ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)
Факультет математики, фізики та інформаційних технологій
(повне найменування інституту/факультету)
Кафедра механіки, автоматизації та інформаційних технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «магістр»
«Інформаційна система автоматизації роботи кафедри»
«Information system for automating the work of the department»

Виконав: здобувач заочної форми навчання
спеціальності 126 Інформаційні системи та технології
Освітня програма Інформаційні системи та технології
Ярошенко Денис Євгенович
(прізвище, ім'я, по-батькові здобувача)

Керівник к.ф.-м.н., доцент Рачинська А.Л.
(науковий ступінь, вчене звання, прізвище та ініціали, підпис)
Рецензент к.т.н., доцент Косой М.Б.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№ ____ від _____ 2024 р.

Захищено на засіданні ЕК №
протокол №____ від _____ 2024 р.
Оцінка ___ / _____ / _____
(за національною шкалою, шкалою ECTS, бали)

Завідувачка кафедри
_____ Алла РАЧИНСЬКА
(підпис)

Голова ЕК
_____ Володимир ВИЧУЖАНІН
(підпис)

Одеса – 2024

АНОТАЦІЯ

У кваліфікаційній роботі розробляється тема «Інформаційна система автоматизації роботи кафедри».

Метою роботи є проектування системи електронного документообігу кафедри ВИШУ. Кафедра є основним навчально-науковим структурним підрозділом ОНУ імені І. І. Мечникова, що проводить освітню і методичну діяльність з однієї або кількох споріднених спеціальностей. Для організації навчального процесу на кафедрі в університеті існує внутрішній документообіг. Найважливішим документом кафедри для забезпечення навчального процесу є документ загального навантаження кафедри та індивідуального навантаження кожного викладача.

В кваліфікаційній роботі спроектовано інформаційну систему для формування документу навантаження кафедри з використанням компонентно-орієнтованого програмування. Формування документи виконується згідно всім необхідним етапам: загальний обсяг навантаження кафедри, індивідуальне навантаження викладача, звіти за семестр та навчальний рік.

Інформаційна система проводить автоматичний розрахунок навантаження однієї кафедри на заданий навчальний рік з урахуванням всіх вхідних документів ОНУ, на основі яких формується навантаження кафедри.

Система має функціонал власного навчання алгоритмам обміну даними між документами, що повинен проводити фахівець кафедри. Для зручності користувача розроблено візуалізацію алгоритмів навчання. Існуючі алгоритми навчання зберігаються і можуть використовувати повторно при необхідності.

ABSTRACT

The qualification work develops the topic "Information system for automating the work of the department".

The purpose of the work is to design an electronic document flow system for the VYSHU department. The department is the main educational and scientific structural unit of the I. I. Mechnikov ONU, which carries out educational and methodological activities in one or more related specialties. To organize the educational process at the department, there is an internal document flow at the university. The most important document of the department for ensuring the educational process is the document of the general workload of the department and the individual workload of each teacher.

In the qualification work, an information system is designed for forming a document of the department's workload using component-oriented programming. The formation of the document is carried out according to all necessary stages: the total workload of the department, the individual workload of the teacher, reports for the semester and the academic year.

The information system automatically calculates the workload of one department for a given academic year, taking into account all incoming documents of ONU, on the basis of which the workload of the department is formed.

The system has the functionality of its own training of data exchange algorithms between documents, which must be carried out by a specialist of the department. For the convenience of the user, visualization of training algorithms has been developed. Existing training algorithms are stored and can be reused if necessary.

ЗМІСТ

стор.

ВСТУП

Актуальність теми.

Електронний документообіг стає поширеним всюди – як в органах державної влади, так і на підприємствах, в IT-компаніях, банках, страхуванні, медицині, освітніх закладах та безлічі інших сфер. В умовах повсюдної цифровізації, ще й під час повномасштабної війни, неможливо працювати лише з паперовими документами, особливо якщо мова йде про великі компанії. Тому електронні документи стають незамінними.

Документообіг – це рух документів в компанії від їхнього створення (чи отримання від контрагента) до передачі в архів. Електронний документообіг – це, відповідно, такий же процес, тільки з документами в електронному вигляді.

Кожна установа має свої процеси роботи з документами, але зазвичай, до них належить створення чи одержання документа, обробка та редагування, узгодження й підписання, надсилання контрагенту чи передача на виконання, архівування. Залежно від виду документа, компанії, налаштованих процесів ці етапи можуть доповнюватись та відрізнятись. Всі процеси, які відбуваються з паперовими документами, можуть бути і в електронному форматі, завдяки чому оптимізуються, стають швидшими та дешевшими.

Документальне забезпечення будь-якого процесу має надзвичайно важливе значення, адже, у підсумку, саме документи є підставою для підтвердження фактів і мають доказову силу. Особливо важливо це розуміти при організації обліку, адже підставою для бухгалтерського обліку господарських операцій є саме первинні документи. Для контролю та впорядкування оброблення даних на підставі первинних документів можуть складатися зведені облікові документи. Інформація, що міститься у прийнятих до обліку первинних документах, систематизується на рахунках бухгалтерського обліку в регістрах синтетичного й аналітичного обліку та

узагальнюється у звітності. Тобто, саме первинні документи (документи, які містять відомості про господарську операцію), є першоджерелом, основою всього уявлення про діяльність підприємства.

В роботі досліджується електронний документообіг кафедри.

Мета та задачі дослідження. Метою роботи є проектування системи електронного документообігу кафедри ВИШУ.

Для досягнення поставленою мети необхідно розв’язати наступні задачі:

1. Проаналізувати існуючий документообіг кафедри ОНУ.
2. Спроектувати архітектуру інформаційної системи документообігу з використанням компонентно-орієнтованого програмування.
3. Розробити функціонал інформаційної системи документообігу з урахуванням всіх необхідних етапів впровадження документації кафедри.
4. Провести аналіз отриманих результатів.

Об’єкт дослідження дипломної роботи – автоматичний документообіг кафедри ОНУ.

Предмет дослідження – документ навантаження кафедри ОНУ.

1 КОМПОНЕНТНО-ОРІЄНТОВАНИЙ ПІДХІД АВТОМАТИЗАЦІЇ ДОКУМЕНТООБІГУ

1.1 Система документообігу організації

Система електронного документообігу – це програмне рішення, яке дозволяє організувати роботу з е-документами та (або) обмін ними всередині компанії чи з зовнішніми отримувачами.

В системі ЕДО можна створювати документи, відправляти та отримувати їх на узгодження, підписувати, керувати рухом документів, розмежовувати доступ до них, зберігати та відправляти в архів тощо.

Залежно від виду документообігу системи теж діляться на внутрішні (корпоративні) та зовнішні.

Системи внутрішнього (корпоративного) документообігу дозволяють обмінюватися документами всередині компанії. А системи зовнішнього електронного документообігу – це рішення для обміну та підписання документів з контрагентами, клієнтами та іншими користувачами поза межами компанії.

Документообіг буває внутрішній та зовнішній. Внутрішній документообіг (його ще іноді називають корпоративним) – це рух документів всередині компанії, тобто обмін документами між різними підрозділами та співробітниками однієї організації.

В межах однієї компанії відбувається створення, узгодження та підписання великої кількості документів – наказів, службових записок, заяв, інструкцій, статутів і багато іншого. Все це можна вести в електронному вигляді та автоматизувати. ЕДО дозволяє не лише перевести документи в електронний формат, а налаштувати процеси та організувати роботу з документами максимально зручно і швидко.

Внутрішній електронний документообіг дозволяє значно оперативніше створювати, узгоджувати та шукати документи, мати історію всіх файлів, організувати легкий доступ відповідальних співробітників до документів,

контролювати, де знаходяться документи та на якому етапі виконання завдань із ними тощо.

Зовнішній документообіг – це рух документів за межами компанії, тобто обмін документами із контрагентами, клієнтами, партнерами, державними органами та іншими зовнішніми організаціями.

Зазвичай зовнішній документообіг передбачає погодження рахунків, актів виконаних робіт, звітів, накладних тощо. Електронний зовнішній документообіг дозволяє миттєво передавати документи контрагентам, не витрачаючи часу й коштів на друк та пересилку. А також ЕДО підвищує захист документів, тож обмін стає значно безпечнішим.

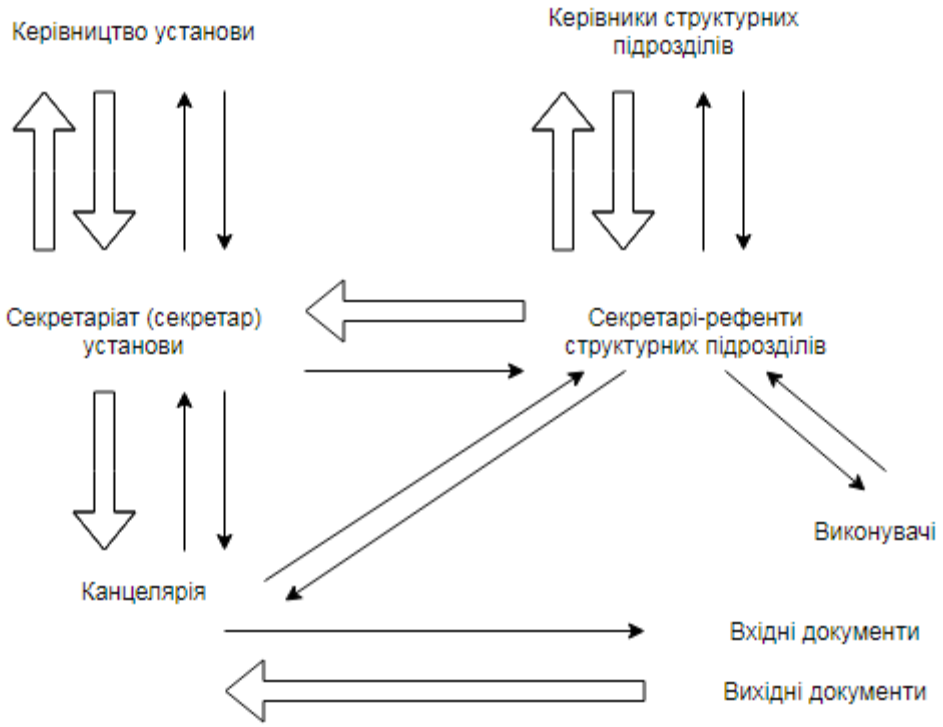

Рисунок 1.1 Схема документообігу для підприємства з обсягом документообігу від 25 до 100 тис. документів на рік.

Одеський національний університет імені І. І. Мечникова є самоврядним (автономним) національним вищим навчальним закладом, що діє згідно з виданою ліцензією на провадження освітньої діяльності за

різними ступенями вищої освіти (в тому числі доктора філософії), виконує фундаментальні та прикладні наукові дослідження, є провідним науковим і методичним центром, має розвинуту інфраструктуру навчальних, наукових і науково-дослідних підрозділів, сприяє поширенню наукових знань та здійснює культурно-освітню діяльність [1].

ОНУ імені І. І. Мечникова є юридичною особою, утвореною у формі державної установи, що діє на засадах неприбутковості. Факультет – це структурний підрозділ ОНУ імені І. І. Мечникова, що об’єднує не менш як три кафедри та/або лабораторії, які у сукупності забезпечують підготовку не менше 200 здобувачів вищої освіти денної форми навчання.

Кафедра є основним навчально-науковим структурним підрозділом ОНУ імені І. І. Мечникова, що проводить освітню і методичну діяльність з однієї або кількох споріднених спеціальностей, спеціалізацій чи навчальних дисциплін (модулів) і здійснює наукову, науково-дослідну та науково-технічну діяльність за певним напрямом.

Завідувач кафедри: 1) організовує навчальний процес на кафедрі; 2) розподіляє обсяг навчального навантаження між викладачами; 3) організовує керівництво аспірантами, докторантами, здобувачами; 4) формує пропозиції щодо зміни штатного розпису; 5) визначає напрями наукової роботи, організовує наукові дослідження; 6) організовує підвищення кваліфікації викладачів і співробітників; 7) дає висновок із відповідними рекомендаціями під час прийняття на роботу науково-педагогічних працівників та продовження трудових угод.

Для організації навчального процесу на кафедрі в університеті існує внутрішній документообіг. На основі освітніх програм різних спеціальностей деканат формує заявки навантаження для кожної кафедри, яка забезпечує на вказаній освітній програмі навчальний процес. На основі заявок навантаження завідувач кафедри або відповідальна особа на кафедри формує внутрішній документ кафедри – навантаження кафедри. Навантаження кафедри має складові: загальний обсяг кафедри та навантаження кожного

викладача кафедри. Більшість документів внутрішнього документообігу кафедри – це документи – Excel.

Microsoft Excel (також іноді називається Microsoft Office Excel [2-4]) - програма для роботи з електронними таблицями, створена корпорацією Microsoft для Microsoft Windows, Windows NT та Mac OS, а також Android, iOS та Windows Phone. Вона надає можливості економіко-статистичних розрахунків, графічні інструменти, мову макропрограмування потоків даних Power Query та, за винятком Excel 2008 під Mac OS X, мову макропрограмування VBA (Visual Basic for Application). Microsoft Excel входить до складу Microsoft Office.

У 1982 році Microsoft запустила на ринок свій перший електронний табличний процесор Multiplan, який був дуже популярним на CP/M системах, але на MS-DOS системах він поступався Lotus 1-2-3. Перша версія Excel призначалася для Mac і була випущена в 1985, а перша версія для Windows була випущена в листопаді 1987 року. Lotus не поспішала випускати 1-2-3 під Windows, і Excel з 1988 року почала обходити з продажу 1-2-3, що врешті-решт допомогло Microsoft досягти позицій провідного розробника програмного забезпечення. Microsoft зміцнювала свою перевагу з випуском кожної нової версії, що мало місце кожні два роки. Поточна версія для платформи Windows - Excel 19 також відома як Microsoft Office Excel 2019. Поточна версія для платформи macOS - Microsoft Excel 2019.

На початку свого шляху Excel став причиною позову про товарний знак від іншої компанії, яка вже продала пакет програм під назвою Excel. В результаті спору Microsoft була зобов'язана використовувати назву Microsoft Excel у всіх своїх офіційних прес-релізах та юридичних документах. Однак згодом ця практика була забута, і Microsoft остаточно усунула проблему, придбавши товарний знак іншої програми. Microsoft також вирішила використовувати літери XL як скорочену назву програми: іконка Windows-програми складається із стилізованого зображення цих двох літер, а розширення стандартних файлів в Excel — .xls.

У порівнянні з першими табличними процесорами Excel представляє безліч нових функцій інтерфейсу користувача, але суть залишається колишньою: як і в програмі-родоначальнику, VisiCalc, організовані в рядки і стовпці клітини-комірки можуть містити дані або формули з відносними або абсолютними посиланнями на інші клітини.

Excel був першим табличний процесор, що дозволяв користувачеві змінювати зовнішній вигляд таблиці на екрані: шрифти, символи та зовнішній вигляд осередків. Він також першим представив метод розумного перерахунку осередків - оновлення тільки осередків, що залежать від змінених осередків: раніше табличні процесори перераховували всі осередки; це робилося або після кожної зміни (що на великих таблицях довго), або за командою користувача (що могло вводити користувача в оману не перерахованими значеннями).

Будучи вперше об'єднаними у Microsoft Office у 1993 році, Microsoft Word та Microsoft PowerPoint отримали новий графічний інтерфейс для відповідності Excel, головного стимулу модернізації ПК на той час.

Починаючи з 1993 року, до складу Excel входить Visual Basic додатків (VBA), мова програмування, заснований на Visual Basic, що дозволяє автоматизувати завдання Excel. VBA є потужним доповненням до програми і в пізніших версіях Excel доступне повнофункціональне інтегроване середовище розробки. Можна створити VBA-код, який повторює дії користувача і таким чином автоматизувати прості завдання. VBA дозволяє створювати форми спілкування з користувачем. Мова підтримує використання (але не створення) DLL ActiveX; пізніші версії дозволяють використовувати елементи об'єктно-орієнтованого програмування.

Функціональність VBA робила Excel легкою метою для макровірусів. І це було серйозною проблемою доти, доки антивірусні продукти не навчилися виявляти їх. Фірма Microsoft, із запізненням вживши заходів для зменшення ризику, додала можливість вибору режиму безпеки:

- повністю відключити макроси

- увімкнути макроси при відкритті документа
- довіряти всім макросам, підписаним із використанням надійних сертифікатів.

Excel використовують представники різних сфер бізнесу. Завдяки зручній роботі з формулами та можливостями внутрішніх обчислень такі таблиці тривалий час виконували дуже важливу функцію: заміну паперових документів та необхідність ручного розрахунку різних параметрів. На перший погляд здається, що перед нами ідеальний варіант автоматизації документообігу, але статистика показує, що співробітники не проходять далі за використання стандартних формул або заповнення таблиць простими даними. Щоб опанувати всі можливості Excel, необхідно вчитися і практикуватися.

Основна перевага використання MS Excel для управлінського обліку та звітності - це гнучкість. Просунуті користувачі можуть самостійно створювати різні варіанти звітів та оперативно адаптувати їх під поточні потреби своєї організації. діяльності підприємства.

Очевидною проблемою при використанні MS Excel є наповнення таблиць MS Excel актуальною вихідною інформацією. Це досить трудомісткою роботою, а також нівелює всю гнучкість використання MS Excel - при зміні таблиць такий код доводиться переписувати.

Ні для кого не секрет, що для автоматизації документообігу на підприємствах багато хто використовує таблиці MS Excel. Деякі користувачі впевнені, що MS Excel зі всім його функціоналом справді здатний замінити повноцінну систему автоматизації документообігу, незважаючи на те, що фахівці з впровадження програмного продукту постійно намагаються роз'яснити: з мінімальними витратами важко отримати добрий результат. Численні шанувальники Excel продовжують чинити опір очевидним фактам, а як результат – втрачають час на відновлення інформації і, відповідно, гроші.

1.2 Технологія Component Object Model

COM (англ. Component Object Model - об'єктна модель компонентів; вимовляється як [ком]) - це технологічний стандарт від компанії Microsoft, призначений для створення програмного забезпечення на основі взаємодіючих компонентів, кожен з яких може використовуватися в багатьох програмах одночасно. Стандарт втілює у собі ідеї поліморфізму та інкапсуляції об'єктно-орієнтованого програмування. На основі COM були реалізовані технології Microsoft OLE Automation, ActiveX, DCOM, COM+, DirectX, а також XPCOM. Основним поняттям, яким оперує стандарт COM, є COM-компонент. Програми, побудовані на стандарті COM, фактично є автономними програмами, а є набір взаємодіючих між собою COM-компонентів. Кожен компонент має унікальний ідентифікатор (GUID) і може одночасно використовувати багато програм. Компонент взаємодіє з іншими програмами через COM-інтерфейси - набори абстрактних функцій та властивостей. Ключовим аспектом COM є те, що ця технологія забезпечує зв'язок між клієнтами та серверами засобами інтерфейсів. Інтерфейс надає можливість клієнту дізнатися, яку функцію підтримує сервер безпосередньо в процесі виконання. Для розширення можливостей сервера потрібно просто додати новий інтерфейс. COM є одночасно і специфікацією та реалізацією. Специфікація визначає правила створення об'єкта та спосіб зв'язку між об'єктами. Відповідно до специфікації об'єкти COM можуть бути написані різними мовами, виконуватися в адресному просторі різних процесів та на різноманітних платформах. Доки об'єкти повністю відповідають специфікації, вони можуть взаємодіяти. COM як реалізація є бібліотекою у разі WINDOWS OLE32.DLL, яка надає ряд основних служб, що підтримують описані специфікації [5-7].

COM клієнт-програмний код, який отримує необхідні послуги від сервера через інтерфейси об'єкта. COM клієнт знає, що він хоче отримати від сервера, але не знає, як його запит виконується всередині сервера.

Type Library містить опис COM об'єктів, його інтерфейсів, методів і його GUID ідентифікаторів.

Class Factory – екземпляр об'єкта, який створює об'єкт COM. Створюється COM сервером за запитом клієнтським додатком 1 інтерфейсу.

COM - інтерфейс-засіб, за допомогою якого об'єкт COM надає свої функціональні можливості для зовнішніх клієнтів. Будучи одного разу опублікованим, інтерфейс не змінюється за жодних обставин. Розширення функціональності можливе лише за допомогою додавання іншого інтерфейсу. Ім'я інтерфейсу завжди починається з великої літера I. Інтерфейс має унікальний іденифікатор - GUID. Для реалізації інтерфейсу може бути використана будь-яка мова програмування, яка підтримує покажчики, структури. Сам по собі інтерфейс об'єктом не є, а є способом отримати доступ до об'єкта. Будь-який інтерфейс є спадкоємцем базового класу IUnknown.

Кожен COM-компонент повинен як мінімум підтримувати стандартний інтерфейс «IUnknown», який надає базові засоби для роботи з компонентом. Інтерфейс «IUnknown» включає три методи: QueryInterface, AddRef, Release. QueryInterface повертає клієнту покажчик на запитаний ним інтерфейс за його IDD (тип класу інтерфейсу). AddRef, Release використовуються для того, щоб об'єкт COM міг самостійно відстежувати тривалість свого існування. Ці методи просто змінюють кількість посилань на об'єкт. Число посилань =0 об'єкт видаляється, AddRef +1 посилання, , Release -1 посилання.

COM сервер - деякий модуль, який містить код для об'єкта COM. Включає в себе хоча б 1 об'єкт COM, який у свою чергу є сукупністю методів і властивостей. Коли клієнт запитує сервер об'єкта COM, він має передати ідентифікатор класу CLSID, який створюється з урахуванням GUID інтерфейсу об'єкта COM. По ньому COM визначає відповідний сервер та створює екземпляр об'єкта COM.

Існує 3 види серверів COM:

In-Process-Server – бібліотека DLL, яка виконується в адресному просторі процесора клієнта.

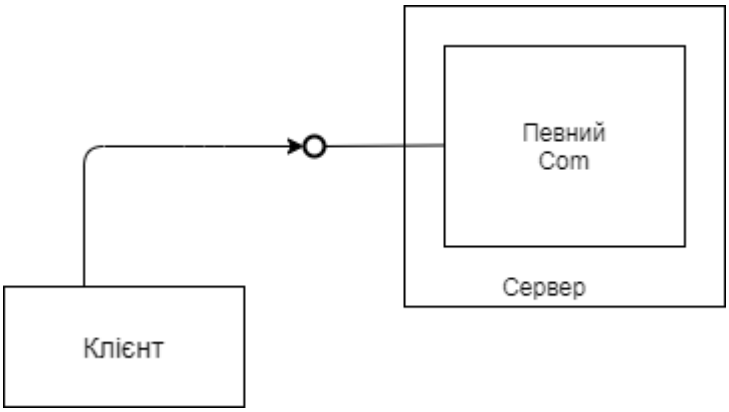

Рисунок 1.2. Перший вид com-серверів

Локальний сервер – інша програма, яка виконується в іншому адресному просторі, але на тому ж комп'ютері, що і клієнтська програма.

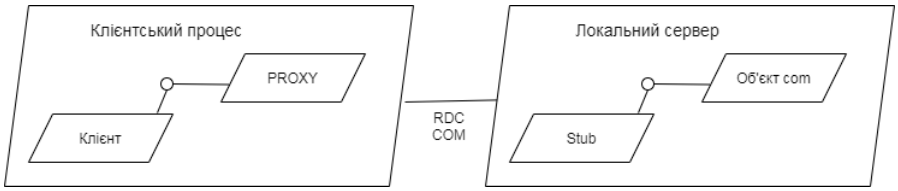

Рисунок 1.3. Другий вид com-серверів

Віддалений сервер є бібліотекою або програмою, яка виконується на іншому комп'ютері.

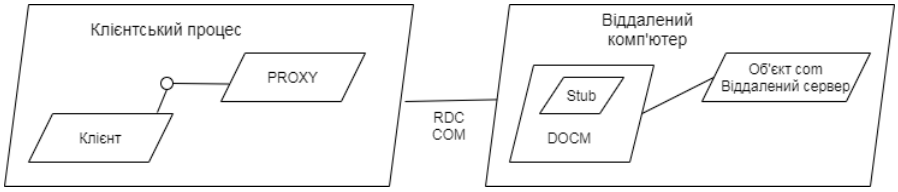

Рисунок 1.4. Третій вид com-серверів

Компонентна модель об'єктів чи технологія COM (Component Object Model) надає можливість одній програмі (клієнту) працювати з об'єктом

іншої програми (сервером). Технологія COM спрощує створення програм, забезпечуючи їх сумісність у різних версіях платформи Windows та відносну незалежність від мови програмування (компоненти COM можуть створюватися різними мовами і далі впроваджуватись у додатку шляхом використання стандартного інтерфейсу) [5-7].

Узагальнено, COM додаток може містити від одного до кількох впроваджених об'єктів, а кожен об'єкт може містити один або кілька інтерфейсів, кожен із яких містить методи об'єкта. Зовнішня програма, використовуючи методи об'єкта, виконує необхідні дії щодо обміну даними із сервером, не турбуючись про особливості внутрішньої реалізації. Сервером може бути файл, що виконується, або бібліотека.

Основним недоліком використання COM об'єктів була необхідність зберігання та реєстрації в системному реєстрі всієї інформації про всі COM компоненти. Відмова від реєстрації екземпляра компонента в реєстрі та розміщення його в доступне місце або каталог програми, частково вирішує проблеми пов'язані з використанням реєстру (відпадає необхідність у реєстрації компонента, присвоєння йому глобальних ідентифікаторів GUID і пошуку по них виконуваного файлу або бібліотеки компонента і т.п.). Реалізацією цього підходу в .NET є технологія збірки.

Збірка (assembly) - це сукупність файлів, що встановлюються на комп'ютер і необхідні виконання програмного коду програми (середовище виконання - NET Runtime Framework не входить у поняття збирання). Найчастіше створюються одне файлові збірки, які з одного EXE чи DLL-файлу.

Середовище виконання (NET Runtime Framework) працює не з інструкціями (командами) для процесорів, а з командами віртуальної машини або проміжним кодом – Microsoft Intermediate Language (MSIL), який компілюється та виконується середовищем. Тому збірка, як мінімум, має один файл коду MSIL, в якому описується збірка, функціональність класів, що входять до неї (так звані метадані) і безпосередньо програмні інструкції.

Метадані є частиною збірки, тому в документації збірки називаються самодокументованими. Крім того, до збірки можуть входити файли ресурсів, графічні файли, додаткові файли EXE, DLL-файли. Програми мовою MSIL створюють так званий "керований код" (managed code). Це означає, що загальномовне середовище CLR (Common Language Runtime), яке є частиною NET Runtime Framework, не просто перетворює MSIL в машинні інструкції, а виконує ці дії з урахуванням зовнішніх установок (Наприклад, Модуль №1 програми може задати власний набір прав і набір прав для Модуля №2). Збірка є мінімальним об'єктом .NET, в якому задаються привілеї, і здійснюється контроль версії.

Збірки, що знаходяться в каталозі додатку (або в його підкаталозі) отримали назви закритих (private) збірок, а збірки, що зберігаються у глобальному кеші збірок (Global Assembly Cache, GAC) - загальних (shared). Щоб будь-який об'єкт COM міг бути використаний у додатку, він має бути оформлений як збірка. При додаванні компонента COM у каталозі програми створюється новий екземпляр закритої збірки (зазвичай одна або кілька бібліотек .dll) і формуються посилання на об'єкти збірки.

Об'єкти, якими оперує com-сервер Excel, є кілька десятків. Усі об'єкти мають ієрархічну структуру. Сам сервер - об'єкт Application або програма Excel, може містити одну або більше книг, посилання на які містить властивість Workbooks. Книги – об'єкти Workbook, можуть містити одну або більше сторінок, посилання на які містить властивість Worksheets або (і) діаграм – властивість Charts. Сторінки - Worksheet, містять об'єкти осередку або групи осередків, посилання на які стають доступними через об'єкт Range. Нижче в ієрархії: рядки, стовпці... Аналогічно і для об'єкта Chart серії ліній, легенди...

Звернемо увагу на те, що інтерфейс C# замість поняття осередку використовує об'єкти Range (вибраний осередок або група осередків). Відзначимо також групу об'єктів ActiveCell, ActiveChart і ActiveSheet, що відносяться до активного вікна (поверх інших). Вони, за набором

властивостей і методів, повністю аналогічні об'єктам Range, Chart і Sheet і, часом, легко полегшують отримання посилання.

Трохи відокремлено від цієї ієрархічної структури об'єктів є властивість Windows об'єкта Excel.Application, призначене для керування вікнами сервера Excel. Властивість Windows містить набір об'єктів Window, які мають, у свою чергу, набір властивостей та методів для керування розмірами, виглядом, масштабом та упорядкуванням відкритих вікон, відображенням заголовків, кольорами тощо. Ці ж можливості доступні і для властивостей та методів об'єкта Excel.Application – ActiveWindow (посилання на активне вікно) [2-4].

Об'єкти Application і Windows прийнято визначати як поля класів, щоб забезпечити доступ до них з будь-якої функції класу форми проекту.

Другим у ієрархії об'єктів Excel.Application є об'єкт Workbook. Інформація про об'єкти Workbook зберігається у вигляді посилань на відкриті робочі книги як Workbooks. Книга в додаток може бути додана тільки через додавання посилання в сукупність Workbooks, а посилання на відкриту книгу може бути отримана по-різному (на ім'я, номер, як посилання на активну книгу).

Документи Excel можна зберегти програмно та звичайним для Excel способом. У будь-якому разі перед виходом із Excel необхідно викликати метод Quit. Якщо властивість Excel.Application DisplayAlerts має значення true, Excel запропонує зберегти незбереженні дані, якщо після старту в документ було внесено зміни. Excel автоматично не повертає цю властивість у значення за замовченням, тому його рекомендується повертати у вихідний стан.

Для відкриття існуючого документа основним методом є Open набору Excel.Workbooks. Для відкриття текстових файлів як робочих книг, баз даних, файлів у форматі .XML використовуються методи OpenText, OpenDatabase або OpenXml. Про використання методів OpenDatabase та OpenXml мова вестиметься в інших темах. У цьому параграфі розглянемо метод Open.

Читання інформації з осередків Excel багато в чому аналогічне виводу. На вибраному аркуші необхідно у вибраній книзі вибрати одну осередок або об'єднану групу осередків (метод get_Range або перетворення до Excel.Range) - після чого достатньо перетворити значення виділених осередків до потрібного типу даних.

Щоб намалювати табличку в Excel, треба навчитися малювати рамки навколо обраного осередку або об'єднаної групи осередків.

Кроки малювання рамки будуть наступні:

- Вибрати осередок або групу осередків на аркуші документа.
- Об'єднати комірки.
- Визначаємо колір ліній обведення. Колір може бути обраний як один з 56 кольорів палітри кольорів Excel і, тому, він задається через колірний індекс (наприклад, excelcells.Borders.ColorIndex=3; - червоний)
- Деякі значення ColorIndex 1 – білий, 2 – чорний, 3 – червоний, 4 – зелений, 6 – жовтий, 41 – синій і т.д.
- Вибрати стиль лінії (Excel.XlLineStyle.xlContinuous). Стиль лінії може бути одним з наступних: xlContinuous, xlDash, xlDashDot, xlDashDotot, xlDot, xlDouble, xlSlantDashDot, xlLineStyleNone.
- Встановити товщину лінії (Excel.XlBorderWeight.lHairline). Товщина лінії може бути однією з наступних: lHairline, xlMedium, xlThick, xlThin.

Події також необхідні для створення функціональної програми для роботи з Excel як і доступ до властивостей та методів. Якщо розглядати використання властивостей та методів, що надаються інтерфейсом COM об'єкта, як прямий зв'язок з керування сервером, то події забезпечують зворотний зв'язок. Завдяки наявності подій, програма може забезпечити програмну функціональність для відгуку на те, що відбувається в програмі.

Об'єкти Excel, як і будь-який контрол C#, мають свої події і програміст, як і контролю, може створити обробник будь-якої з подій. Однак, у нас немає

візуального компонента Excel і не можна подвійним клацанням мишки у віконці навпроти події у вікні Properties на вкладці Events створити обробник. Можна спробувати це зробити вручну.

Основні події об'єктів Excel:

Події об'єкта Application:

a.) пов'язані з поведінкою об'єктів на аркуші

- SheetActivate (відбулася активізація аркуша);

- SheetBeforeDoubleClick (виконаний подвійний клік на аркуші і після цієї події очікується якась реакція сервера - виконання оброблювача за замовчуванням);

- SheetBeforeRightClick (виконаний правий клік на аркуші і після цієї події очікується якась реакція сервера - виконання оброблювача за замовчуванням);

- SheetCalculate (виконано перерахунок формул на аркуші);

- SheetChange (зміна обраного осередку на аркуші);

- SheetDeactivate (аркуш втратив фокус);

SheetFollowHyperlink (користувач пішов за гіперпосиланням);

SheetSelectionChange (змінилося виділення на аркуші).

b.) пов'язані з поведінкою вікна

- WindowActivate (відбулася активізація вікна, якщо Excel на даний момент був активний);

- WindowDeactivate (вікно втратило фокус);

- WindowResize (змінився розмір вікна);

c.) пов'язані з керуванням робочою книгою

- NewWorkbook (створена нова робоча книга);

- WorkbookActivate (книга, один із її листів, отримали фокус);

- WorkbookAddinInstall (виконується інсталяція не встановленого компонента);

- WorkbookAddinUninstall (виконується деінсталяція встановленого компонента);

- WorkbookBeforeClose (після цієї події очікується закриття книги – виконання оброблювача за замовчуванням);

- WorkbookBeforePrint (після цієї події очікується друк аркуша – виконання оброблювача за замовчуванням);

- WorkbookBeforeSave (після цієї події очікується збереження книги – виконання оброблювача за замовчуванням);

- WorkbookDeactivate (книга втратила фокус);

- WorkbookNewSheet (до книги доданий лист);

- WorkbookOpen (відкрито робочу книгу).

Події об'єкта Workbook:

a.) всі події об'єкта Application для пункту a, пов'язані з поведінкою об'єктів на аркуші. Генеруються лише для аркушів даної робочої книжки.

b.) всі події об'єкта Application для b. Генеруються лише для аркушів даної робочої книжки.

c.) пов'язані з керуванням робочою книгою

- Activate (книга, один із її аркушів, отримали фокус);

- BeforeClose (після цієї події очікується закриття книги – виконання оброблювача за замовчуванням);

- BeforePrint (після цієї події очікується друк аркуша – виконання оброблювача за замовчуванням);

- BeforeSave (після цієї події очікується збереження книги - виконання оброблювача за замовчуванням);

- Deactivate (книга втратила фокус);

- NewSheet (до книги доданий лист);

- Open (відкрито робочу книгу).

Події об'єкта Worksheet:

a) пов'язані з поведінкою об'єктів на аркуші

- Activate (відбулася активізація аркуша);

- BeforeDoubleClick (виконаний подвійний клік на аркуші і після цієї події очікується якась реакція сервера - виконання оброблювача за замовчуванням);

- BeforeRightClick (виконаний правий клік на аркуші і після цієї події очікується якась реакція сервера - виконання оброблювача за замовчуванням);

- Calculate (виконано перерахунок формул на аркуші);

- Change (зміна обраного осередку на аркуші);

SheetDeactivate (аркуш втратив фокус);

- SheetFollowHyperlink (користувач пішов за гіперпосиланням);

- SheetSelectionChange (змінилося виділення на аркуші).

Зазначимо, що функції обробників подій Excel передаються не посилання на конкретний об'єкт типу Workbook, Worksheet, Chart, а змінна типу Object, яку перед використанням необхідно явно привести до необхідного типу [7].

Усі події, в імені яких є слово "Before", дозволяють скасувати обробку події за замовчанням. Параметр, що передається обробнику події, зазвичай іменується Cancel, а значення за замовчанням - false. Якщо встановити цей параметр true, Excel не обробляє події за замовчанням.

Взаємодія з серверами автоматизації виконується аналогічно до виконання фонового процесу і тому доступу до елементів, тим, які представлені контролами або є властивостями WindowsForm), з функцій делегатів немає (хоча при виконанні, наприклад textBox1.Text="Перейшли на лист = " + ((Excel.Worksheet) Sh). Name); переривання не буде, але відображення інформації також не буде).

Об'єкти, якими оперує com-сервер Word , також численні як і об'єкти для Excel [8]. Усі об'єкти доступні з програми на C#. Відмінність полягає в тому, що для програми-контролера доступні безпосередньо Word.Application (екземпляр Word без відкритих документів) та Word.Document (екземпляр Word з відкритим або завантаженим документом). Інші об'єкти Word є так

званими внутрішніми об'єктами. Це означає, що вони не можуть бути створені власними силами; так, об'єкт Paragraph не може бути створений без документа (втім, параграф або абзац недоступний і макросу в Word за відсутності відкритого документа).

З точки зору програми, всі об'єкти Word мають ієрархічну структуру. Об'єкт Application - це сервер COM і оболонка для інших об'єктів. Він може містити один або більше об'єктів документа. Об'єкти Document можуть містити такі об'єкти як Paragraph, Table, Range, Bookmark, Chapter, Word, Sentence, Sections, Headers, Footers... Більш правильніше, об'єкт Application може містити колекцію Documents - посилання на об'єкти типу Document, а кожен об'єкт типу Document - Колекцію Paragraphs або посилань на об'єкти типу Paragraph і т.д.

Робота з документами, параграфами, символами, закладками тощо здійснюється шляхом використання властивостей та методів цих об'єктів. Об'єкти під час створення рішень прийнято визначати глобально у тому, щоб забезпечити доступом до них з будь-якої функції проекту.

Як зазначалося, основним у ієрархії об'єктів Word.Application є об'єкт Document. Інформація про об'єкти Document зберігається у вигляді посилань на відкриті документи як Documents. Документ в додаток може бути доданий тільки через додавання посилання в сукупність Documents, а посилання на відкритий документ може бути отримана по-різному (на ім'я, номер, як посилання на активний документ).

Документи Word можна зберегти програмно та звичайним для Word способом. У будь-якому випадку перед виходом із Word необхідно викликати метод Quit. Якщо властивість Word.Application DisplayAlerts має значення true, Word запропонує зберегти дані у разі, коли після старту у документ було внесено будь-які зміни.

Для збереження документа можна використовувати методи Save() або SaveAs. Метод Save() не має параметрів і під час його виклику буде відображено діалогове вікно Word "Збереження документа". Навпаки, метод

SaveAs має безліч параметрів, більшість з яких можна не вказувати, а використовувати як стандартні параметри.

Будь-який документ Word завжди має хоча б один параграф, навіть якщо він порожній, то завжди є курсор введення. Виведення тексту виконується непросто у параграф, а діапазон параграфа - об'єкт Range. Об'єкт Range – це безперервна область документа, що включає позицію початкового та кінцевого символів. Для порожнього параграфа або якщо початкова та кінцева позиції діапазону збігаються - Range представляє курсор введення.

Об'єкт Selection є поточною виділеною областю документа. Всі дії, пов'язані з внесенням до документу будь-яких приватних змін у Word виконуються стосовно виділених фрагментів. Програмно це виконується через об'єкт Selection. Аналогічно, як і при звичайній роботі з Word, при програмному висновку необхідно спочатку виділити фрагмент і далі виконати необхідні дії над цим фрагментом (змінити шрифт, форматування, надрукувати і т.д.). Об'єкт Selection завжди є у документі. Якщо не виділено, він представляє курсор введення (insertion point).

1.3 Використання патернів для проектування ІС

Технічно, патерни (шаблони) проектування - це лише абстрактні приклади правильного використання невеликої кількості комбінацій найпростіших технік ООП. Паттерни проектування - це прості приклади, що показують правильні способи організації взаємодій між класами або об'єктами.

Паттерни (шаблони) проектування – це готові приклади, які описують:

• правильні способи формування внутрішнього стану (полів) та поведінки (методів) об'єкта чи класу;

• правильні способи створення об'єкта (через виклик конструктора чи іншим способом);

• правильні способи об'єднання об'єктів у групи;

• правильні способи організації інформаційних потоків (дзвінків методів та черговості викликів) що дозволяють налагодити гармонійну взаємодію між об'єктами та групами цих об'єктів в об'єктно-орієнтованих системах.

Паттерни проектування допомагають уявити об'єктно-орієнтовану систему формально, відображаючи результати мислення проектувальника в комбінаціях двадцяти трьох точних понять та тверджень.

Об'єктно-орієнтовані шаблони показують відносини та взаємодії між класами чи об'єктами, без визначення того, які кінцеві класи чи об'єкти програми використовуватимуться.

"Низькорівневі" шаблони, що враховують специфіку конкретної мови програмування, називаються ідіомами. Це хороші рішення проектування, характерні для конкретної мови чи програмної платформи, і тому не універсальні.

На найвищому рівні існують архітектурні шаблони, вони охоплюють архітектуру всієї програмної системи.

Алгоритми по суті також є шаблонами, але не проектування, а обчислення, оскільки вирішують обчислювальні завдання.

Важливо розуміти, що патерн – це формула, а приклад використання патерну – це приклад застосування цієї формули.

Усі патерни поділені на три групи. Перша група патернів - шаблони, що породжують (creational patterns), призначені для забезпечення виконання одного з найпоширеніших завдань в ООП: створення об'єктів у системі. У ході роботи більшості об'єктно-орієнтованих систем, незалежно від рівня їхньої складності, створюється безліч екземплярів об'єктів. Шаблони, що породжують, полегшують процес створення об'єктів, надаючи розробнику наступні можливості:

- Єдиний спосіб одержання екземплярів об'єктів. У системі забезпечується механізм створення об'єктів без необхідності ідентифікації певних типів класів програмного коду.

- Простота. Деякі шаблони спрощують процес створення об'єктів настільки, що повністю позбавляють розробника необхідності написання великого і складного програмного коду для отримання екземпляра об'єкта.
- Облік обмежень під час створення. Деякі шаблони дозволяють при створенні об'єктів накладати обмеження на їх тип чи кількість відповідно до встановлених вимог системи.

Патерни, що породжують:

1. Abstract Factory (Абстрактна Фабрика)
2. Builder (Будівельник)
3. Factory Method (Фабричний Метод)
4. Prototype (Прототип)
5. Singleton (Одиночка)

Використання структурних патернів допомагає з класів та об'єктів будувати більші структури.

Структурні патерни рівня класу використовують успадкування для складання композицій з інтерфейсів та реалізацій. Наприклад, у C++ - допустимо використання множинного успадкування реалізації для об'єднання кількох класів на один. У результаті виходить клас, що має властивості всіх своїх батьків. У C# - не допустимо множинне успадкування реалізації (class), але допустимо множинне успадкування інтерфейсів (interface), що дозволяє організовувати композиції інтерфейсів.

Структурні патерни рівня об'єкта використовують композицію об'єктів отримання нової функціональності. Додаткова гнучкість у разі пов'язані з можливістю змінити композицію об'єктів під час виконання, що неприпустимо для статичної композиції класів.

Багато структурні патерни тією чи іншою мірою пов'язані один з одним.

Структурні патерни:

1. Adapter (Адаптер)

2. Bridge (Міст)

3. Composite (Компонувальник)

4. Decorator (Декоратор)

5. Facade (Фасад)

Поведінкові шаблони (behavioral patterns) застосовуються передачі управління у системі. Існують методи організації управління, застосування яких дозволяє досягти значного підвищення як ефективності системи, так і зручності її експлуатації.

Поведінкові патерни:

1. Chain of Responsibility (Ланцюжок Обов'язків)

2. Command (Команда)

3. Interpreter (Інтерпретатор)

4. Iterator (Ітератор)

5. Mediator (Посередник)

6. Memento (Зберігач)

7. Observer (Спостерігач)

8. State (Стан)

9. Strategy (Стратегія)

10. Template Method (шаблонний метод)

11. Visitor (Відвідувач)

Загалом патерн складається з чотирьох основних елементів:

1. Ім'я. Пославшись на нього, ми можемо одразу описати проблему проектування, її вирішення та їх наслідки. Присвоєння патернам імен дозволяє проектувати більш рівні абстракції. За допомогою словника патернів можна обговорювати з колегами, згадувати патерни в документації, в тонкощах представляти дизайн системи. Знаходження хороших імен було одним із найважчих завдань при складанні каталогу.

2. Завдання. Опис того, коли слід застосовувати патерн. Необхідно сформулювати завдання та його контекст. Може описуватись конкретна проблема проектування.

3. Рішення. Опис елементів дизайну, відносин між ними, функції кожного елемента. Конкретний дизайн або реалізація немає на увазі, оскільки патерн - це шаблон, застосовний у різних ситуаціях. Просто дається абстрактний опис задачі проектування і того, як вона може бути вирішена за допомогою якогось дуже узагальненого поєднання елементів (у нашому випадку класів та об'єктів).

4. Результати - це наслідки застосування патерну та різного роду компроміси.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Проектування документообігу кафедри

В роботі побудовано інформаційну технологію [9-10] автоматичного документу обігу кафедри Одеського національного університету. На рисунку 2.1 показано процес розробки найважливішого документу кафедри для забезпечення навчального процесу – загального навантаження кафедри та індивідуального навантаження кожного викладача.

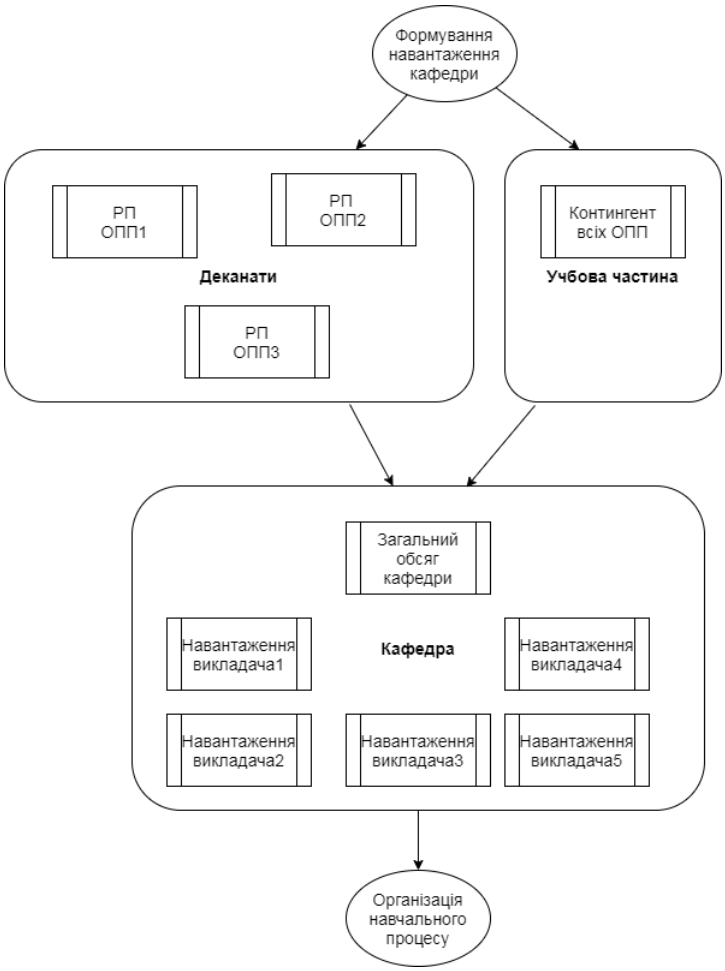

Рисунок 2.1. Документообіг кафедри ОНУ

Першим етапом формування документу навантаження кафедри є формування документу загального обсягу відповідно до документів деканату та учбової частини. Кількість заявок деканату динамічна і залежить від кількості освітніх програм, на яких кафедра повинна забезпечити навчальний

процес. Крім того, сама форма заявок може змінюватися, тому необхідно розробити інформаційну технологію з функціоналом навчання системи розрахунку обсягу навантаження кафедри.

Другим етапом формування документу навантаження кафедри є формування індивідуального навантаження кожного викладача кафедри. Для формування навантаження викладача повинно бути відомо яку частину якої дисципліни викладає конкретно який викладач. Тоді згідно закріплення викладачів за дисциплінами, може бути виконано формування документів індивідуального навантаження викладачів. Так як форма індивідуальних навантаження також може змінюватися, то інформаційна система повинна мати функціонал навчання системи розрахунку індивідуального навантаження викладача.

По закінченню семестрів та навчального року документ навантаження кафедри доповнюється листами звітів викладачів. Тому третім етапом формування документу є створення листів звіту за кожний семестр та за рік. Інформаційна система забезпечує заповнення листів інформацією згідно індивідуального навантаження викладача.

2.2 Інформаційне моделювання

Інформаційна система проводить автоматичний розрахунок навантаження однієї кафедри. Для розрахунку необхідно налаштувати систему, тобто задати:

- кафедру;
- каталог шаблонів;
- навчальний рік;
- шаблон обсягу;
- шаблон заявки.

Зрозуміло, що всі налаштування системи повинні бути єдині для всієї інформаційної системи, для всіх класів та модулів проекту. Для того, щоб

забезпечити єдність таких налаштування спроектовано клас налаштування з використанням паттерну Singleton. Цей шаблон проектування забезпечує наявність у системі лише одного екземпляра заданого класу, дозволяючи іншим класам отримувати доступ до цього екземпляра.

Припустимо, нам знадобився якийсь світовий об'єкт, тобто такий об'єкт, доступ до якого можна було б здійснити з будь-якої точки програми, але при цьому необхідно, щоб він створювався лише один раз. Іншими словами, до цього об'єкта повинні мати доступ всі елементи програми, але працювати вони повинні з одним і тим самим екземпляром. Прикладом такого об'єкта може бути хронологічний список, в якому зберігається інформація про всі дії користувача, які він робив, працюючи з програмою. Об'єкт за визначенням повинен бути доступним для всіх елементів програми, щоб вони могли або заносити в нього відомості про чергову операцію, виконану користувачем, або витягувати з нього дані про останню операцію для її скасування.

Один із можливих методів вирішення цього завдання полягає у створенні глобального об'єкта в головному модулі додатка з подальшою передачею посилання на цей об'єкт усім іншим об'єктам, яким це необхідно. Однак досить важко, приступаючи до розробки програми, правильно визначити спосіб передачі посилання, який би підходив всім об'єктів, так само як і заздалегідь передбачити, яким саме елементам програми знадобиться цей об'єкт. Ще одним недоліком цього рішення є неможливість перешкодити іншим об'єктам створювати додаткові екземпляри глобального об'єкта.

Існує й інший спосіб отримання глобальних значень, що базується на застосуванні статичних змінних. Це дозволяє застосуванню звертатися безпосередньо до кількох спеціальних статичних об'єктів, ув'язнених усередині деякого класу.

Але цей підхід також має низку недоліків.

- Статичний об'єкт — це не найкраще рішення, оскільки він створюється під час завантаження класу, що позбавляє розробника можливості передачі даних перед створенням екземпляра.
- Розробник не може контролювати доступ до статичного об'єкта, який оголошено загальнодоступним.
- Якщо розробник вирішить, що замість одного об'єкта йому знадобиться, скажімо, трійка таких об'єктів, йому доведеться заново переписати весь код клієнтської частини програми. У подібній ситуації дуже корисним стає шаблон Singleton, який забезпечує зручний доступ до всіх елементів додатка до глобального об'єкта.

Щоб забезпечити контроль над створенням екземплярів класу, досить зробити конструктор закритим чи захищеним. Однак тут виникає невелика проблема: як тепер створити хоча б один екземпляр, якщо ця операція недоступна? Ця проблема вирішується за допомогою надання спеціального статичного методу доступу (Instance). Даний метод створює єдиний екземпляр (якщо, звичайно, він вже не існує) і повертає посилання, що викликало його елементу, на створений об'єкт-одиначку. Це посилання, у свою чергу, зберігається як статичний закритий атрибут класу отриманого об'єкта для подальшого використання при викликах.

Хоча об'єкт-одиначку можна створити за допомогою спеціального методу доступу, все ж таки найчастіше він створюється під час завантаження класу. Відкладений виклик конструктора виправдано лише у випадках, коли перед створенням екземпляра необхідно виконати деякі ініціалізаційні операції налагодження.

На рисунку 2.2 показано діаграму побудови шаблону Singleton, згідно якого в проекті побудовано клас AtrIS.

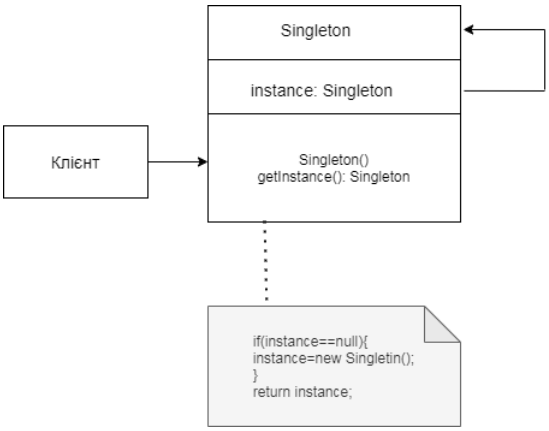

Рисунок 2.2. Діаграма шаблону Singleton

На рисунку 2.3 представлено діаграму класу AtrIS (див. Додаток А). Клас містить два поля oneAtr – екземпляр власного класу та filesZav – список рядків для збереження списку файлів заявок деканатів.

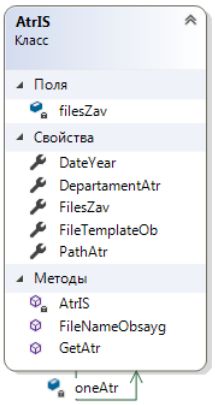

Рисунок 2.3. Діаграма власного класу AtrIS

Крім того, клас містить п’ять властивостей з анонімними полями:

- DateYear – навчальний рік розрахунку документів;
- PathAtr – шлях розташування всіх документів;
- DepartamentAtr – назва кафедри, для якої формуються документи;
- FileTemplateOb – файл шаблону формування документів.

Слід зазначити, що значення полів filesZav, DateYear, DepartamentAtr та FileTemplateOb зберігаються після завершення роботи застосування інформаційної системи. При наступній сесії роботи ІС ці поля приймають

значення згідно збережених даних. Для зміни цих полів існує відповідний функціонал застосування інформаційної системи.

Крім того, клас AtrIS містить три методи (див. Додаток А): AtrIS() – закритий конструктор класу для ініціалізації списку заявок деканату, GetAtr() – відкритий статичний метод для повернення екземпляру класу AtrIS згідно шаблону Singleton, FileNameObsayg() – метод який повертає назву документу навантаження кафедри.

2.3 Визначення користувачів системи

Інформаційна система призначена для завідувача, секретаря та провідного фахівця кафедри, які несуть відповідальність за документацію кафедри.

Для роботи інформаційної системи необхідно завантажити заявки деканату і учбової частини до папки з документами застосування Documents (рис. 2.4). Всі документи сформовані інформаційною системою також зберігаються в цій же папці.

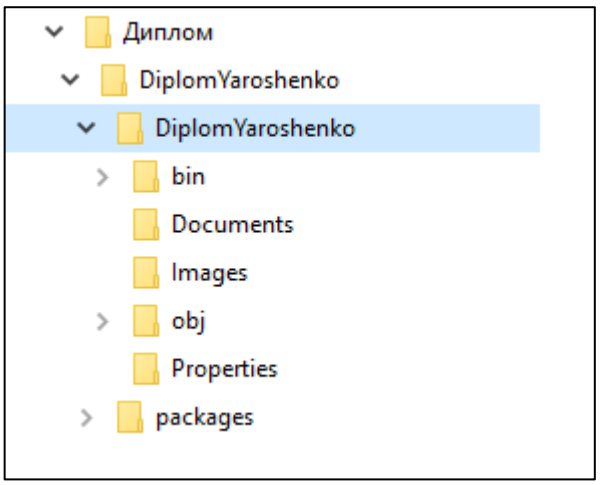

Рисунок 2.4. Файлова організація ІС

В задачі користувача системи входить навчання системи обміну інформації між шаблонами для розрахунку загального обсягу навантаження кафедри та індивідуального навантаження викладача кафедри. Сформовані

задачі зберігаються в файловому просторі інформаційної системи зі власним розширенням.

Для навчання системи першій задачі розрахунку загального обсягу треба сформувати три навчання: обмін даними для кожного семестру навчального року з заявок деканату та обмін даними з учбової частини про контингент здобувачів, кількість лабораторних та практичних груп. Під час формування процесу навчання відкриваються два документи: шаблон документу навантаження кафедри та шаблон заявки деканату. За рахунок обирання відповідних осередків для кожного документу формується алгоритм розрахунку обсягу кафедри. Навчання проводиться для кожного семестру окремо. В результаті в інформаційній системи зберігаються файли з алгоритмами обміну даними з іменами, які дає користувач цієї системи.

Структура документу наступна:

1. Назва файлу шаблону документа навантаження.
2. Назва файлу шаблону заявки деканату.
3. Номер осередку з назвою кафедри, для якої слід сформувати навантаження.
4. Координати осередку шаблону документа навантаження та координати осередку шаблону заявки деканату.

Слід підкреслити, що для кількість координат залежить тільки від користувача, який навчає системи алгоритму розв’язання задачі. Для зручності користувача в інформаційній системи організовано функціонал візуалізації алгоритмів обміну даних, які існують в системі. На рис. 2.5 представлено, як відбувається обмін даними між шаблонами для розрахунку обсягу кафедри за 1 семестр.

Також для розрахунку навантаження кафедри необхідно провести навчання обміну даних між документами Обсяг кафедри та Контингент студентів. В результаті навчання формується документ алгоритму з наступною структурою:

1. Назва файлу шаблону документа навантаження.
2. Назва файлу шаблону контингенту учбової частини.
3. Координати осередку шаблону документа навантаження та координати осередку шаблону контингенту учбової частини.

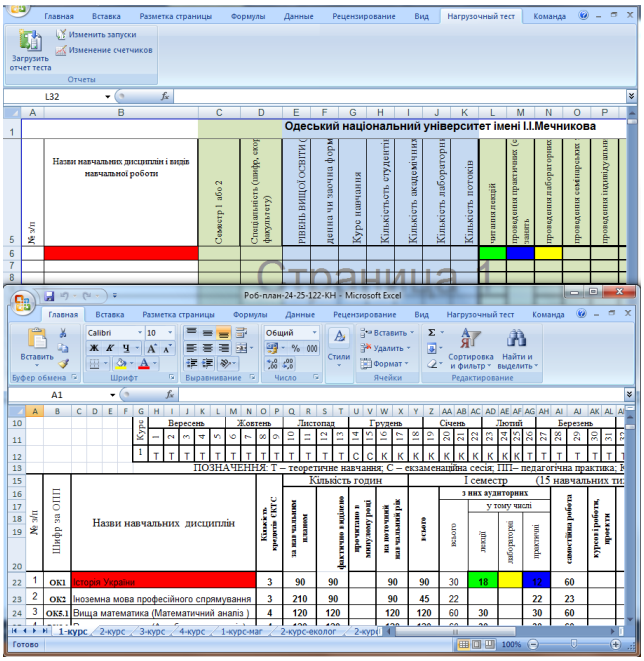

Рисунок 2.5. Візуалізація алгоритму обміну даними за 1 семестр.

На рис. 2.6 представлено, як відбувається обмін даними між шаблонами для розрахунку обсягу кафедри студентів денної форми навчання.

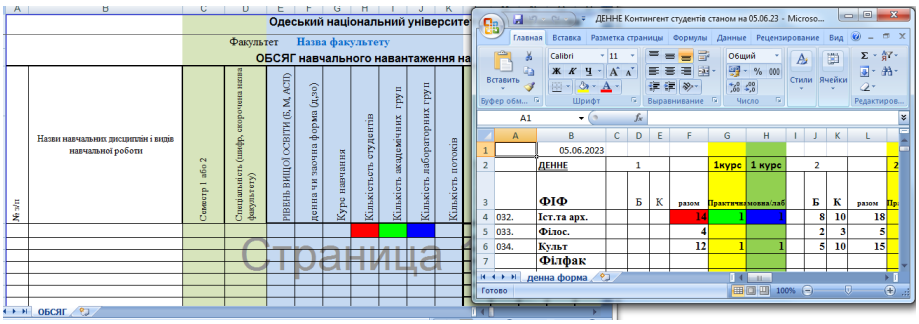

Рисунок 2.6. Візуалізація алгоритму обміну даними для денної форми навчання.

Для розв’язання другої задачі необхідно провести навчання системи для обміну даних між загальним обсягом навантаження та листом викладача. Цей етап можливий тільки після перевірки загального обсягу користувачем системи та передачі системі інформації про розподіл дисциплін між викладачами.

Для формування звіту користувач системи також повинен виконати навчання системи. Таким чином необхідно мати три блоки навчання системи для проведення розрахунку документу навантаження кафедри ОНУ у повному обсязі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Реалізація модулів системи

Інформаційна система складається з трьох модулів: модуль представлення, модуль взаємодії з com-сервером та модуль налаштувань системи, який має дві складові – модуль взаємодії документів даних між собою та модуль взаємодії алгоритмів навчання та даних (рис. 3.1).

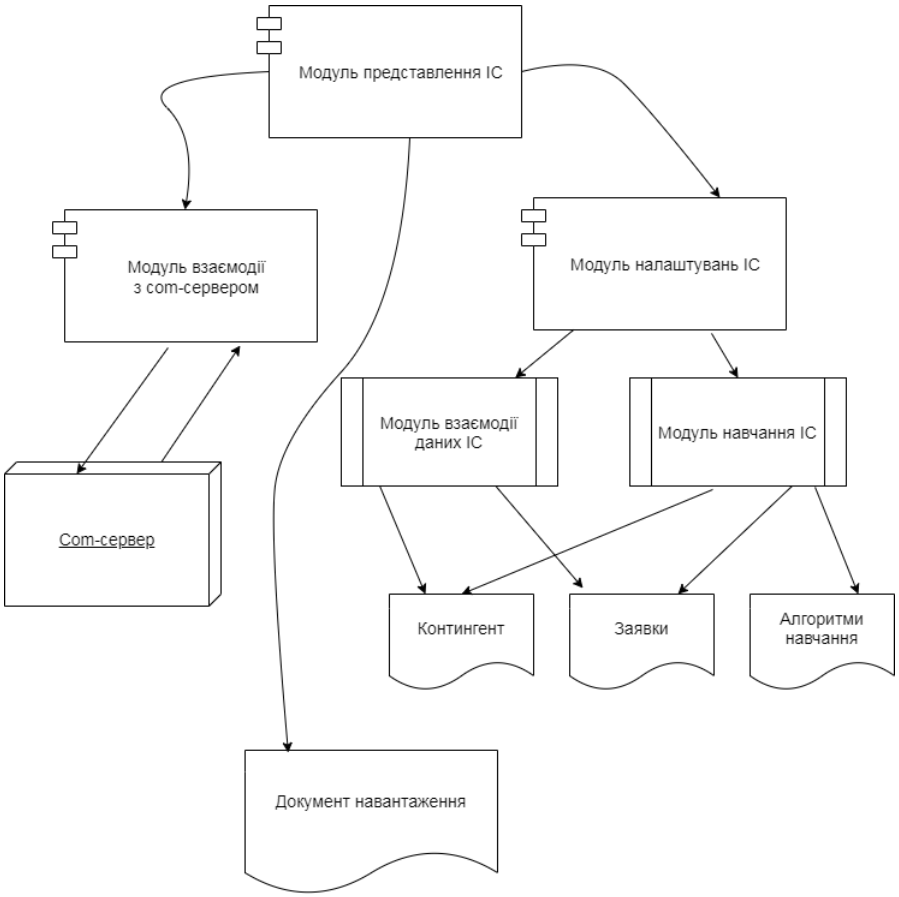

Рисунок 3.1. Схема модулів ІС

Головне вікно модуля представлення на етапі візуальної розробки ІС має вигляд, як показано на рисунку 3.2. В текстових вікнах буде відображатися інформація згідно налаштувань системи, які повинен виконати користувач ІС, що відповідає на кафедрі за формування документу навантаження.

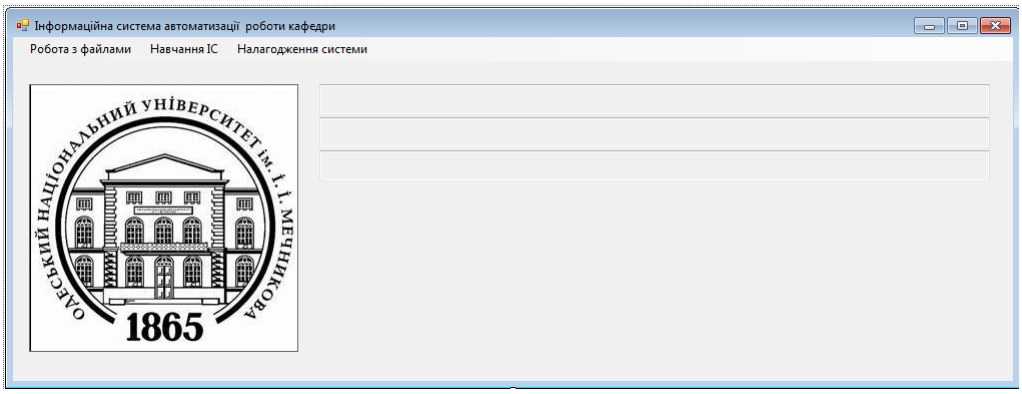

Рисунок 3.2. Головне вікно ІС

У верхній частині вікна знаходиться головне меню, яке представлене на рис. 3.3. Головне меню має три пункти: Робота з файлами, Навчання ІС та Налаштування системи.

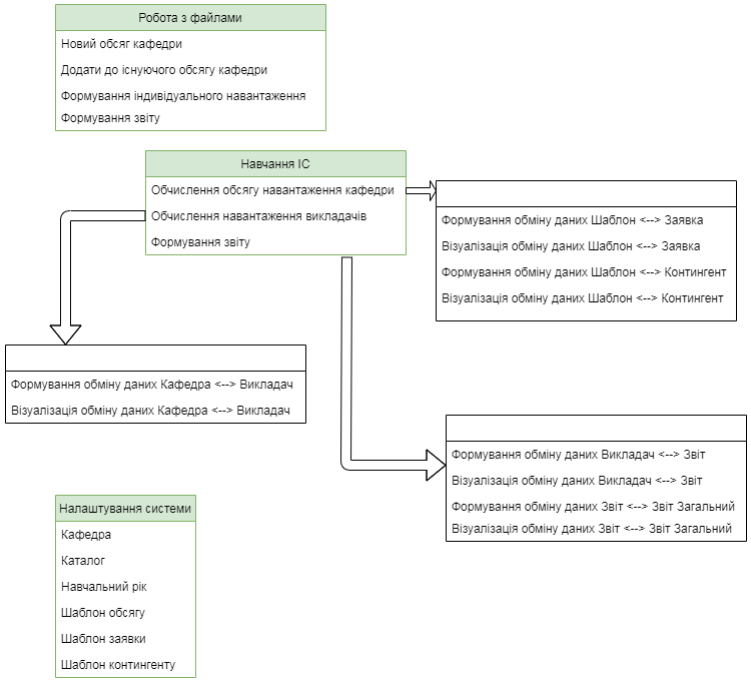

Рисунок 3.3. Головне меню ІС

Для роботи модуля налаштувань ІС використовується клас AtrIS (див. Додаток А), а саме в класі FMain (див. Додаток Б) використовується екземпляр myAtr типу AtrIS, як закрите поле. Ініціалізація екземпляру класу відбувається в конструктору класу FMain за допомогою статичного методу GetAtr() класу AtrIS, так як цей клас реалізує шаблон Singleton. Для

налаштування полів екземпляру myAtr в класі FMain розроблено власний закритий метод AttributIS(). Метод працює з потоком, що пов'язаний з файлом збереження інформації про налагодження системи. Саме збереження інформації відбувається у функції події FMain_FormClosed(), що виникає, коли закривається головне вікно ІС. В результаті роботи функції AttributIS() екземпляр myAtr інкапсулює в собі інформацію про навчальний рік розрахунку документів, шлях розташування всіх документів, назва кафедри, для якої формуються документи та файл шаблону формування документів.

Для проведення розрахунків документів необхідно провести навчання інформаційної системи, а саме задати алгоритми обміну даними:

- Шаблон -- Заявка;
- Шаблон – Контингент;
- Кафедра – Викладач;
- Викладач – Звіт;
- Звіт – Загальний звіт.

Обмін даних «Шаблон -- Заявка» означає, що користувач повинен поставити у відповідність координати комірок із шаблону навантаження загального обсягу кафедри та координат заявок деканату. Для формування алгоритму навчання використовується функція події класу FMain: формуванняАлгоритмуToolStripMenuItem_Click() (див. Додаток Б). В коді функції проводиться перевірка на наявність файлів-заявок в налаштуваннях системи. У випадку відсутності таких файлів користувачеві надається можливість обрати файли заявок за допомогою стандартного діалогового вікна відкриття файлу з можливістю вибору декількох файлів одночасно. Після успішного проходження навчання зберігається файл з алгоритмом обміну даними з іменем, яке задає користувач за допомогою стандартного діалогового вікна збереження файлів. Файл зберігається з розширенням «nav1».

Обмін даних «Шаблон – Контингент» означає, що користувач повинен поставити у відповідність координати комірок із шаблону навантаження загального обсягу кафедри та координат документу учбової частини про контингент студентів, кількість потоків, практичних та лабораторних груп. Для організації процесу використовується функція події класу FMain: формуванняОбміну ДанихШаблонКонтингентToolStripMenuItem_Click() (див. Додаток Б). Функція працює з двома екземплярами власного класу RunExcel (див. Додаток В). Після успішного проходження навчання зберігається файл з алгоритмом обміну даними з іменем, яке задає користувач за допомогою стандартного діалогового вікна збереження файлів. Файл зберігається з розширенням «nav2».

Обмін даних «Кафедра – Викладач» означає, що користувач повинен поставити у відповідність координати комірок з листа загального обсягу навантаження кафедри та листа викладача. Для проведення навчання використовується функція події класу FMain формуванняОбміну ДанихКафедраВикладачToolStripMenuItem_Click () (див. Додаток Б). В функції використовується механізм обробки виключень на обробку виключення нульового посилання.

Обмін даних «Викладач – Звіт» означає, що користувач повинен поставити у відповідність координати комірок з листа шаблону навантаження викладача та листа звіту.

Обмін даних «Звіт – Загальний звіт» означає, що користувач повинен поставити у відповідність координати комірок з листа звіту семестрового та звіту за навчальний рік.

Для організації першого навчання ІС розроблено власний клас AtrNav1 (див. Додаток Г), діаграму якого представлено на рис. 3.4. Клас містить два відкритих списку типу List<Point> для збереження обраних осередків документів обсягу та навантаження, для яких будується алгоритм обміну даними. Крім того, є ще властивості для назви документу обсягу, кафедри, номеру семестру та додаткової інформації для документуобігу.

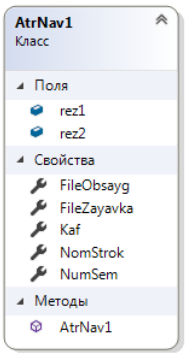

Рисунок 3.4. Діаграма класу AtrNav1

3.2 Реалізація роботи з com-сервером

Для роботи з com-сервером розроблено власний клас RunExcel (див. Додаток В), діаграму якого представлено на рис. 3.5. Клас має закрите поле Excel.Application excelapp. Тип Application інкапсулюється в бібліотеці Microsoft.Office.Interop.Excel. Для зручності в коді введено псевдонім для цього об'єкту:

```
using Excel = Microsoft.Office.Interop.Excel;
```

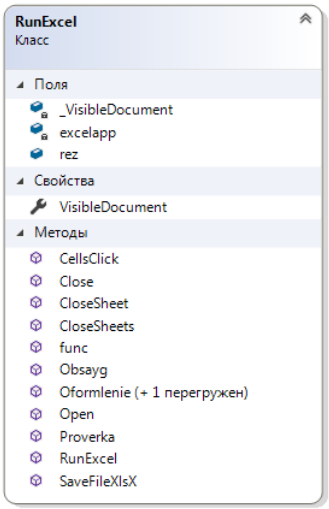

Рисунок 3.5. Діаграма класу RunExcel

Ініціалізація екземпляру класу Application excelapp відбувається безпосередньо підчас його оголошення за допомогою конструктору без вхідних параметрів (див. Додаток В).

Об'єкти, якими оперує сервер Excel, є кілька десятків. Сам сервер - об'єкт Application або програма Excel, може містити одну або більше книг, посилання на які містить властивість Workbooks. Сторінки - Worksheet, містять об'єкти осередку або групи осередків, посилання на які стають доступними через об'єкт Range.

Згідно діаграмі (рис. 3.5) клас RunExcel містить метод Open() з одним вхідним параметром – рядком-назвою файлу,який необхідно буде відкрити. Для відкриття існуючого документа основним методом є Open набору Excel.Workbooks. Для відкриття текстових файлів як робочих книг, баз даних, файлів у форматі .XML використовуються методи OpenText, OpenDatabase або OpenXml. Про використання методів OpenDatabase та OpenXml мова вестиметься в інших темах.

Метод Open має багато параметрів. Але більшість із них, як видно з прикладу, необов'язкові. Розглянемо параметри методу Open:

- FileName - ім'я файлу файлу, що відкривається
- UpdateLinks - спосіб оновлення посилань у файлі
- ReadOnly - при значенні true відкриття тільки для читання
- Format - визначення формату символу роздільника
- Password - пароль доступу до файлу до 15 символів
- WriteResPassword - пароль на збереження файлу
- IgnoreReadOnlyRecommended - при значенні true відключається висновок запиту працювати без внесення змін
- Origin - тип текстового файлу
- Delimiter - розділювач при Format = 6
- Editable - Використовується тільки для надбудов Excel 4.0
- Notify - при значенні true ім'я файлу додається в список нотифікації файлів
- Converter - використовується для передачі індексу конвертера файлу використовується для відкриття файлу
- AddToMRU - при true ім'я файлу додається до списку відкритих файлів

UpdateLinks - дозволяє встановити спосіб оновлення посилань у файлі. Якщо цей параметр не заданий, видається запит на вказівку методу оновлення. Значення: 0 – не оновлювати посилання; 1 – оновлювати зовнішні посилання; 2 - оновлювати лише видалені посилання; 3 – оновлювати всі посилання.

Format - під час роботи з текстовими файлами визначає символ роздільника для полів, які заносяться до різних осередків документа. Значення параметра: 1 – символ табуляції; 2 - кома; 3 - пробіл; 4 – точка з комою; 5 – немає роздільника; 6 – інший символ, визначений у параметрі Delimiter.

Метод Close() класу RunExcel закриває com-сервер. Метод CloseSheet(int i) закриває i-документ. Метод CloseSheets() закриває всі документи з якими працює com-сервер.

Метод Close() com-серверу має наступні вхідні параметри:

- SaveChanges - якщо у книзі немає змін у документі, то параметр ігнорується. Інакше, якщо є зміни, але є посилання на книгу, що закривається. В інших відкритих вікнах - цей параметр також ігнорується. При відсутності посилань та наявності змін - цей параметр визначає, чи повинні бути збережені зміни. При true та певному параметрі Filename – зміни зберігаються, інакше запитується ім'я файлу. При false збереження не відбувається . Якщо Type.Missing – викликається діалогове вікно Save As.
- Filename - це ім'я файлу
- RouteWorkbook - якщо файл не повинен бути перенаправлений іншому одержувачу. Цей параметр ігнорується. Інакше при true файл спрямовується наступному одержувачу. При false пересилання немає.

В класі RunExcel існує дві перевантажені функції Oformlenie().У мові C# передбачено можливість створення кількох функцій з однаковим ім'ям. Це називається перевантаженням функції. Перевантажені функції повинні відрізнятися один від одного списками параметрів: або типом одного або

декількох параметрів, або різною кількістю параметрів, або тим і іншим одночасно.

Перша функція Oformlenie() приймає три вхідних параметра: індекси комірки та третій вхідний параметр – колір, яким треба залити осередок. Для виконання заливки необхідно виділити осередок у об’єкт типу Interior та встановити його властивість ColorIndex.

Друга функція Oformlenie() приймає два вхідних. Ці параметри – це екземпляри власних класів AtrIS (рис. 2.3) та AtrNav1 (рис. 3.4). Ця функція знаходить осередки, які зберігаються в списках екземпляру класу AtrNav1, що були сформовані при навчанні, у двох документах: заява – обсяг та відповідні осередки заливає одним кольором.

Щоб намалювати табличку в Excel, треба навчитися малювати рамки навколо обраного осередку або об'єднаної групи осередків.

Кроки малювання рамки будуть наступні:

- Вибрати осередок або групу осередків на аркуші документа.
- Об'єднати комірки.
- Визначаємо колір ліній обведення. Колір може бути обраний як один з 56 кольорів палітри кольорів Excel і, тому, він задається через колірний індекс (наприклад, excelcells.Borders.ColorIndex=3; - червоний)
- Деякі значення ColorIndex 1 – білий, 2 – чорний, 3 – червоний, 4 – зелений, 6 – жовтий, 41 – синій і т.д.
- Вибрати стиль лінії (Excel.XlLineStyle.xlContinuous). Стиль лінії може бути одним з наступних: xlContinuous, xlDash, xlDashDot, xlDashDotot, xlDot, xlDouble, xlSlantDashDot, xlLineStyleNone.
- Встановити товщину лінії (Excel.XlBorderWeight.lHairline). Товщина лінії може бути однією з наступних: lHairline, xlMedium, xlThick, xlThin.

Можна малювати лінії по будь-якій межі комірки і не по межі комірки, для чого необхідно задати розташування лінії - замість excelcells.Borders задати excelcells.Borders[напрям], де напрямок може бути одним з наступних:

Excel.XlBordersIndex.xlDiagonalDown, Excel.XlBordersIndex. xlDiagonalxlDiagonalUp, Excel.XlBordersIndex.xlDiagonalUp, Excel.XlBordersIndex.xlEdgeBottom, Excel.XlBordersIndex.xlEdgeLeft, Excel.XlBordersIndex.xlEdgeRight, Excel.XlBordersIndex.xlEdgeRight, Excel.XlBordersIndex. .XlBordersIndex.xlInsideVertical.

Згідно рис. 3.5 клас RunExcel містить метод Proverka() з двома вхідними параметрами типу AtrIS та AtrNav1. Цей метод забезпечує перевірку документу на наявність в документі навантаження рядків, що не відповідають формованого навантаження. Якщо такі рядки знайдено то їх видаляють з документу за допомогою методу Delete()com-серверу.

Крім того, клас RunExcel забезпечую формування листів документу навантаження кафедри за допомого методів com-серверу. Функція Obsayg() приймає вхідні параметри для налаштувань системи та екземпляри класів з алгоритмами навчання обміну інформацією між документами системи.

Метод Obsayg() працює одночасно з декількома ми документами com-серверу: шаблон обсягу кафедри, заявки деканату та контингент студентів. В методі проводить робота з комірками документів. З документів заявки деканату та контингентів читається інформація з комірок, а в документ обсягу кафедри записується інформація в комірки. Все відбувається згідно алгоритмам навчання ІС.

Виведення інформації може бути виконане на конкретний аркуш робочої книги Excel у конкретну комірку, тому нам необхідно визначити об'єкти аркуш, в колекції аркушів та комірку. Визначення для осередків і осередку як самостійні об’єкти в C# відсутні, а є поняття області виділених осередків, яка може включати одну або більше осередків, з якими можна виконувати дії. Для виділення використовується метод get_Range, який дозволяє виділити групу осередків через завдання кутових осередків діапазону, і є можливість звернутися безпосередньо до властивостей Rows і Cells – у будь-якому випадку виділеним буде діапазон осередків. Ще один спосіб визначення вибраних осередків - використання методу get_Offset(x,y)

об'єкта Range, що повертає об'єкт Range, що віддаляється від заданої комірки на задану кількість рядків і стовпців, рахуючи від верхнього лівого кута. Це дозволяє працювати з осередками, позиція яких задані щодо обраного осередку чи групи осередків. Можливості об'єднання осередків набагато багатші - можна використовувати об'єднання областей, групу областей, визначення меж виділених осередків і т.п., проте, для більшості практичних завдань цілком перерахованих вище методів. Виділені осередки далі можуть бути поєднані і з ними дії можуть виконуватися як з одним осередком. Для цього використовується метод Merge().

Слід також зазначити ще одну особливість групи властивостей Excel.Application. Як тільки ми отримали посилання на конкретний об'єкт (книгу, аркуш, групу осередків, діаграму), їх можна зробити активними, застосувавши до них метод Activate(). З цієї миті вони доступні через відповідні властивості ActiveWorkbook, ActiveSheet, ActiveChart, ActiveCell. Крім того, при виконанні дій з конкретним об'єктом він автоматично стає активним і дії з ним можуть далі виконуватися з використанням перерахованих властивостей.

Так само можна використовувати і властивість Excel.Application Selection. Тобто, об'єкт може бути визначений як селектований і далі для посилання на об'єкт використовуватися властивість Selection. Хоча це й привабливо, щоб працювати з поточним селективним вибором як засобом зміни властивостей та поведінки об'єкта, але краще все-таки уникати цього. Причина - вибір може бути легко змінений і під час виконання програми, наприклад, кліком мишки в межах документа, що при великому обсязі виведення на 100% призведе до помилки.

Зауважимо, що C# потрібно зчитувати і присвоювати значення властивості Value2, а чи не Value об'єкта Range, оскільки властивість Value містить параметр. C# не підтримує властивості параметрів (за винятком індексних властивостей).

Читання інформації з осередків Excel багато в чому аналогічне виводу. На вибраному аркуші необхідно у вибраній книзі вибрати одну осередок або об'єднану групу осередків (метод get_Range або перетворення до Excel.Range) - після чого достатньо перетворити значення виділених осередків до потрібного типу даних.

Також клас RunExcel (див. рис. 3.5) містить метод SaveFileXlsX() для збереження документу. Документи Excel можна зберегти програмно та звичайним для Excel способом. У будь-якому разі перед виходом із Excel необхідно викликати метод Quit. Якщо властивість Excel.Application DisplayAlerts має значення true, Excel запропонує зберегти незбережені дані, якщо після старту в документ було внесено зміни. Excel автоматично не повертає цю властивість у значення за замовченням, тому його рекомендується повертати у вихідний стан. Зазначимо, що окрім Item, у набору Workbooks, як і у всіх наборів у Microsoft Office, є властивість Count, яка повертає кількість елементів у наборі.

На деяких варіаціях версій Windows і Office запит на збереження може бути присутнім, хоча ми, і відключаємо його у властивості Saved.

Наступне питання - у якому форматі зберігати документ. Для отримання формату документа, що відкривається, і завдання формату зберігається служить властивість Excel.Application DefaultSaveFormat. Властивість має багато значень типу XlFileFormat (які можуть бути легко подивитися в діалоговому вікні "Збереження документа" у полі "Тип файлу", відкривши Excel і вибравши пункт меню "Файл" | "Зберегти як").

Для збереження документів можна використовувати методи Excel.Workbook Save та SaveAs. Метод Save зберігає робочу книгу в папці "Мої документи" з іменами, які присвоюються документу за замовчуванням ("Книга1.xls", "Книга2.xls" ...) або в директорію та з ім'ям під яким документ уже був збережений.

При значенні властивості DisplayAlerts=true Excel запитуватиме - чи записати документ, що зберігається поверх існуючого, при значенні false - ні.

Метод SaveAs дозволяє зберегти документ із зазначенням імені, формату файлу, пароля, режим доступу і т. д. Цей метод, як і метод Save, надає властивості Saved значення true. Метод SaveAs має наступні параметри:

- Filename - ім'я файлу, що зберігається
- FileFormat - формат файлу, що зберігається
- Password - пароль доступу до файлу до 15 символів
- WriteResPassword - пароль на доступ до запису
- ReadOnlyRecommended - при true режим тільки для читання
- CreateBackup - створити резервну копію файлу при true
- AccessMode - режим доступу до робочої книги
- ConflictResolution - спосіб вирішення конфліктів
- AddToMru - при true збережений документ додається до списку раніше відкритих файлів
- TextCodePage - кодова сторінка
- TextVisualLayout – напрямок розміщення тексту
- Local - ідентифікатор ExcelApplication

Для доступу до книги використовуються значення AccessMode xlShared – загальна робоча книга, xlExclusive – монопольний доступ або xlNoChange – заборона зміни режиму доступу.

Параметр ConflictResolution - спосіб вирішення конфліктів при одночасному внесенні декількома користувачами змін в один документ - може мати значення: xlUserResolution - відображення діалогового вікна вирішення конфліктів (параметр за замовчуванням), xlLocalSessionChanges - прийняття змін, внесених користувачем або ChlesOthers.

Для збереження документа можна використовувати метод SaveCopyAs, який зберігає копію робочої книги у файлі.

Крім того, для внутрішньої реалізації функціоналу інформаційної системи в класі RunExcel додано метод VisibleDocument(), що робить

відкритий документ невидимим для користувачів під час обчислення навантаження кафедри.

3.3 Формування документу навантаження кафедри

Головне вікно додатку має вигляд, що представлено на рис. 3.5. За замовченням система налаштована для розрахунку навантаження кафедри механіки, автоматизації та інформаційних технологій за 2024-2025 навчальний рік, скорочена назва кафедри для пошуку в заявках деканату «МАІТ». Поля з інформацією за замовченням недоступні для редагування.

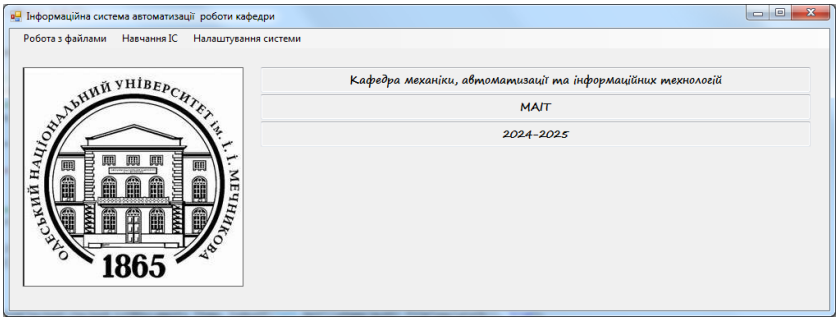

Рисунок 3.5. Головне вікно ІС

У разі необхідності треба провести налаштування ІС за допомогою відповідних пунктів меню (рис. 3.3). На рис. 3.6 представлено діаграму викликів для налаштування назви, абревіатури кафедри та навчального року.

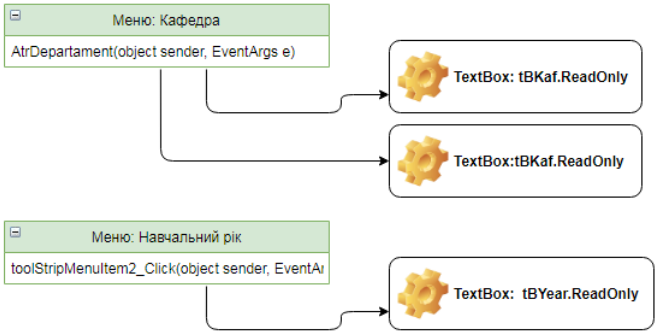

Рисунок 3.6. Діаграма викликів налаштування назви кафедри та навчального року

Для налаштування Каталогу ІС (див. рис. 3.7) використовується стандартне діалогове вікно FolderBrowserDialog. Цей клас надає спосіб видачі запрошення користувача для перегляду, створення та, зрештою, для вибору папки. Цей клас використовується, коли потрібно дозволити користувачеві вибирати тільки папки, але не файли. Огляд папок здійснюється за допомогою дерева. Можуть вибиратися лише папки з файлової системи; Вибір віртуальних папок неможливий. Об'єкт FolderBrowserDialog є модальним діалоговим вікном.

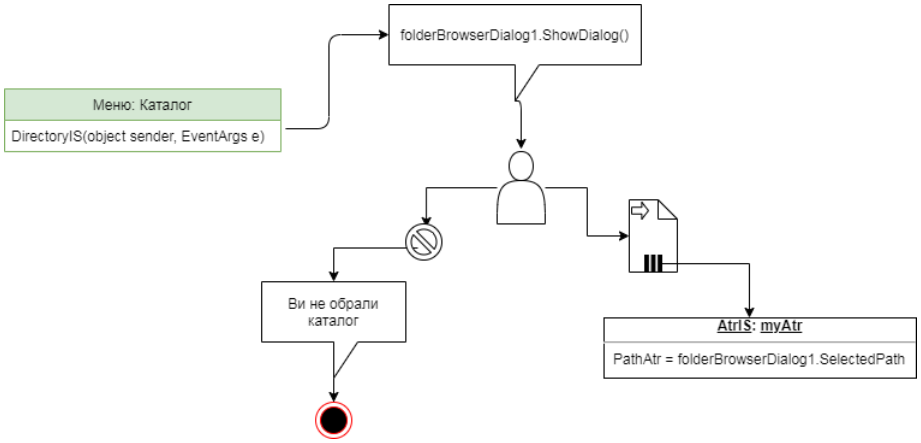

Рисунок 3.7. Діаграма викликів налаштування каталогу ІС

Завантаження шаблону обсягу відбувається за допомогою однойменного пункту меню (див. рис. 3.3). На рис. 3.8 представлено діаграму викликів. Функціонал організовано за допомогою діалогового вікна OpenFileDialog. Клас OpenFileDialog є похідними від базового абстрактного класу FileDialog, який додає загальні файлові характеристики для діалогових вікон під час відкриття та збереження файлів.

Клас FileDialog відображає діалогове вікно, за допомогою якого користувач може вибрати файл. FileDialog – це абстрактний клас, який містить поширену поведінку для класу OpenFileDialog. Цей клас не призначений для прямого використання, проте містить поширену поведінку для згаданих двох класів. Неможливо створити екземпляр класу FileDialog. Для створення діалогового вікна, призначеного для вибору файлу,

використовується OpenFileDialog. FileDialog є модальним діалоговим вікном. Тому, якщо це вікно відображається, частина програми, що залишилася, блокується до тих пір, поки користувач не вибере файл.

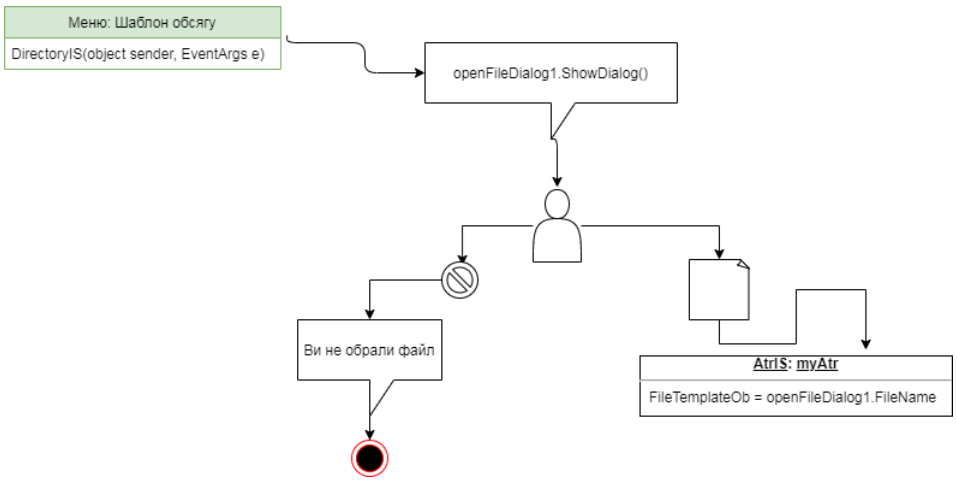

Рисунок 3.8. Діаграма викликів завантаження шаблону обсягу

Для того, щоб діалогове вікно відкривалося з папки документів застосування Documents (див. рис. 2.4), необхідно провести налаштування властивості InitialDirectory. Для налаштування використовується властивість PathAtr екземпляру myAtr, яка задається в функції AttributIS(). Виклик функції відбувається в конструкторі класу FMain.

Аналогічно організована робота ІС для викликів функціоналів Шаблон заявки та Шаблон контингенту. У першому випадку встановлюється властивість FilesZav екземпляру myAtr, а у другому – FileTemplateContingent того ж сама екземпляру.

Для проведення формування документу навантаження кафедри необхідно провести навчання ІС. На рис. 3.9 представлено діаграму викликів організації обміну даними між шаблоном обсягом та заявками деканату. Видно, що відбувається безпосередня робота з com-сервером Excel.

Після формування алгоритму необхідно алгоритм зберегти до файлу за допомогою класу StreamWriter. Об'єкти класу StreamWriter використовуються для запису у файл символів і рядків.

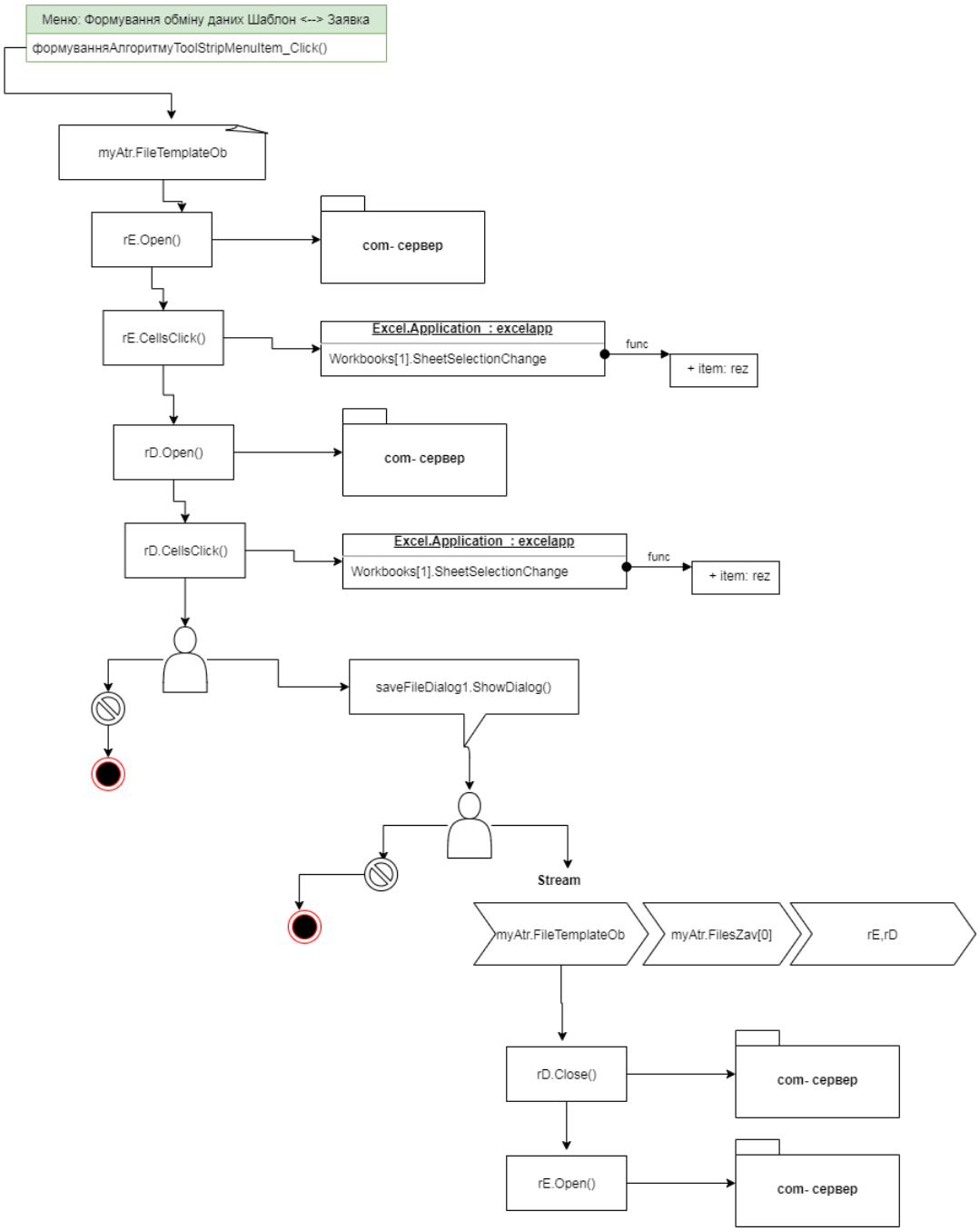

Рисунок 3.9. Діаграма викликів алгоритму навчання

Конструктор класу StreamWriter приймає як параметри ім'я файлу та логічне значення, яке вказує на те, чи слід додавати інформацію до вже існуючого файлу або необхідно створити новий. Якщо це значення дорівнює false, то або відбувається створення нового файлу, або з існуючого файлу спочатку видаляється інформація, що зберігається в ньому, а потім він відкривається на запис. Якщо ж це значення дорівнює true, то файл просто

відкривається, а інформація, що зберігається в ньому, зберігається. Якщо такого файлу немає, створюється новий файл.

Під час ініціалізації екземпляру класу StreamWriter створюється вихідний потік між програмою та файлом. Об'єкти класу StreamWriter мають методи WriteLine() та Write() для запису інформації до файлів.

Згідно діаграмі рис. 3.9 використовується створення власних обробників події клацання на комірках документу com-серверу.

Події також необхідні для створення функціональної програми для роботи з Excel як і доступ до властивостей та методів. Якщо розглядати використання властивостей та методів, що надаються інтерфейсом COM об'єкта, як прямий зв'язок з керування сервером, то події забезпечують зворотний зв'язок. Завдяки наявності подій, програма може забезпечити програмну функціональність для відгуку на те, що відбувається в програмі.

Об'єкти Excel, як і будь-який контрол C#, мають свої події і програміст, як і контролю, може створити обробник будь-якої з подій. Однак, у нас немає візуального компонента Excel і не можна подвійним клацанням мишки у віконці навпроти події у вікні Properties на вкладці Events створити обробник.

Розглянемо основні події об'єктів Excel. Події об'єкта Application:

а.) пов'язані з поведінкою об'єктів на аркуші

- SheetActivate - відбулася активізація аркуша;
- SheetBeforeDoubleClick - виконаний подвійний клік на аркуші і після цієї події очікується якась реакція сервера - виконання оброблювача за замовчуванням;
- SheetBeforeRightClick - виконаний правий клік на аркуші і після цієї події очікується якась реакція сервера - виконання оброблювача за замовчуванням;
- SheetCalculate - виконано перерахунок формул на аркуші;
- SheetChange - зміна обраного осередку на аркуші;
- SheetDeactivate - аркуш втратив фокус;

- SheetFollowHyperlink - користувач пішов за гіперпосиланням;
- SheetSelectionChange - змінилося виділення на аркуші.

b.) пов'язані з поведінкою вікна:

- WindowActivate - відбулася активізація вікна, якщо Excel на даний момент був активний;
- WindowDeactivate - вікно втратило фокус;
- WindowResize - змінився розмір вікна;

c.) пов'язані з керуванням робочою книгою:

- NewWorkbook - створена нова робоча книга;
- WorkbookActivate - книга, один із її листів, отримали фокус;
- WorkbookAddinInstall - виконується інсталяція не встановленого компонента;
- WorkbookAddinUninstall - виконується деінсталяція встановленого компонента;
- WorkbookBeforeClose - після цієї події очікується закриття книги – виконання оброблювача за замовчуванням;
- WorkbookBeforePrint - після цієї події очікується друк аркуша – виконання оброблювача за замовчуванням;
- WorkbookBeforeSave - після цієї події очікується збереження книги – виконання оброблювача за замовчуванням;
- WorkbookDeactivate - книга втратила фокус;
- WorkbookNewSheet - до книги доданий лист;
- WorkbookOpen - відкрито робочу книгу.

Події об'єкта Workbook:

a.) всі події об'єкта Application для пункту a, пов'язані з поведінкою об'єктів на аркуші. Генеруються лише для аркушів даної робочої книжки.

b.) всі події об'єкта Application для b. Генеруються лише для аркушів даної робочої книжки.

c.) пов'язані з керуванням робочою книгою:

- Activate - книга, один із її аркушів, отримали фокус;
- BeforeClose - після цієї події очікується закриття книги – виконання оброблювача за замовчуванням;
- BeforePrint - після цієї події очікується друк аркуша – виконання оброблювача за замовчуванням;
- BeforeSave - після цієї події очікується збереження книги - виконання оброблювача за замовчуванням;
- Deactivate - книга втратила фокус;
- NewSheet - до книги доданий лист;
- Open - відкрито робочу книгу.

Події об'єкта Worksheet:

а) пов'язані з поведінкою об'єктів на аркуші

- Activate - відбулася активізація аркуша;
- BeforeDoubleClick - виконаний подвійний клік на аркуші і після цієї події очікується якась реакція сервера - виконання оброблювача за замовчуванням;
- BeforeRightClick - виконаний правий клік на аркуші і після цієї події очікується якась реакція сервера - виконання оброблювача за замовчуванням;
- Calculate - виконано перерахунок формул на аркуші;
- Change - зміна обраного осередку на аркуші;
- SheetDeactivate - аркуш втратив фокус;
- SheetFollowHyperlink - користувач пішов за гіперпосиланням;
- SheetSelectionChange - змінилося виділення на аркуші.

Аналогічно організовано виклик функцій для функціоналу навчання задач Шаблон – Контингент, Кафедра – Викладач, Викладач – Звіт, Звіт – Загальний звіт. Відрізняється тільки тим, що в останніх трьох задачах робота відбувається з одним документом, але з різними сторінками.

На рисунку 3.10 представлено діаграму функціоналу візуалізації алгоритму навчання обміну даних Шаблон-Заявка.

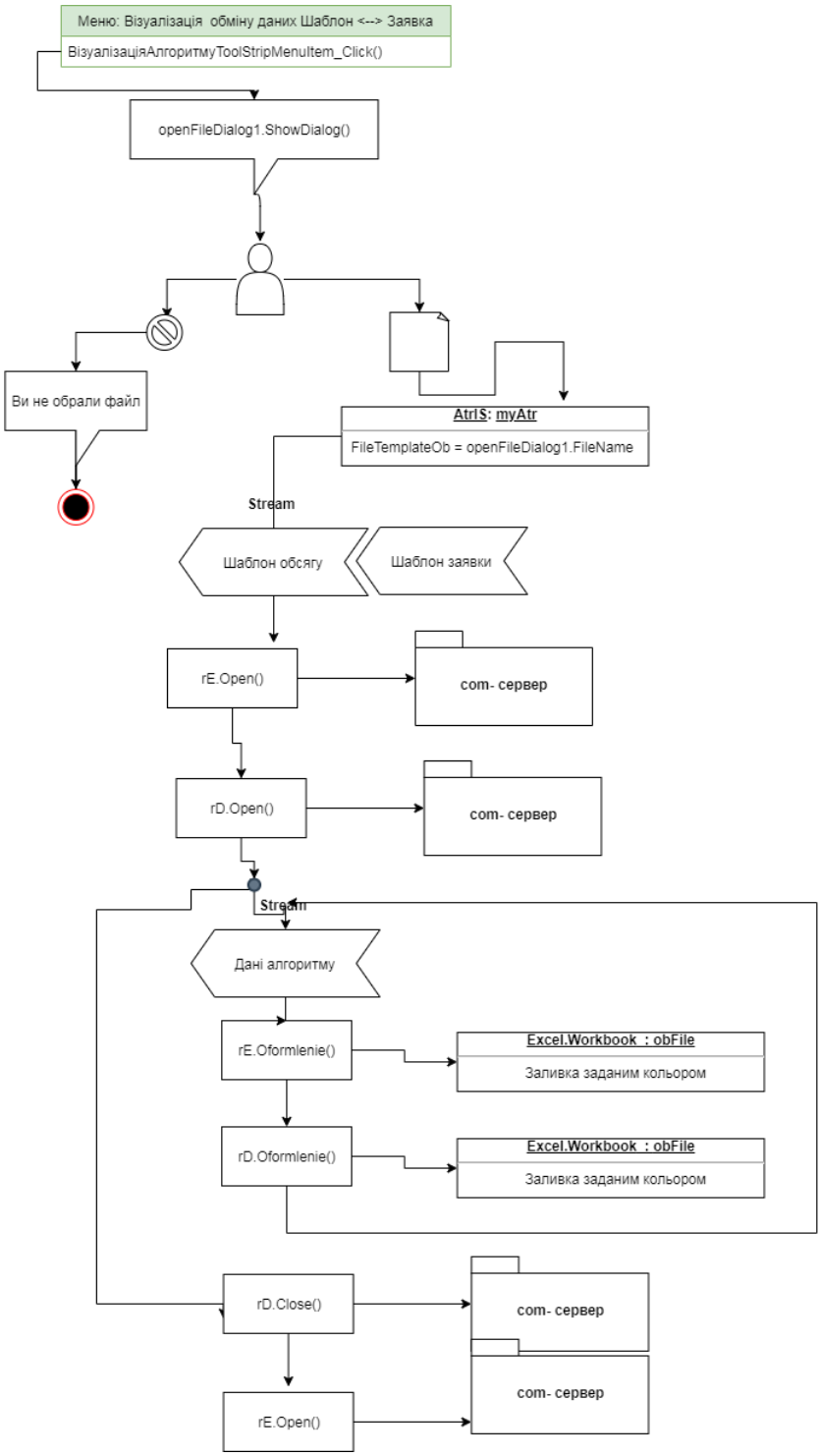

Рисунок 3.10. Діаграма функціоналу візуалізації алгоритму навчання

Слід зазначити, що для організації цього функціоналу використовувалося ціле значення кольору. Як правило, використовується

колірний простір RGB (Red, Green, Blue) складається з усіх можливих кольорів, які можна одержати шляхом змішування червоного, зеленого та синього. Ця колірна модель популярна у фотографії, телебаченні та комп'ютерній графіці. Значення RGB задаються як ціле число від 0 до 255. Наприклад, rgb(255,0,0) відображається червоним кольором. Для червоного параметра встановлено максимальне значення (255), а решта – 0. На жаль, такий підхід не працює в com- сервері Excel, треба використовувати властивість ColorIndex (Excel Graph). Повертає або задає колір кордону, шрифту або внутрішньої області, як показано на рис. 3.11. Колір вказується у вигляді значення індексу в поточній палітрі кольору або у вигляді однієї з наступних констант XlColorIndex: xlColorIndexAutomatic або xlColorIndexNone. Використовуються кольори, починаючи з цілого значення 3.

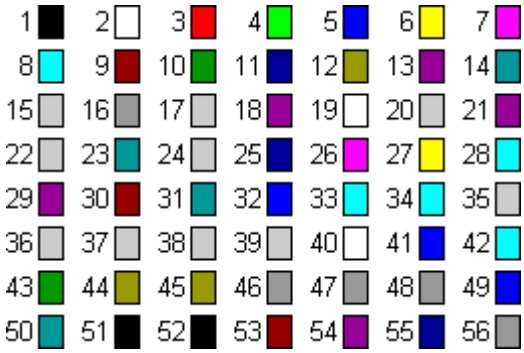

Рисунок 3.11. Палітра кольорів для властивості ColorIndex.

Має сенс ще розглянути функціонал розрахунку загального обсягу кафедри. На рис. 3.12 представлено діаграму викликів функцій. При виконанні цього функціоналу викликаються методи класу RunExcel, що взаємодіють з com-сервером Excel. По-перше, це розрахунок обсягу згідно обраним алгоритмам обміну даними, по-друге, це перевірка отриманого результату та оформлення документу згідно шаблону обсягу.

Аналогічно проводяться розрахунки індивідуального навантаження викладача, звіту за семестр та за навчальний рік.

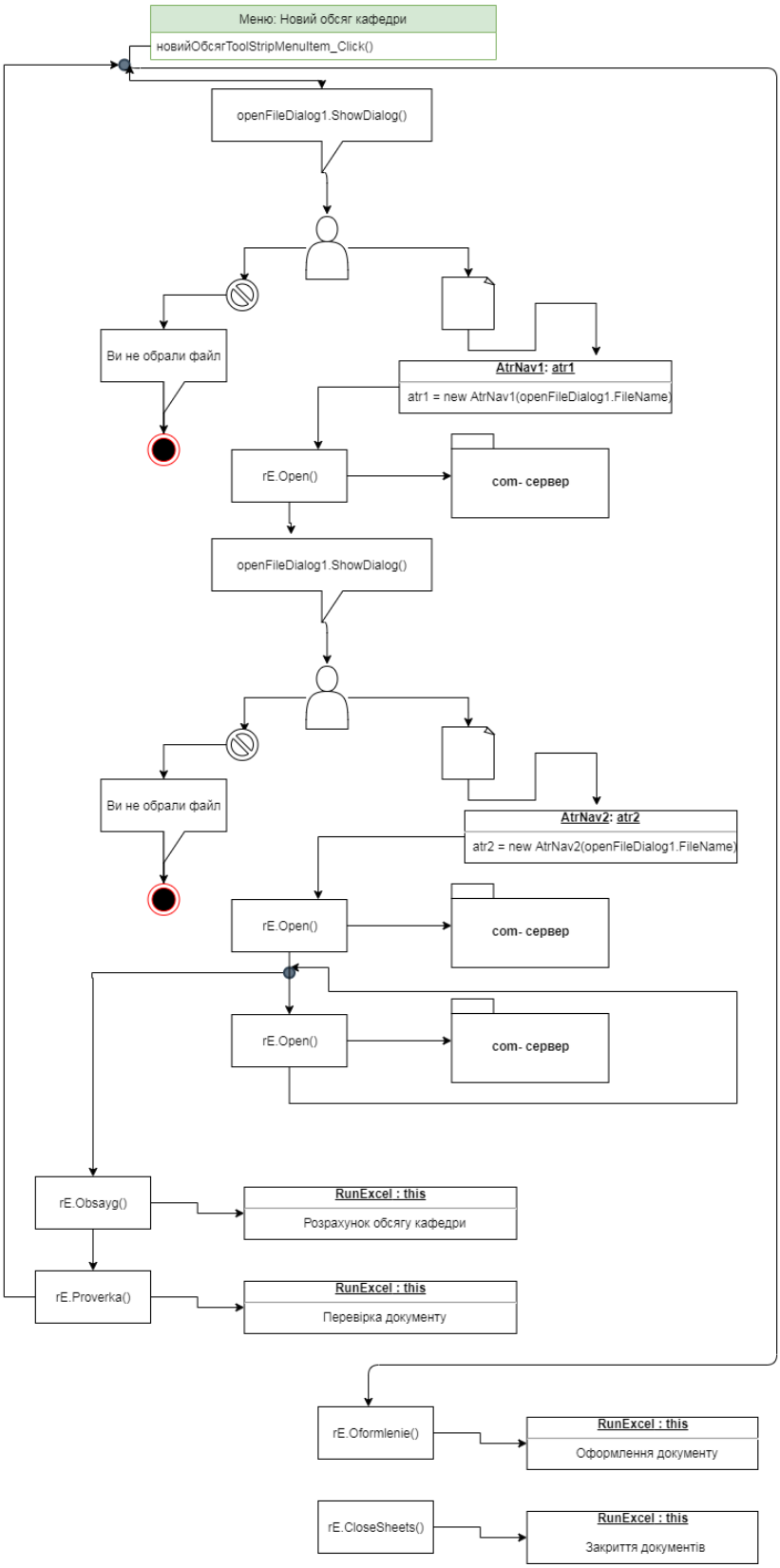

Рисунок 3.12. Діаграма викликів розрахунку навантаження кафедри

ВИСНОВОК

В кваліфікаційній роботі спроектовано інформаційну систему для формування документу навантаження кафедри з використанням компонентно-орієнтованого програмування.

Для досягнення поставленою мети розв’язано наступні задачі:

1. Проаналізовано документообіг кафедри ОНУ.
2. Спроектована інформаційна система формування документу навантаження з урахуванням всіх етапів: формування загального обсягу, індивідуального навантаження викладача, звіту викладачів за семестр та навчальний рік.
3. Розроблено функціонал для автоматичного проведення формування документу навантаження кафедри.

Документ навантаження кафедри обрано, як найважливіший документ для організації навчального процесу. Формування такого документа в ручному форматі потребує багато часу та крім того може приводити до помилок в розрахунках.

Аналіз документообігу ОНУ показав, що на жаль, самі вхідні документи для формування документу навантаження кафедри можуть змінюватися, тому в системі розроблені алгоритми навчання обміну інформацією між документами. Навчання проводить людина, тому щоб уникнути помилок під час навчання розроблений функціонал візуалізації алгоритмів обміну даними для зручної перевірки.

В перспективі до розробленої інформаційної системи треба додати використання com-серверу Word.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Статус Одеського національного університету імені І.І.Мечникова https://onu.edu.ua/pub/bank/userfiles/files/documents/statut-onu-2017.pdf.
2. Greg Harvey Excel 2019 All-in-One For Dummies / For Dummies, 2018. – 816 p.
3. John Walkenbach Excel VBA Programming For Dummies / Wiley, 2013. – 408 p.
4. Michael Alexander (Autor), Richard Kusleika (Autor), John Walkenbach Excel 2019 Bible (Bible (Wiley)) / Wiley John + Sons, 2018. – 1120 p.
5. Randy Abernethy COM/DCOM Unleashed (Unleashed Series) / Sams, 1999 – 700 p.
6. Оберг Роберт Дж. Технология COM + Основы и программирование = Understanding and Programming COM+: A Practical Guide to Windows 2000 First Edition / М.: Вильямс, 2000. – С. 480.
7. Дейл Роджерсон Основы COM. / Forbidden reality, 2-е изд., 2000. – 400 p.
8. Ія Григоришин, Сергій Косяченко, Людмила Кулібаба Створення програмних додатків в середовищі MS Office Visual Basic for Applications / КНТ, Дакор, 2007. – 208 с.
9. Peter Bernus, Handbook on Architectures of Information Systems / Peter Bernus, Kai Mertins, Gunter Schmidt. // Berlin: Springer, 2016. - 886 p.
10. Авраменко В.С. Проектування інформаційних систем / Авраменко В.С., Авраменко А.С. // Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. - 434 с.
11. Daniel Solis. Illustrated C# 2012 / Berkeley: APress, 2012. - 732 p.
12. Ярошенко Д.Є., Рачинська А.Л. Інформаційна система автоматизації роботи кафедри // Інтелектуальні інформаційні системи: Всеукраїнська науково- практична конференція молодих вчених, аспірантів і студентів: ЧНУ ім. Петра Могили. Миколаїв: Вид-во ЧНУ ім. Петра Могили, 2024. – С. 105-107.

ДОДАТОК А

Інтерфейсний модуль

```
public class AtrIS
    {
        static AtrIS oneAtr;
        public string DateYear { get; set; }
        public string PathAtr { get; set; }
        public string DepartamentAtr { get; set; }
        public string FileTemplateOb { get; set; }
        List<string> filesZav;
        AtrIS()
        { filesZav = new List<string>(); }
       static public AtrIS GetAtr()
        {
            if (oneAtr == null)
                oneAtr = new AtrIS();
            return oneAtr;
        }
       public List<string> FilesZav
        {
            get { return filesZav; }
            set { filesZav = value; }
        }
        public string FileNameObsayg()
        {
            string rez;
            rez = DateYear + "_обсяг_" + DepartamentAtr + "xlsx";
            return rez;
        }    }
```

ДОДАТОК Б

Клас модуля представлення

```
public partial class FMain : Form
{
        RunExcel rE, rD;
        AtrIS myAtr;
        public FMain()
        {
            InitializeComponent();
            myAtr = AtrIS.GetAtr();
            AttributIS();
            //екземпляр класу, що інкапсулює ком-сервер екселя
            rE = new RunExcel();
            rD = new RunExcel();
        }
       void AttributIS()
        {
            string s = Application.StartupPath;
            s = Path.GetFullPath(Path.Combine(s, @"../../"))+@"AttIS.txt";
            StreamReader sr = new StreamReader(s);
            string [] rez=sr.ReadToEnd().Split(new char[]
                           {'\n','\r'},StringSplitOptions.RemoveEmptyEntries);
            if (rez.Length > 0)
                tBKaf.Text = rez[0];
            if (rez.Length > 1)
            {
                myAtr.DepartamentAtr = rez[1];
                tBKafAbr.Text = rez[1];
            }
            if (rez.Length > 2)
            {
                tBYear.Text = rez[2];
            }
            if (rez.Length > 3)
                myAtr.PathAtr = rez[3];
            else
                myAtr.PathAtr= Application.StartupPath;
            if (rez.Length > 4)
                myAtr.FileTemplateOb = rez[4];
            if (rez.Length > 5)
                myAtr.FileTemplateContingent = rez[5];
            if (rez.Length > 6)
                for (int i = 6; i < rez.Length; i++)
                    myAtr.FilesZav.Add(rez[i]);
            sr.Close();
        }
       private void DirectoryIS(object sender, EventArgs e)
        {
            if (folderBrowserDialog1.ShowDialog() != DialogResult.OK)
            {
                MessageBox.Show("Ви не обрали каталог");
                return;
            }
            myAtr.PathAtr = folderBrowserDialog1.SelectedPath;
        }
        private void AtrDepartament(object sender, EventArgs e)
        {
            tBKaf.ReadOnly = false;
            tBKafAbr.ReadOnly = false;
        }
       private void шаблонОбсяґуToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
    openFileDialog1.Title = "Оберіть файл обсягу навантаження кафедри";
    openFileDialog1.Filter = "Excel-файли| *.xls; *.xlsx";
    openFileDialog1.InitialDirectory = myAtr.PathAtr;
    openFileDialog1.Multiselect = false;
    if (openFileDialog1.ShowDialog() != DialogResult.OK)
    {
        MessageBox.Show("Ви не обрали файл");
        return;
    }
    myAtr.FileTemplateOb = openFileDialog1.FileName;
}

private void шаблонЗаявкиToolStripMenuItem_Click(object sender, EventArgs e)
{
    openFileDialog1.Title = "Оберіть файл-заявки навантаження кафедри";
    openFileDialog1.Filter = "Excel-файли| *.xls; *.xlsx";
    openFileDialog1.InitialDirectory = myAtr.PathAtr;
    openFileDialog1.Multiselect = true;
    if (openFileDialog1.ShowDialog() != DialogResult.OK)
    {
        MessageBox.Show("Ви не обрали файл");
        return;
    }
    myAtr.FilesZav = openFileDialog1.FileNames.ToList();
}
private void FMain_FormClosed(object sender, FormClosedEventArgs e)
{
    string s = Application.StartupPath;
    s = Path.GetFullPath(Path.Combine(s, @"../../")) + @"AttIS.txt";
    StreamWriter sr = new StreamWriter(s,false);
    sr.WriteLine(tBKaf.Text);
    sr.WriteLine(myAtr.DepartamentAtr);
    sr.WriteLine(tBYear.Text);
    sr.WriteLine(myAtr.PathAtr);
    if(myAtr.FileTemplateOb !=null)
    sr.WriteLine(myAtr.FileTemplateOb);
    if (myAtr.FileTemplateContingent != null)
        sr.WriteLine(myAtr.FileTemplateContingent);
    for (int i = 0; i < myAtr.FilesZav.Count; i++)
        sr.WriteLine(myAtr.FilesZav[i]);

    sr.Close();
    if (rE != null)
        rE.Close();
    if (rD != null)
        rD.Close();
}

private void tBKaf_KeyUp(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        tBKaf.ReadOnly = true;
    }
}
private void tBKafAbr_KeyUp(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        tBKafAbr.ReadOnly = true;
        myAtr.DepartamentAtr = tBKafAbr.Text;
    }
```

```
 }

 private void toolStripMenuItem2_Click(object sender, EventArgs e)
 {
     tBYear.ReadOnly = false;
 }

 private void tBYear_KeyUp(object sender, KeyEventArgs e)
 {
     if (e.KeyCode == Keys.Enter)
     {
         tBYear.ReadOnly = true;
     }
 }
private void новийОбсягToolStripMenuItem_Click(object sender, EventArgs e)
 {
     try
     {
         AtrNav1 atr1 = null;
         AtrNav2 atr2 = null;
         int kolstr = 0;
         for (int kk = 1; kk < 3; kk++)
         {
             openFileDialog1.Title = "Оберіть файл-розрахунку навантаження кафедри
                     в " + Convert.ToString(kk) + " семестрі";
             openFileDialog1.Filter = "Файл навчання| *.nav1";
             openFileDialog1.InitialDirectory = myAtr.PathAtr;
             openFileDialog1.Multiselect = false;
             if (openFileDialog1.ShowDialog() != DialogResult.OK)
             {
                 MessageBox.Show("Ви не обрали файл");
                 return;
             }
             atr1 = new AtrNav1(openFileDialog1.FileName);
             atr1.NumSem = kk;
             if (kk == 1)
                 atr1.NomStrok = atr1.rez1[0].X;
             else
                 atr1.NomStrok = kolstr;

             atr1 = new AtrNav1(openFileDialog1.FileName);
             //відкрить шаблону обсягу
             rE.Open(myAtr.FileTemplateOb);
             //до списку документів в роботі додаємо шаблон обсягу

      checkedListBDocuments.Items.Add(Path.GetFileName(myAtr.FileTemplateOb));
             openFileDialog1.Title = "Оберіть файл-навчання розрахунку обсягу
                           студентів кафедри";
             openFileDialog1.Filter = "Файл навчання| *.nav2";
             openFileDialog1.InitialDirectory = myAtr.PathAtr;
             openFileDialog1.Multiselect = false;
             if (openFileDialog1.ShowDialog() != DialogResult.OK)
             {
                 MessageBox.Show("Ви не обрали файл");
                 return;
             }
             //open Contingent
             if (kk == 1)
                 rE.Open(openFileDialog1.FileName);

             if (myAtr.FilesZav.Count == 0)
             {
                 MessageBox.Show("Заявки навантаження: " + myAtr.FilesZav.Count +
                     Environment.NewLine + "Неможливо виконати поставлену задачу",
```

```
                        "Підключить заявку навантаження:");
                    return;                    }
                //вікриття заявок навантаження
                for (int i = 0; i < myAtr.FilesZav.Count; i++)
                {

                 checkedListBDocuments.Items.Add(Path.GetFileName(myAtr.FilesZav[i]));
                    rE.Open(myAtr.FilesZav[i]);
                }
                 checkedListBDocuments.SetItemChecked(checkedListBDocuments.Items.Inde
            xOf(Path.GetFileName(myAtr.FileTemplateOb)), true);
                //створення копії шаблону обсягу для обчислення обсягу кафедри
                if(kk==1)
                    rE.SaveFileXlsX(tBYear.Text + "_Обсяг_" + myAtr.DepartamentAtr +
                        ".xlsx", myAtr.PathAtr);
                kolstr=rE.Obsayg(myAtr, atr1, atr2);
                atr1.NomStrok = kolstr;
                kolstr -=rE.Proverka(myAtr, atr1);

               }
               rE.Oformlenie(myAtr, atr1);
               rE.CloseSheets();
               MessageBox.Show("Все");
            }
            catch (NullReferenceException)
            {
                MessageBox.Show("Каталог: " + myAtr.PathAtr + Environment.NewLine +
       "Шаблон обсягу: " + myAtr.FileTemplateOb, "Перевірьте налагодження системи:");
            }
        }
private void візуалізаціяАлгоритмуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            openFileDialog1.Title = "Оберіть файл алгоритму розрахунку навантаження
                    кафедри ";
            openFileDialog1.Filter = "Файл навчання| *.nav1";
            openFileDialog1.InitialDirectory = myAtr.PathAtr;
            openFileDialog1.Multiselect = false;
            if (openFileDialog1.ShowDialog() != DialogResult.OK)
            {
                MessageBox.Show("Ви не обрали файл");
                return;
            }
            StreamReader sr = new StreamReader(openFileDialog1.FileName);
            //відкрітття шаблону обсягу
            rE.Open(sr.ReadLine());
            rE.VisibleDocument = true;
            rD.Open(sr.ReadLine());
            rD.VisibleDocument = true;
            sr.ReadLine();
            int k = 3;
            while (!sr.EndOfStream)
            {
                string[] rez = sr.ReadLine().Split('\t');
                rE.Oformlenie(Convert.ToInt32(rez[0]), Convert.ToInt32(rez[1]), k);
                rD.Oformlenie(Convert.ToInt32(rez[2]), Convert.ToInt32(rez[3]), k++);
            }
                sr.Close();
            MessageBox.Show("Навчання успішне?", "Виконання завдання",
                            MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            rE.Close();
            rD.Close();
            }
private void формуванняОбмінуДанихШаблонКонтингентToolStripMenuItem_Click(object sender,
EventArgs e)
```

```
        {
            try
            {
                //відкрття шаблону обсягу
                rE.Open(myAtr.FileTemplateOb);
                rE.VisibleDocument = true;
                rE.CellsClick(myAtr.FileTemplateOb);
                //відкриття шаблону контингенту
                rD.Open(myAtr.FileTemplateContingent);
                rD.VisibleDocument = true;

                rD.CellsClick(myAtr.FileTemplateContingent);
                if (MessageBox.Show("Навчання успішне?", "Виконання завдання",
                     MessageBoxButtons.YesNo, MessageBoxIcon.Question) !=
             DialogResult.Yes)
                {
                    rE.rez.Clear();
                    rD.rez.Clear();
                }
                saveFileDialog1.Title = "Збереження файлу системи обсягу навантаження
                           кафедри-заявка";
                saveFileDialog1.Filter = "Файл навчання| *.nav2";
                saveFileDialog1.InitialDirectory = myAtr.PathAtr;

                if (saveFileDialog1.ShowDialog() != DialogResult.OK)
                {
                    MessageBox.Show("Ви не зберегли інформацію");
                    return;
                }
                StreamWriter sw = new StreamWriter(saveFileDialog1.FileName);
                sw.WriteLine(myAtr.FileTemplateOb);
                sw.WriteLine(myAtr.FileTemplateContingent);
                for (int i = 0; i < rE.rez.Count; i++)
                    sw.WriteLine("{0}\t{1}\t{2}\t{3}", rE.rez[i].X, rE.rez[i].Y,
                                 rD.rez[i].X, rD.rez[i].Y);
                sw.Close();
                rE.Close();
                rD.Close();

            }
            catch (NullReferenceException)
            {
                MessageBox.Show("Каталог: " + myAtr.PathAtr + Environment.NewLine +
        "Шаблон обсягу: " + myAtr.FileTemplateOb, "Перевірьте налагодження системи:");
            }
        }
private void шаблонКонтинтенгентуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            openFileDialog1.Title = "Оберіть файл контингенту";
            openFileDialog1.Filter = "Excel-файли| *.xls; *.xlsx";
            openFileDialog1.InitialDirectory = myAtr.PathAtr;
            openFileDialog1.Multiselect = false;
            if (openFileDialog1.ShowDialog() != DialogResult.OK)
            {
                MessageBox.Show("Ви не обрали файл");
                return;
            }
            myAtr.FileTemplateContingent = openFileDialog1.FileName;
        }
private void візуалізаціяОбмінуДанихШаблонКонтингентToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            openFileDialog1.Title = "Оберіть файл алгоритму розрахунку навантаження
                           кафедри ";
```

```
            openFileDialog1.Filter = "Файл навчання| *.nav2";
            openFileDialog1.InitialDirectory = myAtr.PathAtr;
            openFileDialog1.Multiselect = false;
            if (openFileDialog1.ShowDialog() != DialogResult.OK)
            {
                MessageBox.Show("Ви не обрали файл");
                return;
            }
            StreamReader sr = new StreamReader(openFileDialog1.FileName);
            //відкрітття шаблону обсягу
            rE.Open(sr.ReadLine());
            rE.VisibleDocument = true;
            rD.Open(sr.ReadLine());
            rD.VisibleDocument = true;
            int k = 3;
            while (!sr.EndOfStream)
            {
                string[] rez = sr.ReadLine().Split('\t');
                rE.Oformlenie(Convert.ToInt32(rez[0]), Convert.ToInt32(rez[1]), k);
                rD.Oformlenie(Convert.ToInt32(rez[2]), Convert.ToInt32(rez[3]), k++);
            }
            sr.Close();
            MessageBox.Show("Навчання успішне?", "Виконання завдання",
                    MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            rE.Close();
            rD.Close();
        }
private void формуванняАлгоритмуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            try
            {
                //відкрітття шаблону обсягу
                rE.Open(myAtr.FileTemplateOb);
                rE.VisibleDocument = true;
                rE.CellsClick(myAtr.FileTemplateOb);
                //відкриття шаблону заявки
                if (myAtr.FilesZav.Count == 0)
                {
                    openFileDialog1.Title = "Оберіть файл-заявки навантаження кафедри";
                    openFileDialog1.Filter = "Excel-файли| *.xls; *.xlsx";
                    openFileDialog1.InitialDirectory = myAtr.PathAtr;
                    openFileDialog1.Multiselect = true;
                    if (openFileDialog1.ShowDialog() != DialogResult.OK)
                    {
                        MessageBox.Show("Ви не обрали файл");
                        return;
                    }
                    myAtr.FilesZav = openFileDialog1.FileNames.ToList();
                }
                rD.Open(myAtr.FilesZav[0]);
                rD.VisibleDocument = true;
                rD.CellsClick(myAtr.FilesZav[0]);
                if (MessageBox.Show("Навчання успішне?", "Виконання завдання",
              MessageBoxButtons.YesNo, MessageBoxIcon.Question) != DialogResult.Yes)
                {
                    rE.rez.Clear();
                    rD.rez.Clear();
                }
                saveFileDialog1.Title = "Збереження файлу системи обсягу навантаження
                     кафедри-заявка";
                saveFileDialog1.Filter = "Файл навчання| *.nav1";
                saveFileDialog1.InitialDirectory = myAtr.PathAtr;
                if (saveFileDialog1.ShowDialog() != DialogResult.OK)
                {
```

```
            MessageBox.Show("Ви не зберегли інформацію");
            return;
        }
        StreamWriter sw = new StreamWriter(saveFileDialog1.FileName);
        sw.WriteLine(myAtr.FileTemplateOb);
        sw.WriteLine(myAtr.FilesZav[0]);
        for (int i = 0; i < rE.rez.Count; i++)
            sw.WriteLine("{0}\t{1}\t{2}\t{3}", rE.rez[i].X, rE.rez[i].Y,
             rD.rez[i].X, rD.rez[i].Y);
        sw.Close();
        rE.Close();
        rD.Close();
    }
    catch (NullReferenceException)
    {
     MessageBox.Show("Каталог: " + myAtr.PathAtr + Environment.NewLine + "Шаблон
      обсягу: " + myAtr.FileTemplateOb, "Перевірьте налагодження системи:");
    }
}
```

ДОДАТОК В

Клас модуля взаємодії з com-сервером

```
public class RunExcel
    {
        Excel.Application excelapp = new Excel.Application();
        public List<Point> rez = new List<Point>();
        bool _VisibleDocument;
        public RunExcel()
        {        }
        public void Open(string s)
        {
            try
            {
                excelapp.Workbooks.Open(s,
                         Type.Missing, Type.Missing, Type.Missing, Type.Missing,
                         Type.Missing, Type.Missing, Type.Missing, Type.Missing,
                         Type.Missing, Type.Missing, Type.Missing, Type.Missing,
                         Type.Missing, Type.Missing);
            }
            catch
            {
                throw new NullReferenceException();
            }
        }
public void func(object sender, Excel.Range range)
        {
            rez.Add(new Point(range.Row, range.Column));
        }
        public void Close()
        {
            excelapp.Quit();
        }
public bool VisibleDocument
        {
            get { return _VisibleDocument; }
            set { _VisibleDocument = value;
                excelapp.Visible = _VisibleDocument;
            }
        }
public void SaveFileXlsX(string FileName, string FilePath)
        {
            excelapp.Workbooks[1].SaveAs(FilePath+"\\"+FileName,
Excel.XlFileFormat.xlWorkbookDefault, Type.Missing, Type.Missing, Type.Missing,
Type.Missing, Excel.XlSaveAsAccessMode.xlNoChange, Type.Missing, Type.Missing,
Type.Missing, Type.Missing, Type.Missing);
        }
public int Obsayg(AtrIS myAtr,AtrNav1 nav1, AtrNav2 nav2)
        {
            string rez;
            Excel.Workbook obFile = excelapp.Workbooks[1];
            Excel.Workbook FileConningent = excelapp.Workbooks[2];
            int KolOb = excelapp.Workbooks.Count;
            int p = nav1.NomStrok;
            for (int i = 3; i <= excelapp.Workbooks.Count; i++)
            {
                Excel.Workbook zavFile = excelapp.Workbooks[i];
                string NameSpec="";
                string Riv = "";
                string FormaNav = "";
                if (zavFile.Name.IndexOf("KH") > 0)
                    NameSpec = "122";
                if (zavFile.Name.IndexOf("ICT") > 0)
```

```
                    NameSpec = "126";
                if (zavFile.Name.IndexOf("заочне") > 0)
                    FormaNav="зо";
                else
                    FormaNav = "д";

                for (int j = 1; j <= zavFile.Worksheets.Count; j++)
                {
                                if
            (((Excel.Worksheet)zavFile.Worksheets.get_Item(j)).Name.IndexOf("маг") > 0)
                                Riv = "М";
                            else
                                Riv = "Б";
                                   string kurs =
       Convert.ToString(((Excel.Worksheet)zavFile.Worksheets.get_Item(j)).Name[0]);
                            for (int k = nav1.rez2[0].X; k <= 50; k++)
                            {
                                        rez =
                        Convert.ToString(((Excel.Worksheet)zavFile.Worksheets.get_Item(j)).Cells[k,
                        nav1.Kaf].Value2);
                                if (rez == myAtr.DepartamentAtr)
                                {
                                    for (int i1=0; i1 < nav1.rez1.Count; i1++)
                                    {
                                        dynamic d=
((Excel.Worksheet)zavFile.Worksheets.get_Item(j)).Cells[k, nav1.rez2[i1].Y].Value2;
                                        string myrez = Convert.ToString(d);
                                        ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[p,
nav1.rez1[i1].Y].Value2 = myrez;
                                        ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[p,
3].Value2 = Convert.ToString(nav1.NumSem);
                                        ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[p,
4].Value2 = NameSpec;
                                        ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[p,
5].Value2 = Riv;
                                        ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[p,
6].Value2 = FormaNav;
                                        ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[p,
7].Value2 = kurs;
                                    }
                                    p++;        }                }                            }
                    return p;        }

public void CloseSheet(int i)
        {
            excelapp.Windows[i].Close(false, Type.Missing, Type.Missing);
        }
        public void CloseSheets()
        {
            excelapp.Workbooks.Close();
        }
        public void CellsClick(string i)
        {
                    excelapp.Workbooks[1].SheetSelectionChange += new
       Excel.WorkbookEvents_SheetSelectionChangeEventHandler(func);
        }
        public int Proverka(AtrIS myAtr, AtrNav1 nav1)
        {
            Excel.Workbook obFile = excelapp.Workbooks[1];
            int rez = 0;

                        for (int k = nav1.rez1[1].X; k <=
              ((Excel.Worksheet)obFile.Worksheets[1]).Rows.Count; k++)
            {
```

```
                    if
        (Convert.ToString(((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[k,
        nav1.rez1[0].Y].Value2) == null)
                        break;
                    int kol = 0;
                    for (int i1 = nav1.rez1[1].Y; i1 <= nav1.rez1[nav1.rez1.Count-1].Y; i1++)
                    {
                      string myrez =
Convert.ToString(((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[k, i1].Value2);
                        if (myrez == null)
                            kol++;
                    }
                    if (kol == nav1.rez1.Count - 1)
                    {
                        ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).get_Range("A" +
Convert.ToString(k) + ":AQ" + Convert.ToString(k),
Type.Missing).Rows.Delete(Type.Missing);
                        k--;
                        rez++;}}
                return rez;
            }
      public void CloseSheet(int i)
            {
                excelapp.Windows[i].Close(false, Type.Missing, Type.Missing);
            }
      public void CloseSheets()
            {
                excelapp.Workbooks.Close();
            }
      public void CellsClick(string i)
            {
               excelapp.Workbooks[1].SheetSelectionChange += new
Excel.WorkbookEvents_SheetSelectionChangeEventHandler(func);
            }
      public int Proverka(AtrIS myAtr, AtrNav1 nav1)
            {
                Excel.Workbook obFile = excelapp.Workbooks[1];
                int rez = 0;

                for (int k = nav1.rez1[1].X; k <=
((Excel.Worksheet)obFile.Worksheets[1]).Rows.Count; k++)
                {
                    if
(Convert.ToString(((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[k,
nav1.rez1[0].Y].Value2) == null)
                        break;
                    int kol = 0;
                    for (int i1 = nav1.rez1[1].Y; i1 <= nav1.rez1[nav1.rez1.Count-1].Y; i1++)
                    {
                        string myrez =
Convert.ToString(((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[k, i1].Value2);
                        if (myrez == null)
                            kol++;
                    }
                    if (kol == nav1.rez1.Count - 1)
                    {
                        ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).get_Range("A" +
Convert.ToString(k) + ":AQ" + Convert.ToString(k),
Type.Missing).Rows.Delete(Type.Missing);
                        k--;
                        rez++;
                    }}
                return rez;
    }
```

```
public void Oformlenie(AtrIS myAtr, AtrNav1 nav1)
        {
            Excel.Workbook obFile = excelapp.Workbooks[1];
            for (int k = nav1.rez1[1].X + 1; k <=
            ((Excel.Worksheet)obFile.Worksheets[1]).Rows.Count; k++)
            {
                            if
              (Convert.ToString(((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[k,
              nav1.rez1[0].Y].Value2) == null)
                    break;
                for (int j = 1; j <= 11; j++)
                {
                    ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[k,
j].Borders.ColorIndex = 1;
((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[k, j].Borders.LineStyle =
Excel.XlLineStyle.xlContinuous;
((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[k, j].Borders.Weight =
Excel.XlBorderWeight.xlThin;
                }
            }
        }
        public void Oformlenie(int i, int j, int ColorIndex)
        {
            Excel.Workbook obFile = excelapp.Workbooks[1];
            ((Excel.Worksheet)obFile.Worksheets.get_Item(1)).Cells[i,
j].Interior.ColorIndex = ColorIndex;
        }
    }
```

ДОДАТОК Г

Клас модуля навчання ІС

```
public class AtrNav1
    {
        public List<Point> rez1 = new List<Point>();
        public  List<Point> rez2 = new List<Point>();
        public string FileObsayg { get; set; }

        public string FileZayavka { get; set; }

        public int Kaf { get; set; }

        public int NomStrok { get; set; }

        public int NumSem { get; set; }


        public AtrNav1(string filename)
        {
            StreamReader sr = new StreamReader(filename);
            FileZayavka = sr.ReadLine();
            FileObsayg= sr.ReadLine();
            Kaf = Convert.ToInt32(sr.ReadLine());
            while (!sr.EndOfStream)
            {
                string []rez=sr.ReadLine().Split('\t');
                rez1.Add(new Point(Convert.ToInt32(rez[0]), Convert.ToInt32(rez[1])));
                rez2.Add(new Point(Convert.ToInt32(rez[2]), Convert.ToInt32(rez[3])));
            }
            sr.Close();

        }


    }
```