Міністерство освіти і науки України

Національний авіаційний університет

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

_____ Литвиненко О.Є.

"_____"_____ 2022 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

«БАКАЛАВР»

Тема: Підсистема підтримки прийняття рішень при виникненні пожежі в салоні повітряного судна

Виконавець: Тенягін Д.Д.

Керівник: доц., к.ф.м.-н., Кучерява О.М.

Нормоконтролер: Кучерява О.М.

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютеризованих систем управління

Спеціальність 126 «Інформаційні системи та технології»

(шифр, найменування)

Освітньо-професійна програма «Інформаційні системи та технології»

Форма навчання денна

ЗАТВЕРДЖУЮ

Завідувач кафедри

Литвиненко О.Є.

«_____»_____2022 р.

ЗАВДАННЯ

на виконання дипломного проєкту

Тенягіна Дімитрія Дмитровича

(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломного проєкту: Підсистема підтримки прийняття рішень при виникненні пожежі в салоні повітряного судна

затверджена наказом ректора від 15.02.2022 р. № 251/ст.

2. Термін виконання проекту: з 16.05.2022 р. по 19.06.2022 р

3. Вихідні дані до проєкту: мова програмування *C#*, середовище розробки *Microsoft Visual Studio*, декларативна мова розмітки *XAML*, платформа *.NET Framework*, платформа *Universal Windows Platform*.

4. Зміст пояснювальної записки:

1) Теоретичні аспекти реалізації підсистеми підтримки прийняття рішень
2) Технології для розробки підсистеми підтримки прийняття рішень
3) Розробка та демонстрація підсистеми підтримки прийняття рішень

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) Ілюстрація графічного інтерфейсу програми
2) Схема алгоритму методу *Timer_Tick()*
3) Схема алгоритму конструктора *Airport()*
4) Схема алгоритму методу *Change_Recom()*

Календарний план-графік

№ п/п	Етапи виконання дипломного проєкту	Термін виконання етапів	Відмітка про виконання
1.	Ознайомитись з матеріалами за темою проєкту. Підготувати текст першого розділу пояснювальної записки.	16.05.2022 – 20.05.2022	
2.	Визначити функціонал та архітектуру підсистеми підтримки прийняття рішень.	21.05.2022 – 25.05.2022	
3.	Проаналізувати методи та технології розробки підсистеми підтримки прийняття рішень. Підготувати текст другого розділу пояснювальної записки.	26.05.2022 – 01.06.2022	
4.	Розробка алгоритму, написання та компіляція програмного коду. Підготувати текст третього розділу.	02.06.2022 – 05.06.2022	
5.	Завершити оформлення пояснювальної записки. Оформити графічний матеріал.	06.06.2022 – 10.06.2022	
6.	Підготувати доповідь та презентацію.	15.06.2022 – 16.06.2022	
7.	Захист дипломного проєкту	18.06.2022	

7. Дата видачі завдання: «16» травня 2022 р.

Керівник дипломного проекту _____ Кучерява О.М.
(підпис керівника)

Завдання прийняв до виконання _____ Тенягін Д.Д.
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту «Підсистема підтримки прийняття рішень при виникненні пожежі в салоні повітряного судна»: 49 сторінок, 18 рисунків, 15 використаних джерел, 1 додаток.

МОВА ПРОГРАМУВАННЯ *C#*, СЕРЕДОВИЩЕ РОЗРОБКИ *MICROSOFT VISUAL STUDIO*, ДЕКЛАРАТИВНА МОВА РОЗМІТКИ *XAML*, ПЛАТФОРМА *.NET FRAMEWORK*, ПЛАТФОРМА *UNIVERSAL WINDOWS PLATFORM*.

Об’єкт дипломного проєктування – процес прийняття рішень.

Предмет дипломного проєктування – підсистема підтримки прийняття рішень.

Мета дипломного проєкту – розробка підсистеми підтримки прийняття рішень при виникненні пожежі в салоні повітряного судна.

Методи проєктування – сучасні методи розробки підсистем прийняття рішень, продукційна модель представлення знань, мова програмування *C#*, середовище розробки *Microsoft Visual Studio*, декларативна мова розмітки *XAML*, платформа *.NET Framework*, платформа *Universal Windows Platform*.

Результатом виконання дипломного проєкту є розроблена підсистема для надання підтримки вибору рішення при пожежі в салоні повітряного судна. Дана підсистема забезпечує операторам легкий доступ до моделей і даних для того, щоб підтримати процес прийняття рішень стосовно аварійних ситуацій, пов’язаних з пожежею на борту літака.

Результати дипломного проєктування рекомендується застосувати у сучасних інформаційних системах повітряних суден.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ 6
ВСТУП 7
РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ РЕАЛІЗАЦІЇ ПІДСИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ 10
1.1 Системи підтримки прийняття рішень 10
1.2 Базові компоненти систем підтримки прийняття рішень 11
1.3 Класифікація систем підтримки прийняття рішень 20
1.4 Пожежна безпека у повітряному судні 24
1.5 Висновки до розділу 27
РОЗДІЛ 2 ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ПІДСИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ 29
2.1 Мова програмування *C#* 29
2.2 Декларативна мова розмітки *XAML* 30
2.3 Платформа *.NET Framework* 31
2.4 Інтегроване середовище розробки *Microsoft Visual Studio* 32
2.5 Універсальна платформа *Windows* 35
2.6 Вимоги до системи 38
2.7 Висновки до розділу 38
РОЗДІЛ 3 ФУНКЦІОНУВАННЯ ПІДСИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ 39
3.1 Розробка алгоритму пошуку альтернатив 39
3.2 Проєктування графічного інтерфейсу 41
3.3 Висновки до розділу 45
ВИСНОВКИ 46
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ .. 48

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

СППР	Система підтримки прийняття рішення
ОПР	Особа, що приймає рішення
БД	База даних
СУБД	Система управління базою даних
БМ	База моделей
СУБМ	Система управління базою моделей
БЗ	База знань
ПС	Повітряне судно
CLR	*Common Language Runtime*
XAML	*eXtensible Application Markup Language*

ВСТУП

Актуальність теми. В даний час повітряний транспорт залишається дуже зручним засобом пересування для пасажирських та вантажних потоків. Обладнання літаків та різноманітні інформаційні системи, а також і системи безпеки вдосконалювалися протягом багатьох років. Але навіть при сучасних системах аварійна ситуація може вплинути на здатність екіпажу чітко мислити та приймати рішення.

Для аварійних ситуацій потрібно розробити систему, яка на основі введених, або отриманих іншим способом даних допоможе оператору вибрати найкраще рішення у даній ситуації, наприклад при пожежі у салоні повітряного судна.

Мета і завдання виконання дипломного проекту. Метою дипломного проєкту є розробка підсистеми підтримки прийняття рішень при виникненні пожежі в салоні повітряного судна. Для виконання цілей дипломного проєктування було застосовано навички по роботі в інтегрованому середовищі розробки *Microsoft Visual Studio*, платформі *Universal Windows Platform*, мові програмуванні *C#* та декларативній мові розмітки *XAML*.

Згідно з метою дипломного проєкту, було визначено такі завдання:

- розробка та опис предметної області підсистеми підтримки прийняття рішень при виникненні пожежі в салоні повітряного судна;
- проєктування та розробка алгоритму роботи підсистеми;
- створення програми для тестування підсистеми на мові C#;
- демонстрація функціонування підсистеми.

При розробці підсистеми буде реалізовано:

а) аналіз бази знань та вхідних даних (аналіз тактико-технічних даних літака, сигналів від різних датчиків та введених оператором даних);

б) знаходження алгоритмом альтернатив за отриманими результатами аналізу;

в) виведення рекомендованого рішення та альтернатив за допомоги графічного інтерфейсу.

Об’єкт і предмет проєктування. Об’єктом дипломного проєктування є процес прийняття рішень.

Предметом дипломного проєктування являється підсистема підтримки прийняття рішень.

Авіаційний бізнес лише набирає обороти і польоти на літаках вже максимально доступні кожному. Тому варто подбати безпеку людей у процесі польоту. У результаті дипломного проєктування, відповідно до поставленого завдання, було розроблено програму для тестування підсистеми підтримки прийняття рішень при пожежі у салоні повітряного судна, який не тільки показує оператору найкращий варіант дії, але і інші альтернативи, що надає змогу повисити міри безпеку під час перельотів.

Підсистема буде в складі основної системи підтримки прийняття рішень інформаційної системи повітряного судна.

Методи проєктування. В якості засобу створення підсистеми підтримки прийняття рішення було вирішено використати інтегроване середовище програмування *Microsoft Visual Studio*. Дане середовище було обрано через його здатність підтримувати велику кількість мов програмування та різних технологій.

Мовою програмування для реалізації алгоритмів було обрано *C#*. Мова була вибрана через велику кількість підтримуваних технологій, бібліотек та можливість легко перекомпільовувати підсистему на інші операційні системи.

Для розробки також було застосовано платформу *Universal Windows Platform*. Вона використовується для створення універсальних програм, що запускаються на *Windows 10*, *Windows 10 Mobile* і *Windows 10 IoT* без зміни у коді. Підтримує обрану мову програмування *C#*.

Для збільшення функціоналу мови програмування *C#* було обрано також платформу *.NET Framework*. Ця технологія підтримує створення і виконання

нового покоління додатків. Обрана платформа *Universal Windows Platform* легко поєднується з *.NET Framework.*

При роботі з платформами *Universal Windows Platform* та *.NET Framework* застосовують декларативну мову розмітки *XAML*. Мова *XAML* спрощує створення користувацького інтерфейсу для додатків.

Практичне значення отриманих результатів. Розроблений у даному дипломному проєкті програмний засіб для тестування підсистеми підтримки прийняття рішень дозволить оператору системи приймати найкращу альтернативу у найкоротший час у випадку виникнення пожежі в салоні повітряного судна за допомоги запропонованих підсистемою рішень. Ця інформація допоможе підвищити безпеку людей при польоті у літаку.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ РЕАЛІЗАЦІЇ ПІДСИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

1.1 Системи підтримки прийняття рішень

Система підтримки прийняття рішень (СППР) — це інформаційна система, яка допомагає бізнесу в прийнятті рішень, які вимагають оцінки, рішучості та послідовності дій. Інформаційна система допомагає керувати організацією середнього і високого рівня, аналізуючи величезні обсяги неструктурованих даних і накопичуючи інформацію, яка може допомогти у вирішенні проблем і прийнятті рішень. У СППР технологію обробки даних формує ОПР у процесі взаємодії з системою (рис. 1.1.).

Для цього СППР надає користувачеві набір даних, програмних модулів і моделей, з яких користувач вибирає саме ті ресурси і технології, які дозволять отримати йому потрібну інформацію.

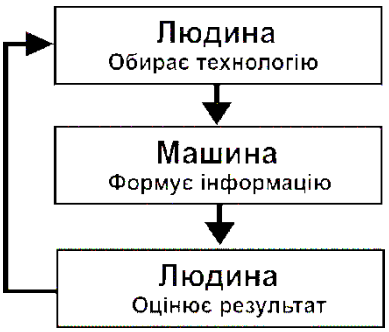

Рис. 1.1. Етапи взаємодії СППР та ОПР

У процесі своєї взаємодії СППР та ОПР утворюють єдину систему. У цій системі СППР можна розглядати як високотехнологічне продовження людини, що підсилює його здібності та розширює можливості (рис. 1.2)

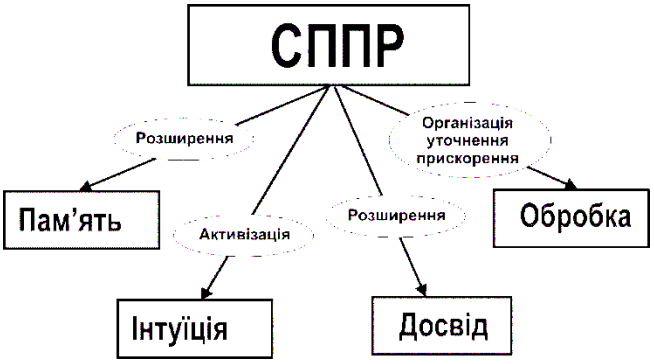

Рис. 1.2. Орієнтація СППР на набір можливостей

Таким чином, з метою розширення можливостей людини СППР у підсумку виконує такі функції:

- підтримує інформаційну модель предметної галузі та забезпечує швидкий і асоціативний доступ до її елементів. Це функція розширення пам'яті ОПР;
- підтримує генерування цілей і нестандартних альтернатив. Це функція активізації творчого мислення та інтуїції ОПР;
- зберігає знання про раніше вирішені проблеми та способи їхнього вирішення. Це функція активізації досвіду ОПР та експертів;
- забезпечує створення, збереження та використання формалізованих моделей. Це функція підтримки математичного інструментарію.

СППР повинні мати такі характеристики, як інтегрованість СППР; достатня потужність; доступність; гнучкість; надійність; працездатність; керованість.

1.2 Базові компоненти систем підтримки прийняття рішень

Вирішення задач проектування складних систем може вивчатися і розвиватися на різних рівнях узагальнення і деталізації. На найвищому рівні спільності основна увага приділяється розробці принципів організації системи та виробленню загального погляду на майбутню систему. Такі загальні аспекти побудови системи називаються *архітектурою системи*.

Архітектура СППР базується на фізичному з'єднанні між її різними компонентами, а також апаратним забезпеченням. А спосіб, яким компоненти або системи об'єднані в мережу, визначає те, як відбувається потік інформації.

Розглянемо приклади формулювання вимог (властивостей) і відповідних їм функцій СППР, необхідних для виконання архітектурного проектування (табл. 1.1).

Таблиця 1.1

Властивості СППР і відповідні їм функції

Властивості СППР	**Функція**
Підтримувати інформаційну модель проблемної сфери й забезпечувати швидкий і асоціативний доступ до її елементів	Розширення пам'яті ОПР
Підтримувати ОПР у процесі генерування цілей і нестандартних альтернатив	Активізація інтуїції ОПР
Указувати можливі напрямки для пошуку й аналізу інформації, що може бути побічно пов'язана з проблемою і враховує фактори людського поводження та соціальну обумовленість рішення	Підтримки поведінкового аспекту прийняття рішень ОПР
Забезпечувати побудову, збереження і використання формальних моделей, що описують окремі аспекти проблемних ситуацій	Багаторазове обчислення рішень на основі різних моделей прийняття рішень

Зберігати знання про раніше вирішені проблеми та способи їх вирішення, забезпечувати їх активнувзаємодію з ОПР	Збереження та активізації досвіду ОПР і експертів у даній проблемній сфері

Незважаючи на те, що сьогодні існує велика кількість різних видів СППР, всі вони характеризуються однотипною структурою, яка включає три основні базові підсистеми (рис. 1.3):

1) *інтерфейс користувача*, основною функцією якого є забезпечення можливості ОПР проводити діалог із системою, використовуючи різні способи введення інформації і формати її виведення;

2) *підсистему роботи з даними*, головна функція якої – збереження, управління, вибірка, відображення, аналіз даних;

3) *підсистему роботи з моделями*, призначенням якої є збереження, управління та вибір моделей для забезпечення користувача відповідями на безліч його запитів.

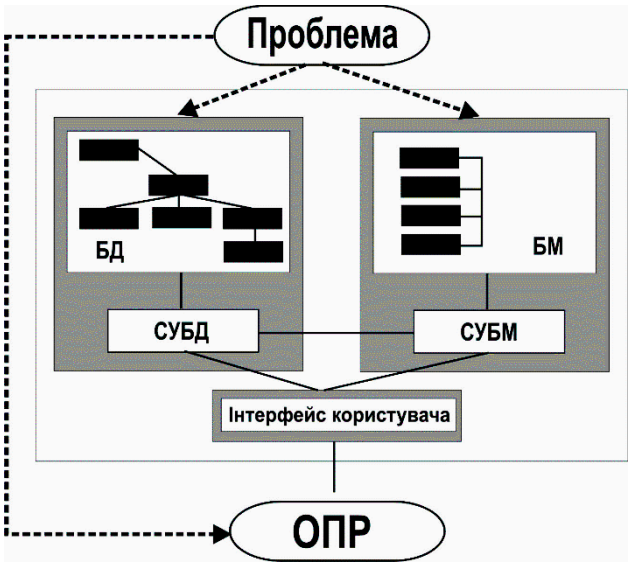

Рис. 1.3. Структура СППР

Підсистема роботи з даними об’єднує базу даних (БД) і систему управління базою даних (СУБД). Даними називають окремі факти, які

характеризують об’єкти, процеси та явища предметної галузі. Певним чином організований набір даних утворює базу даних, а узагальнені програмні засоби, які забезпечують користувачам можливості зберігання, перетворення, вибору та аналізу даних, називають системою управління базою даних.

Підсистема роботи з моделями об’єднує базу моделей (БМ) і систему управління нею (СУБМ). База моделей – це спеціально організований набір формалізованих моделей, насамперед математичних. Кожна математична модель являє собою систему математичних виразів, яка відображає основні властивості та закономірності функціонуваннявідповідного об’єкта. Сукупність програмних засобів, які забезпечують користувачам можливості вибору, застосування та зміни моделей, утворює систему управління базою моделей.

Поняття “*інтерфейс користувача*” означає комплекс програмних засобів, які реалізують діалог користувача з системою на стадії введення інформації та при одержанні результатів.

Ще одним важливим і все частіше використовуваним компонентом СППР є *база знань* (БЗ). Знання – це виявлені людиною закони й закономірності предметної галузі, які дозволяють ставити та вирішувати задачі. Знання, хоча й засновані на емпіричних даних, але являють собою результат розумової діяльності людини, спрямованої на узагальнення її практичного досвіду. У базі знань зберігаються знанняпро раніше вирішені проблеми та способи їхнього вирішення, а також різні рекомендації, які узагальнюють досвід експертів щодо процесу прийняття рішень.

Користувач сприймає систему (в тому числі СППР) через її інтерфейс. Фактично він ототожнює систему з її інтерфейсом. Незалежно від того, наскільки добре реалізовані інші компоненти СППР, без відповідного інтерфейсу користувача система не зможе повною мірою забезпечувати підтримку прийняття рішення.

Підсистема даних СППР складається з бази даних і системи управління базою даних (СУБД).

Особливості підсистеми даних у СППР:

- необхідність виконання великого обсягу операцій з переструктурування даних;
- необхідно передбачати можливість завантаження і наступної обробки даних із зовнішніх джерел (потреба в зовнішніх даних тим вища, чим більш високий рівень керівництва);
- функціонування СУБД у середовищі СППР вимагає більш широкого набору функцій;
- використання нетрадиційних для ЕІС даних (текстова інформація, матеріали САПР та ін.).

Підсистема моделей СППР. Підсистема моделей СППР складається з бази моделей і системи управління базою моделей (СУБМ). Під базою моделей мається на увазі сукупність моделей, спрямованих на дослідження та перевірку безлічі альтернатив.

База моделей може включати:

1) оптимізаційні моделі (моделі математичного програмування, розподіл ресурсів, оптимальне планування, аналіз сіткових графіків тощо;моделі нелінійного і динамічного програмування, моделі аналізу цінних паперів для визначення інвестиційної стратегії та ін.);

2) неоптимізаційні моделі (статистичні моделі на основі аналізу регресії; моделі прогнозування часових рядів; моделі машинної імітації та ін.);

3) моделі на основі понять і методів представлення знань (формальна логіка, моделі продукцій, семантичні мережі, фрейми та гібриди перерахованих способів представлення знань).

Взаємодію користувача з моделями забезпечує *система управління базою моделей,* що є однією з компонентів архітектури СППР.

Основним елементом бази моделей є власне моделі, які можуть бути незалежними одна від одної або утворювати різні структури: ієрархічні або мережні (рис. 1.4). Взаємозв’язок між моделями визначається потребами розв’язуваної задачі й особливостями їх реалізації.

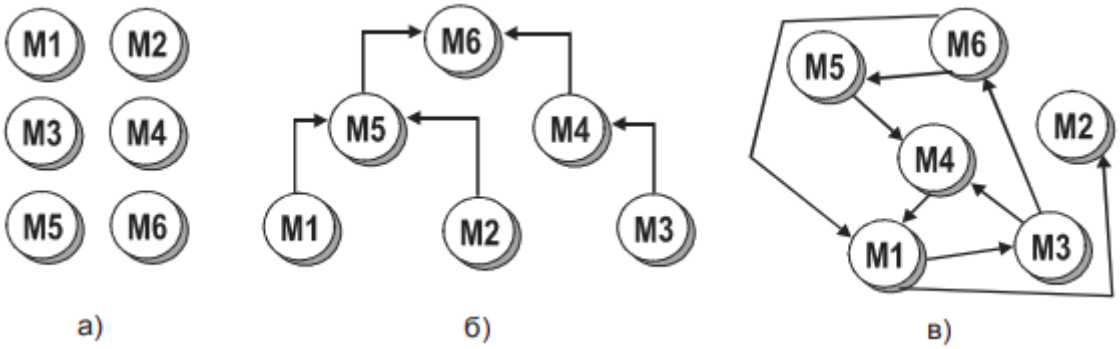

Рис. 1.4. Варіанти структури бази моделей:

а) моделі М1-М6 не залежать одна від одної;

б) моделі М1-М6 утворюють ієрархічну структуру;в) моделі М1-М6 утворюють мережну структуру.

Програмне забезпечення для СУБМ розроблено значно слабкіше порівняно із СУБД і користувацьким інтерфейсом. Тому, як правило, розробники СППР змушені самостійно проектувати і програмно реалізовувати БМ і СУБМ. На рис. 1.5 наведені найбільш розповсюджені способи програмної реалізації БМ і СУБМ.

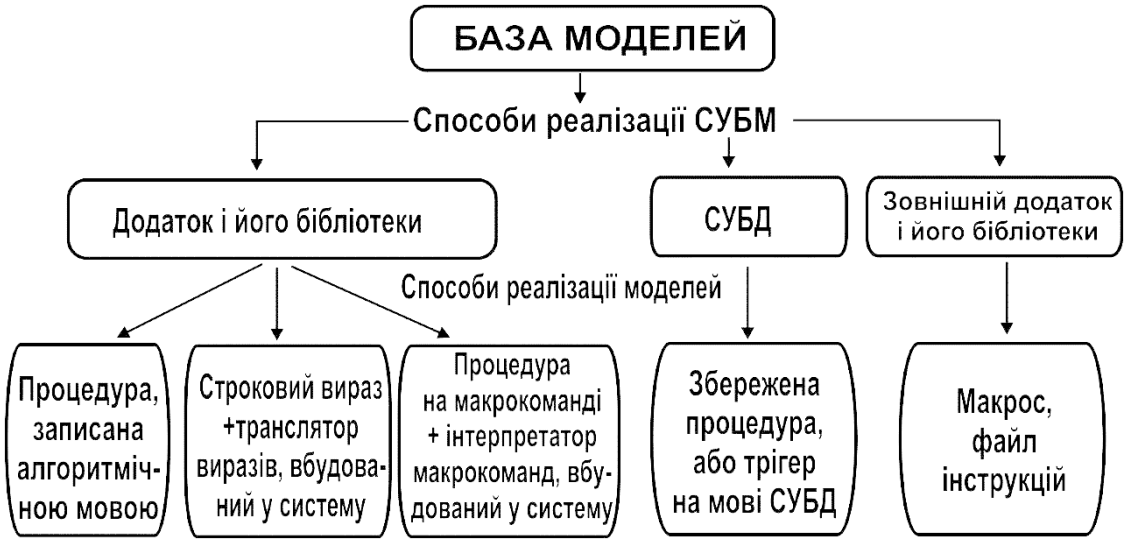

Рис. 1.5. Способи програмної реалізації СУБМ і БМ

Типи архітектури СППР

Узагальнюючи різні визначення, можна зробити висновок, що структура СППР як системи складається із п’яти головних компонентів:

- бази даних;

- системи управління базою даних;
- бази моделей;
- системи управління базою моделей;
- користувацького інтерфейсу.

Залежно від характеру і типу зв’язків між цими компонентами в задачах конструювання СППР виділяють чотири базові різновиди структур СППР: мережна, міст, шарова (типу “сандвіч”), вежа.

Незалежно від типу структури кожна з них містить три компоненти: діалог, базу даних і базу моделей.

Структуру *мережної СППР* зображено на рис. 1.6.

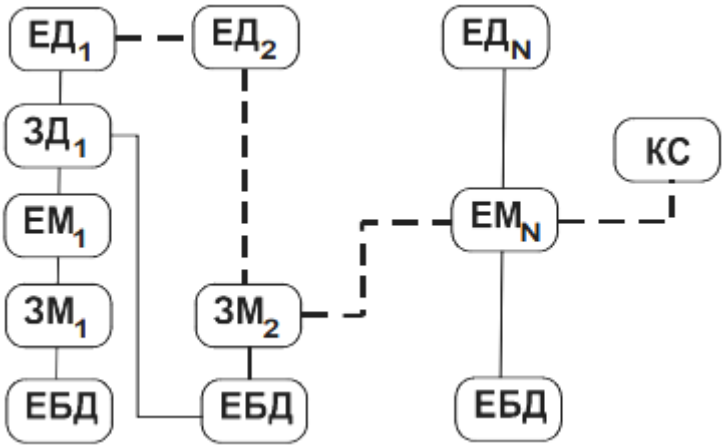

Рис. 1.6. Мережна структура СППР

До складу мережної структури СППР входять:

- елементи діалогу користувача й СППР (ЕД);
- елементи моделювання – складові бази моделей (ЕМ);
- елементи бази даних (ЕБД);
- елементи інтерфейсу – «система обслуговування з’єднань» (обслуговування зв’язків з діалогом (ЗД); обслуговування зв’язків з моделюванням – ЗМ);
- координатор системи обслуговування зв’язків (КС).Переваги структури мережі:
- елементи структури мережі можуть бути неоднорідними, виникатив

різний час і проектуватися різними особами;

• додавання нового елемента діалогу або бази моделей зводитьсятільки до розвитку “системи обслуговування з’єднань”;

• відрізняється простотою і легкістю в інтегруванні окремих (незалежно побудованих) елементів СППР.

Структуру СППР типу “міст” наведено на рис. 1.7.

Рис. 1.7. Структура СППР типу “міст”

До переваг даної структури СППР варто віднести:

• єдину монолітну систему обслуговування з’єднань;

• низьку вартість включення нових функцій.Недоліки:

• локальні елементи, недоступні для інших користувачів;

• вимога, щоб усі локальні елементи обслуговувалися тими самими апаратними засобами.

Структура типу “сандвіч” (шарова) включає безліч елементів моделювання, кожний з яких користується єдиною базою даних і єдиною системою діалогу. Схему структури типу “сандвіч” наведено на рис. 1.8.

Рис. 1.8. Структура СППР типу “сандвіч”

Переваги структури типу “сандвіч”:

- кожен елемент моделювання користується однією і тією ж базою даних;
- пересилання даних між окремими елементами моделювання відбувається через загальну базу даних;
- доцільно використовувати при великих масивах даних і використанні програм, що перетворюють дані.

Недоліки:

- труднощі інтеграції зовнішніх даних (потрібна їх трансформація);
- всі елементи структури вимагають одного програмного забезпечення.

Структуру СППР типу “вежа” зображено на рис. 1.9.

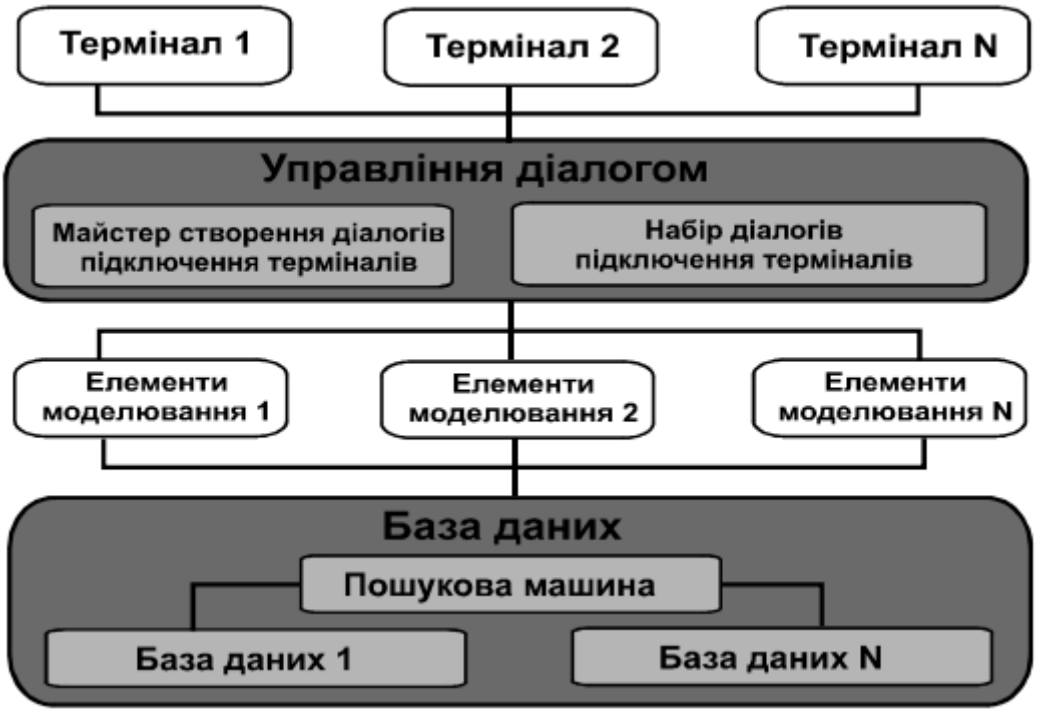

Рис. 1.9. Структура СППР типу “вежа”

До основних переваг цієї структури можна віднести:

- можливість поєднувати різне устаткування апаратного забезпечення;
- можливість поєднувати різні бази даних.
- Недоліки:
- труднощі при інтеграції всіх елементів системи;
- велика залежність від окремих зв’язків діалогу з базою даних;
- експлуатаційні труднощі у випадку наявності багатьох рівнів в організації.

1.3 Класифікація систем підтримки прийняття рішень

Класифікація СППР.

Існує велика кількість СППР, що відрізняються одна від одної цілями, призначенням та предметними галузями. Тому для уніфікації розуміння поняття СППР, підходів до розробки СППР та їхнього використання проводяться класифікації СППР.

До основних *класифікаційних ознак* варто віднести:

- інструментальний підхід до створення СППР;
- ступінь залежності ОПР від СППР;
- міру підтримки ухвалення рішення;
- моделі СППР;
- школи створення СППР.

Класифікація СППР на *основі ступеня залежності ОПР* у процесі прийняття рішення поділяється на:

- термінальну підтримку, що використовується у випадку, коли ОПР має авторитет і відповідні повноваження і несе відповідальність за ухвалення рішення та забезпечення його реалізації;
- групову підтримку, що застосовується у випадку, коли рішення приймаються в результаті переговорів і взаємодії між декількома ОПР;
- організаційну підтримку, що здійснюється у випадку, коли ОПР приймає тільки частину рішень, що передається наступній ОПР для подальшої роботи.

Класифікація на *основі міри підтримки прийняття рішень* містить у собі два підкласи: “Системи, орієнтовані на дані” і “Системи, орієнтовані на моделі”, кожний з яких поділяється на види (табл. 1.2).

Таблиця 1.2

Класифікація СППР за ознакою “Міра підтримки прийняття рішень”

Системи,	Системи накопичення файлів	Забезпечують доступ до елементів даних і містять тільки підсистеми інтерфейсу користувача, БД і СУБД
	Системи аналізу даних	Дозволяють проводити маніпуляцію над даними і одержувати протоколи аналізу з використанням спеціально розроблених і загальних засобів

орієнтовані на дані	Системи аналізу інформації	Забезпечують доступ до декількох баз даних і до невеликих моделей. Ці системи поєднують виходи системи аналізу даних, що орієнтовані на обслуговування запитів користувачів, з даними від зовнішніх джерел інформації
Системи, орієнтовані на моделі	Системи на основі розрахункових моделей	Дозволяють оцінювати наслідки планових дій за допомогою процедур, в основі яких лежать точні, надійні і добре формалізовані моделі
	Системи на основі образних моделей	Генерують оцінки наслідків дій на основі частково визначених імітаційних моделей
	Системи на основі оптимізаційних моделей	Забезпечують вибір напрямків дій шляхом ідентифікації оптимальних рішень, сумісних з набором обмежень
	Системи на основі рекомендаційних моделей	Виробляють конкретні рекомендовані рішення для слабоструктурованих задач. Системи даного класу дають готові рішення задач

Інформація про різні підходи до класифікації СППР наведена в табл. 1.3.

Таблиця 1.3

Класифікація СППР

Класифікаційна ознака	**Типи систем**
Призначення СППР:	Прикладні СППР, СППР-генератори, СППР-

інструментальний підхід	інструментарій
Спеціалізація СППР	Функціонально-спеціалізовані СППР, універсальні СППР
Характер вибору	Цілісний вибір, багатокритеріальний вибір
Тип моделі	Об’єктивна модель, суб’єктивна модель
Ступінь залежності ОПР у процесі прийняття рішень	Термінальна підтримка, групова підтримка, організаційна підтримка
Ієрархічний рівень управління	Вища ланка управління, середня ланка управління, нижча ланка управління
Часовий горизонт	Стратегічне управління, тактичне управління, оперативне управління
Професійна сфера	Мікроекономіка, макроекономіка, оцінка та розповсюдження технологій тощо

Однак слід зазначити, що з точки зору створення і проектування СППР доцільно розглядати їх з погляду моделі СППР і школи створення.

Моделі СППР

Існує ряд систем підтримки прийняття рішень. Їх можна розділити на п’ять типів:

СППР на основі зв'язку

Більшість СППР, керованих комунікаціями, орієнтовані на внутрішні команди, включаючи партнерів. Її мета полягає в тому, щоб допомогти провести зустріч або для співпраці користувачів. Найпоширенішою технологією, яка використовується для розгортання СППР, є веб- або клієнтський сервер. Приклади: чати та програмне забезпечення для обміну миттєвими повідомленнями, онлайн-співпраця та системи мережевих зустрічей.

СППР, керована даними

Більшість СППР, керованих даними, орієнтовані на менеджерів, персонал, а також на постачальників продуктів/послуг. Вона використовується для запиту до бази даних або сховища даних для пошуку конкретних відповідей для певних цілей. Вона розгортається через систему основного фрейма, посилання клієнт/сервер або через Інтернет. Приклади: комп'ютерні бази даних, які мають систему запитів для перевірки.

СППР на основі документів

СППР, керовані документами, є більш поширеними, орієнтованими на широку базу груп користувачів. Метою такої СППР є пошук веб-сторінок і пошук документів за певним набором ключових слів або пошукових термінів. Звичайна технологія, яка використовується для встановлення таких СППР, здійснюється через Інтернет або систему клієнт/сервер.

СППР, орієнтований на знання:

СППР, керовані знаннями, або «база знань», якщо вони відомі, є загальною категорією, яка охоплює широкий спектр систем, що охоплюють користувачів в організації, які її створюють, але можуть також включати інших, які взаємодіють з організацією. По суті, вона використовується для надання консультацій керівництву або для вибору продуктів або послуг. Типовою технологією розгортання, яка використовується для встановлення таких систем, може бути Інтернет або програмне забезпечення, що запускається на автономних ПК.

СППР на основі моделі

СППР на основі моделі — це складні системи, які допомагають аналізувати рішення або вибирати між різними варіантами. Вони використовуються менеджерами та співробітниками бізнесу або людьми, які взаємодіють з організацією, для ряду цілей, залежно від того, як налаштована модель – планування, аналіз рішень тощо. Ці СППР можуть бути розгорнуті за допомогою програмного або апаратного забезпечення в автономному ПК, клієнтській або серверній системи або Інтернет.

1.4 Пожежна безпека у повітряному судні

Пожежі всередині відсіків повітряного судна(ПС), зокрема у пасажирських салонах та кабіні, відносяться до пожеж у замкнутому обсязі. Основним горючим завантаженням при таких пожежах є декоративно-оздоблювальні та конструкційні елементи інтер'єру, що є штучними та натуральними матеріалами оббивки та наповнення крісел, килимові покриття, електропроводка, пластмасові вироби.

Для пожеж усередині фюзеляжу характерні невеликі розміри пожежі, що викликає високу задимленість приміщення, відносно швидке наростання температури у верхній частині приміщень та повільне – у зоні підлоги. При горінні (через 2-3 хв) середнє значення температури в зоні пожежі в 2-4 рази перевищує температуру в зоні підлоги. Середньооб'ємна температура при горінні (до моменту прогоряння обшивки) не перевищує, як правило, 250°С і має деяку тенденцію до зниження. Пожежа має тліючий характер без видимого полум'я, проте вона не припиняється до повного вигоряння горючого завантаження. Горіння відбувається по поверхні стін, стель, пасажирських крісел, але може бути і в обсязі салону за рахунок крапель розплавлених синтетичних матеріалів, що стікають з оздоблювальних та конструкційних елементів стелі салону та кабіни екіпажу.

При прогорянні обшивки пожежа всередині фюзеляжу зазвичай посилюється аж до появи відкритого полум'я та температура у верхній частині салонів різко зростає (до 900 ° С). Висока температура може призвести до розплавлення та загоряння сплавів магнію, що входять до конструкції деяких типів пасажирських крісел, що ускладнює гасіння пожежі.

При пожежах усередині фюзеляжу відбувається швидке наростання концентрації отруйних речовин продуктів горіння та термічного розкладання горючих матеріалів, що зумовлюють основну небезпеку для людей, що знаходяться на борту сонця, що горить.

Характер пожеж усередині фюзеляжу повітряного судна визначає і складність його гасіння, пов'язану з важкодоступністю вогнища пожежі та труднощами визначення його розташування.

При перших ознаках або підозрі на дим і випаровування або пожежу в літаку екіпаж повинен надіти протидимні окуляри та кисневі маски. Окуляри та маски повинні щільно прилягати, щоб мінімізувати будь-яке потрапляння диму та випарів у маску. Важливо, щоб весь екіпаж був повністю знайомий з роботою свого захисного обладнання.

Багато попереджень про дим і пожежу виявляються помилковими. Пасажири та бортпровідники, які повідомляють про незвичайні запахи та випари, можуть бути схильні применшити ситуацію, побоюючись збентеження, якщо вони помиляються. Попередження про пожежу/дим і повідомлення про дим або випари слід сприймати серйозно, доки не буде позитивного підтвердження того, що попередження є помилковими. Якщо це справжня пожежа, то льотному екіпажу може залишитися недовго, щоб розібратися з ситуацією – час має вирішальне значення.

Екіпаж повинен негайно розпочати спуск і розпочати планування аварійної посадки. Слід оголосити надзвичайну ситуацію та повідомити диспетчеру, що літак знижується. У зоні з інтенсивним рухом, коли в безпосередній близькості може бути кілька літаків, було б хорошою ідеєю оголосити надзвичайну ситуацію та попросити зниження та вектори до найближчого аеродрому перед початком зниження. Однак, якщо цей дозвіл не надається негайно, треба спускатися без нього. Посадити літак на землю протягом 15 хвилин після виявлення пожежі є складним завданням, якщо екіпаж перебуваєте на крейсерській висоті на сучасному пасажирському літаку, тому будь-яка затримка початку спуску може стати фатальною.

Те саме не стосується літаків над відкритими океанами або над малонаселеними районами суші, такими як північ Канади чи схід Росії. У цих районах доцільно використовувати всі ресурси, щоб екіпаж завжди знав, де знаходиться найближчий відповідний аеродром, і міг повернути прямо на трасу в разі надзвичайної ситуації. Екіпажі з високою ситуаційною обізнаністю підготовані та знають, ЯК здійснити захід на незнайомий аеродром із незвичайними висотами та процедурами в разі надзвичайної ситуації. Робоче

навантаження буде високим, а видимість може погіршитися, тому підготовка під час низького навантаження може бути життєво важливою. Якщо необхідно прискорити етап маршруту до аварійного аеродрому, екіпаж думає про коригування висоти для досягнення максимальної наземної швидкості. Якщо є кількісні ознаки неконтрольованої пожежі, то існує реальна ймовірність втрати контролю в короткостроковій перспективі, і, отже, приземлення за межами посадової смуги можуть бути єдиним способом пережити пожежу.

Хоча вимога полягає в тому, щоб посадити літак якомога швидше, екіпаж повинен зробити все можливе, щоб ізолювати і контролювати пожежу. Бортпровідники повинні знайти джерело вогню та «боротися» з ним, використовуючи всі доступні ресурси. Екіпажі повинні дотримуватися стандартних процедур щодо гасіння пожежі в польоті.

1.5 Висновки до розділу

В цьому розділі було розглянуто системи підтримки прийняття рішень. Системи підтримки прийняття рішень (СППР) — інформаційні системи, які використовують обладнання, програмне забезпечення, дані, базу моделей і роботу менеджера з метою підтримки всіх стадій прийняття рішень у процесі аналітичного моделювання.

Класифікують такі СППР:

1. Системи, орієнтовані на дані (вибирають інформацію):

- накопичування файлів (*File draver systems*);
- аналізу даних (*Data analysis systems*);
- аналізу інформації (*Analysis information systems*).

2. Системи, орієнтовані на моделі (дають змогу підтримувати прийняття рішень):

- розрахункові або облікові та фінансові моделі;
- репрезентативні або образні;
- оптимізаційні;

- рекомендаційні.

Також було досліджено пожежі у фюзеляжі повітряного судна. Були виявлені дії, які потрібен виконувати екіпаж під час пожежі, а саме:

- екіпірувати захисне обладнання;
- подати сигнал про аварійну ситуацію до диспетчера;
- почати боротися с вогнем у повітряному судні;
- якнайшвидше почати зниження та пошук оптимального місця для аварійної посадки.

РОЗДІЛ 2
ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ПІДСИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

2.1 Мова програмування *C#*

C# — це сучасна, об'єктно-орієнтована мова програмування загального призначення, яка вимовляється як «*C Sharp*». Вона була розроблена Microsoft під керівництвом Андерса Хейлсберга та його команди в рамках ініціативи *.NET* і схвалена Європейською асоціацією виробників комп'ютерів (*ECMA*) та Міжнародною організацією зі стандартів (*ISO*). *C#* є однією з мов для загальномовної інфраструктури. Синтаксично *C#* дуже схожа на *Java* і є простою для користувачів, які знають *C, C++* або *Java*.

Як відзначав сам розробник мови, *C#* створювалася як мова компонентного програмування, і в цьому одна з головних переваг мови, спрямована на можливість повторного використання створених компонентів. З інших об'єктивних факторів відзначимо наступні:

а) C# створювався паралельно з каркасом *FrameworkNet* і повною мірою враховує всі його можливості - як *FCL*, так й *CLR*;

б) C# є повністю об'єктно-орієнтованою мовою, де навіть типи, вбудовані в мову, представлені класами;

в) C# є потужною об'єктною мовою з можливостями спадкування й універсалізації;

г) C# є спадкоємцем мов *C/C++*, зберігаючи кращі риси цих популярних мов програмування;

д) завдяки каркасу *Framework .Net*, що стали надбудовою над операційною системою, програмісти *C#* одержують ті ж переваги роботи з віртуальною машиною, що й програмісти *Java*. Ефективність коду навіть підвищується, оскільки виконавче середовище *CLR* представляє собою компілятор проміжної мови, у той час як віртуальна *Java*-машина є інтерпретатором байт-коду;

е) потужна бібліотека каркасів підтримує зручність побудови різних типів додатків на *C#*, дозволяючи легко будувати *Web*-служби, інші види компонентів, досить просто зберігати й одержувати інформацію з бази даних й інших сховищ даних;

є) реалізація, що сполучає побудову надійного й ефективного коду, є немаловажним чинником, що сприяє успіху *C#* .

Разом з тим *C#* передусім розроблялась як мова програмування прикладного рівня для *CLR* і тому вона залежить, перш за все, від можливостей самої *CLR*. Це стосується, перш за все, системи типів *C#*. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції *CLR*. З розвитком *CLR* від версії 1.1 до 2.0 значно збагатилась і сама *C#*; подібної взаємодії слід чекати і надалі. Проте ця закономірність була порушена з виходом *C#* 3.0, що є розширеннями мови, яке не спирається на розширення платформи *.NET*. *CLR* надає *C#*, як і всім іншим *.NET*-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому *C#*, а проводиться *CLR* для програм, написаних на *C#* точно так, як і це робиться для програм на *VB.NET*, *J#* тощо .

2.2 Декларативна мова розмітки *XAML*

Розшифровується як «*Extensible Application Markup Language*» і вимовляється «*zam-uhl*». *XAML* — це мова розмітки, розроблена *Microsoft* і

використовується для створення інтерфейсів програм. Вона подібна до *HTML*, який визначає вміст веб-сторінки.

Як і інші мови розмітки, *XAML* використовує теги для визначення об’єктів. Теги можуть бути вкладені в інші теги для визначення об'єктів всередині об'єктів. Атрибути об'єкта, такі як ім'я, розмір, форма і колір, визначаються в тегу. Нижче наведено приклад основного тегу *XAML* для кнопки:

<Button x:name = "button" Content = "TechTerms" HorizontalAlignment = "Left" VerticalAlignment = "Top" Margin = "150,300,0,0"/>

Розробники можуть створювати код *XAML* з нуля або використовувати такі програми, як *Microsoft Expression Studio* або *Blend for Visual Studio*, щоб створити код *XAML* за допомогою редактора *WYSIWYG*. *XAML* підтримується будь-якою програмою *Windows*, створеною за допомогою *WPF* або універсальної платформи *Windows*.

При поданні у вигляді тексту файли *XAML* є *XML*-файлами, які зазвичай мають розширення .xaml. Файли можна зберігати в будь-якому кодуванні, що підтримує *XML*, але зазвичай використовується кодування *UTF-8*.

2.3 Платформа *.NET Framework*

.NET Framework — це кероване середовище виконання для *Windows*, яке дозволяє розробникам програмного забезпечення створювати програмне забезпечення однією мовою програмування та бути впевненим, що програма зможе працювати з кодом, написаним іншими мовами. Платформа, яка розроблена для розміщення об’єктного коду незалежно від того, де він зберігається чи виконується, є основною реалізацією технологій *Microsoft .NET*.

Платформа .NET була розроблена для зменшення помилок програмування та підвищення продуктивності за допомогою модульного

підходу до проектування програмного забезпечення. Фреймворк має спільну мову виконання (*CLR*) і бібліотеку класів. *CLR* — це реалізація Microsoft спільної мовної інфраструктури (*CLI*), стандарту, що допомагає різним мовам програмування та бібліотекам працювати разом. *CLR* керує системними службами, такими як пам'ять, виконання потоків, виконання коду, перевірка безпеки коду та компіляція. Бібліотека класів містить перевірений код для повторного використання, який розробники можуть викликати зі своїх власних програм, щоб забезпечити функціональність для таких речей, як введення/виведення файлів, аналіз розширеної мови розмітки (*XML*) і робота з *Windows Forms*.

Інструмент розробки від Microsoft для проектування та розробки програм *.NET* називається *Visual Studio*, і програми зазвичай пишуться на *Visual Basic* (*VB*) або *C#*. *Microsoft Test Framework (MSTest)* можна використовувати для забезпечення якості *(QA)* для додатків *.NET*.

2.4 Інтегроване середовище розробки *Microsoft Visual Studio*

Microsoft Visual Studio — це повне програмне забезпечення (пакет), яке можна використовувати для розробки додатків, будь то бізнес-додатки, персональні програми або компоненти програми, у вигляді консольних програм, програм *Windows* або веб-додатків. *Visual Studio* включає компілятор, пакет *SDK*, інтегроване середовище розробки (*IDE*) і документацію (зазвичай бібліотеку *MSDN*). Компілятори, що входять до пакета *Visual Studio*, включають *Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro* та *Visual SourceSafe*. *Microsoft Visual Studio* можна використовувати для розробки додатків на рідному коді (у формі машинної мови, що працює в *Windows*) або керованому коді (у формі проміжної мови *Microsoft поверх .NET Framework*). Крім того, *Visual Studio* також можна використовувати для розробки додатків *Silverlight*, програм *Windows Mobile* (які працюють поверх *.NET Compact Framework*).

Існує 3 версії *Microsoft Visual Studio*, а саме:

1. *Community*: Це безкоштовна версія, випущена в 2014 році. Усі інші видання платні. Вона містить функції, подібні до професійної версії. Використовуючи цю публікацію, будь-який окремий розробник може розробляти власні безкоштовні або платні програми, такі як програми .*Net*, веб-додатки тощо. У некомерційних організаціях цим питанням можуть користуватися до п’яти користувачів. Його головне призначення — забезпечити підтримку екосистеми (доступ до тисяч розширень) і мов (ви можете кодувати *C#, VB, F#, C++, HTML, JavaScript, Python* тощо).

2. *Professional*: це комерційна версія *Visual Studio*. Вона доступна у *Visual Studio* 2010 і новіших версіях. Вона підтримує редагування *XML* і *XSLT* і включає такі інструменти, як *Server Explorer* та інтеграцію з *Microsoft SQL Server*. *Microsoft* пропонує безкоштовну пробну версію, і після закінчення пробного періоду користувачі повинні заплатити, щоб продовжити її використання. Її головні цілі – забезпечити гнучкість (професійні інструменти розробника для створення будь-якого типу додатків), продуктивність (потужні функції, такі як *CodeLens* підвищують продуктивність команди), спільну роботу (інструменти для швидкого планування проектів, діаграми тощо) та переваги, наприклад, від програмного забезпечення *Microsoft* , плюс *Azure, Pluralsight* тощо..

3. *Enterprise*: це інтегроване наскрізне рішення для команд усіх розмірів із вимогами до якості та масштабу. Microsoft пропонує 90-денну безкоштовну пробну версію цієї версії, після чого користувач повинен заплатити, щоб продовжити її використання. Основна перевага цього видання полягає в тому, що воно має високу масштабованість і надає високоякісне програмне забезпечення.

Має стильний та простий інтерфейс, який легко налаштовується під особисті потреби(рис 2.1).

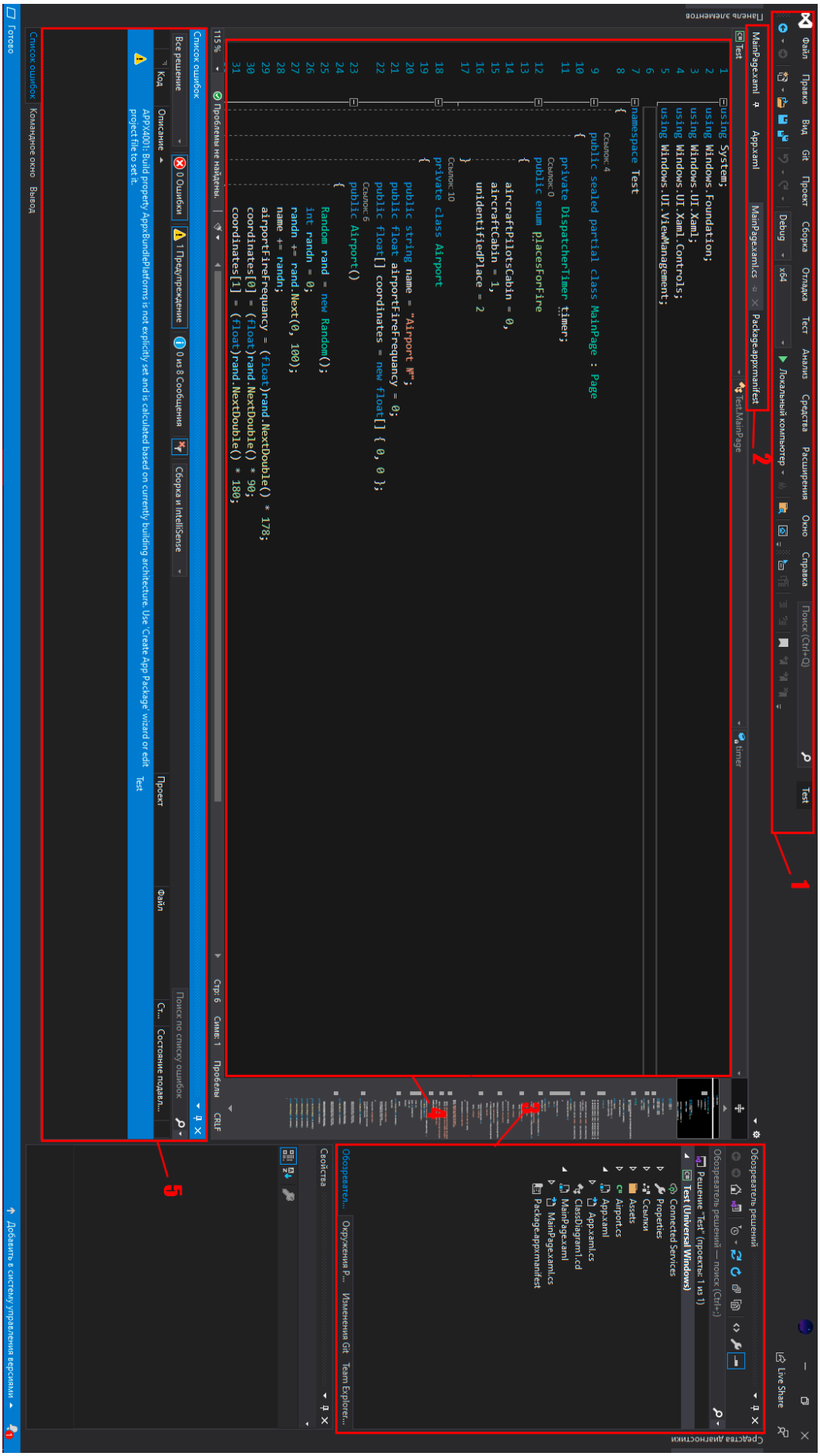

Рис 2.1 Робоче вікно IDE *Visual Studio*

1) список різних меню;
2) відкриті файли за поточну сесію;
3) вікно з ієрархією файлів рішення;
4) вікно з кодом вибраного файлу;
5) вікно з і списком помилок.

2.5 Універсальна платформа *Windows*

Універсальна платформа *Windows* (англ. *Universal Windows Platform*, скор. *UWP*) - платформа, створена *Microsoft* і вперше представлена в *Windows 10*. Метою даної платформи є допомога у створенні універсальних програм, що запускаються як *Windows 10, Windows 10 Mobile і Windows 10 IoT* без зміни у коді. Є підтримка створення таких програм на *C++, C#, VB.NET* і *XAML*. *API* реалізований *C++* і підтримується в *C++, VB.NET, C#, F# і JavaScript*. Розроблена як розширення для *Windows Runtime* (платформи, представленої у *Windows Server 2012* та *Windows 8*), дозволяє запускати програми на різних апаратних платформах. Без проблем взаємодіє з *Visual Studio* (рис 2.2), додаючи модулі для створення програм з графічним інтерфейсом(рис 2.3).

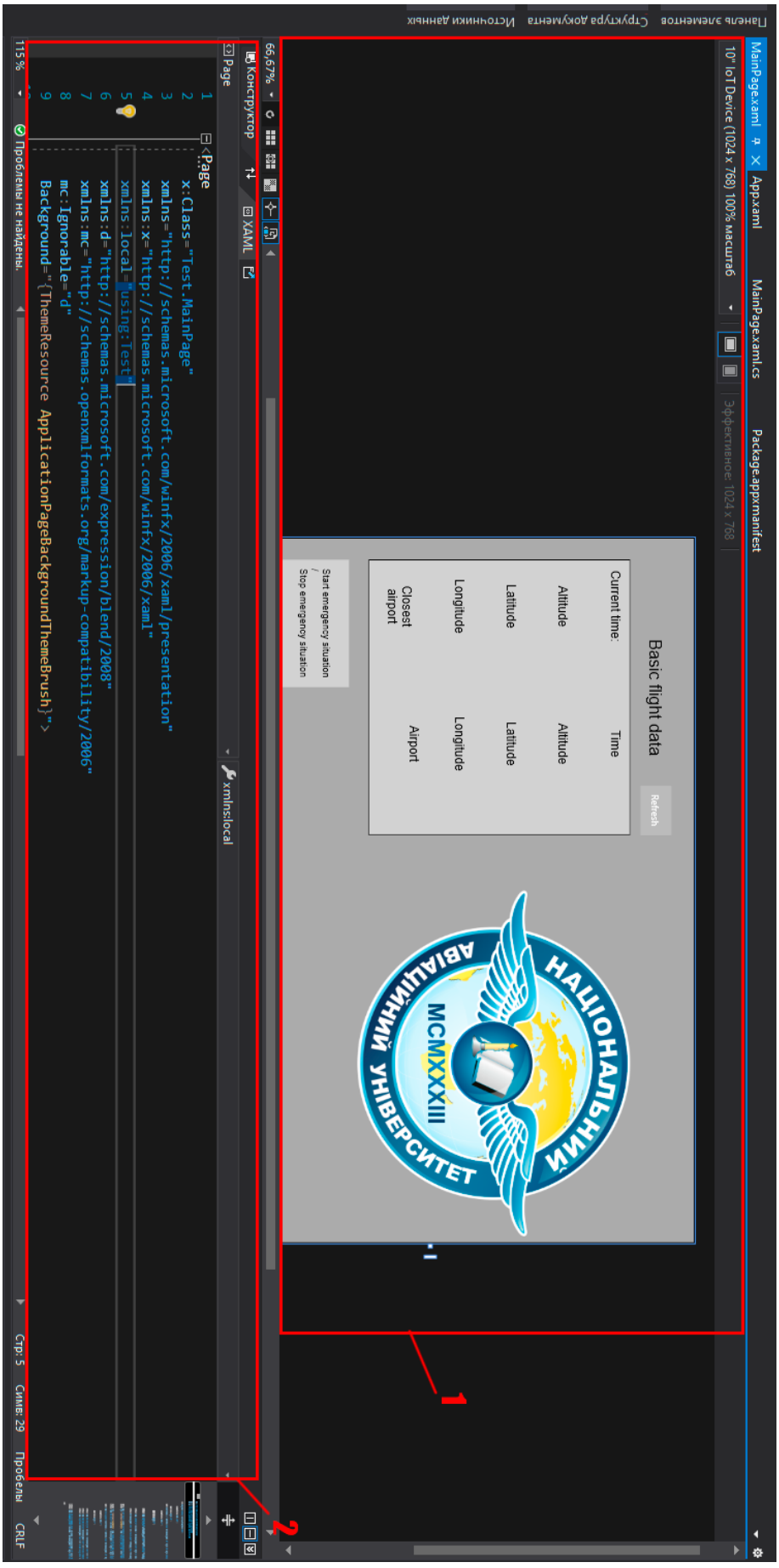

Рис 2.2 Створення додатка за допомоги платформи *UWP* у *Visual Studio*

1) редагування програми за допомоги графічного інтерфейсу;
2) редагування програми за допомоги мови *XAML*.

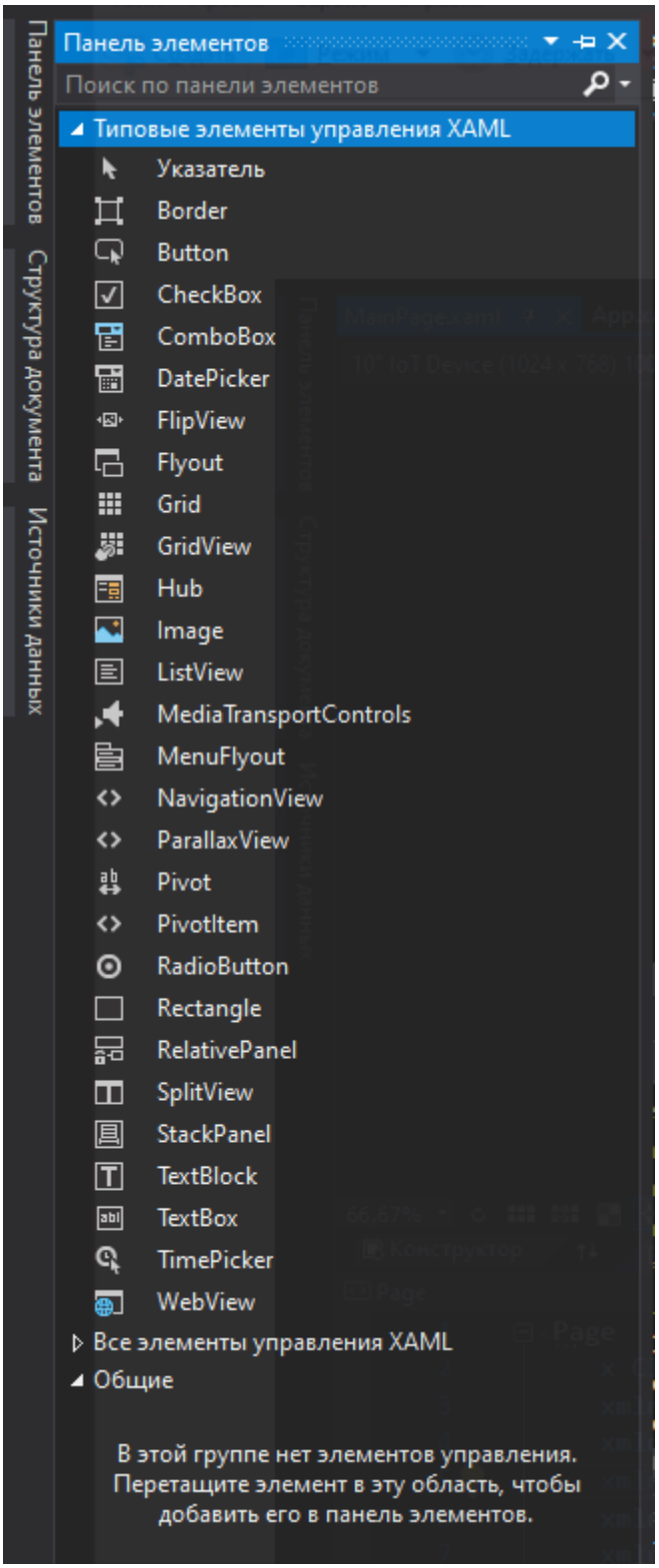

Рис 2.3 Підменю з макетами елементів для створення програм за допомоги платформи *UWP*

2.6 Вимоги до системи

Для стабільної роботи підсистеми, що була розроблена під час дипломного проєктування потрібно мати такі складові:

- Операційна система *Windows Windows 10, Windows 10 IoT, Windows 10 Mobile*
- пакет фреймворка *.NET Framework* 4.8+;

2.7 Висновки до розділу

В цьому розділі було розглянуто засоби та методи, використані для розробки програми для тестування підсистеми підтримки прийняття рішень. Для написання користувацького інтерфейсу було прийняти рішення використовувати платформу *Universal Windows Platform* та підтримувану платформою декларативну мову розмітки *XAML*.

Для написання логіки і алгоритмів підсистеми було прийнято рішення використовувати мову програмування *C#* з використання фреймворку *Microsoft .NET Framework.*

Уся розробка велась у інтегрованому середовищі розробки *Microsoft Visual Studio.*

РОЗДІЛ 3
ФУНКЦІОНУВАННЯ ПІДСИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

У цьому розділі буде розглянуто розробку та використання підсистеми, що була розроблена під час дипломного проєктування та основні її компоненти.

3.1 Розробка алгоритму пошуку альтернатив

Основна частина програмного коду написана на мові програмування *C#*.

Для реалізації та демонстрації роботи алгоритму створення альтернатив при пожежі в салоні повітряного судна потрібно спочатку зімітувати саму пожежу.

Для імітації пожежі у салоні ПС буде згенеровано координати самого ПС, декілька аеропортів. Координати ПС будуть складатися з висоти, широти та довготи. Аеропорти будуть об’єктом класу *Airport* (рис 3.1) і матимуть такі параметри як назва, широта та довгота.

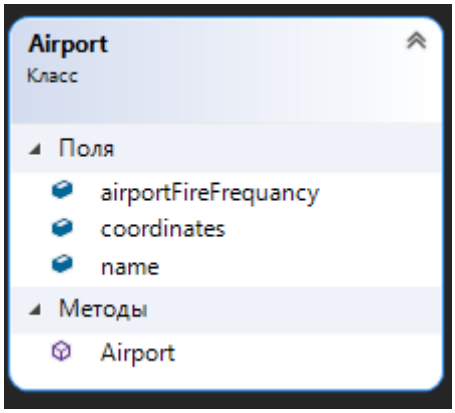

Рис 3.1 Клас *Airport*

Генерація координат являється випадковою. Її реалізовує функція *RefreshCoor():*

```
public void RefreshCoor()
    {
      latitude = (float)rand.NextDouble() * 90;
      longitude = (float)rand.NextDouble() * 180;
      altitude = (int)rand.Next(100, 12000);
      }
```

Координати та назви аеропортів генеруються у конструкторі класу *Airport():*

```
public Airport()
      {
        Random rand = new Random();
        int randn = 0;
        randn += rand.Next(0, 100);
        name += randn;
        airportFireFrequancy = (float)rand.NextDouble() * 178;
        coordinates[0] = (float)rand.NextDouble() * 90;
        coordinates[1] = (float)rand.NextDouble() * 180;
        }
```

Детальніше зімітувати пожежу допоможе її налаштування. Для додаткового налаштування буде доступним присутність вогню та/або диму, їх розташування у ПС.

Нехай ми вказали, що був помічений дим у салоні повітряного судна. Тоді у рекомендованих діях з'явиться пункт *«Put on an oxygen mask. Instruct staff and passengers to put on oxygen masks;»*. Так залежність породить продукційну модель представлення знань. І її правило буде виглядати так:

ЯКЩО ситуація ТО рішення.

Тобто помічений дим це - «ситуація», а *Put on an oxygen mask…* це «рішення».

Можна сказати, що алгоритм аналізує усі доступні дані та поточну ситуацію для конструювання та виведення рекомендованої альтернативи дій.

Програма виводить не лише місце розташування вогню або диму, а і приблизний втрати контролю над ПС. Чим ближче та більше вогню та диму у ПС тим менше залишається часу до повної втрати контролю над літаком. Розрахування часу до цієї події розраховується у методі *Change_Var_For_Alt()*.

3.2 Проєктування графічного інтерфейсу

Користувацький інтерфейс є комунікаційним каналом, за яким здійснюється взаємодія оператора і машини. Тому інтерфейс програми повинен бути простим і зрозумілим.

Інтерфейс програми складається з однієї основної сторінки, яка змінюється внаслідок взаємодії з нею. Початковій варіант вигляду сторінки (рис. 3.2):

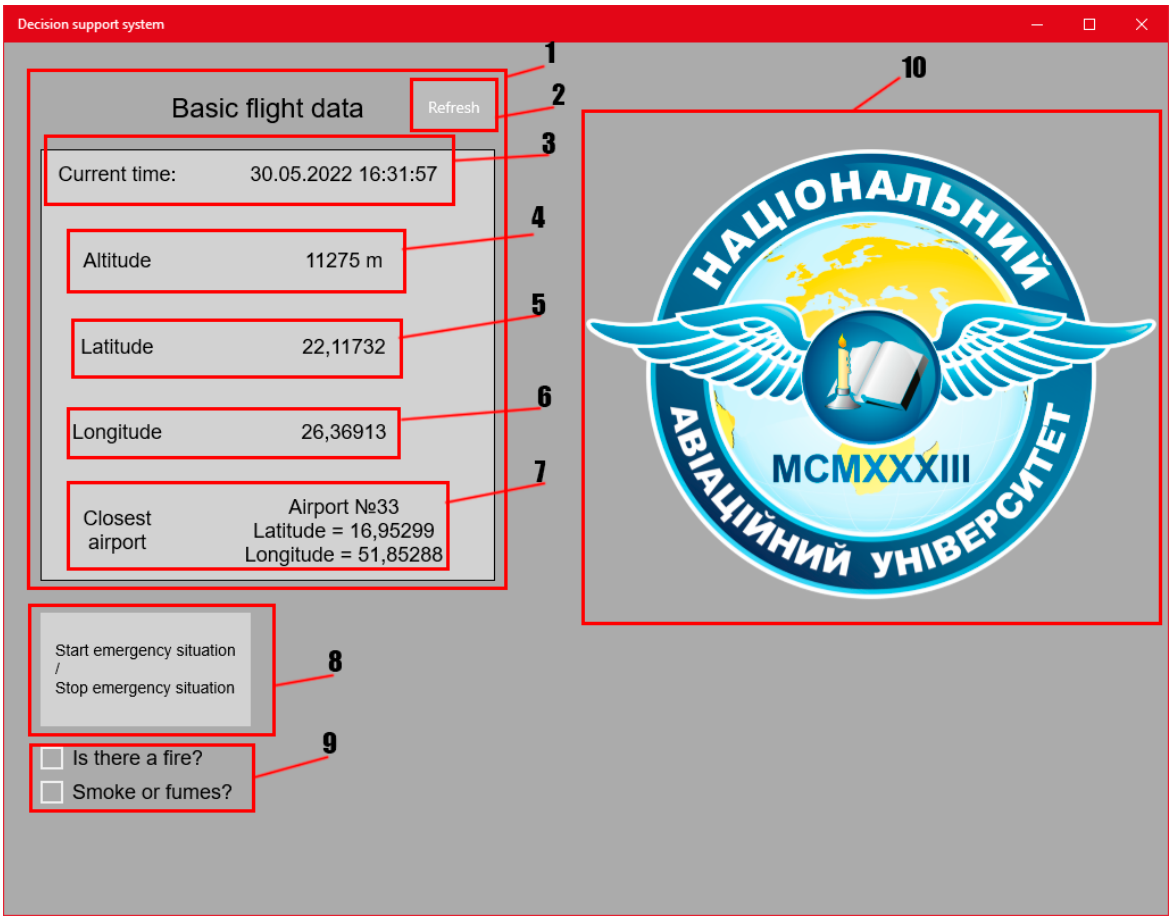

Рис 3.2 Початкове вікно програми

Воно складається з:

1) основні дані польоту;

2) згенерувати дані польоту заново

3) поточний час;

4) висота ПС у метрах;

5) широта;

6) довгота;

7) найближчий аеропорт, його координати;

8) кнопка запуску симуляції пожежі;

9) додаткове налаштування для симуляції(Вибираються для інших результатів роботи алгоритму);

10) емблема.

Частина коду, що відповідає за оновлення основних даних польоту кожну секунду реального часу, описана у функції *Timer_Tick(object sender, object e)* :

```
private void Timer_Tick(object sender, object e)
    {
      Current_Time_Text.Text = DateTime.Now.ToString("G");
      altitude += rand.Next(-2, 2);
      latitude += (float)rand.NextDouble()*0.002f;
      if (latitude>90)
      { latitude -= 180; }
      if (latitude < -90)
      { latitude += 180; }
      longitude += (float)rand.NextDouble()*0.002f;
      if (longitude > 180)
      { longitude -= 360; }
      if (longitude < -180)
      { longitude += 360; }
      Check_Airports();
      Change_Tudes();
      if (etaTimer)
      {
         updateETA();
      }
          }
```

Після запуску симуляції вікно програми зміниться і до нього буде додано нові елементи(рис. 3.3).

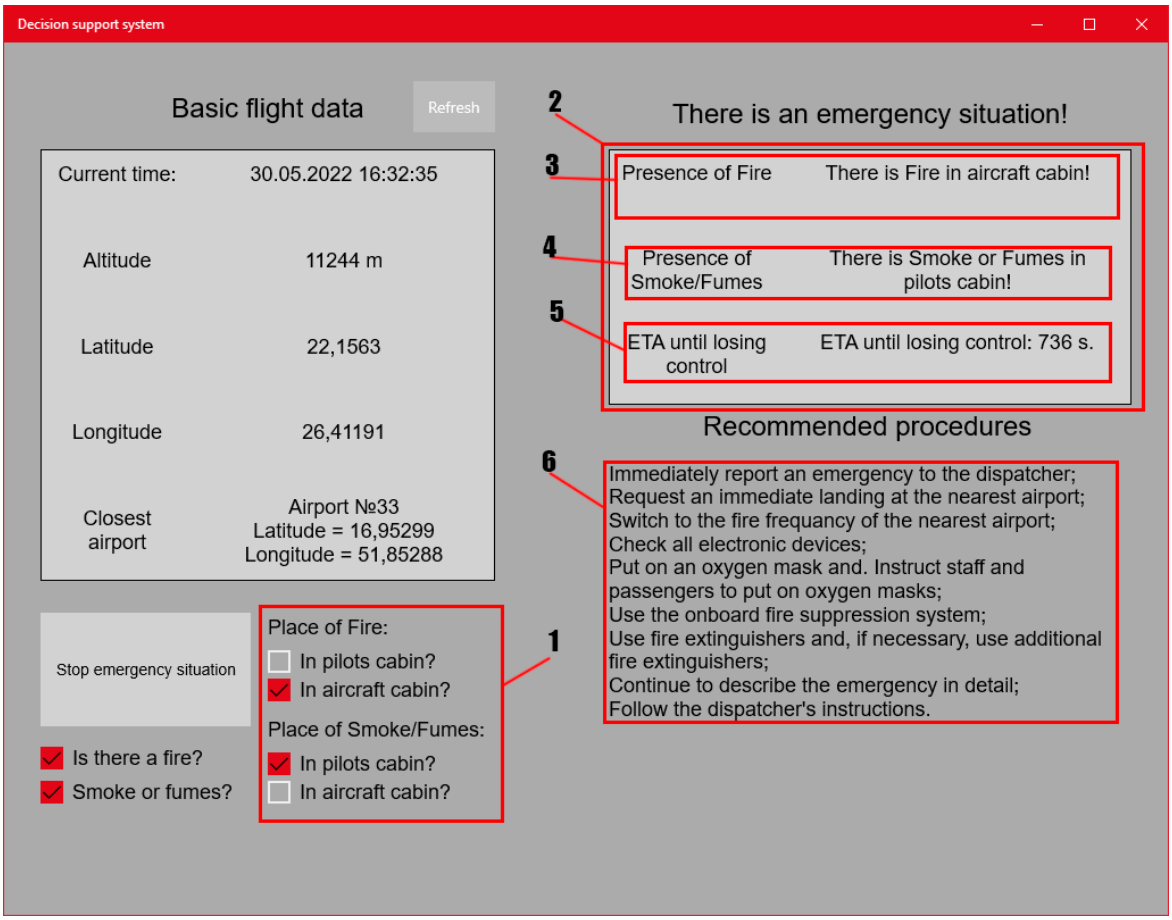

Рис 3.3 Вікно програми після старту симуляції

Нові елементи складаються з:

1) обрані додаткові налаштування симуляції;

2) інформація про поточну ситуацію;

3) місце, де був помічений вогонь(кабіна пілотів, салон ПС, фюзеляж);

4) місце де був помічений дим або випаровування(кабіна пілотів, салон ПС, фюзеляж);

5) приблизний час до утрати контролю над ПС(розташування вогню, диму, випаровування впливає на час);

6) рекомендовані альтернативи в поточній ситуації.

Поява нових елементів у вікні програми контролюється за допомоги методу *Start_Stop_Fire_Click(objects sender, RoutedEventsArgs e)*, який також визиває функції перевірки вибраних додаткових опцій *Check_Checkboxes()*, зміни опису ситуації *Change_Var_For_Alt()* та зміни рекомендованих дій *Change_Recom()* :

```
private void Start_Stop_Fire_Click(object sender, RoutedEventArgs e)
      {
         Check_Checkboxes();
         Change_Var_For_Alt();
         Change_Recom();
         if (Unvisible_Text.Visibility == Visibility.Collapsed)
         {
            Start_Stop_Fire.Content = "Stop emergency situation";
            etaTimer = true;
            Unvisible_Text.Visibility = Visibility.Visible;
            Decision_Border.Visibility = Visibility.Visible;
            Title_for_recommended_actions.Visibility = Visibility.Visible;
            Recommended_procedures.Visibility = Visibility.Visible;
            Logo_Img.Visibility = Visibility.Collapsed;
         }
         else
         {
            Start_Stop_Fire.Content = "Start emergency situation";
            etaTimer = false;
            Unvisible_Text.Visibility = Visibility.Collapsed;
            Decision_Border.Visibility = Visibility.Collapsed;
            Title_for_recommended_actions.Visibility = Visibility.Collapsed;
            Recommended_procedures.Visibility = Visibility.Collapsed;
            Logo_Img.Visibility = Visibility.Visible;
         }
           }
```

Частина програми, яка відповідає за вибір рекомендованих дій, написана в методі *Change_Recom()* :

```
private void Start_Stop_Fire_Click(object sender, RoutedEventArgs e)
      {
         Check_Checkboxes();
         Change_Var_For_Alt();
         Change_Recom();
         if (Unvisible_Text.Visibility == Visibility.Collapsed)
         {
            Start_Stop_Fire.Content = "Stop emergency situation";
            etaTimer = true;
            Unvisible_Text.Visibility = Visibility.Visible;
            Decision_Border.Visibility = Visibility.Visible;
            Title_for_recommended_actions.Visibility = Visibility.Visible;
            Recommended_procedures.Visibility = Visibility.Visible;
            Logo_Img.Visibility = Visibility.Collapsed;
         }
```

```
else
{
    Start_Stop_Fire.Content = "Start emergency situation";
    etaTimer = false;
    Unvisible_Text.Visibility = Visibility.Collapsed;
    Decision_Border.Visibility = Visibility.Collapsed;
    Title_for_recommended_actions.Visibility = Visibility.Collapsed;
    Recommended_procedures.Visibility = Visibility.Collapsed;
    Logo_Img.Visibility = Visibility.Visible;
}
    }
```

3.3 Висновки до розділу

В третьому розділі було розглянуто створення програми до якої входить підсистема підтримки прийняття рішень при виникненні пожежі у салоні повітряного судна.

Було розглянуто основні функції програми:

1) генерація аеропортів;

2) генерація основних даних польоту;

3) розрахування часу до втрати контролю над повітряним судном;

4) використання продукційної моделі представлення знань для виведення рекомендованих альтернатив дій.

Також було представлено та описано графічний інтерфейс програми та його елементів.

ВИСНОВКИ

Під час виконання дипломного проєкту було досліджено технології проєктування та створення систем підтримки прийняття рішень. Були розглянуті типи і види систем підтримки прийняття рішень, їх реалізації та класифікації.

Було додатково досліджено компоненти СППР такі як обладнання, програмне забезпечення, бази даних, бази моделей. Проаналізовано призначення та застосування.

Після було досліджено аварійні ситуації на борту повітряного судна зв’язані з виникненням пожеж. Проаналізовано стандарти дій та процедур, їх послідовність, що виконуються при виникненні пожежі у салоні повітряного судна.

Після чого було розглянуто засоби та методи, використані для розробки підсистеми підтримки прийняття рішень..

- Для написання логіки і алгоритмів підсистеми було прийнято рішення використовувати мову програмування *C#* з використання фреймворку *Microsoft .NET Framework;*
- Уся розробка велась у інтегрованому середовищі розробки *Microsoft Visual Studio;*
- Для написання користувацького інтерфейсу було прийняти рішення використовувати платформу *Universal Windows Platform* та підтримувану платформою декларативну мову розмітки *XAML*.

Далі було розглянуто створену програми до якої входить підсистема підтримки прийняття рішень при виникненні пожежі у салоні повітряного судна.

Було розглянуто основні функції програми:

1) генерація аеропортів;

2) генерація основних даних польоту;

3) розрахування часу до втрати контролю над повітряним судном;

4) використання продукційної моделі представлення знань для виведення рекомендованих альтернатив дій.

Також було представлено та описано графічний інтерфейс програми та його елементів.

У створеній програмі може бути збільшений вибір додаткових опцій для симуляції пожежі у салоні повітряного судна. Також можна отримати знання від експерту для розширення списку правил продукційної моделі представлення знань та заради збільшення кількості альтернатив дій в аварійних ситуаціях зв'язаних з виникненням пожежі у салоні повітряного судна. Варто зауважити, що графічний інтерфейс програми також може бути покращений. До інтерфейсу можливу додати більше функціоналу та видимих даних польоту. Зміна або доопрацювання інтерфейсу також можливе.

Перевагою розробленої підсистеми є її швидкість визначення рекомендованої альтернативи дій та можливість перенести алгоритм на будь-яку мову програмування. Завдяки використанню платформи *UWP* підсистема може вільно адаптуватися під різні пристрої. Програма яка імітує пожежу має простий та зрозумілий інтерфейс для тестування підсистеми підтримки прийняття рішень.

Основні результати дипломного проєкту:

1. Описано системи підтримки прийняття рішень та їх основні компоненти;
2. Описано та застосовано методи та технології для реалізації системи підтримки прийняття рішень;
3. Створено програму імітації пожежі у салоні повітряного судна для тестування підсистеми підтримки прийняття рішень при пожежі у салоні повітряного судна.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Братушка С. М. Системи підтримки прийняття рішень: навчальний посібник для самостійного вивчення дисципліни / [уклад.: С. М. Братушка, С. М. Новак, С. О. Хайлук] ; Державний вищий навчальний заклад “Українська академія банківської справи Національного банку України”. - Суми: ДВНЗ "УАБС НБУ", 2010. - 265 с. ISBN 978-966-8958-56-4
2. Нестеренко О.В. Інтелектуальні системи підтримки прийняття рішень: Навч. посібн. / Нестеренко О.В., Савенков О.І., Фаловський О.О. / За ред. П.І. Бідюка. - Київ: Національна академія управління, 2016. - 188 с. ISBN 978-966-8406-94-2
3. Матеріал з Вікіпедії – вільної енциклопедії / Microsoft Visual Studio: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio (Дата звернення 17.05.2022р).
4. Кодуйте швидше Працюйте ефективніше / Створюйте майбутнє с Visual Studio: https://visualstudio.microsoft.com/ru/ (Дата звернення 17.05.2022).
5. Матеріал з Вікіпедії – вільної енциклопедії / .NET Framework: https://uk.wikipedia.org/wiki/.NET_Framework (Дата звернення 18.05.2022).
6. Крістіан Нейгел та ін. C# 5.0 та платформа .NET 4.5 для професіоналів = Professional C# 5.0 and .NET 4.5. - М.: "Діалектика", 2013. - 1440 с. – ISBN 978-5-8459-1850-5.
7. Матеріал з Вікіпедії – вільної енциклопедії / XAML: https://uk.wikipedia.org/wiki/XAML (Дата звернення 18.05.2022).
8. Andy De George: Огляд XAML (WPF . NET): https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/xaml/?view=netdesktop-6.0&redirectedfrom=MSDN&viewFallbackFrom=netdesktop-5.0 (Дата звернення 18.05.2022).
9. Матеріал з Вікіпедії – вільної енциклопедії / C Sharp: https://uk.wikipedia.org/wiki/C_Sharp (Дата звернення 19.05.2022).
10. Джон Скіт. C# для професіоналів: тонкощі програмування, 3-те видання, новий переклад = C# in Depth, 3rd ed. - М.: «Вільямс», 2014. - 608 с. – ISBN 978-5-8459-1909-0.

11. Матеріал з Вікіпедії – вільної енциклопедії / Universal Windows Platform: https://en.wikipedia.org/wiki/Universal_Windows_Platform (Дата звернення 19.05.2022).

12. John Kennedy: What`s a Universal Windows Platform (UWP) App?: https://docs.microsoft.com/uk-ua/windows/uwp/get-started/universal-application-platform-guide (Дата звернення 20.05.2022).

13. Ларичев О. И., Петровский А. В. Системи підтримки прийняття рішень Сучасний стан та перспективи їх розвитку. // Підсумки науки та техніки. Сер. Технічна кібернетика. — Т.21. М.: ВИНИТИ, 1987, http://www.raai.org/library/papers/Larichev/Larichev_Petrovsky_1987.pdf

14. Сараєв А. Д., Щербіна О. А. Системний аналіз та сучасні інформаційні технології // Праці Кримської Академії наук. - Сімферополь: СОНАТ, 2006. - С. 47-59, https://web.archive.org/web/20070928092729/http://matmodelling.pbnet.ru/Statya_Saraev_Shcherbina.pdf

15. Bonczek R.H., Holsapple C., Whinston A.B. Foundations of Decision Support Systems.- New York: Academic Press, , 1981.