Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Аналіз методів обробки документів в форматі docx для виявлення помилок
(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-3

спеціальності 121

Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-наукової програми

Інженерія програмного забезпечення
(повна назва освітньої програми)

Будянський Олексій Олексійович
(прізвище, ініціали)

Керівник доц. Ревенчук І. А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ проф. Дудар З.В.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет	Комп'ютерних наук
Кафедра	Програмної інженерії
Рівень вищої освіти	другий (магістерський)
Спеціальність	121 - Інженерія програмного забезпечення
освітньо-наукова програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____

(підпис)

«___»_____ 2020 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Будянському Олексію Олексійовичу

1. Тема роботи Аналіз методів обробки документів в форматі docx для виявлення помилок

затверджена наказом по університету від 27.03.2020 № 473

2. Термін подання студентом роботи до екзаменаційної комісії

18.05.2020

3. Вихідні дані до роботи В програмній системі валідації форматування документів передбачити можливість виведення помилок по вхідному документу формату docx; можливість конфігурування налаштувань форматування. Використовувати середовище розробки програмного забезпечення Visual Studio та мову програмування C#, ОС Windows.

4. Перелік питань, що потрібно опрацювати в роботі Формати документів, метоти валідації форматування документів, підходи до аналізу форматуваняя.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) приклад структури файлів, діаграми, вигляд додатку, алгоритми.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
	Об'єктний аналіз поставленої задачі	01-04-2020	
	Розробка моделі взаємодії даних	05-04-2020	
	Розробка структури зберігання даних	10-04-2020	
	Створення коду програми	15-04-2020	
	Тестування і налагодження програми	22-04-2020	
	Підготовка пояснювальної записки.	30-04-2020	
	Підготовка презентації та доповіді	09-05-2020	
	Нормоконтроль, рецензування	11-05-2020	
	Попередній захист	11-05-2020	
	Занесення диплома в електронний архів	11-05-2020	
	Допуск до захисту у зав. кафедри	15-05-2020	
	Дата захисту	18-05-2020	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання: 30.03.2020

Студент _____ Будянський О.О
(підпис) (посада, прізвище, ініціали)

Керівник роботи _____ Доц. Ревенчук І.А
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 69 с., 11 рис., 1 табл., 3 додатки, 20 джерел.

WORD, DOCX, DOC, PDF, ВАЛІДАЦІЯ, ДОКУМЕНТ, ФОРМАТУВАННЯ, СТАНДАРТИ ФОРМАТУВАННЯ

Об’єктом дослідження є аналіз методів обробки документів в форматі docx для виявлення помилок відносно різних стандартів.

Метою роботи є розробка програмної системи, що допоможе реалізувати методи обробки документів та автоматизувати перевірку документів.

Методи розробки базуються на алгоритмах обробки тексту та підходах до роботи з файлами типу XML.

У результаті роботи здійснена програмна реалізація системи для валідації документів формату docx відносно сконфігурованих правил форматування.

WORD, DOCX, DOC, PDF, VALIIDATION, DOCUMENT, FORMATTING, FORMATTING STANDARTS

The aim of the work is to analyze docx document processing methods for detecting errors against different standards.

The purpose of the work is to develop a software system that will help implement document processing methods and automate the process of document verification.

Development methods are based on word processing algorithms and approaches to handle XML files.

As a result, a software for validation of docx format documents with respect to configured formatting rule was implemented.

ЗМІСТ

ВСТУП

У наш час існує багато установ, що працюють з постійним потоком документів, що мають відповідати певним строго визначеним вимогам форматування та оформлення. Більшість з таких організацій стикаються з проблемою перевірки оформлення документів, що є рутинною роботою та займає значну частину робочого часу спеціалістів. Зі схожими проблемами можна зіткнутися майже у кожній сфері, адже документообіг є невід'ємною частиною сучасної професії. Документообіг є діяльністю по організації пересування документів на підприємстві від моменту їх отримання або створення до закінчення виконання: напрямки в архів або відправки з організації. Виділяють три потоки документації: документи, які створюються в організації і використовуються в управлінському процесі працівниками організації (внутрішні), документи, які відправляються в інші організації (вихідні), документи, які надходять з інших організацій (вхідні). Особливістю роботи державних підприємств та структур України є те, що кожен крок та рішення найманого співробітника повинен документуватися, така робота характеризується великою кількістю звітів, а затрати часу на підготовку та перевірку цих паперів можуть сягати 70 відсотків від усієї кількості роботи.

Прикладом такої роботи є перевірка наукових робіт у вищих навчальних закладах. Існують певні документовані стандарти, яким мають відповідати наукові роботи, а саме - ДСТУ 3008-2015 «Звіти у сфері науки і техніки» [1]. Провівши аналіз різних стандартів, було вирішено сконцентруватися саме на ДСТУ 3008-2015, адже він описує правила форматування для найбільш широкого кола елементів документу (таблиці, схеми, списки, рисунки, формули тощо).

Проаналізувавши документообіг у вищих наукових закладах було зроблено висновок, що майже усі документи, що потребують перевірки форматування розповсюджуються у форматі Microsoft Word Document (.doc, .docx) [2]. Особливістю

цього формату є опис документу за допомогою структури xml, що також задокументована у стандартах ECMA (ECMA-376) [3]. Саме ці стандарти дозволяють автоматизувати процес валідації форматування та оформлення документів у форматах .doc та .docx. Даний формат надає великий набір інструментів для автоматизації аналізу форматування, але, у свою чергу, також має ряд недоліків, що ускладнюють перевірку деяких елементів.

Заданий формат дозволяє автоматизувати перевірку наступних основних груп стандартів форматування документу:

- стиль тексту (шрифт, розмір тексту, формат тощо);

- поля, інтервали та відступи (абзацний відступ, поля документу, інтервал між рядками тощо);

- зміст контенту (семантичний аналіз тексту);

- формули, таблиці, діаграми тощо.

Саме тому, було зроблено висновок, що автоматизація даного процесу може суттєво зекономити час та гроші людей у різних сферах та позитивно вплинути на процес діджиталізації в нашій країні. В Україні на державному рівні визнається необхідність формування цифрової економіки, а цифрові технології розглядаються в якості одного із ключових драйверів сталого розвитку, тобто впровадження такої системи у підприємства різних видів, які мають великий обіг звітності може позитивно вплинути на економіку країни в цілому та може надати людям змогу сконцентрувати свою увагу на вирішенні більш значимих та креативних задач, а рутинну роботу буде виконувати машина.

Отже, для вирішення проблеми автоматизації кожного окремого пункту перевірки форматування буде проведений аналіз, яким чином можна отримати інформацію про той чи інший формат, розроблено окремий унікальний метод, використовуючи різні особливості формату docx, або інших форматів, що можуть бути допоміжними у даному питанні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз проблеми

Майже кожне підприємство та організація в Україні має постійний набір документів, які циркулюють між різними установами та мають специфічні вимоги до форматування. До таких документів відносяться звіти різної форми, статті, публікації, тощо. Кількість таких паперів в рамках однієї установи може сягати десятків тисяч за рік, а робота над їх створенням займає значну частку роботи такої установи. Зазвичай, велика кількість часу витрачається саме на рутинну працю перевірки форматування та відповідності стандартам того чи іншого виду документу.

Швидкий розвиток сучасних інформаційних технологій спонукає організації і підприємства впроваджувати інноваційні технології в інформаційному забезпеченні їхньої діяльності. У даний час це має великий вплив на успішну реалізацію управління організацією та прийнятті правильних стратегічних рішень. Управління інформаційним забезпечення діяльності підприємства тісно пов’язане з наявністю сучасних інформаційних ресурсів, а також з можливістю впровадження інноваційних підході у впорядкуванні та обробці документаційних та інформаційних потоків підприємства. Ефективна обробка постійно зростаючих обсягів інформації, основну частину яких складають документи та звіти різних форматів, можлива тільки при умові автоматизованої перевірки та обробки.

Саме тому, автоматизація процесу валідації формату та стилю документів є пріоритетним рішенням, що може суттєво збільшити ефективність роботи та підготовки документів в установах.

Чудовим прикладом такої роботи є перевірка наукових робіт у вищих навчальних закладах. Кожного року в університетах велика кількість студентів та працівників освітніх закладів створюють курсові роботи, атестаційні роботи, дисертації, звіти, тощо. Усі ці документи обов’язково мають відповідати певним

вимогам оформлення, а перевірка оформлення завжди виконується окремим етапом спеціальною людиною. З однієї сторони це шалені витрати часу людей, що займаються перевіркою цих паперів на відповідність вимогам, а з іншої сторони, люди витрачають велику кількість часу на те, щоб відформовувати ці документи таким чином, щоб вони відповідали вказаним стандартам. Тобто, автоматизація даного процесу була б вигідна обом сторонам цього процесу, адже людина змогла б перевіряти свій документ на відповідність без відправки його перевіряючому та одразу бачити свої помилки, а перевіряючи не повинні були б витрачати значну частку свого часу на рутинну роботу та сконцентруватися на більш важливій роботі.

Існують певні документовані стандарти, яким мають відповідати наукові роботи, а саме - ДСТУ 3008-2015 «Звіти у сфері науки і техніки» [1]. Даний стандарт установлює основні вимоги до структурних елементів і правил оформлювання звітів у сфері науки й техніки. Стандарт застосовується для оформлення різних видів звітності про результати теоретичних та практичних досліджень, наукових, конструкторських, дослідницьких та інших робіт.

На основі положень і вимог цього стандарту розроблені інформативні та методичні документи, які встановлюють вимоги до оформлення Нормативно-правових актів і документів, що регламентують підготовку наукових кадрів в Україні: дипломних робіт, дисертаційних робіт, магістерських робіт, бакалаврських робіт, курсових робіт, наукових статей і публікацій до наукових видань.

Вказаний стандарт широко застосовується в усіх сферах наукової діяльності, зокрема під час складання звітів за етапи та річних звітів за різними напрямами, оформлення дисертацій, науково-технічної діяльності, посібників, підручників тощо.

Саме тому основною задачею є автоматизація процесу валідації документів формату docx, за допомогою різних методів обробки документів. Методи повинні базуватися на алгоритмах обробки тексту та підходах до роботи з файлами типу XML.

1.2 Формат документів

Провівши аналіз документообігу у вищих наукових закладах було зроблено висновок, що майже усі документи, що потребують перевірки форматування розповсюджуються у форматі Microsoft Word Document (.doc, .docx) [2].

DOC і DOCX - формати файлів, які використовуються в програмі Word для Microsoft. Основна відмінність DOC від DOCX - їх поточний статус. Формат DOC Microsoft використовувався до версії Word 2003 року. У Word 2007 було представлено DOCX, як новий формат за замовчуванням. Користувачі все ще користуються формато DOC, але числа користувачів зменшується.

Найбільшою проблемою у DOCX є сумісність, оскільки Word 2003 та старіші версії не в змозі відкрити файли DOCX. Це основна проблема при обміні файлами, оскільки не всі люди оновлюють своє програмне забезпечення з кожною новою версією. Щоб вирішити проблему, Microsoft випустила пакет сумісності, який дозволяє старішим версіям Office відкривати DOCX та інші пов'язані формати.

У DOC документ зберігається у бінарному файлі, який містить інформацію про форматування та іншу метадані. З іншого боку, DOCX-файл - це в основі zip-файл, який містить набір XML-файлів, що відносяться до документа. Якщо замінити розширення DOCX на ZIP, можна легко відкрити його за допомогою будь-якого програмного забезпечення для розархівування zip та переглянути або змінити XML-документи.

Формат DOC використовується Microsoft вже досить давно, але його основна суть була в тому, що інші виробники програмного забезпечення не змогли використовувати формат для власних програм. Навіть інші програми для обробки тексту мають труднощі з точним читанням файлів DOC. Основна мета Microsoft з DOCX - створити відкритий стандарт, який можуть також прийняти інші компанії; отже, використання XML як основи.

Через введення DOCX та інших форматів на основі XML, можна із впевненістю сказати, що формат DOC буде стрімко замінятися на користь нових форматів.

Формат DOCX є найпопулярнішим стандартом для обміну файлами документів. Приблизно один мільярд людей, що користуються Microsoft Office, використовують його, як основний. Його найближчий конкурент - формат ODT - підтримується лише Open / LibreOffice та деякими продуктами з відкритим кодом, що робить його далеко не стандартним. Формат PDF не є конкурентом, оскільки PDF-файли не можна редагувати і не містять повної структури документів, тому вони можуть приймати лише обмежені локальні зміни, такі як водяні знаки, підписи тощо. Саме тому більшість ділових документів створюються у форматі DOCX, адже не існує гарної альтернативи замінити його.

Структури, зазначені в цьому форматі, являють собою розширений словник XML, що відображає структуру текстового документу. Розширені елементи та атрибути дозволяють формату вказувати додаткову інформацію про документ або вказувати зміст та форматування частин документа за межами елементів та атрибутів, визначених у специфікації Office Open XML File Formats [4]. Формат був спочатку стандартизований Ecma (як ECMA-376 [5][6]), а також ISO [7] та IEC (як ISO / IEC 29500 [8]) у пізніших версіях.

Безпосередньо в DOCX, додатково до OOXML, елементи та атрибути, складаються з шести груп. Перша група розширює словниковий запас для опису властивостей форматування тексту, додаючи елементи для вказівки текстових ефектів, таких як тінь, світіння, відображення, а також додаючи елементи для визначення типографічних властивостей, таких як лігатури або спосіб відображення міжрядкового інтервалу.

Друга група розширює налаштування, застосовані до документу з обробки тексту, додавши два налаштування для управління збереженням зображень та два параметри, які використовуються, коли документ має декількох авторів.

Третя група розширень передбачає визначення ще чотирьох типів структурованих тегів документів та зміну вигляду структурованих тегів документів.

Четверта група розширень визначає три додаткові атрибути, які відображатимуться у рядках абзацу, розділу чи таблиці. Ці атрибути передбачають однозначну ідентифікацію абзаців або рядків таблиці в частині документа, або надають інформацію про наявність орфографічних помилок в абзаці, або надають інформацію про форматування макета виносок у розділі.

П'ята група визначає вісім нових елементів, які можна використовувати, коли конфліктні редакції є в документі, автором якого є кілька авторів.

Шоста група задає новий атрибут, який відображатиметься на зображенні та вбудованих об'єктах, щоб забезпечити ідентифікатор для цих об'єктів.

Даний формат надає великий набір інструментів для автоматизації аналізу форматування, але, у свою чергу, також має ряд недоліків, що ускладнюють перевірку деяких елементів. Приклад структури docx архіву зображений на рисунку 1.1.

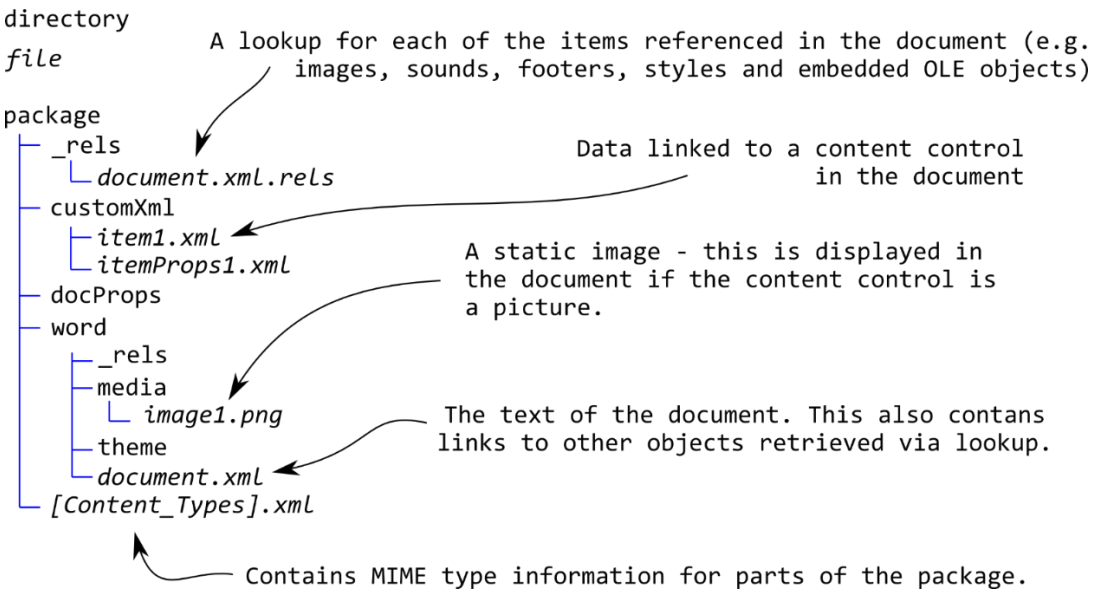


Рисунок 1.1 – Приклад внутрішньої структури XML файлів формату DOCX

Заданий формат дозволяє автоматизувати перевірку наступних основних груп стандартів форматування документу:

- стиль тексту (шрифт, розмір тексту, формат тощо);

- поля, інтервали та відступи (абзацний відступ, поля документу, інтервал між рядками тощо);

- зміст контенту (семантичний аналіз тексту);

- формули, таблиці, діаграми тощо.

Незважаючи на те, що даний формат надає досить велику кількість можливостей для валідації, він має істотні недоліки:

- відсутність будь-якої інформації про розміщення тексту по сторінках;

- неможливість проаналізувати кількість змісту на певній сторінці;

-неможливість проаналізувати правильність розміщення елементів, що займають декілька сторінок.

Однією з основних складностей роботи з форматом docx є те, що багато елементів можуть мати багато рівнів успадкування, наприклад стилі. Стиль може базуватися на іншому стилі, або може бути заснований на "No Style". І стилі можуть базуватися на стилях, заснованих на стилях, глибиною до 9 рівнів. Тож стиль може мати і батьківського, і онукового; і може мати одного або декількох дітей. Стиль успадковує всі атрибути свого батьківського, за винятком випадків, коли ви вказали інше. Наприклад, якщо батьківський стиль є Times New Roman 12pt blue, то його дочірнім кольором буде Times New Roman 12pt blue, якщо не вказано інший шрифт, розмір чи колір. Саме тому, ідентифікація стилю для елементів є достатньо складною задачею, адже треба проаналізувати багато рівнів стилів усіх батьківських елементів, щоб виявити наявний.

Після аналізу, було виділено декілька основних методів обробки документів формату docx:

COM-об'єкти. COM (Компонентна модель об'єкта - Об'єктна модель компонентів) -технологія від компанії Microsoft, що полягає в створенні програмного забезпечення на основі різних взаємодіючих компонентів, кожен із яких доступний для використання багатьма програмнимс продуктами. У сучасних версіях Windows

COM використовується дуже широко. На основі COM були створені технології: Microsoft OLE Automation, ActiveX та інші. Данна технологія дозволяє створити та редагувати документи будь-якої складності, для роботи використовується мова Visual Basic for Applications (VBA) та Microsoft Office API. Недостатки даної технології наступні:

- необхідність встановлення пакетів Microsoft Office і, відповідно, необхідність використання операційної системи з сімейства Windows;

- низька швидкість створення або відкриття нового документа, тому що робота здійснюються через запуск Microsoft Word у фоновому режимі;

- велика надмірність коду навіть при створенні простих документів.

Об'єкти .NET. Для роботи з цими об'єктами потрібна установка програмної платформи .NET Framework. Клас DOTNET дозволяє створювати об'єкти зі збірок .NET, викликати їх методи і використовувати їх властивості. Дане рішення повторює недоліки першого методу, але значно підвищує читаність коду, що відповідає за обробку документа за рахунок використання об'єктно підходу .NET. Іншою перевагою даного методу є переносимість коду, що відповідає за генерацію документа, між програмами, написаними на різних мовах програмування, але на єдиній платформі .NET.

Власна реалізація автоматизованої обробки текстових документів. Документи Microsoft Word з розширенням DOC є бінарними файлами, а документи з розширенням DOCX є файли в відкритому форматі Office Open XML.

Виходячи з вищесказаного можна відзначити, що робота з документами у форматі DOC затруднена через відсутність документації про структуру файлу. Робота з файлами DOCX, навпаки, досить проста за рахунок відкритості формату.

2 АНАЛІЗ АЛГОРИТМІВ ОБРОБКИ ПРАВИЛ ФОРМАТУВАННЯ

Для аналізу кожного окремого правила форматування потрібно розробляти окремий алгоритм, за допомогою якого можна отримати ту чи іншу інформацію, яка дасть змогу зробити висновок, чи коректне форматування за вказаним правилом у вказаному параграфі. Саме тому, у цьому розділі розглянуто декілька показових прикладів аналізу, які наглядно можуть представити основні складності виявлення проблем.

2.1 Аналіз коректності використаного сімейства шрифтів

Інформація про всі використані шрифти документа зберігаються в спеціальному файлі, що називається fontTable.xml. Користувач використовує частину таблиці шрифтів, щоб визначити, які шрифти використовувати для відображення вмісту, коли шрифти, визначені у вмісті, недоступні в системі користувача. Про шрифти зберігаються два основні фрагменти інформації (обидва опціональні): інформація про шрифт для активації заміни шрифту, і одна або кілька вбудованих форм шрифту для використання, коли система користувача не має доступу до шрифту.

Кореневим елементом таблиці шрифтів є елемент Fonts. Корінь повинен містити дочірній font елемент шрифту для кожного шрифту, який використовується в документі. Елемент font повинен містити ім'я шрифту в атрибуті імені. Це ім'я використовується для посилання на шрифт, на який посилається елемент rFonts у властивостях текстового запуску.

Коли шрифти вбудовуються, вони можуть зберігатися як шрифт з растровим зображенням (кожен символ зберігається у вигляді растрового зображення) або у

форматі, що відповідає ISO/IEC 14496-22: 2007 [7]. Пакет може містити нуль або більше частин шрифту. Приклад задання шрифту зображений на рисунку 2.1.

```
<w:fonts>
    <w:font w:name="Arial">
        <w:panose1 w:val="020B0604020202020204" />
        <w:charset w:val="00" />
        <w:family w:val="swiss" />
        <w:pitch w:val="variable" />
        <w:sig w:usb0="20002A87" w:usb1="00000000"
            w:usb2="00000000" w:usb3="00000000"
            w:csb0="000001FF" w:csb1="00000000" />
    </w:font>
    <. . .
</w:fonts>
```

Рисунок 2.1 – Приклад задання шрифту для документа DOCX

Інформація про використані шрифти може зберігатися або безпосередньо у властивостях параграфу, або в стилях. Також там буде вказаний розмір та колір тексту відповідними елементами color та sz (size). Приклад задання інформації про шрифт зображений на рисунку 2.1.

```
<w:rPr>
    <w:rFonts w:ascii="Courier New" w:hAnsi="Times New Roman" w:cs="Times New Roman" />
    <sz w:val="28" />
    <lang w:val="uk-UA" />
    <w:color w:val="FFF200" />
</w:rPr>
```

Рисунок 2.2 – Приклад задання інформації про шрифт, розмір, колір параграфу

Дана структура робить їх валідацію достатньо простою, треба пройтися по кожному параграфу документа, та якщо цей параграф повинен валідуватися, то переконатися, що данні налаштування шрифту є відповідними еталонним.

Параграф може не мати стилів та не мати ніякої інформації про шрифт, тоді треба звертатися до Defaults значення шрифту документа, що зберігається в docDefaults секції.

2.2 Аналіз кількості контенту на сторінці

Однією з вимог для документу за ДСТУ 3008-2015 є кількість наявного текстового контенту на сторінці, тобто не повинно бути напівпорожніх сторінок. Для валідації цього правила, формат docx є не найкращим, адже він не зберігає інформацію про розташування елементів на конкретних сторінках. Тобто, маючи тільки docx ми не зможемо зрозуміти, наскільки та чи інша сторінка заповнена контентом.

Для вирішення цієї задачі було вирішено задіяти інші формати документів, а саме формат pdf. Перевагою формату pdf є те, що дані зберігаються безпосередньо в виді документа зі сторінками, та місце кожного елемента є чітко визначеним. Таким чином, можна перевірити, чи заповнена сторінка контентом на необхідний відсоток. На рисунку 2.3 зображений алгоритм валідації кількості контенту.

Тобто алгоритм даної перевірки наступний:

а) конвертувати документ з формату docx до формату pdf;
б) для кожної сторінки:
    1) знайти розташування останнього тексту ігноруючи нумерацію сторінок та колонтитули;
    2) знайти відношення розташування останнього елемента до розміру сторінки;
    3) перевірити, чи знайдене відношення є більшим, ніж заданий мінімальний процент контенту на сторінці.

Даний алгоритм має свої плюси та певні недоліки. Основним недоліком є те, що проіентне співвідношення є не найкращим показником кількості контенту, адже зазвичай це вимірюється кількістю пустих строк.

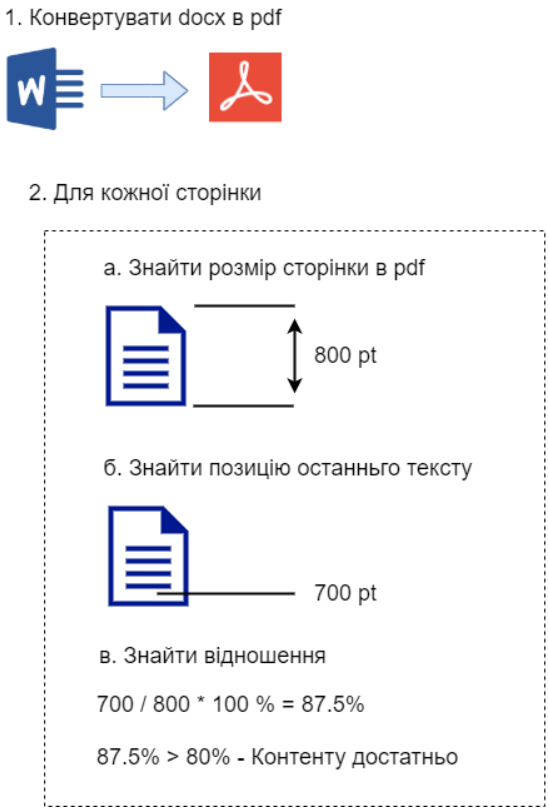


Рисунок 2.3 – Алгоритм валідації кількості контенту на сторінці

Таким чином, використовуючи переваги формату pdf, ми можемо коректо провалідувати це правило. Але для того, щоб зрозуміти чи потрібно аналізувати конкретну сторінку, треба використовувати формат docx, щоб ідентифікувати це по контенту та структурі, адже є сторінки виключення, що можуть мати інший допустимий відсоток, або просто не валідуватися. Звичайно, цей підхід уповільнить роботу алгоритму, адже обробляти тепер потрібно не один, а два документи.

2.3 Аналіз нумерації сторінок

Нумерація сторінок в docx форматі задається за допомогою елемента pgNumType. Він вказує номер сторінки, який відображається на першій сторінці розділу. Якщо це значення пропущене, нумерація продовжуватиметься від найбільшого номера сторінки в попередньому розділі. Інший елемент chapStyle вказує єдиний індекс стилю заголовка, застосований до заголовків глав у документі, який повинен використовуватися як заголовки глав у всіх номерах сторінок цього розділу, шляхом знаходження найближчого заголовка цього стилю та вилучення інформації про нумерацію. Для того щоб проаналізувати правильність нумерації, цієї інформації недостатньо, адже конфігурація форматування відноситься до всієї секції, а окремі сторінки провалідувати неможливо. Для цієї задачі також було прийнято рішення скомбінувати два формати задля отримання результати. Отже, налаштування нумерації та колонтитулів перевіряються за допомогою docx, тоді як відповідність сторінкам ті інші випадки перевіряються за допомогою формату pdf.

Приклад стартової сторінки для нумерації секції зображений на рисунку 2.4.

```
<w:sectPr>
    <w:pgNumType w:start="25" />
</w:sectPr>
```

Рисунок 2.4 – Приклад задання стартової сторінки для нумерації секції

Тобто для кожної сторінки у форматі pdf ми шукаємо текст з номером. Якщо таких декілька, беремо той, що знаходиться ближче до зони нумерації. Далі звіряємо номер з очікуваним номером сторінки. Потім звіряємо положення нумерації на цій конкретній сторінці зі сконфігурованим стандартним положення. Після чого

валідуємо налаштування у форматі docx, а саме розриви нумерації, з яких секцій починається, тощо.

2.4 Валідація інтервалів

Інтервали між абзацами та між рядками абзацу визначається за допомогою елемента <w: spacing> в docx. Значення задаються в двадцятих точки. Звичайний одномісний парагаф має значення <w: line> в 240. Приклад задання інтервалів зображений на рисунку 2.5.

```
<w:pPr>
    <w:spacing
        w:before="120"
        w:after="120"
        w:beforeAutospacing="0"
        w:afterAutospacing="0" />
</w:pPr>

<w:pPr>
    <w:spacing
        w:before="360"
        w:after="120"
        w:line="480"
        w:lineRule="auto"
        w:beforeAutospacing="0"
        w:afterAutospacing="0" />
</w:pPr>

<w:pPr>
    <w:spacing
        w:before="120"
        w:after="120"
        w:beforeAutospacing="0"
        w:afterAutospacing="0" />
</w:pPr>
```

Рисунок 2.5 – Приклад задання інтервалів

Для задання всіх інтервалів використовуються наступні атрибути:

- after - вказує інтервал (в абсолютних одиницях), який буде додано після останнього рядка абзацу;

- before - вказує інтервал (в абсолютних одиницях), який буде додано перед першим рядком абзацу;

- line - вказує кількість вертикального проміжку між рядками тексту в абзаці;

- lineRule - вказує, як обчислюється інтервал між рядками, як зазначено в атрибуті рядка;

- beforeAutospacing вказує, чи повинен інтервал перед абзацом визначати текстовий процесор;

- afterAutospacing - вказує, чи повинен інтервал після абзацу визначати текстовий процесор;

Валідація інтервалів є достатньо простою й виконується для кожного параграфу, але вони можуть каскадно наслідуватися від батьківських елементів, саме тому, валідувати потрібно декілька рівнів каскадно.

2.5 Паралельна обробка алгоритмів

Швидкість обробки одного документа напряму залежить від декількох основних факторів:

- розмір документа, що подається на перевірку;

- кількість валідаційних правил, або, точніше, алгоритмів що будуть повністю пробігати документ на виявлення специфічних помилок;

- кількість можливих паралельних потоків.

Для оптимізації рішення, гарним підходом є паралелізація алгоритму. Якщо комп'ютерна програма або система паралелізована, це розбиває проблему на більш

дрібні фрагменти, які можна самостійно вирішити одночасно за допомогою дискретних обчислювальних ресурсів

Найкращим рішенням було б розпаралелення кожного етапу алгоритму на свій потік, але це неможливо, адже є залежні між собою кроки, що повинні бути виконані послідовно. Жодна програма не може працювати швидше, ніж найбільший ланцюг залежних обчислень, бо вони повинні виконуватися послідовно, цей ланцюг також називають критичним шляхом обчислення.

У даній системі є декілька основних етапів роботи (детальніше на рисунку 2.6):

- перевірка вхідних документів на відповідність вимогам для валідації форматування (перевірка розширення документа, розміру, тощо);
- перероблробка word документа;
- конвертація word документа у pdf документ;
- перероблробка pdf документа;
- валідація форматування вхідних word та pdf документів;
- формування списку помилок форматування та відповідь.

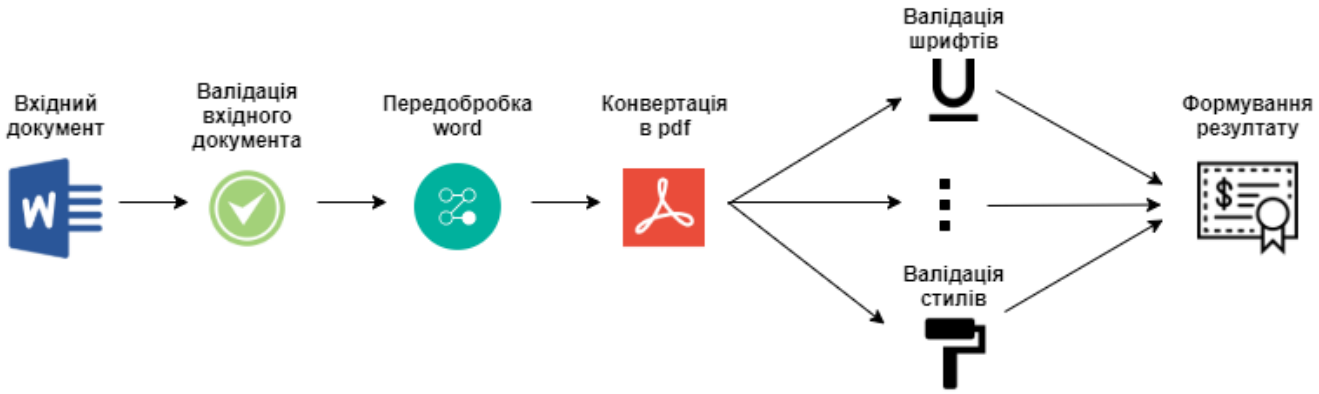


Рисунок 2.6 – Залежні етапи алгоритму валідації форматування документа

Дані етапи є залежними, а тому їх дуже складно розпаралелити. Найдовшим з цих етапів, є етап саме валідації форматування вхідних документів, що має під собою

велику кількість алгоритмів, що вже можуть бути розбиті на декілька потоків. Також довгими є етапи перероблобки та конвертації документів, що в свою чергу не можуть бути розпаралелені. Одним з варіантів прискорення даних кроків є звільнення від конвертації у pdf шляхом запиту готового pdf документа у користувача.

Отже, найбільший ефект буде саме від паралелізації алгоритмів аналізу форматування. Ті алгоритми, що працюють виключно із word або pdf, можуть бути розбиті паралельно досить просто, тоді як алгоритми, що працюють з обома форматами розбиваються складніше й працюють довше.

Якщо взяти кількість валідаційних алгоритмів форматування, що перевіряють тільки один формат документа (word або pdf) за k, тих що перевіряють за обома типа (word та pdf) за h, час роботи одного алгоритма валідації форматування за r (2r - час роботи одного алгоритма валідації для обох форматів), кількість ядер (максимальну паралельність) за n, час роботи підготувальних та фінальних етапів за q то можна вирахувати приблизний час перевірки документа за наступною формулою:

$$t = 5q + \frac{(r * k) + (2r * h)}{n}$$

Тобто основний вплив на ефективність алгоритму має кількість валідаційних кроків, яких можуть бути десятки і саме на цих етапах дуже важлива їх паралелізація задля максимально швидкої відповіді користувачу. Тоді як послідовні частини даного алгоритму не можуть бути розбиті і тому єдиним варіантом прискорення цієї частини є покращення обладнання, на якому буде виконуватися алгоритм, або вертикальна оптимізація.

Приклад результатів апробації такого підходу паралелізації можна знайти у додатку Б, де на прикладі іншого алгоритму прописані результати послідвного алгоритму та паралельного.

3 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Для проектування програмної системи необхідно сформулювати певні вимоги, такі як: яку архітектуру матиме програмна система, з яких частин вона буде складатися, які технології слід застосувати, для кращого результату. Така система повинна поєднувати у собі всі переваги та виключати недоліки вже існуючих систем. Отже, програмна система складається з двох основних частин: сервер та веб частина. Сервер, має надавати публічне API для обробки запитів з веб-додатку. Це API має мати методи для валідації docx та doc документів, та вертати список невідповідностей до сконфігурованих стандартів.

Серверна частина повинна розроблятися за допомогою мови C# та платформи .Net Core. Для розробки API повинен використовуватися фреймворк ASP.NET Core , що є кросплатформенною, високопродуктивною системою з відкритим кодом для створення сучасних додатків, підключених до Інтернету, на базі хмарних технологій.

Клієнтська частина повинна розроблятися за допомогою Angular 8, платформи для побудови мобільних та настільних веб-додатків.

Система повинна бути розгорнута у Microsoft Azure - це сервіс хмарних обчислень, створена корпорацією Майкрософт для побудови, тестування, розгортання та керування програмами та послугами через центри обробки даних, керовані Microsoft.

Отже програма повинна мати наступний функціонал:

а) система має декілька способів взаємодії з користувачем:

1) через консольне застосування:

– на вхід подається документ docx;

– на вихід - список помилок;

2) через веб-клієнт:

– на вхід подається документ docx;

- на вихід - список помилок.

б) система повинна аналізувати документ на предмет порушення заданих правил форматування:

1) документ не повинен бути порожнім;
2) перша сторінка домента повинна бути титульною:
   - відсутня нумерація сторінки;
   - є шапка «Міністерство …»;
   - в кінці сторінки поточний рік по центру;
3) повинна бути сторінка зі змістом:
   - сторінка містить елемент Table of Content;
   - нумерація сторінок збігається з реальною;
   - пункти розміщені по порядку і збігаються з заголовками документа;
   - всі пункти написання в форматі речення (перша буква велика, далі маленькі);
4) перед сторінкою зі змістом знаходиться сторінка з заголовком «РЕФЕРАТ / ABSTRACT»:
   - у першому абзаці тексту вірно зазначено кількість малюнків, таблиць, схем, додатків, джерел;
5) документ повинен містити наступні розділи (заголовки):
   - ВСТУП
   - ВИСНОВКИ
   - ПЕРЕЛІК ПОСИЛАННЯ
6) абзацний відступ по всій записці дорівнює п'яти знакам або 1,25 см. поля:
   - ліве - 2,5 см;
   - праве - 1 см;

- верхнє і нижнє - 2 см;

7) форматування по всьому документу:
   - інтервал основного тексту – полуторний;
   - шрифт тексту - Times New Roman, чорний, 14;
   - по всьому Документу, починаючи з розділу «ВСТУП» повинна бути нумерація (у правому верхньому куті);
   - сторінка не повинна закінчуватися заголовком;
   - використання правильного символу «тире»;
   - сторінку не може починатися картинкою;
8) розділи:
   - кожен розділ починається з нової сторінки;
   - на останній сторінці розділу повинно бути мінімум 10 рядків;
   - розділ не може закінчуватися малюнком, таблицею, схемою;
   - загаловкі основних розділів - великими символами;
   - заголовки і підзаголовки не закінчуються крапкою;
   - тема розділу - з абзацного відступу або по центру;
   - підзаголовки розділів - з абзацного відступу;
   - два відступу після заголовка або підзаголовка перед текстом (звичайний відступ між заголовком і підзаголовком);
   - нумерація заголовків і підзаголовків без точки (не "1.1.», А «1.1»);
   - нумерація розділів - по порядку розміщення;
9) додатки:
   - додатки знаходяться в кінці Документа і нумеруються А, Б, В;
   - форматування додатків ігнорується;
10) список джерел
   - пронумерований;

- згадки джерел присутні в тексті (мінімум одна згадка на джерело), в форматі «[1]»;
- згадки розташовані по порядку згадування в тексті;

11) перерахування:

- використовуються тільки букви, цифри або знак «тире»;

12) рисунки:

- нумерація - наскрізна або по розділах;
- у додатках нумерація - А.1, А - номер додатка;
- підпис рисунка - "Рисунок 1.1 - Назва рисунка";
- відступ перед рисунком, після малюнка і перед текстом після підпису;
- у тексті має бути присутнє посилання на рисунок (перед самим малюнком);
- рисунок розташований по центру;

13) таблиці:

- назва таблиці - над таблицею;
- формат назви - «Таблиця 1.1 - Назва таблиці»;
- назва стовпців таблиці розміщується по центру осередки таблиці;
- присутній відступ перед і після таблиці;

14) формули:

- відступ перед і після формули;

в) результатом аналізу має бути список помилок і попереджень про знайдені порушення форматування.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Опис архітектурних рішень системи

Серверна частина реалізована використовуючи мову програмування C#, .NET Core 2.1 та фреймворк для створення веб-додатків, який реалізує шаблон Model-View-Controller ASP.NET MVC. Розгортається все на сервері, який розміщено у хмарному середовищі Azure, де встановлений, та налаштований IIS. Потужність хмарного середовища повинна витримувати одночасну декілька запитів. Веб частину даної системи буде представляти Angular 9 веб додаток, що буде займатися взаємодією з сервером та відображенням цієї інформації користувачу.

Для парсингу та валідації docx та doc документів використовується бібліотека Spire.Doc for .Net [9]. Spire.Doc for .NET - це професійна бібліотека Word .NET, спеціально розроблена для розробників для створення, читання, запису, перетворення та друку файлів документів Word з будь-якої платформи .NET із швидкою та якісною продуктивністю.

Як незалежний API .NET Word, для Spire.Doc .NET не потрібно встановлювати Microsoft Word ні на стадії розробки, ні на цільових системах. Однак він може додавати можливості створення документів Microsoft Word у.NET будь-яких додатках.

Він підтримує C #, VB.NET, ASP.NET і ASP.NET MVC. Spire.Doc підтримує Word 97-2003 / 2007/2010/2013/2016, і він має можливість конвертувати їх у часто використовувані формати файлів, такі як XML, RTF, TXT, XPS, EPUB, EMF, HTML і навпаки. Крім того, він підтримує високу якість перетворення Word Doc / Docx в PDF за допомогою C #, Word в SVG, Word в PostScript, Word в PCL (Printer Command Language).

Для роботи з файлами типу pdf в системі використовується бібліотека Spire.PDF.[10] Spire.PDF for .NET - це професійний PDF-API, який застосовується для

створення, написання, редагування, обробки та читання PDF-файлів без зовнішніх залежностей у програмі .NET (C #, VB.NET, ASP.NET, .NET Core). Використовуючи цю бібліотеку є можливості для створення PDF-файлів з нуля або повністю, обробляти існуючі PDF документи через C # / VB.NET, не встановлюючи Adobe Acrobat.

Для деяких маніпуляції з PDF документами використовується PDFsharp [11]. PDFsharp - це бібліотека з відкритим кодом .NET, яка легко створює та обробляє PDF-документи на льоту з будь-якої мови .NET. Її також можна використовувати для створення документів PDF, малювання на екрані або надсилання результатів на будь-який принтер.

Структура серверної частина складається з 6 основних проектів, кожен із яких відповідає за свою частину. [12] На рисунку 4.1 можна побачити взаємодію компонентів.

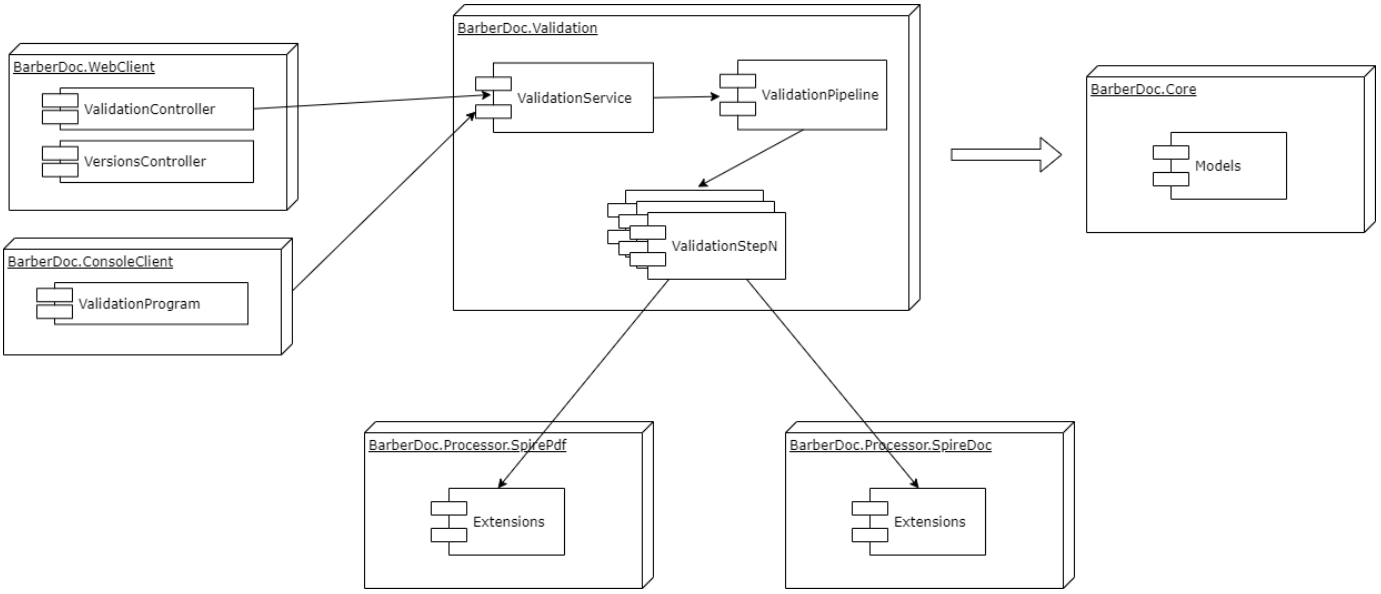


Рисунок 4.1 – Структура проекту

Основними елементами структури є:

- BarberDoc.ConsoleClient – проект, що відповідає за логіку роботи валідації у консолі;

- BarberDoc.WebClient – проект, що відповідає за логіку взаємодії з клієнтом через http запити;

- BarberDoc.Validation – основна логіка валідації знаходиться саме в цій частині. За кожне правило форматування відповідає окремий ValidationStep (їх у системи більше 20), який має свою логіку валідації;

- BarberDoc.Core – відповідає за логіку, яка повинна бути доступна усім проектам системи. Включає в себе основні моделі та інфраструктурні рішення;\

- BarberDoc.Processors.SpirePdf – частина, яка відповідає за взаємодію з docx та doc файлами, через бібліотеку Spire.Doc. Включає в себе основні методи для отримання даних з word документів;

- BarberDoc.Processors.SpirePdf – частина, яка відповідає за взаємодію з Pdf файлами, через бібліотеку Spire.PDF. Включає в себе основні методи для отримання даних з Pdf документів.

Для обробки документу Angular додаток повинен прислати docx або doc документ на ValidationController. Даний контролер, у свою чергу, зконвертує docx документ в pdf через ConvertService (за потреби, якщо користувач сам не представив pdf документ) та відправить на валідацію до ValidationService. ValidationService має ValidationPipeline, як в собі має ValdiationSteps що і відповідають за аналіз форматування по кожному з правил. ValidationService виконує паралельно декілька ValdiationSteps та потім отримує й агрегує з них інформацію з помилками. Далі ці помилки віддаються на ValidationController який у свою чергу віддає їх користувачу. Детально цей процес можна пробачити на рисунку 4.2. Даний процес проходить асинхронно та паралельно, але, оскільки обробка файлу (середній розмір 30 мб) є дуже важким процесом, то данні операції займають достатньо велику кількість часу. Тому, даний сервіс не пристосований до великих навантажень на даний час, та здатен витримати лише декілька паралельних викликів валідації. Саме тому було прийнято рішення максимально розпаралелити валідацію за кожним із правил, щоб прискорити час відповіді на запит до системи [13].

Кожен ValidationStep має у собі індивідуальну складну логіку отримання інформації форматування по тому чи іншому правилу форматування, та може приймати, як вхідний параметр або docx, або pdf та docx.

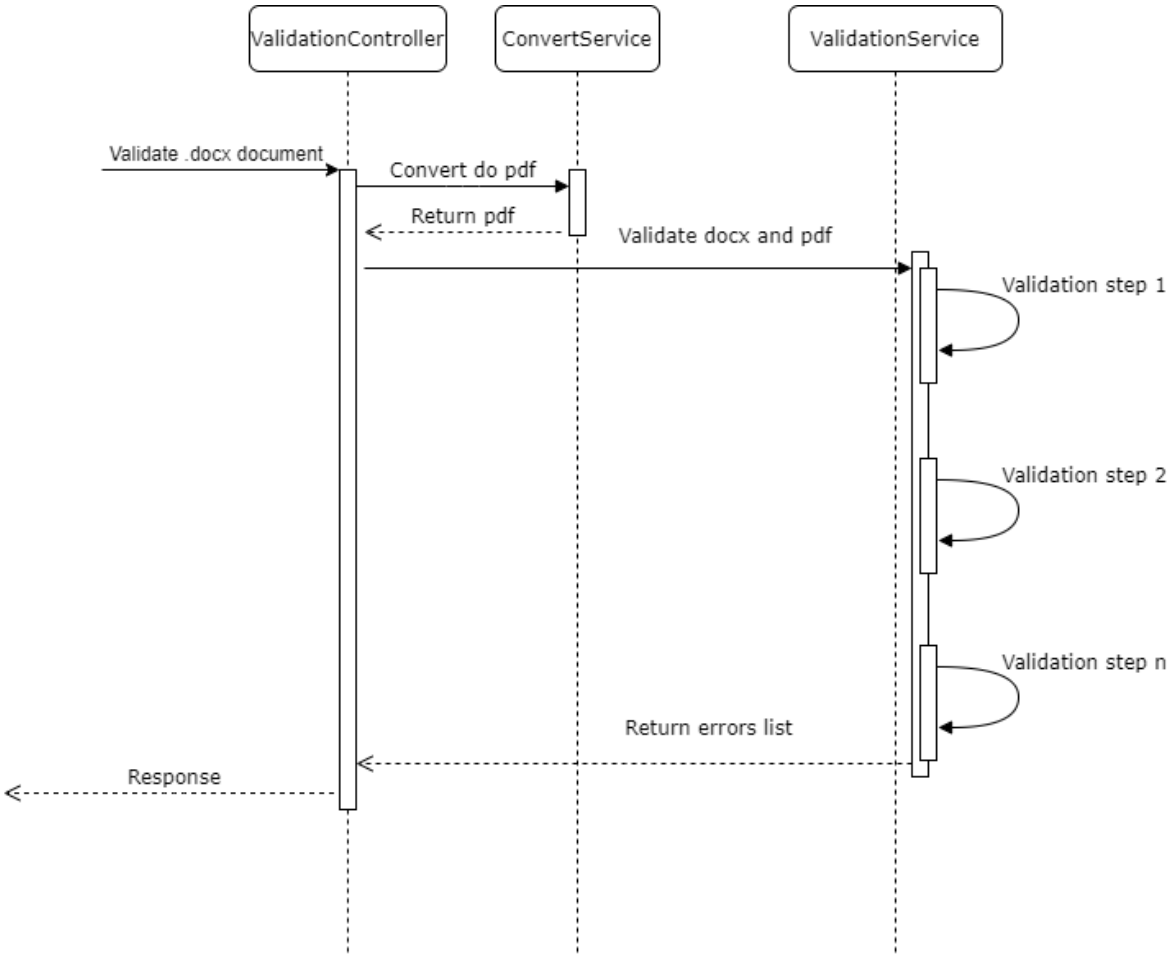


Рисунок 4.2 – Діаграма потоку запиту валідації документу

Даний процес можна прискорити розбивши систему на декілька сервісів, що опрацьовують свій тип документу [14]. Тобто може бути окремий сервіс для обробки docx документів, а інший буде обробляти pdf. Даний підхід може розподілити навантаження на систему, що в свою чергу дасть змогу обробляти більшу кількість запитів одночасно.

4.1 Опис інтерфейсу користувача

Інтерфейс даного програмного забезпечення є дуже простим та ергономічним. Є два поля, куди треба перетягнути doc або docx файл (обов'язково) та pdf файл (опціонально). Pdf файл потрібен для прискорення роботи системи, аби їй не довелось конвертувати його самостійно. Побачити початковий інтерфейс можна на рисунку 4.3.

Далі треба натиснути кнопку "Validate" та дочекатися кінця валідації. Після того, як система проаналізує папери, вся інформація про помилки форматування будуть виведені на екран.

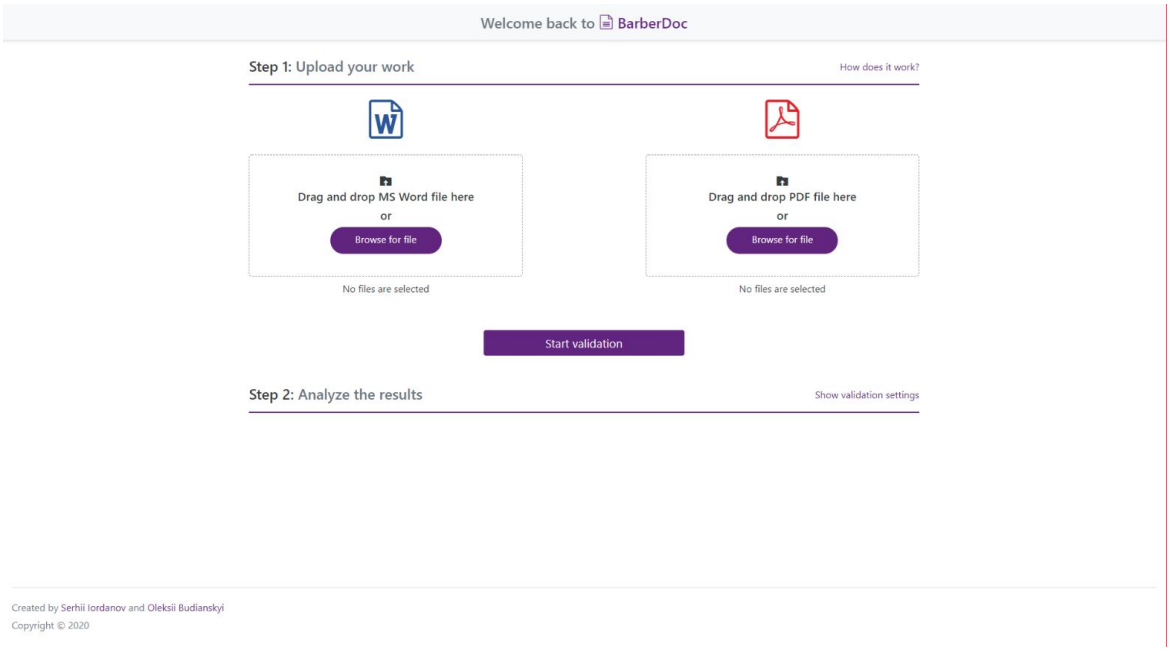
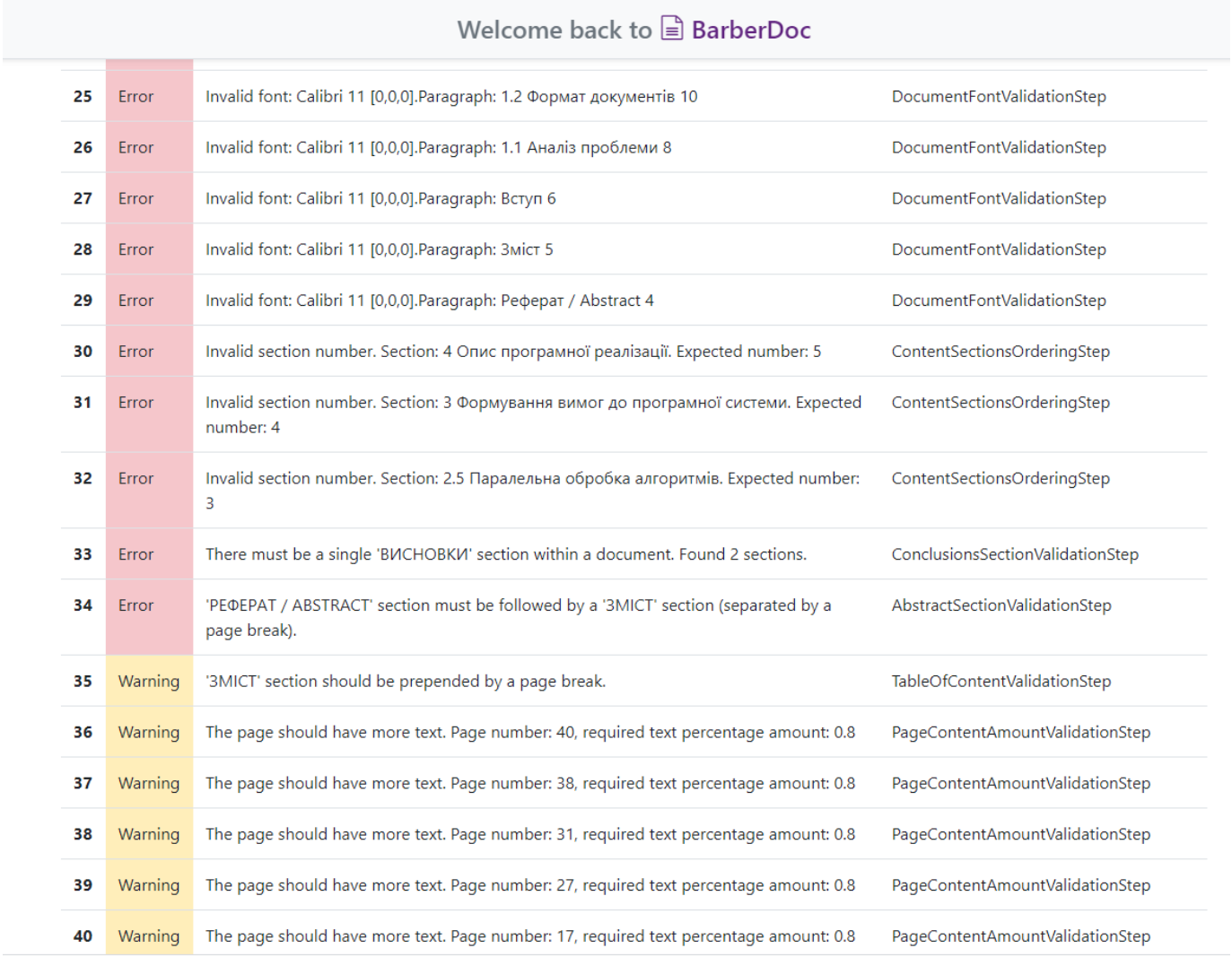


Рисунок 4.3 – Початковий інтерфейс системи

Існує три типи помилок, що відрізняються за своєю значимістю [15]:

- Error – найкритичніші помилки, які повинні бути вирішені в першу чергу;
- Warning – помилки середньої значимості;
- Info - невеликі похибки в форматуванні документа;

Приклад відображення помилок на екрані користувача можна побачити на рисунку 4.4.



25	Error	Invalid font: Calibri 11 [0,0,0].Paragraph: 1.2 Формат документів 10	DocumentFontValidationStep
26	Error	Invalid font: Calibri 11 [0,0,0].Paragraph: 1.1 Аналіз проблеми 8	DocumentFontValidationStep
27	Error	Invalid font: Calibri 11 [0,0,0].Paragraph: Вступ 6	DocumentFontValidationStep
28	Error	Invalid font: Calibri 11 [0,0,0].Paragraph: Зміст 5	DocumentFontValidationStep
29	Error	Invalid font: Calibri 11 [0,0,0].Paragraph: Реферат / Abstract 4	DocumentFontValidationStep
30	Error	Invalid section number. Section: 4 Опис програмної реалізації. Expected number: 5	ContentSectionsOrderingStep
31	Error	Invalid section number. Section: 3 Формування вимог до програмної системи. Expected number: 4	ContentSectionsOrderingStep
32	Error	Invalid section number. Section: 2.5 Паралельна обробка алгоритмів. Expected number: 3	ContentSectionsOrderingStep
33	Error	There must be a single 'ВИСНОВКИ' section within a document. Found 2 sections.	ConclusionsSectionValidationStep
34	Error	'РЕФЕРАТ / ABSTRACT' section must be followed by a 'ЗМІСТ' section (separated by a page break).	AbstractSectionValidationStep
35	Warning	'ЗМІСТ' section should be prepended by a page break.	TableOfContentValidationStep
36	Warning	The page should have more text. Page number: 40, required text percentage amount: 0.8	PageContentAmountValidationStep
37	Warning	The page should have more text. Page number: 38, required text percentage amount: 0.8	PageContentAmountValidationStep
38	Warning	The page should have more text. Page number: 31, required text percentage amount: 0.8	PageContentAmountValidationStep
39	Warning	The page should have more text. Page number: 27, required text percentage amount: 0.8	PageContentAmountValidationStep
40	Warning	The page should have more text. Page number: 17, required text percentage amount: 0.8	PageContentAmountValidationStep

Рисунок 4.4 – Список помилок форматування документа

На даний момент система генерує більше 30 типів помилок, що відображаються користувачу як текст помилки та частина параграфу, в якій цю помилку було знайдено. Основні помилки, що на даний момент може повернути система перераховані у таблиці 4.1.

Таблиця 4.1 – Можливі помилки форматування

Тип	Валідаційне правило	Повідомлення	Опис
Warning	AbstractSection Validation	'{SectionTitle}' should have '{StyleName}' style applied.	Заголовок повинен мати вказаний стиль.
Warning	AbstractSection Validation	'{SectionTitle}' section should be prepended by a page break.	Перед заголовком повинен бути розрив сторінки.
Error	AbstractSection Validation	'{SectionTitle}' section title must be followed by {number} empty lines.	Після зоголвку повинно бути n пустих рядків.
Error	AbstractSection Validation	'{SectionTitle}' section must be followed by a '{NextSectionTitle}' section (separated by a page break).	Заголовок повинен йти у правильній черзі після іншого заголовку.

Продовження таблиці 4.1

Error	AbstractSection Validation	There must be a single '{ExpectedTitle}' section within a document. Found {foundCount} sections.	Кількість вказаних заголовків не відповідає вимогам.
Error	ContentSection Validation	No content sections are found between {introductionSectionTitle} and {conclusionSectionTitle}	Не знайдено секції змісту.
Error	Font Validation	Invalid font: {FontName}{FontSize}{TextColor)}	Шрифт не відповідає вимогам.
Error	DocumentMargins Validation	Invalid section margin {marginType}.Actual margin {actualMargin}. Expected margin: {expectedMargin}.	Відступи не відповідають вимогам.
Error	DocumentSpacings Validation	Invalid {type} spacing {spacingValue}	Інтервали не відповідають вимогам.
Error	EmptyDocument Validation	The document must not be empty	Документ не повинен бути пустим.
Error	PageContentAmount Validation	The document should not have pages without any text. Page number: {currentPage}	Не повинно бути пустих сторінок.

Продовження таблиці 4.1

Warning	PageContentAmount Validation	The page should have more text. Page number: {currentPage} required text percentage amount: {ContentPercentage}	Сторніка повинна мати більше контенту.
Error	PageImage Validation	The page is empty or has only images. Page number: {currentPageNumber}"	Сторінка має тільки зображення.
Error	PageImage Validation	The page should have more text to have image. Page number: {currentPageNumber}	Сторінка повинна мати більше тексту щоб мати зображення.
Error	PageImage Validation	The page should not start with image (at least 2 text lines before). Page number: {currentPageNumber}	Сторінка не повинна починатися з зображення.
Error	PageImage Validation	The page should not end with image (at least 2 text lines after). Page number: {currentPageNumber}	Сторінка не повинна закінчуватися зображенням.
Error	PageNumbering Validation	Can`t find page number in page {currentPageNumber}	На сторінці немає нумерації.

Продовження таблиці 4.1

Warning	PageNumbering Validation	Page number does not reflect real numbering of the pages. Specified page: {pageNumber}. Count: {currentPageNumber}	Нумерація сторінок не відповідає дійсності.
Warning	PageNumbering Validation	Invalid position of page numbering. Page number:{pageNumber}	Позиція нумерації на сторінці не відповідає вимогам.
Error	RestrictedSymbols Validation	Restricted symbol found:{restrictedSymbol}.	Знайдено символи, яких не повинно бути в документі.
Error	TableOfContent Validation	Table of Content section must contain a Table of Content element before the next page break.	Секція змісту повинна мати елемент “Зміст”.
Error	TitlePage Validation	Title (the first) page must be started with a text paragraph and finished with a page break.	Титульна сторінка повинна починатися текстом та закінчуватися розривом сторінки.

Кінець таблиці 4.1

Error	TitlePage Validation	Main title text is not found on title page. Expected: '{ Title}'	Титульна сторінка не має основного тексту.
Error	TitlePage Validation	Current year text on the title page is invalid.	Рік вказаний не вірно на тиутльній сторінці.
Error	TitlePage Validation	University must be placed as a second paragraph of title page header.	Інформація про університет повинна іти другим парагрофом на титульній сторінці.
Error	TitlePage Validation	Title page header text is invalid.	Заголовок титульної сторінки не є вірним.

Наступними кроками у розробці даного додатку є адаптування його до інших стандартів, розбиття на мікросервіси задля прискорення процесу обробки документів, та введення авторизації та автентифікації для обмеження кількості перевірок на одного користувача та рівномірного розподілу між користувачами обчислювального часу [16].

ВИСНОВКИ

В роботі було здійснено огляд на підходи до аналізу форматування документів типу docx. Як було зазначено, наразі не існує систем для online-перевірки документу на відповідність тому чи іншому стандарту, але дане питання є дуже актуальним у різних галузях, адже воно може сильно підвищити ефективність людей, прибравши рутинну роботу і давши змогу зосередитися на більш важливих питаннях, і дана робота здійснює ще один крок у напрямку вирішення цієї проблеми.

За результатами проведених досліджень можна побачити проблему, через яку, автоматизація валідації форматування документів наразі слаборозвинена. Причиною цього є факт того, що існуючі формати для написання документів не в повній мірі надають засоби для верифікації форматування [17].

В ході роботи було виявлено, що потрібно застосовувати комбіновані підходи до аналізу форматування документів, адже немає наразі такого формату документа, що дав би змогу провалідувати його по всім правилам стандартів. Формат docx є дуже вдалим вибором для перевірки [18]:

- стиль тексту (шрифт, розмір тексту, формат тощо);
- поля, інтервали та відступи (абзацний відступ, поля документу, інтервал між рядками тощо);
- зміст контенту (семантичний аналіз тексту);
- формули, таблиці, діаграми тощо.

З іншого боку є формат pdf, що є вдалим вибором для:

- перевірки розташування контенту,
- валідації графічних елементів,
- правильність відображення контенту.

Саме тому, конвертування документів у різні формати, задля отримання усієї необхідної інформації, є основною роботи такої системи.

Переваги розроблених методів полягають у їх гнучкості та широкому колу використання, недоліки ж полягають в часі обробки документів, адже аналіз документа є дуже важким процесом.

В ході роботи було зроблено наступні висновки: автоматизація перевірки форматування документа є цілком реальною задачею, що потребує розвитку, адже має дуже високий потенціал. Використання docx, з дуже гнучкою xml структурою всередині, а також pdf, з бібліотекою для розпізнавання елементів на ньому, є гарним варіантом комбінації, для перевірки тих чи інших правил [19].

Було розроблено програмні модулі мовою C#, для автоматизації перевірки та нормоконтролю атестаційних робіт студентів відносно стандарту ДСТУ 3008-2015 «Звіти у сфері науки і техніки». З використанням бібліотек Spire.Doc for .NET Spire.PDF for.NET, PDFsharp та OpenXML були розроблені методи аналізу документів під різні вимоги стандарту [20].

Такі системи змогли б значно оптимізувати людські ресурси підприємств. На прикладі університету ХНУРЕ, в якому навчаються 11000 студентів, кожен студент витрачає від 1 години до 6 годин на форматування своєї роботи (дипломної роботи, курсової роботи, тощо), тобто це в середньому біля 30 000 годин на рік на рутинну роботи. Зі сторони працівників університету, що займаються нормоконтролем, економія часу може бути ще більш значною.

Саме тому подальша розробка даного програмного застосунку є актуальною та має великі перспективи для монетизації, застосовуючи її у різних сферах діяльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008-2015 [Електронний ресурс] – режим доступу: http://www.knmu.kharkov.ua/attachments/3659_3008-2015.PDF. (дата звернення: 20.03.2020)

2. Technopreneurship in Ukraine. How to Boost Enterpreneurial Competence Development in the Ukrainian IT Industry (The Ukrainian IT Educational System: Basic Facts and Urgent Needs. A. Mendes, Z.Dudar, V. Kauk, T. Shatovska, I. Revenchuk, A. Chupryna, D. Fedasyuk, V. Yakovina, I. Lyutak) edited by Hans Lundberg.//- Linnaeus University Copycenter, 2016.- 160p. ISBN: 978-91-88357-68-7

3. [MS-DOCX]: Word Extensions to the Office Open XML (.docx) File Format [Електронний ресурс] – режим доступу: https://docs.microsoft.com/en-us/openspecs/office_standards/ms-docx/b839fe1f-e1ca-4fa6-8c26-5954d0abbccd. (дата звернення: 20.03.2020)

4. Office Open XML [Електронний ресурс] – режим доступу: http://officeopenxml.com/ (дата звернення: 20.03.2020)

5. Standard ECMA-376 [Електронний ресурс] – режим доступу: https://www.ecma-international.org/publications/standards/Ecma-376.htm (дата звернення: 20.03.2020)

6. Bohdan Sus, Ilona Revenchuk, Nataliia Tmienova,Vira VialkovaDevelopment of Virtual Laboratory Works for Technical and Computer Sciences.- 25th International Conference on Information and Software Technologies, ICIST 2019.- Vilnius, Lithuania, October 10–12, 2019.- pp 383-394

7. ISO/IEC 29500-1:2016 [Електронний ресурс] – режим доступу: https://www.iso.org/standard/71691.html (дата звернення: 20.03.2020)

8. ISO/IEC 14496-22:2007 [Електронний ресурс] – режим доступу: https://www.iso.org/standard/43466.html (дата звернення: 20.03.2020)

9. Spire.Doc for .NET [Електронний ресурс] – режим доступу: https://www.e-iceblue.com/Introduce/word-for-net-introduce.html#.XnTuM6gzaUk (дата звернення: 20.03.2020)

10. Spire.Pdf for .NET [Електронний ресурс] – режим доступу: https://www.e-iceblue.com/Introduce/pdf-for-net-introduce.html#.XnTuiagzaUk (дата звернення: 20.03.2020)

11. Home of PDFsharp and MigraDoc Foundation [Електронний ресурс] – режим доступу: http://www.pdfsharp.net/ (дата звернення: 20.03.2020)

12. І.А. Ревенчук, К.В. Перцьова, О.І. Маренич. Програмна реалізація кластеризації пошукових запитів.- Біоніка інтелекту.-№.-2019.-С.7-14

13. І.А. Ревенчук. Математична модель агрегації даних в соціальних медіа.- Сб. наук. праць "Математичне та комп'ютерне моделювання. Серія: Техн. Науки" за Міжнар. Наук. Конф. “Питання оптимізації обчислень (ПОО-XLIV).- Кам'янець-Подільський.- 2017.- С. 197-203

14. Bondarenko M. F., Dudar Z. V., Revenchuk I.A. Information Systems and Technologies Used in Distance Form of Education at the University.- Informational and Communication Technologies Technologies – Theory and Practice: Proceedings of the International Scientific Conference ICTMC-2010 Devoted to the 80th Anniversary of I.V. Prangishvili. Nova Science Publishers. Series: Computer Science, Technology and Applications. - 2012.- P.485-490. ISBN: 978-1-61470-050-0

15. Кренке Д. Теория и практика построения баз данных. [Текст] / Д.Кренке - 8-е изд. – СПб.: Питер, 2003. – 800 с.

16. Вигерс К.И. Разработка требований к программному обеспечению [Текст] / К.И. Вигерс - Русская Редакция, 2004 г. – 576с.

17. Троелсен Э. C# и платформа .NET. Библиотека программиста. / Э. Троелсен – СПб.: Питер, 2005. – 796 с.3. Мартин, Р. Чистый код. Создание, анализ и рефакторинг: пер. з англ. - Питер, 2010. - 464 с.

18. Постолит А.В. Visual Studio .NET [Текст] / А.В. Постолит. Разработка приложений баз данных. – СПб.: БХВ-Петербург, 2003. - 544 с.

19. Сандерсон, С. ASP.NET MVC Framework с примерами на C# [Текст] / С. Сандерсон.; пер. с англ. Н. Мухин. – М.: ООО «И.Д. Вильямс», 2010. – 560 с.

20. Хорстман, К. Бібліотека професіонала: пер. з англ. – СПб.: Вильяме, 2010. – 644с.