

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра Обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТИРЕНКО

«__» _____ 20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Автоматичне вирішення конфліктів в тексті»

Виконав (-ла):

студент (-ка) IV курсу, групи ІО-61

Рябушкін Валерій Вікторович _____

Керівник:

Професор, д. т. н.,

Луцький Георгій Михайлович _____

Консультант з нормоконтролю:

Професор кафедри ОТ, д.т.н.,

Сімоненко Валерій Павлович _____

Рецензент:

Доцент кафедри АУТС, к.т.н.,

Сокульський Олег Євгенович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп’ютерна інженерія»

Освітньо-професійна програма «Комп’ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Сергій СТИПЕНКО

«__» _____ 20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Рябушкіна Валерія Вікторівича

1. Тема проєкту «Автоматичне вирішення конфліктів в тексті», керівник проєкту Луцький Георгій Михайлович, д. т. н., проф., затверджені наказом по університету від «07» травня 2020р. №1081-с
2. Термін подання студентом проєкту _____
3. Вихідні дані до проєкту: технічна документація, теоретичні дані, інтернет-публікації за темою роботи
4. Зміст пояснювальної записки: проведення аналізу предметної області та огляд існуючих аналогів, проектування і розробка системи автоматичного вирішення конфліктів в тексті при спільному редагуванні документу, проведення тестування розробленої системи
5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)
6. Консультанти розділів проєкту*

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П., проф.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	01.09.2019	
2	Вивчення та аналіз завдання, огляд літератури	27.12.2019	
3	Аналіз існуючих аналогів та виявлення їх недоліків	28.01.2020	
4	Проектування та розробка системи для автоматичного вирішення конфліктів в тексті	10.03.2020	
5	Проведення тестування розробленої системи	20.03.2020	
6	Оформлення матеріалів роботи	20.05.2020	
7	Передзахист	26.05.2020	
8	Захист	...	

Студент

Валерій РЯБУШКІН

Керівник

Георгій ЛУЦЬКИЙ

Анотація

В даній бакалаврській дипломній роботі було реалізовано систему автоматичного вирішення конфліктів в тексті при спільному редагуванні одного і того ж документу. Попередньо проаналізувавши недоліки існуючих аналогів, при реалізації нашої системи було усунено частину з них.

Програма дозволяє наочно поглянути за допомогою графу змін, де були конфлікти і як вони вирішуються. Програмний продукт був створений на мові Java 8.0, JavaScript та ReactJS.

Аннотация

В данной бакалаврской дипломной работе было реализовано систему автоматического разрешения конфликтов в тексте при совместном редактировании одного и того же документа. Предварительно проанализировав недостатки существующих аналогов, при реализации нашей системы было устранено часть из них.

Программа позволяет наглядно взглянуть с помощью графа изменений, где были конфликты и как они решаются. Программный продукт был создан на языке Java 8.0, JavaScript и ReactJS.

Annotation

In this work for a Bachelor's Degree, a system for automatically resolving conflicts in the text during editing the same document together is realized. Having previously analyzed the shortcomings of existing analogues, some of them were eliminated during the implementation of our system.

The program makes it possible to get a look clearly with the help of a graph of changes, where there were conflicts and how they are resolved. The software product was created in Java 8.0, JavaScript, and ReactJS.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	A4		Завдання на дипломний проєкт	2	
2	A4	ДП 6125. 00.000 ВП	Відомість дипломного проєкту	1	
3	A4	ДП 6125. 01.000 ТЗ	Технічне завдання	3	
4	A4	ДП 6125. 02.000 ПЗ	Пояснювальна записка	67	
5	A3	ДП 6125. 03.000 Д1	Схема принципова	1	
6	A3	ДП 6125. 04.000 Д2	Схема функціональна	1	
7	A3	ДП 6125. 05.000 Д3	Схема структурна	1	
8	A4	ДП 6125. 06.000 ДА	Лістинг програми	18	

				ДП 6125. 00.000 ВП		
	ПІБ	Підп.	Дата			
Розробн.	Рябушкін В.В.			Відомість дипломного проєкту	Лист	
Керівн.	Луцький Г.М.				1	
Консульт.					1	
Н/контр.	Сімоненко В.П.				КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-61	
Зав.каф.	Стіренко С. Г.					

ТЕХНІЧНЕ ЗАВДАННЯ

**до дипломного проєкту
освітньо-кваліфікаційного рівня бакалавр**

на тему: “Автоматичне вирішення конфліктів в тексті”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ	2
2. ПІДСТАВИ ДЛЯ РОЗРОБКИ	2
3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	2
4. ДЖЕРЕЛА РОЗРОБКИ.....	2
5. ТЕХНІЧНІ ВИМОГИ.....	2
5.1. Вимоги до розробленої системи.....	2
5.2. Вимоги до програмного забезпечення	3
5.3. Вимоги до апаратного забезпечення.....	3

					<i>ДП 6125. 01.000 ТЗ</i>			
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>	Автоматичне вирішення конфліктів в тексті Технічне завдання	<i>Літ.</i>	<i>Аркуш</i>	<i>Аркушів</i>
<i>Розробив</i>		<i>Рядушкін В.В.</i>				1	3	
<i>Перевір.</i>		<i>Луцький Г.М.</i>						
<i>Н. контр.</i>		<i>Сімоненко В.П.</i>				НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ, ІО-61		
<i>Затверд.</i>		<i>Стіренко С.Г.</i>						

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку курсу «Інженерія програмного забезпечення» та курсу «Оптимізація інформації у комп'ютерних системах». Область застосування: застосування в системах спільного редагування.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського»

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є розробка нової системи для автоматичного вирішення конфліктів під час спільного редагування документів.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література, публікації в періодичних виданнях, публікації в Інтернеті щодо даної предметної області.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленої системи

- Система повинна визначати місцезнаходження конфлікту
- Коректування по часу
- Підтримка одночасного редагування документа більш ніж двома користувачами.

					ДП 6125. 01.000 ТЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		2

5.2. Вимоги до програмного забезпечення

- Java 8.0 і вище
- ECMAScript 8 і вище
- ReactJS

5.3. Вимоги до апаратного забезпечення

- Комп'ютер на базі процесору Intel Core i5 і вище
- Оперативна пам'ять - не менше 8 Гбайт.

					ДП 6125. 01.000 ТЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		3

Пояснювальна записка
до дипломного проєкту
на тему: «Автоматичне вирішення конфліктів в тексті»

Київ – 2020 року

ЗМІСТ

ВСТУП	5
РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ВИРІШЕННЯ КОНФЛІКТІВ ПРИ СПІЛЬНОМУ ДОСТУПІ ДО ФАЙЛІВ.....	10
1.1 Поняття конфлікту при спільному редагуванні текстів.....	10
1.2 Причини виникнення конфліктів при спільному редагуванні текстів...	13
1.3 Наслідки виникнення конфліктів при спільному редагуванні текстів...	15
1.4 Типи конфліктів	16
1.5 Середовище в якому виникають конфлікти.....	18
Висновок до розділу 1	21
РОЗДІЛ 2. ПРЕДСТАВЛЕННЯ ГОТОВИХ РІШЕНЬ ВИРІШЕННЯ КОНФЛІКТІВ В ТЕКСТІ	22
2.1 Методи вирішення конфліктів в тексті.....	22
2.1.1 Блокуючі методи.....	23
2.1.2. Неблокуючі методи.....	27
2.2 Операціональні перетворення.....	33
2.2.1 Моделі узгодженості	36
2.2.1.1 СС модель.....	37
2.2.1.2 ССІ модель.....	37
2.2.1.3 СSM модель.....	38
2.2.1.4 СА модель.....	38
2.2.2. Моделі даних в ОП	39

					<i>ДП 6125. 02.000 ПЗ</i>			
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>	<i>Автоматичне вирішення конфліктів в тексті</i>	<i>Літ.</i>	<i>Аркуш</i>	<i>Аркушів</i>
<i>Розробив</i>	<i>Рядушкін В.В.</i>						2	67
<i>Перевір.</i>	<i>Луцький Г.М.</i>				<i>Пояснювальна записка</i>	<i>НТУУ “КПІ ім. Ізгоря Сікорського”, ФІОТ, ІО-61</i>		
<i>Н. контр.</i>	<i>Сімоненко В.П.</i>							
<i>Затверд.</i>								

2.2.3. Операційна модель	39
2.2.4 Переваги та недоліки ОП	40
2.2.5 Приклади використання ОП в різних додатках спільного редагування.....	41
Висновок до розділу 2	43
РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ	44
3.1 Вибір підходу для реалізації	44
3.2 Модель узгодженності.....	45
3.3 Структура системи ОТ.....	45
3.4 Функції перетворення.....	47
3.5 Протокол взаємодії	47
3.6 Модель взаємодії між користувачами	53
3.7 Модель збереження даних.....	54
3.8 Підхід до обробки взаємодії користувачів	54
3.9 Стек застосованих технологій	54
3.9.1 Мова програмування	54
3.9.2 Системи збірки.....	55
3.9.3 Протокол передачі даних	55
Висновок до розділу 3	56
РОЗДІЛ 4. ТЕСТУВАННЯ СИСТЕМИ	57
4.1 Ілюстрація працездатності розробленої програми	57
4.2 Можливі вдосконалення розробленої програми.....	60
Висновок до розділу 4	62
ВИСНОВКИ.....	63

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		4

ВСТУП

В наш час в умовах діджиталізації та інформатизації ряду процесів, окрім полегшення створення, зберігання та обміну інформацією, виникає ряд загроз, які можуть викликати спотворення вищезазначеної інформації. Однією з таких проблем є одночасний спільний доступ до редагування документів, під час якого можуть не зберегтись окремі зміни, внесені одним чи декількома користувачами. Такі конфлікти можуть виникнути через співпадання часу зберігання файлу, і неможливості порівняння та об'єднання версій не шляхом склеювання, а шляхом додавання унікальних елементів, і ряду інших проблем, таких як протиріччя між внесеними виправленнями. Наприклад, може виникнути ситуація, при якій один з користувачів вирішив видалити певну частину тексту, а інший просто відредагувати її, при цьому остаточна версія документу не може бути створена з додаванням всіх змін одночасно, адже вони є взаємовиключними. В такому випадку доступ розмежовується за пріоритетами, тобто віддається перевага змінам того користувача, який має вищий ранг у певній ієрархії підприємства. І таких прикладів існує досить багато, тому проблема є суттєвою, а потреба у її автоматичному вирішенні – актуальною.

Науковими дослідженнями різноманітних аспектів автоматизації й інформаційного забезпечення систем управління, у тому числі на державному та регіональному рівнях, займалися вчені Є. Балашов, Н. Бусленко, В. Глушков, В. Гончаров, А. Дабагян, А. Іваненко, та ін. Інформаційно-аналітична діяльність органів влади, роль інформації у процесі прийняття державно-управлінських рішень є предметом досліджень А. Ахламова, В. Бакуменка, А. Дегтяра, В. Дорофійенка, І. Древицької, та ін. Питання розвитку інформаційного суспільства в Україні, впровадження в Україні технологій електронного врядування розглядають А. Баранова, О. Голобуцький, М. Демкова, І. Клименко, І. Коліушко, К. Линьова, та ін. Ряд праць, зокрема Л. Березовець, О. Бойченко, Т. Гаман, Р. Калюжного, Л. Полякової, присвячено

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		5

визначенню шляхів удосконалення організаційно-правового, кадрового, інституціонального забезпечення процесів інформатизації органів влади [1].

Дослідження електронного документообігу, впровадження та вибір системи управління документами знайшли віддзеркалення в працях Г. Асєєва; О. Матвієнко, М. Цивіна, а також інших вчених. Зростання обсягів інформації, необхідної для прийняття рішень, зазначає він, призводить до різкого збільшення кількості документів. При цьому традиційні методи роботи з останніми стають малоефективними. На допомогу приходять системи електронного документообігу (СЕД), що дозволяють створювати й опрацьовувати документи електронними засобами. У рамках СЕД електронні версії документів можуть існувати поряд з паперовими або замість них.

Дослідженню особливостей сутності і перспектив розвитку електронної комунікації в умовах глобалізації та регіоналізації світового господарства присвячено роботи таких зарубіжних вчених, як Е.Бейл, Б.Буста, Е.Вілкас, А.Гарнер, К.Кендалл, С.Крог, Х.Лефевр, М.Лінднер, Е.МакКарті, А.Саммерс та ін. Істотний внесок у дослідження шляхів запровадження сучасних форм міжнародної електронної комерції та обміну інформацією в умовах європейської інтеграції України, належить таким вітчизняним ученим, як В.Брижко, О.Василенко, М.Возний, Т.Дубовик, Т.Затонацька, І.Карташова, С.Кривошеєва, Д.Панина, В.Писаренко, та багато ін. Проте, незважаючи на значний інтерес фахівців до інформаційно-аналітичної діяльності та її вдосконалення, багато теоретичних, методологічних і практичних питань залишаються невирішеними. Більшість досліджень сфокусовані на аналізі певних елементів, що формують бізнес-системи електронних комунікацій, визначенні проблем та гальмуючих факторів її розвитку, але серед них не зазначена проблематика доступу до Інтернету та обмежень доступу до окремих ресурсів. Об'єктивна необхідність зазначених проблем, їх практична значимість і недостатня розробленість обумовили вибір теми дослідження.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		6

Метою дослідження є вдосконалення системи автоматичного вирішення конфліктів при спільному редагуванні тексту. Система повинна бути реалізована як браузерний додаток та забезпечувати наступні можливості:

1. Виявляти точне місцезнаходження конфліктів
2. Варіювати швидкодію програми
3. Спростити роботу програміста
4. Зменшити навантаження на мережу
5. Забезпечувати відмовостійкість системи

Завданнями дослідження, виходячи з поставленої мети, є наступні:

- визначення поняття конфлікту при спільному доступі до файлів та видів конфліктів;
- огляд наслідків виникнення конфліктів на різних рівнях доступу та проблем комунікаційного характеру;
- дослідження методології вирішення конфліктів при спільному доступі до файлів;
- огляд існуючих підходів до обробки взаємодії користувачів;
- дослідження причин виникнення конфліктів спільного доступу;
- огляд існуючих методів вирішення конфліктів в текстах;
- формування вимог до інформаційного забезпечення проектованої системи автоматичного вирішення конфліктів спільного доступу;
- побудова та опис елементів моделі автоматичного вирішення конфліктів в тексті;
- реалізація програмного забезпечення для автоматичного вирішення конфліктів та формування відповідних рекомендацій.

Об'єктом дослідження є система автоматичного вирішення конфліктів спільного доступу до документів через в мережі.

Предметом дослідження є організація автоматичного вирішення конфліктів спільного доступу в умовах використання ресурсів мережі.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		7

Методи дослідження. Методологія наукового пошуку базувалася на принципах пізнання, а також, логічному та системному підходах. При розв'язанні задач, направлених на досягнення мети дослідження, у нашій роботі було застосовано сукупність таких методів, як абстрагування, аналіз і синтез, порівняння, узагальнення.

Теоретико-методологічною основою роботи є наукові концепції, розроблені вітчизняними та закордонними науковцями, нормативно-правові акти України та світу у сфері інформатизації, діджиталізації, комунікацій, права та ряду суміжних галузей. В основі виконання нашої роботи лежить використання таких загальнонаукових методів дослідження: логічного, історичного та порівняльного методів для обґрунтування наукових засад і вдосконалення понятійного апарату дослідження; методів спостереження, узагальнення, абстрагування, формалізації, аналізу та синтезу для характеристики методичних аспектів створення та використання інноваційних комунікацій та визначення шляхів їх удосконалення; програмно-цільовий метод з метою обґрунтування механізмів вдосконалення спільного доступу окремих користувачів чи бізнес-компаній з урахуванням конфліктів доступу до окремих спільних ресурсів в мережі.

Інформаційну базу дослідження становлять наукові розробки в сфері комунікацій, менеджменту, інформатизації та ряду суміжних галузей, публікації вітчизняних учених, статті у профільних виданнях, матеріали мережі Інтернет, офіційні матеріали, звітні дані підприємств та організацій, які використовують інноваційні комунікації в менеджменті. База складається з літературних джерел в галузі інформаційно-комунікаційних технологій, інформатики та програмування, системного адміністрування та суміжних галузей.

Практичне значення дослідження полягає в тому, що аналіз проблем спільного доступу та напрямків їх вирішення в процесі застосування інноваційних комунікацій дозволить як підвищити ефективність діяльності

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		8

окремого підприємства в умовах спільного доступу до окремих ресурсів в мережі, так і використовувати отримані результати під час вдосконалення комунікаційної діяльності інших підприємств.

Наукова новизна дослідження. Дослідження проблематики автоматичного вирішення конфліктів спільного доступу в контексті використання інноваційних комунікацій має значну важливість для подальшого розвитку міжорганізаційних комунікацій. Розроблено вдосконалену систему автоматичного вирішення конфліктів в тексті при спільному редагуванні, яка полягає у використанні вдосконаленого алгоритму ОП. Дана система відрізняється від існуючих аналогів та передбачає наступні переваги, які відсутні у всіх існуючих системах, які використовують даний або інші підходи, наприклад підхід з повним або частковим блокуванням документу за допомогою семафорів, а саме:

1) Вперше було розроблено функцію, яка дозволяє визначити точне місцезнаходження конфлікту, який виник при спільному редагуванні.

2) Було вдосконалено модель збереження даних. Тепер, на відміну від існуючих систем, в нашій системі можна вибрати будь-яку модель збереження даних, що дозволяє варіювати швидкодію програми.

3) Було підвищено відмовостійкість системи та зменшено навантаження на сервер за рахунок використання механізму back-pressure, який забезпечує послідовне завантаження текстів в стримінговому режимі.

Структура роботи включає в себе вступ, 4 розділи з підрозділами, висновки та перелік використаних джерел.

					ДП 6125. 02.000 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1.

ТЕОРЕТИЧНІ ЗАСАДИ ВИРІШЕННЯ КОНФЛІКТІВ ПРИ СПІЛЬНОМУ ДОСТУПІ ДО ФАЙЛІВ

1.1 Поняття конфлікту при спільному редагуванні текстів

Для дослідження конфліктів при спільному доступі розглянемо спочатку сутність спільного доступу та його підгрунття.

В епоху широкого використання мережі Інтернет для комунікацій між різними компаніями застосовуються мережеві канали, які є зручними для розповсюдження маркетингової інформації, документообігу та інших дій, які вимагають обміну інформаційними повідомленнями, в тому числі і на міжнародному ринку. Для найбільш швидкого та ефективного інформаційного обміну використовуються різноманітні сучасні програми (месенджери, електронні розсилки, чат-боти тощо) та мережеві ресурси, такі, як соціальні мережі, які дозволяють комунікувати з великою кількістю клієнтів одночасно. Всі ці комунікаційні канали характеризуються як інноваційні комунікації.

Сучасні тенденції розвитку підприємницької сфери вимагають застосування агресивних методів ведення бізнесу, пошуку нових ефективних шляхів виживання на ринку. Забезпечення конкурентоспроможності бізнесу можливе за допомогою застосування сучасних інформаційних технологій, що сприяють осучасненню ведення ділової активності. Одним із видів інформаційних технологій, що безпосередньо є ефективним важелем у діяльності підприємства є Інтернет-технології.

На даний момент онлайн-технології використовують декілька типів архітектури спільного доступу до ресурсів: shared nothing, shared disk, shared everything, а також рішення на основі асинхронної або синхронної реплікації. До того ж в деяких випадках використовуються системи, які використовують розподілені транзакції [2].

При цьому можуть виникати різного виду конфлікти, пов'язані зі спробою одночасного доступу до сайту, мережевого ресурсу, диску чи файлу.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		10

Це може бути як наслідком одночасної потреби регламентованих користувачів у читанні чи редагуванні документу, так і результатом атаки зловмисників, які мають на меті завадити нормальному доступу користувачів.

В даному контексті є доцільним розглянути типи та причини обмежень у доступі до того чи іншого ресурсу чи файлу, структуруючи відомі обмеження у таблиці 1.1.

Таблиця 1.1

Класифікація типів обмежень спільного доступу до файлів

№ з/п	Тип обмеження	Види обмежень
1	Обмеження за масштабом	- локальні - глобальні
2	Обмеження за походженням	- спеціальні - випадкові
3	Обмеження за підгрунтам	- морально-етичні - релігійні - економічні - соціальні - без пекові - технічні
4	Обмеження за ступенем підпорядкованості:	- світові - загальнодержавні - регіональні - місцеві
5	Обмеження за правовою підставою:	- законні - незаконні
6	Обмеження за часом:	- постійні - тимчасові
7	Обмеження за інформаційною складовою:	- обмеження доступу до сайту - обмеження доступу до сегменту - обмеження доступу до персональних сторінок - обмеження доступу до файлів

Обмеження доступу можуть бути локальними (відсутність доступу до мережі через несплату за користування чи технічні проблеми) або глобальними (заборонено доступ до Інтернет в межах всієї країни для всіх без винятку).

Обмеження можуть бути спеціальними (обмежено доступ користувачам, які не є працівниками підприємства) або випадковими (ті ж технічні негаразди у провайдера). За такими ж критеріями розрізняються постійні (довготривалі) та тимчасові обмеження.

Обмеження можуть розповсюджуватись на весь сайт чи його окремі складові (закрита група в ФБ, персональна поштова скринька, до якої обмежений доступ іншим, документ Google Docs, до якого надано доступ тільки за запрошенням, інші види ресурсів). Обмеження може встановлювати як сам власник сайту / документу, так і держава чи інші органи.

На практиці визначення конфлікту залежить від програми, при цьому можливими факторами є семантика предметної області, деталі реалізації, деталізація документа і бажаний рівень паралелізму. Проте в загальному випадку в об'єктно-орієнтованому документі конфлікт виникає, коли два або більше користувачів одночасно змінюють один і той же об'єкт (об'єктно-орієнтований конфлікт). Однак об'єкт в документі може містити і інші об'єкти. Тому конфлікти можуть бути визначені на різних рівнях ієрархії об'єктів. Наприклад, конфлікт може виникнути тільки тоді, коли змінний об'єкт є найнижчим в ієрархії об'єктів. Для подальшого розвитку паралелізму можна визначити, що конфлікт виникає тільки тоді, коли операції одночасно змінюють один і той же атрибут одного і того ж об'єкта на різні значення (конфлікт на основі атрибутів). У даній роботі слід використати загальне визначення конфлікту: конфлікт виникає, коли два або більше користувачів одночасно змінюють одну і ту ж мету з різними намірами. Мета залежить від конкретного додатка і може бути об'єктом, атрибутом об'єкта, словом в документі, буквою в документі, абзацом або навіть цілим документом.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		12

Також конфлікти можуть виникати із-за мережевих налаштувань, розмежування доступу до окремих локальних ресурсів, одночасного доступу більшої за припустиму кількості користувачів та інших причин. Тому доцільно розглядати не тільки самі конфлікти, а й причини їх виникнення та можливі наслідки.

1.2 Причини виникнення конфліктів при спільному редагуванні текстів

Для автоматизації та діджиталізації діяльності підприємств використовуються різні можливості онлайн-документів. Спрощений варіант інформатизації підприємств реалізовано за допомогою засобів Google – Google Dosc, Google Drive та Google Spreadsheet [3].

Проте такий варіант часто є громіздким, а для вибору об'єктів за заданими параметрами потрібно застосувати ряд фільтрів, до того ж результати дій не зберігаються, і спеціалісти не можуть визначити, які об'єкти треба редагувати в першу чергу. Тому потрібне створення більш зручної оболонки для користування даними в онлайн-режимі. Такою може стати база даних на основі Google Docs, через яку можна формулювати запити та отримувати звіти у зручній формі.

Розглянемо, що ж саме є системою управління базою даних та які інструменти існують для роботи з нею. Як було зазначено у попередньому розділі, база даних - це структура даних, в якій зберігається інформація, організована у табличні структури, кожна з яких може містити кілька різних полів. Система управління базами даних (СУБД) – сукупність мовних і програмних засобів, які призначені для створення, ведення і сумісного використання БД багатьма користувачами [4, с. 18].

До основних функцій, які виконують системи управління базами даних, відносяться:

- пряме керування даними у зовнішній пам'яті;
- управління транзакціями;

- контроль буферів пам'яті;
- паралелізм. Одночасний доступ до однієї і тієї ж бази даних кількому користувачам;
- підтримка різних мов БД;
- безпека. Правила безпеки для визначення прав доступу користувачів;
- створення резервних копій та відновлення. Процеси для регулярного резервного копіювання даних і відновлення даних в разі виникнення проблем;
- цілісність. Структура БД і правила покращують цілісність даних;
- опис даних. Словник даних надає опис даних;
- ведення журналу або протоколу виконаних операцій в БД

Фундаментальні можливості СУБД:

- продукування пустої структури БД;
- заповнення та імпортування даних із таблиць іншої бази;
- доступ до даних, засоби пошуку та фільтрації даних.

В наш час, в еру поширення мережевих технологій, СУБД мають мати можливість роботи з віддаленими й розподіленими ресурсами, що знаходяться на серверах Інтернету.

Оснoву планування спеціальних програмних засобів банку даних складає СУБД. З структури СУБД важливо виділити ядро СУБД. Ядро підтримує комплекс різноманітних механізмів роботи з БД, а також надає іншу функціональність та складники, які гарантують функції тестування системи, утиліти, які забезпечують систему додатковими функціями, на кшталт збору статистики, а також механізм налагодження системи.

Виникнення конфліктів при роботі з базою даних чи окремим документом полягає в тому, що декілька користувачів можуть одночасно вносити правки в документ, і інколи ці правки можуть конфліктувати між собою.

Отже, конфлікти при користування онлайн-текстами виникають зазвичай при внесенні одночасно змін декількома користувачами, причому такі випадки

					ДП 6125. 02.000 ПЗ	Арк.
						14
Зм.	Арк.	№ докум.	Підпис	Дата		

можуть мати різну структуру, тому доцільно розглянути наслідки, а також яким чином можуть бути вирішені конфлікти в різних варіантах протиріч.

1.3 Наслідки виникнення конфліктів при спільному редагуванні текстів

Наслідком виникнення конфлікту спільного доступу може стати неможливість вчасно отримати актуальну інформацію, неможливість внести до неї певні зміни чи переглянути зміни, внесені іншими користувачами.

Конфлікт за спільний ресурс в сучасних системах виникає тільки якщо ресурс змінюється. А оскільки ми визначаємо систему як транзакційну, то всі зміни можуть виконуватися тільки всередині транзакцій. Відзначимо ще один очевидний факт, що якщо немає одночасно виконуваних транзакцій, то немає і конфлікту за спільний ресурс. З іншого боку, одночасні транзакції можуть не стикатися за спільний ресурс. Які транзакції будуть конфліктувати, а які ні – визначається трафіком. Складність моделювання конфлікту в системі, що використовує shared everything архітектуру, в тому, що існують два рівня конфлікту. Перший – конфлікт вузлів безпосередньо за сторінку пам'яті, а другий – це конфлікт безпосередньо за ресурс на сторінці (рядок таблиці, індексний вхід, undo вхід і т.д.) [5, с. 65].

Також наслідком виникнення конфлікту може стати порушення безпеки окремого ресурсу, наприклад, через ДДоС-атаки порушено доступ до серверу організації, зміни у документах вчасно не збережені, існує загроза пошкодження, викривлення чи втрати інформації [6].

Нині, в часи коли важливі дані здебільшого зберігаються у електронному вигляді, а не на інших альтернативних носіях, проблема збереження інформації від втрати чи внесення несанкціонованих змін, які потягнуть за собою викривлення вищезазначеної інформації постає на порядку денному. Чим більш вагомими, конфіденційними та значними є дані, тим вищим є відсоток ризику спроб несанкціонованого доступу до них. Причиною та наміром подібних дій та спроб може бути як викрадення копії

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		15

конфіденційної інформації, знищення чи викривлення даних про певних осіб чи діяльність підприємств, організацій, державних органів тощо, так і продукування проблем доступу до інформації в той час, коли вона потрібна користувачам. За цим стоять зловмисники, які мають на меті матеріальне збагачення шляхом здирництва, шантажу, продажу конфіденційної інформації зацікавленим особам, надання послуг умисного викривлення інформації, приховування певних відомостей про посадових чи публічних осіб, а інколи такі вчинки здійснюються просто з метою хуліганства, проте всі подібні дії є злочинними і потребують відповідного захисту від них.

Таким чином, виходячи з типу загрози та можливого наслідку виникнення конфлікту спільного доступу, потрібно оцінювати її, користуючись відповідними методами, а також застосовувати відповідну методiku вирішення конфлікту.

1.4 Типи конфліктів

Конфлікти можуть виникнути при операції злиття, розгалуженні від різних джерел. При змінненні одного і того ж рядка різними користувачами виникає конфлікт. Тобто якщо один користувач внесе зміни в документ та виконає злиття – то конфлікту не буде. Але якщо після цього інший користувач змінить документ в тих же рядках, що і перший – то виникне конфлікт. В такому випадку конфлікт повинен вирішувати другий користувач (власне, через якого і виник конфлікт), або система. Оскільки доступ до ресурсу в онлайн-режимі може відбуватись одночасно або з невеликою різницею в часі, то вирішення «першості» внесення змін може бути нелегким, оскільки час внесення змін у користувача і на сервері може відрізнятись.

Конфлікти бувають вирішувані і не вирішувані. Вирішуваний конфлікт – це конфлікт, який можна отримати рішення без участі третьої особи чи домовленості між користувачами, які створили конфлікт. Вирішуваний конфлікт зазвичай не містить взаємовиключних виправлень в документі. Невирішуваний конфлікт містить повністю або частково протилежні зміни

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		16

(наприклад, один користувач бажає видалити все слово, а інший тільки додати в нього літери чи видалити їх), тоді вирішення конфлікту не може проводитись в автоматичному режимі, окрім випадків, коли встановлена пріоритетність змін за тим чи іншим користувачем.

Серед типів конфліктів доступу можна виокремити наступні типи (Табл. 1.2):

Таблиця 1.2

Класифікація типів конфліктів спільного доступу до файлів

№ з/п	Тип конфлікту	Види конфліктів
1	За масштабом	- локальні конфлікти доступу до файлу чи його частини - глобальні конфлікти доступу до всього ресурсу чи серверу
2	За походженням	- спеціальні конфлікти, які виникають з-за примусового розмежування доступу - випадкові конфлікти, які виникають з-за одночасного редагування
3	За підґрунтям	- без пекові конфлікти, викликані несанкціонованим доступом до файлу - технічні конфлікти, викликані збоюми серверу чи комп'ютеру користувача, переривам в роботі мережі тощо
4	За часом:	- постійні, викликані одночасним редагуванням документу - тимчасові, викликані перевищенням пропускнуої спроможності каналу
5	За інформаційною складовою:	- конфлікт доступу до сайту, де розташований текст - конфлікт доступу до окремих файлів

Існує 2 основні типи вирішення конфліктів доступу:

1) Ручний режим:

- Шляхом ручних змін конфліктних рядків.
- Примусовий запис своєї версії поверх попередньої (зафіксованої).

2) Автоматичний режим:

- Ординальна заміна. Аналог примусового запису, за винятком того, що кожен має свій власний пріоритет, і в залежності від цього пріоритету записуються зміни того, того хто має найвищий пріоритет.

Отже, в залежності від типу конфлікту та важливості його вирішення використовуються різні методи розв'язання проблеми. Проте при великій кількості користувачів ручний режим не є доцільним, адже він значно затримує роботу з текстами, тому доцільно розглянути можливість автоматичного вирішення конфліктів доступу.

1.5 Середовище в якому виникають конфлікти

Колаборативні програми, більш відомі як системи спільного редагування, дозволяють двом або більше розосередженим користувачам, які використовують різні комп'ютери, одночасно та спільно використовувати загальне уявлення віртуальної робочої області і керувати загальним об'єктом даних в цій робочій області. Як правило, копія спільного додатку виконується на комп'ютері кожного користувача для імітації робочої області і маніпулювання локальною копією даних, які спільно обробляються і відображаються. Наприклад, спільний додаток для малювання може відображати один і той же графік на кожному з комп'ютерів групи, а користувач на кожному з цих комп'ютерів може змінювати графік, а також переглядати зміни, внесені іншими користувачами. Один із способів зробити це - надати кожному комп'ютеру користувача локальну копію загального об'єкта даних. Загальний об'єкт даних потім управляється локальною копією користувача спільного додатка. Локальна копія загального об'єкта даних синхронізується з копіями загального об'єкта даних, які підтримуються на інших комп'ютерах за допомогою повідомлень синхронізації, якими обмінюються ці комп'ютери.

Програмне забезпечення для спільного редагування інформації можна розділити на кілька груп:

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		18

- Системи контролю версій. Конфлікти, як правило, вирішуються вручну, синхронізація відбувається тільки за бажанням користувача;

- Desktopні додатки, такі як Abiword, ACE, CoWord, Microsoft Office, MoonEdit та інші;

- Браузерні додатки, такі як Kune, Google Docs, Google Wave (розробка та підтримка припинена), Etherpad (розробка та підтримка припинена), Firepad та інші.

Залишивши за кадром системи контролю версій, докладніше порівняємо між собою варіанти десктопного та браузерного додатки спільного редагування, оскільки вони є передовими в даній предметній області.

Плюси десктопних додатків:

- Після встановлення завжди готові до роботи;

- Можуть працювати без підключення до Інтернету або локальної обчислювальної мережі;

- Як правило, підтримують різноманітні формати файлів.

Мінусом таких додатків є те, що вони вимагають установки на комп'ютер користувача або, як мінімум, завантаження дистрибутива.

Плюси браузерних додатків:

- Доступність. Не потрібно завантажувати і встановлювати дистрибутиви, найчастіше для читання або редагування документа досить перейти за прямим посиланням;

- Кросплатформеність. Вони не обмежуються платформою, для якої велася розробка, працюючи на пристроях з різними операційними системами, в тому числі на планшетах і мобільних телефонах;

Мінуси браузерних додатків:

- Необхідність підключення до мережі Інтернет або ЛВС (Локальна обчислювальна мережа);

- Підтримка різноманітних браузерів і дозволів екрану вимагає певних зусиль від розробника;

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		19

- Обмеженість протоколом НТТР.

Оскільки доступність є важливим параметром, буде розроблено саме браузерний додаток для спільного редагування тексту.

Конфлікт - це поширене явище у співпраці між людьми, і управління конфліктами є однією з центральних проблем при розробці систем спільного редагування.

					ДП 6125. 02.000 ПЗ	Арк.
						20
Зм.	Арк.	№ докум.	Підпис	Дата		

Висновок до розділу 1

Отже, розглянувши поняття конфлікту спільного доступу, ми визначили, що конфліктом є неможливість одночасного спільного доступу для читання, редагування чи внесення інших змін до параметрів файлу чи іншого ресурсу декількома користувачами. Конфлікти можуть виникати через мережеві налаштування, розмежування доступу до окремих локальних ресурсів, одночасного доступу більшої за припустиму кількості користувачів та інших причин. Тому доцільно розглядати не тільки самі конфлікти, а й причини їх виникнення та можливі наслідки.

Розглянувши причини виникнення конфліктів, ми побачили, що конфлікти при користування онлайн-текстами виникають зазвичай при внесенні одночасно змін декількома користувачами, причому такі випадки можуть мати різну структуру, тому доцільно розглянути, яким чином можуть бути вирішені конфлікти в різних варіантах протиріч.

Наслідком виникнення конфлікту спільного доступу може стати неможливість вчасно отримати актуальну інформацію, неможливість внести до неї певні зміни чи переглянути зміни, внесені іншими користувачами. Отже, виходячи з типу загрози та можливого наслідку виникнення конфлікту спільного доступу, потрібно оцінювати її, користуючись відповідними методами, а також застосовувати відповідну методику вирішення конфлікту.

В залежності від типу конфлікту та важливості його вищення використовуються різні методи розв'язання проблеми. Проте при великій кількості користувачів ручний режим не є доцільним, адже він значно затримує роботу з текстами, тому доцільно розглянути можливість автоматичного вирішення конфліктів доступу.

					ДП 6125. 02.000 ПЗ	Арк.
						21
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2.

ПРЕДСТАВЛЕННЯ ГОТОВИХ РІШЕНЬ ВИРІШЕННЯ КОНФЛІКТІВ В ТЕКСТІ

Системи спільного редагування дозволяють користувачам одночасно переглядати і редагувати документи через мережу. За документ можуть виступати комп'ютерні програми і документація до них, веб-сторінки, онлайн-енциклопедії, зображення. Системи управління базами даних без певних модифікацій не можна віднести до систем спільного редагування, оскільки ті, як правило, не повідомляють підключеним користувачам про те, що дані змінилися. Для того, щоб дізнатися про зміну даних, клієнт повинен явно зробити запит у СУБД. Важливою проблемою створення систем спільного редагування є проблема контролю синхронізації і вирішення конфліктів: незалежно від дій користувачів, всі їхні дії повинні бути відображені на всіх клієнтах через час, який трішки перевищує реакцію системи. Чим більший час реакції системи, тим гостріше постає проблема вирішення конфліктів. Не знаючи про дії один одного два користувача можуть, наприклад, видалити один і той же символ, і система повинна розпізнавати такі випадки і вирішувати подібні проблеми. Як було зазначено вище, існує декілька моделей вирішення конфліктів в тексті. Розглянемо деякі з них, які можна застосовувати для вирішення конфліктів при роботі з текстами.

2.1 Методи вирішення конфліктів в тексті

Всі методи вирішення конфліктів в тексті можна розділити на дві великі підкатегорії, а саме:

1. Методи, що блокують документ повністю або частково.
2. Неблокуючі методи.

Щоб зрозуміти сутність цих методів, давайте розглянемо їх детальніше.

					ДП 6125. 02.000 ПЗ	Арк.
						22
Зм.	Арк.	№ докум.	Підпис	Дата		

2.1.1 Блокуючі методи

1. Блокування цілого документу під час редагування.

Це найбільш банальний метод. Проста реалізація і можливість роботи з усіма типами даних є єдиними перевагами цього методу. Список недоліків і незручностей напрочуд більший, а саме:

- В певний момент часу, зміни і будь-які правки може вносити тільки одна особа. Чим більш громіздким буде документ або тека і чим частіше він буде редагуватися, тим очевиднішою стає ця незручність. Уявіть, що технічне завдання складає понад 200 сторінок, при цьому решта користувачів повинна чекати, поки його друг або товариш закінчить редагування в якійсь частині документу.
- Обов'язково повинна виконуватися передача інформації про зміну в документі на сервер, згодом всі учасники повинні завантажити її. Це погіршує темп роботи і збільшує обсяг даних, які пересилаються.
- Дане рішення ніяк не може забезпечити роботу в режимі офлайн. Адже щоб приступити до редагування документу, потрібно заблокувати його на сервері від змін решти осіб. Безумовно, без з'єднання з сервером цього зробити не можна.
- Незняте блокування. Уявімо, що один користувач забув зняти блокування, так ще й на додачу до цього пішов додому або пішов у відпустку. В такому разі, решті користувачів доводиться чекати або шукати недбайливого колегу. Почасти від цього можна від цього можна піти, контролюючи час активності або надавши адміністратору можливість примусово зняти блокування. Втім, ні те ні інше не є ідеальним рішенням, оскільки в першому випадку не можна зрозуміти, на основі чого можна відрізнити відсутність

					ДП 6125. 02.000 ПЗ	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

інтернету від недисциплінованості користувача, а в другому не можна обійтися без втручання людини і чийсь зміни можуть бути втрачені.

Безперечно, даних метод зменшує кількість конфліктів, проте знижує швидкість обробки інформації та внесення змін в документ. Недоцільний при великій кількості користувачів з правом редагування.

Даний метод є досить поширеним у СЕД.

2. Блокування частини документу

Для текстового документа такою одиницею розподілу може як абзац так і таблиця. Це покращує весь процес роботи, а саме:

- Одночасне редагування декількома користувачами одного документа, за умови, що правки вносяться в різні його частини.
- Не можливість повністю заблокувати документ. Попри не зняте блокування, з рештою документа можна продовжувати працювати.

Недоліками цього методу є:

- Як і в вищезгаданому методі не можна забезпечити роботу в режимі офлайн. Для блокування будь-якої з частин документа необхідно зв'язатися з сервером.
- Заблокована при редагуванні частина документу може бути великою. Дійсно, абзац може бути досить великим і містити кілька сотень або тисяч знаків тексту. Ще більше складнощів виникає при роботі з таблицями. Без додаткових хитрощів можна одночасно вирішити змінювати вміст комірок і структуру таблиці - вставляти і видаляти рядки або стовпці. Це означає, що при будь-якій зміні одного з осередків потрібно блокувати всю таблицю. В офісних документах таблиці можуть бути різного рівня вкладеності або ж дуже великого розміру. Особливо неприємним цей момент стає при роботі з табличними документами, де робочий лист і зовсім

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		24

складається з однієї таблиці, а отже, при будь-яких змінах треба його блокувати повністю.

- З точки зору структури, документ можна спрощено уявити як послідовність абзаців і таблиць. І коли ми вставляємо або видаляємо таблиці і абзаци, то ми редагуємо цю послідовність. А це означає, що коректність спільного редагування такій послідовності також необхідно забезпечити. Без запровадження принципово іншого методу синхронізації це повинно бути знову ж блокування, тільки на цей раз об'єкт блокування буде один на весь документ.

3. Метод черги (СМО)

Системи масового обслуговування можна розділити на:

1. Системи з відмовленням – система, у якій заявка, що надійшла в момент коли всі канали зайняті, отримує відмовлення і залишає систему

2. Системи з очікування – система, у якій така заявка не залишає системи, а ставиться у чергу та очікує, поки не звільниться який-небудь канал. Час очікування та кількість місць у черзі можуть бути як необмеженими, так і обмеженими.

За числом каналів обслуговування системи масового обслуговування поділяються на одноканальні, де $n = 1$ та багатоканальні, для яких $n \geq 2$.

Одноканальна СМО – це найпростіша СМО з одним пристроєм для обслуговування. Завдяки такій СМО можливо проаналізувати деякі закономірності керування доступом. Час обслуговування (роботи користувача з документом) є випадковим, але на виході створюється потік обслуговування з інтенсивністю μ , який визначається середнім часом роботи клієнта з текстом [7].

На вхід одноканальної СМО надходить потік запитів на редагування документу інтенсивністю λ . Запит, який прибув у мить, коли система не зайнята, моментально ж починає оброблюватися. Наступний запит, який прибув у мить, коли канал обслуговування зайнятий, отримує відмову.

					ДП 6125. 02.000 ПЗ	Арк.
						25
Зм.	Арк.	№ докум.	Підпис	Дата		

Одноканальна СМО може перебувати лише в одному з двох станів: S0 – канал вільний, S1 – канал зайнятий. Граф станів ймовірних переходів з одного стану в інший наведений на рис. 2.1.

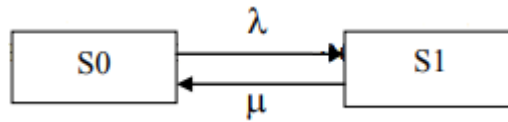


Рис. 2.1. Граф одноканальної СМО з відмовами

Система завжди знаходиться в одному з двох станів, оскільки сума ймовірностей дорівнює 1, отже:

$$p_0 + p_1 = 1, \quad (2.1)$$

де p_0 – ймовірність перебування СМО в стані S0;

p_1 – ймовірність того, що система перебуває у стані S1.

Дії, що виводять її зі стану S0 мають урівноважуватися діями, що повертають систему в стан S0, отже:

$$\lambda p_0 = \mu p_1, \quad (2.2)$$

де λ – інтенсивність потоку запитів;

μ – інтенсивність обслуговування.

З цього виразу визначаємо

$$p_1 = \frac{(\lambda p_0)}{\mu}, \quad (2.3)$$

Враховуючи, що сума ймовірностей завжди дорівнює 1, отримуємо

$$p_0 + (\lambda p_0) / \mu = 1, \quad (2.4)$$

Отже,

$$p_0 = \frac{1}{1 + \frac{\lambda}{\mu}} = \frac{\mu}{\lambda + \mu}, \quad (2.5)$$

І відповідно,

$$p_1 = \frac{\lambda}{\mu} p_0 = \frac{\lambda}{\lambda + \mu}, \quad (2.6)$$

Основні показники СМО з відмовами: відносна пропускна здатність, абсолютна пропускна здатність, а також ймовірність отримання відмови.

					ДП 6125. 02.000 ПЗ	Арк.
						26
Зм.	Арк.	№ докум.	Підпис	Дата		

Для одноканальної СМО з відмовами p_0 являється відносною пропускнуою спроможністю q , тобто $q = p_0$. Тобто це ймовірність того, що у момент часу t канал не зайнятий, і запит буде обслуговуватись.

Біля межі, коли процес уже встановився, значення відносною пропускнуою здатності СМО

$$q = \frac{\mu}{\lambda + \mu}, \quad (2.7)$$

Знайшовши відносну пропускну спроможність q , легко знайти абсолютну пропускну спроможність, яку можна визначити як добуток q на інтенсивність потоку запитів:

$$A = q\lambda = \frac{\lambda\mu}{\lambda + \mu}, \quad (2.8)$$

Ймовірність того, що запит буде обслугований, визначається p_0 , а ймовірність отримання відмови – p_1 . Отже,

$$P_{\text{відмови}} = p_1 = \frac{\lambda}{\lambda + \mu}, \quad (2.9)$$

Даний засіб може бути використано для систем з середньою кількістю користувачів, при цьому можна легко визначити ймовірність відмови у доступі і відповідно до цього планувати доступ таким чином, щоб кількість відмов була мінімальною.

2.1.2. Неблокуючі методи

1. Метод кешування. Даний метод вносить у кеш всі зміни, внесені іншими користувачами під час того, як черговий користувач вносить свої зміни, і редагування відбувається як одночасно, так і поступово, тому що користувачі можуть внести зміни одночасно за часом їх створення але в документі вони будуть підтягнуті з кешу поступово, і остаточна версія документу буде готова тільки після завершення роботи користувачів з документом.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		27

2. Метод рангів (ієрархій).

Метод аналізу ієрархій (МАІ) був розроблений Томасом Л. Сааті в 1970 році і є систематичною процедурою для ієрархічного подання елементів, що визначають суть будь-якої проблеми.

Завдання, для вирішення яких може бути застосований МАІ:

- проблема багатокритеріального вибору. Вибір однієї альтернативи з наявного набору альтернатив на основі деяких критеріїв;
- ранжування. Багатокритеріальне упорядкування заданої множини альтернатив;
- визначення пріоритетів альтернатив та критеріїв в задачах багатокритеріального вибору;
- розподіл ресурсів. Розподіл ресурсів між альтернативами з заданої множини;
- порівняльний аналіз;
- розробка рекомендацій по оптимізації внутрішніх процесів організації на основі успішного досвіду конкурентів;
- управління якістю. Аналіз різних аспектів якості та шляхи покращення якості.

У відповідності з формулюванням завдання прийняття рішення структура моделі прийняття рішення в методі аналізу ієрархій представляє собою схему (граф), яка включає:

- набір альтернативних рішень;
- головний критерій рейтингування рішень;
- набір груп однотипних факторів, що впливають на рейтинг;
- множину зв'язків, що вказують на вплив рішень, критеріїв та факторів один на одного.

Структура моделі відображає результат аналізу ситуації ухвалення рішення. Стосовно документу та вирішення конфліктів в ньому даний метод полягає в тому, що кожен з користувачів чи груп користувачів має певний ранг

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		28

(ієрархію), і при виникненні конфлікту приймаються за остаточну версію зміни, внесені користувачем з вищим рангом, тоді ж зміни, які викликали конфлікт і внесені користувачами нижчих рангів, відхиляються.

3. Засіб використання розподілених транзакцій. Використання даного методу засноване на системі блокчейн, тобто наборі блоків у розподіленій системі. Дана система захищена від підробок, адже окремі її компоненти зберігаються на різних носіях, фізично віддалених один від одного на величезні відстані, і кожен наступний блок пов'язаний з деревом попередніх блоків. Тому підробка всієї системи неможлива, а підробка окремого блоку зруйнує зв'язок з попередніми блоками, тому не має сенсу. Саме захищеність даної системи від зовнішніх впливів і є однією з основних цінностей, тому все більша кількість організацій та держав переходять на зберігання важливих даних саме за технологією Блокчейн.

Людам, які недостатньо досвідчені у системі розподілених даних та новітніх комп'ютерних і обчислювальних технологіях, можуть вважати дану систему ненадійною та створеною для обману недосвідчених користувачів. Проте дана система на даний момент є найбільш захищеною, адже алгоритм шифрування захищає від запуску в систему видозміненого блоку, а перевірка достовірності через весь ланцюжок не дозволяє розповсюдити фальсифіковану інформацію по мережі.

Побудова гілки транзакцій відбувається таким чином, що транзакція не може бути продубльована, адже при проходженні по гілці вона має отримати підтвердження з декількох різних незалежних джерел, і при входженні у мережу вона отримує унікальний ідентифікатор, за допомогою якого всі дублюючі транзакції будуть відсічені. Дана система може бути застосована для вирішення конфліктів при дублюванні змін, і зберігає зміни окремо за транзакціями.

4. Differential synchronization (DS). Це симетричний алгоритм, який використовує нескінченний цикл фонових різницевих(diff) і патч-операцій. Не

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		29

існує жодної вимоги, які вражають серверні тристоронні злиття (3-way merge). Рисунок 2 представляє собою ідеалізовану схему потоку даних для DS. Він передбачає наявність двох документів які знаходяться на одному комп'ютері без мережі [8].

Текст клієнта і текст сервера можна редагувати у будь-який час. Мета полягає в тому, щоб завжди тримати ці два тексти якомога ближче один до одного.

1. Клієнтський текст дифундує на загальну тінь (Common shadow).
2. Це повертає список правок, які були виконані в клієнтському тексті.
3. Клієнтський текст копіюється в загальну тінь. Ця копія повинна бути ідентична значенням клієнтського тексту на кроці 1, тому в багатопотоковому середовищі повинен бути зроблений знімок тексту.
4. Ці зміни застосовуються до тексту сервера на основі максимальних зусиль.
5. Текст сервера оновлюється разом з результатом виправлення. Кроки 4 і 5 повинні бути атомарними, але вони не повинні бути блокуючими; вони можуть повторюватися до тих пір, поки текст сервера не залишиться незмінним досить довго.

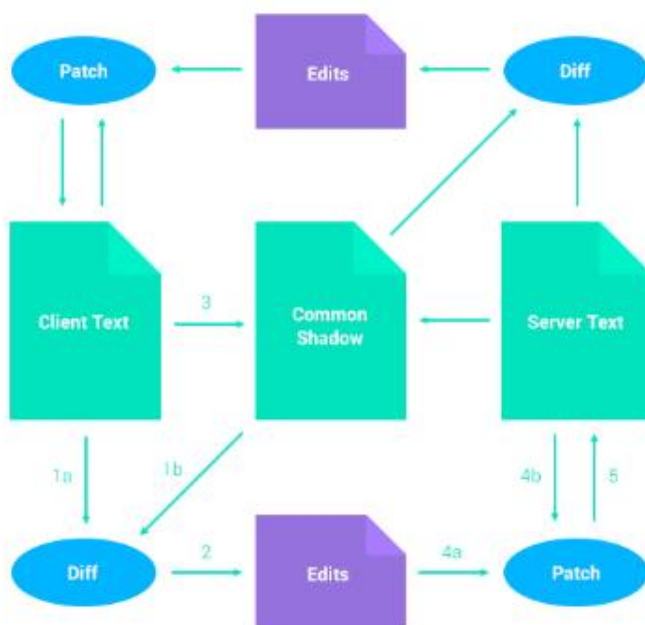


Рис 2.2. Ідеалізована схема потоку даних для DS

Тепер процес повторюється симетрично в протилежному напрямку. На цей раз загальна тінь збігається з текстом клієнта в попередній половині синхронізації, тому результуюча різниця поверне зміни, внесені в текст сервера, а не результат виправлення на кроці 5.

Роздільна функція полягає в тому, що алгоритм виправлення нечіткої, тобто патчі можуть бути застосовані навіть у тому випадку, якщо документ змінився. Таким чином, якщо клієнт набрав кілька натискань клавіш за час, необхідний для завершення синхронізації, виправлення(патчі) з сервера, швидше за все, будуть мати досить впізнаваний контекст, щоб їх можна було успішно застосувати. Однак, якщо деякі або всі виправлення не будуть виконані на кроці 4, вони автоматично відобразатимуться негативно при наступній зміні і будуть виправлені з тексту клієнта.

Забігаючи наперед, можна з впевненістю сказати, що даний алгоритм нам не підходить, оскільки тристороннє злиття для документа зі складною структурою є нетривіальним. При цьому необхідно мати формальну гарантію того, що він дасть однакові результати при злиттях в різних порядках і напрямках.

5. Операціональні перетворення(ОП). Техніка операціонального перетворення полягає в тому, що всі наступні операції перетворюються так, щоб враховувати ефект попередніх. При цьому не використовується блокування і в будь-який момент часу кожний користувач може вільного редагувати будь-який фрагмент тексту. Сьогодні ОП користується великим попитом в системах спільного редагування, для вирішення конфліктів та інших задач.

Техніка операціонального перетворення, незважаючи на складність реалізації, надає користувачам найбільшу зручність при роботі з текстом, і одночасно забезпечує досить якісне підтримання синхронності тексту у різних користувачів [9]. Саме цей метод використаний при розробці програми.

6. Безконфліктний реплікований тип даних (CRDT) - це структура даних, яка може бути реплікована в паралельному додатку без необхідності ручного вирішення конфліктів. Залежно від реалізації поновлення можуть бути відправлені як дельти (відповідність операціям в ОТ), так і шляхом спільного використання повного стану. Якщо ваші клієнти часто спілкуються один з одним, то набагато ефективніше ділитися тільки окремими операціями. Однак якщо зв'язок відбувається через дуже ненадійну мережу, наприклад мобільний або супутникову, відправка повного CRDT час від часу може бути кращим варіантом.

Основна ідея CRDT аналогічна ідеї ОП. Різниця полягає в тому, що клієнтам не потрібно перетворювати вхідні повідомлення, так як вони вже містять всю інформацію, необхідну для вирішення конфліктів з локальним станом. CRDT зберігають достатньо інформації в структурі даних, щоб створити операції, які можуть бути об'єднані без будь-якого перетворення [10].

Всякий раз, коли клієнт застосовує операцію, він може посилатися на символ за ідентифікатором, а не по позиції. Оскільки ідентифікатор унікальний, то операція буде як і раніше діє при отриманні іншим клієнтом, навіть якщо за цей час відбулися паралельні модифікації. Видалення відбувається аналогічним чином, але насправді ми не видаляємо символ зі структури даних, а просто помічаємо його віддаленим так званим надгробком.

Залишаються відкритими два питання: Як ми призначаємо нові ідентифікатори і як ми справляємося з двома паралельними вставками в одній і тій же позиції?

Нові ідентифікатори генеруються наступним чином: кожен клієнт відстежує найвищий числовий ідентифікатор, пов'язаний з його ідентифікатором клієнта. Щоразу, коли він вставляє новий символ, він збільшує найвищий числовий ідентифікатор і призначає новий ідентифікатор, що складається зі свого власного ідентифікатора клієнта і збільшеного числа.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		32

Паралельні вставки в одній і тій же позиції повинні бути дозволені таким чином, щоб привести до узгодженого стану.

Проте, CRDT - це не месія. Він підходить тільки для певного набору проблем і викликає додаткові накладні витрати на зберігання, оскільки їм необхідно відстежувати достатню кількість інформації історії в структурі даних, щоб забезпечити злиття вхідних операцій або станів.

2.2 Операціональні перетворення

Варто докладніше розглянути цю групу алгоритмів, адже саме її ми застосували при реалізації системи.

Перш ніж розповісти про ОП, варто зрозуміти, чому ця група алгоритмів була розроблена. Приблизно два десятиліття тому, коли спільні розподілені обчислення ставали все більш реальною реальністю, були визначені три фундаментальні якості для створення придатного і надійного спільного програмного забезпечення, в якому кілька співробітників редагують один документ, так звані властивості узгодженості[11] :

1. Збереження причинності (Causality preservation)
2. Збереження наміру (Intention preservation)
3. Конвергенція

З огляду на ці міркування, давайте розглянемо ситуацію, коли у нас є два редактора – Андрій та Влад, які намагаються відредагувати рядок, використовуючи наївний підхід. У цьому прикладі розглянемо ситуацію, коли Андрій намагається вставити символи в передню частину рядка, а Влад - видалити символи з передньої частини рядка. Оскільки ми живемо в реальному світі, існує кінцева затримка між двома правками, і на рис. 2.3 ми бачимо, як вони будуть застосовуватися до рядка прикладу. Як можна бачити, результат Влада закінчується з правильною версією рядка, в той час як результат Андрія виявився неправильним. Це пов'язано з тим, що коли Влад вказував своє оновлення delete, він використовував індекси початкового рядка

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		33

для створення специфікації. Коли Андрій оновив свій рядок, намір початкового видалення було втрачено.

Операціональні перетворення дозволяють розподіленим, спільним системам пом'якшити цю проблему шляхом перетворення специфікації оновлень співробітників з урахуванням історії застосування попередніх оновлень.

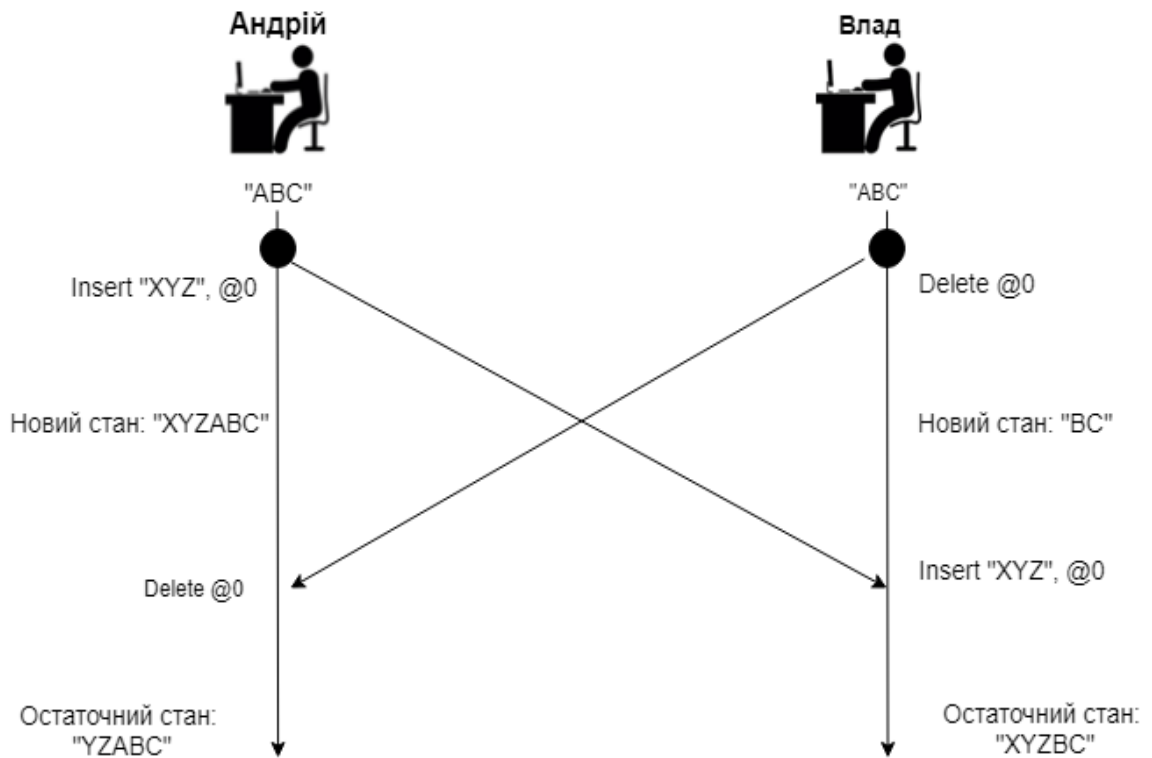


Рис 2.3. Невдалий сценарій, при якому обидва редактора намагаються змінити текст в одному і тому ж місці

Що ж дивного в проблемі Андрія? Ну, по-перше, ми бачимо, що перед командою видалення у Андрія дійсно були всі дані, які йому були потрібні, щоб опинитися в потрібному стані! При аналізі цієї проблеми реальна проблема полягала просто в незрозумінні того, що мав на увазі Влад. Отже, чи можна виправити це непорозуміння? Андрій знав, що він додав три символи, і якщо всі клієнти мали однаковий початковий стан(текст), Андрій міг перетворити команду Влада, щоб мати сенс в контексті його локального оновлення. Як Андрій це робить? Він просто розглядає свою історію перегляду і те, як документ був змінений з моменту первинного рядка. Звідти

Андрій може зрозуміти, що з початкового рядка все тепер зрушено на три, а потім перетворити команду Влада, щоб зберегти намір. По суті, цей процес відомий як операціональне перетворення, і цей приклад конкретно відомий як перетворення включення, при якому попередня операція робить деякий вплив на поточну операцію.

Як би школа не вчила нас твердити про неправильне, не менш важливо враховувати і те, що в кінцевому підсумку спрацювало правильно. У цьому прикладі ми бачимо, що текст Влада насправді виявився правильним. А чому це так? Як виявилось, оскільки видалення відбулося до вставки в тому ж самому місці, ефект видалення може бути виключений з обробки вставки. Хоча це перетворення також було перетворенням включення, ми відзначаємо, що ніякого чистого ефекту на перетворення не було. Це вказує на складну логіку, що лежить в основі цих перетворень, і показує, що тільки те, що було попередньо оновлене, не обов'язково означає, що операція повинна бути перетворена.

Щоб проілюструвати процес операціонального перетворення, розглянемо рисунок 2.4, на якому кожен співробітник аналізує і потенційно перетворює дане оновлення. Перш ніж застосувати оновлення до локального стану, клієнт з'ясовує, чи потрібно йому перетворити зовнішню операцію для збереження наміру. В кінцевому рахунку, завдяки цьому процесу обидва клієнта можуть застосовувати оновлення, які зберігають намір і сходяться до одного і того ж правильного стану.

					ДП 6125. 02.000 ПЗ	Арк.
						35
Зм.	Арк.	№ докум.	Підпис	Дата		

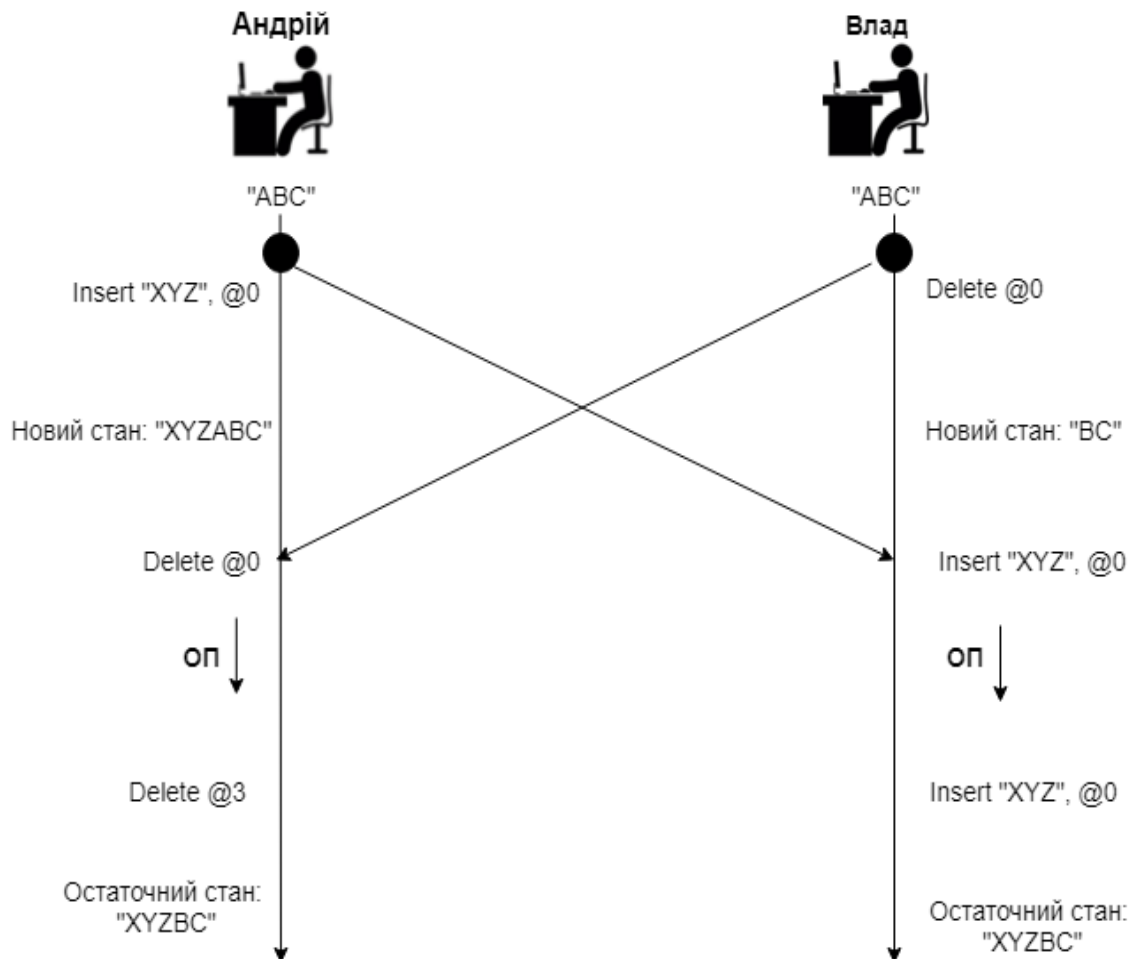


Рис 2.4. Вдалиий сценарій, при якому обидва редактора намагаються змінити текст в одному і тому ж місці застосовуючи ОП

2.2.1 Моделі узгодженості

Моделі узгодженості визначають правила, які визначають, які дані кожен клієнт може бачити в пам'яті, коли певний набір операцій було виконано різними клієнтами. Прикладом виникнення неузгодженості може служити ситуація, коли один клієнт записує свою власну локальну копію даних в розподіленому сховищі даних, але ще не поширив ці зміни на інших клієнтів. Якщо інші клієнти хочуть читати або записувати ці дані, то вони можуть використовувати стару версію даних, яка не враховує зміни іншого клієнта [12].

Для систем спільного редагування було визначено кілька моделей узгодженості для формальної перевірки властивостей, які клієнти повинні

підтримувати для успішного розв'язання конфліктів в локальних копіях своїх документів.

2.2.1.1 СС модель

Для систем спільного редагування потрібні дві властивості узгодженості:

1) Каузальність (Causality) або властивість старшинства по передування: гарантує, що порядок виконання причинно-залежних операцій буде таким же, як їх природний причинно-наслідковий порядок, відповідно до процесу спільної роботи. Каузальний зв'язок між двома операціями формально визначається співвідношенням Лампорта "відбулося до" (happened-before). Якщо дві операції каузально незалежні, вони є паралельними. Дві паралельні операції можуть виконуватися в різному порядку на двох різних копіях документів.

2) Збіжність (Convergence): гарантує, що всі репліковані копії загального документа повинні бути ідентичними на всіх сайтах (тобто, всі операції, що генеруються будуть виконані на всіх сайтах), після виконання всіх операцій.

Так як паралельні операції можуть бути виконані в різному порядку і операції редагування зазвичай не комутативні, копії документа на різних сайтах можуть відрізнятися (бути несумісними).

2.2.1.2 ССІ модель

ССІ модель була запропонована в якості загального фреймворка для управління узгодженістю в системах спільного редагування. У середині ССІ моделі згруповані разом три властивості узгодженості:

1) Збереження причинності: те ж саме, що і властивість старшинства по передування в СС моделі.

2) Збіжність: те ж саме, що і властивість збіжності в СС моделі.

3) Збереження наміру: гарантує, що ефект виконання операції у будь-якому стані документа, буде таким же, як і прогнозувалося при виконанні операції. Намір операції Op визначається як ефект виконання, який може

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		37

бути досягнутий шляхом застосування Op на стані документа, з якого Op була згенерована.

CCI модель розширює CC модель новим критерієм: збереження наміру. Істотна різниця між конвергенцією і збереженням наміру полягає в тому, що перше завжди може бути досягнуто шляхом серіалізації протоколу, однак останнє не може бути досягнуто за рахунок серіалізації будь-якого протоколу, якщо операції завжди будуть виконуватися в їх початковому варіанті. Забезпечення якості збереження несеріалізованого наміру була головною технічною задачею. Виявилось, що ОП дуже добре підходить для забезпечення конвергенції та збереження наміру в системах спільного редагування. Саме ця модель використана при реалізації програмного забезпечення.

2.2.1.3 CSM модель

Умова збереження наміру не була формально визначена в CCI моделі для цілей формальних доказів. Модель узгодженості CSM складається з наступних формальних умов:

- 1) Каузальність: визначає те ж саме, що і в CC моделі.
- 2) Наслідки виконання однієї операції (Single-operation effects): наслідок від виконання будь-якої операції у будь-якому стані виконання досягає того ж ефекту, що і в його початковому стані.
- 3) Наслідки виконання декількох операцій (Multi-operation effects): залежність наслідків будь-яких двох операцій підтримуються і після того, як вони обидві будуть виконані у будь-яких станах.

2.2.1.4 CA модель

Наведена вище CSM модель вимагає, щоб в системі був визначений загальний порядок всіх об'єктів. Отже, специфікація зводиться до нових об'єктів, що подаються операціями вставки. Однак, специфікація загального порядку тягне за собою застосування залежних від додатка стратегій, таких як при розриві зв'язків для вставки (тобто, коли нові об'єкти вставляються двома

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		38

поточними операціями в одну і ту ж позицію). Таким чином, загальний порядок стає залежним від додатка. Більш того, у функціях перетворення і процедурі управління алгоритму загальний порядок повинен зберігатися, що призводить до збільшення складності часу / простору алгоритму.

2.2.2. Моделі даних в ОП

Моделі даних в ОП - це представлення даних в конкретній спільній сесії [13]. Модель даних може являти собою документ, електронну таблицю, зображення або будь-який інший об'єкт, з яким різні клієнти можуть спільно працювати. Сама базова модель даних ОП - це модель лінійного простору в пам'яті, наприклад, одного рядка.

Останні розробки в області ОП дозволили розробити більш складні моделі даних ОП, такі як моделі, що включають редагування формату rich-text з розміткою або розширений стан додатку, представлений в JSON (JavaScript Object Notation). Однак в міру ускладнення моделей даних, що використовуються в ОП, ускладнюються і моделі, необхідні для подання конкретної операції з даними.

2.2.3. Операційна модель

Операція є фундаментальним блоком механізму ОП, і в найпростішій моделі даних ОП операція - це просто вставка або видалення одного символу в пам'яті [14]. Крім того, більш складні операції можуть бути створені шляхом комбінування вставок і вилучень, таких як операція підстановки (комбінація операції вставки і видалення).

Для просунутих моделей даних, які можуть використовувати більше, ніж лінійний простір в пам'яті, ми часто знаходимо складніші операції, такі як переміщення даних між адресами пам'яті. Наприклад, текстові процесори ОП можуть додавати операції для початку і закриття кордонів анотацій для представлення тегів розмітки, а структури ОП JSON можуть виконувати операції редагування для певних шляхів пошуку об'єкта JSON.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		39

У минулих дослідженнях була зроблена спроба узагальнити дані ОТ і операційної моделі, а також дозволити тільки операції вставки, видалення і заміни даних. Однак для більшості сучасних реалізацій ОТ використовуються прикладні дані і операційні моделі. Вони також вимагають функцій перетворення операцій для підтримки властивості збереження наміру, по одному для кожної пари операцій, оскільки кожен тип операції повинен бути здатний бути зміненим алгоритмом управління ОТ щодо іншого типу операції. Таким чином, для N специфічних для додатка операцій потрібно $N \times N$ функцій перетворення для ОТ.

2.2.4 Переваги та недоліки ОП

Порівнявши всі існуючі методи як не автоматичного так і автоматичного вирішення конфліктів в тексті, я зробив вибір на користь ОТ, тому що:

- Дана група алгоритмів вирішує майже всі існуючі конфлікти при спільному редагуванні текстів
- Сервер повинен мати тільки один простір станів, яке є історією операцій, які він застосував. Коли він отримує операцію клієнта, йому потрібно тільки перетворити операцію відповідно до історії операцій, застосувати перетворену операцію і потім транслювати її.
- Можливості системи ОТ можуть бути розширені шляхом зміни одного компонента без впливу на інший. Наприклад, алгоритми управління можуть бути розширені для підтримки нових можливостей (наприклад, розширення від підтримки узгодженості до скасування групи) без зміни існуючих функцій перетворення; функції перетворення можуть бути розширені для підтримки нових функцій (наприклад, вирішення конфліктів, блокування) або нових додатків з різними моделями даних і операцій без зміни алгоритмів управління (наприклад, один і той же алгоритм управління може використовуватися для підтримки паралельного редагування, вирішення конфліктів, блокування і вказівки).

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		40

Як і всі існуючі алгоритми автоматичного вирішення конфліктів текстів, ОП також має свої недоліки:

- Дана група алгоритмів дуже складна в реалізації
- Необхідно зберігати кожен змін документ

2.2.5 Приклади використання ОП в різних додатках спільного редагування.

ОП був застосований до широкого спектру програм для спільної роботи в режимі реального часу і в режимі не реального часу. Розглянемо найпопулярніші і великі з них:

1) Спільне редагування тексту. Приклади систем:

- Subethaedit: спільний редактор в реальному часі, призначений для Mac OS X.
- CoVim: текстовий редактор для спільного користування, який дозволяє декільком користувачам одночасно редагувати один і той же текстовий документ в Vim в реальному часі.

2) Спільне редагування графіки. Наявна система:

- CoFlash: спільний мультимедійний редактор в реальному часі, який уможливорює одночасні зміни в документі декількома клієнтами в Adobe Flash

3) Спільне редагування HTML / XML і форматування документів.

Приклади систем:

- EtherPad: веб-редактор для спільної роботи в режимі реального часу.
- CoSKEditor: веб-редактор для спільної роботи в режимі реального часу, що дозволяє одночасне редагування одного і того ж документу в SKEditor декільком користувачам.

4) Спільна обробка текстів. Наявна система:

					ДП 6125. 02.000 ПЗ	Арк.
						41
Зм.	Арк.	№ докум.	Підпис	Дата		

- CodoxWord: артільний текстовий процесор реального часу. Дозволяє рівночасно редагувати один і той же документ декільком користувачам в Microsoft Word.

5) Інструменти спільного створення слайдів та презентацій. Існуюча система:

- CoPowerPoint: спільний редактор слайдів в режимі реального часу, який уможливорює одночасне редагування одного і того ж документу в Microsoft PowerPoint декільком користувачам.

6) Спільне редагування електронних таблиць. Приклади систем:

- CoCalc: спільний редактор електронних таблиць в режимі реального часу, який дозволяє декільком користувачам одночасно редагувати один і той же документ в OpenOffice.org

7) Спільні автоматизовані засоби проектування цифрових медіа.

Приклади систем:

- CoMaya: інструмент для спільного 3D-проектування в реальному часі, який дозволяє декільком дизайнерам одночасно редагувати один і той же документ в Autodesk Maya.

8) Спільні веб-додатки, сервіси та фреймворки. Приклади систем:

- Google Docs: веб-сумісний текстовий процесор в режимі реального часу.
- Google Wave: де, ОП - основний метод спільної роботи в реальному часі, що лежить в основі даного проекту, який об'єднує електронну пошту, миттєві повідомлення, блоги, Вікі і соціальні мережі.
- Novell Pulse: корпоративна програмна платформа для спільної роботи в реальному часі, що використовує протокол Федерації Google Wave
- Open Cooperative Web Framework: проект Dojo Foundation, який використовує алгоритми операціонального перетворення для реалізації концепцій кооперативних веб-сайтів.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		42

Висновок до розділу 2

Готові рішення вирішення конфліктів доступу в тексті базуються на ряді різних моделей. Одні моделі базуються на блокуванні доступу до ресурсу під час редагування іншими користувачами, другі визначають черговість та ієрархію внесення змін, треті користуються алгоритмом узгодження змін від користувачів, якщо воно можливе. Отже, існує ряд механізмів та методів вирішення конфліктів при спільному редагуванні текстів, окремі з них вже застосовуються в даних додатках, проте вони є досить недосконалими, адже вони автоматично не вирішують всіх конфліктів або вимагають ручного затвердження змін, або ж відхиляють конфліктуючі зміни без можливості прийняття компромісного рішення. Тому доцільно розглянути можливість розробки програмного продукту для автоматичного вирішення вищезазначених конфліктів

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		43

РОЗДІЛ 3.

РОЗРОБКА СИСТЕМИ

3.1 Вибір підходу для реалізації

Основна ідея більшості додатків спільного редагування текстів та документів полягає у використанні зіставлення вмісту різних версій документа. Для наочності будемо вважати, що у нас є два користувача - Андрій і Влад. Вміст документа однаковий як в першого користувача так і в другого, тобто синхронізовано між ними. В момент отримання оновлених даних від Андрія, сервер обчислює різницю між версією Андрія та своєю, і силкується поєднати дві версії документів в одну найкращим існуючим шляхом. Відтак сервер відсилає отриману версію Владу. У разі наявності будь-яких невідправлених змін від Влада на сервер, цикл повторюється, оскільки потрібно поєднати серверний варіант з локальним варіантом Влада, і потім знову відправити оновлені дані на сервер. Здебільшого даний підхід функціонує не вельми коректно, оскільки виникає велика кількість помилок.

При реалізації нашої програми використаний інакший підхід, а саме: документи зберігаються як послідовність змін. Просте порівняння тексту документа не застосовується, а натомість вносяться зміни по порядку. Знаючи, як і де саме клієнт видозмінював вміст документа і з огляду на бажання та затію користувача можна чітко встановити остаточний варіант після поєднання всіх змін та редагувань.

Щоб зрозуміти коли нам слід впроваджувати зміни потрібен протокол взаємодії.

Всі операції над змістом документа можна розділити на три різні операції, а саме:

- видалення тексту;
- вставка тексту;
- застосування стилів до текста.

					ДП 6125. 02.000 ПЗ	Арк.
						44
Зм.	Арк.	№ докум.	Підпис	Дата		

Таким чином, коли користувач редагує документ, всі його дії зберігаються рівно в цьому наборі операцій, і вони дописуються в кінець журналу змін. При відображенні документа журнал змін буде виконаний, застосовуючи записані операції.

3.2 Модель узгодженості

Проаналізувавши і порівнявши всі існуючі моделі узгодженості, найкращою і найдоцільнішою моделлю для нашої системи виявилася модель ССІ, яка забезпечує виконання трьох основних властивостей, а саме:

- 1) Збіжності
- 2) Збереження наміру
- 3) Збереження причинності

3.3 Структура системи ОП

Стратегією структурування системи ОП є відокремлення алгоритмів контролю перетворення (transformation control algorithms) високого рівня (або інтеграції) від функції перетворення (transformation functions) низького рівня, і вказівками відносин (обов'язки і обмеження) між цим двома шарами в якості властивостей і умов (properties and conditions) перетворення в середині системи ОП, як показано на рис. 3.1 [15]. Інакше кажучи, стратегією роботи ОП системи є її розділ на три складові:

1) Алгоритми контролю перетворення (АКП): відповідають за визначення того, коли операція (мета перетворення) готова до трансформації, які операції повинні бути перетворені і в якому порядку повинні виконуватися перетворення. Ці алгоритми є універсальними в тому сенсі в тому сенсі, що вони спираються на загальні відносини (наприклад, контекст і паралелізм) між операціями, які виконують цю роботу. Алгоритми управління забезпечують вхідні операції для функцій перетворення.

2) Функції перетворення: відповідають за виконання реальних трансформацій на цільовій операції відповідно з урахуванням впливу еталонних процедур. Функції перетворення залежать від типів і параметрів

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		45

операцій. Функції перетворення виробляють вихідні операції для алгоритмів управління.

3) Вимоги і властивості перетворень: визначають відносини між АКП і функціями і служать вимогами коректності алгоритму ОТ, які можуть бути використані для перевірки правильності алгоритму контролю перетворення або функції по відношенню до певних вимог коректності. Для правильного функціонування системи ОТ необхідно, щоб АКП і функції перетворення відповідали один одному з точки зору цих умов і властивостей.

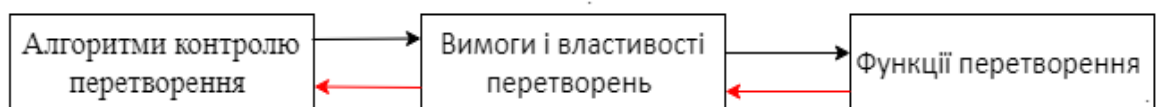


Рисунок 3.1. Багаторівнева структура системи ОТ

Відділення АКП від специфічних для операцій функцій перетворення має наступні переваги:

1) Зниження складності проектування системи ОТ: розбиття системи ОТ на більш дрібні і керовані компоненти з чітко визначеними умовами перетворення і властивостями між ними допомагає знизити складність проектування і дозволяє змінювати один компонент без впливу на інший.

2) Підвищення модульності і повторюваності компонентів ОТ: алгоритми керування і функції перетворення можуть бути розроблені і перевірені окремо. Компоненти, призначені для різних систем ОТ, можуть бути об'єднані для створення нових систем ОТ для обслуговування нових цілей, якщо вони сумісні щодо умов перетворення і властивостей, необхідних один одним.

3) Полегшення розширення можливостей: можливості системи ОТ можуть бути розширені шляхом зміни одного компонента без впливу на інший. Наприклад, АКП можуть бути розширені для підтримки нових можливостей (наприклад, розширення від підтримки узгодженості до скасування групи) без зміни існуючих функцій перетворення; функції

перетворення можуть бути розширені для підтримки нових функцій (наприклад, вирішення конфліктів, блокування) або нових додатків з різними моделями даних і операцій без зміни алгоритмів керування (наприклад, один і той же АКП може використовуватися для підтримки паралельного редагування, вирішення конфліктів, блокування і вказівки).

3.4 Функції перетворення

Існує два види функцій перетворення (для скасування групування і підтримки узгодженості):

1) Включення або пряма трансформація (Inclusion or Forward Transformation): Позначається як IT(Opa, Opb). Перетворює операцію Opa перед операцією Opb таким чином, щоб вплив Opb ефективно включається в параметри Opa, тобто враховується ефект ОП (наприклад, дві вставки).

2) Винятки або зворотня трансформація (Exclusion or Backward Transformation):

Позначається як ET (Opa, Opb), яке перетворює операцію Oa перед іншою операцією Opb так, що вплив Opb ефективно виключається з параметрів Opa(наприклад, вставка після видалення) [16].

3.5 Протокол взаємодії

Пояснимо роботу нашого алгоритму на прикладі з вищезгаданими користувачами – Андрієм і Владом.

Кожен з них занотовує для себе таку інформацію:

- 1) Версія (id, ревізія) останньої зміни, відправлену з сервера клієнту.
- 2) Поточний стан документа, який бачить конкретний редактор.
- 3) Будь-які зміни, зроблені локально, відправлені на сервер, але без підтвердження від сервера.
- 4) Будь-які зміни, зроблені локально, але не відправлені на сервер (очікують свого надсилання)

Сервер запам'ятовує наступну інформацію:

- 1) Поточний стан документа на момент останньої обробленої зміни

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		47

2) Список всіх змін, які він отримав, але ще не обробив (черга обробки).

3) Повна історія всіх оброблених змін – лоз ревізій.

Всі зміни обробляються покроково з врахуванням попередніх редакцій документу. Кожен крок перевіряється на узгодження і програма в автоматичному режимі обирає зміну, яка має слідувати наступною (за часом надходження на сервер) і вбудовує її в документ з врахуванням його поточного стану.

Спочатку документ порожній. Нехай Андрій набирає фразу "Hello".

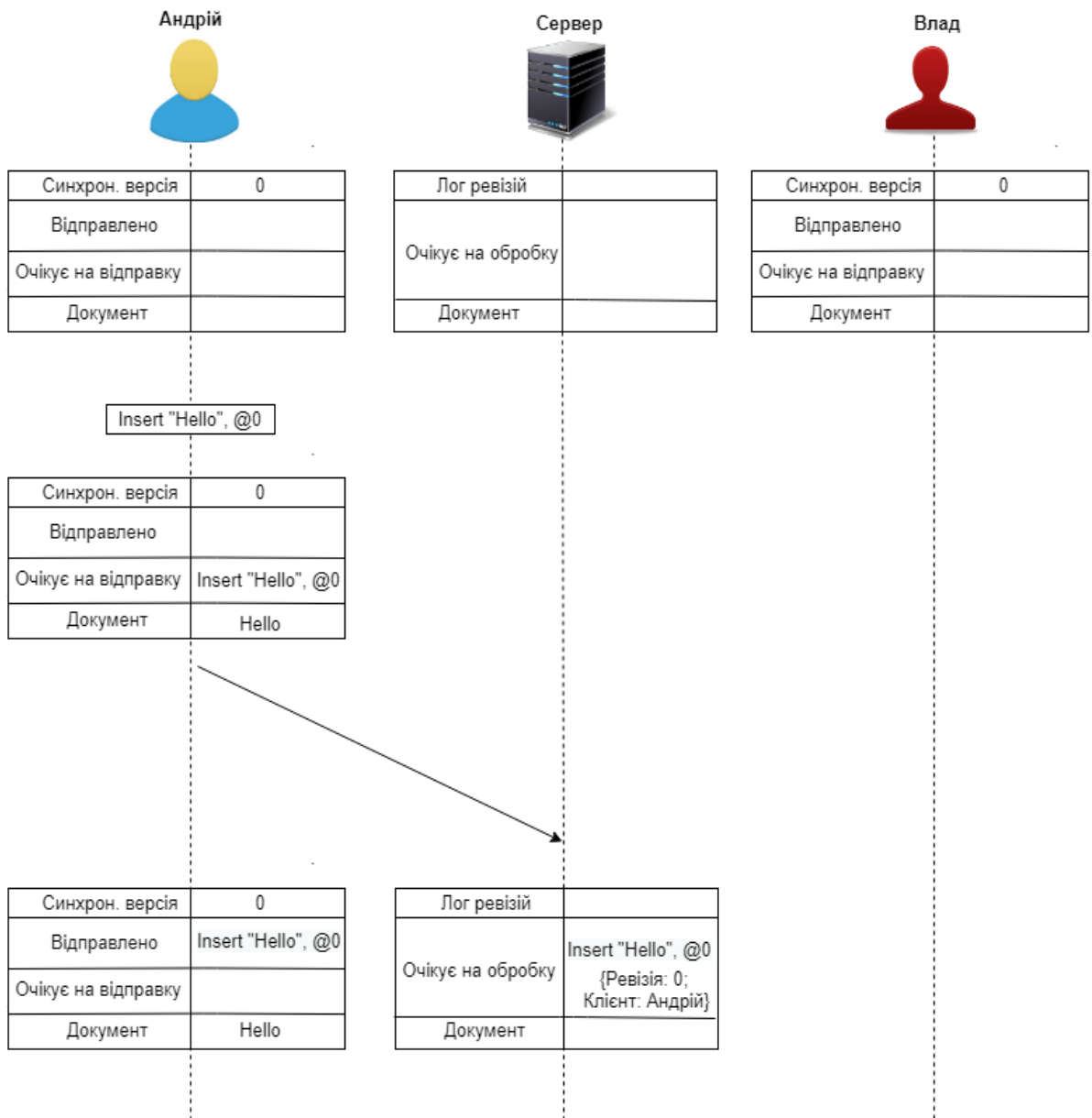


Рис 3.2. Ілюстрація роботи алгоритму на першому кроці

Зм.	Арк.	№ докум.	Підпис	Дата

Замовник на початку додав цю зміну до переліку змін, які чекають відправки, а слідом за тим вислав на сервер і перемістив конфігурації в перелік надісланих (Відправлено).

Андрій не припиняє писати і вже дописав текст "Київ". Паралельно цьому Влад надрукував знак "!" у своєму пустому документі, оскільки він ще не отримав змін в документі, які зробив Влад.

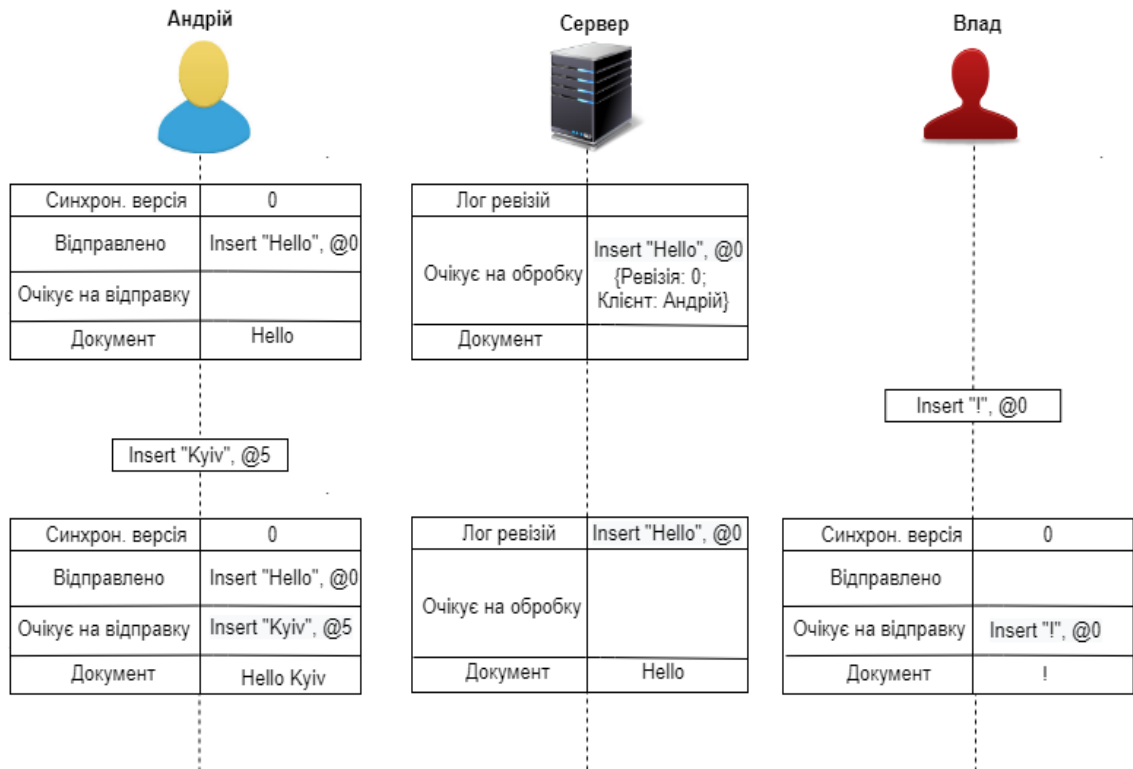


Рис 3.3. Ілюстрація роботи алгоритму на другому кроці

Андрійове «Insert "Київ", @5» було внесено до переліку, які чекають відправки, але не було ще надіслано, внаслідок того, що не можна відправляти більше ніж одну конфігурацію за один раз, а попереднє ще не було підтверджено. Варто зауважити, власне що сервер також зберіг конфігурації Андрія в своєму лозі ревізій. Далі сервер посилає їх Владу і відправляє підтвердження Андрію (про те, власне що Андрієві перші зміни були благополучно оброблені).

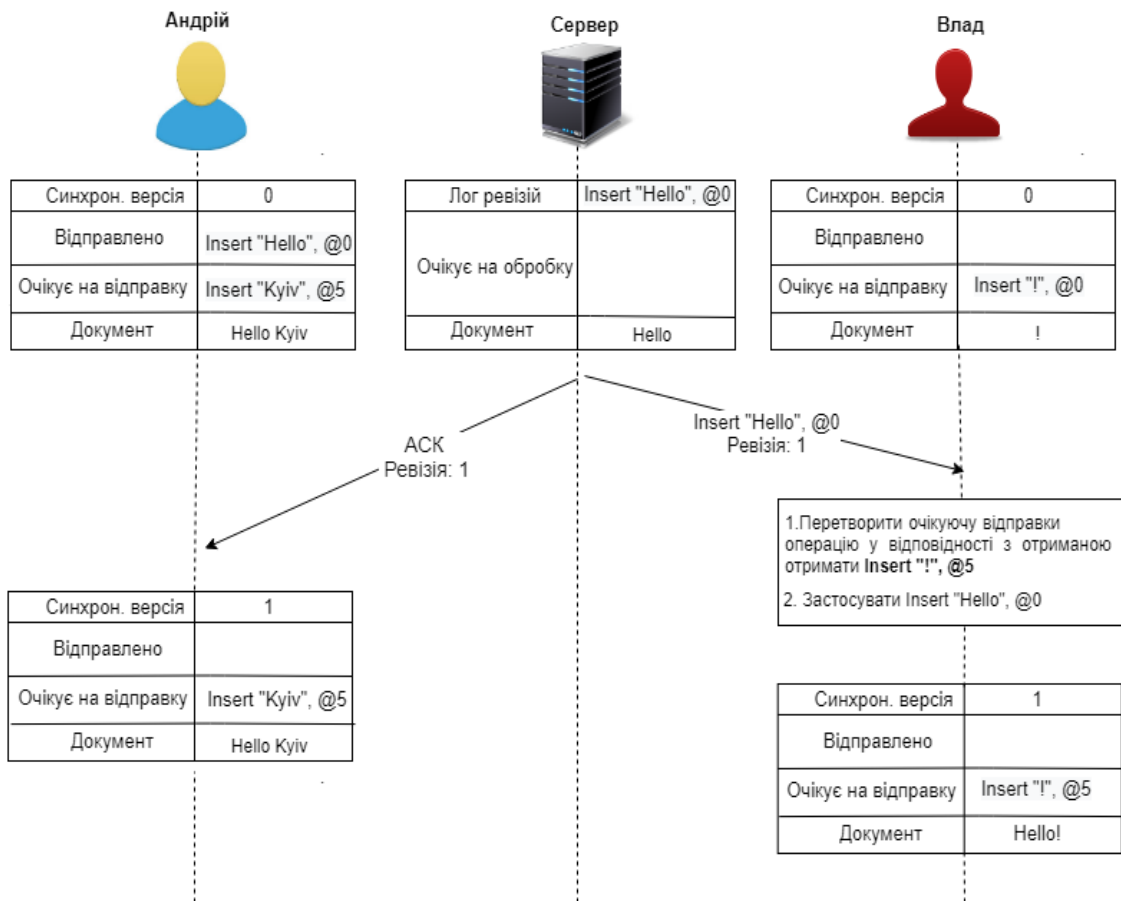


Рис 3.4. Ілюстрація роботи алгоритму на третьому кроці

Замовник Влад отримує Андрієві конфігурації від сервера і використовує ОП відносно змін, що чекають відправки на сервер «Insert "!", @0». Підсумком стає зміна індексу вставки з 0 на 5. Також зауважимо, власне що обидва клієнти оновили номер останньої синхронізованої ревізії з сервером на 1. І насамкінець, замовник Андрія видаляє відповідну зміну з переліку очікуючих підтвердження від сервера.

Далі Андрій і Влад надсилають власні зміни-конфігурації на сервер.

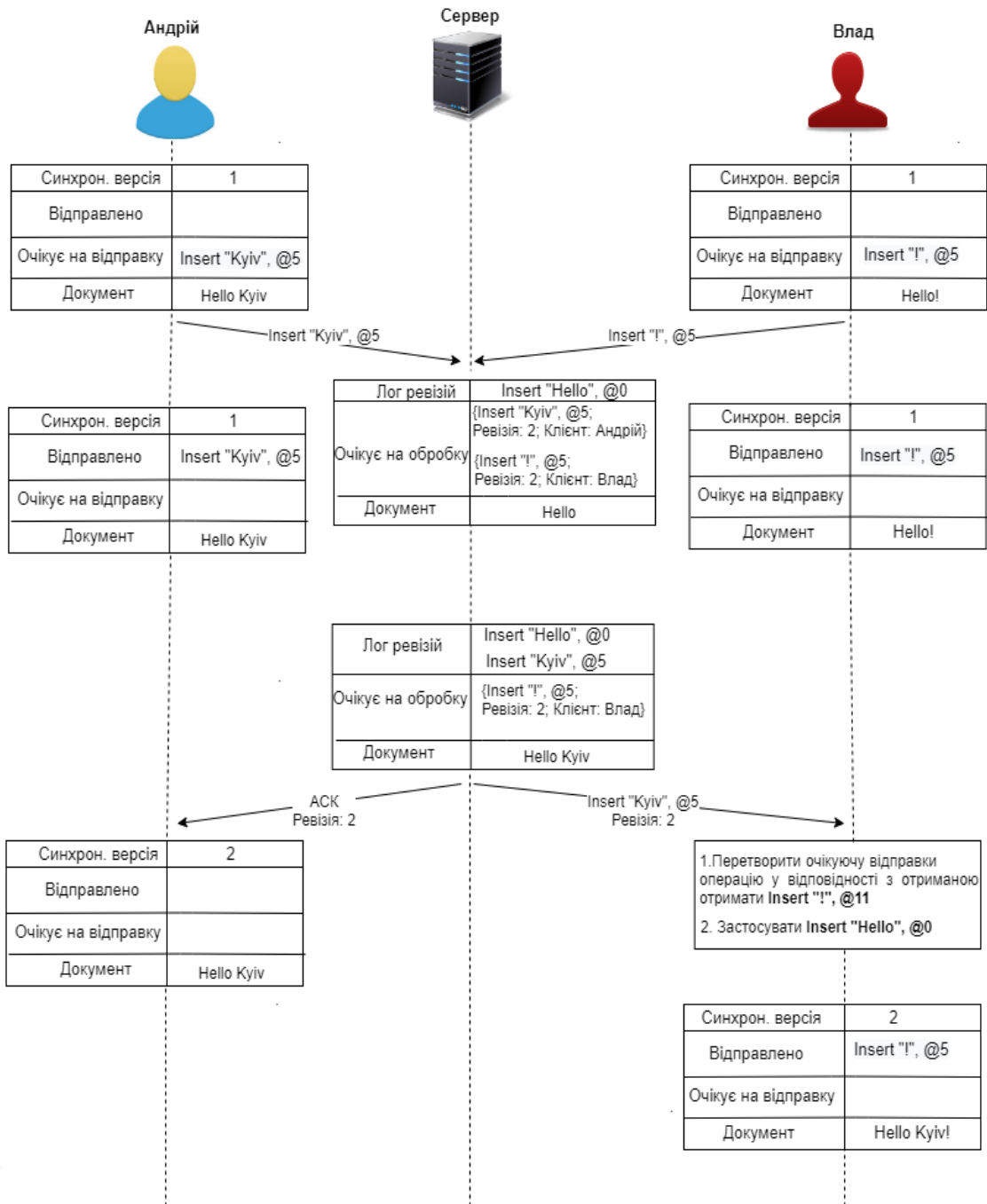


Рис 3.5. Ілюстрація роботи алгоритму на четвертому кроці

Сервер отримує Андрієві зміни до Владових змін, в наслідок цього в першу чергу він опрацьовує їх, а потім відправляє Андрію підтвердження. Ще він відправляє їх Владу, і його замовник видозмінює їх відносно поки ще непідтверджених змін «Insert "!", @5».

Згодом відбувається доленосна подія. Сервер починає опрацьовувати конфігурації Влада, ті, які на думку Влада, мають номер ревізії 2. Але сервер у себе вже зафіксував конфігурації під номером 2, в наслідок цього він

використовує перетворення до всіх змін, про які замовник Влада ще не знає (в даному випадку – «Insert "Kyiv", @5»), і зберігає результат під номером 3.

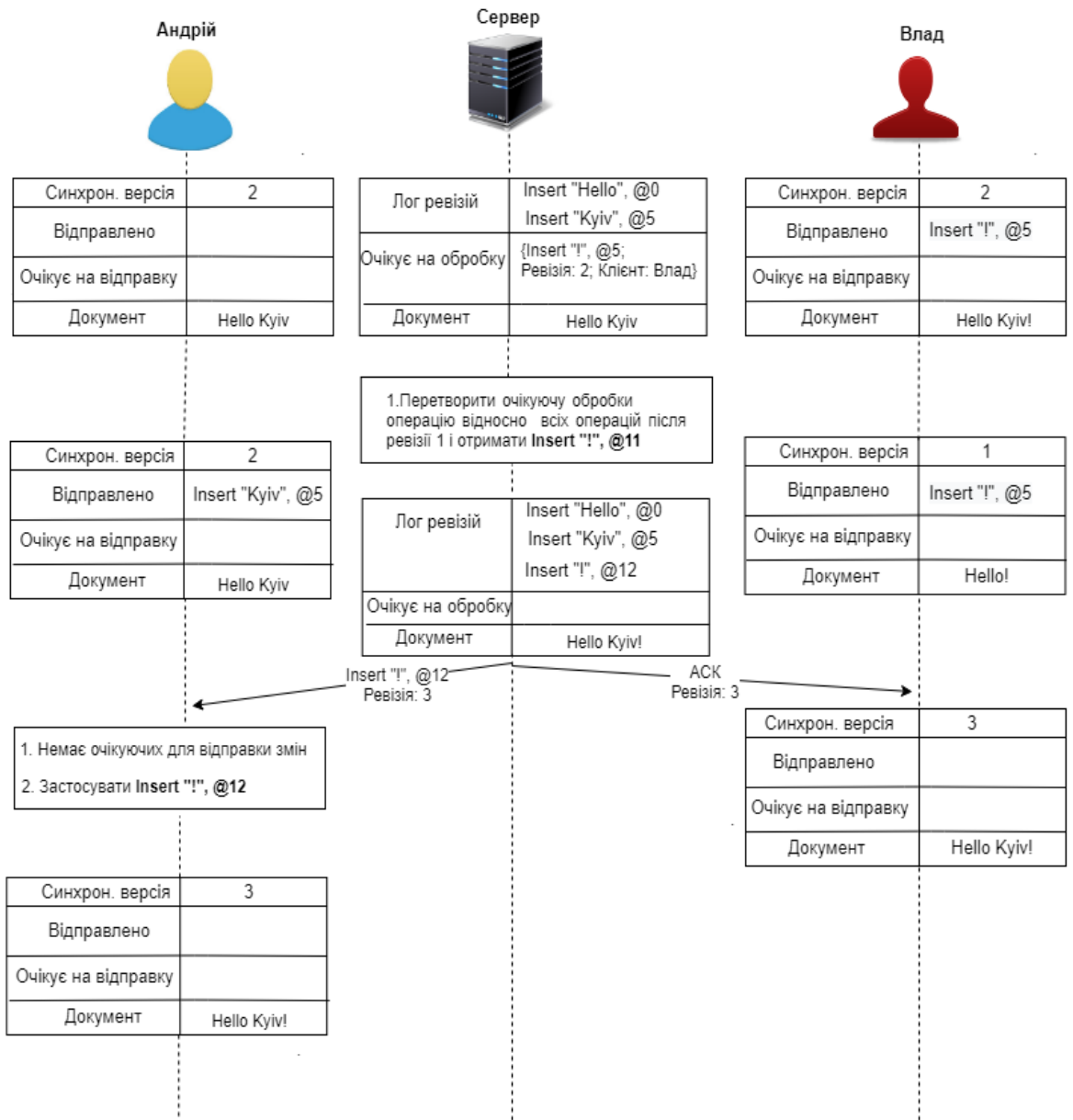


Рис 3.6. Ілюстрація роботи алгоритму на п'ятому кроці

Наочно показано, що індекс в Владовій зміні був збільшений на 5, щоб помістити слово Андрія.

Для Андрія процедура завершено, а Владу всього на всього залишилося отримати нову конфігурацію від сервера і відправити доказ цього.

3.6 Модель взаємодії між користувачами

1-і алгоритми на базі ОП не передбачали застосування центрального сервера. Всі користувачі зв'язувалися між собою peer-to-peer. Відповідно до цього, поточний стан документа у користувачів виражався послідовністю операцій (Op1, Op2, ... OpN), при цьому у будь-який момент часу кількість, склад і порядок даних операцій у будь-якого користувача може бути свій. В даному випадку неможливо визначити єдиний жорсткий порядок між всіх операцій, що генеруються, можливо встановити лише тільки слабкий порядок по happened before аспекту. Операції, між якими немає такого ставлення, числяться конкуруючими.

Даний підхід має багато недоліків, серед яких складність реалізації, зменшення продуктивності роботи та збільшення вимог до перетворення.

Проаналізувавши цю інформацію було вибрано клієнт-серверний варіант у якому всі користувачі рівні між собою і відсутнє p2p з'єднання [17]. Використовуючи модель клієнт-сервер стан документів можна описати простою ревізією.

Система реалізована по протоколу HTTP з архітектурою запиту-відповіді. На жаль, дана архітектура обмежує способи, якими сервер може взаємодіяти з клієнтом. Сервер не може ініціювати з'єднання з клієнтом і повинен чекати, поки користувач не зв'яжеться з ним і не запросить інформацію для відправки даних. Це є певною проблемою при реалізації алгоритму операціонального перетворення, оскільки сервер повинен відправляти поновлення клієнтам якомога швидше, як тільки інший користувач вніс зміни. Один із способів обійти це полягає в тому, щоб опитати клієнта через певний проміжок часу. Наприклад, замість відправки повідомлення про оновлення кожен раз, коли який-небудь клієнт щось змінює, сервер повинен запам'ятати результати всіх перетворень, проведених ним в відношенні повідомлень від клієнта, а потім відправити їх клієнту, коли клієнт опитує. Проблема з опитуванням полягає в тому, що існує велика кількість накладних витрат, які виникають при

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		53

створенні нового http-з'єднання кожні 100-200 мілісекунд. У багатьох випадках опитування непродуктивні, оскільки сервер не має ніякої інформації для клієнта, і відповідь від сервера полягає просто в тому, що немає ніяких нових даних для клієнта.

3.7 Модель збереження даних

Всі зміни при редагуванні зберігаються в пам'яті, для того, щоб збільшити швидкодію програми, позбавитися блокувань програми через звернення до файлової системи. Але за бажанням, можна вибрати будь-який альтернативний спосіб. Так як дана система дозволяє зберігати дані у вигляді час-значення-змін, це дозволяє використати будь-яку базу даних, що дозволить нам зберігати реплікацію на фізичному носії.

3.8 Підхід до обробки взаємодії користувачів

При реалізації системи був використаний асинхронний метод, щоб мінімізувати навантаження на сервер [18]. Оскільки вирішення конфліктів в реальному часу не потребується, цим самим ми досягаємо більшої ефективності роботи нашої програми, так як ми використовуємо значну кількість даних, тому економимо ресурс мережі при передачі.

3.9 Стек застосованих технологій

3.9.1 Мова програмування

Ми використали мову Java 8.0, тому що вона кроссплатформенна, тобто дозволяє запускати програмне забезпечення в незалежності від операційної системи і від апаратного забезпечення, також вона має швидкодію на рівні програм написаних під конкретну платформу, тому цей вибір найбільш доцільним для даної роботи.

Мова JS та ReactJS були використані для клієнтської частини, для відображення результату та синхронізації з сервером.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		54

3.9.2 Системи збірки

Maven був використаний для збірки проєкту на Java, дозволяє завантажити всі залежності для даного проєкту, а також дозволяє розробляти нашу програму по версіям.

NPM був використаний для завантаження залежностей для клієнтської частини, автоматизує збірку фінального проєкту.

3.9.3 Протокол передачі даних

Замість старого протоколу HTTP/1.1 був використаний протокол HTTP/2, тому що він має такі переваги [19]:

1. Одне з'єднання. Для завантаження веб-сайту використовується тільки одне з'єднання з сервером, і це з'єднання залишається відкритим до тих пір, поки веб-сайт відкритий. Це скорочує кількість переходів туди і назад, необхідних для налаштування кількох TCP-з'єднань.
2. Ущільнення. Кілька запитів дозволені одночасно, на одному і тому ж поєднанні. Раніше, з HTTP / 1.1, кожна передача повинна була б чекати завершення інших передач.
3. Пріоритизація. Запити - це призначені рівні залежності, які сервер може використовувати для більш швидкої доставки ресурсів з більш високим пріоритетом.
4. Бінарність. Робить HTTP / 2 простішим для аналізу сервером, більш компактним і менш схильним до помилок. Не витрачається додатковий час на переклад інформації з тексту в двійковий формат, який є рідною мовою комп'ютера.
5. Стиснення заголовків. HTTP / 2 використовує стиснення HPACK, що знижує накладні витрати. Багато заголовки були відправлені з однаковими значеннями в кожному запиті в HTTP / 1.1.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		55

Висновок до розділу 3

Таким чином, використовуючи концепцію операціонального перетворення та модель узгодження ССІ, ми розробили алгоритм вирішення конфліктів при спільному доступі до тексту в автоматичному режимі. За допомогою узгодження відбувається злиття версій редагування до спільного результату – враховуються одночасно внесені зміни в різні елементи тексту або навіть у одне й те ж слово, якщо конфлікт є вирішуваним, тобто зміни не є взаємовиключними. Всі користувачі можуть впевнитись у тому, що їх зміни були враховані та записані, а також переглянути кінцеву версію документу і продовжити своє редагування за потреби. Отже, даний алгоритм є дієвим і може бути реалізований програмно для автоматичного застосування.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		56

РОЗДІЛ 4.

ТЕСТУВАННЯ СИСТЕМИ

4.1 Ілюстрація працездатності розробленої програми

Інтерфейс користувача складається з двох вікон:

1. Поле для введення тексту
2. Поле для перегляду графу змін.

Найкращим способом протестувати і проілюструвати роботу нашої програми є наступний приклад: два користувачі хочуть одночасно працювати з одним документом в мережі.

Нехай перший користувач ввів рядок «Привіт Одеса, Київ, Миколаїв, Харків, Боярка». Відразу після цього праворуч від користувача з'явився оновлений граф змін, який зображений на рис 4.2, на якому зображена операція Ins (Insert - вставити), індекс згідно до якого була виконана операція в тексті та сам текст який слід вставити або видалити в існуючий рядок. Вершина зеленого кольору означає, що стан цієї вершини є останнім оновленням рядка. Варто зауважити, що граф змін буде всім, хто підключений до мережі. Через певний проміжок часу двоє користувачів почали одночасно редагувати даний рядок, а саме, перший користувач видалив слово «Харків» і замість нього вставив слово «Львів». Інший користувач видалив слово «Київ» і замість нього вставив слово «Париж». Результат цього спільного редагування зображено на рис 4.3. Розглянемо цю процедуру детальніше. В цьому нам допоможе граф змін, який відображає кожному зміну в одиницю часу, який зображений на рис 4.4.

Як зазначено вище, перший користувач ввів початковий рядок «Привіт Одеса, Київ, Миколаїв, Харків, Боярка», який знаходиться у вершині 9ae1971, ці зміни залишилися там збережені. Далі перший клієнт і другий почали одночасно редагувати рядок, а саме перший видалив слово «Київ» на позиції 14 (Del@14[Київ]) і додав літеру «П», другий учасник видалив слово «Харків» Del@30[Харків] і додав літеру «Л». Ці зміни збереглися у верших 49b26de і

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		57

73b55f8 відповідно. При цьому виник конфлікт. Далі цей конфлікт був вирішений і результат записався у вершину 5584dce. Одночасно з вирішенням цього конфлікту перший користувач додав «ариж» які збереглися у вершині 61973d2.

А другий користувач ввів букви «ьвів», які збереглися у вершині 9392f2c. Далі вирішився конфлікт з вводом першого користувача, результат цього конфлікту записаний у вершині b62dc1a. І в кінці відбулося об'єднання вводу другого користувача і конфлікту, який був вирішений на попередньому кроці. Результат цього записаний у вершині dbdd282.

Отже, завдяки графу змін можна наочно проілюструвати і показати роботу алгоритму при спільному редагуванні тексту.

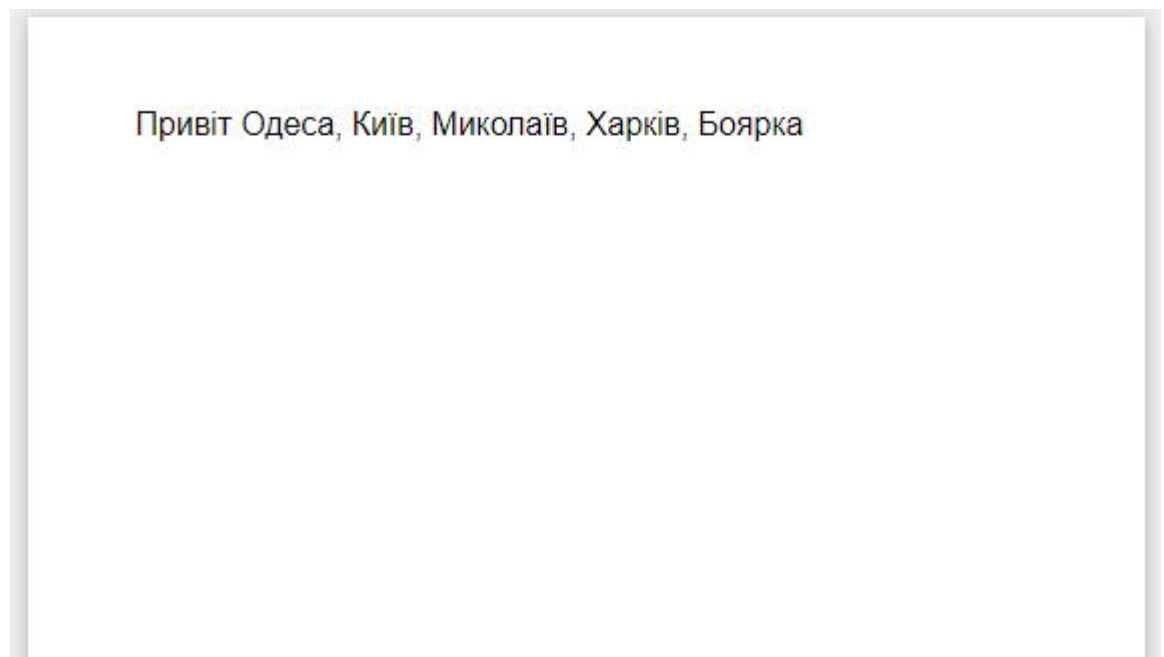


Рис 4.1. Поле для введення тексту. Зображено початкові дані, введені першим користувачем

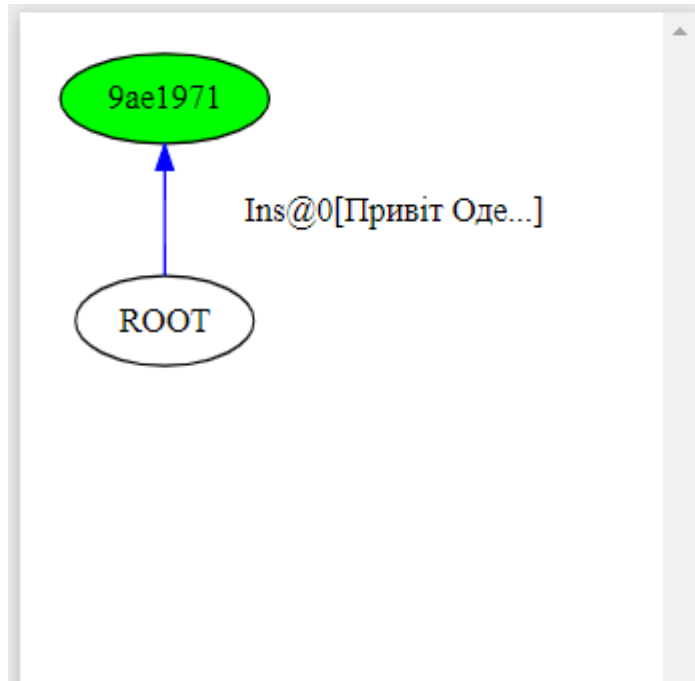


Рис 4.2. Граф змін, який відповідає первинному рядку, який зображений на рис 4.1.

Привіт Одеса, Париж, Миколаїв, Львів, Боярка

Рис 4.3. Поле для введення тексту. Змінені дані, введені як першим так і другим користувачем

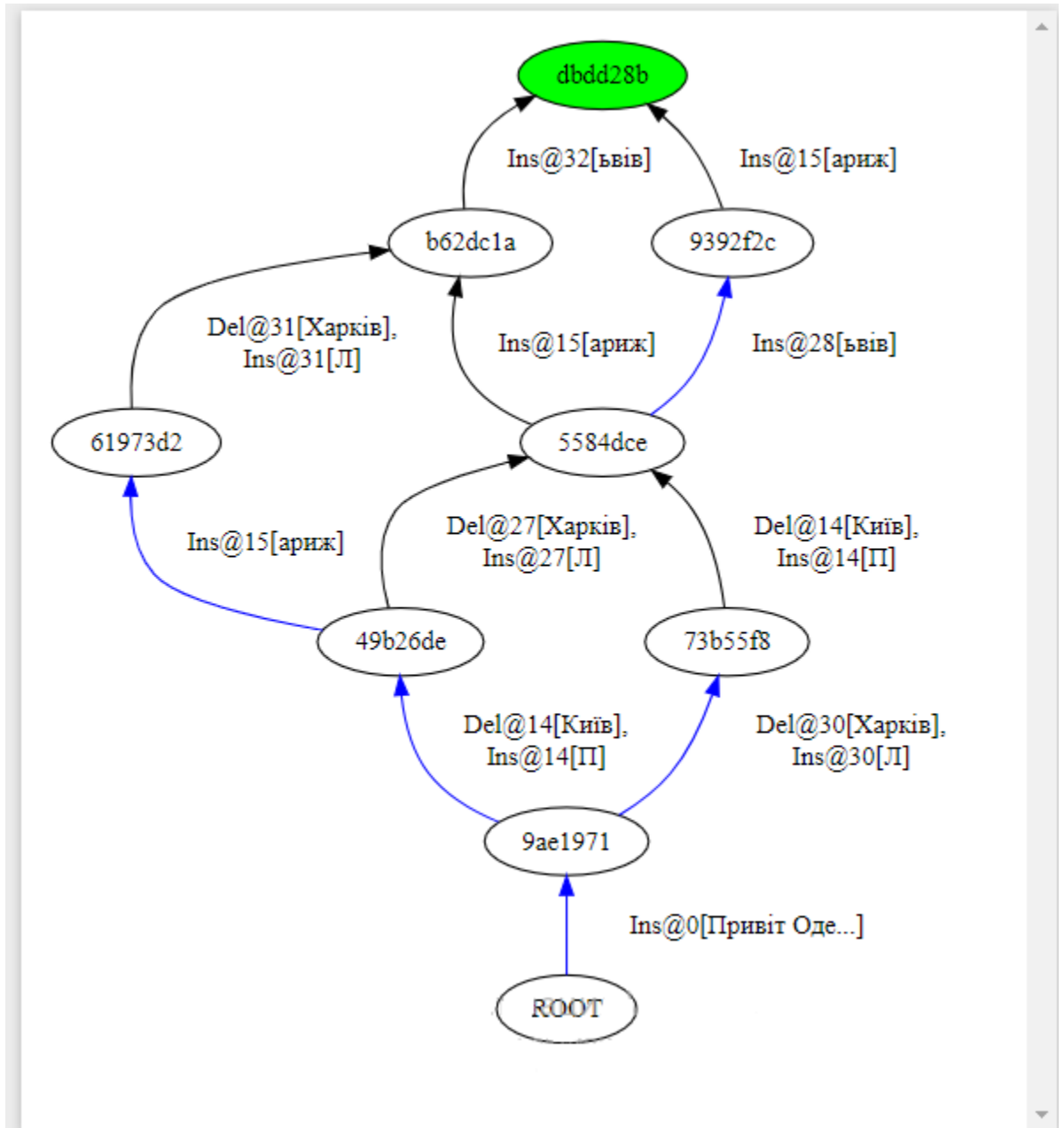


Рис 4.4. Граф змін, який відповідає зміненому рядку, який зображений на рис 4.3.

4.2 Можливі вдосконалення розробленої програми

Також важливо визначити шляхи подальшого розвитку програми, які значно поліпшать комфорт при роботі, доступність документів, та продуктивність праці:

1. Як згадувалося раніше, спільна робота між користувачами може бути синхронною або асинхронною. В даній роботі був використаний тільки асинхронний спосіб роботи. В майбутньому

можна розробити інтегровану систему, що підтримує як синхронну, так і асинхронну спільну роботу.

2. Наразі при редагуванні тексту наочно не видно, хто саме редагував текст, тому для вирішення цієї проблеми можна додати кольорові маркери, тобто для кожного користувача визначити свій особистий колір.
3. Змінити модель клієнт-сервер на серверну push-архітектуру, де повнодуплексне з'єднання з сервером залишається відкритим. Це дозволяє серверу запитувати інформацію у клієнта у будь-який момент.
4. Повідомлення про зміну документа
5. Можливість вставити в текст посилання на рядок документа, ввівши текст «# xxx», де «xxx» - номер рядка. При кліці на посиланню курсор повинен встановлюватися на початок цього рядка в документі.
6. Можливість блокування зміни окремих частин документа;

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		61

Висновок до розділу 4

Було проведено тестування розробленої програми на прикладі, при якому були вирішені певні конфлікти при редагуванні тексту декількома користувачами. Переконалися в коректності та можливості роботи системи. Вся інформація про редагування та вирішення конфліктів відображена в зручному вигляді для аналізу. Проаналізували недоліки системи та запропонували внести певні вдосконалення в майбутньому для збільшення продуктивності та поліпшення комфорту під час роботи.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		62

ВИСНОВКИ

Даний дипломний проект присвячений розробці системи автоматичного вирішення конфліктів при спільному редагуванні текстів, що забезпечує базове форматування тексту, підтримку синхронності текстів у різних користувачів і придатний для реального застосування. Проаналізувавши відому інформацію про дану проблему, стало зрозуміло, що вона є актуальною та суттєвою.

У ході виконання роботи було поставлено та вирішено наступні завдання:

- проведено визначення поняття конфлікту при спільному редагуванні документів та види конфліктів.

- здійснено огляд наслідків виникнення конфліктів на різних рівнях доступу.

- проведено дослідження причин виникнення конфліктів спільного редагуванні.

- реалізовано дослідження методології вирішення конфліктів при спільному редагуванні текстів

- надано огляд більшості існуючих методів вирішення конфліктів в текстах. Існує ряд механізмів вирішення конфліктів спільного доступу у текстах, окремі з них вже застосовуються різними системами, проте вони є досить недосконалими, або мають обмежений функціонал, адже вимагають або ручного затвердження змін, або ж відхиляють конфліктуючі зміни без можливості прийняття компромісного рішення. Проаналізувавши всю відому інформацію, було зроблено вибір на користь групи алгоритмів операціонального перетворення.

- Проведено огляд всіх існуючих систем, які використовують ОП. Виявивши всі переваги та недоліки таких систем, було прийнято рішення написати систему яка могла б знешкодити більшість недоліків, при цьому залишивши всі переваги існуючих рішень.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		63

- було здійснено детальний опис алгоритму, визначено основні сутності розробленої програми.

- протестовано та проілюстровано роботу реалізованої системи.

Розроблена система може застосовуватися на практиці при спільному написанні текстів (автор / автори і редактор / редактори), спільних перекладах текстів (дозволяє як одночасно працювати над різними фрагментами і при цьому бачити загальний прогрес, так і одночасно працювати над одним фрагментом тексту), обговоренні текстових документів (при цьому можливо робити явно видимі позначки в тексті), парному програмуванні та в інших випадках.

Даний продукт має великі можливості масштабованості та вдосконалення і може бути налаштований для роботи з різною кількістю користувачів та документів.

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		64

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Коваль Р.А. Інформаційно-аналітичне забезпечення діяльності органів влади. // Механізми державного управління. – Класичний приватний університет. – Запоріжжя, 2008.
2. High Performance Parallel Database Processing and Grid Databases. Ch. 10/ David Taniar, Clement H. C. Leung, Wenny Rahayu, Sushant Goel //Wiley Publishing, 2008, isbn: 9780470107621
3. Войтович Н.В., Найдьонова А.В. Використання хмарних технологій Google та сервісів web 2.0 в освітньому процесі. Методичні рекомендації. – Дніпро: ДПТНЗ «Дніпровський центр ПТОТС», 2017 – 113 с.
4. Захарова І.В. Основи інформаційно-аналітичної діяльності /І.В. Захарова, Л. Я. Філіпова. – К. : ЦУЛ, 2013. – 336с
5. Гусєв Є.І. Способи організації спільного доступу до розподіленим сторінок пам'яті в системах хмарних обчислень / Є.І. Гусєв // Дис. к.т.н. спец. 05.13.05. – К., НТУУ «КПІ», 2017. – 156 с.
6. Цирульник С. М. DDoS-атаки й методи боротьби з ними / С. М. Цирульник, Д. В. Кисюк, Т. О. Говорущенко // Науковий вісник Чернівецького університету. – 2009. – Вип. 446. – С. 131–134.
7. Аналітичні моделі масового обслуговування в задачах проектування інформаційних систем : навчально-довідковий посібник / уклад.: А. А. Косолапов; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: «LikePrint», ФОП Гечка Т.О., 2015. – 186 с.
8. Neil Fraser, Differential synchronization, Proceedings of the 9th ACM symposium on Document engineering, p.13-20, 2009, New York, NY, United States.
9. Operational transformation. [Електронний ресурс]. – Режим доступу: URL: https://en.wikipedia.org/wiki/Operational_transformation

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		65

10. To OT or CRDT, that is the question. [Електронний ресурс]. – Режим доступу: URL: <https://www.tiny.cloud/blog/real-time-collaboration-ot-vs-crdt/>
11. C. Sun, Y. Yarrg, Y. Zhang, ad D. Chen: “A consistency model and suppoting schemes for real-time cooperative editing systems,” In Proc. of The 19th Australasian Computer Science Confemncs, pp. 582-591, Melbourne, Jan. 1996.
12. Chengzheng Sun, Xiaohua Jia, Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems, ACM Transactions on Computer-Human Interaction, Vol. 5, No. 1, p.63-108, March 1998.
13. Chengzheng Sun, David Sun, Operation context and context-based operational transformation, Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work, p.279-288, 2006, New York, NY, United States
14. Chengzheng Sun, Clarence Ellis, Operational transformation in real-time group editors: issues, algorithms, and achievements, Proceedings of the 1998 ACM conference on Computer supported cooperative work, p.59-68, November 14-18, 1998, Seattle, Washington, United States.
15. Chengzheng Sun, Yi Xu, and Agustina, Exhaustive Search of Puzzles in Operational Transformation, February 15-19, 2014, Baltimore, MD, USAA
16. Santosh Kumawat, Ajay Khunteta. A Survey on Operational Transformation Algorithms: Challenges, Issues and Achievements International Journal of Computer Applications, July 2010
17. Client–server model [Електронний ресурс]. – Режим доступу: URL: https://en.wikipedia.org/wiki/Client-server_model
18. H. Shen, C. Sun, “Flexible Merging for Asynchronous Collaborative Systems”, Proc. of CoopIS/DOA/ODBASE 2002, pp. 304-321.

					ДП 6125. 02.000 ПЗ	Арк. 66
Зм.	Арк.	№ докум.	Підпис	Дата		

19. HTTP/1.1 vs HTTP/2: What's the Difference? [Електронний ресурс]. –

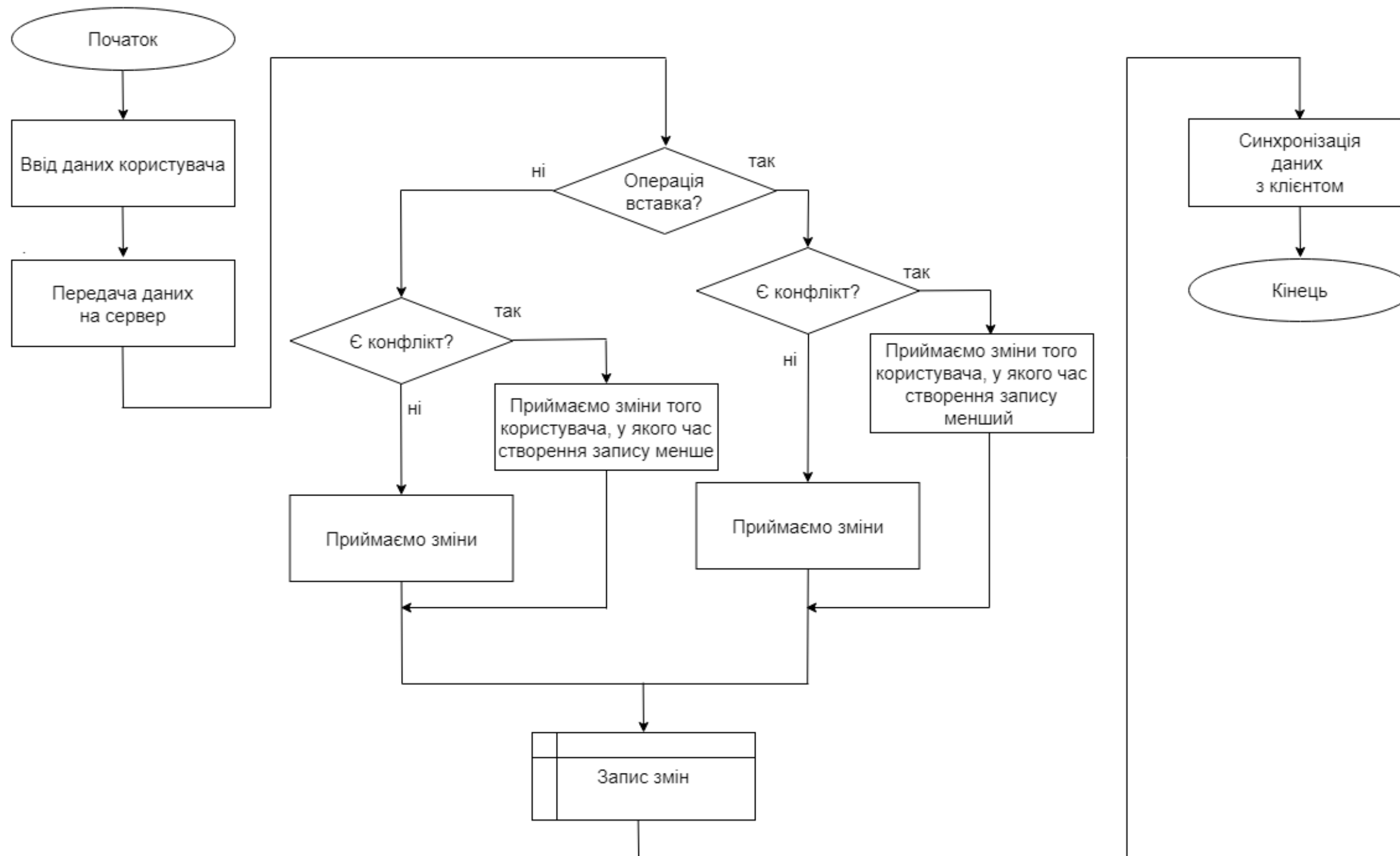
Режим

доступу:

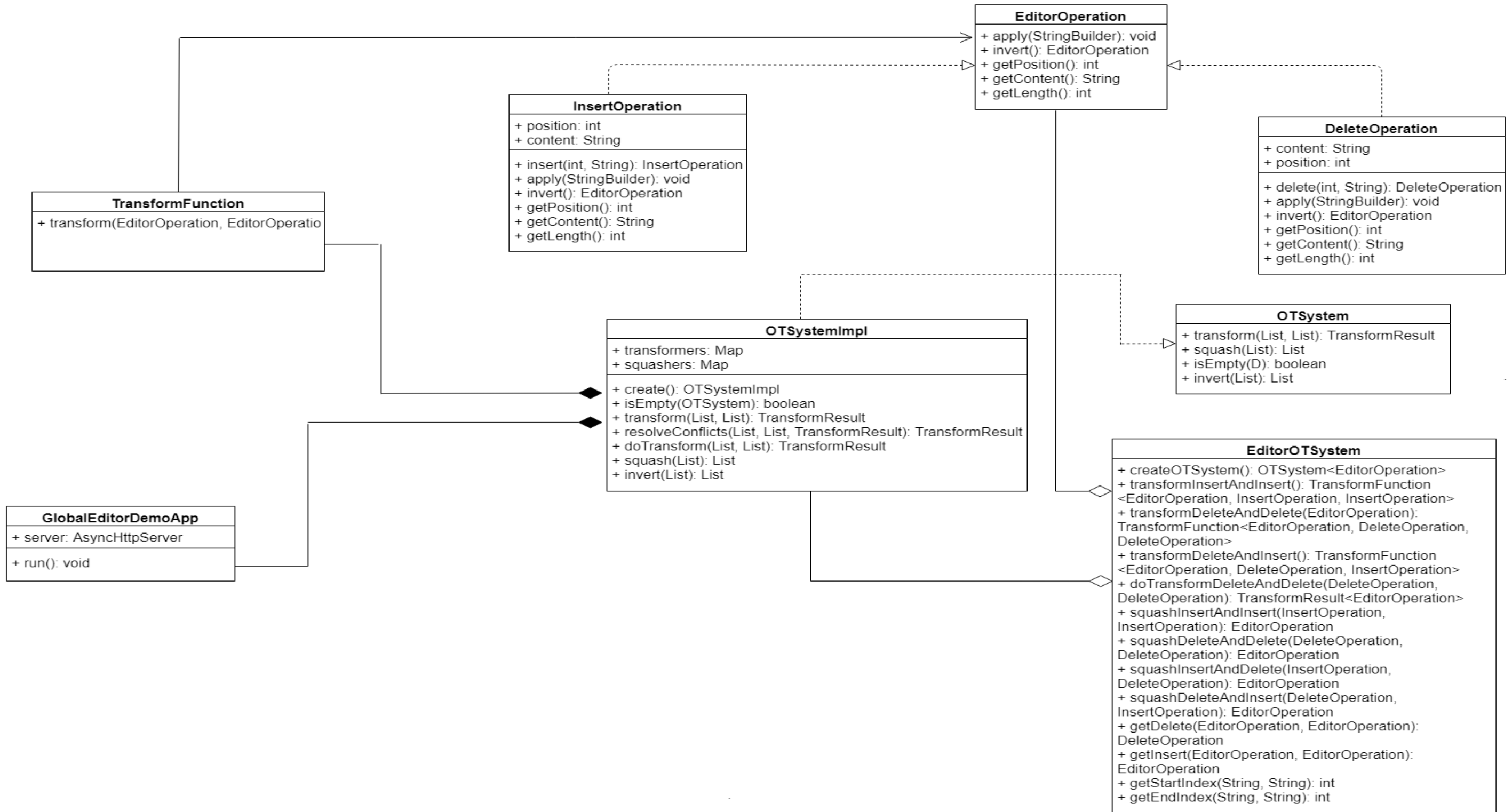
URL:

<https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference>

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		67



					<i>ДП 6125. 03.000 Д1</i>			
					<i>Схема принципова</i>	<i>Літера</i>	<i>Маса</i>	<i>Масштаб</i>
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>				
<i>Розроб.</i>		<i>Рядушкін В.В.</i>						
<i>Перевір.</i>		<i>Луцький Г.М.</i>						
<i>Т. контр.</i>						<i>Аркуш 1</i>	<i>Аркушів 1</i>	
<i>Н. контр.</i>		<i>Сімоменко В.П.</i>			<i>НТУУ "КПІ ім. Ігоря Сікорського" ФІОТ Група ІО-61</i>			
<i>Затв.</i>								



					ДП 6125. 04.000 Д2			
					Схема функціональна	Літера	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Рядушкін В.В.						
Перевір.		Луцький Г.М.						
Т. контр.						Аркуш 1	Аркушів 1	
Н. контр.		Сімоменко В.П.			НТУУ "КПІ ім. Ігоря Сікорського" ФІОТ Група ІО-61			
Затв.								

ДОДАТОК А
Лістинг програми

					<i>ДП 6125. 06.000 ДА</i>			
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>	<i>Лістинг програми</i>	<i>Літ.</i>	<i>Аркуш</i>	<i>Аркушів</i>
<i>Розробив</i>	<i>Рядушкін В.В.</i>						<i>1</i>	<i>18</i>
<i>Перевір.</i>	<i>Луцький Г.М.</i>							
<i>Н. контр.</i>	<i>Сімоненко В.П.</i>							
<i>Затверд.</i>								
						<i>НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ, ІО-61</i>		

```

public final class GlobalEditorDemoApp extends Launcher {
    public static final String EAGER_SINGLETONS_MODE =
    "eagerSingletonsMode";

    public static final String PROPERTIES_FILE = "server.properties";
    public static final String CREDENTIALS_FILE = "credentials.properties";
    public static final String DEFAULT_LISTEN_ADDRESSES = "*:8080";
    public static final String DEFAULT_SERVER_ID = "Editor Node";

    @Inject
    @Named("Example")
    AsyncHttpServer server;

    @Override
    protected Collection<Module> getModules() {
        return asList(
            ServiceGraphModule.defaultInstance(),
            ConfigModule.create(() ->
                Config.create()
                    .with("http.listenAddresses", DEFAULT_LISTEN_ADDRESSES)
                    .with("node.serverId", DEFAULT_SERVER_ID)
                    .override(Config.ofProperties(PROPERTIES_FILE, true))
                        .combine(Config.ofProperties(CREDENTIALS_FILE, true)))
                    .override(ofProperties(System.getProperties()).getChild("config")))
                .printEffectiveConfig(),
            new OTCommonModule<EditorOperation>() {
                @Override
                protected void configure() {
                    bind(new TypeLiteral<StructuredCodec<EditorOperation>>())

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		3

```

    {}).toInstance(OPERATION_CODEEC);
        bind(new TypeLiteral<Function<EditorOperation, String>>())
    {}).toInstance(Objects::toString);
        bind(new TypeLiteral<OTSystem<EditorOperation>>())
    {}).toInstance(createOTSystem());
    }
    },
    override(new GlobalNodesModule())
        .with(new ExampleCommonModule()));
}

@Override
protected void run() throws Exception {
    awaitShutdown();
}

public static void main(String[] args) throws Exception {
    new
GlobalEditorDemoApp().launch(parseBoolean(System.getProperty(EAGER_SING
LETONS_MODE)), args);
}
}

public class DeleteOperation implements EditorOperation {
    public static final StructuredCodec<DeleteOperation> DELETE_CODEEC =
object(DeleteOperation::new,
    "pos", DeleteOperation::getPosition, INT_CODEEC,
    "content", DeleteOperation::getContent, STRING_CODEEC);
}

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		4

```

@NotNull
private final String content;
private final int position;

public DeleteOperation(int position, @NotNull String content) {
    this.position = position;
    this.content = content;
}

public static DeleteOperation delete(int position, @NotNull String content) {
    return new DeleteOperation(position, content);
}

@Override
public void apply(StringBuilder builder) {
    builder.delete(position, position + content.length());
}

@Override
public EditorOperation invert() {
    return insert(position, content);
}

@Override
public int getPosition() {
    return position;
}

@NotNull

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		5

```
@Override  
public String getContent() {  
    return content;  
}
```

```
@Override  
public int getLength() {  
    return content.length();  
}
```

```
@Override  
public boolean equals(Object o) {  
    if (this == o) return true;  
    if (o == null || getClass() != o.getClass()) return false;  
  
    DeleteOperation that = (DeleteOperation) o;  
    if (position != that.position) return false;  
    if (!content.equals(that.content)) return false;  
  
    return true;  
}
```

```
@Override  
public int hashCode() {  
    int result = content.hashCode();  
    result = 31 * result + position;  
    return result;  
}
```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		6

```

    @Override
    public String toString() {
        return "Del@" + position + '[' + limit(content, 10) + ']';
    }
}

public interface EditorOperation {
    void apply(StringBuilder builder);
    EditorOperation invert();
    int getPosition();
    String getContent();
    int getLength();
}

public class EditorOTState implements OTState<EditorOperation> {
    private StringBuilder stringBuilder = new StringBuilder();

    @Override
    public void init() {
        stringBuilder = new StringBuilder();
    }

    @Override
    public void apply(EditorOperation op) {
        op.apply(stringBuilder);
    }

    public String getState() {
        return stringBuilder.toString();
    }
}
}

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		7

```

public class EditorOTSystem {
    private EditorOTSystem() {
        throw new AssertionError();
    }
    public static OTSystem<EditorOperation> createOTSystem() {
        return OTSystemImpl.<EditorOperation>create()
            .withTransformFunction(InsertOperation.class, InsertOperation.class,
transformInsertAndInsert())
            .withTransformFunction(DeleteOperation.class, DeleteOperation.class,
transformDeleteAndDelete())
            .withTransformFunction(DeleteOperation.class, InsertOperation.class,
transformDeleteAndInsert())
            .withEmptyPredicate(InsertOperation.class, op ->
op.getContent().isEmpty())
            .withEmptyPredicate(DeleteOperation.class, op ->
op.getContent().isEmpty())
            .withInvertFunction(InsertOperation.class, op ->
singletonList(op.invert()))
            .withInvertFunction(DeleteOperation.class, op ->
singletonList(op.invert()))
            .withSquashFunction(InsertOperation.class, InsertOperation.class,
EditorOTSystem::squashInsertAndInsert)
            .withSquashFunction(DeleteOperation.class, DeleteOperation.class,
EditorOTSystem::squashDeleteAndDelete)
            .withSquashFunction(DeleteOperation.class, InsertOperation.class,
EditorOTSystem::squashDeleteAndInsert)
            .withSquashFunction(InsertOperation.class, DeleteOperation.class,

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		8

```

EditorOTSystem::squashInsertAndDelete);
}
@NotNull
private static TransformFunction<EditorOperation, InsertOperation,
InsertOperation> transformInsertAndInsert() {
return (left, right) -> {
if (left.getPosition() == right.getPosition() &&
left.getContent().equals(right.getContent())) {
return TransformResult.empty();
}
if (left.getPosition() == right.getPosition() &&
left.getContent().compareTo(right.getContent()) < 0 ||
left.getPosition() < right.getPosition()) {
return TransformResult.of(insert(right.getPosition() + left.getLength(),
right.getContent()), left);
}
return TransformResult.of(right, insert(left.getPosition() + right.getLength(),
left.getContent()));
};
}
@NotNull
private static TransformFunction<EditorOperation, DeleteOperation,
DeleteOperation> transformDeleteAndDelete() {
return (left, right) -> {
if (left.getPosition() > right.getPosition()) {
TransformResult<EditorOperation> result =
doTransformDeleteAndDelete(right, left);
return TransformResult.of(result.right, result.left);
} else {

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		9

```

        return doTransformDeleteAndDelete(left, right);
    }
};
}
@NotNull
private static TransformFunction<EditorOperation, DeleteOperation,
InsertOperation> transformDeleteAndInsert() {
    return (left, right) -> {
        if (left.getPosition() == right.getPosition()) {
            return TransformResult.of(right, delete(right.getPosition() +
right.getLength(), left.getContent().substring(0, left.getLength())));
        }
        if (right.getPosition() > left.getPosition() && left.getPosition() +
left.getLength() > right.getPosition()) {
            int index = right.getPosition() - left.getPosition();
            return TransformResult.right(delete(left.getPosition(),
left.getContent().substring(0, index) +
right.getContent() + left.getContent().substring(index)));
        }
        if (left.getPosition() > right.getPosition()) {
            return TransformResult.of(right, delete(left.getPosition() +
right.getLength(), left.getContent()));
        }
        return TransformResult.of(insert(right.getPosition() - left.getLength(),
right.getContent()), left);
    };
}
private static TransformResult<EditorOperation>
doTransformDeleteAndDelete(DeleteOperation left, DeleteOperation right) {

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		10

```

if (left.equals(right)) {
    return TransformResult.empty();
}
if (left.getPosition() == right.getPosition()) {
    if (left.getLength() > right.getLength()) {
        return TransformResult.right(delete(left.getPosition(),
            left.getContent().substring(left.getLength() - right.getLength())));
    } else {
        return TransformResult.left(delete(left.getPosition(),
            right.getContent().substring(right.getLength() - left.getLength())));
    }
}
if (left.getPosition() + left.getLength() > right.getPosition()) {
    if (left.getPosition() + left.getLength() >= right.getPosition() +
right.getLength()) {
        return TransformResult.right(delete(left.getPosition(),
            left.getContent().substring(0, right.getPosition() - left.getPosition()) +
            left.getContent().substring(right.getPosition() - left.getPosition() +
right.getLength())));
    }
    return TransformResult.of(
        delete(left.getPosition(), right.getContent().substring(left.getPosition() +
left.getLength() - right.getPosition())),
        delete(left.getPosition(), left.getContent().substring(0, right.getPosition()
- left.getPosition()))
    );
}
return TransformResult.of(delete(right.getPosition() - left.getLength(),
right.getContent()), left);

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		11

```

}
@Nullable
private static EditorOperation squashInsertAndInsert(InsertOperation first,
InsertOperation second) {
    if (first.getPosition() <= second.getPosition() && first.getPosition() +
first.getLength() >= second.getPosition()) {
        return insert(first.getPosition(),
            first.getContent().substring(0, second.getPosition() - first.getPosition()) +
            second.getContent() +
            first.getContent().substring(second.getPosition() -
first.getPosition()));
    } else {
        return null;
    }
}
@Nullable
private static EditorOperation squashDeleteAndDelete(DeleteOperation first,
DeleteOperation second) {
    if (second.getPosition() <= first.getPosition() && second.getPosition() +
second.getLength() >= first.getPosition()) {
        return delete(second.getPosition(),
            second.getContent().substring(0, first.getPosition() -
second.getPosition()) +
            first.getContent() +
            second.getContent().substring(first.getPosition() -
second.getPosition()));
    } else {
        return null;
    }
}

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		12

```

}
@Nullable
private static EditorOperation squashInsertAndDelete(InsertOperation first,
DeleteOperation second) {
    if (second.getPosition() == first.getPosition() &&
second.getContent().equals(first.getContent())) {
        return insert(0, "");
    }
    if (second.getPosition() <= first.getPosition() &&
        second.getPosition() + second.getLength() >= first.getPosition() &&
        second.getPosition() + second.getLength() >= first.getPosition() +
first.getLength()) {
        return getDelete(first, second);
    }
    if (first.getPosition() <= second.getPosition() &&
        first.getPosition() + first.getLength() >= second.getPosition() &&
        first.getPosition() + first.getLength() >= second.getPosition() +
second.getLength()) {
        return getDelete(second, first).invert();
    }
    return null;
}
@Nullable
private static EditorOperation squashDeleteAndInsert(DeleteOperation first,
InsertOperation second) {
    if (first.getPosition() == second.getPosition()) {
        if (second.getLength() <= first.getLength()) {
            if (first.getContent().startsWith(second.getContent())) {
                return delete(first.getPosition() + second.getLength(),

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		13

```

        first.getContent().substring(second.getLength()));
    }
    if (first.getContent().endsWith(second.getContent())) {
        return delete(first.getPosition(),
            first.getContent().substring(0, first.getLength() -
second.getLength()));
    }
    } else {
        return getInsert(first, second);
    }
}
return null;
}
@NotNull
private static DeleteOperation getDelete(EditorOperation first, EditorOperation
second) {
    return delete(second.getPosition(),
        second.getContent().substring(0, first.getPosition() - second.getPosition())
+
        second.getContent().substring(first.getPosition() - second.getPosition()
+ first.getLength()));
}
@Nullable
private static EditorOperation getInsert(EditorOperation first, EditorOperation
second) {
    assert second.getLength() > first.getLength();
    assert first.getPosition() == second.getPosition();
    String content = second.getContent();
    String subContent = first.getContent();

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		14

```

int index = first.getPosition();
int startIndex = getStartIndex(subContent, content);
int endIndex = getEndIndex(subContent, content);
if (startIndex == -1 && content.length() - subContent.length() == endIndex) {
    return insert(index, content.substring(0, endIndex));
}
if (subContent.length() == startIndex) {
    return insert(index + startIndex, content.substring(startIndex));
}
if (startIndex != -1 && endIndex != -1 && content.length() -
subContent.length() >= endIndex - startIndex) {
    return insert(index + startIndex,
        content.substring(startIndex, startIndex + content.length() -
subContent.length()));
}
return null;
}

private static int getStartIndex(String subContent, String content) {
    int startIndex = -1;
    for (int i = 0; i < subContent.length(); i++) {
        if (content.charAt(i) == subContent.charAt(i)) {
            startIndex = i + 1;
        } else {
            break;
        }
    }
    return startIndex;
}

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		15

```

private static int getEndIndex(String subContent, String content) {
    int endIndex = -1;
    for (int i = content.length() - 1, j = subContent.length() - 1; i >= 0 && j >= 0;
i--) {
        if (content.charAt(i) == subContent.charAt(j--)) {
            endIndex = i;
        } else {
            break;
        }
    }
    return endIndex;
}

public class InsertOperation implements EditorOperation {
    public static final StructuredCodec<InsertOperation> INSERT_CODEEC =
object(InsertOperation::new,
        "pos", InsertOperation::getPosition, INT_CODEEC,
        "content", InsertOperation::getContent, STRING_CODEEC);
    private final int position;
    @NotNull
    private final String content;
    public InsertOperation(int position, @NotNull String content) {
        this.position = position;
        this.content = content;
    }
    public static InsertOperation insert(int position, @NotNull String content) {
        return new InsertOperation(position, content);
    }
}

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		16

```

@Override
public void apply(StringBuilder builder) {
    builder.insert(position, content);
}

@Override
public EditorOperation invert() {
    return delete(position, content);
}

@Override
public int getPosition() {
    return position;
}

@NotNull
@Override
public String getContent() {
    return content;
}

@Override
public int getLength() {
    return content.length();
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    InsertOperation that = (InsertOperation) o;
    if (position != that.position) return false;
    if (!content.equals(that.content)) return false;
}

```

```

    return true;
}
@Override
public int hashCode() {
    int result = position;
    result = 31 * result + content.hashCode();
    return result;
}
@Override
public String toString() {
    return "Ins@" + position + '[' + limit(content, 10) + ']';
}
}

```

					ДП 6125. 02.000 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		18