхом дослідження функціональності й ефективності застосування відповідної експериментальної інформаційної системи.

Висновки до розділу 1

В розділі було проведено аналіз предметної області на предмет використання технології комп’ютерного тестування для перевірки рівня знань особи, яка навчається. Також проведено аналіз наукових статей та публікацій, присвячених проблемі автоматизованої генерації тестових завдань. Існуючі розроблені методи та технології генерації тестових завдань не дозволяють генерувати тестові завдання різних типів із повним покриттям семантичної структури навчального матеріалу. Також існуючі на сучасному етапі підходи до генерації тестових завдань мають характерні недоліки, які полягають у відсутності зв’язку створених тестових завдань із елементами семантичної структури ІНМ. Тому розробка інформаційної технології, яка буде вирішувати ці проблеми, визнана актуальною.

Було проведено аналіз сучасних систем управління навчанням для визначення їхніх переваг та можливостей. В результаті аналізу, для роботи було обрано систему Moodle.

На основі проведених досліджень було сформовано мету та завдання, які потрібно вирішити в межах виконання даної магістерської дипломної роботи. Метою дипломної роботи магістра визначено розробку інформаційної технології для автоматизованої генерації тестових завдань із повним покриттям семантичної структури навчального матеріалу.

Розділ 2

Розробка інформаційної технології автоматизованої генерації тестових завдань

2.1 Інформаційна модель семантичної структури навчального курсу

Як було описано в розділі 1.1, в процесі навчання ІНМ в більшості випадків подається у формі підручника, навчального посібника чи конспекту лекцій. Він являється основним джерелом інформації в навчальному курсі. Найбільш звичною його структурою є система тем, заголовків та підзаголовків, оскільки така структура допомагає автору подати матеріал в логічно зв'язаних структурних блоках. Інформаційною моделлю такого подання ІНМ є ієрархічна структура заголовків та зв'язків між ними (рис 2.1):

$$H \cup R_1, \tag{2.1}$$

де H – множина заголовків, R_1 – множина зв'язків між заголовками.

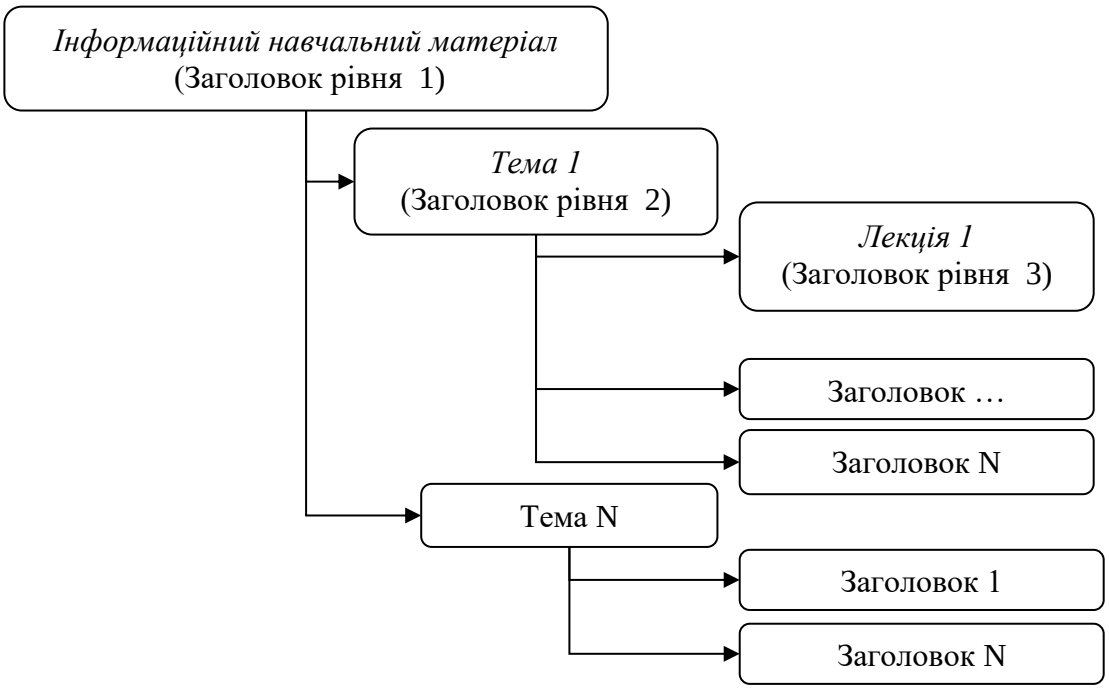


Рисунок 2.1 – Приклад ієрархічного подання рубрик ІНМ

Важливим елементом ІНМ є наявність ключових термінів для кожного розділу документу, які відображають його семантичний вміст. Появи ключових термінів у окремих реченнях ІНМ дозволяють генерувати такі тестові завдання, які покривають елементи семантичної структури ІНМ. В такому випадку ІНМ можна подати у вигляді:

$$I = H \cup R_1 \cup K \cup R_2 \cup S \cup R_3, \quad (2.2)$$

де I – ІНМ, S – множина речень, K – множина ключових термінів, R_2 – множина зв'язків між ключовими термінами та реченнями, R_3 – множина зв'язків між заголовками та реченнями.

ТНМ є найбільш розповсюдженим різновидом навчального матеріалу, який призначений для визначення рівня засвоєння ІНМ шляхом виконання тестування особи, що вивчає навчальний курс. ТНМ можна подати у вигляді:

$$T = Q \cup R_4 \cup R_5, \quad (2.3)$$

де T – ТНМ, Q – множина тестових завдань, R_4 – множина зв'язків між тестовим завданнями та реченнями, R_5 – множина зв'язків між тестовим завданнями та ключовими термінами.

Сукупність множини ТНМ і ІНМ формують структуру навчального курсу, яка подається у вигляді (Додаток Б):

$$C = I \cup T \quad (2.4)$$

де C – навчальний курс.

Згідно із формул (2.2), (2.3) та (2.4) структуру навчального курсу можна записати у вигляді (рисунки 2.2, 2.3):

$$C = H \cup R_1 \cup K \cup R_2 \cup S \cup R_3 \cup Q \cup R_4 \cup R_5. \quad (2.5)$$

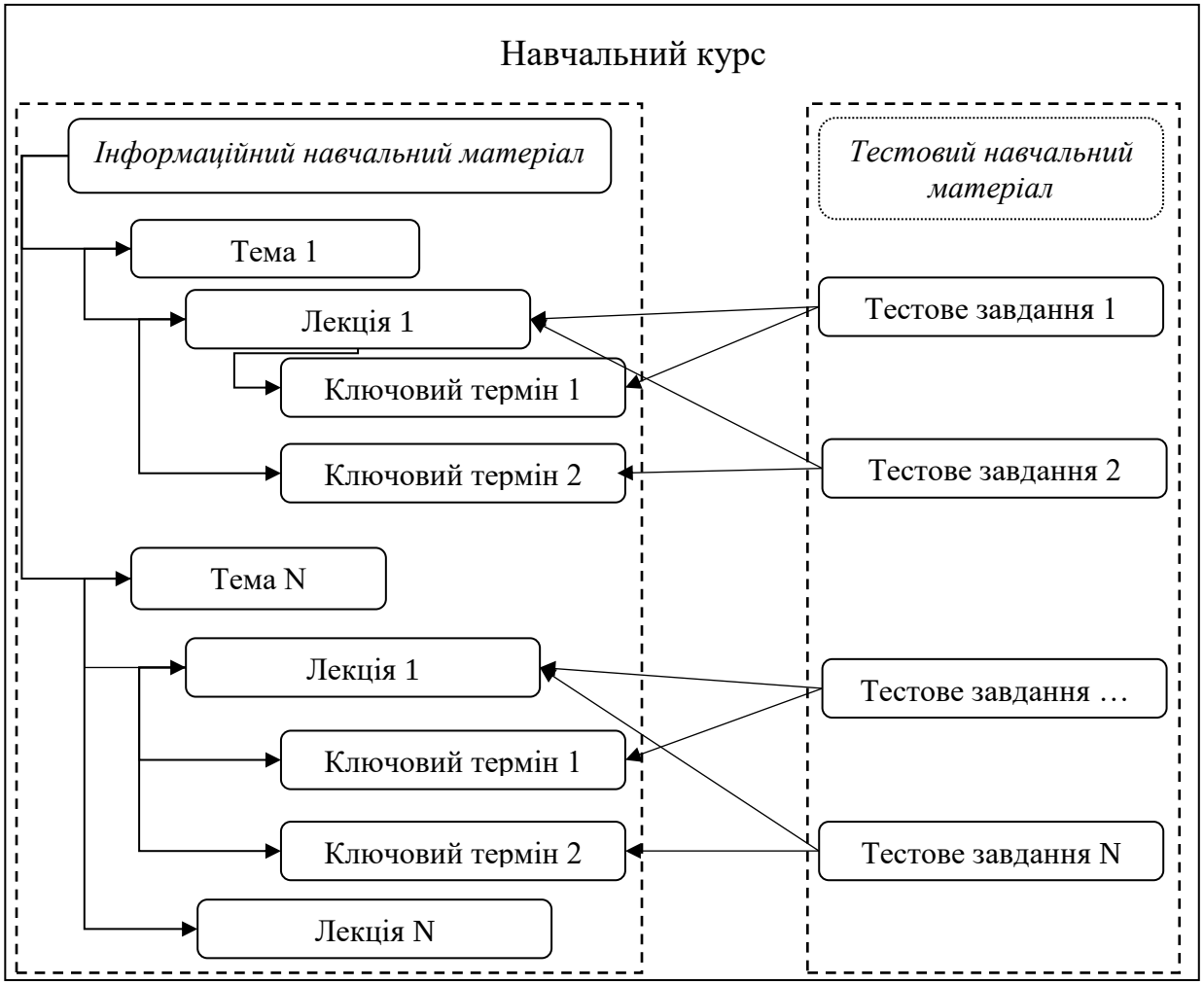


Рисунок 2.2 – Зразок подання навчального курсу дисципліни

На рисунку 2.3 зображено випадок, коли в контенті тестового завдання використовується три ключових терміни (Ключовий термін 1, Ключовий термін 2, Ключовий термін 3). Для формування елементів тестевого завдання (запитання, відповідей, дистракторів) було використано контент трьох речень (Речення 5, Речення 6, Речення 7). Таке тестове завдання може бути використано для перевірки лише найменш важливого терміну із трьох використаних.

Рисунок 2.3 – Зв'язок тестового завдання з елементами ІНМ

Таким чином, розроблена модель дозволяє використовувати згенеровані тестові завдання для перевірки рівня знань визначених ключових термінів у межах визначених рубрик ІНМ, контролюючи при цьому обсяг використаних в тестовому завданні складніших термінів. Це, зокрема, відкриває можливість для використання автоматизовано згенерованих множин тестових завдань для адаптивного тестування [28].

2.2 Метод автоматизованої генерації тестових завдань

2.2.1 Використання продукційної моделі для генерації тестових завдань

Процес генерації тестових завдань викладачем складається з наступних етапів:

1. Пошук важливого текстового фрагменту навчального матеріалу, який потрібно протестувати.

2. Генерація тестового завдання для перевірки знання знайденого фрагменту.

Такий принцип генерації тестових завдань можна інтерпретувати за допомогою *продукційної моделі* подання знань [29]. Така модель дозволяє подати знання у вигляді конструкції, схема якої наведена на рисунку 2.4.

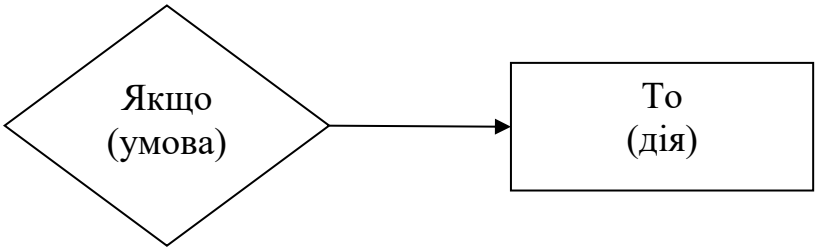


Рисунок 2.4 – Схема правила продукції

Наведені на рисунку 2.4 конструкції називаються ядрами продукції. Умовне ядро продукції містить певне речення-зразок, твердження, умова за якою здійснюється пошук потрібно елемента в базі знань. Дану частину також називають антецедентом.

Друге ядро продукції, або його ще називають дія або консеквент, описує висновок або дію яка повинна виконатися при успішному виконанні умови першого ядра. Виконувані дії не завжди являються кінцевим результатом, а можуть бути лише проміжним, що визначає умову для наступних ітерацій застосування продукційних правил.

Практичним прикладом застосування такої моделі знань є медичні системи для визначення діагнозу пацієнта. Тобто в даному випадку умовна частина може містити симптоми, а друга частина – діагноз для якого визначені дані симптоми. В такому випадку, якщо симптоми реального пацієнта співпадають і з перерахованими симптомами в антецеденті тоді встановлюється відповідний діагноз визначений в консеквенті.

При застосуванні продукційної моделі до завдання генерації тестових завдань схему продукціного правила можна представити так як зображено на рисунку 2.5.

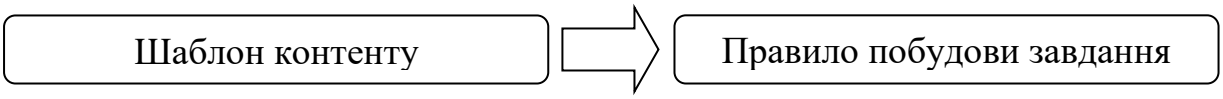


Рисунок 2.5 – Схема продукційного правила для генерації тестових завдань

Кожний антецедент правила продукції тестового завдання містить містить два обов’язкові елементи: ключовий термін так конектор (рис. 2.6). *Конектором* являється слово або символ із тексту, що приєднує ключовий термін із тезою. Відповідно якщо застосування антецеденту до текстового фрагменту дає позитивний результат тоді з даного фрагменту можна сформувати тестове завдання. Тест буде формуватися на основі описаних консеквентів.

Приклад застосування продукційних правил для генерації тестових завдань зображено на рисунку 2.7. В наведеному прикладі антецедентом визначається три вимоги до речення із тексту, у випадку виконання яких правило буде активним. Консеквент визначає 4 послідовні дії, які потрібно виконати для генерації тестового завдання. Таким чином, множину правил продукції можна використовувати як інструмент для генерації тестових завдань.

Для генерації тестових завдань логічного типу, одиничного вибору, множинного вибору та короткої відповіді можна виділити 8 консеквентів.

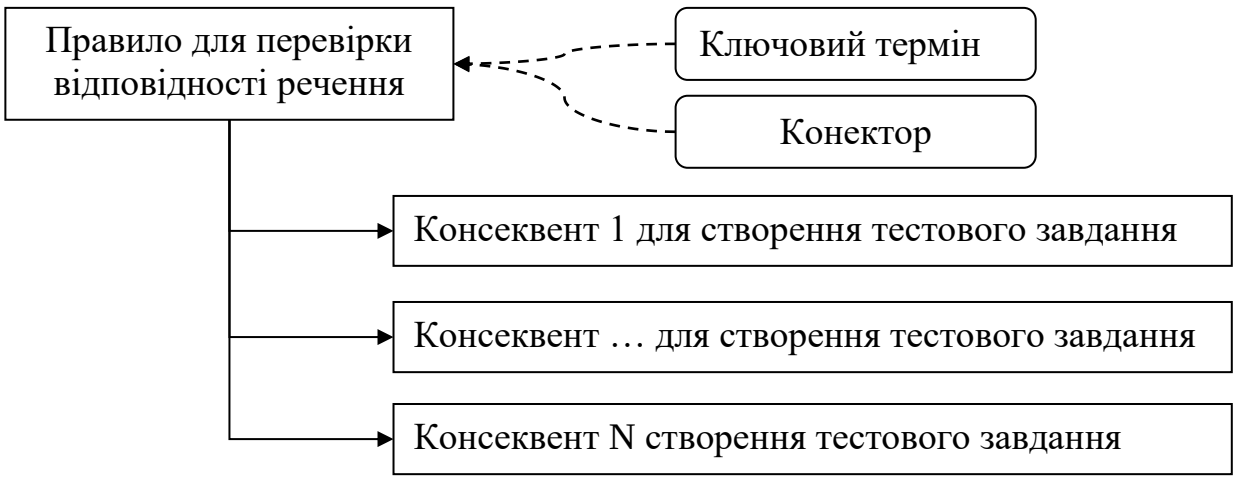


Рисунок 2.6 – Схема продукційного правила для генерації тестових завдань

Рисунок 2.7 – Приклад продукційного правила для створення прототипів тестових завдань

Для генерації тестових завдань логічного типу виділяється наступні 3 консеквенти:

– генерація логічного питання із речення без внесення будь-яких змін;

– генерація логічного питання із речення в якому заміняється ключовий термін на неправильний варіант;

– генерація логічного питання в якому заміняється тезисна частина на неправильний варіант.

Для генерації тестових завдань одиничного вибору використовуються такі консеквенти:

– генерація тестового завдання із вибором правильного ключового терміну в варіантах відповіді;

– генерація тестового завдання із вибором правильного означення в варіантах відповіді.

Для тестових завдань множинного вибору використовується консеквент при якому формується список на вибір кількох елементів з переліку, які відносяться до заданого реченню.

Для генерації тестових завдань із введенням відповіді використовуються консеквенти:

– генерація тестового завдання на введення терміну пропущеного у реченні;

– генерація тестового завдання на введення терміну, який відповідає наведеному означенню.

Таким чином, для генерації тестових завдань використання правил продукції є перспективним, оскільки кожне з правил дозволятиме забезпечити формальне подання конкретного алгоритму створення тестового завдання відповідної структури.

2.2.2 Схема та опис методу

Схема методу автоматизованої генерації тестових завдань зображена на рисунку 2.8. *Вхідними даними* методу є рубрика h із множини H для якої потрібно згенерувати тестові завдання та множину речень S, відповідну рубриці h. Третім параметром методу є множина зв'язків між реченнями множини S та рубрикою h. Наступними вхідними даними методу є множина ключових термінів K рубрики h та множина зв'язків між ключовими термінами та реченнями, в яких вони використовуються. Для роботи методу потрібна множина P створених продукційних правил для генерації тестових завдань.

На *Етапі 1* методу генерації тестових завдань відбувається скорочення множини речень S, що дасть змогу оперувати меншим об'ємом даних що пришвидшить роботу методу. Даний етап розпочинається із *Кроку 1.1* на якому відбувається вибір s_i речення із множини S. На *Кроці 1.2* обирається ключовий термін k_b із множини K. Ключова робота даного етапу виконується на *Кроці 1.3,* де відбувається перевірка входження k_b ключового терміну в s_i речення, шляхом пошуку зв'язку між терміном k_b та реченням s_i в множині R_2. Якщо відповідний зв'язок вдалося знайти, то речення додається в множину S', а якщо ні – виконання методу переходить на *Крок 1.2* і обирається наступний елемент k_{b+1} із множини K. Коли множина K буде вичерпана, метод повертається на *Крок 1.1* та обирається наступне речення s_i із множини S. *Етап 1* завершується після вичерпання множини речень S. В результаті роботи даного етапу буде сформовано множину актуальних речень S', яка передається на наступний етап.

Етап 2 відповідає за формування множини прототипів тестового завдання. На *Кроці 2.1* відбувається вибір s_i речення із множини S', а на *Кроці 2.2* вибирається продукційне правило p_j із множини P. Застосування антецедента правила продукції p_j до речення s_i відбувається на *Кроці 2.3*. У випадку успішності *Кроку 2.3* виконання переходить до *Кроку 2.4,* де відбувається

застосування консеквента правила продукції p_j до речення s_i для створення тестового завдання.

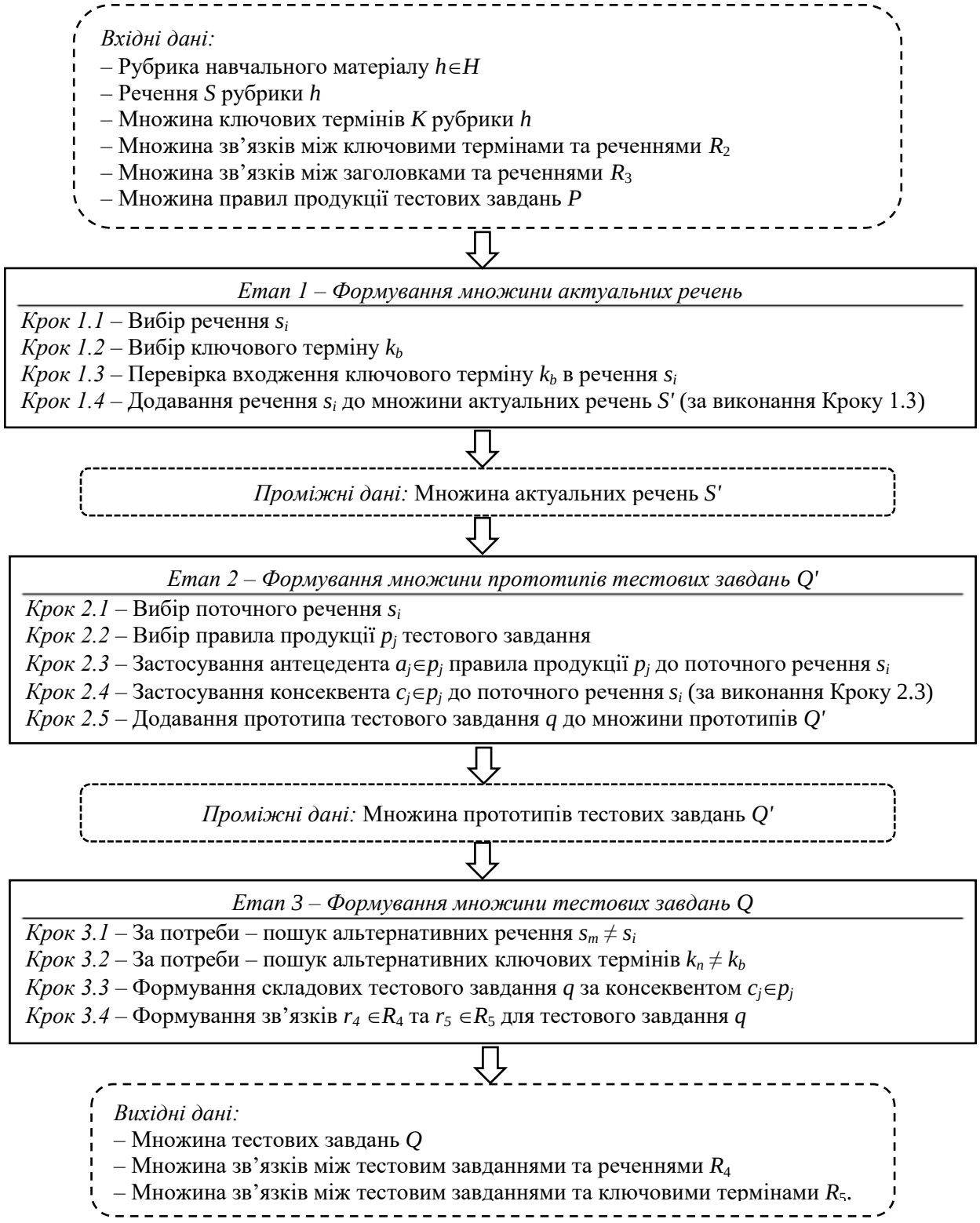


Рисунок 2.8 – Схема методу автоматизованої генерації тестових завдань

Варто зазначити, що на даному кроці не завжди вдається створити тестове завдання відповідно до правил визначених у консеквенті. Пов'язано це з тим, що деякі консеквенти вимагають застосування випадково підібраних фрагментів тестового завдання, яких не можливо визначити на даному етапі методу. Тому результатом застосування консеквента на *Кроці 2.4* є прототип тестового завдання, який вноситься в множину *Q'* на *Кроці 2.5.* Після виконання *Кроку 2.5* або неуспішного застосування антецедента на *Кроці 2.3* відбувається перехід на *Крок 2.2*. Коли множина *P* вичерпується виконання переходить на *Крок 1.1* та обирається наступне речення із множини *S'*. Виконання *Етапу 2* завершується після того, як будуть оброблені всі елементи множини *S'*.

Заключним є *Етап 3,* де на основі прототипів із множини *S'* завершується генерація тестових завдань. За потреби на *Кроці 3.1* і *Кроці 3.2* відбувається пошук альтернативних речень $s_m \neq s_i$, та ключових термінів $k_n \neq k_b$ для доповнення прототипів тестового завдання відповідно до консеквентів, після чого на *Кроці 3.3* генеруються кінцеві версії тестових завдань, які додаються в множину *Q*. *Крок 3.4* відповідає за створення та наповнення множин зв'язків R_4 та R_5.

Вихідними даними методу генерації тестових завдань є множина тестових завдань *Q* для рубрики начального матеріалу $h \in H$, яка була передана у вхідний параметр, а також множина зв'язків R_4 та R_5 в межах рубрики h. Для наповнення даних навчального курсу, що визначено в (2.5), розроблений метод потрібно застосувати циклічно для кожної рубрики ІНМ.

2.3 Інформаційна технологія автоматизованої генерації тестових завдань

Для вирішення завдання автоматизованої генерації тестових завдань до навчальних матеріалів було розроблено ІТ (рисунку 2.9) на основі методу розробленого в розділі 2.2. Як *вхідна дані* ІТ приймає правила продукції

тестових завдань та ІНМ (у вигляді рубрик навчального матеріалу, ключових термінів, зв’язків між заголовками, зв’язків між ключовими термінами та заголовками навчального матеріалу, зв’язків між реченням та заголовками).

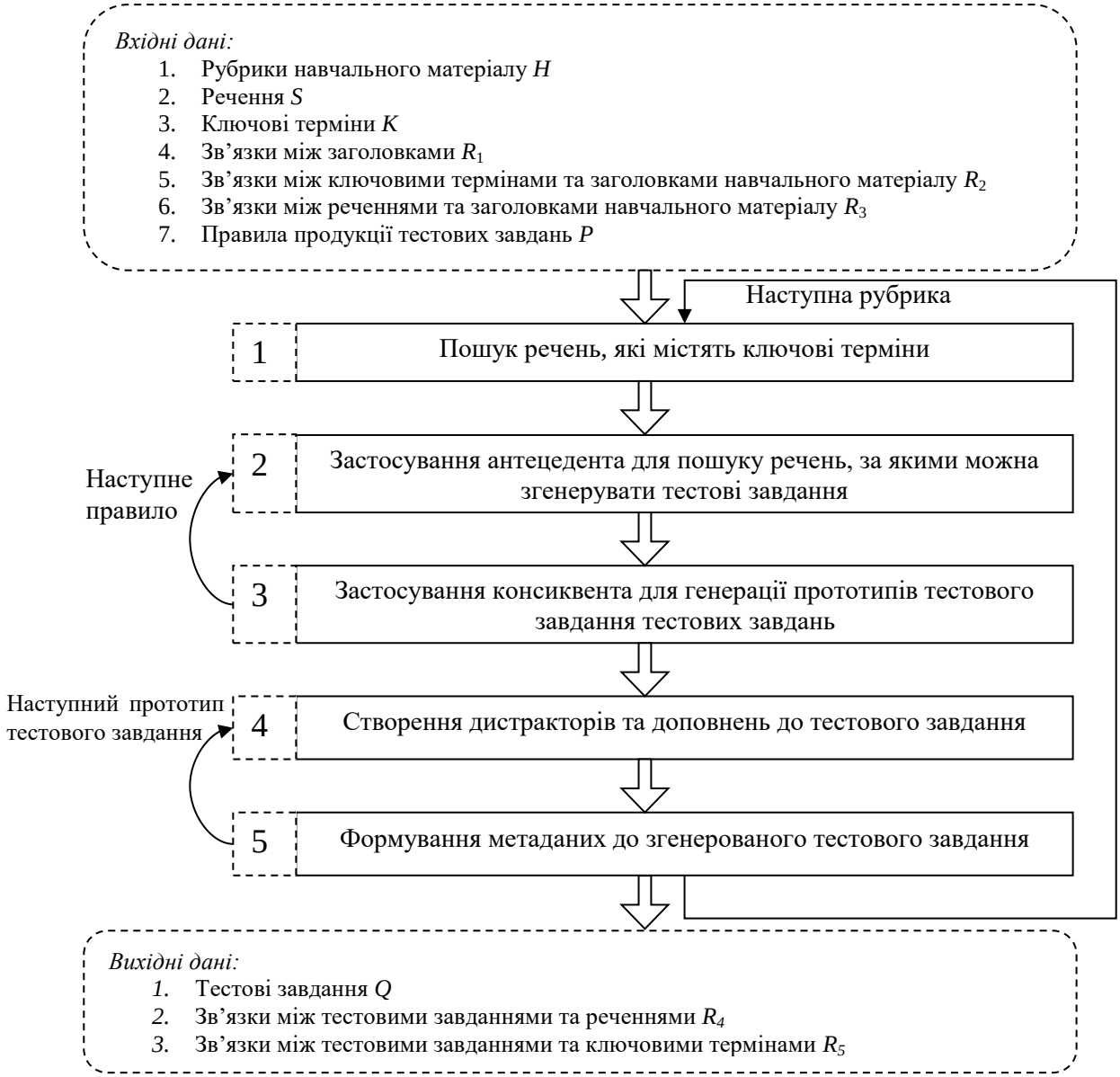


Рисунок 2.9 – Схема ІТ для генерації тестових завдань

Перший етап «Формування множини речень, які включають ключові терміни» розпочинає свою роботу із обробки речень рубрики навчального матеріалу для скорочення кількість речень до яких буде застосоване правило продукції. Оскільки основним елементом продукційного правила є ключовий

термін, то відкинувши речення, які не містять ключового терміну, вдасться уникнути пустого застосовувати правила.

На *другому етапі* обробляються речення, які надішли з попереднього етапу. До кожного отриманого речення застосовується антецедент правила продукції. Речення, які успішно пройшли перевірку умовами антецеденту, передаються на наступний етап.

На *третьому етапі* до отриманих речень застосовується консеквент правила продукції, тобто правило створення тестового завдання. Генерується максимальна кількість прототипів тестових завдань, але не завжди на даному етапі можна згенерувати повноцінний прототип тестового завдання. Це пов’язано із тим, що для генерації може бути потрібна випадково обрана теза, випадково обраний ключовий термін тощо. Такий елемент можна отримати лише тоді, коли існує певна вибірка однотипних прототипів на тестові завдання. Другий та третій етап виконуються циклічно, доки не буде використано всю множину правил продукції.

Наступні два етапи виконуються циклічно для кожного прототипу тестового завдання. На *четвертому етапі* до прототипу тестового завдання додаються дистрактори та доповнення. Потрібні фрагменти обираються із інших прототипів тестових завдань. Це потрібно для того, щоб максимально зв’язати тестові завдання між собою, що дасть можливість точніше перевірити знання визначених ключових термінів. Із доповненого прототипу тестового завдання генерується повноцінне тестове завдання, яке передається на наступний етап.

На *п’ятому етапі* згенероване тестове завдання додається в результуючу вибірку тестових завдань. Також створюються зв’язки між реченням на основі якого згенеровано тестове завдання та тестовим завданням й зв’язки між ключовими термінами та тестовими завданнями.

Описані етапи генерації тестів застосовуються для кожної рубрики документу. В результаті виконання розробленої ІТ генерації тестових завдань

заповнюється ТНМ для переданого на вхід ІНМ, що дозволяє отримати повністю заповнену множину даних навчального курсу (2.5).

Висновки до розділу 2

В розділі було проведено розробку моделі навчального курсу для автоматизованої генерації тестових завдань. Розроблена модель включає в себе наступні елементи: множину заголовків, множину речень, множину зв’язків між заголовками, множину ключових термінів, множину зв’язків між ключовими термінами та реченнями, множину зв’язків між заголовками та реченнями, множину тестових завдань, множину зв’язків між тестовим завданнями та реченнями, множину зв’язків між тестовим завданнями та ключовими термінами.

Розроблено метод автоматизованої генерації тестових завдань до навчальних матеріалів, що дозволяє генерувати тестові завдання за контентом навчального матеріалу на основі правил продукції.

На основі даного методу розроблено інформаційну технологію автоматизованої генерації тестових завдань, що дозволяє за поданням семантичної структури інформаційного навчального матеріалу одержувати множину тестових завдань, призначених для перевірки рівня знань визначених ключових термінів рубрик ІНМ.

Розділ 3

Розробка інформаційної системи автоматизованої генерації тестових завдань

3.1 Схема функцій інформаційної системи автоматизованої генерації тестових завдань

Для виконання завдання автоматизованої генерації тестових завдань, експериментальна інформаційна система повинна виконувати 8 основних функцій (рисунок 3.1):

1) завантаження вхідних даних у вигляді семантичної структури ІНМ,
2) зчитування продукційних правил,
3) генерація тестових завдань з однією правильною відповіддю,
4) генерація тестових завдань із декількома правильними відповідями,
5) генерація тестових завдань із можливістю введення відповіді,
6) генерація тестових із варіантами відповіді «Так» або «Ні»,
7) редагування тестових завдань,
8) експорт тестових завдань в середовище для тестування.

Функція «Зчитування продукційних правил». Оскільки для генерації тестових завдань використовуються правила продукції і кількість таких правил може бути не обмеженою, то вони не можуть бути вписані в код майбутньої реалізації системи. Тому продукційні правила потрібно записувати і зчитувати з локального файлу або бази даних. Саме за це відповідає поточна функція.

Функції 3 – 6 відповідають за основну роботу системи. В кожній функції генерується тестове завдання відповідного типу опираючись на продукційні правила, ключові терміни та контент документу.

Результатом роботи системи є множина згенерованих тестів, які потрібно експортувати в середовище для тестування. Саме за це відповідає функція «Експорт тестових завдань в середовище для тестування».

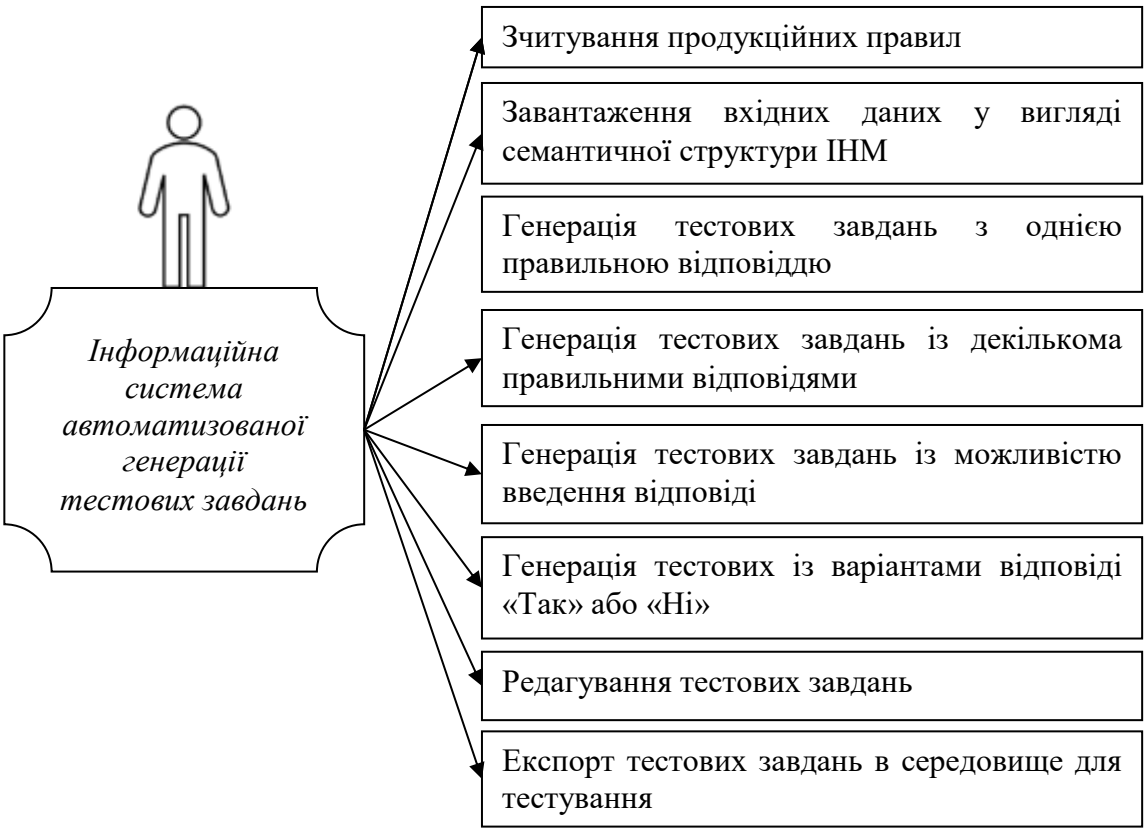


Рисунок 3.1 – Функції інформаційної системи автоматизованої генерації тестових завдань

Архітектура інформаційної системи має бути побудована в такий спосіб, щоб забезпечити коректне виконання наведених функцій інформаційної системи автоматизованої генерації тестових завдань.

3.2 Даталогічна модель бази даних інформаційної системи

Для отримання вхідних та збереження вихідних даних інформаційної системи автоматизованої генерації тестових завдань потрібна БД, даталогічна модель якої зображена на рисунку 3.2.

В таблицях «ІНМ», «Заголовки», «Абзаци», «Речення», «Ключові терміни», «Терміни в реченнях», «Терміни» зберігаються дані про ІНМ, структура якого визначена в (2.2). Модуль інформаційної системи, який відповідає за генерацію тестових завдань, отримує дані зі перерахованих таблиць

та повністю відтворює інформаційний навчальний матеріал та генерує тестові завдання, які зберігаються в таблицю «Тестові завдання». В таблиці також зберігається інформація про те, який ключовий термін покривається згенерованим тестовим завданням, з якого речення згенеровано та який тип тестового завдання. Список типів тестових завдань визначений в таблиці «Типи завдань». Відповіді до згенерованого тестового завдання зберігаються в однойменні таблиці «Відповіді».

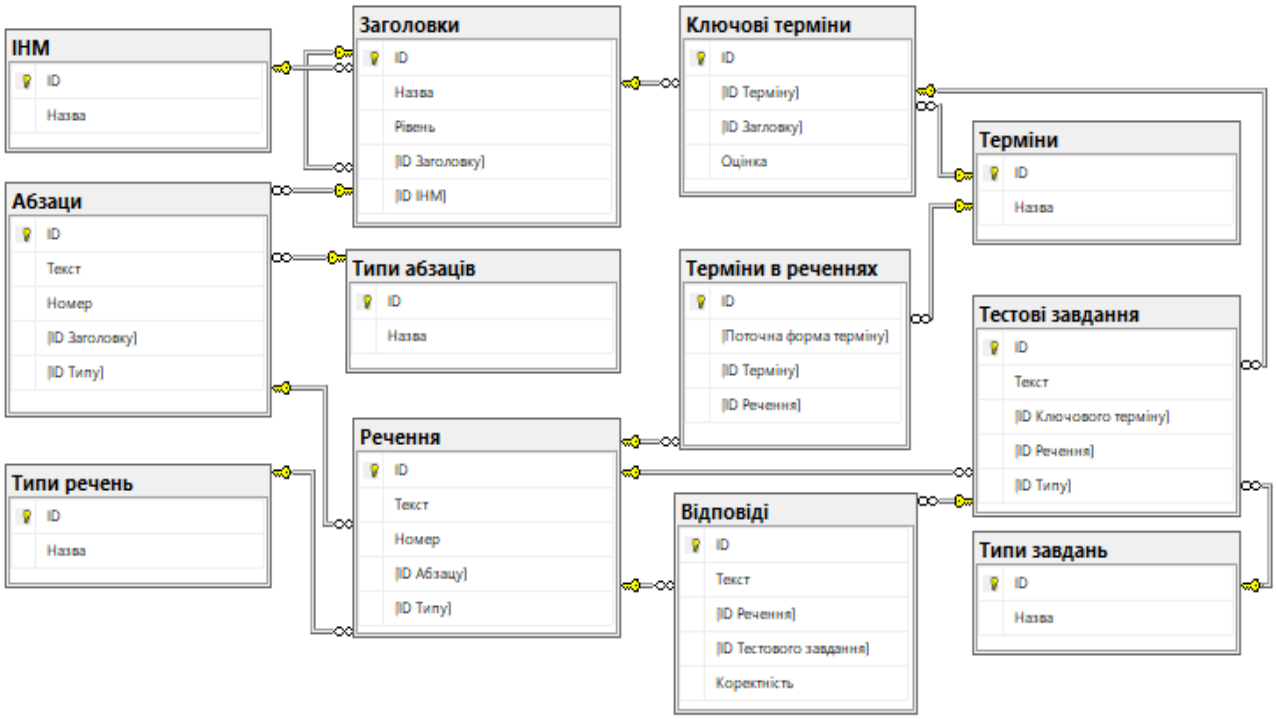


Рисунок 3.2 – Даталогічна структура бази даних інформаційної системи автоматизованої генерації тестових завдань

Описана структура бази даних дозволяє зберігати дані потрібні для інформаційної системи генерації тестових завдань та повністю покриває структуру навчального курсу, наведену в (2.5).

3.3 Аналіз засобів імпорту тестових завдань у середовище Moodle

Оскільки в пункті 1.2 було обрано систему управління навчанням Moodle, то система генерації тестових завдань повинна мати механізм взаємодії із даним середовищем. Тобто система автоматизованої генерації тестових завдань повинна вміти експортувати згенеровані завдання в систему Moodle.

LMS Moodle підтримує наступні формати імпорту та експорту тестових завдань із файлу [30]:

1) Вбудовані відповіді (пропущені слова);
2) Aiken;
3) Blackboard;
4) Examview (екзаменаційний);
5) GIFT;
6) Missing word (пропущене слово);
7) Moodle XML;
8) WebCT (web course tools – веб засоби для курсів).

Таблиця 3.1 – Основні елементи розмітки GIFT

Символи	Опис
//text	Коментар
::title::	Запитання тестового завдання
[…format…]	Формат поточної частини тексту. Може набувати значення: [html], [moodle], [plain],[markdown].
{	Початок фрагменту із варіантами відповіді
{T}, {F}	Позначається вірне або не вірне твердження в завданнях логічного типу
=	Вірна відповідь
~	Невірна відповідь
=item -> match	Відповіді для встановлення відповідності
=%n%answer:tolerance	Відсоткове число встановлення важливості відповідей в тестових завдання на вибір декількох відповідей
}	Кінець фрагменту із варіантами відповіді
\n	Перехід на новий рядок

Із перерахованих форматів імпорту/експорту тестових завдань в систему Moodle було обрано формат GIFT та Moodle XML, оскільки дані формати підтримують всі потрібні типи тестових завдань. Також тестові завдання різного типу можуть бути збережені в одному файлі одночасно. Синтаксична структура обраних форматів не складна і проста в реалізації.

GIFT формату імпорту/експорту тестових завдань має власний набір символів для побудови даного формату [31], перелік та опис кожного наведено в таблиці 3.1. Для імпорту/експорту тестових завдань в форматі Moodle XML використаються теги наведені в таблиці 3.2 [32].

Таблиця 3.2 – Основні елементи розмітки XML

Елемент	**Опис**
quiz	Узагальнюючий тег для набору з усіх тестів
question	Визначає межі тестового завдання. Поточний тег може містити атрибут type для визначення типу тестового завдання. Атрибут може набувати наступних значень: multichoice, truefalse, shortanswer, matching, cloze, essay, numerical, description.
name	Назва тестового завдання
text	Тег використовується для вставки текстового фрагменту в інших тегах.
questiontext	Використовується для визначення завдання
answer	Відповіді до тестового завдання. Атрибут fraction визначає який варіант відповіді є коректним (100), а який ні (0).
subquestion	Використовується для визначення пари «Теза - Відповідь» в завданнях для встановлення відповідності.
shuffleanswers	Використовується для тестів визанчення чи потрібно перемішувати варіанти відповідей. Тег може містити значення true (так) або false (ні).

В наступному лістингу наведено приклади компоновки тегів для визначення тестового завдання на відповідність та вибору декількох відповідей [32]:

```
<quiz>
<question type="matching">
 <subquestion>
```

```
     <text>Перший елемент першої пари</text>
     <answer>
         <text>Другий елемент першої пари</text>
     </answer>
 </subquestion>
 <subquestion>
     <text>Перший елемент другої пари</text>
     <answer>
         <text>Другий елемент другої пари</text>
     </answer>
 </subquestion>
</question>

<question type="multichoice">
 <answer fraction="100">
     <text>Перша правильна відповідь</text>
 </answer>
 <answer fraction="100">
     <text>Друга правильна відповідь </text>
</answer>
 <answer fraction="0">
     <text>Неправильна відповідь</text>
 </answer>
 <shuffleanswers>true</shuffleanswers>
 <single>false</single>
 <answernumbering>abc</answernumbering>
</question>
<quiz>
```

3.4 Розробка правил генерації тестових завдань

3.4.1 Розробка колекції тегів для формального опису правил продукції тестових завдань

Для того, щоб мати формалізоване подання правил генерації тестових завадань, було розроблено набір тегів. Отримані набори можна розділи на два типи за призначенням:

1) теги ідентифікації текстового контенту (табл. 3.2);
2) теги для генерації тестових завдань (табл. 3.3).

В таблиці 3.4 наведено перелік атребутів які можуть бути застосовані то тегів, для додатковго уточнення. Також було розроблено теги для компановки правил продукції, наведені в таблиці 3.5.

Таблиця 3.2 – Теги для ідентифікації елементів контенту

Тег	Опис
<signature />	Тег верхнього рівня в якому компонуються всі теги для антецедента ідентифікації контенту
<TermGroup />	Частина тексту, який являє собою ключовий термін
<ThesisGroup />	Частина тексту (теза), в які описується визначення терміну
<RandomThesisGroup />	Випадково обране визначення іншого терміну
<RandomTermGroup />	Випадково обраний термін
<Connector />	Слово або символ із тексту за допомогою якого поєднується термін із тезою (– , – це, є, називається, тощо)
<BeginSentence />	Тег який відповідає за частину тексту початку речення до TermGroup або ThesisGroup

Таблиця 3.3 – Теги компановки правил для генерації тестів

<signature>	Тег верхнього рівня в якому компонуються всі теги для консеквента генерації тестів
<rightAnswer />	Тег в якому компонуються всі теги для генерації правильної відповіді
<wrongAnswer />	Тег в якому компонуються всі теги для генерації неправильної відповіді
<FALSE />	Тег який використовується для зазначення відповіді «Ні» в завданях логічного типу
<TRUE />	Тег який використовується для зазначення відповіді «Так» в завданнях логічного типу

Таблиця 3.4 – Атребути для тегів

id	Ідентифікатор правила продукції
Case	Використовується для визначення рівня регістру першої букви фрагменту тексту, який відповідає обраному тегу. Атребут може набувати двох значень «upper», «lower» («lower» за умовчуванням)

Таблиця 3.5 – Теги компановки правил продукції

<masks>	Тег самого верхнього рівня, який містить перелік всіх правил продукції
<mask>	Тег який містить правило продукції
<testMasks>	Тег верхнього рівня для всіх консеквентів генерації тестових завдань

Наведені у таблицях теги є своєрідним псевдокодом і можуть бути замінені.

3.4.2 Розробка правил продукції генерації тестових завдань логічного типу

Моделі генерації тестових завдань логічного типу полягають у генерації завдань і двома варіантами відповіді: Так або Ні. В даному випадку можливі два типи моделі. Для першого типу характерна генерація тестових завдань із правильним твердженням, для якого правильною відповіддю буде «Так». Відповідно другий тип призначений для генерації тестового завдання із хибного твердження, для якого правильною відповіддю буде «Ні».

Для генерації тестового завдання першого типу із правильно відповіддю «Так» може використовуватися наступне правило:

```
<mask type="YesNo">
      <signature>
            <TermGroup case="upper"/>
            <Connector />
            <ThesisGroup />
      </signature>
      <rightAnswer>
            <TRUE />
      </rightAnswer>
      <wrongAnswer>
            <FALSE />
      </wrongAnswer>
</mask>
```

При використанні наведеного правила, до знайденого твердження у тексті навчального матеріалу не застосовуються перетворення оскільки воно вірне.

Для генерації тестового завдання другого типу (із правильно відповіддю «Ні») застосовується два правила:

1. Замість конкретного терміну підставляється будь-який випадковий термін

```
<mask type="YesNo" >
      <signature>
            <RandomTermGroup case="upper"/>
            <Connector />
            <ThesisGroup />
      </signature>
      <rightAnswer>
            <FALSE />
      </rightAnswer>
      <wrongAnswer>
            <TRUE />
      </wrongAnswer>
</mask>
```

2. Замість коректної тези використовується будь-яка випадкова теза

```
<mask type="YesNo">
      <signature>
            <TermGroup case="upper"/>
            <Connector />
            <RandomThesisGroup />
      </signature>
      <rightAnswer>
            <FALSE />
      </rightAnswer>
      <wrongAnswer>
            <TRUE />
      </wrongAnswer>
</mask>
```

При використані правила із випадково підібраним терміном або тезою потрібно відзначити те, що випадкове значення повинно бути отримане із тієї самої секції документу для якої генерується тестове завдання. Це потрібно для

того, щоб згенероване тестове завдання максимально відповідало контексту використаної частини документу.

3.4.3 Розробка правил продукції генерації тестових завдань одиночного вибору

Застосування правил для генерації тестового завдання одиничного вибору полягають у тому, щоб згенерувати із тексту, знайденого відповідним антецедентом, запитання для яких пропонується два або більше варіантів відповідей, де вірною є лише одна. У такому випадку існує два типи консеквентів. Перший тип відповідає за генерацію завдання із правильним терміном у варіантах відповідей:

```
<mask type="SingleChoice" >
      <signature>
            <ThesisGroup case="upper"/>
            <Connector />
      </signature>
      <rightAnswer>
            <TermGroup case="upper"/>
      </rightAnswer>
      <wrongAnswer>
            <RandomTermGroup case="upper"/>
      </wrongAnswer>
</mask>
```

Другий тип передбачає генерацію завдання із правильною тезою у варіантах відповідей:

```
<mask type="SingleChoice" >
      <signature>
            <TermGroup case="upper"/>
            <Connector />
      </signature>
      <rightAnswer>
            <ThesisGroup case="upper"/>
      </rightAnswer>
      <wrongAnswer>
            <RandomThesisGroup case="upper"/>
      </wrongAnswer>
```

```
</mask>
```

3.4.4 Розробка правил продукції генерації тестових завдань множинного вибору

Правила для генерації тестових завдань множинного вибору полягає у генерації завдання, що складаються із двох і більше варіантів відповідей і де водночас вірною є декілька відповідей.

Наступна продукційне правило є прикладом антецедента для пошуку речення з якого можна згенерувати тестове завдання множинного вибору:

```
<mask>
      <signature>
            <BeginSentence />
            <TermGroup />
            <Connector />
            <SpecSymbol>:</SpecSymbol>
            <List />
      </signature>
      <connector>є</connector>
      <connector>що</connector>
      <connector>наступні</connector>
      …
      <connector>такі</connector>
      <connector>бувають</connector>
</mask>
```

Наприклад за допомогою такого антецедента буде вибрано наступне речення із навчального матеріалу: «До емпіричних методів дослідження належать: експерименти, наукові дослідження, спостереження, вимірювання». Ключовим терміном є «Емпіричний метод». Конектор – належить.

Консеквент для генерації тестового завдання може бути:

```
<mask type="MultipleChoice">
      <signature>
            <BeginSentence />
            <TermGroup />
            <Connector />
            <SpecSymbol>:</SpecSymbol>
```

```
      </signature>
      <rightAnswer>
            <SplitedList case="upper"/>
      </rightAnswer>
      <wrongAnswer>
            <RandomSplitedList case="upper"/>
      </wrongAnswer>
</mask>
```

3.4.5 Розробка правил продукції генерації тестових завдань із введенням відповіді

Продукційне правило для генерації тестового завдання із введенням відповіді полягає в генерації завдання в якому для користувача не пропонуються можливі варіанти відповіді, а відбувається перевірка значення введеного користувачем. Прикладом такого продукційного правила є:

```
<mask>
      <signature>
            <TermGroup />
            <Connector />
            <EndSentence />
      </signature>
      <connector> – це</connector>
      <connector>називається</connector>
      …
      <connector>називаються</connector>
      <testmask>
            <mask type="InputAnswer" >
                  <signature>
                        <InputSpace />
                        <Connector />
                        <ThesisGroup />
                  </signature>
                  <rightAnswer>
                        <TermGroup case="upper"/>
                  </rightAnswer>
            </mask>
      </testmask>
<mask>
```

За допомогою такого правила відбувається пошук речення в яке відповідає зв’язкам «термін – конектор – пояснення терміну». Наприклад, із

речення: «Наука – це сфера діяльності людини, спрямована на отримання (вироблення і систематизацію у вигляді теорій, гіпотез, законів природи або суспільства тощо) нових знань про навколишній світ», буде згенеровано наступне тестове наступне тестове завдання: «_____ – це сфера діяльності людини, спрямована на отримання (вироблення і систематизацію у вигляді теорій, гіпотез, законів природи або суспільства тощо) нових знань про навколишній світ», де правильною відповідь є «Наука».

Для тестових завдань даного типу дуже важливою частиною є те, щоб виключити можливість ігнорування введених відповідей через не співпадіння закінчень у веденому значені, та у визначеному варіанті правильної відповіді. Для вирішення даної проблеми термін, який зазначений як правильна відповідь повинен бути у називному відмінку. Також у функцію даної модель не потрапляє генерація хибних варіантів відповідей, оскільки це не потрібно.

3.5 Опис комбінації засобів розробки інформаційної системи

Для розробки інформаційної системи генерації тестових завдань було обрано програмну платформу Microsoft .NET Framework [33, 34]. На даний момент платформа стрімко розвивається і вже існує її кросплатформена версія під назвою .NET Core [35]. Платформа включає велику бібліотеку класів, на основі якої можна реалізовувати додатки різного типу (Web сервіси, Desktop програми тощо) із використанням різних сервісів, таких як: доступ до файлової системи, робота із базою даних тощо. Завдяки своїй структурі, .NET підтримує багато мов програмування серед яких: C#, C++/CLI, F#, Visual Basic тощо. Із перерахованих мов програмування найбільш інтенсивно розвивається C#, яка постійно розвивається разом із платформою .NET.

Для розробки інформаційної системи генерації тестових завдань було обрано тип додатку «Desktop application» із використанням технології WPF [36] для побудови графічного інтерфейсу додатку. Технологія WPF в порівняні із

Windows Forms [37] надає можливість створення гнучких елементів інтерфейсу користувача із шаблонами та стилями, а також є можливість створення графічних анімацій. WPF технологія надає можливість розділили логіку відображення від бізнес-логіки додатку.

Оскільки для розробки інформаційної системи обрано платформу .NET, то для повної сумісності було обрано СКБД MS SQL Server [38] для роботи з БД. Для взаємодії із СКБД було обрано технологію Entity Framework [39]. Робота з базою даних реалізується за допомогою Entity Framework, що надає можливість працювати з базою даних із об’єктно-орієнтованим підходом на базі .NET. Entity Framework переводить роботу із базою даних на вищий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища даних. На рівні Entity Framework відбувається робота з об’єктами даних, в той час як на фізичному рівні потрібно оперувати таблицями, індексами тощо.

3.6 Архітектура інформаційної системи автоматизованої генерації тестових завдань

Відповідно до визначених функцій інформаційної системи було розроблено архітектуру інформаційної системи автоматизованої генерації тестових завдань схема якої відображена на рисунку 3.3.

Інформаційна система автоматизованої генерації тестових завдань складається із 7 підсистем:

- Підсистема роботи з графічним інтерфейсом;
- Підсистема генерації тестових завдань;
- Підсистема роботи з правилами продукції;
- Підсистема роботи з базою даних;
- Підсистема експорту тестових завдань у форматі GIFT;
- Підсистема експорту тестових завдань у форматі Moodle XML.

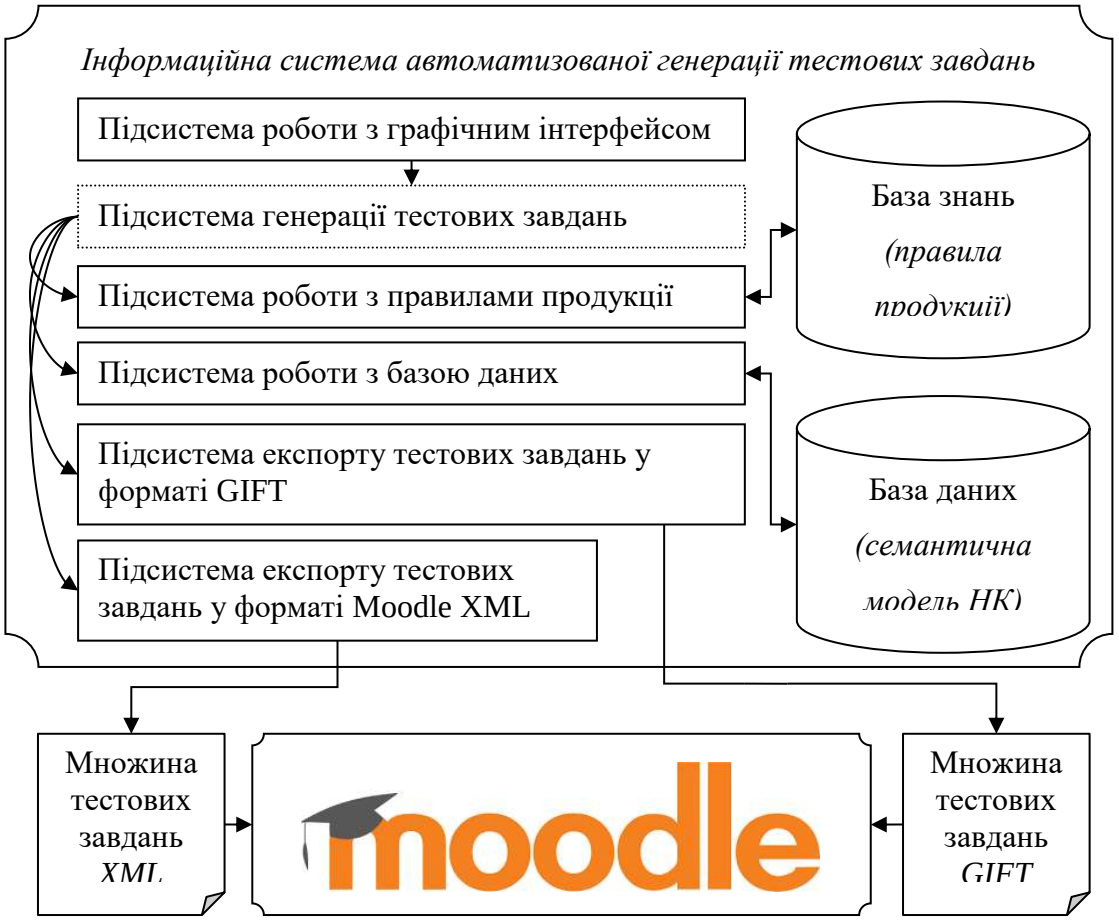


Рисунок 3.3 – Архітектура інформаційної системи автоматизованої генерації тестових завдань

Оскільки для розробки графічного інтерфейсу було обрано технологію WPF, то для побудови архітектури підсистема роботи з графічним інтерфейсом було обрано шаблон проектування MVVM. Даний архітектурний шаблон призначений для розділення бізнес-логіки та логіки графічного інтерфейсу. Мета даного шаблону, як і шаблону MVC, полягає в досягнені наступного принципу: «Утримання UI-коду простим і вільним від логіки програми, щоб полегшити підтримку коду» [40]. MVVM складається з трьох шарів, які відображенні на рисунку 3.4. В результаті застосування шаблону MVVM було побудовано діаграму класів підсистеми генерації тестових завдань, яка зображена на рисунку 3.5

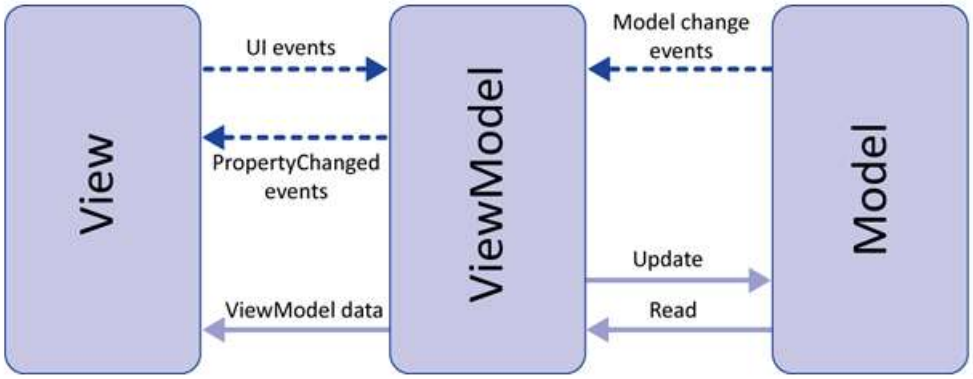


Рисунок 3.4 – Схема MVVM патерну [40]

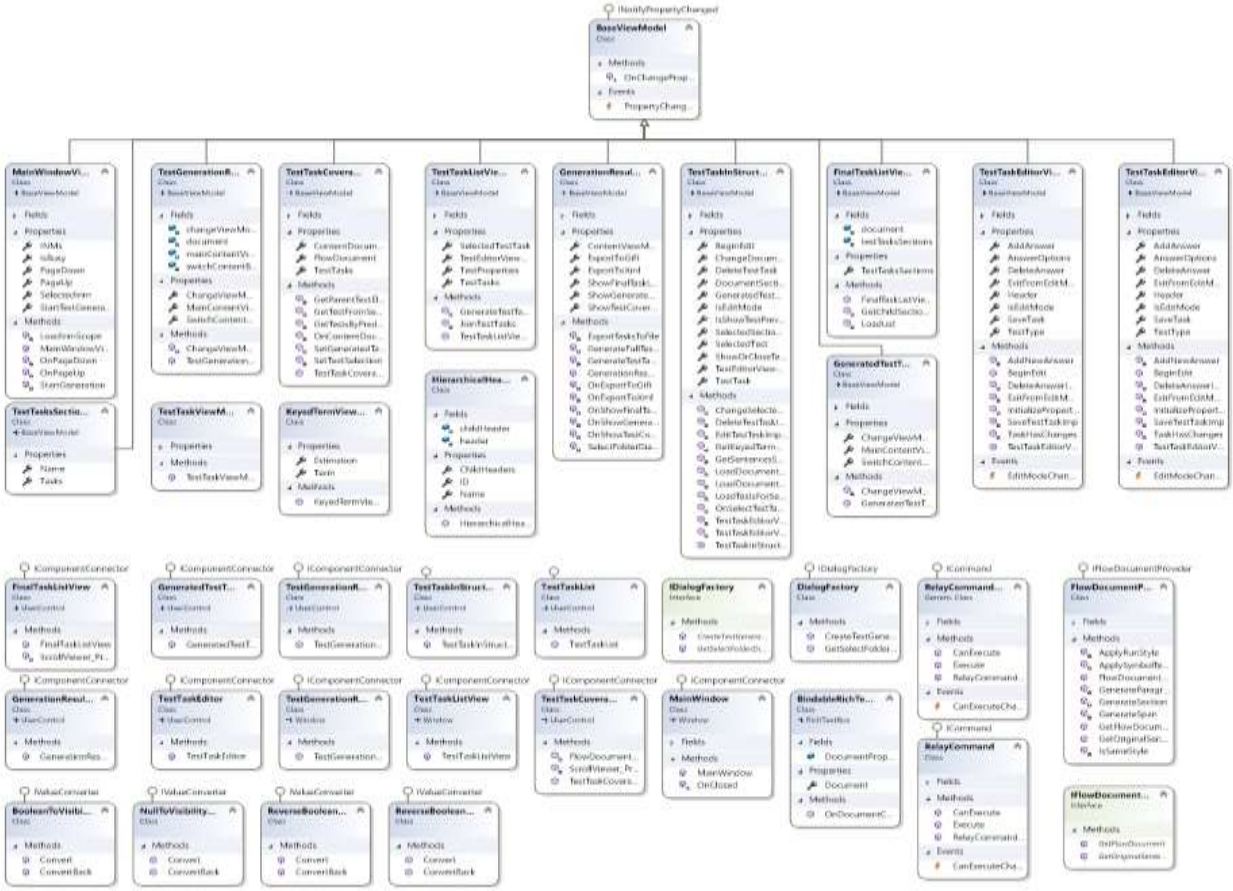


Рисунок 3.5 – Діаграма класів підсистеми роботи з графічним інтерфейсом

В результаті побудови архітектури інших підсистем (підсистем бізнес-логіки) інформаційної системи автоматизованої генерації тестових завдань було отримано діаграму класів, зображену на рисунку 3.6 та в розгорнутому вигляді у Додатку В.

Рисунок 3.6 – Діаграма класів підсистем бізнес-логіки інформаційної системи

Наведені архітектура та діаграма класів інформаційної системи забезпечують можливість створення інформаційної системи автоматизованої генерації тестових завдань, відповідної до розробленої інформаційної технології.

3.7 Розробка програмних модулів

3.7.1 Модуль «Робота з продукційними правилами»

Головною функцією модуля для роботи з продукційними правилами являється зчитування їх з файла з розширенням xml. Для того, щоб зберігати інформацію про правило з структурою описану в розділі 2.2 – реалізовано класи TaskMask і XmlMask.

Клас XmlMask реалізований для збереження інформації про унікальне правило продукції, яке використовується для накладання на речення текстового документу. У випадку успішного її застосування до речення – з останнього можна створити тестові завдання по спеціально створених для цього правилах. Властивості даного класу перераховані в таблиці 3.6.

Таблиця 3.6 – Властивості класу XmlMask

Назва	**Опис**
ID	Ідентифікатор правила продукції.
Mask	Повертає правило, яка застосовується для визначення чи підходить речення для генерації тестового завдання
Connectors	Список конекторів
TaskMask	Повертає правило для генерації тестових завдань з речення до якого було успішно застосовано правило з властивості Mask

Клас TaskMask призначений для зберігання інформації про продукційне правило для генерації безпосередньо тестових завдань. Клас складається з властивостей перерахованих в таблиці 3.7

Таблиця 3.7 – Властивості класу TaskMask

Назва	**Опис**
ID	Ідентифікатор продукційного правила.
TestType	Повертає тип правила
QuestionMask	Містить правило для створення запитання в тестовому завдані.
RightAnswerMask	Повертає правило призначену для створення правильної відповіді.
WrongAnswerMask	Повертає правило призначену для створення не правильної відповіді.

Клас MaskManager реалізовує інтерфейс IMaskManager і призначений для керування правилами продукції. Наразі його головною функцією є завантаження правил з файлу формату xml. Завантажені правила можна отримати з властивості Masks. А для безпосередньо завантаження правил реалізовано метод LoadMasks.

3.7.2 Модуль «Тестові завдання»

ETestType є перерахунком типів тестових завдань і містить наступні значення:

– YesNo;
– SingleChoice;
– MultipleChoice;
– InputAnswer;

Клас TestTask призначений для збережння інформації про тестове завдання. Даний клас містить один метод Clone, який призначений для клонування поточного об’єкта тестового завдання. Також в класі реалізовані властивості перераховані в таблиці 3.8.

Таблиця 3.8 – Основні властивості класу TaskTask

Назва	**Опис**
TestType	Повертає тип тестового завдання. Може мітить тише значення з перерахунку ETestType.
Keyterm	Ключовий термін який використовувався при генерації тестового завдання.
Task	Повертає згенеровано запитання для тестового завдання.
Sentences	Речення з якого було згенеровано тестове завдання.
GeneratedByTaskMask	Повертає правило за допомогою якого було згенеровано тестове завдання.
AnswerOption	Повертає список варіантів відповідей.

Клас AnswerOption призначений для збереження інформацію про варіант відповіді на тестове завдання. Даний клас реалізовує інтерфейс ICloneable для створення копії екземпляра класу AnswerOption. Також клас має 2 властивості, описані в таблиці 3.9.

Таблиця 3.9 – Властивості класу AnswerOption

Назва	**Опис**
Answer	Властивість як повертає текстову відповідь на тестове завдання.
IsRightAnswer	Властивість типу bool. Якщо вона повертає true тоді варіант відповіді правильний, якщо false – неправильний.

Клас TestTaskWorkpiece являється накопичувачем інформація про майбутнє тестове завдання. Він відіграє роль своєрідної «заготовки» для тестового завдання. В таблиці 3.10 наведено основні властивості даного класу.

Таблиця 3.10 – Основні властивості класу TestTaskWorkpiece

Назва	**Опис**
KeyTerm	Повертає ключовий термін який використовується для генерації тестового завдання..
GeneratedByTaskMask	Повертає правило, яке використовується для генерації тестового завдання.
QuestionParts	Повертає список елементів значення тегів продукційного правила для генерації запитання для тестового завдання.
RightAnswerParts	Повертає список елементів значення тегів правила для генерації правильної відповіді на тестове завдання.
WrongAnswerParts	Повертає список елементів значення тегів правила для генерації запитання на тестове завдання.
Sentences	Повертає речення з якого формується тестове завдання.
SentencesParts	Повертає список елементів, які відповідають значенням тегів правил, яка застосовувалася при розбиті речення.

3.7.4 Модуль «Робота з базою даних»

Робота з базою даних реалізується за допомогою Entity Framework, що надає можливість працювати з базою даних із об’єктно-орієнтованим підходом на базі .NET. Даний модуль включає набір класів, які об’єктно-орієнтовано

представляють кожну таблицю бази даних. Клас INM перезначений для роботи із таблицею «IHM», відповідно 1 об’єкт даного класу представляє собою 1 запис в таблиці IHM. Властивості класу INM наведенні в таблиці 3.11.

Таблиця 3.11 – Основні властивості класу INM

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці IHM
Name	Повертає або задає значення запису таблиці IHM із колонки «Назва»

Клас Header призначений для роботи із таблицею «Заголовки». Основні властивості класу Header наведені в таблиці 3.12.

Таблиця 3.12 – Основні властивості класу Header

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Заголовки»
Name	Повертає або задає значення колонки «Назва» для відповідного запису із таблиці «Заголовки»
Level	Повертає або задає значення колонки «Рівень» для відповідного запису із таблиці «Заголовки»
ParentHeaderID	Повертає або задає значення вторинного ключа «ID Заголовку» який вказує на ID заголовка якому він належить
InmID	Повертає або задає значення вторинного ключа «ID IHM» значення якого повертає ID IHM, якому належить заголовок

Клас Paragraph призначений для роботи із таблицею «Абзаци». Основні властивості класу Paragraph наведені в таблиці 3.13.

Таблиця 3.13 – Основні властивості класу Paragraph

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Абзаци»
Number	Повертає або задає значення колонки «Номер» для відповідного запису із таблиці «Абзаци»
HeaderID	Повертає або задає значення вторинного ключа «ID Заголовку» який вказує на ID типу заголовку якому належить абзац
TypeID	Повертає або задає значення вторинного ключа «ID Типу», який вказує на ID типу абзацу

Клас ParagraphType призначений для роботи із таблицею «Типи абзаців». Основні властивості класу ParagraphType наведені в таблиці 3.14.

Таблиця 3.14 – Основні властивості класу ParagraphType

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Типи абзаців»
Name	Повертає або задає значення колонки «Назва» для відповідного запису із таблиці «Типи абзаців»

Клас SentenceType призначений для роботи із таблицею «Типи речень». Основні властивості класу SentenceType наведені в таблиці 3.15.

Таблиця 3.16 – Основні властивості класу SentenceType

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Типи речень»
Name	Повертає або задає значення колонки «Назва» для відповідного запису із таблиці «Типи речень»

Клас Sentence призначений для роботи із таблицею «Речення». Основні властивості класу Sentence наведені в таблиці 3.16.

Таблиця 3.16 – Основні властивості класу Sentence

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Речення»
Text	Повертає або задає значення колонки «Текст» для відповідного запису із таблиці «Речення»
Number	Повертає або задає значення колонки «Номер» для відповідного запису із таблиці «Речення»
ParagraphID	Повертає або задає значення вторинного ключа «ID Абзацу», який вказує на ID абзацу якому належить речення
TypeID	Повертає або задає значення вторинного ключа «ID Типу», який вказує на ID типу речення

Клас Term призначений для роботи із таблицею «Терміни». Основні властивості класу Term наведені в таблиці 3.17.

Таблиця 3.17 – Основні властивості класу Term

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Терміни»
Name	Повертає або задає значення колонки «Назва» для відповідного запису із таблиці «Терміни»

Клас TermInSentence призначений для роботи із таблицею «Терміни». Основні властивості класу TermInSentence наведені в таблиці 3.18.

Таблиця 3.18 – Основні властивості класу TermInSentence

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Терміни»
CurrentForm	Повертає або задає значення колонки «Поточна форма терміну» для відповідного запису із таблиці «Терміни»
TermID	Повертає або задає значення вторинного ключа «ID Терміну», який вказує на ID терміну, якому належить якому належить поточна форма терміну в речені
SentenceID	Повертає або задає значення вторинного ключа «ID Речення», який вказує на ID речення, якому належить поточна форма терміну в реченні

Клас TermInSentence призначений для роботи із таблицею «Терміни в реченнях». Основні властивості класу TermInSentence наведені в таблиці 3.19.

Таблиця 3.19 – Основні властивості класу TermInSentence

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Терміни»
CurrentForm	Повертає або задає значення колонки «Поточна форма терміну» для відповідного запису із таблиці «Терміни»
TermID	Повертає або задає значення вторинного ключа «ID Терміну», який вказує на ID терміну, якому належить якому належить поточна форма терміну в речені
SentenceID	Повертає або задає значення вторинного ключа «ID Речення», який вказує на ID речення, якому належить поточна форма терміну в реченні

Клас KeyedTerm призначений для роботи із таблицею «Ключові терміни». Основні властивості класу KeyedTerm наведені в таблиці 3.20.

Таблиця 3.20 – Основні властивості класу KeyedTerm

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Ключові терміни»
Estimate	Повертає або задає значення колонки «Поточна форма терміну» , що відповідає за оцінку ключового терміна
TermID	Повертає або задає значення вторинного ключа «ID Терміну», який вказує на ID терміну, що визначений як ключовий в меж рубрики ІНМ
HeaderID	Повертає або задає значення вторинного ключа «ID Загловку», який вказує на ID заголовку, якому належить ключовий термін

Клас TestTask призначений для роботи із таблицею «Тестові завдання». Основні властивості класу TestTask наведені в таблиці 3.21.

Таблиця 3.21 – Основні властивості класу TestTask

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Тестові завдання»
Text	Повертає або задає значення колонки «Текст» для відповідного запису із таблиці «Тестові завдання»
KeyedTermID	Повертає або задає значення вторинного ключа «ID Ключового терміну», який вказує на ID ключового терміну, для якого згенеровано тестове завдання
SentenceID	Повертає або задає значення вторинного ключа «ID Речення», який вказує на ID речення, на основі якого згенеровано тестове завдання
TypeID	Повертає або задає значення вторинного ключа «ID Типу», який вказує на ID типу тестового завдання

Клас TestTaskType призначений для роботи із таблицею «Тестові завдання». Основні властивості класу TestTaskType наведені в таблиці 3.22.

Таблиця 3.22 – Основні властивості класу TestTaskType

Назва	**Опис**
ID	Повертає або задає значення первинного ключа запису таблиці «Тестові завдання»
Name	Повертає або задає значення колонки «Назва» для відповідного запису із таблиці «Тестові завдання»

Для кожного класу, що надають можливість працювати із записами таблиць бази даних, розроблено відповідні класи «репозиторії», які інкапсулюють всі потрібні запити до відповідної таблиці. Таким чинному в даному розділі було розроблено модуль, який працює із базою даних.

3.7.3 Модуль «Генерація тестових завдань»

Даний модуль являється основним в поточній роботі оскільки саме він відповідає за головне завдання проекту – генерацію тестових завдань. Класом, який організовує роботу даного модуля є «TestGenerator». Він реалізовує інтерфейс «ITestGenerator» і реалізовує його методи. Опис даних методів наведений в таблиці 3.23.

Таблиця 3.23 – Основні методи класу TestGenerator

Назва	**Опис**
Generate	Організовує генерацію тестових завдань.
SaveTestToRepository	Зберігає згенеровану множину тестових завдань в БД
GenerateParallel	Організовує генерацію тестових завдань в паралельному режимі для кожної секції.

Наступні 6 класів реалізовані для побудови тестових завдань. Серед цих класів «MainTestBuilder» є основним і виконує в більшості роль розподілу генерації тестових завдань по більш спеціалізованих для цього класах. Даний

клас реалізовує інтерфейс «IMainTestBuilder». Реалізовані методи описані в таблиці 3.24.

Таблиця 3.24– Основні методи класу MainTestBuilder

Назва	**Опис**
BuildTask(ISentences, Term)	Запускає генерацію тестових завдань по переданому реченню та ключовому терміні. Передає виконання на спеціалізовані класи для генерації тестів кожного виду.
BuildTask(List<ISentences>, Term)	Запускає генерацію тестових завдань по списку речень та ключовому терміні. Передає виконання на спеціалізовані класи для генерації тестів кожного виду.
FinalizeTaskBuild	Виконує завершальний етап генерації тестових завдань.

Таблиця 3.25 – Основні методи класу BaseTaskBuilder

Назва	**Опис**
ApplyStyleAttributes(XmlNode, string)	Застосовує стильові ознаки, передбачені тегом продукційного правила для значення, яке відповідає переданому тегу.
GetMaskRandomTagValue(XmlNode, TestTaskWorkpiece)	Метод повертає значення для тегів, що відповідають за випадкові значення. Даний метод є віртуальний, що дає можливість перевизначити його в класі наслідника.
GetMaskTagValue(XmlNode, TestTaskWorkpiece)	Метод повертає значення для всіх тегів (окрім тегів для випадкових значень). Даний метод є віртуальний, що дає можливість перевизначити його в класі наслідника.
CreateTask(TestTaskWorkpiece)	Створює тестове завдання по отриманому об’єкті TestTaskWorkpiece
FinalizeTaskBuild()	Завершує генерацію тестових завдань. Під час виконання даного методу відбувається підстановка випадкових значень та застосування стилів.

Для кожного виду тестових завдань реалізовані спеціальні класи. Генерація тестових завдань із можливістю вибору декількох відповідей реалізована в класі MultipleChoiceTestBuilder, з можливістю вибору однієї відповіді – SingleChoiceTestBuilder, введення відповіді – InputAnswerTestBuilder, з вибором варіанту «Так» або «Ні» – YesNoTestBuilder. Вони наслідують абстрактний клас BaseTaskBuilder в якому реалізовано базовий функціонал, а при потребі перевизначають функції під свою спеціалізацію. В таблиці 3.25 наведений опис методів класу BaseTaskBuilder.

3.7.6 Модуль «Експорт тестових завдань в GIFT»

Поточний модуль приймає на вхід список тестових завдань для експорту в GIFT форматі та шлях до місця, де повинен зберегтися файл із експортованими тестовими завданнями.

Модуль містить інтерфейс ITestTaskConverter із одним методом Convert. Даний інтерфейс слугує контрактом для конвертації різних типів тестових завдань у різні формати. На даний момент в системі реалізовано 3 класи конвертації тестових завдань в GIFT формат:

– YesNoConverter – відповідає за конвертацію тестових завдань логічного типу;

– InputAnswerConverter – відповідає за конвертацію тестових завдань із введенням відповіді;

– SingleChoiceConverter – відповідає за конвертацію тестових завдань із одною або багатьма правильними відповідями.

Вхідні дані даного модуля передаються в метод Export класу GiftTestTasksExporter, який реалізує інтерфейс ITestTasksExporter. Даний клас відповідає за організацію робити вище перерахованих класів конвертерів і зберігає результат конвертації в файл за шляхом, отриманим у вхідних параметрах.

3.7.7 Модуль «Експорт тестових завдань в Moodle XML»

Вхідними даним модуля є список тестових завдань, який потрібно експортувати в xml файл в форматі Moodle XML, та фізичний шлях місця збереження файлу з тестами. Вхідні дані даного модуля передаються в метод Export класу XmlTestTaskExporter, який реалізує інтерфейс ITestTasksExporter. Клас XmlTestTaskExporter відповідає за організацію роботи класів конвертації тестових завдань різних типів.

Модуль містить базовий клас BaseTaskToXmlConverter із набором базових методів для конвертації тестових завдань підтримуваних типів в формат XML. Класи, які відповідають за конвертацію конкретного типу тестового завдання, успадковують клас BaseTaskToXmlConverter та розширюють функціонал специфічною логікою конвертації для відповідного типу. До класів конвертації тестових завдань належать:

– YesNoConverter – конвертує тестове завдання логічного типу в XML формат;

– InputAnswerConverter – конвертує тестове завдання із введенням відповіді в XML формат;

– MultyChoiceConverter – конвертує тестове завдання із одною або багатьма правильними відповідями відповіді в XML формат.

Результатом роботи модуля є збережений файл із розширенням xml із тестовими завданнями в форматі Moodle XML.

3.7.7 Взаємодія модулів системи

Робота інформаційної системи розпочинається з модуля «Графічний інтерфейс», де користувач має можливість обрати ІНМ для генерації тестових завдань.

Обраний ІНМ, для якого потрібно згенерувати тестові завдання, та об’єкт класу MaskManager із завантаженими правилами продукції тестових завдань передаються на вхід підсистеми генерації тестових завдань в клас TestGenerator. Генерація тестових завдань розпочинається з виклику методу Generate об’єкта класу TestGenerator. Даний метод за допомогою взаємодії із модулем, який відповідає за роботу з базою даних, отримує перелік рубрик ІНМ та циклічно передає кожну рубрику в метод BuildTask класу MainTaskBuilder. Метод BuildTask за допомогою модуля роботи із базою даних отримає речення, які містять ключові терміни.

Отримавши речення, метод переходить до роботи із правилами продукції тестових завдань, а саме циклічно проходить по реченням із ключовими термінами та перевіряє, чи кожне речення відповідає правилам, визначеним в антецеденті, і, якщо перевірка пройшла успішно, то речення передаються у спеціальні класи для генерації тестових завдань, а саме: MultipleTestBuilder, SingleChoiceTestBuilder, YesNoTestBuilder та InputAnswerTestBuilder. В кожен з перерахованих класів речення передається в метод TaskBuild. Він обробляє отримане речення і формує всю можливу інформацію для майбутнього тестового завдання, але не створює самого завдання. Це пов’язано із тим, що тестові завдання можуть містити випадкові варіанти відповідей або інші елементи, які не можна отримати одразу з одного речення. Для цього сформована інформація зберігається в серединні класу.

Безпосередньо формування кінцевих тестових завдань відбувається в методах FinalizeTaskBuild кожного із класів, які викликаються після того як кожне речення було оброблено методом TaskBuild.

Множина згенерованих тестових завдань повертається в метод Generate класу TestGenerator який передає її в метод SaveTestToRepository для збереження її в БД працюючи із модулем «Робота із базою даних».

Після завершення процесу генерації тестових завдань та збереження згенерованих тестових завдань в базу даних, виконання переходить до модуля

«Графічний інтерфейс», який надає можливість переглядати згенеровані тестові завдання, редагувати і видаляти.

Висновки до розділу 3

В даному розділі проведено аналіз функцій експериментальної інформаційної системи автоматизованого генерування тестових завдань та було виділено функції, які вона повинна виконувати: завантаження вхідних даних у вигляді семантичної структури ІНМ, зчитування продукційних правил, генерація тестових завдань з однією правильною відповіддю, генерація тестових завдань із декількома правильними відповідями, генерація тестових завдань із можливістю введення відповіді, генерація тестових із варіантами відповіді «Так» або «Ні», редагування тестових завдань, експорт тестових завдань в середовище для тестування. Проведено огляд можливих форматів завантаження множини автоматизовано згенерованих тестових завдань в середовище Moodle. З-поміж розглянутих форматів було обрано формати GIFT та Moodle XML.

Проведено розробку набору тегів для формалізованого подання правил продукції тестових завдання для інформаційної технології генерації тестових завдань.

Відповідно визначеним функціям інформаційної системи, було розроблено архітектуру інформаційної системи автоматизованої генерації тестових завдань, яка складається із наступних підсистем: підсистеми роботи з графічним інтерфейсом; підсистеми генерації тестових завдань; підсистеми генерації тестових завдань; підсистеми роботи з правилами продукції; підсистеми роботи з базою даних; підсистеми експорту тестових завдань у форматі GIFT; підсистеми експорту тестових завдань у форматі Moodle XML. Проведено розробку модулів інформаційної системи, програмна реалізація яких буде забезпечувати виконання функцій інформаційної системи, що дає можливість автоматизовано генерувати тестові завдання до ІНМ.

Розділ 4

Дослідження ефективності інформаційної технології автоматизованої генерації тестових завдань

4.1 Дослідження функціональності експериментальної інформаційної системи генерації тестових завдань

В даній роботі розроблена інформаційна система для автоматизованої генерації тестових завдань для навчального курсу на основі ключових термінів кожної рубрики ІНМ та з можливістю експорту згенерованих тестових завдання в файл у GIFT форматі.

При запуску файлу AIS.TestGenerator.UI.exe першим з'являється діалог для введення вхідних параметрів системи (рисунок 4.1). В даному вікні відображається перелік доступних ІНМ із БД. При виборі ІНМ стає доступна кнопка «Згенерувати», при натиску на яку запуститься процес генерації тестових. Також доступна можливість перегляду попередньо згенерованих тестових завдань для обраного ІНМ.

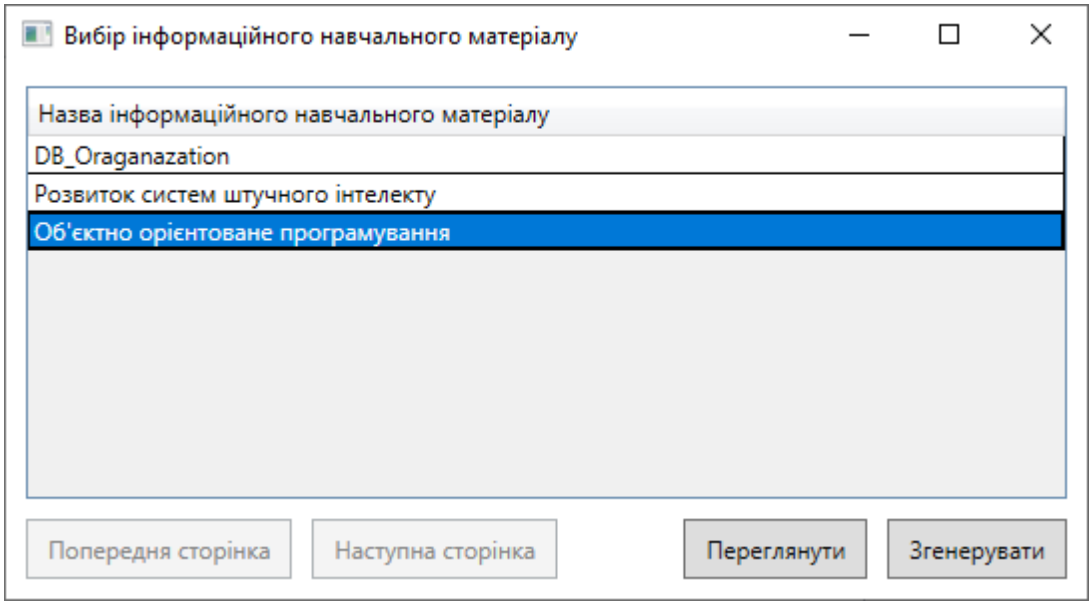


Рисунок 4.1 – Вікно для вибору інформаційного навчального матеріалу

Після завершення процесу генерації тестових завдань з’явиться діалогове вікно зображене на рисунку 4.2.

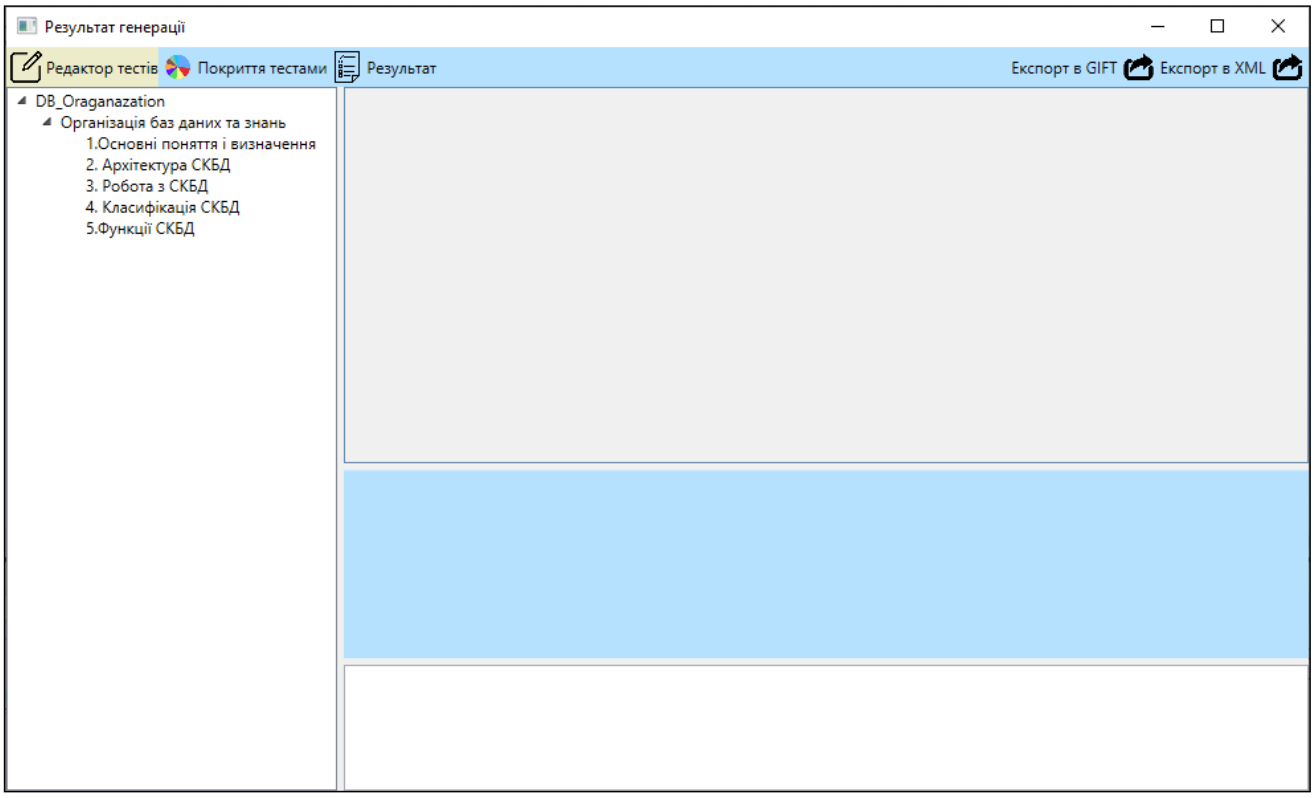


Рисунок 4.2 – Вікно результату генерації тестових завдань

Вікно «Результат генерації тестових завдань» має дві робочі області (рисунок 4.3). Перша область відповідає за навігацію між різними режимами перегляду. Друга область відповідає за відображення режиму перегляду. Поточне вікно запускається із обраним в навігаційному меню елемента «Редагування тестів».

У вікні «Результат генерації» при обраному в навігаційній панелі елемента «Редактор тестів» відображається дерево із структурою текстового документу. При виборі секції в дереві структури документу з’являється список тестових завдань згрупований по ключовим термінам (рисунок 4.4).

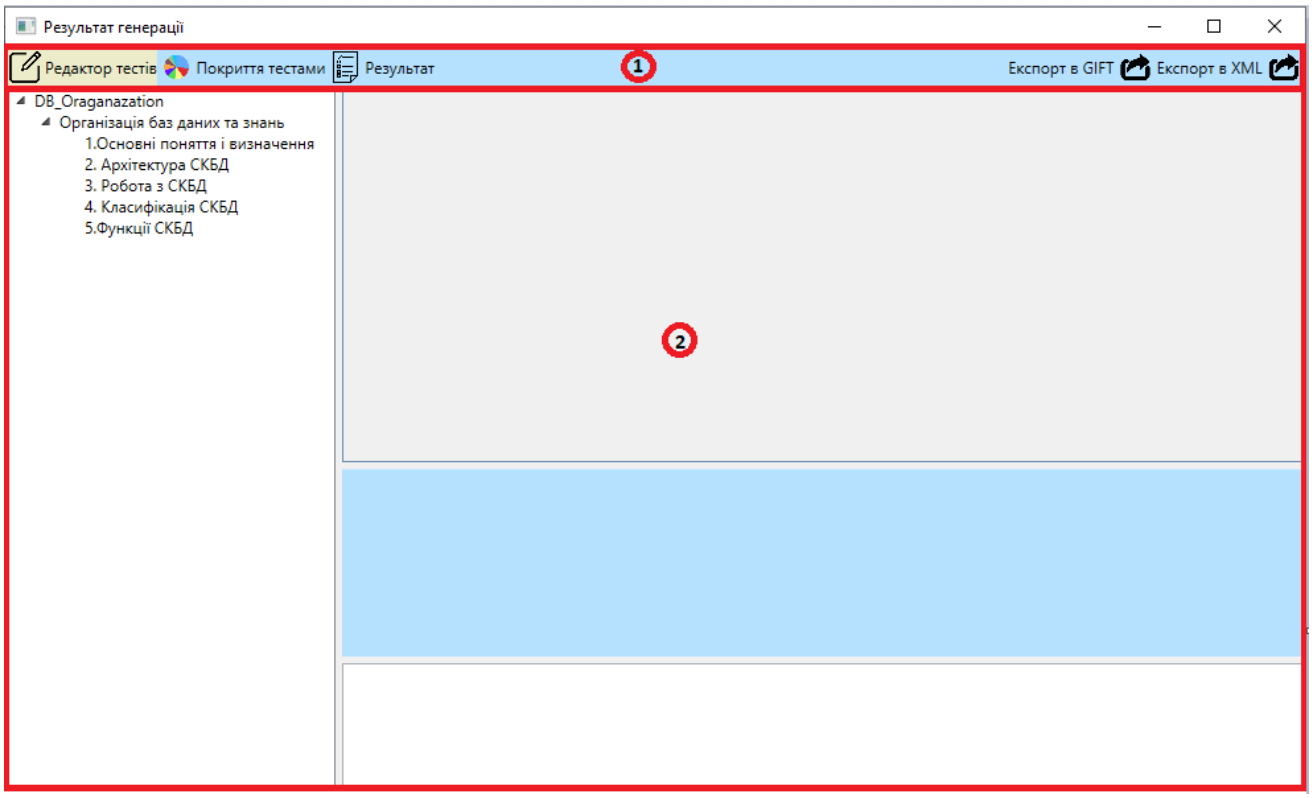


Рисунок 4.3 – Робочі області вікна «Результат генерації тестових завдань»

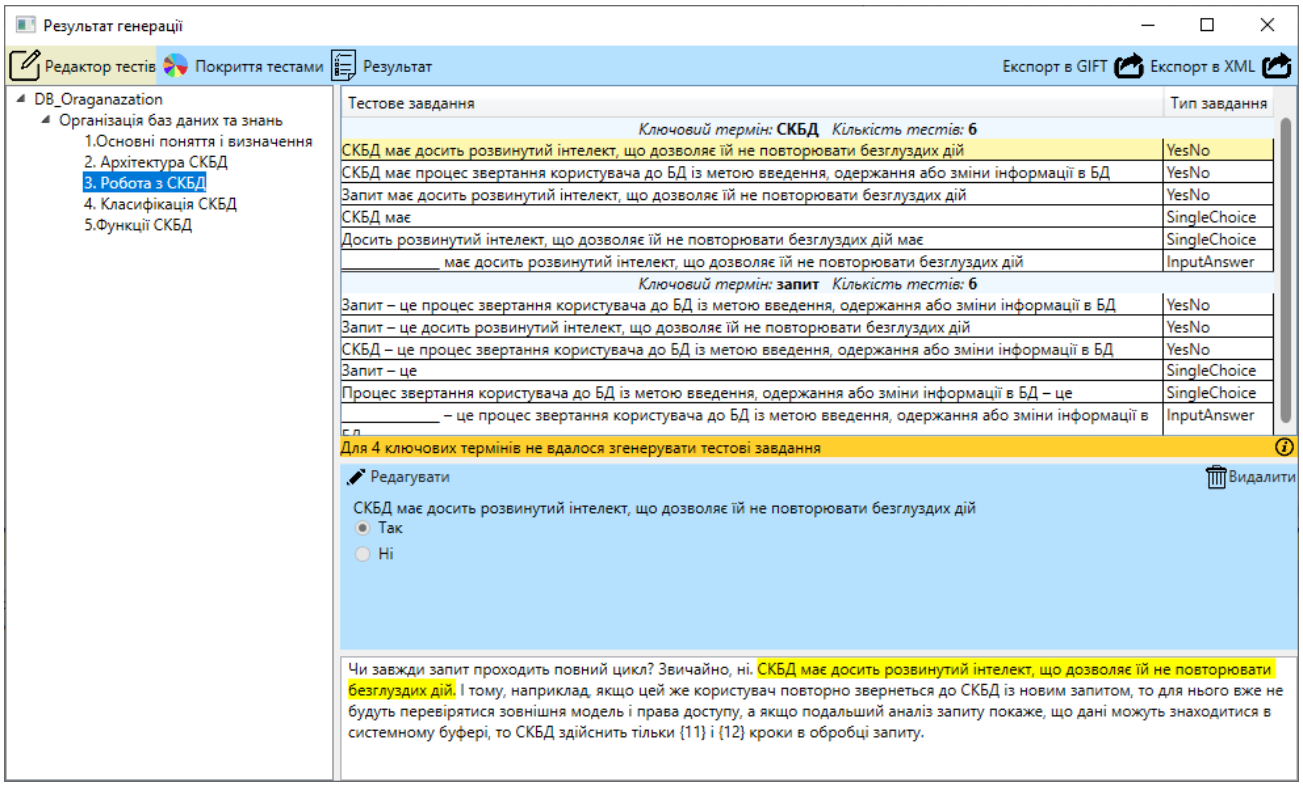


Рисунок 4.4 – Відображення ключових термінів для обраної секції

Якщо в межах обраної рубрики для певних ключових термінів не вдалося згенерувати тестові завдання, тоді під списком тестових завдань буде відображено відповідне повідомлення для користувача. Для перегляду списку невикористаних ключових термінів потрібно натиснути відповідну кнопку (рисунок 4.5).

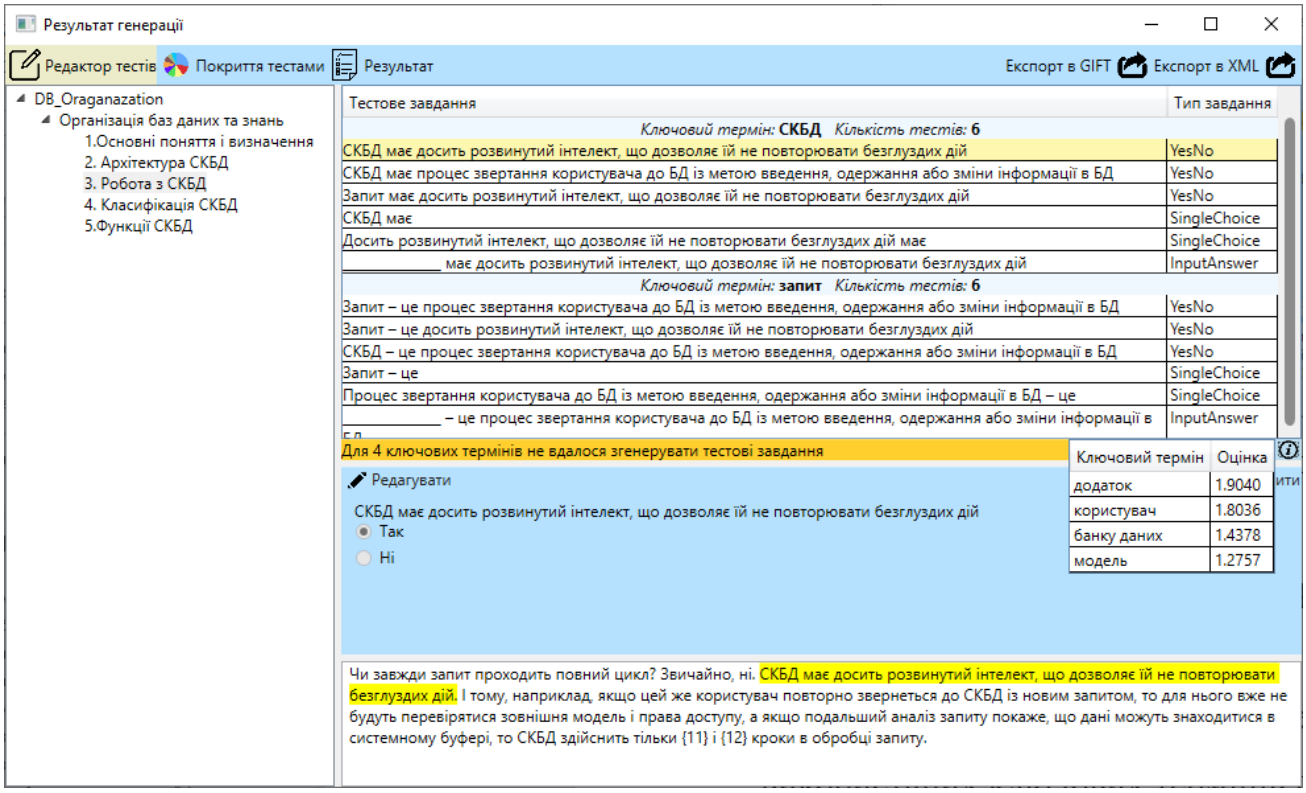


Рисунок 4.5 – Відображення невикористаних ключових термінів

Обране тестове завдання із переліку тестових завдання відображається в під списком у відповідному форматі в залежності від типу тестового завдання. Обране завдання можна відредагувати натиснувши кнопку «Редагувати» і видаляти – натиснувши на однойменну кнопку. Також нижче під тестовим завданням відображається фрагмент контенту із якого згенеровано тестове завдання.

Перехід в навігаційній панелі до категорії «Покриття тестами» надає коритувачу можливіть перевірити, яка частина текстового контетну покрита тестовими завданнями. Коритувачу потрібно натиснути на фрагмент тексту,

який цікавить і якщо по обраному фрагменті існують згенеровані тетові завдання, то вони будуть відображені в правій частині вікна. На рисунку 4.6 зображено приклад використання категорії «Покриття тестами».

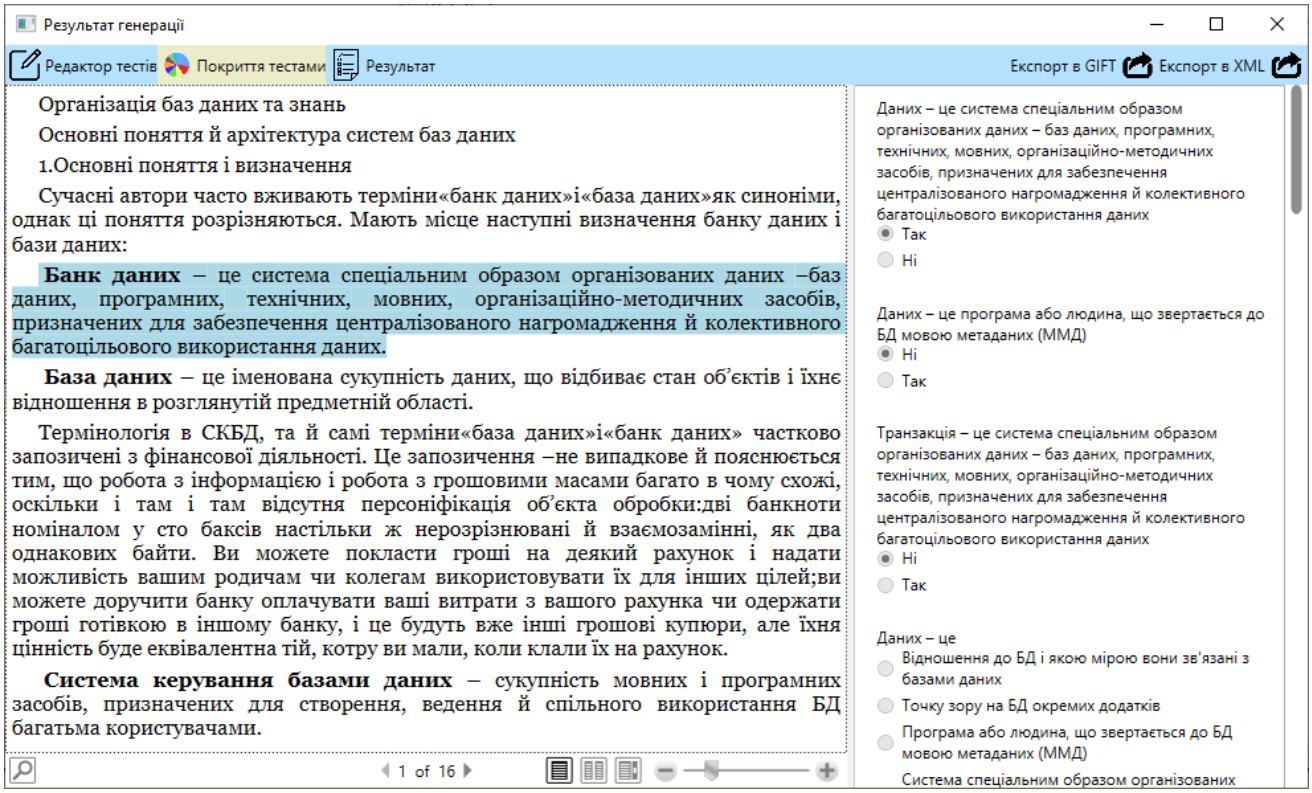


Рисунок 4.6 – Приклад роботи в категорії «Покритя тестами»

В категорії «Результат» реалізовано можливіть перегляду всіх згенерованих тестових завдань для кожної рубрики ІНМ. Приклад роботи зображений на рисунку 4.7.

Після обробки згенерованих тестових завдань можна експортувати тестові завдання в середовище для тестування. В додатку реалізовано експорт в формат GIFT та Moodle XML, які описанні в розділі 3.1.2 та 3.1.3 і використовується в Moodle LMS. Для того, щоб розпочати експорт потрібно натиснути на кнопку «Експорт в GIFT» або «Експорт в XML». Після натиску з'являється діалог для вибору місця збереження файлу із експортованими тестовими завданнями, та введення назви файлу (рисунок 4.8).

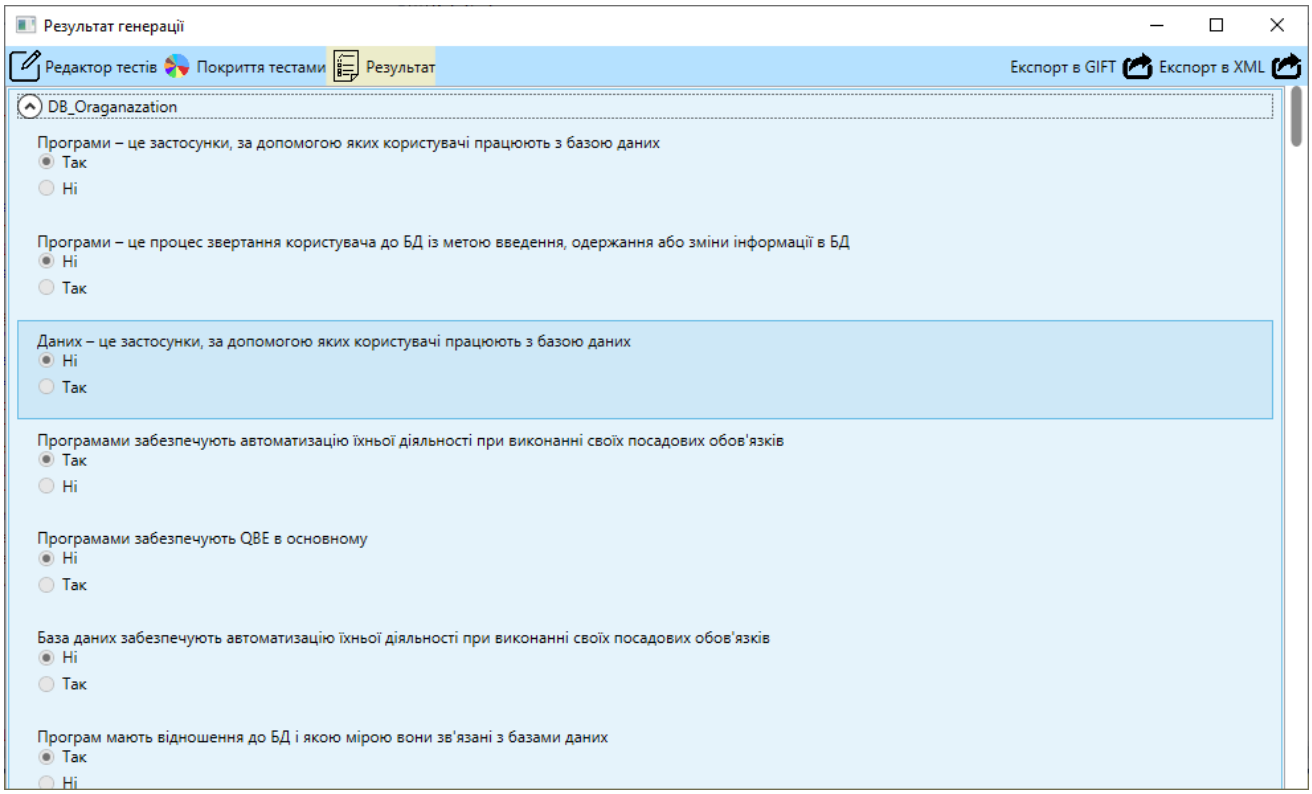


Рисунок 4.7 – Скріншот обраної категорії «Результат»

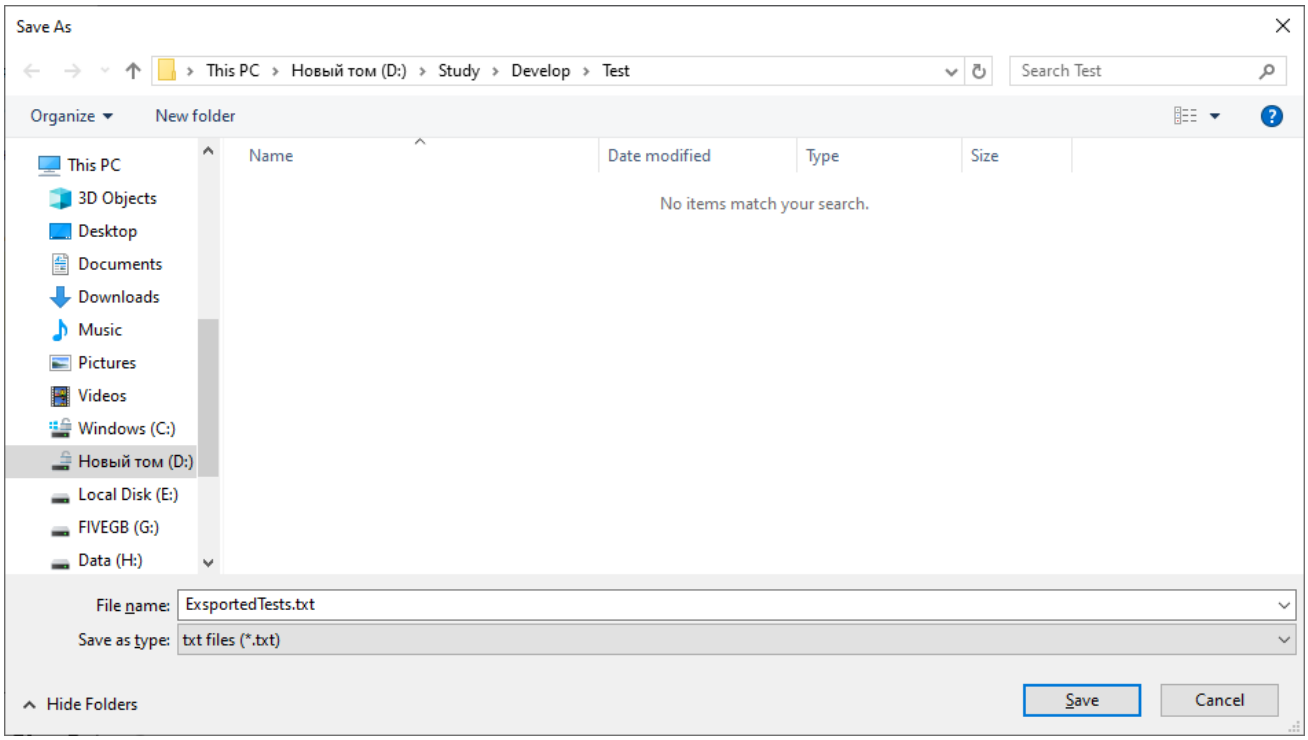


Рисунок 4.8 – Вибір назви і папки для збереження експортованих тестових завдань

Після вибору місця збереження файлу, тестові завдання будуть експортовані у відповідну папку. На рисунку 4.9 зображено приклад контенту файлу із експортованими тестовими завданнями у форматі GIFT.

Рисунок 4.9 – Приклад контенту файлу із експортованими тестовими завданнями у форматі GIFT

Щоб використати тестові завдання в системі Moodle потрібно перейти в навчальний курс та створити елемент «Тест» і перейти в режим його редагування. Для імпорту згенерованих тестових завдань потрібно перейти в банк питань і обрати вкладку «Імпорт». В обраній вкладці вибрати формат експортованих тестовий завдань (в даному випадку GIFT) та вибрати файл із тестовими завданнями (рисунок 4.10).

В результаті імпорту тестових завдань буде сформований Банк питань для обраного тесту (рис. 4.11).

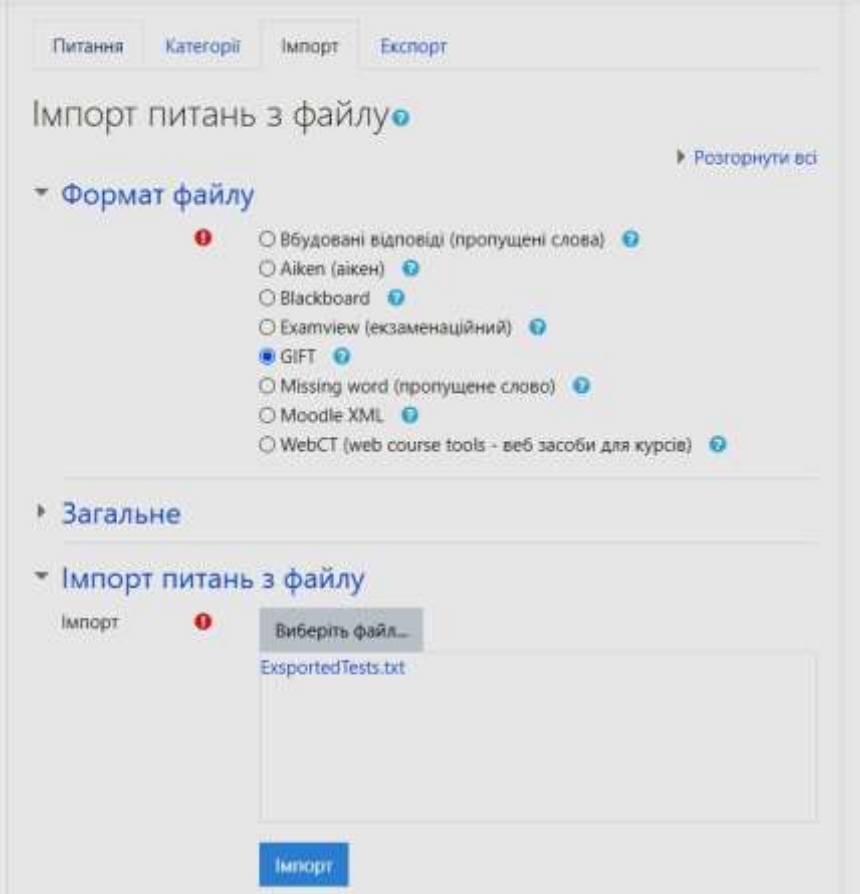


Рисунок 4.10 – Імпорт тестових завдань в систему Moodle

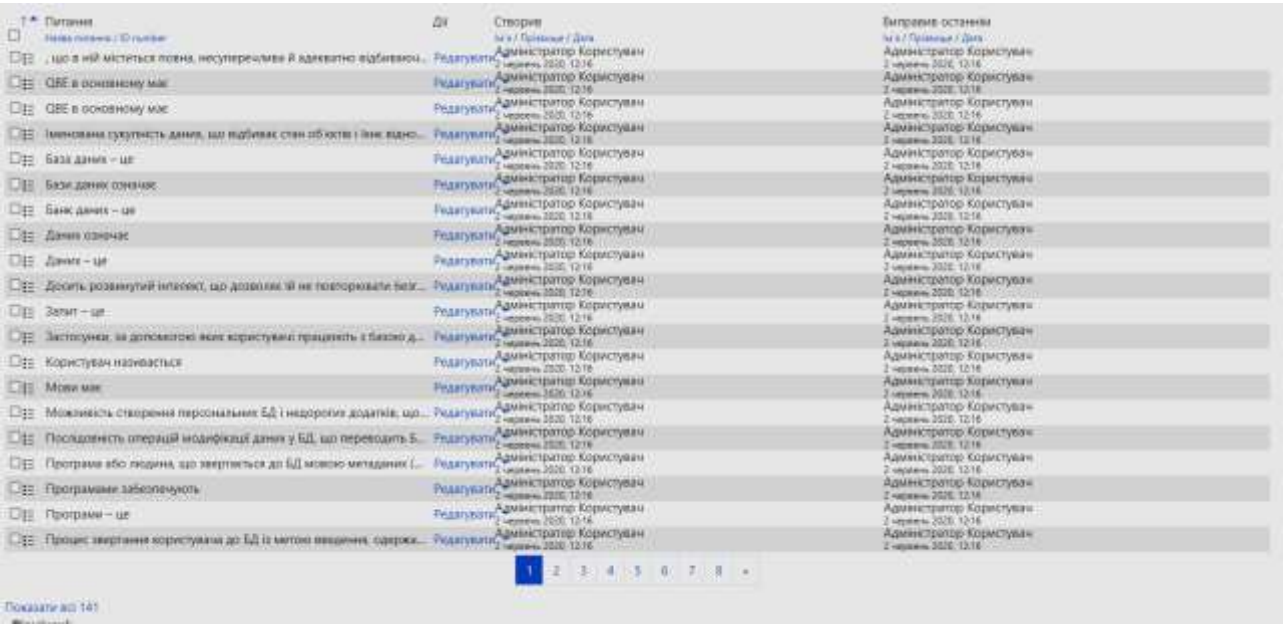


Рисунок 4.11 – Сформований Банк питань на основі імпортованих тестових завдань

Отже, в даному розділі було проведено дослідження функціональності розробленої інформаційної технології з автоматизованої генерації тестових завдань шляхом опису і перевірки бізнес-процесів у роботі програмного продукту, починаючи із вибору ІНМ і закінчуючи експортом згенерованих тестових завдань в форматі GIFT та імпортом завдань до середовища Moodle.

4.2 Тестування функціональності інформаційної системи

4.2.1 Покриття функціоналу Unit тестами

Для тестування розробленої системи були проведене модульне тестування [41]. При тестуванні класу MasksManager було реалізовано 3 тести. Тест Ctor_ShouldThrowExeption призначений для перевірки того, що при створені об’єкту із переданими пустим параметром в конструктор, буде отримана помилка. Тест має наступну реалізацію:

```
[Test]
public void Ctor_ShouldThrowExeption()
{
      //Assert
      Assert.Throws<ArgumentNullException>(() =>
      {
            //Act
            MasksManager masksManager = new MasksManager(string.Empty);
      });
}
```

Тест Ctor_ShouldNotThrowExeption перевіряє, що при переданому не пустого параметру в конструктор класу помилок не буде. Юніт тест має наступну реалізацію:

```
[Test]
public void Ctor_ShouldNotThrowExeption()
{
      //Arange
      string filePath = "../../Mask/TestData/SingleMask.xml";
      Assert.DoesNotThrow(() =>
      {
```

```
            //Act
            MasksManager masksManager = new MasksManager(filePath);
      });
}
```

Тест LoadMask_ShouldLoadExpetedMasks перевіряє, що при виклику методу LoadMask продукційні правила будуть коректно завантажені із файла. Реалізація тесту наступна:

```
[Test]
public void LoadMask_ShouldLoadExpetedMasks()
{
      //Arange
      string expectedMask = "<signature><BeginSentence /><Connector /><TermGroup
/></signature>";
      string maskFilePath = @"../../Mask/TestData/SingleMask.xml";
      List<(string, string, string)> expectedTaskMasks = new List<(string, string,
string)>()
      {
            ("<signature><TermGroup case=\"upper\" /><Connector /><ThesisGroup
/></signature>", "<rightAnswer><TRUE /></rightAnswer>", "<wrongAnswer><FALSE
/></wrongAnswer>"),
            ("<signature><TermGroup case=\"upper\" /><Connector /></signature>",
"<rightAnswer><ThesisGroup case=\"upper\" /></rightAnswer>",
"<wrongAnswer><RandomThesisGroup case=\"upper\" /></wrongAnswer>"),
            ("<signature><ThesisGroup case=\"upper\" /><Connector /><InputSpace
/></signature>", "<rightAnswer><TermGroup case=\"upper\" /></rightAnswer>",
"<wrongAnswer />")
      };

      string maskFileFullPath = Path.Combine(AppContext.BaseDirectory, maskFilePath);
      MasksManager masksManager = new MasksManager(maskFileFullPath);

      //At
      masksManager.LoadMasks();

      //Assert
      Assert.AreEqual(1, masksManager.Mask.Count);
      Assert.AreEqual(expectedMask, masksManager.Mask[0].Mask.OuterXml);
      Assert.AreEqual(3, masksManager.Mask[0].Connectors.Count);
      Assert.AreEqual(3, masksManager.Mask[0].TaskMasks.Count);
      for (int i = 0; i < masksManager.Mask[0].TaskMasks.Count; i++)
      {
            TaskMask currentTaskMask = masksManager.Mask[0].TaskMasks[i];
            Assert.AreEqual(expectedTaskMasks[i].Item1,
currentTaskMask.QuestionMask.OuterXml);
            Assert.AreEqual(expectedTaskMasks[i].Item2,
currentTaskMask.RightAnswerMask.OuterXml);
            Assert.AreEqual(expectedTaskMasks[i].Item3,
currentTaskMask.WrongAnswerMask.OuterXml);
      }
}
```

Результат відпрацювання частини модульних тестів зображений на рисунку 4.12.

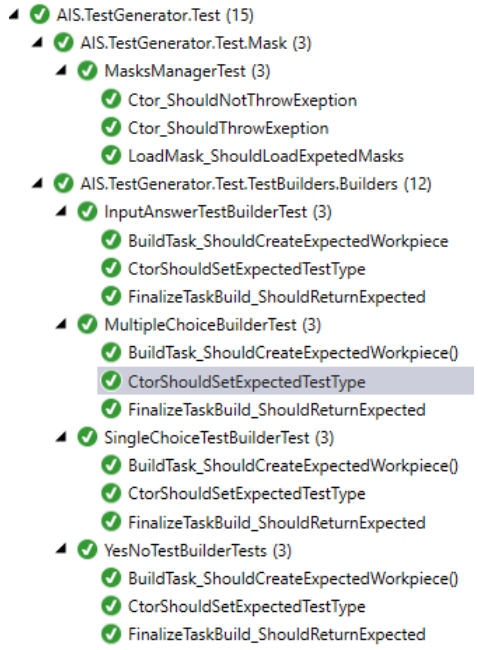


Рисунок 4.12 – Результат виконання юніт тестів

Аналогічним чином було протестовано основні методи інформаційної системи. Результат тестування Unit тестами визначив, що всі методи системи виконуються коректно.

4.2.2 Покриття функціоналу GUI тестами

Під час проведення тестування інтерфейсу програмного продукту відбувається імітація роботи користувача із додатком. Тестування інтерфейсу користувача може відбуватися в ручному режимі, коли спеціально підготовлена людина тестує програмний продукт, або в автоматизованому режимі за допомогою певного програмного забезпечення. Для тестування розробленої системи автоматизованої генерації тестових завдань було обрано

автоматизованих підхід до перевірки інтерфейсу програмного продукту за допомогою TestComplete [42].

Було реалізовано автоматизований тест, який покриває перевірку елементів інтерфейсу користувача в першому діалозі, який призначений для вибору документу та введення коефіцієнтів. Фрагмент лістингу реалізовано тесту наведено нижче:

```
function verifyMainWindowsControls(){
this.commonControlVerification.checkIsControlExist(this.mainWindow.getPathTextBox
(), this.logsControlName.filePath);
this.commonControlVerification.checkIsControlExist(this.mainWindow.getSelectButto
n(), this.logsControlName.browse);

this.commonControlVerification.checkIsControlExist(this.mainWindow.getBoldCoeffic
ientNumberUpDown(), this.logsControlName.bold);

this.commonControlVerification.checkIsControlExist(this.mainWindow.getItalicCoeff
icientNumberUpDown(), this.logsControlName.ittalic);
this.commonControlVerification.checkIsControlExist(this.mainWindow.getUnderlineCo
efficientNumberUpDown(), this.logsControlName.underline);
this.commonControlVerification.checkIsControlExist(this.mainWindow.getStartButton
(), this.logsControlName.run);
this.commonControlVerification.checkIsControlReadOnly(this.mainWindow.getPathText
Box(), this.logsControlName.filePath);
this.commonControlVerification.checkIsControlEnabled(this.mainWindow.getSelectBut
ton(), this.logsControlName.browse);
this.commonControlVerification.checkIsControlEnabled(this.mainWindow.getBoldCoeff
icientNumberUpDown(), this.logsControlName.bold);
this.commonControlVerification.checkIsControlEnabled(this.mainWindow.getItalicCoe
fficientNumberUpDown(), this.logsControlName.ittalic);

this.commonControlVerification.checkIsControlEnabled(this.mainWindow.getUnderline
CoefficientNumberUpDown(), this.logsControlName.underline);

this.commonControlVerification.checkIsControlDisabled(this.mainWindow.getStartBut
ton(), this.logsControlName.run);
  clickSelectButton();
  checkIsOpenDialogAppear();
  setPathToTestFile();
  clickOpenButton();

this.commonControlVerification.checkIsControlEnabled(this.mainWindow.getStartButt
on(), this.logsControlName.run);
}
```

Результат виконання тесту зображений на рисунку 4.14.

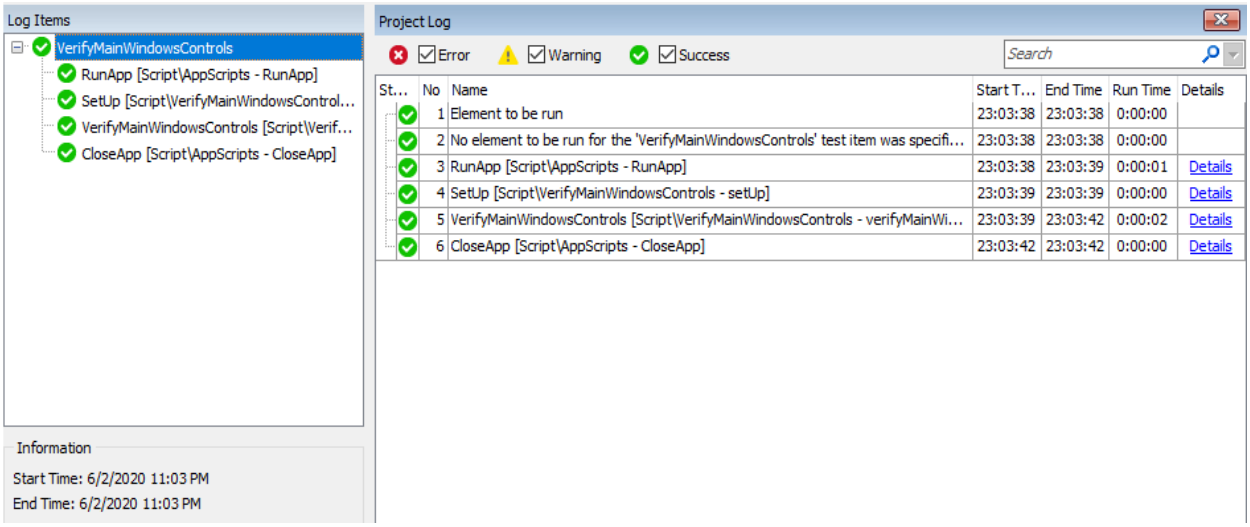


Рисунок 4.14 – Результат виконання автоматизованого UI тесту для діалогу введення даних

Наступним було реалізовано автоматизований тест для перевірки графічних елементів діалогу, який відображає результати генерації тестових завдань. Наступний лістинг відображає фрагмент реалізації тесту:

```
function verifyLeftSidebarInGenerationResultDialog(){
  clickSelectButton();
  setPathToTestFile();
  clickOpenButton();
  clickStartGenerationButton();
  waitTestGenerationResultDialog();

  const testEditorButton = this.testGenerationResult.getTestEditorButton();
  this.commonControlVerification.checkIsControlExist(testEditorButton,
this.logsControlName.testEditor);
  this.commonControlVerification.checkIsControlEnabled(testEditorButton,
this.logsControlName.testEditor);

  const unknownWordsButton = this.testGenerationResult.getUnknownWordsButton();
  this.commonControlVerification.checkIsControlExist(unknownWordsButton,
this.logsControlName.testEditor);
  this.commonControlVerification.checkIsControlEnabled(unknownWordsButton,
this.logsControlName.testEditor);

  const testCoverageButton = this.testGenerationResult.getTestCoverageButton();
  this.commonControlVerification.checkIsControlExist(testCoverageButton,
this.logsControlName.testEditor);
  this.commonControlVerification.checkIsControlEnabled(testCoverageButton,
this.logsControlName.testEditor);

  const generationResultButton =
this.testGenerationResult.getGenerationResultButton();
  this.commonControlVerification.checkIsControlExist(generationResultButton,
this.logsControlName.testEditor);
```

```
  this.commonControlVerification.checkIsControlEnabled(generationResultButton,
this.logsControlName.testEditor);

  const giftExportButton = this.testGenerationResult.getGiftExportButton();
  this.commonControlVerification.checkIsControlExist(giftExportButton,
this.logsControlName.testEditor);
  this.commonControlVerification.checkIsControlEnabled(giftExportButton,
this.logsControlName.testEditor);
       }
```

Результат виконання автоматизованого тесту зображений на рисунку 4.15

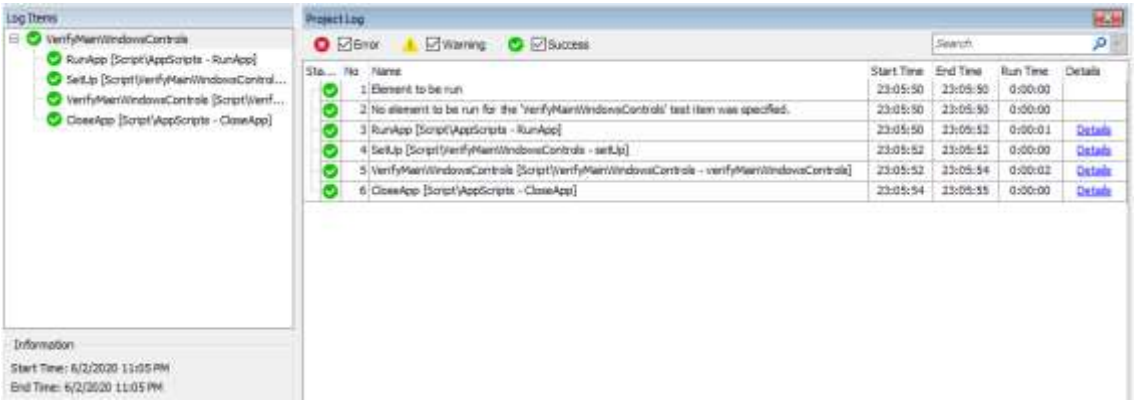


Рисунок 4.15 – Результат виконання автоматизованого UI тесту для діалогу із результатом генерації тестових завдань

В результаті реалізації автоматизованих UI тестів для розробленої інформаційної системи, було перевірено функціональність інтерфейсу, що підтвердило спроможність коректного виконання всіх функцій системи.

4.3 Дослідження покриття семантичної структури навчального курсу згенерованими тестовими завданнями

Для проведення дослідження була сформована тестова вибірка навчальних матеріалів із 20 дисциплін спеціальності «122 – Комп’ютерні науки». Обрані навчальні матеріали використовуються на кафедрі Комп’ютерних наук та інформаційних технологій Хмельницького національного університету в навчальному процесі. Вибірка формувалася на основі матеріалів із Модульного

середовища ХНУ [43] та Навчального центру заочно-дистанційної освіти ХНУ [44].

Дане дослідження призначене на визначення відсоткової частини контенту ІНМ, яка покрита згенерованими тестових завдань за допомогою ІТ, запропонованої в розділі 2.3. Для оцінки повноти покриття контенту ІНМ було визначено відношення T_H, T_P і T_S кількості використаних семантичних блоків ІНМ (рубрик, абзаців, речень) для генерації тестових завдань до загальної кількості відповідних семантичних блоків ІНМ:

$$T_H = \frac{UB_H}{AB_H},\ T_P = \frac{UB_P}{AB_P},\ T_S = \frac{UB_S}{AB_S} \tag{4.1}$$

де UB_H, UB_P і UB_S – кількість семантичних блоків, які використано для генерації тестового завдання; AB_H, AB_P і AB_S – загальна кількість відповідних семантичних блоків в контенті ІНМ. Для визначення середнього значення покриття визначеної вибірки тестовими завданнями використовуються наступні відношення:

$$\overline{T_H} = \frac{\sum_{i=0}^{n} T_{H,i}}{n}, \overline{T_P} = \frac{\sum_{i=0}^{n} T_{P,i}}{n}, \overline{T_S} = \frac{\sum_{i=0}^{n} T_{S,i}}{n} \tag{4.2}$$

де n – загальна кількість ІНМ у виборці.

В результаті проведення дослідження було визначено, що для тестової вибірки середнім значенням покриття тестовими завданнями семантичних блоків ІНМ є: $\overline{T_H} = 94.72\%, \overline{T_P} = 74.87\%, \overline{T_S} = 52.23\%$. Мінімальними та максимальними показниками покриття визначено:

– для рубрик: $T_{H,min} = 89.93\%$, $T_{H,max} = 100\%$

– для абзаців: $T_{P,min} = 55.68\%$, $T_{P,max} = 94.37\%$

– для речень: $T_{S,min} = 33.52\%$, $T_{S,max} = 71.12\%$

Наведені значення покриття тестовими завданнями ІНМ графічно відображені на рисунку 4.16.

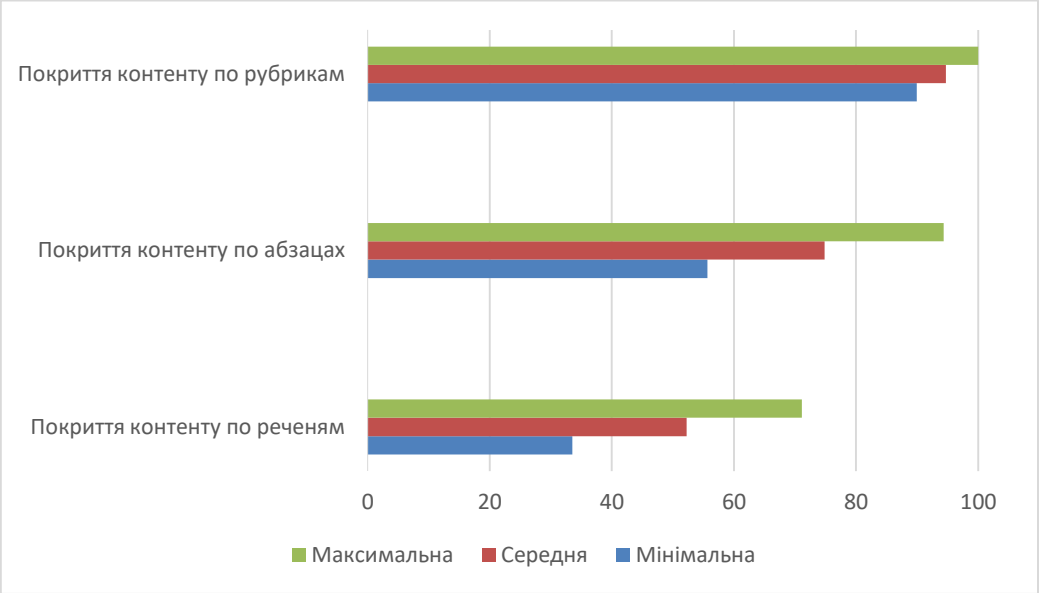


Рисунок 4.16 – Граничні та середні показники покриття тестовими завданнями рубрик, абзаців та речень

Варто відзначити, що запропонована ІТ базується на продукційних правилах генерації тестових завдань, в яких обов'язковою частиною антецедентів є ключовий термін, тому його поява в реченні ІНМ слугує своєрідним маркером для генерації тестового завдання. Оскільки, ключові терміни в реченнях можуть замінятися синонімами або займенниками, в такому випадку антецедент правила продукції не спрацює. Також в навчальному матеріалі часто використовуються різного роду пояснювальні приклади для ключових термінів, але водночас приклади можуть не включати ключові терміни. Саме тому відсоток покриття речень тестовими завданнями не є таким великим, в порівнянні із абзацами та рубриками.

4.4 Дослідження коректності автоматизовано згенерованих тестових завдань

Дослідження проводиться з метою визначення величини коректно автоматизовано згенерованих тестових завдань для ІНМ за допомогою розробленої ІТ. Для проведення дослідження використовувалися автоматизовано

згенеровані тестові завдання до вибірки ІНМ, що використовувалася в досліджені, яке описане в пункті 4.3.

Для проведення оцінки коректно автоматизовано згенерованих тестових завдань потрібно визначити:

– відсоток *CT* тестових завдань, які не потребують редагування;

– відсоток *ET* відредагованих тестових завдань;

– відсоток *DT* видалених тестових завдань.

Для розрахунку перелічених величин в межах однієї рубрики ІНМ потрібно визначити відношення CT_H, ET_H і DT_H за наступними формулами:

$$CT_H = \frac{NCT}{nt}, ET_H = \frac{NET}{nt}, DT_H = \frac{NDT}{nt} \quad (4.3)$$

де nt – загальна кількість автоматизовано згенерованих тестових завдань для рубрики ІНМ; *NCT* – кількість тестових завдань, що не потребують змін; *NET* – кількість відредагованих тестових завдань; *NDT* – кількість видалених тестових завдань.

Для визначення *CT*, *ET* і *DT* в межах одного ІНМ використовуються наступні формули:

$$CT = \frac{\sum_{i=0}^{nh} CT_{H,i}}{nh}, ET = \frac{\sum_{i=0}^{nh} ET_{H,i}}{nh}, DT = \frac{\sum_{i=0}^{nh} CT_{H,i}}{nh} \quad (4.4)$$

де *nh* – кількість рубрик в ІНМ.

Щоб визначити середнє значення відредагованих *ET*, видалених *DT* та використаних без змін автоматизовано згенерованих тестових завдань *CT* використовуються наступні відношення:

$$\overline{CT} = \frac{\sum_{i=0}^{n} CT_i}{n}, \overline{ET} = \frac{\sum_{i=0}^{n} ET_i}{n}, \overline{DT} = \frac{\sum_{i=0}^{n} DT_i}{n} \quad (4.5)$$

де *n* – кількість ІНМ в тестовій вибірці.

В результаті проведеного дослідження встановлено, що середні значення $\overline{CT}, \overline{ET}$ і $\overline{DT}$ для тестової вибірки ІНМ склали (рисунок 4.17):

$$\overline{CT} = 24.38\% \ , \overline{ET} = 26.75\%, \overline{DT} = 48.87\%$$

Рисунок 4.17 – Відсоткове співвідношення використаних без змін, відредагованих та видалених тестових завдань

Отримані значення видалених автоматизовано згенерованих тестових завдань $\overline{DT} = 48.87\%$ аргументуються надлишковістю автоматизовано згенерованих тестових завдань, а не їхньою некоректністю. Середній відсоток відкоригованих тестових завдань $\overline{ET} = 26.75\%$ спричинений потребую узгодження слів у тестовому завдані по відмінках і роду. Згенеровані тестові завдання, які не потребували редагування склали $\overline{CT} = 24.38\%$, а в результуючі вибірці тестових завдань частина не редагованих тестових завдань склала 47.68%.

Висновки до розділу 4

В розділі було проведено дослідження функціональності розробленої експериментальної інформаційної системи автоматизованої генерації тестових завдань. Система надає можливість для обраного ІНМ автоматизовано згенерувати тестові завдання та переглядати множину тестових завдань; можна переглядати множину тестових завдань по рубрикам ІНМ в межах якої тестові завдання було згенеровано. Також ІС дозволяє видаляти та редагувати

згенеровані тестові завдання для покращення результуючої вибірки тестових завдань. Кінцеву вибірку тестових завдань можна експортувати в файл формату GIFT або Moodle XML для подальшого завантаження тестів в систему Moodle.

Для експериментальної ІС було проведено тестування за допомогою модульних та GUI тестів. Проведено дослідження покриття ІНМ автоматизовано згенерованими тестовими завданнями й встановлено, що середні показники покриття тестами на рівні заголовків складають 94,72%, абзаців – 74.87%, речень – 52.23%.

Проведено дослідження шляхів подальшого використання множини автоматизовано згенерованих тестових завдань та встановлено, що 24,38% тестових завдань користувачем було включено в кінцеву вибірку тестових завдань без редагування, 26,75% згенерованих тестових завдань було відредаговано та включено в кінцеву вибірку, а 48,87% видалено через їх надлишковість.

Загальні висновки

В результаті виконання дипломної роботи магістра було розв’язано науково-технічну задачу створення інформаційної технології автоматизованої генерації тестових завдань. У рамках роботи було поставлено та вирішено наступні завдання:

1) проведено аналіз сучасних методів автоматизації генерації тестів для визначення існуючих можливостей генерації тестових завдань із повним покриттям семантичної структури навчального матеріалу;

2) вдосконалено інформаційну модель семантичної структури навчального курсу для забезпечення можливості з достатньою для генерації тестових завдань інформативністю виконувати формальне подання навчальних матеріалів;

3) розроблено метод автоматизованої генерації тестових завдань до навчальних матеріалів за допомогою правил продукції;

4) розроблено інформаційну технологію автоматизованої генерації тестових завдань за допомогою отриманих моделі та методу;

5) розроблено інформаційну систему генерації тестових завдань на основі розробленої інформаційної технології;

6) проведено прикладне дослідження ефективності інформаційної технології генерації тестових завдань шляхом дослідження функціональності й ефективності застосування відповідної експериментальної інформаційної системи.

В результаті проведеної роботи були отримані такі наукові положення:

– вдосконалено інформаційну модель семантичної структури навчального курсу, яка відрізняється тим, що містить подання навчальних матеріалів у вигляді системи рубрикації, множин ключових термінів до кожної із рубрик та тестових завдань для перевірки ключових термінів у визначених місцях їх використання;

– розроблено новий метод автоматизованої генерації тестових завдань до навчальних матеріалів, що дозволяє генерувати тестові завдання за контентом навчального матеріалу на основі правил продукції;

– розроблено нову інформаційну технологію автоматизованої генерації тестових завдань, що дозволяє за поданням семантичної структури інформаційного навчального матеріалу одержувати множину тестових завдань, призначених для перевірки рівня знань визначених термінів;

– розроблено нову інформаційну систему генерації тестових завдань, що за створеною інформаційною технологією дозволяє генерувати та експортувати в середовище тестування множини тестових завдань.

Розроблена експериментальна інформаційна система виконує наступні функції: завантаження вхідних даних у вигляді семантичної структури ІНМ, зчитування продукційних правил, генерація тестових завдань з однією правильною відповіддю, генерація тестових завдань із декількома правильними відповідями, генерація тестових завдань із можливістю введення відповіді, генерація тестових із варіантами відповіді «Так» або «Ні», редагування тестових завдань, експорт тестових завдань в середовище для тестування.

Проведене прикладне дослідження функціональності експериментальної інформаційної системи підтвердило її відповідність розробленій інформаційній технології й визначило можливість виконувати перетворення вхідних даних у вихідні, відповідно до послідовності кроків інформаційної технології автоматизованої генерації тестових завдань. За результатом використання експериментальної інформаційної системи вдалося встановити, що розроблена модель навчального курсу, метод та ІТ дозволяє отримати наступний ефект:

– покриття семантичної структури ІНМ множиною згенерованих тестових завдань складає 94.72% для рубрик, 74.87% для абзаців та 52.23% для речень;

– можливість використовувати згенеровані тестові завдання для перевірки рівня знань визначених ключових термінів у межах визначених рубрик ІНМ;

– можливість контролю використання тестових завдань у тестуванні залежно від обсягу наявних у контенті термінів різних рівнів семантичної значущості;

– можливість використання створених множин тестових завдань для адаптивного тестування.

Наукові положення дипломної роботи автором висвітлено в 5 наукових публікаціях [47, 48, 49, 50, 51], в тому числі 1 стаття – в фаховому виданні [47], включеному в перелік МОН України. За виконання дипломної роботи магістра прийнято участь у науково-практичних конференціях: Всеукраїнській науково-практичній конференції молодих науковців і студентів «Інтелектуальний потенціал – 2018»; XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп’ютерних наук АПКН-2019»; XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп’ютерних наук АПКН-2020».

Перелік посилань

1. Задворна С. Г. Структурування навчальної інформації: основні дидактичні аспекти / С. Г. Задворна // Вісник Черкаського університету. Серія «Педагогічні науки» Вип. 196. Ч. 2. С. 39–43.

2. Бєляк О. М. Структура навчальної інформації як складова підготовки студентів немовних спеціальностей / О. М. Бєляк // Наука і освіта – 2014 – С.12-15.

3. Рубрикація тексту [Електронний ресурс]. – Режим доступу: https://kozhemyaka.at.ua/index/rubrikacija_tekstu/0-56

4. Структура тексту [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Структура _тексту

5. Ковалів Ю.І. / Літературознавча енциклопедія. Київ – 2007.

6. Анотування і реферування [Електронний ресурс]. – Режим доступу: https://pidru4niki.com/1513061640666/dokumentoznavstvo/anotuvannya_referuvannya_naukovih_tekstiv

7. Великий тлумачний словник сучасної української мови / В. Т. Бусел. – Київ; Ірпінь – 2005.

8. Ключове слова [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Ключове_слово

9. Семантичне ядро [Електронний ресурс]. – Режим доступу: https://igroup.com.ua/seo-articles/cemantychne-yadro/

10. Методи і форми контролю успішності студентів [Електронний ресурс]. – Режим доступу: https://pidruchniki.com/70171/pedagogika/metodi_formi_kontrolyu_uspishnosti_studentiv

11. Кухар Л.О. Конструювання тестів / Л. О. Кухар, В. П. Сергієнко // Луцьк – 2010.

12. Форми тестових завдань. Форма подання тестового завдання [Електронний ресурс]. – Режим доступу:

https://pidru4niki.com/12590605/informatika/formi_testovih_zavdan_forma_podannya_testovogo_zavdannya

13. Методичні рекомендації щодо формування тестових завдань і тестів [Електронний ресурс]. – Режим доступу: https://science.lpnu.ua/sites/default/files/attach/2016/1677/26_metodychni_rekomendaciyi_-296-311.pdf

14. Титенко С. В. Генерація тестових завдань у системі дистанційного навчання на основі моделі формалізації дидактичного тексту [Електронний ресурс]. – Режим доступу: http://www.setlab.net/?view=Tytenko-test-generation

15. Мельник А. М. Автоматична генерація тестових завдань різних типів / А. М. Мельник // Вісник Хмельницького національного університету. Хмельницький – С.124-129.

16. Мельник А. М. Інформаційна технологія автоматичної генерації тестових завдань з керованою складністю / А. М. Мельник, Р. М. Пасічник, Р. П. Шевчук // Інформаційні технології в технічних системах – С.57-61.

17. Попов А. М. Проектування тестових завдань закритого типу на базі моделі онтології на основі когнітивних прототипів / А. М. Попов, А. М. Мельник // Медична інформатика та інженерія. Хмельницький – 2015. – С.46-51

18. Практична реалізація технології автоматизації тестування на основі понятійно-тезисної моделі [Електронний ресурс]. – Режим доступу: http://www.setlab.net/?view=Tytenko_Virt06

19. Титенко С. В. Генерація тестових завдань у системі дистанційного навчання на основі моделі формалізації дидактичного тексту / С. В. Титенко // Наукові вісті НТУУ «КПІ» – 2009 – С.47-57.

20. Аналіз методів генерації тестових завдань [Електронний ресурс]. – Режим доступу: http://www.setlab.net/?view=Tanchenko_IAI_2013_new

21. Дерябкін В.П. Інтелектуальна інформаційна система тестування знань / В.П. Дерябкін, С.А. Піявскій, Н.М. Пузанков // Міжнародна науково-технічна конференція «Перспективні інформаційні технології». Самара – 2015 – С. 141-145.

22. Системи управління навчанням [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Learning_management_system

23. Moodle [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Moodle

24. Скорюкова Я. Г. Особливості використання системи e-Learning Server 3000 у процесі вивчення графічних дисциплін./ Я. Г. Скорюкова, Собчук Н.В., О.В. Слободянюк, М.С. Гречанюк // Сучасні інформаційні технології та інноваційні методики навчання в підготовці фахівців: методологія, теорія, досвід, проблеми. Вінниця – 2017 – с. 171-175.

25. Дибля Н. О. Організація процесу дистанційного навчання з допомогою програмного продукту eLERNING Server 3000 / Н. О. Дибля, Д. Л. Кречман // Комп'ютерні інструменти в освіті, 2002. №3-4 – с. 123-138.

26. ATutor [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ATutor

27. Blackboard [Електронний ресурс]. – Режим доступу: https://www.blackboard.com/

28. Адаптивне тестування [Електронний ресурс]. – Режим доступу: https://vpa.org.ua/glossary/adaptivne-testuvannya/

29. Продукційна модель [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Продукційна_модель

30. Імпорт тестових завдань в середовище LMS Moodle [Електронний ресурс]. – Режим доступу: https://docs.moodle.org/310/en/Import_questions

31. GIFT формат [Електронний ресурс]. – Режим доступу: https://docs.moodle.org/38/en/GIFT_format

32. Moodle XML формат [Електронний ресурс]. – Режим доступу: https://docs.moodle.org/38/en/Moodle_XML_format

33. .Net Framework [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/.NET_Framework

34. .Net Framework [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview

35. Net Core [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/core/introduction

36. Windows Presentation Foundation [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8

37. Windows Forms [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/windows-forms-overview?view=netframeworkdesktop-4.8

38. Microsoft SQL Server [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server

39. Entity Framework [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/entityframework/1.1.php

40. MVVM [Електронний ресурс]. – Режим доступу: https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1

41. Модульне тестування [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Модульне_тестування

42. TestComplete структура [Електронний ресурс]. – Режим доступу: https://smartbear.com/product/testcomplete/features/

43. Модульне середовище для навчання ХНУ [Електронний ресурс]. – Режим доступу: https://msn.khnu.km.ua/

44. Навчальний центр заочно-дистанційної освіти ХНУ. [Електронний ресурс]. – Режим доступу: https://de.khnu.km.ua/p.aspx

45. Білоус Г. А. Параметризація моделі тестового завдання при автоматизованому формуванні тестів / Г. А. Білоус, О. В. Мазурець // Матеріали VII Міжнародної науково-практичної конференції «Інформаційні управляючі системи та технології ICST-ODESSA-2018». Одеса – 2018. – С.64-66.

46. Бармак О. В. Інформаційна технологія автоматизованого формування тестових завдань /О. В. Бармак, О. В. Мазурець, В. І. Кліменко // Вісник

Хмельницького національного університету. Технічні науки. – 2017. – № 5. – С. 93–103.

47. Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В Ковальчук, В. О. Слободзян // Вісник Хмельницького національного університету. Хмельницький – 2018. – С.61-69.

48. Ковальчук О. В. Використання програмного розширення Spire.Doc для автоматизації роботи з цифровими документами / О. В. Ковальчук, Г. А Білоус, В. О. Слободзян // Збірник наукових праць за матеріалами XI всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2019». Хмельницький – 2019 – с. 116-122.

49. Ковальчук О. В. Дослідження практичної ефективності інформаційної технології автоматизованого визначення семантичних термінів навчальних матеріалів / О.В. Ковальчук, О.В. Мазурець. // Сучасні технології в механіці : зб. наук. пр., Хмельницький – 2018 – с. 141-148 .

50. Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В Ковальчук, В. О. Слободзян, Г. А. Білоус. // Інтелектуальний потенціал – 2018 : збірник наукових праць молодих науковців і студентів, Хмельницький – 2018 – с. 51-56 .

51. Ковальчук О. В. Метод генерації тестових завдань до навчальних матеріалів на основі продукційних правил / О. В Ковальчук, О. В. Мазурець, // Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020». Хмельницький – 2020 – с. 51-56.

ДОДАТКИ

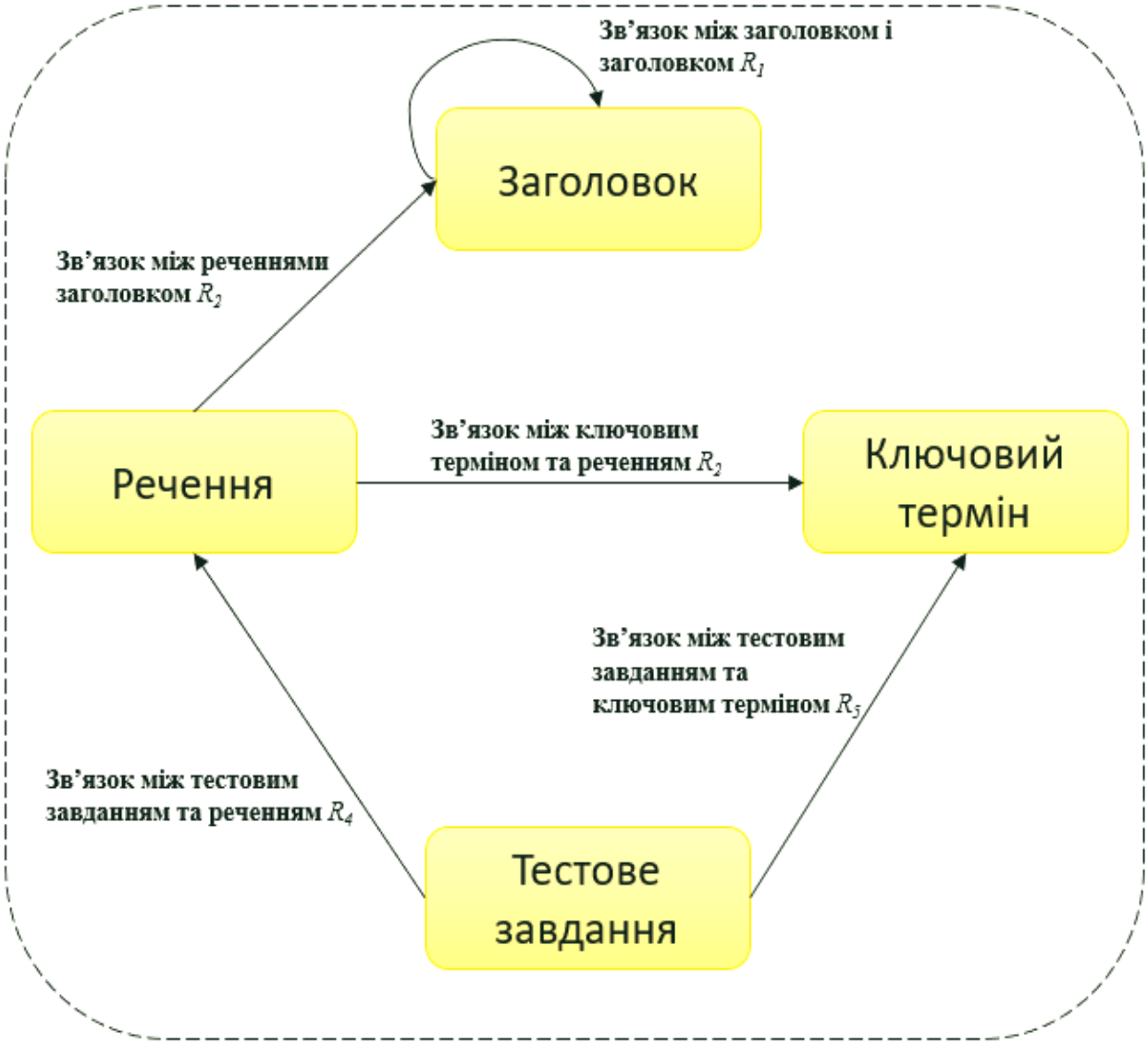
Додаток А

Результати аналізу відомих методів автоматизованої генерації тестових завдань

Назва	Опис
Методи з використанням семантичних мереж (Танченко С. С., Титенко С. В., Гагарин А. А.)	В основі методу покладено використання бази знань, структурною одиницею якої є ланцюг «поняття – зв'язок – поняття». Базу знань заповнює експерт (викладач) на основі інформаційного навчального матеріалу. Тестові завдання генеруються шляхом видаленням одного із елементів ланцюга та постановкою запитання про відсутню ланку. Недоліком даного методу є надзвичайно великі трудові затрати на формування і оновлення бази знань відповідно до навчального матеріалу.
Метод параметризованих тестових завдань (Брусиловський П., Каші Е., Патгак С.)	Метод полягає у формуванні набору тестових завдань на основі шаблонів, в яких змінюється певний параметр на автоматично згенероване значення за допомогою певної формули або алгоритму. Таким чином може бути згенерована велика кількість однотипних завдань, які при тестуванні будуть індивідуальними для кожної особи, яка проходить тестування. Недоліком методу є предметна спрямованість: метод підходить лише для перевірки специфічних за природою практичних знань у точних науках.
Метод створення тестових завдань за понятійно-тезисною моделлю (Титенко С.В., Гагарін О.О.)	В методі використовується база знань, в яку входять «поняття» та «тези» із навчального матеріалу. Тестове завдання створюється на основі шаблону із підстановкою елементів із множини «понять» і «тез». Недоліком методу є великі затрати часу на заповнення бази знань.
Метод автоматизованого формування тестових завдань за ключовими термінами (Мазурець О. В., Бармак О. В.)	Побудова тестових завдань для перевірки знань визначеного терміну за семантичною структурою навчального матеріалу. Дозволяє створювати велику кількість прототипів тестових завдань різних типів та різної педагогічної якості. Недолік – метод не дозволяє вести повний контроль контенту інформаційного навчального матеріалу, який задіяно при тестуванні внаслідок того, що не ведеться облік всіх використаних фрагментів ІНМ для генерації тестового завдання.
Метод генерації тестових завдань до формалізованих за системою семантичних класів (Мельник А. М., Пасічник Р. М.)	Метод ґрунтується на формалізації навчальних матеріалів за допомогою системи семантичних класів, що дозволяє поділити текст на сукупності тверджень. Кожне твердження розбивається на компоненти, які описують процеси, поняття або їх характеристики, які поєднуються сполучниками. Для генерації тесту деяке твердження розбивається на основну та альтернативну частини, які можуть містити одну або декілька компонент. Альтернативна частина тесту поповнюється аналогічними за лінгвістичним змістом, синтаксично узгодженими, частинами інших тверджень. Недоліком методу є потреба в попередньому ручному заповнені семантичних класів відповідно до навчального матеріалу.

Додаток Б

Схема зв’язків інформаційного навчального матеріалу



Додаток В

Розгорнута структура класів автоматизованої системи автоматизованої системи генерації тестових завдань

Додаток Г

Програмні коди

Проєкт AIS.TestGenerator

Лістинг
AIS.TestGenerator\ISectionContainTestTask.cs

```
using AIS.TermSearcher.Entities.Section;
using System.Collections.Generic;

namespace AIS.TestGenerator
{
 public interface ISectionContainTestTask : INewSection
 {
 List<TestTask.TestTask> TestTasks { get; set; }
 }
}
```

Лістинг
AIS.TestGenerator\ITestGenerator.cs

```
using System;
using System.Collections.Generic;

namespace AIS.TestGenerator
{
 public interface ITestGenerator
 {
 Dictionary<Guid, List<TestTask.TestTask>> TestTasks { get; }

 Dictionary<Guid, List<TermDefinition>> TermDefinitions { get; }

 void Generate();

 void GenerateParallel();
 }
}
```

Лістинг
AIS.TestGenerator\SectionContainTestTask.cs

```
using AIS.DocxReader.Entities.Section;
using AIS.TermSearcher.Entities.Section;
using System.Collections.Generic;

namespace AIS.TestGenerator
{
 class SectionContainTestTask : NewSection, ISectionContainTestTask
 {
 public List<TestTask.TestTask> TestTasks { get; set; }

 public SectionContainTestTask(INewSection section, List<TestTask.TestTask> testTasks)
 {
 this.Level = section.Level;
 this.Text = section.Text;
 this.Name = section.Name;
 this.Guid = section.Guid;
 this.KeyTerms = section.KeyTerms;
 this.Paragraphs = section.Paragraphs;
 this.Sections = section.Sections;
 this.TestTasks = testTasks;
 }

 public SectionContainTestTask(INewSection section) : this(section, new List<TestTask.TestTask>())
 {
 }
 }
}
```

Лістинг
AIS.TestGenerator\SentencesPart.cs

```
namespace AIS.TestGenerator
{
 class SentencesPart
 {
 string Name { get; set; }
 string Value { get; set; }
 }
}
```

Лістинг
AIS.TestGenerator\TermDefinition.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TestGenerator.Masks;
using System.Text.RegularExpressions;

namespace AIS.TestGenerator
{
 public class TermDefinition
 {
 public Term Keyterm { get; set; }

 public string Definition { get; set; }

 public GroupCollection DefinitionPart { get; set; }
 }
}
```

Лістинг
AIS.TestGenerator\TestGenerator.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Section;
using AIS.DocxReader.Entities.Sentences;
using AIS.TermSearcher.Entities.Section;
using AIS.TermSearcher.Entities.Term;
using AIS.TestGenerator.Masks;
using AIS.TestGenerator.TestBuilders;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace AIS.TestGenerator
{
 internal class TestGenerator : ITestGenerator
 {
 private readonly IDocument document;
 private readonly Dictionary<Guid, List<Term>> keyterms;
 private readonly IMasksManager masksManager;

 public TestGenerator(IDocument document, Dictionary<Guid, List<Term>> keyterms, IMasksManager masksManager)
 {
 if (document == null)
 {
 throw new ArgumentNullException(nameof(document));
 }
 if (keyterms == null)
 {
 throw new ArgumentNullException(nameof(keyterms));
 }
 if (masksManager == null)
 {
 throw new ArgumentNullException(nameof(masksManager));
 }

 this.document = document;
 this.keyterms = keyterms;
 this.masksManager = masksManager;
 }

 public Dictionary<Guid, List<TestTask.TestTask>> TestTasks { get; private set; }

 public Dictionary<Guid, List<TermDefinition>> TermDefinitions { get; private set; }

 public void Generate()
 {
 this.TermDefinitions = new Dictionary<Guid, List<TermDefinition>>();
 this.TestTasks = new Dictionary<Guid, List<TestTask.TestTask>>();
 for (int i = 0; i < document.Sections.Count; i++)
 {
 document.Sections[i] = GenerateSection(document.Sections[i]);
 }
 }

 public void SaveToGift(Guid guidSection)
 {
 }

 public ISection GenerateSection(ISection section)
 {
 INewSection analizedSection = section as INewSection;
 if (analizedSection == null)
 {
 throw new Exception("тип секцій не підтримується для аналізу");
 }

 ISectionContainTestTask testTaskSection = new SectionContainTestTask(analizedSection);
 for (int i = 0; i < section.Sections.Count; i++)
 {
 testTaskSection.Sections[i] = GenerateSection(section.Sections[i]);
 }

 MainTestBuilder testBuilder = new MainTestBuilder(this.masksManager);
 foreach (var keyTerm in analizedSection.KeyTerms)
 {
 string joinedKeyTerm = string.Join("|", keyTerm.AllExistedForm.Select(x => $"\\b{x.Text}\\b"));
 var sentencesContainedKeyTerm = from paragraph in section.AllParagraphs
 from sentence in paragraph.Sentences
 where Regex.Match(sentence.Text, joinedKeyTerm, RegexOptions.IgnoreCase).Success
 && sentence.SentencesType == ESentencesType.Declarative
 select sentence;
```

```
testBuilder.BuildTask(sentencesContainedKeyTerm.ToList(), keyTerm);
 }

 testTaskSection.TestTasks = testBuilder.FinalizeTaskBuild();
 return testTaskSection;
 }

 public void GenerateParallel()
 {
 Parallel.For(0, document.Sections.Count, sectionIndex =>
 {
 document.Sections[sectionIndex] = GenerateSectionParallel(document.Sections[sectionIndex]);
 });
 }

 private ISection GenerateSectionParallel(ISection section)
 {
 ISectionContainTestTask testTaskSection = new SectionContainTestTask(section as INewSection);
 Parallel.For(0, testTaskSection.Sections.Count, sectionIndex =>
 {
 testTaskSection.Sections[sectionIndex] = GenerateSectionParallel(testTaskSection.Sections[sectionIndex]);
 });

 MainTestBuilder testBuilder = new MainTestBuilder(this.masksManager);
 foreach (var keyTerm in testTaskSection.KeyTerms)
 {
 string joinedKeyTerm = string.Join("|", keyTerm.AllExistedForm.Select(x => $"\\b{x.Text}\\b"));
 var sentencesContainedKeyTerm = from paragraph in testTaskSection.AllParagraphs
 from sentence in paragraph.Sentences
 where Regex.Match(sentence.Text, joinedKeyTerm, RegexOptions.IgnoreCase).Success
 && sentence.SentencesType == ESentencesType.Declarative
 select sentence;


testBuilder.BuildTask(sentencesContainedKeyTerm.ToList(), keyTerm);
 }

 testTaskSection.TestTasks = testBuilder.FinalizeTaskBuild();
 return testTaskSection;
 }
 }
}
```

Лістинг AIS.TestGenerator\TestGeneratorFactory.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Entities.Term;
using AIS.TestGenerator.Masks;
using System;
using System.Collections.Generic;

namespace AIS.TestGenerator
{
 public static class TestGeneratorFactory
 {
 public static ITestGenerator CreateTestGenerator(IDocument document, Dictionary<Guid, List<Term>> keyWord, IMasksManager masksManager)
 {
 return new TestGenerator(document, keyWord, masksManager);
 }

 public static IMasksManager CreateMaskManager(string pathToCriterions)
 {
 return new MasksManager(pathToCriterions);
 }
 }
}
```

Лістинг AIS.TestGenerator\Export\ITestTaskConverter.cs

```
namespace AIS.TestGenerator.Export
{
 interface ITestTaskConverter<T>
 {
 T Convert(TestTask.TestTask testTask);
 }
}
```

Лістинг AIS.TestGenerator\Export\ITestTasksExporter.cs

```
using System.Collections.Generic;

namespace AIS.TestGenerator.Export
{
 public interface ITestTasksExporter
 {
 void Export(List<TestTask.TestTask> testTasks, string filePath);
 }
}
```

Лістинг AIS.TestGenerator\Export\TestTaskExporterFactory.cs

```
using AIS.TestGenerator.Export.GIFT;

namespace AIS.TestGenerator.Export
{
 public class TestTaskExporterFactory
 {
 public ITestTasksExporter GetGiftTestTaskExporter()
 {
 return new GiftTestTasksExporter();
 }
 }
}
```

Лістинг AIS.TestGenerator\Export\GIFT\GiftTestTasksExporter.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Threading.Tasks;
using AIS.TestGenerator.Export.GIFT.TaskToGiftConverter;
using AIS.TestGenerator.TestTask;

namespace AIS.TestGenerator.Export.GIFT
{
 internal class GiftTestTasksExporter : ITestTasksExporter
 {
 private readonly Dictionary<ETestType, ITestTaskConverter<string>> converters = new Dictionary<ETestType, ITestTaskConverter<string>>()
 {
 { ETestType.YesNo, new YesNoConverter() },
 { ETestType.SingleChoice, new SingleChoiceConverter() },
 { ETestType.MultipleChoice, new SingleChoiceConverter() },
 { ETestType.InputAnswer, new InputAnswerConverter() }
 };

 public void Export(List<TestTask.TestTask> testTasks, string filePath)
 {
 StringBuilder taskStringBuilder = new StringBuilder();
 Parallel.ForEach(testTasks, (item) =>
 {
 string taskGift = ConvertTaskToGift(item);
 lock (taskStringBuilder)
 {

taskStringBuilder.Append(Environment.NewLine);
 taskStringBuilder.Append(taskGift);

taskStringBuilder.Append(Environment.NewLine);
 }
 });

 using (FileStream stream = File.Create(filePath))
 {
 using (StreamWriter file = new StreamWriter(stream))
 {

file.Write(taskStringBuilder.ToString());
 }
 }
 }

 private string ConvertTaskToGift(TestTask.TestTask testTask)
 {
 ITestTaskConverter<string> convertor = this.converters[testTask.TestType];
 string convertResult = convertor.Convert(testTask);
 return convertResult;
 }
 }
}
```

Лістинг AIS.TestGenerator\Export\GIFT\TaskToGiftConverter\GiftConstants.cs

```
namespace AIS.TestGenerator.Export.GIFT.TaskToGiftConverter
{
 static class GiftConstants
 {
 public const string GiftTaskTemplate = @"{0} {{{1}}}";

 public const string RigthAnswer = "=";
 public const string WrongAnswer = "~";
 public const string Conformity = "->";
 public const string True = "T";
 public const string False = "F";
 }
}
```

Лістинг AIS.TestGenerator\Export\GIFT\TaskToGiftConverter\InputAnswerConverter.cs

```
using System.Text;
using AIS.TestGenerator.TestTask;

namespace AIS.TestGenerator.Export.GIFT.TaskToGiftConverter
```

```
{
 class       InputAnswerConverter       :
ITestTaskConverter<string>
 {
 public string Convert(TestTask.TestTask
testTask)
 {
 StringBuilder   optionBuilder   =   new
StringBuilder();
 foreach (AnswerOption answerOption in
testTask.AnswerOptions)
 {

optionBuilder.AppendLine($"{GiftConstants
.RigthAnswer} {answerOption.Answer}");
 }

 string          convertedTask          =
string.Format(GiftConstants.GiftTaskTempl
ate,                        testTask.Task,
optionBuilder.ToString());
 return convertedTask;
 }
 }
}
```

Лістинг AIS.TestGenerator\Export\GIFT\TaskToGiftConverter\SingleChoiceConverter.cs

```
using System.Text;
using AIS.TestGenerator.TestTask;

namespace
AIS.TestGenerator.Export.GIFT.TaskToGiftC
onverter
{
 class      SingleChoiceConverter      :
ITestTaskConverter<string>
 {
 public string Convert(TestTask.TestTask
testTask)
 {
 StringBuilder   optionBuilder   =   new
StringBuilder();
 foreach (AnswerOption answerOption in
testTask.AnswerOptions)
 {
 if (answerOption.IsRightAnswer)
 {

optionBuilder.AppendLine($"{GiftConstants
.RigthAnswer} {answerOption.Answer}");
 }
 else
 {

optionBuilder.AppendLine($"{GiftConstants
.WrongAnswer} {answerOption.Answer}");
 }
 }

 string          convertedTask          =
string.Format(GiftConstants.GiftTaskTempl
ate,                        testTask.Task,
optionBuilder.ToString());
 return convertedTask;
 }
 }
}
```

Лістинг AIS.TestGenerator\Export\GIFT\TaskToGiftConverter\YesNoConverter.cs

```
using System.Linq;
using AIS.TestGenerator.Resources;

namespace
AIS.TestGenerator.Export.GIFT.TaskToGiftC
onverter
{
 class          YesNoConverter          :
ITestTaskConverter<string>
 {
 public string Convert(TestTask.TestTask
testTask)
 {
 string             answer             =
testTask.AnswerOptions.First(x         =>
x.IsRightAnswer).Answer                 ==
MaskTagTranslate.True                   ?
GiftConstants.True : GiftConstants.False;

 string          convertedTask          =
string.Format(GiftConstants.GiftTaskTempl
ate, testTask.Task, answer);
 return convertedTask;
 }
 }
}
```

Лістинг AIS.TestGenerator\Masks\IMasksManager.cs

```
using System.Collections.Generic;

namespace AIS.TestGenerator.Masks
{
 public interface IMasksManager
 {
 List<XmlMask> Mask { get; }

 void LoadMasks();
 }
}
```

Лістинг AIS.TestGenerator\Masks\MasksManager.cs

```
using AIS.TestGenerator.TestTask;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Xml;

namespace AIS.TestGenerator.Masks
{
 public     class     MasksManager     :
IMasksManager
 {
 private         readonly         string
pathToFileWithMasks;

 public MasksManager(string pathToFile)
 {
 if (string.IsNullOrEmpty(pathToFile))
 {
 throw                                  new
ArgumentNullException(nameof(pathToFile))
;
 }

 this.pathToFileWithMasks = pathToFile;
 }

 public List<XmlMask> Mask { get; private
set; }

 public void LoadMasks()
 {
 XmlDocument     document     =     new
XmlDocument();
 document.Load(this.pathToFileWithMasks);

 var             xmlMasks             =
document.SelectNodes("//masks/mask");
 this.Mask = new List<XmlMask>();


 foreach (XmlNode xmlMask in xmlMasks)
 {
 List<string>        connectors        =
((XmlNode)xmlMask).SelectNodes("connector
").Cast<XmlNode>()
 .Select(x => x.InnerText)
 .ToList();

 XmlNode          maskSignature          =
xmlMask.SelectSingleNode("signature");
 var           testMaskNodes           =
xmlMask.SelectNodes("testMasks/mask");
 List<TaskMask>    testMasks    =    new
List<TaskMask>();
 foreach     (var     testMaskNode     in
testMaskNodes)
 {
 string            testType            =
((XmlNode)testMaskNode).Attributes["type"
].Value;
 ETestType convertedTestType;
 if   (Enum.TryParse<ETestType>(testType,
out convertedTestType))
 {
 string           testMaskId           =
((XmlNode)testMaskNode).Attributes["id"].
InnerText;
 XmlNode          testTaskMask          =
((XmlNode)testMaskNode).SelectSingleNode(
"signature");
 XmlNode         rightAnswerMask        =
((XmlNode)testMaskNode).SelectSingleNode(
"rightAnswer");
 XmlNode         wrongAnswerMask        =
((XmlNode)testMaskNode).SelectSingleNode(
"wrongAnswer");

 TaskMask taskMask;
 if (wrongAnswerMask == null)
 {
 taskMask   =   new   TaskMask(testMaskId,
testTaskMask,          convertedTestType,
rightAnswerMask);
 }
 else
 {
 taskMask   =   new   TaskMask(testMaskId,
testTaskMask,          convertedTestType,
rightAnswerMask, wrongAnswerMask);
 }

 testMasks.Add(taskMask);
 }
 }

 string               id               =
xmlMask.Attributes["id"].Value;
 this.Mask.Add(new
XmlMask(Guid.Parse(id),    maskSignature,
connectors, testMasks));
 }
 }

 public    void    AddMasks(List<XmlMask>
newMask)
 {
 throw new NotImplementedException();
 }

 public void EditMask(XmlMask newMask)
 {
 throw new NotImplementedException();
 }

 public void DeleteMask(string Id)
 {
 throw new NotImplementedException();
 }
 }
}
```

Лістинг AIS.TestGenerator\Masks\TaskMask.cs

```
using AIS.TestGenerator.TestTask;
using System;
using System.Xml;

namespace AIS.TestGenerator.Masks
{
 public class TaskMask
 {
 public TaskMask(string ID, XmlNode mask,
ETestType        testType,        XmlNode
rightAnswerMask, XmlNode wrongAnswerMask)
```

```
:             this(ID,             mask,
testType,rightAnswerMask)
 {
 if (wrongAnswerMask == null)
 throw                                new
ArgumentNullException(nameof(wrongAnswerM
ask));

 this.WrongAnswerMask = wrongAnswerMask;
 }

 public   TaskMask(string   ID,   XmlNode
questionMask, ETestType testType, XmlNode
rightAnswerMask)
 {
 if (string.IsNullOrEmpty(ID))
 throw                                new
ArgumentNullException(nameof(ID));

 if (rightAnswerMask == null)
 throw                                new
ArgumentNullException(nameof(rightAnswerM
ask));

 this.ID = ID;
 this.QuestionMask = questionMask;
 this.TestType = testType;
 this.RightAnswerMask = rightAnswerMask;
 }

 public string ID { get; private set; }

 public ETestType TestType { get; private
set; }

 public   XmlNode   QuestionMask   {   get;
private set; }

 public  XmlNode  RightAnswerMask  {  get;
private set; }

 public  XmlNode  WrongAnswerMask  {  get;
private set; }
 }
}
```

Лістинг AIS.TestGenerator\Masks\XmlMask.cs

```
using System;
using System.Collections.Generic;
using System.Xml;

namespace AIS.TestGenerator.Masks
{
 public class XmlMask
 {
 public  XmlMask(Guid  id,  XmlNode  mask,
List<string>  connectors,  List<TaskMask>
taskMasks)
 {
 if (id == null)
 throw                                new
ArgumentNullException(nameof(id));
 if (mask == null)
 throw                                new
ArgumentNullException(nameof(mask));
 if (connectors == null)
 throw                                new
ArgumentNullException(nameof(connectors))
;
 if (taskMasks == null)
 throw                                new
ArgumentNullException(nameof(taskMasks));

 this.ID = id;
 this.Mask = mask;
 this.Connectors = connectors;
 this.TaskMasks = taskMasks;
 }

 public Guid ID { get; private set; }

 public XmlNode Mask { get; private set;
}

 public List<string> Connectors { get;
private set; }

 public List<TaskMask> TaskMasks { get;
private set; }
 }
}
```

Лістинг AIS.TestGenerator\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly
is controlled through the following
//  set  of  attributes.  Change  these
attribute   values   to   modify   the
information
// associated with an assembly.
[assembly:
AssemblyTitle("AIS.TestGenerator")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly:
AssemblyProduct("AIS.TestGenerator")]
[assembly: AssemblyCopyright("Copyright ©
2018")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the
types in this assembly not visible
// to COM components.  If you need to
access a type in this assembly from
// COM, set the ComVisible attribute to
true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of
the typelib if this project is exposed to
COM
[assembly: Guid("295dc189-051a-4bcd-a87d-
4b2291c499e4")]

// Version information for an assembly
consists of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you
can  default  the  Build  and  Revision
Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly:
AssemblyFileVersion("1.0.0.0")]
```

Лістинг AIS.TestGenerator\Properties\Resources.Designer.cs

```
//------------------------------------------
------------------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
//  Changes  to  this  file  may  cause
incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------
------------------------------------------

namespace AIS.TestGenerator.Properties {
 using System;

 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or
Visual Studio.
 // To add or remove a member, edit your
.ResX file then rerun ResGen
 // with the /str option, or rebuild your
VS project.

[global::System.CodeDom.Compiler.Generate
dCodeAttribute("System.Resources.Tools.St
ronglyTypedResourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUs
erCodeAttribute()]

[global::System.Runtime.CompilerServices.
CompilerGeneratedAttribute()]
 internal class Resources {

 private                              static
global::System.Resources.ResourceManager
resourceMan;

 private                              static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.
SuppressMessageAttribute("Microsoft.Perfo
rmance",
"CA1811:AvoidUncalledPrivateCode")]
 internal Resources() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrow
sableAttribute(global::System.ComponentMo
del.EditorBrowsableState.Advanced)]
 internal                             static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if  (object.ReferenceEquals(resourceMan,
null)) {
 global::System.Resources.ResourceManager
temp                 =                 new
global::System.Resources.ResourceManager(
"AIS.TestGenerator.Properties.Resources",
typeof(Resources).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 ///  Overrides  the  current  thread's
CurrentUICulture property for all
 /// resource lookups using this strongly
typed resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrow
sableAttribute(global::System.ComponentMo
del.EditorBrowsableState.Advanced)]
 internal                             static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }
 }
```

```
}
```

Лістинг AIS.TestGenerator\Resources\MaskAttributeNameValue.Designer.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
//  Changes to this file may cause incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace AIS.TestGenerator.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual Studio.
 // To add or remove a member, edit your .ResX file then rerun ResGen
 // with the /str option, or rebuild your VS project.

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
 internal class MaskAttributeNameValue {

 private static global::System.Resources.ResourceManager resourceMan;

 private static global::System.Globalization.CultureInfo resourceCulture;


[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
 internal MaskAttributeNameValue() {
 }

 /// <summary>
 /// Returns the cached ResourceManager instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Resources.ResourceManager ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null)) {
 global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("AIS.TestGenerator.Resources.MaskAttributeNameValue", typeof(MaskAttributeNameValue).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's CurrentUICulture property for all
 /// resource lookups using this strongly typed resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Globalization.CultureInfo Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to case.
 /// </summary>
 internal static string Case {
 get {
 return ResourceManager.GetString("Case", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to upper.
 /// </summary>
 internal static string Upper {
 get {
 return ResourceManager.GetString("Upper", resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.TestGenerator\Resources\MaskTag.Designer.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
//  Changes to this file may cause incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace AIS.TestGenerator.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the StronglyTypedResourceBuilder
 // class via a tool like ResGen or Visual Studio.
 // To add or remove a member, edit your .ResX file then rerun ResGen
 // with the /str option, or rebuild your VS project.

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
 internal class MaskTag {

 private static global::System.Resources.ResourceManager resourceMan;

 private static global::System.Globalization.CultureInfo resourceCulture;


[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
 internal MaskTag() {
 }

 /// <summary>
 /// Returns the cached ResourceManager instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Resources.ResourceManager ResourceManager {
 get {
 if (object.ReferenceEquals(resourceMan, null)) {
 global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("AIS.TestGenerator.Resources.MaskTag", typeof(MaskTag).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 /// Overrides the current thread's CurrentUICulture property for all
 /// resource lookups using this strongly typed resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
 internal static global::System.Globalization.CultureInfo Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar to AnyWord.
 /// </summary>
 internal static string AnyWord {
 get {
 return ResourceManager.GetString("AnyWord", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to BeginSentence.
 /// </summary>
```

```
 internal static string BeginSentence {
 get {
 return
ResourceManager.GetString("BeginSentence"
, resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Caps.
 /// </summary>
 internal static string Caps {
 get {
 return ResourceManager.GetString("Caps",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to ].
 /// </summary>
 internal static string CloseTag {
 get {
 return
ResourceManager.GetString("CloseTag",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Connector.
 /// </summary>
 internal static string Connector {
 get {
 return
ResourceManager.GetString("Connector",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to EndSentence.
 /// </summary>
 internal static string EndSentence {
 get {
 return
ResourceManager.GetString("EndSentence",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Inflexion.
 /// </summary>
 internal static string Inflexion {
 get {
 return
ResourceManager.GetString("Inflexion",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to InputSpace.
 /// </summary>
 internal static string InputSpace {
 get {
 return
ResourceManager.GetString("InputSpace",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to _____.
 /// </summary>
 internal          static          string
InputSpaceReplacement {
 get {
 return
ResourceManager.GetString("InputSpaceRepl
acement", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to List.
 /// </summary>
 internal static string List {
 get {
 return ResourceManager.GetString("List",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to [.
 /// </summary>
 internal static string OpenTag {
 get {
 return
ResourceManager.GetString("OpenTag",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to RandomSplitedList.
 /// </summary>
 internal static string RandomSplitedList
{
 get {
 return
ResourceManager.GetString("RandomSplitedL
ist", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to RandomTermGroup.
 /// </summary>
 internal static string RandomTermGroup {
 get {
 return
ResourceManager.GetString("RandomTermGrou
p", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to RandomThesisGroup.
 /// </summary>
 internal static string RandomThesisGroup
{
 get {
 return
ResourceManager.GetString("RandomThesisGr
oup", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to SpecSymbol.
 /// </summary>
 internal static string SpecSymbol {
 get {
 return
ResourceManager.GetString("SpecSymbol",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to SplitedList.
 /// </summary>
 internal static string SplitedList {
 get {
 return
ResourceManager.GetString("SplitedList",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to TermGroup.
 /// </summary>
 internal static string TermGroup {
 get {
 return
ResourceManager.GetString("TermGroup",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to ThesisGroup.
 /// </summary>
 internal static string ThesisGroup {
 get {
 return
ResourceManager.GetString("ThesisGroup",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to TRUE.
 /// </summary>
 internal static string True {
 get {
 return ResourceManager.GetString("True",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to WithoutSpace.
 /// </summary>
 internal static string WithoutSpace {
 get {
 return
ResourceManager.GetString("WithoutSpace",
resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.TestGenerator\Resources\MaskTagTranslate.Designer.cs

```
//------------------------------------------
------------------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
//  Changes  to  this  file  may  cause
incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------
------------------------------------------

namespace AIS.TestGenerator.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or
Visual Studio.
 // To add or remove a member, edit your
.ResX file then rerun ResGen
```

```
 // with the /str option, or rebuild your
VS project.

[global::System.CodeDom.Compiler.Generate
dCodeAttribute("System.Resources.Tools.St
ronglyTypedResourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUs
erCodeAttribute()]

[global::System.Runtime.CompilerServices.
CompilerGeneratedAttribute()]
 internal class MaskTagTranslate {

 private                             static
global::System.Resources.ResourceManager
resourceMan;

 private                             static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.
SuppressMessageAttribute("Microsoft.Perfo
rmance",
"CA1811:AvoidUncalledPrivateCode")]
 internal MaskTagTranslate() {
 }

 /// <summary>
 ///  Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrow
sableAttribute(global::System.ComponentMo
del.EditorBrowsableState.Advanced)]
 internal                            static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if  (object.ReferenceEquals(resourceMan,
null)) {
 global::System.Resources.ResourceManager
temp                =                   new
global::System.Resources.ResourceManager(
"AIS.TestGenerator.Resources.MaskTagTrans
late",
typeof(MaskTagTranslate).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 ///   Overrides   the   current   thread's
CurrentUICulture property for all
 /// resource lookups using this strongly
typed resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrow
sableAttribute(global::System.ComponentMo
del.EditorBrowsableState.Advanced)]
 internal                            static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to ].
 /// </summary>
 internal static string CloseTag {
 get {
 return
ResourceManager.GetString("CloseTag",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Hi.
 /// </summary>
 internal static string False {
 get {
 return
ResourceManager.GetString("False",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to [.
 /// </summary>
 internal static string OpenTag {
 get {
 return
ResourceManager.GetString("OpenTag",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Так.
 /// </summary>
 internal static string True {
 get {
 return ResourceManager.GetString("True",
resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.TestGenerator\Resources\RegexTag.Designer.cs

```
//------------------------------------------
----------------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
//   Changes   to   this   file   may   cause
incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------
----------------------------------------

namespace AIS.TestGenerator.Resources {
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 //  class  via  a  tool  like  ResGen  or
Visual Studio.
 // To add or remove a member, edit your
.ResX file then rerun ResGen
 // with the /str option, or rebuild your
VS project.

[global::System.CodeDom.Compiler.Generate
dCodeAttribute("System.Resources.Tools.St
ronglyTypedResourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUs
erCodeAttribute()]

[global::System.Runtime.CompilerServices.
CompilerGeneratedAttribute()]
 internal class RegexTag {

 private                             static
global::System.Resources.ResourceManager
resourceMan;

 private                             static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.
SuppressMessageAttribute("Microsoft.Perfo
rmance",
"CA1811:AvoidUncalledPrivateCode")]
 internal RegexTag() {
 }

 /// <summary>
 ///  Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrow
sableAttribute(global::System.ComponentMo
del.EditorBrowsableState.Advanced)]
 internal                            static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if  (object.ReferenceEquals(resourceMan,
null)) {
 global::System.Resources.ResourceManager
temp                =                   new
global::System.Resources.ResourceManager(
"AIS.TestGenerator.Resources.RegexTag",
typeof(RegexTag).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 ///   Overrides   the   current   thread's
CurrentUICulture property for all
 /// resource lookups using this strongly
typed resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrow
sableAttribute(global::System.ComponentMo
del.EditorBrowsableState.Advanced)]
 internal                            static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to (?&lt;AnyWord&gt;[^.!?]*).
 /// </summary>
 internal static string AnyWord {
 get {
 return
ResourceManager.GetString("AnyWord",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to            (?&lt;BeginSentence&gt;[A-
Я|IЇ]+[^.?!\n]*).
 /// </summary>
 internal static string BeginSentence {
 get {
 return
ResourceManager.GetString("BeginSentence"
, resourceCulture);
 }
 }
```

```
 /// <summary>
 /// Looks up a localized string similar to (?&lt;EndSentence&gt;.*?\.).
 /// </summary>
 internal static string EndSentence {
 get {
 return ResourceManager.GetString("EndSentence", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to ((([\r\n]*.+;){1,}[\r\n]*[^\.]+\.)|((.+,){1,}(.*?\.))).
 /// </summary>
 internal static string List {
 get {
 return ResourceManager.GetString("List", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to [;|,|.].
 /// </summary>
 internal static string ListParse {
 get {
 return ResourceManager.GetString("ListParse", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to ((.+,){1,}(.*\.?)).
 /// </summary>
 internal static string OneLineList {
 get {
 return ResourceManager.GetString("OneLineList", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to |.
 /// </summary>
 internal static string SeparatorForOptions {
 get {
 return ResourceManager.GetString("SeparatorForOptions", resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar to (\s|\p{P}\s).
 /// </summary>
 internal static string WhiteSpaceOrPunctuation {
 get {
 return ResourceManager.GetString("WhiteSpaceOrPunctuation", resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.TestGenerator\TestBuilders\IMainTestBuilder.cs

```
using AIS.TermSearcher.Entities.Term;
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.TestGenerator.TestBuilders
{
 interface IMainTestBuilder
 {
 void BuildTask(ISentences sentences, Term keyTerm);

 void BuildTask(List<ISentences> sentences, Term keyTerm);

 List<TestTask.TestTask> FinalizeTaskBuild();
 }
}
```

Лістинг AIS.TestGenerator\TestBuilders\MainTestBuilder.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TestGenerator.Masks;
using AIS.TestGenerator.Resources;
using AIS.TestGenerator.TestBuilders.Builders;
using AIS.TestGenerator.TestTask;
using ASI.DocxReader.Entities.Sentences;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Xml;

namespace AIS.TestGenerator.TestBuilders
{
 class MainTestBuilder : IMainTestBuilder
 {
 private const int mistake = 15;
 private readonly IMasksManager masksManager;

 private Dictionary<ETestType, ITestBulder> testBuiders;

 public MainTestBuilder(IMasksManager masksManager)
 {
 if (masksManager == null) throw new ArgumentNullException(nameof(masksManager));

 this.masksManager = masksManager;
 testBuiders = new Dictionary<ETestType, ITestBulder>();
 }

 public void BuildTask(ISentences sentences, Term keyTerm)
 {
 foreach (XmlMask mask in this.masksManager.Mask)
 {
 if (TryParseSentences(mask, sentences, keyTerm, out Dictionary<string, string> tagValue))
 {
 foreach (TaskMask taskMask in mask.TaskMasks)
 {
 ITestBulder testBulder;
 if (!testBuiders.TryGetValue(taskMask.TestType, out testBulder))
 {
 testBulder = this.CreateTestBuilder(taskMask.TestType);
 testBuiders.Add(taskMask.TestType, testBulder);
 }

 testBulder.BuildTask(sentences, tagValue, keyTerm, taskMask);
 }
 }
 }
 }

 public void BuildTask(List<ISentences> sentences, Term keyTerm)
 {
 foreach (ISentences item in sentences)
 {
 BuildTask(item, keyTerm);
 }
 }

 public List<TestTask.TestTask> FinalizeTaskBuild()
 {
 List<TestTask.TestTask> tasks = new List<TestTask.TestTask>();
 foreach (var testBuilder in this.testBuiders.Values)
 {
 List<TestTask.TestTask> finalizedTestTask = testBuilder.FinalizeTaskBuild();
 tasks.AddRange(finalizedTestTask);
 }

 return tasks;
 }

 private bool TryParseSentences(XmlMask mask, ISentences sentences, Term term, out Dictionary<string, string> outTagValues)
 {
 int previousTagEndPosition = 0;
 outTagValues = null;
 Dictionary<string, string> tagValues = new Dictionary<string, string>();
 bool isBeginTag = false;
 foreach(XmlNode tagNode in mask.Mask)
 {
 string currentTagValue = string.Empty;
 bool isTagParsed = false;
 if (previousTagEndPosition >= sentences.Text.Length)
 return false;

 if (MaskTag.BeginSentence == tagNode.Name)
 {
 isBeginTag = true;
 continue;
 }
 else if (MaskTag.Connector == tagNode.Name)
 {
 isTagParsed = TryParseConnector(mask.Connectors, sentences.Text, ref previousTagEndPosition, out currentTagValue);
 }
 else if (MaskTag.TermGroup == tagNode.Name)
 {
 isTagParsed = TryParseTermGroup(term, sentences.Text, ref previousTagEndPosition, out currentTagValue);
 }
 else if (MaskTag.EndSentence == tagNode.Name)
 {
 isTagParsed = true;
 currentTagValue = sentences.Text.Substring(previousTagEndPosition);
 }
 else if (MaskTag.List == tagNode.Name)
 {
 isTagParsed = TryParseList(sentences.Text, ref previousTagEndPosition, out currentTagValue);
 }
 else if (MaskTag.SpecSymbol == tagNode.Name && !string.IsNullOrEmpty(tagNode.InnerText))
```

```
 {
 isTagParsed                                = TryParseSpecSymbol(tagNode.InnerText, sentences.Text,                          ref previousTagEndPosition,                out currentTagValue);
 }

 if (!isTagParsed)
 return false;

 if (isBeginTag)
 {
 isBeginTag = false;
 tagValues.Add(MaskTag.BeginSentence, sentences.Text.Substring(0, previousTagEndPosition                  - currentTagValue.Length));
 }

 tagValues.Add(tagNode.Name, currentTagValue);
 }

 outTagValues = tagValues;
 return true;
 }

 private                        ITestBulder CreateTestBuilder(ETestType testType)
 {
 switch (testType)
 {
 case ETestType.YesNo:
 return new YesNoTestBuilder();
 case ETestType.SingleChoice:
 return new SingleChoiceTestBuilder();
 case ETestType.InputAnswer:
 return new InputAnswerTestBuilder();
 case ETestType.MultipleChoice:
 return new MultipleChoiceBuilder();
 default:
 throw new NotImplementedException();
 }
 }

 private                                bool TryParseConnector(List<string> connectors,  string  sentences,  ref  int endPosition, out string value)
 {
 string        joinedConnectors        = string.Join("|",  connectors.Select(x  => $"{x}\\b"));
 int         maxConnectorLength         = connectors.Max(x => x.Length);
 int           lengthSentences          = (maxConnectorLength    +    mistake)    % (sentences.Length - endPosition);
 Match             connector            = Regex.Match(sentences.Substring(endPositi on, lengthSentences), joinedConnectors);
 if (!connector.Success)
 {
 value = string.Empty;
 return false;
 }

 value = connector.Value;
 endPosition      =      endPosition     + connector.Index + connector.Value.Length;
 return true;
 }

 private    bool    TryParseTermGroup(Term term,    string    sentences,    ref    int endPosition, out string value)
 {
 string joinedKeyTerms = string.Join("|", term.AllExistedForm.Select(x           => $"\\b{x.Text}\\b"));
 int lengthSentences = endPosition == 0 ? sentences.Length  :  (term.Text.Length  + mistake)     %     (sentences.Length     - endPosition);
 Match              keyTerm             = Regex.Match(sentences.Substring(endPositi on,   lengthSentences),   joinedKeyTerms, RegexOptions.IgnoreCase);
 if (!keyTerm.Success)
 {
 value = string.Empty;
 return false;
 }

 value = keyTerm.Value;
 endPosition     =     keyTerm.Index     + keyTerm.Value.Length;
 return true;
 }

 private     bool     TryParseList(string sentences,  ref  int  endPosition,  out string value)
 {
 Match               list               = Regex.Match(sentences.Substring(endPositi on), RegexTag.OneLineList);
 if (!list.Success)
 {
 value = string.Empty;
 return false;
 }

 value = list.Value;
 endPosition      =      list.Index      + list.Value.Length;
 return true;
 }

 private  bool  TryParseSpecSymbol(string specSymbol,  string  sentences,  ref  int endPosition, out string value)
 {
 int   lengthSentences   =   mistake   % (sentences.Length - endPosition);
 int           specSymbolIndex          = sentences.Substring(endPosition, lengthSentences).IndexOf(specSymbol);
 if (specSymbolIndex == -1)
 {
 value = string.Empty;
 return false;
 }

 value = specSymbol;
 endPosition    +=    specSymbolIndex    + specSymbol.Length;
 return true;
 }
 }
}
```

Лістинг AIS.TestGenerator\TestBuilders\Builders\BaseTaskBuilder.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TestGenerator.Masks;
using AIS.TestGenerator.Resources;
using AIS.TestGenerator.TestTask;
using ASI.DocxReader.Entities.Sentences;
using AIS.Helper.Extention.String;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Xml;

namespace AIS.TestGenerator.TestBuilders.Builders
{
 abstract    class    BaseTaskBuilder    : ITestBulder
 {
 protected       readonly       ETestType currentTestType;

 protected  Dictionary<string,  TaskMask> usedTaskMasks  =  new  Dictionary<string, TaskMask>();
 protected        List<TestTaskWorkpiece> generatedTaskWorkpieces       =       new List<TestTaskWorkpiece>();
 protected Random random = new Random();

 public         BaseTaskBuilder(ETestType testType)
 {
 this.currentTestType = testType;
 }

 public     void     BuildTask(ISentences sentences,   Dictionary<string,   string> sentencesParts,  Term  keyTerm,  TaskMask taskMask)
 {
 if (!this.usedTaskMasks.TryGetValue(taskMask.ID, out TaskMask result))
 {
 this.usedTaskMasks.Add(taskMask.ID, taskMask);
 }

 TestTaskWorkpiece  taskWorkpiece  =  new TestTaskWorkpiece
 {
 KeyTerm = keyTerm,
 GeneratedByTaskMask = taskMask,
 Sentences = sentences,
 SentencesParts = sentencesParts
 };

 CreateQuestionParts(taskWorkpiece);

 this.generatedTaskWorkpieces.Add(taskWorkpiece);
 }

 public           List<TestTask.TestTask> FinalizeTaskBuild()
 {
 List<TestTask.TestTask>  result  =  new List<TestTask.TestTask>();
 foreach    (var    taskWorkpieces    in this.generatedTaskWorkpieces)
 {
 int       taskIndexForBigerTerm       = result.FindIndex(x                       => x.Keyterm.Infinitive.Contains(taskWorkpieces.KeyTerm.Infinitive)
 && taskWorkpieces.KeyTerm.Infinitive != x.Keyterm.Infinitive
 && taskWorkpieces.GeneratedByTaskMask.QuestionMask.InnerXml                        == x.GeneratedByTaskMask.QuestionMask.InnerXml
 &&             x.Sentences             == taskWorkpieces.Sentences);

 int      taskIndexForSmallerTerm      = result.FindIndex(x                       => taskWorkpieces.KeyTerm.Infinitive.Contains(x.Keyterm.Infinitive)
 && taskWorkpieces.KeyTerm.Infinitive != x.Keyterm.Infinitive
 && taskWorkpieces.GeneratedByTaskMask.QuestionMask.InnerXml                        == x.GeneratedByTaskMask.QuestionMask.InnerXml
 &&             x.Sentences             == taskWorkpieces.Sentences);
 if  (taskIndexForSmallerTerm  ==  -1  && taskIndexForBigerTerm == -1)
 {
 result.Add(CreateTask(taskWorkpieces));
 }
 else if(taskIndexForSmallerTerm != -1)
 {
 result[taskIndexForSmallerTerm]        = CreateTask(taskWorkpieces);
 }
 }
```

```
 return result;
 }

 private                TestTask.TestTask
CreateTask(TestTaskWorkpiece
testTaskWorkpiece)
 {
 string            question            =
FinalizeQuestion(testTaskWorkpiece);
 List<AnswerOption>    answerOptions    =
CreateAnswers(testTaskWorkpiece);
 return new TestTask.TestTask(question,
 answerOptions,
 currentTestType,
 testTaskWorkpiece.KeyTerm,
 testTaskWorkpiece.Sentences,
 testTaskWorkpiece.GeneratedByTaskMask);
 }

 protected         virtual         string
GetMaskTagValue(XmlNode              tag,
TestTaskWorkpiece taskWorkpiece)
 {
 string tagValue;
 if  (MaskTag.TermGroup  ==  tag.Name  ||
MaskTag.Connector == tag.Name)
 {
 tagValue                                =
taskWorkpiece.SentencesParts[tag.Name];
 }
 else    if    (MaskTag.ThesisGroup    ==
tag.Name)
 {
 List<string>      dictionaryKeys      =
taskWorkpiece.SentencesParts.Keys.ToList(
);
 int              termIndex              =
dictionaryKeys.IndexOf(MaskTag.TermGroup)
;
 int           connectorIndex           =
dictionaryKeys.IndexOf(MaskTag.Connector)
;

 if (connectorIndex > termIndex)
 {
 tagValue                                =
taskWorkpiece.SentencesParts[MaskTag.EndS
entence];
 }
 else
 {
 tagValue                                =
taskWorkpiece.SentencesParts[MaskTag.Begi
nSentence];
 }
 }
 else                                  if
(taskWorkpiece.SentencesParts.TryGetValue
(tag.Name, out string value))
 {
 tagValue = value;
 }
 else
 {
 tagValue = tag.Name;
 }

 return tagValue.Trim(' ');
 }

 protected         virtual         string
GetMaskRandomTagValue(XmlNode        tag,
TestTaskWorkpiece taskWorkpiece)
 {
 string tagValue = tag.Name;
 if (MaskTag.RandomTermGroup == tag.Name)
 {
 string xpath = "./" + MaskTag.TermGroup;
 var          filteredWorkpiece          =
this.generatedTaskWorkpieces.Where(x   =>
x.GeneratedByTaskMask.QuestionMask.Select
SingleNode(xpath) != null
 &&           x.KeyTerm.Text           !=
taskWorkpiece.KeyTerm.Text).ToList();

 if (filteredWorkpiece.Count > 0)
 {
 int            randomIndex            =
random.Next(filteredWorkpiece.Count);
 tagValue                                =
filteredWorkpiece[randomIndex].QuestionPa
rts[MaskTag.TermGroup];
 }
 }
 else  if  (MaskTag.RandomThesisGroup  ==
tag.Name)
 {
 string      xpath      =      "./"      +
MaskTag.ThesisGroup;
 var          filteredWorkpiece          =
this.generatedTaskWorkpieces.Where(x   =>
x.GeneratedByTaskMask.QuestionMask.Select
SingleNode(xpath) != null
 &&           x.Sentences.Text           !=
taskWorkpiece.Sentences.Text).ToList();

 if (filteredWorkpiece.Count > 0)
 {
 int            randomIndex            =
random.Next(filteredWorkpiece.Count);
 tagValue                                =
filteredWorkpiece[randomIndex].QuestionPa
rts[MaskTag.ThesisGroup];
 }
 }

 return tagValue;
 }

 protected   abstract   List<AnswerOption>
CreateAnswers(TestTaskWorkpiece
taskWorkpiece);

 protected                           string
FinalizeQuestion(TestTaskWorkpiece
taskWorkpiece)
 {
 List<string>       tags       =       new
List<string>(taskWorkpiece.QuestionParts.
Keys);
 foreach       (XmlNode       tag       in
taskWorkpiece.GeneratedByTaskMask.Questio
nMask.ChildNodes)
 {
 string             tagValue             =
taskWorkpiece.QuestionParts[tag.Name];
 if               (string.Equals(tag.Name,
taskWorkpiece.QuestionParts[tag.Name]))
 {
 if (tag.Name.Contains("Random"))
 {
 tagValue   =   GetMaskRandomTagValue(tag,
taskWorkpiece);
 }
 else
 {
 tagValue     =     GetMaskTagValue(tag,
taskWorkpiece);
 }
 }

 taskWorkpiece.QuestionParts[tag.Name]  =
ApplyStyleAttributes(tag, tagValue);
 }

 return          string.Join("          ",
taskWorkpiece.QuestionParts.Values);
 }

 protected                           string
ApplyStyleAttributes(XmlNode  tag,  string
tagValue)
 {
 string formatedTagValue = tagValue;
 foreach   (XmlAttribute   attribute   in
tag.Attributes)
 {
 if           (attribute.Name           ==
MaskAttributeNameValue.Case)
 {
 if           (attribute.Value           ==
MaskAttributeNameValue.Upper)
 {
 formatedTagValue                        =
tagValue.FirstCharToUpper();
 }
 }
 }

 return formatedTagValue;
 }

 private                               void
CreateQuestionParts(TestTaskWorkpiece
taskWorkpiece)
 {
 taskWorkpiece.QuestionParts     =     new
Dictionary<string, string>();
 foreach       (XmlNode       tag       in
taskWorkpiece.GeneratedByTaskMask.Questio
nMask.ChildNodes)
 {
 string  tagValue  =  GetMaskTagValue(tag,
taskWorkpiece);

taskWorkpiece.QuestionParts.Add(tag.Name,
tagValue);
 }
 }
 }
}
```

Лістинг AIS.TestGenerator\TestBuilders\Builders\InputAnswerTestBuilder.cs

```
using AIS.TestGenerator.Resources;
using AIS.TestGenerator.TestTask;
using System.Collections.Generic;
using System.Xml;

namespace
AIS.TestGenerator.TestBuilders.Builders
{
 class      InputAnswerTestBuilder      :
BaseTaskBuilder
 {
 public     InputAnswerTestBuilder()     :
base(ETestType.InputAnswer)
 {

 }

 protected         override         string
GetMaskTagValue(XmlNode              tag,
TestTaskWorkpiece taskWorkpiece)
 {
 string tagValue;
 if    (string.Equals(MaskTag.InputSpace,
tag.Name))
 {
 tagValue                                =
MaskTag.InputSpaceReplacement;
 }
 else
 {
 tagValue   =   base.GetMaskTagValue(tag,
taskWorkpiece);
 }

 return tagValue;
 }

 protected   override   List<AnswerOption>
CreateAnswers(TestTaskWorkpiece
taskWorkpiece)
 {
 var         rightAnswerMaskTags         =
taskWorkpiece.GeneratedByTaskMask.RightAn
swerMask.ChildNodes;
 string[]       tagValues       =       new
string[rightAnswerMaskTags.Count];
 for     (int     i     =     0;     i     <
rightAnswerMaskTags.Count; i++)
 {
 XmlNode              tag              =
rightAnswerMaskTags.Item(i);
```

```
 string        generatedTagValue        =
GetMaskTagValue(tag, taskWorkpiece);
 generatedTagValue                      =
ApplyStyleAttributes(tag,
generatedTagValue);

 tagValues[i] = generatedTagValue;
 }

 string       generatedRightAnswer      =
string.Join(" ", tagValues);
 return new List<AnswerOption>()
 {
 new   AnswerOption(generatedRightAnswer,
true)
 };
 }
 }
}
```

Лістинг AIS.TestGenerator\TestBuilders\Builders\ITestBulder.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TestGenerator.Masks;
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace
AIS.TestGenerator.TestBuilders.Builders
{
 interface ITestBulder
 {
 void   BuildTask(ISentences   sentences,
Dictionary<string,                string>
sentencesParts,  Term  keyTerm,  TaskMask
taskMask);

 List<TestTask.TestTask>
FinalizeTaskBuild();
 }
}
```

Лістинг AIS.TestGenerator\TestBuilders\Builders\MultipleChoiceBuilder.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Xml;
using AIS.TestGenerator.Resources;
using AIS.TestGenerator.TestTask;

namespace
AIS.TestGenerator.TestBuilders.Builders
{
 class      MultipleChoiceBuilder      :
BaseTaskBuilder
 {
 public     MultipleChoiceBuilder()     :
base(ETestType.MultipleChoice)
 {

 }

 protected   override   List<AnswerOption>
CreateAnswers(TestTaskWorkpiece
taskWorkpiece)
 {
 List<AnswerOption>    answers    =    new
List<AnswerOption>();

answers.AddRange(CreateRightAnswers(taskW
orkpiece));

answers.AddRange(CreateWrongAnswers(taskW
orkpiece));
 return answers;
 }

 private               List<AnswerOption>
CreateRightAnswers(TestTaskWorkpiece
taskWorkpiece)
 {
 List<AnswerOption>    answers    =    new
List<AnswerOption>();
 int             answerMask             =
taskWorkpiece.GeneratedByTaskMask.RightAn
swerMask.ChildNodes.Count;
 if      (answerMask      >      1      ||
taskWorkpiece.GeneratedByTaskMask.RightAn
swerMask.ChildNodes[0].Name            !=
MaskTag.SplitedList)
 {
 throw        new        Exception($"Mask
'{taskWorkpiece.GeneratedByTaskMask.Right
AnswerMask.InnerXml}'       is       not
supported.");
 }

 string              list              =
taskWorkpiece.SentencesParts[MaskTag.List
];
 string[] listItems = Regex.Split(list,
RegexTag.ListParse);
 return    listItems.Select(x    =>    new
AnswerOption(x, true)).ToList();
 }

 private               List<AnswerOption>
CreateWrongAnswers(TestTaskWorkpiece
taskWorkpiece)
 {
 List<AnswerOption>    answers    =    new
List<AnswerOption>();
 int             answerMask             =
taskWorkpiece.GeneratedByTaskMask.WrongAn
swerMask.ChildNodes.Count;
 if      (answerMask      >      1      ||
taskWorkpiece.GeneratedByTaskMask.RightAn
swerMask.ChildNodes[0].Name            !=
MaskTag.SplitedList)
 {
 throw        new        Exception($"Mask
'{taskWorkpiece.GeneratedByTaskMask.Right
AnswerMask.InnerXml}'       is       not
supported.");
 }

 var         filteredWorkpiece         =
this.generatedTaskWorkpieces.Where(x   =>
x.SentencesParts.ContainsKey(MaskTag.List
)
 &&          x.Sentences.Text          !=
taskWorkpiece.Sentences.Text).ToList();

 if (filteredWorkpiece.Count > 0)
 {
 int             randomIndex            =
random.Next(filteredWorkpiece.Count);
 string           randomList           =
filteredWorkpiece[randomIndex].SentencesP
arts[MaskTag.List];
 string[]        randomListItems        =
Regex.Split(randomList,
RegexTag.ListParse);

 int             skipedIndex            =
random.Next(randomListItems.Length);
 answers                                =
randomListItems.Skip(skipedIndex)
 .Take(randomListItems.Length           -
skipedIndex)
 .Select(x    =>    new    AnswerOption(x,
false)).ToList();
 }

 return answers;
 }
 }
}
```

Лістинг AIS.TestGenerator\TestBuilders\Builders\SingleChoiceTestBuilder.cs

```
using AIS.TestGenerator.Masks;
using AIS.TestGenerator.TestTask;
using System.Collections.Generic;
using System.Linq;
using System.Xml;

namespace
AIS.TestGenerator.TestBuilders.Builders
{
 class      SingleChoiceTestBuilder      :
BaseTaskBuilder
 {
 public    SingleChoiceTestBuilder()    :
base(ETestType.SingleChoice)
 {
 }

 protected   override   List<AnswerOption>
CreateAnswers(TestTaskWorkpiece
taskWorkpiece)
 {
 List<AnswerOption>    answers    =    new
List<AnswerOption>();
 string generatedAnswer;
 int sameAnswerCount = 0;
 while    (answers.Count    <    3    &&
sameAnswerCount < 3)
 {
 generatedAnswer                        =
GenerateAnswer(taskWorkpiece.GeneratedByT
askMask.WrongAnswerMask, taskWorkpiece);
 if (answers.FirstOrDefault(x => x.Answer
== generatedAnswer) != null)
 {
 sameAnswerCount++;
 continue;
 }

 AnswerOption    wrongAnswer    =    new
AnswerOption(generatedAnswer, false);
 answers.Add(wrongAnswer);
 sameAnswerCount = 0;
 }

 generatedAnswer                        =
GenerateAnswer(taskWorkpiece.GeneratedByT
askMask.RightAnswerMask, taskWorkpiece);
 AnswerOption    rightAnswer    =    new
AnswerOption(generatedAnswer, true);

 answers.Add(rightAnswer);
 return answers;
 }

 private  string  GenerateAnswer(XmlNode
mask,                   TestTaskWorkpiece
currentTaskWorkpiece)
 {
 string[]      tagValues      =      new
string[mask.ChildNodes.Count];
 for    (int    i    =    0;    i    <
mask.ChildNodes.Count; i++)
 {
 XmlNode tag = mask.ChildNodes.Item(i);
 string generatedValue;
 if (tag.Name.Contains("Random"))
 {
 generatedValue                         =
GetMaskRandomTagValue(tag,
currentTaskWorkpiece);
 }
 else
 {
 generatedValue  =  GetMaskTagValue(tag,
currentTaskWorkpiece);
 }

 tagValues[i] = ApplyStyleAttributes(tag,
generatedValue);
 }

 return string.Join(" ", tagValues);
 }
 }
}
```

Лістинг AIS.TestGenerator\TestBuilders\Builders\YesNoTestBuilder.cs

```
using AIS.TestGenerator.Resources;
using AIS.TestGenerator.TestTask;
using System.Collections.Generic;

namespace AIS.TestGenerator.TestBuilders.Builders
{
 class YesNoTestBuilder : BaseTaskBuilder
 {
 public YesNoTestBuilder() : base(ETestType.YesNo)
 {

 }

 protected override List<AnswerOption> CreateAnswers(TestTaskWorkpiece taskWorkpiece)
 {
 List<AnswerOption> answers = new List<AnswerOption>();
 AnswerOption rightAnswer = new AnswerOption()
 {
 Answer = InterpretAnswer(taskWorkpiece.GeneratedByTaskMask.RightAnswerMask.ChildNodes[0].Name),
 IsRightAnswer = true
 };

 AnswerOption wrongAnswer = new AnswerOption()
 {
 Answer = InterpretAnswer(taskWorkpiece.GeneratedByTaskMask.WrongAnswerMask.ChildNodes[0].Name),
 IsRightAnswer = false
 };

 answers.Add(rightAnswer);
 answers.Add(wrongAnswer);
 return answers;
 }

 private string InterpretAnswer(string answer)
 {
 return string.Equals(answer, MaskTag.True) ? MaskTagTranslate.True : MaskTagTranslate.False;
 }
 }
}
```

Лістинг AIS.TestGenerator\TestTask\AnswerOption.cs

```
using System;
namespace AIS.TestGenerator.TestTask
{
 public class AnswerOption : ICloneable
 {
 public AnswerOption()
 {

 }

 public AnswerOption(string answer, bool isRightAnswer)
 {
 this.Answer = answer;
 this.IsRightAnswer = isRightAnswer;
 }

 public string Answer { get; set; }

 public bool IsRightAnswer { get; set; }

 public object Clone()
 {
 return new AnswerOption()
 {
 Answer = this.Answer,
 IsRightAnswer = this.IsRightAnswer
 };
 }
 }
}
```

Лістинг AIS.TestGenerator\TestTask\ETestType.cs

```
namespace AIS.TestGenerator.TestTask
{
 public enum ETestType
 {
 YesNo,
 SingleChoice,
 MultipleChoice,
 InputAnswer
 }
}
```

Лістинг AIS.TestGenerator\TestTask\TestTask.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TestGenerator.Masks;
using ASI.DocxReader.Entities.Sentences;
using System;
using System.Collections.Generic;
using System.Linq;

namespace AIS.TestGenerator.TestTask
{
 public class TestTask : ICloneable
 {
 public TestTask(string task, List<AnswerOption> answerOptions, ETestType testType, Term keyterm, ISentences sentences, TaskMask generatedByTaskMask)
 {
 this.Task = task;
 this.AnswerOptions = answerOptions;
 this.TestType = testType;
 this.Keyterm = keyterm;
 this.Sentences = sentences;
 this.GeneratedByTaskMask = generatedByTaskMask;
 }

 public string Task { get; set; }

 public List<AnswerOption> AnswerOptions { get; set; }

 public ETestType TestType { get; set; }

 public Term Keyterm { get; set; }

 public ISentences Sentences { get; set; }

 public TaskMask GeneratedByTaskMask { get; set; }

 public object Clone()
 {
 List<AnswerOption> clonedAnswerOptions = AnswerOptions.Select(x => (AnswerOption)x.Clone()).ToList();
 return new TestTask(this.Task, clonedAnswerOptions, this.TestType, this.Keyterm, this.Sentences, this.GeneratedByTaskMask);
 }
 }
}
```

Лістинг AIS.TestGenerator\TestTask\TestTaskWorkpiece.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TestGenerator.Masks;
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.TestGenerator.TestTask
{
 class TestTaskWorkpiece
 {
 public Dictionary<string, string> QuestionParts { get; set; }

 public TaskMask GeneratedByTaskMask { get; set; }

 public Term KeyTerm{ get; set; }

 public Dictionary<string, string> RightAnswerParts { get; set; }

 public Dictionary<string, string> WrongAnswerParts { get; set; }

 public ISentences Sentences { get; set; }

 public Dictionary<string, string> SentencesParts { get; set; }
 }
}
```

Лістинг AIS.TestGenerator\TestTask\Comparers\AnswerOptionComparer.cs

```
using System.Collections.Generic;

namespace AIS.TestGenerator.TestTask.Comparers
{
 class AnswerOptionComparer : IEqualityComparer<AnswerOption>
 {
 public bool Equals(AnswerOption firstAnswer, AnswerOption secondAnswer)
 {
 return string.Equals(firstAnswer.Answer, secondAnswer.Answer) && firstAnswer.IsRightAnswer == secondAnswer.IsRightAnswer;
 }

 public int GetHashCode(AnswerOption answer)
 {
 int hashCode = answer.Answer == null ? 0 : answer.Answer.GetHashCode();
 return hashCode ^ answer.IsRightAnswer.GetHashCode();
 }
 }
}
```

Лістинг AIS.TestGenerator\TestTask\Comparers\TestTaskComparer.cs

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

namespace AIS.TestGenerator.TestTask.Comparers
{
 public class TestTaskComparer : IEqualityComparer<TestTask>
 {
 public bool Equals(TestTask firstTask, TestTask secondTask)
 {
 bool isTaskTheSame = firstTask.Task == secondTask.Task;
 bool isAnswersSame = IsAnswersSame(firstTask.AnswerOptions, secondTask.AnswerOptions);
 bool isTestTypeSame = firstTask.TestType == secondTask.TestType;
```

```
 bool       isGeneratedBySameMask       =
firstTask.GeneratedByTaskMask.ID       ==
secondTask.GeneratedByTaskMask.ID;

 return isTaskTheSame && isAnswersSame &&
isTestTypeSame && isGeneratedBySameMask;
 }

 public int GetHashCode(TestTask obj)
 {
 return obj.TestType.GetHashCode()
 ^ obj.Task.GetHashCode()
 ^
obj.GeneratedByTaskMask.ID.GetHashCode();
 }

 private                                bool
IsAnswersSame(List<AnswerOption>
firstAnswers,          List<AnswerOption>
secondAnswers)
 {
 AnswerOptionComparer
answerOptionComparer        =        new
AnswerOptionComparer();
 return
firstAnswers.Except(secondAnswers,
answerOptionComparer).Count() != 0
 &&    secondAnswers.Except(firstAnswers,
answerOptionComparer).Count() != 0;
 }
 }
}
```

Проєкт AIS.TestGenerator.UI

Лістинг AIS.TestGenerator.UI\AnswerOptionComparer.cs

```
using AIS.TestGenerator.TestTask;
using System.Collections.Generic;

namespace AIS.TestGenerator.UI
{
 internal  class  AnswerOptionComparer  :
IEqualityComparer<AnswerOption>
 {
 public   bool   Equals(AnswerOption   x,
AnswerOption y)
 {
 return string.Equals(x.Answer, y.Answer)
&& x.IsRightAnswer == y.IsRightAnswer;
 }

 public int GetHashCode(AnswerOption obj)
 {
 int hashCode = obj.Answer == null ? 0 :
obj.Answer.GetHashCode();
 return            hashCode            +
obj.IsRightAnswer.GetHashCode();
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\App.xaml

```
<Application
x:Class="AIS.TestGenerator.UI.App"

xmlns="http://schemas.microsoft.com/winfx
/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/win
fx/2006/xaml"
 xmlns:local="clr-
namespace:AIS.TestGenerator.UI"
 StartupUri="MainWindow.xaml">
 <Application.Resources>

 </Application.Resources>
</Application>
```

Лістинг AIS.TestGenerator.UI\App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Navigation;

namespace AIS.TestGenerator.UI
{
 /// <summary>
 /// Interaction logic for App.xaml
 /// </summary>
 public partial class App : Application
 {
 }
}
```

Лістинг AIS.TestGenerator.UI\DialogFactory.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Entities.Word;
using AIS.TestGenerator.UI.ViewModels;
using AIS.TestGenerator.UI.Views;
using Microsoft.Win32;
using System.Collections.Generic;
using System.Windows;

namespace AIS.TestGenerator.UI
{
 class DialogFactory : IDialogFactory
 {
 public                                 Window
CreateTestGenerationResult(IDocument
document, List<Word> UnKnownWords)
 {
 GenerationResultViewModel  viewModel  =
new   GenerationResultViewModel(document,
UnKnownWords, this);
 TestGenerationResult    view    =    new
TestGenerationResult();
 view.DataContext = viewModel;

 return view;
 }

 public                             FileDialog
GetSelectFolderDialog()
 {
 SaveFileDialog   openFileDialog   =   new
SaveFileDialog();
 return openFileDialog;
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\IDialogFactory.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Entities.Word;
using Microsoft.Win32;
using System.Collections.Generic;
using System.Windows;

namespace AIS.TestGenerator.UI
{
 public interface IDialogFactory
 {
 Window
CreateTestGenerationResult(IDocument
document, List<Word> UnKnownWords);

 FileDialog GetSelectFolderDialog();
 }
}
```

Лістинг AIS.TestGenerator.UI\MainWindow.xaml

```
<Window          xmlns:UserControls="clr-
namespace:AIS.TestGenerator.UI.Views.User
Controls"
x:Class="AIS.TestGenerator.UI.MainWindow"

xmlns="http://schemas.microsoft.com/winfx
/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/win
fx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/exp
ression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.o
rg/markup-compatibility/2006"
 xmlns:viewModel="clr-
namespace:AIS.TestGenerator.UI.ViewModels
"
 xmlns:userControls="clr-
namespace:AIS.TestGenerator.UI.Views.User
Controls"

xmlns:gif="http://wpfanimatedgif.codeplex
.com"
 xmlns:local="clr-
namespace:AIS.TestGenerator.UI"
 mc:Ignorable="d"
 Title="MainWindow"          Height="180"
Width="700"               MinHeight="180"
MinWidth="300">
 <Window.Resources>
 <ResourceDictionary>
 <DataTemplate          DataType="{x:Type
viewModel:StyleCoefficientEditorViewModel
}">
 <userControls:StyleCoefficientEditor/>
 </DataTemplate>
 <ResourceDictionary.MergedDictionaries>
 <ResourceDictionary
Source="pack://application:,,,/Views/Styl
e/Converters.xaml" />
 </ResourceDictionary.MergedDictionaries>
 </ResourceDictionary>
 </Window.Resources>
 <Grid>
 <Grid Margin="10">
 <Grid.RowDefinitions>
 <RowDefinition Height="23"/>
 <RowDefinition Height="70"/>
 <RowDefinition Height="*"/>
 <RowDefinition Height="23"/>
 </Grid.RowDefinitions>
 <Grid Grid.Row="0">
 <Grid.ColumnDefinitions>
 <ColumnDefinition Width="*"/>
 <ColumnDefinition Width="5"/>
 <ColumnDefinition Width="80"/>
 </Grid.ColumnDefinitions>
 <TextBox   Grid.Row="0"   Grid.Column="0"
x:Name="filePathTextBox"
IsReadOnly="True"          Text="{Binding
SelectedFilePath}"/>
 <Button   Grid.Row="0"   Grid.Column="2"
x:Name="browseButton"   Content="Вибрати"
Command="{Binding BrowseFiles}"/>
 </Grid>
 <GroupBox                   Grid.Row="1"
Header="Коефіцієнти">
 <ContentControl        Content="{Binding
StyleCoefficientViewModel}"/>
 </GroupBox>
 <Button      Grid.Row="3"      Width="80"
x:Name="startButton"
HorizontalAlignment="Right"
Content="Розпочати"     Command="{Binding
StartTestGenerator}" Grid.Column="1"/>
 </Grid>
 <Border              BorderBrush="Black"
BorderThickness="1"
Background="#80000000"
Visibility="{Binding              IsBusy,
Converter={StaticResource
booleanToVisibilityConverter}}"
Grid.RowSpan="4">
 <Grid>
 <Image
gif:ImageBehavior.RepeatBehavior="Forever
"
gif:ImageBehavior.AnimatedSource="Resourc
es/loading.gif"/>
 </Grid>
 </Border>
```

```
 </Grid>
</Window>
```

Лістинг
AIS.TestGenerator.UI\MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace AIS.TestGenerator.UI
{
 /// <summary>
 ///      Interaction      logic      for
MainWindow.xaml
 /// </summary>
 public partial class MainWindow : Window
 {
 public MainWindow()
 {
 InitializeComponent();
 DialogFactory   dialogFactory   =   new
DialogFactory();
 this.DataContext          =          new
ViewModels.MainWindowViewModel(dialogFact
ory);
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\RelayCommand.cs

```
using System;
using System.Windows.Input;

namespace AIS.TestGenerator.UI
{
 public class RelayCommand<T> : ICommand
 {
 private  readonly  Action<T>  execute  =
null;
 private readonly Predicate<T> canExecute
= null;

 public RelayCommand(Action<T> execute) :
this(execute, null)
 {
 }

 public  RelayCommand(Action<T>  execute,
Predicate<T> canExecute)
 {
 if (execute == null)
 {
 throw                                  new
ArgumentNullException(nameof(execute));
 }

 this.execute = execute;
 this.canExecute = canExecute;
 }

 public bool CanExecute(object parameter)
 {
 return  canExecute  ==  null  ?  true  :
canExecute((T)parameter);
 }

 public        event        EventHandler
CanExecuteChanged
 {
 add { CommandManager.RequerySuggested +=
value; }
 remove { CommandManager.RequerySuggested
-= value; }
 }

 public void Execute(object parameter)
 {
 execute((T)parameter);
 }
 }

 public class RelayCommand : ICommand
 {
 private readonly Action execute;
 private readonly Func<bool> canExecute;

 public  RelayCommand(Action  execute)  :
this(execute, null)
 {
 }

 public    RelayCommand(Action    execute,
Func<bool> canExecute)
 {
 if (execute == null)
 {
 throw                                  new
ArgumentNullException(nameof(execute));
 }

 this.execute = execute;
 this.canExecute = canExecute;
 }

 public bool CanExecute(object parameter)
 {
 return  canExecute  ==  null  ?  true  :
canExecute();
 }

 public        event        EventHandler
CanExecuteChanged
 {
 add { CommandManager.RequerySuggested +=
value; }
 remove { CommandManager.RequerySuggested
-= value; }
 }

 public void Execute(object parameter)
 {
 execute();
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\Controls\BindableRic
hTextBox.cs

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;

namespace AIS.TestGenerator.UI.Controls
{
 class BindableRichTextBox : RichTextBox
 {
 public          static          readonly
DependencyProperty DocumentProperty =
 DependencyProperty.Register("Document",
typeof(FlowDocument),
 typeof(BindableRichTextBox),         new
FrameworkPropertyMetadata
 (null,                                 new
PropertyChangedCallback(OnDocumentChanged
)));

 public new FlowDocument Document
 {
 get
 {
 return
(FlowDocument)this.GetValue(DocumentPrope
rty);
 }
 set
 {
 this.SetValue(DocumentProperty, value);
 }
 }

 public            static            void
OnDocumentChanged(DependencyObject   obj,
DependencyPropertyChangedEventArgs args)
 {
 RichTextBox        richTextBox        =
(RichTextBox)obj;
 if (args.NewValue == null)
 {
 richTextBox.Document.Blocks.Clear();
 }
 else
 {
 richTextBox.Document                   =
(FlowDocument)args.NewValue;
 }
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\Converters\BooleanTo
VisibilityConverter.cs

```
using System;
using System.Globalization;
using System.Windows;
using System.Windows.Data;

namespace AIS.TestGenerator.UI.Converters
{
 class   BooleanToVisibilityConverter   :
IValueConverter
 {
 public object Convert(object value, Type
targetType, object parameter, CultureInfo
culture)
 {
 return          ((bool)value)          ?
Visibility.Visible                      :
Visibility.Collapsed;
 }

 public object ConvertBack(object value,
Type   targetType,   object   parameter,
CultureInfo culture)
 {
 throw new NotImplementedException();
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\Converters\NullToVis
ibilityConverter.cs

```
using System;
using System.Globalization;
using System.Windows;
using System.Windows.Data;

namespace AIS.TestGenerator.UI.Converters
{
 class    NullToVisibilityConverter    :
IValueConverter
 {
 public object Convert(object value, Type
targetType, object parameter, CultureInfo
culture)
 {
 return     value     ==     null     ?
Visibility.Collapsed                    :
Visibility.Visible;
 }

 public object ConvertBack(object value,
Type   targetType,   object   parameter,
CultureInfo culture)
 {
 throw new NotImplementedException();
 }
```

```
 }
}
```

Лістинг
AIS.TestGenerator.UI\Converters\ReverseBooleanConverter.cs

```
using System;
using System.Globalization;
using System.Windows.Data;

namespace AIS.TestGenerator.UI.Converters
{
 [ValueConversion(typeof(bool), typeof(bool))]
 class ReverseBooleanConverter : IValueConverter
 {
 public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
 {
 if (targetType != typeof(bool))
 throw new InvalidOperationException("The target must be a boolean");

 return !(bool)value;
 }

 public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
 {
 if (targetType != typeof(bool))
 throw new InvalidOperationException("The target must be a boolean");

 return !(bool)value;
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\Converters\ReverseBooleanToVisibilityConverter.cs

```
using System;
using System.Globalization;
using System.Windows;
using System.Windows.Data;

namespace AIS.TestGenerator.UI.Converters
{
 class ReverseBooleanToVisibilityConverter : IValueConverter
 {
 public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
 {
 return ((bool)value) ? Visibility.Collapsed : Visibility.Visible;
 }

 public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
 {
 throw new NotImplementedException();
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\Converters\TestTasksNumberPerKeyTerm.cs

```
using AIS.TermSearcher.Entities.Term;
using System;
using System.Globalization;
using System.Linq;
using System.Windows.Data;

namespace AIS.TestGenerator.UI.Converters
{
 class TestTasksNumberPerKeyTerm : IMultiValueConverter
 {
 public object Convert(object[] values, Type targetType, object parameter, CultureInfo culture)
 {
 ISectionContainTestTask currentSection = values[0] as ISectionContainTestTask;
 Term term = values[1] as Term;
 int testTasksNumber = 0;
 if (currentSection != null && term != null)
 {
 testTasksNumber = currentSection.TestTasks.Count(x => x.Keyterm.Text == term.Text);
 }

 return testTasksNumber.ToString();
 }

 public object[] ConvertBack(object value, Type[] targetTypes, object parameter, CultureInfo culture)
 {
 throw new NotImplementedException();
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\DocumentProcessing\FlowDocumentProvider.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Paragraph;
using AIS.DocxReader.Entities.Section;
using AIS.DocxReader.Entities.TextContainer;
using AIS.DocxReader.Entities.Word;
using ASI.DocxReader.Entities.Sentences;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Documents;
using FlowDocSection = System.Windows.Documents.Section;
using FlowParagraph = System.Windows.Documents.Paragraph;
using StringResources = AIS.TestGenerator.UI.Properties.Resources;

namespace AIS.TestGenerator.UI.DocumentProcessing
{
 class FlowDocumentProvider : IFlowDocumentProvider
 {
 private readonly IDocument document;

 public FlowDocumentProvider(IDocument document)
 {
 if (document == null)
 throw new ArgumentNullException(nameof(document));

 this.document = document;
 }

 public FlowDocument GetFlowDocument()
 {
 FlowDocument document = new FlowDocument();
 document.ColumnWidth = 10000;
 document.LineHeight = 5;
 var sections = from childSection in this.document.Sections
 select GenerateSection(childSection);

 document.Blocks.AddRange(sections);
 return document;
 }

 public List<(Type, int)> GetOriginalSentencesPath(Run run)
 {
 TextElement textElement = run;
 List<(Type, int)> sentencesPath = new List<(Type, int)>();

 int indexPreviousElement = -1;
 Type previousType = null;
 Inline previousInline = null;
 Block previousBlock = null;
 while (textElement != null)
 {
 if (textElement.Parent is Span castedSentences)
 {
 previousType = typeof(ISentences);
 previousInline = castedSentences;
 }
 else if (textElement.Parent is FlowParagraph castedParagraph)
 {
 indexPreviousElement = castedParagraph.Inlines.ToList().IndexOf(previousInline);
 sentencesPath.Add((previousType, indexPreviousElement));

 previousType = typeof(IParagraph);
 previousBlock = castedParagraph;
 }
 else if (textElement.Parent is FlowDocSection castedSection)
 {
 indexPreviousElement = castedSection.Blocks.Where(x => x.GetType() == previousBlock.GetType()).ToList().IndexOf(previousBlock);
 sentencesPath.Add((previousType, indexPreviousElement));

 previousType = typeof(ISection);
 previousBlock = castedSection;
 }
 else if (textElement.Parent is FlowDocument castedDocument)
 {
 indexPreviousElement = castedDocument.Blocks.ToList().IndexOf(previousBlock);
 sentencesPath.Add((previousType, indexPreviousElement));
 }

 textElement = textElement.Parent as TextElement;
 }

 sentencesPath.Reverse();
 return sentencesPath;
 }

 private FlowDocSection GenerateSection(ISection section)
 {
 FlowDocSection createdSection = new FlowDocSection();
 foreach (IParagraph paragraph in section.Paragraphs)
 {
 FlowParagraph createdParagraph = GenerateParagraph(paragraph);

createdSection.Blocks.Add(createdParagraph);
 }

 var createdChildSections = from childSection in section.Sections
 select GenerateSection(childSection);
```

```
createdSection.Blocks.AddRange(createdChi
ldSections);
 return createdSection;
 }

 private                    FlowParagraph
GenerateParagraph(IParagraph
docParagraph)
 {
 FlowParagraph    paragraph    =    new
FlowParagraph();
 paragraph.TextIndent = 20;
 foreach   (ISentences   sentences   in
docParagraph.Sentences)
 {
 Span         createdSentences         =
GenerateSpan(sentences);
 paragraph.Inlines.Add(createdSentences);
 }

 return paragraph;
 }

 private   Span   GenerateSpan(ISentences
sentences)
 {
 Span newSpan = new Span();
 IWordStyle previousStyle = null;
 foreach (ITextContainer textContainer in
sentences.TextContainers)
 {
 if (textContainer.Words.Count == 0)
 {
 continue;
 }

 string       textContainerValue       =
ApplySymbolTextContainers(textContainer);
 IWord            firstWord            =
textContainer.Words.First();
 if    (previousStyle    !=    null    &&
IsSameStyle(previousStyle,
firstWord.WordStyle))
 {
 (newSpan.Inlines.LastInline as Run).Text
+= textContainerValue;
 }
 else
 {
 Run         newRun         =         new
Run(textContainerValue);
 ApplyRunStyle(newRun,
firstWord.WordStyle);
 newSpan.Inlines.Add(newRun);
 }
 }

 return newSpan;
 }

 private                            string
ApplySymbolTextContainers(ITextContainer
container)
 {
 StringBuilder textContainerBuilder = new
StringBuilder();
 string text = container.Text;
 if     (container.StartSymbolType     ==
ESymbolType.FullStop                    ||
container.StartSymbolType               ==
ESymbolType.Comma                       ||
container.StartSymbolType               ==
ESymbolType.None)
 {

textContainerBuilder.Append(StringResourc
es.Space);
 }
 else  if  (container.StartSymbolType  ==
ESymbolType.OpenBracket                 ||
container.StartSymbolType               ==
ESymbolType.Quotation)
 {

textContainerBuilder.Append(container.Sta
rtSymbol);
 }


textContainerBuilder.Append(container.Tex
t);
 if      (container.EndSymbolType      ==
ESymbolType.Hyphen)
 {

textContainerBuilder.Append(StringResourc
es.Space);
 }

 if      (container.EndSymbolType      !=
ESymbolType.OpenBracket                 ||
container.StartSymbolType               !=
ESymbolType.Quotation)
 {

textContainerBuilder.Append(container.End
Symbol);
 }

 return textContainerBuilder.ToString();
 }

 private   void   ApplyRunStyle(Run   run,
IWordStyle wordStyle)
 {
 if (wordStyle.Bold)
 {
 run.FontWeight                          =
System.Windows.FontWeights.Bold;
 }

 if (wordStyle.Underline)
 {
 run.TextDecorations                     =
System.Windows.TextDecorations.Underline;
 }

 if (wordStyle.Italic)
 {
 run.FontStyle                           =
System.Windows.FontStyles.Italic;
 }
 }

 private    bool    IsSameStyle(IWordStyle
firstStyle, IWordStyle secondStyle)
 {
 return        firstStyle.Bold        ==
secondStyle.Bold &&
 firstStyle.Italic  ==  secondStyle.Italic
&&
 firstStyle.Underline                    ==
secondStyle.Underline;
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\DocumentProcessing\IFlowDocumentProvider.cs

```
using ASI.DocxReader.Entities.Sentences;
using System;
using System.Collections.Generic;
using System.Windows.Documents;

namespace
AIS.TestGenerator.UI.DocumentProcessing
{
 interface IFlowDocumentProvider
 {
 FlowDocument GetFlowDocument();

 List<(Type,                          int)>
GetOriginalSentencesPath(Run run);
 }
}
```

Лістинг
AIS.TestGenerator.UI\Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Resources;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Windows;

// General Information about an assembly
is controlled through the following
//  set  of  attributes.  Change  these
attribute   values   to   modify   the
information
// associated with an assembly.
[assembly:
AssemblyTitle("AIS.TestGenerator.UI")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly:
AssemblyProduct("AIS.TestGenerator.UI")]
[assembly: AssemblyCopyright("Copyright ©
2018")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the
types in this assembly not visible
// to COM components.  If you need to
access a type in this assembly from
// COM, set the ComVisible attribute to
true on that type.
[assembly: ComVisible(false)]

//In order to begin building localizable
applications, set
//<UICulture>CultureYouAreCodingWith</UIC
ulture> in your .csproj file
//inside a <PropertyGroup>. For example,
if you are using US english
//in   your   source   files,   set   the
<UICulture> to en-US. Then uncomment
//the  NeutralResourceLanguage  attribute
below. Update the "en-US" in
//the line below to match the UICulture
setting in the project file.

//[assembly:
NeutralResourcesLanguage("en-US",
UltimateResourceFallbackLocation.Satellit
e)]


[assembly: ThemeInfo(
 ResourceDictionaryLocation.None, //where
theme specific resource dictionaries are
located
 //(used if a resource is not found in
the page,
 // or application resource dictionaries)

ResourceDictionaryLocation.SourceAssembly
//where the generic resource dictionary
is located
 //(used if a resource is not found in
the page,
 // app, or any theme specific resource
dictionaries)
)]


// Version information for an assembly
consists of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you
can  default  the  Build  and  Revision
Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
```

```
[assembly: AssemblyVersion("1.0.0.0")]
[assembly:
AssemblyFileVersion("1.0.0.0")]
```

Лістинг
AIS.TestGenerator.UI\Properties\Resources
.Designer.cs

```
//------------------------------------------
------------------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
//  Changes to this file may cause
incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------
------------------------------------------

namespace AIS.TestGenerator.UI.Properties
{
 using System;


 /// <summary>
 /// A strongly-typed resource class, for
looking up localized strings, etc.
 /// </summary>
 // This class was auto-generated by the
StronglyTypedResourceBuilder
 // class via a tool like ResGen or
Visual Studio.
 // To add or remove a member, edit your
.ResX file then rerun ResGen
 // with the /str option, or rebuild your
VS project.

[global::System.CodeDom.Compiler.Generate
dCodeAttribute("System.Resources.Tools.St
ronglyTypedResourceBuilder", "15.0.0.0")]

[global::System.Diagnostics.DebuggerNonUs
erCodeAttribute()]

[global::System.Runtime.CompilerServices.
CompilerGeneratedAttribute()]
 internal class Resources {

 private                              static
global::System.Resources.ResourceManager
resourceMan;

 private                              static
global::System.Globalization.CultureInfo
resourceCulture;


[global::System.Diagnostics.CodeAnalysis.
SuppressMessageAttribute("Microsoft.Perfo
rmance",
"CA1811:AvoidUncalledPrivateCode")]
 internal Resources() {
 }

 /// <summary>
 /// Returns the cached ResourceManager
instance used by this class.
 /// </summary>

[global::System.ComponentModel.EditorBrow
sableAttribute(global::System.ComponentMo
del.EditorBrowsableState.Advanced)]
 internal                             static
global::System.Resources.ResourceManager
ResourceManager {
 get {
 if  (object.ReferenceEquals(resourceMan,
null)) {
 global::System.Resources.ResourceManager
temp                 =                 new
global::System.Resources.ResourceManager(
"AIS.TestGenerator.UI.Properties.Resource
s", typeof(Resources).Assembly);
 resourceMan = temp;
 }
 return resourceMan;
 }
 }

 /// <summary>
 ///  Overrides the current thread's
CurrentUICulture property for all
 /// resource lookups using this strongly
typed resource class.
 /// </summary>

[global::System.ComponentModel.EditorBrow
sableAttribute(global::System.ComponentMo
del.EditorBrowsableState.Advanced)]
 internal                             static
global::System.Globalization.CultureInfo
Culture {
 get {
 return resourceCulture;
 }
 set {
 resourceCulture = value;
 }
 }

 /// <summary>
 /// Looks up a localized resource of
type System.Drawing.Bitmap.
 /// </summary>
 internal  static  System.Drawing.Bitmap
add {
 get {
 object              obj              =
ResourceManager.GetObject("add",
resourceCulture);
 return ((System.Drawing.Bitmap)(obj));
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Сформовано маскою.
 /// </summary>
 internal static string CreatedByMask {
 get {
 return
ResourceManager.GetString("CreatedByMask"
, resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to &lt;masks&gt;
 ///                                &lt;mask
id=&quot;1C33DD321D4214A2ABC22A122FB3D4AB
&quot;&gt;
 /// &lt;signature&gt;
 /// &lt;BeginSentence /&gt;
 /// &lt;Connector /&gt;
 /// &lt;TermGroup /&gt;
 /// &lt;/signature&gt;
 ///         &lt;connector&gt;         –
це&lt;/connector&gt;
 ///
&lt;connector&gt;називається&lt;/connecto
r&gt;
 ///
&lt;connector&gt;називаються&lt;/connecto
r&gt;
 ///
&lt;connector&gt;називають&lt;/connector&
gt;
 ///
&lt;connector&gt;описує&lt;/connector&gt;
 ///
&lt;connector&gt;описують&lt;/connector&g
t;
 ///
&lt;connector&gt;означає&lt;/connector&gt
;
 ///
&lt;connector&gt;означають&lt;/connector&
gt;
 ///
&lt;connector&gt;представляє&lt;/connecto
r&gt;
 ///
&lt;connector&gt;представляють&lt;/connec
tor&gt;
 ///  &lt;conn  [rest  of  string  was
truncated]&quot;;.
 /// </summary>
 internal static string Criterions {
 get {
 return
ResourceManager.GetString("Criterions",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized resource of
type System.Drawing.Bitmap.
 /// </summary>
 internal  static  System.Drawing.Bitmap
delete {
 get {
 object              obj              =
ResourceManager.GetObject("delete",
resourceCulture);
 return ((System.Drawing.Bitmap)(obj));
 }
 }

 /// <summary>
 /// Looks up a localized resource of
type System.Drawing.Bitmap.
 /// </summary>
 internal  static  System.Drawing.Bitmap
Edit {
 get {
 object              obj              =
ResourceManager.GetObject("Edit",
resourceCulture);
 return ((System.Drawing.Bitmap)(obj));
 }
 }

 /// <summary>
 /// Looks up a localized resource of
type System.Drawing.Bitmap.
 /// </summary>
 internal  static  System.Drawing.Bitmap
exit {
 get {
 object              obj              =
ResourceManager.GetObject("exit",
resourceCulture);
 return ((System.Drawing.Bitmap)(obj));
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Word document (*.docx)|*.docx.
 /// </summary>
 internal static string FileFilter {
 get {
 return
ResourceManager.GetString("FileFilter",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Ключовий термін.
 /// </summary>
 internal static string KeyTerm {
 get {
 return
ResourceManager.GetString("KeyTerm",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Список.
```

```
 /// </summary>
 internal static string List {
 get {
 return ResourceManager.GetString("List",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized resource of
type System.Drawing.Bitmap.
 /// </summary>
 internal static System.Drawing.Bitmap
loading {
 get {
 object obj =
ResourceManager.GetObject("loading",
resourceCulture);
 return ((System.Drawing.Bitmap)(obj));
 }
 }

 /// <summary>
 /// Looks up a localized resource of
type System.Drawing.Bitmap.
 /// </summary>
 internal static System.Drawing.Bitmap
question {
 get {
 object obj =
ResourceManager.GetObject("question",
resourceCulture);
 return ((System.Drawing.Bitmap)(obj));
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to .
 /// </summary>
 internal static string Space {
 get {
 return
ResourceManager.GetString("Space",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Структура.
 /// </summary>
 internal static string Structure {
 get {
 return
ResourceManager.GetString("Structure",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized resource of
type System.Drawing.Bitmap.
 /// </summary>
 internal static System.Drawing.Bitmap
tests {
 get {
 object obj =
ResourceManager.GetObject("tests",
resourceCulture);
 return ((System.Drawing.Bitmap)(obj));
 }
 }

 /// <summary>
 /// Looks up a localized resource of
type System.Drawing.Bitmap.
 /// </summary>
 internal static System.Drawing.Bitmap
testsBuilder {
 get {
 object obj =
ResourceManager.GetObject("testsBuilder",
resourceCulture);
 return ((System.Drawing.Bitmap)(obj));
 }

 /// <summary>
 /// Looks up a localized string similar
to Тип тестового завдання.
 /// </summary>
 internal static string TestTaskType {
 get {
 return
ResourceManager.GetString("TestTaskType",
resourceCulture);
 }
 }

 /// <summary>
 /// Looks up a localized string similar
to Сформовано з речення.
 /// </summary>
 internal static string UsedSentences {
 get {
 return
ResourceManager.GetString("UsedSentences"
, resourceCulture);
 }
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Properties\Settings.Designer.cs

```
//------------------------------------------
------------------------------------------
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause
incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//------------------------------------------
------------------------------------------

namespace AIS.TestGenerator.UI.Properties
{



[global::System.Runtime.CompilerServices.
CompilerGeneratedAttribute()]

[global::System.CodeDom.Compiler.Generate
dCodeAttribute("Microsoft.VisualStudio.Ed
itors.SettingsDesigner.SettingsSingleFile
Generator", "15.7.0.0")]
 internal sealed partial class Settings :
global::System.Configuration.ApplicationS
ettingsBase {

 private static Settings defaultInstance
=
((Settings)(global::System.Configuration.
ApplicationSettingsBase.Synchronized(new
Settings())));

 public static Settings Default {
 get {
 return defaultInstance;
 }
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\ViewModels\BaseViewModel.cs

```
using System.ComponentModel;

namespace AIS.TestGenerator.UI.ViewModels
{
 class BaseViewModel :
INotifyPropertyChanged
 {
 public event PropertyChangedEventHandler
PropertyChanged;

 protected void OnChangeProperty(string
property)
 {
 if (PropertyChanged != null)
 {
 PropertyChanged.Invoke(this, new
PropertyChangedEventArgs(property));
 }
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\ViewModels\FinalTaskListViewModel.cs

```
using AIS.DocxReader.Entities.Document;
using System;
using System.Collections.Generic;
using System.Linq;

namespace AIS.TestGenerator.UI.ViewModels
{
 class FinalTaskListViewModel :
BaseViewModel
 {
 private readonly IDocument document;

 public FinalTaskListViewModel(IDocument
document)
 {
 if (document == null) throw new
ArgumentNullException(nameof(document));

 this.document = document;
 LoadList();
 }

 private List<TestTasksSectionViewModel>
testTasksSections;
 public List<TestTasksSectionViewModel>
TestTasksSections
 {
 get
 {
 return this.testTasksSections;
 }
 }

 private void LoadList()
 {
 this.testTasksSections =
new List<TestTasksSectionViewModel>();
 foreach (var childSection in
document.Sections)
 {

this.testTasksSections.AddRange(GetChildS
ectionList(childSection as
ISectionContainTestTask));
 }
 }

 private List<TestTasksSectionViewModel>
GetChildSectionList(ISectionContainTestTa
sk currentSection)
 {
 List<TestTasksSectionViewModel> sections
= new List<TestTasksSectionViewModel>();
 TestTasksSectionViewModel
currentTestStructure =
new
TestTasksSectionViewModel
 {
 Name = currentSection.Name,
 Tasks =
new
List<TestTaskEditorViewModel>()
 };

 sections.Add(currentTestStructure);

currentTestStructure.Tasks.AddRange(curre
ntSection.TestTasks.Select(task =>
new
TestTaskEditorViewModel(task, false)));
```

```
 foreach (var childSection in currentSection.Sections)
 {

sections.AddRange(GetChildSectionList(childSection as ISectionContainTestTask));
 }

 return sections;
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\ViewModels\GeneratedTestTasksViewModel.cs

```
using AIS.DocxReader.Entities.Document;
using System;
using System.Windows.Input;

namespace AIS.TestGenerator.UI.ViewModels
{
 class GeneratedTestTasksViewModel : BaseViewModel
 {
 private readonly IDocument document;

 public GeneratedTestTasksViewModel(IDocument document)
 {
 if (document == null) throw new ArgumentNullException(nameof(document));

 this.document = document;
 this.changeViewModel = new RelayCommand(ChangeViewModelImp);
 ChangeViewModelImp();
 }

 private ICommand changeViewModel;
 public ICommand ChangeViewModel
 {
 get { return this.changeViewModel; }
 }

 private string switchContentButtonText;
 public string SwitchContentButtonText
 {
 get
 {
 return this.switchContentButtonText;
 }
 set
 {
 this.switchContentButtonText = value;

OnChangeProperty(nameof(this.SwitchContentButtonText));
 }
 }

 BaseViewModel mainContentViewModel;
 public BaseViewModel MainContentViewModel
 {
 get
 {
 return this.mainContentViewModel;
 }
 set
 {
 this.mainContentViewModel = value;

OnChangeProperty(nameof(this.MainContentViewModel));
 }
 }

 private void ChangeViewModelImp()
 {
 if (this.MainContentViewModel is TestTaskInStructureViewModel)
 {
 this.MainContentViewModel = new TestTaskListViewModel(this.document);
 this.SwitchContentButtonText = Properties.Resources.Structure;
 }
 else
 {
 this.MainContentViewModel = new TestTaskInStructureViewModel(this.document);
 this.SwitchContentButtonText = Properties.Resources.List;
 }
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\ViewModels\GenerationResultViewModel.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Entities.Word;
using AIS.TestGenerator.Export;
using AIS.TestGenerator.TestTask.Comparers;
using AIS.TestGenerator.TestTask;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Input;

namespace AIS.TestGenerator.UI.ViewModels
{
 class GenerationResultViewModel : BaseViewModel
 {
 private readonly IDocument document;
 private readonly List<Word> unKnownWords;
 private readonly IDialogFactory dialogFactory;

 public GenerationResultViewModel(IDocument document, List<Word> unKnownWords, IDialogFactory dialogFactory)
 {
 this.document = document ?? throw new ArgumentNullException(nameof(document));
 this.unKnownWords = unKnownWords ?? throw new ArgumentNullException(nameof(unKnownWords));
 this.dialogFactory = dialogFactory ?? throw new ArgumentNullException(nameof(dialogFactory));

 this.showGeneratedTests = new RelayCommand(OnShowGeneratedTests);
 this.showUnknownWords = new RelayCommand(OnShowUnknownWords);
 this.showTestCoverage = new RelayCommand(OnShowTestCoverage);
 this.showFinalTaskList = new RelayCommand(OnShowFinalTaskList);
 this.exportToGift = new RelayCommand(OnExportToGift);
 OnShowGeneratedTests();
 }

 #region Properties
 private BaseViewModel contentViewModel;
 public BaseViewModel ContentViewModel
 {
 get { return this.contentViewModel; }
 set
 {
 this.contentViewModel = value;

OnChangeProperty(nameof(this.ContentViewModel));
 }
 }
 #endregion

 #region Commands
 private ICommand showGeneratedTests;
 public ICommand ShowGeneratedTests
 {
 get { return this.showGeneratedTests; }
 }

 private ICommand showUnknownWords;
 public ICommand ShowUnknownWords
 {
 get { return this.showUnknownWords; }
 }

 private ICommand showTestCoverage;
 public ICommand ShowTestCoverage
 {
 get { return this.showTestCoverage; }
 }

 private ICommand showFinalTaskList;
 public ICommand ShowFinalTaskList
 {
 get { return this.showFinalTaskList; }
 }

 private ICommand exportToGift;
 public ICommand ExportToGift
 {
 get { return this.exportToGift; }
 }
 #endregion

 private void OnShowGeneratedTests()
 {
 this.ContentViewModel = new GeneratedTestTasksViewModel(this.document);
 }

 private void OnShowUnknownWords()
 {
 this.ContentViewModel = new UnknownWordsViewModel(this.unKnownWords);
 }

 private void OnShowFinalTaskList()
 {
 this.ContentViewModel = new FinalTaskListViewModel(this.document);
 }

 private void OnShowTestCoverage()
 {
 this.ContentViewModel = new TestTaskCoverageViewModel(this.document);
 }

 private void OnExportToGift()
 {
 Microsoft.Win32.FileDialog fileDialog = dialogFactory.GetSelectFolderDialog();
 fileDialog.Filter = "txt files (*.txt)|*.txt";
 bool? dialogResult = fileDialog.ShowDialog();
 if (dialogResult.HasValue && dialogResult.Value)
 {
 List<TestTask.TestTask> taskToSave = GenerateFullTestsList();

 //var singleChose = taskToSave.Where(x => x.TestType == TestTask.ETestType.SingleChoice).Take(10);
 //var inputAnswer = taskToSave.Where(x => x.TestType == TestTask.ETestType.InputAnswer).Take(10);
 //var multipleChoice = taskToSave.Where(x => x.TestType == TestTask.ETestType.MultipleChoice).Take(10);
 //var yesNo = taskToSave.Where(x => x.TestType == TestTask.ETestType.YesNo).Take(10);
```

```
 //taskToSave            =             new
List<TestTask.TestTask>(40);
 //taskToSave.AddRange(singleChose);
 //taskToSave.AddRange(inputAnswer);
 //taskToSave.AddRange(multipleChoice);
 //taskToSave.AddRange(yesNo);

 TestTaskExporterFactory  exporterFactory
= new TestTaskExporterFactory();
 ITestTasksExporter    giftExporter    =
exporterFactory.GetGiftTestTaskExporter()
;
 giftExporter.Export(taskToSave,
fileDialog.FileName);
 }
 }

 private          List<TestTask.TestTask>
GenerateFullTestsList()
 {
 IEnumerable<TestTask.TestTask> testTasks
= Enumerable.Empty<TestTask.TestTask>();
 foreach       (var       section       in
this.document.Sections)
 {
 IEnumerable<TestTask.TestTask>
childSectionTests                        =
GenerateTestTaskListForSection(section as
ISectionContainTestTask);
 testTasks                                =
testTasks.Union(childSectionTests,    new
TestTaskComparer());
 }

 return testTasks.ToList();
 }

 private   IEnumerable<TestTask.TestTask>
GenerateTestTaskListForSection(ISectionCo
ntainTestTask parentSection)
 {
 IEnumerable<TestTask.TestTask> testTasks
= parentSection.TestTasks;
 foreach       (var       section       in
parentSection.Sections)
 {
 IEnumerable<TestTask.TestTask>
childSectionTests                        =
GenerateTestTaskListForSection(section as
ISectionContainTestTask);
 testTasks                                =
testTasks.Union(childSectionTests,    new
TestTaskComparer());
 }

 return testTasks;
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\ViewModels\MainWindo
wViewModel.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Searcher;
using AIS.TermSearcher.Searcher.Settings;
using AIS.TestGenerator.Masks;
using Microsoft.Win32;
using System;
using System.Windows.Input;

namespace AIS.TestGenerator.UI.ViewModels
{
 class       MainWindowViewModel       :
BaseViewModel
 {
 private     readonly     IDialogFactory
dialogFactory;

 private IMasksManager masksManager;
 public
MainWindowViewModel(IDialogFactory
dialogFactory)
 {
 if (dialogFactory == null)
 {
 throw                                  new
ArgumentNullException(nameof(dialogFactor
y));
 }

 this.dialogFactory = dialogFactory;

 this.startTestGenerator      =      new
RelayCommand(StartGeneration,    ()    =>
!string.IsNullOrEmpty(this.SelectedFilePa
th));
 this.browseFiles          =          new
RelayCommand(OnBrowseFiles);

 this.styleCoefficientViewModel   =   new
StyleCoefficientEditorViewModel();
 this.masksManager                        =
TestGeneratorFactory.CreateMaskManager("R
esources/Criterions.xml");
 this.masksManager.LoadMasks();
 }

 #region Commands
 private ICommand startTestGenerator;

 public ICommand StartTestGenerator
 {
 get { return this.startTestGenerator; }
 }

 private ICommand browseFiles;
 public ICommand BrowseFiles
 {
 get { return this.browseFiles; }
 }

 #endregion

 #region Properties
 private string selectedFilePath;
 public string SelectedFilePath
 {
 get { return this.selectedFilePath; }
 set
 {
 this.selectedFilePath = value;

OnChangeProperty(nameof(this.SelectedFile
Path));
 }
 }

 private  StyleCoefficientEditorViewModel
styleCoefficientViewModel;
 public   StyleCoefficientEditorViewModel
StyleCoefficientViewModel
 {
 get              {              return
this.styleCoefficientViewModel; }
 }

 private bool isBusy;
 public bool IsBusy
 {
 get{ return this.isBusy; }
 set
 {
 this.isBusy = value;
 OnChangeProperty(nameof(this.IsBusy));
 }
 }
 #endregion

 private async void StartGeneration()
 {
 ISearchManager searchManager = null;
 ITestGenerator testGenerator;
 string mesage = "";
 await
System.Threading.Tasks.Task.Factory.Start
New(() =>
 {
 this.IsBusy = true;
 SearchSetting   searchSetting   =   new
SearchSetting
 {
 CorrectionSetting          =          new
CorrectionSetting()
 {
 BoldFontCoefficient                      =
(float)this.styleCoefficientViewModel.Bol
d,
 ItalicFontCoefficient                    =
(float)this.styleCoefficientViewModel.Ita
lic,
 UnderlineFontCoefficient                 =
(float)this.styleCoefficientViewModel.Und
erline
 }
 };

 DocxReader.DocxReader   reader   =   new
DocxReader.DocxReader(this.SelectedFilePa
th);
 IDocument document = reader.Read();

 searchManager                            =
TermSearcherFactory.CreateSearchManager(d
ocument, 5, searchSetting);
 searchManager.Search();

 testGenerator                            =
TestGeneratorFactory.CreateTestGenerator(
searchManager.AnalizedDocument,
searchManager.NauseaResult,
masksManager);
 testGenerator.GenerateParallel();
 });

 var             unknownWord             =
searchManager.UnknownWords;
 this.IsBusy = false;
 var               dialog                =
this.dialogFactory.CreateTestGenerationRe
sult(searchManager.AnalizedDocument,
unknownWord);
 dialog.ShowDialog();
 }

 private void OnBrowseFiles()
 {
 OpenFileDialog   openFileDialog   =   new
OpenFileDialog();
 openFileDialog.Filter                    =
Properties.Resources.FileFilter;
 if (openFileDialog.ShowDialog() == true)
 {
 this.SelectedFilePath                    =
openFileDialog.FileName;
 }
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\ViewModels\StyleCoef
ficientEditorViewModel.cs

```
namespace AIS.TestGenerator.UI.ViewModels
{
 class  StyleCoefficientEditorViewModel  :
BaseViewModel
 {
 public StyleCoefficientEditorViewModel()
 {
 this.bold = 1;
 this.italic = 1;
 this.underline = 1;
 }

 #region
 private double bold;
 public double Bold
 {
 get { return this.bold; }
 set
 {
 this.bold = value;
 OnChangeProperty(nameof(this.Bold));
```

```
 }
 }

 private double italic;
 public double Italic
 {
 get { return this.italic; }
 set
 {
 this.italic = value;
 OnChangeProperty(nameof(this.Italic));
 }
 }

 private double underline;
 public double Underline
 {
 get { return this.underline; }
 set
 {
 this.underline = value;

OnChangeProperty(nameof(this.underline));
 }
 }
 #endregion
 }
}
```

Лістинг
AIS.TestGenerator.UI\ViewModels\TestGenerationResultViewModel.cs

```
using AIS.DocxReader.Entities.Document;
using System;
using System.Windows.Input;

namespace AIS.TestGenerator.UI.ViewModels
{
 class TestGenerationResultViewModel : BaseViewModel
 {
 private readonly IDocument document;

 public TestGenerationResultViewModel(IDocument document)
 {
 if (document == null) throw new ArgumentNullException(nameof(document));

 this.document = document;
 this.changeViewModel = new RelayCommand(ChangeViewModelImp);
 ChangeViewModelImp();
 }

 private ICommand changeViewModel;
 public ICommand ChangeViewModel
 {
 get{ return this.changeViewModel; }
 }

 private string switchContentButtonText;
 public string SwitchContentButtonText
 {
 get
 {
 return this.switchContentButtonText;
 }
 set
 {
 this.switchContentButtonText = value;

OnChangeProperty(nameof(this.SwitchContentButtonText));
 }
 }

 BaseViewModel mainContentViewModel;
 public BaseViewModel MainContentViewModel
 {
 get
 {
 return this.mainContentViewModel;
 }
 set
 {
 this.mainContentViewModel = value;

OnChangeProperty(nameof(this.MainContentViewModel));
 }
 }

 private void ChangeViewModelImp()
 {
 if (this.MainContentViewModel is TestTaskListViewModel)
 {
 this.MainContentViewModel = new TestTaskInStructureViewModel(this.document);
 this.SwitchContentButtonText = Properties.Resources.List;
 }
 else
 {
 this.MainContentViewModel = new TestTaskListViewModel(this.document);
 this.SwitchContentButtonText = Properties.Resources.Structure;
 }
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\ViewModels\TestTaskCoverageViewModel.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Paragraph;
using AIS.DocxReader.Entities.Section;
using AIS.TestGenerator.UI.DocumentProcessing;
using ASI.DocxReader.Entities.Sentences;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Documents;
using System.Windows.Input;
using FlowParagraph = System.Windows.Documents.Paragraph;

namespace AIS.TestGenerator.UI.ViewModels
{
 class TestTaskCoverageViewModel : BaseViewModel
 {
 private readonly IDocument document;
 private readonly IFlowDocumentProvider flowDocumentProvider;

 private TextElement selectedTextElement;

 public TestTaskCoverageViewModel(IDocument document)
 {
 if (document == null)
 throw new ArgumentNullException(nameof(document));

 this.document = document;
 this.flowDocumentProvider = new FlowDocumentProvider(document);

 contentDocumentClick = new RelayCommand<Run>(OnContentDocumentClick);
 }

 private ICommand contentDocumentClick;
 public ICommand ContentDocumentClick
 {
 get
 {
 return this.contentDocumentClick;
 }
 }

 private FlowDocument flowDocument;
 public FlowDocument FlowDocument
 {
 get
 {
 if (flowDocument == null)
 {
 this.flowDocument = this.flowDocumentProvider.GetFlowDocument();
 }

 return this.flowDocument;
 }
 }

 private List<TestTaskEditorViewModel> testTasks;
 public List<TestTaskEditorViewModel> TestTasks
 {
 get { return this.testTasks; }
 set
 {
 this.testTasks = value;
 OnChangeProperty(nameof(TestTasks));
 }
 }

 private void OnContentDocumentClick(Run clickedRun)
 {
 List<(Type type, int index)> path = this.flowDocumentProvider.GetOriginalSentencesPath(clickedRun);
 IParagraph paragraph = null;
 ISentences sentences = null;
 object documentPart = this.document.Sections[path[0].index];
 for (int i = 1; i < path.Count; i++)
 {
 if (path[i].type == typeof(ISection))
 {
 documentPart = (documentPart as ISection).Sections[path[i].index];
 }
 else if (path[i].type == typeof(IParagraph))
 {
 documentPart = (documentPart as ISection).Paragraphs[path[i].index];
 paragraph = documentPart as IParagraph;
 }
 else if (path[i].type == typeof(ISentences))
 {
 documentPart = (documentPart as IParagraph).Sentences[path[i].index];
 sentences = documentPart as ISentences;
 }
 }

 Func<TestTask.TestTask, bool> predicate = task => task.Sentences == sentences;

 TextElement selectTextElement = null;
 List<TestTask.TestTask> testTasks = GetTestsByPredicate(predicate);
 if (testTasks.Count != 0)
 {
 selectTextElement = GetParentTextElementByType(clickedRun, typeof(Span));
 }
 else
 {
 predicate = task => paragraph.Sentences.Contains(task.Sentences);

 selectTextElement = GetParentTextElementByType(clickedRun, typeof(FlowParagraph));
 testTasks = GetTestsByPredicate(predicate);
```

```
 }

 SetGeneratedTask(testTasks);
 SetTextSelection(selectTextElement);
 }

 private                              void
SetGeneratedTask(List<TestTask.TestTask>
testTasks)
 {
 List<TestTaskEditorViewModel>
testTaskViewModels          =          new
List<TestTaskEditorViewModel>();
 foreach (var test in testTasks)
 {
 testTaskViewModels.Add(new
TestTaskEditorViewModel(test, false));
 }

 this.TestTasks = testTaskViewModels;
 }

 private                              void
SetTextSelection(TextElement textElement)
 {
 if (this.selectedTextElement != null)
 {
 this.selectedTextElement.Background    =
textElement.Background;
 }

 textElement.Background                 =
System.Windows.Media.Brushes.LightBlue;
 this.selectedTextElement = textElement;
 }

 private                       TextElement
GetParentTextElementByType(Run  run,  Type
type)
 {
 TextElement element = run;
 while (element.GetType() != type)
 {
 element = element.Parent as TextElement;
 }

 return element;
 }

 private          List<TestTask.TestTask>
GetTestsByPredicate(Func<TestTask.TestTas
k, bool> predicate)
 {
 List<TestTask.TestTask>  result  =  new
List<TestTask.TestTask>();
 foreach      (var      section      in
document.Sections)
 {

result.AddRange(GetTestFromSectionByPredi
cate(section  as  ISectionContainTestTask,
predicate));
 }

 return result;
 }

 private          List<TestTask.TestTask>
GetTestFromSectionByPredicate(ISectionCon
tainTestTask                       section,
Func<TestTask.TestTask, bool> predicate)
 {
 List<TestTask.TestTask>    testTasks    =
section.TestTasks.Where(predicate).ToList
();
 foreach     (var     childSection     in
section.Sections)
 {

testTasks.AddRange(GetTestFromSectionByPr
edicate(childSection                    as
ISectionContainTestTask, predicate));
 }

 return testTasks;
 }
}
```

Лістинг AIS.TestGenerator.UI\ViewModels\TestTaskEditorViewModel.cs

```
using AIS.TestGenerator.TestTask;
using System;
using System.Collections.ObjectModel;
using System.Linq;
using System.Windows;
using System.Windows.Input;

namespace AIS.TestGenerator.UI.ViewModels
{
 class      TestTaskEditorViewModel      :
BaseViewModel
 {
 private    readonly    TestTask.TestTask
currentTestTask;
 private                 TestTask.TestTask
clonedTestTask;

 public
TestTaskEditorViewModel(TestTask.TestTask
testTask, bool editMode)
 {
 if (testTask == null)
 {
 throw                                  new
ArgumentNullException(nameof(testTask));
 }

 this.currentTestTask = testTask;
 this.isEditMode = editMode;
 this.AnswerOptions          =          new
ObservableCollection<AnswerOption>(this.c
urrentTestTask.AnswerOptions);

 this.clonedTestTask                    =
(TestTask.TestTask)testTask.Clone();

 this.AddAnswer           =           new
RelayCommand(AddNewAnswer);
 this.DeleteAnswer          =          new
RelayCommand<object>(DeleteAnswerImplemen
tation);
 this.SaveTask            =            new
RelayCommand(SaveTestTaskImp);
 this.ExitFromEditMode        =        new
RelayCommand(ExitFromEditModeImp);
 }

 #region Events
 public event EventHandler SaveTestTask;
 public         event         EventHandler
EditModeChanged;
 #endregion

 #region Properties

 public ETestType TestType
 {
 get               {               return
this.currentTestTask.TestType; }
 set  {  this.currentTestTask.TestType  =
value; }
 }

 public string Header
 {
 get { return this.currentTestTask.Task;
}
 set
 {
 this.currentTestTask.Task = value;
 OnChangeProperty(nameof(Header));
 }
 }

 private
ObservableCollection<AnswerOption>
answerOptions;

 public
ObservableCollection<AnswerOption>
AnswerOptions
 {
 get { return this.answerOptions; }
 set
 {
 this.answerOptions = value;
 OnChangeProperty(nameof(AnswerOptions));
 }
 }

 private bool isEditMode;

 public bool IsEditMode
 {
 get { return this.isEditMode; }
 set
 {
 this.isEditMode = value;
 OnChangeProperty(nameof(IsEditMode));
 if (this.EditModeChanged != null)
 {

this.EditModeChanged.Invoke(this.isEditMo
de, EventArgs.Empty);
 }
 }
 }

 #endregion

 #region Commands
 private ICommand addAnswer;

 public ICommand AddAnswer
 {
 get { return this.addAnswer; }
 set { this.addAnswer = value; }
 }

 private ICommand deleteAnswer;

 public ICommand DeleteAnswer
 {
 get { return this.deleteAnswer; }
 set { this.deleteAnswer = value; }
 }

 private ICommand saveTask;

 public ICommand SaveTask
 {
 get { return this.saveTask; }
 set { this.saveTask = value; }
 }

 public ICommand exitFromEditMode;
 public ICommand ExitFromEditMode
 {
 get { return this.exitFromEditMode; }
 set { this.exitFromEditMode = value; }
 }

 #endregion

 private                              void
DeleteAnswerImplementation(object
deleteObject)
 {
 AnswerOption    answerOptionToDelete    =
deleteObject as AnswerOption;
 if (answerOptionToDelete != null)
 {

this.AnswerOptions.Remove(answerOptionToD
elete);
 }
 }

 private void AddNewAnswer()
 {
 this.AnswerOptions.Add(new
AnswerOption());
 }
```

```
 private void SaveTestTaskImp()
 {
 if (this.SaveTestTask != null)
 {
 this.currentTestTask.AnswerOptions = this.AnswerOptions.ToList();

this.SaveTestTask.Invoke(this.currentTestTask, EventArgs.Empty);
 }
 }

 private void ExitFromEditModeImp()
 {
 this.IsEditMode = false;
 if (TaskHasChanges())
 {
 MessageBoxResult result = MessageBox.Show("Ви маєте не збережені зміни. Зберегти зміни?", "Увага", MessageBoxButton.YesNo, MessageBoxImage.Warning);
 if (result == MessageBoxResult.Yes)
 {
 SaveTestTaskImp();
 }
 else
 {
 this.currentTestTask.AnswerOptions = this.clonedTestTask.AnswerOptions;
 this.currentTestTask.Keyterm = this.clonedTestTask.Keyterm;
 this.Header= this.clonedTestTask.Task;
 this.AnswerOptions = new ObservableCollection<AnswerOption>(this.clonedTestTask.AnswerOptions);

 this.clonedTestTask = (TestTask.TestTask)this.currentTestTask.Clone();
 }
 }
 }

 private bool TaskHasChanges()
 {
 return !string.Equals(this.clonedTestTask.Task, this.Header)
 || this.clonedTestTask.AnswerOptions.Except(this.AnswerOptions, new AnswerOptionComparer()).Count() != 0
 || this.AnswerOptions.Except(this.clonedTestTask.AnswerOptions, new AnswerOptionComparer()).Count() != 0;
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\ViewModels\TestTaskInStructureViewModel.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Section;
using AIS.DocxReader.Entities.Sentences;
using AIS.TermSearcher.Entities.Term;
using ASI.DocxReader.Entities.Sentences;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;

namespace AIS.TestGenerator.UI.ViewModels
{
 class TestTaskInStructureViewModel : BaseViewModel
 {
 private readonly IDocument document;

 private bool isShowTestPreview;
 private ICommand showOrCloseTestPreview;
 private ICommand changeDocumentSection;
 private ISectionContainTestTask selectedDocumentSection;

 public TestTaskInStructureViewModel(IDocument document)
 {
 if (document == null)
 {
 throw new ArgumentNullException(nameof(document));
 }

 this.document = document;

 this.ShowOrCloseTestPreview = new RelayCommand(() => this.IsShowTestPreview = !this.IsShowTestPreview);
 this.ChangeDocumentSection = new RelayCommand<ISection>(this.ChangeSelectedDocumentSection);
 this.KeytermSelected = new RelayCommand(this.KeytermSelectionChanged);
 this.EditTestTask = new RelayCommand(this.EditTestTaskImplementation, () => this.SelectedTest != null);
 this.DeleteTestTask = new RelayCommand(this.DeleteTestTaskImplementation, () => this.SelectedTest != null);
 this.IsEditMode = false;
 }

 #region Properties
 public List<ISection> DocumentSections
 {
 get { return this.document.Sections; }
 }

 public bool IsShowTestPreview
 {
 get { return this.isShowTestPreview; }
 set { this.isShowTestPreview = value; }
 }

 private ObservableCollection<Term> sectionKeyterm;
 public ObservableCollection<Term> SectionKeyterm
 {
 get { return sectionKeyterm; }
 set
 {
 sectionKeyterm = value;

OnChangeProperty(nameof(SectionKeyterm));
 }
 }

 private bool isEditMode;
 public bool IsEditMode
 {
 get { return isEditMode; }
 set
 {
 isEditMode = value;
 OnChangeProperty(nameof(IsEditMode));
 }
 }

 private Term selectedKeyterm;
 public Term SelectedKeyterm
 {
 get { return selectedKeyterm; }
 set { selectedKeyterm = value; }
 }

 private ObservableCollection<TestTask.TestTask> testTasks;
 public ObservableCollection<TestTask.TestTask> TestTasks
 {
 get { return this.testTasks; }
 set
 {
 this.testTasks = value;
 OnChangeProperty(nameof(TestTasks));
 }
 }

 private TestTask.TestTask selectedTest;
 public TestTask.TestTask SelectedTest
 {
 get { return this.selectedTest; }
 set
 {
 this.selectedTest = value;

OnChangeProperty(nameof(this.SelectedTest));
 OnSelectTestTaskChanded();
 }
 }

 private TestTaskEditorViewModel testEditorViewModel;
 public TestTaskEditorViewModel TestEditorViewModel
 {
 get { return testEditorViewModel; }
 set
 {
 this.testEditorViewModel = value;

OnChangeProperty(nameof(this.TestEditorViewModel));
 }
 }

 private FlowDocument generatedTestDocumentRegion;
 public FlowDocument GeneratedTestDocumentRegion
 {
 get { return generatedTestDocumentRegion; }
 set
 {
 generatedTestDocumentRegion = value;

OnChangeProperty(nameof(GeneratedTestDocumentRegion));
 }
 }

 #endregion

 #region Commands
 public ICommand ShowOrCloseTestPreview
 {
 get { return this.showOrCloseTestPreview; }
 set { this.showOrCloseTestPreview = value; }
 }

 public ICommand ChangeDocumentSection
 {
 get { return this.changeDocumentSection; }
 set { this.changeDocumentSection = value; }
 }

 private ICommand keytermSelected;
 public ICommand KeytermSelected
 {
 get { return this.keytermSelected; }
 set { this.keytermSelected = value; }
 }

 private ICommand editTestTask;

 public ICommand EditTestTask
 {
 get { return editTestTask; }
 set { editTestTask = value; }
```

```
 }

 private ICommand deleteTestTask;

 public ICommand DeleteTestTask
 {
 get { return deleteTestTask; }
 set { deleteTestTask = value; }
 }
 #endregion

 private                                void
ChangeSelectedDocumentSection(ISection
selectedItem)
 {
 if           (selectedItem           is
ISectionContainTestTask castedISection)
 {
 this.SectionKeyterm         =         new
ObservableCollection<Term>(castedISection
.KeyTerms);
 selectedDocumentSection                =
castedISection;
 }
 }

 private void KeytermSelectionChanged()
 {
 this.TestTasks           =           new
ObservableCollection<TestTask.TestTask>(t
his.selectedDocumentSection.TestTasks
 .Where(x     =>     x.Keyterm.Text     ==
this.SelectedKeyterm.Text));
 }

 private void OnSelectTestTaskChanded()
 {
 if (this.SelectedTest != null)
 {
 this.TestEditorViewModel      =      new
TestTaskEditorViewModel(this.SelectedTest
, this.IsEditMode);
 this.TestEditorViewModel.SaveTestTask +=
TestTaskEditorViewModel_SaveTestTask;
 this.TestEditorViewModel.EditModeChanged
+=
TestTaskEditorViewModel_EditModeChanged;

 this.GeneratedTestDocumentRegion       =
GetUsedDocumentRegionForGenerateSelectedT
est();
 }
 else
 {
 this.TestEditorViewModel = null;
 this.GeneratedTestDocumentRegion = null;
 }
 }

 private                                void
TestTaskEditorViewModel_EditModeChanged(o
bject sender, EventArgs e)
 {
 if (sender is bool)
 {
 this.IsEditMode = (bool)sender;
 }
 }

 private                                void
TestTaskEditorViewModel_SaveTestTask(obje
ct sender, EventArgs e)
 {
 var       currentSelectedTestTask      =
this.SelectedTest;
 KeytermSelectionChanged();
 this.SelectedTest                      =
currentSelectedTestTask;
 }

 private                                void
EditTestTaskImplementation()
 {
 this.TestEditorViewModel.IsEditMode    =
true;
 }

 private                                void
DeleteTestTaskImplementation()
 {

this.selectedDocumentSection.TestTasks.Re
move(this.SelectedTest);
 if      (selectedDocumentSection      is
ISectionContainTestTask castedISection)
 {
 this.SectionKeyterm         =         new
ObservableCollection<Term>(castedISection
.KeyTerms);
 }

 KeytermSelectionChanged();
 }

 private                        FlowDocument
GetUsedDocumentRegionForGenerateSelectedT
est()
 {
 var        usedDocumentParagraph       =
this.selectedDocumentSection.AllParagraph
s.First(x                              =>
x.Text.Contains(this.selectedTest.Sentenc
es.Text));

 FlowDocument    flowDocument    =    new
FlowDocument();
 Paragraph paragraph = new Paragraph();
 ISentences        lastSentences        =
usedDocumentParagraph.Sentences.Last();
 foreach    (ISentences    sentences    in
usedDocumentParagraph.Sentences)
 {
 string        sentencesSeparator       =
GetSentencesSeparator(sentences.Sentences
Type);
 string        sentencesFullText        =
sentences.Text + sentencesSeparator;
 Run     selectedSentences     =     new
Run(sentencesFullText);
 if             (sentences             ==
this.selectedTest.Sentences)
 {
 selectedSentences.Background    =    new
SolidColorBrush(Colors.Yellow);
 }


paragraph.Inlines.Add(selectedSentences);
 if (sentences != lastSentences)
 {
 paragraph.Inlines.Add(new Run(" "));
 }
 }

 return new FlowDocument(paragraph);
 }

 private                                string
GetSentencesSeparator(ESentencesType
sentencesType)
 {
 switch (sentencesType)
 {
 case ESentencesType.Interrogative:
 return "?";
 case ESentencesType.Exclamatory:
 return "!";
 default:
 return ".";
 }
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\ViewModels\TestTaskListViewModel.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Section;
using System;
using System.Collections.Generic;

namespace AIS.TestGenerator.UI.ViewModels
{
 class      TestTaskListViewModel      :
BaseViewModel
 {
 private readonly IDocument document;

 public   TestTaskListViewModel(IDocument
document)
 {
 if   (document   ==   null)   throw   new
ArgumentException(nameof(document));

 this.document = document;

 List<TestTask.TestTask> joinedTestTask =
new List<TestTask.TestTask>();
 foreach      (var      section      in
document.Sections)
 {

joinedTestTask.AddRange(JoinTestTasks(sec
tion));
 }

 this.TestTasks = joinedTestTask;
 }

 public List<TestTask.TestTask> TestTasks
 {
 get; private set;
 }

 private                TestTask.TestTask
selectedTestTask;
 public                 TestTask.TestTask
SelectedTestTask
 {
 set
 {
 selectedTestTask = value;

OnChangeProperty(nameof(TestEditorViewMod
el));

OnChangeProperty(nameof(TestProperties));
 }
 }

 public           TestTaskEditorViewModel
TestEditorViewModel
 {
 get
 {
 TestTaskEditorViewModel result = null;
 if (selectedTestTask != null)
 {
 result               =               new
TestTaskEditorViewModel(this.selectedTest
Task, false);
 }

 return result;
 }
 }

 public    Dictionary<string,    string>
TestProperties
 {
 get
 {
 Dictionary<string, string> properties =
null;
 if (this.selectedTestTask != null)
 {
 properties                             =
GenerateTestTaskProperties();
 }

 return properties;
 }
 }

 private          List<TestTask.TestTask>
JoinTestTasks(ISection section)
```

```
 {
 List<TestTask.TestTask> testTasks = new List<TestTask.TestTask>();
 if (section is ISectionContainTestTask castedSection)
 {

testTasks.AddRange(castedSection.TestTasks);
 }

 foreach (var childSection in section.Sections)
 {
 if (childSection is ISectionContainTestTask castedchildSection)
 {

testTasks.AddRange(castedchildSection.TestTasks);
 }
 }

 return testTasks;
 }

 private Dictionary<string, string> GenerateTestTaskProperties()
 {
 Dictionary<string, string> properties = new Dictionary<string, string>();

properties.Add(Properties.Resources.CreatedByMask,
this.selectedTestTask.GeneratedByTaskMask.QuestionMask.InnerXml);

properties.Add(Properties.Resources.KeyTerm, this.selectedTestTask.Keyterm.Text);

properties.Add(Properties.Resources.UsedSentences,
this.selectedTestTask.Sentences.Text);

properties.Add(Properties.Resources.TestTaskType,
this.selectedTestTask.TestType.ToString());

 return properties;
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\ViewModels\TestTasksSectionViewModel.cs

```
using System.Collections.Generic;

namespace AIS.TestGenerator.UI.ViewModels
{
 class TestTasksSectionViewModel : BaseViewModel
 {
 public string Name { get; set; }

 public List<TestTaskEditorViewModel> Tasks { get; set; }
 }
}
```

Лістинг AIS.TestGenerator.UI\ViewModels\UnknownWordsViewModel.cs

```
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AIS.TestGenerator.UI.ViewModels
{
 class UnknownWordsViewModel : BaseViewModel
 {
 public UnknownWordsViewModel(List<Word> unKnownWords)
 {
 if (unKnownWords == null) throw new ArgumentNullException(nameof(unKnownWords));

 this.unKnownWords = unKnownWords;
 }

 #region Properties
 private List<Word> unKnownWords;
 public List<Word> UnKnownWords
 {
 get{ return this.unKnownWords; }
 }
 #endregion

 #region Commands

 #endregion
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\TestGenerationResult.xaml

```
<Window
x:Class="AIS.TestGenerator.UI.Views.TestGenerationResult"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:ie="clr-namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
 xmlns:viewModel="clr-namespace:AIS.TestGenerator.UI.ViewModels"
 xmlns:userControls="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 xmlns:local="clr-namespace:AIS.TestGenerator.UI.Views"
 mc:Ignorable="d"
 Title="TestGenerationResult" Height="500" Width="1000" MinHeight="600" MinWidth="800">
 <Window.Resources>
 <ResourceDictionary>
 <DataTemplate DataType="{x:Type viewModel:GenerationResultViewModel}">
 <userControls:GenerationResultView/>
 </DataTemplate>
 </ResourceDictionary>
 </Window.Resources>
 <ContentControl Grid.Row="1" Content="{Binding}"/>
</Window>
```

Лістинг AIS.TestGenerator.UI\Views\TestGenerationResult.xaml.cs

```
using System.Windows;

namespace AIS.TestGenerator.UI.Views
{
 /// <summary>
 /// Interaction logic for TestGenerationResult.xaml
 /// </summary>
 public partial class TestGenerationResult : Window
 {
 public TestGenerationResult()
 {
 InitializeComponent();
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\TestTaskListView.xaml

```
<Window
x:Class="AIS.TestGenerator.UI.Views.TestTaskListView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:viewModel="clr-namespace:AIS.TestGenerator.UI.ViewModels"
 xmlns:userControls="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 xmlns:local="clr-namespace:AIS.TestGenerator.UI.Views"
 mc:Ignorable="d"
 Title="TestTaskListView" Height="450" Width="800">
 <Window.Resources>
 <ResourceDictionary>
 <DataTemplate DataType="{x:Type viewModel:TestTaskEditorViewModel}">
 <userControls:TestTaskEditor/>
 </DataTemplate>
 <ResourceDictionary.MergedDictionaries>
 <ResourceDictionary Source="Style/ScrollBar.xaml" />
 <ResourceDictionary Source="Style/Converters.xaml" />
 </ResourceDictionary.MergedDictionaries>
 </ResourceDictionary>
 </Window.Resources>
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="0.60*"/>
 <RowDefinition Height="5"/>
 <RowDefinition Height="0.4*"/>
 </Grid.RowDefinitions>
 <DataGrid ItemsSource="{Binding TestTasks}" AutoGenerateColumns="False" SelectedItem="{Binding SelectedTestTask}" HeadersVisibility="Column" IsReadOnly="True">
 <DataGrid.Columns>
 <DataGridTextColumn Header="Тестові завдання" Width="*" Binding="{Binding Task}">
 <DataGridTextColumn.HeaderStyle>
 <Style TargetType="DataGridColumnHeader">
 <Setter Property="HorizontalContentAlignment" Value="Center" />
 </Style>
 </DataGridTextColumn.HeaderStyle>
 </DataGridTextColumn>
 </DataGrid.Columns>
 </DataGrid>
 <GridSplitter Grid.Row="1" HorizontalAlignment="Stretch" Height="5"/>
 <Grid Grid.Row="2">
 <Grid.ColumnDefinitions>
 <ColumnDefinition/>
```

```
 <ColumnDefinition Width="5"/>
 <ColumnDefinition/>
 </Grid.ColumnDefinitions>

 <Grid Background="#ececec" Grid.Row="0"
Grid.Column="0">
 <ContentControl Content="{Binding
TestEditorViewModel, Mode=OneWay}"/>
 </Grid>
 <GridSplitter Grid.Column="1"
HorizontalAlignment="Stretch" Width="5"
/>
 <DataGrid Grid.Column="2"
IsReadOnly="True" ItemsSource="{Binding
TestProperties}"
AutoGenerateColumns="False"
HeadersVisibility="Column">
 <DataGrid.Columns>
 <DataGridTemplateColumn
Header="Властивість" Width="100">
 <DataGridTemplateColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTemplateColumn.HeaderStyle>
 <DataGridTemplateColumn.CellTemplate>
 <DataTemplate>
 <TextBlock Text="{Binding Key}"
TextWrapping="Wrap"
VerticalAlignment="Center"/>
 </DataTemplate>
 </DataGridTemplateColumn.CellTemplate>
 </DataGridTemplateColumn>

 <DataGridTemplateColumn
Header="Значення" Width="0.6*">
 <DataGridTemplateColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTemplateColumn.HeaderStyle>
 <DataGridTemplateColumn.CellTemplate>
 <DataTemplate>
 <TextBlock Text="{Binding Value}"
TextWrapping="Wrap"
VerticalAlignment="Center"/>
 </DataTemplate>
 </DataGridTemplateColumn.CellTemplate>
 </DataGridTemplateColumn>
 </DataGrid.Columns>
 </DataGrid>
 </Grid>
 </Grid>
</Window>
```

Лістинг AIS.TestGenerator.UI\Views\TestTaskListView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace AIS.TestGenerator.UI.Views
{
 /// <summary>
 /// Interaction logic for
TestTaskListView.xaml
 /// </summary>
 public partial class TestTaskListView :
Window
 {
 public TestTaskListView()
 {
 InitializeComponent();
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\Style\Converters.xaml

```
<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx
/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/win
fx/2006/xaml"
 xmlns:local="clr-
namespace:AIS.TestGenerator.UI.Views.Styl
e"
 xmlns:converters="clr-
namespace:AIS.TestGenerator.UI.Converters
">
 <converters:NullToVisibilityConverter
x:Key="nullToVisibilityConverter"/>
 <converters:BooleanToVisibilityConverter
x:Key="booleanToVisibilityConverter"/>

<converters:ReverseBooleanToVisibilityCon
verter
x:Key="reverseBooleanToVisibilityConverte
r"/>
 <converters:ReverseBooleanConverter
x:Key="reverseBooleanConverter"/>
 <converters:TestTasksNumberPerKeyTerm
x:Key="testTasksNumberPerKeyTerm" />
</ResourceDictionary>
```

Лістинг AIS.TestGenerator.UI\Views\Style\ScrollBar.xaml

```
<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx
/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/win
fx/2006/xaml"
 xmlns:local="clr-
namespace:AIS.TestGenerator.UI.Views.Styl
e">
 <!--Scrollbar Thumbs-->
 <Style x:Key="ScrollThumbs"
TargetType="{x:Type Thumb}">
 <Setter Property="Template">
 <Setter.Value>
 <ControlTemplate TargetType="{x:Type
Thumb}">
 <Grid x:Name="Grid">
 <Rectangle HorizontalAlignment="Stretch"
VerticalAlignment="Stretch" Width="Auto"
Height="Auto" Fill="Transparent" />
 <Border x:Name="Rectangle1"
CornerRadius="5"
HorizontalAlignment="Stretch"
VerticalAlignment="Stretch" Width="Auto"
Height="Auto"
Background="{TemplateBinding Background}"
/>
 </Grid>
 <ControlTemplate.Triggers>
 <Trigger Property="Tag"
Value="Horizontal">
 <Setter TargetName="Rectangle1"
Property="Width" Value="Auto" />
 <Setter TargetName="Rectangle1"
Property="Height" Value="7" />
 </Trigger>
 </ControlTemplate.Triggers>
 </ControlTemplate>
 </Setter.Value>
 </Setter>
 </Style>
 <!--ScrollBars-->
 <Style x:Key="{x:Type ScrollBar}"
TargetType="{x:Type ScrollBar}">
 <Setter
Property="Stylus.IsFlicksEnabled"
Value="false" />
 <Setter Property="Foreground"
Value="#8C8C8C" />
 <Setter Property="Background"
Value="Transparent" />
 <Setter Property="Width" Value="8" />
 <Setter Property="Template">
 <Setter.Value>
 <ControlTemplate TargetType="{x:Type
ScrollBar}">
 <Grid x:Name="GridRoot" Width="8"
Background="{TemplateBinding
Background}">
 <Grid.RowDefinitions>
 <RowDefinition Height="0.00001*" />
 </Grid.RowDefinitions>
 <Track x:Name="PART_Track" Grid.Row="0"
IsDirectionReversed="true"
Focusable="false">
 <Track.Thumb>
 <Thumb x:Name="Thumb"
Background="{TemplateBinding Foreground}"
Style="{DynamicResource ScrollThumbs}" />
 </Track.Thumb>
 <Track.IncreaseRepeatButton>
 <RepeatButton x:Name="PageUp"
Command="ScrollBar.PageDownCommand"
Opacity="0" Focusable="false" />
 </Track.IncreaseRepeatButton>
 <Track.DecreaseRepeatButton>
 <RepeatButton x:Name="PageDown"
Command="ScrollBar.PageUpCommand"
Opacity="0" Focusable="false" />
 </Track.DecreaseRepeatButton>
 </Track>
 </Grid>
 <ControlTemplate.Triggers>
 <Trigger SourceName="Thumb"
Property="IsMouseOver" Value="true">
 <Setter Value="{DynamicResource
ButtonSelectBrush}" TargetName="Thumb"
Property="Background" />
 </Trigger>
 <Trigger SourceName="Thumb"
Property="IsDragging" Value="true">
 <Setter Value="{DynamicResource
DarkBrush}" TargetName="Thumb"
Property="Background" />
 </Trigger>
 <Trigger Property="IsEnabled"
Value="false">
 <Setter TargetName="Thumb"
Property="Visibility" Value="Collapsed"
/>
 </Trigger>
 <Trigger Property="Orientation"
Value="Horizontal">
 <Setter TargetName="GridRoot"
Property="LayoutTransform">
 <Setter.Value>
 <RotateTransform Angle="-90" />
 </Setter.Value>
 </Setter>
 <Setter TargetName="PART_Track"
Property="LayoutTransform">
 <Setter.Value>
 <RotateTransform Angle="-90" />
 </Setter.Value>
 </Setter>
 <Setter Property="Width" Value="Auto" />
 <Setter Property="Height" Value="8" />
 <Setter TargetName="Thumb"
Property="Tag" Value="Horizontal" />
 <Setter TargetName="PageDown"
Property="Command"
Value="ScrollBar.PageLeftCommand" />
 <Setter TargetName="PageUp"
Property="Command"
Value="ScrollBar.PageRightCommand" />
 </Trigger>
 </ControlTemplate.Triggers>
 </ControlTemplate>
```

```
 </Setter.Value>
 </Setter>
 </Style>
</ResourceDictionary>
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\FinalTaskListView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.FinalTaskListView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:viewModel="clr-namespace:AIS.TestGenerator.UI.ViewModels"
 xmlns:userControls="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 xmlns:local="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 mc:Ignorable="d"
 d:DesignHeight="450"
d:DesignWidth="800">
 <UserControl.Resources>
 <ResourceDictionary>
 <DataTemplate          DataType="{x:Type viewModel:TestTaskEditorViewModel}">
 <userControls:TestTaskEditor/>
 </DataTemplate>
 </ResourceDictionary>
 </UserControl.Resources>
 <Grid>
 <ScrollViewer              Height="Auto" PreviewMouseWheel="ScrollViewer_PreviewMouseWheel">
 <ListView          ItemsSource="{Binding TestTasksSections}"
ScrollViewer.VerticalScrollBarVisibility="Auto"
ScrollViewer.HorizontalScrollBarVisibility="Disabled">
 <ListView.ItemTemplate>
 <DataTemplate>
 <Expander>
 <Expander.Header>
 <TextBlock     Text="{Binding     Name}" Width="{Binding
 RelativeSource={RelativeSource
 Mode=FindAncestor,
 AncestorType={x:Type Expander}},
 Path=ActualWidth}"/>
 </Expander.Header>
 <ListView ItemsSource="{Binding Tasks}" ScrollViewer.HorizontalScrollBarVisibility="Disabled"     Background="Transparent" BorderThickness="0">
 <ListView.ItemContainerStyle>
 <Style TargetType="ListViewItem">
 <Setter
Property="HorizontalContentAlignment"
Value="Stretch"/>
 </Style>
 </ListView.ItemContainerStyle>
 <ListView.ItemTemplate>
 <DataTemplate>
 <ContentControl Content="{Binding}" />
 </DataTemplate>
 </ListView.ItemTemplate>
 </ListView>
 </Expander>
 </DataTemplate>
 </ListView.ItemTemplate>
 </ListView>
 </ScrollViewer>
 </Grid>
</UserControl>
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\FinalTaskListView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace
AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for FinalTaskListView.xaml
 /// </summary>
 public partial class FinalTaskListView : UserControl
 {
 public FinalTaskListView()
 {
 InitializeComponent();
 }

 private                              void ScrollViewer_PreviewMouseWheel(object sender, MouseWheelEventArgs e)
 {
 ScrollViewer scv = (ScrollViewer)sender;

scv.ScrollToVerticalOffset(scv.VerticalOffset - e.Delta);
 e.Handled = true;
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\GeneratedTestTasksView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.GeneratedTestTasksView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:local="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 xmlns:viewModel="clr-namespace:AIS.TestGenerator.UI.ViewModels"
 mc:Ignorable="d"
 d:DesignHeight="450"
d:DesignWidth="800">
 <UserControl.Resources>
 <ResourceDictionary>
 <DataTemplate          DataType="{x:Type viewModel:TestTaskEditorViewModel}">
 <local:TestTaskEditor/>
 </DataTemplate>
 <DataTemplate          DataType="{x:Type viewModel:TestTaskListViewModel}">
 <local:TestTaskList/>
 </DataTemplate>
 <DataTemplate          DataType="{x:Type viewModel:TestTaskInStructureViewModel}">
 <local:TestTaskInStructure/>
 </DataTemplate>
 <ResourceDictionary.MergedDictionaries>
 <ResourceDictionary
Source="pack://application:,,,/Views/Style/ScrollBar.xaml" />
 <ResourceDictionary
Source="pack://application:,,,/Views/Style/Converters.xaml" />
 </ResourceDictionary.MergedDictionaries>
 </ResourceDictionary>
 </UserControl.Resources>
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="20"/>
 <RowDefinition/>
 </Grid.RowDefinitions>
 <Menu Grid.Row="0">
 <Menu.ItemsPanel>
 <ItemsPanelTemplate>
 <DockPanel
HorizontalAlignment="Stretch"/>
 </ItemsPanelTemplate>
 </Menu.ItemsPanel>
 <MenuItem               Header="{Binding SwitchContentButtonText}"
HorizontalAlignment="Right"
Command="{Binding ChangeViewModel}"/>
 </Menu>
 <ContentControl             Grid.Row="1"
Content="{Binding
MainContentViewModel}"/>
 </Grid>
</UserControl>
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\GeneratedTestTasksView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace
AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for GeneratedTestTasksView.xaml
 /// </summary>
 public           partial           class GeneratedTestTasksView : UserControl
 {
 public GeneratedTestTasksView()
 {
 InitializeComponent();
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\GenerationResultView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.GenerationResultView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:local="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 xmlns:viewModel="clr-namespace:AIS.TestGenerator.UI.ViewModels"
 mc:Ignorable="d"
 d:DesignHeight="450" d:DesignWidth="800">
 <UserControl.Resources>
 <ResourceDictionary>
 <DataTemplate          DataType="{x:Type viewModel:GeneratedTestTasksViewModel}">
 <local:GeneratedTestTasksView/>
 </DataTemplate>
 <DataTemplate          DataType="{x:Type viewModel:UnknownWordsViewModel}">
 <local:UnknownWordsView/>
 </DataTemplate>
 <DataTemplate          DataType="{x:Type viewModel:TestTaskCoverageViewModel}">
 <local:TestTaskCoverageView/>
 </DataTemplate>
 <DataTemplate          DataType="{x:Type viewModel:FinalTaskListViewModel}">
 <local:FinalTaskListView/>
 </DataTemplate>
 </ResourceDictionary>
 </UserControl.Resources>
 <Grid>
 <Grid.ColumnDefinitions>
 <ColumnDefinition Width="75"/>
 <ColumnDefinition Width="*"/>
 </Grid.ColumnDefinitions>
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition/>
 <RowDefinition Height="60"/>
 </Grid.RowDefinitions>
 <StackPanel Background="#ececec">
 <RadioButton          x:Name="testEditor" Height="60"              IsChecked="True" Command="{Binding ShowGeneratedTests}">
 <RadioButton.Template>
 <ControlTemplate>
 <Grid x:Name="Container">
 <Grid.RowDefinitions>
 <RowDefinition/>
 <RowDefinition Height="0.4*"/>
 </Grid.RowDefinitions>
 <Image Source="/Resources/testsBuilder.png" />
 <TextBlock  Grid.Row="1"  Text="Редактор тестів"      HorizontalAlignment="Center" FontSize="9"/>
 </Grid>
 <ControlTemplate.Triggers>
 <Trigger Property="RadioButton.IsChecked" Value="True">
 <Setter           TargetName="Container" Property="Background" Value="#ececca"/>
 </Trigger>
 <Trigger Property="RadioButton.IsMouseOver" Value="True">
 <Setter           TargetName="Container" Property="Background" Value="#ececda"/>
 </Trigger>
 </ControlTemplate.Triggers>
 </ControlTemplate>
 </RadioButton.Template>
 </RadioButton>
 <RadioButton       x:Name="unknownWords" Height="60"            Command="{Binding ShowUnknownWords}">
 <RadioButton.Template>
 <ControlTemplate TargetType="RadioButton">
 <Grid x:Name="Container">
 <Grid.RowDefinitions>
 <RowDefinition/>
 <RowDefinition Height="0.4*"/>
 </Grid.RowDefinitions>
 <Image  Source="/Resources/question.png" />
 <TextBlock  Grid.Row="1"  Text="Невідомі слова"       HorizontalAlignment="Center" FontSize="9"/>
 </Grid>
 <ControlTemplate.Triggers>
 <Trigger Property="RadioButton.IsChecked" Value="True">
 <Setter           TargetName="Container" Property="Background" Value="#ececca"/>
 </Trigger>
 <Trigger Property="RadioButton.IsMouseOver" Value="True">
 <Setter           TargetName="Container" Property="Background" Value="#ececde"/>
 </Trigger>
 </ControlTemplate.Triggers>
 </ControlTemplate>
 </RadioButton.Template>
 </RadioButton>
 <RadioButton       x:Name="testCoverage" Height="60"            Command="{Binding ShowTestCoverage}">
 <RadioButton.Template>
 <ControlTemplate TargetType="RadioButton">
 <Grid x:Name="Container">
 <Grid.RowDefinitions>
 <RowDefinition/>
 <RowDefinition Height="0.4*"/>
 </Grid.RowDefinitions>
 <Image Source="/Resources/testCoverage.png" />
 <TextBlock  Grid.Row="1"  Text="Покриття тестами"     HorizontalAlignment="Center" FontSize="9"/>
 </Grid>
 <ControlTemplate.Triggers>
 <Trigger Property="RadioButton.IsChecked" Value="True">
 <Setter           TargetName="Container" Property="Background" Value="#ececca"/>
 </Trigger>
 <Trigger Property="RadioButton.IsMouseOver" Value="True">
 <Setter           TargetName="Container" Property="Background" Value="#ececde"/>
 </Trigger>
 </ControlTemplate.Triggers>
 </ControlTemplate>
 </RadioButton.Template>
 </RadioButton>
 <RadioButton   x:Name="generationResult" Height="60"            Command="{Binding ShowFinalTaskList}">
 <RadioButton.Template>
 <ControlTemplate TargetType="RadioButton">
 <Grid x:Name="Container">
 <Grid.RowDefinitions>
 <RowDefinition/>
 <RowDefinition Height="0.4*"/>
 </Grid.RowDefinitions>
 <Image Source="/Resources/tests.png" />
 <TextBlock Grid.Row="1" Text="Результат" HorizontalAlignment="Center" FontSize="9"/>
 </Grid>
 <ControlTemplate.Triggers>
 <Trigger Property="RadioButton.IsChecked" Value="True">
 <Setter           TargetName="Container" Property="Background" Value="#ececca"/>
 </Trigger>
 <Trigger Property="RadioButton.IsMouseOver" Value="True">
 <Setter           TargetName="Container" Property="Background" Value="#ececde"/>
 </Trigger>
 </ControlTemplate.Triggers>
 </ControlTemplate>
 </RadioButton.Template>
 </RadioButton>

 </StackPanel>
 <Button Grid.Row="1" x:Name="giftExport" Padding="0"          Background="#ececec" Command="{Binding ExportToGift}">
 <Border Margin="0">
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition/>
 <RowDefinition Height="0.4*"/>
 </Grid.RowDefinitions>
 <Image    Source="/Resources/export.png" Grid.Row="0"/>
 <TextBlock    Text="Експорт    в    GIFT" Grid.Row="1" FontSize="9"/>
 </Grid>
 </Border>


 </Button>
 </Grid>

 <ContentControl          Grid.Column="1" Grid.Row="0"           Content="{Binding ContentViewModel}"/>
 </Grid>
</UserControl>
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\GenerationResultView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for GenerationResultView.xaml
 /// </summary>
 public           partial           class GenerationResultView : UserControl
 {
 public GenerationResultView()
 {
 InitializeComponent();
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\StyleCoefficientEditor.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.StyleCoefficientEditor"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:wpfTool="clr-namespace:Xceed.Wpf.Toolkit;assembly=Xceed.Wpf.Toolkit"
 xmlns:local="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 mc:Ignorable="d"
 d:DesignHeight="50" d:DesignWidth="400">
 <UserControl.Resources>
 <Style TargetType="TextBlock">
 <Setter     Property="VerticalAlignment" Value="Center"/>
 <Setter   Property="HorizontalAlignment" Value="Center"/>
 </Style>
 <Style TargetType="wpfTool:DoubleUpDown">
 <Setter                Property="Margin" Value="5,2,5,2"/>
 <Setter             Property="Increment" Value="0.1"/>
 </Style>
 </UserControl.Resources>
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition/>
 <RowDefinition/>
 </Grid.RowDefinitions>
 <Grid.ColumnDefinitions>
 <ColumnDefinition/>
 <ColumnDefinition/>
 <ColumnDefinition/>
 </Grid.ColumnDefinitions>
 <TextBlock  Grid.Row="0"  Grid.Column="0" Text="Жирний"/>
 <TextBlock  Grid.Row="0"  Grid.Column="1" Text="Курсив"/>
 <TextBlock  Grid.Row="0"  Grid.Column="2" Text="Підкреслений"/>

 <wpfTool:DoubleUpDown       Grid.Row="1" Grid.Column="0"  x:Name="boldCoefficient" Value="{Binding Bold}"/>
 <wpfTool:DoubleUpDown       Grid.Row="1" Grid.Column="1"
x:Name="italicCoefficient"
Value="{Binding Italic}"/>
 <wpfTool:DoubleUpDown       Grid.Row="1" Grid.Column="2"
x:Name="underlineCoefficient"
Value="{Binding Underline}"/>
 </Grid>
</UserControl>
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\StyleCoefficientEditor.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for StyleCoefficientEditor.xaml
 /// </summary>
 public           partial           class StyleCoefficientEditor : UserControl
 {
 public StyleCoefficientEditor()
 {
 InitializeComponent();
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\TestGenerationResultView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.TestGenerationResultView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:local="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 mc:Ignorable="d"
 d:DesignHeight="450"
d:DesignWidth="800">
 <UserControl.Resources>
 <Style TargetType="RadioButton">
 <Style.Triggers>
 <Trigger            Property="IsChecked" Value="True">
 <Setter            Property="Background" Value="Green"/>
 </Trigger>
 </Style.Triggers>
 </Style>
 </UserControl.Resources>
 <Grid>
 <Grid.ColumnDefinitions>
 <ColumnDefinition Width="60"/>
 <ColumnDefinition Width="*"/>
 </Grid.ColumnDefinitions>
 <StackPanel Background="#ececec">
 <RadioButton                 Height="60" IsChecked="True">
 <RadioButton.Template>
 <ControlTemplate TargetType="RadioButton">
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition/>
 <RowDefinition Height="0.4*"/>
 </Grid.RowDefinitions>
 <TextBlock   Grid.Row="1"   Text="Тестові завдання"    HorizontalAlignment="Center" FontSize="7"/>
 </Grid>
 </ControlTemplate>
 </RadioButton.Template>
 </RadioButton>
 <RadioButton Height="60">
 <RadioButton.Template>
 <ControlTemplate TargetType="RadioButton">
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition/>
 <RowDefinition Height="0.4*"/>
 </Grid.RowDefinitions>
 <TextBlock  Grid.Row="1"  Text="Невідомі слова"       HorizontalAlignment="Center" FontSize="7"/>
 </Grid>
 </ControlTemplate>
 </RadioButton.Template>
 </RadioButton>
 </StackPanel>
 </Grid>
</UserControl>
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\TestGenerationResultView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for TestGenerationResultView.xaml
 /// </summary>
 public           partial           class TestGenerationResultView : UserControl
 {
 public TestGenerationResultView()
 {
 InitializeComponent();
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\TestTaskCoverageView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.TestTaskCoverageView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:local="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 xmlns:i="clr-namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
 xmlns:viewModel="clr-namespace:AIS.TestGenerator.UI.ViewModels"

xmlns:gif="http://wpfanimatedgif.codeplex.com"
 xmlns:userControls="clr-namespace:AIS.TestGenerator.UI.Views.UserControls"
 mc:Ignorable="d"
 d:DesignHeight="450"
d:DesignWidth="800">
 <UserControl.Resources>
 <ResourceDictionary>
 <DataTemplate          DataType="{x:Type viewModel:TestTaskEditorViewModel}">
 <userControls:TestTaskEditor/>
 </DataTemplate>
 <ResourceDictionary.MergedDictionaries>
```

```
 <ResourceDictionary
Source="pack://application:,,,/Views/Style/Converters.xaml" />
 </ResourceDictionary.MergedDictionaries>
 </ResourceDictionary>
 </UserControl.Resources>
 <Grid>
 <Grid.ColumnDefinitions>
 <ColumnDefinition Width="0.65*"/>
 <ColumnDefinition Width="5"/>
 <ColumnDefinition Width="0.35*"/>
 </Grid.ColumnDefinitions>
 <FlowDocumentReader
x:Name="flowDocReader" Document="{Binding
FlowDocument}"
PreviewMouseLeftButtonUp="FlowDocumentReader_PreviewMouseLeftButtonUp" >
 </FlowDocumentReader>
 <GridSplitter            Grid.Column="1"
HorizontalAlignment="Stretch"/>
 <ScrollViewer            Grid.Column="2"
Height="Auto"
PreviewMouseWheel="ScrollViewer_PreviewMouseWheel">
 <ListView       Background="Transparent"
ItemsSource="{Binding         TestTasks}"
ScrollViewer.HorizontalScrollBarVisibility="Disabled">
 </ListView>
 </ScrollViewer>
 </Grid>
</UserControl>
```

Лістинг
AIS.TestGenerator.UI\Views\UserControls\TestTaskCoverageView.xaml.cs

```
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;

namespace
AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for
TestTaskCoverageView.xaml
 /// </summary>
 public           partial           class
TestTaskCoverageView : UserControl
 {
 public TestTaskCoverageView()
 {
 InitializeComponent();
 }

 private                                void
FlowDocumentReader_PreviewMouseLeftButtonUp(object sender, MouseButtonEventArgs e)
 {
 if (flowDocReader.Selection.IsEmpty)
 {
 Run             clickedRun             =
flowDocReader.Selection.Start.Parent   as
Run;

 object         contentClickCmd         =
this.DataContext.GetType().GetProperty("ContentDocumentClick").GetValue(this.DataContext, null);

contentClickCmd.GetType().GetMethod("Execute").Invoke(contentClickCmd,          new
object[] { clickedRun });
 var  viewModel  =  this.DataContext  as
ViewModels.TestTaskCoverageViewModel;
 }
 }

 private                                void
ScrollViewer_PreviewMouseWheel(object
sender, MouseWheelEventArgs e)
 {
 ScrollViewer scv = (ScrollViewer)sender;
scv.ScrollToVerticalOffset(scv.VerticalOffset - e.Delta);
 e.Handled = true;
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\Views\UserControls\TestTaskEditor.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.TestTaskEditor"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:testGenerator="clr-
namespace:AIS.TestGenerator.TestTask;assembly=AIS.TestGenerator"
 xmlns:local="clr-
namespace:AIS.TestGenerator.UI.Views.UserControls"
 mc:Ignorable="d"
 d:DesignHeight="450"
d:DesignWidth="800">
 <UserControl.Resources>
 <ResourceDictionary>
 <DataTemplate x:Key="Menu">
 <Menu            VerticalAlignment="Top"
Grid.Row="0">
 <Menu.ItemsPanel>
 <ItemsPanelTemplate>
 <DockPanel
HorizontalAlignment="Stretch"/>
 </ItemsPanelTemplate>
 </Menu.ItemsPanel>
 <MenuItem  Padding="0"  Command="{Binding
AddAnswer}"    ToolTip    ="Додати    нову
відповідь"           Visibility="{Binding
IsEditMode,     Converter={StaticResource
booleanToVisibilityConverter}}">
 <MenuItem.Header>
 <Image Source="/Resources/add.png"/>
 </MenuItem.Header>
 </MenuItem>
 <MenuItem  Padding="0"  Command="{Binding
SaveTask}"           Visibility="{Binding
IsEditMode,     Converter={StaticResource
booleanToVisibilityConverter}}">
 <MenuItem.Header>
 <Image Source="/Resources/save.png"/>
 </MenuItem.Header>
 </MenuItem>
 <MenuItem                     Padding="0"
HorizontalAlignment="Right"
Command="{Binding      ExitFromEditMode}"
ToolTip="Вийти       з       редагування"
Visibility="{Binding          IsEditMode,
Converter={StaticResource
booleanToVisibilityConverter}}">
 <MenuItem.Header>
 <Image Source="/Resources/exit.png"/>
 </MenuItem.Header>
 </MenuItem>
 </Menu>
 </DataTemplate>
 <ResourceDictionary.MergedDictionaries>
 <ResourceDictionary
Source="pack://application:,,,/Views/Style/ScrollBar.xaml" />
 <ResourceDictionary
Source="pack://application:,,,/Views/Style/Converters.xaml" />
 </ResourceDictionary.MergedDictionaries>
 </ResourceDictionary>
 </UserControl.Resources>
 <ContentControl         x:Name="control"
Content="{Binding}" Grid.Row="1" >
 <ContentControl.Resources>
 <DataTemplate x:Key="SingleChoice">
 <Border Padding="5">
 <ScrollViewer
VerticalScrollBarVisibility="Auto"
Margin="5,0,0,5" >
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="Auto" />
 <RowDefinition Height="*"/>
 </Grid.RowDefinitions>
 <TextBlock    Text="{Binding    Header}"
TextWrapping="Wrap"></TextBlock>
 <ItemsControl               Grid.Row="1"
ItemsSource="{Binding AnswerOptions}">
 <ItemsControl.ItemTemplate>
 <DataTemplate>
 <RadioButton         GroupName="Answers"
IsChecked="{Binding       IsRightAnswer}"
IsEnabled="False"
VerticalContentAlignment="Center"
Margin="0,0,0,5">
 <RadioButton.Content>
 <TextBlock    Text="{Binding    Answer}"
TextWrapping="Wrap"/>
 </RadioButton.Content>
 </RadioButton>
 </DataTemplate>
 </ItemsControl.ItemTemplate>
 </ItemsControl>
 </Grid>
 </ScrollViewer>
 </Border>
 </DataTemplate>
 <DataTemplate
x:Key="EditableSingleChoice">
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="20"/>
 <RowDefinition Height="*"/>
 </Grid.RowDefinitions>
 <ContentPresenter           Grid.Row="0"
Content="{Binding}"
ContentTemplate="{StaticResource Menu}"/>
 <ScrollViewer               Grid.Row="1"
VerticalScrollBarVisibility="Auto"
Margin="5,0,0,5">
 <Grid Grid.Row="1" Margin="5">
 <Grid.RowDefinitions>
 <RowDefinition Height="Auto" />
 <RowDefinition Height="*"/>
 </Grid.RowDefinitions>
 <TextBox   Grid.Row="0"   Text="{Binding
Header,
UpdateSourceTrigger=PropertyChanged}"
Background="Transparent"
BorderThickness="0" TextWrapping="Wrap"/>
 <ItemsControl               Grid.Row="1"
ItemsSource="{Binding AnswerOptions}">
 <ItemsControl.ItemTemplate>
 <DataTemplate>
 <RadioButton         GroupName="Answers"
IsChecked="{Binding       IsRightAnswer}"
VerticalContentAlignment="Center"
Margin="0,0,0,5">
 <RadioButton.Content>
 <Grid>
 <Grid.ColumnDefinitions>
 <ColumnDefinition/>
 <ColumnDefinition Width="25"/>
 </Grid.ColumnDefinitions>
 <TextBox   MinWidth="70"   Height="Auto"
Background="Transparent"
BorderThickness="0,0,0,2"  Text="{Binding
Answer,
UpdateSourceTrigger=PropertyChanged}"
TextWrapping="Wrap"/>
 <Button    Grid.Column="1"    Width="20"
Height="20"
HorizontalContentAlignment="Center"
BorderThickness="0"
VerticalContentAlignment="Center"
Margin="5,0,0,0" Background="Transparent"
```

```
Command="{Binding
DataContext.DeleteAnswer,
ElementName=control}"
CommandParameter="{Binding}" >
 <Image Source="/Resources/delete.png"/>
 </Button>
 </Grid>
 </RadioButton.Content>
 </RadioButton>
 </DataTemplate>
 </ItemsControl.ItemTemplate>
 </ItemsControl>
 </Grid>
 </ScrollViewer>
 </Grid>
 </DataTemplate>
 <DataTemplate x:Key="MultipleChoice">
 <Border Padding="5">
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition              Height="Auto"
MaxHeight="35"/>
 <RowDefinition Height="*"/>
 </Grid.RowDefinitions>
 <TextBlock    Text="{Binding    Header}"
TextWrapping="Wrap"></TextBlock>
 <ScrollViewer               Grid.Row="1"
VerticalScrollBarVisibility="Auto"
Margin="5,0,0,5">
 <ItemsControl      ItemsSource="{Binding
AnswerOptions}">
 <ItemsControl.ItemTemplate>
 <DataTemplate>
 <CheckBox            IsChecked="{Binding
IsRightAnswer}"         IsEnabled="False"
VerticalContentAlignment="Center"
Margin="0,0,0,5">
 <CheckBox.Content>
 <TextBlock    Text="{Binding    Answer}"
TextWrapping="Wrap"/>
 </CheckBox.Content>
 </CheckBox>
 </DataTemplate>
 </ItemsControl.ItemTemplate>
 </ItemsControl>
 </ScrollViewer>
 </Grid>
 </Border>
 </DataTemplate>
 <DataTemplate
x:Key="EditableMultipleChoice">
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="20"/>
 <RowDefinition Height="*"/>
 </Grid.RowDefinitions>
 <ContentPresenter           Grid.Row="0"
Content="{Binding}"
ContentTemplate="{StaticResource Menu}"/>
 <Grid Grid.Row="1" Margin="5">
 <Grid.RowDefinitions>
 <RowDefinition              Height="Auto"
MaxHeight="35"/>
 <RowDefinition Height="*"/>
 </Grid.RowDefinitions>
 <TextBox   Grid.Row="0"   Text="{Binding
Header,   UpdateSourceTrigger=LostFocus}"
Background="Transparent"
BorderThickness="0"
TextWrapping="Wrap"></TextBox>
 <ScrollViewer               Grid.Row="1"
VerticalScrollBarVisibility="Auto"
Margin="5,0,0,5">
 <ItemsControl      ItemsSource="{Binding
AnswerOptions}">
 <ItemsControl.ItemTemplate>
 <DataTemplate>
 <CheckBox            IsChecked="{Binding
IsRightAnswer}"
VerticalContentAlignment="Center"
Margin="0,0,0,5">
 <CheckBox.Content>
 <Grid>
 <Grid.ColumnDefinitions>
 <ColumnDefinition/>
 <ColumnDefinition Width="25"/>
 </Grid.ColumnDefinitions>
 <TextBox   MinWidth="70"   Height="Auto"
Background="Transparent"
BorderThickness="0,0,0,2"  Text="{Binding
Answer,
UpdateSourceTrigger=PropertyChanged}"
TextWrapping="Wrap"/>
 <Button    Grid.Column="1"    Width="20"
Height="20"
HorizontalContentAlignment="Center"
BorderThickness="0"
VerticalContentAlignment="Center"
Margin="5,0,0,0" Background="Transparent"
Command="{Binding
DataContext.DeleteAnswer,
ElementName=control}"
CommandParameter="{Binding}" >
 <Image Source="/Resources/delete.png"/>
 </Button>
 </Grid>
 </CheckBox.Content>
 </CheckBox>
 </DataTemplate>
 </ItemsControl.ItemTemplate>
 </ItemsControl>
 </ScrollViewer>
 </Grid>
 </Grid>
 </DataTemplate>
 <DataTemplate x:Key="InputAnswer">
 <ScrollViewer               Grid.Row="3"
VerticalScrollBarVisibility="Auto"
Margin="5,0,0,5" >
 <Border Padding="5">
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="Auto" />
 <RowDefinition Height="5"/>
 <RowDefinition Height="20"/>
 <RowDefinition Height="*"/>
 </Grid.RowDefinitions>
 <TextBlock    Text="{Binding    Header}"
TextWrapping="Wrap"></TextBlock>
 <TextBlock   Text="Варіанти   відповіді:"
Grid.Row="2"/>
 <ItemsControl               Grid.Row="3"
ItemsSource="{Binding AnswerOptions}">
 <ItemsControl.ItemTemplate>
 <DataTemplate>
 <TextBlock    Text="{Binding    Answer}"
TextWrapping="Wrap"/>
 </DataTemplate>
 </ItemsControl.ItemTemplate>
 </ItemsControl>
 </Grid>
 </Border>
 </ScrollViewer>
 </DataTemplate>
 <DataTemplate
x:Key="EditableInputAnswer">
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="20"/>
 <RowDefinition Height="*"/>
 </Grid.RowDefinitions>
 <ContentPresenter           Grid.Row="0"
Content="{Binding}"
ContentTemplate="{StaticResource Menu}"/>
 <ScrollViewer               Grid.Row="1"
VerticalScrollBarVisibility="Auto"
Margin="5,0,0,5" >
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="Auto" />
 <RowDefinition Height="5"/>
 <RowDefinition Height="20"/>
 <RowDefinition Height="*"/>
 </Grid.RowDefinitions>
 <TextBox   Grid.Row="0"   Text="{Binding
Header,
UpdateSourceTrigger=PropertyChanged}"
Background="Transparent"
BorderThickness="0" TextWrapping="Wrap"/>
 <TextBlock   Text="Варіанти   відповіді:"
Grid.Row="2"/>
 <ItemsControl               Grid.Row="3"
ItemsSource="{Binding AnswerOptions}">
 <ItemsControl.ItemTemplate>
 <DataTemplate>
 <Grid HorizontalAlignment="Left">
 <Grid.ColumnDefinitions>
 <ColumnDefinition/>
 <ColumnDefinition Width="25"/>
 </Grid.ColumnDefinitions>
 <TextBox   MinWidth="70"   Height="Auto"
Background="Transparent"
BorderThickness="0,0,0,2"  Text="{Binding
Answer,
UpdateSourceTrigger=PropertyChanged}"
TextWrapping="Wrap"/>
 <Button    Grid.Column="1"    Width="20"
Height="20"
HorizontalContentAlignment="Center"
BorderThickness="0"
VerticalContentAlignment="Center"
Margin="5,0,0,0" Background="Transparent"
Command="{Binding
DataContext.DeleteAnswer,
ElementName=control}"
CommandParameter="{Binding}" >
 <Image Source="/Resources/delete.png"/>
 </Button>
 </Grid>
 </DataTemplate>
 </ItemsControl.ItemTemplate>
 </ItemsControl>
 </Grid>
 </ScrollViewer>
 </Grid>
 </DataTemplate>
 </ContentControl.Resources>
 <ContentControl.Style>
 <Style TargetType="ContentControl">
 <Style.Triggers>
 <MultiDataTrigger>
 <MultiDataTrigger.Conditions>
 <Condition             Binding="{Binding
Path=TestType}"          Value="{x:Static
testGenerator:ETestType.YesNo}"/>
 <Condition             Binding="{Binding
Path=IsEditMode}" Value="False"/>
 </MultiDataTrigger.Conditions>
 <Setter
Property="ContentControl.ContentTemplate"
Value="{StaticResource SingleChoice}" />
 </MultiDataTrigger>
 <MultiDataTrigger>
 <MultiDataTrigger.Conditions>
 <Condition             Binding="{Binding
Path=IsEditMode}" Value="True"/>
 <Condition             Binding="{Binding
Path=TestType}"          Value="{x:Static
testGenerator:ETestType.YesNo}"/>
 </MultiDataTrigger.Conditions>
 <MultiDataTrigger.Setters>
 <Setter
Property="ContentControl.ContentTemplate"
Value="{StaticResource
EditableSingleChoice}"/>
 </MultiDataTrigger.Setters>
 </MultiDataTrigger>
 <MultiDataTrigger>
 <MultiDataTrigger.Conditions>
 <Condition             Binding="{Binding
Path=TestType}"          Value="{x:Static
testGenerator:ETestType.SingleChoice}"/>
 <Condition             Binding="{Binding
Path=IsEditMode}" Value="False"/>
 </MultiDataTrigger.Conditions>
 <MultiDataTrigger.Setters>
 <Setter
Property="ContentControl.ContentTemplate"
Value="{StaticResource SingleChoice}"/>
 </MultiDataTrigger.Setters>
 </MultiDataTrigger>
 <MultiDataTrigger>
 <MultiDataTrigger.Conditions>
 <Condition             Binding="{Binding
Path=IsEditMode}" Value="True"/>
```

```
 <Condition             Binding="{Binding
Path=TestType}"          Value="{x:Static
testGenerator:ETestType.SingleChoice}"/>
 </MultiDataTrigger.Conditions>
 <MultiDataTrigger.Setters>
 <Setter
Property="ContentControl.ContentTemplate"
Value="{StaticResource
EditableSingleChoice}" />
 </MultiDataTrigger.Setters>
 </MultiDataTrigger>
 <MultiDataTrigger>
 <MultiDataTrigger.Conditions>
 <Condition             Binding="{Binding
Path=TestType}"          Value="{x:Static
testGenerator:ETestType.MultipleChoice}"/
>
 <Condition             Binding="{Binding
Path=IsEditMode}" Value="False"/>
 </MultiDataTrigger.Conditions>
 <MultiDataTrigger.Setters>
 <Setter
Property="ContentControl.ContentTemplate"
Value="{StaticResource   MultipleChoice}"
/>
 </MultiDataTrigger.Setters>
 </MultiDataTrigger>
 <MultiDataTrigger>
 <MultiDataTrigger.Conditions>
 <Condition             Binding="{Binding
Path=TestType}"          Value="{x:Static
testGenerator:ETestType.MultipleChoice}"/
>
 <Condition             Binding="{Binding
Path=IsEditMode}" Value="True"/>
 </MultiDataTrigger.Conditions>
 <MultiDataTrigger.Setters>
 <Setter
Property="ContentControl.ContentTemplate"
Value="{StaticResource
EditableMultipleChoice}" />
 </MultiDataTrigger.Setters>
 </MultiDataTrigger>
 <MultiDataTrigger>
 <MultiDataTrigger.Conditions>
 <Condition             Binding="{Binding
Path=TestType}"          Value="{x:Static
testGenerator:ETestType.InputAnswer}"/>
 <Condition             Binding="{Binding
Path=IsEditMode}" Value="False"/>
 </MultiDataTrigger.Conditions>
 <MultiDataTrigger.Setters>
 <Setter
Property="ContentControl.ContentTemplate"
Value="{StaticResource InputAnswer}" />
 </MultiDataTrigger.Setters>
 </MultiDataTrigger>
 <MultiDataTrigger>
 <MultiDataTrigger.Conditions>
 <Condition             Binding="{Binding
Path=TestType}"          Value="{x:Static
testGenerator:ETestType.InputAnswer}"/>
 <Condition             Binding="{Binding
Path=IsEditMode}" Value="True"/>
 </MultiDataTrigger.Conditions>
 <MultiDataTrigger.Setters>
 <Setter
Property="ContentControl.ContentTemplate"
Value="{StaticResource
EditableInputAnswer}" />
 </MultiDataTrigger.Setters>
 </MultiDataTrigger>
 </Style.Triggers>
 </Style>
 </ContentControl.Style>
 </ContentControl>
</UserControl>
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\TestTaskEditor.xaml.cs

```
using System.Windows.Controls;

namespace
AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for
TestTaskEditor.xaml
 /// </summary>
 public  partial  class  TestTaskEditor  :
UserControl
 {
 public TestTaskEditor()
 {
 InitializeComponent();
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\TestTaskInStructure.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserC
ontrols.TestTaskInStructure"

xmlns="http://schemas.microsoft.com/winfx
/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/win
fx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.o
rg/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/exp
ression/blend/2008"
 xmlns:ie="clr-
namespace:System.Windows.Interactivity;as
sembly=System.Windows.Interactivity"
 xmlns:viewModel="clr-
namespace:AIS.TestGenerator.UI.ViewModels
"
 xmlns:userControls="clr-
namespace:AIS.TestGenerator.UI.Views.User
Controls"
 xmlns:controls="clr-
namespace:AIS.TestGenerator.UI.Controls"
 xmlns:local="clr-
namespace:AIS.TestGenerator.UI.Views.User
Controls"
 mc:Ignorable="d"
 d:DesignHeight="450"
d:DesignWidth="800">
 <UserControl.Resources>
 <ResourceDictionary>
 <DataTemplate          DataType="{x:Type
viewModel:TestTaskEditorViewModel}">
 <userControls:TestTaskEditor/>
 </DataTemplate>
 <ResourceDictionary.MergedDictionaries>
 <ResourceDictionary
Source="pack://application:,,,/Views/Styl
e/Converters.xaml" />
 </ResourceDictionary.MergedDictionaries>
 </ResourceDictionary>
 </UserControl.Resources>
 <Grid>
 <Grid x:Name="MainContent">
 <Grid.ColumnDefinitions>
 <ColumnDefinition Width="250"/>
 <ColumnDefinition Width="5"/>
 <ColumnDefinition />
 </Grid.ColumnDefinitions>
 <Grid Grid.Column="0">
 <Grid.RowDefinitions>
 <RowDefinition Height="0.7*"/>
 <RowDefinition Height="5"/>
 <RowDefinition Height="0.3*"/>
 </Grid.RowDefinitions>
 <TreeView                    Grid.Row="0"
ItemsSource="{Binding  DocumentSections}"
x:Name="documentStructure"
IsEnabled="{Binding           IsEditMode,
Converter={StaticResource
reverseBooleanConverter}}">
 <TreeView.ItemContainerStyle>
 <Style              TargetType="{x:Type
TreeViewItem}">
 <Setter            Property="IsExpanded"
Value="True"/>
 </Style>
 </TreeView.ItemContainerStyle>
 <TreeView.ItemTemplate>
 <HierarchicalDataTemplate
ItemsSource="{Binding Sections}">
 <TextBlock Text="{Binding Name}"/>
 </HierarchicalDataTemplate>
 </TreeView.ItemTemplate>

 <ie:Interaction.Triggers>
 <ie:EventTrigger
EventName="SelectedItemChanged">
 <ie:InvokeCommandAction
Command="{Binding ChangeDocumentSection}"
CommandParameter="{Binding
ElementName=documentStructure,
Path=SelectedItem}"/>
 </ie:EventTrigger>
 </ie:Interaction.Triggers>
 </TreeView>
 <GridSplitter               Grid.Row="1"
HorizontalAlignment="Stretch"/>
 <DataGrid                   Grid.Row="2"
ItemsSource="{Binding    SectionKeyterm}"
IsEnabled="{Binding          IsEditMode,
Converter={StaticResource
reverseBooleanConverter}}"
AutoGenerateColumns="False"
IsReadOnly="True"  SelectedItem="{Binding
SelectedKeyterm}">
 <DataGrid.Columns>
 <DataGridTemplateColumn Header="Ключовий
термін" Width="0.6*">
 <DataGridTemplateColumn.CellTemplate>
 <DataTemplate>
 <TextBlock>
 <Hyperlink             Command="{Binding
Path=DataContext.KeytermSelected,
RelativeSource=          {RelativeSource
FindAncestor,        AncestorType={x:Type
DataGrid}}}">
 <TextBlock                Text="{Binding
Text}"></TextBlock>
 </Hyperlink>
 </TextBlock>
 </DataTemplate>
 </DataGridTemplateColumn.CellTemplate>
 </DataGridTemplateColumn>
 <DataGridTextColumn      Header="Оцінка"
Binding="{Binding            Evaluation}"
Width="0.3*">
 <DataGridTextColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTextColumn.HeaderStyle>
 </DataGridTextColumn>
 <DataGridTextColumn       Header="Тести"
Width="0.2*" >
 <DataGridTextColumn.Binding>
 <MultiBinding Converter="{StaticResource
testTasksNumberPerKeyTerm}">
 <Binding ElementName="documentStructure"
Path="SelectedItem"/>
 <Binding />
 </MultiBinding>
 </DataGridTextColumn.Binding>
 <DataGridTextColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTextColumn.HeaderStyle>
 </DataGridTextColumn>
 </DataGrid.Columns>
 <DataGrid.CellStyle>
 <Style TargetType="DataGridCell">
 <Style.Triggers>
```

```
 <Trigger           Property="IsSelected"
Value="True">
 <Setter            Property="Background"
Value="White" />
 <Setter            Property="Foreground"
Value="Black" />
 </Trigger>
 </Style.Triggers>
 </Style>
 </DataGrid.CellStyle>
 </DataGrid>
 </Grid>
 <GridSplitter            Grid.Column="1"
HorizontalAlignment="Stretch" />
 <Grid Grid.Column="2">
 <Grid.RowDefinitions>
 <RowDefinition Height="23"/>
 <RowDefinition Height="0.6*"/>
 <RowDefinition Height="5"/>
 <RowDefinition Height="0.3*"/>
 <RowDefinition Height="5"/>
 <RowDefinition Height="0.2*"/>
 </Grid.RowDefinitions>
 <Grid.ColumnDefinitions>
 <ColumnDefinition/>
 </Grid.ColumnDefinitions>

 <Menu  Grid.Row="0"  IsEnabled="{Binding
IsEditMode,     Converter={StaticResource
reverseBooleanConverter}}">
 <Menu.ItemsPanel>
 <ItemsPanelTemplate>
 <DockPanel
HorizontalAlignment="Stretch"/>
 </ItemsPanelTemplate>
 </Menu.ItemsPanel>
 <MenuItem            Header="Редагувати"
Command="{Binding          EditTestTask}"
Margin="5,0,0,0">
 <MenuItem.Icon>
 <Image      Source="/Resources/Edit.png"
Width="15" Height="15"/>
 </MenuItem.Icon>
 </MenuItem>
 <MenuItem    HorizontalAlignment="Right"
Header="Видалити"       Command="{Binding
DeleteTestTask}">
 <MenuItem.Icon>
 <Image Source="/Resources/basket.png"/>
 </MenuItem.Icon>
 </MenuItem>
 </Menu>
 <DataGrid                   Grid.Row="1"
ItemsSource="{Binding         TestTasks}"
HeadersVisibility="Column"
IsEnabled="{Binding           IsEditMode,
Converter={StaticResource
reverseBooleanConverter}}"
AutoGenerateColumns="False"
IsReadOnly="True"  SelectedItem="{Binding
SelectedTest}" SelectionMode="Single">
 <DataGrid.Columns>
 <DataGridTemplateColumn  Header="Тестове
завдання" Width="*">
 <DataGridTemplateColumn.CellTemplate>
 <DataTemplate>
 <TextBlock     Text="{Binding     Task}"
TextWrapping="Wrap"/>
 </DataTemplate>
 </DataGridTemplateColumn.CellTemplate>
 <DataGridTemplateColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTemplateColumn.HeaderStyle>
 </DataGridTemplateColumn>
 <DataGridTextColumn          Header="Тип
завдання"  Width="100"  Binding="{Binding
TestType}">
 <DataGridTextColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTextColumn.HeaderStyle>
 </DataGridTextColumn>
 </DataGrid.Columns>
 </DataGrid>
 <GridSplitter               Grid.Row="2"
HorizontalAlignment="Stretch"/>
 <Grid Background="#ececec" Grid.Row="3">
 <ContentControl        Content="{Binding
TestEditorViewModel, Mode=OneWay}" />
 </Grid>
 <GridSplitter               Grid.Row="4"
HorizontalAlignment="Stretch"/>
 <controls:BindableRichTextBox
Grid.Row="5"            IsReadOnly="True"
x:Name="richTextBoxName"
Document="{Binding
GeneratedTestDocumentRegion}"/>
 </Grid>
 </Grid>
 </Grid>
</UserControl>
```

Лістинг
AIS.TestGenerator.UI\Views\UserControls\TestTaskInStructure.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace
AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for
TestTaskInStructure.xaml
 /// </summary>
 public partial class TestTaskInStructure
: UserControl
 {
 public TestTaskInStructure()
 {
 InitializeComponent();
 }
 }
}
```

Лістинг
AIS.TestGenerator.UI\Views\UserControls\TestTaskList.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserC
ontrols.TestTaskList"

xmlns="http://schemas.microsoft.com/winfx
/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/win
fx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.o
rg/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/exp
ression/blend/2008"
 xmlns:viewModel="clr-
namespace:AIS.TestGenerator.UI.ViewModels
"
 xmlns:userControls="clr-
namespace:AIS.TestGenerator.UI.Views.User
Controls"
 xmlns:local="clr-
namespace:AIS.TestGenerator.UI.Views.User
Controls"
 mc:Ignorable="d"
 d:DesignHeight="450"
d:DesignWidth="800">
 <UserControl.Resources>
 <ResourceDictionary>
 <DataTemplate           DataType="{x:Type
viewModel:TestTaskEditorViewModel}">
 <userControls:TestTaskEditor/>
 </DataTemplate>
 </ResourceDictionary>
 </UserControl.Resources>
 <Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="0.60*"/>
 <RowDefinition Height="5"/>
 <RowDefinition Height="0.4*"/>
 </Grid.RowDefinitions>
 <DataGrid           ItemsSource="{Binding
TestTasks}"   AutoGenerateColumns="False"
SelectedItem="{Binding SelectedTestTask}"
HeadersVisibility="Column"
IsReadOnly="True">
 <DataGrid.Columns>
 <DataGridTextColumn      Header="Тестові
завдання"   Width="*"   Binding="{Binding
Task}">
 <DataGridTextColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTextColumn.HeaderStyle>
 </DataGridTextColumn>
 </DataGrid.Columns>
 </DataGrid>
 <GridSplitter                Grid.Row="1"
HorizontalAlignment="Stretch"
Height="5"/>
 <Grid Grid.Row="2">
 <Grid.ColumnDefinitions>
 <ColumnDefinition/>
 <ColumnDefinition Width="5"/>
 <ColumnDefinition/>
 </Grid.ColumnDefinitions>

 <Grid  Background="#ececec"  Grid.Row="0"
Grid.Column="0">
 <ContentControl        Content="{Binding
TestEditorViewModel, Mode=OneWay}"/>
 </Grid>
 <GridSplitter            Grid.Column="1"
HorizontalAlignment="Stretch"   Width="5"
/>
 <DataGrid                Grid.Column="2"
IsReadOnly="True"   ItemsSource="{Binding
TestProperties}"
AutoGenerateColumns="False"
HeadersVisibility="Column">
 <DataGrid.Columns>
 <DataGridTemplateColumn
Header="Властивість" Width="100">
 <DataGridTemplateColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTemplateColumn.HeaderStyle>
 <DataGridTemplateColumn.CellTemplate>
 <DataTemplate>
 <TextBlock     Text="{Binding     Key}"
TextWrapping="Wrap"
VerticalAlignment="Center"/>
 </DataTemplate>
 </DataGridTemplateColumn.CellTemplate>
 </DataGridTemplateColumn>
```

```
 <DataGridTemplateColumn
Header="Значення" Width="0.6*">
 <DataGridTemplateColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTemplateColumn.HeaderStyle>
 <DataGridTemplateColumn.CellTemplate>
 <DataTemplate>
 <TextBlock     Text="{Binding     Value}"
TextWrapping="Wrap"
VerticalAlignment="Center"/>
 </DataTemplate>
 </DataGridTemplateColumn.CellTemplate>
 </DataGridTemplateColumn>
 </DataGrid.Columns>
 </DataGrid>
 </Grid>
 </Grid>
</UserControl>
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\TestTaskList.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace
AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for
TestTaskList.xaml
 /// </summary>
 public  partial  class  TestTaskList  :
UserControl
 {
 public TestTaskList()
 {
 InitializeComponent();
 }
 }
}
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\UnknownWordsView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserC
ontrols.UnknownWordsView"

xmlns="http://schemas.microsoft.com/winfx
/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/win
fx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.o
rg/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/exp
ression/blend/2008"
 xmlns:local="clr-
namespace:AIS.TestGenerator.UI.Views.User
Controls"
 mc:Ignorable="d"
 d:DesignHeight="450"
d:DesignWidth="800">
 <UserControl.Resources>
 <ResourceDictionary>
 <ResourceDictionary.MergedDictionaries>
 <ResourceDictionary
Source="pack://application:,,,/Views/Styl
e/ScrollBar.xaml" />
 <ResourceDictionary
Source="pack://application:,,,/Views/Styl
e/Converters.xaml" />
 </ResourceDictionary.MergedDictionaries>
 </ResourceDictionary>
 </UserControl.Resources>
 <Grid>
 <Grid.ColumnDefinitions>
 <ColumnDefinition Width="150"/>
 <ColumnDefinition/>
 </Grid.ColumnDefinitions>
 <DataGrid    AutoGenerateColumns="False"
ItemsSource="{Binding      UnKnownWords}"
IsReadOnly="True" SelectionMode="Single">
 <DataGrid.Columns>
 <DataGridTextColumn       Header="Слово"
Width="*" Binding="{Binding Text}">
 <DataGridTextColumn.HeaderStyle>
 <Style
TargetType="DataGridColumnHeader">
 <Setter
Property="HorizontalContentAlignment"
Value="Center" />
 </Style>
 </DataGridTextColumn.HeaderStyle>
 </DataGridTextColumn>
 </DataGrid.Columns>
 </DataGrid>
 </Grid>
</UserControl>
```

Лістинг AIS.TestGenerator.UI\Views\UserControls\UnknownWordsView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace
AIS.TestGenerator.UI.Views.UserControls
{
 /// <summary>
 ///      Interaction      logic      for
UnknownWordsView.xaml
 /// </summary>
 public partial class UnknownWordsView :
UserControl
 {
 public UnknownWordsView()
 {
 InitializeComponent();
 }
 }
}
```

Проєкт AIS.TestGenerator.UI.GUITests

Лістинг Script\Framework\CommonScripts\AppScripts.js

```
function RunApp(){
 TestedApps.AIS_TestGenerator_UI.Run();
}

function CloseApp(){

TestedApps.AIS_TestGenerator_UI.Terminate
();
}
```

Лістинг Script\Framework\CommonScripts\CommonControlVerification.js

```
let Control = require("Control");

class CommonControlVerification{
 checkIsControlExist(control,
logControlName){
 if(control.exists()){
 Log.Checkpoint(`'${logControlName}'
control exists.`);
 }
 else{
 Log.Error(`'${logControlName}'   control
does not exist.`);
 }
 }

 checkIsControlReadOnly(control,
logControlName){
 if(control.isReadOnly()){
 Log.Checkpoint(`${logControlName}
control is read-only`);
 }
 else{
 Log.Error(`$logControlName}  control  is
not read-only`);
 }
 }

 checkIsControlDisabled(control,
logControlName){
 if(!control.enabled()){
 Log.Checkpoint(`${logControlName}
control is disabled`);
 }
 else{
 Log.Error(`$logControlName}  control  is
enabled`);
 }
 }

 checkIsControlEnabled(control,
logControlName){
 if(control.enabled()){
 Log.Checkpoint(`${logControlName}
control is enabled`);
 }
 else{
 Log.Error(`${logControlName}  control  is
disabled`);
 }
 }
}

module.exports                          =
CommonControlVerification;
```

Лістинг Script\Framework\ControlWrappers\Button.js

```
let Control = require("Control");

class Button extends Control{

 getLabel(){
 return this.control.WPFControlText;
 }
}

module.exports = Button;
```

Лістинг Script\Framework\ControlWrappers\Control.js

```
class Control {

 constructor (mapper) {
 this.control = mapper;
 }

 exists() {
 return this.control.Exists;
 }
```

```
 enabled(){
 return this.control.Enabled;
 }

 visible(){
 return this.control.Visible;
 }

 fullName(){
 return this.control.FullName;
 }

 click() {
 this.control.Click();
 }

 isReadOnly(){
 return this.control.IsReadOnly;
 }

 width(){
 return this.control.Width;
 }

 height(){
 return this.control.Height;
 }
}

module.exports = Control;
```

Лістинг Script\Framework\ControlWrappers\NumberUpDown.js

```
let Control = require("Control");

class NumberUpDown extends Control{

 getValue(){
 return this.control.Value;
 }
}

module.exports = NumberUpDown;
```

Лістинг Script\Framework\ControlWrappers\TextBox.js

```
let Control = require("Control");

class TextBox extends Control {

 setText(text){
 this.control.SetText(text);
 }

 getText(){
 return this.control.Text;
 }
}

module.exports = TextBox;
```

Лістинг Script\Framework\Dialogs\MainWindow.js

```
let Button = require("Button");
let            NumberUpDown            =
require("NumberUpDown");
let TextBox = require("TextBox");

class MainWindow{
 globalPath() {
 return
NameMapping.Sys.Process.WPFObject("HwndSo
urce: MainWindow", "MainWindow");
 }

 isAppeared() {
 return this.globalPath().Exists;
 }

 getPathTextBox(){
 return                                new
TextBox(this.globalPath().FindChild(["Clr
ClassName",            "WPFControlName"],
["TextBox", "filePathTextBox"], 5));
 }

 getSelectButton(){
 return                                new
Button(this.globalPath().FindChild(["ClrC
lassName",  "WPFControlName"],  ["Button",
"browseButton"], 5));
 }

 getBoldCoefficientNumberUpDown(){
 return                                new
NumberUpDown(this.globalPath().FindChild(
["ClrClassName",       "WPFControlName"],
["DoubleUpDown", "boldCoefficient"], 7));
 }

 getItalicCoefficientNumberUpDown(){
 return                                new
NumberUpDown(this.globalPath().FindChild(
["ClrClassName",       "WPFControlName"],
["DoubleUpDown",    "italicCoefficient"],
7));
 }

 getUnderlineCoefficientNumberUpDown(){
 return                                new
NumberUpDown(this.globalPath().FindChild(
["ClrClassName",       "WPFControlName"],
["DoubleUpDown",    "italicCoefficient"],
7));
 }

 getStartButton(){
 return                                new
Button(this.globalPath().FindChild(["ClrC
lassName",  "WPFControlName"],  ["Button",
"startButton"], 5));
 }
}

module.exports = MainWindow;
```

Лістинг Script\Framework\Dialogs\TestGenerationResult.js

```
let Button = require("Button");

class TestGenerationResult{
 globalPath() {
 return
NameMapping.Sys.Process.FindChild("Name",
"*HwndSource: TestGenerationResult*");
 }

 isAppeared() {
 return this.globalPath().Exists;
 }

 getTestEditorButton(){
 return                                new
Button(this.globalPath().FindChild("WPFCo
ntrolName", "testEditor", 6));
 }

 getUnknownWordsButton(){
 return                                new
Button(this.globalPath().FindChild("WPFCo
ntrolName", "unknownWords", 6));
 }

 getTestCoverageButton(){
 return                                new
Button(this.globalPath().FindChild("WPFCo
ntrolName", "testCoverage", 6));
 }

 getGenerationResultButton(){
 return                                new
Button(this.globalPath().FindChild("WPFCo
ntrolName", "generationResult", 6));
 }

 getGiftExportButton(){
 return                                new
Button(this.globalPath().FindChild("WPFCo
ntrolName", "giftExport", 6));
 }
}

module.exports = TestGenerationResult;
```

Лістинг Script\Framework\Dialogs\WindowsOpenFileDialog.js

```
let Button = require("Button");
let            NumberUpDown            =
require("NumberUpDown");
let TextBox = require("TextBox");

class WindowsOpenFileDialog{
 globalPath() {
 return
NameMapping.Sys.Process.Window("*",
"Open", 1);
 }

 isAppeared() {
 return this.globalPath().Exists;
 }

 getFileNameTextBox(){
 return                                new
TextBox(this.globalPath().FindChild(["Wnd
Class", "Index"], ["Edit", 1], 5));
 }

 getOpenButton(){
 return                                new
Button(this.globalPath().FindChild(["WndC
lass",      "WndCaption"],      ["Button",
"&Open"], 5));
 }

 getCancelButton(){
 return                                new
NumberUpDown(this.globalPath().FindChild(
["WndClass",   "WndCaption"],   ["Button",
"Button"], 5));
 }
}

module.exports = WindowsOpenFileDialog;
```

Лістинг Script\Tests\VerifyLeftSidebarInGenerationResultDialog.js

```
let      CommonControlVerification      =
require("CommonControlVerification");
let MainWindow = require("MainWindow");
let        TestGenerationResult        =
require("TestGenerationResult");
let        WindowsOpenFileDialog        =
require("WindowsOpenFileDialog");

const            testFileName            =
"DB_Oraganazation_docx";

function
verifyLeftSidebarInGenerationResultDialog
(){
 clickSelectButton();
 setPathToTestFile();
 clickOpenButton();
 clickStartGenerationButton();
 waitTestGenerationResultDialog();

 const         testEditorButton         =
this.testGenerationResult.getTestEditorBu
tton();

this.commonControlVerification.checkIsCon
trolExist(testEditorButton,
this.logsControlName.testEditor);

this.commonControlVerification.checkIsCon
trolEnabled(testEditorButton,
this.logsControlName.testEditor);

 const        unknownWordsButton        =
this.testGenerationResult.getUnknownWords
Button();
```

```
this.commonControlVerification.checkIsCon
trolExist(unknownWordsButton,
this.logsControlName.testEditor);

this.commonControlVerification.checkIsCon
trolEnabled(unknownWordsButton,
this.logsControlName.testEditor);

 const        testCoverageButton        =
this.testGenerationResult.getTestCoverage
Button();

this.commonControlVerification.checkIsCon
trolExist(testCoverageButton,
this.logsControlName.testEditor);

this.commonControlVerification.checkIsCon
trolEnabled(testCoverageButton,
this.logsControlName.testEditor);

 const      generationResultButton      =
this.testGenerationResult.getGenerationRe
sultButton();

this.commonControlVerification.checkIsCon
trolExist(generationResultButton,
this.logsControlName.testEditor);

this.commonControlVerification.checkIsCon
trolEnabled(generationResultButton,
this.logsControlName.testEditor);

 const         giftExportButton         =
this.testGenerationResult.getGiftExportBu
tton();

this.commonControlVerification.checkIsCon
trolExist(giftExportButton,
this.logsControlName.testEditor);

this.commonControlVerification.checkIsCon
trolEnabled(giftExportButton,
this.logsControlName.testEditor);
}

function setUp(){
 this.commonControlVerification   =   new
CommonControlVerification();
 this.mainWindow = new MainWindow();
 this.testGenerationResult      =      new
TestGenerationResult();
 this.windowsOpenFileDialog     =     new
WindowsOpenFileDialog();

 this.logsControlName = {
 testEditor: 'Редактор тестів',
 browse: 'Невідомі слова',
 bold: 'Покриття тестами',
 italic: 'Результат',
 giftExport: 'Експорт в Gift'
 }
}

function clickSelectButton(){

this.mainWindow.getSelectButton().click()
;
 Log.Checkpoint("Click          'Вибрати'
button");
}

function setPathToTestFile(){
 let            fullFileName            =
Files.FileNameByName(testFileName);

this.windowsOpenFileDialog.getFileNameTex
tBox().setText(fullFileName);
}

function clickOpenButton(){

this.windowsOpenFileDialog.getOpenButton(
).click();
 Log.Checkpoint("Click 'Open' button");
}

function clickStartGenerationButton(){

this.mainWindow.getStartButton().click();
 Log.Checkpoint("Click        'Розпочати'
button");
}

function
waitTestGenerationResultDialog(){
 do
 {
 Delay(1000,   "Wait   test   generation
complite");
 }
 while
(!this.testGenerationResult.isAppeared())
;
}
Лістинг
Script\Tests\VerifyMainWindowsControls.js

let      CommonControlVerification      =
require("CommonControlVerification");
let MainWindow = require("MainWindow");
let        WindowsOpenFileDialog        =
require("WindowsOpenFileDialog");

const            testFileName            =
"DB_Oraganazation_docx";

function verifyMainWindowsControls(){

this.commonControlVerification.checkIsCon
trolExist(this.mainWindow.getPathTextBox(
), this.logsControlName.filePath);

this.commonControlVerification.checkIsCon
trolExist(this.mainWindow.getSelectButton
(), this.logsControlName.browse);

this.commonControlVerification.checkIsCon
trolExist(this.mainWindow.getBoldCoeffici
entNumberUpDown(),
this.logsControlName.bold);

this.commonControlVerification.checkIsCon
trolExist(this.mainWindow.getItalicCoeffi
cientNumberUpDown(),
this.logsControlName.ittalic);

this.commonControlVerification.checkIsCon
trolExist(this.mainWindow.getUnderlineCoe
fficientNumberUpDown(),
this.logsControlName.underline);

this.commonControlVerification.checkIsCon
trolExist(this.mainWindow.getStartButton(
), this.logsControlName.run);

this.commonControlVerification.checkIsCon
trolReadOnly(this.mainWindow.getPathTextB
ox(), this.logsControlName.filePath);

this.commonControlVerification.checkIsCon
trolEnabled(this.mainWindow.getSelectButt
on(), this.logsControlName.browse);

this.commonControlVerification.checkIsCon
trolEnabled(this.mainWindow.getBoldCoeffi
cientNumberUpDown(),
this.logsControlName.bold);

this.commonControlVerification.checkIsCon
trolEnabled(this.mainWindow.getItalicCoef
ficientNumberUpDown(),
this.logsControlName.ittalic);

this.commonControlVerification.checkIsCon
trolEnabled(this.mainWindow.getUnderlineC
oefficientNumberUpDown(),
this.logsControlName.underline);

this.commonControlVerification.checkIsCon
trolDisabled(this.mainWindow.getStartButt
on(), this.logsControlName.run);
 clickSelectButton();
 checkIsOpenDialogAppear();
 setPathToTestFile();
 clickOpenButton();

this.commonControlVerification.checkIsCon
trolEnabled(this.mainWindow.getStartButto
n(), this.logsControlName.run);
}

function setUp(){
 this.commonControlVerification   =   new
CommonControlVerification();
 this.mainWindow = new MainWindow();
 this.windowsOpenFileDialog     =     new
WindowsOpenFileDialog();

 this.logsControlName = {
 filePath: 'File path',
 browse: 'Browse',
 bold: 'Bold',
 ittalic: 'Italic',
 underline: 'Underline',
 run: 'Run'
 }
}

function clickSelectButton(){

this.mainWindow.getSelectButton().click()
;
 Log.Checkpoint("Click          'Вибрати'
button");
}

function checkIsOpenDialogAppear(){

if(this.windowsOpenFileDialog.isAppeared(
)){
 Log.Checkpoint("Open     dialog     has
appeared");
 }
 else{
 Log.Error("Open    dialog    has    not
appeared");
 }
}

function setPathToTestFile(){
 let            fullFileName            =
Files.FileNameByName(testFileName);

this.windowsOpenFileDialog.getFileNameTex
tBox().setText(fullFileName);
}

function clickOpenButton(){

this.windowsOpenFileDialog.getOpenButton(
).click();
 Log.Checkpoint("Click 'Open' button");
}
```

Додаток Д

Ксерокопії наукових публікацій, виконаних при роботі над дипломною роботою магістра

1.Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В Ковальчук, В. О. Слободзян // Вісник Хмельницького національного університету. Хмельницький – 2018. – С.61-69.

2.Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В Ковальчук, В. О. Слободзян, Г. А. Білоус. // Інтелектуальний потенціал – 2018 : збірник наукових праць молодих науковців і студентів, Хмельницький – 2018. – с. 51-56.

3.Ковальчук О. В. Дослідження практичної ефективності інформаційної технології автоматизованого визначення семантичних термінів навчальних матеріалів / О. В. Ковальчук, О. В. Мазурець. // Сучасні технології в механіці : зб. наук. пр., Хмельницький – 2018 – с. 141-148.

4.Ковальчук О.В. Використання програмного розширення Spire.Doc для автоматизації роботи з цифровими документами / О. В. Ковальчук, Г. А Білоус., В. О. Слободзян // Збірник наукових праць за матеріалами XI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2019». Хмельницький – 2019. – с. 116-122.

5.Ковальчук О. В. Метод генерації тестових завдань до навчальних матеріалів на основі продукційних правил / О. В Ковальчук, О. В. Мазурець, // Збірник наукових праць за матеріалами XII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2020». Хмельницький – 2020. – с. 51-56.

ISSN 2307-5732

НАУКОВИЙ ЖУРНАЛ **1.2018**

ВІСНИК

Хмельницького національного університету

Технічні науки
Technical sciences

SCIENTIFIC JOURNAL
HERALD OF KHMELNYTSKYI NATIONAL UNIVERSITY

2018, Issue 1, Volume 257

Хмельницький

ВІСНИК
ХМЕЛЬНИЦЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
серія: Технічні науки

Затверджений як фахове видання (перереєстрація)
Наказ МОН 04.07.2014 №793

Засновано в липні 1997 р. ***Виходить 6 разів на рік***

Хмельницький, 2018, № 1 (257)

Засновник і видавець: Хмельницький національний університет
(до 2005 р. – Технологічний університет Поділля, м. Хмельницький)

Включено до наукометричних баз:

Google Scholar	http://scholar.google.com.ua/citations?hl=uk&user=aIUP9OYAAAAJ
Index Copernicus	http://jml2012.indexcopernicus.com/passport.php?id=4538&id_lang=3
РИНЦ	http://elibrary.ru/title_about.asp?id=37650
Polish Scholarly Bibliography	https://pbn.nauka.gov.pl/journals/46221

Технічний редактор Горященко К. Л., к.т.н.
Редактор-коректор Броженко В. О.

Рекомендовано до друку рішенням вченої ради Хмельницького національного університету, протокол № 10 від 30.01.2018 р.

Адреса редакції: редакція журналу "Вісник Хмельницького національного університету"
Хмельницький національний університет
вул. Інститутська, 11, м. Хмельницький, Україна, 29016

☎ (038-2) 62-51-08 **web:** http://journals.khnu.km.ua/vestnik
e-mail: visnyk.khnu@gmail.com http://vestnik.ho.com.ua
http://lib.khnu.km.ua/visnyk_tup.htm

Зареєстровано Міністерством України у справах преси та інформації.
Свідоцтво про державну реєстрацію друкованого засобу масової інформації
Серія КВ № 9722 від 29 березня 2005 року

ЗМІСТ

РАДІОТЕХНІКА, ЕЛЕКТРОНІКА ТА ТЕЛЕКОМУНІКАЦІЇ

ТЕХНОЛОГІЇ ЛЕГКОЇ ПРОМИСЛОВОСТІ

МАШИНОЗНАВСТВО ТА ОБРОБКА МАТЕРІАЛІВ В МАШИНОБУДУВАННІ

УДК 004

О.В. МАЗУРЕЦЬ, О.В. КОВАЛЬЧУК, В.О. СЛОБОДЗЯН
Хмельницький національний університет

ВИКОРИСТАННЯ СПЕЦІАЛІЗОВАНИХ ПРОГРАМНИХ РОЗШИРЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З ЦИФРОВИМИ ДОКУМЕНТАМИ НАВЧАЛЬНИХ МАТЕРІАЛІВ

В статті досліджено проблему автоматизації роботи з електронними документами навчальних матеріалів у задачах визначення структури змістовних блоків у електронному документі навчального матеріалу та пошуку ключових термінів у контенті навчального матеріалу. Встановлено, що для реалізації програмної обробки електронних документів є доцільним використання спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентам відповідних файлів. З метою визначення найбільш ефективного програмного розширення проведено аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки електронних документів: Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll. За результатами аналізу встановлено, що бібліотека Microsoft.Office.Interap.Word.dll надає доступ до всіх функцій MS Office, так як працює безпосередньо з PIA, для її роботи потрібна ліцензія для MS Office на кожному клієнтському комп'ютері, MS Office завантажується у фоновому режимі, внаслідок чого займає значний обсяг оперативної пам'яті. При роботі з docx-файлом за допомогою DocumentFormat.OpenXml.dll не потрібна наявність MS Office та запуск в фоновому режимі, проте зростає складність програмного використання бібліотеки внаслідок необхідності регулярного співставлення ідентифікатора стиля з контейнером властивостей стилів. Перевагою Spire.Doc.dll визначено реалізацію функцій автоматичного співставлення стилів текстових блоків їх властивостям на рівні розширення. В результаті аналізу розглянутих бібліотек було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки електронних документів. Встановлено, що перенесення функції автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з електронним документом, так і процес програмування. Розглянуто практичні особливості використання розширення Spire.Doc.dll при роботі з електронними документами .DOCX. З метою визначення можливості застосування обраного інструменту в задачах роботи з електронними документами було проведено його практичну апробацію в складі програмної системи для вирішення прикладної задачі побудови структури електронного документу та пошуку ключових слів у його контенті. В результаті встановлено можливість та достатню ефективність використання розширення Spire.Doc.dll для роботи з електронними документами навчальних матеріалів.

Ключові слова: електронний документ, цифровий документ, навчальні матеріали, ключові терміни, Microsoft.Office.Interap.Word, DocumentFormat.OpenXml, Spire.Doc.

O.V. MAZURETS, O.V. KOVALCHYK, V.O. SLOBODZIAN
Khmelnytsky National University

USING SPECIALIZED SOFTWARE PACKEGES FOR AUTOMATION OF WORK WITH DIGITAL DOCUMENTS OF EDUCATIONAL MATERIALS

At article investigated a problem of the automation work with electronic documents of educational materials in the tasks for determining the structure of content blocks at electronic documents educational material and search of key terms in content of educational materials. It is established that for software processing of electronic documents need to use specialized program components that provide the necessary tools for work with content of files. Microsoft.Office.Interop.Word.dll, DocumentFormat.OpenXml.dll and Spire.Doc.dll were using for choose the most effective program components for automation work with electronic documents. In process audit established that Microsoft.Office.Interap.Word.dll provide access to all function of MS Office because it works with PIA. This library needs a license for MS Office at every computers of client and MS Office loaded in the background, in result it occupies many RAM. DocumentFormat.OpenXml.dll uses for work with docx file. This component does not need to install MS Office and work in the background but the complexity of use this library is increasing to the need to regularly match the style ID with the styles properties container. The advantage of Spire Doc.dll is the implementation of automatic functions for matching of style of text blocks to their properties. In result audit this libraries there was chosen Spire.Doc.dll for use to automation of processing of electronic documents. It is established that in result transfer of functions for automatic matching styles of text block to their properties from the level of the functional code of the application to the level of the functional library simplify work of the application with electronic document and programming process. There was considered practical features of Spire.Doc.dll for working with electric documents ".DOCX". There was practical use made this tool in program system for determine its the possibilities at using in the tasks for work with electronic documents. The system is designed to build the structure of an electronic document and search keywords in its content. Explored the possibility and effectiveness of using the Spire.Doc.dll library to work with electronic educational materials documents.

Keyword: digital document, electronic document, key terms, educational materials, Microsoft.Office.Interap.Word, DocumentFormat.OpenXml, Spire.Doc.

Постановка проблеми в загальному вигляді

Автоматизація вирішення ряду задач у галузі сучасної вищої освіти (оцінка відповідності навчальних матеріалів вимогам, оцінка відповідності наборів тестових завдань навчальним матеріалам, автоматизована генерація прототипів тестових завдань, реалізація гнучких алгоритмів тестування, допомога та контроль якості при формуванні навчальних матеріалів, допомога та контроль якості при формуванні тестів до навчальних матеріалів, автоматизація формування рефератів та анотацій до елементів навчальних матеріалів тощо) забезпечується через застосування онтології як методу формального опису знань, що містяться в навчальних матеріалах. Модель онтології навчального матеріалу може складатися з ключових

слів, ключових термінів, структури навчального матеріалу, атрибутів ключових слів та ключових термінів, що визначають їх властивості та забезпечують прив'язку до елементів структури навчального матеріалу [1]. За такої моделі, онтологія навчального матеріалу є методом виявлення сенсу навчального матеріалу та засобом вирішення наведеного ряду практичних задач.

Основними з етапів побудови онтології навчального матеріалу є пошук ключових термінів у контенті навчального матеріалу та побудова його логічної структури. Вхідними даними є електронний документ навчального матеріалу. Для автоматизації виконання обох наведених етапів потрібна програмна обробка відповідних цифрових файлів (зазвичай формату .DOCX). Для ефективної реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають об'єктно-орієнтований інструментарій для програмної роботи з контентом відповідних файлів.

Аналіз останніх досліджень

Проблему автоматизації побудови логічної структури навчального матеріалу (наприклад: Дисципліна / Розділ / Тема) пропонується вирішувати шляхом визначення ієрархії змістовних блоків у цифровому документі [1], таким чином формуючи верхній рівень вертикальної онтології відповідної навчальної дисципліни. Проблему пошуку ключових термінів у контенті навчального матеріалу пропонується вирішувати шляхом використання відповідної інформаційної технології на основі дисперсійної оцінки [2], таким чином формуючи нижній рівень онтології навчальної дисципліни.

Визначено, що онтологія навчального матеріалу розглянутої моделі може бути методом виявлення сенсу навчального матеріалу й ефективно використана в розробці інформаційних технологій для роботи з навчальними матеріалами [1]. Зокрема, з застосуванням такої моделі онтології навчального матеріалу було розв'язано ряд практичних задач: оцінка відповідності навчальних матеріалів вимогам [3], оцінка відповідності наборів тестових завдань навчальним матеріалам [4], автоматизована генерація прототипів тестових завдань [5], реалізація гнучких алгоритмів тестування [6], автоматизація формування рефератів та анотацій до елементів навчальних матеріалів [7] та інші.

Для реалізації програмної обробки цифрових документів є доцільним використання розглянутих в даній статті спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентом відповідних файлів, зокрема: Microsoft.Office.Interap.Word.dll [8], DocumentFormat.OpenXml.dll [9], Spire.Doc.dll [10].

Постановка задачі

Метою статті є аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів і визначення найбільш ефективного програмного розширення. Методом визначення можливості застосування обраного інструменту в задачах роботи з цифровими документами є його практична апробація в складі програмної системи для вирішення відповідних прикладних задач.

Викладення основних матеріалів дослідження

На сучасному етапі для зберігання електронних документів навчальних матеріалів використовується файл з розширенням .DOCX (або створених на його основі .HTML, .PDF тощо). На відміну від файлів .DOC, які зберігають дані документа в одному бінарному файлі, файли .DOCX створюються за допомогою відкритого формату XML, в якому зберігаються документи як колекції окремих файлів і папок в стиснутому пакеті. .DOCX-файли містять .XML-файли і три папки, docProps, Word, і _rels (Рис. 1а), які містять властивості документа, його зміст (Рис. 1б) і відношення між файлами, тему (Рис. 1в) та включені файли (Рис. 1г). .DOCX-файли розроблені, щоб зробити вміст документів доступним і відкритим – так, текстовий документ чи зображення зберігаються як окремі прості файли в такій колекції у складі одного файлу .DOCX.

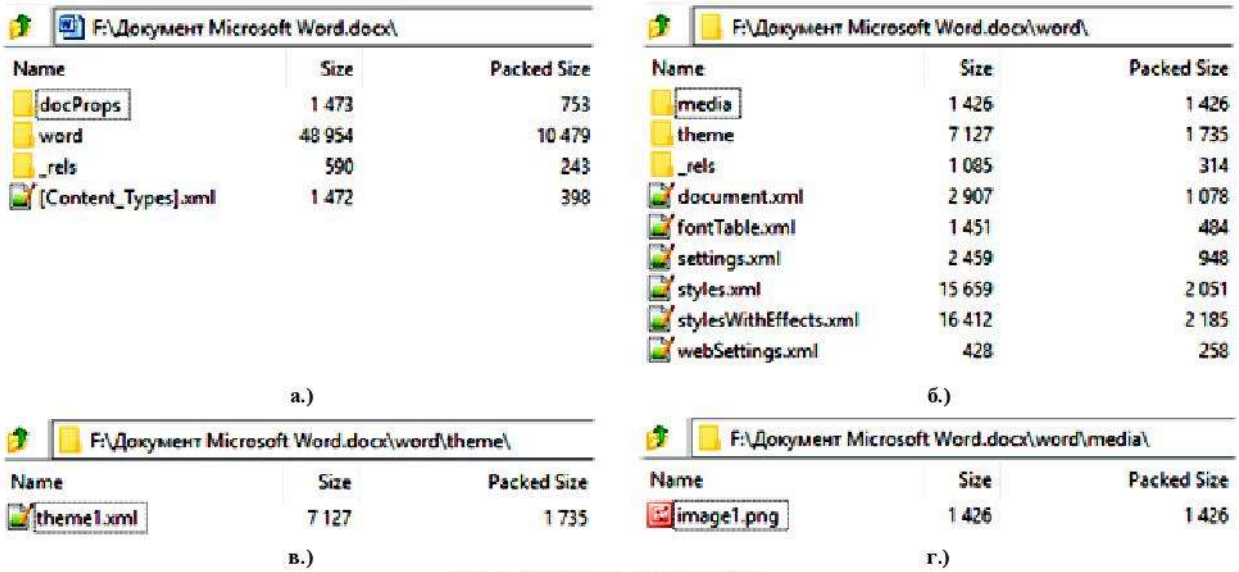


Рис. 1. Структура .DOCX-файлу

Office OpenXML є форматом електронних офісних документів, таблиць, презентацій та діаграм, розроблений компанією Microsoft. Даний формат є міжнародним стандартом, який затверджений в Міжнародній організації зі стандартизації (ISO) [11]. Він включає в себе такі первинні мови розмітки:

- WordprocessingML – для обробки текстових документів;
- SpeadsheetXL – електронних таблиць;
- PresentationML – для презентацій;
- DrawingML – для векторної графіки, діаграм.

Хоча Word реалізує свою функціональність через об'єктну модель, збереження даних відбувається у наборі XML-файлів, що ускладнює можливості для автоматизації прямого програмного парсингу [12]. Оскільки файл стилів та текст знаходяться в різних файлах у складі .XML, потрібно зчитувати одразу ряд файлів та на базі їх взаємозв'язків програмно формувати об'єктну модель документа, що є не найкращим рішенням. Тому для реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають об'єктно-орієнтований інструментарій для програмної роботи з контентом відповідних файлів. Найбільш відомими з таких спеціалізованих програмних комплексів є розширення Microsoft.Office.Interap.Word.dll [8], DocumentFormat.OpenXml.dll [9] та Spire.Doc.dll [10].

Microsoft.Office.Interap.Word.dll

Бібліотека Microsoft.Office.Interap.Word.dll [8] дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM.

За допомогою процесів автоматизації Office Automation програми, написані мовами, такими як Visual C# .NET, отримують можливість програмно керувати іншими програмами. Автоматизація Microsoft Word (далі Word) дозволяє виконувати відповідні дії, такі як створення нового документу, додавання даних у документ або створення таблиць. З програмами Word та іншими програмами Microsoft Office практично всі дії, які користувач може виконувати вручну за допомогою користувацького інтерфейсу, також можуть бути виконані програмним шляхом за допомогою автоматизації Office Automation.

Word реалізує цю програмну функціональність через об'єктну модель [13]. Об'єктна модель є набором класів і методів, які служать аналогами логічних компонентів Word. Наприклад, існує об'єкт Application, об'єкт Document та об'єкт Paragraph, кожен з яких містить функціональність цих частин Word (рис. 2).

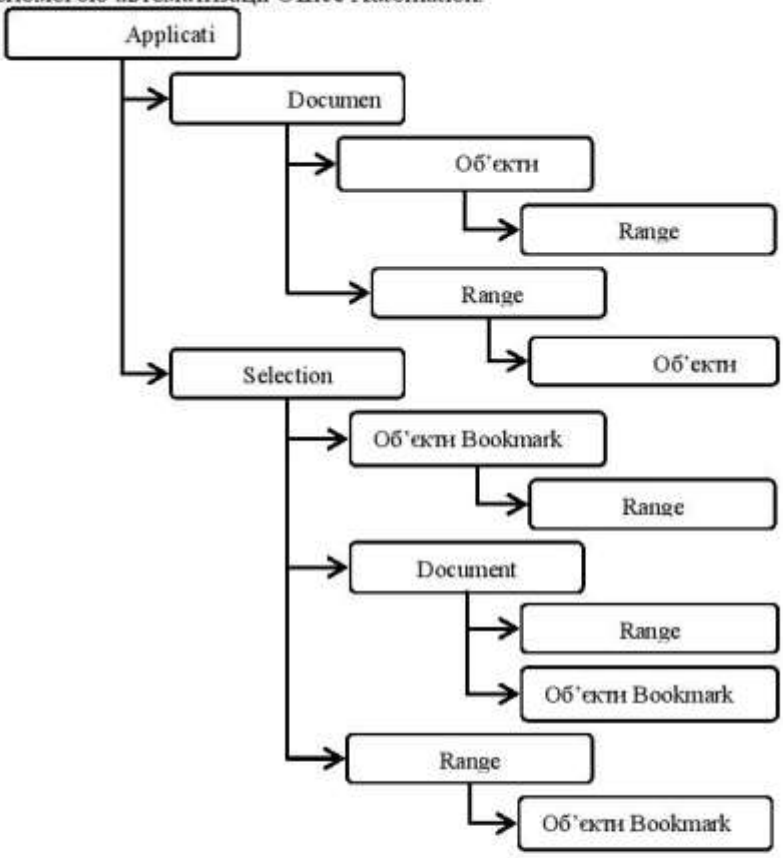


Рис. 2. Об'єктна модель Word

Щоб використовувати функції програми MS Office в проекті, можна використовувати первинний взаємозв'язок PIA (Primary Interop Assembly). PIA дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM. При створенні нового проекту Office, Visual Studio додає посилання на PIA, необхідний для побудови проекту. При цьому, у деяких сценаріях доводиться додавати посилання на додаткові PIA (наприклад, якщо необхідно використовувати функцію MS Office Word в проекті для MS Office Excel).

Для роботи з інтерфейсом Microsoft.Office.Interop.Word.dll необхідно встановити відповідну збірку, для чого в меню Tools потрібно обрати пункт NuGet Package Manager та запустити Package Manager Console та ввести наступну команду:

```
PM> Install-Package Microsoft.Office.Interop.Word
```

Після того як менеджер пакетів повідомить про успішне завершення встановлення, можна починати працювати з необхідними інтерфейсами.

DocumentFormat.OpenXml.dll

Бібліотека класів .NET DocumentFormat.OpenXml.dll (версії 2.0 / 2.5 / 2.8.1) [9] дає можливість розробникам програмного забезпечення працювати з пакетом Microsoft Office. Компанія надає дану бібліотеку безкоштовно в повному доступі, й її можна завантажити з офіційного сайту Microsoft або

засобами Visual Studio.

Для роботи з файлами формату .DOCX, який має зображену на рисунку 1 структуру, використовується мова розмітки WordprocessingML [14]. На рисунку 3 зображено приклад структури цієї мови розмітки.

```
<?xml version="1.0" encoding="utf-8"?>
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
<w:body>
<w:p>
<w:r>
<w:t> Текст </w:t>
</w:r>
```

Рис. 3. Приклад WordprocessingML-розмітки .DOCX-документу

Теги, наведені на рис. 3, мають такі властивості:

- *document* – є основою частиною документа
- *body* – контейнер для збору структур рівня блоку, які складають основну історію;
- *p* – абзац (paragraph);
- *r* – контейнер який містить в собі текст з однаковими властивостями (run);
- *t* – власне текстовий елемент (text).

Всі інші теги, які можуть використовуватися, є допоміжними до вищенаведених і додають їм стильових властивостей.

Spire.Doc.dll

Spire.Doc.dll [10] для .NET є повністю незалежною бібліотекою класів .NET Word, спеціально створеною для розробників, й дозволяє швидко генерувати, відкривати, редагувати та зберігати документи Word різних версій. Конвертація функціональних можливостей дозволяє розробникам здійснювати перетворення Word в інші актуальні формати документів (.PDF, .EPub, .HTML, .RTF, .Image, .XML тощо). Можна створювати та обробляти вже існуючі документи Word динамічно.

Бібліотека Spire.Doc.dll забезпечує роботу з майже всіма елементами документа Word, а саме: сторінки, розділи, заголовки, колонтитули, абзаци, переліки, таблиці, текст, виноски, поля, гіперпосилання, закладки, коментарі, зображення, стилі, фонові параметри, функції друку, налаштування документа та захисту. Крім того, підтримуються графічні об'єкти, включаючи форми, текстові поля, зображення, OLE-об'єкти та елементи керування. Бібліотека Spire.Doc.dll підтримує функцію пошуку та заміни, вирівнювання, перерву сторінки, поле заповнення, об'єднання документів, копіювання документів, друк тощо.

Для роботи з бібліотекою Spire.Doc необхідно встановити відповідну збірку, для чого потрібно запустити Package Manager Console та виконати наступну команду:

```
PM> Install-Package Spire.Doc
```

Після завершення встановлення можна починати працювати з бібліотекою.

Дискусія

Хоч бібліотека Microsoft.Office.Interap.Word.dll надає доступ до всіх функцій MS Office, так як працює безпосередньо з PIA, для її роботи потрібна ліцензія для MS Office на кожному клієнтському комп'ютері. Крім того, при використанні Automation, MS Office завантажується у фоновому режимі, внаслідок чого займає деякий обсяг оперативної пам'яті та завантажує велику кількість файлів і DLL. Додатки MS Office були розроблені як додатки для користувацького інтерфейсу, і через це Microsoft.Office.Interop.Word.dll працює повільно. Microsoft не рекомендує використовувати Office Automation (або будь-який Office Interop) на сервері.

На відміну від Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll не вимагають наявності MS Office на машині користувача та його запуску в фоновому режимі.

При роботі з docx-файлом за допомогою DocumentFormat.OpenXml.dll потрібно дотримуватися тієї ж ієрархії, що і сама структура розмітки тобто: *document-paragraph-run-text*. Також відповідно до розглянутих вище особливостей взаємозв'язків всередині документу, при присвоєнні параграфу деякого стилю, у властивостях параграфа надається лише ідентифікатор (ID) на даний стиль, натомість сам стиль описується окремо у файлі style.xml. Внаслідок необхідності регулярного співставлення ID стиля з контейнером style.xml для одержання характеристик стилів абзаців, зростає складність програмного використання бібліотеки.

Spire.Doc.dll не вимагає встановлення в систему кожного користувача MS Office, тобто є можливість повністю незалежної від нього роботи. Повна версія Spire.Doc.dll є платною, а безкоштовна версія FreeSpire.Doc має ряд обмежень (наприклад обробка не більше 500 абзаців і 25 таблиць) [15]. Перевагою Spire.Doc.dll визначено відсутність необхідності співставлення ID стиля з контейнером style.xml, оскільки дана функція реалізована на рівні бібліотеки.

Таким чином, в результаті аналізу сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів, було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки цифрових документів. Основними перевагами Spire.Doc.dll встановлено відсутність необхідності наявності MS Office на машині користувача та запуску в фоновому режимі, а також реалізацію функцій автоматичного співставлення стилів текстових блоків їх властивостям на рівні розширення.

Перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з цифровим документом, так і процес програмування.

Програмна реалізація

При зчитуванні файлу Spire.Doc.dll перетворює документ в об'єктну модель. Ця модель має структуру, яка починається документом і закінчується об'єктом *TextRange* (рис. 4). Приклад використання такої об'єктної моделі документу MS Office показаний на рис. 5.

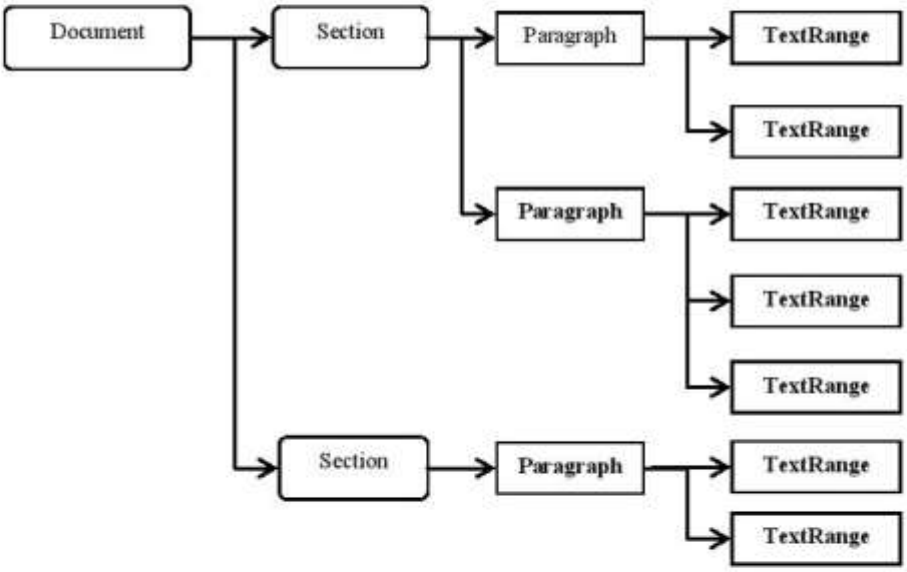


Рис. 4. Загальна структура об'єктної моделі документу

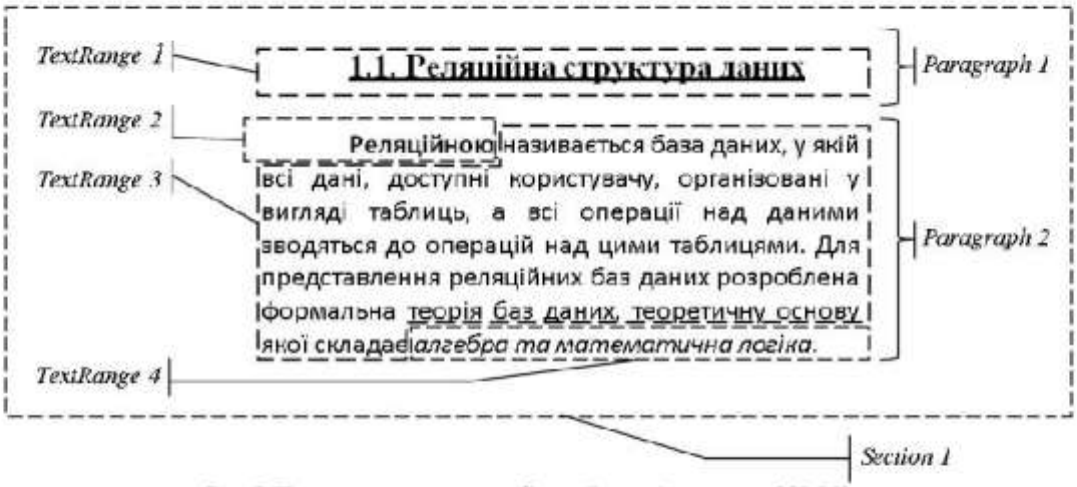


Рис. 5. Приклад використання об'єктної моделі документу MS Office

Відповідно до об'єктної моделі документу, MS Office використовує розділи (*Section*), щоб вказати частини документа, що мають різну орієнтацію сторінки, стовпці або заголовки та нижні колонтитули. Розбиття на Section дозволяє користувачеві вказати місце, де розпочинається і закінчується інше форматування. Тому Section використовуються коли потрібне використання заголовків, колонтитулів, схем, нумерації сторінок, розмірів аркушу, поля чи різні рівні захисту. Об'єкти Section містяться в об'єкті Document (рис. 5), в колекції Selections. А розділи, в свою чергу, містять в собі наступний елемент структури – абзаци (*Paragraph*).

Paragraph в MS Word визначає будь-який текст, який закінчується жорстким поверненням, яке формується, наприклад, при натисканні клавіші Enter. Формат абзацу дозволяє контролювати зовнішній вигляд фрагменту, якщо є окремі абзаци. Наприклад, можна змінити відстань між рядками або вирівнювання тексту по лівому краю на вирівнювання по центру. Можна додати індивідуальні стилі,

відступи для абзаців, або визначити заголовки і списки. В об'єктній моделі *Paragraph* знаходиться в *Document -> Sections -> Section -> Paragraphs* (рис. 6). Одержати відомості про стилі *Paragraph* можна за допомогою методу *GetStyle()* (рис. 7).

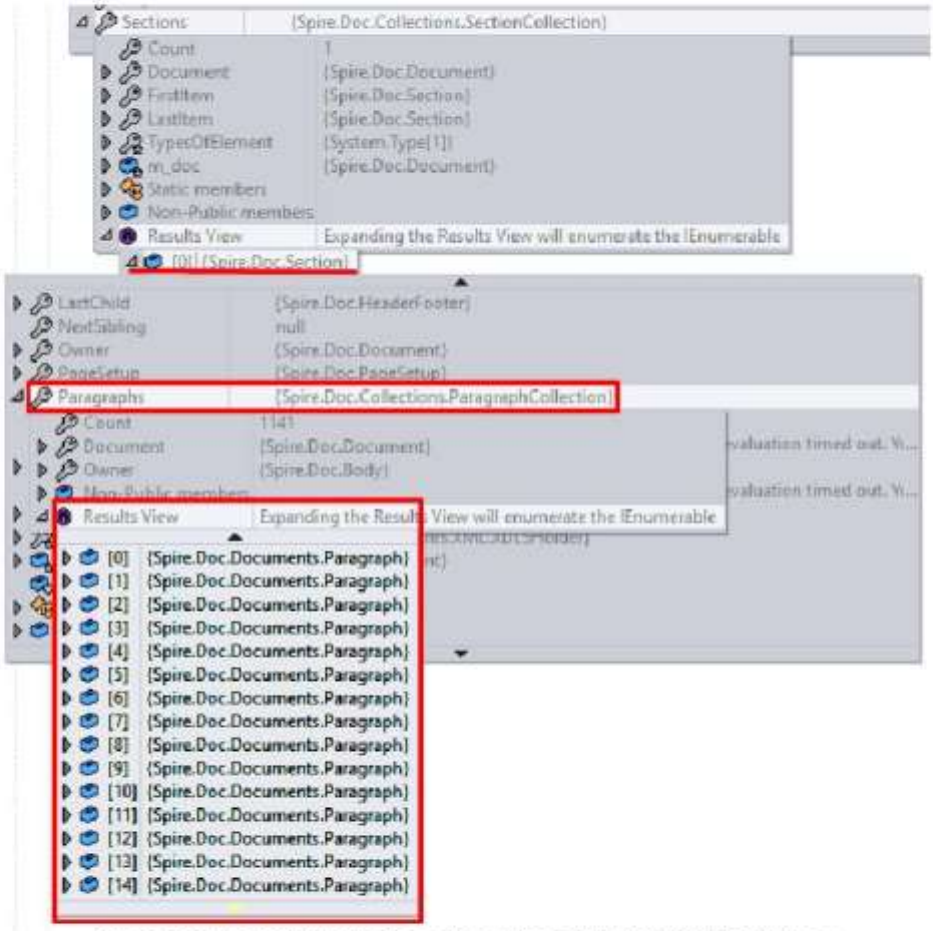


Рис. 6. Позиція визначеного об'єкта Paragraph в об'єктній моделі документа

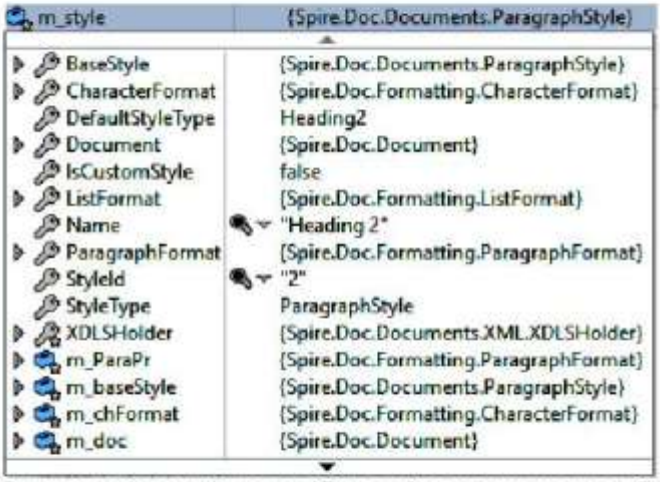


Рис. 7. Одержання відомостей про стилі Paragraph за допомогою методу GetStyle()

TextRange є найнижчим рівнем структури документу, що визначає фрагмент тексту однакового стилю. Тому для того, щоб при розв'язку прикладних задач отримати окремо кожне слово з власними властивостями стилів, потрібна розробка додаткового алгоритму для розбиття цих фрагментів *TextRange* на слова. *TextRange* знаходиться в *Paragraph -> Items*. Через роботу з відповідними властивостями, можна одержати текст (рис. 8) та стилі (рис. 9), що відносяться до визначеного *TextRange*.

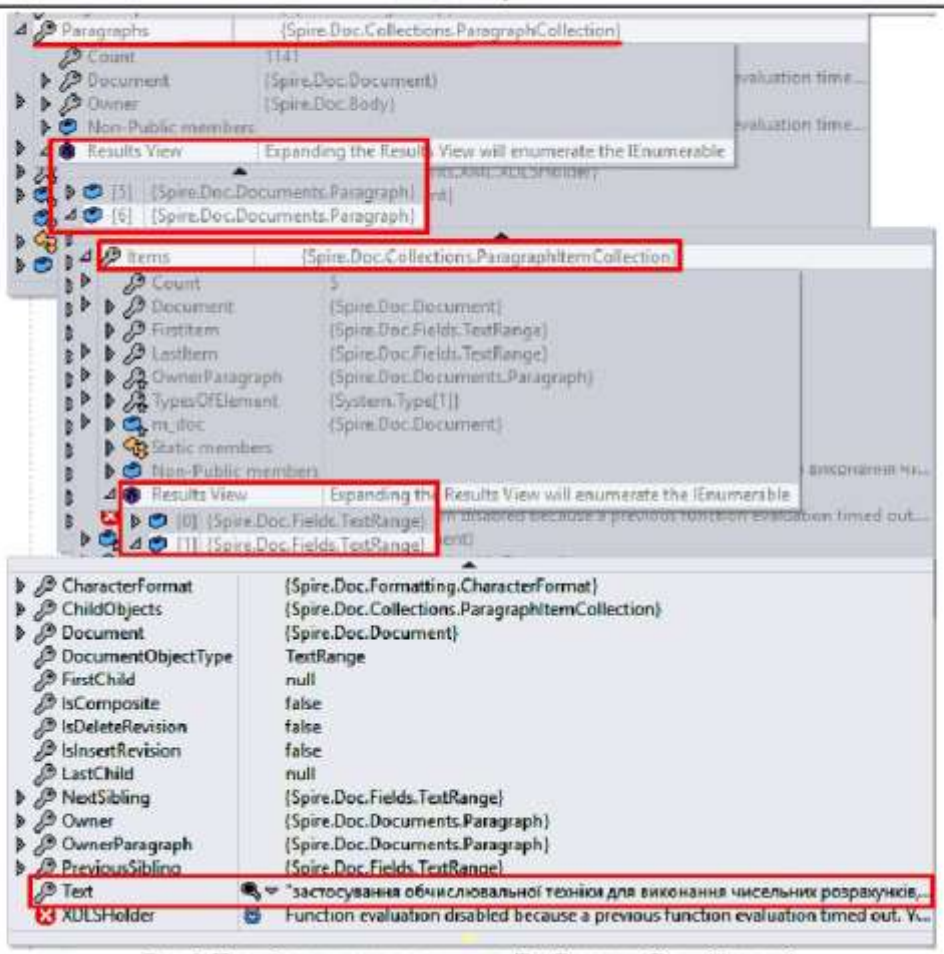


Рис. 8. Позиція визначеного тексту в TextRange у об'єктній моделі

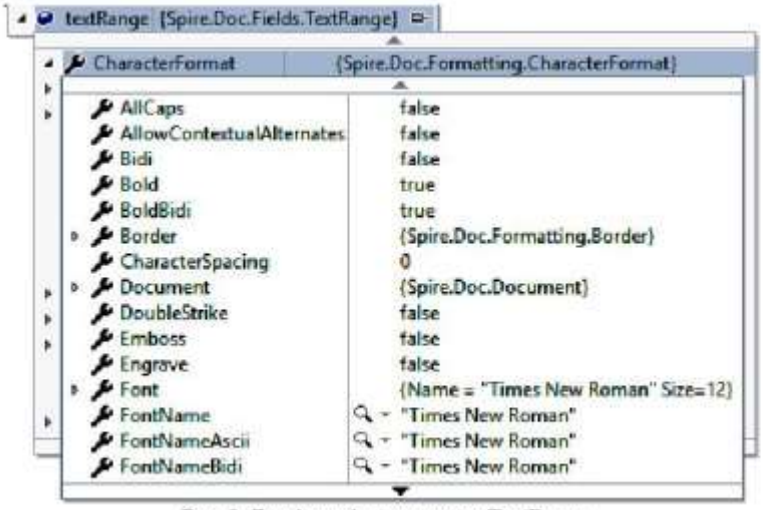


Рис. 9. Деякі стилі визначеного TextRange

Paragraph може містити лише один з кількох визначених форматів: *Level1, Level2, Level3, Level4, Level5, Level6, Level7, Level8, Level9, Body*. Відповідно до цього значення, можна визначити заголовки та їх рівні. Кожен похідний елемент має посилання на батьківський, і таким чином існує можливість визначити прив'язку певного елементу до певного модулю, заголовку чи підзаголовку при визначенні структури змістовних блоків у електронному документі навчального матеріалу.

З метою визначення можливості застосування розширення Spire.Doc.dll в задачах роботи з електронними документами, було проведено його практичну апробацію в рамках розробки програмної системи для вирішення прикладної задачі побудови структури електронного документу та пошуку ключових слів у його контенті. Зокрема, на рисунку 10 показано приклад обробки файлу *«Організація баз даних і знань.docx»* з автоматичним визначенням структури навчального матеріалу за допомогою аналізу системи заголовків та переліків ключових слів для контенту кожного елементу структури.

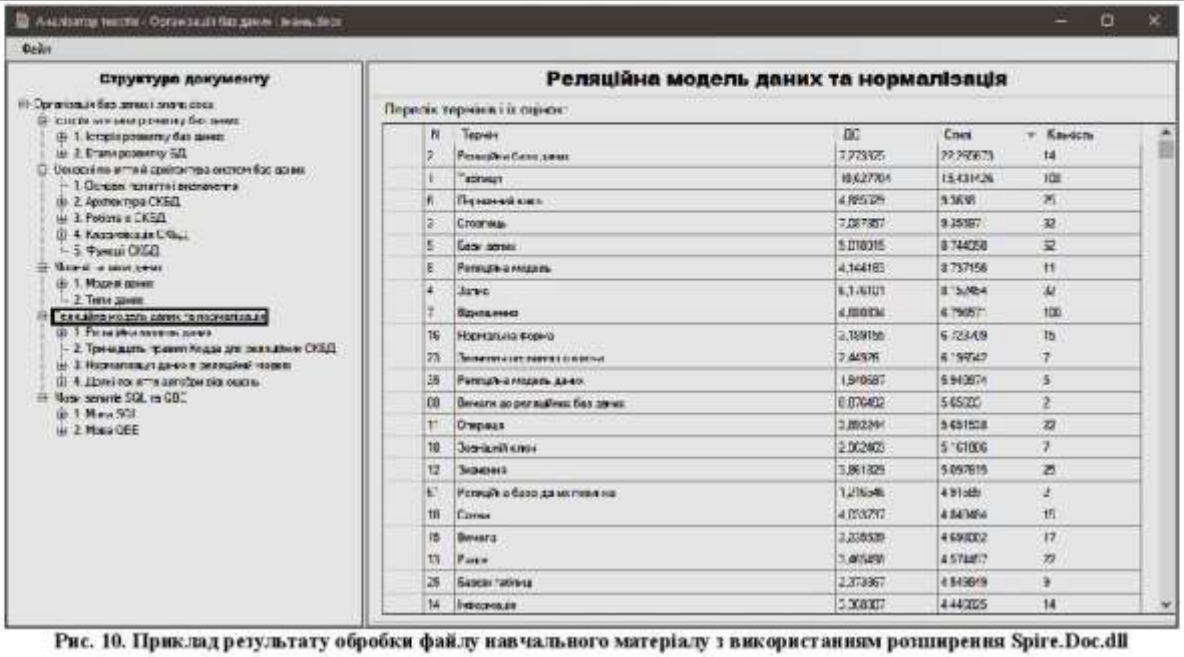


Рис. 10. Приклад результату обробки файлу навчального матеріалу з використанням розширення Spire.Doc.dll

Маючи стилі форматування *Paragraph* та *TextRange*, можна визначити заголовки, рівні заголовків, текст для аналізу чи ігнорування, прив'язку певного фрагменту тексту до заголовку.

При цьому, визначення властивостей *Paragraph* надало можливість для формування структури документу, а обробка властивостей елементів *TextRange* визначила необхідний для програмної обробки контент, що дозволило визначити множини ключових слів за допомогою методу дисперсійного оцінювання.

За результатами використання розширення Spire.Doc.dll в розробленому програмному комплексі встановлено, що дана бібліотека надає достатній інструментарій для роботи з цифровими документами навчальних матеріалів у рамках розглянутих актуальних задач, й може бути ефективно використана в реалізації інформаційних технологій для автоматизації роботи з навчальними матеріалами.

Висновки

Отже, у статті було досліджено проблему автоматизації роботи з цифровими документами навчальних матеріалів у задачах визначення структури змістовних блоків у цифровому документі навчального матеріалу та пошуку ключових термінів у контенті навчального матеріалу. Встановлено, що для реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентом відповідних файлів.

З метою визначення найбільш ефективного програмного розширення було проведено аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів: Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll. В процесі аналізу встановлено, що бібліотека Spire.Doc.dll є оптимальним варіантом для використання при автоматизації обробки цифрових документів. Встановлено, що перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з цифровим документом, так і процес програмування.

Розглянуто практичні особливості використання розширення Spire.Doc.dll при роботі з цифровими документами .DOCX. З метою визначення можливості застосування обраного інструменту в задачах роботи з цифровими документами було проведено його практичну апробацію в складі програмної системи для вирішення прикладної задачі побудови структури цифрового документу та пошуку ключових слів у його контенті. В результаті встановлено можливість та достатню ефективність використання розширення Spire.Doc.dll для роботи з цифровими документами навчальних матеріалів.

Подальші дослідження спрямовані на впровадження розширення Spire.Doc.dll в розробку автоматизованих систем роботи з цифровими документами навчальних матеріалів для вирішення прикладних задач.

Література

1. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів / О. В. Мазурець // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2017. – № 6. – С. 223–229.

2. Бармак О. В. Інформаційна технологія автоматизованого визначення термінів у навчальних матеріалах / О. В. Бармак, О. В. Мазурець // Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах». – Хмельницький, 2015. – № 2. – С. 94–102.

3. Коренчук О. В. Система інтелектуального аналізу відповідності лекційних матеріалів навчальних

дисциплін стандартам освіти / О. В. Коренчук, О. В. Мазурець // Збірник наукових праць за матеріалами восьмої міжнародної науково-технічної конференції «Актуальні проблеми комп'ютерних технологій 2014». – Хмельницький, 2014. – С. 191–201.

4. Поліщук А. О. Інформаційна технологія автоматизації аналізу відповідності тестових завдань лекційним матеріалам навчальних дисциплін / А. О. Поліщук, О. В. Мазурець // Збірник наукових праць за матеріалами дев'ятої міжнародної науково-технічної конференції «Актуальні проблеми комп'ютерних технологій 2015». – Хмельницький, 2015. – С. 207–218.

5. Бармак О. В. Інформаційна технологія автоматизованого формування тестових завдань / О. В. Бармак, О. В. Мазурець, В. І. Кліменко // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2017. – № 5. – С. 93–103.

6. Бармак О. В. Застосування інформаційної технології гнучкого тестування рівня знань у середовищі Moodle / О. В. Бармак, О. В. Мазурець, А. О. Матвійчук // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2017. – № 3. – С. 103–115.

7. Бармак О. В. Інформаційна технологія автоматизованого анотування та реферування цифрових текстів / О. В. Бармак, О. В. Мазурець, А. В. Живілік // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2017. – № 4. – С. 147–158.

8. Considerations for server-side Automation of Office [Електронний ресурс]. – Режим доступу : https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office/

9. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу : https://www.iso.org/search/x/query/Open%2520XML

10. Spire.Doc for .NET [Електронний ресурс]. – Режим доступу : https://www.nuget.org/packages/Spire.Doc/

11. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу : https://www.iso.org/search/x/query/Open%2520XML

12. Office Open XML [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Office_Open_XML

13. How to automate Microsoft Excel from Microsoft Visual C#.NET [Електронний ресурс]. – Режим доступу : https://support.microsoft.com/en-us/help/302084/how-to-automate-microsoft-excel-from-microsoft-visual-c--net

14. DocX for creates or modifies Microsoft Word files [Електронний ресурс]. – Режим доступу : https://github.com/xceedsoftware/DocX

15. Free Spire.Doc for .NET [Електронний ресурс]. – Режим доступу : https://www.e-iceblue.com/Introduce/free-doc-component.html

References

1. Mazurets O. V. Ontologichiy pidhid do pobudovi semantichnoi modely navchalnih materialiv / O. V. Mazurets // Naukovyy zhurnal „Herald of Khmelnytskyi National University" seriya: Tekhnichni nauky. Khmelnytsky, 2017, № 6. – S. 223-229.

2. Barmak O. V., Mazurets O. V. Informatsiyna tekhnolohiya avtomatyzovanoho vyznachennya terminiv u navchalnykh materialakh / O. V. Barmak, O. V. Mazurets // Mizhnarodnyy naukovo-tekhnichnyy zhurnal «Vymiryuvalna ta obchyslyuvalna tekhnika v tekhnolohichnykh protsesakh» – Khmelnytsky, 2015. – №2. – C.94–102.

3. Korenchuk O. V., Mazurets O. V. Systema intelektualnoho analizu vidpovidnosti lektsiynykh materialiv navchalnykh dystsyplin standartam osvity / O. V. Korenchuk, O. V. Mazurets // Zbirnyk naukovykh prats za materialamy vosmoyi mizhnarodnoyi naukovo-tekhnichnoyi konferentsiyi «Aktualni problemy kompyuternykh tekhnolohiy 2014». Khmelnytsky – 2014. – S.191-201.

4. Polishchuk A. O., Mazurets O. V. Informatsiyna tekhnolohiya avtomatyzatsiyi analizu vidpovidnosti testovykh zavdan lektsiynym materialam navchalnykh dystsyplin / A. O. Polishchuk, O. V. Mazurets // Zbirnyk naukovykh prats za materialamy devyatoyi mizhnarodnoyi naukovo-tekhnichnoyi konferentsiyi «Aktualni problemy kompyuternykh tekhnolohiy 2015». Khmelnytsky – 2015. – S.207-218.

5. Barmak O. V., Mazurets O. V., Klimenko V. I. Informatsiyna tekhnolohiya avtomatyzovanoho formuvannya testovykh zavdan / O. V. Barmak, O. V. Mazurets, V. I. Klimenko // Naukovyy zhurnal „Herald of Khmelnytskyi National University" seriya: Tekhnichni nauky. Khmelnytsky, 2017, № 5. – S.93-103.

6. Barmak O. V., Mazurets O. V., Matviychuk A. O. Zastosuvannya informatsiynoyi tekhnolohiyi hnuchkoho testuvannya rivnya znan u seredovyshchi Moodle / O. V. Barmak, O. V. Mazurets, A. O. Matviychuk // Naukovyy zhurnal „Herald of Khmelnytskyi National University" seriya: Tekhnichni nauky. Khmelnytsky, 2017, №3. – S.103-115.

7. Barmak O. V., Mazurets O. V., Zhyvilik A. V. Informatsiyna tekhnolohiya avtomatyzovanoho anotuvannya ta referuvannya tsyfrovykh tekstiv / O. V. Barmak, O. V. Mazurets, A. V. Zhyvilik // Naukovyy zhurnal „Herald of Khmelnytskyi National University" seriya: Tekhnichni nauky. Khmelnytsky, 2017, № 4. – S. 147-158.

8. Considerations for server-side Automation of Office URL: https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office/

9. International Organization for Standardization - Open XML file formatsb. URL: https://www.iso.org/search/x/query/Open%2520XML

10. Spire.Doc for .NET URL: https://www.nuget.org/packages/Spire.Doc/

11. International Organization for Standardization - Open XML file formatsb. URL: https://www.iso.org/search/x/query/Open%2520XML

12. Office Open XML. URL: https://uk.wikipedia.org/wiki/Office_Open_XML

13. How to automate Microsoft Excel from Microsoft Visual C#.NET. URL: https://support.microsoft.com/en-us/help/302084/how-to-automate-microsoft-excel-from-microsoft-visual-c--net

14. DocX for creates or modifies Microsoft Word files. URL: https://github.com/xceedsoftware/DocX

15. Free Spire.Doc for .NET. URL: https://www.e-iceblue.com/Introduce/free-doc-component.html

Рецензія/Peer review : 19.01.2018 р. Надрукована/Printed :28.01.2018 р.
Рецензент: стаття рецензована редакційною колегією

НПК МНІС ІП-2018

ЗБІРНИК НАУКОВИХ ПРАЦЬ МОЛОДИХ НАУКОВЦІВ І СТУДЕНТІВ

ЧАСТИНА 1

ПРИСВЯЧУЄТЬСЯ 30-РІЧЧЮ ПІДГОТОВКИ ІТ-ФАХІВЦІВ В ХМЕЛЬНИЦЬКОМУ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Хмельницький національний університет

Військовий інститут Київського національного університету ім.Тараса Шевченка

ПВНЗ "Університет економіки і підприємництва"

Тернопільський інститут агропромислового виробництва

Інтелектуальний потенціал - 2018

збірник наукових праць молодих науковців і студентів

Присвячується 30-річчю підготовки ІТ- фахівців в Хмельницькому національному університеті

сформовано за матеріалами
Всеукраїнської науково-практичної конференції
молодих науковців і студентів «Інтелектуальний потенціал – 2018»

14-16 листопада 2018р.

Частина 1
Комп'ютерні науки та інформаційні технології проектування

Хмельницький
2018

ББК 74.480.278
С.88

«Інтелектуальний потенціал – 2018» - збірник наукових праць молодих науковців і студентів з нагоди 30-річчя підготовки ІТ- фахівців в ХНУ/Колектив авторів – Хмельницький: ПВНЗ УЕП, 2018. – Ч.1: Комп'ютерні науки та інформаційні технології проектування. – 132 с.

ЗМІСТ

подається приклад X, про класову приналежність якого нічого не відомо. Класифікатор повинен дати відповідь, до якої класу відноситься вектор X.

Для побудови поділяючої гіперплощини використовується радіальна базисна функція:

$$k(x,x') = \exp(-\gamma\|x-x'\|^2) \text{ при } \gamma > 0,$$

де x – поточний вектор, x' – центральний вектор, γ – параметр, що нормалізує.

Попередня обробка допомагає зменшити вплив шумовий складової, а також усунути невеликі дефекти.

Отже, в роботі представлений автоматизований алгоритм класифікації на основі модифікованого оператора локальних бінарних околиць. Для класифікації дескрипторів і поділу на класи використаний метод опорних векторів. Приклади, представлені в роботі, демонструють ефективність алгоритму при класифікації на складно текстурних зображеннях.

Література

1. Lin Jin, Liu Yayu. Potholes Detection Based on SVM in the Pavement Distress Image, Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, pp. 544 – 547, 2010.

2. Методы компьютерной обработки изображений./Под редакцией В.А. Сойфера – М.: Физматлит, 2003.-784с.

3. Barni M., Bartolini F., Cappellini V. Image processing for virtual restoration of artworks. IEEE Multimedia, vol. 7, no. 2, pp. 34-37, 2000.

4. Zhou J., Huang P., and Chiang F. Wavelet-based pavement distress classification. TRR: Journal of the Transportation Research Board, pp. 89–98, 2005.

Метод формального опису елементів моделей автоматизованого формування тестових завдань

Мазурець О.В., Ковальчук О.В., Слободзян В.О., Білоус Г.А.
Хмельницький національний університет

Одним із основних способів контролю знань в навчальних інформаційних системах є комп'ютерне тестування, яке крім контрольної функції застосовується для навчання, тренінгу, розвитку здібностей. Інформаційні технології дають можливість суттєво зменшити трудові затрати на створення тестових завдань з можливістю їх постійного оновлення, що формує актуальний напрямок наукових досліджень.

Інформаційна технологія автоматизованого формування тестових завдань на основі контенту навчальних матеріалів [1] дозволяє на основі вхідних даних у вигляді контенту файлу формату .docx навчальних матеріалів або його визначеної частини автоматизовано формувати вихідні

дані у вигляді файлу з тестами для імпорту у віртуальне навчальне середовище. Розв'язок задачі автоматизації формування тестових завдань містить ряд послідовних етапів перетворення інформації, до яких відносяться визначення семантичних термінів у контенті навчальних матеріалів [2], формування структури навчальних матеріалів [3], безпосередньо автоматизоване формування тестових завдань [1] й перенесення результатів у область взаємодії з кінцевим користувачем – підсистему тестування середовища Moodle [4].

Характерною рисою автоматизованого формування тестових завдань за даною інформаційною технологією є використання набору моделей тестових завдань для перетворення фрагментів контенту у тестові завдання.

Модель тестового завдання являє собою структуру, що складається з наступних функціональних елементів:

– маска для фрагменту тексту з поняттям, що призначена для ідентифікації фрагментів контенту із заданим терміном, до яких воно може бути застосовано;

– параметри моделі тестового завдання, що визначають особливості й ефективність його застосування: тип запитання, що може бути створено за цією моделлю; базова оцінка очікуваної якості застосування моделі у діапазоні (0;1), що одержується методом експертної оцінки результатів; набір правил корекції базової оцінки цього правила конвертації, що в залежності від вказаних особливостей фрагменту можуть знизити базову оцінку, при активації виступаючи множниками у діапазоні (0;1);

– маска для формування тестового завдання, що містить алгоритм перетворення даного фрагменту у елементи тестового завдання.

Тобто модель у даному випадку представляє собою структуру, в яку входять: набір з'єднувачів (слів або символів які вирізняють певний фрагмент тексту як релевантний відносно терміну); правило конвертації фрагментів з тексту у тестове завдання, та дані про словоформу (відмінок/рід). На вхід моделі завжди дається певна зв'язка [термін – слово маркер – теза], за якою здійснюється формування тестового завдання та підбір варіантів відповідей.

При використанні даного підходу спершу встановлюються вимоги до набору тестових завдань, визначаються актуальні моделі тестових завдань й на основі найбільш прийнятних із них формується набір тестових завдань. В результаті за кожною з обраних моделей тестових завдань формується одне тестове запитання у форматі .gift, що може бути конвертоване у середовище Moodle. Кінцевий результат зберігається в одному файлі тесту.

Таким чином, при формуванні тестових завдань різних типів використовуються моделі, наділені параметрами та критеріями, характерними лише для конкретного типу питання та будови терміну. В межах розглянутої інформаційної технології, з метою формування моделей

тестових завдань є необхідною розробка набору тегів для формального опису елементів правил конвертації.

Метою роботи є розробка множини тегів як методу формального опису елементів моделей автоматизованого формування тестових завдань. Це дозволить проводити як зручне й ефективне створення нових чи редагування існуючих моделей формування тестових завдань, так і компактне зберігання розроблених моделей для подальшого використання.

Відповідно до розглянутих складових моделей формування тестових завдань, виділяються теги для ідентифікації елементів контенту та теги для формування тестів. Теги для ідентифікації елементів контенту використовуються в масках ідентифікації й наведені в таблиці 1.

Таблиця 1 – Теги для ідентифікації елементів контенту

Тег	Опис
[TermGroup]	Фрагмент тексту, який являє собою термін, що складається з набору слів
[ThesisGroup]	Фрагмент тексту (теза), який дає визначення терміну
[RandomTermGroup]	Випадково обраний інший термін
[RandomThesisGroup]	Випадково обране визначення іншого терміну
[Connector]	Слово або символ із тексту, що поєднує термін з тезою (– , – це, є, називається, тощо)
[BeginSentence]	Фрагмент тексту від початку речення до TermGroup або ThesisGroup (може бути null)
[Inflexion]	Тег повертає нормальну форму наступного елементу
[Caps]	Переведення першої букви до верхнього регістру
[ReCaps]	Переведення першої букви до нижнього регістру

Нижче наведений приклад для ідентифікації двох моделей з істинним твердженням, базової (Basic) та реверсивної (Reversed):

```
[Caps][TermGroup][connector][ThesisGroup]. {TRUE}
[Caps][ThesisGroup][connector][TermGroup]. {TRUE}
```

Теги для формування тестів використовуються в масках формування й наведені у таблиці 2.

Таблиця 2 – Теги для формування тестів

Тег	Опис
{Header}	Візуальний визначник для тексту питання
{Answer}	Візуальний визначник для варіанту відповіді
{FALSE}	Вказівка на неправильний варіант відповіді
{TRUE}	Вказівка на правильний варіант відповіді

Нижче зображений приклад моделі формування тестового завдання одиничного вибору, де в якості тексту питання (Header) вказана теза, а у

варіантах відповідей (Answer 1 – N) назви термінів:

```
Header -> [Caps][ThesisGroup]["-, - це"]
Answer 1 -> [TermGroup]                          {TRUE}
Answer 2 -> [RandomTermGroup]    {FALSE}
Answer 3 -> [RandomTermGroup]    {FALSE}
Answer N -> [RandomTermGroup]    {FALSE}
```

Наведені теги можна віднести до елементів псевдокоду, або змінних, які під час роботи алгоритму мають різні значення, а іноді можуть не мають жодних (null). У випадках, коли тег може повернути порожнє значення, він вказується з додаванням знака дорівнює і нуля, наступним чином: [Tag = 0].

Наведений підхід дозволяє відкрите програмування алгоритму роботи різноманітних моделей генерування тестових завдань, що визначає множину формалізованих таким чином моделей як базу знань відповідної інформаційної системи для автоматизованого формування тестових завдань за навчальними матеріалами.

Для прикладу наведено процес формування тестового завдання типу «із введенням тексту» з використанням однієї з моделей формування тестових завдань для відповідного типу завдань.

Моделі формування завдань із введенням тексту полягають у формулюванні одного твердження з відсутнім ключовим словом та формуванні можливих варіантів відповідей, які не пропонуються користувачеві, а використовуються лише для перевірки введеного тексту. Відповідно, можлива модель, мета якої – забезпечити максимальну кількість правильних відповідей для коректної перевірки тексту, введеного користувачем.

При формуванні маски ідентифікації, за основу твердження береться певна пара [термін – теза], причому важливо, щоб термін був у початковій формі (називному відмінку однини), щоб виключити можливість ігнорування правильних відповідей через несхожість у закінченнях або словоформах. Далі відбувається формування набору правильних відповідей. Необхідним і достатнім є хоча б один варіант відповіді, якщо він повністю покриває можливі форми вживання даного терміну. Множина варіантів відповідей формуються на підставі параметрів моделі.

При формуванні правильних відповідей необхідно враховувати: абревіатуру терміну; скорочення; слова іншомовного походження (якщо є у тексті); відмінок/рід терміну.

Оскільки варіанти відповідей завжди сховані, функції цієї моделі не включають у себе підбір хибних термінів, а навпаки – модель має гарантувати, що всі можливі форми вживання цього поняття будуть розпізнані як правильна відповідь.

Отже, для сформованого на рисунку 1 тестового питання типу «Коротка відповідь» було використано термін «Вимірювання» та відповідну модель формування тестових завдань із відкритою відповіддю. Термін взятий з фрагменту тексту: «Вимірювання – це множина однотипних даних, що

утворюють одну з граней гіперкуба» з теми «Моделі даних» навчального матеріалу «Організація баз даних та знань».

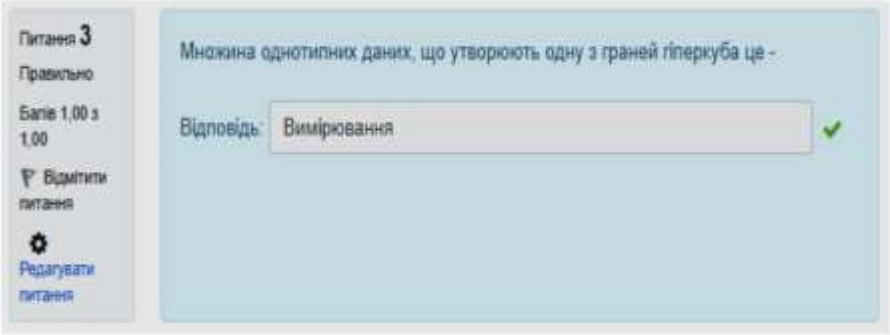


Рисунок 1 – Приклад використання у Moodle сформованого тестового завдання типу «Коротка відповідь»

Необхідно відмітити, що правильними відповідями є обидві з наведених, а для повної відповіді на питання достатньо вказати одну з них.

В даному випадку актуальний фрагмент тексту був знайдений наступною маскою ідентифікації:

```
[BeginSentence][ReCaps][TermGroup][Connector][ThesisGroup].
```

А генерація тестового завдання за актуалізованою моделлю відбулася шляхом використання відповідної маски формування тестового завдання із таким кодом:

```
Header -> [Caps][_____][connector][ThesisGroup]:
Answer 1 -> [Abbreviation] {TRUE}
Answer 2 -> [TermGroup1] {TRUE}
Answer N -> [TermGroupN] {TRUE}
```

Відповідний сформований фрагмент GIFT-файлу, який сформований за даною моделлю формування тестового завдання типу «із введенням тексту» із відкритою відповіддю, наступний:

```
// question
Множина однотипних даних, що утворюють одну з граней
гіперкуба це -
{
=%100% Вимірювання#
=%100% Dimension#
}
```

Отже, розроблені теги для моделей формування тестових завдань дозволяють формально описати процес формування тестових завдань із рахуванням всіх особливостей та параметрів, й забезпечити автоматизацію імпорту доступних тестових завдань у середовищі Moodle. Тег формального

опису моделей є елементом псевдокоду, який призначений для формального опису структури моделі, її вхідних та вихідних параметрів.

За умови достатньої відповідності семантичним та структурним вимогам і коректного співвідношення між обсягом контенту навчального матеріалу та параметрів генерації набору тестових завдань, одержується репрезентативний тест, що може бути використаний як безпосередньо для тестування, так і як сировина для подальшої роботи розробника тестів.

Література

1. Бармак О. В., Мазурець О. В., Кліменко В. І. Інформаційна технологія автоматизованого формування тестових завдань / О. В. Бармак, О. В. Мазурець, В. І. Кліменко // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2017, №5. – С.93-103.

2. Мазурець О. В. Інформаційна технологія автоматизованого визначення семантичних термінів в елементах навчальних матеріалів / О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №3. – С.223-230

3. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів / О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2017, №6. – С.223-229.

4. Moodle – Open-source learning platform. – [Електронний ресурс]. – Режим доступу: https://moodle.org/

Система підбору методом спільної фільтрації продукції інтернет-магазину літератури

Мартинюк Б.І., Лищук О.А.
Хмельницький національний університет

З розвитком економічних відносин у суспільстві, виникла потреба швидко і ефективно збувати продукцію. А в людей, що купують товари виникла потреба максимально раціонально використовувати власний час. Саме тому інтенсивний розвиток веб-технологій зумів задовільнити всі сторони даних суспільних відносин. Зокрема почали з'являтися інтернет-магазини, які з кожним роком стають більш функціональними і традиційні магазини не здатні з ними конкурувати. Адже в електронній комерції є такі функції, які не здатні використовувати традиційний метод торгівлі. А саме йдеться про потужний функціонал швидкого підбору товару, де немає людського фактору, що значно пришвидшує пошук необхідного товару для

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

“СУЧАСНІ ТЕХНОЛОГІЇ В МЕХАНІЦІ”

ЗБІРНИК НАУКОВИХ ПРАЦЬ

19-21 КВІТНЯ 2018Р.
м. ХМЕЛЬНИЦЬКИЙ

ЗМІСТ

СЕКЦІЯ 1 • SECTION 1

СЕКЦІЯ 2 • SECTION 2

СЕКЦІЯ 3 • SECTION 3

5. A.Marzo, S.A.Seah, B.W.Drinkwater, D.R.Sahoo, B.Long, S.Subramanian. Holographic acoustic elements for manipulation of levitated objects, *Nature Communications* 6, Article number: 8661, 2015.
6. D. Karpul, J. Tapson, M. Rapson, A. Jongens, G. Cohen, Limiting factors in acoustic separation of carbon particles in air, J. Acoust. Soc. Am. 127 (2010) 2153–2158.

УДК 004.4

ПРОГРАМНИЙ КОМПЛЕКС WEB-ВІДОБРАЖЕННЯ CAD-МОДЕЛЕЙ СИСТЕМИ DYNAMO

Дитинюк В.О., Скрипник Т.К.
Хмельницький Національний Університет, Україна

Сьогодні людство дійшло до етапу свого розвитку коли неможливо назвати таку галузь, в якій би не знайшла примінення комп'ютеризація та автоматизація процесів. Ця тенденція значно зменшує відсоток рутинної роботи, натомість дає змогу сконцентруватися на творчій діяльності, що в свою чергу оптимізує результати окремого людського індивіду і цілих областей і сфер в цілому. Cloud-сервіси забезпечують зручний мережевий доступ до загального пула обчислювальних ресурсів з урахуванням мінімальних потреб в апаратному забезпеченні на клієнтській стороні. Одним з перспективних напрямків, де успішно застосовуються хмарні технології це напрямок проектування різноманітних об'єктів (від деталей чіпа мікрокомп'ютера до грандіозних хмарочосів). Для цих цілей використовуються технології автоматизованого проектування (computer - aided design - CAD) і автоматизованої розробки або конструювання (computer aided engineering - CAE). Ці системи дозволяють створювати креслення або моделі реальних об'єктів як на площині так і просторі, а також проводити з цими моделями різноманітні розрахунки.

Візуальна система Dynamo є одним з прикладів таких технологій, які забезпечують хмарні обрахунки використовуючи інтерфейс "вузлів", який може бути зручним як для прогрмістів, інженерів так і для звичайних користувачів. Dynamo використовує бібліотеки CAD-системи Revit а отже підтримує виконання скриптів на кількох мовах програмування.

Метою даної роботи було об'єднати функціонал програмного комплексу Dynamo з підходом віддаленого хмарного виконання обрахунків. Для цього було імплементовано на мові програмування C#

додаток Reach який є проміжною ланкою між клієнтським додатком та функціоналом ядра Dynamo. Reach відповідає за коректну обробку запитів, виклик відповідних методів для формування сцени та рендеру геометричних примітивів, компонування данних та відправку інформації про геометрію на клієнтський додаток через WebSocket з'єднання. Клієнтський додаток імплементовано на мові JavaScript на фреймворку Backbone.js за використанням бібліотеки Three.js, яка забезпечує рендеринг отриманої з Reach геометрії в браузері за допомогою WebGL. Інтерфейс додатку максимально наближений до інтерфейу самого Dynamo, з можливістю вибора нодів, запуску кастомних скриптів та іншого фукціонала.

В роботі проведено грунтовний аналіз предметної області і програмного забезпечення предметної області, що дозволило чітко встановити мету та сформулювати задачі дослідження.

В роботі використано сучасний підхід до розробки програмного забезпечення – об'єктно-орієнтований аналіз і проектування та сучасні case-засоби створення UML–моделі розроблюваної системи.

В результаті аналізу сучасних засобів програмування для реалізації комплексу було використано мову програмування C#/Javascript. В якості інтегрованого середовища для написання програми використано IDE Microsoft .NET Framework 4.5. Для створення UML моделі використано case-систему RationalRose.

Проведено аналіз та проектування системи, здійснено реалізацію основних проектних модулів системи, розроблено інструкцію з користування системою.

УДК 004.9

ДОСЛІДЖЕННЯ ПРАКТИЧНОЇ ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ СЕМАНТИЧНИХ ТЕРМІНІВ НАВЧАЛЬНИХ МАТЕРІАЛІВ

Ковальчук О.В., Мазурець О.В.
Хмельницький національний університет, Україна
E-mail: losha.kovalchyk1998@gmail.com

Базовим засобом реалізації дистанційної освіти є інформаційні технології. Це визначає необхідність формалізації та стандартизації навчального процесу. Загальноприйнятим є підхід застосування навчальних матеріалів у вигляді цифрових документів визначеної

структури як інструменту навчання. Для роботи із курсами навчальних дисциплін використовуються спеціалізовані віртуальні навчаючі середовища. При їх використанні, потенційна якість отриманих освітніх послуг безпосередньо визначається якістю навчальних матеріалів курсу. В умовах вузької спеціалізації курсів дисциплін, їх значної численності та інтенсивного оновлення, перспективним шляхом оцінки якості навчальних курсів та їх елементів є автоматизація вирішення відповідного ряду задач у сучасній вищій освіті.

З семантичної точки зору, базовою властивістю контенту є його семантика, яку формалізовано відображають у вигляді мережі, вузлами якої є терміни, що несуть семантичне навантаження, а дуги відображають характер зв'язку між вузлами [1]. Відтак, аналіз термінів, що використовуються у навчальних матеріалах, дозволяє побудувати семантичну модель навчального курсу й вирішити ряд похідних задач.

Метою роботи є висвітлення загальних аспектів інформаційної технології автоматизованого визначення множин ключових семантичних термінів у електронних документах навчальних матеріалів й дослідження її ефективності.

Семантика навчального матеріалу виражається його логічною структурою (наприклад: Дисципліна / Розділ / Тема) та поняттями, що розглядаються в ньому. Множини ключових термінів кожного елементу ієрархії змістовних блоків навчального матеріалу можуть мати довільну кількість елементів й у сукупності формують загальну множину ключових термінів навчального матеріалу. За такої моделі, онтологія навчального матеріалу може бути методом виявлення сенсу навчального матеріалу. Пошук множин ключових семантичних термінів у навчальних матеріалах необхідний для всіх елементів ієрархії змістовних блоків, тому інформаційна технологія має використовуватись не тільки для електронного документу загалом, а я для його елементів.

Загальну схему інформаційної технології автоматизованого визначення множини ключових семантичних термінів у електронних документах навчальних матеріалів відображено на рисунку 1.

Рис. 1 – Загальна схема інформаційної технології автоматизованого множини ключових семантичних термінів

В результаті аналізу системи заголовків навчальних матеріалів як електронних документів забезпечується сегментація контенту навчальних матеріалів та вибір фрагменту для аналізу. Після цього проводиться розбиття фрагменту контенту електронного документу, на менші фрагменти – фрази. Під фразою розуміється семантично цілісний вузол, що виокремлений форматуванням тексту чи розділовими знаками, й локалізує місцезнаходження окремих термінів. Одержання в результаті виконання блоку множини фраз дає можливість в подальшому обробляти на предмет пошуку термінів кожну з фраз окремо. Наступним кроком проводиться формування множини всіх можливих термінів, що присутні у досліджуваному контенті. До множини термінів навчального матеріалу M_T включаються всі можливі неперервні впорядковані послідовності слів, які не виходять за межі фраз та відповідають умові:

$$M_T = \left\{ \left\langle x_1, x_2, x_3, x_4, x_5, x_6 \right\rangle \middle| \begin{array}{l} x_1 \in M_I \text{ Y } M_{\Pi} \\ x_2, x_3, x_4, x_5, x_6 \in M_M \\ \left\langle x_1, x_2, x_3, x_4, x_5, x_6 \right\rangle \text{ Y } M_I \neq \varnothing \end{array} \right\} \quad (1)$$

де M_M – множина семантично значущих елементів (іменників M_I та прикметників $M_{ПК}$) та семантично зв'язуючих елементів (сполучників M_C, часток $M_Ч$ та прийменників $M_{ПЙ}$), $M_M = M_I \text{ Y } M_{ПК} \text{ Y } M_C \text{ Y } M_Ч \text{ Y } M_{ПЙ} \text{ Y} \varnothing$. Сегментація по термінах проводиться з використанням бази даних корпусу слів

української мови та в якості вихідних даних формує множину термінів M_T, що містяться в оброблюваному фрагменті електронного документу навчального матеріалу.

Лематизація та калькуляція термінів дозволяє на основі множини термінів M_T сформувати множину лемо-незалежних термінів M_{TI}, а також співставити кожному із них кількість появ у досліджуваному тексті. Для цього спершу проводиться лематизація кожного слова у кожній фразі в множині M_T. Після чого одержана множина обробляється й компактифікується таким чином, що всі ідентичні повторення термінів видаляються, а кожному терміну співставляється величина K_n, що відображає встановлену кількість появ даного терміну n у вхідній множині M_T.

Оскільки на етапі формування множини термінів M_T до неї додавались усі можливі варіанти термінів в межах фраз без поглинання більшими словосполученнями менших, в даному блоці проводиться аналіз необхідності такого поглинання. Одержана в результаті множина лемо-незалежних термінів M_{TI} містить терміни, що використовуються у навчальному матеріалі з кількісним показником використання, але не визначає важливість даних термінів.

Лематизація текстового контенту переводить текст електронного документу навчального матеріалу, що аналізується, до відповідної послідовності слів у інфінітивному стані. Вони дозволяють проводити подальше оцінювання дисперсії слів.

Метод дисперсійного оцінювання дозволяє відділити із загальної множини широковживаних у тексті слів слова, що розташовані рівномірно й показав свою високу ефективність у попередніх дослідженнях [2]. Пошук та дисперсійне оцінювання важливих слів у параграфі призначені для оцінки важливості кожного слова в досліджуваному тексті, що проводиться з використанням методу дисперсійного оцінювання [3].

Оцінка важливості v_n кожного терміна n із множини M_{TI} обчислюється за формулою:

$$v_n = \sum_{i=1}^{x_n} \frac{K_n \sigma_n}{k_n}, \qquad (2)$$

де K_n – кількість появ терміну n в множині M_{TI}; k_n – кількість появ i-го слова терміну n в лематизованому текстовому контенті визначеного фрагменту електронного документу; σ_n – дисперсійна оцінка для i-го слова терміну n; x_n – кількість слів у терміні n.

Множина ключових термінів формується на основі лемо-незалежних термінів із множини M_{TI} з найбільшими значеннями оцінки важливості, а їх кількість випливає із визначення відомого показника з семантичної обробки текстів, щільності ключових слів $P_{ШТ}$.

Розглянута інформаційна технологія автоматизованого визначення семантичних термінів в елементах навчальних матеріалів дозволяє на основі електронного документу навчального матеріалу автоматизовано отримувати відповідний перелік ключових термінів навчального матеріалу.

Запропонована інформаційна технологія автоматизованого визначення семантичних термінів в елементах навчальних матеріалів була реалізована в дослідницькому програмному продукті. Структурно програмний продукт складається з ряду класів. Так, пошук ключових термінів та робота з ключовими термінами реалізовані у класі WordCombination. Клас IMainForm забезпечує інтерфейс для головної форми взаємодії з користувачем, а MainForm є класом користувацької форми. Для збереження і роботи з комбінаціями слів (словосполученням) використовується клас Combination. Клас WigthCombination наслідує клас Combination і розширює його можливостями обрахунку ваги словосполучення. IWorkWithServer реалізує інтерфейс для роботи з базою даних, а WorkWithServer забезпечує роботи з базою даних Microsoft SQL Server, який використовується як для збереження даних роботи, так і для використання бази даних корпусу слів української мови. Клас PresenterWork використовується для взаємодії графічної частина й логіки програми. WigthWord є класом для обрахунку ваги слова в контексті досліджуваного фрагменту тексту. Для зберігання множини слів і всіх пов'язаних із ним даних використовується клас Word. Клас SelectTerm зберігає терміни, виділені в тексті і тип виділення для подальшого аналізу важливості термінів. Section – клас, який приймає текст в межах певного параграфа й організовує подальшу обробку даного фрагменту. Для первинного аналізу тексту і розбиття його на параграфи (контент, прив'язаний до одного елементу заголовку Heading, використовується клас ProcessText.

Вхідними даними для системи є електронний документ навчального матеріалу, а вихідними даними є відповідна множина ключових термінів. Зокрема, на рисунку 2 показано приклад обробки теми «Нейромережі когнітрон та неокогнітрон» навчального матеріалу дисципліни «Методи та системи штучного інтелекту».

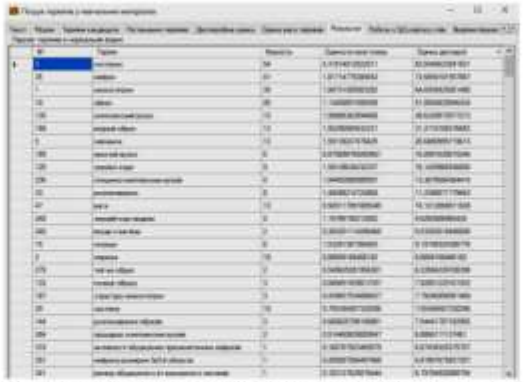


Рис. 2 – Приклад роботи розробленого програмного продукту

Ефективність практичного застосування розглянутої інформаційної технології автоматизованого визначення семантичних термінів в елементах навчальних матеріалів може бути визначена шляхом оцінки результатів використання відповідного програмного продукту за показниками точності та повноти [4].

Точність пошуку P (Precision, відношення кількості релевантних ключових термінів, знайдених автоматично, до загальної кількості знайдених ключових термінів в досліджуваному тексті) та повнота пошуку R (Recall, відношення кількості релевантних ключових термінів, знайдених автоматично, до загальної кількості релевантних ключових термінів в досліджуваному тексті) обчислюються за наступними формулами:

$$P = \frac{\left| M_{TK}^{E} \text{ I } M_{TK} \right|}{\left| M_{TK} \right|}, \; R = \frac{\left| M_{TK}^{E} \text{ I } M_{TK} \right|}{\left| M_{TK}^{E} \right|}, \qquad (4)$$

де M_{TK}^{E} – множина релевантних ключових термінів, сформована експертом; M_{TK} – множина знайдених автоматично ключових термінів.

Середня точність пошуку $\overline{P}$ та середня повнота пошуку $\overline{R}$ визначаються наступним чином:

$$\overline{P} = \frac{\sum_{i=1}^{k} P_k}{k}, \; \overline{R} = \frac{\sum_{i=1}^{k} R_k}{k}, \qquad (5)$$

де k – кількість навчальних матеріалів у тестовій вибірці.

Для визначення ефективності практичного застосування інформаційної технології автоматизованого визначення семантичних термінів в елементах навчальних матеріалів, тестовим програмним продуктом було оброблено тестову вибірку з 50 файлів навчальних курсів. Так, у результаті тестування розглянутого на прикладі рисунку 2 навчального матеріалу за показника щільності ключових слів 7% було отримано наступне:

– до множини ключових термінів автоматично було віднесено наступний перелік термінів: *когнітрон, неокогнітрон, нейрон, комплексний вузол, простий вузол, образ, вхідний образ, навчання;*

– до множини ключових термінів експертом було віднесено наступний перелік термінів: *когнітрон, неокогнітрон, нейрон, збуджуючий нейрон, гальмуючий нейрон, комплексний вузол, простий вузол.*

Відповідно до математичних моделей (4), даному випадку точність пошуку склала 0,625, а повнота пошуку склала 0,714. Відповідно до (5), середня точність пошуку для дослідженої вибірки з 50 файлів навчальних курсів склала 0,732, а середня повнота пошуку склала 0,697. Мінімальна точність пошуку одержана 0,512, мінімальна повнота пошуку – 0,581; максимальна точність пошуку – 0,929, максимальна повнота пошуку – 1,000.

Таким чином, було запропоновано інформаційну технологію автоматизованого визначення множини ключових семантичних термінів у контенті елементів навчальних матеріалів, що ґрунтується на пошуку використаних фраз у тексті та дисперсійній оцінці важливості слів. Вхідними даними інформаційної технології є електронний документ навчального матеріалу та обраний елемент для аналізу, вихідними даними є відповідна множина ключових семантичних термінів навчального матеріалу.

Розглянуто тестовий програмний продукт, що дозволяє автоматизовано визначати множину ключових семантичних термінів за даною інформаційною технологією. Проведені дослідження підтвердили можливість ефективно формувати множини ключових семантичних термінів елементів навчальних матеріалів з середніми

показниками точності пошуку до 73,2% та повноти пошуку до 69,7%. Аналіз отриманих результатів виявив, що відсутність програмно визначених термінів у множині автора не завжди характеризує недолік розглядуваної технології. Деякі семантично важливі терміни автори суб'єктивно ігнорують, в той час як іншу категорію складають поняття, на яких автори акцентують надмірну увагу попри їх другорядність в рамках матеріалу, що викладається.

Встановлена ефективність запропонованої інформаційної технології сприяє її використанню для вирішення ряду актуальних задач [1].

Література

1. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів / О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2017, №6. – С.223-229.
2. Бармак О. В., Мазурець О. В. Інформаційна технологія автоматизованого визначення термінів у навчальних матеріалах / О. В. Бармак, О. В. Мазурець // Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах» – Хмельницький, 2015. – №2. – С.94–102.
3. Ortuño M., Carpena P., Bernaola P., Muñoz E., Somoza A.M. Keyword detection in natural languages and DNA / M. Ortuño, P. Carpena, P. Bernaola, E. Muñoz, A. M. Somoza // Europhys. Lett, 2002. – 57(5). – P. 759-764.
4. Powers D. M. W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation / D. M. W. Powers // Journal of Machine Learning Technologies, Vol. 2, No. 1. (2011), p. 37-63.

УДК 004.4

МАТЕМАТИЧНІ МОДЕЛІ ДЛЯ ВИЗНАЧЕННЯ СЕМАНТИЧНИХ ТЕРМІНІВ У КОНТЕНТІ НАВЧАЛЬНИХ МАТЕРІАЛІВ

Кондаков О.В., Мазурець О.В., Скрипник Т.К.
Хмельницький національний університет, Україна
E-mail: exe.chong@gmail.com

Поширення інформаційних технологій та розвиток глобальної мережі та телекомунікацій привели до значних змін у вищій освіті. Одними із проявів цих змін стало виникнення дистанційної форми освіти й розвиток спеціалізованих навчальних середовищ, найбільш

розповсюдженим із яких наразі є Moodle [1]. Застосування спеціалізованих навчальних середовищ й електронних навчальних курсів вимагає вирішення ряду задач автоматизації, зокрема: автоматизація побудови семантичної моделі навчальних курсів, оцінка відповідності навчальних матеріалів вимогам, допомога та контроль якості при формуванні навчальних матеріалів, оцінка відповідності наборів тестових завдань навчальним матеріалам, допомога та контроль якості при формуванні тестів до навчальних матеріалів, автоматизована генерація прототипів тестових завдань, реалізація гнучких алгоритмів тестування, автоматизація формування анотацій і рефератів до елементів навчальних матеріалів тощо.

Вирішення всіх цих задач може бути реалізоване через автоматизацію побудови семантичної моделі навчальних курсів та її використання у відповідних інформаційних технологіях. Одним із способів вирішення задачі оцінки семантичної відповідності є аналіз термінологічної бази навчальних матеріалів. Тому задача автоматизації визначення семантичних термінів у навчальних матеріалах є актуальною задачею сучасної освіти.

Термінами можуть бути як ключові слова, так і ключові словосполучення. Ключові словосполучення можуть містити довільну кількість слів, і бути семантичними мережами малої ємності. В рамках аналізу їх склад доцільно спрощувати до двох слів. При цьому в процесі пошуку як мінімум одне із цих слів можна розглядати як термін в межах навчальних матеріалів. Тож питання автоматизації пошуку ключових слів у контенті навчальних матеріалів є першочерговою задачею в процесі вирішення розглядуваної проблеми.

Метою роботи є дослідження сучасних відомих методів аналізу текстів для оцінки їх ефективності й придатності до використання у задачі автоматизації пошуку ключових семантичних термінів у контенті навчальних матеріалів.

Застосування різноманітних методів аналізу текстів дозволяє зіставити окремим словам або словосполученням тексту деякі певним чином поставлені у відповідність числові вагові значення, що вказують на міру їх важливості в досліджуваному тексті. Ці методи розрізняються за алгоритмами обрахунку вказаних вагових значень [2]. Найбільш розповсюдженими методами аналізу текстів є частотна оцінка, оцінка TFIDF та дисперсійна оцінка.

Частотна оцінка TF (term frequency) є частотою згадувань певного слова і у тексті, що розглядається, й обчислюється наступним чином [3]:

Міністерство освіти і науки України
Хмельницький національний університет

АПКН 2019

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XI всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2019»

14-15 листопада 2019

Том 1
Роботи студентів та молодих вчених Факультету програмування та комп'ютерних і телекомунікаційних систем ХНУ

Хмельницький 2019

ЗМІСТ

УДК 004.8

Артюхова Д.І.

Хмельницький національний університет

СПОСІБ ОБМЕЖЕННЯ МНОЖИНИ КЛЮЧОВИХ ТЕРМІНІВ У ЦИФРОВИХ ТЕКСТАХ

Стаття присвячена проблемі визначення достатнього обсягу множини ключових термінів для цифрових текстів. Запропонований спосіб дозволяє автоматизовано обмежувати множину ключових термінів шляхом пошуку достатнього значення відношення кількості ключових термінів у тексті до загальної кількості слів у даному тексті.

The article deals with the problem of defining a sufficient number of key terms for digital texts. The proposed method allows to automatically limit many key terms by finding a sufficient ratio of the number of key terms in the text to the total number of words in the text.

Семантичний аналіз текстів є важким математичним завданням, яке ускладнюється необхідністю обробки природної мови. Множину ключових слів цифрового текстового документу називають пошуковим образом цього документу. Множина ключових слів тексту є найбільш семантично стиснутим результатом семантичного аналізу тексту, й якість її автоматизованого формування в рамках семантичного аналізу тексту визначається ефективністю використовуваних методів пошуку ключових слів.

Для автоматизації пошуку ключових слів використовуються різноманітні методи аналізу текстів, таких як TFIDF, частотна оцінка та дисперсійна оцінки тощо [1]. Хоча використання таких методів є ефективним, вихідними даними є множина ключових термінів із кількістю елементів, рівною кількості оригінальних слів у документі. Оскільки кожному елементу співставляється деяке певним чином поставлене у відповідність числове вагове значення [2], то множина може бути впорядкована за важливістю відповідних термінів, але проблему складає необхідність обмеження вихідної множини [3]. Очевидно, що даний етап є останнім при формування вихідної множини ключових термінів (рис. 1).

Запропоновано визначати рекомендовану кількість ключових термінів у цифрових текстах номінально похідною від параметру щільності ключових термінів [4]. Щільність ключових термінів є

УДК 004.4

Ковальчук О.В., Білоус Г.А., Слободзян В.О.

Хмельницький національний університет

ВИКОРИСТАННЯ ПРОГРАМНОГО РОЗШИРЕННЯ SPIRE.DOC ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З ЦИФРОВИМИ ДОКУМЕНТАМИ

З метою визначення найбільш ефективного програмного розширення проведено аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки електронних документів: Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll. В результаті аналізу розглянутих бібліотек, було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки електронних документів. Встановлено, що перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з електронним документом, так і процес програмування.

Microsoft.Office.Interop.Word.dll, DocumentFormat.OpenXml.dll and Spire.Doc.dll were useing for chooce the most effective program components for automation work with electronic documents. In result audit this libraries there was chosen Spire.Doc.dll for use to automation of processing of electronic documents. It is established that in result transfer of functions for automatic matching styles of text block to their properties from the level of the functional code of the application to the level of the functional library simplify work of the application with electronic document and programming process.

Для ефективної реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають об'єктно-орієнтований інструментарій для програмної роботи з контентом відповідних файлів [1]. Для реалізації програмної обробки цифрових документів є доцільним використання розглянутих в даній статті спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентом відповідних файлів, зокрема: Microsoft.Office.Interap.Word.dll [2], DocumentFormat.OpenXml.dll [3], Spire.Doc.dll [4].

Метою роботи є аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів і визначення найбільш ефективного програмного розширення.

На сучасному етапі для зберігання електронних документів навчальних матеріалів використовується файл з розширенням .DOCX (або створених на його основі .HTML, .PDF тощо). На відміну від файлів .DOC, які зберігають дані документа в одному бінарному файлі, файли .DOCX створюються за допомогою відкритого формату XML, в якому зберігаються документи як колекції окремих файлів і папок в стиснутому пакеті. .DOCX-файли містять .XML-файли і три папки, docProps, Word, і _rels (Рис. 1), які містять властивості документа, його зміст і відношення між файлами, тему та включені файли. .DOCX-файли розроблені, щоб зробити вміст документів доступним і відкритим – так, текстовий документ чи зображення зберігаються як окремі прості файли в такій колекції у складі одного файлу .DOCX.

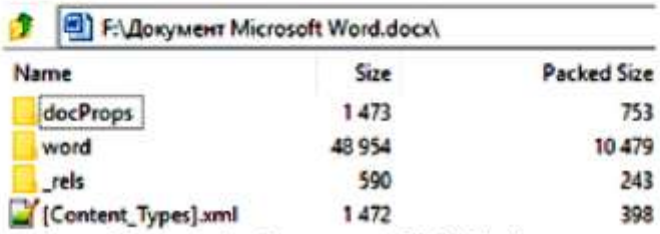


Рисунок 1 – Структура .DOCX-файлу

Office OpenXML є форматом електронних офісних документів, таблиць, презентацій та діаграм, розроблений компанією Microsoft. Даний формат є міжнародним стандартом, який затверджений в Міжнародній організації зі стандартизації (ISO) [5]. Він включає в себе такі первинні мови розмітки:

- WordprocessingML – для обробки текстових документів;
- SpeadsheetXL – електронних таблиць;
- PresentationML – для презентацій;
- DrawingML – для векторної графіки, діаграм.

Хоча Word реалізує свою функціональність через об'єктну модель, збереження даних відбувається у наборі XML-файлів, що ускладнює можливості для автоматизації прямого програмного парсингу [6]. Оскільки файл стилів та текст знаходяться в різних файлах у складі .XML, потрібно зчитувати одразу ряд файлів та на базі їх взаємозв'язків програмно формувати об'єктну модель документа, що є не найкращим рішенням.

Бібліотека ***Microsoft.Office.Interap.Word.dll*** [2] дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM.

За допомогою процесів автоматизації Office Automation програми, написані мовами, такими як Visual C# .NET, отримують можливість програмно керувати іншими програмами. Автоматизація Microsoft Word (далі Word) дозволяє виконувати відповідні дії, такі як створення нового документу, додавання даних у документ або створення таблиць. З програмами Word та іншими програмами Microsoft Office практично всі дії, які користувач може виконувати вручну за допомогою користувацького інтерфейсу, також можуть бути виконані програмним шляхом за допомогою автоматизації Office Automation.

Word реалізує цю програмну функціональність через об'єктну модель [7]. Об'єктна модель є набором класів і методів, які служать аналогами логічних компонентів Word. Наприклад, існує об'єкт Application, об'єкт Document та об'єкт Paragraph, кожен з яких містить функціональність цих частин Word.

Щоб використовувати функції програми MS Office в проекті, можна використовувати первинний взаємозв'язок PIA (Primary Interop Assembly). PIA дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM. При створенні нового проекту Office, Visual Studio додає посилання на PIA, необхідний для побудови проекту. При цьому, у деяких сценаріях доводиться додавати посилання на додаткові PIA (наприклад, якщо необхідно використовувати функцію MS Office Word в проекті для MS Office Excel).

DocumentFormat.OpenXml.dll

Бібліотека класів .NET DocumentFormat.OpenXml.dll (версії 2.0 / 2.5 / 2.8.1) [3] дає можливість розробникам програмного забезпечення працювати з пакетом Microsoft Office. Компанія надає дану бібліотеку безкоштовно в повному доступі, й її можна завантажити з офіційного сайту Microsoft або засобами Visual Studio.

Для роботи з файлами формату .DOCX, використовується мова розмітки WordprocessingML [8]. На рисунку 2 зображено приклад структури цієї мови розмітки.

Теги, наведені на рисунку 2, мають такі властивості:

- *document* – є основою частиною документа
- *body* – контейнер для збору структур рівня блоку, які складають основну історію;
- *p* – абзац (paragraph);
- *r* – контейнер який містить в собі текст з однаковими властивостями (run);
- *t* – власне текстовий елемент (text).

Всі інші теги, які можуть використовуватися, є допоміжними до вищенаведених і додають їм стильових властивостей.


```
<?xml version="1.0" encoding="utf-8"?>
<w:document
xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2
006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t> Текст </w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document
```

Рисунок 2 – Приклад WordprocessingML-розмітки .DOCX-документу

Spire.Doc.dll [4] для .NET є повністю незалежною бібліотекою класів .NET Word, спеціально створеною для розробників, й дозволяє швидко генерувати, відкривати, редагувати та зберігати документи Word різних версій. Конвертація функціональних можливостей дозволяє розробникам здійснювати перетворення Word в інші актуальні формати документів (.PDF, .EPub, .HTML, .RTF, .Image, .XML тощо). Можна створювати та обробляти вже існуючі документи Word динамічно.

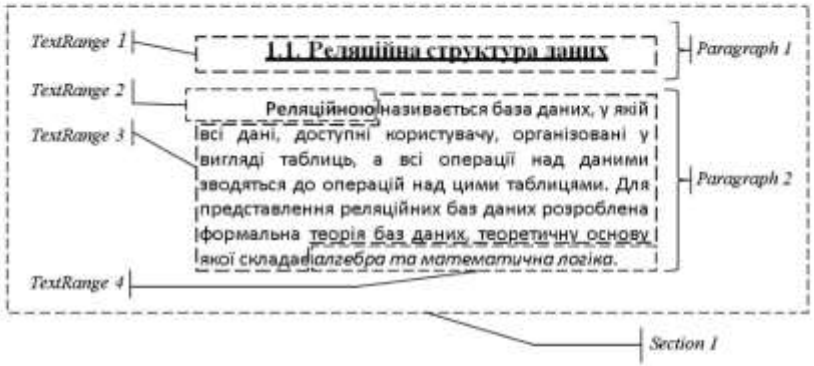


Рисунок 3 – Приклад використання об'єктної моделі MS Office

Бібліотека Spire.Doc.dll забезпечує роботу з майже всіма елементами документа Word, а саме: сторінки, розділи, заголовки, колонтитули, абзаци, переліки, таблиці, текст, виноски, поля, гіперпосилання, закладки, коментарі, зображення, стилі, фонові параметри, функції друку, налаштування документа та захисту. Крім того, підтримуються графічні об'єкти, включаючи форми, текстові поля,

зображення, OLE-об'єкти та елементи керування. Бібліотека Spire.Doc.dll підтримує функцію пошуку та заміни, вирівнювання, перерву сторінки, поле заповнення, об'єднання документів, копіювання документів, друк тощо.

При зчитуванні файлу Spire.Doc.dll перетворює документ в об'єктну модель. Ця модель має структуру, яка починається документом і закінчується об'єктом *TextRange*. Приклад використання такої об'єктної моделі документу MS Office показаний на Рисунку 3.

Одержати відомості про стилі *Paragraph* можна за допомогою методу *GetStyle()* (Рис. 4). Через роботу з відповідними властивостями, можна одержати текст та стилі (Рис. 5), що відносяться до визначеного *TextRange*.

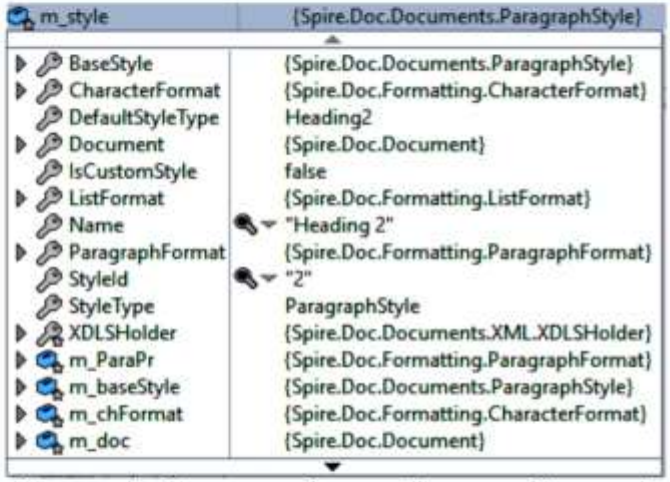


Рисунок 4 – Одержання відомостей про стилі Paragraph

Хоч бібліотека Microsoft.Office.Interap.Word.dll надає доступ до всіх функцій MS Office, так як працює безпосередньо з PIA, для її роботи потрібна ліцензія для MS Office на кожному клієнтському комп'ютері. Крім того, при використанні Automation, MS Office завантажується у фоновому режимі, внаслідок чого займає деякий обсяг оперативної пам'яті та завантажує велику кількість файлів і DLL. Додатки MS Office були розроблені як додатки для користувацького інтерфейсу, і через це Microsoft.Office.Interop.Word.dll працює повільно. Microsoft не рекомендує використовувати Office Automation (або будь-який Office Interop) на сервері.

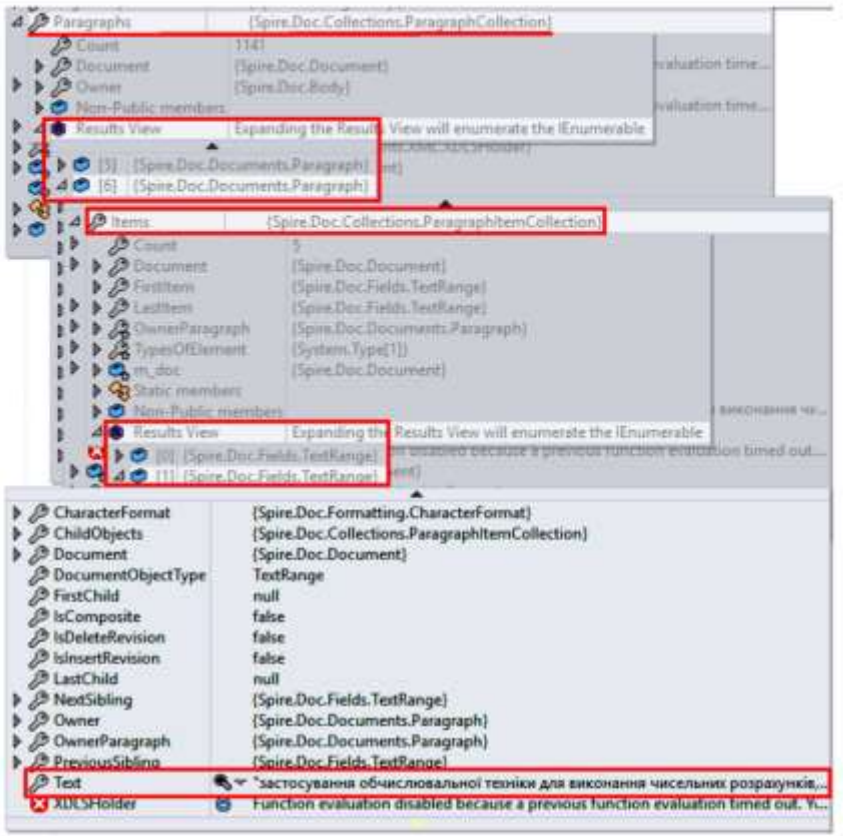


Рисунок 5 – Позиція визначеного тексту в TextRange у об'єктній моделі

На відміну від Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll не вимагають наявності MS Office на машині користувача та його запуску в фоновому режимі.

При роботі з docx-файлом за допомогою DocumentFormat.OpenXml.dll потрібно дотримуватися тієї ж ієрархії, що і сама структура розмітки тобто: *document-paragraph-run-text*. Також відповідно до розглянутих вище особливостей взаємозв'язків всередині документу, при присвоєнні параграфу деякого стилю, у властивостях параграфа надається лише ідентифікатор (ID) на даний стиль, натомість сам стиль описується окремо у файлі style.xml. Внаслідок необхідності регулярного співставлення ID стиля з контейнером style.xml для одержання характеристик стилів абзаців, зростає складність програмного використання бібліотеки. Перевагою Spire.Doc.dll визначено відсутність

необхідності співставлення ID стиля з контейнером style.xml, оскільки дана функція реалізована на рівні бібліотеки.

Таким чином, в результаті аналізу сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів, було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки цифрових документів. Основними перевагами Spire.Doc.dll встановлено відсутність необхідності наявності MS Office на машині користувача та запуску в фоновому режимі, а також реалізацію функцій автоматичного співставлення стилів текстових блоків їх властивостям на рівні розширення.

Перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з цифровим документом, так і процес програмування.

Перелік посилань

1. Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №1. – С.61-69.
2. Considerations for server-side Automation of Office [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office/
3. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу: https://www.iso.org/search/x/query/Open%2520XML
4. Spire.Doc for .NET [Електронний ресурс]. – Режим доступу: https://www.nuget.org/packages/Spire.Doc/
5. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу: https://www.iso.org/search/x/query/Open%2520XML
6. Office Open XML [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Office_Open_XML
7. How to automate Microsoft Excel from Microsoft Visual C#.NET [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/help/302084/how-to-automate-microsoft-excel-from-microsoft-visual-c--net
8. DocX for creates or modifies Microsoft Word files [Електронний ресурс]. – Режим доступу: https://github.com/xceedsoftware/DocX
9. Free Spire.Doc for .NET [Електронний ресурс]. – Режим доступу: https://www.e-iceblue.com/Introduce/free-doc-component.html

УДК 004.912

Колісник О.Ю., Багрій Р.О., Скрипник Т.К.

Хмельницький національний університет

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ФОРМУВАННЯ ТЕКСТОВИХ ПОВІДОМЛЕНЬ ЗА ДОПОМОГОЮ РУХІВ РУКИ ЛЮДИНИ

Незважаючи на зростаючу популярність пристроїв, що слідкують за рухами людини, розробка ефективних методів введення тексту для цих пристроїв залишається не дослідженою. У роботі запропоновано інформаційну технологію, що надає можливість введення текстових повідомлень з використанням пристрою Leap Motion, за допомогою якого користувач керує вказівником на віртуальній клавіатурі за допомогою рухів руки. Зокрема, досліджено можливості пристрою Leap Motion, визначено перелік рухів для керування. Також досліджено методи введення літер та запропоновано дизайн віртуальної клавіатури.

Despite the increasing popularity of devices that track human movements, the development of effective text input methods for these devices remains unknown. The paper proposes information technology that provides enter text messages using the Leap Motion device, with which the user controls the pointer on the virtual keyboard using hand gestures. In particular, the possibilities of the Leap Motion device were investigated, a list of movements for control was determined. It also explored text input methods and proposed a virtual keyboard design.

Вступ Багато робіт присвячено методам введення тексту на персональних комп'ютерах і мобільних пристроях. Проте, роботи, пов'язані з безконтактним введенням тексту, не достатньо поширені. Це відбувається головним чином тому, що завжди виникають проблеми при проектуванні та реалізації віртуальної поверхні для введення тексту, відстеженню і сприйняттю дій користувача та відображенню взаємодії між діями користувача і поверхнею введення.

Ведення тексту за допомогою рухів руки людини передбачає відсутність фізичної клавіатури або сенсорного екрану при взаємодії користувача з комп'ютером. Отже, виникає необхідність в розробці віртуальної клавіатури, візуальні елементи якої повинні бути пов'язана з вхідним сигналом, що передається від зовнішнього пристрою.

Метою дослідження є розробка інформаційної технології, яка б дала можливість введення текстових повідомлень за допомогою рухів руки з використанням зовнішніх пристроїв.

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ

за матеріалами XII всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2020»

9-10 листопада 2020

Хмельницький 2020

ЗМІСТ

УДК 316.74:004.4

Алексейко В. О.

Хмельницький національний університет

ФЕЙКОВІ НОВИНИ ЯК ФЕНОМЕН СУЧАСНОСТІ

Розглянуто феномен фейкових новин. Представлені класифікація, характеристика і різновиди фейкових новин. Описано приклади створення фейків за допомогою штучного інтелекту. Наведено приклади боротьби з фейками, етапи їх виявлення та правила, яких необхідно дотримуватися, щоб не стати жертвою фейкової інформації.

The phenomenon of fake news is considered. The classification, characteristics and varieties of fake news are presented. Examples of creating fakes through the artificial intelligence are described. Examples of the struggle against fakes, the stages of their detection and the rules that must be followed in order not to fall victim of fake information are given.

Останнім часом спостерігається масовий відтік користувачів з більш надійних традиційних засобів масової інформації, таких як газети, радіо і телебачення, в нові формати: соціальні мережі, YouTube, подкасти, онлайн-журнали, новинні додатки. Завдяки мережі Інтернет з'явилась можливість миттєвого доступу до широкого спектру джерел інформації. Крім того, існують численні послуги, що дозволяють ділитися новинами з мільйонами людей по всьому світу. Великий обсяг інформації, до якої люди отримують доступ, зазвичай, не перевіряється і вважається достовірним, що призводить до появи такої проблеми як дезінформація користувачів за допомогою публікації фейкових новин.

Сучасне цифрове суспільство вступає в епоху пост правди, де фейкові новини починають витісняти звичайні. У 2020 році світ накрила хвиля фейків. За статистикою, понад дві третини європейців стикаються з недостовірними новинами принаймні раз на тиждень, в США майже половина інтернет-користувачів несвідомо поширювали фейкові новини. Соціальні мережі в один голос заявляють, що боротьба з неправдивою інформацією один із їхніх пріоритетів. Зокрема Facebook щодня видаляє близько мільйона підробних акаунтів. Пришвидшитись у цій боротьбі компанії змусила пандемія, яка породила безліч фейків й теорій змов [1].

Метою роботи є огляд різновидів фейкових новин та способів їх виявлення.

Наразі не існує єдиного визначення поняття «фейкові новини». Слово fake має суто негативне значення, яке співвідноситься з поняттями «неправда», «обман» і «маніпулювання». Fake news – це новини, засновані на неправдивій інформації, творці якої орієнтовані на обман і маніпулювання. [2].

Таким чином інформаційна система дозволить позбавити співробітника від рутинної повсякденної роботи. Автоматизація дозволить значно скоротити час. У друкованій формі автоматично будуть вказані всі необхідні реквізити відповідно до встановленої форми друку документів.

Договір № КОК01 від 24.10.2018

Код клієнта	КЛ02	Дохід за 6 місяців	66 000,00
П.І.Б.	Шульженко Марина Анатоліївна	Номер паспорту	МР258010
Адреса	Херсон, Фрунзе вул., Будинок 2, кв. 45	Ідентифікаційний код	659367696
Телефон	+380 (96) 512 36 47	Вид послуги	Кредит

Код обліку кредиту	Термін (в міс.)	Сума	Відсоткова ставка	Переплачено	Всього до сплати	Щомісячний платіж
КОК01	24	85000	2,9	2567,71	87567,71	3649,64

Рисунок 4 – Багатотабличний звіт

Реалізація запропонованих заходів щодо вдосконалення механізму управління клієнтською базою дозволить комерційним банкам створити умови для формування стійкої бази лояльних клієнтів, підвищити рентабельність їх обслуговування і зміцнити свої позиції на ринку банківських послуг.

Перелік посилань

1. Бекаревич, Ю. Б. Самоучитель Microsoft Access 2013 / Ю. Б. Бекаревич, Н. В. Пушкіна. – Санкт Петербург.: БХВ-Петербург, 2014. – 464 с
2. Гайна Г. А. Основи проектування баз даних: Навчальний посібник / Г. А. Гайна. – Київ: КНУБА, 2005. – 204 с.
3. Лютий І. О. Банківський маркетинг: підручник для студентів вищих навчальних закладів / І. О. Лютий, О. О. Солодка. – Київ: Центр учбової літератури, 2010. – 776 с.
4. MS Access. Бази даних MS Access. MS Access 2007. [Електронний ресурс]: – URL: http://hi-news.pp.ua/kompyuteri/1607-ms-access-bazi-danih-ms-access-ms-access-2007.html

УДК 004.91

Ковальчук О. В., Мазурець О. В.

Хмельницький національний університет

МЕТОД ГЕНЕРАЦІЇ ТЕСТОВИХ ЗАВДАНЬ ДО НАВЧАЛЬНИХ МАТЕРІАЛІВ НА ОСНОВІ ПРОДУКЦІЙНИХ ПРАВИЛ

Розглянуто метод автоматизованої генерації тестових завдань до навчальних матеріалів. В основі методу покладено продукційну модель подання знань. Розроблений метод забезпечує повне покриття тестовими завданнями семантичної структури навчального матеріалу. Згенерована множина тестових завдань може використовуватися для перевірки знань ключових термінів в межах рубрик навчального матеріалу. Також завдяки створеним зв'язкам між тестовим завданням та рубрикою і між тестовим завданням та ключовим терміном з'являється можливість контролювати використання тестових завдань залежно від семантичної значущості термінів різних рівнів навчального матеріалу. Сформовані множини зв'язків та тестових завдань дозволяють проводити адаптивне тестування знань навчального матеріалу.

The method for the automated generation of test tasks has been considered. The proposed method is based on production rules. The method provides full coverage of the semantic structure of education material by tests. Generated set of tests can be used for testing knowledge of key terms within the headings of the educational material. Also, created relationships between tests and headings and relationships between tests and key terms provide the ability to control the using of tests during testing depending on the semantic value of key terms on different levels of education material. Generated sets of tests and sets of relationships can be used for adaptive testing of educational material knowledge.

На сьогоднішній день існують методи та інформаційні технології автоматизованої генерації тестових завдань. Даному напрямку присвячено ряд наукових публікації, серед яких варто виділити роботи Титенко С. В. [1], Пасічника Р. М. та Мельника А. М. [2, 3], Попова А. М. [4].

Існуючі на сучасному етапі підходи до генерації тестових завдань мають характерні недоліки, які полягають у відсутності зв'язку створених тестових завдань із елементами семантичної структури інформаційного навчального матеріалу. Це спричиняє нерівномірність покриттям тестом навчального матеріалу на етапі перевірки рівня знань.

Метою роботи є розробка методу автоматизованої генерації тестових завдань, які покривають всю семантичну структуру навчального матеріалу в вигляді системи рубрикації та множин ключових термінів кожної із рубрик.

Робота методу автоматизованої генерації тестових завдань ґрунтується на використанні моделі подання навчального курсу дисципліни як сукупність множин

інформаційного та тестового навчального матеріалу. В такому випадку навчальний курс можна записати наступним чином:

$$C = H \cup R_1 \cup K \cup R_2 \cup S \cup R_3 \cup Q \cup R_4 \cup R_5 \quad (1)$$

де H – множина рубрик (заголовків) інформаційного навчального матеріалу, R_1 – множина зв'язків між заголовками, K – множина ключових термінів, R_2 – множина зв'язків між ключовими термінами та реченнями, R_3 – множина зв'язків між заголовками та реченнями, Q – множина тестових завдань, R_4 – множина зв'язків між тестовим завданнями та реченнями, R_5 – множина зв'язків між тестовим завданнями та ключовими термінами.

Правила продукції є логічною формою представлення і перетворення знань [5]. Ядро правила продукції складається з двох частин таких як: антецедент і консеквент. Антецедент являється умовною частиною в якій комбінуються набір умов, при здійснені яких активується правило і відбувається виконання дій, визначених у консеквенті. При застосуванні даного підходу для створення тестового завдання, антецедентом являється шаблон речення, за яким відбувається пошук речення в контенті, а консеквентом – алгоритм створення тестового завдання на основі знайденого речення.

Використання правил продукції для генерації тестових завдань подібне до алгоритму формування тестів людиною, тому продукційні правила були обрані за основу в методі генерації тестових завдань. Також в процесі дослідження встановлено, що обов'язковою частиною правила є ключовий термін та зв'язуючий фрагмент (далі конектор). Наявність ключового терміну дозволяє обирати важливі фрагменти текстового контенту навчального матеріалу. Конектором є слово або символ, який зв'язує ключовий термін із тезою. Прикладом конектора є наступні фрагменти: «це», «називається», «таких як:» і т.д. (рисунок 1).

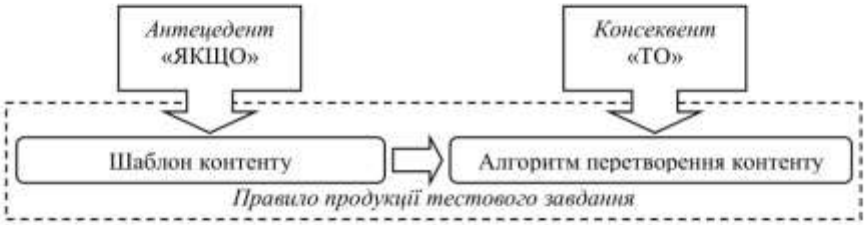


Рисунок 1 – Схема продукційного правила для формування прототипів тестових завдань

Запропонований метод автоматизованої генерації тестових завдань складається із трьох етапів і його схема зображена на рисунку 2. У вхідні параметри метод приймає рубрику h із множини H для якої потрібно згенерувати тестові завдання та множину речень S, відповідну рубриці h. Третім параметром методу є множина зв'язків між реченнями множини S та рубрикою h. Наступними вхідними

даними методу є множина ключових термінів K рубрики h та множина зв'язків між ключовими термінами та реченнями, в яких вони використовуються. Для роботи методу потрібна множина заздалегідь створених продукційних правил для генерації тестових завдань.

На *Етапі 1* методу генерації тестових завдань відбувається скорочення множини речень S, що дасть змогу оперувати меншим об'ємом даних що пришвидшить роботу методу. Даний етап розпочинається із *Кроку 1.1* на якому відбувається вибір s_i речення із множини S. На *Кроці 1.2* обирається ключовий термін k_b із множини K. Ключова робота даного етапу виконується на *Кроці 1.3*, де відбувається перевірка входження k_b ключового терміну в s_i речення, шляхом пошуку зв'язку між терміном k_b та реченням s_i в множині R_2. Якщо відповідний зв'язок вдалося знайти, то речення додається в множину S', а якщо ні – виконання методу переходить на *Крок 1.2* і обирається наступний елемент k_{b+1} із множини K. Коли множина K буде вичерпана, метод повертається на *Крок 1.1* та обирається наступне речення s_i із множини S. *Етап 1* завершується після вичерпання множини речень S. В результаті роботи даного етапу буде сформовано множину актуальних речень S', яка передається на наступний етап.

Етап 2 відповідає за формування множини прототипів тестового завдання. На *Кроці 2.1* відбувається вибір s_i речення із множини S', а на *Кроці 2.2* вибирається продукційне правило p_j із множини P. Застосування антецедента правила продукції p_j до речення s_i відбувається на *Кроці 2.3*. У випадку успішності *Кроку 2.3* виконання переходить до *Кроку 2.4*, де відбувається застосування консеквента правила продукції p_j до речення s_i для створення тестового завдання. Варто зазначити, що на даному кроці не завжди вдається створити тестове завдання відповідно до правил визначених у консеквенті. Пов'язано це з тим, що деякі консеквенти вимагають застосування випадково підібраних фрагментів тестового завдання, яких не можливо визначити на даному етапі методу. Тому результатом застосування консеквента на *Кроці 2.4* є прототип тестового завдання, який вноситься в множину Q' на *Кроці 2.5*. Після виконання *Кроку 2.5* або неуспішного застосування антецедента на *Кроці 2.3* відбувається перехід на *Крок 2.2*. Коли множина P вичерпується виконання переходить на *Крок 1.1* та обирається наступне речення із множини S'. Виконання *Етапу 2* завершується після того, як будуть оброблені всі елементи множини S'.

Заключним є *Етап 3*, де на основі прототипів із множини S' завершується генерація тестових завдань. За потреби на *Кроці 3.1* і *Кроці 3.2* відбувається пошук альтернативних речень $s_m \neq s_b$ та ключових термінів $k_n \neq k_b$ для доповнення прототипів тестового завдання відповідно до консеквентів, після чого на *Кроці 3.3* генеруються кінцеві версії тестових завдань, які додаються в множину Q. *Крок 3.4* відповідає за створення та наповнення множин зв'язків R_4 та R_5.

Вихідними даними методу генерації тестових завдань є множина тестових завдань Q для рубрики начального матеріалу $h \in H$, яка була передана у вхідний параметр, а також множина зв'язків R_4 та R_5 в межах рубрики h. Для наповнення

даних навчального курсу визначено в (1) розроблений метод потрібно застосувати циклічно для кожної рубрики інформаційного навчального матеріалу.

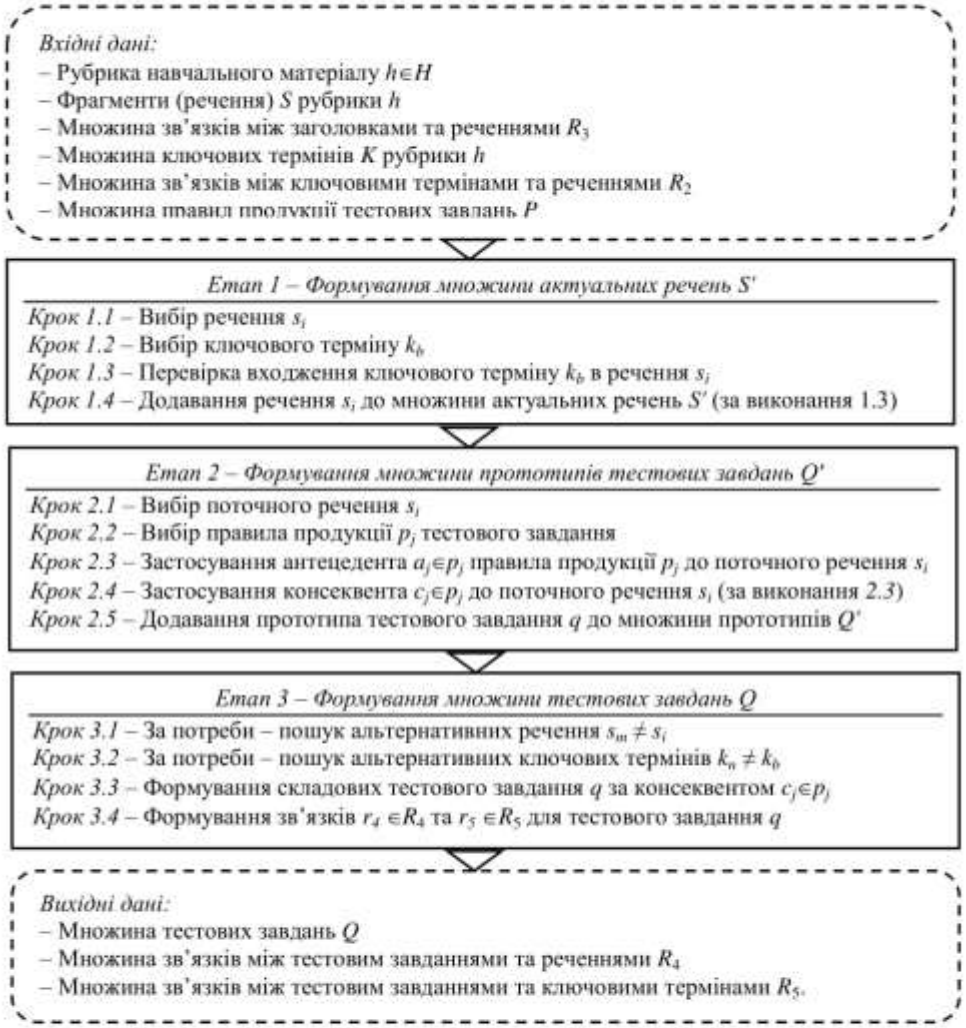


Рисунок 2 – Схема методу генерації тестових завдань

Для дослідження ефективності роботи методу автоматизованої генерації тестових завдань було реалізовано програмний додаток (рисунок 3) на платформі .NET Framework та за допомогою мови програмування C#. З метою покращення

якості згенерованих тестових завдань в програмному продукті було передбачено можливість їх видаляти та редагувати.

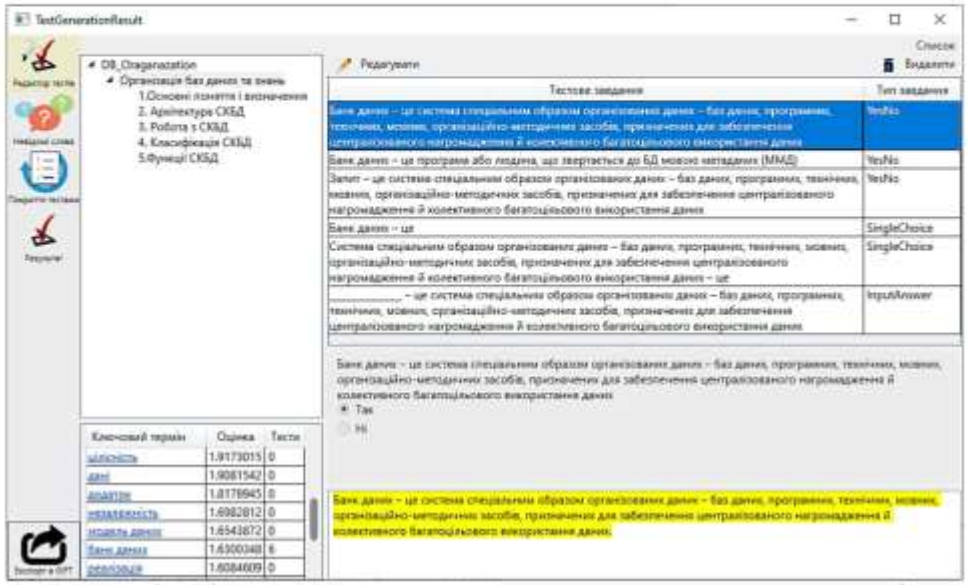


Рисунок 3 – Вікно програмного продукту для перегляду згенерованих тестів

Отже, за допомогою описаного методу можна згенерувати тестові завдання до інформаційного навального. При цьому використання розробленого методу дозволяє отримати наступний якісний ефект:

- множина згенерованих тестових завдань повністю покриває семантичну структуру;
- можливість використовувати згенеровані тестові завдання для перевірки рівня знань визначених ключових термінів у межах визначених рубрик інформаційного навчального матеріалу;
- можливість контролю використання тестових завдань у тестуванні залежно від обсягу наявних у контенті термінів різних рівнів семантичної значущості;
- можливість використання згенерованих тестових завдань для адаптивного тестування.

Перелік посилань

1. Титенко С. В. Генерація тестових завдань у системі дистанційного навчання на основі моделі формалізації дидактичного тексту [Електронний ресурс]. – Режим доступу: http://www.setlab.net/?view=Tytenko-test-generation
2. Мельник А. М. Автоматична генерація тестових завдань різних типів / А. М. Мельник // Вісник Хмельницького національного університету. Хмельницький – С.124-129

3. Мельник А. М.. Інформаційна технологія автоматичної генерації тестових завдань з керованою складністю / А. М. Мельник, Р. М. Пасічник, Р. П. Шевчук // Інформаційні технології в технічних системах – С.57-61
4. Попов А. М. Проектування тестових завдань закритого типу на базі моделі онтології на основі когнітивних прототипів / А. М. Мельник // Медична інформатика та інженерія. Хмельницький – 2015. – С.46-51
5. Продукційні правила [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki /Продукційні_правила.
6. Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №1. – С.61-69.
7. Ковальчук О.В.. Використання програмного розширення SPIRE.DOC для автоматизації роботи з цифровими документами / О.В. Ковальчук, Г.А Білоус., В.О. Слободзян // Збірник наукових праць за матеріалами XI всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2019». Хмельницький – 2019 – с. 116-122 .
8. Ковальчук О.В. Дослідження практичної ефективності інформаційної технології автоматизованого визначення семантичних термінів навчальних матеріалів / О.В. Ковальчук, О.В. Мазурець. // Сучасні технології в механіці : зб. наук. пр., Хмельницький – 2018 – с. 141-148 .
9. Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В Ковальчук, В. О. Слободзян, Г. А. Білоус. // Інтелектуальний потенціал – 2018 : збірник наукових праць молодих науковців і студентів, Хмельницький – 2018 – с. 51-56.

УДК 004.932.2

Ковдря В. Ю.

Національний авіаційний університет

АВТОМАТИЗОВАНА СИСТЕМА СЕГМЕНТАЦІЇ ЦИФРОВИХ ЗОБРАЖЕНЬ НА ОСНОВІ ДИСКРЕТНИХ СТРУКТУР

Розглянуто задачу сегментації цифрових зображень на основі графового підходу. Запропонована автоматизована система забезпечує якісну сегментацію і може бути використана для обробки даних з аерофотозйомки. Досліджено вплив параметрів k і minSize на результати сегментації. Проведено порівняння роботи алгоритму у різних кольорових просторах: RGB та Lab.

The problem of segmentation of digital images based on graph approach is considered. The proposed automated system provides high-quality segmentation and can be used to process aerial photography. The influence of k and minSize parameters on segmentation results is investigated. The work of algorithm is compared in different color spaces: RGB and Lab.

Задача сегментації зображень залишаються складними для комп'ютерного зору. Проте її використання є виправданим і необхідним у задачах високого рівня. Розділення сегментів на зображенні дозволяє зменшити навантаження на модулі розпізнавання, індексування тощо [1].

Метою роботи є розробка автоматизованої системи для сегментації цифрових зображень і тестування отриманої системи на даних з аерофотозйомки.

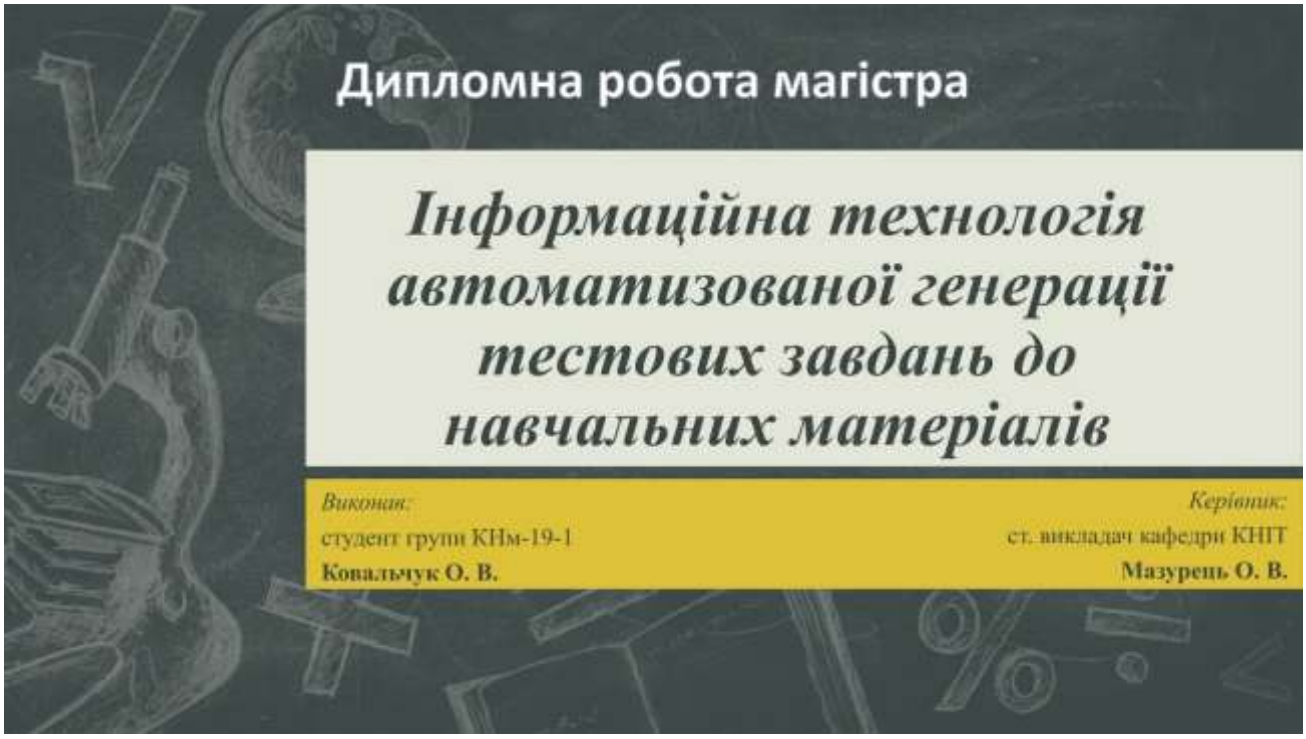
Завдання:

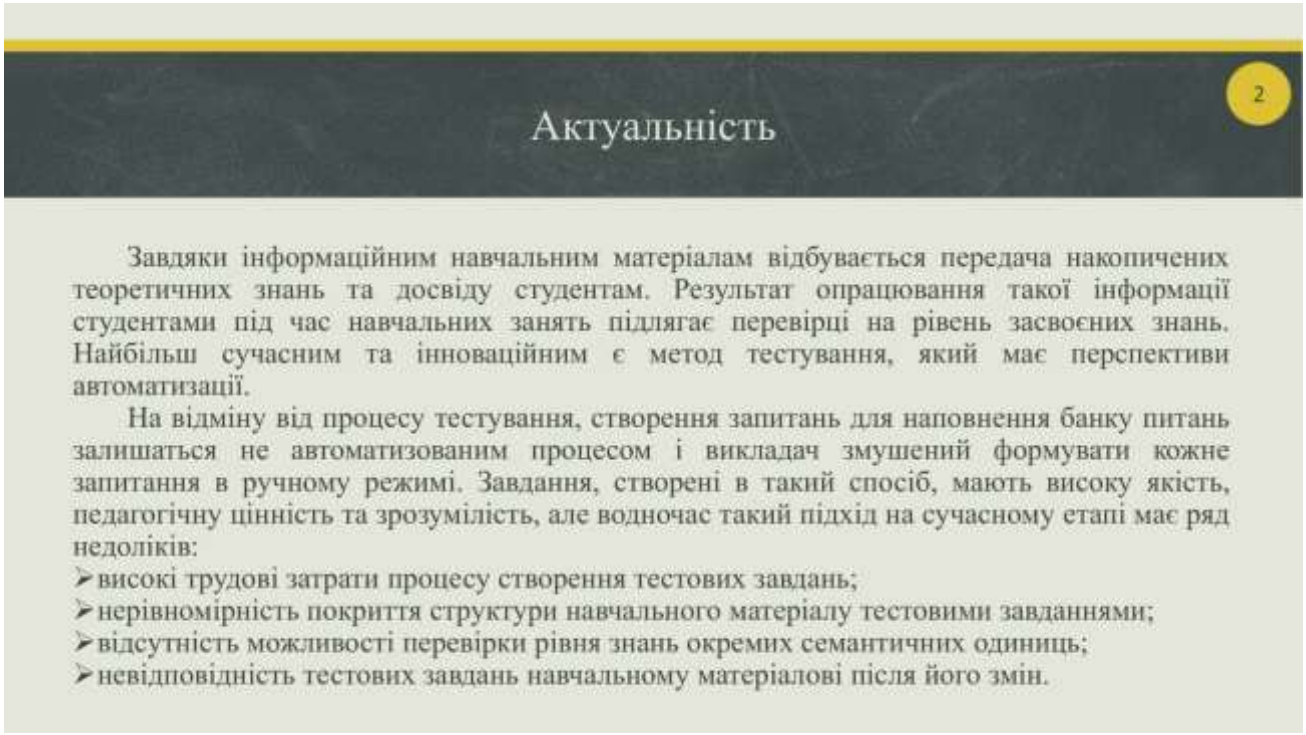
1) вивчити метод ефективної сегментації на ґратковому графі;
2) розробити алгоритм сегментації цифрового зображення методом ефективної сегментації на ґратковому графі;
3) розробити обчислювальну схему для отримання сегментації цифрових зображень;
4) створити автоматизовану систему на основі розробленого алгоритму;
5) оцінити якість отриманої сегментації;
6) провести тестування автоматизованої системи на реальних даних.

У графовому підході сегментація S являє собою розбиття V на компоненти таким чином, що кожен компонент $C \in S$ відповідає з'єднаним компонентом у графі $G' = (V, E')$, де $E' \in E$ [2]. Іншими словами, будь-яка сегментація є підмножиною ребер в E. На малюнку показано граф для сегментації (рисунок 1.а) і результат сегментації на ньому (рисунок 1.б).

Додаток Е

Презентаційний матеріал





Сучасні підходи до автоматизованої генерації тестових завдань

3

Назва	Опис
Семантичні мережі (використовують метод: Танченко С. С., Титенко С. В., Гагарин А. А.)	В основі методу покладено використання бази знань, структуровано однаковою якої є ланцюг «поняття – зв'язок – поняття». Базу знань заповнює експерт (викладач) на основі інформаційного навчального матеріалу. Тестові завдання генеруються шляхом опущення одного із елементів ланцюга та постановкою запитання про відсутню ланку. *Недоліком* даного методу є великі трудові затрати на формування і оновлення бази знань відповідно до навчального матеріалу.
Параметризовані тестові завдання (використовують метод: Брусиловський, П., Сосновський С., Каппі Е., Паттак С.)	Метод полягає у формуванні набору тестових завдань на основі шаблонів в якому змінюється певний параметр за автоматично згенероване значення за допомогою певної формули або алгоритму. Таким чином може бути згенерована велика кількість однотипних завдань, які при тестуванні будуть індивідуальними для кожної особи, яка проходить тестування. *Недоліком* методу є предметна спрямованість і підходить лише для перевірки знань практичних знань у точних науках.
Генерації тестових завдань на основі понятійно-тезисній моделі (використовують метод: Титенко С.В., Гагаріна О.О.)	В методі використовується база знань в яку входять поняття та тези із навчального матеріалу. Тестове завдання створюється на основі шаблону із підстановкою елементів із множини понять і тез. Недоліком методу є великі затрати часу на заповнення бази знань.
Автоматизована генерація тестових завдань за ключовими термінами *(Мазурець О. В., Бармак О. В.)*	Побудова тестових завдань для перевірки знань визначеного терміну за семантичною структурою навчального матеріалу. Метод не дозволяє вести повний контроль контенту ВІМ, який задано при тестуванні внаслідок того, що не ведеться облік використаних фрагментів ВІМ для генерації тестового завдання. В той час як в ряді випадків.
Генерації тестових завдань до формалізованих на основі системи семантичних класів навчального матеріалу (Мельник А. М., Пасічник Р. М.)	Метод генерації тестових завдань ґрунтується на формалізації навчальних матеріалів за допомогою системи семантичних класів, що дозволяє поділити текст на сукупності тверджень. Кожне твердження розбивається на компоненти, які описують процеси, поняття або їх характеристики, які поєднуються кількома. Для генерації тесту деяке твердження розбивається на основну та альтернативну частини, які можуть містити одну або декілька компонент. Альтернативна частина тесту повинностися аналогічними за лінгвістичним змістом, синтаксично узгодженими, частинами інших тверджень. *Недоліком* методу є потреба в ручному заповненні семантичних класів відповідно до навчального матеріалу.

Постановка задачі

4

Мета дипломної роботи магістра – розробка інформаційної технології автоматизованої генерації тестових завдань до інформаційних навчальних матеріалів в основі якого покладено метод автоматизованої генерації тестових завдань за допомогою продукційних правил.

Для досягнення мети потрібно вирішити наступні ***завдання***:

1) провести аналіз сучасних методів автоматизації генерації тестів для визначення існуючих можливостей генерації тестових завдань із повним покриттям семантичної структури навчального матеріалу;
2) вдосконалити інформаційну модель семантичної структури навчального курсу для забезпечення можливості з достатньою для генерації тестових завдань інформативністю виконувати формальне подання навчальних матеріалів;
3) розробити метод автоматизованої генерації тестових завдань до навчальних матеріалів за допомогою правил продукції;
4) розробити інформаційну технологію автоматизованої генерації тестових завдань за допомогою отриманих моделі та методу;
5) розробити інформаційну систему генерації тестових завдань на основі розробленої інформаційної технології;
6) провести прикладне дослідження ефективності інформаційної технології генерації тестових завдань шляхом дослідження функціональності й ефективності застосування відповідної експериментальної інформаційної системи.


Інформаційна модель семантичної структури навчального курсу
6
Інформаційний навчальний матеріал
Тема 1
Лекція 1
Ключовий термін 1
Ключовий термін N
Тема N
Лекція 1
Ключовий термін 1
Ключовий термін N
Лекція N
Тестовий навчальний матеріал
Тестове завдання 1
Тестове завдання 2
Тестове завдання 3
Тестове завдання ...
Тестове завдання N


Інформаційна модель семантичної структури навчального курсу
7
Інформаційний навчальний матеріал
Тема 1
Лекція 1
Ключовий термін 1
Речення 1
Речення 2
Ключовий термін 2
Речення 3
Речення 4
Ключовий термін 3
Речення 5
Речення 6
перевіряє
Тестове завдання
використовує

Інформаційна модель семантичної структури навчального курсу

8

$$C = I \cup T = H \cup R_1 \cup K \cup R_2 \cup S \cup R_3 \cup Q \cup R_4 \cup R_5$$

C – навчальний курс
I – інформаційний навчальний матеріал
T – тестовий навчальний матеріал
H – множина заголовків
K – множина ключових термінів
S – множина речень
R_1 – множина зв'язків між заголовками
R_2 – множина зв'язків між ключовими термінами та реченнями
R_3 – множина зв'язків між заголовками та реченнями
R_4 – множина зв'язків між тестовим завданнями та реченнями
R_5 – множина зв'язків між тестовим завданнями та ключовими термінами

Метод автоматизованої генерації тестових завдань

9

Продукціна модель є однією із моделей представлення знань, яка дозволяє представляти знання у вигляді правил, що складаються із антецедентів (набір умов) та консеквентів (набір дій).

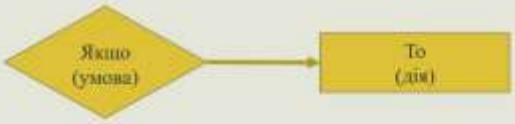
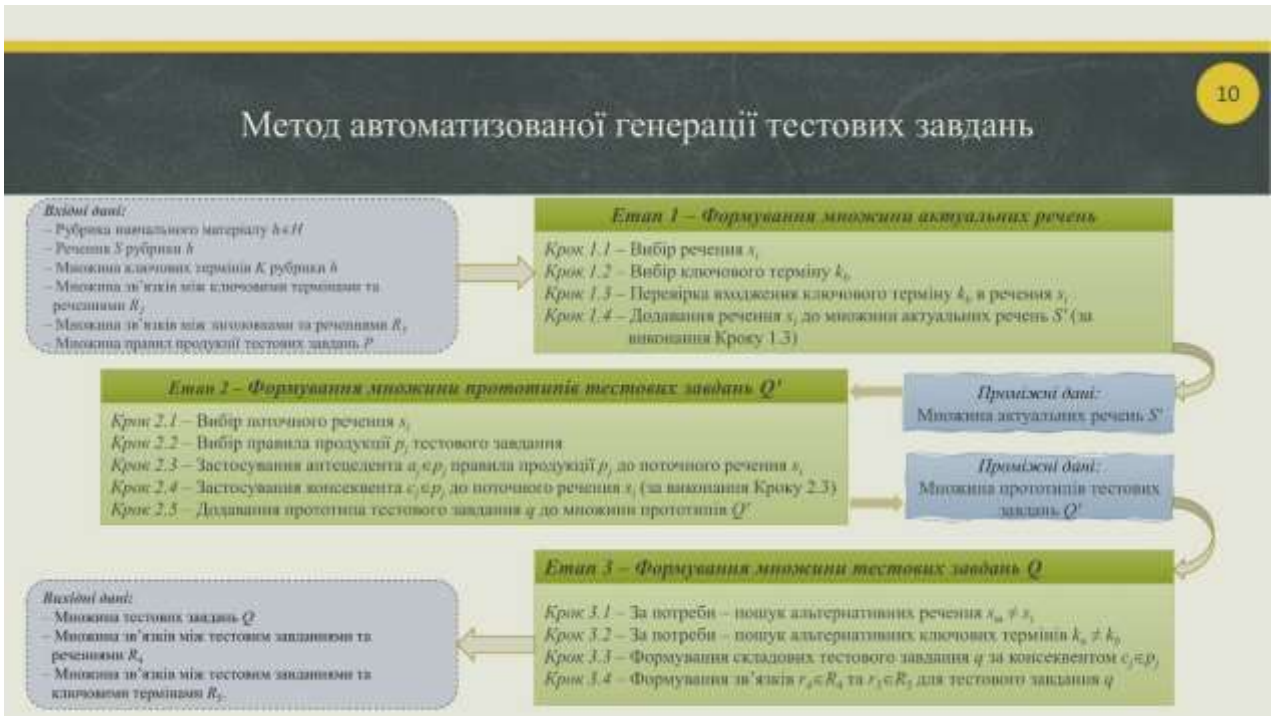
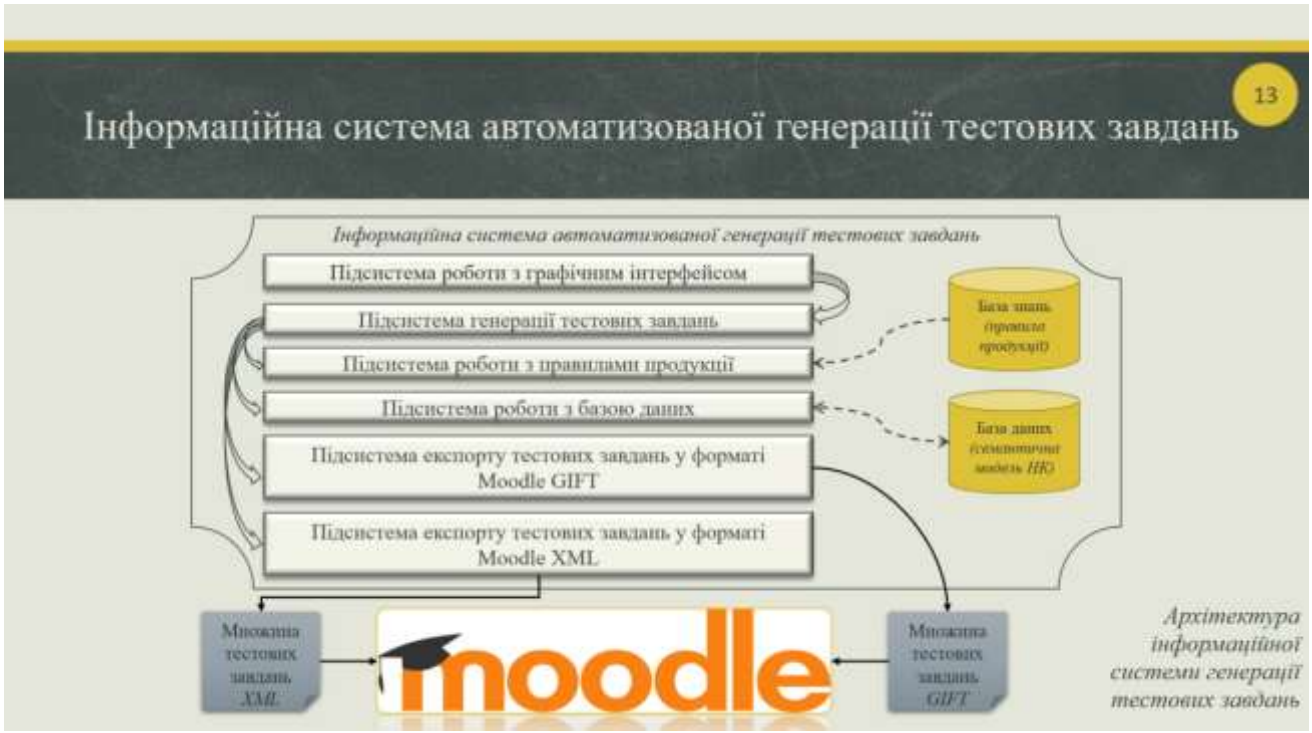


Схема правила продукції для генерації тестового завдання


Метод автоматизованої генерації тестових завдань
10
Вхідні дані:
– Рубрика навчального матеріалу h∈H
– Речення S рубрики h
– Множина ключових термінів K рубрики h
– Множина зв'язків між ключовими термінами та реченнями R2
– Множина зв'язків між заголовками та реченнями R3
– Множина правил продукцій тестових завдань P
Етап 1 – Формування множини актуальних речень
Крок 1.1 – Вибір речення si
Крок 1.2 – Вибір ключового терміну kj
Крок 1.3 – Перевірка входження ключового терміну kj в речення si
Крок 1.4 – Додавання речення si до множини актуальних речень S' (за виконання Кроку 1.3)
Проміжні дані:
Множина актуальних речень S'
Етап 2 – Формування множини прототипів тестових завдань Q'
Крок 2.1 – Вибір поточного речення si
Крок 2.2 – Вибір правила продукції pj тестового завдання
Крок 2.3 – Застосування антецедента aj∈pj правила продукції pj до поточного речення si
Крок 2.4 – Застосування консеквента cj∈pj до поточного речення si (за виконання Кроку 2.3)
Крок 2.5 – Додавання прототипа тестового завдання q до множини прототипів Q'
Проміжні дані:
Множина прототипів тестових завдань Q'
Етап 3 – Формування множини тестових завдань Q
Крок 3.1 – За потреби – пошук альтернативних речення sm ≠ si
Крок 3.2 – За потреби – пошук альтернативних ключових термінів kn ≠ kj
Крок 3.3 – Формування складових тестового завдання q за консеквентом cj∈pj
Крок 3.4 – Формування зв'язків r4∈R4 та r5∈R5 для тестового завдання q
Вихідні дані:
– Множина тестових завдань Q
– Множина зв'язків між тестовим завданнями та реченнями R4
– Множина зв'язків між тестовим завданнями та ключовими термінами R5


Інформаційна технологія автоматизованої генерації тестових завдань
11
1 Пошук речень, які містять ключові терміни
2 Застосування антецедента для пошуку речень, за якими можна згенерувати тестові завдання
3 Застосування консеквента для генерації прототипів тестового завдання тестових завдань
4 Створення дистракторів та доповнень до тестового завдання
5 Формування метаданих до згенерованого тестового завдання
Наступне правило
Наступний прототип тестового завдання
Наступна рубрика


Інформаційна система автоматизованої генерації тестових завдань
12
Зчитування продукційних правил
Завантаження вхідних даних у вигляді семантичної структури ІНМ
Редагування тестових завдань
Експорт тестових завдань в середовище для тестування
Функції інформаційної системи автоматизованої генерації тестових завдань
Генерація тестових завдань з однією правильною відповіддю
Генерація тестових із варіантами відповіді «Так» або «Ні»
Генерація тестових завдань із можливістю введення відповіді
Генерація тестових завдань із декількома правильними відповідями


Інформаційна система автоматизованої генерації тестових завдань
13
Інформаційна система автоматизованої генерації тестових завдань
Підсистема роботи з графічним інтерфейсом
Підсистема генерації тестових завдань
Підсистема роботи з правилами продукції
Підсистема роботи з базою даних
Підсистема експорту тестових завдань у форматі Moodle GIFT
Підсистема експорту тестових завдань у форматі Moodle XML
Множина тестових завдань XML
moodle
Множина тестових завдань GIFT
Архітектура інформаційної системи генерації тестових завдань

Інформаційна система автоматизованої генерації тестових завдань

14

Дослідження покриття семантичної структури навчального матеріалу згенерованими тестовими завданнями

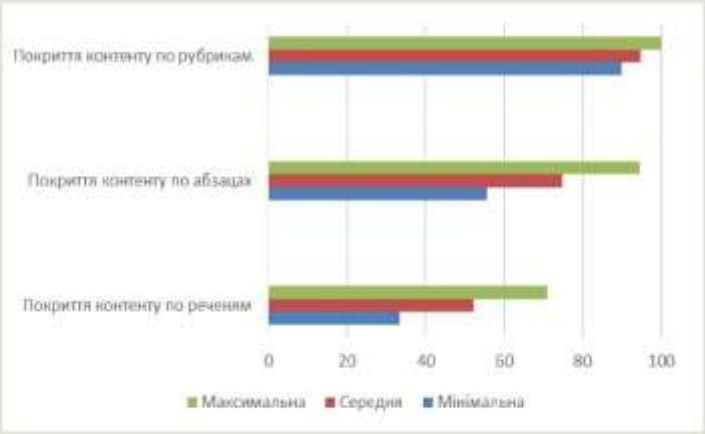
15

Граничні показники T покриття семантичних блоків інформаційного навчального матеріалу склали:

- для рубрик: $T_{H,min} = 89.93\%$, $T_{H,max} = 100\%$
- для абзаців: $T_{P,min} = 55.68\%$, $T_{P,max} = 94.37\%$
- для речень: $T_{S,min} = 33.52\%$, $T_{S,max} = 71.12\%$

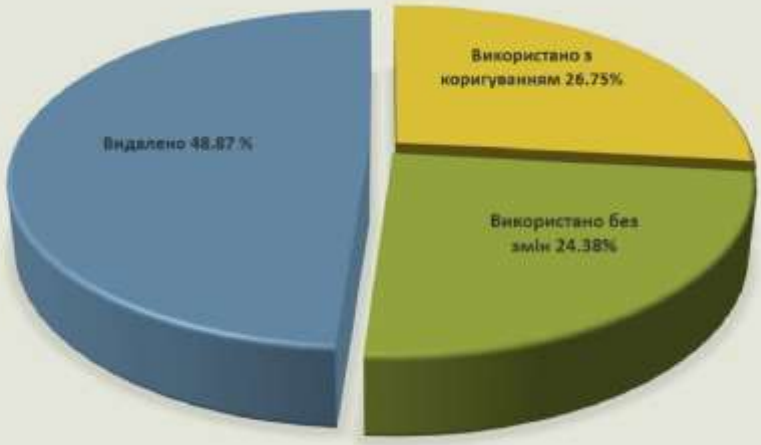
Середні показники $\overline{T}$ покриття семантичних блоків інформаційного навчального матеріалу склали:

- для рубрик: $\overline{T_H} = 94.72\%$
- для абзаців: $\overline{T_P} = 74.87\%$
- для речень: $\overline{T_S} = 52.23\%$



Дослідження шляхів використання автоматизовано згенерованих тестових завдань

16



Висновки

17

За результатами виконання дипломної роботи магістра було вирішено всі поставлені завдання, що дало можливість досягнути мету роботи – розроблено інформаційну технологію автоматизованої генерації тестових завдань до інформаційних навчальних матеріалів, в основі якого покладено метод автоматизованої генерації тестових завдань за допомогою продукційних правил.

В результаті експериментального дослідження розробленої інформаційної технології встановлено, що із всієї вибірки автоматизовано згенерованих тестових завдань без коригування в результуючу вибірку розробниками включено у середньому 24.38% тестових завдань, 26.75% було проредаговано, 48.87% – видалено. Середній рівень покриття семантичної структури інформаційного навчального матеріалу множиною згенерованих тестових завдань складає 94.72% для рубрик, 74.87% для абзаців та 52.23% для речень. Також встановлено, що розроблені модель, метод та інформаційна технологія дозволяють отримати наступний ефект:

– можливість використовувати згенеровані тестові завдання для перевірки рівня знань визначених ключових термінів у межах визначених рубрик ІНМ;

– можливість контролю використання тестових завдань у тестуванні залежно від обсягу наявних у контенті термінів різних рівнів семантичної значущості;

– можливість використання створених множин тестових завдань для адаптивного тестування.

18

Отримані положення наукової новизни

✓ Вдосконалено **інформаційну модель** семантичної структури навчального курсу, яка відрізняється тим, що містить подання навчальних матеріалів у вигляді системи рубрикації, множини ключових термінів до кожної із рубрик та тестових завдань для перевірки ключових термінів у визначених місцях їх використання.

✓ Розроблено новий **метод** автоматизованої генерації тестових завдань до навчальних матеріалів, що дозволяє генерувати тестові завдання за контентом навчального матеріалу на основі правил продукції.

✓ Розроблено нову **інформаційну технологію** автоматизованої генерації тестових завдань, що дозволяє за поданням семантичної структури інформаційного навчального матеріалу одержувати множину тестових завдань, призначених для перевірки рівня знань визначених термінів.

✓ Розроблено нову **інформаційну систему** генерації тестових завдань, що за створеною інформаційною технологією дозволяє генерувати та експортувати в середовище тестування множини тестових завдань.

19

Апробація результатів дипломної роботи магістра та публікації

Публікації

Основні наукові положення роботи автором висвітлено в 5 наукових публікаціях, серед яких 1 в фаховому виданні, включеному в перелік МОН України:

1. Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В Ковальчук, В. О. Слободзян // Вісник Хмельницького національного університету. Хмельницький – 2018. – С.61-69.
2. Ковальчук О. В. Використання програмного розширення Sprint.Doc для автоматизації роботи з цифровими документами / О. В. Ковальчук, Г. А. Білоус., В.О. Слободзян // Збірник наукових праць за матеріалами XI всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2019». Хмельницький – 2019 – с. 116-122.
3. Ковальчук О. В. Дослідження практичної ефективності інформаційної технології автоматизованого визначення семантичних термінів навчальних матеріалів / О.В. Ковальчук, О.В. Мазурець. // Сучасні технології в механіці : зб. наук. пр., Хмельницький – 2018 – с. 141-148.
4. Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В Ковальчук, В. О. Слободзян, Г. А. Білоус. // Інтелектуальний потенціал – 2018 : збірник наукових праць молодих науковців і студентів, Хмельницький – 2018 – с. 51-56.
5. Ковальчук О. В. Метод генерації тестових завдань до навчальних матеріалів на основі продукційних правил / О. В Ковальчук, О. В. Мазурець. // Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020». Хмельницький – 2020 – с. 51-56.

Конференції

Основні наукові та практичні результати доповідалися на наступних 4 науково-практичних конференціях:

1) Міжнародній конференції молодих науковців «Сучасні технології в механіці – 2018» (19-21 квітня 2018 року);
2) Всеукраїнській науково-практичній конференції молодих науковців і студентів «Інтелектуальний потенціал – 2018» (14-16 листопада 2018 року);
3) XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» (14-15 листопада 2019 року);
4) XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (9-10 листопада 2020 року).

Wed Dec 02 11:38:23 EET 2020, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 82017 Назва: Інформаційна технологія автоматизованої генерації тестових завдань до навчальних матеріалів Додано в БД: 2020-12-02 Автора: Ковальчук Олексій Володимирович Керівники: Мазурець О.В. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	103568	915	3477 (3%)	43 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ КАФЕДРИ

КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: **Інформаційна технологія автоматизованої генерації тестових завдань до навчальних матеріалів**

Автор: **Ковальчук Олексій Володимирович**

Спеціальність: **122 Комп'ютерні науки**

Науковий керівник: **ст. викладач Мазурець Олександр Вікторович**

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	-
3	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	-
4	Інше:	-

Підтвердження: Показник збігів є незначним і складає 6.36% (найбільша схожість 0.67% з одним джерелом). Виявлені в роботі запозичення є законними і не є плагіатом, оскільки стосуються огляду існуючих рішень, мають відповідні посилання на джерела у Переліку посилань і розміщені в розділах, які не описують безпосередньо авторське дослідження. Випадки віднесення до плагіату прикладів вхідних даних інформаційних систем та елементів бібліографічного опису джерел у Переліку посилань є природніми і не є плагіатом. Робота приймається до захисту.

02.12.2020

Дата

Підпис керівника

Підпис завідувача кафедри

Хмельницький національний університет

Кафедра комп'ютерних наук та інформаційних технологій

ВІДГУК ОПОНЕНТА

на дипломну роботу магістра

Магістра гр. КНм-19-1 Ковальчука Олексія Володимировича

На тему: *Інформаційна технологія автоматизованої генерації тестових завдань до навчальних матеріалів.*

1. Актуальність і значення теми.

Із розвитком інформаційних технологій освітня галузь отримала новітній засіб перевірки знань у вигляді комп'ютерного тестування. Викладачу достатньо сформувати набір тестових завдань на основі навчального матеріалу та заповнити банк тестових завдань, а автоматизована система тестування сформує вибірку завдань для студента та перевірить результат тестування. На відміну від процесу тестування процес створення тестових завдань є неавтоматизованим, як результат викладач змушений завжди створювати тестові завдання в ручному режимі, що робить процес створення тестових завдань важким та трудомістким. Тому є доцільною розробка інформаційної технології для автоматизованої генерації тестових завдань, розподілених за семантичною структурою навчального матеріалу, що надасть можливість використовувати згенеровані тестові завдання для перевірки рівня знань визначених ключових термінів у межах визначених рубрик інформаційного навчально матеріалу.

2.Оцінка якості та достовірності проведених досліджень.

Отримані результати добре співвідносяться з результатами, наведеними в наукових роботах. Проведені в роботі дослідження підтвердили достовірність одержаних результатів.

3. Оцінка запропонованих заходів та пропозицій, практичної цінності та ефективності.

Проведені в роботі дослідження представляють науково-технічну цінність та є ефективним дослідженням в галузі інформаційних технологій, що дає можливість використати запропоновану інформаційну технологію для автоматизованої генерації тестових завдань до навчальних матеріалів.

4. Загальний висновок та оцінка.

Дипломна робота магістра виконана в повному обсязі. Пояснювальна записка оформлена відповідно до вимог дипломних робіт. За своєю структурою, практичними цінностями, поставленій меті та вирішеними задачами робота відповідає вимогам вищої школи і вимогам, що пред'являються до освітньо-кваліфікаційного рівня «магістр». Тому автор роботи О. В. Ковальчук заслуговує присвоєння кваліфікації магістра з комп'ютерних наук.

Робота заслуговує на оцінку «відмінно».

Опонент Мартенюк В.В. к.т.н., проф.