ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова
праця на правах рукопису

МАЗУРЕЦЬ ОЛЕКСАНДР ВІКТОРОВИЧ

УДК 004.91

ДИСЕРТАЦІЯ
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АВТОМАТИЗОВАНОГО СТРУКТУРУВАННЯ НАВЧАЛЬНИХ МАТЕРІАЛІВ ТА СТВОРЕННЯ ТЕСТІВ ДЛЯ АДАПТИВНОГО КОНТРОЛЮ РІВНЯ ЗНАНЬ

05.13.06 – Інформаційні технології
05 – Технічні науки

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

О. В. Мазурець

Науковий керівник: Бармак Олександр Володимирович
доктор технічних наук, професор

Ідентичність всіх примірників дисертації
ЗАСВІДЧУЮ:
Вчений секретар спеціалізованої
вченої ради К 58.082.02

/ М.І. Комар /

Тернопіль – 2019

АНОТАЦІЯ

Мазурець О. В. Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.06 Інформаційні технології. – Хмельницький національний університет. – Тернопільський національний економічний університет, Тернопіль, 2019.

Дисертаційна робота присвячена розробці інформаційної технології для автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань. Інформаційна технологія дає змогу за вхідними даними навчальних матеріалів курсу автоматизовано одержувати множину тестових завдань.

Відомі методи та підходи до автоматизації створення тестових завдань характеризуються суттєвими вимогами до приведення інформаційних навчальних матеріалів до потрібного вигляду та структури, що вимагає додаткових витрат часу та зусиль. Проте більшість інформаційних навчальних матеріалів можуть бути автоматизовано оброблені для визначення їх семантичної структури. Побудова відповідної моделі семантичної структури для навчального курсу відкриває можливість автоматизованого створення розподілених за такою структурою інформаційного навчального матеріалу тестових завдань. Для створення тестових завдань використано продукційну модель формального представлення знань для подання правил формування граматичних конструкцій тестових завдань, характерних для ручного створення тестів.

Удосконалено інформаційну модель семантичної структури навчального курсу, яка на відміну від відомих забезпечує можливість формального подання семантичної структури навчального курсу з інформативністю, достатньою для автоматизованого створення тестових

завдань, які розподілені за структурою навчальних матеріалів. Інформаційна модель семантичної структури навчального курсу формально подає інформаційний навчальний матеріал та тестовий навчальний матеріал у вигляді множин: заголовків, ключових термінів, тестових завдань, зв'язків. До множини зв'язків належать підмножини зв'язків: між заголовками й заголовками, між заголовками й ключовими термінами, між заголовками та тестовими завданнями, між ключовими термінами та тестовими завданнями.

Удосконалено метод формування структури навчальних матеріалів та пошуку в них ключових термінів, який на відміну від відомих дозволяє врахувати можливість взаємного поглинання термінів і ідентифікувати як термін не тільки слова, а й словосполучення.

Вперше розроблено метод автоматизованого формування тестових завдань до навчальних матеріалів, який використовує продукційну модель для подання правил формування тестових завдань і дозволяє одержувати множини тестових завдань без додаткової формалізації навчальних матеріалів. Кожне правило продукції тестових завдань складається з антецеденту (формально подає речення, які потенційно придатні для створення тестових завдань) та консеквенту (формально подає алгоритми створення тестових завдань різних типів і конструкцій).

Вперше розроблено інформаційну технологію автоматизованого створення тестових завдань до навчальних матеріалів, яка дозволяє автоматизовано одержувати множину тестових завдань, що придатні для проведення адаптивного тестування, за вхідними даними у вигляді інформаційних навчальних матеріалів. Створюються тестові завдання наступних типів, що використовуються в навчальних середовищах: логічного типу, одиничного вибору, множинного вибору та завдань на введення тексту.

Проведено експериментальне тестування інформаційної технології автоматизованого створення тестових завдань до навчальних матеріалів шляхом створення й використання відповідної програмної системи. Розроблена програмна система для автоматизованого створення тестів надає

можливість створення тестових завдань для цілеспрямованої перевірки рівня знань визначених термінів у структурі навчальних матеріалів, що необхідно для адаптивного тестування, яке дозволяє визначати рівень знань меншою від існуючих алгоритмів кількістю тестових завдань, й досліджено її ефективність.

За результатами експериментального тестування інформаційної технології автоматизованого створення тестових завдань до навчальних матеріалів було визначено, що за її використання забезпечується повне покриття начального матеріалу, оскільки правила продукції застосовуються для всіх рівнів семантичної структури навчальних матеріалів. Автоматизація процесу формування тестових завдань забезпечує суттєве скорочення часу на розробку тестових завдань. Дані, що містяться у моделі (заголовки, терміни, тестові завдання, зв’язки) дають можливість проведення адаптивного контролю рівня одержаних знань.

Ключові слова: тестові завдання, тести, інформаційна технологія, дисперсійна оцінка, семантичний аналіз, ключові слова, ключові терміни.

SUMMARY

Mazurets O. V. Information Technology for Automated Structuring of Educational Materials and Creation of Tests for Adaptive Control of the Level of Knowledge. – Qualifying scientific work on the right of manuscript.

The thesis for the degree of candidate of technical sciences, specialty 05.13.06 «Information technology». – Khmelnytskiy National University. – Ternopil National Economic University. – Ternopil, 2019.

The dissertation is devoted to the development of information technology for the automated structuring of educational materials and the creation of tests for adaptive control of the knowledge level. Information technology allows to automatically receive sets of test tasks based on input data in the form of educational materials of course. The analysis of existing methods for searching key

terms in digital texts and methods of automated creation of test tasks for educational materials is implemented.

Known methods and approaches to automation of creation of test tasks are characterized by essential requirements for transformation of informational educational materials to the necessary appearance and structure, which requires additional time and effort. However, most informational educational materials can be automated processed to determine their semantic structure. Building the necessary model of the semantic structure of the educational course provides an opportunity for the automated creation of test tasks that are distributed according to such structure of the informational educational material. When creating test tasks, the production model for the formal representation of knowledge was used to represent the rules for creating grammatical constructs of test tasks, which are characteristic of the process of manual creation of tests.

The information model of the semantic structure of the educational course has been improved, which, unlike the known ones, provides a formal presentation of the semantic structure of the educational course. The semantic structure of the educational course is informative enough to automate the creation of test tasks, which are distributed by the structure of the educational materials. The information model of the semantic structure of educational course formally provides informational educational material and testing educational material in the form of sets: headings, key terms, test tasks, relations. The set of relations include subsets of relations: between headings and headings, between headings and key terms, between headings and test tasks, between key terms and test tasks.

The method of forming the structure of educational materials and searching for key terms in them has been improved. This method, unlike the known ones, allows accounting the possibility of mutual absorption of terms and identifying as terms not only words but also phrases.

New method for the automated creation of test tasks for educational materials has been developed, which uses a production model to represent the rules for the creation of test tasks. The method allows receiving test tasks sets without

further formalization of educational materials. Each test task production rule consists of an antecedent (formally submits sentences that are potentially suitable for creating test tasks) and a consequence (formally submits algorithms for creating test tasks of different types and designs).

New information technology for automated creation of test tasks for educational materials has been developed, which allows receiving automatically sets of test tasks according to the input data in the form of educational materials. The resulting sets of test tasks are suitable for adaptive testing. Test tasks are created for the following types used in educational environments: logical type, single choice, multiple choice and text input tasks.

Experimental testing of information technology for automated creation of test tasks for educational materials was carried out by creating and using the corresponding software system. The developed software system for automated test creation provides the ability to create test tasks to purposefully determine the level of knowledge of certain key terms in the structure of educational materials. This is necessary for adaptive testing, which makes it possible to determine the level of knowledge by the number of test tasks, which is less than in the existing algorithms, and investigated its effectiveness.

According to the results of experimental testing of information technology of automated creation of test tasks for educational materials, it was determined that full coverage of the educational material is ensured, since the production rules apply to all levels of the semantic structure of educational materials. Automation of the process of creation of test tasks provides a significant reduction of time for the development of test tasks. The data contained in the model (headings, key terms, test tasks, relations) make it possible to carry out adaptive control of the level of knowledge.

Keywords: test tasks, tests, information technology, disperse evaluation, semantic analysis, keywords, key terms.

Список публікацій здобувача

Статті у виданнях, зареєстрованих в наукометричній базі Scopus:

1. Krak I., Barmak O., Mazurets O. The practice investigation of the information technology efficiency for automated definition of terms in the semantic content of educational materials. CEUR Workshop Proceedings. 2016. Vol. 1631. P. 237–245.

2. Krak I., Barmak O., Mazurets O. The practice implementation of the information technology for automated definition of semantic terms sets in the content of educational materials. CEUR Workshop Proceedings. 2018. Vol. 2139. P. 245–254.

3. Barmak O., Krak I., Mazurets O., Pavlov S., Smolarz A., Wojcik W. Research of efficiency of information technology for creation of semantic structure of educational materials. Advances in Intelligent Systems and Computing. 2020. Vol. 1020. P. 554–569.

Статті у фахових виданнях:

4. Бармак О. В., Мазурець О. В. Методи автоматизації визначення семантичних термінів у навчальних матеріалах. Вісник Хмельницького національного університету. Серія : Технічні науки. 2015. № 2. С. 209–213.

5. Бармак О. В., Мазурець О. В. Інформаційна технологія автоматизованого визначення термінів у навчальних матеріалах. Вимірювальна та обчислювальна техніка в технологічних процесах. 2015. № 2. С. 94–102.

6. Бармак О. В., Мазурець О. В., Матвійчук А. О. Застосування інформаційної технології гнучкого тестування рівня знань у середовищі Moodle. Вісник Хмельницького національного університету. Серія : Технічні науки. 2017. № 2. С. 103–114.

7. Бармак О. В., Мазурець О. В., Кліменко В. І. Інформаційна технологія автоматизованого формування тестових завдань. Вісник Хмельницького національного університету. Серія : Технічні науки. 2017. № 5. С. 93–103.

8. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2017. № 6. С. 223–229.

9. Мазурець О. В., Ковальчук О. В., Слободзян В. О. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 1. С. 61–69.

10. Мазурець О. В. Інформаційна технологія автоматизованого визначення семантичних термінів в елементах навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 3. С. 223–230.

11. Мазурець О. В. Розробка множини тегів для формального опису елементів моделей автоматизованого формування тестових завдань. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 5. С. 26–31.

12. Крак Ю. В., Бармак О. В., Мазурець О. В. Практична реалізація інформаційної технології автоматизованого визначення множини семантичних термінів в контенті навчальних матеріалів. Проблеми програмування. 2018. № 2–3. С. 245–254.

13. Бармак О. В., Мазурець О. В. Інформаційна модель семантичної структури навчального курсу. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 6. Т. 1. С. 92–97.

14. Мазурець О. В. Інформаційна технологія автоматизованого створення тестів до навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2019. № 4. С. 84–91.

15. Мазурець О. В. Метод автоматизованого формування тестових завдань. Вісник Хмельницького національного університету. Серія : Технічні науки. 2019. № 5. С.189–194.

Патенти на корисну модель:

16. Мазурець О. В. Пат. на корисну модель 129903. Україна, МПК G06F 17/00. Спосіб визначення переліку ключових слів у тексті. № u201707242; заявл. 10.07.2017; опубл. 26.11.2018, Бюл. № 22.

17. Мазурець О. В. Пат. на корисну модель 137386. Україна, МПК G06F 17/00. Спосіб обмеження переліку ключових слів тексту. № u201900552; заявл. 18.01.2019; опубл. 25.10.2019, Бюл. № 20.

Опубліковані праці апробаційного характеру:

18. Мазурець О. В. Інформаційна технологія побудови онтологічної моделі навчального курсу для оцінювання отриманих знань. Інформаційні управляючі системи та технології : матеріали III Міжнародної науково-практичної конференції. (м. Одеса, 23–25 вересня 2014 р.). Одеса, 2014. С. 81–83.

19. Бармак О. В., Крак Ю. В., Мазурець О. В. Інформаційна технологія автоматизованого визначення термінів у навчальних матеріалах. Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту : матеріали Міжнародної наукової конференції. (с. Залізний Порт, 25–28 травня 2015 р.). Херсон, 2015. С. 26–28.

20. Мазурець О. В. Особливості застосування інформаційної технології автоматизованого визначення термінів у навчальних матеріалах. Інформаційні управляючі системи та технології : матеріали IV Міжнародної науково-практичної конференції. (м. Одеса, 22–24 вересня 2015 р.). Одеса, 2015. С. 62–65.

21. Мазурець О. В. Дослідження ефективності інформаційної технології автоматизованого визначення термінів у навчальних матеріалах. ICACIT-2015 : матеріали III Міжнародної конференції з автоматичного управління та інформаційних технологій (м. Київ, 11–13 грудня 2015 р.). Київ, 2015. С. 136–139.

22. Кліменко В. І., Мазурець О. В. Аналіз сучасних методів генерації тестових завдань. Актуальні проблеми комп'ютерних технологій : збірник наукових праць за матеріалами X міжнародної науково-технічної конференції. (м. Хмельницький, 31 травня 2016 р.). Хмельницький, 2016. С. 77–84.

23. Крак Ю. В., Бармак О. В., Мазурець О. В. Практичне дослідження ефективності інформаційної технології автоматизованого визначення семантичних термінів в контенті навчальних матеріалів. Прикладне програмне забезпечення УкрПРОГ'2016 : матеріали X Міжнародної науково-практичної конференції по програмуванню. (м. Київ, 24–26 травня 2016 р.). Київ, 2016. С. 237–245.

24. Мазурець О. В. Особливості автоматизованого формування тестових завдань. Інформаційні управляючі системи та технології : матеріали V Міжнародної науково-практичної конференції. (м. Одеса, 20–22 вересня 2016 р.). Одеса, 2016. – С. 71-73.

25. Мазурець О. В., Якимюк О. М. Моделі оцінки ефективності методів пошуку ключових термінів у контенті навчальних матеріалів. Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи : матеріали Всеукраїнської науково-практичної конференції з міжнародною участю. (м. Тернопіль, 9–10 листопада 2017 р.). Тернопіль, 2017. С. 258–261.

26. Мазурець О. В. Інформаційна технологія гнучкого тестування рівня знань у середовищі MOODLE. Інформаційні управляючі системи та технології : матеріали VI Міжнародної науково-практичної конференції. (м. Одеса, 20–22 вересня 2017 р.). Одеса, 2017. С. 83–85.

27. Крак Ю. В., Бармак О. В., Мазурець О. В. Практична реалізація інформаційної технології автоматизованого визначення множини семантичних термінів в контенті навчальних матеріалів. Прикладне програмне забезпечення УкрПРОГ'2018 : матеріали XI Міжнародної науково-практичної конференції по програмуванню. (м. Київ, 22–24 травня 2018 р.). Київ, 2018. С. 245–254.

28. Мазурець О. В. Використання множини тегів для формального опису моделей формування тестових завдань. Інформаційні управляючі системи та технології : матеріали VII Міжнародної науково-практичної конференції. (м. Одеса, 17–18 вересня 2018 р.). Одеса, 2018. С. 69–72.

29. Бармак О. В., Мазурець О. В. Дослідження точності та повноти автоматизованого визначення семантичних термінів у навчальних матеріалах. Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи : матеріали III Міжнародної науково-практичної конференції. (м. Тернопіль, 5 квітня 2019 р.). Тернопіль, 2019. С. 98–101.

30. Крак Ю. В., Бармак О. В., Мазурець О. В. Інформаційна модель семантичної структури навчального курсу для генерації тестових завдань. Моделювання та дослідження стійкості динамічних систем : матеріали XIX Міжнародної науково-практичної конференції. (м. Київ, 22–24 травня 2019 р.). Київ, 2019. С. 365–367.

31. Мазурець О. В. Застосування продукційної моделі для автоматизованої генерації тестових завдань. Інформаційні управляючі системи та технології : матеріали VIII Міжнародної науково-практичної конференції. (м. Одеса, 23–25 вересня 2019 р.). Одеса, 2019. С. 52–54.

ЗМІСТ

Стор.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ІНМ (ІЕМ)	Інформаційний навчальний матеріал (informational education material)
ІТ	Інформаційна технологія
НК (ЕС)	Навчальний курс (educational course)
ТНМ (ТЕМ)	Тестовий навчальний матеріал (testing education material)
DE	Disperse evaluation (дисперсійна оцінка)
IDF	Inverse document frequency (зворотна документарна частота)
IST	Information Society Technologies
HVG	Horizontal Visibility Graph (граф горизонтальної видимості)
LMS	Learning Management System (Система керування навчанням)
MS Office	Microsoft Office
Moodle	Modular Object-Oriented Dynamic Learning Environment (модульне об'єктно-орієнтоване динамічне навчальне середовище)
SEO	Search Engine Optimization (пошукова оптимізація)
TF	Term frequency (частотна оцінка)
XML	eXtensible Markup Language (розширювана мова размітки)
XHTML	eXtensible HyperText Markup Language (розширювана мова гіпертекстової размітки)

ВСТУП

Дисертаційна робота присвячена розробці ІТ для автоматизованого формування тестових завдань до ІНМ, яка дозволяє за вхідними даними ІНМ курсу автоматизовано одержувати множину тестових завдань для цілеспрямованої перевірки рівня знань визначених термінів у структурі ІНМ, що необхідно для адаптивного тестування.

Актуальність роботи. У галузі сучасної вищої освіти потенційна якість отриманих освітніх послуг прямо залежить від якості ІНМ та ефективності засобів контролю рівня засвоєних знань. Важливу роль у розв’язанні проблеми ефективного контролю рівня засвоєних знань, що постає з розвитком нових технологій і підвищенням ступеня інформатизації суспільства й освіти, відіграють комп’ютерні засоби перевірки знань. Одним із основних способів контролю знань у навчальних інформаційних системах залишається комп’ютерне тестування.

За умов вузької спеціалізації курсів навчальних дисциплін, їх чисельності та інтенсивного оновлення застосування ІТ дає змогу зменшити затрати часу на створення тестового навчального матеріалу (ТНМ) за допомогою автоматизації процесу його створення, а також забезпечити можливість використання створених тестових завдань для цілеспрямованої перевірки рівня знань визначених термінів у визначених елементах структури ІНМ, що необхідно для адаптивного тестування й актуалізує наведений напрямок наукових досліджень.

Для вирішення поставлених задач необхідні семантичний аналіз контенту ІНМ та автоматизація процесу створення тестових завдань. Значний внесок у напрямок семантичного аналізу цифрових текстів внесли Анісімов А. В., Бармак О. В., Бісікало О. В., Бродер А., Виноград Т. В., Гальперін І. Р., Греймас А. Ж., Кобозева І. М., Крак Ю. В., Кронгауз М. А., Лакофф Дж., Ланде Д. В., Леонтьєв Н. Н., Лунн Х. П., Люгер Д. Ф., Маккарті Д., Ньюелл А., Піттс У., Попов Е. В., Поспєлов Д. А., Рубашкін В. Ш., Саченко А. О., Севбо І.

П., Фостер Д. М., Хопкрофт Дж., Шемакін Ю. І., Широков В. А. та інші. Вагомий внесок у напрямок автоматизації тестування, розробки та застосування навчальних і тестувальних середовищ зробили Аванесов В. С., Башмаков І. А., Клайн П., Литвиненко В. І., Мельник А. М., Пасічник В. В., Пасічник Р. М., Рафальська О. О., Снитюк В. Е., Теленик С. Ф., Титенко С. В., Томашевський В. М., Тонконогий В. М., Федусенко О. В, Федорук П. І. й інші.

Існуючі методи та підходи щодо автоматизації створення тестових завдань характеризуються суттєвими вимогами до приведення ІНМ до потрібного вигляду та структури, що вимагає додаткових витрат часу та зусиль. Такий підхід актуальний для ІНМ, які важко піддаються автоматизованому семантичному аналізу. Проте більшість ІНМ можуть бути автоматизовано оброблені для визначення їх семантичної структури. Побудова відповідної моделі семантичної структури для навчального курсу відкриває можливість автоматизованого створення розподілених за такою структурою навчального матеріалу тестових завдань. Для створення тестових завдань використано продукційну модель формального представлення знань для подання правил формування граматичних конструкцій тестових завдань, характерних для ручного створення тестів.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконана на кафедрі комп’ютерних наук та інформаційних технологій Хмельницького національного університету. Зміст роботи, її основні завдання відповідають державним науково-технічним програмам, які було сформульовано в законах України «Про наукову і науково-технічну діяльність», «Про Національну програму інформатизації», а також планам найважливіших науково-дослідних робіт Міністерства освіти і науки України, п.6.2.2. «Перспективні інформаційні технології і системи». Дисертаційна робота є складовою частиною досліджень, що виконувались за держбюджетною темою 1Б-2019 «Агентно-орієнтована система підвищення безпеки та якості програмного забезпечення комп’ютерних систем» (номер

державної реєстрації 0119U100662) у Хмельницькому національному університеті, в якій здобувач був виконавцем окремих розділів.

Мета і задачі роботи. Мета роботи полягає в розробці ІТ, що уможливить автоматизоване створення тестових завдань, розподілених за структурою ІНМ. Для досягнення поставленої мети визначені наступні задачі дослідження:

1) провести аналіз існуючих методів пошуку ключових термінів у цифрових текстах і методів автоматизованого формування тестових завдань до ІНМ;

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

3) удосконалити метод формування структури ІНМ та пошуку у них ключових термінів, включаючи словосполучення, в ієрархічному розрізі рубрик ІНМ;

4) розробити метод автоматизованого формування тестових завдань до ІНМ за допомогою правил продукції, який не вимагатиме додаткової формалізації ІНМ;

5) розробити ІТ автоматизованого створення тестових завдань за допомогою отриманих моделей та методів;

6) виконати експериментальну перевірку ІТ автоматизованого створення тестових завдань.

Об'єкт дослідження – процес створення тестових завдань із використанням інформаційних технологій.

Предмет дослідження – моделі, методи, підходи та засоби інформаційної технології автоматизованого створення тестових завдань.

Методи дослідження. Для розв'язання поставлених задач використовуються основні положення методів аналізу даних, теорії графів, теорії множин. Для визначення ключових слів використано метод дисперсійної оцінки; для формування тестових завдань використано метод

подання знань за допомогою правил продукції; для реалізації ІТ – методології проектування інформаційних систем та об’єктно-орієнтований підхід.

Наукова новизна одержаних результатів полягає в наступному:

1. Удосконалено інформаційну модель семантичної структури навчального курсу, яка на відміну від відомих забезпечує можливість формального подання семантичної структури навчального курсу з інформативністю, достатньою для автоматизованого створення розподілених за структурою ІНМ тестових завдань.

2. Удосконалено метод формування структури навчальних матеріалів та пошуку в них ключових термінів, який на відміну від відомих дозволяє врахувати можливість взаємного поглинання термінів і ідентифікувати як термін не тільки слова, а й словосполучення.

3. Вперше розроблено метод автоматизованого формування тестових завдань до навчальних матеріалів, який використовує продукційну модель для подання правил формування тестових завдань і дозволяє одержувати множини тестових завдань без додаткової формалізації ІНМ;

4. Вперше розроблено ІТ автоматизованого створення тестових завдань до навчальних матеріалів, яка дозволяє за вхідними даними ІНМ автоматизовано одержувати множини тестових завдань, що придатні для проведення адаптивного тестування.

Практичне значення одержаних результатів. У результаті виконаного дисертаційного дослідження за ІТ розроблено відповідне експериментальне програмне забезпечення, яке підтвердило вірність запропонованих наукових положень. Застосування ІТ дає можливість за обраним цифровим документом з ІНМ: автоматизовано визначати їхню структуру і формувати впорядковані за семантичною важливістю множини ключових термінів та автоматизовано створювати множини тестових завдань для цілеспрямованої перевірки рівня знань у структурі ІНМ при адаптивному тестуванні.

Отримані результати досліджень покладено в основу експериментальних програмних застосувань, що ілюструють можливості ІТ. Програмне забезпечення використовувалось у роботі Навчального простору «VECTOR» (м. Хмельницький, акт про впровадження від 01.04.2019 р.); в Приватному малому підприємстві «Лінк» (м. Хмельницький, акт про впровадження від 15.04.2019 р.), в Товаристві з обмеженою відповідальністю “Науково-технічна фірма «Інфосервіс»” (м. Хмельницький, акт про впровадження від 07.05.2019 р.); у навчальному процесі Хмельницького національного університету при викладанні дисциплін «Методи та системи штучного інтелекту», «Проектування інформаційних систем» та «Інтелектуальний аналіз даних» (м. Хмельницький, акт про впровадження від 22.05.2019 р., ХНУ).

Особистий внесок здобувача. Усі наукові і практичні результати дисертаційної роботи одержані автором самостійно й опубліковані [1-31], зокрема, в одноосібно підготовлених працях: ІТ автоматизованого створення тестів до ІНМ [14]; інформаційна модель семантичної структури навчального курсу [8]; метод формування структури ІНМ та пошуку в них ключових термінів [10]; метод автоматизованого формування тестових завдань до ІНМ [11, 15].

У друкованих працях, опублікованих у співавторстві, автору належать основні ідеї, теоретична та практична розробка положень, відображених у характеристиці наукової новизни отриманих результатів, а саме: дослідження методів пошуку ключових слів та термінів [4]; інформаційна модель семантичної структури навчального курсу [13]; метод формування структури ІНМ та пошуку в них ключових термінів [1, 2, 3, 5]; метод автоматизованого формування тестових завдань до ІНМ [6]; ІТ автоматизованого формування тестових завдань до ІНМ [7]; програмне забезпечення за ІТ автоматизованого формування тестових завдань до ІНМ [9, 12]. Отримані патенти на корисну модель [16, 17].

Апробація результатів роботи та публікації. Основні наукові та практичні результати доповідалися та обговорювалися на міжнародних та всеукраїнських наукових конференціях [18-31], зокрема: «Інформаційні управляючі системи та технології (ICST-ODESSA)» (Одеса, 2014–2019); «Конференція з автоматичного управління та інформаційних технологій (ICACIT)» (Київ, 2015); «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту» (Херсон, 2015); «Актуальні проблеми комп'ютерних технологій (АПКТ)» (Хмельницький, 2016); «Укр'ПРОГ» (Київ, 2016, 2018); «Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи» (Тернопіль, 2017, 2019); «Моделювання та дослідження стійкості динамічних систем (DSMSI)» (Київ, 2019).

Результати дисертаційних досліджень доповідались на наукових семінарах кафедри комп'ютерних наук та інформаційних технологій Хмельницького національного університету (2014–2019рр.).

Публікації результатів. Основні положення і результати дисертації були опубліковані в 31 науковій праці, зокрема: 3 публікації – у виданнях зареєстрованих у наукометричній базі Scopus, 12 статей – у фахових виданнях (5 з яких є одноосібними), 2 патенти на корисну модель та 14 публікацій – у матеріалах конференцій.

Структура та обсяг роботи. Дисертаційна робота складається з анотації, вступу, 4 розділів, висновків, списку використаних джерел із 181 найменування на 20 сторінках та 7 додатків. Загальний обсяг дисертації становить 217 сторінок, з них 151 сторінка основного тексту. Дисертація містить 49 рисунків та 17 таблиць.

РОЗДІЛ 1.
ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ ВИЗНАЧЕННЯ РІВНЯ ЗАСВОЄНИХ ЗНАНЬ

У галузі сучасної вищої освіти потенційна якість отриманих освітніх послуг залежна від якості навчальних матеріалів та ефективності засобів контролю рівня засвоєних знань. Важливу роль у вирішенні проблеми ефективного контролю рівня засвоєних знань, яка постає з розвитком нових технологій і підвищенням ступеня інформатизації суспільства й освіти, відіграють комп'ютерні засоби перевірки знань. Одним із основних способів контролю знань в навчальних інформаційних системах залишається комп'ютерне тестування [32, 33].

В умовах вузької спеціалізації курсів навчальних дисциплін, їх чисельності та інтенсивного оновлення, єдиним шляхом забезпечення ефективного тестового контролю рівня засвоєних знань є застосування інформаційних технологій. Інформаційні технології дають можливість суттєво зменшити трудові затрати на створення тестових завдань з можливістю їх постійного оновлення, що формує актуальний напрямок наукових досліджень.

1.1 Семантична структура навчального курсу

1.1.1 Підходи до побудови семантичної структури

Різноманітним аспектам побудови семантичної структури навчального курсу присвячено значну кількість праць українських і закордонних авторів: Пасічника В. В., Теленика С. Ф., Рафальської О. О., Снитюка В. Е., Титенка С. В., Тонконогого В. М., Федусенко О. В, Федорука П. І., Langmann R., Ngoc Vu Huy, Zinovatna S. та інших. Проблема побудови структури навчального курсу відноситься до педагогічної галузі [34, 35], проте формальне подання

семантичної структури навчального курсу дозволяє вирішити широке коло проблем в освіті засобами інформаційних технологій.

Формалізація в більшості випадків виконується шляхом розробки відповідної моделі [36, 37]. На сучасному етапі особливу актуальність одержали питання побудови моделей навчальних курсів для дистанційної освіти та індивідуального підходу до навчання [38, 39, 40, 41]. Структура є природною властивістю навчального матеріалу як структурованого тексту [42, 43], відповідно, будь-яка модель формалізації навчального матеріалу повинна відобразити його структурність [44].

Серед способів структурування навчальної інформації виділяють: метод проектів (Дж. Дьюї), методи модульного (Дж. Рассел) і програмованого (Б. Ф. Скіннер, Н. Краудер) навчання, методи опорних конспектів (В. Ф. Шаталов), узагальнення (В. В. Давидов, А. В. Усова) [35, 42, 45] тощо.

Структурна семантика вивчає зв’язок між значеннями термінів у реченні й як це значення може бути складено з окремих елементів речення [46]. Відповідно до структурної семантики, семантична структура може бути визначена як цілісна структура пов’язаних понять [47]. У загальному випадку семантичне уявлення є графом, семантичною мережею, що відображає бінарні відношення між парами вузлів – семантичних одиниць тексту. Глибина семантичного аналізу може бути різною [48].

Відповідно, *семантична структура навчального курсу* як сукупності навчальних матеріалів має містити два види елементів (вузли та зв’язки) [49] й відображати структуру й семантику складових навчальних матеріалів [33, 50].

1.1.2 Класифікація навчальних матеріалів

Навчальний матеріал є системою, яка має структуру зі специфічними елементами й зв’язками між ними [43]. Навчальний матеріал розглядають як сукупність двох видів інформації: основної та допоміжної. Кінцева мета подання основної інформації є перетворення її в знання або вміння.

Допоміжна інформація має на меті забезпечення надійності (гарантованості) у засвоєнні основної інформації. Знання розглядають як сприйняту, зрозумілу, усвідомлену й включену в систему наявних знань інформацію [42].

Навчальний матеріал залежно від виконуваних функцій може бути згрупований наступним чином: програмний, інформаційний, операційний, контролюючий, актуалізуючий, стимулюючий, діагностуючий [43].

У даній роботі розглядаються та досліджуються *інформаційний навчальний матеріал* і *тестовий діагностуючий навчальний матеріал*. Інформаційний навчальний матеріал (ІНМ) розглядається як засіб подання знань, які надаються людині, що вивчає курс. Тестовий діагностуючий навчальний матеріал (ТНМ) є різновидом діагностуючого навчального матеріалу й розглядається як засіб перевірки рівня одержаних з ІНМ знань [33].

1.1.3 Структура подання цифрових навчальних матеріалів

На сучасному етапі для зберігання електронних документів навчальних матеріалів зазвичай використовується файл з розширенням .docx (або створений на його базі .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. Формат файлів .docx розроблений, 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 [51].

Для реалізації програмної обробки цифрових документів є доцільним використання існуючих спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентом відповідних файлів, зокрема: Microsoft.Office.Interap.Word.dll [52], DocumentFormat.OpenXml.dll [53], Spire.Doc.dll [54]. Проведені дослідження

[9] виявили переваги застосування програмного розширення Spire.Doc.dll для парсингу цифрових документів ІНМ.

Структура є природною властивістю тексту, з чого випливає, що подання ІНМ має відображати його структурність. Елементом структури ІНМ є рубрика – певним чином позначена частина тексту. Таким чином, рубрикація є системою взаємного зв'язку та підпорядкування рубрик ІНМ, що візуально може бути подана взаємним зв'язком та підпорядкуванням заголовків цих рубрик. Функціями рубрикації є структуризація та систематизація ІНМ, а також полегшення його використання. На кількість рубрик впливає загальний обсяг та складність тексту (чим важчий для сприйняття текст, тим щільнішим має бути поділ), проте навчальний курс як система знань невідворотно передбачає наявність структури рубрикації [33].

Кількість рівнів вкладеності елементів структури ІНМ може бути різною, нерівномірною в межах одного ІНМ, назви рівнів різних ІНМ можуть відрізнятись (наприклад, Дисципліна / Модуль / Розділ / Тема / Лекція / Параграф; Дисципліна / Розділ / Тема; Дисципліна / Розділ / Підрозділ / Параграф тощо). На рис. 1.1 подано приклад, у якому назвами рівнів елементів структури ІНМ є «Дисципліна», «Розділ» та «Тема».

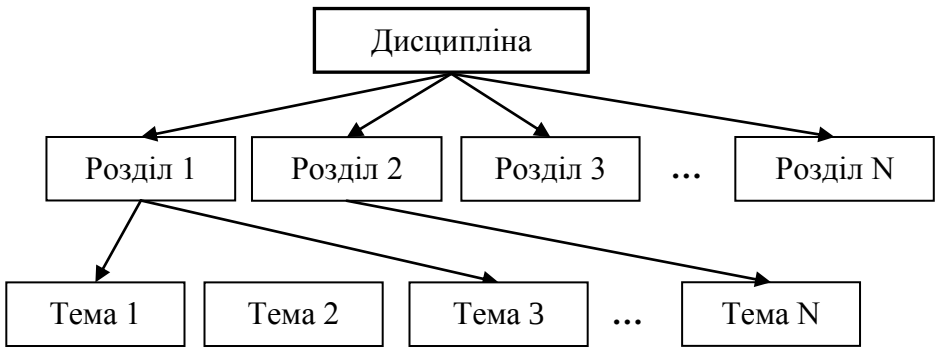


Рисунок 1.1 – Приклад формального подання структури ІНМ

Тому окрім назви (семантичного ідентифікатора) кожного елементу структури ІНМ є важливим номер рівня цього елементу в ієрархічній структурі ІНМ, в той час як назва рівню не є семантично значущою.

ІНМ мають структуру, потрібну для первинної структуризації всього контенту ІНМ, причому характерними властивостями такого підходу є наступне:

– елементи структури ІНМ розбивають контент ІНМ на окремі фрагменти;

– кожен елемент структури ІНМ, і відповідно фрагмент контенту, має назву – семантичний ідентифікатор (заголовок);

– кожен елемент структури ІНМ, і відповідно фрагмент контенту, може містити аналогічним чином підпорядковані елементи структури, що розділяють ці фрагменти на менші фрагменти з привласненням їм власних назв.

Можливості для парсингу ІНМ надає об’єктна модель електронного текстового документу [55], яка передбачає використання стилів заголовків для позначення назв рубрик.

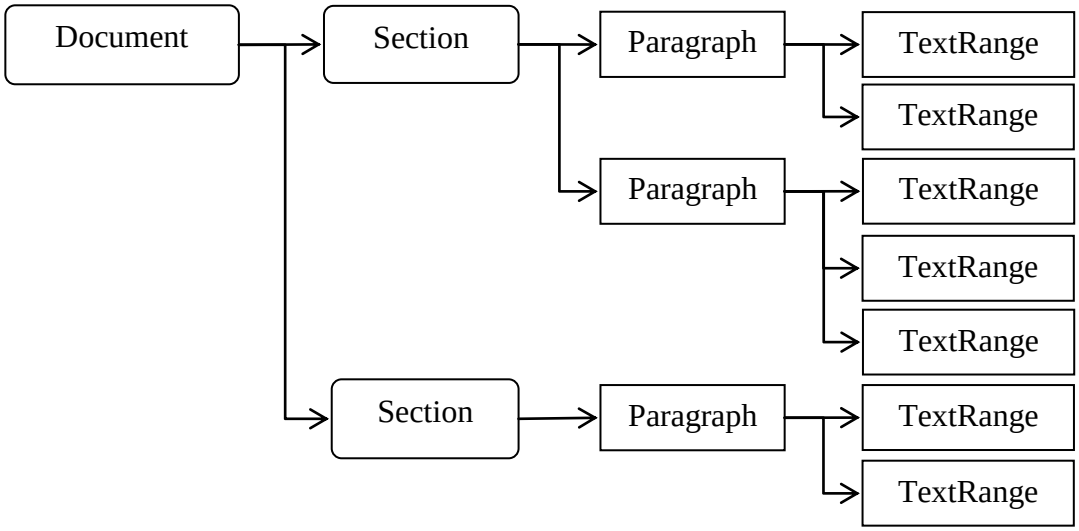


Рисунок 1.2 – Загальна структура об’єктної моделі документу MS Office

Відповідно до об’єктної моделі документу [56], формат .docx MS Office використовує розділи (*Section*), які містяться в об’єкті *Document* (рис. 1.2) і містять в собі менші елементи структури – абзаци (*Paragraph*), які можуть мати стиль заголовку (*Heading*). Наприклад, у файлах Microsoft Word назви

дисциплін можуть відповідати елементам стандартного стилю «*Heading* 1», назви розділів – «*Heading* 2», назви тем – «*Heading* 3» тощо [57]. Таким чином, структура ІНМ як цифрових документів регламентується мовами розмітки цифрових документів й реалізується через систему стилів заголовків.

TextRange є найнижчим рівнем структури документу, що визначає фрагмент тексту однакового стилю в межах *Paragraph* [58].

Відтак до множини локальних областей пошуку термінів документу (фраз) включаються неперервні впорядковані послідовності слів, що не виходять за межі контейнерів цифрового документу *TextRange* та не перериваються розділовими знаками. Одержані в результаті множини фраз дозволяють опрацьовувати кожну з фраз окремо на предмет пошуку термінів.

1.1.4 Електронні навчальні середовища

На сьогоднішній день у навчальних закладах України використовується широкий набір програмного забезпечення для організації навчання засобами інформаційних технологій [59, 60, 61]. У залежності від критерію класифікації, розрізняють програмні системи: спеціалізовані та широковживані, безкоштовні та платні, з відкритим кодом та комерційні тощо [62].

До найбільш використовуваних належать Moodle [63], ATutor [64], eLearning Server 3000 [65], Blackboard [66], IBM Lotus LearningSpace [67], Sakai LMS [68] тощо. Всі ці системи відповідають загальноприйнятим у світі стандартам організації дистанційного навчання: вони модульні, інтерактивні, персоніфіковані, адаптовані, відповідають вимогам безпеки тощо. Наведені системи дозволяють викладати ІНМ та проводити перевірку рівня одержаних із них знань використовуючи тестування. Окремо можна відзначити різноманітні програмні системи, призначені тільки для проведення тестування [69, 70, 71].

У навчальному процесі багатьох вишів України використовують систему Moodle (модульне об’єктно-орієнтоване середовище навчання) – безкоштовну, відкриту систему дистанційного навчання [72]. Система орієнтована насамперед на організацію взаємодії між викладачем та учнями, хоча підходить і для організації традиційних дистанційних курсів, а також підтримки очного навчання. Система Moodle надає широкий ряд можливостей для організації навчання [63]. В контексті даної роботи важливі наступні властивості Moodle:

– можливість проведення тестування (рис. 1.3), причому алгоритм тестування може бути модифіковано додатковими модулями;

– додаткові модулі у вигляді плагінів дозволяють модифікувати процес тестування та аналіз відповідей, що дозволяє реалізувати алгоритми адаптивного тестування (рис. 1.4);

– широкий інструментарій для роботи з множиною тестових завдань, в тому числі можливість завантаження створених тестів у форматах GIFT, MOODLE XML, XHTML тощо.

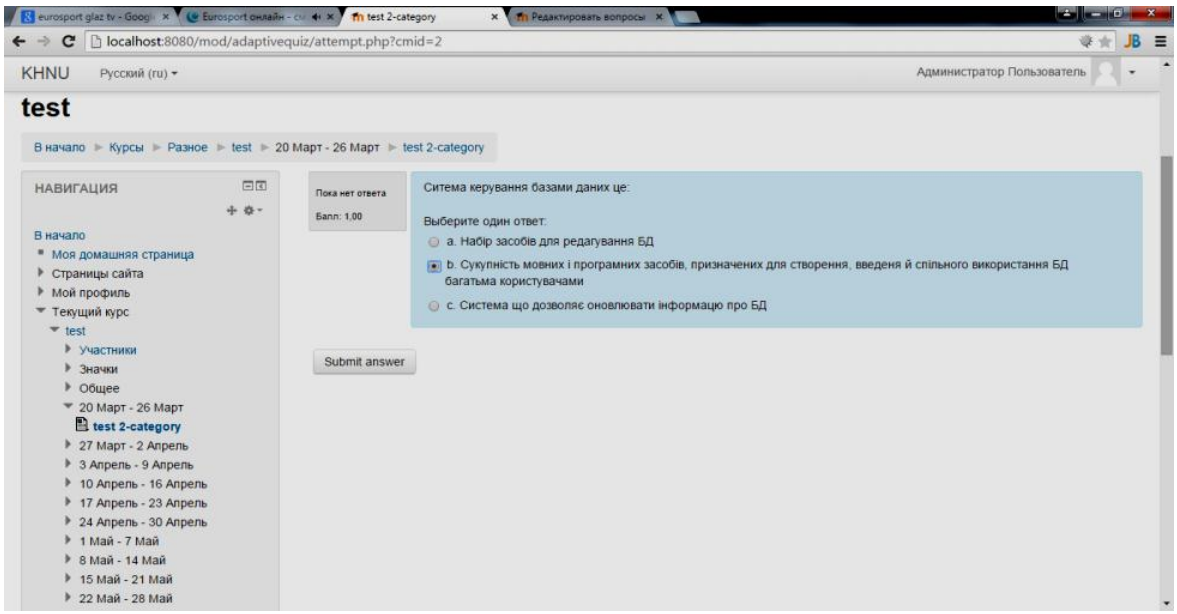


Рисунок 1.3 – Приклад тестування у середовищі Moodle

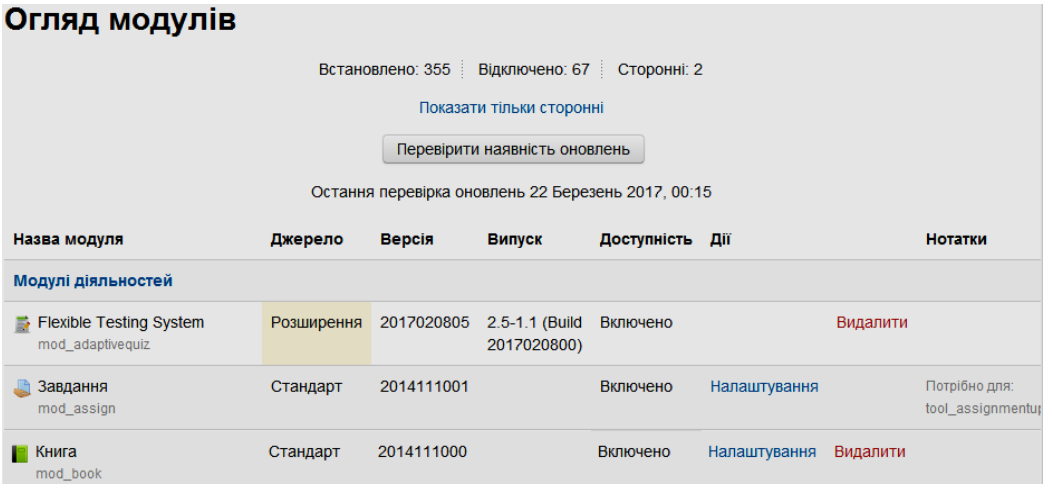


Рисунок 1.4 – Приклад встановлення додаткових модулів у середовищі Moodle

Таким чином, широковживане насьогодні в вишах України середовище Moodle можна використовувати для проведення тестування з використанням множини окремо створених тестових завдань та необхідних для цього метаданих.

1.1.5 Особливості навчальних курсів у вищій освіті

Навчальний курс дисципліни у вищій школі є логічно організованою системою теорій з рекомендаціями шляхів їх застосування на практиці [40]. Семантично структуру навчального курсу зазвичай формують за індуктивною або дедуктивною схемою [73] подачі навчального матеріалу.

Індуктивна структура курсу передбачає побудову викладення навчального матеріалу від фактів і прикладів до загальних положень і наукових висновків. Дедуктивна структура курсу передбачає починати викладення навчального матеріалу загальними положеннями із наступним оглядом можливості їх застосування на конкретних прикладах [74].

Загальні рекомендації з побудови структури навчальних матеріалів викладено в положеннях законодавчих актів, норм та вимог Міністерства освіти і науки України: Законів України «Про вищу освіту», «Про видавничу справу», «Про авторське право і суміжні права», Наказу Міністерства

освіти і науки України «Щодо видання навчальної літератури для вищої школи» [75, 76].

Відповідно до діючих норм [77], структура ІНМ у вищій освіті є ієрархічною, кожен рівень визначає структурний та семантичний вміст підрівнів. Так, якщо ІНМ складається з модулів, то кожен модуль може містити перелік тем, які повинні бути висвітлені в його межах; у свою чергу, кожна тема потребує множини лекцій для її викладення.

Для забезпечення якісної перевірки знань, важливим аспектом є умова побудови тестових завдань, які повністю покривають таку ієрархічну структуру ІНМ [57]. Для цього необхідне співставлення кожного тестового завдання з конкретним фрагментом контенту ІНМ та семантичної одиниці (наприклад терміну, поняття), знання якої ним перевіряється [78].

1.2 Сучасний стан комп’ютеризації контролю знань

При якісному конструюванні тестів можна забезпечити відповідний рівень дискримінативності. Задача розробки множини тестових завдань вимагає не тільки їх семантичної якості, збалансованості за рівним складності та типами, а й повноцінного та рівномірного покриття ними навчального матеріалу [79]. На сучасному етапі широко впроваджується метод адаптивного контролю рівня знань [80], і сформована множина тестових завдань повинна бути придатною для такого використання.

1.2.1 Напрямки комп’ютеризації контролю знань

Різноманітним аспектам тестування [81], розробки та застосування навчальних і тестувальних середовищ за сучасними інформаційними технологіями [82, 83, 84], питанням розробки баз даних і знань програмних систем перевірки рівня знань [85, 86] присвячено багато праць вітчизняних і закордонних авторів: Аванесова В. С., Башмакова І. А., Гагаріна О. О.,

Клайна П., Литвиненко В. І., Мельника А. М., Пасічника В. В., Пасічника Р. М., Рафальської О. О., Снитюка В. Е., Теленика С. Ф., Титенко С. В., Томашевського В. М., Тонконогого В. М., Федусенко О. В, Федорука П. І., Schwarz, Weber та інших. Більшість з них здійснювали дослідження в сфері проведення тестувань [87, 88, 89, 90], наповнення бази тестових завдань за допомогою засобів підтримки ручного створення тестових завдань [91], оцінки складності тестових завдань [92, 93], безпеки процесу тестування та відтворення результатів [94, 95, 96].

Проблеми автоматизації процесу тестування і обробки його результатів ґрунтовно досліджені в літературі. Однак, задачі автоматизації формування банку тестових завдань досліджені в недостатній мірі [81]. Наразі питання формування банку тестових завдань у більшості випадків залишається виключно прерогативою розробника, який працює без використання інтелектуальних засобів автоматизації даного процесу [97]. Інформаційні технології дають можливість суттєво зменшити трудові затрати на створення самих тестових завдань з можливістю їх постійного оновлення, що формує актуальний напрямок наукових досліджень.

Прагнення автоматизувати формування тестових завдань наштовхуються на область штучного інтелекту і на проблеми формалізації знань та їх подальшого використання в процесі створення тестів. Традиційний підхід до створення засобів тестування фактично є комп’ютеризацію ручного тестування й полягає у використанні інформаційно-комунікаційних технологій замість паперової роботи, що дає додаткові можливості щодо управління формуванням тестів з множини створених тестових завдань та автоматичної перевірки результатів. За таких умов професійно розроблені експертом завдання мають високу якість і зрозумілість; однак істотним недоліком підходу є висока трудомісткість самого процесу формування тестових завдань.

Однією з проблем є об'єктивізація процесу оцінювання знань й умінь [33]. Її вирішення супроводжується необхідністю розробки автоматизованих систем навчання й контролю знань за умови розв’язання задач, що визначають

актуальні напрямки підвищення ефективності комп'ютеризації контролю знань. Визначено наступні напрямки підвищення ефективності комп'ютеризації контролю знань:

– автоматизація створення множин тестових завдань з метою зменшення часу та трудомісткості розробки тесту;

– автоматизація контролю за розподілом тестових завдань у множині з метою досягнення збалансованих за рівним складності та типом тестів, а також повноцінного та рівномірного покриття набором тестових завдань навчального матеріалу;

– застосування адаптивних алгоритмів тестування з метою мінімізувати інформаційну надлишковість та час, необхідний для визначення рівня засвоєних знань.

1.2.2 Підходи до автоматизації формування тестових завдань

Серед відомих засобів автоматизації формування тестових завдань [22] необхідно відзначити метод параметризованих задач, метод генерації тестових завдань за понятійно-тезисною моделлю, а також метод генерації тестових завдань за формалізацією структурованих текстових тверджень. Дослідженням в напрямку автоматизації формування тестових завдань присвячено багато наукових публікації [98, 99, 100, 101, 102], також варто виділити роботи таких науковців як Титенко С. В. [97], Снитюка В. Е. [33], Пасічника Р. М. [103], Мельника А. М. [104], Турчак А. М. [105].

На сучасному етапі розвивається підхід до *формування тестових завдань за онтологією* предметної області [106]. З цією метою поняття ІНМ подаються у вигляді множини ланцюжків «сутність – зв'язок – опис», чому передує нотація $C : <N, E, L, D>$, де N – нотація, $E=\{E_i\}$ – кортеж сутностей предметної області, L – зв'язки, $D=\{D_i\}$ – кортеж властивостей. Існують програмні засоби для роботи з онтологіями, наприклад мова OWL,

запропонована в проекті Semantic Web [107], а також редактор онтологій і фреймворк для побудови баз знань Protege [108].

Шляхом маніпулювання елементами наведеної моделі отримують запитання, відповіді й дистрактори (некоректні варіанти відповідей) тестових завдань. Наведений метод дозволяє проводити автоматизоване формування тестових завдань за онтологією, проте він може бути застосований лише для предметних областей, поняття яких подані у відповідному вигляді. Формування онтології є окремим складним завданням; із відомих підходів до автоматизації процесу побудови онтології слід відзначити *метод генерації множини ядер продукційних правил* для формування бібліотеки методів для побудови онтологій [105, 109].

Відомий *метод параметризованих задач* [98], який дозволяє створювати тестові завдання відкритого типу. Тестове завдання є створеним автором текстом (шаблон), в якому деякі елементи можна змінювати (параметри). Змінюючи відповідно до заданого алгоритму параметри, за створеним шаблоном одержуються нові варіанти завдання.

Недоліком методу визначають трудомісткість формування множини шаблонів тестових завдань [110]. Перевагою методу є можливість створення великої кількості тестових завдань за малою кількістю шаблонів.

В ряді робіт [97, 111] розглядається *метод генерації тестових завдань за понятійно-тезисною моделлю*. Метод передбачає формальне подання ІНМ як множини понять (предмет обговорення із предметної області) $C=\{c_1,\ldots,c_{n1}\}$ та множини тез (відомості з ІНМ про поняття) $T=\{t_1,\ldots, t_{n2}\}$. Тези зв'язані з поняттями ($CT: T \rightarrow C$), кожне поняття може мати довільну кількість тез ($TC: C \rightarrow 2^T$). Поняття й тези можуть бути віднесені до певних семантичних класів: означення, проблеми, методи тощо [110]. Генерація тестів відбувається на основі створених шаблонів, контент яких формують елементи множин понять і тез.

Перевагою методу генерації тестових завдань на основі понятійно-тезисної моделі є можливість автоматизованого створення семантично й

синтаксично коректних тестових завдань. Проте цей метод вимагає значних затрат часу на формалізацію вручну контенту ІНМ до потрібного вигляду (тези, поняття, класи) й накладає обмеження на семантичну структуру ІНМ як сукупність понять та тез [112].

Відомий *метод генерації тестових завдань на основі формалізації структурованих текстових тверджень* [113, 114], який використовує семантичну, формальну та синтаксичну типізацію елементів тексту. Для цього пропонується формальне подання фрагментів контенту ІНМ як сукупності компонентів, які описують процеси, поняття або їх характеристики. Компоненти поєднуються в речення сполучниками, й фрагменти контенту подаються у вигляді *Df = <IdE, CS, CNJ, TCNJ,* Pr*_IdE, IdSc, IdFc>*, де *Df* - текстове твердження; *IdE* – ідентифікатор компоненти твердження; *CS* – зміст компоненти; *CNJ* – представлення зв'язку між компонентами тверджень; *TCNJ* – тип зв'язку між компонентами; *Pr_IdE* – посилання на батьківський елемент дерева (ідентифікатор відповідної компоненти); *IdSc* – ідентифікатор семантичного класу; *IdFc* – ідентифікатор формального класу.

Формальні класи подання (текстового, числового, формульного, алгоритмічного, ілюстративного та програмного коду), абстрактні семантичні класи (означення, задачі, методи, ефективність методів, приклади реалізації), зв'язки однорідності альтернатив (для формування семантично коректних дистракторів) й інші засоби дозволяють формувати семантично й синтаксично узгоджені й коректні тестові завдання, що є перевагою даного методу [115]. Недоліком є потреба в значних затратах часу для приведення контенту ІНМ до потрібного формального вигляду й обмеження області формування тестових завдань рамками наявних в моделі елементів.

Таким чином, існуючі методи є ефективними для використання у визначених випадках, проте більшість із них вимагає суттєвої і трудомісткої попередньої підготовки ІНМ. В умовах великої кількості та інтенсивного оновлення курсів навчальних дисциплін така попередня обробка ІНМ є складною задачею; проте в деяких випадках, наприклад автоматизації

формування математичних задач методом параметризованих тестів, безальтернативною. Втім, значна частина контенту ІНМ багатьох курсів навчальних дисциплін містить переважно текстовий контент, який характеризується послідовністю й семантичною зв'язністю подання. Ця особливість відриває шлях до розробки методу автоматизованого формування тестових завдань, який не вимагає суттєвої попередньої обробки ІНМ.

1.2.3 Види тестових завдань

Дидактичний тест є множиною тестових завдань, яка дає змогу якісно виміряти рівень засвоєння знань та сформованості вмінь і навичок учнів з предмета [116].

Питання класифікації типів тестових завдань наразі однозначно не вирішено. Існують різні системи класифікації. В залежності від критерію класифікації, одержують різні класифікації типів тестових завдань. Наприклад, за способом виконання тестові завдання поділяють на закриті (коли пропонуються вибір із кількох наперед сформульованих відповідей, з яких одна або кілька правильні) та відкриті (потребують символьного введення відповіді).

У середовищі Moodle використовуються наступні типи питань в тестових завданнях [63]:

– множинний вибір – студент обирає відповідь на питання з декількох запропонованих йому варіантів, причому питання можуть мати одну (одиничний вибір) або одразу декілька (множинний вибір) правильних відповідей;

– логічний тип (Правильно/Неправильно) – відповідь на питання студент обирає між двома варіантами – «Так» чи «Ні»;

– відповідність – кожному елементу відповідей першої групи потрібно зіставити елемент відповідей другої групи;

– коротка відповідь (на введення тексту) – відповіддю на питання є слово або коротка фраза, яку вводять за допомогою клавіатури, при цьому допускається кілька правильних відповідей (варіантів написання), які можуть мати різні оцінки, також відповіді можуть бути (або не бути) чутливими до регістру;

– числовий – відповіддю на питання про виконання обчислювальних операцій є число, при цьому числова відповідь може мати заданий інтервал гранично допустимої похибки відхилення від правильного значення;

– есе – студент коротко викладає свій погляд на запропоновану проблему;

– вбудовані відповіді (embedded answers) – запропоновано текст, безпосередньо в який вставляються короткі відповіді, числові відповіді або множинний вибір, як в «робочому зошиті».

Середовище Moodle визначає максимальний обсяг доцільних для створення типів тестових завдань, ґрунтуючись на сучасному рівні розвитку ІТ. Також Moodle можна використовувати як експериментальну установку для апробації створених множин тестових завдань. Виходячи з цього, визначено такі типи тестових завдань, які доцільно формувати автоматизовано: одиничний вибір, множинний вибір, логічного типу та коротка відповідь (завдання на введення тексту). Тестові завдання на відповідність та з вбудованими відповідями є поєднанням інших типів тестових завдань, а есе виступає різновидом спілкування з студентом й не передбачає створення засобів для автоматизованої перевірки.

1.2.4 Продукційні правила для створення тестових завдань

Процес ручного створення тестів характеризується циклічним повторенням однотипного процесу, в якому кожна ітерація циклу відповідає вирішенню задачі створення одного тестового завдання. Причому такі ітерації об’єднуються в серії, що характеризуються аналізом одного окремого

локального фрагменту контенту ІНМ. Фрагменти ІНМ зазвичай обираються послідовно, згідно черги їх викладення. При аналізі обраного фрагменту контенту (одиниця ітерації) розробник оцінює можливість застосування відомих йому правил перетворення контенту ІНМ в контент елементів тестового завдання [116]. Таким чином, алгоритм ручного створення тестових завдань полягає в тому, що по черзі обираються логічно відокремлені фрагменти вхідного контенту й до кожного з них проводиться перебір окремо сформованої множини правил; у випадку збігу умов правила і параметрів контенту на їх основі формується тестове завдання (рис. 1.5).

Рисунок 1.5 – Приклад використання правила продукції для формування тестового завдання

З наведеного можна зробити висновок, що процес ручного створення тестів базується на використанні заснованої на правилах моделі (продукційної моделі, моделі правил продукції), й дозволяє подавати знання у вигляді конструкцій типу «Якщо (умова), то (дія)» (рис. 1.6) [117, 118]. Таку модель використовують продукційні експертні системи, що працюють аналогічним чином для вирішення практичних задач наведеного характеру [119, 120, 121].

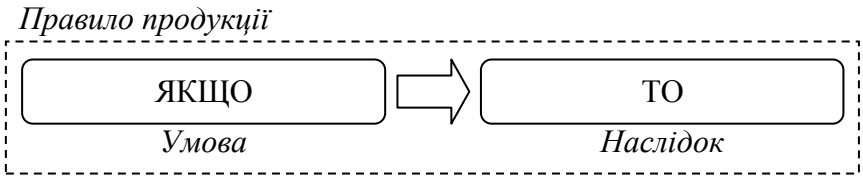


Рисунок 1.6 – Логічна структура правила продукції

У продукційних системах знання подані у формі множини правил продукції (продукційних правил), за якими формуються висновки, що повинні бути зроблені (або не зроблені) в різних ситуаціях [117, 118, 122]. Висновки робляться за методами прямого або зворотного логічного висновку. Залежно від методу логічного висновку розрізняють два види продукційних систем: системи з прямим логічним висновком та системи із зворотним логічним висновком [123, 124].

Загальна стратегія вирішення задач полягає в розбитті їх на фрагменти, які можна легше обробити. При цьому, системи з прямим логічним висновком знаходяться під управлінням фактів. Вони починають свою роботу з відомих початкових фактів і продовжують, використовуючи правила для створення висновків або виконання певних дій. Системи із зворотним логічним висновком керуються гіпотезами. Вони починають свою роботу з гіпотези, або мети, яку користувач намагається довести і продовжують, відшукуючи правила, які дозволять довести коректність гіпотези [125, 126].

У випадку с процесом створення тестів, прямий логічний висновок відповідає створенню нових тестових завдань за правилами; зворотний логічний висновок відповідає пошуку правил, що були використані при перетворенні фрагменту існуючого контенту ІНМ в елемент тестового завдання.

З наведеного можна зробити висновок, що використання продукційної моделі для подання алгоритмів створення тестових завдань відповідає процесу ручного створення тестів, який характеризується циклічним повторенням однотипного процесу, в якому кожна ітерація циклу відповідає вирішенню задачі створення одного тестового завдання. Формалізація процесу ручного створення тестових завдань за використання заснованої на правилах моделі знань відкриває можливості для його автоматизації.

1.2.5 Аналіз використання адаптивного тестування

На сучасному етапі в області комп’ютерного тестування знань визначено перспективним впровадження технологій адаптивного тестування в електронних навчальних середовищах, чому присвячені праці вітчизняних та закордонних авторів [44, 99, 127, 128, 129, 130, 131, 132, 133, 134].

Адаптивним називається тест, в якому складність чи інші властивості тестових завдань змінюються залежно від правильності відповідей випробуваного. Наприклад, якщо студент правильно відповідає на тестові завдання, складність наступних завдань підвищується, якщо неправильно – знижується. Також є можливість внесення додаткових питань в області, яку особа знає не дуже добре для більш тонкого з’ясування рівня знань у цих галузях. Дана модель застосовується для тестування студентів за допомогою комп’ютерних засобів, тому що «вручну» неможливо заздалегідь розмістити стільки питань і в тому порядку, скільки і в якому вони повинні бути пред’явлені особі, що тестується [33]. Системи адаптивного тестування визнаються більш ефективними за існуючі класичні системи [99].

Відповідно до існуючих визначень [33], ефективною вважається така система тестування, у якій навчання й контроль знань проходять за мінімально можливий час, причому забезпечується їхня повнота, а також відсутність або мінімальна присутність інформаційної надмірності або недостатності і максимально можлива об’єктивізація отриманих результатів. Відомо, що чисельним вираженням ефективності є критерій ефективності. Рішення завдання пошуку максимального значення критерію ефективності пов'язане з формалізацією й оптимізацією зазначених вище понять.

При проектуванні процесу навчання та контролю знань, потрібне графоподібне подання навчального матеріалу та множини тестових завдань [135]. Кожний з рівнів навчального матеріалу може містити ряд підрівнів. Крім горизонтальної структури в графі питань присутній і вертикальний поділ, за допомогою якого найчастіше виділяються теми НК. В

процесі тестування бажаним є підхід, коли перехід на наступний рівень залежить від відповідей на питання попереднього рівня, що є одним зі способів усунення інформаційної недостатності [33]. Інформаційна надмірність виникає внаслідок неправильної реалізації структури контролю знань.

Відомі наступні варіанти адаптивного тестування [33].

– Всім учням на початку тестування даються завдання середньої складності й потім, залежно від відповідей, кожному визначається завдання складніше або легше.

– Контроль знань починається з будь-якого рівня складності завдань і далі ітераційно наближається до реального рівня складності, що відповідає рівню знань.

– Існує база знань, з питаннями, розподіленими по рівнях складності. При правильній відповіді наступне завдання обирається із групи більш складних завдань, при неправильному – з більш легких.

– Тестування починається з перевірки знання найважливішого терміна та у випадку коректних відповідей підвищується складність шляхом перевірки знання менш важливих термінів.

– Обхід за структурою НК зверху вниз, у випадку невірної відповіді на найбільш прості питання тестування у частині структури, що слідує нижче, не проводиться.

Підсумовуючи наведене, обов’язковою вимогою до автоматизованого створення множин тестових завдань визначено одночасне формування необхідних зв’язків кожного тестового завдання з семантичними та структурними елементами ІНМ. Це дозволить використовувати їх для тестування не тільки за традиційним алгоритмом (підбір тестових завдань у тест випадковим чином), а й для тестування з використанням технології адаптивного тестування, що відповідає сучасним тенденціям в розвитку систем тестування.

1.3 Особливості семантичного аналізу навчальних матеріалів

Проаналізовані існуючі методи автоматизації формування тестових завдань переважно вимагають тривалої та трудомісткої попередньої підготовки ІНМ. У роботі пропонується автоматизувати процес створення семантичної структури ІНМ засобами семантичного аналізу текстів до вигляду, достатнього для подальшого автоматизованого формування тестових завдань. При цьому вважається, що семантичний вміст тексту може бути переданий відповідною множиною ключових термінів (слів і словосполучень) [136]. Існують методи, що можуть бути використані для автоматизованого формування таких множин.

1.3.1 Семантичний аналіз та сучасні SEO-системи

Основи досліджень в галузі семантичного аналізу текстів заклали такі вчені як Liddy E. D., Sowa J. F., Harris R., Brown K., Tadeusiewicz R., Chomsky N., Herrmann D., Chapman N., Lyons J., Кришталь О. А., Лефевр В. А.. На сучасному етапі вирішенню проблеми автоматизованої аналітичної обробки текстової інформації присвячені праці багатьох вітчизняних та іноземних авторів, серед яких Анісімов А. В., Бармак О. В., Бісікало О. В., Бродер А., Виноград Т. В., Гальперін І. Р., Греймас А. Ж., Кобозева І. М., Крак Ю. В., Кронгауз М. А., Лакофф Дж., Ланде Д. В., Леонтьєв Н. Н., Лунн Х. П., Люгер Д. Ф., Маккарті Д., Ньюелл А., Піттс У., Попов Е. В., Поспєлов Д. А., Рубашкін В. Ш., Саченко А. О., Севбо І. П., Фостер Д. М., Хопкрофт Дж., Шемакін Ю. І., Широков В. А. та інші.

В ІНМ найбільш поширеним є текстовий формат подання інформації, оскільки він є найбільш звичним для людини [33, 137]. У ряді робіт сучасних авторів доводиться, що семантичний вміст тексту може бути переданий його згорнутим (стиснутим) поданням, яке в залежності від ступеню стиснення та методу одержання може приймати вигляд анотації [138], реферату [139] або

множини ключових слів [140]. Для автоматизації їх формування використовуються методи та засоби семантичного аналізу текстів [141, 142].

Дослідження в напрямку семантичного аналізу текстів у Європі та США привертають увагу відомих фірм і державних установ найвищого рівня. Європейський Союз наразі координує ряд програм у галузі автоматичної обробки тексту [143], зокрема Human Language Technology Sector of the Information Society Technologies (IST) Programme, у яких основні дослідження присвячені автоматизації процесу семантичного та синтаксичного аналізу електронних текстів [144].

Ключовий термін є таким словом чи словосполученням у тексті, яке здатне в сукупності з іншими термінами семантично передавати основний зміст тексту [145, 146, 147]. Як правило ключовий термін має високий ступінь повторюваності у тексті та має здатність конденсувати, згортати інформацію, виражену цілим текстом, об’єднувати його основний зміст [148].

Терміни призначені для семантичної концентрації сенсу (замінюють більший набір слів – означення), їх засвоєння важливе для подальшого викладення матеріалу. Тому розуміння термінів ІНМ є обов’язковим елементом знань, що одержуються з навчальних дисциплін [75, 76, 149].

Поняття ключових термінів як носіїв найбільш семантично вагомої інформації про текст активно використовується в інформатиці, зокрема, завданнях інформаційного пошуку [150, 151, 152]. Для ІТ зазначений напрямок знайшов широке відображення в системах аналітичної обробки текстів для пошукової оптимізації (SEO-системах) [153].

Більшість існуючих програмних систем, створених для автоматизованої аналітичної обробки цифрових текстів, призначені переважно для SEO-аналізу текстів [154, 155]. Наприклад, аналізатор від біржі контенту «Адвего» [156] вираховує кількість слів, кількість граматичних помилок тощо. Сервіс також дозволяє побачити перелік ключових слів, які вираховуються за методом частотної оцінки, та забезпечує аналіз текстів на плагіат. Сервіс для семантичного аналізу текстів «Istio» [157] визначає вагу слів в тексті для

складання схеми активних гіперпосилань (анкор-листа). Багатофункціональна SEO-платформа «Serpstat» [158] забезпечує поглиблений аналіз текстового контенту, аналіз пошукових запитів, розподіл запитів по дереву сайту, пошук схожих фраз тощо; аналіз ключових слів, зокрема, допомагає використовувати найефективніші ключові слова в контенті.

Таким чином, сучасний рівень розвитку IT визначає широке використання методів і засобів семантичного аналізу текстів, які надають можливість подання семантичного вмісту цифрового тексту відповідною множиною ключових термінів. При цьому, така множина може бути одержана автоматизовано.

1.3.2 Методи пошуку ключових термінів у цифрових текстах

Множина ключових термінів тексту є найбільш семантично стиснутим результатом семантичного аналізу тексту [136, 159, 160, 161]. Для автоматизації пошуку ключових слів використовуються різноманітні методи аналізу текстів [162, 163, 164], серед яких широке розповсюдження [165] одержали метод частотної оцінки TF, метод оцінки TFIDF та метод дисперсійного оцінювання DE. Ці методи дозволяють поставлені у відповідність окремим словам або словосполученням тексту деякі певним чином визначені числові вагові значення, що вказують на міру їх важливості в досліджуваному тексті [166, 167].

Частотна оцінка *TF* (term frequency) є частотою появ певного слова *i* у тексті, що розглядається, й обчислюється наступним чином [168]:

$$TF_i = \frac{n(i)}{N}, \tag{1.1}$$

де *n(i)* – кількість появ слова *i* у тексті, *N* – загальна кількість слів у тексті.

Оцінка *TFIDF* є добутком частоти появ слова у тексті *TF* та зворотної документарної частоти слова *IDF* (inverse document frequency) [169]:

$$TFIDF = TF * IDF, \;\; IDF_i = \log \frac{D}{d_i}, \tag{1.2}$$

де D – кількість альтернативних документів для порівняння або фрагментів, на які розбивається текст при аналізі; d_i – кількість документів або фрагментів, у яких дане слово присутнє.

Дисперсійна оцінка *DE* (disperse evaluation) за змістом близька до оцінки TFIDF, та є оцінкою дискримінантної сили слів. Дисперсійний аналіз є статистичним методом оцінки зв'язку між факторними й результативними ознаками в різних групах, відібраний випадковим чином, заснований на визначенні розходжень (розкиду) значень ознак. В основі дисперсійного аналізу лежить аналіз відхилень всіх одиниць досліджуваної сукупності від середнього арифметичного. Як міра відхилень береться дисперсія – середній квадрат відхилень. Відхилення, викликані впливом факторної ознаки (фактора) порівнюються з величиною відхилень, викликаних випадковими обставинами. Якщо відхилення, викликані факторною ознакою, більш істотні, ніж випадкові відхилення, то вважається, що фактор впливає на результуючу ознаку. Аналіз значень дисперсійної оцінки дозволяє відділити із загального переліку широковживаних у тексті слів слова, що розташовані рівномірно. Якщо деяке слово A в тексті, що складається з N слів, позначене як A_k^n, де індекс k – номер появи даного слова в тесті, а n – позиція даного слова в тексті, то інтервалом між послідовними появами слова при таких позначеннях буде величина:

$$\Delta A_k^m = A_{k+1}^m - A_k^n = m - n, \tag{1.3}$$

де на m-ій і n-ій позиціях у тесті знаходиться слово A, яке зустрілось k+1-ий і k-ий рази. Тоді дисперсійна оцінка розраховується наступним чином [161]:

$$DE = \frac{\sqrt{(\Delta A^2) - (\Delta A)^2}}{(\Delta A)} \tag{1.4}$$

де (ΔA) – середнє значення послідовності $\Delta A_1, \Delta A_2, \Delta A_k$; (ΔA^2) – послідовності A_1^2, A_2^2, A_k^2; K – кількість появи слова A в тексті.

У ряді робіт наводяться результати досліджень, згідно яким метод дисперсійного оцінювання дозволяє одержувати найбільш релевантну множину ключових термінів як у цифрових текстах загалом [165], так і власне в ІНМ [4].

Поряд з послідовним аналізом текстів, використовується побудова мереж, вузлами яких є елементи – слова, фрагменти природної мови. Це дозволяє виявляти структурні елементи тексту, без яких він втрачає свою цілісність. При цьому є актуальною задача визначення інформаційно-значущих структурних елементів. Такі елементи можуть використовуватися також для пошуку складних компонентів тексту, таких як колокації, надфразові єдності [170], зокрема, при пошуку подібних фрагментів у різних текстах [171].

В подальшому визначені ключові слова часто використовуються для побудови мереж з текстів, так званих мереж слів (Language Network). Є різні способи інтерпретації вузлів і зв'язків, що призводить до різних видів подання таких мереж. Вузли можуть бути з’єднані між собою, коли відповідні їм слова стоять поруч у тексті [172], належать одному абзацу чи реченню [173], синтаксично або семантично з’єднані [174]. У рамках теорії складних мереж (Complex Network) [175] запропоновано кілька методів побудови мереж на основі часових рядів [170], серед яких можна назвати граф горизонтальній видимості (Horizontal Visibility Graph – HVG) [165, 176].

Відома семантична мережа (Semantic Web) [177], що є загальнодоступною глобальною семантичною мережею, формованою на базі Всесвітньої мережі Інтернет шляхом стандартизації подання інформації у вигляді, придатному для машинної обробки. Семантична мережа створена для того, щоб зробити інформацію придатною для автоматичного аналізу.

Використання семантичних мереж може допомогти в розв’язку задачі формування ключових словосполучень із окремих слів, проте не сприяє розв’язку задачі пошуку ключових слів. Область можливого аналізу тексту обмежується обсягом власне мережі, а її модифікація вимагає значних

трудових витрат на формування бази знань, необхідності залучення експерта по предметної області й інженера по знаннях [110, 178].

Таким чином, задача пошуку ключових термінів у ІНМ може бути розв’язана з використанням одного з розглянутих методів пошуку ключових слів.

1.3.3 Семантичний аналіз навчальних матеріалів

ІНМ в більшості випадків формується у вигляді підручника, навчального посібника чи конспекту лекцій. Особливістю семантичного аналізу відповідних цифрових документів є обов’язкова наявність структури документу й необхідність її врахування при обробці контенту та формуванні результатів [136]. Структура документу ІНМ є відображенням рубрикації – системи взаємного зв’язку та підпорядкування рубрик, що візуально виражена взаємним зв’язком та підпорядкуванням *заголовків* цих рубрик .

Кожна рубрика (елемент структури ІНМ) має власний контент й відповідно власне семантичне навантаження та зосереджує увагу на окремих термінах. Завдання пошуку ключових термінів у навчальних матеріалах ускладнюється тим, що як термін може виступати не тільки слово, а й словосполучення з кількох слів, причому як термін може бути використане скорочення чи абревіатура.

Таким чином, при пошуку ключових термінів у текстовому контенті ІНМ слід враховувати наявність структури документу. Кожен елемент структури (рубрики) незалежно від рівня підпорядкованості має власне семантичне ядро, відповідно й множину ключових термінів. Як наслідок, множини ключових термінів виступають семантичним розширенням структури документу.

Також варто зауважити, що елементом, який передає сенс у ІНМ, може бути спеціальний об’єкт (формула, рисунок, схема тощо). Виявлення таких семантично важливих об’єктів для включення їх у множину ключових

термінів можна організувати через врахування їх частоти використання через спеціальні посилання на них або їх сигнатуру.

1.4 Висновки до розділу та постановка задачі

Проведений аналіз підтвердив актуальність задачі автоматизованого створення тестових завдань для перевірки рівня отриманих знань за структурою навчальних матеріалів. Розв’язок цієї задачі суттєво зменшить час на створення ТНМ, забезпечить його належний якісний рівень та дасть змогу реалізувати адаптивне тестування. Адаптивне тестування дозволить визначати рівень знань меншою від існуючих підходів кількістю тестових завдань та забезпечить при цьому максимальну повноту та мінімальну надмірність процесу контролю знань. Автоматизація створення множин тестових завдань дозволить забезпечити рівномірне покриття ІНМ тестовими завданнями. Існуючі ж методи та підходи до автоматизації створення тестових завдань, що переважно характеризуються суттєвими вимогами до приведення ІНМ до потрібного вигляду та структури, вимагають додаткових витрат часу та зусиль.

В результаті проведеного аналізу існуючих підходів сформульовані наступні завдання дослідження, метою яких є автоматизація формування розподілених за структурою ІНМ тестових завдань:

1) провести аналіз існуючих методів пошуку ключових термінів у цифрових текстах і методів автоматизованого формування тестових завдань до ІНМ;

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

3) удосконалити метод формування структури ІНМ та пошуку у них ключових термінів, включаючи словосполучення, в ієрархічному розрізі рубрик ІНМ;

4) розробити метод автоматизованого формування тестових завдань до ІНМ за допомогою правил продукції, який не вимагатиме додаткової формалізації ІНМ;

5) розробити ІТ автоматизованого створення тестових завдань за допомогою отриманих моделей та методів;

6) виконати експериментальну перевірку ІТ автоматизованого створення тестових завдань.

РОЗДІЛ 2.
МОДЕЛІ ТА МЕТОДИ ДЛЯ АВТОМАТИЗОВАНОГО СТВОРЕННЯ ТЕСТІВ ДО НАВЧАЛЬНИХ МАТЕРІАЛІВ

ІТ автоматизованого створення тестів до ІНМ курсу навчальної дисципліни призначена для автоматизованого перетворення вхідної інформації поданої у вигляді електронного документу ІНМ у вихідну інформацію у вигляді множини тестових завдань. Електронний документ ІНМ (наприклад, підручник або конспект лекцій) є файлом із слабо структурованим текстовим контентом (наприклад, формату .docx), що містить структуру документу в вигляді заголовків і відповідні їм текстові теоретичні відомості. Колекція тестових завдань є множиною різних за параметрами (тип запитання, кількість правильних відповідей, терміни які використовуються в завданні тощо) тестових завдань, які можуть бути використані для автоматизованої перевірки рівня засвоєння знань за допомогою існуючих навчальних середовищ.

Розглянута ІТ базується на використанні інформаційної моделі семантичної структури НК. Вона містить множини заголовків, ключових термінів і тестових завдань, співставлення між якими формується множиною зв’язків.

Для наповнення елементів моделі, пропонується застосовувати метод формування структури навчальних матеріалів та пошуку у них ключових термінів й метод автоматизованого формування тестових завдань. Використання зазначених двох методів забезпечить наповнення елементів інформаційної моделі семантичної структури НК, яка в свою чергу може бути використана як для автоматизованого створення тестів до навчальних матеріалів, так і з іншою метою, зокрема для забезпечення адаптивного тестування рівня знань.

2.1 Загальна схема інформаційної технології

ІТ автоматизованого створення тестів до навчальних матеріалів базується на використанні інформаційної моделі семантичної структури НК. Інформаційна модель семантичної структури НК з використанням двох методів (метод формування структури навчальних матеріалів та пошуку у них ключових термінів, метод автоматизованого формування тестових завдань), визначає структуру ІТ (рис. 2.1).

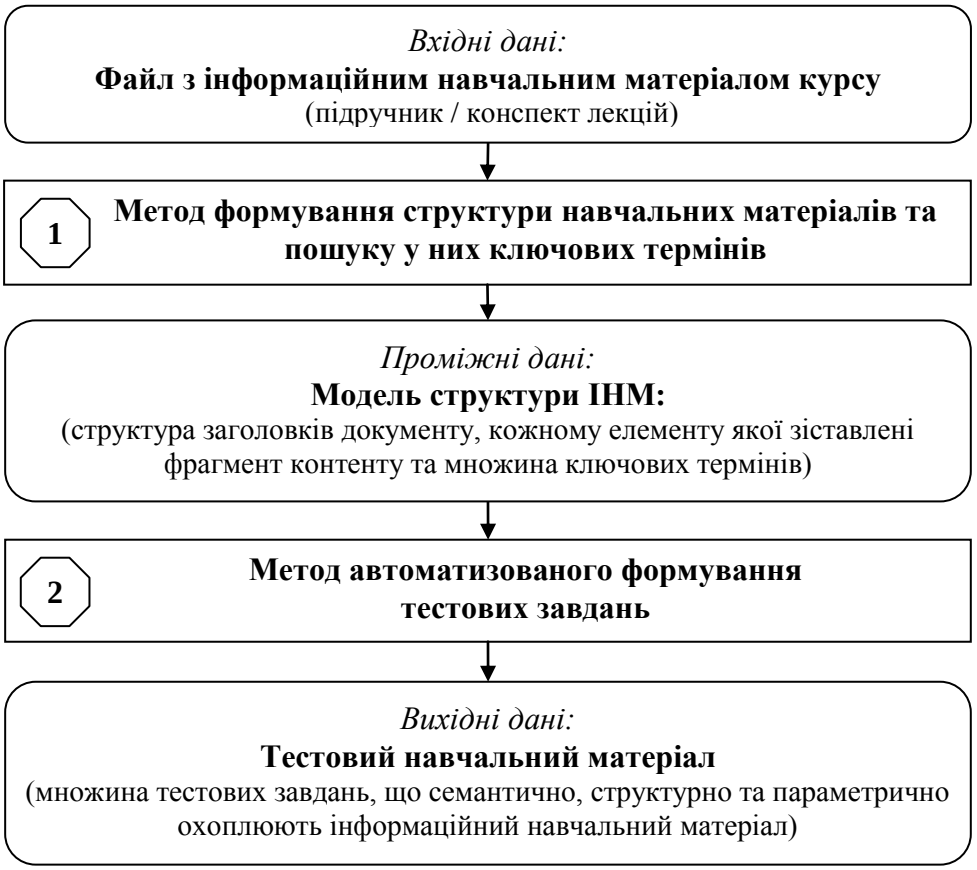


Рисунок 2.1 – Загальна схема застосування інформаційної технології

Вхідними даними ІТ автоматизованого створення тестів до навчальних матеріалів є файл електронного документу (наприклад, формату .docx) зі слабо структурованим текстовим контентом ІНМ, що містить структуру документу у вигляді заголовків та зіставлених їм текстових теоретичних відомостей. Таке

формування електронного документу ІНМ є загальноприйнятим і похідним від розмітки електронного текстового документу.

В процесі виконання блоку 1 (*метод формування структури навчальних матеріалів та пошуку у них ключових термінів*) ІТ автоматизовано визначається структура заголовків документу, кожному елементу якої зіставляється відповідний фрагмент контенту ІНМ. У кожному з цих фрагментів тексту автоматизовано визначаються подані за показником семантичної важливості множини ключових термінів. Визначення множин ключових термінів відбувається з використанням методу дисперсійного оцінювання, виконується оцінювання семантичної важливості ключових термінів, автоматизоване обмеження множин ключових термінів. Передбачена можливість коригування одержаних множин шляхом фільтрування за стилями документу, синтаксичними шаблонами або вручну, для уточнення результату.

Проміжні дані є результатом виконання блоку 1 і вхідними даними блоку 2.

В процесі виконання блоку 2 (*метод автоматизованого формування тестових завдань*) ІТ за допомогою правил продукції автоматизовано генеруються тестові завдання. Для цього кожен елемент контенту (речення) з кожної рубрики документу перевіряється на наявність ключових термінів з даної рубрики. Якщо термін присутній в реченні, то проводиться вибір правила продукції, яке задовольняє умовам. У результаті, автоматизовано створюється нове тестове завдання. Передбачена можливість коригування тексту тестового завдання та загальної кількості тестових завдань, що може покращити множину тестових завдань.

Вихідними даними ІТ є множина тестових завдань, які розрізняються за параметрами (тип запитання, кількість правильних відповідей, правило за яким сформоване тестове завдання, терміни які використовуються в завданні тощо) й можуть бути використані для перевірки рівня засвоєння відповідних знань за допомогою існуючих навчальних середовищ та систем тестування.

Одержана множина містить тестові завдання, що семантично, структурно та параметрично охоплюють відповідний вхідний ІНМ.

Таким чином, наповнення й використання моделі семантичної структури НК за допомогою наведеної ІТ дозволяє за вхідними даними у вигляді файлу електронного документу з ІНМ одержувати вихідні дані у вигляді множини тестових завдань. Також наповнення моделі відкриває можливість для подальшого вирішення ряду актуальних задач автоматизації, наприклад використання одержаної множини тестових завдань для проведення адаптивного тестування рівня засвоєння знань, визначення відповідності ІНМ вимогам, допомога при розробці ІНМ, оцінка відповідності наборів тестових завдань інформаційним навчальним матеріалам, допомога при створенні тестів тощо.

2.2 Інформаційна модель семантичної структури навчального курсу

До навчального курсу входить навчальний матеріал, який можна класифікувати як програмний, інформаційний, операційний, актуалізуючий, стимулюючий, діагностуючий тощо (див. п. 1.1.2). В межах даної роботи інформаційна модель семантичної структури навчального курсу містить формальне подання ІНМ та ТНМ.

2.2.1 Структура навчального курсу

Інформаційний навчальний матеріал в більшості випадків формується у вигляді підручника, навчального посібника чи конспекту лекцій, і є основним носієм інформації в навчальному курсі, призначеної для набуття знань та частини вмінь суб’єктом, що вивчає навчальний курс.

Діагностуючий навчальний матеріал містить завдання, які дозволяють оцінити рівень засвоєнні ІНМ, виявити прогалини в знаннях, причини неправильних дій суб'єкта, що вивчає навчальний курс.

Тестовий навчальний матеріал є найбільш розповсюдженим різновидом діагностуючого навчального матеріалу й призначений для визначення рівня засвоєння ІНМ шляхом використання комп'ютерного чи паперового тестування суб'єкта, що вивчає навчальний курс.

Для побудови інформаційної моделі узагальнимо матеріали навчального курсу як сукупність *множин сутностей* (заголовків, слів, ключових термінів, тестових завдань, зв'язків тощо) подану у вигляді ієрархічної структури.

Наразі ІНМ існують у вигляді слабо структурованих цифрових документів (наприклад, форматів .doc, .docx, .html, .pdf тощо). В роботі пропонується подати семантичну структуру ІНМ як ієрархічну структуру заголовків документа (рубрикація, наприклад: Дисципліна / Розділ / Тема) та співставлених заголовкам множин ключових термінів (рис. 2.2). Ключові терміни в таких множинах ранжовані за рівнем семантичної важливості в межах відповідних заголовкам фрагментів контенту документа.

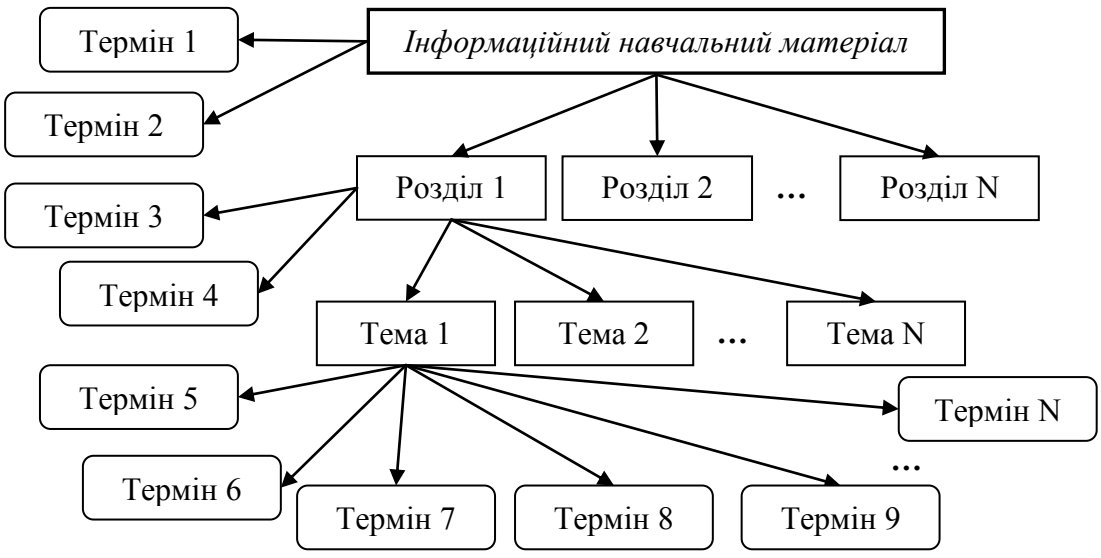


Рисунок 2.2 – Приклад графічного подання семантичної структури ІНМ

Ключовими термінами можуть виступати як слова, так і словосполучення й абревіатури (табл. 2.1).

Таблиця 2.1 – Підходи до пошуку ключових термінів різних типів

Тип терміну	**Підхід до формалізації**
Слово	Окремий елемент тексту, що характеризується неперервною (без прогалин/пробілів) сукупністю символів у тексті. До цього типу належать також слова з апострофами, двокореневі слова (з'єднані суфіксами) та складені слова (з'єднані дефісом) тощо. Одному слову відповідає один елемент множини унікальних слів документу. Приклади слів: «об'єкт», «дані», «змінна-відношення».
Словосполучення	Словосполучення є стійкими сукупностями важливих слів, що згруповані у визначеній послідовності та у такій комбінації неодноразово присутні в текстовому контенті. Синтакстично словосполучення як елемент тексту є сталою впорядкованою сукупністю слів. Одному словосполученню відповідає кілька елементів множини унікальних слів документу. Приклади словосполучень: «база даних», «штучний інтелект».
Абревіатури та скорочення	Є стійкою зв'язною сукупністю символів, тому може визначатись та оброблятись як слово. Одній абревіатурі чи скороченню відповідає один елемент множини унікальних слів документу. Приклади абревіатур та скорочень: «ШІ», «БД», «SQL», «шт.».

Для подання ІНМ у вигляді ієрархічної семантичної структури НМ, визначимо наступні множини сутностей ІНМ, що є сукупностями відповідних атрибутів (рис. 2.3): множина заголовків, множина термінів і множина зв'язків між сутностями.

До множини зв'язків, які відображають факт і властивості взаємозв'язку між сутностями, належать: зв'язки між заголовками та заголовками; зв'язки між заголовками та ключовими термінами.

Іншими словами, ІНМ подається деревоподібною структурою, кожен елемент якої є семантичним вузлом, причому нижні рівні подаються вузлами у

вигляді ключових термінів, а решта рівнів є заголовками для відповідних фрагментів контенту документа ІНМ.

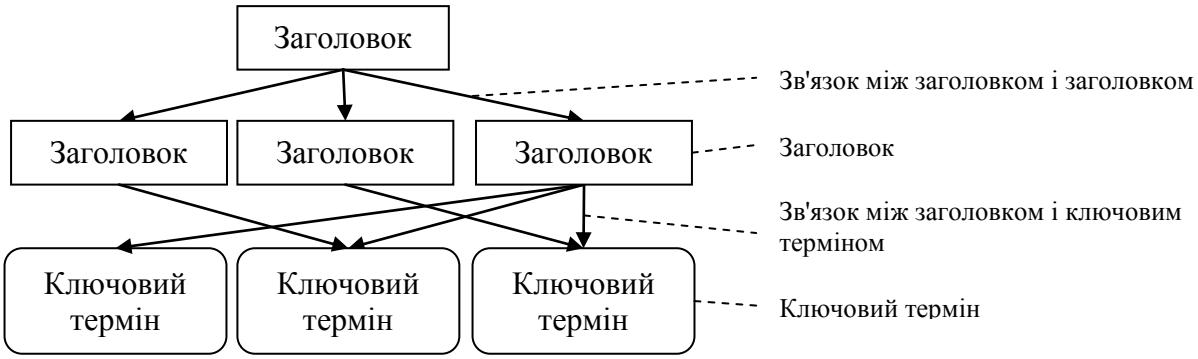


Рисунок 2.3 – Приклад використання елементів множини сутностей ІНМ для подання семантичної структури НМ

ТНМ є засобом перевірки рівня одержаних при вивченні ІНМ знань, зокрема засвоєння відповідних множин ключових термінів. ТНМ подається тестами. Тест формально є множиною однорідних елементів – тестових завдань; в процесі тестування із них формується тестувальна вибірка. Оскільки призначенням окремого тестового завдання є перевірка рівня засвоєння певного локального обсягу знань із ІНМ, що на практиці визначається рівнем розуміння й володіння відповідною термінологічною базою, то визначені терміни мають бути присутні в контенті тестового завдання. Відповідно, сутностями у моделі семантичної структури ТНМ є заголовки, ключові терміни, тестові завдання, а також зв’язки для визначення відношень між ними.

Запропоновано наступні множини сутностей ТНМ, що є сукупностями відповідних атрибутів (рис. 2.4): множина заголовків; множина тестових завдань; множина зв’язків між сутностями. До підмножини зв’язків, які відображають факт і властивості взаємозв’язку між сутностями, належать наступні різновиди: зв’язки між заголовками та ключовими термінами; зв’язки між заголовками та тестовими завданнями; зв’язки між ключовими термінами та тестовими завданнями.

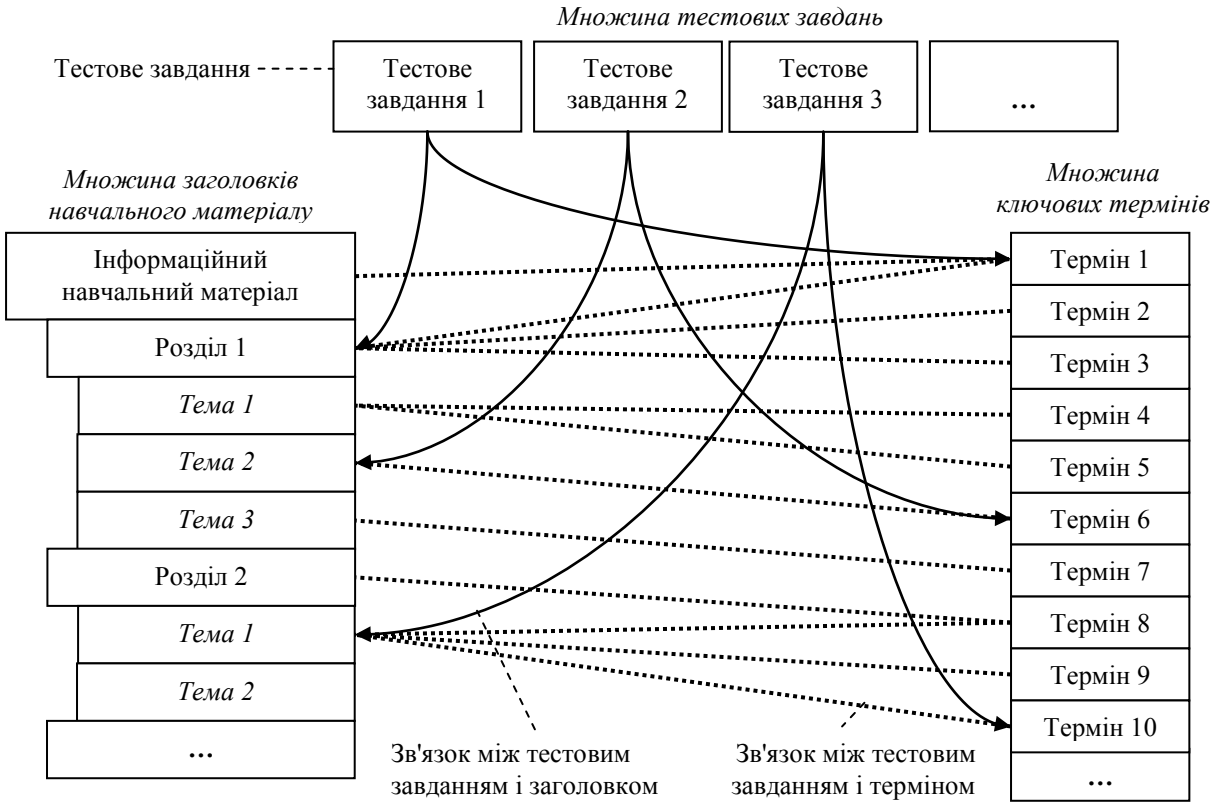


Рисунок 2.4 – Приклад зв’язків елементів множини тестових завдань з елементами множин ІНМ

Множину тестових завдань формують всі тестові завдання, створені для перевірки рівня засвоєння знань із відповідних ІНМ. Зв’язки вказують на наявність ключового терміну в контенті тестового завдання, що дозволяє використовувати тестове завдання для визначення рівня розуміння співставленого з ним ключового терміну.

2.2.2 Моделювання матеріалів навчального курсу

Розглянуту вище структуру навчального курсу (educational course, *EC*) можна подати у вигляді:

$$\{IEM, TEM\} \subset EC, \tag{2.1}$$

де *IEM* – інформаційний навчальний матеріал (informational education material), *TEM* – тестовий навчальний матеріал (testing education material).

Семантична структура *EC* як об'єднання *IEM* ∪ *TEM* може бути подана у вигляді:

$$IEM \cup TEM = \{M_{Heading} \cup M_{Term} \cup M_{TestEx} \cup M_{Rel}\}, \quad (2.2)$$

де $M_{Heading}$ – множина заголовків, M_{Term} – множина термінів, M_{TestEx} – множина тестових завдань, M_{Rel} – множина зв'язків.

Відповідно до (2.2), кожен елемент множини *IEM* ∪ *TEM* є елементом підмножини $M_{Heading}$, M_{Term}, M_{TestEx} або M_{Rel}:

$$IEM \cup TEM = \{x \mid x \in M_{Heading} \vee x \in M_{Term} \vee x \in M_{TestEx} \vee x \in M_{Rel}\} \quad (2.3)$$

В залежності від типів елементів, які сполучаються за допомогою елементів множини M_{Rel}, її структура може бути подана у вигляді:

$$M_{Rel} = M_{Rel:H\text{-}H} \cup M_{Rel:H\text{-}T} \cup M_{Rel:H\text{-}TE} \cup M_{Rel:T\text{-}TE}, \quad (2.4)$$

де $M_{Rel:H\text{-}H}$ – множина зв'язків між заголовками й заголовками, $M_{Rel:H\text{-}T}$ – множина зв'язків між заголовками й ключовими термінами, $M_{Rel:H\text{-}TE}$ – множина зв'язків між заголовками та тестовими завданнями, $M_{Rel:T\text{-}TE}$ – множина зв'язків між ключовими термінами та тестовими завданнями.

Відповідно до (2.4), кожен елемент множини M_{Rel} є елементом підмножини $M_{Rel:H\text{-}H}$, $M_{Rel:H\text{-}T}$, $M_{Rel:T\text{-}W}$, $M_{Rel:H\text{-}TE}$ або $M_{Rel:T\text{-}TE}$:

$$M_{Rel} = \{x \mid x \in M_{Rel:H\text{-}H} \vee x \in M_{Rel:H\text{-}T} \vee x \in M_{Rel:H\text{-}TE} \vee x \in M_{Rel:T\text{-}TE}\}. \quad (2.5)$$

Згідно (2.2) та (2.4), семантична структура ІНМ і ТНМ навчального курсу може бути приведена до вигляду (рис. 2.5):

$$IEM \cup TEM = \{M_{Heading} \cup M_{Term} \cup M_{TestEx} \cup M_{Rel:H\text{-}H} \cup M_{Rel:H\text{-}T} \cup M_{Rel:H\text{-}TE} \cup M_{Rel:T\text{-}TE}\}. \quad (2.6)$$

Кожна з складових *IEM* та *TEM* у моделі *EC* має власне подання й структуру.

Семантична структура ІНМ (*IEM*) може бути подана у вигляді:

$$\{M_{Heading} \cup M_{Term} \cup M_{Rel:H\text{-}H} \cup M_{Rel:H\text{-}T}\} \subset IEM \subset EC. \quad (2.7)$$

Відповідно до (2.7), кожен елемент множини *IEM* є елементом підмножини $M_{Heading}$, M_{Term}, $M_{Rel:H\text{-}H}$ або $M_{Rel:H\text{-}T}$:

$$IEM = \{x \mid x \in M_{Heading} \vee x \in M_{Term} \vee x \in M_{Rel:H\text{-}H} \vee x \in M_{Rel:H\text{-}T}\}. \quad (2.8)$$

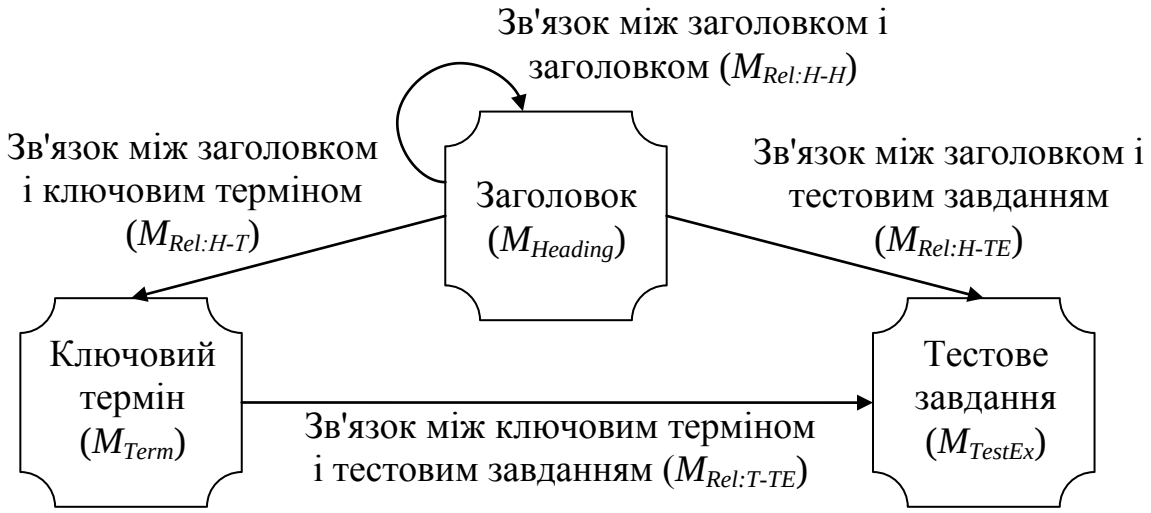


Рисунок 2.5 – Взаємозв’язок між параметрами моделі

Наведена сукупність множин (заголовків $M_{Heading}$, термінів M_{Term}, зв’язків $M_{Rel:H-H}$ і $M_{Rel:H-T}$) є достатньою для формування семантичної структури ІНМ. При цьому елементи множин заголовків і термінів формують вузли такої структури; а елементи множин зв’язків формують дуги.

Семантична структура ТНМ (*TEM)* може бути подана у вигляді:

$$\{M_{Heading} \cup M_{Term} \cup M_{TestEx} \cup M_{Rel:H-TE} \cup M_{Rel:T-TE}\} \subset TEM \subset EC. \quad (2.9)$$

Відповідно до (2.9), кожен елемент множини *TEM* є елементом підмножини $M_{Heading}$, M_{Term}, M_{TestEx}, $M_{Rel:H-TE}$ або $M_{Rel:T-TE}$:

$$TEM = \{x |\ x \in M_{Heading} \vee x \in M_{Term} \vee x \in M_{TestEx} \vee x \in M_{Rel:H-T} \vee x \in M_{Rel:H-TE} \vee x \in M_{Rel:T-TE}\}. \quad (2.10)$$

Згідно (2.7) та (2.9), множини $M_{Heading}$, M_{Term} та $M_{Rel:H-T}$ належать одночасно *IEM* та *TEM*:

$$IEM \cap TEM = \{M_{Heading} \cup M_{Term} \cup M_{Rel:H-T}\}. \quad (2.11)$$

Таким чином, виконано формальне подання семантичної структури навчального курсу у вигляді інформаційної моделі, що містить множину заголовків $M_{Heading}$, множину термінів M_{Term}, множину тестових завдань M_{TestEx} та множину зв’язків M_{Rel}.

2.2.3 Параметри моделі

Множини, що використовувались при формуванні інформаційної моделі НК, є її параметрами. Кожен параметр визначає структуру елементів, що входять до нього. Далі розглянуто структуру множин: заголовків $M_{Heading}$, термінів M_{Term}, тестових завдань M_{TestEx} і зв'язків M_{Rel}.

Множина заголовків. Заголовком у даній роботі вважається задана автором ІНМ семантично значуща назва певного змістовного блоку – розділу, теми, параграфу тощо. Множина заголовків M_{Heding} складається з цих назв.

Зважаючи на існуючі загальноприйняті вимоги до структури інформаційного навчального матеріалу курсу (наприклад, «Назва дисципліни / Розділ / Тема; Розділ / Підрозділ / Параграф» тощо), можна зробити висновок про відповідність заголовків фрагментам контенту ієрархічної моделі. Таким чином, до множини заголовків відносяться всі виділені автором назви фрагментів контенту (розділів, параграфів, тем, лекцій тощо) з можливим обмеженням за рівнем вкладеності.

Кожен елемент множини заголовків M_{Heding} є кортежем наступного вигляду:

$$m_{Heding} = (ID, HName, HContent, Grade), \tag{2.12}$$

де атрибут *ID* – унікальний ідентифікатор елементу, *HName* – назва заголовку, *HContent* – вміст відповідної рубрики, *Grade* – рівень заголовку в ієрархічній структурі.

Відповідно, для цих атрибутів існують множини ідентифікаторів M_{ID}, назв заголовків M_{HName}, контентів $M_{HContent}$ та рівнів в ієрархії M_{Grade}, які дозволяють формувати впорядковані четвірки множини M_{Heding}:

$$\begin{aligned} M_{\text{Heding}} = \{ & (ID, HName, HContent, Grade) \mid ID \in Z, \\ & HName \in M_{HName}, HContent \in M_{HContent}, Grade \in M_{Grade} \}. \end{aligned} \tag{2.13}$$

До множини M_{HName} входять символьні (текстові) назви всіх присутніх в ІНМ ідентифікаторів фрагментів контенту.

До множини $M_{HContent}$ входять відповідні рубрикам вмісти форматованого тексту або адреси їх знаходження.

Множина M_{Grade} містить позначення рівнів. Відповідно, потужність множини M_{Grade} дорівнює кількості рівнів в ієрархічній структурі заголовків.

Множина ідентифікаторів M_{ID} містить унікальні цілі числа ($M_{ID} \in Z$), для однозначної ідентифікації елементів у множині M_{Heding}.

У зв'язку з тим, що припустиме існування тотожних комбінацій значень *HName* та *Grade* для елементів множини M_{Heding}, то включення до кортежу атрибуту *ID* дозволяє уникнути неоднозначності при роботі з елементами множини M_{Heding}.

Множина ключових термінів. Множина ключових термінів M_{Term} складається з визначених для ІНМ ключових термінів. Ключові терміни є семантично значущими назвами понять, розуміння яких є обов'язковим для ефективного засвоєння контенту певного фрагменту ІНМ, одержання відповідних знань і вмінь.

Іншими словами, кожен елемент множини термінів M_{Term} є кортежем наступного вигляду:

$$m_{Term} = (ID, TermName, TermNum), \tag{2.14}$$

де атрибут *ID* – унікальний ідентифікатор елементу, *TermName* – символьна назва терміну, *TermNum* – кількість слів у терміні.

Для цих атрибутів існують множини назв термінів $M_{TermName}$ та кількостей слів у терміні $M_{TermNum}$, які дозволяють формувати впорядковані трійки множини M_{Term}:

$$\begin{aligned} M_{Term} = \{ & (ID, TermName, TermNum) \mid ID \in Z, \\ & TermName \in M_{TermName}, TermNum \in M_{TermNum} \}. \end{aligned} \tag{2.15}$$

Множину ідентифікаторів M_{ID} складають унікальні цілі числа ($M_{ID} \in Z$), що потрібні для однозначної ідентифікації елементів у множині M_{Term}.

До множини назв термінів $M_{TermName}$ входять символьні назви ключових термінів у тому вигляді, в якому вони присутні в контенті ІНМ, включаючи леми. Лемами є похідні словоформи, що одержуються в результаті зміни

параметрів нормальної форми терміна (наприклад, відмінок) згідно синтаксичних взаємозв'язків слів у тексті. Кожен елемент множини $M_{TermName}$ є цілком впорядкованою множиною слів, з яких складається термін.

Множина $M_{TermNum}$ містить кількості слів, що входять до відповідних термінів з множини M_{Term}. Потужність множини $M_{TermNum}$ визначається максимальною кількістю слів, що може входити до складу терміна. Ключовий термін може бути словом, словосполученням чи абревіатурою, й атрибут *TermNum* може бути використаний для вказання типу терміну. При *TermNum* = 1 ключовий термін є словом, скороченням слова чи абревіатурою, при *TermNum* ≠ 1 ключовий термін є словосполученням.

Множина тестових завдань. Множина тестових завдань M_{TestEx} містить всі тестові завдання. Властивості (атрибути) кожного елемента в множині є незалежними від матеріалу, рівень засвоєння якого перевіряється. Атрибути, що визначаються методом експертної оцінки та не впливають на структуру й інші властивості тестового завдання (час тестування, рівень складності тощо) в даній моделі не розглядаються.

Кожен елемент множини тестових завдань M_{TestEx} є кортежем наступного вигляду:

$$m_{TestEx} = (ID, Type, TEContent, Answers), \qquad (2.16)$$

де *ID* – унікальний ідентифікатор елементу, *Type* – тип питання, *TEContent* – контент власне тестового завдання, *Answers* – кількість відповідей.

Для цих атрибутів тестового завдання існують множини ідентифікаторів M_{ID}, типів питань M_{Type}, контентів тестових завдань $M_{HContent}$, кількостей відповідей $M_{Answers}$, які дозволяють формувати впорядковані четвірки множини M_{TestEx}:

$$M_{TestEx} = \{(ID, Type, TEContent, Answers) \mid ID \in M_{ID}, \\ Type \in M_{Type}, TEContent \in M_{TEContent}, Answers \in Z\}. \qquad (2.17)$$

Множину ідентифікаторів M_{ID} складають унікальні назви (наприклад, цілі числа $M_{ID} \in Z$), що використовуються для однозначної ідентифікації елементів у множині M_{TestEx} за їх значеннями.

До множини типів питань M_{Type} входять цілі числа ($Type \in Z$), що вказують на номер типу питання в тестовому завданні. Наприклад, сучасними середовищами навчання передбачено питання множинного вибору, логічного вибору (Так/Ні), відповідності, короткої відповіді, числової відповіді, есе та вбудованої відповіді. Параметр типу питання безпосередньо впливає на логічну однорідність структури тесту й є первинним класифікатором тестових завдань.

До множини $M_{TEContent}$ входять власне тестові завдання або адреси їх знаходження.

До множини кількостей відповідей $M_{Answers}$ входять цілі числа ($Answers \in Z$), що визначають, скільки правильних відповідей встановлено у тестовому завданні. Кількість правильних відповідей може варіюватися в залежності від типу завдання й виділяється одна або кілька правильних відповідей. Цей параметр потрібен для деяких систем тестування (наприклад, Moodle за ним визначає тип деяких тестових завдань) та може використовуватись при обрахунку складності тестового завдання.

Атрибути кожного елементу множини M_{TestEx} дозволяють класифікувати тестові завдання за властивостями, інваріантними до рубрик ІНМ або ключових термінів, які використані в них. Для визначення відношень між тестовими завданнями та заголовками й ключовими термінами використовуються елементи відповідних множин зв’язків.

Множина зв’язків між елементами семантичної структури. До множини зв’язків M_{Rel} входять елементи, які визначають зв’язки між двома елементами з множин $M_{Heading}$, M_{Term} та M_{TestEx}. Окрім визначення зв’язку між двома атрибутами, елементи множини M_{Rel} можуть містити додаткові атрибути, що визначають характер цього зв’язку.

Відповідно, кожен елемент множини зв'язків є кортежем наступного вигляду:

$$m_{Rel} = (TypeRel, Obj1, Obj2, Feature), \quad (2.18)$$

де *TypeRel* – ціле число ($TypeRel \in Z$), що вказує на тип зв'язку; *Obj*1– перша сутність з співвідношення; *Obj*2 – друга сутність з співвідношення; *Feature* – атрибут, що вказує на характеристику зв'язку.

Згідно з (2.4), в залежності від приналежності атрибутів *Obj*1 та *Obj*2 окремим множинам з переліку ($M_{Heading}$, M_{Term}, M_{TestEx}), атрибут *TypeRel* приймає значення, вказані у табл. 2.2.

Таблиця 2.2 – Перелік значень атрибута *TypeRel*

Множина зв'язків	**Значення *TypeRel***	**Приналежність Obj1**	**Приналежність Obj2**	**Значення Feature**
$M_{Rel:H\text{-}H}$	1	$M_{Heading}$	$M_{Heading}$	відсутнє (Null)
$M_{Rel:H\text{-}T}$	2	$M_{Heading}$	M_{Term}	числовий показник важливості ключового терміну
$M_{Rel:H\text{-}TE}$	3	$M_{Heading}$	M_{TestEx}	числовий показник номеру використаного речення
$M_{Rel:T\text{-}TE}$	4	M_{Term}	M_{TestEx}	тип використання терміна у тестовому завданні

Таким чином, модель семантичної структури навчального курсу містить наступні складові: множину заголовків $M_{Heading}$, множину термінів M_{Term}, множину тестових завдань M_{TestEx}, множину зв'язків M_{Rel}. При цьому, визначено наступні підмножини M_{Rel}: між заголовками та заголовками $M_{Rel:H\text{-}H}$, між заголовками та ключовими термінами $M_{Rel:H\text{-}T}$, між заголовками та тестовими завданнями $M_{Rel:H\text{-}TE}$, між ключовими термінами та тестовими завданнями $M_{Rel:T\text{-}TE}$.

Для наповнення множин моделі, застосовуються методи:

– метод формування структури навчальних матеріалів та пошуку в них ключових термінів для визначення елементів моделі: множини заголовків,

множини зв’язків між заголовками, множини ключових термінів, множини зв’язків між заголовками і термінами;

– метод автоматизованого формування тестових завдань для визначення елементів моделі: множини тестових завдань, множини зв’язків між ключовими термінами та тестовими завданнями, множини зв’язків між заголовками і тестовими завданнями.

Використання цих методів дозволяє здійснювати повне визначення елементів моделі.

2.3 Метод формування структури навчальних матеріалів та пошуку в них ключових термінів

Метод призначений для автоматизованого визначення структури документу ІНМ. У ньому кожному елементу рубрики зіставляється відповідний фрагмент контенту ІНМ та визначаються за показником семантичної важливості обмежені множини ключових термінів. Загальну схему методу наведено на рис. 2.6.

Вхідними даними методу є електронний документ ІНМ (файл слабо структурованого текстового документу *IEM*, наприклад з розширенням .docx) для обробки та множина параметрів обмеження моделі, до яких належать максимальна кількість слів у терміні – n та гранична щільність ключових термінів у тексті – Q.

У *Блоці 1* автоматизовано визначається структура з $q = |M_{Heading}|$ рубрик документу, кожному елементу якої зіставляється відповідний фрагмент контенту ІНМ. Як інструмент парсингу використовуються існуючі програмні та методологічні засоби об’єктного представлення вмісту електронних текстових документів, наведені в п. 1.1.3. Для одержання ієрархічної структури рубрик документу $IEM \Rightarrow G' = M_{Heading} \cup M_{Rel:H\text{-}H}$ виконуються такі дії.

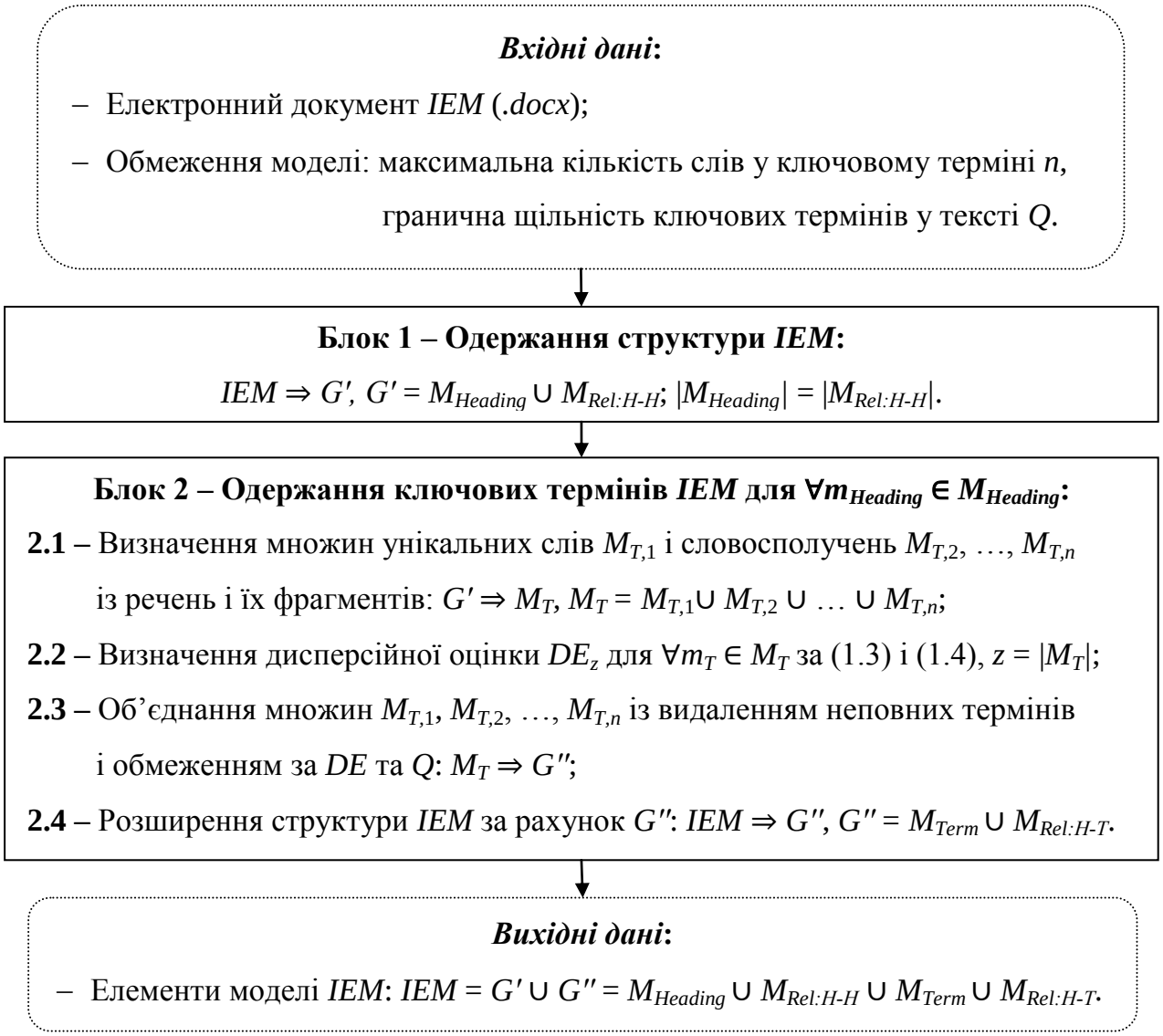


Рисунок 2.6 – Загальна схема методу формування структури навчальних матеріалів та пошуку в них ключових термінів

1. Властивості стилів кожного абзацу перевіряються на використання стилів рубрикації (заголовків). Для кожного заголовку визначається рівень в ієрархії, в результаті одержуються атрибути *ID*, *HName* та *Grade* кортежів множини $M_{Heading}$.

2. Рівень *Grade* кожного заголовку з $M_{Heading}$ порівнюється з рівнями попередніх заголовків для визначення його підпорядкованості. У результаті одержуються атрибути *Obj*1 та *Obj*2 кортежів $M_{Rel:H\text{-}H}$. При цьому кожному елементу $M_{Heading}$ ставиться у відповідність один елемент $M_{Rel:H\text{-}H}$ (кореневий елемент позначається зв'язком самого з собою): $|M_{Heading}| = |M_{Rel:H\text{-}H}|$.

3. Визначається контент кожного з q елементів $M_{Heading}$ наступним чином: контентом елементу $z = \overline{1,q}$ множини $M_{Heading}$ є фрагмент контенту *IEM* починаючи від заголовку цього елементу й закінчуючи початком наступного заголовку, рівень якого не нижчий за рівень елемента *z*. При цьому контент елементів-нащадків одночасно входить до складу контенту елемента-предка в ієрархічній структурі заголовків. В результаті одержується атрибут *HContent* кортежів множини $M_{Heading}$ (рис. 2.7).

Рисунок 2.7 – Схема обробки контенту та формування структури ІНМ

У *Блоці 2* для кожного зі співставлених заголовкам *z* фрагментів контенту ІНМ $\forall m_{Heading} \in M_{Heading}$ визначається за показником семантичної

важливості множина ключових термінів. Для цього використовуються наступні кроки.

Для визначення множин унікальних слів і словосполучень (*Крок 2.1*) спершу формується подання обраного елементу $M_{Heading}$ як впорядкованої множини слів M_{TXT}. Множина M_{TXT} складається зі слів тексту в порядку їх слідування у документі.

Кожен елемент M_{TXT} визначається як неперервна послідовність допустимих символів. Наприклад, при обробці ІНМ, виконаних українською мовою, до такої множини будуть включені неперервні послідовності символів з наступної множини допустимих символів у словах $M_{SymWord}$:

$$M_{SymWord} = M_{abc} \cup M_{ABC} \cup M_{абв} \cup M_{АБВ} \cup M_{123} \cup M_{SymOK}, \qquad (2.19)$$

де M_{abc} – множина символів англійського алфавіту латиницею в нижньому регістрі, M_{ABC} – множина символів англійського алфавіту латиницею в верхньому регістрі, $M_{абв}$ – множина символів українського алфавіту кирилицею в нижньому регістрі, $M_{АБВ}$ – множина символів українського алфавіту кирилицею в верхньому регістрі, M_{123} – множина арабських цифр (M_{123} ={0, 1, 2, 3, 4, 5, 6, 7, 8, 9}), M_{SymOK} – множина допустимих символів всередині слів (M_{SymOK} ={’, -}, апостроф і дефіс).

Кожному слову в множині M_{TXT} зіставляється порядковий номер, за допомогою якого в подальшому при дисперсійному оцінюванні відбувається пошук відстаней між однаковими термінами. Якщо термін складається з кількох слів, то при визначенні його порядкового номера (позиції) приймається номер першого слова.

Окремо виконується пошук меж текстових контейнерів (фраз) з метою розбиття відповідного елементові множини $M_{Heading}$ фрагменту контенту цифрового документу на фрази. Під фразою мається на увазі семантично цілісний вузол, що виокремлений автором стилістичним форматуванням тексту чи розділовими знаками, й локалізує місцезнаходження окремих термінів. Терміни не можуть виходити за межі фраз. Відтак, межами текстового контейнеру є:

– зміни стилістичних властивостей тексту;

– розділові знаки;

– абзаци/параграфи.

Після цього формується множина унікальних слів і словосполучень тексту $M_T = M_{T,1} \cup M_{T,2} \cup \ldots \cup M_{T,n}$; кожному елементу M_T зіставляється кількість появ у фразах досліджуваного елементу множини $M_{Heading}$. До множини унікальних слів $M_{T,1}$ належать всі унікальні слова тексту. До множин унікальних словосполучень $\{M_{T,2}, \ldots, M_{T,n}\}$ належать всі унікальні послідовності слів тексту, що не виходять за межі фраз та мають довжину відповідно від 2 до n слів.

Обрахунок відстаней між однаковими елементами $m_T \in M_T$ в тексті є початковим на *кроці 2.1* – дисперсійному оцінюванні термінів, за якого для $\forall m_T \in M_T$ визначаються всі відстані ΔA між сусідніми їх появами. На відміну від існуючої формули обрахунку відстаней (1.3), яка для k появ елементу визначає $k - 1$ відстаней, використовується одержання k відстаней у наступний спосіб. Для кожної k-ї появи m_T з множини M_{T1} в множині M_{TXT} на позиції m_k^t обраховується одне значення відстані $\Delta A_{m,k}^t$:

$$\Delta A_{m,k}^t = \begin{cases} 0, \ \textit{якщо } m_{\min}^t = m_{\max}^t; \\ m_{\min}^t + \left|M_{TXT}\right| - m_{\max}^t, \ \textit{якщо } m_k^t = m_{\max}^t; \\ m_{k+1}^t - m_k^t, \ \textit{в іншому випадку} \end{cases} \qquad (2.20)$$

де $m_{\min}^t$ – позиція першої появи m_T у множині M_{TXT}, (ΔT^2) $m_{\max}^t$ – позиція останньої появи m_T у множині M_{TXT}.

Значення дисперсійної оцінки DE_i обраховується згідно з (1.4) та (2.20) для кожного з $i = \overline{1, z}$ елементів множини M_{T1} за відстанями між послідовними появами цього терміну в тексті, $z = |M_T|$.

Оскільки при формуванні множин $M_{T,1}$, $M_{T,2}$, …, $M_{T,n}$ до них додавались усі можливі варіанти слів і словосполучень в межах фраз без поглинання

більшими словосполученнями менших, на *кроці 2.3* проводиться аналіз необхідності такого поглинання.

Поглинання елементів у множині M_T проводиться, якщо вони переважно (більше ніж у половині випадків) є частиною іншого елемента із більшою кількістю слів. Якщо в множині M_T існує елемент t_1 (k_1 – кількість появ t_1 в множині M_T), що є впорядкованою множиною з x_1 слів, та елемент t_2 (k_2 – кількість появ t_2 в множині M_T), що є впорядкованою множиною з x_2 слів, причому t_1 є підмножиною t_2 ($t_1 \subset t_2$), а $x_1<x_2$, то при виконанні умови $2k_1<k_2$ елемент t_1 видаляється з множини M_T. У результаті в множині M_T видаляються елементи, які використовуються переважно як частини інших елементів.

Після цього з M_T видаляються всі елементи, для яких $DE_i = 0$. Таке значення вказує на те, що даний термін присутній у тексті лише один раз, й не може бути інтерпретований як ключовий.

Елементи в множині M_T сортуються за значенням їх дисперсійної оцінки *DE*, після чого їх кількість обмежується відповідно до вхідного параметру граничної щільності ключових термінів у тексті *Q*. Щільність ключових слів *Q* є відношенням кількості слів ключових термінів у тексті до загальної кількості слів у тексті й для навчальних матеріалів становить 0,11-0,15; цей параметр може бути перевизначений у вхідних даних. Відповідно, до порожньої множини ключових термінів M_{Term} додаються елементи з множини M_T з найбільшими значеннями *DE* доти, доки справджується рівність:

$$\sum_{i=1}^{n} \frac{K_n x_n}{X_{txt}} \le Q, \qquad (2.21)$$

де K_n – кількість появ терміну n в множині M_{TXT}; x_n – кількість слів у терміні n; X_{txt} – загальна кількість слів у тексті; n – поточна кількість термінів у множині M_{Term}.

В результаті одержуються всі атрибути кортежів множин M_{Term} та $M_{Rel:H\text{-}T}$. При цьому за (2.18) атрибути кортежів множин M_{Term} вказують, в контенті якого елементу множини $M_{Heading}$ ($Obj1$) який термін із множини M_{Term}

(*Obj*2) має яке значення *DE* (*Feature*). Значення *Feature* розглядається як числовий показник семантичної важливості терміну *Obj*2 у рубриці *Obj*1.

Вихідними даними методу є структура ІНМ $G' = M_{Heading} \cup M_{Rel:H\text{-}H}$ та множини ключових термінів для кожного елементу структури $G'' = M_{Term} \cup M_{Rel:H\text{-}T}$.

Таким чином, метод формування структури навчальних матеріалів та пошуку у них ключових термінів дозволяє визначити елементи наступних множин моделі семантичної структури НК: $M_{Heading}$, M_{Term}, $M_{Rel:H\text{-}H}$, $M_{Rel:H\text{-}T}$.

2.4 Метод автоматизованого формування тестових завдань

Основною метою використання моделі семантичної структури НК є забезпечення автоматизованого формування множини тестових завдань M_{TestEx} різних типів до заданого ІНМ чи його визначеного елементу $M_{Heading}$, забезпечуючи при цьому повне охоплення набором тестових завдань семантики ІНМ у вигляді відповідної множини ключових термінів M_{Term}.

2.4.1 Призначення методу

Метод автоматизованого формування тестових завдань дозволяє наповнити наступні множини моделі семантичної структури НК:

– множина тестових завдань M_{TestEx};

– множина зв’язків між заголовками і тестовими завданнями $M_{Rel:H\text{-}TE}$;

– множина зв’язків між ключовими термінами і тестовими завданнями $M_{Rel:T\text{-}TE}$.

Множина тестових завдань M_{TestEx}, кожен елемент якої відповідно до (2.17) є кортежем вигляду M_{TestEx} = (*ID*, *Type*, *TEContent*, *Answers*), містить всі тестові завдання визначеного тесту до заданого ІНМ чи його визначеної частини.

Атрибут *TEContent* кожного елементу множини M_{TestEx} містить контент тестового завдання чи вказівку на його місцезнаходження. До контенту тестового завдання належать наступні елементи:

– контент запитання;

– множина відповідей, кожен елемент якої є кортежем виду: контент відповіді; оцінка відповіді;

– множина опцій тестового завдання, кожен елемент якої є кортежем виду: тип питання, час відповіді, максимальна кількість балів, рівень складності.

Множину зв'язків між заголовками і тестовими завданнями $M_{Rel:H\text{-}TE}$, кожен елемент якої відповідно до (2.17) та табл. 2.2 є кортежем вигляду $M_{Rel:H\text{-}TE}$ = (3, *Heading*, *TestEx*, *Cont*), складають елементи, що визначають зв'язки між одним елементом множини $M_{Heading}$ та одним елементом множини M_{TestEx}. Заголовок визначає межі контенту, в яких актуальне тестове завдання. Такий зв'язок дозволяє встановити, рівень знання якого елементу ІНМ перевіряє конкретне тестове завдання.

Множину зв'язків між ключовими термінами і тестовими завданнями $M_{Rel:T\text{-}TE}$ складають елементи, які визначають зв'язки між одним елементом множини M_{Term} та одним елементом множини M_{TestEx}, й кожен її елемент відповідно до (2.17) та табл. 2.2 є кортежем вигляду $M_{Rel:T\text{-}TE}$ = (4, *Term*, *TestEx*, *Loc*). Оскільки у контенті одного тестового завдання (як запитання, так і множини відповідей) можуть використовуватись кілька ключових термінів, то й відповідних одному тестовому завданню зв'язків із ключовими термінами може бути кілька.

2.4.2 Правила продукції тестових завдань

Процедурні знання чи правила є набором певних процедур для перетворення знань як даних. Модель продукції найкраще відображає процедурний характер знань. Основним конструктивним елементом такої

моделі є правило продукції, яке можна подати так: *ЯКЩО <умова> ТО <висновок чи дія>*, тобто правило складається з умовної та дієвої частини. Можливість використання продукційних правил для створення тестових завдань була визначена в п. 1.2.4.

Умовою (антецедентом) є деяке речення-шаблон, за яким здійснюється пошук, а дією (консеквентом) – алгоритм перетворення речення в контент складових тестового завдання, що виконуються при успішних результатах пошуку (рис. 2.8).

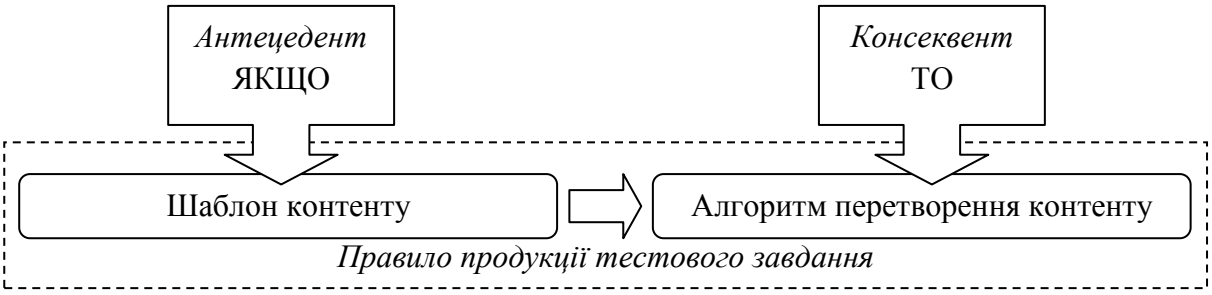


Рисунок 2.8 – Схема продукційного правила для формування прототипів тестових завдань

Прикладом продукційного правила для генерації прототипу тестових завдань може бути подане на рис. 2.9. В даному випадку, антецедент визначає три вимоги до речення, у випадку одночасного виконання яких правило активується. При застосуванні антецеденту використовуються активний термін та множина сполучних фрагментів. Консеквент визначає послідовність дій із чотирьох кроків, необхідну для формулювання контенту тестового завдання. При застосуванні консеквенту використовуються: контент речення, активний термін та множина ключових термінів для даного фрагменту ІНМ. Таким чином, множина правил продукції M_{Rule} є інструментом для створення тестових завдань.

Кожне правило продукції тестових завдань $\forall m_{Rule} \in M_{Rule}$ є кортежем із двох елементів – антеценеднта та консеквента, які формують імплікацію:

$$m_{Rule} = (a \Rightarrow c), \tag{2.22}$$

де $a \in A$ – антецедент правила (A – множина антецедентів), $c \in C$ – консеквент правила (C – множина консеквентів).

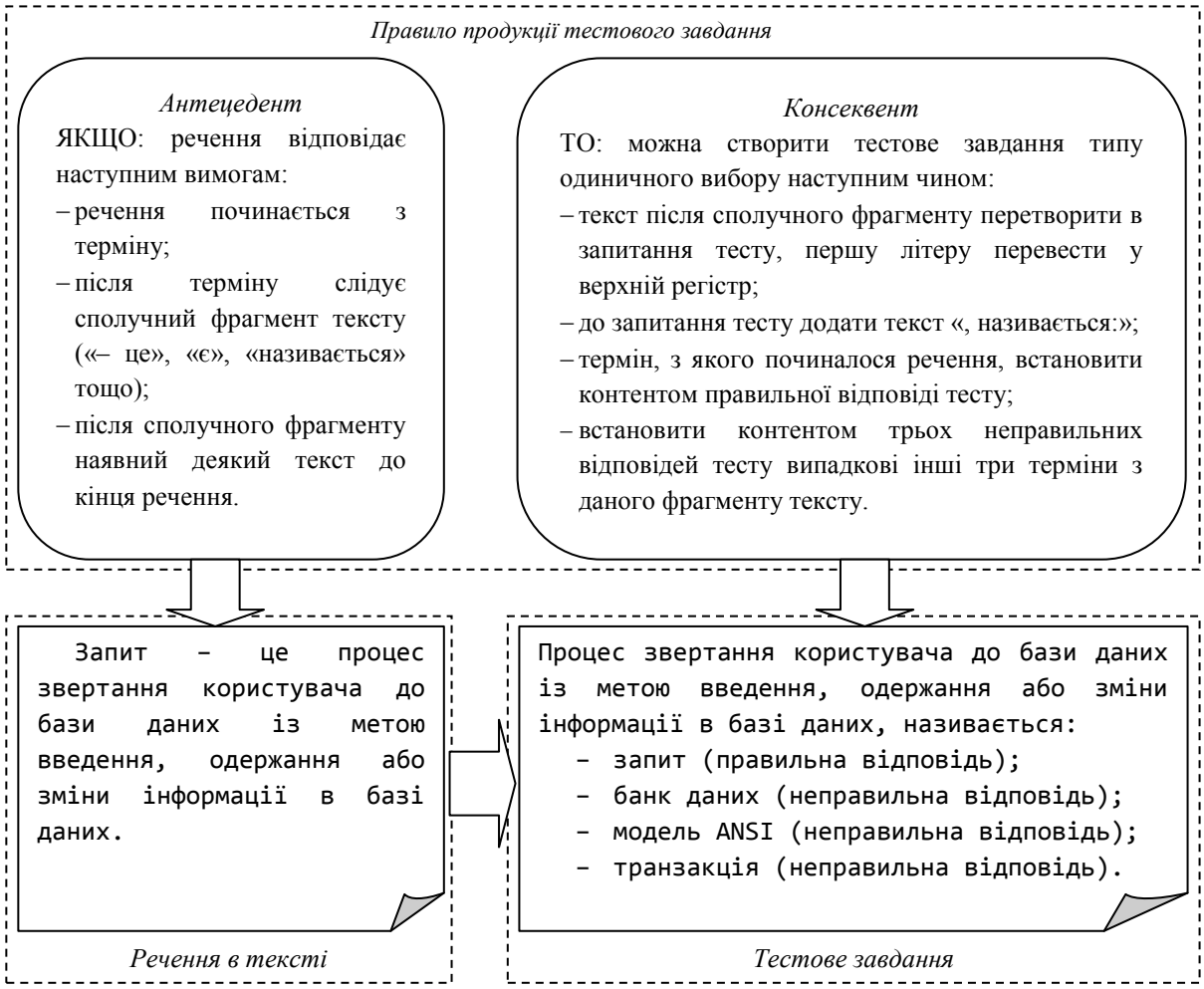


Рисунок 2.9 – Приклад продукційного правила для створення прототипів тестових завдань

Проведені дослідження, що полягали в зіставленні створених авторами вручну тестових завдань та використаних для цього фрагментів ІНМ, параметри яких наведені в Додатку Д, дозволили встановити існуючі типи консеквентів a та антецедентів c.

Множина антецедентів містить 2 елементи:

$$A = \{a_1, a_2\}, \tag{2.23}$$

де означальний антецедент a_1 є формальним поданням речення, в якому присутній термін та його синтаксично виражене означення; згадувальний

антецедент a_2 є формальним поданням речення, в якому присутній термін без його синтаксично вираженого означення.

Відповідно до п. 1.2.3, необхідно створити тестові завдання наступних типів: логічного типу, одиничного вибору, множинного вибору та короткої відповіді. Кожен консеквент $c \in C$ містить алгоритм створення тестових завдань одного визначеного типу. Множина консеквентів містить 11 елементів:

$$C = \{c_{a1}, c_{a2}, c_{a3}, c_{b1}, c_{b2}, c_{b3}, c_{b4}, c_{c1}, c_{c2}, c_{d1}, c_{d2}\}. \quad (2.24)$$

Для створення тестових завдань логічного типу використовуються консеквенти:

c_{a1} – для постановки логічного питання до речення чи його частини без семантичних змін;

c_{a2} – для постановки логічного питання до речення чи його частини з заміною терміну чи його означення на некоректний варіант;

c_{a3} – для постановки логічного питання до речення чи його частини з додаванням заперечних часток.

Для створення тестових завдань одиничного вибору використовуються консеквенти:

c_{b1} – на вибір коректного терміну на задане означення терміна;

c_{b2} – на вибір коректного означення, що відповідає заданому терміну;

c_{b3} – на вибір одного коректного переліку;

c_{b4} – на вибір коректного терміна, пропущеного в реченні.

Для створення тестових завдань множинного вибору використовуються консеквенти:

c_{c1} – на вибір відповідних заданому терміну кількох коректних речень, дистрактори формуються шляхом заміни інших термінів у реченнях на заданий;

c_{c2} – на вибір кількох елементів з переліку, релевантних заданому реченню.

Для створення тестових завдань із короткою відповіддю (завдання на введення тексту) використовуються консеквенти:

c_{d1} – на введення терміну, відповідного означенню;

c_{d2} – на введення терміну, пропущеного у реченні.

Не кожен консеквент може імплікуватися з визначеного антецеденту. Можливі комбінації елементів $a \in A$ та $c \in C$ наведено в табл.. 2.3. Вони формують множину M_{Rule}, що містить 17 елементів:

$$M_{Rule} = \{m_{1a1}, m_{2a1}, m_{1a2}, m_{2a2}, m_{1a3}, m_{2a3}, m_{1b1}, m_{1b2}, m_{1b3}, m_{2b3}, m_{2b4}, m_{1c1}, m_{2c1}, m_{1c2}, m_{2c2}, m_{1d1}, m_{2d2}\}. \quad (2.25)$$

Таблиця 2.3 – Можливі комбінації антецедентів і консеквентів правил продукції

Консеквенти	**Правила по антецедентах**	
	a_1	a_2
c_{a1} – для постановки логічного питання до речення чи його частини без семантичних змін	$m_{1a1} = (a_1 \Rightarrow c_{a1})$	$m_{2a1} = (a_2 \Rightarrow c_{a1})$
c_{a2} – для постановки логічного питання до речення чи його частини з заміною терміну чи його означення на некоректний варіант	$m_{1a2} = (a_1 \Rightarrow c_{a2})$	$m_{2a2} = (a_2 \Rightarrow c_{a2})$
c_{a3} – для постановки логічного питання до речення чи його частини з додаванням заперечних часток	$m_{1a3} = (a_1 \Rightarrow c_{a3})$	$m_{2a3} = (a_2 \Rightarrow c_{a3})$
c_{b1} – на вибір коректного терміну на задане означення терміна	$m_{1b1} = (a_1 \Rightarrow c_{b1})$	–
c_{b2} – на вибір коректного означення, що відповідає заданому терміну	$m_{1b2} = (a_1 \Rightarrow c_{b2})$	–
c_{b3} – на вибір одного коректного переліку	$m_{1b3} = (a_1 \Rightarrow c_{b3})$	$m_{2b3} = (a_2 \Rightarrow c_{b3})$
c_{b4} – на вибір коректного терміна, пропущеного в реченні	–	$m_{2b4} = (a_2 \Rightarrow c_{b4})$
c_{c1} – на вибір відповідних заданому терміну кількох коректних речень, дистрактори формуються шляхом заміни інших термінів у реченнях на заданий	$m_{1c1} = (a_1 \Rightarrow c_{c1})$	$m_{2c1} = (a_2 \Rightarrow c_{c1})$
c_{c2} – на вибір кількох елементів з переліку, релевантних заданому реченню	$m_{1c2} = (a_1 \Rightarrow c_{c2})$	$m_{2c2} = (a_2 \Rightarrow c_{c2})$
c_{d1} – на введення терміну, відповідного означенню	$m_{1d1} = (a_1 \Rightarrow c_{d1})$	–
c_{d2} – на введення терміну, пропущеного у реченні	–	$m_{2d2} = (a_2 \Rightarrow c_{d2})$

Наведені множини антецедентів і консеквентів були визначені за результатом ручного порівняння (наведено у Додатку Д) контенту 31 навчального курсу з модульного середовища ХНУ [180] та середовища заочно-дистанційної освіти ХНУ [181] й створених для їх перевірки відповідно 31 тестів (містять загалом 1983 тестових завдання). Слід зауважити, що при цьому 11,70% склали тестові завдання, в яких були використані дистрактори поза контенту ІНМ, 6,45% склали тестові завдання, в яких були використані вставлені спеціальні об’єкти з контенту ІНМ (формула, рисунок, схема тощо). Для перевірки достатності сформованих множин антецедентів і консеквентів було проведено перевірку їх придатності для опису правил перетворення у тестовій вибірці, яка включала контент 9 навчальних курсів та створених для їх перевірки відповідно 9 тестів (містять загалом 438 тестових завдань) й не входили до першої вибірки. Виявилось, що сформовані множини антецедентів і консеквентів є достатніми. Додаткових комбінацій елементів $a \in A$ та $c \in C$ не було знайдено. При цьому тестові запитання, що належать не використаним у роботі типам (таким як тестові запитання на правильну послідовність та тестові запитання на відповідність), до розгляду не приймалися. Використані в дослідженні матеріали належать Тестовій вибірці №1, параметри якої наведено в п.4.2.1.

Таким чином, множиною з 2 антецедентів можна описати всі речення, які потенційно придатні для створення тестових завдань. Множина з 11 консеквентів охоплює всі алгоритми створення тестових завдань типів, що використовуються в навчальних середовищах: логічного, одиничного вибору, множинного вибору та завдань на введення тексту. Множини антецедентів і консеквентів формують множину з 17 правил продукції, які дозволяють створювати всі можливі тестові завдання за всіма потенційно придатними реченнями.

2.4.3 Схема методу

Схему методу автоматизованого формування тестових завдань подано на рис. 2.10.

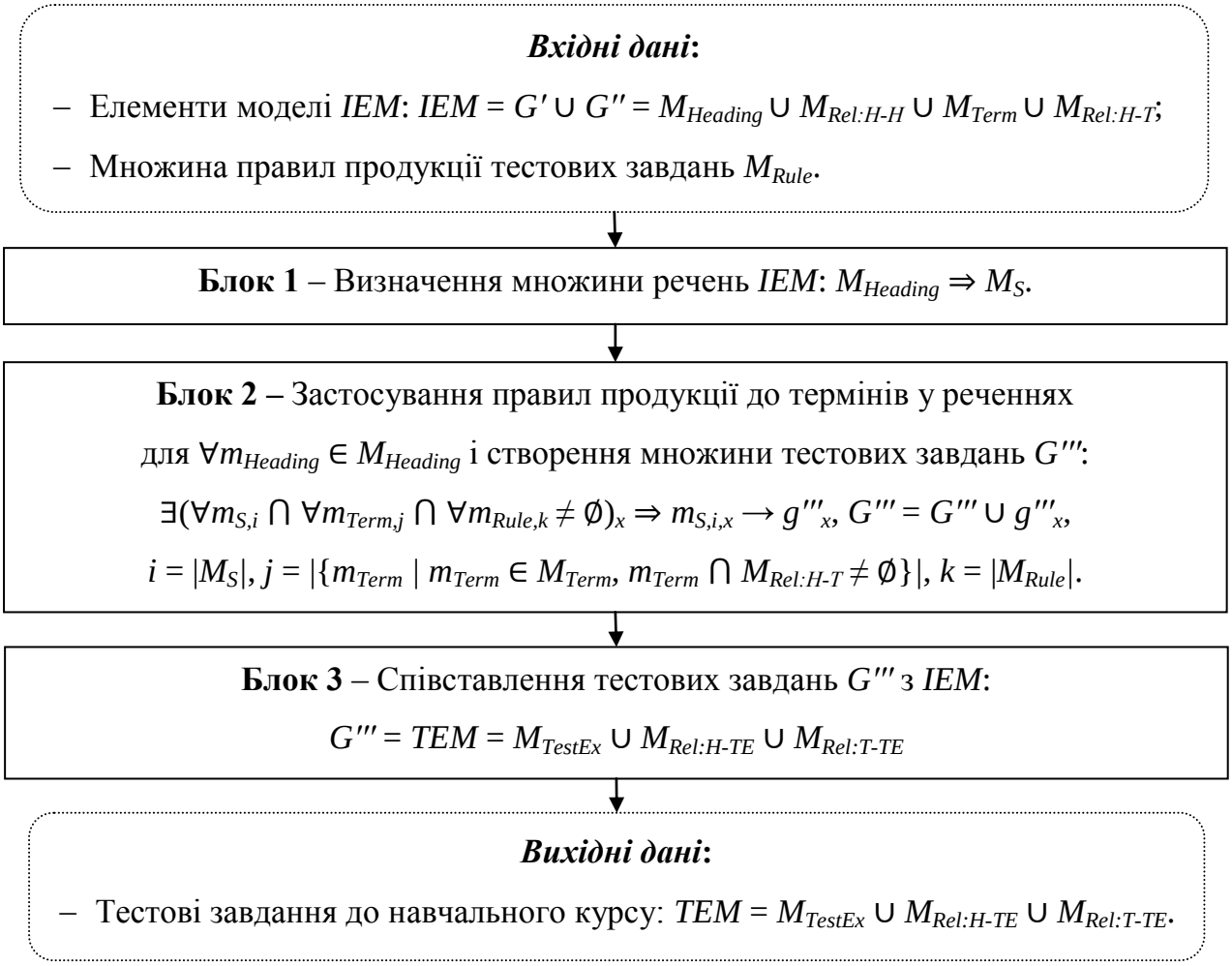


Рисунок 2.10 – Загальна схема методу автоматизованого формування тестових завдань

Вхідними даними методу автоматизованого формування тестових завдань є контент ІНМ чи його визначеного елементу структури $M_{Heading}$ та відповідна множина ключових термінів $M_{Term} \cup M_{Rel:H\text{-}T}$; вихідними даними є множина тестових завдань M_{TestEx}, а також множини зв’язків – між заголовками та тестовими завданнями $M_{Rel:H\text{-}TE}$ і між ключовими термінами та тестовими

завданнями $M_{Rel:T\text{-}TE}$. Для роботи методу необхідна множина правил продукції тестових завдань M_{Rule}, створених окремо і заздалегідь.

Спершу (*Блок 1*) шляхом парсингу контенту обраного елементу ІНМ (атрибут *HContent* кортежів множини $M_{Heading}$) формується множина фрагментів M_S, кожен з яких є реченням або в деяких випадках (наприклад, переліках) – множиною речень. Фрагменти локалізують потенційний контент для створення окремих тестових завдань.

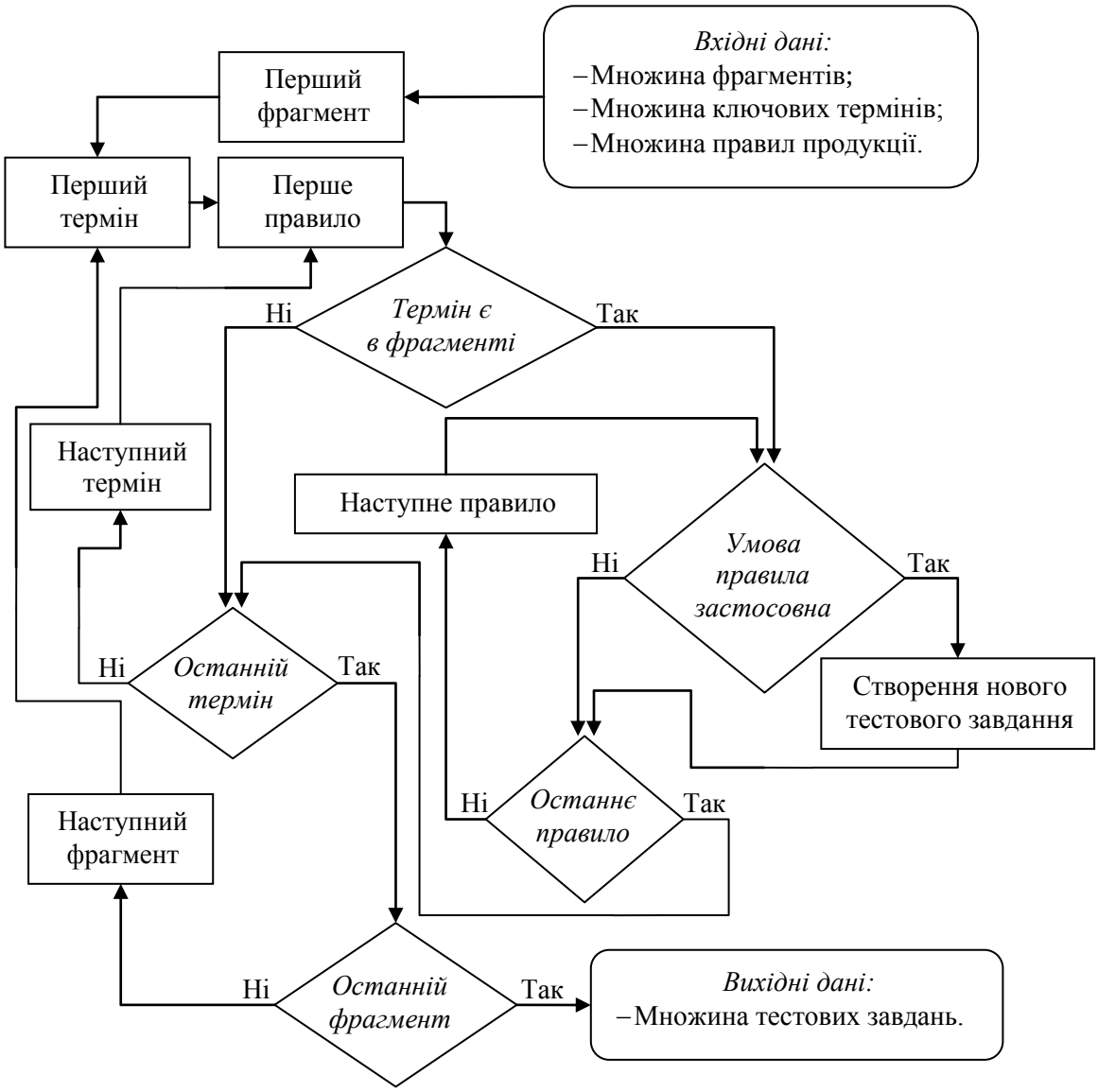


Рисунок 2.11 – Схема створення тестових завдань

Для створення множини тестових завдань G''' (*Блок 2*) кожен елемент $m_S \in M_S$ з кожної рубрики документу $m_{Heading} \in M_{Heading}$ перевіряється на

наявність кожного ключового терміну $m_{Term} \in M_{Term}$, зіставленого даній рубриці $m_{Term} \cap M_{Rel:H\text{-}T} \neq \emptyset$.. Якщо термін m_{Term} присутній в фрагменті m_S, то проводиться перебір правил продукції M_{Rule} на предмет відповідності умови (антецедента) правила. Кожен випадок відповідності $\exists(\forall m_{S,i} \cap \forall m_{Term,j} \cap \forall m_{Rule,k} \neq \emptyset)_x$ має наслідком автоматичне створення нового тестового завдання g'''_x (рис. 2.11). Дієва частина правила продукції (консеквент) визначає алгоритм перетворення контенту фрагменту m_S у тестове завдання g'''.

Шляхом перебору фрагментів, термінів та правил продукції виконується пошук відповідності антецедента обраного правила в фрагменті контенту ІНМ (рис. 2.12). Якщо відповідність встановлено – відповідно до консеквента даного правила формується нове тестове завдання, перевіряється на наявність мінімальної для даного типу тестових завдань кількості необхідних елементів, й додається до множини тестових завдань M_{TestEx}. Після чого (*Блок 3*) формуються відповідні зв'язки як елементи множини зв'язків між заголовками та тестовими завданнями $M_{Rel:H\text{-}TE}$ й множини зв'язків між ключовими термінами та тестовими завданнями $M_{Rel:T\text{-}TE}$.

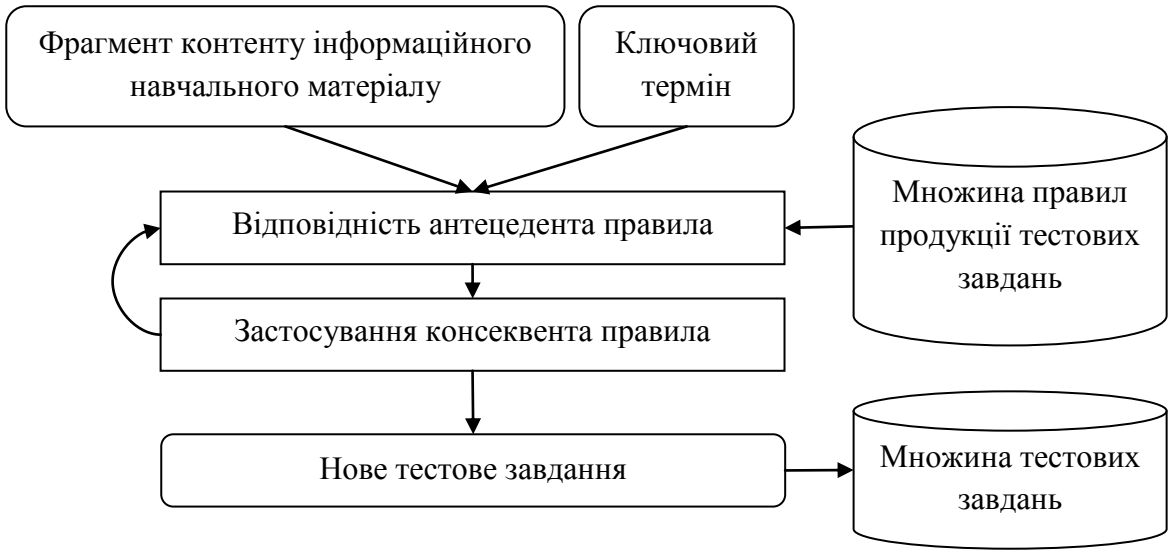


Рисунок 2.12 – Схема створення нових тестових завдань на основі правил

Вихідними даними методу автоматизованого формування тестових завдань є всі атрибути елементів множин ТНМ: $G''' = TEM = M_{TestEx} \cup M_{Rel:H\text{-}TE} \cup M_{Rel:T\text{-}TE}$.

Для покращення створених тестових завдань передбачена можливість ручного коригування контенту кожного одержаного тестового завдання та автоматизованого коригування загальної кількості тестових завдань.

2.4.4 Параметри тестових завдань

Для забезпечення процесу тестування, важливими параметрами тестового завдання, які не стосуються безпосередньо його контенту, є такі: тип питання, час відповіді, максимальна кількість балів, рівень складності. Хоча передбачається ручне або фіксоване визначення більшості цих параметрів, автоматизація їх обрахунку є окремими задачами. Наприклад, час відповіді на тестове завдання може бути обрахований залежно від типу питання та кількості символів у контенті тестового завдання.

Тестове завдання має множину відповідей з параметрами: контент відповіді та оцінка відповіді. Оцінка відповіді визначається так: якщо максимальний бал за правильну відповідь на тестове завдання рівний B, а кількість правильних відповідей N_{True}, то кожна правильна відповідь отримує бал $B_{True}=B/N_{True}$. Відповідно, якщо кількість хибних відповідей N_{False}, то кожна хибна відповідь отримує бал $B_{False}=-B/N_{False}$.

В межах моделі семантичної структури НК, можлива фіксація проміжних даних, які можна розглядати як семантичні параметри тестового завдання, а саме:

– термін, рівень засвоєння якого перевіряється тестовим завданням;

– номер фрагменту (речення), що було використано для створення тестового завдання (тобто знання якого й перевіряється);

– номер правила продукції, використаного для формування тестового завдання;

– номер типу тестового завдання;

– оцінка якості використання правила – необов’язковий параметр, що може бути обрахований додатково.

Наведені семантичні параметри тестового завдання можуть бути використані для зменшення нерівномірності за цими параметрами тестових завдань у результуючій множині.

Для цього необхідно сформувати множину семантичних параметрів тестових завдань $M_{TestExPar}$, кожен елемент якої є кортежем виду:

$$M_{TestExPar} = \{ID, Term, Sentence, Rule, Type, PointsEx\}, \quad (2.26)$$

де *ID* – унікальний ідентифікатор тестового завдання, *Term* – термін, *Sentence* – використане речення, *Rule* – використане правило продукції, *Type* – тип питання, *PointsEx* – оцінка якості використання правила.

Таким чином, за результатами використання методу автоматизованого формування тестових завдань формується таблиця тестових завдань із їх семантичними параметрами, подана у таблиці 2.4.

Таблиця 2.4 – Структура таблиці семантичних параметрів тестових завдань

№ тестового завдання	Термін	№ речення, що використано для формування тестового завдання	№ правила продукції	№ типу тестового завдання	Оцінка якості використання правила
…	…	…	…	…	…

У результаті використання методу може бути одержано надмірну кількість тестових завдань. Пропонується автоматизовано зменшувати їх кількість при одночасному зменшенні нерівномірності тестових завдань за параметрами *Term, Sentence, Rule, Type* й при видаленні тестових завдань із меншими значеннями параметра *PointsEx*.

Висновки до розділу

1. Вдосконалено інформаційну модель семантичної структури навчального курсу з метою забезпечення можливості з достатньою інформативністю виконувати формальне подання навчальних матеріалів курсу дисципліни. Автоматичне визначення параметрів моделі дозволяє автоматизовано створювати розподілені за структурою ІНМ тестові завдання.

2. Вдосконалено метод формування структури навчальних матеріалів та пошуку в них ключових термінів для отримання ключових слів та словосполучень у ієрархічному розрізі рубрик навчальних матеріалів курсу дисципліни. Це дозволяє врахувати можливість взаємного поглинання термінів і ідентифікувати як термін не тільки слова, а й словосполучення. При пошуку ключових термінів для кожного з них визначається умовна оцінка його важливості у відповідній рубриці, а множини ключових термінів для кожної з рубрик автоматизовано обмежуються за показником щільності ключових термінів у контенті відповідної рубрики.

3. Запропоновано метод автоматизованого формування тестових завдань до навчальних матеріалів за продукційною моделлю представлення знань для подання правил формування тестових завдань, який не вимагає додаткової формалізації навчальних матеріалів. В результаті використання методу автоматизованого формування тестових завдань отримують множину тестових завдань, що різні за параметрами (тип запитання, кількість правильних відповідей, правило за яким сформоване тестове завдання, терміни які використовуються в завданні тощо) й можуть бути використані для перевірки рівня засвоєння знань за допомогою існуючих навчальних середовищ та систем тестування. Множина містить тестові завдання, що семантично, структурно та параметрично охоплюють відповідний вхідний ІНМ.

РОЗДІЛ 3.
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АВТОМАТИЗОВАНОГО СТВОРЕННЯ ТЕСТІВ

3.1 Схема та кроки інформаційної технології

ІТ автоматизованого створення тестів до навчальних матеріалів використовує запропоновану у п. 2.2 інформаційну модель семантичної структури НК та методи:

– формування структури навчальних матеріалів та пошуку у них ключових термінів (див. п. 2.3);

– автоматизованого формування тестових завдань (див. п. 2.4).

Схематично застосування означених методів у ІТ подано на рис. 3.1.

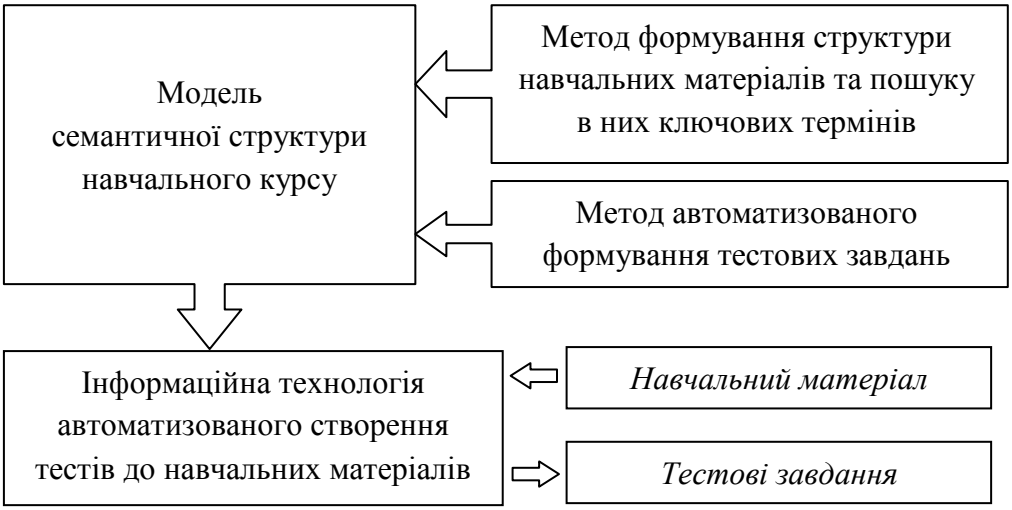


Рисунок 3.1 – Схема застосування методів ІТ

Як видно з наведеної схеми (див. рис. 3.1) досягнення мети ІТ відбувається шляхом послідовного застосування розглянутих методів (рис. 3.2).

Вхідними даними ІТ є файл електронного документу з контентом ІНМ, наприклад, формату .docx. Файл має містити слабо структурований текстовий контент, для рубрикації бажане використання заголовків, які визначають

структуру текстового контенту. Ніякої спеціальної попередньої обробки вхідних даних не вимагається, втім підвищення рівня виконання вимог роботи з електронними документами (використання стилів) та вимог до роботи з навчальними матеріалами (див. п 3.3.1) може суттєво підвищити якість вихідних даних ІТ.

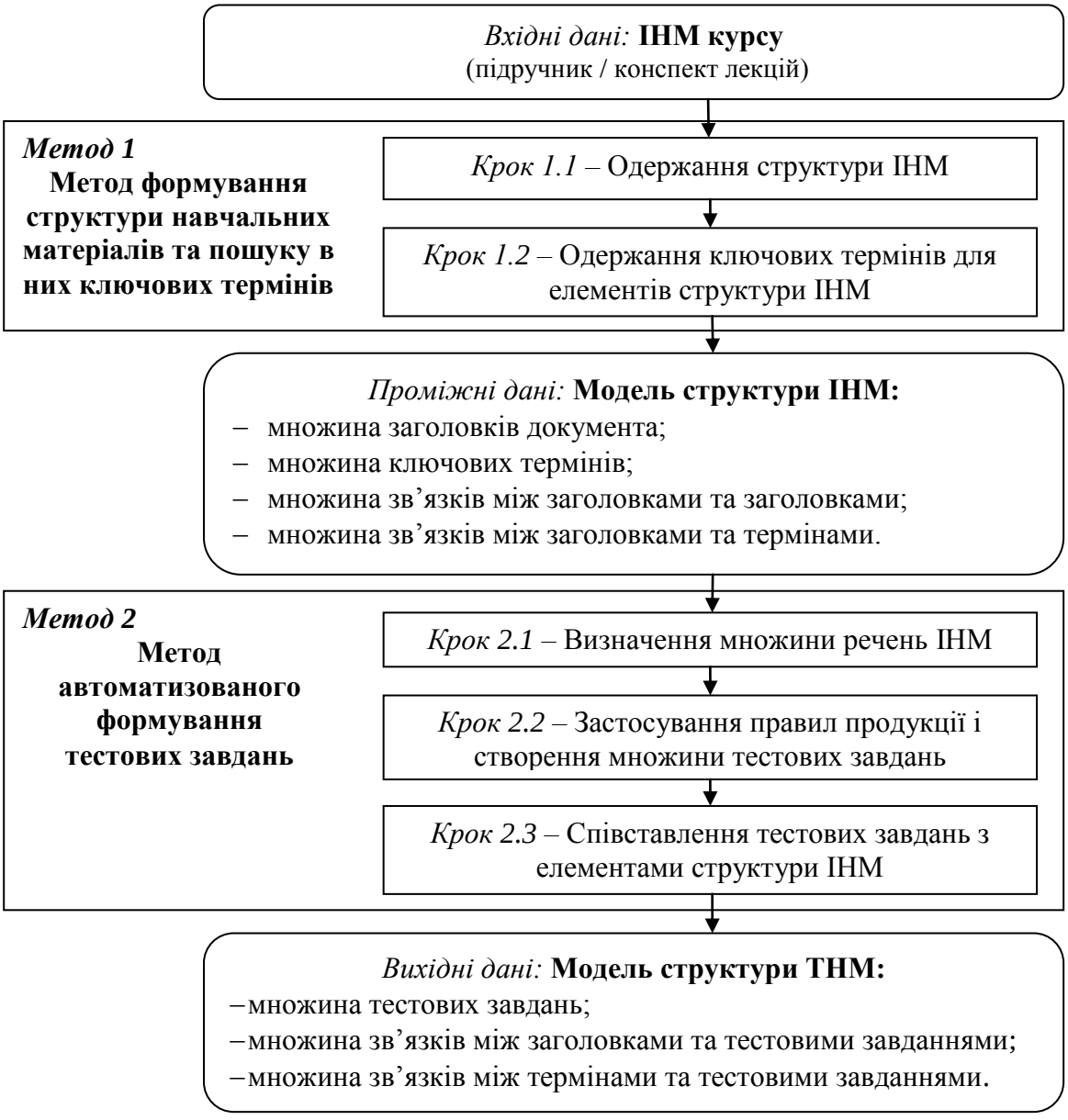


Рисунок 3.2 – Послідовність кроків ІТ

Метод 1 (формування структури навчальних матеріалів та пошуку в них ключових термінів) є першим етапом роботи ІТ. Результатом роботи методу є формування частини моделі семантичної структури НК, яка

стосується ІНМ, а саме верхнього рівня структури ІНМ у вигляді рубрикації (структури заголовків) та нижнього рівня структури ІНМ у вигляді окремих впорядкованих за важливістю множин ключових термінів для кожного з елементів верхнього рівня структури ІНМ.

На кроці 1.1 при аналізі вмісту документу обробляється контент документу для визначення стилів його складових. Відповідно до них, формується структура заголовків, одним з атрибутів кожного з яких є приналежний йому обсяг контенту документу. В подальшому обробка контенту кожного з одержаних елементів відбувається окремо і незалежно. При цьому контент підлеглого елементу одночасно входить до складу контенту елемента-батька в ієрархічній структурі заголовків.

Після цього на кроці 1.2 для кожного елемента в структурі ІНМ проводиться аналіз контенту для пошуку ключових термінів із використанням методу дисперсійного оцінювання. При цьому терміни-кандидати обираються в межах фраз, обмежених єдиним стилістичним оформленням та знаками пунктуації. Після визначення дисперсійної оцінки термінів, яка розглядається як показник їх важливості, відбувається поглинання термінів, що є частинами інших термінів з більшою кількістю слів. Одержана множина термінів сортується за значенням важливості терміна й обмежується за встановленим значенням показника щільності ключових термінів у відповідному контенті.

Результат роботи методу 1 не прив’язаний до певної мови й на оцінку важливості терміна не впливають стильові ознаки в контенті. Втім, передбачена необов’язкова можливість коригування отриманих множин ключових термінів, що може підвищити якість вихідних даних:

– шляхом перевизначення показника важливості терміна за стилями документу, наприклад при виділенні в контенті терміна жирним, курсивом чи в складі заголовку (див. п. 3.3.3);

– шляхом фільтрування за синтаксичними шаблонами, наприклад якщо термін є одним словом то згідно проведених досліджень він може бути тільки іменником (див. п. 3.3.2);

– видаленням деяких термінів вручну у випадках, коли попри їх важливість контенті користувач не вважає їх вивчення метою даного фрагменту ІНМ.

За результатом використання методу 1 забезпечується визначення параметрів відповідних множин семантичної структури НК (множини заголовків, множини ключових термінів, множини зв'язків між заголовками і заголовками, множини зв'язків між заголовками і термінами), які є вхідними даними для методу 2 в межах ІТ.

Метод 2 (автоматизованого формування тестових завдань) є другим етапом роботи ІТ. Результатом його роботи є формування частини моделі семантичної структури НК, яка стосується тестових завдань, а саме множини тестових завдань та зв'язків кожного тестового завдання як із елементом структури ІНМ, на основі якого й для перевірки якого створене тестове завдання, так і з ключовим терміном, рівень засвоєння якого тестове завдання перевіряє.

Обробка контенту кожного з елементів структури ІНМ для створення тестових завдань відбувається окремо і незалежно. На кроці 2.1 цей контент розбивається на окремі речення. Інструментом автоматизованого створення тестових завдань слугує множина правил продукції тестових завдань, кожне з яких містить умову, яка визначає придатність обраного речення для створення тестового завдання до обраного терміна, та наслідок у вигляді алгоритму перетворення контенту речення у відповідне тестове завдання.

На кроці 2.2 кожне речення контенту обраного елементу структури ІНМ перевіряється на наявність кожного ключового терміну, що визначений як ключовий у цьому елементі структури ІНМ. Якщо термін використовується в реченні, то проводиться перебір правил продукції для визначення відповідності умов правил. Кожен випадок відповідності має наслідком автоматичне створення нового тестового завдання.

У випадку, коли метою застосування ІТ є не максимізація кількості тестових завдань у вихідній множині, а й їх рівномірний розподіл за

визначеними параметрами (тип запитання, термін який перевіряється, елемент структури ІНМ тощо), передбачена необов'язкова можливість (див. п. 2.4.4) коригування загальної кількості тестових завдань шляхом видалення частини з них за категоріями, де їх кількість більша від середньої.

На кроці 2.3 за результатом використання методу 2 забезпечується наповнення відповідних множин семантичної структури НК: множини тестових завдань, множини зв'язків між ключовими термінами і тестовими завданнями, множини зв'язків між заголовками і тестовими завданнями.

Вихідними даними ІТ є множина тестових завдань. Ця множина може бути подана в потрібному користувачеві вигляді, наприклад в форматі .xml чи .gift для подальшого використання в середовищі Moodle.

Також *вихідними даними* ІТ є семантична структура НК, подана у вигляді ряду множин, яка може бути використана для допомоги при практичному використанні сформованої множини тестових завдань, зокрема для забезпечення адаптивного тестування. Іншими напрямками використання одержаних вихідних даних є автоматизоване вирішення ряду задач – визначення відповідності ІНМ вимогам, допомога при розробці ІНМ, оцінка відповідності наборів тестових завдань до ІНМ, допомога при створенні тестів й інших.

Для прикладної реалізації ІТ потрібне її функціональне подання та визначення вимог до вхідних даних, а також створення множини правил продукції тестових завдань.

3.2 Функціональне подання інформаційної технології як інформаційної системи

Функціональне подання запропонованих у ІТ методів перетворення інформації для автоматизованого формування тестових завдань зображено на рис. 3.3. у вигляді IDEF0-діаграми [179].

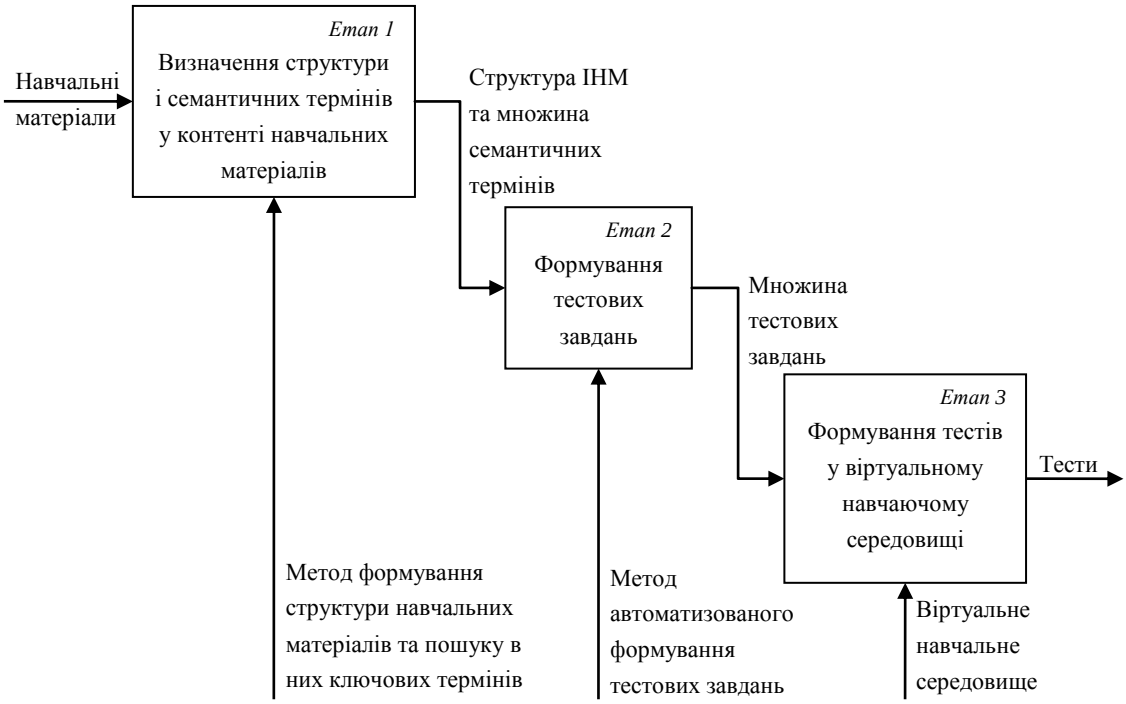


Рисунок 3.3 – Діаграма етапів вирішення задачі автоматизації створення тестових завдань

Кожному етапу з наведених відповідає деяка послідовність перетворення даних і компоненти системи, які прийматимуть у цьому участь. На рис. 3.4 зображено відповідну схему розподілу функцій за компонентами системи.

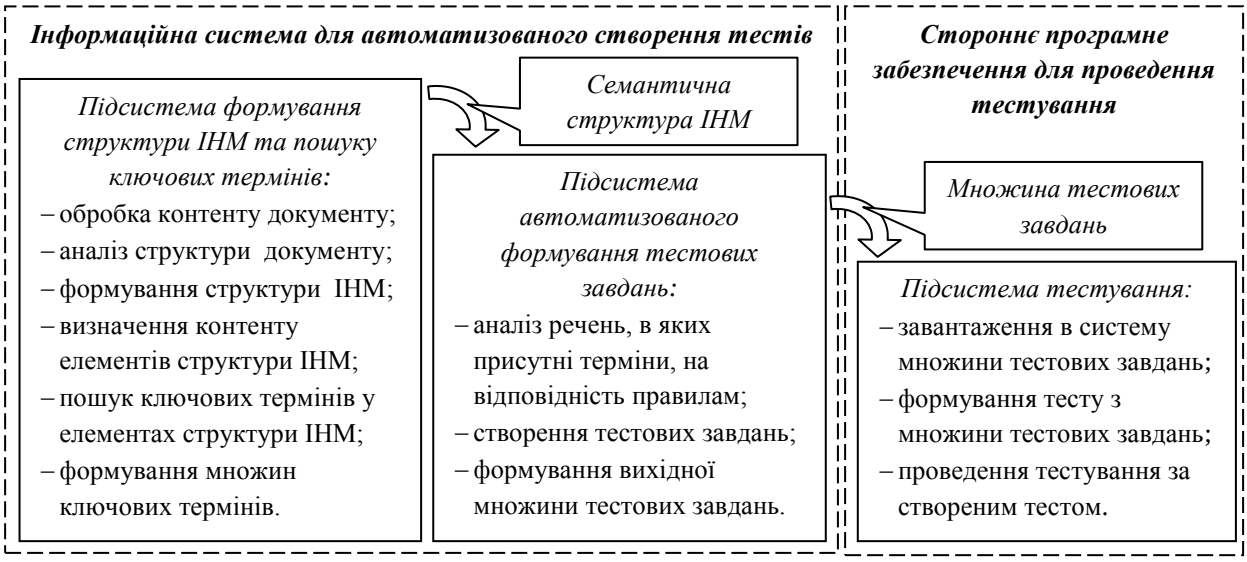


Рисунок 3.4 – Схема розподілу функцій за компонентами системи

Розробка програмних системи, що за допомогою відомих алгоритмів реалізують процес тестування (завантаження в систему множини тестових завдань, формування тесту з множини тестових завдань, проведення тестування за створеним тестом), виходить за межі даної роботи. Для перевірки працездатності створених множини тестових завдань було використано середовище Moodle. При цьому вихідні дані інформаційної системи для автоматизованого створення тестів є вичерпними для подальшого їх використання сторонніми системами тестування.

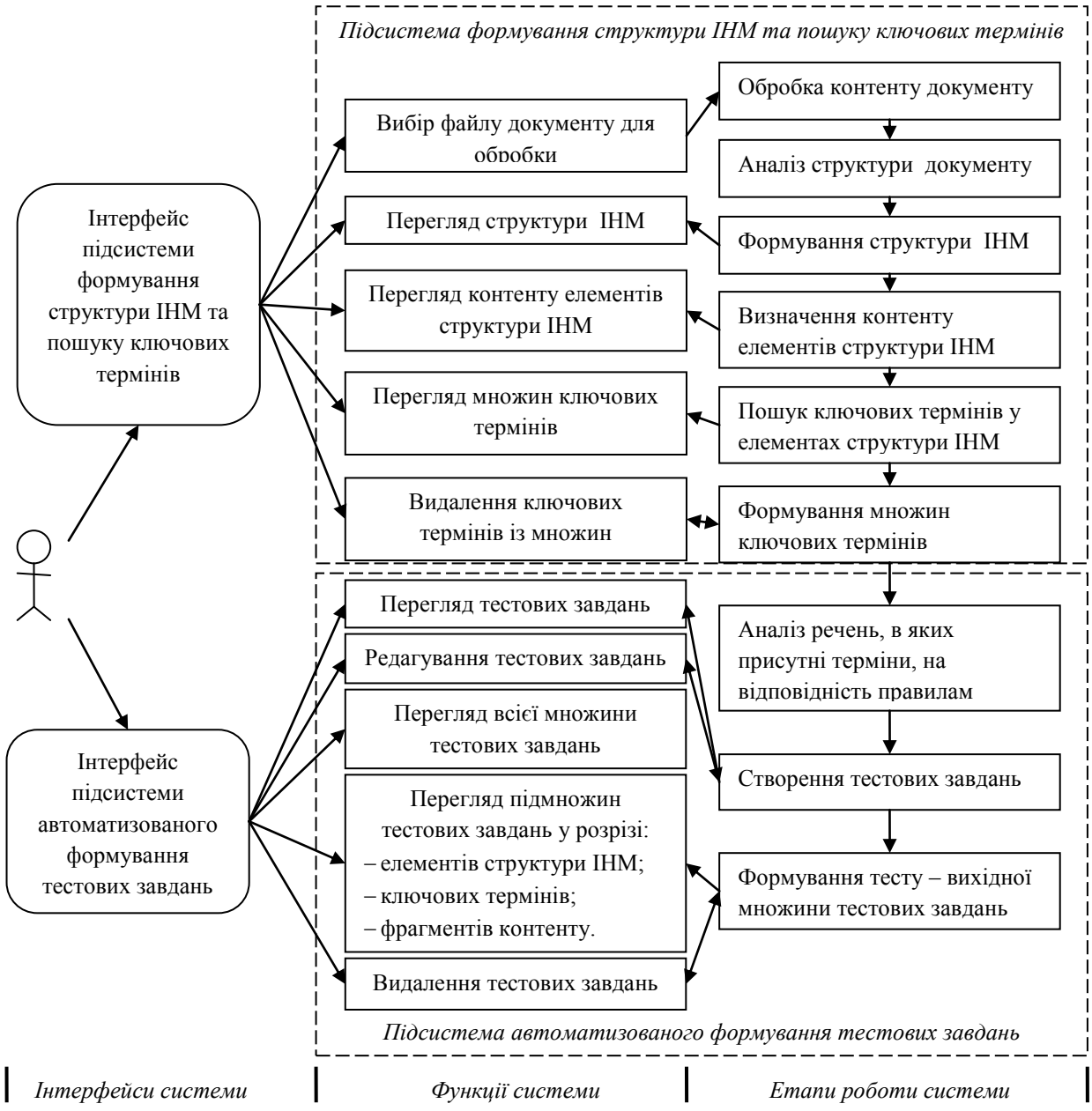


Рисунок 3.5 – Функції користувача

Виходячи зі схеми розподілу функцій за компонентами системи, визначено дві підсистеми:

1) підсистема формування структури ІНМ та пошуку ключових термінів, що виконує функції обробки контенту вхідного документу, аналізу структури документу, формування структури ІНМ визначення контенту елементів структури ІНМ, пошуку ключових термінів у елементах структури ІНМ та формування множин ключових термінів;

2) підсистема автоматизованого формування тестових завдань, що виконує функції аналізу речень в яких присутні терміни на відповідність правилам автоматизованого створення тестових завдань та формування вихідної множини тестових завдань.

Цей розподіл функцій за підсистемами визначає обсяг можливих дій користувача, які наведено на рис. 3.5.

Функції користувача можна розділити на три категорії:

1. Обов’язкові активні функції користувача, без яких робота системи є неможливою, до яких належить вибір файлу документу для обробки.

2. Необов’язкові активні функції користувача, які впливають на кінцевий результат роботи системи, до яких належать:

– видалення обраних ключових термінів із множин знайдених ключових термінів в контенті елементів структури ІНМ;

– редагування обраного тестового завдання;

– видалення обраного тестового завдання з множини створених тестових завдань.

3. Необов’язкові спостережні функції користувача, які не впливають на кінцевий результат роботи системи і призначені для огляду проміжних даних роботи системи, й до яких належать:

– перегляд створеної системою структури ІНМ;

– перегляд контенту, віднесеного до кожного з елементів структури ІНМ;

– перегляд множин ключових термінів, знайдених у кожному з елементів структури ІНМ;

– перегляд контенту кожного з створених системою тестових завдань;

– перегляд множини всіх створених тестових завдань;

– перегляд підмножин тестових завдань у розрізі обраного елементу структури ІНМ, обраного ключового терміну чи обраного фрагменту контенту ІНМ (речень чи їх груп).

Таким чином, обов'язковою функцією користувача системи автоматизованого створення тестів є вибір файлу документу ІНМ для обробки. Подальші кроки, що приводять до формування множини тестових завдань, система здатна виконувати самостійно. Решта функцій користувача є необов'язковими (для покращення результату роботи системи) або спостережними (для розуміння процесу одержання проміжних і вихідних даних).

3.3 Особливості застосування інформаційної технології

3.3.1 Вимоги до вхідних даних інформаційної технології

Єдиною *обов'язковою вимогою* до вхідних даних ІТ є наявність текстового контенту ІНМ у вигляді алфавітних символів та прогалин в файлі.

Інші властивості вхідних даних є необов'язковими, хоча впливають на роботу ІТ чи окремих програмних систем, розроблених за нею. Вони розглянуті далі.

Рубрикація контенту ІНМ є бажаною, оскільки дозволяє формувати впорядковані множини ключових термінів і множини тестових завдань в розрізі рубрик ІНМ, що в свою чергу надає можливість створити більш репрезентативний ТНМ та перевіряти при тестуванні засвоєння знань за елементами структури ІНМ. У випадку, якщо рубрикація відсутня, буде сформовано дерево структури документу з 1 елементом множини заголовків,

назва якого відповідна назві файлу. Згідно документарної моделі даних, спеціальні засоби рубрикації (стилі, заголовки, рубрики тощо) передбачені в усіх слабо структурованих текстових документах (формати .doc, .docx, .html, .pdf тощо) (розглянуто в п. 2.2.1), а її наявність є похідною від вимог до структури ІНМ навчальних дисциплін (розглянуто в п. 1.1.3).

Неоднозначне іменування термінів в ІНМ свідчить про недотримання норм формування та ведення наукової літератури (розглянуто в п. 1.1.5). Хоча є допустимим використання абревіатур, часткових скорочень та повних назв (наприклад, «СКБД», «Система керування БД», «Система керування базами даних»), зокрема кількома мовами, для використання в рамках окремого матеріалу обирається лише один варіант. Це пов’язано з тим, що як для машинної, так і для розумової ідентифікації термінів бажаною є уніфікація ідентифікатора. Навіть у випадку введення нового скорочення, позиція введення скорочення є позицією першого згадування повної назви терміну в тексті, причому в цій же позиції у тексті присутня також і повна назва терміну (зазвичай, у дужках). Причому після введення скорочення в подальшому повна назва терміну не використовується. Відповідно, усунення неоднозначної іменованості термінів не тільки бажане з точки зору норм формування та ведення наукової літератури, а й дозволить уникнути випадків, коли при автоматичному аналізі контенту навчальних матеріалів подання одного поняття буде розглядатись як кілька окремих термінів.

Обробка спеціальних об’єктів (формула, рисунок, схема тощо) в контенті ІНМ можлива шляхом розгляду їх позиції в тексті як окремих термінів. Для визначення позицій ідентичних спеціальних об’єктів використовується їх сигнатура (наприклад, геометричні та дискові розміри формул). Як спеціальні об’єкти, крім вставлених об’єктів, також можна розглядати семантично значущі класи (наприклад: прізвища, роки, століття), для побудови тестів для їх перевірки можливе їх включення до множини ключових термінів. У випадку ж, коли навпаки необхідно виключити з процесу обробки контент певних спеціальних елементів (таблиці, програмні

коди тощо), існує можливість зробити це шляхом використання стилів оформлення, відмінних від стилів рубрикації (заголовків) та основного тексту, що є природнім при роботі з слабо структурованими текстовими документами.

Подання текстового контенту бажано у вигляді алфавітних символів відповідної мови. При програмному парсингу контенту ІНМ для пошуку ключових термінів виконується обробка розділових знаків. При цьому власне розділові знаки (наприклад: !№;%:?*()_+=,.@#$^&*<>"|\/…{}–»«•§) видаляються, але якщо вони є частиною слів (для української мови це апострофи «’» та дефіси «-») – то розглядаються як частина слів. Тому рекомендовано, наприклад, коректно використовувати дефіси «-» та тире «–», оскільки дефіс є частиною складних слів, у той час як тире використовується для пунктуації та семантичного конструювання речення. У випадку нехтування цією вимогою, можливі помилки при автоматизованому визначенні меж слів, словосполучень та речень у контенті ІНМ. Роздільниками слів визначено розділові знаки й прогалини. Тестування інформаційної технології проводилось для ІНМ виконаних українською, англійською мовами та з використанням обох мов одночасно. Однак, оскільки методи інформаційної технології проводить обробку тексту на рівні окремих слів як неподільних елементів, можливе її використання для роботи з мовами, де слово формується в інший спосіб (наприклад, ієрогліфами). В будь-якому разі, до обробки приймається контент у вигляді алфавітних символів відповідної мови, в той час як обробку текстового контенту спеціальних елементів (наприклад, тексту на зображеннях) не передбачено.

Зазначені вимоги до вхідних даних, покликані покращити результати роботи складових ІТ, відповідають існуючим нормам оформлення електронних документів та ведення наукової літератури, що ставлять на меті покращити сприйняття контенту людиною.

Для підвищення ефективності використання ІТ можлива попередня технічна обробка тексту, що наближає контент ІНМ до наведених вище рекомендацій і полягає переважно в усуненні неоднозначного іменування

термінів та обробці розділових знаків. Таким чином, бажаним є випадок, коли контент ІНМ структурований засобами рубрикації, містить переважно текстове подання у вигляді алфавітних символів та прогалин й відповідає існуючим нормам оформлення електронних документів і ведення наукової літератури.

3.3.2 Параметри ключових термінів, синтаксична фільтрація

ІТ використовує ряд параметрів, одержаних внаслідок кількісного аналізу зразків існуючих ключових термінів. Це, наприклад, допустима кількість слів у терміні, синтаксичні параметри ключових термінів тощо. Далі розглянуто особливості цих параметрів для ІТ.

За результатами проведеного аналізу вибірки з 1308 елементів навчальних матеріалів із визначеними укладачем (автором) репрезентативними множинами ключових термінів, встановлено, що всі елементи наведених множин M_T відповідають наступним закономірностям [5]:

– кількість слів у терміні n = 1..6;

– якщо термін є словом (n = 1), то воно входить до множини іменників M_I;

– якщо термін є словосполученням (n > 1), то до його складу входять елементи множини M_M. До складу множини M_M входять множини семантично значущих елементів (іменників M_N, числівників M_{Num} і прикметників M_A) та семантично зв'язуючих елементів (сполучників M_S і прийменників M_P);

– якщо n > 1, то до складу словосполучення входить принаймні один елемент із множини іменників M_N;

– якщо n > 1, то першим (k = 1) та останнім (k = n) словом є елементи множини семантично значущих елементів $M_N \cup M_{Num} \cup M_A$;

– якщо n > 1, то між елементами словосполучення відсутні розділові знаки (окрім дефісу та апострофу, які можуть бути частинами слова).

При цьому, якщо термін є словосполученням, то для слів з його складу визначено наступні обмеження:

– До множини сполучників M_S належать тільки сполучники, які належать до виду простих, що морфологічною будовою не пов'язані з іншими частинами мови. Їх множина є визначеною: M_S = {*і, й, а, чи, та*}.

– До множини прийменників M_P належать тільки прийменники, які належать до виду первинних, тобто які мають праслов'янське походження. Їх множина є визначеною: M_P = {*без, у, в, від, для, по, через, при, над, під, до, з, із, як, за*}.

– До множини числівників M_{Num} належать тільки порядкові числівники (наприклад: *«дев'яності», «перша», «двадцять»* $\in M_{Num}$). Також до цієї множини можна віднести групи цифр розмірності не більше 4 (наприклад: *«2019», «1053», «256»* $\in M_{Num}$).

При дослідженні було сформовано таблицю з наступними атрибутами:

1. № документу;
2. Назва навчального матеріалу; посилання або вказівка на джерело;
3. Перелік ключових термінів матеріалу, визначений його автором;
4. Кількість ключових термінів;
5. Кількість ключових термінів, що містять n слів для випадків: $n = 1$, $n = 2$, $n = 3$, $n = 4$, $n = 5$, $n = 6$, $n > 6$;
6. Кількість ключових термінів, що відповідають портрету терміна;
7. Ключові терміни, що не відповідають портрету терміна.

Шляхом аналізу даних наведеної таблиці (фрагмент таблиці для аналізу параметрів ключових термінів ІНМ наведено в Додатку Г), були поступово розширені й відкориговані наведені вище обмеження та параметри (портрет) ключових термінів доти, поки в останньому стовпці не залишились виключно семантично некоректні та помилкові зразки термінів.

За результатами аналізу, було одержано висновки (рис. 3.6), що характеризують залежність кількості ключових термінів у всій вибірці по відношенню до кількості слів у них.

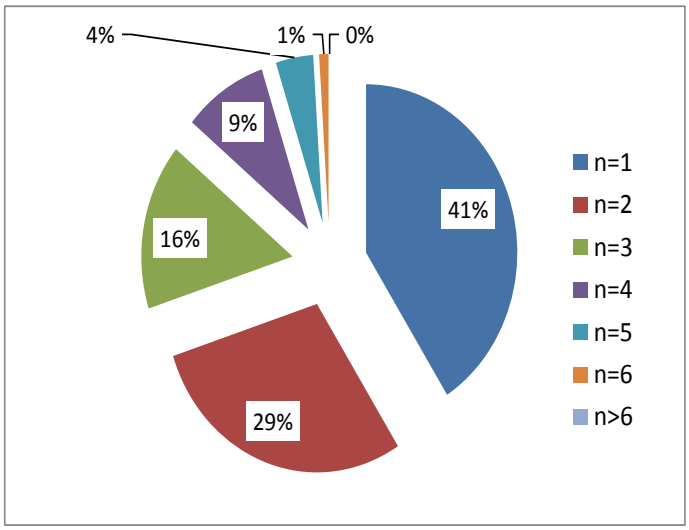


Рисунок 3.6 – Графік залежності кількості ключових термінів у вибірці від кількості слів у їх складі

Таким чином, при виконанні ІТ, після автоматизованого визначення множини ключових термінів ІНМ з неї видаляються ті терміни, що не відповідають наступній умові:

$$M_T = \{\langle x_1, x_2, x_3, x_4, x_5, x_6\rangle | x_1 \in M_N \cup M_{Num} \cup M_A, \\ x_2 \in M_M, x_3 \in M_M, x_4 \in M_M, x_5 \in M_M, \\ x_6 \in M_N \cup M_{Num} \cup M_A, \langle x_1, x_2, x_3, x_4, x_5, x_6\rangle \cap M_N \neq \varnothing\}. \quad (3.1)$$

Подальші кроки виконуються без змін. Така фільтрація одержаних елементів множини ключових термінів є необов’язково, але у випадку наявності відповідної до мови та галузі ІНМ синтаксичної бази даних дозволяє підвищити якість формування множин ключових термінів і якість вирішення ряду похідних задач, таких як узгодження слів у контенті тестових завдань.

3.3.3 Врахування семантичної ваги ключових термінів

Після застосування методу формування структури навчальних матеріалів та пошуку в них ключових термінів, для кожного заданого фрагменту ІНМ визначається множина ключових термінів із співставленим кожному з термінів числовим значенням, яке відображає його семантичну вагу. Зважаючи на особливості використання цих термінів в контенті заданого фрагменту ІНМ, можна перевизначити цей показник з метою більш коректного формування результуючої вибірки ключових термінів.

Для цього значення обрахованого показника семантичної ваги σ множиться на деякий ваговий коефіцієнт:

$$\sigma = \sigma_0 k_i, \tag{3.2}$$

де k_i – ваговий коефіцієнт для i-ї ознаки.

Такими ознаками можуть бути випадки використання терміну в початковому чи завершальному (вступному та висновковому) абзацах ІНМ, використання терміну у заголовках, виділення спеціальним стильовим відображенням (жирний, курсив) тощо.

При аналізі розглянутої вище вибірки ІНМ було встановлено наступне:

– автори часто використовують ключові терміни фрагменту ІНМ для формулювання заголовків цих фрагментів (ця ознака має двонаправлений характер), проте кількісно ці випадки залежать від потужності відповідної множини ключових термінів;

– автори часто використовують спеціальні візуальні виділення (жирний, курсив) для позначення в тексті ключових термінів, особливо якщо подається їх визначення, проте кількісно ці випадки залежать від ретельності візуального оформлення ІНМ;

– якщо автор використовує спеціальні візуальні виділення ключового терміна, то в більшості випадків виділяється власне термін, і не використовуються різні спеціальні візуальні ефекти для виділення складових

одного терміна, крім випадків некоректного використання форматування слабо структурованих текстових документів.

Отже, для збільшення значення обрахованого показника семантичної ваги σ терміну, запропоновано використання наступних вагових коефіцієнтів для таких ознак:

– k_{Head} – для врахування кожного випадку використання терміна у заголовку елемента навчального матеріалу;

– k_{Bold} – для врахування кожного випадку використання виділення жирним нарисом терміна у контенті елемента навчального матеріалу;

– k_{Italic} – для врахування кожного випадку використання виділення курсивним нарисом терміна у контенті елемента навчального матеріалу.

Наприклад, якщо прийнято k_{Head} = 1,03, k_{Bold} = 1,025 та k_{Italic} = 1,015, й для деякого терміна значення згідно методу пошуку ключових термінів обрахованого показника семантичної ваги σ_0 = 1,72345, при одноразовому використанні цього терміна у заголовку й двома випадками виділення його – курсивом та жирним курсивом, то результуюче значення показника семантичної ваги σ буде наступним:

$$\sigma = 1{,}72345*1{,}03*1{,}015*1{,}025*1{,}015 = 1{,}87453.$$

Таким чином, з метою більш коректного формування результуючої множини ключових термінів запропоновано врахування особливостей використання цих термінів у контенті ІНМ шляхом математичного врахування випадків появи наведених ознак.

3.3.4 Алгоритм адаптивного вибору тестових завдань

Придатність визначених за ІТ даних інформаційної моделі для використання для адаптивного тестування потребує визначення алгоритму, за яким таке тестування можливе. У п. 1.2.5 наведено існуючі алгоритми адаптивного тестування, спільною рисою яких є залежність операції вибору наступного тестового завдання від відповіді на поточне тестове завдання.

Серед алгоритмів адаптивного тестування слід виділити такі, що пропонують поступове збільшення складності тестових завдань до виконання умови – неправильної відповіді або досягнення максимальної складності. Такі дії пропонується повторювати для кожного розділу ІНМ (структуру розділів ІНМ розглянуто в п. 1.1.5) в межах тесту [33].

В п. 1.3.1 визначено можливість подання семантичного вмісту цифрового тексту відповідною множиною ключових термінів. Причому, згідно [165], числове значення семантичної значущості різних ключових термінів є різною, що дозволяє формувати впорядковані за зменшенням семантичної значущості множини ключових термінів.

Наведена в п. 3.1 ІТ дозволяє одержувати дані, необхідні проведення для адаптивного тестування наступного зразка. Тестування починається з перевірки знання найважливішого терміна та у випадку коректних відповідей підвищується складність шляхом перевірки знання менш важливих термінів (рис. 3.7).

Наведений алгоритм гнучкого вибору тестових запитань дозволяє проводити тестування за максимальною кількістю елементів структури ІНМ, розпочинаючи тестування від найбільш загального терміну. Складність запитань збільшується у випадку коректної відповіді. Якщо ж користувач не спроможний дати коректну відповідь навіть на найлегше запитання, то немає сенсу продовжувати опитування блоку та перевіряти знання на складніших запитаннях. Тому повна перевірка знань елементів структури ІНМ закінчується достроково у разі некоректних відповідей або ж закінчується після коректних відповідей на усі запитання, включаючи найскладніші.

У разі завершення тестування по одному елементу структури ІНМ, відбувається перехід до тестування наступного, якщо такий є. Тест визнається пройденим у випадку, коли оброблені усі елементи структури ІНМ.

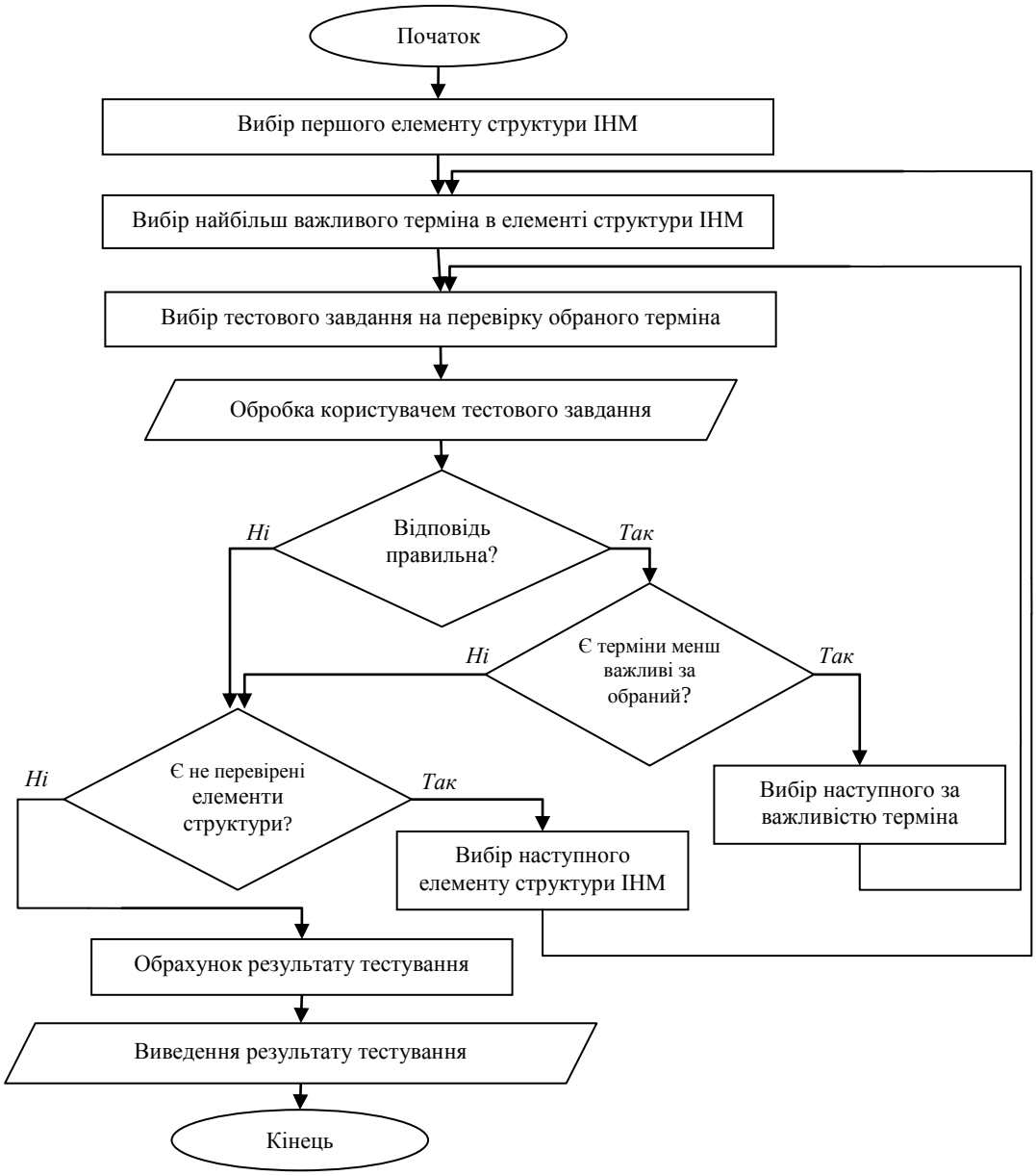


Рисунок 3.7 – Алгоритмічна схема адаптивного вибору тестових запитань

Оцінка за тест визначається як оцінка за кожен елемент структури ІНМ, помножена на ваговий коефіцієнт цього блоку, що визначається розробником навчального курсу.

3.4 Особливості елементів моделі семантичної структури

Особливості визначення **елементів множини заголовків.** При форматуванні слабо структурованих текстових документів, у вигляді яких подаються ІНМ, для визначення назв рівнів елементів структури

використовуються спеціальні елементи форматування (стилі) – заголовки. Як заголовок у даній роботі розглядається задана автором назва певного змістовного блоку в рамках контенту ІНМ. Таким чином, до множини заголовків M_{Heding} входять всі заголовки документу. Відповідно до (2.13) кожен елемент множини заголовків $M_{Heading}$ має атрибути *ID*, *HName*, *HContent* та *Grade*, які запропоновано автоматизовано визначати поданим у табл. 3.1 чином.

Таблиця 3.1 – Підходи до одержання значень атрибутів елементів множини заголовків M_{Heding}

Назва атрибуту	**Фізичний зміст атрибуту**	**Підхід до одержання значення атрибуту**
ID	унікальний ідентифікатор елементу	Автоматично привласнені унікальні цілі числа ($M_{ID} \in Z$), що використовуються для однозначної ідентифікації елементів у множині M_{Heding} за їх значеннями
HName	семантичний ідентифікатор відповідного фрагменту контенту ІНМ	Символьна назва заголовку (рубрики), створена автором ІНМ та включена до контенту цифрового документу ІНМ
HContent	контент чи вказівка на місцезнаходження контенту відповідного елементу ІНМ	Фрагмент контенту ІНМ починаючи від заголовку цього елементу й закінчуючи початком наступного заголовку, рівень якого *Grade* не нижчий за рівень даного елемента
Grade	рівень заголовку в ієрархічній структурі	Автоматично визначений номер рівня заголовку або автоматизовано привласнений виходячи з загальної кількості визначених автором рівнів у ІНМ

Особливості визначення **елементів множини термінів.** Відповідно до (2.14), кожен елемент множини термінів M_{Term} має атрибути *ID*, *TermName* та *TermNum*. Вони автоматизовано визначаються поданим у табл. 3.2 чином.

Особливості визначення **елементів множини тестових завдань.** Відповідно до (2.16), кожен елемент множини тестових завдань M_{TestEx} має атрибути *ID*, *Type*, *TEContent* та *Answers*, які запропоновано автоматизовано визначати поданим у табл. 3.3 чином. Атрибути, що визначаються методом експертної оцінки та не впливають на структуру й інші властивості тестового

завдання (час тестування, рівень складності тощо) в даній моделі не розглядаються.

Таблиця 3.2 – Підходи до одержання значень атрибутів елементів множини термінів M_{Term}

Назва атрибуту	**Фізичний зміст атрибуту**	**Підхід до одержання значення атрибуту**
ID	унікальний ідентифікатор елементу	Автоматично привласнені унікальні цілі числа ($M_{ID} \in Z$), що використовуються для однозначної ідентифікації елементів у множині M_{Term}
TermName	символьна назва терміну	всі присутні в контенті унікальні неперервні сукупності символів, що містять пробіл та елементи множині символів у словах $M_{SymWord}$ й не виходять за межі текстових контейнерів (фраз)
TermNum	кількість слів у терміні	кількість послідовностей символів з множини $M_{SymWord}$, які розділені пробілами, $TermNum \in [1; 6]$

Таблиця 3.3 – Підходи до одержання значень атрибутів елементів множини тестових завдань M_{TestEx}

Назва атрибуту	**Фізичний зміст атрибуту**	**Підхід до одержання значення атрибуту**
ID	унікальний ідентифікатор елементу	автоматично привласнені унікальні цілі числа ($M_{ID} \in Z$), що використовуються для однозначної ідентифікації елементів у множині M_{TestEx}
Type	номер типу питання	одержуються в результаті використання методу автоматизованого формування тестових завдань як число ($Type \in Z$), відповідне типу створеного тестового завдання
TEContent	контент тестового завдання чи вказівка на його місцезнаходження	одержуються в результаті використання методу автоматизованого формування тестових завдань як елемент, що однозначно визначає контент створеного тестового завдання
Answers	кількість відповідей	одержуються в результаті використання методу автоматизованого формування тестових завдань як число ($Answers \in Z$), відповідне кількості відповідей у створеному тестовому завданні

Особливості визначення **елементів множини зв'язків.** До множини зв'язків M_{Rel} належать зв'язки, що в залежності від елементів, які ними сполучаються, формують підмножини (див. табл. 2.2):

– множини зв’язків між заголовками та заголовками $M_{Rel:H\text{-}H}$;

– множини зв’язків між заголовками та ключовими термінами $M_{Rel:H\text{-}T}$;

– множини зв’язків між ключовими термінами та словами $M_{Rel:T\text{-}W}$;

– множини зв’язків між заголовками та тестовими завданнями $M_{Rel:H\text{-}TE}$;

– множини зв’язків між ключовими термінами та тестовими завданнями $M_{Rel:T\text{-}TE}$.

В залежності від характеру зв’язку (2.18), подана у вигляді M_{Rel} = (*TypeRel*, *Obj*1, *Obj*2, *Feature*), де *TypeRel* – індекс типу зв’язку; *Obj*1– перша сутність з співвідношення, *Obj*2 – друга сутність з співвідношення, *Feature* – характеристика зв’язку, що має відмінний характер для різних типів зв’язків. Шляхи визначення атрибутів елементів цих множин розглянуто далі.

Особливості визначення **елементів множини зв’язків між заголовками.** Підхід до наповнення множини зв’язків між заголовками базується на використанні існуючих властивостей ІНМ, поданих у вигляді цифрових слабо структурованих текстових документів, та природньої потреби в фрагментації та ієрархічній структуризації ІНМ.

До *множини зв’язків між заголовками* входять елементи, які визначають бінарні зв’язки між двома елементами множини M_{Heding}. Тому для $M_{Rel:H\text{-}H}$ (2.18) набуває вигляду, згідно якого кожен елемент множини зв’язків між заголовками $M_{Rel:H\text{-}H}$ є кортежем наступного вигляду:

$$M_{Rel:H–H} = (1, Heading1, Heading2, 0), \qquad (3.3)$$

де *Heading1* – вказівка на елемент множини M_{Heding}, що вказує на заголовок – «батька» в ієрархічній структурі заголовків; *Heading2* – вказівка на елемент множини M_{Heding}, що вказує на заголовок – «нащадка» в ієрархічній структурі заголовків.

Хоча атрибут *Heading1* визначає елемент множини M_{Heding}, що вказує на заголовок – «батька», а атрибут *Heading2* визначає елемент множини M_{Heding}, що вказує на заголовок – «нащадка» в ієрархічній структурі заголовків, дані вказівники не можуть бути ідентичними (заголовок зв’язаний сам з собою) чи визначати елементи множини M_{Heding} з однаковими значеннями *Grade* (зв’язок

між елементами одного рівня), тому впорядкованість атрибутів *Obj*1 та *Obj*2 у зв'язку не є обов'язковою. Підпорядкованість двох заголовків у зв'язку можна визначити автоматично шляхом порівняння їх значень атрибуту *Grade*, більше число буде вказувати на заголовок – «батька», а менше – на заголовок – «нащадка».

Таким чином, відповідно до (2.18) та (3.3), кожен елемент множини зв'язків між заголовками $M_{Rel:H–H}$ має незмінні атрибути *TypeRel*=1 й *Feature*=0, а також змінні атрибути *Obj*1 = *Heading*1 та *Obj*2 = *Heading*2, які запропоновано автоматизовано визначати поданим у табл. 3.4 чином.

Таблиця 3.4 – Підходи до одержання значень атрибутів елементів множини зв'язків між заголовками $M_{Rel:H–H}$

Назва атрибуту	**Фізичний зміст атрибуту**	**Підхід до одержання значення атрибуту**
TypeRel	ціле число (*TypeRel*$\in Z$), що вказує на тип зв'язку	Для елементів множини зв'язків між заголовками $M_{Rel:H–H}$ завжди *TypeRel*=1
*Obj*1	ціле число, що відповідає атрибуту *ID* заголовка – «батька»	ідентифікатор першої сутності *Heading*1 у зв'язку, яка автоматизовано обирається з множини заголовків M_{Heding}
*Obj*2	ціле число, що відповідає атрибуту *ID* заголовка – «нащадка»	ідентифікатор другої сутності *Heading*2 у зв'язку, яка автоматизовано обирається з множини заголовків M_{Heding}
Feature	додаткова характеристика зв'язку	Для елементів множини заголовків $M_{Rel:H–H}$ завжди *Feature*=0

У зв'язку з тим, що комбінація значень атрибутів *TypeRel, Obj*1 та *Obj*2 для множини зв'язків між заголовками $M_{Rel:H–H}$ є унікальною, то комбінацію цих атрибутів можна використати як унікальний ідентифікатор елементів множини зв'язків між заголовками $M_{Rel:H–H}$.

Оскільки структура заголовків ІНМ є ієрархічною, можливі зв'язки тільки між елементами множини M_{Heding} різних рівнів ієрархії *Grade*; й неможливі зв'язки між елементами множини M_{Heding} з однаковим рівнем ієрархії *Grade*. Це визначає також неможливість формування змістовного зв'язку елементу множини M_{Heding} з самим собою, тому цей випадок

передбачено для визначення найвищого елементу в ієрархічній структурі заголовків ІНМ.

Елементи множин заголовків M_{Heding} і зв'язків між заголовками $M_{Rel:H–H}$ дозволяють сформувати семантичну структуру НМ – мережеву структуру з заголовків та зв'язків підпорядкування між ними, що має ієрархічну властивість. Така структура, що складається з осередків (заголовків) та дуг (зв'язків між заголовками) має бути сформована першою та є основою для подальшої прив'язки до її осередків ключових термінів і тестових завдань.

Особливості визначення **елементів множини зв'язків між заголовками й ключовими термінами** До *множини* $M_{Rel:H\text{-}T}$ входять елементи, які визначають бінарні зв'язки між одним елементом множини M_{Heding} та одним елементом множини M_{Term}. Для множини $M_{Rel:H\text{-}T}$ (2.18) набуває вигляду, згідно якого кожен елемент множини зв'язків між заголовками й ключовими термінами $M_{Rel:H\text{-}T}$ є кортежем наступного вигляду:

$$M_{Rel:\ H\text{-}T} = (2, Heading, Term, Weight), \qquad (3.4)$$

де *Heading* – вказівка на елемент множини M_{Heding}, що вказує на заголовок фрагменту контенту, в рамках якого вивчається термін; *Term* – вказівка на елемент множини M_{Term}, що вказує на ключовий термін в цьому контенті; *Weight* – числовий показник важливості ключового терміну в межах відповідного заголовку *Heading* фрагменту контенту ІНМ.

Відповідно до (2.18) та (3.4), кожен елемент множини зв'язків між заголовками й ключовими термінами $M_{Rel:H\text{-}T}$ має незмінний атрибут *TypeRel*=1, а також змінні атрибути *Obj*1 = *Heading*, *Obj*2 = *Term* та *Feature* = *Weight*, які запропоновано автоматизовано визначати поданим у табл. 3.5 чином.

Оскільки в рамках семантичної моделі навчального курсу між одним заголовком *Heading* та одним терміном *Term* може бути лише один зв'язок (*Heading*, *Term*, *Weight*), факт існування якого є незалежним від кількості згадувань даного терміна *Term* у відповідному заголовкові *Heading* контенті,

то у множині $M_{Rel:H-T}$ може бути лише по одному елементу з унікальними комбінаціями значень атрибутів *Heading* та *Term*.

Таблиця 3.5 – Підходи до одержання значень атрибутів елементів множини зв'язків між заголовками й ключовими термінами $M_{Rel:H-T}$

Назва атрибуту	**Фізичний зміст атрибуту**	**Підхід до одержання значення атрибуту**
TypeRel	ціле число ($TypeRel \in Z$), що вказує на тип зв'язку	для елементів множини зв'язків між заголовками $M_{Rel:H-T}$ завжди $TypeRel=2$
*Obj*1	ціле число, що відповідає атрибуту *ID* заголовка, у межах контенту якого термін визначено як ключовий	одержуються в результаті використання методу формування структури навчальних матеріалів та пошуку у них ключових термінів як атрибуту *ID* елементу множини M_{Heding}
*Obj*2	ціле число, що відповідає атрибуту *ID* ключового терміну	одержуються в результаті використання методу формування структури навчальних матеріалів та пошуку у них ключових термінів як атрибуту *ID* елементу множини M_{Term}
Feature	натуральне число, важливість терміна, для $M_{Rel:H-T}$ *Feature= Weight*	одержуються в результаті використання методу формування структури навчальних матеріалів та пошуку у них ключових термінів як числовий показник важливості терміна

Особливості визначення **елементів множини зв'язків між заголовками та тестовими завданнями** До *множини* $M_{Rel:H-TE}$ входять елементи, які визначають бінарні зв'язки між одним елементом множини $M_{Heading}$ та одним елементом множини M_{TestEx}. Для множини $M_{Rel:H-TE}$ (2.18) набуває вигляду, згідно якого кожен елемент цієї множини є кортежем наступного вигляду:

$$M_{Rel:H-TE} = (3, Heading, TestEx, Cont), \quad (3.5)$$

де *Heading* – вказівка на елемент множини $M_{Heading}$, за контентом якого створене тестове завдання; *TestEx* – вказівка на елемент множини M_{TestEx}, що вказує на визначене тестове завдання, *Cont* – ціле число ($Cont \in Z$), що вказує на порядковий номер речення в складі контенту елементу множини $M_{Heading}$, за яким і для перевірки якого створене тестове завдання.

Таким чином, відповідно до (2.18) та (3.5), кожен елемент множини $M_{Rel:H\text{-}TE}$ має незмінний атрибут *TypeRel*=3, а також змінні атрибути *Obj*1 = *Heading*, *Obj*2 = *TestEx* й *Feature* = *Cont*, які запропоновано автоматизовано визначати поданим у табл. 3.6 чином.

Елементи множини зв'язків між заголовками та тестовими завданнями $M_{Rel:H\text{-}TE}$ дозволяють встановити, кожне тестове завдання призначене для перевірки контенту якого елементу структури ІНМ, й який саме фрагмент ІНМ перевіряється з точністю до визначеного речення. Потреба в існуванні цієї множини викликана тим, що визначення зв'язку терміну й тестового завдання є недостатнім, адже визначений термін може визначатись як ключовий у декількох елементах структури ІНМ, а для подальшого забезпечення проведення адаптивного тестування цього недостатньо.

Таблиця 3.6 – Підходи до одержання значень атрибутів елементів множини зв'язків між заголовками та тестовими завданнями $M_{Rel:H\text{-}TE}$

Назва атрибуту	**Фізичний зміст атрибуту**	**Підхід до одержання значення атрибуту**
TypeRel	ціле число (*TypeRel*$\in$*Z*), що вказує на тип зв'язку	для елементів множини зв'язків між термінами та словами $M_{Rel:H\text{-}TE}$ завжди *TypeRel*=3
*Obj*1	ціле число, що відповідає атрибуту *ID* заголовка, за контентом якого створене тестове завдання	одержуються в результаті використання методу автоматизованого формування тестових завдань як атрибут *ID* елементу множини M_{Heding}
*Obj*2	ціле число, що відповідає атрибуту *ID* тестового завдання	одержуються в результаті використання методу автоматизованого формування тестових завдань як атрибут *ID* елементу множини M_{TestEx}
Feature	натуральне число, важливість терміна, для $M_{Rel:H\text{-}TE}$ *Feature= Cont*	одержуються в результаті використання методу автоматизованого формування тестових завдань як порядковий номер речення, за яким створене тестове завдання

Особливості визначення **елементів множини зв'язків між ключовими термінами та тестовими завданнями** До *множини* $M_{Rel:T\text{-}TE}$ входять елементи, які визначають зв'язки між одним елементом множини M_{Term} та одним елементом множини M_{TestEx}. Елементи даної множини зв'язків

дозволяють вказувати на використання визначеного ключового терміна в контенті визначеного тестового завдання.

Для множини $M_{Rel:T\text{-}TE}$ (2.18) набуває вигляду, згідно якого кожен елемент множини зв'язків між ключовими термінами та тестовими завданнями $M_{Rel:T\text{-}TE}$ є кортежем наступного вигляду:

$$M_{Rel:T\text{-}TE} = (4, Term, TestEx, Loc), \quad (3.6)$$

де *Term* – вказівка на елемент множини M_{Term}, що визначає ключовий термін, який використовується в тестовому завданні; *TestEx* – вказівка на елемент множини M_{TestEx}, що вказує на визначене тестове завдання, *Loc* – ціле число ($Loc \in Z$), що вказує на тип використання або місце використання терміну *Term* в тестовому завданні *TestEx*.

Згідно (2.18) та (3.6), кожен елемент множини $M_{Rel:T\text{-}TE}$ має незмінний атрибут $TypeRel = 4$, а також змінні атрибути $Obj1 = Term$, $Obj2 = TestEx$ й $Feature = Loc$, які запропоновано автоматизовано визначати поданим у табл. 3.7 чином.

Таблиця 3.7 – Підходи до одержання значень атрибутів елементів множини зв'язків між ключовими термінами та тестовими завданнями $M_{Rel:T\text{-}TE}$

Назва атрибуту	**Фізичний зміст атрибуту**	**Підхід до одержання значення атрибуту**
TypeRel	ціле число ($TypeRel \in Z$), що вказує на тип зв'язку	для елементів множини зв'язків між термінами та словами $M_{Rel:T\text{-}TE}$ завжди $TypeRel=4$
*Obj*1	ціле число, що відповідає атрибуту *ID* терміна, який використовується в тестовому завданні	одержуються в результаті використання методу автоматизованого формування тестових завдань як атрибут *ID* елементу множини M_{Term}
*Obj*2	ціле число, що відповідає атрибуту *ID* тестового завдання	одержуються в результаті використання методу автоматизованого формування тестових завдань як атрибут *ID* елементу множини M_{TestEx}
Feature	натуральне число, важливість терміна, для $M_{Rel:T\text{-}TE}$ $Feature = Loc$	одержуються в результаті використання методу автоматизованого формування тестових завдань як вказівник на тип або місце використання терміну в тестовому завданні

Показник типу або місця використання терміну в тестовому завданні *Loc* приймає значення за результатами формального віднесення наявного використання терміну до одної з категорій (наприклад: значення «1» визначає використання терміна в нормальному вигляді у контенті тестового запитання; значення «5» визначає використання терміна в лематизованому вигляді у контенті правильної відповіді до тестового запитання тощо). Оскільки можливе кілька включень одного терміна до контенту одного тестового завдання, то відповідно припустима одночасна наявність кількох елементів множини $M_{Rel:T\text{-}TE}$ з ідентичними значеннями атрибутів *Term* і *TestEx*. Також відповідно припустима одночасна наявність кількох ідентичних елементів множини $M_{Rel:T\text{-}TE}$ (з ідентичними значеннями всіх атрибутів); відповідно, кількість таких елементів теж є значущим показником моделі семантичної структури навчального курсу.

3.5 Застосування правил продукції тестових завдань

Із характеристик продукційних систем, наведених в п. 1.2.4, випливає, що при обробці одного фрагменту тексту для визначення рівня знань одного терміна можливе використання кількох різних правил продукції.

Для продукції тестових завдань далі наведено формальне подання за допомогою тегів відповідних правил (див. п. 2.4.2).

3.5.1 Подання правил продукції

Для формального опису правил продукції, зокрема для ідентифікації елементів контенту антецедентом та для формування тестових завдань консеквентом, використано множину тегів. Запропоновані теги можна віднести до елементів псевдокоду, або змінних, які під час роботи алгоритму приймають різні значення. Теги для ідентифікації елементів контенту антецедентом наведено у табл. 3.8.

Таблиця 3.8 – Теги для ідентифікації елементів контенту антецедентом

Тег	Опис
<Antecedent>	Вказує на початок антецедента правила продукції
[Term]	Термін, що складається з множини слів
[Thesis]	Фрагмент тексту (теза), який дає визначення терміну
[RandomTerm]	Випадково обраний інший термін
[RandomThesis]	Випадково обране визначення іншого терміну
[Connector]	Елемент множини сполучних фрагментів, що зазвичай поєднують в тексті термін з його визначенням
[ConnectorFeature]	Елемент множини сполучних фрагментів, що зазвичай поєднують в тексті термін з описом його деякої властивості
[ConnectorMulti]	Елемент множини сполучних фрагментів, після яких зазвичай подається в тексті перелік однорідних елементів
[Text]	Деякий текст, що не ідентифікується іншими тегами
[Caps1]	Переведення першої букви до верхнього регістру
[ReCaps1]	Переведення першої букви до нижнього регістру
[TermSentence]	Фрагмент тексту, який містить термін
[Counter]	Елемент переліку
[NonCounter]	Завершальний елемент переліків
[Union]	Сполучники елементів переліку
_X	Порядковий номер X елементу одного типу в реченні: [Text_X]
=0	Можлива відсутність контенту відповідного тегу: [Counter=0]

Далі наведено деякі приклади фрагментів тексту, що визначають вміст тегів.

Тег [Connector] визначає множину $M_{Connector}$, й для його активації необхідна наявність одного з її елементів, наприклад таких:

«–», «– це», «є», «називається», «означає», «представляє» $\in M_{Connector}$.

Тег [ConnectorFeature] визначає множину $M_{ConnectorFeature}$, й для його активації необхідна наявність одного з її елементів, наприклад таких:

«описує», «включає», «входить до», «належить» «має» $\in M_{ConnectorFeature}$.

Тег [ConnectorMulti] визначає множину $M_{ConnectorMulti}$ сполучних фрагментів, після яких зазвичай подається в тексті перелік однорідних елементів. Для його активації необхідна наявність одного з елементів, наприклад:

«:», «розрізняють», «наступні:», «бувають», «такі:» $\in M_{ConnectorMulti}$.

Тег [Union] визначає множину M_{Unions} сполучників елементів переліків, і для його активації необхідна наявність одного з елементів, наприклад:

«,», « і», «та», «й», «, а також» $\in M_{ConnectorMulti}$.

Тег [NonCounter] визначає множину $M_{NonCounter}$ завершальних елементів переліків, й для його активації необхідна наявність одного з елементів:

«тощо», « і т.д.», «та інші», «і т.п.», «і подібні.», «.» $\in M_{ConnectorMulti}$.

Для наповнення наведених множин використано результати кількісного аналізу контенту ІНМ при його ручному використанні для формування тестів.

Розроблені теги для формування тестів консеквентом подано в табл. 3.9. Ця множина тегів призначена для ідентифікації складових тестового завдання та їх вихідних параметрів. До таких складових тестового завдання належать текст питання та текст варіантів відповіді, а параметрами варіантів відповіді є її правильність чи хибність.

Таблиця 3.9 – Теги для формування тестів консеквентом правила продукції

Тег	Опис
<Consequence>	Вказує на початок консеквента правила продукції
{Header}	Візуальний визначник для тексту питання
{Answer}	Візуальний визначник блоку відповідей
{FALSE}	Вказівка на неправильний варіант відповіді
{TRUE}	Вказівка на правильний варіант відповіді

З використанням тегів, опис антецедента та консеквента правила продукції зменшується за об’ємом та набуває більшої формалізації, що спрощує розробку відповідних алгоритмів. Приклад формалізації правила продукції для створення тестового завдання з множинним вибором подано для наступного правила:

Антецедент. Якщо речення відповідає наступним вимогам:

– речення починається з фрагменту, що містить ключовий термін;

– після фрагменту, що містить ключовий термін, слідує сполучний фрагмент тексту із числа наступних: «:», «розрізняють», «наступні:», «бувають», «такі:» тощо;

– після сполучного фрагменту наявні 2-4 елементи тексту, що розділяються комою чи відповідним сполучником;

– перелічення елементів переривається при зустрічі завершального елементу переліків із числа наступних: «тощо», « і т.д.», «та інші», «і т.п.», «і подібні.», «.».

Консеквент. Можна створити тестове завдання типу множинного вибору наступним чином:

– текст з початку речення, що містить ключовий термін, до сполучного фрагменту перетворити в запитання тесту;

– контентом 2 правильних відповідей тесту встановити 1 та 2 елементи тексту після сполучного фрагменту, що розділяються комою;

– далі, контентом 1-2 правильних відповідей тесту встановити 3 та 4 елементи тексту після сполучного фрагменту, що розділяються комою або сполучником, якщо такі є;

– далі, контентом 2 неправильних відповідей тесту встановити випадкові інші два терміни з даного фрагменту тексту.

В результаті подання цього правила продукції засобами тегів воно прийме наступний вигляд:

<Antecedent> [TermSentence] [ConnectorMulti] [Counter_1] [Union] [Counter_2] [Union] [Counter_3=0] [Union] [Counter_4=0] [NonCounter]

<Consequence> {Header} [TermSentence] [ConnectorMulti] {Answer} {TRUE} [Counter_1] {TRUE} [Counter_2] {TRUE} [Counter_3=0] {TRUE} [Counter_4=0] {FALSE} [RandomTerm] {FALSE} [RandomTerm]

На рис. 3.7 схематично зображено процес використання розглянутого правила продукції для створення тестового завдання. Вхідними даними є

обраний для підбору правил фрагмент контенту ІНМ. Вихідними даними є прототип тестового завдання як елемент ТНМ, що може бути використаний для експорту в підсистеми тестування навчальних середовищ.

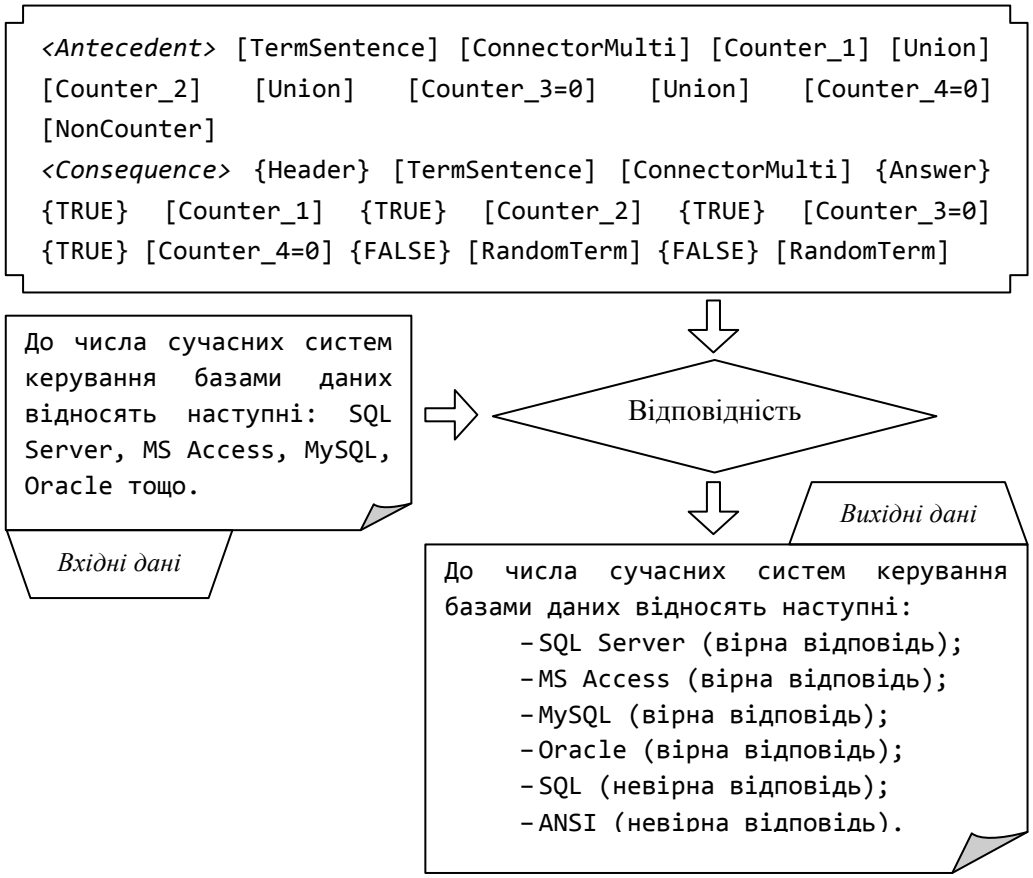


Рисунок 3.7 – Приклад використання правила продукції

Результатом множини випадків застосування правил продукції є множина тестових завдань. Вона може бути перенесена до відповідного середовища тестування за допомогою передбачених у ньому засобів. Наприклад, у середовищі Moodle можливий імпорт множин тестових завдань у форматах .XML, .XHTML та .GIFT (див. п. 1.1.4). Так, у форматі .GIFT наведений приклад сформованого тестового завдання прийме вигляд, зображений на рис. 3.8. При експорті наведеного контенту у форматі .GIFT, тестове завдання набуде вигляду, наведеного на рис. 3.9.

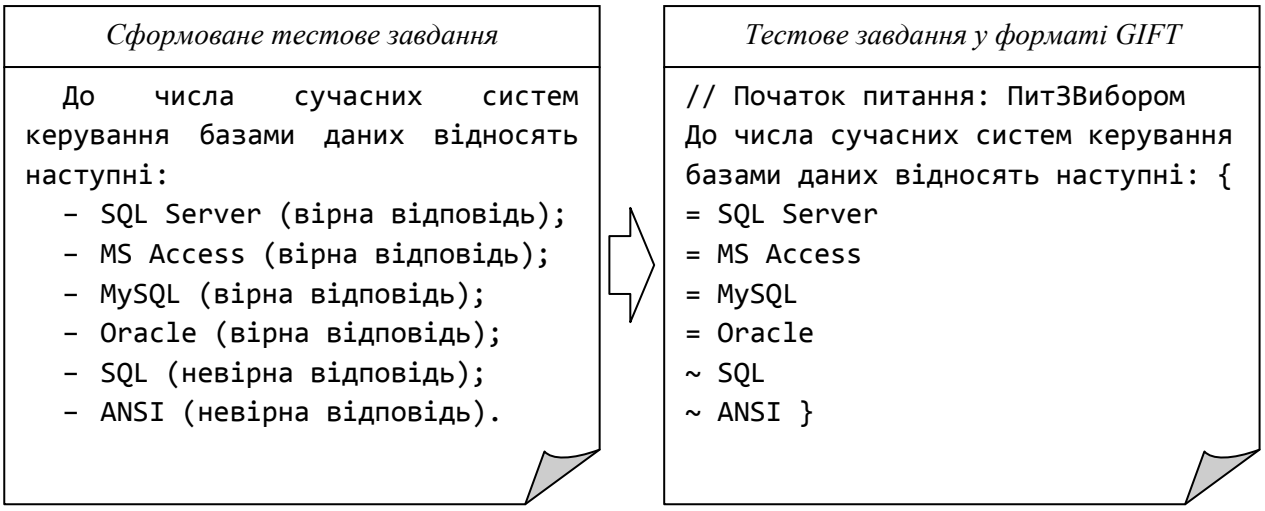


Рисунок 3.8 – Приклад збереження створеного тестового завдання у форматі .GIFT

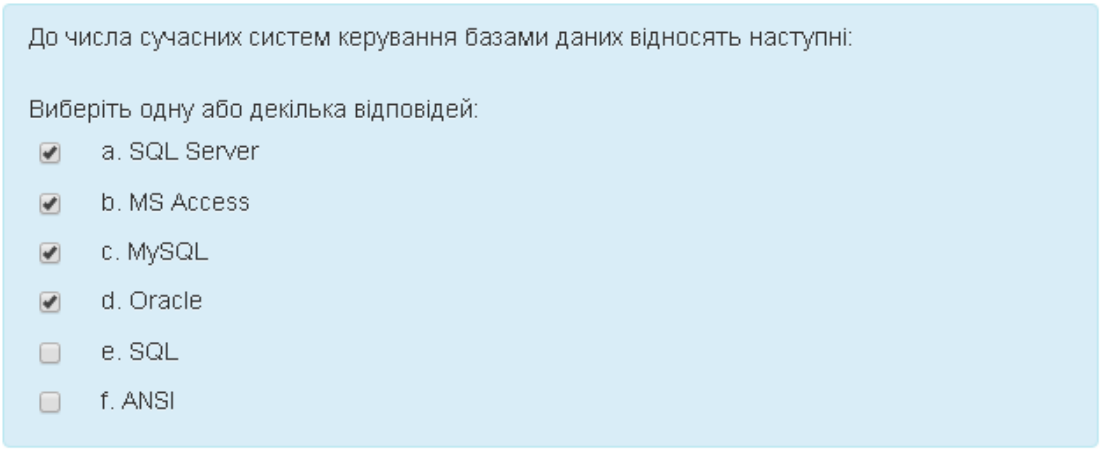


Рисунок 3.9 – Використання у середовищі Moodle створеного тестового завдання, експортованого у форматі .GIFT

При необхідності, окрім множини тестових завдань, для подальшої роботи з тестами можуть бути використані інші вихідні дані інформаційної технології (див. рис. 3.2).

3.5.2 Приклади правил продукції тестових завдань

В п. 2.4.2 розглянуто 2 антецеденти (a_1, a_2) та 11 консеквентів (c_{a1}, c_{a2}, c_{a3}, c_{b1}, c_{b2}, c_{b3}, c_{b4}, c_{c1}, c_{c2}, c_{d1}, c_{d2}) правил продукції тестових завдань, що

формують множину з 17-ти їх можливих комбінацій як правил продукції (m_{1a1}, m_{2a1}, m_{1a2}, m_{2a2}, m_{1a3}, m_{2a3}, m_{1b1}, m_{1b2}, m_{1b3}, m_{2b3}, m_{2b4}, m_{1c1}, m_{2c1}, m_{1c2}, m_{2c2}, m_{1d1}, m_{2d2}). Далі наведено зразки правил продукції для створення тестових завдань різних типів.

Правила продукції тестових завдань логічного типу згідно табл. 2.4 наступні: m_{1a1}, m_{2a1}, m_{1a2}, m_{2a2}, m_{1a3}, m_{2a3}. Для них створено наступні зразки формального опису.

Для $m_{1a1} = (a_1 \Rightarrow c_{a1})$:

<Antecedent> [Text=0] [Term] [Connector] [Thesis]

<Consequence> {Header} [AnswerT1] [Term] [Connector] [Thesis] {Answer} {TRUE}

Для $m_{1a2} = (a_1 \Rightarrow c_{a2})$:

1) *<Antecedent>* [Text=0] [Term] [Connector] [Thesis]

<Consequence> {Header} [AnswerT1] [RandomTerm] [Connector] [Thesis] {Answer} {FALSE}

2) *<Antecedent>* [Text=0] [Term] [Connector] [Thesis]

<Consequence> {Header} [AnswerT1] [Term] [Connector] [RandomThesis] {Answer} {FALSE}

Для $m_{1a3} = (a_1 \Rightarrow c_{a3})$:

<Antecedent> [Text=0] [Term] [Connector] [Thesis]

<Consequence> {Header} [AnswerT1] [Term] [Connector] [Not] [Thesis] {Answer} {FALSE}

Для $m_{2a1} = (a_2 \Rightarrow c_{a1})$:

<Antecedent> [Text=0] [Term] [Text=0]

<Consequence> {Header} [AnswerT1] [Text=0] [Term] [Text=0] {Answer} {TRUE}

Для $m_{2a2} = (a_2 \Rightarrow c_{a2})$:

<Antecedent> [Text=0] [Term] [Text=0]

<Consequence> {Header} [AnswerT1] [Text=0] [RandomTerm] [Text=0] {Answer} {FALSE}

Для $m_{2a3} = (a_2 \Rightarrow c_{a3})$:

<Antecedent> [Text=0] [Term] [Text=0]

<Consequence> {Header} [AnswerT1] [Text=0] [Term] [Not] [Text=0] {Answer} {FALSE}

Правила продукції тестових завдань одиничного вибору згідно табл. 2.4 наступні: m_{1b1}, m_{1b2}, m_{1b3}, m_{2b3}, m_{2b4}. Для них створено наступні зразки формального опису.

Для $m_{1b1} = (a_1 \Rightarrow c_{b1})$:

1) *<Antecedent>* [Text=0] [Term] [Connector] [Thesis]

<Consequence> {Header} [AnswerT2-1] [Thesis] [Connector] {Answer} {TRUE} [Term] {FALSE} [RandomTerm_1] {FALSE} [RandomTerm_2] {FALSE} [RandomTerm_3]

2) *<Antecedent>* [Thesis] [Connector] [Term] [Text=0]

<Consequence> {Header} [AnswerT2-1] [Thesis] [Connector] {Answer} {TRUE} [Term] {FALSE} [RandomTerm_1] {FALSE} [RandomTerm_2] {FALSE} [RandomTerm_3]

Для $m_{1b2} = (a_1 \Rightarrow c_{b2})$:

1) *<Antecedent>* [Text=0] [Term] [Connector] [Thesis]

<Consequence> {Header} [AnswerT2-2] [Term] [Connector] {Answer} {TRUE} [Thesis] {FALSE} [RandomThesis_1] {FALSE} [RandomThesis_2] {FALSE} [RandomThesis_3]

2) *<Antecedent>* [Thesis] [Connector] [Term] [Text=0]

<Consequence> {Header} [AnswerT2-2] [Term] [Connector] {Answer} {TRUE} [Thesis] {FALSE} [RandomThesis_1] {FALSE} [RandomThesis_2] {FALSE} [RandomThesis_3]

Для $m_{1b4} = (a_1 \Rightarrow c_{b4})$:

<Antecedent> [Text=0] [Term] [Text=0]

<Consequence> {Header} [AnswerT2-4] [Text=0] “_____” [Text=0] {Answer} {TRUE} [Term] {FALSE} [RandomTerm_1] {FALSE} [RandomTerm_2] {FALSE} [RandomTerm_3]

Правила продукції тестових завдань множинного вибору згідно табл. 2.4 наступні: m_{1c1}, m_{2c1}, m_{1c2}, m_{2c2}. Для них створено наступні зразки формального опису.

Для $m_{2c2} = (a_2 \Rightarrow c_{c2})$:

<Antecedent> [Text=0] [Term] [Text=0] [ConnectorMulti] [Counter_1] [Union] [Counter_2] [Union] [Counter_3=0] [Union] [Counter_4=0] [NonCounter]

<Consequence> {Header} [Text=0] [Term] [Text=0] [ConnectorMulti] {Answer} {TRUE} [Counter_1] {TRUE} [Counter_2] {TRUE} [Counter_3=0] {TRUE} [Counter_4=0] {FALSE} [RandomTerm] {FALSE} [RandomTerm]

Правила продукції тестових завдань на введення тексту згідно табл. 2.4 наступні: m_{1d1}, m_{2d2}. Для них створено наступні зразки формального опису.

Для $m_{1d1} = (a_1 \Rightarrow c_{d1})$:

1) *<Antecedent>* [Text=0] [Term] [Connector] [Thesis]

<Consequence> {Header} [AnswerT4-1] [Thesis] [Connector] {Answer} {TRUE} [Term]

2) *<Antecedent>* [Thesis] [Connector] [Term] [Text=0]

<Consequence> {Header} [AnswerT4-1] [Thesis] [Connector] {Answer} {TRUE} [Term]

Для $m_{2d2} = (a_2 \Rightarrow c_{d2})$:

<Antecedent> [Text=0] [Term] [Text=0]

<Consequence> {Header} [AnswerT4-2] [Text=0] “_____” [Text=0] {Answer} {TRUE} [Term]

В Додатку Г наведено приклади тестових завдань, створених за розглянутими зразками правил продукції.

В залежності від використаної мови, предметної області та інших особливостей ІНМ, множина зразків правил продукції та множини вмісту тегів можуть бути доповнені в межах поданих в табл. 2.4 правил продукції.

Висновки до розділу

1. Запропоновано ІТ автоматизованого створення тестів до навчальних матеріалів, що дозволяє за вхідними даними у вигляді ІНМ курсу автоматизовано одержувати множину тестових завдань. Важливою рисою ІТ є формування тестових завдань для цілеспрямованої перевірки рівня знань

конкретних ключових термінів у ієрархічному розрізі рубрик навчальних матеріалів курсу дисципліни.

2. Розроблено архітектуру інформаційної системи для автоматизованого створення тестів, що складається з двох підсистем (підсистеми формування структури ІНМ та пошуку ключових термінів й підсистеми автоматизованого формування тестових завдань), й яка створена за ІТ автоматизованого створення тестів до навчальних матеріалів. Визначено схему розподілу функцій за компонентами системи. Підсистема формування структури ІНМ та пошуку ключових термінів виконує функції обробки контенту вхідного документу, аналізу структури документу, формування структури ІНМ, визначення контенту елементів структури ІНМ, пошуку ключових термінів у елементах структури ІНМ та формування множин ключових термінів. Підсистема автоматизованого формування тестових завдань виконує функції аналізу речень в яких присутні терміни на відповідність правилам автоматизованого створення тестових завдань та формування вихідної множини тестових завдань. Визначено та класифіковано функції користувача інформаційної системи.

3. Наведено особливості застосування ІТ, зокрема рекомендовані вимоги до попередньої технічної обробки тексту, яка полягає в усуненні неоднозначного іменування термінів та обробці розділових знаків.

4. Визначено шляхи практичного одержання елементів множин моделі семантичної структури НК за допомогою метода формування структури навчальних матеріалів та пошуку у них ключових термінів й метода автоматизованого формування тестових завдань в межах розробленої ІТ.

5. Розроблено зразки правил продукції, необхідні для формування тестових завдань, які наведено за допомогою формального опису з використанням тегів. Створено правила продукції тестових завдань наступних типів: логічного типу, одиничного вибору, множинного вибору та із введенням тексту. Одержана множина з 17 правил продукції дозволяє створювати всі можливі тестові завдання за всіма потенційно придатними реченнями.

РОЗДІЛ 4.
ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АВТОМАТИЗОВАНОГО ФОРМУВАННЯ ТЕСТОВИХ ЗАВДАНЬ ДО НАВЧАЛЬНИХ МАТЕРІАЛІВ

Для експериментального тестування ІТ автоматизованого формування тестових завдань до навчальних матеріалів розроблено відповідні прикладні програмні системи. За результатами їх роботи по вирішенню задач формування структури навчальних матеріалів і пошуку в них ключових термінів, автоматизованого формування тестових завдань та їх використання для проведення адаптивного тестування можна визначити ефективність розробленої ІТ.

4.1 Програмне забезпечення для експериментального тестування

4.1.1 Інформаційна система для автоматизованого створення тестів

Відповідно до наведеної в п. 3.1 та 3.2 структури ІТ, створена на її базі інформаційна система складається з двох підсистем – підсистеми формування структури ІНМ та пошуку ключових термінів й підсистеми автоматизованого формування тестових завдань.

Підсистема формування структури ІНМ та пошуку ключових термінів виконує функції обробки контенту вхідного документу для формування структури документу та пошуку ключових термінів.

Підсистема автоматизованого формування тестових завдань виконує функції аналізу речень в яких присутні терміни на відповідність правилам автоматизованого створення тестових завдань та формування вихідної множини тестових завдань.

Програмна структура інформаційної системи складається з класів, згрупованих у модулі, які зображені на рис. 4.1. Докладно структуру цих

модулів інформаційної системи для автоматизованого створення тестів розглянуто в Додатку А, а лістинги програмного коду класів модулів інформаційної системи для автоматизованого створення тестів наведено в Додатку Б.

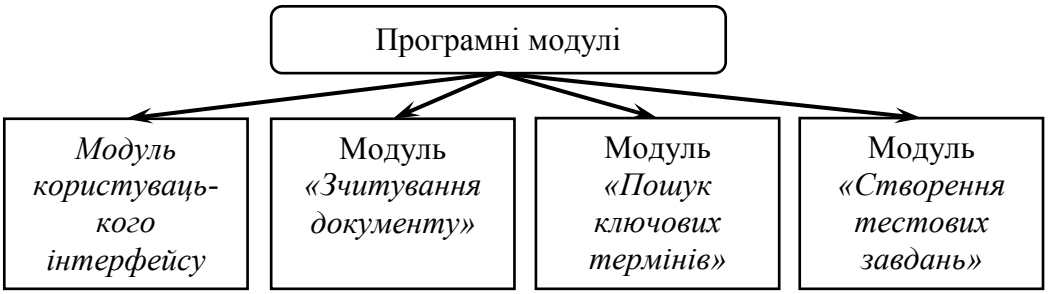


Рисунок 4.1 – Схема програмних модулів системи

Після вибору документу та параметрів роботи визначається структура ІНМ (відображається в формі дерева), для кожного вузла якої формується множина ключових термінів (рис. 4.2).

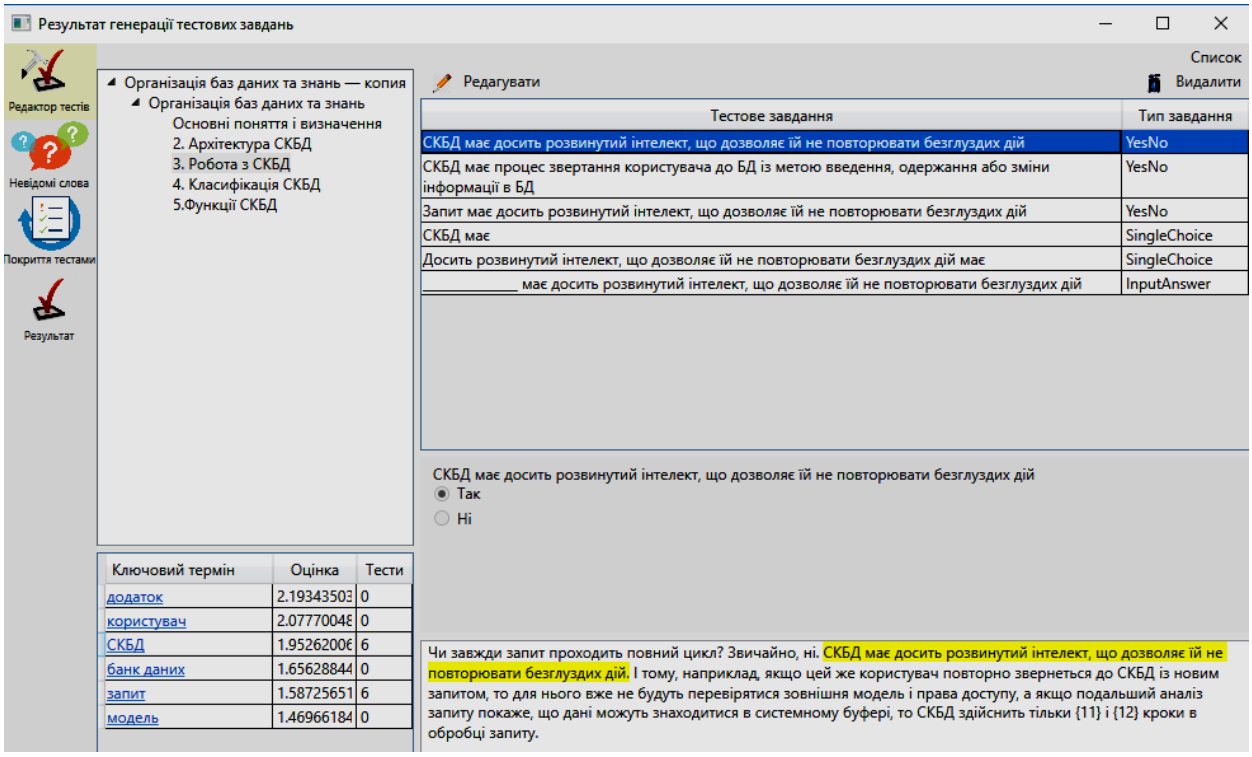


Рисунок 4.2 – Робота з створеними тестовими завданнями

Відображається перелік ключових термінів з оцінкою важливості та кількістю створених тестових завдань для кожного терміну. До обраного ключового терміну відображаються тестові завдання, призначені для його перевірки. Кожне створене тестове завдання можна відкоригувати чи видалити. Передбачена можливість перегляду покриття контенту ІНМ тестовими завданнями (рис. 4.3).

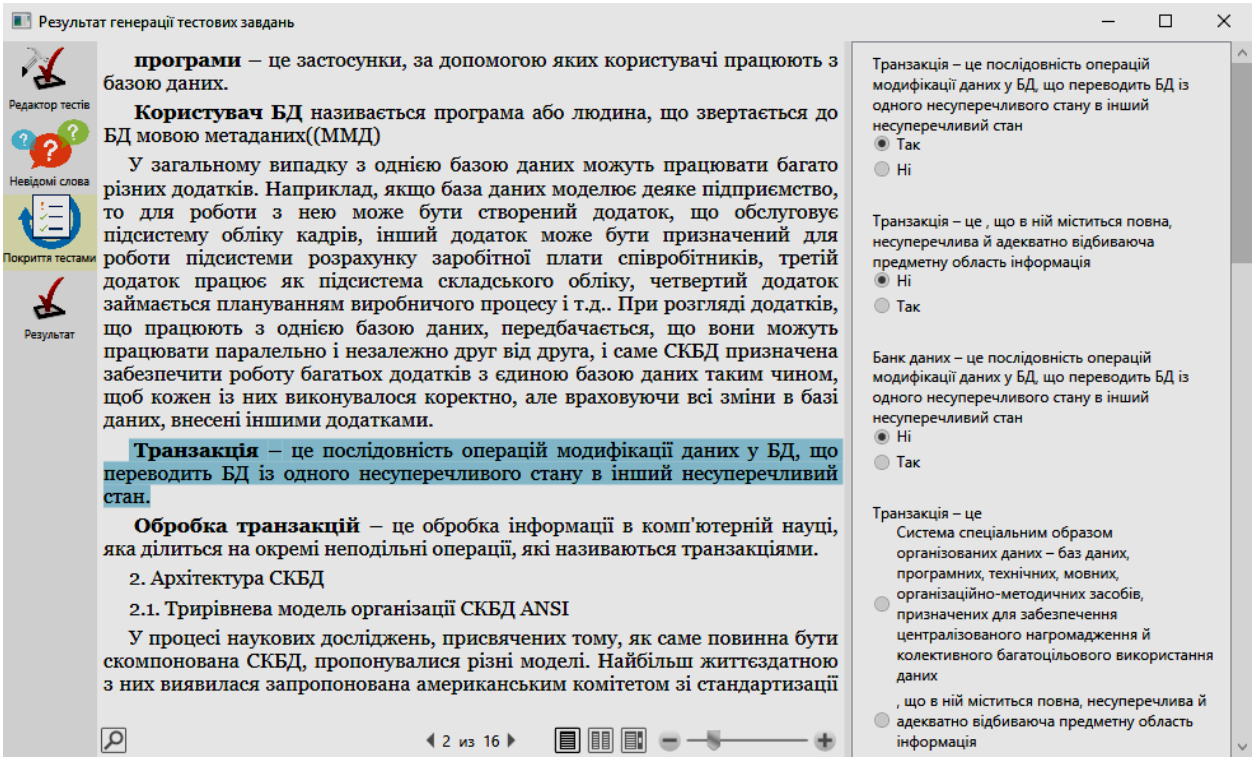


Рисунок 4.3 – Перегляд покриття текстового контенту тестовими завданнями

В робочій області «Результат» реалізована можливіть перегляду всіх створених тестових завдань для кожної рубрики документу ІНМ (рис. 4.4). Множина тестових завдань може бути збережена як файл .gift для експорту в навчальне середовище та в БД системи для подальшої роботи з метаданими тесту (зв'язок тестових завдань з ключовими термінами та заголовками тощо).

Створене застосування дає змогу формування множини тестових завдань та наповнення всіх множин моделі семантичної структури навчального курсу відповідно до розробленої ІТ.

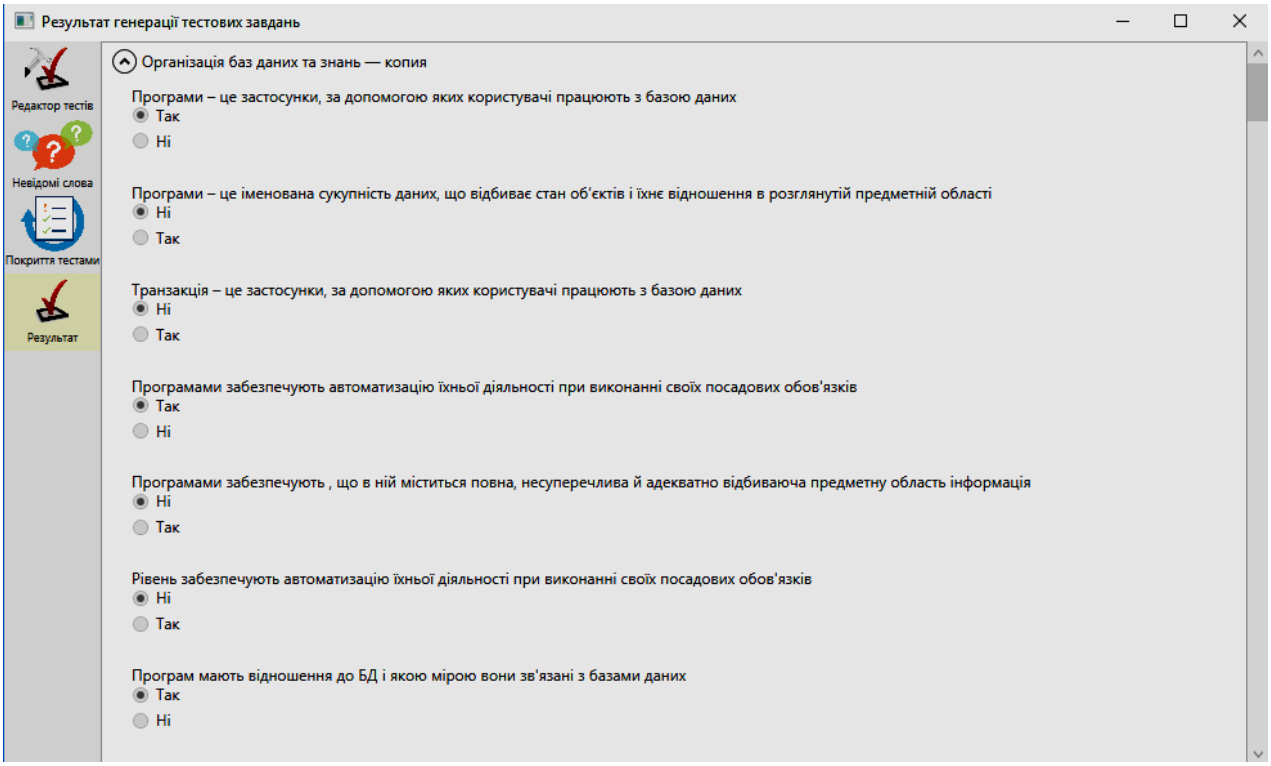


Рисунок 4.4 – Перегляд створених тестових завдань

Етапами роботи інформаційної системи для автоматизованого створення тестів є: визначення структури й семантичних термінів у контенті навчальних матеріалів шляхом застосування методу формування структури навчальних матеріалів та пошуку в них ключових термінів; формування тестових завдань шляхом застосування методу автоматизованого формування тестових завдань; перенесення результатів в область взаємодії з користувачем – окремого програмного продукту чи підсистеми тестування віртуального навчального середовища, наприклад Moodle.

4.1.2 Застосування для адаптивного тестування

Передбачена в моделі семантичної структури навчального курсу (див. п. 2.2) можливість збереження даних не тільки окремого тестового завдання, а й його зв’язку з множинами ключових термінів і заголовків, реалізована в розробленій ІТ (див. п. 3.1), забезпечує можливість використання створених множин тестових завдань не тільки для проведення тестування за класичним

алгоритмом (випадковий підбір тестових завдань до тесту), а й для адаптивного тестування, при якому використовується алгоритм, наведений у п. 3.3.4).

Розроблено програмну систему для адаптивного тестування, що складається з:

– підсистеми розробки тестів, створеної з використанням мови програмування C# на платформі .NET Framework;

– засобу для проведення адаптивного тестування у вигляді плагіна до середовища Moodle, створеного з використанням мови програмування PHP.

Підсистема розробки тестів дозволяє завантажувати вихідні дані програмного продукту, описаного в п. 4.1.1, у вигляді множини тестових завдань та метаданих до них, які визначають зв’язки з множиною ключових термінів та з множиною заголовків, коригувати ці дані та формувати пакет тестових завдань у вигляді файлу .xml (рис. 4.5).

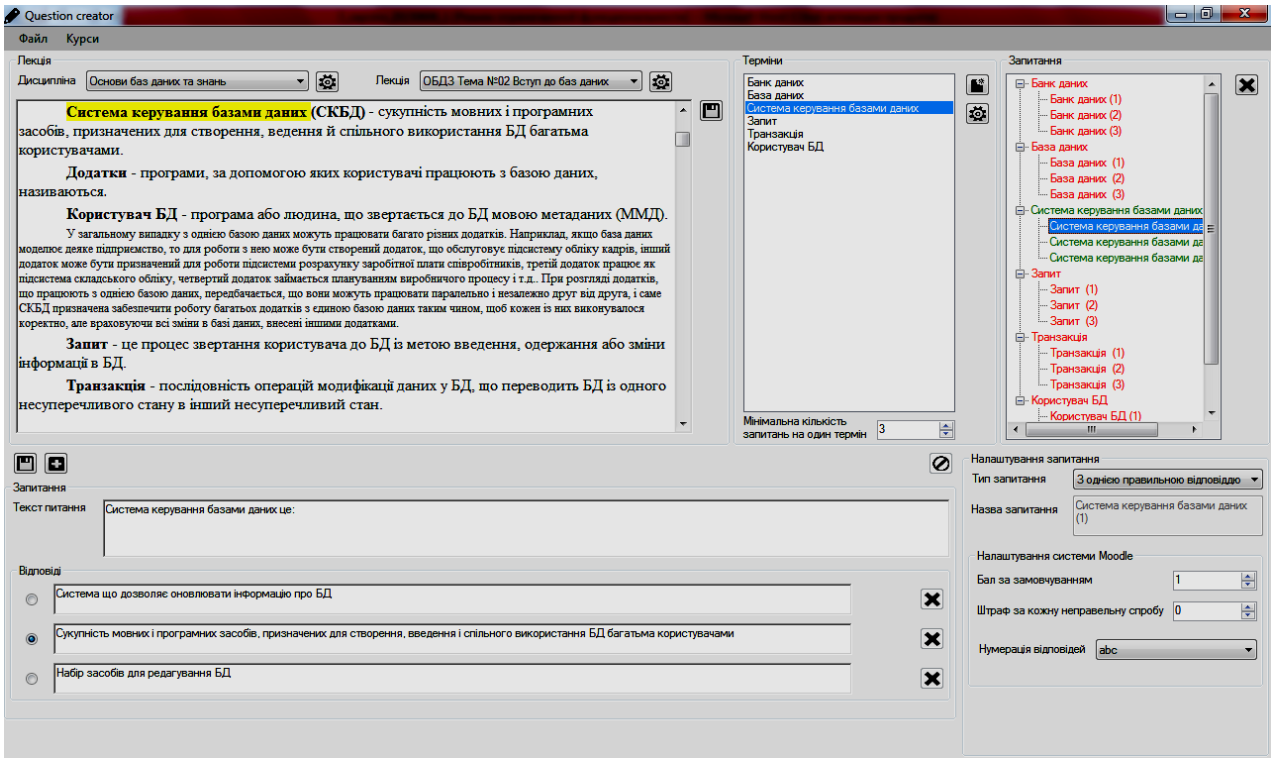


Рисунок 4.5 – Інтерфейс підсистеми розробки тестів

Цей файл формується за стандартами Moodle так, щоб він міг бути завантажений в це середовище стандартними засобами Moodle. Після завантаження ці запитання зберігаються в базі даних Moodle і можуть бути використані будь-якою системою тестування в межах цього середовища.

Засіб для проведення адаптивного тестування реалізовано у вигляді плагіна до середовища Moodle, який додає йому функціональну можливість проведення адаптивного тестування за одним із існуючих (див. п. 1.2.5) алгоритмів (рис. 4.6).

Реалізація функціональності модуля адаптивного тестування потребує збереження тестових завдань, налаштувань тесту, та результатів тестування у базі даних. Система Moodle має свою базу, яка дозволяє додавати до неї таблиці, необхідні для роботи нового модуля (рис. 4.7).

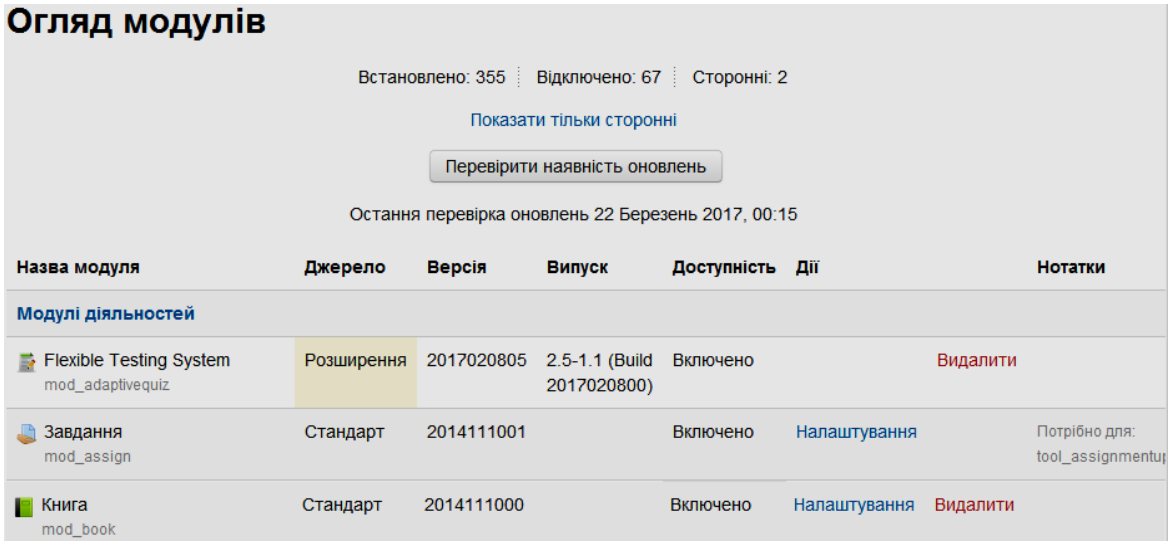


Рисунок 4.7 – Встановлений плагін для проведення адаптивного тестування у середовищі Moodle

Розроблена програмна система призначена для дослідження можливості й ефекту від використання створених відповідно до IT множин тестових завдань для адаптивного тестування.

4.2 Експериментальне тестування інформаційної технології

4.2.1 Визначення шляхів дослідження

За запропонованою ІТ розроблена інформаційна система, призначена для:

– створення розміченої структури документу ІНМ та знаходження множин ключових термінів, що відображають сенс кожного елементу;

– створення для кожного ключового терміну тестових завдань різних типів та синтаксичних конструкцій, які максимально повно і рівномірно покривають контент відповідної рубрики й придатні для проведення адаптивного тестування.

Валідність запропонованої ІТ можливо підтвердити проведенням наступних досліджень для тестової вибірки документів з ІНМ:

1) оцінка *точності та повноти визначення множин ключових термінів* для контенту рубрик ІНМ – шляхом порівняння одержаних автоматично множин із множинами ключових термінів, сформованих авторами ІНМ;

2) оцінка *повноти покриття контенту ключовими термінами* з створеної множини – шляхом визначення відношення кількості окремих семантичних блоків (рубрик, абзаців і речень) тексту із присутніми ключовими термінами до загальної кількості таких блоків у тексті;

3) оцінка *повноти покриття контенту створеними тестовими завданнями* (як кожного з типів, так і загалом) – шляхом визначення відношення кількості використаних для створення тестових завдань рубрик, абзаців і речень тексту до загальної їх кількості в тексті; *рівномірність покриття контенту створеними тестовими завданнями* є кількісним поданням повноти покриття контенту створеними тестовими завданнями й обраховується одночасно з повнотою;

4) *зменшення часу, необхідного на формування множини тестових завдань* – шляхом визначення різниці між часом, необхідним на відповідну роботу за її ручного виконання, та часом, необхідним на досягнення аналогічного результату за використання розробленої технології та ручної корекції автоматизовано створеної множини тестових завдань;

5) можливість використання створених тестів для адаптивного тестування.

Ефективність отримання *структури ІНМ* не потребує окремих досліджень, оскільки є похідною від коректності використання стилів електронних документів авторами ІНМ, й при дотриманні базових вимог роботи з стилями визначається ефективністю використаних існуючих спеціалізованих програмних розширень для парсингу та обробки одержаних даних.

Для проведення досліджень було використано такі тестові вибірки ІНМ.

Тестова вибірка №1 складається з множини всіх фахових дисциплін освітнього рівнів «бакалавр» та «магістр» спеціальності «122 – Комп'ютерні науки», що використовуються в навчальному процесі на кафедрі Комп'ютерних наук та інформаційних технологій Хмельницького національного університету, й розміщені в системі Модульного середовища для навчання ХНУ [180] і Навчального центру заочно-дистанційної освіти ХНУ [181]. Вибірка має наступні структурні параметри: кількість дисциплін – 40, кількість заголовків 1 та 2 рівня (іменованих одиниць структури, придатних для локальних пошуку ключових термінів та створення множин тестових завдань) – 203, загальна кількість тестових завдань у вибірці – 2421. Технічні параметри наведено на рис. 4.8. Тестова вибірка №1 використовується як вхідні данні при роботі ІТ, зокрема для формування структури ІНМ, пошуку ключових термінів та створення множин тестових завдань. При створенні правил продукції тестових завдань проводився аналіз створених авторами тестових завдань та відповідних ІНМ із цієї вибірки.

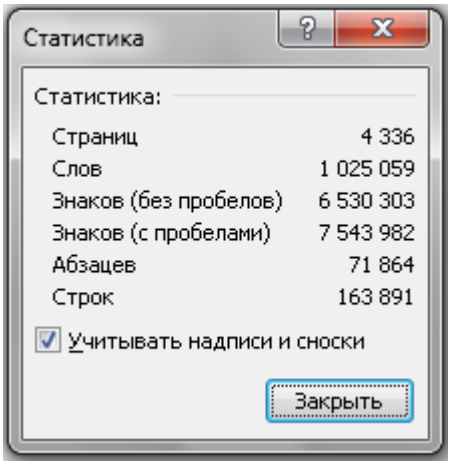


Рисунок 4.8 – Технічна параметри тестової вибірки №1 з ІНМ

Тестова вибірка №2 сформована з ІНМ, що знайдені у відкритому доступі й взяті із різних джерел з метою репрезентативності. Тестова вибірка №2 містить 1600 елементів ІНМ, кожен з яких містить визначену автором множину ключових слів. Вона призначена для дослідження ефективності пошуку ключових термінів запропонованим методом та для кількісних досліджень (максимальна кількість слів у терміні, кількість ключових термінів і ІНМ тощо). Приклад відомостей за даною вибіркою подано в додатку Г.

Далі наведено хід та результати досліджень.

4.2.2 Дослідження з вибору алгоритму пошуку ключових термінів

Умови дослідження. Було проведено аналіз вибірки ІНМ шляхом використання відомих методів для визначення ключових семантичних термінів і подальшого порівняння одержаних множин ключових термінів із множиною, створеною автором ІНМ. Згідно п. 1.3.2, для дослідження було використано методи: частотний аналіз, аналіз TFIDF та дисперсійний аналіз. Для проведення експериментів за наведеною вище схемою було розроблено тестове програмне забезпечення [4, 29], що реалізує обробку контенту ІНМ трьома розглянутими методами.

В процесі обробки контенту переліки ключових слів, отримані за відповідними методами, обмежуються за кількісним порогом й формують множини B_1, B_2, B_3. В подальшому ці множини порівнюються із множиною B_A, утвореною переліком ключових термінів, який сформовано автором навчального матеріалу. Перетин цих множин $B_k \cap B_A$ визначає ефективність відповідного методу *k*.

Максимальна область перетину авторського переліку зі сформованими застосунком переліками $B_k \cap B_A \rightarrow max$ визначає найбільш ефективний метод автоматизації пошуку ключових семантичних термінів у контенті навчальних матеріалів.

Ефективність наведених методів пропонується визначати за наступною формулою:

$$E_k = \frac{N_{Ak}}{N_A} \cdot 100\%, \tag{4.1}$$

де N_{Ak} – кількість термінів у авторському (B_A) та сформованому за *k*-им методом (B_k) переліками термінів, що співпали ($B_k \cap B_A$); N_A – кількість термінів у переліку термінів B_k, сформованому експертом (автором).

Результати дослідження. В результаті тестування (на прикладі ІНМ «Введення у реляційну модель даних» навчального курсу «Вступ до реляційних баз даних» [15]) розробленим програмним забезпеченням отримуються три переліки ключових термінів за відповідними методами аналізу та проводиться їх порівняння у сукупності з авторським переліком. Деякі результати порівняння наведено у Табл. 4.1.

На основі наведених даних дослідження, за формулою (4) побудовано діаграму ефективності розглянутих методів формування переліку ключових термінів у порівнянні з авторським переліком (рис. 4.9). Ефективність методу частотної оцінки склала 33,3%, методу оцінки TFIDF – 30,3%, методу дисперсної оцінки – 84,8%.

Аналогічним чином було досліджено 30 лекцій із ІНМ різних НК й обраховано середню ефективність кожного із методів. Середня ефективність

методу частотної оцінки склала 27,1%, методу оцінки TFIDF – 45,5% та методу дисперсійної оцінки – 88,3% (рис. 4.10).

Таблиця 4.1 – Фрагмент порівняльної таблиці аналізу термінів

№ п/п	Термін	Визначено автором	Частотний аналіз	Аналіз TFIDF	Аналіз дисперсії
1.	реляційна база даних	+	+		+
2.	тип даних	+	+	+	+
3.	домен	+		+	+
4.	реляційна модель даних	+			+
5.	обмеження цілісності	+		+	+
6.	заголовок відношення	+			+
7.	значення відношення	+	+		+
8.	перша нормальна форма	+			+
9.	модель даних	+	+		+
10.	СКБД	+	+		+
11.	реляційне числення	+			
12.	цілісність сутності	+		+	+
13.	булевий тип	+			+
14.	зовнішній ключ	+		+	+
15.	SQL	+			
16.	заголовок відношення				+
17.	унікальність значень			+	

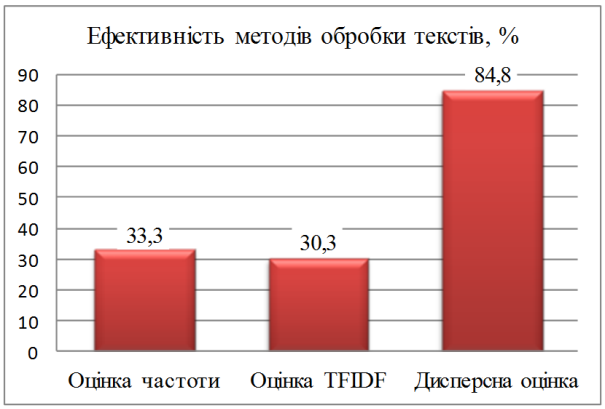


Рисунок 4.9 – Діаграма ефективності методів обробки текстів

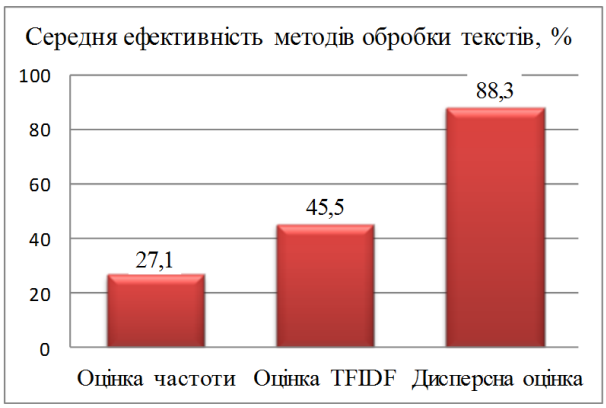


Рисунок 4.10 – Діаграма середньої ефективності методів обробки текстів

Таким чином, метод дисперсійної оцінки продемонстрував найвищу ефективність серед досліджуваних методів, показавши при цьому мінімальну

ефективність 67,7%, максимальну – 100%. Це обгрунтовує доцільність вибору методу дисперсійної оцінки для використання і методі формування структури навчальних матеріалів та пошуку в них ключових термінів

4.2.3 Дослідження повноти і точності пошуку ключових термінів

Для перевірки спроможності визначення множини ключових термінів у контенті ІНМ за розробленою ІТ, проведене порівняння знайдених ключових термінів із термінами запропонованими автором (експертом). Для дослідження було використано тестовий програмний продукт, описаний у п. 4.1.1. При дослідженні використано тестову вибірку №2, наведену вище.

Оцінювання результатів дослідження виконувалось за показниками точності (Precision) та повноти (Recall). Точність пошуку P (відношення кількості релевантних ключових термінів, знайдених автоматично, до загальної кількості знайдених ключових термінів в досліджуваному тексті) та повнота пошуку R (відношення кількості релевантних ключових термінів, знайдених автоматично, до загальної кількості релевантних ключових термінів в досліджуваному тексті) обчислюються наступним чином [79]:

$$P = \frac{\left|M_E \bigcap M_A\right|}{\left|M_A\right|},\ R = \frac{\left|M_E \bigcap M_A\right|}{\left|M_E\right|}, \tag{4.1}$$

де M_E – множина релевантних ключових термінів, сформована експертом; M_A – множина знайдених автоматично ключових термінів.

Відповідно, середня точність пошуку $\overline{P}$ та середня повнота пошуку $\overline{R}$ визначаються за наступними формулами:

$$\overline{P} = \frac{\sum_{i=1}^{k} P_k}{k},\ \overline{R} = \frac{\sum_{i=1}^{k} R_k}{k}, \tag{4.2}$$

де k – кількість документів навчальних матеріалів у тестовій вибірці.

При дослідженні до уваги беруться наступні підмножини M_E та M_A: множина спільних у M_E й M_A ключових термінів ($M_E \bigcap M_A$), множина

ключових термінів тільки в множині експерта ($M_E \setminus M_A$) та множина ключових термінів тільки в множині програми ($M_A \setminus M_E$). Для визначення співвідношень між цими множинами було визначено наступні показники:

$$D_S = \frac{|M_E \cap M_A|}{|M_E \cup M_A|},\ D_E = \frac{|M_E \setminus M_A|}{|M_E \cup M_A|},\ D_A = \frac{|M_A \setminus M_E|}{|M_E \cup M_A|}, \quad (4.3)$$

де D_S – спільна частка (відношення кількості спільних в двох множинах термінів до кількості унікальних термінів в об'єднаній множині); D_E – частка експерта (відношення кількості наявних тільки в множині експерта термінів до кількості унікальних термінів в об'єднаній множині); D_A – частка програми (відношення кількості наявних тільки в програмно визначеній множині термінів до кількості унікальних термінів в об'єднаній множині).

Кроки дослідження:

1) одержання множин ключових термінів для кожного з документів вибірки;

2) визначення показників точності та повноти пошуку ключових термінів для кожного з документів вибірки;

3) визначення середніх і граничних показників точності та повноти пошуку ключових термінів;

4) інтерпретація результатів дослідження.

Для описаного в п. 4.2.2 (див. табл. 4.1) прикладу обробки теми «Нейронні мережі когнітрон та неокогнітрон» дисципліни «Методи та системи штучного інтелекту» за показника щільності ключових слів Q = 7% до множини ключових термінів було віднесено наступні: когнітрон, неокогнітрон, нейрон, комплексний вузол, простий вузол, образ, вхідний образ. Результати порівняння одержаної множини ключових термінів з авторською множиною наведено у табл. 4.2. Відповідно до (4.1), у даному випадку точність пошуку склала 0,625, а повнота пошуку склала 0,714.

Загалом, для тестової вибірки №2, відповідно до (4.1) та (4.2) середня точність пошуку склала $\overline{P}$ = 0,732, середня повнота пошуку склала $\overline{R}$ = 0,697. Мінімальна точність пошуку за вибіркою склала P_{min} = 0,512, мінімальна

повнота пошуку R_{min} = 0,581; максимальна точність пошуку P_{max} = 0,929, максимальна повнота пошуку R_{max} = 1,000. Візуально результати зображено на рис. 4.11.

Таблиця 4.2 – Порівняльна таблиця аналізу множин термінів

№ п/п	Ключовий термін	Визначено автором	Визначено автоматично
1.	когнітрон	+	+
2.	неокогнітрон	+	+
3.	нейрон	+	+
4.	збуджуючий нейрон	+	
5.	гальмуючий нейрон	+	
6.	комплексний вузол	+	+
7.	простий вузол	+	+
8.	образ		+
9.	вхідний образ		+
10.	навчання		+

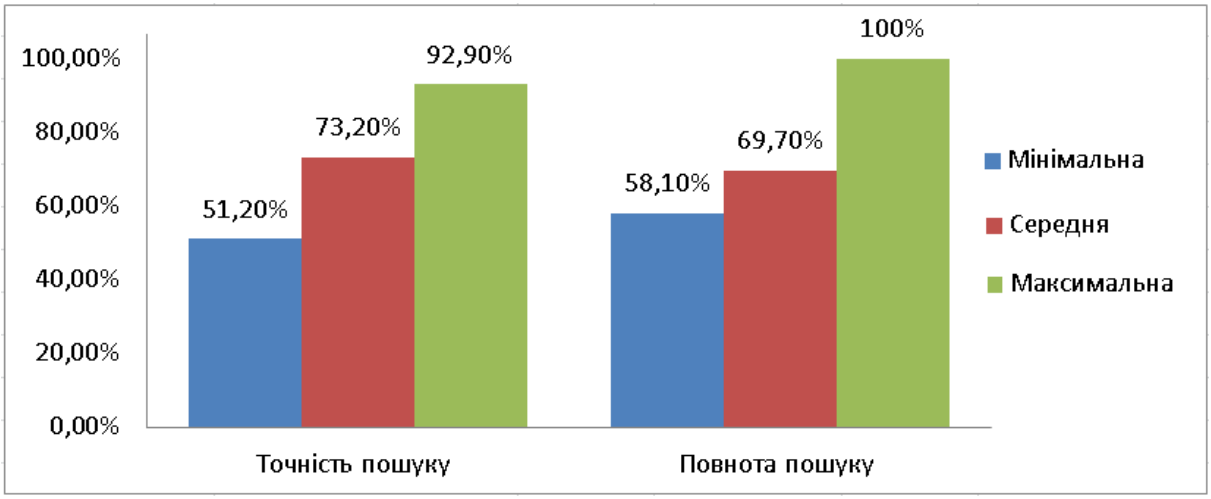


Рисунок 4.11 – Результати обрахунку точності та повноти пошуку ключових термінів для тестової вибірки №2

Середні показники для D_S, D_E та D_A, обраховані за (4.3) та усереднені аналогічно до (4.2) для тестової вибірки №2, становлять: $\overline{D_S}$ = 60,10%, $\overline{D_A}$ = 13,79%, $\overline{D_E}$ = 26,11%, що візуально відображено на рис. 4.12.

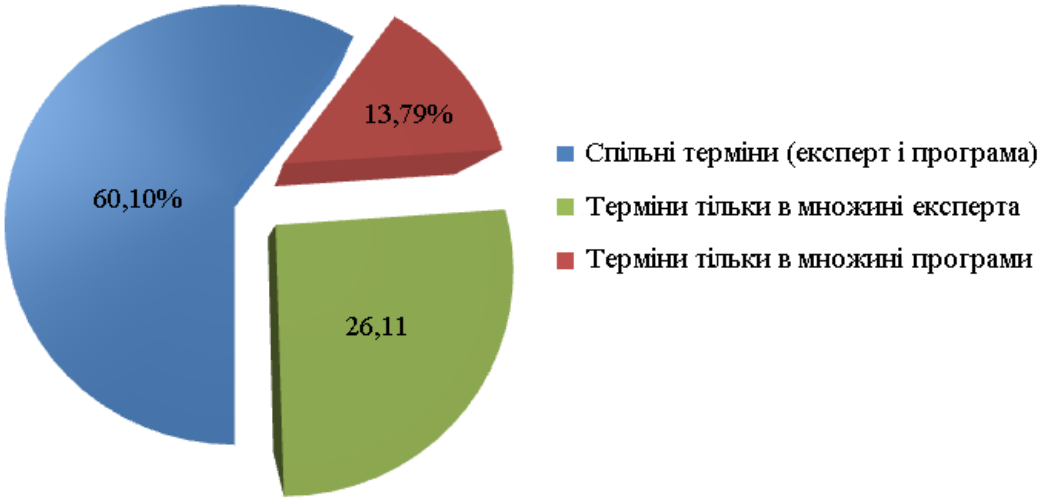


Рисунок 4.12 – Результати аналізу входжень ключових термінів до результуючих множин для тестової вибірки №2

Аналіз одержаних результатів виявив, що відсутність програмно визначених термінів у множині автора не завжди характеризує недолік запропонованої технології. Деякі семантично важливі терміни автори суб'єктивно ігнорують, в той час як іншу категорію складають терміни, на яких автори акцентують надмірну увагу попри їх другорядність в рамках матеріалу, що викладається. Тому крім обрахунку P та R доцільно розглядати не тільки терміни із множини експерта, що не були знайдені програмно, а й автоматично знайдені терміни, які не увійшли до множини експерта, шляхом обрахунку $\overline{D_S}$, $\overline{D_A}$ та $\overline{D_E}$.

Низькі значення показника точності пошуку (до P = 51,2%) спостерігались переважно внаслідок одержання множини ключових термінів більшої потужності за потужність множини ключових термінів авторів ІНМ. Це є результатом більш глибокого семантичного аналізу ІНМ тестовою програмною системою або визначення авторами ІНМ замалої для наявного обсягу контенту кількості ключових термінів. Зменшення параметру щільності ключових термінів у тексті Q надає можливість одержувати множини ключових термінів меншої потужності й таким чином збільшити точність пошуку ключових термінів у таких зразках ІНМ.

Фактори, що спричинили одержання низьких значень показника повноти пошуку (до R = 58,1%), визначено як програмні (збільшення параметру щільності ключових термінів у тексті Q дозволяє включити більшу кількість ключових термінів до результуючої множини), так і суб'єктивні. Аналіз результатів виявив наступні причини, що призвели до формування авторами ІНМ некоректних множин ключових термінів:

– включення до множини недостатньої або надмірної кількості ключових термінів для наявного обсягу контенту ІНМ;

– використання ідентичної множини ключових термінів як для всього ІНМ, так і для кожного з його складових;

– включення до множини ключових термінів обраних фрагментів заголовків без використання цих термінів у контенті;

– багаторазове використання в контенті ІНМ кількох синонімічних назв для одного терміну («СКБД», «система керування БД», «система керування базами даних»), що некоректно для навчальних матеріалів;

– включення до множини таких термінів, що мали бути розглянуті в ІНМ, але це не було зроблено;

– інші суб'єктивні фактори, наприклад визначення виключно термінів, що складаються з одного слова (база» і «дані» замість «база даних») або узагальнюючих для кількох термінів назв («основні поняття», «різновиди класів»).

Досягнення максимальних одержаних значень показників точності пошуку (до P = 92,9%) та повноти пошуку (до R = 100%) стало можливим у випадках, коли автори ІНМ змогли уникнути вищезгаданих факторів та коректно визначали необхідну кількість ключових термінів у множині. Окремо слід відзначити використання авторами SEO-систем, що забезпечує формування технічно коректних множин ключових термінів.

Згадана можливість коригування параметру щільності ключових термінів у тексті P, передбачена в розробленій інформаційній технології, надає інструмент для впливу на оціночні параметри точності пошуку та

повноти пошуку, а саме: при збільшенні параметру щільності *Q* збільшується повнота пошуку *R*, проте зменшується точність пошуку *P*. В дослідженні було використано загальне для всіх документів вибірки значення *Q* = 7%; підбір окремого значення для кожного документу дозволить покращити середні показники $\overline{P}$ та $\overline{R}$, проте зменшить коректність експерименту.

Як слідує з наведеного вище, збільшення значення *Q* збільшує $\overline{D_A}$, а зменшення *Q* збільшує $\overline{D_E}$. За оптимального для вибірки значення *Q* = 7% одержано $\overline{D_S}$ = 60,10%, при цьому частка некоректно визначених ключових термінів склала $\overline{D_A}$ = 13,79%, що свідчать про можливість використання розробленої інформаційної технології для одержання валідних множин ключових термінів, релевантних до множин ключових термінів авторів ІНМ.

Таким чином, одержані результати дослідження свідчать, що розроблена ІТ надає можливість для одержання множин ключових термінів, релевантних до множин ключових термінів авторів ІНМ. Встановлені відхилення точності пошуку пояснюються людським фактором, а відхилення повноти пошуку можуть бути відкориговані передбаченими в ІТ засобами. Це надає підставу розглядати автоматизовано одержані за створеною ІТ множини ключових термінів як рівень семантичної структури ІНМ, що коректно передає сенс навчального матеріалу та потребує створення множин тестових завдань для перевірки рівня засвоєння цих термінів.

4.2.4 Дослідження повноти покриття контенту ключовими термінами

Дане дослідження проводилось з метою визначити частку контенту ІНМ, що містить включені до результуючої множини ключових термінів, і відповідно може бути використана для створення тестових завдань розробленою ІТ. За матеріал дослідження було використано розділи навчальних дисциплін тестової вибірки №1, параметри якої наведено в

п. 4.2.1. Для дослідження було використано тестовий програмний продукт, описаний у п. 4.1.1.

За одиницю виміру повноти покриття контенту ІНМ ключовими термінами взято відношення K_H, K_P і K_S кількості семантичних блоків (відповідно підрубрик N_H, абзаців N_P і речень N_S), що містять знайдені ключові терміни, до загальної кількості таких блоків (відповідно N_{HH}, N_{PP} і N_{SS}) у контенті кожного ІНМ:

$$K_H = \frac{N_H}{N_{HH}},\ K_P = \frac{N_P}{N_{PP}},\ K_S = \frac{N_S}{N_{SS}}, \tag{4.4}$$

$$\overline{K_H} = \frac{\sum_{i=1}^{h} K_{H,i}}{h},\ \overline{K_P} = \frac{\sum_{i=1}^{h} K_{P,i}}{h},\ \overline{K_S} = \frac{\sum_{i=1}^{h} K_{S,i}}{h}, \tag{4.5}$$

де $\overline{K_H}$, $\overline{K_P}$ та $\overline{K_S}$ – середні значення для відповідно K_H, K_P та K_S для тестової вибірки; h – кількість оброблюваних документів ІНМ у вибірці.

З метою дослідження характеру семантичних блоків, що розглядаються, було використано похідні показники: L_H, L_P та L_S для відношення кількості слів у семантичних блоках (відповідно підрубрик, абзаців і речень), що містять знайдені ключові терміни, до загальної кількості слів у ІНМ F_W. Значення показників L_H, L_P та L_S й їх середні значення $\overline{L_H}$, $\overline{L_P}$ та $\overline{L_S}$ за тестовою вибіркою обраховуються наступним чином:

$$L_H = \frac{F_H}{F_W},\ L_P = \frac{F_P}{F_W},\ L_S = \frac{F_S}{F_W}, \tag{4.6}$$

$$\overline{L_H} = \frac{\sum_{i=1}^{h} L_{H,i}}{h},\ \overline{L_P} = \frac{\sum_{i=1}^{h} L_{P,i}}{h},\ \overline{L_S} = \frac{\sum_{i=1}^{h} L_{S,i}}{h}, \tag{4.7}$$

де F_H, F_P та F_S – кількість слів відповідно у підрубриках, абзацах та реченнях, в яких знайдені ключові терміни.

Наприклад, у результаті обробки розділу «Моделювання даних та нормалізація» дисципліни «Організація баз даних та знань» з значеннями максимальної кількості слів у терміні n = 5 та граничної щільності ключових

термінів Q = 11%, одержано множину ключових термінів, наведену у табл. 4.2, до якої включено наступні унікальні терміни: база даних, таблиця, первинний ключ, запит, нормальна форма, нормалізація, відношення, кортеж, дані.

В даному випадку показники покриття контенту ключовими термінами за (4.4) склали: K_H = 100%, K_P = 73,3%, K_S = 76,1%. При цьому, відповідні показники покриття контенту за кількістю слів відповідно до (4.6) одержано наступні: L_H = 100%, L_P = 79,8%, L_S = 85,9%.

Таблиця 4.2 – Приклад результату пошуку ключових термінів

№ п/п	**Ключовий термін *T***	**Кількість у тексті *k***	**Частота *TF***	**Дисперсійна оцінка *σ***
1	бази даних	64	0,0169	3,3720
2	таблиця	73	0,0175	2,9294
3	первинного ключа	55	0,0162	3,3720
4	запитом	47	0,0110	2,9294
5	нормальна форма	49	0,0165	2,6444
6	нормалізації	41	0,0165	2,5320
7	відношення	34	0,0110	2,3817
8	кортеж	40	0,0137	2,2732
9	баз даних	67	0,0220	2,1460
10	нормальної форми	34	0,0302	1,9918
11	запит	52	0,0302	1,9917
12	даних	79	0,0302	1,9915

Загалом по тестовій вибірці №2 з h = 203 документів ІНМ за (4.5) одержано середні показники покриття контенту $\overline{K_H}$ = 100%, $\overline{K_P}$ = 82,5% та $\overline{K_S}$ = 89,1%. Граничні значення цих показників склали: $K_{H,min}$ = 100%, $K_{H,max}$ = 100%, $K_{P,min}$ = 60,1%, $K_{P,max}$ = 100%, $K_{S,min}$ = 66,5%, $K_{S,max}$ = 100%. Аналогічні середні показники покриття контенту за кількістю слів відповідно до (4.7) одержано $\overline{L_H}$ = 100%, $\overline{L_P}$ = 88,1% та $\overline{L_S}$ = 91,4%, при наступних граничних значеннях: $L_{H,min}$ = 100%, $L_{H,max}$ = 100%, $L_{P,min}$ = 67,6%, $L_{P,max}$ = 100%, $L_{S,min}$ = 69,7%, $L_{S,max}$ = 100%. Візуальне подання результатів аналізу покриття контенту ключовими термінами зображено на рис. 4.13.

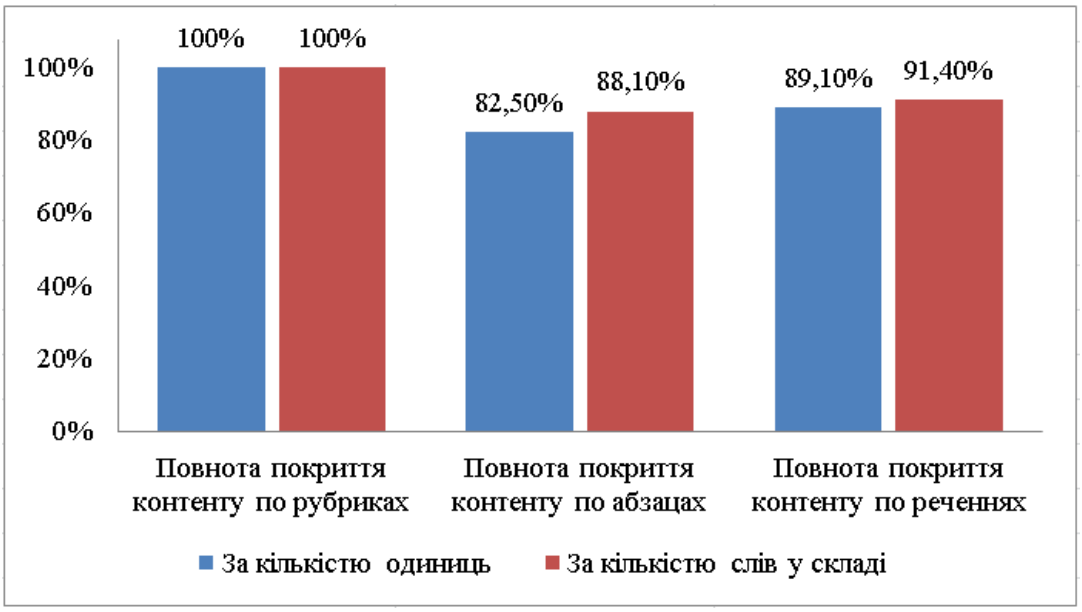


Рисунок 4.13 – Середні значення покриття контенту ключовими термінами в складі рубрик, абзаців та речень

Таким чином, за значення Q = 11% ключові терміни з автоматизовано одержаної множини містяться в усіх підрубриках ІНМ, зокрема в середньому в 82,5% абзаців (що містять 88,1% текстового контенту), або в 89,1% речень (що містять 91,4% текстового контенту). Збільшення значення параметру граничної щільності ключових термінів Q дозволить збільшити наведені значення, проте призведе до включення до результуючої множини маловажливих та похідних термінів, перевіряти знання яких за допомогою тестів не потрібно.

Оскільки одержано $\overline{K_S} > \overline{K_P}$, то можна зробити висновок, що ключові терміни не присутні переважно в абзацах, що складаються з невеликої кількості речень. Наприклад, «Далі наведено основні властивості цих класів.». Результат $\overline{L_S} > \overline{K_S}$ свідчить, що ключові терміни не присутні переважно в коротких реченнях. Ці факти пояснюються загальновідомими синтаксичними особливостями побудови текстів, коли деякі речення (а іноді й абзаци) мають не семантичну, а зв'язувальну важливість.

Суттєве зниження показників повноти покриття контенту ($K_{P,min}$ = 60,1% та $K_{S,min}$ = 66,5%, $L_{P,min}$ = 67,6% та $L_{S,min}$ = 69,7%) спричиняє поява

додаткових суб'єктивних факторів, які характеризуються включенням спеціальних фрагментів контенту, зокрема програмних кодів та практичних прикладів. Наявність таких включень характерна для операційного навчального матеріалу (методичні вказівки, практикуми тощо), а для перевірки одержаних знань та вмінь потребує створення тестів за допомогою параметричного методу. Включення цих елементів до ІНМ хоча й не забороняється, проте не потребує тестової перевірки рівня засвоєння.

4.2.5 Дослідження повноти та рівномірності покриття контенту тестовими завданнями

Дослідження проводилось з метою визначити частку контенту ІНМ, яка використовується для створення тестових завдань, і відповідно рівень знання якої може бути перевірений створеним за допомогою запропонованої інформаційної технології тестом. Як і в п. 4.2.4, за матеріал дослідження було використано розділи навчальних дисциплін тестової вибірки №1. Для дослідження було використано призначений для автоматизованого формування тестових завдань тестовий програмний продукт, описаний у п. 4.1.1.

В п. 4.2.4 було досліджено повноту покриття контенту ІНМ ключовими термінами шляхом обрахунку відношення кількості семантичних блоків (підрубрик K_H, абзаців K_P і речень K_S), що містять знайдені ключові терміни, до загальної кількості таких блоків у контенті кожного ІНМ. Для дослідження повноти та рівномірності покриття контенту створеними тестовими завданнями використано аналогічні до (4.4) і (4.5) параметри KT_H, KT_P і KT_S, а також KT_O – відношення кількості використаних для створення тестових завдань появ ключових термінів до загальної кількості появ термінів у ІНМ:

$$KT_H = \frac{NT_H}{N_{HH}},\ KT_P = \frac{NT_P}{N_{PP}},\ KT_S = \frac{NT_S}{N_{SS}},\ KT_O = \frac{NT_O}{N_{OO}}, \qquad (4.8)$$

де NT_H, NT_P, NT_S і NT_O – кількості семантичних блоків (відповідно підрубрик, абзаців, речень і появ), що використані для створення тестових завдань; N_{HH}, N_{PP}, N_{SS} і N_{OO} – загальна кількість відповідних блоків у контенті кожного ІНМ.

Середні значення для тестової вибірки $\overline{KT_H}$, $\overline{KT_P}$, $\overline{KT_S}$ та $\overline{KT_O}$ для відповідних параметрів KT_H, KT_P, KT_S та KT_O обраховуються:

$$\overline{KT_H} = \frac{\sum_{i=1}^{h} KT_{H,i}}{h}, \quad \overline{KT_P} = \frac{\sum_{i=1}^{h} KT_{P,i}}{h}, \quad \overline{KT_S} = \frac{\sum_{i=1}^{h} KT_{S,i}}{h}, \quad \overline{KT_O} = \frac{\sum_{i=1}^{h} KT_{O,i}}{h}, \qquad (4.9)$$

де h – кількість оброблюваних документів ІНМ у вибірці.

При цьому, при використанні певного семантичного блоку до загальної кількості одиниця додається тільки один раз, незалежно від кількості створених за ним тестових завдань чи знайдених у ньому ключових термінів.

Для оцінки ж *рівномірності покриття контенту ІНМ* створеними тестовими завданнями, окрім граничних показників для NT_H, NT_P, NT_S і NT_O, було визначено кількості створених тестових завдань різних типів до кожної появи ключових термінів.

Поява ключового терміну в контенті ІНМ є найменшим семантичним блоком, який слугує маркером для створення тестових завдань. Вона формує потребу в розумінні учнем даного терміну для розуміння сенсу всього фрагменту (зазвичай речення) ІНМ, й таким чином визначає можливість використання цього фрагменту для перевірки розуміння терміну. Більші фрагменти (підрубрики, абзаци, речення) можуть містити довільну (більшу одиниці) кількість появ різних ключових термінів, тому обрахунок кількості створених тестових завдань для таких фрагментів не проводився.

Середня кількість використаних правил продукції тестових завдань певного типу (або сукупна), що були виконані до появ ключових термінів у контенті ІНМ, обраховується:

$$\overline{KO_u} = \frac{\sum_{i=1}^{h}\sum_{j=1}^{t}\sum_{v=1}^{o} KO_{u,i,j,v}}{h \cdot t \cdot o}, \qquad (4.10)$$

де h – кількість оброблюваних документів ІНМ у вибірці; t – кількість ключових термінів у множині для i-го документу ІНМ; o – кількість появ j-го ключового терміну i-го документу ІНМ; $KO_{u,i,j,v}$ – кількість створених тестових завдань u-го типу (або сукупна) для v-ї появи j-го ключового терміну i-го документу ІНМ.

Кроки дослідження:

1) визначення кількостей семантичних блоків (підрубрик, абзаців, речень і появ) для документів ІНМ тестової вибірки;

2) одержання множин тестових завдань для документів ІНМ тестової вибірки;

3) визначення кількостей використаних правил продукції тестових завдань кожного типу для кожної появи кожного ключового терміну в кожному документі ІНМ тестової вибірки;

4) визначення відношення кількості використаних для створення тестових завдань семантичних блоків до загальної кількості семантичних блоків у кожному документі ІНМ тестової вибірки;

5) визначення середніх і граничних показників для відношення кількості використаних для створення тестових завдань семантичних блоків до загальної кількості семантичних блоків по тестовій вибірці;

6) визначення середньої кількості використаних правил продукції тестових завдань кожного типу та загалом до появ ключових термінів у контенті ІНМ по тестовій вибірці;

7) інтерпретація результатів дослідження.

Одержано наступні результати дослідження.

Середні значення відношення кількості використаних для створення тестових завдань семантичних блоків до загальної кількості семантичних блоків по тестовій вибірці №2 з h = 203 документів ІНМ за (4.8) і (4.9) одержано наступні: $\overline{KT_H}$ = 100%, $\overline{KT_P}$ = 82,6%, $\overline{KT_S}$ = 91,2% та $\overline{KT_O}$ = 97,8%. Граничні значення цих показників склали: $KT_{H,min}$ = 100%, $KT_{H,max}$ = 100%, $KT_{P,min}$ = 60,3%, $KT_{P,max}$ = 100%, $KT_{S,min}$ = 66,9%, $KT_{S,max}$ = 100%, $KT_{O,min}$ = 95,3%,

$KT_{O,max}$ = 100%. Візуальне подання результатів аналізу повноти покриття контенту тестовими завданнями зображено на рис. 4.14.

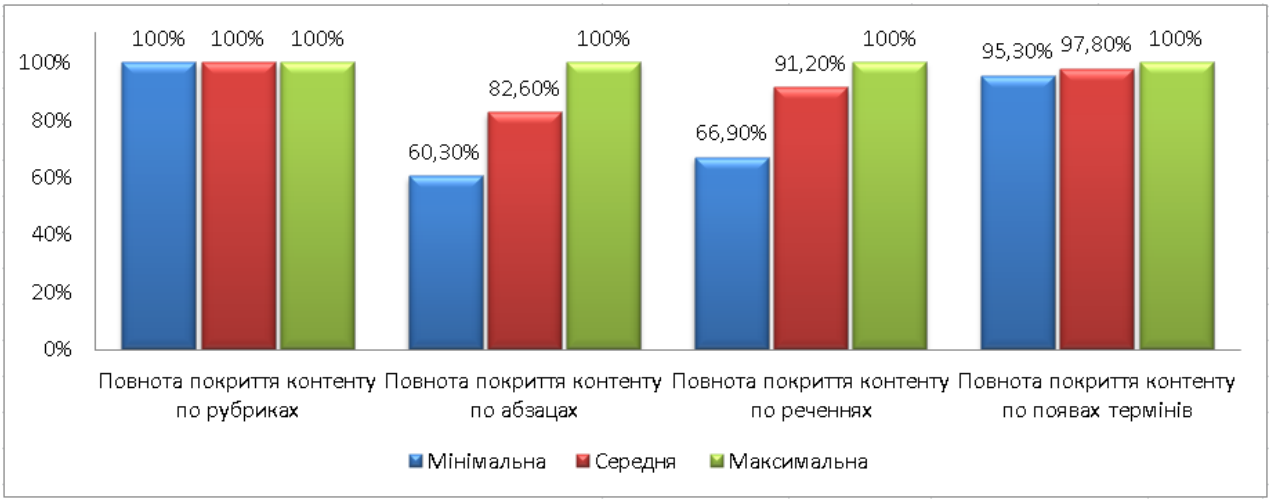


Рисунок 4.14 – Середні та граничні значення показників KT_H, KT_P, KT_S і KT_O повноти покриття контенту тестовими завданнями по тестовій вибірці №2

Обрахунок середньої кількості використаних правил продукції тестових завдань по типах та загалом до поява ключових термінів у контенті ІНМ $\overline{KO_u}$ по тестовій вибірці №2 з h = 203 документів за (4.10) дав результати наведені в табл. 4.3. До обрахунку бралися $\overline{KT_O}$ = 97,8% випадків, коли поява терміну актуалізувала хоча б один антецедент правил продукції.

Таблиця 4.3 – Кількість створених тестових завдань (у дужках наведено кількість використаних правил продукції) по типах та загалом до поява ключових термінів у контенті ІНМ по тестовій вибірці №2

№ п/п	Тип тестового завдання	Кількість створених тестових завдань, шт.		
		Мінімальна	Середня	Максимальна
1	Логічного типу	2 (2)	2,23 (2,28)	8 (3)
2	Одиничного вибору	2 (2)	3,19 (2,61)	17 (3)
3	Множинного вибору	1 (1)	1,95 (1)	5 (1)
4	Із введенням тексту	1 (1)	1,06 (1)	3 (1)
5	Загалом на одну появу	6 (6)	8,43 (6,87)	14 (8)

Таким чином, показник $\overline{KT_O}$ = 97,8% свідчить, що в 97,8% випадків поява ключового терміну в контенті ІНМ призводить до створення як мінімум одного тестового завдання. Це пояснюється тим, що ряд консеквентів (c_{a1}, c_{a2}, c_{b3}, c_{b4}, c_{c1}, c_{d2} – див. п. 2.4.2) дають можливість створення тестових завдань за наявним контентом без суттєвих обмежень за структурою речення чи синтаксисом. Решту 2,2% складає поява ключових термінів у реченнях, в яких відсутній мінімальний набір елементів контенту його для ідентифікації хоча б одним зразком антецедента правила (наприклад, «Далі буде розглянуто основні властивості *наслідування*.»), при одночасній неможливості включення до обробки суміжних фрагментів контенту (наприклад, якщо це окремий абзац і наступний абзац не є переліком).

Слід зауважити, що всі розглянуті появи ключових термінів зосереджені в середньому в $\overline{K_P}$ = 82,5% абзаців, або $\overline{K_S}$ = 89,1% речень. Різниця $\overline{KT_P} > \overline{K_P}$ пояснюється тим, що при побудові деяких тестових завдань (зокрема, для консеквенту c_{c2}) може використовуватись також контент прилеглих абзаців. Різниця $\overline{KT_S} > \overline{K_S}$ одержується в звязку з тим, що при побудові ряду тестових завдань за появи ключового терміну в реченні, яке ідентифікується як надкоротке означальне, може використовуватись також контент попереднього речення (наприклад, «Можливий випадок, коли фізично віддалені бази даних віртуально використовуються як єдине ціле. Такий випадок називається *розподіленою базою даних*.»).

Аналіз кількості використаних правил продукції для створення тестових завдань по їх типах свідчить, що до кожної появи (в $\overline{KT_O}$ = 97,8% випадків) ключового терміну в контенті створюється мінімум одне тестове завдання кожного типу. Таким чином, завдяки поєднанню в правилах продукції однакових антецедентів та різних консеквентів (див. табл. 2.4) досягається мінімально необхідна рівномірність розподілу тестових завдань за типами та використаними ключовими термінами.

4.2.6 Дослідження зменшення часу на формування множини тестових завдань

Дослідження проводилось з метою кількісного визначити зменшення часу, необхідного на формування множини тестових завдань шляхом визначення різниці між часом, необхідним на цю роботу за її ручного виконання, та часом, необхідним на досягнення аналогічного результату за використання розробленої ІТ та подальшого ручного коригування автоматизовано створеної множини тестових завдань. За матеріал дослідження було використано розділи навчальних дисциплін з тестової вибірки №1 (загалом 40 зразків). Метою встановлено розробку однакової кількості тестових завдань для кожного з зразків ІНМ (по 40-100 завдань). Для автоматизованого формування тестових завдань було використано програмний продукт, описаний у п. 4.1.1. Для ручної розробки та коригування множини тестових завдань було використано середовище Moodle на базі сайту Хмельницького національного університету. Суб'єктами роботи з розробки та коригування тестів були використані викладачі кафедри Комп'ютерних наук та інформаційних технологій Хмельницького національного університету (загалом 5 осіб).

Ефект зменшення часу на формування множини тестових завдань *TT* до *i*-го елементу ІНМ з вибірки обраховується за формулою:

$$\Delta TT_i = \frac{TR_i - TP_i}{TR_i} \cdot 100, \quad \overline{\Delta TT} = \frac{\sum_{i=1}^{n} \Delta TT_i}{n}, \tag{4.11}$$

де TR_i – час, затрачений на ручне створення множини тестових завдань до *i*-го елементу ІНМ з вибірки; TP_i – час, затрачений на створення множини тестових завдань з використанням програмного продукту до *i*-го елементу ІНМ з вибірки; $\overline{\Delta TT}$ – середнє значення для вибірки з *n* елементів ІНМ.

При ручному створенні множини тестових завдань з використанням редактора тестів середовища Moodle, до результату *TR* слід враховувати час, який витрачається розробником на розробку, перегляд сформованої множини тестових завдань, повторне коригування й інші дії, які мають наслідком формування множини тестових завдань. При створенні множини тестових завдань з використанням розробленого за ІТ програмного продукту, витрачений на досягнення результату час *TP*, містить складові: час, витрачений програмним продуктом на обробку вхідних даних, формування множини тестових завдань, їх збереження та завантаження в середовище Moodle; час, витрачений викладачем на ознайомлення з сформованими тестовими завданнями, видалення некоректних та надлишкових зразків; час, витрачений викладачем на коригування деяких тестових завдань.

Окремо в ході дослідження до відсотку загальної кількості автоматизовано створених за інформаційною технологією тестових завдань *NT* було визначено відсотки кількості тестових завдань за подальшими діями розробника тесту: $NT = NTG + NTC + NTB = 100$ %, де *NTG* – відсоток кількості тестових завдань, що були включені до результуючої вибірки без змін; *NTC* – відсоток кількості тестових завдань, що були включені до результуючої вибірки й зазнали ручного коригування; *NTB* – відсоток кількості тестових завдань, що були видалені як некоректні чи надлишкові.

Середні значення для *NTG*, *NTC* та *NTB* обраховуються:

$$\overline{NTG} = \frac{\sum_{i=1}^{n} NTG_i}{n}, \quad \overline{NTC} = \frac{\sum_{i=1}^{n} NTC_i}{n}, \quad \overline{NTB} = \frac{\sum_{i=1}^{n} NTB_i}{n}. \qquad (4.12)$$

Одержано наступні результати дослідження. Середнє значення ефекту зменшення часу на формування множини тестових завдань за (4.11) склало $\overline{\Delta TT} = 60{,}25\%$, при цьому мінімальне значення було одержано $TT_{min} = 26{,}91\%$, максимальне – $TT_{max} = 74{,}53\%$. Візуальне подання результатів аналізу повноти покриття контенту тестовими завданнями зображено на рис. 4.15.

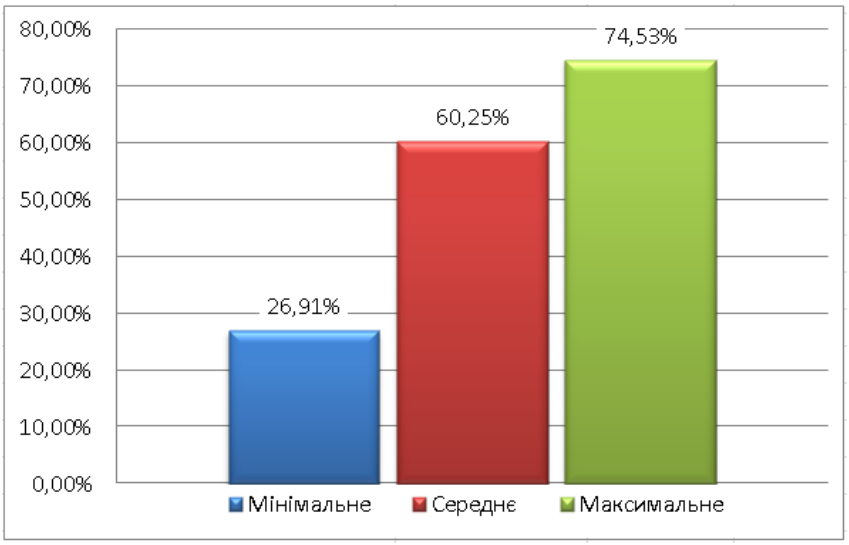


Рисунок 4.15 – Середнє та граничні значення показника зменшення часу на формування множини тестових завдань

Середні значення відсотків кількості тестових завдань за подальшими діями розробника тесту склали: $\overline{NTG}$ = 23,65%, $\overline{NTC}$ = 27,14%, $\overline{NTB}$ = 49,21% (рис. 4.16).

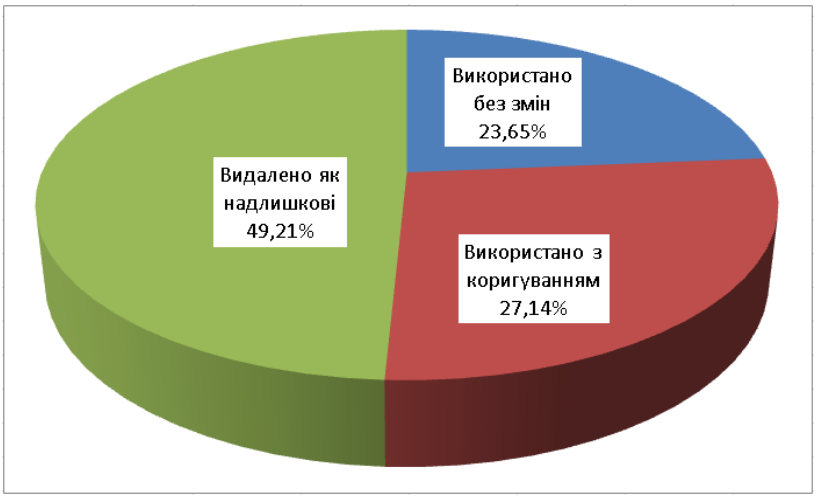


Рисунок 4.16 – Середні значення відсотків кількості тестових завдань за подальшими діями розробника тесту

Середнє значення ефекту зменшення часу на формування множини тестових завдань $\overline{\Delta TT}$ = 60,25% свідчить, що формування множини тестових завдань за використання розробленої ІТ та подальшого ручного коригування

автоматизовано створеної множини тестових завдань дозволяє більш швидко досягти мети, ніж досягнення аналогічного результату за ручного формування множини тестових завдань. Мінімальне значення TT_{min} = 26,91% вказує, що в деяких випадках викладачі відчувають складнощі при подальшій роботі з автоматизовано створеною множиною тестових завдань. Це пояснюється як суб'єктивним фактором, так і різною швидкістю адаптації до автоматизації створення тестових завдань у різних суб'єктів. Втім, оскільки $TT_{min} < 0$, використання розробленої ІТ та подальшого ручного коригування автоматизовано створеної множини тестових завдань дозволяє завжди досягти поставленої мети за менший час.

Наявний відсоток кількості видалених тестових завдань $\overline{NTB}$ = 49,21% пояснюється переважно надлишковою кількістю створених автоматизовано тестових завдань, аніж їх некоректністю, оскільки в жодному випадку викладачеві не знадобилося створення нових тестових завдань. Коригування, здійснене до ряду автоматизовано створених тестових завдань $\overline{NTC}$ = 27,14%, стосувалося переважно синтаксичного узгодження елементів тестового завдання та редагування дистракторів. При цьому, значна кількість тестових завдань $\overline{NTG}$ = 23,65%, що складає 46,56% від загальної кількості завдань у результуючій множині, була використана без коригування та змін.

4.2.7 Дослідження використання створених тестових завдань для адаптивного тестування

Для дослідження було використано призначену для проведення адаптивного тестування тестову програмну систему, описану в п. 4.1.2. За матеріал дослідження було використано множини тестових завдань, одержані за допомогою призначеного для автоматизованого формування тестових завдань тестового програмного продукту, описаного у п. 4.1.1, шляхом

обробки вхідних даних у вигляді ряду розділів навчальних дисциплін із тестової вибірки №1.

Використовуючи ідентичну множину тестових завдань, було запропоновано піддослідним (студентам за власним бажанням) пройти тестування з використанням базового алгоритму проходження тесту середовища Moodle та з використанням адаптивного алгоритму проходження тесту у середовищі Moodle. Достатня кількість тестових завдань за традиційного алгоритму визначається шляхом їх випадкового додавання до тесту до вичерпання вказаного граничного часу. Достатня кількість тестових завдань за адаптивного алгоритму визначається шляхом одержання кінцевого результату по рівню знань за кожним із заголовків ІНМ.

Дослідження ефекту від використання створених множин тестових завдань для адаптивного тестування виконано шляхом оцінки різниці часу на проходження тесту та кількості тестових завдань, використаних для тестування, за наведеними двома алгоритмами, як для кожної категорії результату (за національною шкалою), так і в середньому.

Ефект від використання адаптивного тестування за різницею в часі тестування обраховується за формулою:

$$\Delta T_i = \frac{TC_i - TA_i}{TC_i} \cdot 100, \quad \overline{\Delta T_i} = \frac{\sum_{j=1}^{n} \Delta T_{i,j}}{n}, \tag{4.13}$$

де TC_i – час, затрачений на тестування за традиційного алгоритму i-м досліджуваним; TA_i – час, затрачений на тестування за адаптивного алгоритму i-м досліджуваним; $\overline{\Delta T_i}$ – середнє значення для вибірки з n проходжень тесту двома алгоритмами.

Ефект від використання адаптивного тестування за різницею в кількості одержаних на тестуванні тестових завдань обраховується так:

$$\Delta N_i = \frac{NC_i - NA_i}{NC_i} \cdot 100, \quad \overline{\Delta N_i} = \frac{\sum_{j=1}^{n} \Delta N_{i,j}}{n}, \tag{4.14}$$

де NC_i – кількість завдань, одержана на тестування за традиційного алгоритму i-м досліджуваним; NA_i – кількість завдань, одержана на тестування за адаптивного алгоритму i-м досліджуваним; $\overline{\Delta N_i}$ – середнє значення для вибірки з n проходжень тесту двома алгоритмами.

Кроки дослідження:

1) автоматизоване створення множини тестових завдань для проходження тестування;

2) проходження тестування студентами за традиційного алгоритму;

3) проходження тестування студентами за адаптивного алгоритму;

4) визначення різниці в часі тестування й різниці кількості одержаних тестових завдань для кожного піддослідного;

5) визначення середніх і граничних показників для різниці в часі тестування й різниці кількості одержаних тестових завдань по тестовій вибірці;

6) інтерпретація результатів дослідження.

На етапі підготовки тесту було виконано автоматизоване створення множини тестових завдань для проходження тестування для навчального курсу «Організація баз даних та знань». Нижче наведено приклади створених тестових завдань для кожного з типів.

Так, на основі фрагменту «*Зріз (Slice) представляє собою підмножину гіперкуба, отриману в результаті фіксації одного або декількох вимірів*.» сформоване тестове завдання логічного типу до терміну «Зріз» (рис. 4.17).

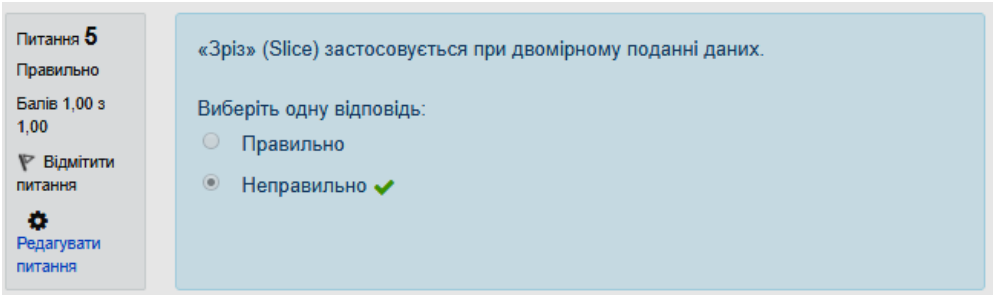


Рисунок 4.17 – Приклад тестового завдання логічного типу у Moodle

На основі фрагменту контенту «*Основною перевагою багатовимірної моделі даних є зручність і ефективність аналітичної обробки великих обсягів даних, пов'язаних з часом.*» сформоване тестове завдання одиничного вибору до терміну «багатовимірна модель даних» (рис. 4.18).

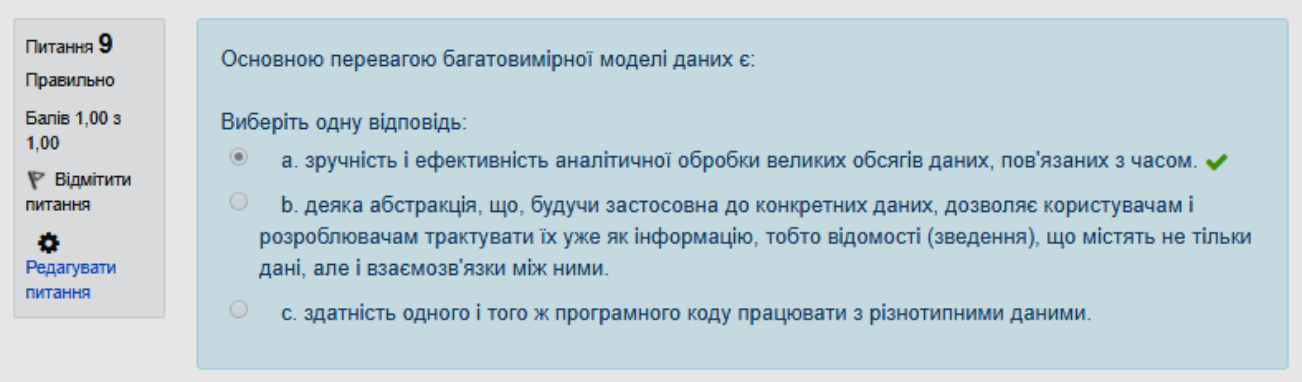


Рисунок 4.18 – Приклад тестового завдання одиничного вибору у Moodle

На основі фрагменту контенту «*До числа класичних відносяться наступні моделі даних: ієрархічна; мережева; реляційна.*» сформоване тестове завдання множинного вибору до терміну «класична модель даних» (рис. 4.19).

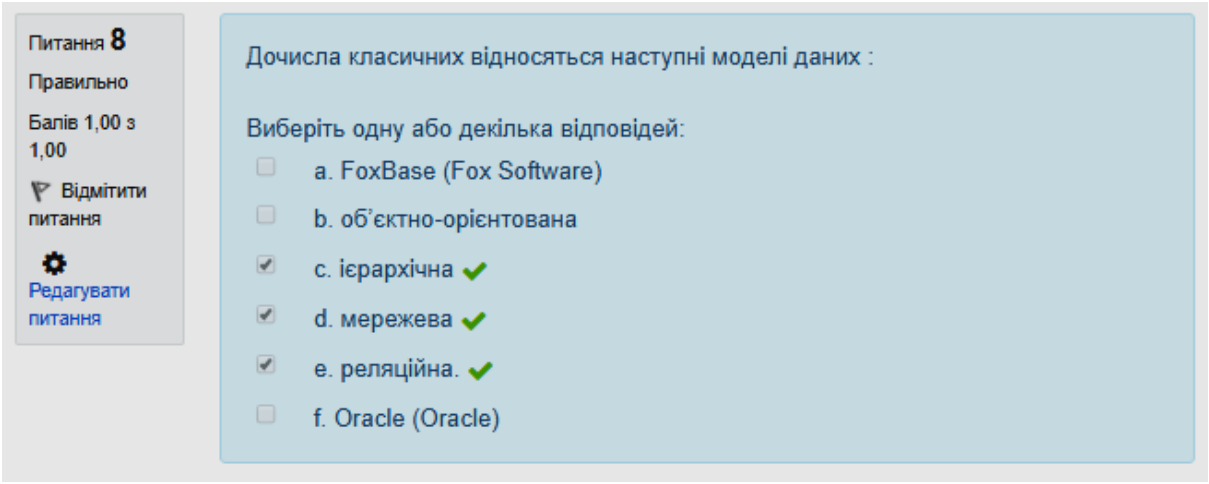


Рисунок 4.19 – Приклад тестового завдання множинного вибору у Moodle

На основі фрагменту контенту «*Вимірювання – це множина однотипних даних, що утворюють одну з граней гіперкуба.*» сформоване

тестове завдання на введення тексту до терміну «класична модель даних» (рис. 4.20).

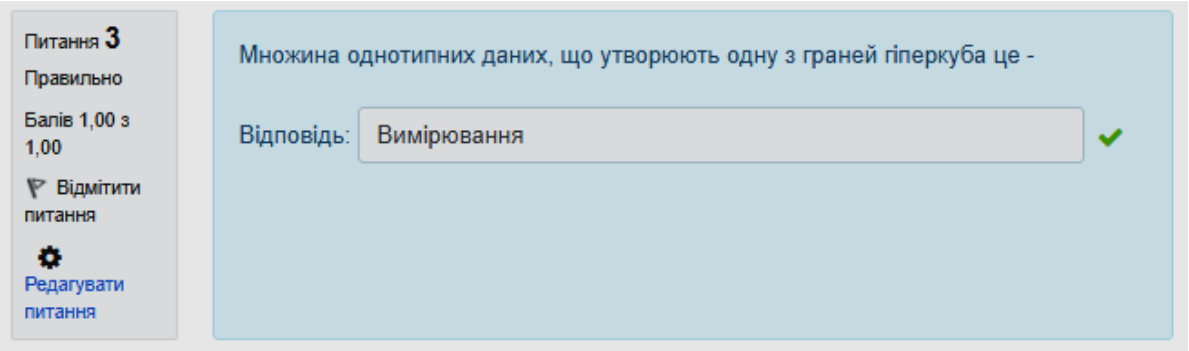


Рисунок 4.20 – Приклад тестового завдання на введення тексту у Moodle

Одержана вибірка містить 154 тестових завдання, з яких логічного типу – 41, одиничного вибору – 44, множинного вибору – 33, із введенням тексту – 36 завдань.

У табл. 4.4 наведено фрагмент результатів проходження тесту у середовищі Moodle з використанням створеної множини тестових завдань. Використовуючи один і той самий набір тестових завдань, студенти проходили тестування з використанням базового алгоритму середовища Moodle та з використанням адаптивного алгоритму.

Таблиця 4.4 – Приклад (фрагмент) результатів дослідження

№ п/п	Суб'єкт тестування	Базовий алгоритм			Адаптивне тестування		
		Затрачений час, хв.	Кількість завдань	Отримана оцінка	Затрачений час, хв.	Кількість завдань	Отримана оцінка
1.	Студент 1	24	15	3,77	15	11	3,21
2.	Студент 2	33	15	4,5	36	14	4,31
3.	Студент 3	27	16	2,12	15	5	1,1
4.	Студент 4	25	15	4,65	21	17	4,48
5.	Студент 5	36	14	4,45	41	9	3,95
6.	Студент 6	23	16	3,58	17	6	2,46
7.	Студент 7	28	15	5,00	26	14	4,46
8.	Студент 8	23	15	4,57	18	11	4,22
9.	Студент 9	31	16	4,15	34	8	3,8
10.	Студент 10	43	15	4,4	23	9	3,96
…	…	…	…	…	…	…	…

Середній час проходження тестів для 55 студентів за категоріями згідно одержаної за національною шкалою оцінки з використанням базового алгоритму середовища Moodle склав: для оцінки «незадовільно» $\overline{TC_2}$ = 25,48 хв., для оцінки «задовільно» $\overline{TC_3}$ = 27,33 хв., для оцінки «добре» $\overline{TC_4}$ = 29,02 хв., для оцінки «відмінно» $\overline{TC_5}$ = 28,36 хв., середній для всіх категорій $\overline{TC_S}$ = 27,62 хв.. Середній час проходження за аналогічних умов з використанням адаптивного алгоритму проходження тесту склав: для оцінки «незадовільно» $\overline{TA_2}$ = 13,27 хв., для оцінки «задовільно» $\overline{TA_3}$ = 15,58 хв., для оцінки «добре» $\overline{TA_4}$ = 24,17 хв., для оцінки «відмінно» $\overline{TA_5}$ = 27,54 хв., середній для всіх категорій $\overline{TA_S}$ = 21,95 хв.. Візуальне подання результатів проходження тестів за середнім часом зображено на рис. 4.21.

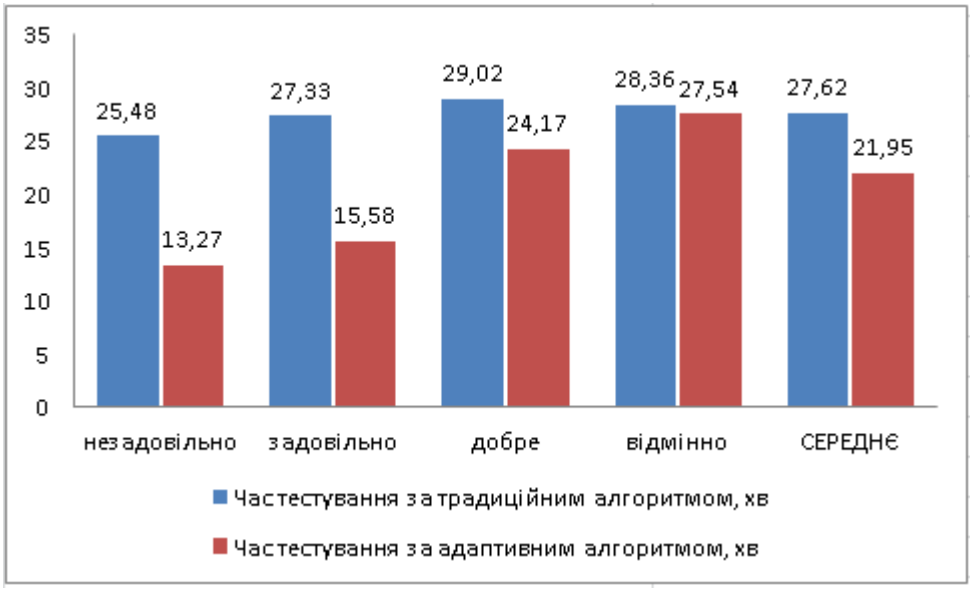


Рисунок 4.21 – Середній час проходження тестів, хв.

Одержані результати тестування дозволили за (4.13) визначити ефект від використання адаптивного тестування за різницею в часі тестування: для оцінки «незадовільно» $\overline{\Delta T_2}$ = 47,92 %, для оцінки «задовільно» $\overline{\Delta T_3}$ = 42,99 %,

для оцінки «добре» $\overline{\Delta T_4}$ = 16,71 %, для оцінки «відмінно» $\overline{\Delta T_5}$ = 2,89 %, середнє значення за всіма категоріями $\overline{\Delta Ts}$ = 20,53 %.

Середня кількість одержаних тестових завдань за використання базового алгоритму середовища Moodle склала: для оцінки «незадовільно» $\overline{NC_2}$ = 15,31 од., для оцінки «задовільно» $\overline{NC_3}$ = 14,83 од., для оцінки «добре» $\overline{NC_4}$ = 15,11 од., для оцінки «відмінно» $\overline{NC_5}$ = 15,07 од., середній для всіх категорій $\overline{NC_S}$ = 15,13 од.. Середня кількість одержаних тестових завдань за використання адаптивного алгоритму проходження тесту склав: для оцінки «незадовільно» $\overline{NA_2}$ = 6,49 од., для оцінки «задовільно» $\overline{NA_3}$ = 8,67 од., для оцінки «добре» $\overline{NA_4}$ = 12,25 од., для оцінки «відмінно» $\overline{NA_5}$ = 16,12 од., середній для всіх категорій $\overline{NA_S}$ = 11,83 од. (рис. 4.22).

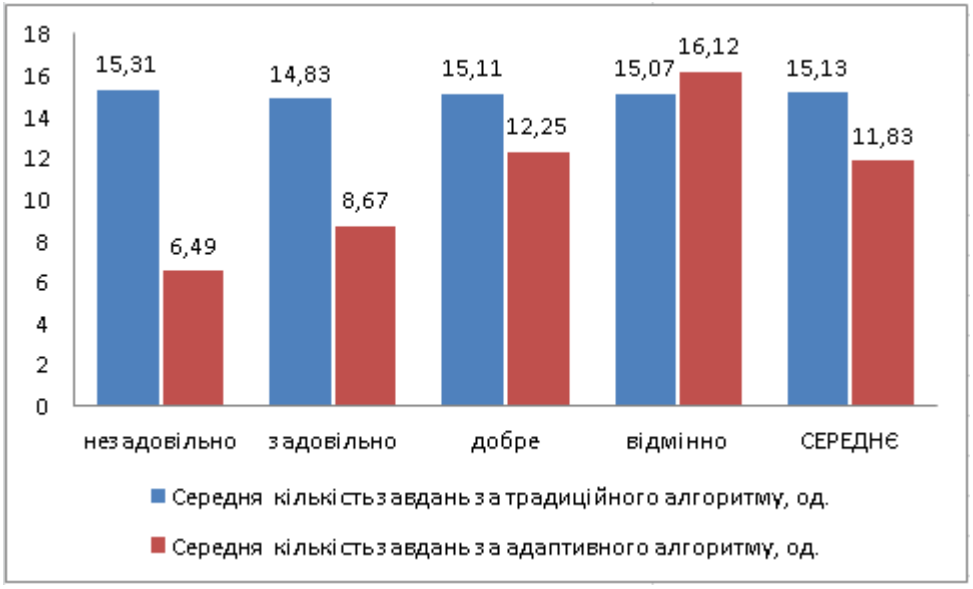


Рисунок 4.22 – Середня кількість одержаних тестових завдань, од.

Одержані результати тестування дозволили за (4.14) визначити ефект від використання адаптивного тестування за кількістю одержаних тестових завдань: для оцінки «незадовільно» $\overline{\Delta N_2}$ = 57,61 %, для оцінки «задовільно»

$\overline{\Delta N_3}$ = 41,54 %, для оцінки «добре» $\overline{\Delta N_4}$ = 18,93 %, для оцінки «відмінно» $\overline{\Delta N_5}$ = -6,97 %, середнє значення за всіма категоріями $\overline{\Delta Ns}$ = 17,28 %.

Результати дослідження підтвердили можливість використання створених за розробленою ІТ множин тестових завдань не тільки для проведення тестування за традиційним алгоритмом тестування, а й для адаптивного тестування.

В середньому для визначення рівня знань за адаптивного тестування знадобилося використання на $\overline{\Delta Ns}$ = 17,28 % меншої кількості завдань. При цьому чим вищим виявився рівень знань піддослідного, тим більша кількість завдань знадобилася для визначення рівня знань за адаптивного тестування, від $\overline{NA_2}$ = 6,49 од. для оцінки «незадовільно» до $\overline{NA_5}$ = 16,12 од. для оцінки «відмінно». Причому, якщо для оцінки «незадовільно» ця кількість була суттєво меншою за показник традиційного алгоритму ($\overline{\Delta N_2}$ = 57,61 %), то для оцінки «відмінно» кількість одержаних тестових завдань виявилась навіть дещо більшою ($\overline{\Delta N_5}$ = -6,97 %), що було пов’язано з необхідністю максимального заглиблення перевірки в кожен елемент структури ІНМ.

Загалом, за $\overline{TA_S} < \overline{TC_S}$, алгоритм адаптивного тестування в середовищі Moodle забезпечив більш швидке на $\overline{\Delta Ts}$ = 20,53 % проходження тесту за базовий алгоритм проходження тесту середовища Moodle. Чим вищим виявився рівень знань студента, тим більша кількість завдань знадобилася для визначення рівня знань за адаптивного тестування, й відповідно тим більше часу було витрачено на їх вирішення – якщо для оцінки «незадовільно» $\overline{\Delta T_2}$ = 47,92 %, то для оцінки «відмінно» $\overline{\Delta T_5}$ = 2,89 %. В той час як за традиційного алгоритму ці показники відрізняються несуттєво.

Хоча $\overline{\Delta N_5} < 0$, одержано $\overline{\Delta T_5} > 0$. Це можна пояснити тим, що за адаптивного тестування тестові завдання подавались в логічно послідовному

порядку, що дало можливість студентам більш зосереджено їх опрацьовувати і зменшило час на «переключення» між різними рубриками.

Загалом, алгоритм адаптивного тестування в середовищі Moodle забезпечив більш швидке на $\overline{\Delta Ts}$ = 20,53 % проходження тесту, при цьому для визначення рівня знань знадобилося використання на $\overline{\Delta Ns}$ = 17,28 % меншої кількості завдань.

На об’єктивність результатів суттєво впливає коректність сформованих тестових завдань. Адже для створення коректного тестового набору, тестові завдання формуються так, щоб вони рівномірно покривали ІНМ. При коректно сформованому наборі тестових запитань, запропонований підхід дозволяє більш точно визначити рівномірність рівня отриманих знань та виявити прогалини в розумінні вивченого матеріалу.

4.2.8 Результати використання інформаційної технології

Створена множина тестових завдань оцінюється за рядом показників, до яких належать дискримінативність, повнота і рівномірність покриття навчального матеріалу, валідність тесту [33, 116]. Оскільки використання ІТ є лише одним з етапів створення результуючої множини тестових завдань, неможливо об’єктивно визначити значення наведених показників. Але можна визначити вплив на кінцевий результат властивостей ІТ за результатами її експериментального тестування.

Дискримінативність множини тестових завдань – визначає, які студенти краще вирішують тестові завдання, які опанували ІНМ чи ні. Дискримінативність зростає, якщо завдання вирішується краще студентами, які опанували ІНМ, й зменшується, якщо завдання вирішується краще студентами, які не опанували ІНМ. У зв’язку з тим, що використання ІТ передбачає можливість подальшого коригування створених тестових завдань, в даному випадку цей показник є суб’єктивним. Втім, оскільки всі тестові завдання створюються на основі контенту ІНМ, можна стверджувати, що

ознайомлення з ІНМ й засвоєння відповідних знань безпосередньо впливають на підвищення кількості правильних відповідей при проходженні відповідного тесту.

Якість дистракторів. У тестових завданнях, створених вручну, дистрактори можуть неявно містити підказки внаслідок виходу їх сенсу за межі відповідного контенту ІНМ (наприклад, на питання «Різниця бітів між двома векторами це відстань» відповіді: «Хемінга», «Хемінгуея», «Трампа»). На відміну від тестових завдань, створених вручну, ІТ передбачає формування дистракторів за контентом цього ж ІНМ, що виключає подібні випадки. Синтаксична узгодженість відповідей з контентом завдання при необхідності може бути виконана вручну.

Повнота покриття ІНМ – визначає, яку частину контенту ІНМ покриває множина створених тестових завдань. Відповідно до досліджень у п. 2.4.5, для створення тестових завдань використано в середньому контент 100% рубрик, або 82,6% абзаців, або 91,2% речень з контенту ІНМ.

Рівномірність покриття ІНМ множиною тестових завдань – визначає, чи рівномірно і достатньо покритий ними контент ІНМ. В зв’язку з тим, що ІТ передбачає створення надлишкової кількості тестових завдань (згідно п. 2.4.6, викладач видаляє в середньому 49,21% створених автоматизовано тестових завдань) з метою подальшого ручного вибору найбільш вдалих для включення у тест, цей показник є суб’єктивним. Втім, згідно п. 2.4.5, ключові терміни присутні в середньому в 91,2% речень з контенту ІНМ, й у середньому в 97,8% випадків появи терміна, до кожного випадку створюється мінімум одне тестове завдання кожного типу. Це дозволяє досягти відповідної рівномірності покриття контенту ІНМ тестовими завданнями без створення вручну додаткових тестових завдань.

Валідність тесту – характеристика, яка визначає можливість тестових завдань відповідати саме певному рівню складності, тобто їх не можна виконати за нижчого рівня знань (функціональна валідність тесту). Оскільки для перевірки рівня знань певного елементу структури ІНМ (наприклад, теми)

у тестовому завданні використовуються елементи її ж контенту, забезпечується валідність тесту за структурою ІНМ. Оскільки ключові терміни, знання яких перевіряється, мають різну ступінь семантичної важливості (згідно п. 1.1.1 та п. 1.3.2) й відповідно різну міру присутності і висвітленості в контенті ІНМ, одержана відповідність показника семантичної важливості терміна та рівня складності тестового завдання забезпечує валідність тесту за семантикою ІНМ. Як результат, використання створених за ІТ тестових завдань для класичного та адаптивного тестування (наведено в п. 4.2.7) дозволило виявити студентів із різними рівнями знань.

Висновки до розділу

1. Для експериментального тестування ІТ автоматизованого формування тестових завдань до навчальних матеріалів було розроблено дві відповідні прикладні програмні системи. Інформаційна система для автоматизованого створення тестів функціонально відповідна розробленій ІТ й складається з двох підсистем – підсистеми формування структури ІНМ та пошуку ключових термінів і підсистеми автоматизованого формування тестових завдань. Застосування для адаптивного тестування використовує передбачену в моделі семантичної структури НК можливість збереження даних не тільки окремого тестового завдання, а й його зв’язку з множинами ключових термінів і заголовків, для адаптивного тестування, й складається з двох підсистем – підсистеми розробки тестів і засобу для проведення адаптивного тестування у вигляді плагіна до середовища Moodle.

2. Дослідження ефективності пошуку ключових термінів у контенті елементів ІНМ підтвердили можливість ефективно автоматизовано формувати множини ключових семантичних термінів до контенту елементів ІНМ з показниками точності пошуку до 92,9% та повноти пошуку до 100,0%. При цьому встановлено, що за значення показника щільності 11% ключові терміни

з автоматизовано одержаної множини містяться в середньому в 89,1% речень, тобто 91,4% текстового контенту.

3. Дослідження повноти та рівномірності покриття контенту тестовими завданнями виявили, що при використанні інформаційної системи для автоматизованого створення тестів у середньому в 97,8% випадків появи ключового терміну в контенті створюється мінімум одне тестове завдання кожного типу. Завдяки поєднанню в правилах продукції однакових антецедентів та різних консеквентів досягається мінімально необхідна рівномірність розподілу тестових завдань за типами та використаними ключовими термінами.

4. Перевагами представлення алгоритмів формування тестових завдань у вигляді продукційних правил визначено модульність (окремі правила продукції можуть буди додані, видалені чи відредаговані незалежно від інших), наочність та однаковість їх структури, простоту створення та розуміння окремих правил й простоту механізму логічного виведення.

5. З метою визначення можливості використання створених тестів для адаптивного тестування, було використано відповідне застосування, за використання якого студенти проводили тестування з традиційним алгоритмом проходження тесту середовища Moodle та з використанням адаптивного алгоритму проходження тесту. Алгоритм адаптивного тестування в середовищі Moodle забезпечив у середньому на 20,53 % більш швидке проходження тесту, при цьому для визначення рівня знань знадобилося використання в середньому на 17,28 % меншої кількості завдань.

6. За результатами експериментального тестування ІТ автоматизованого формування тестових завдань до навчальних матеріалів було визначено, що розроблений метод автоматизованого формування тестових завдань не втрачаючи переваг ручного методу (контроль рівномірності покриття контенту навчальних матеріалів, контроль повноти уваги до семантичних термінів, контроль рівномірності уваги до семантичних термінів, можливість використання тесту для адаптивного тестування, низькі

трудозатрати для підготовки контенту навчальних матеріалів), зберігає переваги існуючих методів формування тестових завдань (висока швидкість формування множини тестових завдань).

7. Встановлено наступні властивості розробленої ІТ. Оскільки правила продукції застосовуються для всіх рівнів семантичної структури навчальних матеріалів, забезпечується повне покриття начального матеріалу. Автоматизація процесу формування тестових завдань забезпечує суттєве скорочення часу на розробку тестових завдань. Дані, що містяться у моделі (заголовки, терміни, тестових завдання, зв’язки) дають можливість проведення адаптивного контролю рівня одержаних знань.

ВИСНОВКИ

Дисертаційна робота є закінченим науковим дослідженням, розв’язує науково-технічну задачу створення ІТ автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань. У рамках роботи поставлені та вирішені такі завдання:

1. За аналізом існуючих методів для пошуку ключових термінів у цифрових текстах і автоматизованого формування тестових завдань до ІНМ обґрунтовано потребу у створенні ІТ автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань.

2. Удосконалено інформаційну модель семантичної структури навчального курсу, яка на відміну від відомих забезпечує можливість формального подання семантичної структури навчального курсу з інформативністю, достатньою для автоматизованого створення розподілених за структурою ІНМ тестових завдань.

3. Удосконалено метод формування структури навчальних матеріалів та пошуку у них ключових термінів, який на відміну від відомих дає змогу врахувати можливість взаємного поглинання термінів та ідентифікувати як термін не тільки слова, а й словосполучення.

4. Розроблено новий метод автоматизованого формування тестових завдань до навчальних матеріалів, який використовує продукційну модель для подання правил формування тестових завдань і дозволяє одержувати множини тестових завдань без додаткової формалізації ІНМ;

5. Розроблено нову ІТ автоматизованого створення тестових завдань до навчальних матеріалів, яка дає змогу за вхідними даними ІНМ автоматизовано одержувати множину тестових завдань, що придатні для проведення адаптивного тестування.

Виконано експериментальну перевірку інформаційної технології автоматизованого створення тестових завдань шляхом розробки й

використання інформаційної системи для автоматизованого створення тестів. Результати експериментального тестування запропонованої ІТ довели її спроможність розв’язувати поставлені задачі. При цьому, середня точність пошуку ключових термінів складає 73,2%, середня повнота пошуку – 69,7%. За показником щільності 11% знайдені ключові терміни містяться в середньому в 89,1% речень, що складають 91,4% текстового контенту. У середньому в 97,8% випадків появи ключового терміну в контенті створюється мінімум одне тестове завдання кожного типу. Завдяки поєднанню в правилах продукції однакових антецедентів та різних консеквентів досягається мінімально необхідна повнота розподілу тестових завдань за типами та використаними ключовими термінами. Використання створених тестів для адаптивного тестування забезпечило у середньому на 20,57% більш швидке проходження тесту, при цьому для визначення рівня знань знадобилося використання в середньому на 19,33% меншої кількості завдань. Використання розробленої ІТ дозволяє в 100% випадків досягти поставленої мети за менший час (у середньому на 60,25%) у порівнянні ручним створенням тестових завдань. Водночас із числа прийнятих до роботи тестових завдань 46,56% не потребують коригування чи змін.

Розроблені положення знайшли практичне застосування у роботі Навчального простору «VECTOR» (акт про впровадження від 01.04.2019 р.), Приватного малого підприємства «Лінк» (акт про впровадження від 15.04.2019 р.), Товариства з обмеженою відповідальністю “Науково-технічна фірма «Інфосервіс»” (акт про впровадження від 07.05.2019 р.) та у навчальному процесі Хмельницького національного університету (акт про впровадження від 22.05.2019 р.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Krak I., Barmak O., Mazurets O. The practice investigation of the information technology efficiency for automated definition of terms in the semantic content of educational materials. CEUR Workshop Proceedings. 2016. Vol. 1631. P. 237–245.

2. Krak I., Barmak O., Mazurets O. The practice implementation of the information technology for automated definition of semantic terms sets in the content of educational materials. CEUR Workshop Proceedings. 2018. Vol. 2139. P. 245–254.

3. Barmak O., Krak I., Mazurets O., Pavlov S., Smolarz A., Wojcik W. Research of efficiency of information technology for creation of semantic structure of educational materials. Advances in Intelligent Systems and Computing. 2020. Vol. 1020. P. 554–569.

4. Бармак О. В., Мазурець О. В. Методи автоматизації визначення семантичних термінів у навчальних матеріалах. Вісник Хмельницького національного університету. Серія : Технічні науки. 2015. № 2. С. 209–213.

5. Бармак О. В., Мазурець О. В. Інформаційна технологія автоматизованого визначення термінів у навчальних матеріалах. Вимірювальна та обчислювальна техніка в технологічних процесах. 2015. № 2. С. 94–102.

6. Бармак О. В., Мазурець О. В., Матвійчук А. О. Застосування інформаційної технології гнучкого тестування рівня знань у середовищі Moodle. Вісник Хмельницького національного університету. Серія : Технічні науки. 2017. № 2. С. 103–114.

7. Бармак О. В., Мазурець О. В., Кліменко В. І. Інформаційна технологія автоматизованого формування тестових завдань. Вісник Хмельницького національного університету. Серія : Технічні науки. 2017. № 5. С. 93–103.

8. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2017. № 6. С. 223–229.

9. Мазурець О. В., Ковальчук О. В., Слободзян В. О. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 1. С. 61–69.

10. Мазурець О. В. Інформаційна технологія автоматизованого визначення семантичних термінів в елементах навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 3. С. 223–230.

11. Мазурець О. В. Розробка множини тегів для формального опису елементів моделей автоматизованого формування тестових завдань. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 5. С. 26–31.

12. Крак Ю. В., Бармак О. В., Мазурець О. В. Практична реалізація інформаційної технології автоматизованого визначення множини семантичних термінів в контенті навчальних матеріалів. Проблеми програмування. 2018. № 2–3. С. 245–254.

13. Бармак О. В., Мазурець О. В. Інформаційна модель семантичної структури навчального курсу. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 6. Т. 1. С. 92–97.

14. Мазурець О. В. Інформаційна технологія автоматизованого створення тестів до навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2019. № 4. С. 84–91.

15. Мазурець О. В. Метод автоматизованого формування тестових завдань. Вісник Хмельницького національного університету. Серія : Технічні науки. 2019. № 5. С.189–194.

16. Мазурець О. В. Пат. на корисну модель 129903. Україна, МПК G06F 17/00. Спосіб визначення переліку ключових слів у тексті. № u201707242; заявл. 10.07.2017; опубл. 26.11.2018, Бюл. № 22.

17. Мазурець О. В. Пат. на корисну модель 137386. Україна, МПК G06F 17/00. Спосіб обмеження переліку ключових слів тексту. № u201900552; заявл. 18.01.2019; опубл. 25.10.2019, Бюл. № 20.

18. Мазурець О. В. Інформаційна технологія побудови онтологічної моделі навчального курсу для оцінювання отриманих знань. Інформаційні управляючі системи та технології : матеріали III Міжнародної науково-практичної конференції. (м. Одеса, 23–25 вересня 2014 р.). Одеса, 2014. С. 81–83.

19. Бармак О. В., Крак Ю. В., Мазурець О. В. Інформаційна технологія автоматизованого визначення термінів у навчальних матеріалах. Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту : матеріали Міжнародної наукової конференції. (с. Залізний Порт, 25–28 травня 2015 р.). Херсон, 2015. С. 26–28.

20. Мазурець О. В. Особливості застосування інформаційної технології автоматизованого визначення термінів у навчальних матеріалах. Інформаційні управляючі системи та технології : матеріали IV Міжнародної науково-практичної конференції. (м. Одеса, 22–24 вересня 2015 р.). Одеса, 2015. С. 62–65.

21. Мазурець О. В. Дослідження ефективності інформаційної технології автоматизованого визначення термінів у навчальних матеріалах. ICACIT-2015 : матеріали III Міжнародної конференції з автоматичного управління та інформаційних технологій (м. Київ, 11–13 грудня 2015 р.). Київ, 2015. С. 136–139.

22. Кліменко В. І., Мазурець О. В. Аналіз сучасних методів генерації тестових завдань. Актуальні проблеми комп’ютерних технологій : збірник наукових праць за матеріалами X міжнародної науково-технічної конференції. (м. Хмельницький, 31 травня 2016 р.). Хмельницький, 2016. С. 77–84.

23. Крак Ю. В., Бармак О. В., Мазурець О. В. Практичне дослідження ефективності інформаційної технології автоматизованого визначення семантичних термінів в контенті навчальних матеріалів. Прикладне програмне забезпечення УкрПРОГ’2016 : матеріали X Міжнародної науково-практичної конференції по програмуванню. (м. Київ, 24–26 травня 2016 р.). Київ, 2016. С. 237–245.

24. Мазурець О. В. Особливості автоматизованого формування тестових завдань. Інформаційні управляючі системи та технології : матеріали V Міжнародної науково-практичної конференції. (м. Одеса, 20–22 вересня 2016 р.). Одеса, 2016. – С. 71-73.

25. Мазурець О. В., Якимюк О. М. Моделі оцінки ефективності методів пошуку ключових термінів у контенті навчальних матеріалів. Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи : матеріали Всеукраїнської науково-практичної конференції з міжнародною участю. (м. Тернопіль, 9–10 листопада 2017 р.). Тернопіль, 2017. С. 258–261.

26. Мазурець О. В. Інформаційна технологія гнучкого тестування рівня знань у середовищі MOODLE. Інформаційні управляючі системи та технології : матеріали VI Міжнародної науково-практичної конференції. (м. Одеса, 20–22 вересня 2017 р.). Одеса, 2017. С. 83–85.

27. Крак Ю. В., Бармак О. В., Мазурець О. В. Практична реалізація інформаційної технології автоматизованого визначення множини семантичних термінів в контенті навчальних матеріалів. Прикладне програмне забезпечення УкрПРОГ’2018 : матеріали XI Міжнародної науково-практичної конференції по програмуванню. (м. Київ, 22–24 травня 2018 р.). Київ, 2018. С. 245–254.

28. Мазурець О. В. Використання множини тегів для формального опису моделей формування тестових завдань. Інформаційні управляючі системи та технології : матеріали VII Міжнародної науково-практичної конференції. (м. Одеса, 17–18 вересня 2018 р.). Одеса, 2018. С. 69–72.

29. Бармак О. В., Мазурець О. В. Дослідження точності та повноти автоматизованого визначення семантичних термінів у навчальних матеріалах. Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи : матеріали III Міжнародної науково-практичної конференції. (м. Тернопіль, 5 квітня 2019 р.). Тернопіль, 2019. С. 98–101.

30. Крак Ю. В., Бармак О. В., Мазурець О. В. Інформаційна модель семантичної структури навчального курсу для генерації тестових завдань. Моделювання та дослідження стійкості динамічних систем : матеріали XIX Міжнародної науково-практичної конференції. (м. Київ, 22–24 травня 2019 р.). Київ, 2019. С. 365–367.

31. Мазурець О. В. Застосування продукційної моделі для автоматизованої генерації тестових завдань. Інформаційні управляючі системи та технології : матеріали VIII Міжнародної науково-практичної конференції. (м. Одеса, 23–25 вересня 2019 р.). Одеса, 2019. С. 52–54.

32. Павлиш В. Н., Зайцева М. Н. Теоретичні основи контролю в навчальному процесі вищої школи. Навчальні праці ДНТУ. 2011. №9. С. 130–136.

33. Снитюк В. Е., Юрченко К. Н. Интеллектуальное управление оцениванием знаний. Черкассы, 2013. 262 с.

34. Мартинюк А. Проблема структурування знань та її значення в навчальному процесі. Вісник Львівського університету. 2008. № 24. С. 28–37.

35. Медведєва А. С. Підготовка майбутніх учителів до структурування навчальної інформації у дидактичному процесі загальноосвітньої школи (на матеріалі математики і фізики): дис. канд. пед. наук: 13.00.04. Одеса, 2003.

36. Елизаренко Г. Н. Проектирование компьютерных курсов обучения: концепция, язык, структура. К.: НТУУ “КПІ”, 2001.

37. Кравченко Ю. В., Оксіюк О. Г. Концепція структурування інформаційного ресурсу системи дистанційного навчання. Сучасні інформаційні технології у сфері безпеки і охорони. 2009. № 1(4). С. 6–11.

38. Krissilov V., Ngoc Vu Huy, Zinovatna S. Information model of distance learning system in terms of data communication in heterogeneous Internet networks. Праці Одеського політехнічного університету. 2018. №1(54). С. 62–68.

39. Федусенко О. В., Рафальська О. О. Розробка загальної концептуальної моделі дистанційного розгалуженого курсу. Управління розвитком складних систем. 2011. Вип. 8. С. 92–96.

40. Гладуш В. А., Лисенко Г. І. Педагогіка вищої школи: теорія, практика, історія. Навчальний посібник. Д., 2014. 416 с.

41. Lyalina Y., Langmann R., Krisilov V.The Interaction Model in iLearning Environments and Its Use in the Smart Lab Concept. International Journal of Online Engineering. 2011. Вип. 7. № 4. С. 16–19.

42. Бєляк О. М. Структурування навчальної інформації як складова підготовки студентів немовних спеціальностей. Наука і освіта. 2014. № 3. С. 12–15.

43. Коробов Є. Т. Навчальний матеріал та його структура. URL: http://www.rusnauka.com/19_DSN_2010/Pedagogica/69745.doc.htm (дата звернення: 03.11.2017).

44. Снитюк В. Е., Юрченко К. Н. Элементы знаниеориентированных систем профессиональной подготовки адаптивного типа. Вісник Херсонського національного технічного університету. 2010. № 2 (38). С. 180–186.

45. Мельник А. М. Інформаційна технологія автоматичної генерації тестових завдань з керованою складністю: Автореф. дис. канд. техн. наук: спец. 05.13.05 «Інформаційні технології». Тернопільський національний економічний університет. Тернопіль, 2011. 157 с.

46. Григорьев Г. В., Ручкина Е. М. Вариативность терминологических определений как одна из ключевых проблем современного. Культура и цивилизация. 2017. Вып. 7А. С. 438–450.

47. Keith A. Natural Language Semantics. Blackwell Publishers Ltd. Oxford, 2001. 251 p.

48. Cruse A. Meaning in Language. An Introduction to Semantics and Pragmatics. Second Edition. Oxford University Press. New York, 2004. 137 p.

49. Серажим К. С. Семантичний і семіотичний аспекти аналізу текстів. Вісник Київського національного університету імені Тараса Шевченка. Журналістика. Київ, 2013. № 20. С.34–36.

50. Задворна С. Г. Структурування навчальної інформації: основні дидактичні аспекти. Вісник Черкаського університету. Серія «Педагогічні науки». 2001. Вип. 196. Ч. 2. С. 39–43.

51. Open XML file format. International Organization for Standardization. URL: https://www.iso.org/search/x/query/Open_XML. (дата звернення: 05.12.2017).

52. Considerations for server-side Automation of Office. URL: [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office/. (дата звернення: 05.12.2017).

53. DocumentFormat.OpenXml. URL: https://www.nuget.org/packages/DocumentFormat.OpenXml/. (дата звернення: 05.12.2017).

54. SpiteDoc. URL: https://www.e-iceblue.com/Introduce/word-for-net-introduce.html. (дата звернення: 05.12.2017).

55. Spire.Doc for .NET. URL: https://www.nuget.org/packages/Spire.Doc/. (дата звернення: 05.12.2017).

56. DocX for creates or modifies Microsoft Word files. URL: https://github.com/xceedsoftware/DocX. (дата звернення: 05.12.2017).

57. Загальні відомості про стилі у програмі Word. URL: http://office.microsoft.com/uk-ua/word-help/HA102647012.aspx. (дата звернення: 05.12.2017).

58. Free Spire.Doc for .NET. URL: https://www.e-iceblue.com/Introduce/free-doc-component.html. (дата звернення: 05.12.2017).

59. Коротун О. В. Методологічні засади змішаного навчання в умовах вищої освіти. Інформаційні технології в освіті. 2016. №. 3 (28). С. 117–129.

60. Ткачук В. В., Семеріков С. О., Єчкало Ю. В. Створення електронних навчально-методичних комплексів у мобільно орієнтованому середовищі навчання ВНЗ. Новітні комп'ютерні технології. Кривий Ріг : Видавничий центр ДВНЗ «Криворізький національний університет». 2017. Том XV. С. 189–196.

61. Шабельник Т. В., Лисенко Ю. Г. Моделі дистанційних навчальних технологій як фактор підвищення конкурентоспроможності сучасного закладу вищої освіти. Матеріали II міжнародної науково-практичної конференції «Інтернаціоналізація вищої освіти України в умовах полікультурного світового простору: стан, проблеми, перспективи». Маріуполь. 2018. С. 96–97.

62. Борисовська Ю. О., Козлова О. С., Лисенко О. А. Аналіз сучасних платформ дистанційного навчання. Вісник Херсонського державного технічного університету. 2010. № 2 (38). С. 491–496.

63. Moodle – Open-source learning platform. URL: https://moodle.org/. (дата звернення: 14.05.2019).

64. A Tutor – Expert Tutoring Advice. URL: https://atutor.ca/. (дата звернення: 14.05.2019).

65. Скорюкова Я. Г., Собчук Н. В., Слободянюк О. В., Гречанюк М. С. Особливості використання системи e-Learning Server 3000 при навчанні графічним дисциплінам. Сучасні інформаційні технології та інноваційні методики навчання в підготовці фахівців: методологія, теорія, досвід, проблеми. Вінниця : ВДПУ. 2017. Вип. 48. С. 166–171.

66. Blackboard. Education Technology & Services. URL: https://www.blackboard.com/ (дата звернення: 14.05.2019).

67. A quick course in LearningSpace. URL: https://www.ibm.com/developerworks/lotus/library/ls-quickcourse_LVC/index.html. (дата звернення: 14.05.2019).

68. Sakai Learning Management System. URL: https://www.sakailms.org/. (дата звернення: 14.05.2019).

69. Комп'ютерна програма тестування знань OpenTEST 2. URL: http://opentest.com.ua/kompyuternaya-programma-testirovaniya-znanij-opentest-2/. (дата звернення: 26.02.2017).

70. Контрольно-діагностична система Test-W2. URL: http://teach-inf.at.ua/load/programi/testi/ test_w2_kontrolno_diagnostichna_sistema/16-1-0-7. (дата звернення: 26.02.2017).

71. Тесторіум – система тестування знань. URL: http://www.testorium.net/docs/about#/docs/about. (дата звернення: 26.02.2017).

72. Демида Б., Сагайдак С., Копил І. Системи дистанційного навчання: огляд, аналіз, вибір. Вісник Національного університету «Львівська політехніка». Комп'ютерні науки та інформаційні технології. 2011. № 694. С. 98−107.

73. Каплінський В. В. Методика викладання у вищій школі: Навчальний посібник. Вінниця: ТОВ «Ніланд ЛТД», 2015. 224 с.

74. Методика викладання у вищій школі. Форми й методи навчання у вищій школі. URL: https://pidruchniki.com/88914/pedagogika/ /zformi_metodi_navchannya_vischiy_shkoli. (дата звернення: 12.04.2018).

75. Бюлетень Вищої атестаційної комісії України. 2009. № 6; 2011. №11–12.

76. Закон України «Про вищу освіту» від 17 січня 2002 р. N 2984-III із змінами, внесеними законодавчими документами у 2010 р. // Відомості Верховної Ради (ВВР), 2002, N 20, ст.134; зі змінами. URL: http://www.test.if.ua/wp-content/uploads/2011/01/zakon_pro_vyschu_osvitu.pdf (дата звернення: 12.04.2018).

77. Про порядок розробки складових нормативного та навчально-методичного забезпечення підготовки фахівців з вищою освітою. Наказ міністерства освіти України №285 від 31.07.1998. С.10–11.

78. Титенко С. В. Побудова дидактичної онтології на основі аналізу елементів понятійно-тезисної моделі Наукові вісті НТУУ «КПІ». 2010. № 1. С. 82–87.

79. Снитюк В. Е. Концептуальные принципы и методы проектирования систем автоматизированного контроля знаний. АСУ и приборы автоматики. 2003. Вып. 123. С. 40–43.

80. Войтович І. С., Іващенко А. А. Використання адаптивного тестування в навчальному процесі вищого навчального закладу. Наукові записки КДПУ ім. В. Винниченка. Серія: Проблеми методики фізико-математичної і технологічної освіти. 2014. Вип. 6 (2). С. 3–8.

81. Сергієнко В. П., Малежик М. П., Сіткар Т. В. Комп'ютерні технології в тестуванні: навч. посіб. Луцьк: «Волиньполіграф», 2012. 290 с.

82. Мельник А. М., Пасічник Р. М., Шевчук Р. П. Автоматична генерація тестових завдань як засіб підвищення ефективності процесу навчання. Тези доповідей II Міжнародної науково-практичної конференції «Інформаційні технології та комп'ютерна інженерія». Харків, 2011. С. 57–59.

83. Мельник В. Д. Інтелектуальні технології контролю знань в комп'ютеризованому навчанні. Ukraine–EU. Modern Technology, Business and Law. 2015. С. 60–64.

84. Глібцов Н. Н., Крусь А. А. Реалізація підсистеми тестування в системах дистанційного навчання. Керуючі системами та машинами. 2001. № 3. С. 70–78.

85. Титенко С. В., Гагарін А. А. Практична реалізація технології автоматизації тестування на основі понятійно-тезисної моделі. URL: http://www.setlab.net/?view=Tytenko_Virt06. (дата звернення: 02.11.2017).

86. Андруховський А. Б. Застосування xml-сервісу для побудови системи педагогічного тестування. Матеріали XVI Всеукраїнської наукової

конференції «Сучасні проблеми прикладної математики та інформатики». Львів, 2009. С. 15–16.

87. Снитюк В.Е. Управление процессом оценивания знаний. Структурированность и многокритериальность. Материалы Международной конференции «KDS плюс MeL». К.: 2012. С. 51–52.

88. Нетавська Е. Г. Концептуальні принципи реалізації і структури інструментів контролю знань на базі онтології. In Proc. XIIIth Int. Conf. «Knowledge-Dialogue-Solutions». Bulgaria, Varna, 2007. №2. С. 464–470.

89. Нетавська Е. Г. Структурно-онтологічний підхід до оптимізації процесу контролю знань. Штучний інтелект. 2006. №3. С. 213–219.

90. Тихонов, Ю. Л., Лахно В. А. Онтологічний підхід до контролю знань в е-освіті. Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. Харків: НТУ «ХПІ». 2018. № 16 (1292). С. 128–133.

91. Челышкова М. Б. Теория и практика конструирования педагогических тестов: Учебное пособие. М.: Логос, 2002. 432 с.

92. Снитюк В. Е., Юрченко К. Н. Корректировка сложности вопросов в компьютерных системах профессиональной подготовки // Сборник трудов VII Межд. конф. «Интернет-Образование-Наука-2010». Винница: ВНТУ. 2010. С. 121–122.

93. Мельник А. М., Пасічник Р. М. Модель оцінки складності тестових завдань. Науковий вісник Чернівецького університету: Комп’ютерні системи та компоненти. 2009. № 479. С. 108–113.

94. Чеботарева Н. Е., Федорихин В. А., Симонов В. М., Шильников А. В., Жога Л. В. Универсальный комплекс тестовых заданий различных уровней сложности как эффективное средство систематической оценки качества знаний студентов по физике. Физическое образование в вузах. 2003. Т. 9. № 2. С.45–53.

95. Мисник Л. Д. Методи управління технологіями навчання і тестування знань студентів ВУЗ. Східно-Європейський журнал передових технологій. 2002. № 1/2 (55). С. 24–27.

96. Гайтан Е. Н. Функциональные возможности современных систем автоматизации контроля качества обучения. Сравнительный анализ. Проблеми програмування. 2015. № 2. С. 101–118.

97. Титенко С. В. Генерація тестових завдань у системі дистанційного навчання на основі моделі формалізації дидактичного тексту. Наукові вісті НТУУ «КПІ» 2009. № 1 (63). С. 47–57.

98. Кручинин В. В., Морозова Ю. В. Модели и алгоритмы генерации задач в компьютерном тестировании. Известия Томского политехнического университета. Томский политехнический университет (ТПУ). 2004. Т. 307. № 5. С. 127–131.

99. Нехаев И. Н. Постановка задачи эффективного адаптивного тестирования уровня знаний. Вестник Московского городского педагогического университета. Серия «Информатика и информатизация образования». 2008. № 15. С. 124–127.

100. Давыдова Н. А., Рудинский И. Д. Автоматизированный синтез тестовых заданий для систем педагогического контроля знаний. Информатизация образования и науки. 2013. № 1 (17). С. 77–90.

101. Gutl Ch., Lankmayr K., Weinhofer J., Hofler M.Enhanced Approach of Automatic Creation of Test Items to foster Modern Learning Setting. Electronic Journal of e-Learning. 2011. Т. 9. № 1. С. 23–38.

102. Швецов А. Н., Сергушичева А. П. Автоматическая генерация тестов контроля знаний при подготовке технических специалистов. Сборник трудов международной конференции «Научно-технический прогресс в черной металлургии 2017». 2017. С. 285–293.

103. Мельник А. М., Пасічник Р. М. Метод генерації тестових завдань на основі системи семантичних класів. Вісник Тернопільського державного технічного університету. 2010. Т. 15. № 1. С. 187–193.

104. Мельник А. М. Методи та засоби автоматичної генерації тестових завдань різних форм. Матеріали ІІ–ї науково-практичної конференції «Інноваційні комп’ютерні технології у вищій школі». Львів, 2010. С. 147–152.

105. Турчак А. М., Єгорова О. В., Златкін А. А. Генерація множини ядер продукційних правил в системах управління оцінюванням знань. Молодий вчений. 2016. Т. 11. С. 39–42.

106. Рижов О. А., Попов А. М. Проектування тестових завдань закритого типу на базі моделі онтології на основі когнітивних прототипів. Медична інформатика та інженерія. № 2. 2015. С. 46–51.

107. Сирота С. В., Ліскін В. О. Огляд сучасних онтологокерованих інформаційних систем та сервісів і перспективи їх застосування в електронній освіті. Технологічний аудит та резерви виробництва. 2015. Т. 5. № 6 (25). С. 58–60.

108. Сирота С. В., Ліскін В. О. Розробка генератора тестів для Moodle на базі онтології. Східноєвропейський журнал передових технологій. 2015. №. 5 (2). С. 44–48.

109. Найханова Л. В. Генерация множества ядер продукционных правил в задаче автоматического построения библиотеки декларативным методом. Информационные технологии. 2008. № 10. С. 37–42.

110. Танченко С. С., Титенко С. В., Гагарин А. А. Анализ методов генерации тестовых заданий. XIII международная научная конференция имени Т. А. Таран «Интеллектуальный анализ информации ИАИ-2013». К. : Просвіта. 2013. С. 220–226

111. Титенко С. В. Автоматизация построения тестовых заданий в системах дистанционного обучения на основе понятийно-тезисной модели. Educational Technology & Society. 2013. № 16 (1). С. 482–499.

112. Танченко С. С., Титенко С. В. Усунення мовної неузгодженості в тестових завданнях, згенерованих на основі понятійно-тезисної моделі. URL: http://www.setlab.net/?view=ST_Tanchenko_IAI_2014. (дата звернення: 02.11.2017).

113. Мельник А. М., Пасічник Р. М. Система автоматичної генерації тестових завдань. Тези доповідей міжнародної науково–практичної

конференції «Інформаційні технології та комп'ютерна інженерія». Вінниця, 2010. С. 408–409.

114. Мельник А. М. Автоматична генерація тестових завдань різних типів. Вісник Хмельницького національного університету. 2010. № 4. С. 124–129.

115. Мельник А. М., Пасічник Р. М., Шевчук Р. П. Інформаційна технологія автоматичної генерації тестових завдань з керованою складністю. Системи обробки інформації. Харків, 2011. № 3 (93). С. 57–61.

116. Аванесов В. С. Композиция тестовых заданий: Учебная книга для преподавателей вузов, учителей школ, аспирантов и студентов пед. вузов. 2-е изд., испр. и доп. М.: Адепт, 1998. 218 с.

117. Al Mamun M. A., Hannan M. A., Hussain A., Basri H. Theoretical model and implementation of a real time intelligent bin status monitoring system using rule based decision algorithms. Expert Systems with Applications. 2016. Т. 48. С. 76–88.

118. Liu H., Gegov M. A., Cocea. Rule-based systems: a granular computing perspective. Granular Computing. 2016. Т. 1. №. 4. С. 259–274.

119. Кравець П., Киркало Р. Системи прийняття рішень з нечіткою логікою. Вісник Національного університету «Львівська політехніка». 2009. № 650. С. 115–123.

120. Тарасенко В., Корченко О., Терейковський І. Метод застосування продукційних правил для подання експертних знань в нейромережевих засобах розпізнавання мережевих атак на комп'ютерні системи. Безпека інформації. 2013. № 3 (19). С. 168–174.

121. Bassel G. W., Glaab E., Marquez J., Holdsworth M. J., Bacardit J. Functional network construction in Arabidopsis using rule-based machine learning on large-scale data sets. The Plant Cell. 2011. Т. 23. №. 9. С. 3101–3116.

122. Moghimi M., Varjani A. Y. New rule-based phishing detection method. Expert systems with applications. 2016. Т. 53. С. 231–242.

123. Олійник А. О. Видобування продукційних правил на основі негативного відбору. Радіоелектроніка, інформатика, управління. 2016. № 1 (36). С. 40–49.

124. Hutto C. J. A parsimonious rule-based model for sentiment analysis of social media text / C. J. Hutto, G. E. Vader // Eighth international AAAI conference on weblogs and social media. URL: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/download/8109/8122 (дата звернення: 14.01.2015).

125. Hu X., Pedrycz W., Castillo O., Melin P. Fuzzy rule-based models with interactive rules and their granular generalization. Fuzzy Sets and Systems. 2017. Т. 307. С. 1–28.

126. Kehrer T., Alshanqiti A., Heckel R. Automatic inference of rule-based specifications of complex in-place model transformations. International Conference on Theory and Practice of Model Transformations. Springer, 2017. С. 92–107.

127. Brusilovsky P., Rollinger C., Peyl C. Adaptive and Intelligent Technologies for Web-based Education. Special Issue on Intelligent Systems and Teleteaching, Konstliche Intelligenz. 1999. № 4. С. 19–25.

128. Любченко В. В., Нестеренко А. В. Адаптивность и ее составляющие в обучающих системах. Сборник трудов VII международной конференции «Интеллектуальный анализ информации ИАИ-2007». Киев. 2007. С. 229–232.

129. Durlach P. J., Lesgold A. M. Adaptive Technologies for Training and Education. Cambridge : Cambridge University Press, 2012. 380 с.

130. Іванова О. Н., Кононов О. Н. Адаптивне тестування на практиці діагностики вмінь та знань. URL: http://www.ht.bitnet.ru/press/articles/?view=art129. (дата звернення: 16.04.2018).

131. Гайтан О. М. Елементи технології реалізації автоматизованого адаптивного контролю знань студентів в комп'ютерних системах навчання. Радіоелектронні і комп'ютерні системи. 2014. № 4. С. 97–105.

132. Information Extraction. URL: https://www.itl.nist.gov/iaui/894.02/ /related_projects/muc/index.html. (дата звернення: 15.04.2018).

133. Серажим К. С. Семантичний і семіотичний аспекти аналізу текстів. Вісник Київського національного університету імені Тараса Шевченка. Журналістика. Київ, 2013. № 20. С. 34–36.

134. Luhn H. P. The automatic creation of literature abstracts. Advances in automatic text summarization. The MIT Press, 1999. С. 15–21.

135. Ильичева Н. В., Горелова А. В., Бочкарева Н. Ю. Аннотирование и реферирование. Самара: Вид-во Самарського державного університету, 2003. 100 с.

136. Ненич Л. Про принципи відбору ключових слів у рефератах. Вісник Книжкової палати. 2000. № 9. С. 22–23.

137. Семантичний аналіз. URL: http://cropas.by/seo-slovar/semanticheskij-analiz/. (дата звернення: 05.03.2018).

138. Властивості інформації. URL: https://sites.google.com/site/ /in4matuka/vlastivosti-informacii. (дата звернення: 05.03.2018).

139. Human Language Technologies for Europe. URL: http://tcstar.org/pubblicazioni/D17_HLT_ENG.pdf. (дата звернення: 11.09.2016).

140. Language Engineering pages of the European Commission. URL: http://www.linglink.lu/. (дата звернення: 11.09.2016).

141. Зяблова О. А. Определение термина в когнитивно-дискурсивной парадигме знания. Проблемы и методы современной лингвистики. 2005. № 1. С. 43–54.

142. Головина Е. В., Щербакова М. В. Теоретические аспекты изучения терминов. Филологические науки. Вопросы теории и практики. 2017. № 11 (77), Ч. 2. С. 57–59.

143. Григорьев Г. В., Ручкина Е. М., Васильев Л. Г. Состояние развития понятийно-структурной организации терминологии через призму парадигматических отношений и вариативности терминологических

дефиниций. Филологические науки. Вопросы теории и практики. 2019. Т. 12. № 4. С. 244–251.

144. Бісікало О. В. Формальні методи образного аналізу та синтезу природномовних конструкцій: монографія. Вінниця: ВНТУ, 2013. 316 с.

145. Корниевская Т. А. Термин как объект исследования в лингвистике. Историческая и социально-образовательная мысль. 2015. Т. 7. № 5. Ч. 1. С. 234–237.

146. Chen J., Dosyn D., Lytvyn V., Sachenko A.Smart Data Integration by Goal Driven Ontology Learning. Advances in Big Data. 2016. Т. 529. С. 283–292.

147. Пескова О. В. Алгоритмы классификации полнотекстовых документов. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика. М.: МИЭМ, 2011. С. 170–212.

148. Алгоритм LSA для пошуку схожих документів. URL: https://netpeak.net/ru/blog/algoritm-lsa-dlya-poiska-pohozhih-dokumentov/. (дата звернення: 15.01.2018).

149. Aggarwal C. C., Zhai C. Mining Text Data. Springer, 2012. 527 с.

150. Онлайн-система семантичного аналізу тексту «Itop». URL: https://itop.media/tools.php?i=semantics. (дата звернення: 17.01.2018).

151. Расширювана платформа видобутку текстів. URL: http://citforum.univ.kiev.ua/internet/xml/platform/. (дата звернення: 17.01.2018).

152. Семантичний SEO-аналізатор тексту «Адвего». URL: https://advego.com/text/seo/. (дата звернення: 17.01.2018).

153. Сервіс для семантичного аналізу текстів «Istio». URL: https://istio.com/rus/text/analyz/. (дата звернення: 17.01.2018).

154. Багатофункціональна SEO-платформа Serpstat. URL: https://serpstat.com/uk/. (дата звернення: 17.01.2018).

155. Jamaati M., Mehri A. Text Mining by Tsallis Entropy. Physica A: Statistical Mechanics and its Applications. 2018. № 490. С. 1368–1376.

156. Даркулова К., Ергешова Г. Необходимость выделения ключевых слов для свёртывания текста. VI Международная студенческая электронная

научная конференция «Студенческий научный форум». Лингвистический анализ научного текста. Южно-Казахстанский государственный университет им. Мухтара Ауэзова. Шымкент, 2014. URL: http://www.scienceforum.ru/. (дата звернення: 26.02.2018).

157. Абрамов Е. Г. Подбор ключевых слов для научной статьи. Научная периодика: проблемы и решения. 2011. №2. С. 35–40.

158. Mehri A., Jamaati M., Mehri H. Word ranking in a single document by Jensen–Shannon divergence. Physics Letters A. 2015. Т. 379, Ч. 28–29. С. 1627–1632.

159. Разработка метода поиска ключевых слов для задачи визуализации научных статей. URL: http://www.dialog-21.ru/media/3995/sandrikova.pdf. (дата звернення: 24.02.2018).

160. Недильченко О. С. Этапы и методы автоматического извлечения ключевых слов. Молодой ученый. 2017. №22. С. 60-62. URL: https://moluch.ru/archive/156/44044/. (дата звернення: 24.02.2018).

161. Ландэ Д. В., Снарский А. А. Компактифицированный горизонтальный граф видимости для сети слов. Труды Международной научной конференции «Интеллектуальный анализ информации ИАИ-2013. Знания и рассуждения». НТУУ «КПІ», Киев: 2013. С. 158–164.

162. Ландэ Д. В. Візуалізація статистики входження слів. Матеріали міжнародної конференції MegaLing'2009. Київ, 2009. С.63–64

163. Скороходько Э. Ф. Роль системно- и текстообусловленных характеристик термина в частотном индексировании научных текстов. Научно-техническая информация. Серия 2. 2002. № 8. С. 1–6.

164. Ortuño M., Carpena P., Bernaola P., Muñoz E., Somoza A. M. Keyword detection in natural languages and DNA. Europhys. Lett. 2002. № 57 (5). С. 759–764.

165. J. Ventura, Silva J. New Techniques for Relevant Word Ranking and Extraction. Proceedings of 13th Portuguese Conference on Artificial Intelligence. Springer-Verlag, 2007. С. 691–702.

166. Amancio D. R., Oliveira Jr O. N., Costa L. F. Structure-Semantics Interplay in Complex Networks and its Effects on the Predictability of Similarity in Texts. Physica A: Statistical Mechanics and its Applications. 2012. Т. 391. № 18. С. 4406–4419.

167. Guzmán-Vargas L., Obregón-Quintana B., Aguilar-Velázquez D., Hernández-Pérez R., Liebovitch L. Word-Length Correlations and Memory in Large Texts: A Visibility Network Analysis. Entropy. 2015. Т. 17. № 11. С. 7798–7810.

168. Casas B., Hernández-Fernández A., Català N., Ferrer-i-Cancho R., Baixeries J. Polysemy and Brevity Versus Frequency in Language. Computer Speech & Language. 2019. Т. 58. С. 19–50.

169. Xie P., Yang B., Zhang Z., Andrade R. F. Exact evaluation of the causal spectrum and localization properties of electronic states on a scale-free network. Physica A: Statistical Mechanics and its Applications. 2018. Т. 502. С. 40–48.

170. Haber A., Molnar F., Motter A. E. State Observation and Sensor Selection for Nonlinear Networks. IEEE transactions on control of network systems. 2017. Т. 5. № 2. С. 694–708.

171. Caldeira S., Petit Lobao T. C., Andrade R. F. S., Neme A., J. G. V. Mirand M. G. The Network of Concepts in Written Texts. The European Physical Journal B. Condensed Matter and Complex Systems. 2006. Т. 49. № 4. С. 523–529.

172. Томашевський В. М., Дмитрик І. М. Аналіз моделей навчання та контролю знань. Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: збірник наукових праць. 2008. № 49. С.147–152.

173. Погребнюк І. М., Томашевський В. М. Моделювання сценаріїв адаптивного навчання з використанням мереж Петрі. Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка : збірник наукових праць. 2012. Вип. 55. С. 38–45.

174. Федорук П. І. Адаптивні тести: статистичні методи аналізу результатів тестового контролю знань. Математические машины и системы. 2007. № 3–4. С. 122–138.

175. Федорук П. І. Реалізація методу адаптивного тестування у системах дистанційного навчання. Медична інформатика та інженерія. 2008. № 1. С. 66–71.

176. Lande D. V., Snarskii A. A., Yagunova E. V. The use of horizontal visibility graphs to identify the words that define the informational structure of a text. 12th Mexican International Conference on Artificial Intelligence. IEEE. 2013. С. 209–215.

177. Berners-Lee T., Hendler J., Lassila O. The Semantic Web. Scientific American. 2001. № 284. Ч. 5. С. 34–43.

178. Segaran T., Evans C., Taylor J. Programming the Semantic Web. O'Reilly Media. 2009. 302 с.

179. IDEF5 – Ontology Description Capture Method. URL: http://www.idef.com/IDEF5.htm. (дата звернення: 21.05.2018).

180. Модульне середовище для навчання ХНУ. URL: https://msn.khnu.km.ua/. (дата звернення: 17.06.2019).

181. Навчальний центр заочно-дистанційної освіти ХНУ. URL: https://de.khnu.km.ua/p.aspx. (дата звернення: 17.06.2019).

ДОДАТКИ

Додаток А. Структура основних модулів інформаційної системи для автоматизованого створення тестів

Додаток Б. Лістинги програмного коду основних модулів інформаційної системи для автоматизованого створення тестів

Додаток В. Приклади тестових завдань, створених за розробленими в роботі зразками правил продукції

Додаток Г. Фрагмент таблиці для аналізу параметрів ключових термінів інформаційних навчальних матеріалів

Додаток Д. Результати дослідження особливостей ручного формування тестових завдань авторами навчальних матеріалів

Додаток Е. Акти впровадження та реалізації результатів наукових досліджень

Додаток Ж. Список публікацій за темою дисертації та відомості про апробацію результатів дисертації

Додаток А

Структура основних модулів інформаційної системи для автоматизованого створення тестів

Інформаційна система для автоматизованого створення тестів містить три логічні модулі (рисунок А.1) та модуль користувацького інтерфейсу.

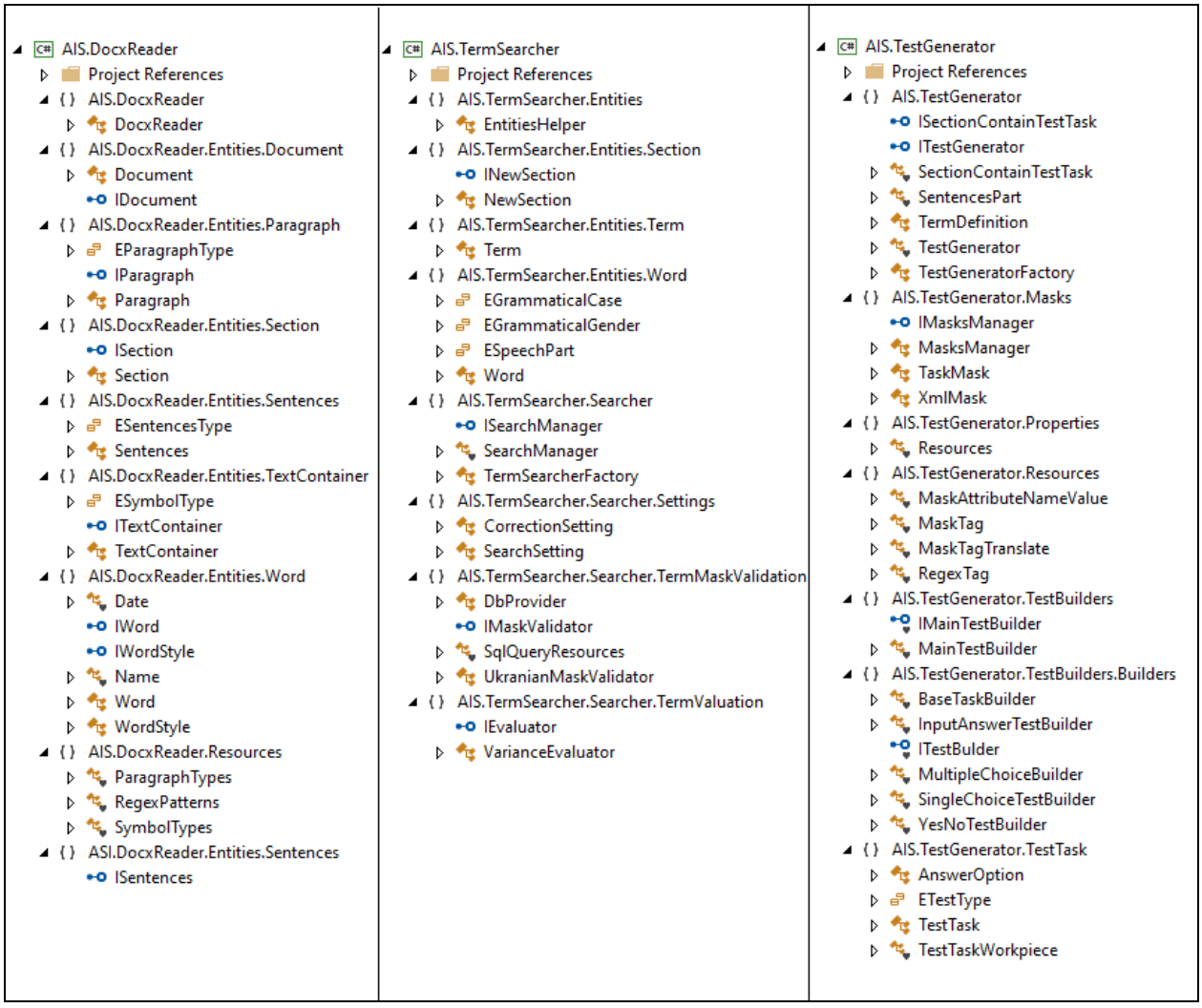


Рисунок А.1 – Структура логічних модулів інформаційної системи

Логічний модуль «Зчитування документу» AIS.DocxReader – модуль для роботи з електронним документом ІНМ, зокрема типу .docx. Головною функцією цього модуля є зчитування документу та його парсинг. Також модуль відповідає за побудову структури документу.

Логічний модуль «Пошук ключових термінів» AIS.TermSearcher – відповідає за формування множин термінів у межах визначених рубрик (заголовків), їх компактифікацію, оцінку важливості та фіксацію результатів.

Логічний модуль «Створення тестових завдань» AIS.TestGenerator – головний модуль, що відповідає за роботу з правилами продукції, типами завдань та формування кінцевого результату – множини тестових завдань та метаданих до них, які визначаються складовими інформаційної моделі семантичної структури навчального курсу.

Модуль користувацького інтерфейсу AIS.TestGenerator.UI (рисунок А.2).

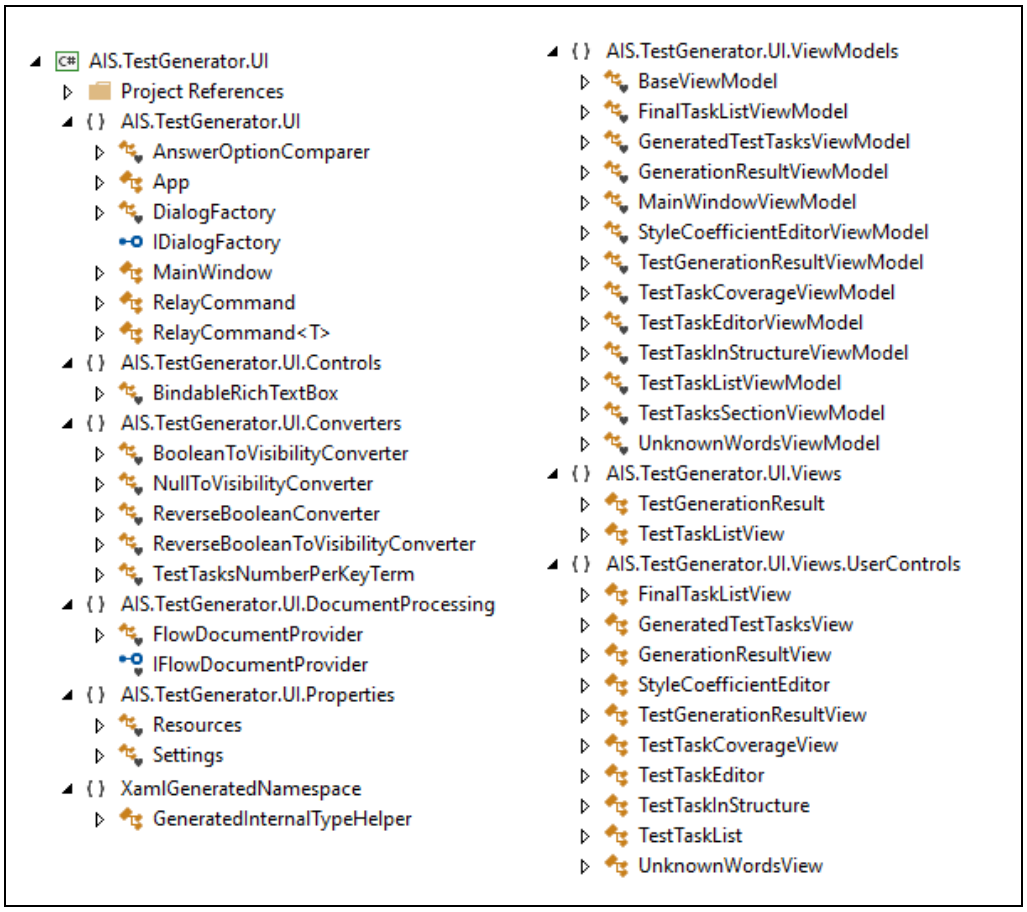


Рисунок А.2 – Структура модулю користувацького інтерфейсу

Наведені модулі послідовно взаємодіють між собою. Обраний для створення тестових завдань цифровий документ з ІНМ передається на обробку в модуль «Зчитування документу» (рисунок А.3). Зчитування документа розпочинається викликом метода *Read* класу *DocxReader*. Результатом

виконання метода є об’єкт класу *Document*, який мітить об’єктну модель документу.

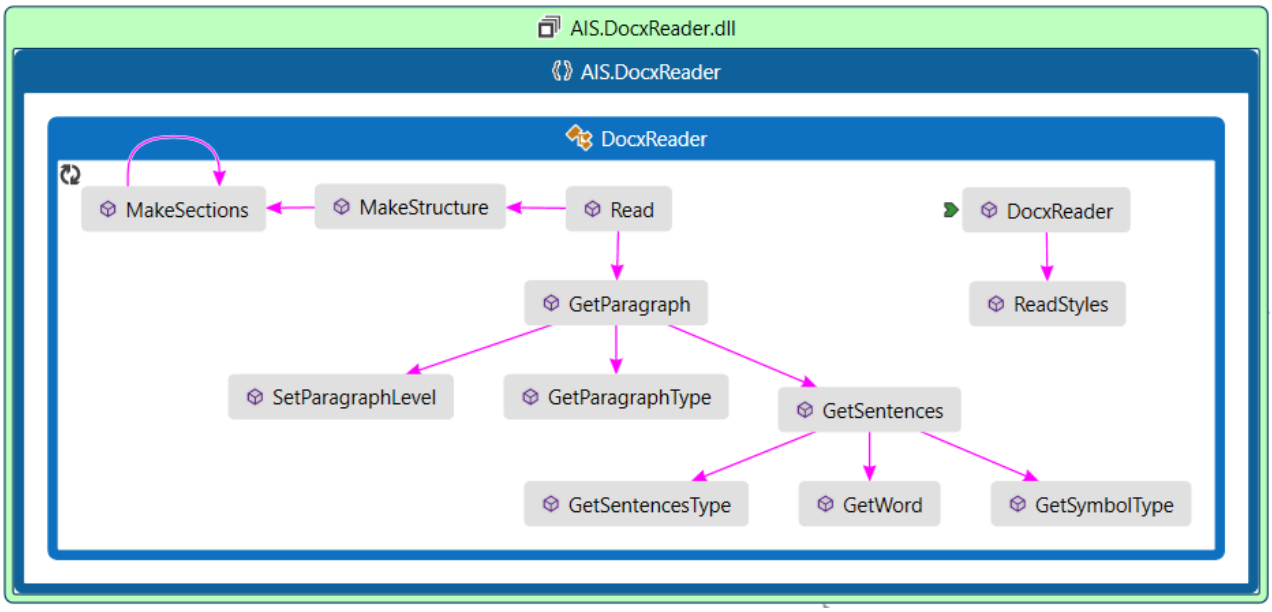


Рисунок А.3 – Карта коду модуля «Зчитування документу»

Отриманий об’єкт зчитаного цифрового документу з ІНМ в попередньому модулі передається на обробку в модуль «Пошук ключових термінів» (рисунок А.4). Пошук термінів розпочинається з виклику методу *Search* в класі *SearhManager*. В методі відбувається перебір всіх секцій документу, кожна з яких передається на обробку методу *Evalute* класу *VarianceEvaluator*. Результатом виконання методу є секція документу, яка містить базову множину ключових термінів. Далі обробка отриманих ключових термінів відбувається в класі *SearhManager*. Кожному слову ключового терміна співставляється інформація про його властивості. При необхідності, секції з ключовими термінами передаються на обробку в клас *UkranianMaskValidator*. Цей клас відфільтровує отримані ключові терміни, відкидаючи ті, що не підходять по антецедентах продукційних правил. Результат роботи даного класу повертається до класу *SearhManager*. На наступних етапах отримані ключові терміни проходять фільтрацію для виключення із множини повторів, проведення взаємного поглинання термінів

й їхня кількість обмежується відповідно до встановленого значення показника щільності ключових термінів у тексті. Результатом виконання модуля є об’єкт класу *Document*, в якому кожна секція мітить відповідні ключові терміни.

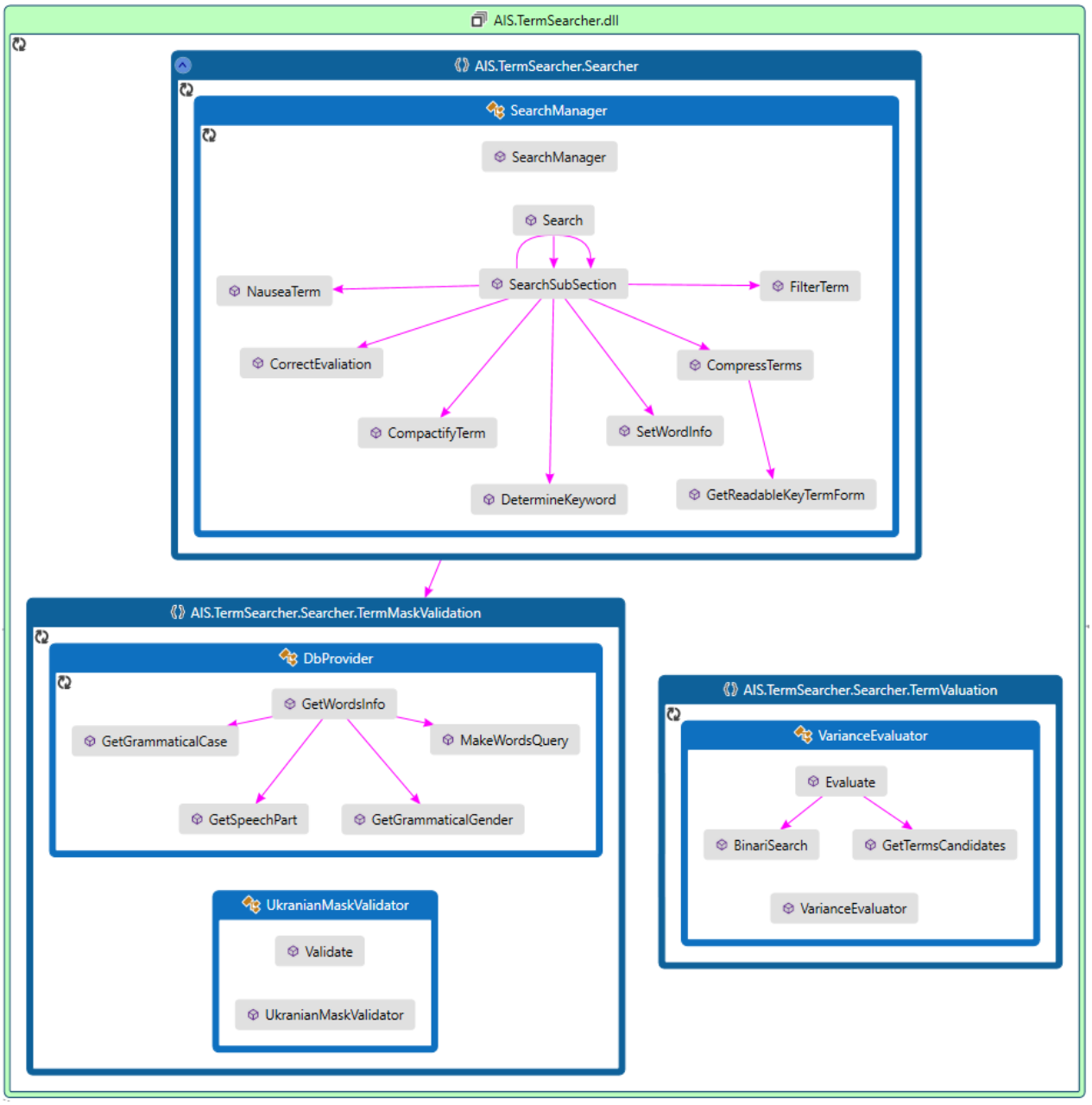


Рисунок А.4 – Карта коду модуля «Пошук ключових термінів»

Результат модуля «Пошук ключових термінів» передається модулю «Створення тестових завдань» (рисунок А.5). Модуль розпочинає свою роботу з того, що завантажує правила продукції для створення тестових завдань, які містяться в базі знань системи у вигляді окремого файлу .xml. Завантаження

антецедентів та консеквентів правил продукції відбувається в класі *MaskManager* методом *Load*, який зберігає структуру правила продукції в класі *XmlMask* і повертає перелік завантажених правил.

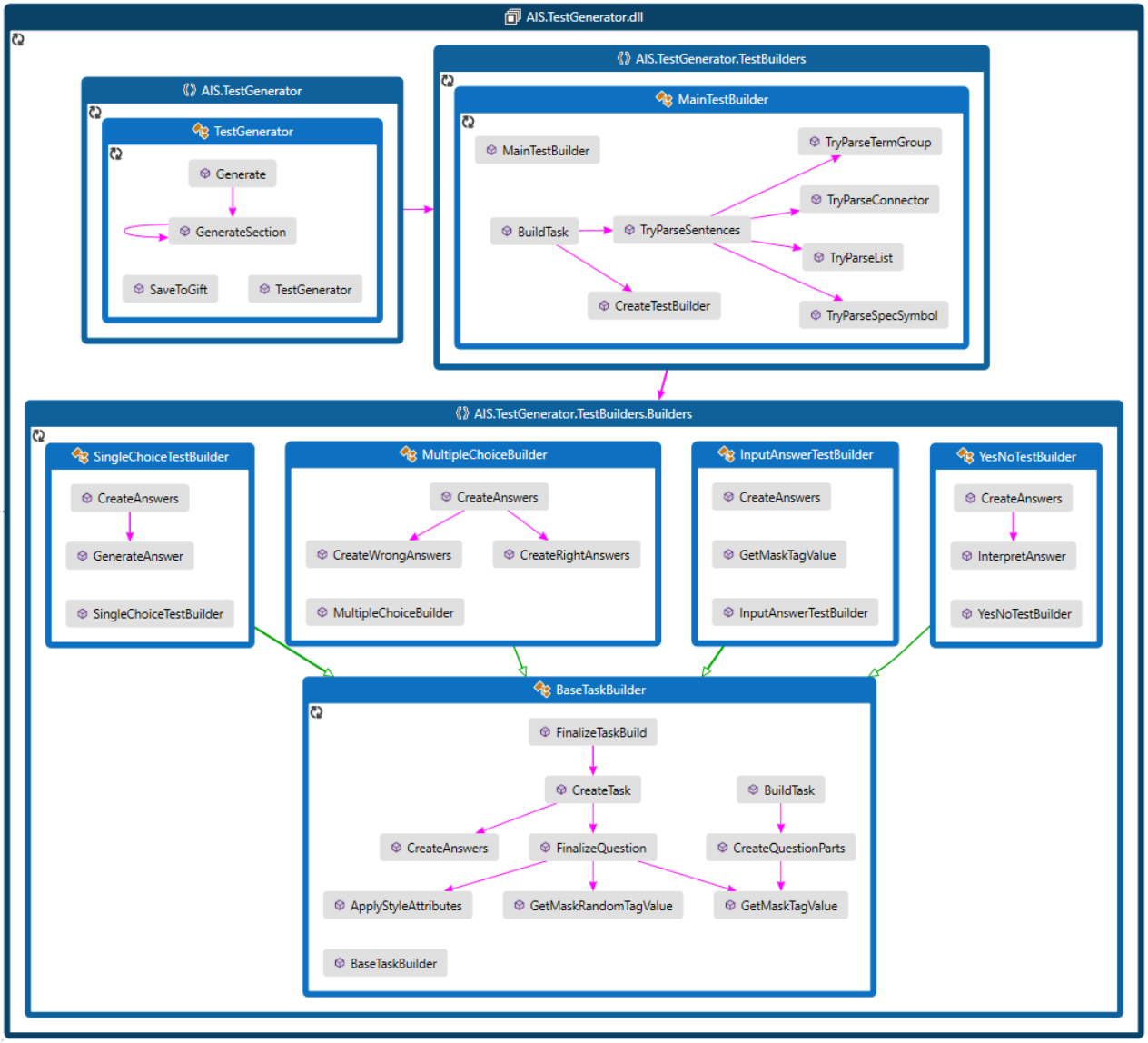


Рисунок А.5 – Карта коду модуля «Створення тестових завдань»

Об’єкт класу *MaskManager* із завантаженими продукційними правилами і документ із ключовими термінами передаються в конструктор класу *TestGenerator*. Створення тестових завдань розпочинається з виклику методу *Generate* або *GenerateParallel* (в залежності від потрібного режиму) об’єкта класу *TestGenerator*. Даний метод послідовно аналізує речення кожної секції текстового документу і у випадку, якщо воно містить ключовий термін,

передає речення в метод *BuildTask* класу *MainTaskBuilder*. Метод послідовно перевіряє, чи передане речення відповідає антецеденту правил продукції; якщо перевірка пройшла успішно, то речення передаються у спеціальні класи для створення тестових завдань: *MultipleTestBuilder*, *SingleChoiceTestBuilder*, *YesNoTestBuilder* та *InputAnswerTestBuilder* (відповідно до типу тестового завдання, що формується консеквентом поточного правила продукції). В кожному із перерахованих класів речення передається в метод *TaskBuild*, який обробляє отримане речення й формує всю потрібну інформацію для тестового завдання, але не створює самого завдання, оскільки вони можуть містити випадкові варіанти відповідей або інші компоненти, які не можна отримати виключно з даного речення. Для цього сформована інформація зберігається в середині класу. Безпосередньо формування кінцевих тестових завдань відбувається в методах *FinalizeTaskBuild* кожного із класів. Їх викликає однойменний метод з класу *MainTaskBuilder*, коли алгоритм дійшов до кінця документу. Після виклику методу *FinalizeTaskBuild* класу *MainTaskBuilder* сформовані тестові завдання записуються в об’єктну модель текстового документу.

Додаток Б

Лістинги програмного коду основних модулів інформаційної системи для автоматизованого створення тестів

AIS.DocxReader

DocxReader.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Paragraph;
using AIS.DocxReader.Entities.Section;
using AIS.DocxReader.Entities.Sentences;
using AIS.DocxReader.Entities.TextContainer;
using AIS.DocxReader.Entities.Word;
using AIS.DocxReader.Resources;
using ASI.DocxReader.Entities.Sentences;
using DocumentFormat.OpenXml.Packaging;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;
using Wordprocessing = DocumentFormat.OpenXml.Wordprocessing;

namespace AIS.DocxReader
{
 public class DocxReader
 {
  private readonly string filePath;
  private readonly WordprocessingDocument wordDocument;
  private readonly Regex regexWord;

  private string[] headingStyles;
  private string[] normalStyles;
  private string[] titleStyles;
  private string[] listStyles;

  private int lastParagraphIndex = 0;
  private int lastHeadingLevel = -1;

  public DocxReader(string filePath)
  {
   this.filePath = filePath;
   wordDocument = WordprocessingDocument.Open(filePath, false);

   regexWord = new
Regex($"{RegexPatterns.Exclusion}|{RegexPatterns.Date}|{RegexPatterns.
Name}|{RegexPatterns.Word}|{RegexPatterns.Separator}");

   ReadStyles();
  }

  public Document Read()
  {
   Document document = new Document()
   {
    Name = Path.GetFileNameWithoutExtension(filePath),
    Sections = new List<ISection>()
   };

   List<IParagraph> paragraphs = new List<IParagraph>();

   foreach (object wordParagraph in
wordDocument.MainDocumentPart.Document.ChildElements[0].ChildElements)
   {
    if (wordParagraph is Wordprocessing.Paragraph
currentWordParagraph)
    {
     IParagraph paragraph = GetParagraph(currentWordParagraph);

     if (paragraph.Sentences.Any())
     {
      paragraphs.Add(paragraph);
     }
    }
   }

   document.Sections.Add(MakeStructure(paragraphs));

   return document;
  }

  private ISection MakeStructure(List<IParagraph> paragraphs)
  {
   Section rootSection = new Section
   {
    Level = -1,
    Name = Path.GetFileNameWithoutExtension(filePath),
    Paragraphs = new List<IParagraph>(),
    Sections = new List<ISection>(),
    Guid = Guid.NewGuid()
   };

   MakeSections(rootSection, paragraphs);

   return rootSection;
  }

  private void ReadStyles()
  {
   headingStyles = ParagraphTypes.Heading.Split(' ');
   normalStyles = ParagraphTypes.Normal.Split(' ');
   titleStyles = ParagraphTypes.Title.Split(' ');
   listStyles = ParagraphTypes.List.Split(' ');
  }

  private void MakeSections(ISection currentSection, List<IParagraph>
paragraphs)
  {
   while (lastParagraphIndex + 1 < paragraphs.Count)
   {
    IParagraph currentParagraph = paragraphs[lastParagraphIndex];

    if (currentParagraph.Level == currentSection.Level)
    {
     if (currentParagraph.Type == EParagraphType.Heading ||
currentParagraph.Type == EParagraphType.Title)
     {
      break;
     }

     currentSection.Paragraphs.Add(currentParagraph);
     lastParagraphIndex++;
    }
    else if (currentParagraph.Level > currentSection.Level)
    {
     string headerText;

     if (currentParagraph.Type == EParagraphType.Heading ||
currentParagraph.Type == EParagraphType.Title)
     {
      headerText = currentParagraph.Text;
     }
     else
     {
      headerText = currentParagraph.Sentences[0].Text;
     }

     Section section = new Section
     {
      Sections = new List<ISection>(),
      Paragraphs = new List<IParagraph>(),
      Level = currentParagraph.Level,
      Guid = Guid.NewGuid(),
      Name = headerText
     };
     section.Paragraphs.Add(currentParagraph);

     lastParagraphIndex++;

     MakeSections(section, paragraphs);
     currentSection.Sections.Add(section);
    }
    else if (currentParagraph.Level < currentSection.Level)
    {
     break;
    }
   }
  }

  private IParagraph GetParagraph(Wordprocessing.Paragraph
wordParagraph)
  {
   Paragraph paragraph = new Paragraph
   {
    Sentences = GetSentences(wordParagraph),
    Type = GetParagraphType(wordParagraph),
    Text = wordParagraph.InnerText
   };

   SetParagraphLevel(paragraph, wordParagraph);

   return paragraph;
  }

  private List<ISentences> GetSentences(Wordprocessing.Paragraph
wordParagraph)
  {
   List<IWord> words = new List<IWord>();
   List<ISentences> sentences = new List<ISentences>();
   List<ITextContainer> textContainers = new List<ITextContainer>();
   WordStyle lastWordStyle = new WordStyle();

   TextContainer textContainer = new TextContainer()
   {
    Words = new List<IWord>(),
    StartSymbolType = ESymbolType.Start
   };

   int lastSentencesIndex = 0;
   int lastTextContainerIndex = 0;

   MatchCollection wordMatches =
regexWord.Matches(wordParagraph.InnerText);
   var wordMathesByStyle = GetWordMathesByStyle(wordParagraph);

   foreach (Match wordMatch in wordMatches)
   {
    if (string.IsNullOrEmpty(wordMatch.Value) ||
string.IsNullOrWhiteSpace(wordMatch.Value))
    {
     continue;
    }

    if (wordMatch.Groups["separator"].Success)
    {
     ESentencesType sentencesType = GetSentencesType(wordMatch.Value);
     ESymbolType symbolType = GetSymbolType(wordMatch.Value);

     textContainer.EndSymbolType = symbolType;
     textContainer.EndSymbol = wordMatch.Value;
     textContainer.Text =
wordParagraph.InnerText.Substring(lastTextContainerIndex,
wordMatch.Index - lastTextContainerIndex).Trim();
     textContainers.Add(textContainer);
     textContainer = new TextContainer()
     {
      Words = new List<IWord>(),
      StartSymbolType = symbolType,
      StartSymbol = wordMatch.Value
     };

     lastTextContainerIndex = wordMatch.Index + 1;

     if (sentencesType == ESentencesType.None)
     {
      continue;
     }

     Sentences currentSentences = new Sentences()
     {
      Text = wordParagraph.InnerText.Substring(lastSentencesIndex,
wordMatch.Index - lastSentencesIndex).Trim(),
      Words = words,
      SentencesType = sentencesType,
      TextContainers = textContainers
     };

     sentences.Add(currentSentences);

     lastSentencesIndex = wordMatch.Index + 1;
     words = new List<IWord>();
     textContainers = new List<ITextContainer>();
    }
    else
    {
     IWord word = GetWord(wordMatch);
     WordStyle wordStyle = new WordStyle();

     if (wordMathesByStyle.ContainsKey(wordMatch.Index))
     {
      wordStyle = wordMathesByStyle[wordMatch.Index].Style;
      word.WordStyle = wordStyle;
     }
     else
     {
      word.WordStyle = wordStyle;
     }

     words.Add(word);

     if (lastWordStyle.Equals(wordStyle))
     {
      textContainer.Words.Add(word);
     }
     else
     {
      textContainer.EndSymbolType = ESymbolType.None;
      textContainer.Text =
wordParagraph.InnerText.Substring(lastTextContainerIndex,
wordMatch.Index - lastTextContainerIndex).Trim();
      if (textContainer.Words.Any())
      {
       textContainers.Add(textContainer);
      }
      textContainer = new TextContainer()
      {
       Words = new List<IWord>() { word },
       StartSymbolType = ESymbolType.None
      };

      lastTextContainerIndex = wordMatch.Index;
     }

     lastWordStyle = wordStyle;
    }
   }

   if (words.Any())
   {
    textContainer.EndSymbolType = ESymbolType.None;
    textContainer.Text =
wordParagraph.InnerText.Substring(lastTextContainerIndex,
wordParagraph.InnerText.Length - lastTextContainerIndex).Trim();
    textContainers.Add(textContainer);

    sentences.Add(new Sentences()
    {
     Text = wordParagraph.InnerText.Substring(lastSentencesIndex,
wordParagraph.InnerText.Length - lastSentencesIndex).Trim(),
     Words = words,
     SentencesType = ESentencesType.None,
     TextContainers = textContainers
    });
   }

   return sentences;
  }

  private ESymbolType GetSymbolType(string symbolType)
  {
   if (SymbolTypes.FullStop.Contains(symbolType))
    return ESymbolType.FullStop;
   else if (SymbolTypes.Comma.Contains(symbolType))
    return ESymbolType.Comma;
   else if (SymbolTypes.OpenBracket.Contains(symbolType))
    return ESymbolType.OpenBracket;
   else if (SymbolTypes.CloseBracket.Contains(symbolType))
    return ESymbolType.CloseBracket;
   else if (SymbolTypes.Hyphen.Contains(symbolType))
    return ESymbolType.Hyphen;
   else if (SymbolTypes.Quotation.Contains(symbolType))
    return ESymbolType.Quotation;
   else return ESymbolType.Other;
  }

  private Dictionary<int, (Match Match, WordStyle Style)>
GetWordMathesByStyle(Wordprocessing.Paragraph wordParagraph)
  {
   Dictionary<int, (Match, WordStyle)> wordMatches = new
Dictionary<int, (Match, WordStyle)>();
   int lastIndex = 0;

   foreach (var paragraphChild in wordParagraph.ChildElements)
   {
    if (paragraphChild is Wordprocessing.Run run)
    {
     WordStyle wordStyle = new WordStyle()
     {
      Bold = run.RunProperties?.Bold != null,
      Italic = run.RunProperties?.Italic != null,
      Underline = run.RunProperties?.Underline != null,
     };

     foreach (Match match in regexWord.Matches(run.InnerText))
     {
      wordMatches.Add(lastIndex + match.Index, (match, wordStyle));
     }

     lastIndex += run.InnerText.Length;
    }
    else if (!string.IsNullOrWhiteSpace(paragraphChild.InnerText))
    {
     foreach (Match match in
regexWord.Matches(paragraphChild.InnerText))
     {
      wordMatches.Add(lastIndex + match.Index, (match, new
WordStyle()));
     }
     lastIndex += paragraphChild.InnerText.Length;
    }
   }

   return wordMatches;
  }

  private IWord GetWord(Match word)
  {
   if (word.Groups["name"].Success)
   {
    return new Name()
    {
     Text = word.Groups["name"].Value,
     Lastname = word.Groups["lastname"].Value,
     Initials = word.Groups["initials"].Value
    };
   }
   else if (word.Groups["year"].Success || word.Groups["day"].Success)
   {
    return new Date()
    {
     Text = word.Value
    };
   }
   else if (word.Groups["word"].Success ||
word.Groups["exclusion"].Success)
   {
    return new Word()
    {
     Text = word.Value
    };
   }
   //todo: check if the condition is possible
   else return null;
  }

  private ESentencesType GetSentencesType(string separator)
  {
   switch (separator)
   {
    case ".":
     return ESentencesType.Declarative;
    case "?":
     return ESentencesType.Interrogative;
    case "!":
     return ESentencesType.Exclamatory;
    default:
     return ESentencesType.None;
   }
  }

  private void SetParagraphLevel(Paragraph paragraph,
Wordprocessing.Paragraph wordParagraph)
  {
   if (paragraph.Type == EParagraphType.Heading)
   {
    if
(int.TryParse(wordParagraph.ParagraphProperties.ParagraphStyleId.Val.I
nnerText, out int level))
    {
     paragraph.Level = level;
     lastHeadingLevel = level;
    }
    else
    {
     paragraph.Level = 0;
     lastHeadingLevel = 0;
    }
   }
   else if (paragraph.Type == EParagraphType.Title)
   {
    paragraph.Level = 0;
    lastHeadingLevel = 0;
   }
   else
   {
    paragraph.Level = lastHeadingLevel;
   }
  }

  private EParagraphType GetParagraphType(Wordprocessing.Paragraph
wordParagraph)
  {
   if (wordParagraph.ParagraphProperties?.ParagraphStyleId == null)
   {
    return EParagraphType.Normal;
   }

   string styleId =
wordParagraph.ParagraphProperties.ParagraphStyleId.Val.InnerText;

   if (headingStyles.Contains(styleId) || styleId.Contains("Heading"))
   {
    return EParagraphType.Heading;
   }
   else if (titleStyles.Contains(styleId))
   {
    return EParagraphType.Title;
   }
   else if (listStyles.Contains(styleId))
   {
    return EParagraphType.List;
   }
   else if (normalStyles.Contains(styleId))
   {
    return EParagraphType.Normal;
   }
   else
   {
    return EParagraphType.None;
   }
  }

  ~DocxReader()
  {
   wordDocument.Dispose();
  }
 }
}
```

Document.cs

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

namespace AIS.DocxReader.Entities.Document
{
 public class Document : IDocument
 {
  public string Name { get; set; }

  public List<ISection> Sections { get; set; }
 }
}
```

EParagraphType.cs

```
namespace AIS.DocxReader.Entities.Paragraph
{
 public enum EParagraphType : sbyte
 {
  None = -1,
  Normal,
  List,
  Table,
  Heading,
  Title
 }
}
```

Paragraph.cs

```
using ASI.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Paragraph
{
 public class Paragraph : IParagraph
 {
  public int Level { get; set; }

  public string Text { get; set; }

  public EParagraphType Type { get; set; }

  public List<ISentences> Sentences { get; set; }
 }
}
```

Section.cs

```
using AIS.DocxReader.Entities.Paragraph;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;

namespace AIS.DocxReader.Entities.Section
{
 public class Section : ISection
 {
  private string text;
  private List<IParagraph> allParagraphs;

  public int Level { get; set; }

  public string Name { get; set; }

  public string Text
  {
   get
   {
    if (string.IsNullOrEmpty(text))
    {
     text = GetText();
    }

    return text;
   }
   protected set { text = value; }
  }

  public Guid Guid { get; set; }

  public List<ISection> Sections { get; set; }

  public List<IParagraph> Paragraphs { get; set; }

  public List<IParagraph> AllParagraphs
  {
   get
   {
    if (allParagraphs == null)
    {
     allParagraphs = new List<IParagraph>();
     allParagraphs.AddRange(this.Paragraphs);
     allParagraphs.AddRange(from s in Sections
             from p in s.AllParagraphs
             select p);
    }

    return allParagraphs;
   }
  }

  private string GetText()
  {
   StringBuilder stringBuilder = new StringBuilder();

   if (Paragraphs != null)
   {
    foreach (var paragraph in Paragraphs)
    {
     stringBuilder.AppendLine(paragraph.Text);
    }
   }

   if (Sections != null)
   {
    foreach (var section in Sections)
    {
     stringBuilder.AppendLine(section.Text);
    }
   }

   return stringBuilder.ToString();
  }
 }
}


ESentencesType.cs
namespace AIS.DocxReader.Entities.Sentences
{
 public enum ESentencesType
 {
  /// <summary>
  /// Default value. It is used when a sentences type impossible to know
  /// </summary>
  None,
  /// <summary>
  /// Ends with '.'
  /// </summary>
  Declarative,
  /// <summary>
  /// Ends with '?'
  /// </summary>
  Interrogative,
  /// <summary>
  /// Ends with '!'
  /// </summary>
  Exclamatory
 }
}
```

Sentences.cs

```
using AIS.DocxReader.Entities.TextContainer;
using AIS.DocxReader.Entities.Word;
using AIS.DocxReader.Entities.Sentences;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.Sentences
{
 public class Sentences : ISentences
 {
  public string Text { get; set; }

  public List<IWord> Words { get; set; }

  public List<ITextContainer> TextContainers { get; set; }

  public ESentencesType SentencesType { get; set; }
 }
}
```

ESymbolType.cs

```
namespace AIS.DocxReader.Entities.TextContainer
{
 public enum ESymbolType
 {
  None,   //Style conteiner
  Comma,  //,
  Fullstop,  //.!?…
  OpenBracket, //[{((
  CloseBracket, //]}))
  Hyphen,  //-‒–—―
  Quotation, //'"‘’“”"
  Start,
  End,
  Other
 }
}
```

TextContainer.cs

```
using AIS.DocxReader.Entities.Word;
using System.Collections.Generic;

namespace AIS.DocxReader.Entities.TextContainer
{
 public class TextContainer : ITextContainer
 {
  public string Text { get; set; }

  public List<IWord> Words { get; set; }

  public ESymbolType StartSymbolType { get; set; }

  public ESymbolType EndSymbolType { get; set; }

  public string EndSymbol { get; set; }

  public string StartSymbol { get; set; }
 }
}
```

Date.cs

```
namespace AIS.DocxReader.Entities.Word
{
 class Date : IWord
 {
  public IWordStyle WordStyle { get; set; }

  public string Text { get; set; }
 }
}
```

Name.cs

```
namespace AIS.DocxReader.Entities.Word
{
 class Name : IWord
 {
  public IWordStyle WordStyle { get; set; }

  public string Text { get; set; }

  public string Lastname { get; set; }

  public string Initials { get; set; }
 }
}
```

Word.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public class Word : IWord
 {
  public IWordStyle WordStyle { get; set; }

  public string Text { get; set; }
 }
}
```

WordStyle.cs

```
namespace AIS.DocxReader.Entities.Word
{
 public class WordStyle : IWordStyle
 {
  public bool Bold { get; set; }

  public bool Italic { get; set; }

  public bool Underline { get; set; }

  public override bool Equals(object obj)
  {
   if (!(obj is WordStyle style))
   {
    return false;
   }
   else if (this.Bold == style.Bold &&
      this.Italic == style.Italic &&
      this.Underline == style.Underline)
   {
    return true;
   }

   return false;
  }
 }
}
```

AIS.TermSearcher

EntitiesHelper.cs

```
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.ComponentModel;

namespace AIS.TermSearcher.Entities
{
 public static class EntitiesHelper
 {
  private static Dictionary<string, ESpeechPart> speechPartMapping;
  private static Dictionary<string, EGrammaticalGender>
grammaticalGenderMapping;
  private static Dictionary<string, EGrammaticalCase>
grammaticalCaseMapping;

  public static Dictionary<string, ESpeechPart> SpeechPartMapping
  {
   get
   {
    if (speechPartMapping == null)
    {
     speechPartMapping = new Dictionary<string, ESpeechPart>();
     foreach (ESpeechPart speechPart in
Enum.GetValues(typeof(ESpeechPart)))
     {
      speechPartMapping.Add(speechPart.GetUkrainianValue().ToLower(),
speechPart);
     }

     return speechPartMapping;
    }
    else return speechPartMapping;
   }
  }

  public static Dictionary<string, EGrammaticalGender>
GrammaticalGenderMapping
  {
   get
   {
    if (grammaticalGenderMapping == null)
    {
     grammaticalGenderMapping = new Dictionary<string,
EGrammaticalGender>();

     foreach (EGrammaticalGender speechPart in
Enum.GetValues(typeof(EGrammaticalGender)))
     {

grammaticalGenderMapping.Add(speechPart.GetUkrainianValue().ToLower(),
speechPart);
     }
    }

    return grammaticalGenderMapping;
   }
  }

  public static Dictionary<string, EGrammaticalCase>
GrammaticalCaseMapping
  {
   get
   {
    if (grammaticalCaseMapping == null)
    {
     grammaticalCaseMapping = new Dictionary<string,
EGrammaticalCase>();

     foreach (EGrammaticalCase speechPart in
Enum.GetValues(typeof(EGrammaticalCase)))
     {

grammaticalCaseMapping.Add(speechPart.GetUkrainianValue().ToLower(),
speechPart);
     }
    }

    return grammaticalCaseMapping;
   }
  }

  public static string GetUkrainianValue(this ESpeechPart eSpeechPart)
  {
   var type = typeof(ESpeechPart);
   var memInfo = type.GetMember(eSpeechPart.ToString());
   var attributes =
memInfo[0].GetCustomAttributes(typeof(DescriptionAttribute), false);
   return ((DescriptionAttribute)attributes[0]).Description;
  }

  public static string GetUkrainianValue(this EGrammaticalGender
eGrammaticalGender)
  {
   var type = typeof(EGrammaticalGender);
   var memInfo = type.GetMember(eGrammaticalGender.ToString());
   var attributes =
memInfo[0].GetCustomAttributes(typeof(DescriptionAttribute), false);
   return ((DescriptionAttribute)attributes[0]).Description;
  }

  public static string GetUkrainianValue(this EGrammaticalCase
eGrammaticalCase)
  {
   var type = typeof(EGrammaticalCase);
   var memInfo = type.GetMember(eGrammaticalCase.ToString());
   var attributes =
memInfo[0].GetCustomAttributes(typeof(DescriptionAttribute), false);
   return ((DescriptionAttribute)attributes[0]).Description;
  }
 }
}
```

SearchManager.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.DocxReader.Entities.Section;
using AIS.TermSearcher.Entities.Section;
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using AIS.TermSearcher.Searcher.Settings;
using AIS.TermSearcher.Searcher.TermMaskValidation;
using AIS.TermSearcher.Searcher.TermValuation;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

namespace AIS.TermSearcher.Searcher
{
 class SearchManager : ISearchManager
 {
  #region Fields and Properties
  private readonly IMaskValidator maskValidator;
  private readonly IEvaluator evaluator;
  private readonly SearchSetting searchSetting;
  private readonly int wordCountInTerm;
  private readonly IDocument document;
  private Dictionary<string, Word> wordsInfo;

  private Dictionary<Guid, List<Term>> keyTerms;
  private Dictionary<Guid, List<Term>> keyTermMaskResult;
  private Dictionary<Guid, List<Term>> keyTermAfterAbsorption;
  private Dictionary<Guid, List<Term>> compresedKeyTerms;
  private Dictionary<Guid, List<Term>> correctedKeyTerms;
  private Dictionary<Guid, List<Term>> nauseaResult;

  public Dictionary<Guid, List<Term>> KeyTerms
  {
   get
   {
    return this.keyTerms;
   }
  }

  public Dictionary<Guid, List<Term>> CompresedKeyTerms
  {
   get
   {
    return this.compresedKeyTerms;
   }
  }

  public Dictionary<Guid, List<Term>> CorrectedKeyTerms
  {
   get
   {
    return this.correctedKeyTerms;
   }
  }

  public Dictionary<Guid, List<Term>> KeyTermAfterAbsorption
  {
   get
   {
    return this.keyTermAfterAbsorption;
   }
  }

  public Dictionary<Guid, List<Term>> KeyTermMaskResult
  {
   get
   {
    return this.keyTermMaskResult;
   }
  }

  public Dictionary<Guid, List<Term>> NauseaResult
  {
   get
   {
    return this.nauseaResult;
   }
  }

  public IDocument AnalizedDocument => this.document;

  public List<Word> UnknownWords => this.wordsInfo.Values.Where(x =>
!x.IsFromDb).ToList();
  #endregion

  public SearchManager(IMaskValidator maskValidator, IEvaluator
evaluator, IDocument document, SearchSetting searchSetting, int
wordCountInTerm)
  {
   if (document == null)
    throw new ArgumentNullException(nameof(document));
   if (evaluator == null)
    throw new ArgumentNullException(nameof(evaluator));
   if (maskValidator == null)
    throw new ArgumentNullException(nameof(maskValidator));
   if (searchSetting == null)
    throw new ArgumentNullException(nameof(searchSetting));

   this.evaluator = evaluator;
   this.searchSetting = searchSetting;
   this.wordCountInTerm = wordCountInTerm;
   this.maskValidator = maskValidator;
   this.wordsInfo = new Dictionary<string, Word>();

   this.document = new Document
   {
    Name = document.Name,
    Sections = new List<ISection>()
   };

   foreach (var section in document.Sections)
   {
    this.document.Sections.Add(new NewSection(section));
   }
  }

  public void Search()
  {
   this.keyTerms = new Dictionary<Guid, List<Term>>();
   this.keyTermMaskResult = new Dictionary<Guid, List<Term>>();
   this.compresedKeyTerms = new Dictionary<Guid, List<Term>>();
   this.correctedKeyTerms = new Dictionary<Guid, List<Term>>();
   this.keyTermAfterAbsorption = new Dictionary<Guid, List<Term>>();
   this.nauseaResult = new Dictionary<Guid, List<Term>>();

   foreach (var section in this.document.Sections)
   {
    SearchSubSection(section as INewSection);
   }
  }

  private void SearchSubSection(INewSection section)
  {
   foreach (var subSection in section.Sections)
   {
    SearchSubSection(subSection as INewSection);
   }

   DetermineKeyword(section);
   SetWordInfo(section);
   FilterTerm(section);
   CompressTerms(section);
   CompactifyTerm(section);
   CorrectEvaliation(section, this.searchSetting.CorrectionSetting);
   NauseaTerm(section);
  }

  private void DetermineKeyword(INewSection section)
  {
   List<Term> keyTerms = new List<Term>();
   for (int i = 1; i <= this.wordCountInTerm; i++)
   {
    List<Term> result = this.evaluator.Evaluate(section, i);
    keyTerms.AddRange(result);
   }

   this.keyTerms.Add(section.Guid, keyTerms);

   section.KeyTerms = keyTerms;
  }

  private void SetWordInfo(INewSection section)
  {
   List<Word> allWords = new List<Word>();
   foreach (var term in section.KeyTerms)
   {
    allWords.AddRange(term.Words);
   }

   List<string> uniqueWords = allWords
    .ToLookup(word => word.Text)
    .Select(a => a.First().Text)
    .ToList();

   using (DbProvider dbProvider = new DbProvider())
   {
    Dictionary<string, Word> newWordsInfo =
dbProvider.GetWordsInfo(uniqueWords.Except(this.wordsInfo.Keys).ToList
());
    foreach (var word in newWordsInfo)
    {
     this.wordsInfo.Add(word.Key, word.Value);
    }
   }

   foreach (var term in section.KeyTerms)
   {
    foreach (var word in term.Words)
    {
     Word wordInfo = this.wordsInfo[word.Text];

     word.IsFromDb = true;
     word.Infinitive = wordInfo.Infinitive;
     word.GrammaticalGender = wordInfo.GrammaticalGender;
     word.GrammaticalCase = wordInfo.GrammaticalCase;
     word.SpeechPart = wordInfo.SpeechPart;
    }
   }

  private void FilterTerm(INewSection section)
  {
   section.KeyTerms = this.maskValidator.Validate(section.KeyTerms);
   this.keyTermMaskResult.Add(section.Guid, section.KeyTerms);
  }

  private void CompressTerms(INewSection section)
  {
   List<Term> compresedKeyTerm = new List<Term>();
   List<string> uniqueInfinitive = section.KeyTerms.Select(x =>
x.Infinitive).Distinct().ToList();

   foreach (var infinitive in uniqueInfinitive)
   {
    var keyTermForms = section.KeyTerms.Where(x => x.Infinitive ==
infinitive).ToList();
    double maxEvaluation = keyTermForms.Max(x => x.Evaluation);
    if (maxEvaluation == 0)
    {
     continue;
    }

    int count = keyTermForms.Sum(x => x.Count);
    Term readableForm = GetReadableKeyTermForm(keyTermForms);
    if (readableForm == null)
    {
     readableForm = keyTermForms.First(x => x.Evaluation ==
maxEvaluation);
    }

    readableForm.AllExistedForm = keyTermForms;
    readableForm.Count = count;
    readableForm.Evaluation = maxEvaluation;
    readableForm.Words = readableForm.Words;

    compresedKeyTerm.Add(readableForm);
   }

   section.KeyTerms = compresedKeyTerm;
   this.compresedKeyTerms.Add(section.Guid, compresedKeyTerm);
  }

  private void CompactifyTerm(INewSection section)
  {
   List<Term> termsToSave = new List<Term>();

   for (int i = 1; i < this.wordCountInTerm; i++)
   {
    var smallTerms = section.KeyTerms.Where(t => t.Words.Count == i);
    var bigTerms = section.KeyTerms.Where(t => t.Words.Count == i +
1);

    foreach (var smallTerm in smallTerms)
    {
     var hasBigVersion = bigTerms.FirstOrDefault(t =>
      new
Regex($"^{smallTerm.Infinitive}\\s|\\s{smallTerm.Infinitive}$").IsMatc
h(t.Infinitive)
```

```
        && smallTerm.Count <= t.Count * 2);

      if (hasBigVersion == null)
      {
       termsToSave.Add(smallTerm);
      }
     }
    }

    section.KeyTerms = termsToSave;
    this.keyTermAfterAbsorption.Add(section.Guid, termsToSave);
   }

   private void CorrectEvaliation(INewSection section,
CorrectionSetting correctionSetting)
   {
    List<Term> correctedKeyTerm = new List<Term>();

    foreach (var term in section.KeyTerms)
    {
     bool isBold = term.Words.Any(word => word.WordStyle?.Bold !=
null);
     bool isItalic = term.Words.Any(word => word.WordStyle?.Italic !=
null);
     bool isUnderline = term.Words.Any(word =>
word.WordStyle?.Underline != null);

     if (isBold)
      term.Evaluation *= correctionSetting.BoldFontCoefficient;

     if (isItalic)
      term.Evaluation *= correctionSetting.ItalicFontCoefficient;

     if (isUnderline)
      term.Evaluation *= correctionSetting.UnderlineFontCoefficient;

     correctedKeyTerm.Add(term);
    }

    this.correctedKeyTerms.Add(section.Guid, correctedKeyTerm);
   }

   private void NauseaTerm(INewSection section)
   {
    int termCount = (int)(section.KeyTerms.Count * 0.15);

    List<Term> nausea = section.KeyTerms
     .OrderByDescending(x => x.Evaluation)
     .Take(termCount)
     .ToList();

    section.KeyTerms = nausea;
    this.nauseaResult.Add(section.Guid, nausea);
   }

   private Term GetReadableKeyTermForm(List<Term> options)
   {
    Term firstTerm = options.First();

    if (firstTerm.Words.Count == 1)
    {
     return new Term(firstTerm.Infinitive)
     {
      Words = firstTerm.Words
     };
    }

    foreach (var keyTerm in options)
    {
     var selectedKeyTerm = keyTerm.Words.FirstOrDefault(x =>
x.SpeechPart == ESpeechPart.Noun && x.Text == x.Infinitive);
     if (selectedKeyTerm != null)
     {
      return keyTerm;
     }
    }

    return null;
   }
  }
}
```

TermSearcherFactory.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Searcher.Settings;
using AIS.TermSearcher.Searcher.TermMaskValidation;
using AIS.TermSearcher.Searcher.TermValuation;

namespace AIS.TermSearcher.Searcher
{
 public static class TermSearcherFactory
 {
  public static ISearchManager CreateSearchManager(IDocument document,
int wordInTermCount, SearchSetting searchSetting)
  {
   VarianceEvaluator varianceEvaluator = new VarianceEvaluator();
   UkranianMaskValidator mask = new UkranianMaskValidator();

   return new SearchManager(mask, varianceEvaluator, document,
searchSetting, wordInTermCount);
  }
 }
}
```

NewSection.cs

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

namespace AIS.TermSearcher.Entities.Section
{
 public class NewSection : DocxReader.Entities.Section.Section,
INewSection
 {
  public List<Term.Term> KeyTerms { get; set; }

  public NewSection() { }

  public NewSection(ISection section)
  {
   Level = section.Level;
   Text = section.Text;
   Name = section.Name;
   Guid = section.Guid;
   Paragraphs = section.Paragraphs;
   Sections = new List<ISection>();

   if (section.Sections != null && section.Sections.Any())
   {
    foreach (var item in section.Sections)
    {
     Sections.Add(new NewSection(item));
    }
   }
  }
 }
}
```

Term.cs

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

namespace AIS.TermSearcher.Entities.Term
{
 public class Term
 {
  private string text;
  private string infinitive;

  public string Text
  {
   get
   {
    if (string.IsNullOrEmpty(this.text))
    {
     this.text = string.Join(" ", this.Words.Select(x => x.Text));
    }

    return this.text;
   }
  }

  public string Infinitive
  {
   get
   {
    if (string.IsNullOrEmpty(this.infinitive))
    {
     this.infinitive = string.Join(" ", this.Words.Select(x =>
x.Infinitive));
    }

    return this.infinitive;
   }
  }

  public double Evaluation { get; set; }

  public List<Term> AllExistedForm { get; set; }

  public List<Word.Word> Words { get; set; }

  public int Count { get; set; }

  public int Position { get; set; }

  public Term()
  {

  }

  public Term(string text)
  {
   this.text = text;
  }
 }
}
```

EGrammaticalCase.cs

```
using System.ComponentModel;

namespace AIS.TermSearcher.Entities.Word
{
 public enum EGrammaticalCase
 {
  /// <summary>Default</summary>
  [Description("None")]
  None,
  /// <summary>Називний</summary>
  [Description("Називний")]
  Nominative,
  /// <summary>Родовий</summary>
  [Description("Родовий")]
  Genitive,
  /// <summary>Давальний</summary>
  [Description("Давальний")]
  Dative,
  /// <summary>Знахідний </summary>
  [Description("Знахідний ")]
  Accusative,
  /// <summary>Орудний </summary>
  [Description("Орудний ")]
  Instrumental,
  /// <summary>Місцевий</summary>
  [Description("Місцевий")]
  Prepositional,
  /// <summary>Кличний</summary>
  [Description("Кличний")]
  Vocative
 }
}
```

EGrammaticalGender.cs

```
using System.ComponentModel;

namespace AIS.TermSearcher.Entities.Word
{
 public enum EGrammaticalGender
 {
  /// <summary>Default</summary>
  [Description("Відсутній")]
  None,
  /// <summary>Чоловічий</summary>
  [Description("Чоловічий")]
  Masculine,
  /// <summary>Жіночий</summary>
  [Description("Жіночий")]
  Feminine,
  /// <summary>Середній</summary>
  [Description("Середній")]
  Neuter
 }
}
```

ESpeechPart.cs

```
using System.ComponentModel;

namespace AIS.TermSearcher.Entities.Word
{
 public enum ESpeechPart
 {
  /// <summary>Default</summary>
  [Description("None")]
  None,
  /// <summary>Іменник</summary>
  [Description("Іменник")]
  Noun,
  /// <summary>Прикметник</summary>
  [Description("Прикметник")]
  Adjective,
  /// <summary>Числівник</summary>
  [Description("Числівник")]
  Numeral,
  /// <summary>Займенник</summary>
  [Description("Займенник")]
  Pronoun,
  /// <summary>Дієслово</summary>
  [Description("Дієслово")]
  Verb,
  /// <summary>Дієприкметник</summary>
  [Description("Дієприкметник")]
  Participle,
  /// <summary>Дієприслівник</summary>
  [Description("Дієприслівник")]
  Transgressive,
  /// <summary>Прислівник</summary>
  [Description("Прислівник")]
  Adverb,
  /// <summary>Сполучник</summary>
  [Description("Сполучник")]
  Conjunction,
  /// <summary>Прийменник</summary>
  [Description("Прийменник")]
  Preposition,
  /// <summary>Частка</summary>
  [Description("Частка")]
  Particle,
 }
}
```

Word.cs

```
using AIS.DocxReader.Entities.Word;

namespace AIS.TermSearcher.Entities.Word
{
 public class Word : IWord
 {
  public string Text { get; set; }

  public string Infinitive { get; set; }

  public IWordStyle WordStyle { get; set; }

  public bool IsFromDb { get; set; }

  public int Position { get; set; }

  public EGrammaticalGender GrammaticalGender { get; set; }

  public EGrammaticalCase GrammaticalCase { get; set; }

  public ESpeechPart SpeechPart { get; set; }
 }
}
```

CorrectionSetting.cs

```
namespace AIS.TermSearcher.Searcher.Settings
{
 public class CorrectionSetting
 {
  public float BoldFontCoefficient { get; set; }

  public float ItalicFontCoefficient { get; set; }

  public float UnderlineFontCoefficient { get; set; }
 }
}
```

SearchSetting.cs

```
using AIS.TermSearcher.Searcher.Settings;

namespace AIS.TermSearcher.Searcher.Settings
{
 public class SearchSetting
 {
  public CorrectionSetting CorrectionSetting { get; set; }
 }
}
```

DbProvider.cs

```
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using static AIS.TermSearcher.Entities.EntitiesHelper;

namespace AIS.TermSearcher.Searcher.TermMaskValidation
{
 public class DbProvider : IDisposable
 {
  private readonly SqlConnection connection;

  public DbProvider()
  {
   string connectionString = @"Data Source=DESKTOP-
3KQK1PR\SQLEXPRESS2016;Initial
Catalog=TESTWORDSSQL;Trusted_Connection=true";
   this.connection = new SqlConnection(connectionString);
   this.connection.Open();
  }

  public Dictionary<string, Word> GetWordsInfo(List<string> words)
  {
   Dictionary<string, Word> wordsInfo = new Dictionary<string,
Word>();

   if (words.Any())
   {
    string query = MakeWordsQuery(words);
    SqlCommand command = new SqlCommand(query, this.connection);

    using (SqlDataReader reader = command.ExecuteReader())
    {
     while (reader.Read())
     {
      string key = reader["Word"].ToString().ToLower();
      if (wordsInfo.ContainsKey(key))
      {
       continue;
      }

      Word word = new Word
      {
       IsFromDb = true,
       Infinitive = reader["Infinitive"].ToString(),
       GrammaticalGender =
GetGrammaticalGender(reader["GrammaticalGender"].ToString()),
       GrammaticalCase =
GetGrammaticalCase(reader["GrammaticalCase"].ToString()),
       SpeechPart = GetSpeechPart(reader["SpeechPart"].ToString())
      };
      wordsInfo.Add(key, word);
     }
    }

    List<string> unknownWords = words.Except(wordsInfo.Keys).ToList();
    foreach (var word in unknownWords)
    {
     wordsInfo.Add(word, new Word()
     {
      IsFromDb = false,
      Text = word,
      GrammaticalGender = EGrammaticalGender.None,
      GrammaticalCase = EGrammaticalCase.None,
      SpeechPart = ESpeechPart.None
     });
    }
   }

   return wordsInfo;
  }

  private string MakeWordsQuery(List<string> words)
  {
   string wordsQuery = string.Format(SqlQueryResources.Words,
words.First().ToLower().Replace("'", "''"));

   foreach (var word in words.Skip(1))
   {
    wordsQuery += " OR " + string.Format(SqlQueryResources.Words,
word.ToLower().Replace("'", "''"));
   }

   return string.Format(SqlQueryResources.GetWordInfo, wordsQuery);
  }

  private EGrammaticalGender GetGrammaticalGender(string
grammaticalGender)
  {
   return GrammaticalGenderMapping[grammaticalGender.ToLower()];
  }

  private ESpeechPart GetSpeechPart(string speechPart)
  {
   speechPart = speechPart.ToLower();

   if (SpeechPartMapping.ContainsKey(speechPart))
   {
    return SpeechPartMapping[speechPart];
   }
   else
   {
    return ESpeechPart.None;
   }
  }

  private EGrammaticalCase GetGrammaticalCase(string grammaticalCase)
  {
   grammaticalCase = grammaticalCase.ToLower();

   if (GrammaticalCaseMapping.ContainsKey(grammaticalCase))
   {
    return GrammaticalCaseMapping[grammaticalCase];
   }
   else
   {
    return EGrammaticalCase.None;
   }
  }

  public void Dispose()
  {
   this.connection.Close();
  }
 }
}
```

UkranianMaskValidator.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;

namespace AIS.TermSearcher.Searcher.TermMaskValidation
{
 public class UkranianMaskValidator : IMaskValidator
 {
  public List<Term> Validate(List<Term> terms)
  {
   List<Term> resultTerm = new List<Term>();

   foreach (var term in terms)
   {
    int wordCount = term.Words.Count;

    if (!term.Words.Any(word => word.SpeechPart == ESpeechPart.Noun))
     continue;

    if (term.Words.First().SpeechPart != ESpeechPart.Noun &&
term.Words.First().SpeechPart != ESpeechPart.Adjective)
     continue;

    if (term.Words.Last().SpeechPart != ESpeechPart.Noun &&
term.Words.Last().SpeechPart != ESpeechPart.Adjective)
     continue;

    int n = 0;
    for (int i = 0; i < wordCount; i++)
    {
     if (term.Words[i].SpeechPart == ESpeechPart.Noun
      || term.Words[i].SpeechPart == ESpeechPart.Adjective
      || term.Words[i].SpeechPart == ESpeechPart.Conjunction
      || term.Words[i].SpeechPart == ESpeechPart.Particle)
     {
      n++;
     }
    }

    if (n == wordCount)
    {
     resultTerm.Add(term);
    }
   }

   return resultTerm;
  }
 }
}
```

VarianceEvaluator.cs

```
using AIS.TermSearcher.Entities.Section;
using AIS.TermSearcher.Entities.Term;
using AIS.TermSearcher.Entities.Word;
using System;
using System.Collections.Generic;
using System.Linq;

namespace AIS.TermSearcher.Searcher.TermValuation
{
 public class VarianceEvaluator : IEvaluator
 {
  public List<Term> Evaluate(INewSection section, int wordCount)
  {
   List<Term> resultValuation = new List<Term>();
   List<Term> termsCondidates = GetTermsCandidates(section,
wordCount);

   if (!termsCondidates.Any())
   {
    return resultValuation;
   }

   int lastTermPosition = termsCondidates.Last().Position;
   termsCondidates.Sort((term1, term2) => string.Compare(term1.Text,
term2.Text));

   List<Term> uniqueTermsCondidates = termsCondidates
    .ToLookup(term => term.Text)
    .Select(a => a.First())
    .ToList();

   foreach (var uniqueTerm in uniqueTermsCondidates)
   {
    double evaluation;
    int sumDist = 0, sumDistSquer = 0;

    List<Term> selectedTerms = BinariSearch(termsCondidates,
uniqueTerm.Text);
    selectedTerms = selectedTerms.OrderBy(a => a.Position).ToList();

    int termCount = selectedTerms.Count;
    if (termCount == 1)
    {
     evaluation = 0;
    }
    else
    {
     for (int j = 0; j < selectedTerms.Count - 1; j++)
     {
      sumDist += (selectedTerms[j + 1].Position -
selectedTerms[j].Position) - wordCount;
      sumDistSquer += (int)Math.Pow((selectedTerms[j + 1].Position -
selectedTerms[j].Position) - wordCount, 2);
     }

     sumDist += (lastTermPosition - selectedTerms.Last().Position +
selectedTerms.First().Position) - wordCount;
     sumDistSquer += (int)Math.Pow((lastTermPosition -
selectedTerms.Last().Position + selectedTerms.First().Position) -
wordCount, 2);

     evaluation = Math.Sqrt(((double)sumDistSquer /
(selectedTerms.Count)) - Math.Pow(((double)sumDist /
```

```
(selectedTerms.Count)), 2)) / ((double)sumDist /
(selectedTerms.Count));
    }

    if (evaluation > 0)
    {
     resultValuation.Add(new Term()
     {
      Evaluation = evaluation,
      Count = selectedTerms.Count,
      Words = uniqueTerm.Words
     });
    }
   }

   return resultValuation.OrderByDescending(p =>
p.Evaluation).ToList();
  }

  public List<Term> GetTermsCandidates(INewSection section, int
wordCount)
  {
   List<Word> words;
   List<Term> termsCondidates = new List<Term>();
   int position = 0;

   foreach (var paragraph in section.AllParagraphs)
   {
    foreach (var sentences in paragraph.Sentences)
    {
     foreach (var textContainer in sentences.TextContainers)
     {
      words = textContainer.Words.Select(word => new Word()
      {
       Text = word.Text.ToLower(),
       WordStyle = word.WordStyle
      }).ToList();

      for (int i = 0; i < words.Count; i++)
      {
       Term term = new Term()
       {
        Words = new List<Word>(),
        Position = position
       };

       for (int j = 0; j < wordCount && i + j < words.Count; j++)
       {
        term.Words.Add(words[i + j]);
       }

       if (term.Words.Count == wordCount)
       {
        termsCondidates.Add(term);
       }

       position++;
      }
     }
    }
   }

   return termsCondidates;
  }

  List<Term> BinariSearch(List<Term> allKeywordOptions, string value)
  {
   List<Term> option = new List<Term>();
   int start = 0, end = allKeywordOptions.Count;
   while (start < end)
   {
    int center = (start + end) / 2;
    switch (String.Compare(value, allKeywordOptions[center].Text))
    {
     case -1:
      end = center;
      break;
     case 0:
      int increment = 0;
      bool left = true, rigth = true;
      while (left || rigth)
      {
       if (left && (center - increment < 0 || allKeywordOptions[center
- increment].Text != value))
        left = false;
       if (rigth && (center + increment + 1 >= allKeywordOptions.Count
|| allKeywordOptions[center + increment + 1].Text != value))
        rigth = false;
       if (left)
        option.Add(allKeywordOptions[center - increment]);
       if (rigth)
        option.Add(allKeywordOptions[center + increment + 1]);
       increment++;
      }

      return option;
     case 1: start = center + 1; break;
    }
   }

   return option;
  }
 }
}
```

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>())
  {
  }
 }
}
```

SentencesPart.cs

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

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; }
 }
}
```

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;
  }
 }
}
```

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);
  }
 }
}
```

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();
  }
 }
}
```

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(wrongAnswerMask));

   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(rightAnswerMask));

   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; }
 }
}
```

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; }
 }
}
```

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(endPosition,
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(endPosition,
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(endPosition),
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;
  }
 }
}
```

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
   };
  }
 }
}
```

ETestType.cs

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

TestTask.cs

```
using AIS.TermSearcher.Entities.Term;
using AIS.TestGenerator.Masks;
using AIS.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);
  }
 }
}
```

TestTaskWorkpiece.cs

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

namespace AIS.TestGenerator.TestTask
{
 public 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; }
 }
}
```

```
TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs


TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs


TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs


TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs


TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs


TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs
```

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
{
 public 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.EndSentence];
    }
    else
    {
     tagValue = taskWorkpiece.SentencesParts[MaskTag.BeginSentence];
    }
   }
   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.SelectSingleNode(xpath) != null
                 && x.KeyTerm.Text !=
taskWorkpiece.KeyTerm.Text).ToList();

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

    if (filteredWorkpiece.Count > 0)
    {
     int randomIndex = random.Next(filteredWorkpiece.Count);
     tagValue =
filteredWorkpiece[randomIndex].QuestionParts[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.QuestionMask.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.QuestionMask.ChildNodes)
   {
    string tagValue = GetMaskTagValue(tag, taskWorkpiece);
    taskWorkpiece.QuestionParts.Add(tag.Name, tagValue);
   }
  }
 }
}
```

InputAnswerTestBuilder.cs

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

namespace AIS.TestGenerator.TestBuilders.Builders
{
 public 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.RightAnswerMask.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)
   };
  }
 }
}
```

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
{
 public class MultipleChoiceBuilder : BaseTaskBuilder
 {
  public MultipleChoiceBuilder() : base(ETestType.MultipleChoice)
  {

  }

  protected override List<AnswerOption>
CreateAnswers(TestTaskWorkpiece taskWorkpiece)
  {
   List<AnswerOption> answers = new List<AnswerOption>();
   answers.AddRange(CreateRightAnswers(taskWorkpiece));
   answers.AddRange(CreateWrongAnswers(taskWorkpiece));
   return answers;
  }
```

```
  private List<AnswerOption> CreateRightAnswers(TestTaskWorkpiece
taskWorkpiece)
  {
   List<AnswerOption> answers = new List<AnswerOption>();
   int answerMask =
taskWorkpiece.GeneratedByTaskMask.RightAnswerMask.ChildNodes.Count;
   if (answerMask > 1 ||
taskWorkpiece.GeneratedByTaskMask.RightAnswerMask.ChildNodes[0].Name
!= MaskTag.SplitedList)
    {
     throw new Exception($"Mask
'{taskWorkpiece.GeneratedByTaskMask.RightAnswerMask.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.WrongAnswerMask.ChildNodes.Count;
   if (answerMask > 1 ||
taskWorkpiece.GeneratedByTaskMask.RightAnswerMask.ChildNodes[0].Name
!= MaskTag.SplitedList)
    {
     throw new Exception($"Mask
'{taskWorkpiece.GeneratedByTaskMask.RightAnswerMask.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].SentencesParts[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;
   }
  }
}
```

App.xaml

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

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

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

DialogFactory.cs

```
using AIS.DocxReader.Entities.Document;
using AIS.TermSearcher.Entities.Word;
using AIS.TestGenerator.UI.ViewModels;
using AIS.TestGenerator.UI.Views;
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);
   TestGenerationResult view = new TestGenerationResult();
   view.DataContext = viewModel;

   return view;
  }
 }
}
```

MainWindow.xaml

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

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:gif="http://wpfanimatedgif.codeplex.com"
        xmlns:local="clr-namespace:AIS.TestGenerator.UI"
        mc:Ignorable="d"
        Title="Параметри" 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/Style/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" IsReadOnly="True"
Text="{Binding SelectedFilePath}"/>
    <Button Grid.Row="0" Grid.Column="2" Content="Вибрати"
Command="{Binding BrowseFiles}"/>
   </Grid>
   <GroupBox Grid.Row="1" Header="Коефіцієнти">
    <ContentControl Content="{Binding StyleCoefficientViewModel}"/>
   </GroupBox>
   <Button Grid.Row="3" Width="80" 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="Resources/loading.gif"/>
   </Grid>
  </Border>
 </Grid>
</Window>
```

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();
  }
 }
}
```

BindableRichTextBox.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(DocumentProperty);
   }

   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;
   }
  }
 }
}
```

BooleanToVisibilityConverter.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();
  }
 }
}
```

NullToVisibilityConverter.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();
  }
 }
}
```

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;
  }
 }
}
```

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();
  }
 }
}
```

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();
  }
 }
}
```

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;
```

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();
```

```
  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(createdChildSections);
   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(StringResources.Space);
   }
   else if (container.StartSymbolType == ESymbolType.OpenBracket ||
container.StartSymbolType == ESymbolType.Quotation)
   {
    textContainerBuilder.Append(container.StartSymbol);
   }

   textContainerBuilder.Append(container.Text);
   if (container.EndSymbolType == ESymbolType.Hyphen)
   {
    textContainerBuilder.Append(StringResources.Space);
   }

   if (container.EndSymbolType != ESymbolType.OpenBracket ||
container.StartSymbolType != ESymbolType.Quotation)
   {
    textContainerBuilder.Append(container.EndSymbol);
   }

   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;
  }
 }
}
```

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(GetChildSectionList(childSection
as ISectionContainTestTask));
   }
  }

  private List<TestTasksSectionViewModel>
GetChildSectionList(ISectionContainTestTask currentSection)
  {
   List<TestTasksSectionViewModel> sections = new
List<TestTasksSectionViewModel>();
   TestTasksSectionViewModel currentTestStructure = new
TestTasksSectionViewModel
   {
    Name = currentSection.Name,
    Tasks = new List<TestTaskEditorViewModel>()
   };

   sections.Add(currentTestStructure);

currentTestStructure.Tasks.AddRange(currentSection.TestTasks.Select(ta
sk => new TestTaskEditorViewModel(task, false)));
   foreach (var childSection in currentSection.Sections)
   {
    sections.AddRange(GetChildSectionList(childSection as
ISectionContainTestTask));
   }

   return sections;
  }
 }
}
```

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;
   }
  }
 }
}
```

GenerationResultViewModel.cs

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

namespace AIS.TestGenerator.UI.ViewModels
{
 class GenerationResultViewModel : BaseViewModel
 {
  private readonly IDocument document;
  private readonly List<Word> unKnownWords;
  public GenerationResultViewModel(IDocument document, List<Word>
unKnownWords)
  {
   if (document == null) throw new
ArgumentNullException(nameof(document));

   this.document = document;
   this.unKnownWords = unKnownWords;

   this.showGeneratedTests = new RelayCommand(OnShowGeneratedTests);
   this.showUnknownWords = new RelayCommand(OnShowUnknownWords);
   this.showTestCoverage = new RelayCommand(OnShowTestCoverage);
   this.showFinalTaskList = new RelayCommand(OnShowFinalTaskList);
   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; }
  }
  #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);
  }
 }
}
```

MainWindowViewModel.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(dialogFactory));
   }

   this.dialogFactory = dialogFactory;

   this.startTestGenerator = new RelayCommand(StartGeneration, () =>
!string.IsNullOrEmpty(this.SelectedFilePath));
   this.browseFiles = new RelayCommand(OnBrowseFiles);

   this.styleCoefficientViewModel = new
StyleCoefficientEditorViewModel();
   this.masksManager =
TestGeneratorFactory.CreateMaskManager("Resources/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.SelectedFilePath));
   }
  }

  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 message = "";
   await System.Threading.Tasks.Task.Factory.StartNew(() =>
   {
    this.IsBusy = true;
    SearchSetting searchSetting = new SearchSetting
    {
     CorrectionSetting = new CorrectionSetting()
     {
      BoldFontCoefficient =
(float)this.styleCoefficientViewModel.Bold,
      ItalicFontCoefficient =
(float)this.styleCoefficientViewModel.Italic,
      UnderlineFontCoefficient =
(float)this.styleCoefficientViewModel.Underline
     }
    };

    DocxReader.DocxReader reader = new
DocxReader.DocxReader(this.SelectedFilePath);
    IDocument document = reader.Read();

    searchManager = TermSearcherFactory.CreateSearchManager(document,
5, searchSetting);
    searchManager.Search();

    testGenerator =
TestGeneratorFactory.CreateTestGenerator(searchManager.AnalizedDocumen
t, searchManager.NauseaResult, masksManager);
    testGenerator.GenerateParallel();
   });

    var unknownWord = searchManager.UnknownWords;
    this.IsBusy = false;
    var dialog =
this.dialogFactory.CreateTestGenerationResult(searchManager.AnalizedDo
cument, unknownWord);
    dialog.ShowDialog();
  }

  private void OnBrowseFiles()
  {
   OpenFileDialog openFileDialog = new OpenFileDialog();
   openFileDialog.Filter = Properties.Resources.FileFilter;
   if (openFileDialog.ShowDialog() == true)
   {
    this.SelectedFilePath = openFileDialog.FileName;
   }
  }
 }
}
```

StyleCoefficientEditorViewModel.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
 }
```

```
}
```

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;
   }
  }
 }
}
```

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.TestTask, bool> predicate)
  {
   List<TestTask.TestTask> result = new List<TestTask.TestTask>();
   foreach (var section in document.Sections)
   {
    result.AddRange(GetTestFromSectionByPredicate(section as
ISectionContainTestTask, predicate));
   }

   return result;
  }

  private List<TestTask.TestTask>
GetTestFromSectionByPredicate(ISectionContainTestTask section,
Func<TestTask.TestTask, bool> predicate)
  {
   List<TestTask.TestTask> testTasks =
section.TestTasks.Where(predicate).ToList();
   foreach (var childSection in section.Sections)
   {
    testTasks.AddRange(GetTestFromSectionByPredicate(childSection as
ISectionContainTestTask, predicate));
   }

   return testTasks;
  }
 }
}
```

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.currentTestTask.AnswerOptions)
;

   this.clonedTestTask = (TestTask.TestTask)testTask.Clone();

   this.AddAnswer = new RelayCommand(AddNewAnswer);
   this.DeleteAnswer = new
RelayCommand<object>(DeleteAnswerImplementation);
   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.isEditMode, 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(answerOptionToDelete);
   }
  }

  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;
  }
 }
}
```

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>(this.selectedDocumentSection.T
estTasks
                .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 =
GetUsedDocumentRegionForGenerateSelectedTest();
      }
      else
      {
       this.TestEditorViewModel = null;
       this.GeneratedTestDocumentRegion = null;
      }
    }

    private void TestTaskEditorViewModel_EditModeChanged(object sender,
EventArgs e)
    {
     if (sender is bool)
      {
       this.IsEditMode = (bool)sender;
      }
    }

    private void TestTaskEditorViewModel_SaveTestTask(object sender,
EventArgs e)
    {
     var currentSelectedTestTask = this.SelectedTest;
     KeytermSelectionChanged();
     this.SelectedTest = currentSelectedTestTask;
    }

    private void EditTestTaskImplementation()
    {
     this.TestEditorViewModel.IsEditMode = true;
    }

    private void DeleteTestTaskImplementation()
    {
     this.selectedDocumentSection.TestTasks.Remove(this.SelectedTest);
     if (selectedDocumentSection is ISectionContainTestTask
castedSection)
     {
      this.SectionKeyterm = new
ObservableCollection<Term>(castedISection.KeyTerms);
     }

     KeytermSelectionChanged();
    }

    private FlowDocument GetUsedDocumentRegionForGenerateSelectedTest()
    {
     var usedDocumentParagraph =
this.selectedDocumentSection.AllParagraphs.First(x =>
x.Text.Contains(this.selectedTest.Sentences.Text));

     FlowDocument flowDocument = new FlowDocument();
     Paragraph paragraph = new Paragraph();
     ISentences lastSentences = usedDocumentParagraph.Sentences.Last();
     foreach (ISentences sentences in usedDocumentParagraph.Sentences)
     {
      string sentencesSeparator =
GetSentencesSeparator(sentences.SentencesType);
      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 ".";
     }
    }
}
```

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(section));
    }

    this.TestTasks = joinedTestTask;
   }

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

   private TestTask.TestTask selectedTestTask;
   public TestTask.TestTask SelectedTestTask
   {
    set
    {
     selectedTestTask = value;
     OnChangeProperty(nameof(TestEditorViewModel));
     OnChangeProperty(nameof(TestProperties));
    }
   }

   public TestTaskEditorViewModel TestEditorViewModel
   {
    get
    {
     TestTaskEditorViewModel result = null;
     if (selectedTestTask != null)
     {
      result = new TestTaskEditorViewModel(this.selectedTestTask,
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;
   }
  }
}
```

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; }
 }
}
```

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
 }
}
```

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.Interac
tivity"
  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="Результат генерації тестових завдань" 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>
```

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>
```

TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs

TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs

TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs

TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs

TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs

TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs

Converters.xaml

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

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:local="clr-
namespace:AIS.TestGenerator.UI.Views.Style"
                    xmlns:converters="clr-namespace:AIS.TestGenerator.UI.Converters">
 <converters:NullToVisibilityConverter
x:Key="nullToVisibilityConverter"/>
 <converters:BooleanToVisibilityConverter
x:Key="booleanToVisibilityConverter"/>
 <converters:ReverseBooleanToVisibilityConverter
x:Key="reverseBooleanToVisibilityConverter"/>
 <converters:ReverseBooleanConverter x:Key="reverseBooleanConverter"/>
 <converters:TestTasksNumberPerKeyTerm
x:Key="testTasksNumberPerKeyTerm" />
</ResourceDictionary>
```

ScrollBar.xaml

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

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:local="clr-
namespace:AIS.TestGenerator.UI.Views.Style">
 <!--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>
```

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>
```

GeneratedTestTasksView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.GeneratedTestTasksVie
w"

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>
```

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>
  <StackPanel Background="#ececec">
   <RadioButton 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 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 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 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>
  <ContentControl Grid.Column="1" Grid.Row="0" Content="{Binding
ContentViewModel}"/>
 </Grid>
</UserControl>
```

StyleCoefficientEditor.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.StyleCoefficientEdito
r"

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" Value="{Binding
Bold}"/>
  <wpfTool:DoubleUpDown Grid.Row="1" Grid.Column="1" Value="{Binding
Italic}"/>
  <wpfTool:DoubleUpDown Grid.Row="1" Grid.Column="2" Value="{Binding
Underline}"/>
 </Grid>
</UserControl>
```

TestGenerationResultView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.TestGenerationResultV
iew"

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>
```

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.Interac
tivity"
             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>
```

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"
      <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>
  <ContentControl Grid.Column="1" Grid.Row="0" Content="{Binding
ContentViewModel}"/>
 </Grid>
</UserControl>
```

StyleCoefficientEditor.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.StyleCoefficientEdito
r"

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" Value="{Binding
Bold}"/>
  <wpfTool:DoubleUpDown Grid.Row="1" Grid.Column="1" Value="{Binding
Italic}"/>
  <wpfTool:DoubleUpDown Grid.Row="1" Grid.Column="2" Value="{Binding
Underline}"/>
 </Grid>
</UserControl>
```

TestGenerationResultView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.TestGenerationResultV
iew"

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>
```

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.Interac
tivity"
             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>
```

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>
   <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>
 </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>
       <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>
```

TestTaskInStructure.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.TestTaskInStructure"

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:ie="clr-
namespace:System.Windows.Interactivity;assembly=System.Windows.Interac
tivity"
             xmlns:viewModel="clr-
namespace:AIS.TestGenerator.UI.ViewModels"
             xmlns:userControls="clr-
namespace:AIS.TestGenerator.UI.Views.UserControls"
             xmlns:controls="clr-
namespace:AIS.TestGenerator.UI.Controls"
             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.MergedDictionaries>
    <ResourceDictionary
Source="pack://application:,,,/Views/Style/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.Column="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.Column="2" HorizontalAlignment="Stretch"/>
    <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>
</UserControl>
```

TestTaskList.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.TestTaskList"

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>
  <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" Name="properties" IsReadOnly="True"
ClipboardCopyMode="ExcludeHeader" SelectionMode="Single"
SelectionChanged="DataGrid_SelectionChanged" 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>
```

UnknownWordsView.xaml

```
<UserControl
x:Class="AIS.TestGenerator.UI.Views.UserControls.UnknownWordsView"

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>
  <ResourceDictionary>
   <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.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>
```

Додаток В

Приклади тестових завдань, створених за розробленими в роботі зразками правил продукції

Таблиця В.1 – Приклад створення тестового завдання логічного типу

Фрагмент контенту, яким активовано антецедент правила продукції: `Транзакція – це послідовність операцій модифікації даних у БД, що переводить БД із одного несуперечливого стану в інший несуперечливий стан.`
Ключовий термін, знання якого перевіряється: «транзакція»

Використане правило продукції: $m_{1a1} = (a_1 \Rightarrow c_{a1})$.
Використаний зразок правила продукції:
`<Antecedent> [Text=0] [Term] [Connector] [Thesis]`
`<Consequence> {Header} [AnswerT1] [Term] [Connector] [Thesis] {Answer} {TRUE}`

Вигляд сформованого тестового завдання: Транзакція – це послідовність операцій модифікації даних у БД, що переводить БД із одного несуперечливого стану в інший несуперечливий стан ◉ Так ○ Ні

Таблиця В.2 – Приклад створення тестового завдання логічного типу

Фрагмент контенту, яким активовано антецедент правила продукції: `Модель даних – це деяка абстракція, що, будучи застосовна до конкретних даних, дозволяє користувачам і розроблювачам трактувати їх уже як інформацію, тобто відомості (зведення), що містять не тільки дані, але і взаємозв'язки між ними.`
Ключовий термін, знання якого перевіряється: «модель даних»

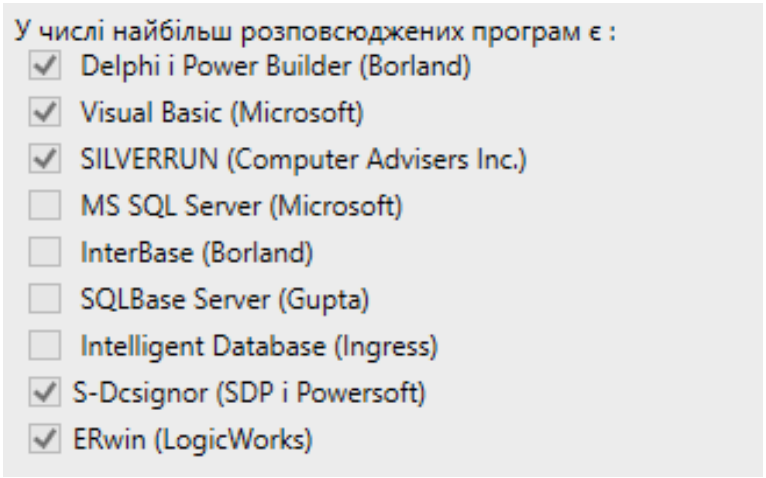
Використане правило продукції: $m_{1a2} = (a_1 \Rightarrow c_{a2})$.
Використаний зразок правила продукції:
`<Antecedent> [Text=0] [Term] [Connector] [Thesis]`
`<Consequence> {Header} [AnswerT1] [RandomTerm] [Connector] [Thesis] {Answer} {FALSE}`

Вигляд сформованого тестового завдання: Записи – це деяка абстракція, що, будучи застосовна до конкретних даних, дозволяє користувачам і розроблювачам трактувати їх уже як інформацію, тобто відомості (зведення), що містять не тільки дані, але і взаємозв'язки між ними ◉ Ні ○ Так

Таблиця В.3 – Приклад створення тестового завдання множинного вибору

Фрагмент контенту, яким активовано антецедент правила продукції: `У числі найбільш розповсюджених програм є: Delphi і Power Builder (Borland), Visual Basic (Microsoft), SILVERRUN (Computer Advisers Inc.), S-Dcsignor (SDP і Powersoft) і ERwin (LogicWorks).`
Ключовий термін, знання якого перевіряється: «програм»

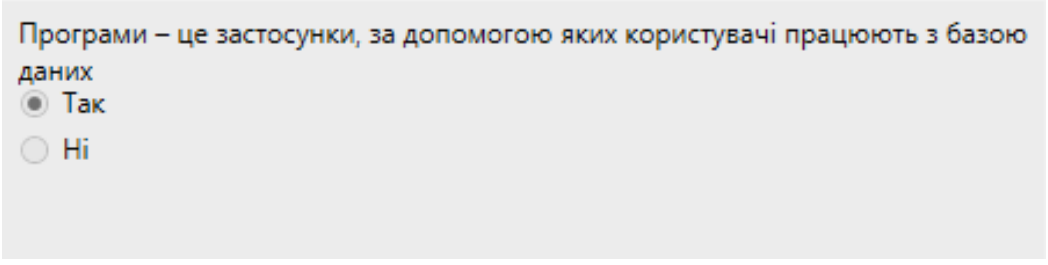
Використане правило продукції: $m_{2c2} = (a_2 \Rightarrow c_{c2})$.
Використаний зразок правила продукції:
`<Antecedent> [Text=0] [Term] [Text=0] [ConnectorMulti] [Counter_1] [Union] [Counter_2] [Union] [Counter_3=0] [Union] [Counter_4=0] [Union] [Counter_5=0] [NonCounter]`
`<Consequence> {Header} [Text=0] [Term] [Text=0] [ConnectorMulti] {Answer} {TRUE} [Counter_1] {TRUE} [Counter_2] {TRUE} [Counter_3=0] {TRUE} [Counter_4=0] {TRUE} [Counter_5=0] {FALSE} [RandomTerm] {FALSE} [RandomTerm] {FALSE} [RandomTerm] {FALSE} [RandomTerm]`

Вигляд сформованого тестового завдання: 

Таблиця В.4 – Приклад створення тестового завдання логічного типу

Фрагмент контенту, яким активовано антецедент правила продукції: `Програми – це застосунки, за допомогою яких користувачі працюють з базою даних.`
Ключовий термін, знання якого перевіряється: «програма»

Використане правило продукції: $m_{1a1} = (a_1 \Rightarrow c_{a1})$.
Використаний зразок правила продукції:
`<Antecedent> [Text=0] [Term] [Connector] [Thesis]`
`<Consequence> {Header} [AnswerT1] [Term] [Connector] [Thesis] {Answer} {TRUE}`

Вигляд сформованого тестового завдання: 

Таблиця В.5 – Приклад створення тестового завдання на введення тексту

Фрагмент контенту, яким активовано антецедент правила продукції: `Транзакція – це послідовність операцій модифікації даних у БД, що переводить БД із одного несуперечливого стану в інший несуперечливий стан.`
Ключовий термін, знання якого перевіряється: «транзакція»

Використане правило продукції: $m_{1d1} = (a_1 \Rightarrow c_{d1})$.
Використаний зразок правила продукції:
`<Antecedent> [Thesis] [Connector] [Term] [Text=0]`
`<Consequence> {Header} [AnswerT4-1] [Thesis] [Connector] {Answer} {TRUE} [Term]`

Вигляд сформованого тестового завдання: _____ – це послідовність операцій модифікації даних у БД, що переводить БД із одного несуперечливого стану в інший несуперечливий стан Варіанти відповіді: Транзакція

Таблиця В.6 – Приклад створення тестового завдання одиничного вибору

Фрагмент контенту, яким активовано антецедент правила продукції: `Транзакція – це послідовність операцій модифікації даних у БД, що переводить БД із одного несуперечливого стану в інший несуперечливий стан.`
Ключовий термін, знання якого перевіряється: «транзакція»

Використане правило продукції: $m_{1b2} = (a_1 \Rightarrow c_{b2})$.
Використаний зразок правила продукції:
`<Antecedent> [Text=0] [Term] [Connector] [Thesis]`
`<Consequence> {Header} [AnswerT2-2] [Term] [Connector] {Answer} {TRUE} [Thesis] {FALSE} [RandomThesis_1] {FALSE} [RandomThesis_2] {FALSE} [RandomThesis_3]`

Вигляд сформованого тестового завдання: Транзакція – це Застосунки, за допомогою яких користувачі працюють з базою даних Система спеціальним образом організованих даних – баз даних, програмних, технічних, мовних, організаційно-методичних засобів, призначених для забезпечення централізованого нагромадження й колективного багатоцільового використання даних Процес звертання користувача до БД із метою введення, одержання або зміни інформації в БД Послідовність операцій модифікації даних у БД, що переводить БД із одного несуперечливого стану в інший несуперечливий стан

Таблиця В.7 – Приклад створення тестового завдання на введення тексту

Фрагмент контенту, яким активовано антецедент правила продукції: `База даних – це іменована сукупність даних, що відбиває стан об’єктів і їхнє відношення в розглянутій предметній області.`
Ключовий термін, знання якого перевіряється: «`база даних`»

Використане правило продукції: $m_{1d1} = (a_1 \Rightarrow c_{d1})$.
Використаний зразок правила продукції:
`<Antecedent> [Text=0] [Term] [Connector] [Thesis]`
`<Consequence> {Header} [AnswerT4-1] [Thesis] [Connector] {Answer} {TRUE} [Term]`

Вигляд сформованого тестового завдання: _____ – це іменована сукупність даних, що відбиває стан об'єктів і їхнє відношення в розглянутій предметній області Варіанти відповіді: База даних

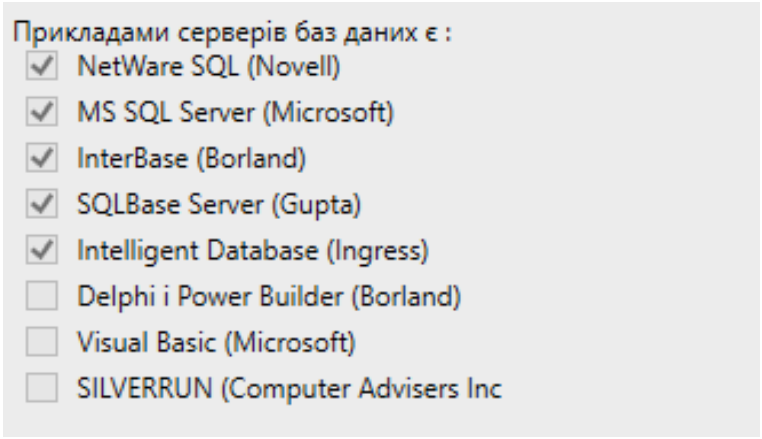
Таблиця В.8 – Приклад створення тестового завдання одиничного вибору

Фрагмент контенту, яким активовано антецедент правила продукції: `Програми – це застосунки, за допомогою яких користувачі працюють з базою даних.`
Ключовий термін, знання якого перевіряється: «`програми`»

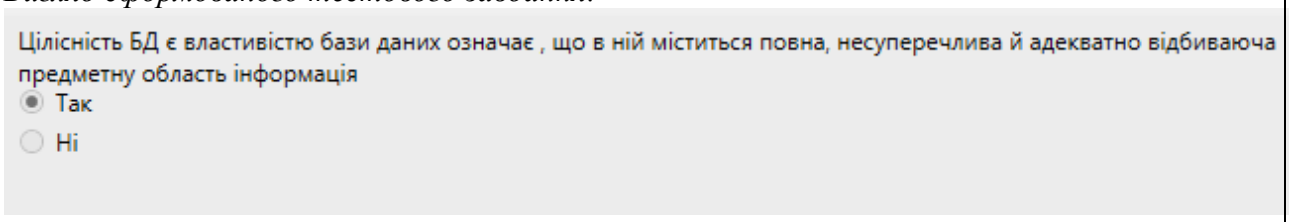
Використане правило продукції: $m_{1b2} = (a_1 \Rightarrow c_{b2})$.
Використаний зразок правила продукції:
`<Antecedent> [Text=0] [Term] [Connector] [Thesis]`
`<Consequence> {Header} [AnswerT2-2] [Term] [Connector] {Answer} {TRUE} [Thesis] {FALSE} [RandomThesis_1] {FALSE} [RandomThesis_2]`

Вигляд сформованого тестового завдання: Програми – це Система спеціальним образом організованих даних – баз даних, програмних, технічних, мовних, організаційно-методичних засобів, призначених для забезпечення централізованого нагромадження й колективного багатоцільового використання даних Точку зору на БД окремих додатків Застосунки, за допомогою яких користувачі працюють з базою даних

Таблиця В.9 – Приклад створення тестового завдання множинного вибору

Фрагмент контенту, яким активовано антецедент правила продукції: Прикладами серверів баз даних є: NetWare SQL (Novell), MS SQL Server (Microsoft), InterBase (Borland), SQLBase Server (Gupta), Intelligent Database (Ingress).
Ключовий термін, знання якого перевіряється: «баз даних»
Використане правило продукції: $m_{2c2} = (a_2 \Rightarrow c_{c2})$. *Використаний зразок правила продукції:* *<Antecedent>* [Text=0] [Term] [Text=0] [ConnectorMulti] [Counter_1] [Union] [Counter_2] [Union] [Counter_3=0] [Union] [Counter_4=0] [Union] [Counter_5=0] [NonCounter] *<Consequence>* {Header} [Text=0] [Term] [Text=0] [ConnectorMulti] {Answer} {TRUE} [Counter_1] {TRUE} [Counter_2] {TRUE} [Counter_3=0] {TRUE} [Counter_4=0] {TRUE} [Counter_5=0] {FALSE} [RandomTerm] {FALSE} [RandomTerm] {FALSE} [RandomTerm]
Вигляд сформованого тестового завдання: 

Таблиця В.10 – Приклад створення тестового завдання логічного типу

Фрагмент контенту, яким активовано антецедент правила продукції: Цілісність БД є властивістю бази даних, яка означає, що в ній міститься повна, несуперечлива й адекватно відбиваюча предметну область інформація.
Ключовий термін, знання якого перевіряється: «цілісність»
Використане правило продукції: $m_{2a1} = (a_2 \Rightarrow c_{a1})$. *Використаний зразок правила продукції:* *<Antecedent>* [Text=0] [Term] [Text=0] *<Consequence>* {Header} [AnswerT1] [Text=0] [Term] [Text=0] {Answer} {TRUE}
Вигляд сформованого тестового завдання: 

Таблиця В.11 – Приклад створення тестового завдання одиничного вибору

Фрагмент контенту, яким активовано антецедент правила продукції: `База даних – це іменована сукупність даних, що відбиває стан об’єктів і їхнє відношення в розглянутій предметній області.`
Ключовий термін, знання якого перевіряється: «`база даних`»

Використане правило продукції: $m_{1b1} = (a_1 \Rightarrow c_{b1})$.
Використаний зразок правила продукції:
`<Antecedent> [Text=0] [Term] [Connector] [Thesis]`
`<Consequence> {Header} [AnswerT2-1] [Thesis] [Connector] {Answer} {TRUE} [Term] {FALSE} [RandomTerm_1] {FALSE} [RandomTerm_2] {FALSE} [RandomTerm_3]`

Вигляд сформованого тестового завдання: Іменована сукупність даних, що відбиває стан об'єктів і їхнє відношення в розглянутій предметній області – це ○ Даних ○ Програми ○ Банк даних ◉ База даних

Таблиця В.12 – Приклад створення тестового завдання одиничного вибору

Фрагмент контенту, яким активовано антецедент правила продукції: `База даних – це іменована сукупність даних, що відбиває стан об’єктів і їхнє відношення в розглянутій предметній області.`
Ключовий термін, знання якого перевіряється: «`база даних`»

Використане правило продукції: $m_{1b2} = (a_1 \Rightarrow c_{b2})$.
Використаний зразок правила продукції:
`<Antecedent> [Text=0] [Term] [Connector] [Thesis]`
`<Consequence> {Header} [AnswerT2-2] [Term] [Connector] {Answer} {TRUE} [Thesis] {FALSE} [RandomThesis_1] {FALSE} [RandomThesis_2] {FALSE} [RandomThesis_3]`

Вигляд сформованого тестового завдання: База даних – це ○ QBE в основному ○ Процес звертання користувача до БД із метою введення, одержання або зміни інформації в БД ○ Послідовність операцій модифікації даних у БД, що переводить БД із одного несуперечливого стану в інший несуперечливий стан ◉ Іменована сукупність даних, що відбиває стан об'єктів і їхнє відношення в розглянутій предметній області

Додаток Г

Фрагмент таблиці для аналізу параметрів ключових термінів інформаційних навчальних матеріалів

№ з/п	Назва навчального матеріалу; посилання або вказівка на джерело	Множина ключових термінів матеріалу, визначений його автором	Кількість ключових термінів	Кількість слів *n* у терміні	Кількість ключових термінів, що відповідають портрету терміна	Ключові терміни, що не відповідають портрету терміна

1.	Демографічні процеси в Україні як об'єкт державного регулювання *http://studies.in.ua/pravo/20.pdf*	– демографічні процеси,
		– демографічна криза,
		– народжуваність,
		– смертність,
		– тривалість життя,
		– міграція
			6	*n* = 1: 3;
				n = 2: 3;
				n = 3: 0;
				n = 4: 0;
				n = 5: 0;
				n = 6: 0;
				n > 6: 0.
					6	-

2.	Політична культура як чинник демократичних перетворень *http://studies.in.ua/pravo/polit_cultur.pdf*	– політичний процес,
		– політична культура,
		– політична субкультура,
		– цінності,
		– політичні міфи
			5	*n* = 1: 1;
				n = 2: 4;
				n = 3: 0;
				n = 4: 0;
				n = 5: 0;
				n = 6: 0;
				n > 6: 0.
					5	-

3.	Глобалізація: поняття, вплив на сучасні державу і право *http://studies.in.ua/pravo/global.pdf*	– глобалізація,
		– сутнісні ознаки глобалізації,
		– аспекти глобалізації,
		– вплив глобалізації на державу і право
			4	*n* = 1: 1;
				n = 2: 1;
				n = 3: 1;
				n = 4: 0;
				n = 5: 0;
				n = 6: 1;
				n > 6: 0.
					3	– вплив глобалізації на державу і право

4.	Громадянське суспільство і держава: політико-правові та соціальні аспекти взаємодії *http://studies.in.ua/pravo/2012_182_170_10.pdf*	– громадянське суспільство,
		– держава,
		– демократія,
		– соціальна система,
		– соціальний інститут,
		– громадянське об'єднання,
		– солідарний інтерес
			7	*n* = 1: 2;
				n = 2: 5;
				n = 3: 0;
				n = 4: 0;
				n = 5: 0;
				n = 6: 0;
				n > 6: 0.
					7	-

5.	Медіатизація політики як ключовий вектор трансформації легітимаційних та делегітимаційних практик в Україні *http://studies.in.ua/pravo/mediat_polit_vektor.pdf*	– медіатизація,
		– медіа логіка,
		– медіакратія,
		– легітимація,
		– делегітимація
			5	*n* = 1: 4;
				n = 2: 1;
				n = 3: 0;
				n = 4: 0;
				n = 5: 0;
				n = 6: 0;
				n > 6: 0.
					5	-

6.	Політичний дискурс засобів масової інформації України як індикатор суспільних процесів *http://studies.in.ua/pravo/polit_discurs_smi.pdf*	– політичний дискурс,
		– мас-медіа,
		– медіатизація політики,
		– символічний простір,
		– інформація
			5	*n* = 1: 2;
				n = 2: 3;
				n = 3: 0;
				n = 4: 0;
				n = 5: 0;
				n = 6: 0;
				n > 6: 0.
					5	-

7.	Змішана виборча система: світовий досвід застосування *http://studies.in.ua/pravo/zmishana_v_sis.pdf*	– змішана виборча система,
		– лінійне змішування,
		– структурне змішування,
		– змішана пропорційна система,
		– змішана паралельна система,
		– система мінімального представництва
			6	*n* = 1: 0;
				n = 2: 2;
				n = 3: 3;
				n = 4: 0;
				n = 5: 0;
				n = 6: 0;
				n > 6: 0.
					6	-

8.	Криві другого порядку *https://msn.khnu.km.ua/pluginfile.php/367818/mod_resource/content/2/L_1_26.pdf*	– коло;
		– еліпс;
		– гіпербола;
		– парабола;
		– рівняння еліпса, гіперболи
			5	*n* = 1: 4;
				n = 2: 0;
				n = 3: 1;
				n = 4: 0;
				n = 5: 0;
				n = 6: 0;
				n > 6: 0.
					4	– рівняння еліпса, гіперболи

9.	Теорія множин *https://msn.khnu.km.ua/pluginfile.php/360354/mod_resource/content/1/Теорія множин.pdf*	– множина;
		– відношення;
		– функції
			3	*n* = 1: 3;
				n = 2: 0;
				n = 3: 0;
				n = 4: 0;
				n = 5: 0;
				n = 6: 0;
				n > 6: 0.
					3	-

10.	Основні теореми диференціального числення *https://msn.khnu.km.ua/pluginfile.php/367815/mod_resource/content/2/L_1_23.pdf*	– теорема Ролля;
		– теорема Лагранжа;
		– теорема Коші;
		– правило Лопіталя
			4	*n* = 1: 0;
				n = 2: 4;
				n = 3: 0;
				n = 4: 0;
				n = 5: 0;
				n = 6: 0;
				n > 6: 0.
					4	-

…	…	…	…	…	…	…

Додаток Д

Результати дослідження особливостей ручного формування тестових завдань авторами навчальних матеріалів

№ з/п	Назва навчального курсу	Кількість тестових завдань			
		Загальна кількість тестових завдань *	З них – відповідають правилам * **	З них – використані спеціальні об’єкти з контенту**	З них – використані дистрактори поза контенту**
1.	Алгоритмізація та програмування	81	81	6	3
2.	Геометричне моделювання	54	54	13	6
3.	Інсталяція та експлуатація мережевих операційних систем	60	60	2	3
4.	Інтегровані комп’ютерні схеми	51	51	15	2
5.	Інтелектуальних аналіз даних	75	75	1	5
6.	Комп’ютерна графіка	48	48	5	0
7.	Комп’ютерні мережі	92	92	10	4
8.	Крос-платформене програмування	61	61	6	23
9.	Математичні методи дослідження операцій	55	55	4	17
10.	Методи та системи штучного інтелекту	100	100	6	12
11.	Мови SQL-запитів	64	64	0	19
12.	Мови об’єктно-орієнтованого програмування	60	60	1	13
13.	Надійність систем	83	83	3	15
14.	Об’єктно-орієнтоване програмування	67	67	2	2
15.	Операційні системи	70	70	4	6
16.	Організація баз даних та знань	82	82	1	10
17.	Основи математичної логіки	45	45	5	2
18.	Основи програмної інженерії та тестування програмного забезпечення	76	76	3	21
19.	Проектування інформаційних систем	53	53	3	3
20.	Проектування баз даних	60	60	6	2
21.	Проектування розподілених баз даних та знань	41	41	3	7
22.	Системне програмування	58	58	4	1
23.	Теорія алгоритмів	65	65	9	7
24.	Технології захисту інформації	47	47	0	3
25.	Технології комп’ютерного проектування	63	63	4	8
26.	Технологія розподілених систем та паралельних обчислень	80	80	0	5
27.	Технологія створення програмних продуктів	57	57	2	2
28.	Транзитивні системи та їх аналіз	32	32	0	3
29.	Управління ІТ-проектами	68	68	4	10
30.	WEB-технології та WEB-дизайн (1 семестр)	74	74	2	6
31.	WEB-технології та WEB-дизайн (2 семестр)	61	61	4	12
∑	*Загалом (тестових завдань)*	*1983*	*1983*	*128*	*232*
∑	*Загалом (відсотків тестових завдань)*	*100%*	*100%*	*6,45%*	*11,70%*

* не включаючи інші типи тестових завдань; ** множини можуть перетинатись

Додаток Е

Акти впровадження та реалізації результатів наукових досліджень

Науково-технічна фірма
"ІНФОСЕРВІС"

м.Хмельницький,
вул. Театральна, 54 офіс № 211
ЄДРПОУ 14169783
☎ (0382) 79-40-57

7.05.2019 № 1

ДОВІДКА

про впровадження результатів дисертаційної роботи
Мазурця Олександра Вікторовича «Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань»

В процесі виробничої діяльності ТОВ "Науково-технічна фірма «Інфосервіс»" знайшли застосування наступні результати дисертаційної роботи Мазурця Олександра Вікторовича «Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань:

– інформаційна модель семантичної структури навчального курсу, яка забезпечує формальне подання структури цифрових текстових документів та тестів до них;

– метод формування структури навчальних матеріалів та пошуку у них ключових термінів, що дозволяє отримувати ключові слова та словосполучення в розрізі рубрик відповідних цифрових текстових документів;

– метод автоматизованого створення тестових завдань до навчальних матеріалів, який використовує продукційну модель подання знань для створення правил формування тестових завдань;

– інформаційна технологія автоматизованого створення тестових завдань до навчальних матеріалів, яка дозволяє за вхідними даними цифрових текстових документів автоматизовано одержувати множину тестових завдань.

У ТОВ "Науково-технічна фірма «Інфосервіс»" зазначені результати використовуються при розробці промислового програмного забезпечення для створення інформаційних технологій у модулях для пошуку ключових термінів у цифрових текстах та для автоматизованого створення тестових завдань.

ТОВ Науково-технічна фірма "ІНФОСЕРВІС" Код 14169783 Україна м. Хмельницький

Директор _____ Павлишин В.В.

7.05.2019

ЗАТВЕРДЖУЮ
Директор Навчального
простору «VECTOR»
Сергій Володимирович Мандзій С.В.
«01» квітня 2019 р.

АКТ

про впровадження результатів науково-дослідної роботи «Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань» Мазурця Олександра Вікторовича

Комісія у складі

Голова: заступник директора з методичної роботи Молчанова М.О.

Члени комісії: старший викладач, відповідальний за систему тестування Гащук Т.О., викладач Мороз О.О., викладач Ряба А.О., викладач Палій А.А.

цим актом засвідчує, що в Навчальному просторі «VECTOR» знайшли застосування результати дисертаційної роботи Мазурця Олександра Вікторовича «Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань», а саме елементи інформаційної технології створення тестів (інформаційна модель семантичної структури навчального курсу, метод формування структури навчальних матеріалів та пошуку у них ключових термінів, метод автоматизованого формування тестових завдань до навчальних матеріалів).

У Навчальному просторі «VECTOR» зазначені результати використовувались при розробці інноваційних методів та засобів навчання з використанням інформаційних технологій. Застосування розробленого на основі наведених елементів програмного забезпечення дозволило зменшити в 6,5 разів час, необхідний для створення та підготовку до використання наборів тестових завдань. Забезпечення засобами розробленої інформаційної технології формування структури навчального матеріалу та визначення зв'язків створених тестових завдань з елементами структури та контенту навчального матеріалу дозволило реалізувати в навчальному процесі тестування за адаптивним алгоритмом, що дозволило зменшити в середньому на 18 % час, необхідний на проходження тестування учнів на рівень знань модулів навчальних дисциплін.

Голова комісії: Заступник директора з методичної роботи _____ Молчанова М.О.

Члени комісії: Старший викладач _____ Гащук Т.О.

викладач _____ Мороз О.О.

викладач _____ Ряба А.О.

викладач _____ Палій А.А.

ДОВІДКА

про впровадження результатів дисертаційної роботи
Мазурця Олександра Вікторовича
«Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань»

В процесі виробничої діяльності приватного малого підприємства «Лінк» при розробці програмного забезпечення для внутрішнього тестування рівня знань з техніки безпеки та робочих обов'язків працівників знайшли застосування результати дисертаційної роботи Мазурця Олександра Вікторовича «Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань».

Зокрема, було використано такі результати: інформаційна модель семантичної структури навчального курсу, яка забезпечує формальне подання структури цифрових текстових документів та тестів до них; метод формування структури навчальних матеріалів та пошуку у них ключових термінів, що дозволяє отримувати ключові слова та словосполучення в розрізі рубрик відповідних цифрових текстових документів; метод автоматизованого створення тестових завдань до навчальних матеріалів, який використовує продукційну модель подання знань для створення правил формування тестових завдань; інформаційна технологія автоматизованого створення тестових завдань до навчальних матеріалів, яка дозволяє за вхідними даними цифрових текстових документів автоматизовано одержувати множину тестових завдань.

Створене з використанням зазначених результатів програмне забезпечення дозволило автоматизовано створювати множини тестових завдань та використовувати їх для проведення адаптивного тестування. При цьому внаслідок застосування створеного програмного забезпечення вдалося досягти пришвидшення створення тестових завдань в середньому в 3,7 рази у порівнянні з ручним методом, а застосування адаптивного тестування дозволило в порівнянні з алгоритмом випадкового вибору питань зменшити затрачений на тестування час у середньому на 11%, при цьому забезпечивши повне покриття за структурою теоретичного матеріалу.

Директор ПМП «Лінк» _____ *Ноль Р.С.* *15.04.2019*

приватне мале підприємство «ЛІНК» ідентифікаційний код №22777114 Україна м.Хмельницький

«ЗАТВЕРДЖУЮ»
Проректор з науково-педагогічної
роботи, д.е.н., професор
_____ Войнаренко М.П.
"22" травня 2019 р.

АКТ

про впровадження в навчальний процес Хмельницького національного університету результатів дисертаційної роботи старшого викладача кафедри комп'ютерних наук та інформаційних технологій Мазурця Олександра Вікторовича «Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань»

Ми, комісія у складі: завідувача кафедри комп'ютерних наук та інформаційних технологій д.т.н., професора Сорокатого Р.В., ст. викладача кафедри комп'ютерних наук та інформаційних технологій Скрипник Т.К., склала цей акт про те, що результати дисертаційної роботи Мазурця О.В., а саме елементи інформаційної технології автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань, використовуються в навчальному процесі на кафедрі комп'ютерних наук та інформаційних технологій для спеціальності 122 «комп'ютерні науки», зокрема в курсах дисциплін «Методи та системи штучного інтелекту», «Проектування інформаційних систем» та «Інтелектуальний аналіз даних».

На основі матеріалів дисертаційної роботи Мазурця О.В. студентами виконано і успішно захищено ряд магістерських робіт.

Акт обговорений та схвалений на засіданні кафедри комп'ютерних наук та інформаційних технологій (протокол №10 від 22 травня 2019 р.).

Зав.каф. комп'ютерних наук та інформаційних технологій,
д.т.н., професор Р.В. Сорокатий

Секретар кафедри КНІТ,
ст.викладач Т.К. Скрипник

Додаток Ж

Список публікацій праць за темою дисертації та відомості про апробацію результатів дисертації

Список публікацій за темою дисертації

Статті у виданнях, зареєстрованих в наукометричній базі Scopus:

1. Krak I., Barmak O., Mazurets O. The practice investigation of the information technology efficiency for automated definition of terms in the semantic content of educational materials. CEUR Workshop Proceedings. 2016. Vol. 1631. P. 237–245.

2. Krak I., Barmak O., Mazurets O. The practice implementation of the information technology for automated definition of semantic terms sets in the content of educational materials. CEUR Workshop Proceedings. 2018. Vol. 2139. P. 245–254.

3. Barmak O., Krak I., Mazurets O., Pavlov S., Smolarz A., Wojcik W. Research of efficiency of information technology for creation of semantic structure of educational materials. Advances in Intelligent Systems and Computing. 2020. Vol. 1020. P. 554–569.

Статті у фахових виданнях:

4. Бармак О. В., Мазурець О. В. Методи автоматизації визначення семантичних термінів у навчальних матеріалах. Вісник Хмельницького національного університету. Серія : Технічні науки. 2015. № 2. С. 209–213.

5. Бармак О. В., Мазурець О. В. Інформаційна технологія автоматизованого визначення термінів у навчальних матеріалах. Вимірювальна та обчислювальна техніка в технологічних процесах. 2015. № 2. С. 94–102.

6. Бармак О. В., Мазурець О. В., Матвійчук А. О. Застосування інформаційної технології гнучкого тестування рівня знань у середовищі Moodle. Вісник Хмельницького національного університету. Серія : Технічні науки. 2017. № 2. С. 103–114.

7. Бармак О. В., Мазурець О. В., Кліменко В. І. Інформаційна технологія автоматизованого формування тестових завдань. Вісник Хмельницького національного університету. Серія : Технічні науки. 2017. № 5. С. 93–103.

8. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2017. № 6. С. 223–229.

9. Мазурець О. В., Ковальчук О. В., Слободзян В. О. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 1. С. 61–69.

10. Мазурець О. В. Інформаційна технологія автоматизованого визначення семантичних термінів в елементах навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 3. С. 223–230.

11. Мазурець О. В. Розробка множини тегів для формального опису елементів моделей автоматизованого формування тестових завдань. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 5. С. 26–31.

12. Крак Ю. В., Бармак О. В., Мазурець О. В. Практична реалізація інформаційної технології автоматизованого визначення множини семантичних термінів в контенті навчальних матеріалів. Проблеми програмування. 2018. № 2–3. С. 245–254.

13. Бармак О. В., Мазурець О. В. Інформаційна модель семантичної структури навчального курсу. Вісник Хмельницького національного університету. Серія : Технічні науки. 2018. № 6. Т. 1. С. 92–97.

14. Мазурець О. В. Інформаційна технологія автоматизованого створення тестів до навчальних матеріалів. Вісник Хмельницького національного університету. Серія : Технічні науки. 2019. № 4. С. 84–91.

15. Мазурець О. В. Метод автоматизованого формування тестових завдань. Вісник Хмельницького національного університету. Серія : Технічні науки. 2019. № 5. С.189–194.

Патенти на корисну модель:

16. Мазурець О. В. Пат. на корисну модель 129903. Україна, МПК G06F 17/00. Спосіб визначення переліку ключових слів у тексті. № u201707242; заявл. 10.07.2017; опубл. 26.11.2018, Бюл. № 22.

17. Мазурець О. В. Пат. на корисну модель 137386. Україна, МПК G06F 17/00. Спосіб обмеження переліку ключових слів тексту. № u201900552; заявл. 18.01.2019; опубл. 25.10.2019, Бюл. № 20.

Опубліковані праці апробаційного характеру:

18. Мазурець О. В. Інформаційна технологія побудови онтологічної моделі навчального курсу для оцінювання отриманих знань. Інформаційні управляючі системи та технології : матеріали III Міжнародної науково-практичної конференції. (м. Одеса, 23–25 вересня 2014 р.). Одеса, 2014. С. 81–83.

19. Бармак О. В., Крак Ю. В., Мазурець О. В. Інформаційна технологія автоматизованого визначення термінів у навчальних матеріалах. Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту : матеріали Міжнародної наукової конференції. (с. Залізний Порт, 25–28 травня 2015 р.). Херсон, 2015. С. 26–28.

20. Мазурець О. В. Особливості застосування інформаційної технології автоматизованого визначення термінів у навчальних матеріалах. Інформаційні управляючі системи та технології : матеріали IV Міжнародної науково-практичної конференції. (м. Одеса, 22–24 вересня 2015 р.). Одеса, 2015. С. 62–65.

21. Мазурець О. В. Дослідження ефективності інформаційної технології автоматизованого визначення термінів у навчальних матеріалах. ICACIT-2015 : матеріали III Міжнародної конференції з автоматичного

управління та інформаційних технологій (м. Київ, 11–13 грудня 2015 р.). Київ, 2015. С. 136–139.

22. Кліменко В. І., Мазурець О. В. Аналіз сучасних методів генерації тестових завдань. Актуальні проблеми комп'ютерних технологій : збірник наукових праць за матеріалами X міжнародної науково-технічної конференції. (м. Хмельницький, 31 травня 2016 р.). Хмельницький, 2016. С. 77–84.

23. Крак Ю. В., Бармак О. В., Мазурець О. В. Практичне дослідження ефективності інформаційної технології автоматизованого визначення семантичних термінів в контенті навчальних матеріалів. Прикладне програмне забезпечення УкрПРОГ'2016 : матеріали X Міжнародної науково-практичної конференції по програмуванню. (м. Київ, 24–26 травня 2016 р.). Київ, 2016. С. 237–245.

24. Мазурець О. В. Особливості автоматизованого формування тестових завдань. Інформаційні управляючі системи та технології : матеріали V Міжнародної науково-практичної конференції. (м. Одеса, 20–22 вересня 2016 р.). Одеса, 2016. – С. 71-73.

25. Мазурець О. В., Якимюк О. М. Моделі оцінки ефективності методів пошуку ключових термінів у контенті навчальних матеріалів. Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи : матеріали Всеукраїнської науково-практичної конференції з міжнародною участю. (м. Тернопіль, 9–10 листопада 2017 р.). Тернопіль, 2017. С. 258–261.

26. Мазурець О. В. Інформаційна технологія гнучкого тестування рівня знань у середовищі MOODLE. Інформаційні управляючі системи та технології : матеріали VI Міжнародної науково-практичної конференції. (м. Одеса, 20–22 вересня 2017 р.). Одеса, 2017. С. 83–85.

27. Крак Ю. В., Бармак О. В., Мазурець О. В. Практична реалізація інформаційної технології автоматизованого визначення множини семантичних термінів в контенті навчальних матеріалів. Прикладне програмне забезпечення

УкрПРОГ’2018 : матеріали XI Міжнародної науково-практичної конференції по програмуванню. (м. Київ, 22–24 травня 2018 р.). Київ, 2018. С. 245–254.

28. Мазурець О. В. Використання множини тегів для формального опису моделей формування тестових завдань. Інформаційні управляючі системи та технології : матеріали VII Міжнародної науково-практичної конференції. (м. Одеса, 17–18 вересня 2018 р.). Одеса, 2018. С. 69–72.

29. Бармак О. В., Мазурець О. В. Дослідження точності та повноти автоматизованого визначення семантичних термінів у навчальних матеріалах. Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи : матеріали III Міжнародної науково-практичної конференції. (м. Тернопіль, 5 квітня 2019 р.). Тернопіль, 2019. С. 98–101.

30. Крак Ю. В., Бармак О. В., Мазурець О. В. Інформаційна модель семантичної структури навчального курсу для генерації тестових завдань. Моделювання та дослідження стійкості динамічних систем : матеріали XIX Міжнародної науково-практичної конференції. (м. Київ, 22–24 травня 2019 р.). Київ, 2019. С. 365–367.

31. Мазурець О. В. Застосування продукційної моделі для автоматизованої генерації тестових завдань. Інформаційні управляючі системи та технології : матеріали VIII Міжнародної науково-практичної конференції. (м. Одеса, 23–25 вересня 2019 р.). Одеса, 2019. С. 52–54.

Продовження додатку Ж

Таблиця Ж.1 – Відомості про апробацію результатів дисертації

№ з/п	Назва конференції	Місце проведення	Дата проведення	Форма участі
1.	«Інформаційні управляючі системи та технології ICST-ODESSA-2014» : III Міжнародна науково-практична конференція	м. Одеса	23–25 вересня 2014 р.	заочна
2.	«Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту» : Міжнародна наукова конференція	с. Залізний Порт	25–28 травня 2015 р.	очна
3.	«Інформаційні управляючі системи та технології ICST-ODESSA-2015» : IV Міжнародна науково-практична конференція	м. Одеса	22–24 вересня 2015 р.	заочна
4.	«ICACIT-2015» : III Міжнародна конференція з автоматичного управління та інформаційних технологій	м. Київ	11–13 грудня 2015 р.	очна
5.	Актуальні проблеми комп'ютерних технологій АПКТ-2016» : X Міжнародна науково-технічна конференція	м. Хмельницький	31 травня 2016 р.	очна
6.	«Прикладне програмне забезпечення УкрПРОГ'2016» : X Міжнародна науково-практична конференція по програмуванню	м. Київ	24 – 26 травня 2016 р.	очна
7.	«Інформаційні управляючі системи та технології ICST-ODESSA-2016» : V Міжнародна науково-практична конференція	м. Одеса	20–22 вересня 2016 р.	заочна
8.	Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи» : Всеукраїнська науково-практична конференція з міжнародною участю	м. Тернопіль	9–10 листопада 2017 р.	очна
9.	«Інформаційні управляючі системи та технології ICST-ODESSA-2017» : VI Міжнародна науково-практична конференція	м. Одеса	20–22 вересня 2017 р.	заочна
10.	«Прикладне програмне забезпечення УкрПРОГ'2018» : XI Міжнародна науково-практична конференція по програмуванню	м. Київ	22–24 травня 2018 р.	очна
11.	«Інформаційні управляючі системи та технології ICST-ODESSA-2018» : VII Міжнародна науково-практична конференція	м. Одеса	17–18 вересня 2018 р.	заочна
12.	«Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи» : III Міжнародна науково-практична конференція	м. Тернопіль	5 квітня 2019 р.	очна
13.	«Моделювання та дослідження стійкості динамічних систем DSMSI-2019» : XIX Міжнародна науково-практична конференція	м. Київ	22–24 травня 2019 р.	очна
14.	«Інформаційні управляючі системи та технології ICST-ODESSA-2019» : VIII Міжнародна науково-практична конференція	м. Одеса	23–25 вересня 2019 р.	заочна