СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

Монографія

За загальною редакцією
д-ра екон. наук, професора В. С. Пономаренка

Харків
ХНЕУ ім. С. Кузнеця
2022

УДК 004(0.034)
С91

Авторський колектив: д-р техн. наук, професор Н. Г. Аксак – розділ 1; д-р пед. наук, професор Л. Е. Гризун, канд. техн. наук, доцент О. В. Щербаков – розділ 2; канд. техн. наук, доцент В. А. Золотарьов – розділ 3; канд. екон. наук, доцент І. О. Золотарьова – розділ 4; д-р техн. наук, професор М. М. Корабльов – розділ 5; канд. техн. наук, доцент В. П. Коцюба – розділ 6; канд. техн. наук, доцент М. Ю. Лосєв – розділ 7; канд. техн. наук, доцент О. В. Фролов – розділ 8; д-р техн. наук, професор С. В. Мінухін – розділ 9.

Рецензенти: завідувач кафедри репрографії Навчально-наукового видавничо-поліграфічного інституту Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", д-р техн. наук, доцент *О. О. Палюх*; професор кафедри комп'ютерних технологій і мехатроніки Харківського національного автомобільно-дорожнього університету, д-р техн. наук *О. П. Алексієв*; професор кафедри комп'ютерних наук та інформаційних технологій Української академії друкарства, д-р техн. наук *О. В. Тимченко*.

Рекомендовано до видання рішенням ученої ради Харківського національного економічного університету імені Семена Кузнеця.
Протокол № 4 від 25.05.2022 р.

Самостійне електронне текстове мережеве видання

Сучасні інформаційні технології та системи [Електронний ресурс] : монографія / Н. Г. Аксак, Л. Е. Гризун, О. В. Щербаков та ін. ; за заг. ред. д-ра екон. наук, професора В. С. Пономаренка. – Харків : ХНЕУ ім. С. Кузнеця, 2022. – 271 с.
С91
ISBN 978-966-676-850-9

Розглянуто сучасний стан та перспективи розвитку сучасних інформаційних технологій і систем різних видів та різного прикладного характеру.

Монографія становить інтерес як для фахівців, сферу діяльності яких безпосередньо пов'язано з розробленням прикладних інформаційних технологій і систем, так і для більш широкого кола фахівців. Вона буде корисною викладачам, аспірантам і студентам, що спеціалізуються в галузі інформаційних технологій, і всім, хто серйозно цікавиться проблемами інформаційного суспільства.

УДК 004(0.034)

ISBN 978-966-676-850-9

Вступ

Сучасні інформаційні технології й системи використовують для створення електронних ринків, де можна відстежити та проконтролювати оплати. Крім того, вони впливають на створення додаткових робочих місць і розширюють можливості для ухвалення рішення в медичній, освітній та інших сферах.

Ускладнення економічних процесів потребує нових підходів до їхнього дослідження на основі принципів самоорганізації систем.

У монографії наведено результати наукових досліджень у галузі інформаційних технологій і систем, презентовані на Міжнародній науково-практичній конференції "Інформаційні технології та системи", що відбулася 14 – 15 квітня 2022 року на базі Харківського національного економічного університету імені Семена Кузнеця. У ній відображено найбільш цікаві результати за найсучаснішими науковими напрямами застосування, проєктування та визначення проблем і перспектив інформаційних технологій у технічних системах тощо.

У розділі 1 монографії розроблено проєкт адаптивної системи електронного навчання, заснованого на багатоагентному підході. Систему реалізовано на основі розподіленої трирівневої архітектури, що дозволяє рекомендувати навчальні об'єкти, адаптовані до профілю студента, відповідно до його характеристик та уподобань. Реалізовано основні функції щодо рекомендацій навчальних шляхів, відповідно до рівня знань студентів та стилів навчання, серед студентів і викладачів. Розроблено застосунок мовою C++ із графічним інтерфейсом, що є інтегрованим середовищем для розроблення багатоагентних систем, яке здатне взаємодіяти з іншими застосунками в мережі. Реалізовано демоверсію багатоагентної системи е-навчання, для роботи якої використано технологію CORBA. Така система може бути інтегрованою з будь-якою системою управління навчанням.

У розділі 2 монографії розглянуто шляхи розв'язання науково-прикладної проблеми навігації в закритих приміщеннях. Запропоновано алгоритмічні й інтерфейсні рішення для проєктування та реалізації інформаційно-навігаційної системи університету. Висвітлено особливості архітектури означеної системи та основні етапи її розроблення в контексті запропонованих рішень. Схарактеризовано функціонал реалізованої системи. Установлено, що у процесі проєктування вдалося подолати

основні обмеження, притаманні аналогічним системам, що реалізують навігацію у приміщенні. Проаналізовано результати апробації системи в освітній практиці ХНЕУ ім. С. Кузнеця. Отримані під час апробації відгуки від користувачів засвідчують доцільність розроблення та застосування системи для навігації в університеті.

У розділі 3 монографії, згідно з рекомендаціями CERT-UA, розглянуто основні рекомендації із захисту інформації в інформаційно-комунікаційних мережах. Наведено перелік категорій і типів кіберінцидентів, відповідно до рекомендацій Європейської агенції з кібербезпеки, призначений для впровадження єдиної таксономії як інструменту для обміну інформацією щодо кіберінцидентів. Описано категорії шпигунського програмного забезпечення та запропоновано методи захисту від них. Відповідно до методики OWASP Top-10, визначено основні загрози вебресурсам, наведено рекомендації з виявлення конкретних вразливостей і запропоновано методи захисту від них. Розроблено рекомендації щодо організації безпечної віддаленої роботи установи.

У розділі 4 монографії зазначено, що сфера штучного інтелекту нині переживає невпинне зростання, на етапах досліджень і розроблень виникає велика кількість моделей у різних галузях, включно з інформаційними технологіями, фінансами та інженерією. Розглянуто методики, спрямовані на управління проєктами. Аналіз значною мірою зосереджено на гібридних системах, які становлять обчислювальні моделі методів змішаного навчання. Нині ці моделі перебувають на ранній стадії розроблення. Надано класифікацію всіх сфер управління проєктами та методи навчання, які використовують у кожній із них, презентуючи коротке дослідження різних методів штучного інтелекту, які використовують сьогодні, та галузей управління проєктами, у яких застосовують агенти.

У розділі 5 монографії розроблено мультиагентну систему підтримки ухвалення клінічних рішень (СПУКР) для діагностики захворювань, яка складається із трьох модулів високого рівня: інтерфейсного модуля, модуля виконання та модуля знань. Систему реалізовано на основі синергізму між мультиагентними системами (МАС) та міркуваннями на основі прецедентів, що дозволяє підвищити ефективність упровадження механізмів навчання й адаптації до особливостей пацієнтів. Агенти МАС є інтелектуальними та становлять складні програмні системи, які мають здатність пристосовуватися до певної обставини, що викликає зміну їхньої поведінки або характеристик, із метою забезпечення здатності

до адаптації та розв'язання поставленої перед ними проблеми. Було проведено тестові дослідження для визначення діагнозу захворювання серця, які показали здатність пропонованої СПУКР вирішувати поставлені завдання та можливість її використання для ухвалення рішень у реальних умовах.

У розділі 6 монографії проаналізовано принципи управління комп'ютерними мережами, особливості побудови й режими роботи систем моніторингу, аналізу й ідентифікації станів мережі в реальному часі. Розглянуто варіанти застосування, особливості функціонування, переваги й недоліки найбільш уживаних програмних продуктів моніторингу та діагностування устаткування IP-мереж. Обґрунтовано, що тільки комплексне застосування уніфікованого програмного забезпечення, апаратних і технічних рішень дозволить здійснювати контроль та проводити автоматичний моніторинг мережевих пристроїв та оперативне усунення виявлених несправностей і виправлення неполадок у комп'ютерній мережі.

У розділі 7 монографії здійснено аналіз особливостей функціонування системи управління сучасних інформаційних мереж, розглянуто організацію процесу контролю й управління обміну даними в мережах із комутацією пакетів, переваги та недоліки алгоритмів маршрутизації пакетів у мережах, а також можливості ефективного моделювання процесу мультиоб'єктного управління ресурсами адаптивної комп'ютерної мережі.

У розділі 8 монографії розглянуто асимптотично-оптимальну лінійну інтерполяцію плоских кривих, яка задовольняє умові близької до мінімальної кількості ланок ламаної. Було запропоновано алгоритми обчислення значень послідовності вузлів апроксимації на основі чисельного інтегрування функції-регулятора з подальшою лінійною та сплайновою інтерполяцією її значень. Обґрунтовано методику оцінювання результатів моделювання апроксимації реальних кривих, що ґрунтується на статистичному опрацюванні рядів відносних похибок ланок ламаної. Здійснено моделювання апроксимації реальних кривих і досліджено вплив на показники розподілу похибок кількісної характеристики ступеня дискретизації інтегральної функції-регулятора вузлів, залежно від методу інтерполяції значень інтегральної функції. Досліджено вплив кількісної характеристики ступеня дискретизації інтегральної функції-регулятора вузлів інтерполяції на показники розподілу похибок, а також оптимізацію

параметра функції-регулятора вузлів для випадку кривих із наявними точками перегину.

У розділі 9 розглянуто питання щодо основних інструментів та програмного забезпечення виконання завдань машинного навчання, що має можливість виконуватися на розподілених системах, якими вибрано фреймворк Apache Spark. Розглянуто середовище, у якому він працює, та типові приклади застосування на кластері Azure HDInsight. Проаналізовано механізми розподілу ресурсів у кластерах під управлінням програмної платформи Apache Spark та досліджено наявні параметри конфігурації, що впливають на продуктивність роботи кластера. Для оцінювання продуктивності кластерів загального призначення розглянуто наявні тестові набори для проведення експериментальних досліджень, зокрема для тестування моделей машинного навчання. Для проведення експериментів вибрано бенчмарк Spark-Perf як тестовий набір із розширеними можливостями щодо моделей та алгоритмів машинного навчання. Розроблено методичне забезпечення для налаштування програмного забезпечення для різних конфігурацій кластерів Azure.

Розділ 1
Багатоагентна система електронного навчання

1.1. Вступ і формулювання завдання

Пандемія Covid-19 швидко поширюється світом і не має жодних ознак припинення. У всьому світі станом за даними ВООЗ на 17:48 CE 30 листопада 2021 року було зареєстровано 261 435 768 підтверджених випадків COVID-19, зокрема 5 207 634 померлих. Станом на 28 листопада 2021 року було введено 7 772 799 316 доз вакцини.

Більше того, епідемія вплинула на всі сфери економіки та політики, а також на соціальне життя всіх людей у всьому світі. Соціальне відокремлення й ізоляція, необхідні для уникнення поширення епідемії, сформували нові звички та види діяльності через інтернет. Освіта є однією зі сфер, яка має великий вплив, оскільки вона є необхідною багатьом людям і в кожній окремій країні. Студенти є обмеженими у своїх можливостях ходити до закладів освіти та брати участь у колективних заходах, тому навчання переважно здійснюють за методами електронного або змішаного навчання. Такий підхід надав університетам ефективний метод навчання в поєднанні електронного навчання із традиційним навчанням, залежно від змісту та ресурсів, які університети можуть надати для курсів електронного навчання на різних рівнях. Сприятливі можливості для освітньої діяльності містять інтерактивні дії в режимі онлайн, а також зберігання й оцінювання даних, які можна здійснити на інтернет-платформі, щоб студенти та викладачі могли виконувати свої завдання.

Термін "електронне навчання" охоплює широкий набір застосувань і процесів. E-learning (е-навчання) – це загальний термін, що описує будь-який тип навчання, що залежить або поліпшується онлайн-спілкуванням із використанням новітніх інформаційно-комунікаційних технологій. Сфера такого навчання є дуже широкою. Ця тема є однаково новою для виробників курсів, постачальників технологій і кінцевих користувачів (тобто для тих, хто навчаються), вона ще не знайшла спільної думки та ринкової позиції.

Протягом останніх кількох років електронному навчанню приділяли надто багато уваги, особливо провідні постачальники технологій. Однак у міру розвитку технологій та бізнесу розвивається й термінологія.

Е-навчання обіцяє надати нові потужні інструменти для підвищення компетентностей і можливостей, швидкості та продуктивності, незалежно від геолокації. Подібно до того, як розвиток ІТ докорінно змінив природу спілкування та принцип виконання роботи в багатьох сферах діяльності, так і поява технологій е-навчання принципово змінює характер навчання. Людей все більше заохочують навчатися самостійно та вивчати лише те, що їм дійсно потрібно для оптимального виконання своїх завдань.

Основна частина ефективного е-навчання є інтерактивною. Оскільки людина також має володіти навичками в саморегуляції, проте здебільшого є необхідність у зверненні до викладача. Е-навчання може задовольнити потреби в навчанні 24 години на добу, 7 днів на тиждень, водночас традиційні навчальні заходи не мають такої можливості. Електронне навчання дозволяє здійснювати навчання для окремих осіб у зручний для них спосіб.

Отже, сучасне використання терміна "електронне навчання" має різні значення в різних контекстах.

У секторах вищої освіти та бізнесу це стосується гнучкого доставлення вмісту та програм через інтернет. Тут е-навчання можна охарактеризувати як таке, що керується ідеями підвищення продуктивності та зниження витрат, особливо в умовах глобалізованого бізнес-середовища з акцентом на доставлення контенту та управління онлайн-курсами.

У контексті широкої освітньої спільноти термін "електронне навчання" охоплює різноманітні практики, технології та теоретичні положення. Його не лише зосереджено на онлайн-контекстах, а воно містить повний спектр комп'ютерних навчальних платформ і методів доставлення, жанрів, форматів, як-от мультимедіа, освітнє програмування, моделювання, ігри та використання нових медіа на стаціонарних і мобільних платформах.

Електронне навчання розвивається разом зі Всесвітньою мережею загалом. Сьогодні електронне навчання переважно проходить у формі онлайн-курсів. Як наслідок, провідна технологія навчання, що використовують сьогодні, є типом системи, яка організовує та проводить

онлайн-курси, – це система управління навчанням (СУН). Ця частина програмного забезпечення стала поширеною в середовищі навчання. Такі компанії, як WebCT, Blackboard і Desire2Learn, установили продукти в тисячах університетів та коледжів і ними користуються десятки тисяч викладачів та студентів. Система управління навчанням бере навчальний у зміст та організовує його стандартним способом як курс, розподілений на модулі й уроки, підкріплений вікторинами, тестами та дискусіями, а також у багатьох сучасних системах, інтегрованих в інформаційну систему студентів коледжу чи університету. Традиційні теорії дистанційного навчання, наприклад трансакційної дистанції, описані Майклом Г. Муром, було адаптовано для онлайнового світу. Контент, організований за цією традиційною моделлю, подано або повністю в режимі онлайн, або разом із більш традиційними семінарами для студентів на чолі з викладачем, дотримуючись визначеної навчальної програми, яку необхідно виконувати в заздалегідь визначеному темпі.

З іншого боку, що буде відбуватися, якщо онлайн-навчання перестане бути засобом, а стане більше схожим на платформу? У цьому разі програма для електронного навчання почне нагадувати інструмент для ведення блогів, тобто один вузол у мережі, пов'язаний з іншими вузлами та службами створення вмісту, якими користуються інші студенти. Це стає персональним навчальним центром, де вміст повторно використовують і перемішують, відповідно до власних потреб та інтересів студента, або сукупністю взаємодійних програм – середовищем, а не системою. Буде інструментом особистого портфоліо: студенти будуть мати своє особисте місце для створення та демонстрації власних робіт.

Такий підхід до навчання означає, що навчальний уміст створюють і поширюють зовсім іншим способом. Контент електронного навчання агрегують студенти, використовуючи свою особисту RSS-стрічку або подібну програму. Звідти його переробляють для використання іншим студентом.

Інтелектуальні системи навчання дозволяють студентам здобути освітні ресурси, які відповідають їхнім вимогам чи інтересам. Використання мультиагентного підходу й застосування методів машинного навчання, також може поліпшити систему е-навчання та зробити її більш інтелектуальною. Однак системам електронного навчання, що використовують мультиагентний підхід, бракує включення всіх бажаних характеристик

(спільності, гнучкості, адаптивності, інтерактивності та безпеки) в одну й ту саму систему.

Тому актуальним є проблема створення адаптивної системи е-навчання, яка була б спроможною відповідати потребам кожного студента. Якщо адаптивне та динамічне навчання поєднують із різними стилями навчання, то такий підхід буде найбільш ефективним і поліпшить результативність та знання студента, порівняно із традиційним навчанням.

Для розв'язання цієї проблеми пропонують багатоагентну систему електронного навчання, що здатна надавати кожному студентові персоналізований контент [1].

1.2. Фактори, що впливають на застосування електронного навчання

Системи електронного та змішаного навчання використовують в університетах, і вони застосовують студентам як онлайн, так і традиційні методи навчання. Курси можуть бути в онлайновій, традиційній або змішаній формі, залежно від кожного закладу освіти. Зазвичай студенти часто потребують допомоги, щоб дістати інформацію про те, яка форма підходить для їхніх навичок та здібностей. Зазвичай, це вирішує ректорат. Однак іноді не дуже добре виходить. Одним із підходів для розв'язання цієї проблеми може бути використання аналізу наявних даних. Тому можна виділити чотири фактори впливу, включно з деякими їхніми рисами: студентів, викладачів, закладу освіти та навчальної дисципліни (рис. 1.1).

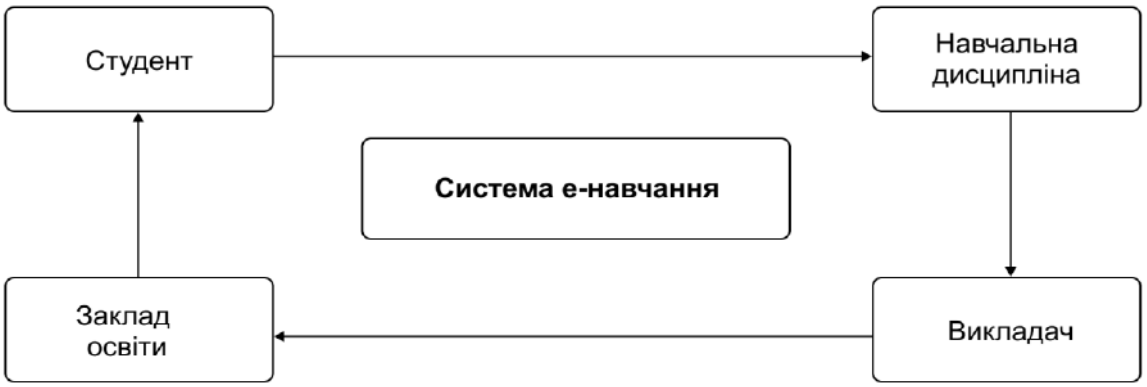


Рис. 1.1. **Фактори впливу**

Вибірки даних зібрано від студентів, які вже закінчили курси, може бути використано для розв'язання двох завдань: по-перше, для оцінювання впливу факторів на застосування електронного навчання на основі кластеризації за допомогою мапи Когонена [19]; по-друге, для розроблення моделі прогнозування, щоб допомогти студентам у виборі відповідного методу навчання [26].

Риси факторів впливу, спрямовані на оцінювання системи електронного навчання, наведено в табл. 1.1.

Таблиця 1.1

Риси факторів впливу

Фактори	Риси	Опис
1	2	3
Студент	x_1	Рік народження
	x_2	Стать
	x_3	Місце розташування
	x_4	Рік навчання
	x_5	Галузь знань
	x_6	Середній бал
Викладач	x_7	Кваліфікація викладача
	x_8	Зміст освітніх компонентів
	x_9	Устаткування, якість інтернету
	x_{10}	Доступність для учасників освітнього процесу
Заклад освіти	x_{11}	Рівень вищої освіти
	x_{12}	Строк навчання за освітньою програмою
	x_{13}	Форми здобуття освіти на ОП
	x_{14}	Матеріально-технічне забезпечення
	x_{15}	Мова (мови) викладання
	x_{16}	Умови для реалізації права на освіту особам з особливими освітніми потребами
Навчальна дисципліна	x_{17}	Придатність курсу до електронного навчання
	x_{18}	Навчально-методичне забезпечення
	x_{19}	Забезпечення якості освітнього компонента

Закінчення табл. 1.1

1	2	3
	x_{20}	Форми та процедури проведення контрольних заходів
	x_{21}	Форми й методи навчання

Для вимірювання впливу факторів, які впливають на систему електронного навчання, вибрано метод кластеризації з використанням мапи Когонена [19]. Вихідні дані, визначені в результаті оцінювання системи студентами, перетворюють на множину зразків $X = \{X^1,...,X^M\}$, де $X^j = \{x_1^j,...,x_n^j\}^T$ – вхідний вектор, $j = \overline{1,M}$, M – кількість опитуваних студентів, $n = 1,...,21$. Мережа складається з одного шару, має n вхідних нейронів, що відповідають координатам аналізованих зразків, і k^2 вихідних нейронів, що становлять квадратну решітку розміром $k \times k$, $W = (w_{jk})_{j=\overline{1,M}}^{k=\overline{1,n}}$ – матриця вагових коефіцієнтів, заданих випадковим способом, визначає положення нейронів на площині.

Уведено вектор параметрів $E = (\varepsilon_1,...,\varepsilon_M)$, який буде характеризувати віддаленість розташування j-го нейрона на площині (на стадії ініціалізації $\varepsilon_j = 0$).

Алгоритм навчання. На вхід послідовно подають вхідні зразки, для кожного з яких визначають номер нейрона-переможця $v(X^i)$ – нейрона, розташованого найближче до поточного зразка:

$$v(X^i) = \arg\min_j y_j^i,\ j = \overline{1,M}, \tag{1.1}$$

де $y_j^i = \left\| W_j - X^i \right\|^2$;

W_j – j-й стовпець матриці вагових коефіцієнтів.

Далі виконують корекцію вагових коефіцієнтів за такою формулою:

$$\begin{gathered} w_{c,k} \sim w_{c,k} + \alpha(x_k^i - w_{c,k}); \\ c = \overline{\nu(X^i) - r, \nu(X^i) + r}, \end{gathered} \tag{1.2}$$

де α – параметр швидкості навчання;

$\nu(X^i)$ – номер нейрона переможця.

Для нейрона-переможця параметр ε_v збільшується на величину, що дорівнює евклідовій відстані між цими нейроном та зразком, поданим на вхід мережі:

$$\varepsilon_v \sim \varepsilon_v + \left\| W_v - X^i \right\|^2. \qquad (1.3)$$

Для візуалізації було вибрано прямокутну самоорганізовану мапу Когонена розміром $k \times k$, побудовану шляхом навчання мережі Когонена з використанням модифікованого алгоритму регуляризації з параметром $\alpha = 0{,}1$.

Показник значущості, що характеризує, наскільки вихідне поле залежить від кожного із вхідного, розраховують після побудови моделі кластеризації (рис. 1.2).

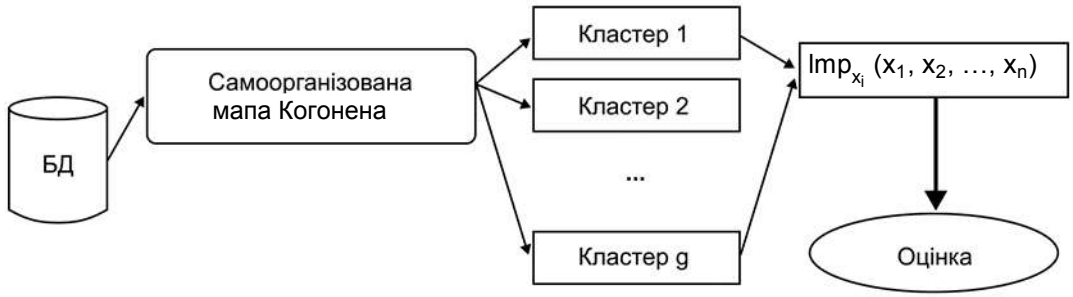


Рис. 1.2. **Модель кластеризації**

Якщо є *n* вхідних атрибутів, тоді формула для розрахунку значущості кожної риси має такий вигляд:

$$\mathrm{Imp}_{x_i}(x_1, x_2, \ldots, x_n) = \left(\frac{\sum_{j=1}^{M} x_1^j}{M}, \frac{\sum_{j=1}^{M} x_2^j}{M}, \ldots, \frac{\sum_{j=1}^{M} x_n^j}{M} \right). \qquad (1.4)$$

Модель вибору методу навчання з використанням оцінки (1.3) показано на рис. 1.3. За результатами вимірювань можна вдосконалювати систему електронного навчання та надавати студентам своєчасну інформацію.

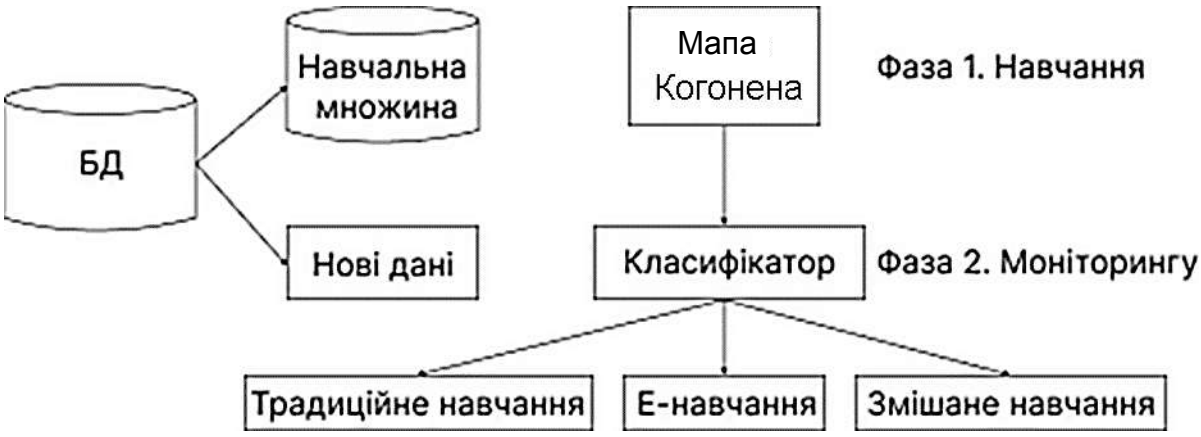


Рис. 1.3. **Модель вибору методу навчання**

Вибір методу навчання здійснюють у два етапи. На першій фазі навчальну множину використовують для побудови моделі кластеризації та визначення класифікатора. Навчальну множину формують на основі визначених даних від студентів, які закінчили курс та оцінили систему.

На другій фазі класифікатор використовують для прогнозування типу навчання – електронного, традиційного або змішаного. Це, з одного боку, може допомогти студенту вибрати відповідний метод вивчення курсу, з іншого – допоможе закладам освіти краще підтримувати студентів, а також урізноманітнити форми навчання та поліпшити якість.

У завданні кластеризації характеристики даних завжди впливають на остаточний результат. Залежно від мети, різні фактори по-різному впливають на різні системи. Отже, кількість і значення атрибутів є одним із факторів, що визначають результати моделі. Питання про те, як найкраще закінчити курс, завжди постає перед кожним студентом. Це залежить від багатьох факторів впливу, але таку інформацію не завжди повністю надають студентам. Тому модель прогнозує відповідний метод для предмета, який студенти готуються вибрати. Крім того, використовуючи алгоритми класифікації, але на різних наборах даних, можна визначити фактори, які впливають на процес навчання в університеті.

1.3. Модель багатоагентної системи електронного навчання

Запропоновану систему електронного навчання призначено для того, щоб рекомендувати персоналізований навчальний ресурс на основі

характеристик та уподобань студентів. Основними компонентами багатоагентної системи електронного навчання є модель умісту, модель студента та модель адаптації. Ці компоненти взаємодіють між собою, забезпечуючи індивідуальну адаптацію системи, відповідно до моделі студента, яка збирає всю інформацію про нього, та моделі вмісту, що описує навчальні ресурси та їхню структуру. Система може вибирати найбільш прийнятну структуру змісту для кожного студента, беручи до уваги його рівень знань, стиль навчання та стиль викладання.

Модель умісту охоплює навчальні ресурси, структуровані таким способом, що полегшує процес адаптації під час використання інтелектуальних систем е-навчання та складається із чотирьох шарів, поданих у вигляді ієрархічної мережі: навчальна дисципліна, змістовий модуль, тема та навчальна одиниця (НО) (рис. 1.4).

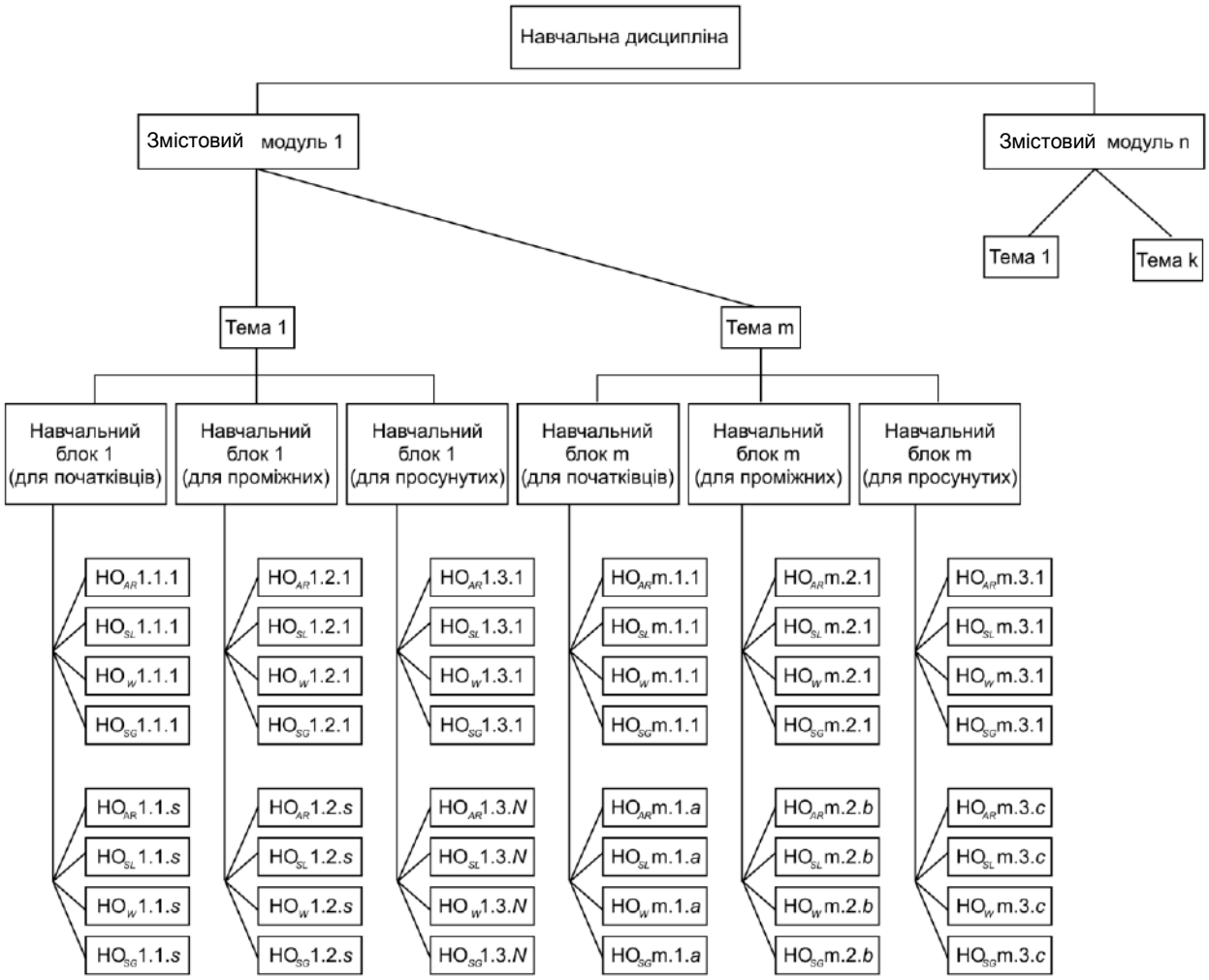


Рис. 1.4. **Організація курсу**

Шар "Навчальна дисципліна" розташовано у верхній частині, він презентує корінь моделі вмісту. Під коренем є кілька змістових модулів, кожний із яких стосується певної теми.

Третій шар "Тема" містить навчальні блоки, фрагментовані за необхідним обсягом та рівнем складності для початківців, проміжних або просунутих студентів. Кожний навчальний блок складається з навчальних одиниць, кожну з яких надано в різних варіантах, відповідно до стилю викладання.

Четвертий рівень містить НО, який становить найменший елемент та має бути добре проіндексованим для наступного пошуку або повторного використання навчального матеріалу. Через це всі елементи мають метадані – інформацію про сам компонент (заголовок, автора, рівень освіти, рівень складності, тип інтерактивності тощо). Застосування метаданих дозволяє зробити більш простим пошук та розпізнавання навчального матеріалу.

У табл. 1.2 наведено метадані, які може бути використано для кожного елементу, описаного на рис. 1.4.

Таблиця 1.2

Метадані компонента навчальної дисципліни

Компоненти	Метадані
Навчальна дисципліна	Ім'я, рівень складності, опис курсу, мова, тривалість
Змістовий модуль	Назва, рівень розділу, передумова, опис розділу
Тема	Ім'я, мета, передумова, рівень об'єкта, рівень складності
НО	ID навчальної одиниці, назва, передумова, остання зміна, стиль навчання, ключове слово, рівень інтерактивності, тип формату, тип зв'язку

Модель студента. Адаптивні системи зазвичай використовують різні дані користувача, залежно від мети адаптації. У цьому підході користувачем є студент, тому система мусить мати можливість адаптувати навчальний досвід, відповідно до потреб студентів. Модель студента містить історію навчання, прогрес студента в поточному навчальному

семестрі, стиль навчання та іншу інформацію про студентів. Модель ґрунтується на п'яти вимірах:

1. Загальна особиста інформація: ім'я, вік, стать – усе те, що залишається незмінним протягом усього використання системи.

2. Особисті вподобання, наприклад, бажаний домен, мова, тип носія, тощо.

3. Стилі навчання визначають, відповідно до моделі Фельдера – Сільвермана (ФС). Оскільки можливості e-learning дозволяють персоналізувати освіту для кожного студента або навчати студентів з однаковими вподобаннями, модель ФС пропонує диференціювати вподобання студентів за чотири шкалами та надає рекомендації викладачам, як правильно формувати навчальні курси, щоб успішно навчати студентів із протилежними вподобаннями за кожною зі шкал.

4. Інформація про статус особистого навчання, яка поєднує рівень знань, історію та цілі навчання. Цю інформацію постійно оновлюють у процесі навчання для адаптації системи.

На рис. 1.5 наведено онтологію, яку використовують для визначення цих вимірів.

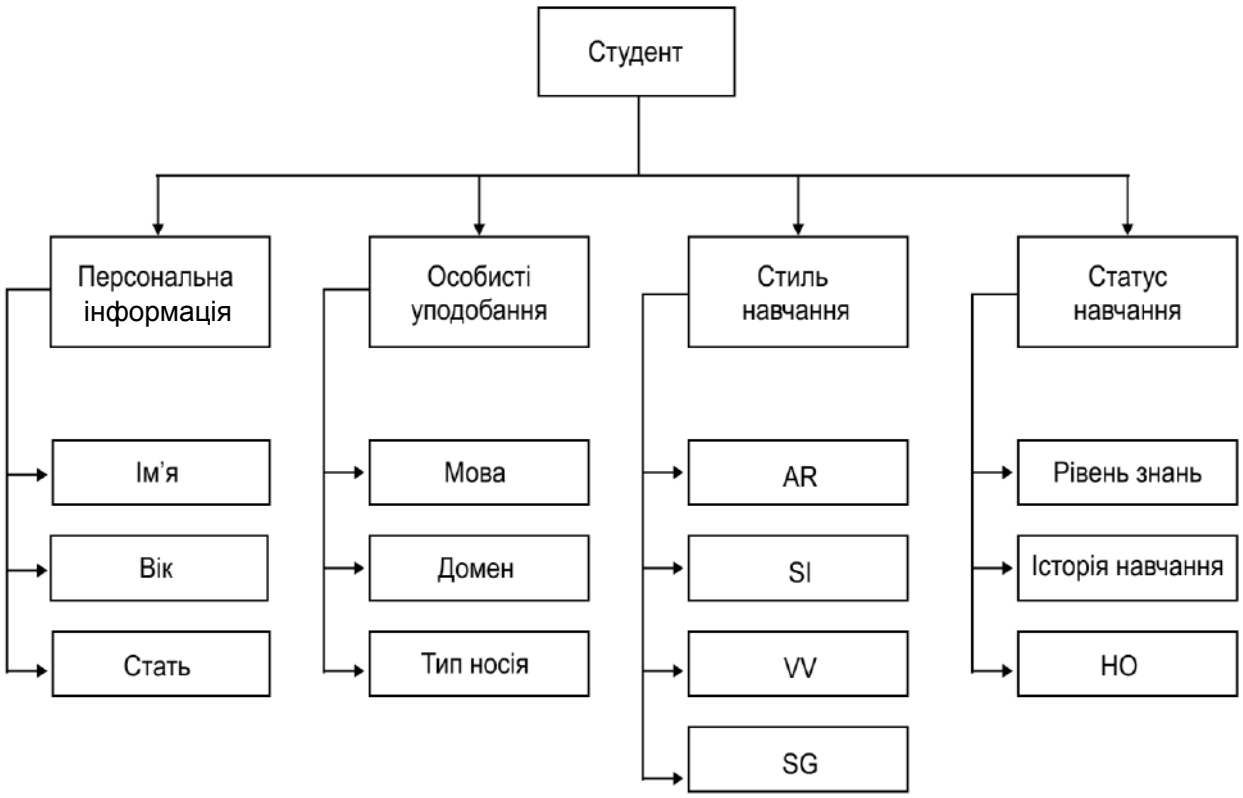


Рис. 1.5. **Онтологія студента**

Це стандартне подання даних студентів, які можна збирати двома способами: безпосередньо від студента або шляхом аналізу його результатів за допомогою системи управління навчанням (LMS). Студент надає особисту інформацію та заповнює online-анкету ILS (Index of Learning Styles) для визначення стилю навчання. Після вибору курсу для навчання, студенти складають початковий тест для визначення рівня знань, а потім формують його профіль. Усією цією інформацією управляє агент.

Модель ФС складається із чотирьох стилів навчання, як-от: *активний-рефлективний (active-reflective), сенситивний-інтуїтивний (sensing-intuitive), візуальний-вербальний (visual-verbal)* та *послідовний-глобальний (sequential-global)*. Індекс стилів навчання – це інструмент, що дозволяє оцінити переваги студента за чотирма шкалами. Його реалізовано у вигляді опитного листа із 44 питань (11 для кожного стилю) і перебуває у вільному доступі в інтернеті (https://www.webtools.ncsu.edu/learningstyles/submit.php). Приклад анкетування наведено на рис. 1.6.

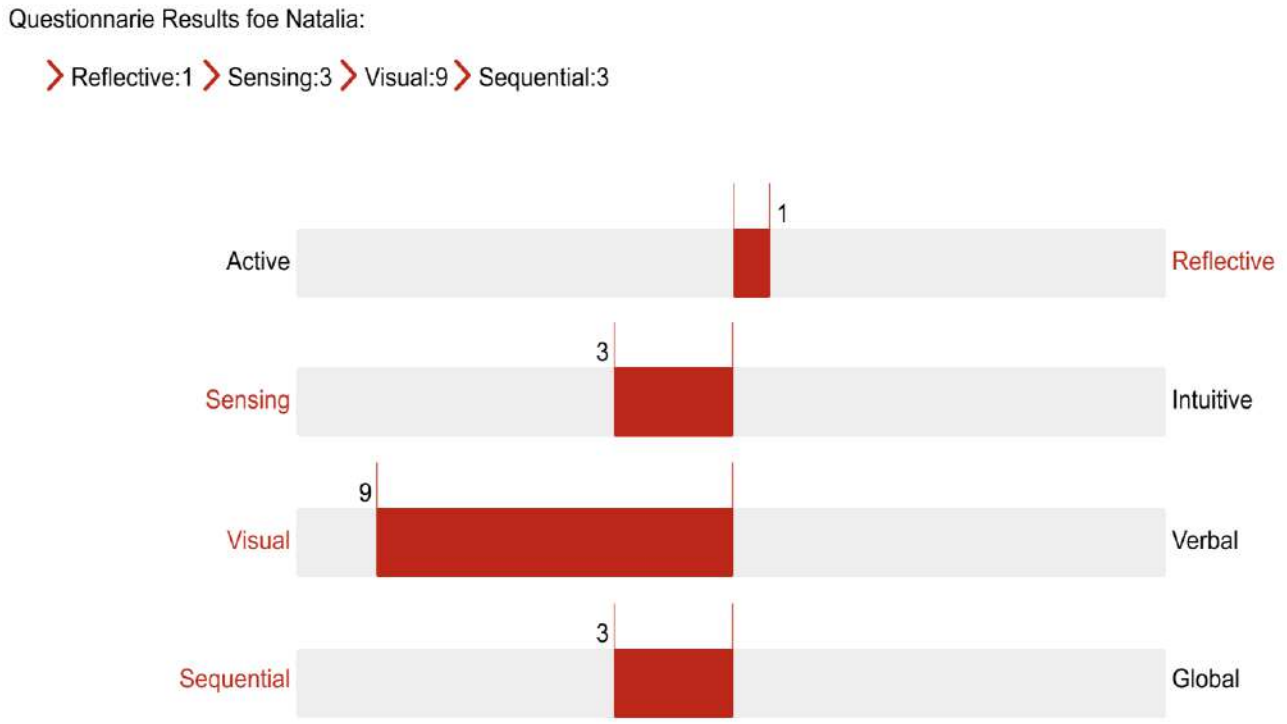


Рис. 1.6. **Результати анкетування**

Коли хтось заповнює анкету ILS у режимі онлайн, негайно формується відповідь із балами за всіма чотирма стилями, короткими поясненнями їхнього значення та посиланнями на більш детальну інформацію про те, як ці оцінки слід інтерпретувати. ILS є доступним безкоштовно для осіб, які бажають оцінити свої власні вподобання, або для викладачів, які бажають використовувати його для навчання в аудиторії чи у процесі дослідження.

Кожне питання складається із двох варіантів відповідей, наприклад:

Коли мені дають інструкції, як дістатися нового місця, я віддаю перевагу:

намальованій карті;

текстовому опису.

Кожен стиль навчання пов'язано з 11 пунктами, водночас кожен вимір має дві протилежні категорії (*a* або *b*) та відповідає тій чи тій категорії (наприклад, *активний – a / рефлективний – b* або *візуальний – a / вербальний – b*).

Кожне питання належить до однієї із чотирьох шкал, а відповіді – протилежним сторонам:

$$< \mathrm{AR}, \mathrm{SI}, \mathrm{VV}, \mathrm{SG} >, \tag{1.5}$$

де AR – оцінка за активно-рефлективною шкалою;

SI – оцінка за сенсорно-інтуїтивною шкалою;

VV – оцінка за візуально-вербальною;

SG – оцінка за послідовно-глобальною.

Кожна оцінка – це непарне ціле число з відрізка [–11; +11], яке формується так:

$$\mathrm{P}(\mathrm{i}) = -\mathrm{N}(\mathrm{i})_{\mathrm{a}} + \mathrm{N}(\mathrm{i})_{\mathrm{b}}, \tag{1.6}$$

де $\mathrm{P}(\mathrm{i})$ – переваги студента в і-му стилі $(\mathrm{i} = \mathrm{AR}, \mathrm{SI}, \mathrm{VV}, \mathrm{SG})$;

$\mathrm{N}(\mathrm{i})_{\mathrm{a}}$ – кількість відповідей, що відповідають стороні a шкали і;

$\mathrm{N}(\mathrm{i})_{\mathrm{b}}$ – кількість відповідей, що відповідають протилежній стороні b цієї ж шкали.

Визначена оцінка для кожного параметра вказує на переваги студента для тієї чи тієї категорії (рис. 1.7).

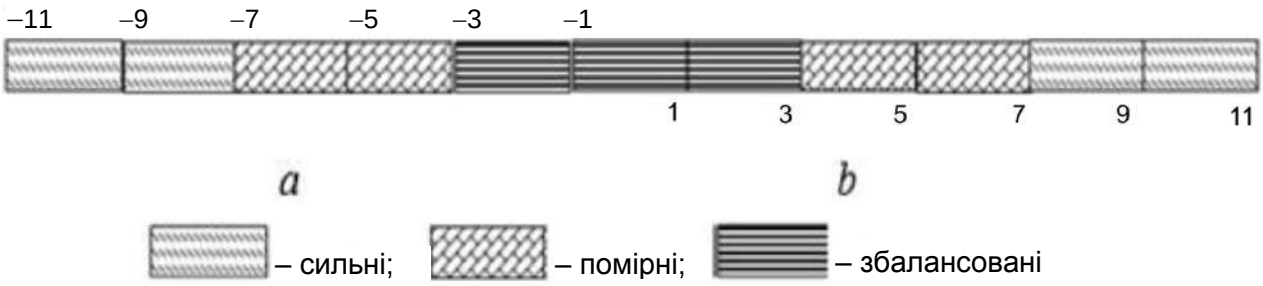


Рис. 1.7. **Типи переваг**

Наприклад, якщо оцінка за стиль дорівнює 1 або 3, студент має досить добре збалансоване вподобання за двома категоріями цього виміру і лише помірно віддає перевагу тому чи тому. Якщо оцінка дорівнює 5 або 7, то перевагу віддають помірно одній категорії цього виміру. Тож студент може навчатися в обох стилях, але в уподобаному стилі йому буде набагато легше, а результати будуть кращими. Якщо оцінка дорівнює 9 або 11, перевагу віддають одній категорії, у студента можуть виникнути труднощі, йому буде дуже важко навчатися у протилежному стилі.

Якщо перевага студента більш схиляється до виміру *візуальний* (як у прикладі *visual*=9), то такі студенти найкраще запам'ятовують те, що бачать – рисунки, діаграми, блок-схеми, часові лінії, фільми та демонстрації. Для таких студентів треба визначити діаграми, ескізи, схеми, фотографії, блок-схеми чи будь-яку іншу візуальну презентацію матеріалу курсу. Запитати викладача, звернутися до довідників і подивитися, чи є в наявності будь-які відеоматеріали з курсу. Підготувати концептуальну мапу, перелічувавши ключові точки, об'єднавши їх у квадрати чи кола та накресливши лінії зі стрілками між поняттями, щоб показати зв'язки. Позначити нотатки кольоровим маркером, щоб усе, що стосується однієї теми, було однаковим кольором.

На противагу студенти, які більш схиляються до виміру *вербальний*, дістають більше інформації від слів – письмових та усних пояснень. Кожен дізнається більше, якщо інформацію подають як візуально, так

і вербально. Для таких студентів треба писати конспекти або плани навчального матеріалу своїми словами. Робота у групах може бути особливо ефективною: вони досягають розуміння матеріалу, слухаючи пояснення однокурсників, і дізнаються ще більше, якщо дістають пояснення.

Отже, кожен охочий може дізнатися розподіл своїх переваг у стилях навчання за моделлю Фельдера – Сільвермана, а викладач може використовувати ці дані у процесі побудови курсу та подальшому навчанні студентів. Але варто мати на увазі, що переваги можуть досить змінюватися із часом, а також залежать від предметної галузі. Тож модель ФС може бути застосовною для диференціювання онлайн-курсів за чотирма шкалами стилів навчання.

Модель адаптації має на меті створити відповідний навчальний шлях для поліпшення навчання. Вона використовує інформацію, що зберігають у моделі студента та моделі вмісту, щоб надати відповідні рекомендації. Модель адаптації забезпечує послідовність НО, відповідно до стилю навчання, рівня знань студента. Як подано в моделі вмісту, вибравши навчальну одиницю, студент може переглядати різні НО вибраної навчальної дисципліни. Навчальний блок може містити теоретичні файли, лекції, приклади, вправи, самостійне спостереження, досліди, лабораторні та практичні роботи тощо. Студент може елементи читати, слухати або переглядати відео, виконувати вправу, брати участь у дослідах тощо. Завдання системи – вибрати НО, відповідно до профілю студента з урахуванням стилю викладання.

1.4. Архітектура багатоагентної системи е-навчання

Для моделювання процесу комунікації та координації, щоб рекомендувати НО, адаптовані до профілю студента, відповідно до його характеристик та уподобань пропонують багаторівневу архітектуру системи е-навчання, засновану на мультиагентному підході. Цю архітектуру взаємопов'язано із двома платформами (рис. 1.8), а саме середовищем розроблення багатоагентних систем AIgentCreator, написане на мові програмування C++ із використанням графічної бібліотеки Qt4, та системою Moodle LMS. Взаємодію агентів здійснюють за допомогою розробленого

мовою XML протоколу передавання даних засобами технології CORBA (Common Object Request Broker Architecture).

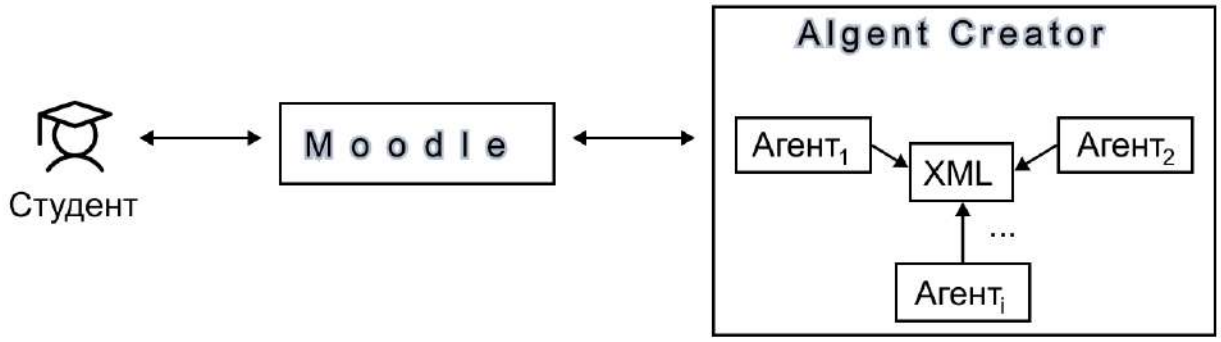


Рис. 1.8. **Архітектура багатоагентної системи**

Moodle LMS – це навчальна платформа, розроблена для того, щоб надати викладачам, адміністраторам та студентам єдину надійну, безпечну та інтегровану систему для створення персоналізованих навчальних середовищ. Можна завантажити це програмне забезпечення на свій власний вебсервер. Moodle надають безкоштовно як програмне забезпечення з відкритим вихідним кодом під загальною суспільною ліцензією GNU. Будь-хто може адаптувати, розширити або модифікувати Moodle як для комерційних, так і для некомерційних проєктів без будь-яких ліцензійних зборів і дістати вигоду від економічної ефективності, гнучкості та інших переваг використання Moodle.

Платформа *AIgentCreator* є конструктором для створення багатоагентної системи (МАС), яка складається з кількох частин, як-от:

графічний конструктор;

скелет (шаблон) агентської програми;

контролер (серверний застосунок, який дозволяє управляти та здійснювати взаємодію між агентами) [32].

Графічний конструктор є середовищем для розроблення МАС, який дозволяє додавати в робочу область агентів і пов'язувати їх між собою за допомогою відповідних інструментів. Кожен агент має список властивостей, як-от: ім'я, обробники подій (запуск агента, відправлення повідомлення, приймання повідомлення та ін.).

Скелет дозволяє на основі введених даних генерувати вихідний код агентського застосунку. Цей код є практично однаковим для всіх агентів – це API (Application Programming Interface) для опису реальної функціональності агентів.

Контролер є поштовим сервером для пересилання повідомлень між агентами.

На рис. 1.9 показано діаграму використання розробленого проєкту. Діаграма параметрів використання відображає набір дій, які можна виконувати.

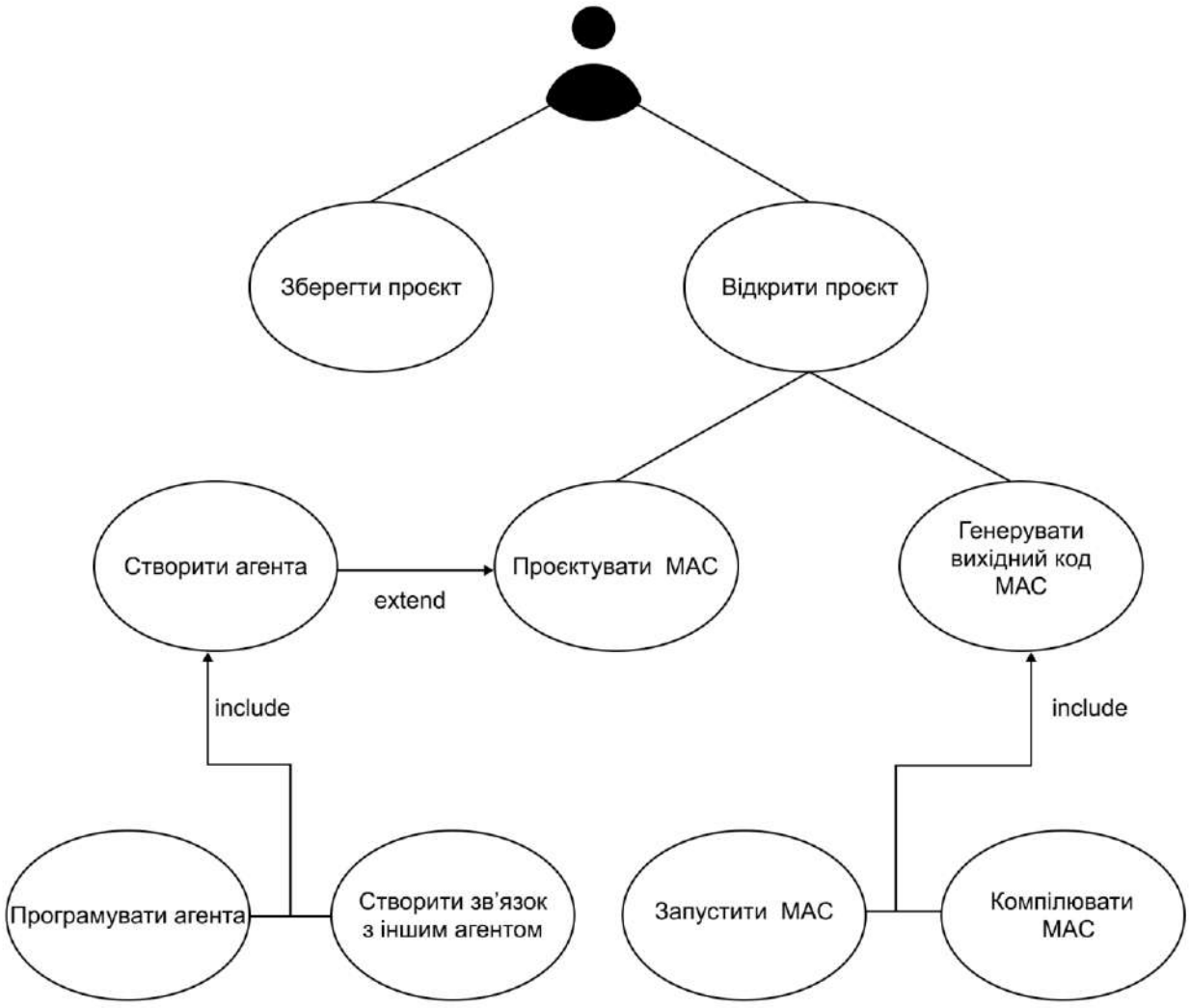


Рис. 1.9. **Діаграма варіантів використання розробленого середовища**

Дії "відкрити" та "зберегти проєкт" дозволяють використовувати дискові накопичувачі для зберігання та відновлення розроблених проєктів. Збережений проєкт є директорією, у якій розташовано обробники подій для кожного агента та файл з описом проєкту у форматі XML. Процес генерування – це компіляція та розгортання МАС на комп'ютері. Запуск МАС здійснюють за допомогою згенерованого файлу-сценарію, який по черзі запускає кожного агента.

Проєктування МАС передбачає створення множини агентів, які взаємодіють у програмному середовищі. Створення агента відбувається шляхом додавання відповідного компонента в робочу галузь програми. Як видно з рис. 1.9, створення агента містить дві інші дії: зв'язування створеного агента з іншими агентами та програмування агента.

Зв'язування відбувається шляхом поєднання двох агентів у робочій області. Зв'язок між агентами може бути одно- та двоспрямованим. У першому разі повідомлення можуть передаватися тільки в одному напрямку, а другому – агенти мають можливість обмінюватися повідомленнями в обох напрямках.

Програмування агентів відбувається за допомогою програми файлів-обробників на дії агентів. Ці обробники є стандартним файлом, написаним мовою сценаріїв (Perl, Python та ін.). Обробники викликаються з агентської програми автоматично в разі генерування будь-яких подій (ініціалізації, отримання повідомлення, подій таймера та ін.). Із метою вилучити з компіляції код, який не використовують у конкретному агенті (наприклад, один з обробників події), використовують директиви препроцесора умовної компіляції.

Специфічні блоки коду – це препроцесорні інструкції, які компілюють лише за умови наявності спеціального прапорця для цього блоку. Алгоритм генерування коду показано на рис. 1.10.

Коли всі агенти у списку оброблено та для них заповнено шаблони, генератор коду заповнює шаблон системи складання. Під *системою складання* мають на увазі набір правил для утиліти GNU Make, яка управляє процесом компіляції.

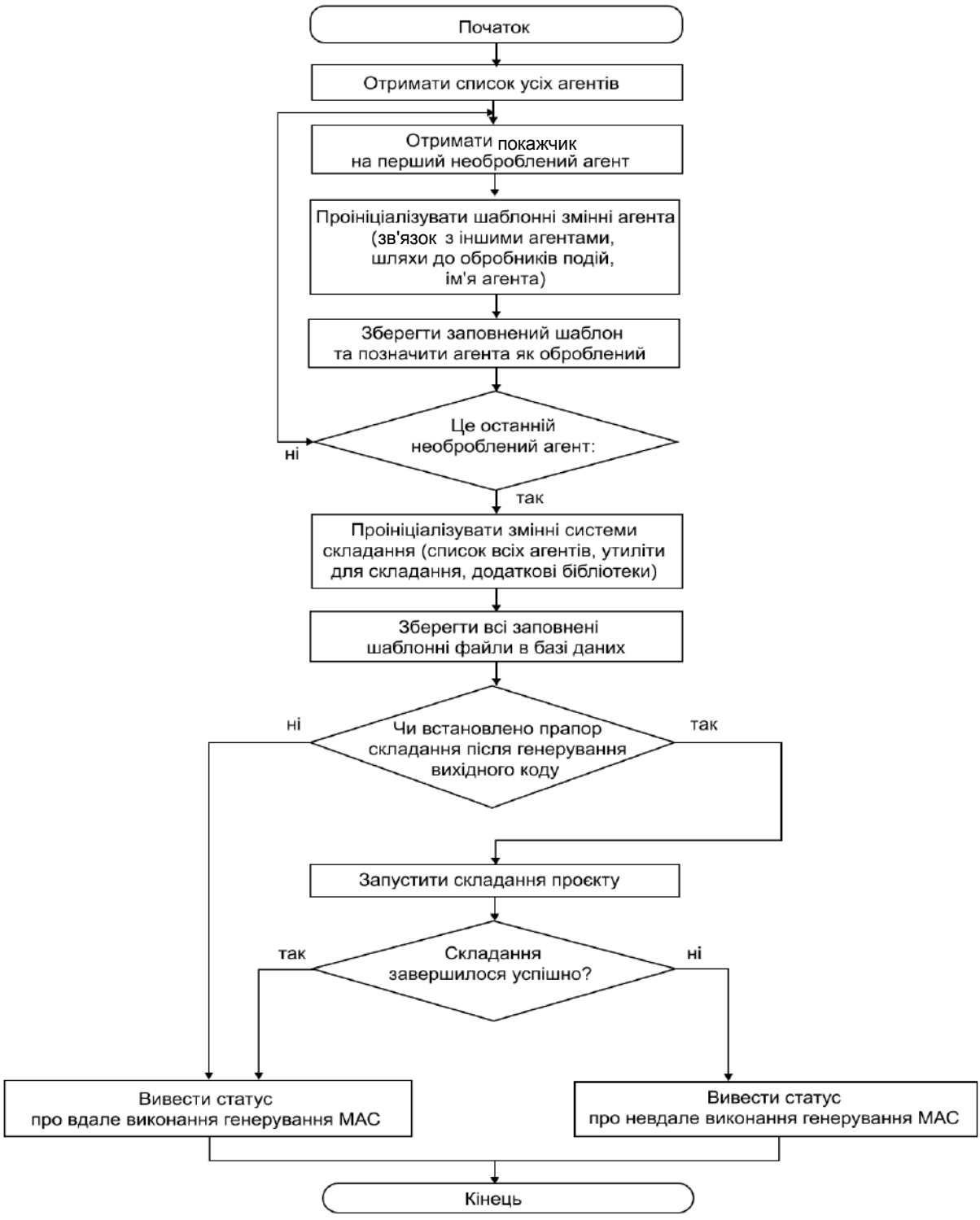


Рис. 1.10. **Алгоритм генерування вихідного коду MAC**

Модель багатоагентної системи подають у такому вигляді:

$$MAS = \langle A, E, Qt_{Creator} \rangle, \tag{1.7}$$

де $A = \{A_i\}$ – множина агентів, що функціонують у середовищі AIgentCreator, і набуває таких значень:

$i = learning$ – "агент навчання" ($A_{learning}$) інкапсулює особисті дані студента;

$i = intermediate$ – "агент-посередник" ($A_{intermediate}$) надсилає запити студента відповідним агентам, та повертає навчальний матеріал у відповідному для студента форматі;

$i = control$ – "агент регулювання" ($A_{control}$) підтримує процес синхронізації між середовищем *AIgentCreator* та платформою Moodle;

$i = rating\_points$ – "агент оцінювання" (A_{rating_points}) оцінює рівень навчання студента та перевіряє, чи запропоновано системою НО для його поліпшення;

$i = adaptation$ – "агент адаптації" ($A_{adaptation}$) відповідає за генерування навчального шляху студента, відповідно до даних, отриманих від агента регулювання та агента навчання;

$i = content$ – "агент умісту" ($A_{content}$) відповідає за пошук НО в базі даних, відповідно до запиту від агента адаптації;

$i = tracing$ – "агент відстеження" ($A_{tracking}$) виявляє нові події в системі;

$E = \left\{E_i^{states}\right\}$ – множина станів середовища AIgentCreator.

Агент – це програмна сутність, яка здатна самостійно здійснювати будь-яку діяльність, тобто, це програма, яка після запуску здатна виконувати закладені в ній дії. На рис. 1.11 показано діаграму використання, де відображено основну функціональність агентів.

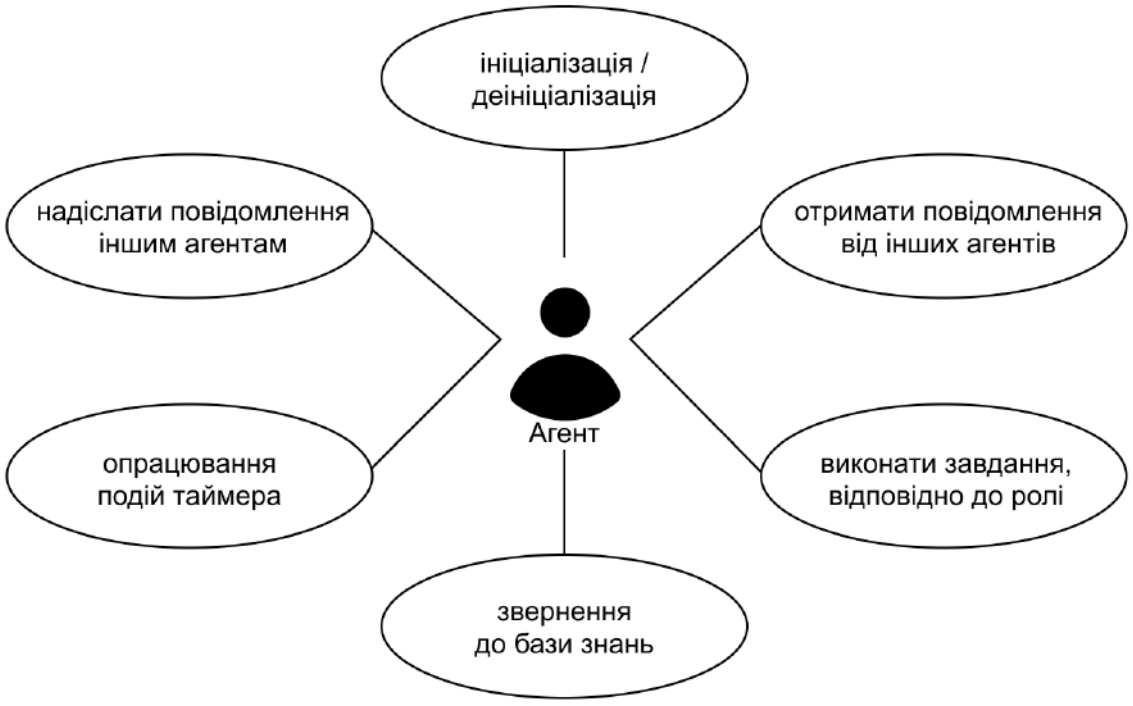


Рис. 1.11. **Діаграма використання агентської програми**

Ініціалізацію виконують як першу подію, що опрацьовує агент. Її ініціюють під час запуску агента і виконують лише один раз. Призначено її для виконання дій, які сигналізували б іншим агентам про те, що агента було запущено або для створення чи отримання будь-яких ресурсів, необхідних для подальшої роботи програми, або для встановлення таймера.

Деініціалізацію виконують після завершення роботи програми. Як і ініціалізацію, цю дію виконують лише один раз.

Отримання повідомлень від інших агентів є однією з головних функцій агента. Так агент може діставати вказівки або інформацію з мережі.

Надсилання повідомлень призначено для передавання іншим агентам указівок або інформації.

Опрацювання подій таймера можна використовувати для періодичної перевірки будь-якого ресурсу в мережі або для перевірки статусу виконання будь-якого завдання іншим агентом.

Кожен агент оснащено базою знань, що складається зі сценаріїв, яких слід використовувати, якщо виникають однакові запити.

Архітектура ґрунтується на декількох агентах, призначених для спілкування та обміну даними, із метою надання студентам відповідної інформації, наприклад, навчальних ресурсів, що відповідають їхнім характеристикам та уподобанням. Агенти в цій системі мають співпрацювати для здійснення процесу адаптації НО.

У багатоагентній системі (1.7) кожний агент Ai, $Ai \in A$ надано таким набором:

$$A_i = \left\{ A_i^{action}, E_i^{states}, F_{behavior}, A_i^{states}, F_{states}, F_{decision} \right\}, \quad (1.8)$$

де A_i^{action} – множина дій агентів;

відображення $F_{behavior} : A_i^{states} \times E_i^{states} \to 2^{A_i^{states}}$ формує поведінку середовища;

A_i^{states} – множина внутрішніх станів агентів;

відображення $F_{states} : A_i^{states} \times E_i^{states} \to A_i^{states}$ оновлює стани агентів;

відображення $F_{decision} : A_i^{states} \to E_i^{states}$ дозволяє ухвалювати рішення для виконання дій агентом за поточним внутрішнім станом.

Розподілену архітектуру багатоагентної системи е-навчання засновано на використанні методів комунікації та координації, який дозволяє

запропонованій системі електронного навчання рекомендувати НО, адаптовані до профілю студента, відповідно до його характеристик та уподобань. Для цього використовують набір автономних агентів $\{A_i\}$, які забезпечують поведінку (завдання та дії, які потрібно виконати) для створення адекватної відповіді студентові. Сутність кожного агента змодельовано послідовністю ролей та завдань, які вони будуть виконувати, відповідно до потреб.

Послідовність дій (діаграма взаємодії), що виконує і-й агент із моменту запуску й до кінця його існування в середовищі, що моделюють показано на рис. 1.12. На ній подано трьох учасників – і-й агент, сервер та інші агенти середовища.

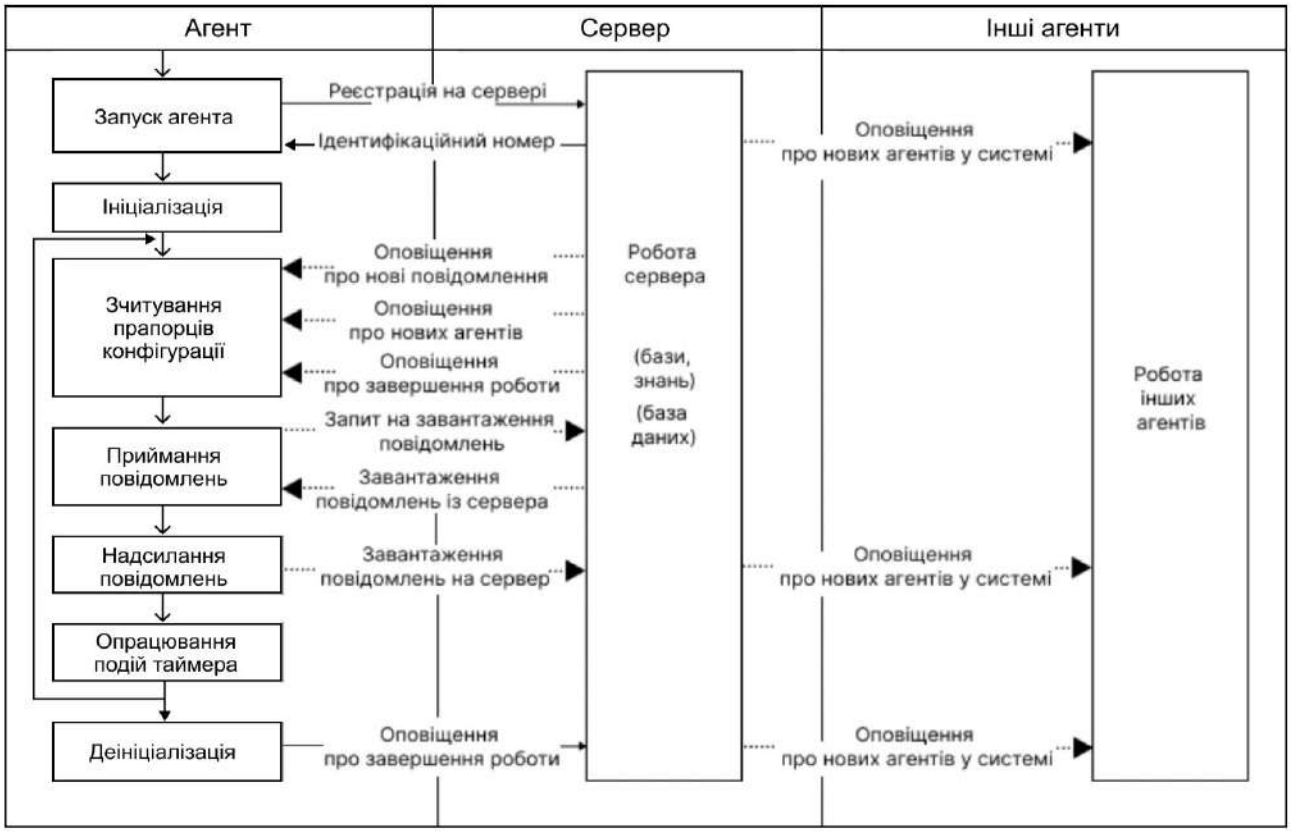


Рис. 1.12. **Діаграма діяльності (взаємодії) агента в системі**

Прямокутники є діями (операціями), що виконує агент. Вертикальними стрілками вказано послідовність виконання цих дій. Горизонтальними – повідомлення, якими агент обмінюється із сервером. Суцільна стрілка вказує, що це повідомлення відправляють у будь-якому разі,

а пунктирна – означає, що факт відправлення повідомлення залежить від будь-якої умови.

Розгляд діаграми починають із запуску агента. У цей момент широкомовним розсиланням надсилають запит на реєстрацію на сервері. Сервер під час отримання такого запиту додає інформацію про цього агента (поточне місце розташування та його тип ($A_{learning}$,$A_{intermediate}$, $A_{control}$,A_{rating_points},$A_{adaptation}$,$A_{content}$, $A_{tracking}$)) у свою базу даних і на основі цих даних генерує унікальний ідентифікаційний код, який передає застосунку. Цей код використовують під час надсилання повідомлень між агентами та сервером. Після реєстрації на сервері агент виконує певний для нього сценарій.

Приймання повідомлень, як і відправлення, здійснюють у разі, якщо для агента вказано сценарій опрацювання даних, що приймають. Агент надсилає запит на отримання даних, сервер надсилає їх агенту й потім запускає сценарій (рис. 1.13). Завантажене повідомлення зберігають у файлі на накопичувачі, а до сценарію передають його дескриптор.

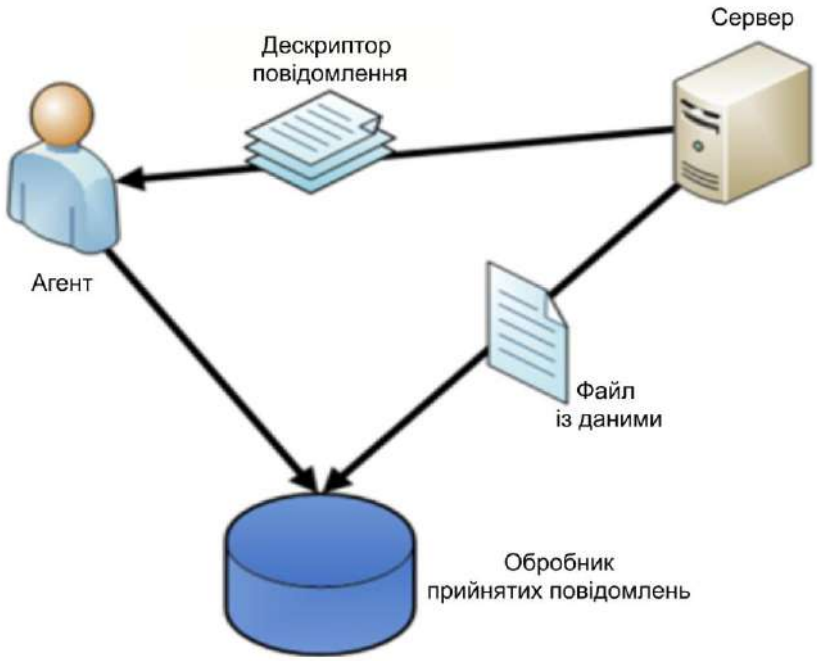


Рис. 1.13. **Механізм обміну повідомленнями**

Під час надсилання повідомлення запускають відповідний сценарій. Сценарій призначено для опрацювання даних, які буде надіслано. Потім

формують дескриптор, у якому вказують розмір повідомлення, шлях розміщення повідомлення на диску, ідентифікатори агентів, які мають його отримати, та контрольну суму для перевірки цілісності повідомлення. Цей дескриптор відправляють на сервер, який завантажує це повідомлення за вказаною адресою. Потім він аналізує дескриптор і надсилає сповіщення про нове повідомлення всім агентам, зазначеним у дескрипторі.

Агентно-орієнтовану модель взаємодії між середовищем AIgentCreator, і платформою Moodle LMS показано на рис. 1.14.

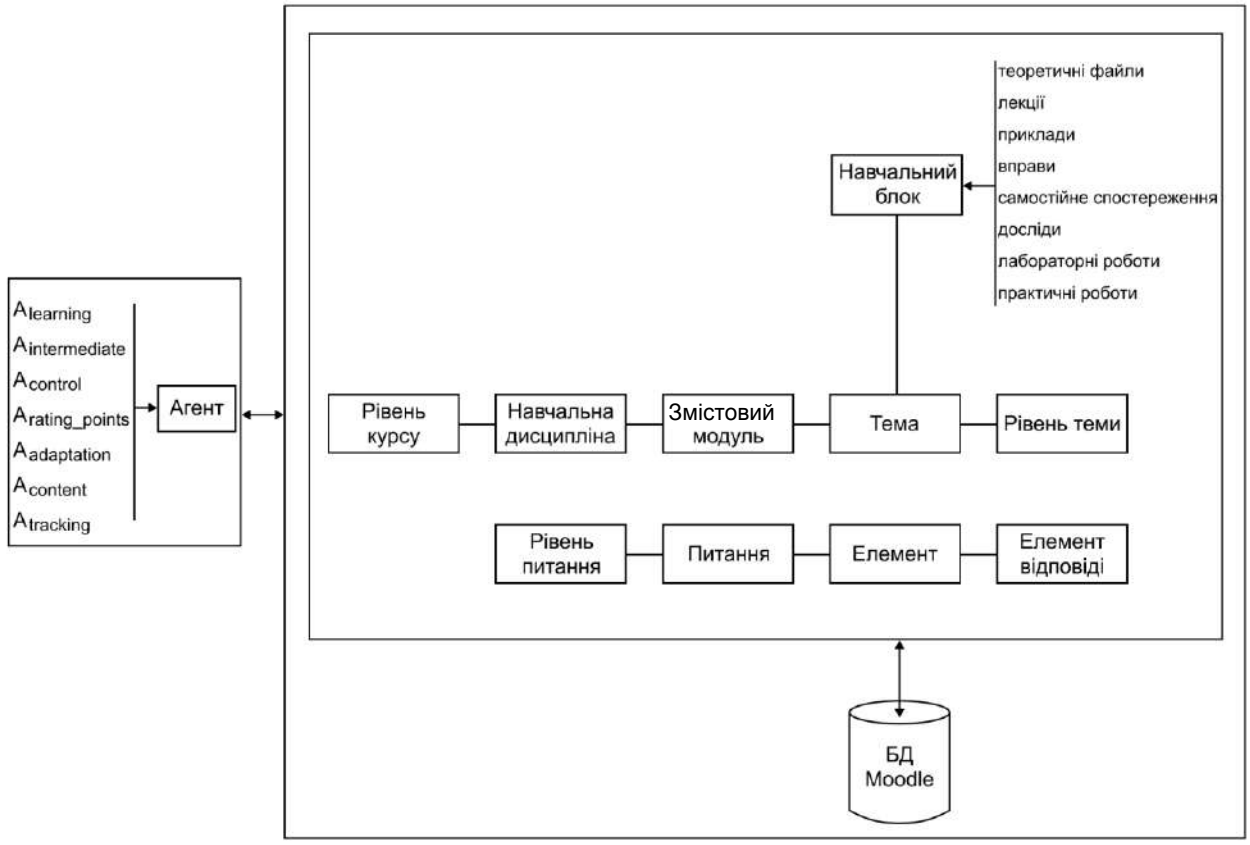


Рис. 1.14. **Модель взаємодії між середовищем AIgentCreator та платформою Moodle LMS**

Процес комунікації та координації між цими двома системами здійснюють за допомогою розробленого мовою XML протоколу передавання даних засобами технології CORBA. Незалежність від мов програмування та операційних середовищ, фундаментальна підтримка об'єктно-орієнтованого програмування та багато інших унікальних характеристик, зробили

CORBA провідним стандартом у галузі інфраструктурного middleware. Основою технології CORBA є такі:

IDL (Interface Definition Language) – це мова, що дозволяє описати всі аспекти віддаленої взаємодії;

схеми відображення IDL-оголошень на конкретні мови програмування;

ORB (Object Request Broker) – це об'єктна магістраль, що дозволяє передавати запити від клієнтів до серверів та назад;

послуги (Common Object Services) CORBA – обслуговування динамічного формування запитів (DII), репозитарію інтерфейсів (IR), динамічного опрацювання запитів (DSI) та ін.;

GIOP – це специфікація протоколу (команди та формати інформації, що передають);

IIOP (Internet Inter-Orb Protocol) використовують GIOP для TCP/IP. IIOP є конкретною реалізацією абстрактних визначень GIOP.

Агенти в цій системі співпрацюють для здійснення процесу адаптації НО.

Агент адаптації інтерфейсу управляє всім зв'язком між інтерфейсом та системою (рис. 1.15). Цей агент діє як посередник, який передає студентські запити відповідним агентам, а потім повертає навчальний матеріал у відповідному для студента форматі.

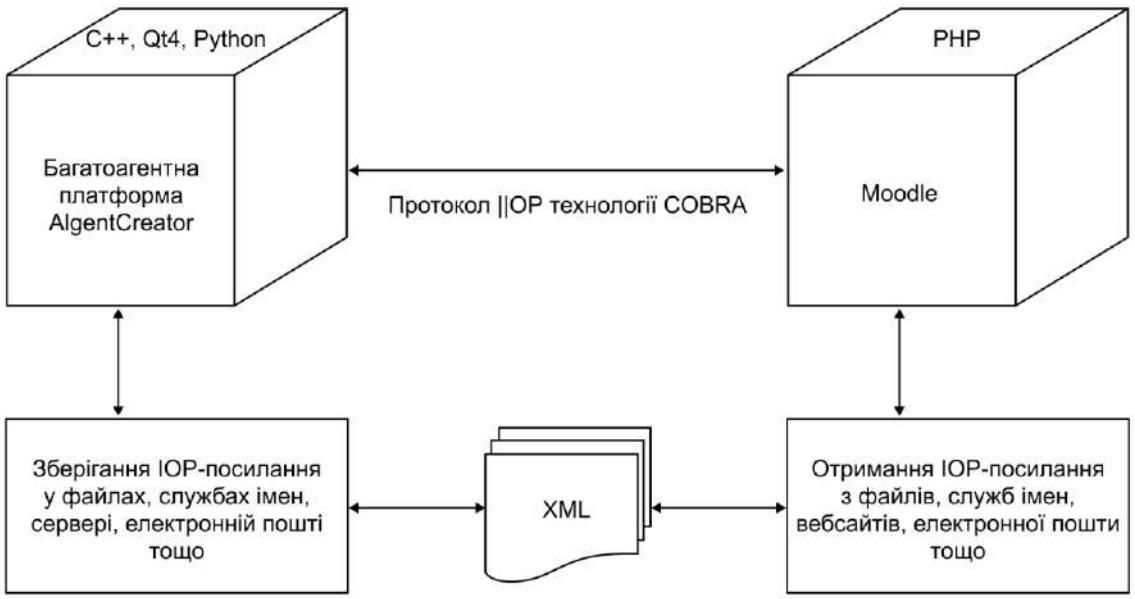


Рис. 1.15. **Модель зв'язку між використовуваними платформами**

Саме для того, щоб здійснити зв'язок між двома платформами, має бути інтерфейс, створений, завдяки використанню технології CORBA.

Кожен студент має власного агента навчання.

1.5. Моделювання мультиагентної системи розподіленої торговельної фірми

Реалізовано демоверсію багатоагентної системи е-навчання, для роботи якої необхідно таке: бібліотека Qt4; утиліта складання GNU make; компілятор GCC; інтерпретатор Perl або Python; бібліотека CORBA. Більшість із цього ПЗ поставляють з операційними системами Linux і MacOS (крім бібліотек CORBA і Qt4). Для операційної системи Windows це програмне забезпечення необхідно встановити вручну.

У системі агент адаптації дістає персональну інформацію, а потім передає її агентові вмісту, щоб вибрати НО та їхні взаємозв'язки. НО презентують стани, а відносини між ними – дії.

Агент умісту повертає всі ресурси, які відповідають профілю студента з вербальним стилем навчання. Агент адаптації має вибрати найбільш вигідний шлях для поточного студента. У цьому прикладі агент умісту містить такі НО: конспект лекцій, методичні вказівки, приклад програми, лабораторні роботи, дослідницьку роботу та залік із такими діями:

$1.2.1_{VV}$ Listen Lecture;

$1.2.2_{VV}$ Read Methodical;

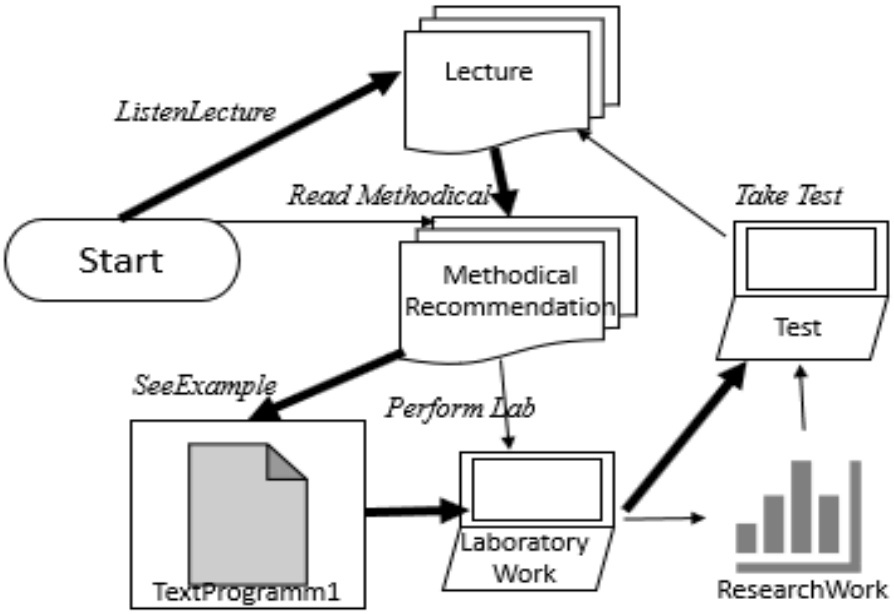
$1.2.3_{VV}$ SeeExample;

$1.2.4_{VV}$ PerformLab;

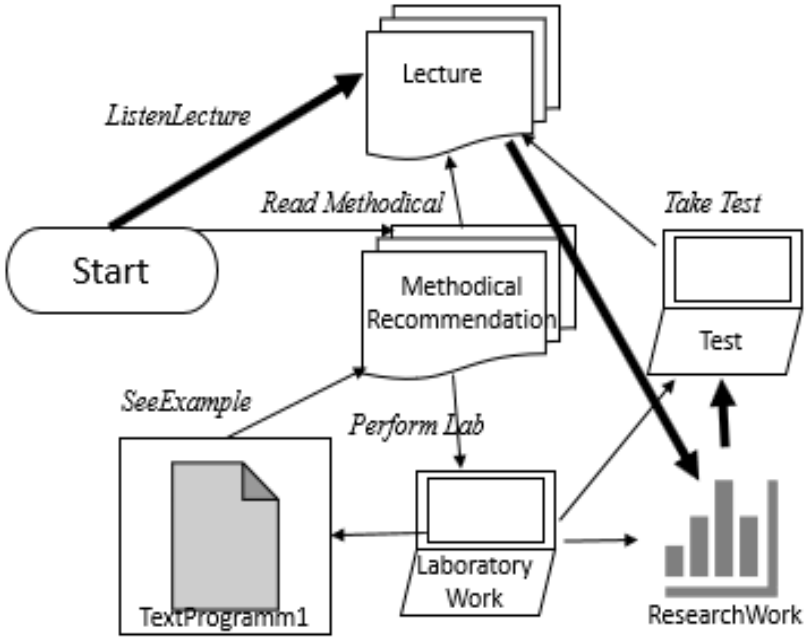
$1.2.5_{VV}$ ResearchWork;

$1.2.6_{VV}$ TakeTest.

Наприклад, для студента із середнім рівнем знань і вербальним стилем навчання буде побудовано такий шлях: 0 – $1.2.1_{VV}$ – $1.2.2_{VV}$ – $1.2.3_{VV}$ – $1.2.4_{VV}$ – $1.2.6_{VV}$ (рис. 1.16а, товста лінія на цьому рисунку).



а



б

Рис. 1.16. **Приклади шляхів для студентів із вербальним стилем навчання**

Якщо ж студент із просунутим рівнем знань і вербальним стилем навчання, то найкращим шляхом навчання буде такий: 0 – $1.2.1_{VV}$ – $1.2.5_{VV}$ – $1.2.6_{VV}$ (рис. 1.16б, товста лінія на цьому рисунку).

1.6. Висновки

Запропоновано розподілену трирівневу архітектуру, засновану на багатоагентному підході до моделювання адаптивної системи електронного навчання. Це дозволяє рекомендувати навчальний шлях, адаптований до профілю студента, відповідно до його характеристик та уподобань з урахуванням стилю викладання.

Архітектура, яку використовують у цій системі, ґрунтується на комунікації дій агентів системи, водночас кожен агент виконує чітко визначене завдання.

Розроблено інтегроване середовище для побудови багатоагентних систем.

Реалізовано демоверсію багатоагентної системи е-навчання, для роботи якої використано технологію CORBA.

Як мову опису поведінки агентів використовують мови, що інтерпретуються (Perl або Python).

Запропоновану систему може бути інтегровано з будь-якою системою управління навчанням.

Розділ 2
Алгоритмічні та інтерфейсні рішення для проєктування мобільної інформаційно-навігаційної системи університету

2.1. Вступ і формулювання завдання

Ефективне розв'язання науково-прикладної проблеми навігації та орієнтування в закритих приміщеннях набуває сьогодні все більшої актуальності. Більшість відомих технологічних рішень такої проблеми спрямовано на забезпечення інформаційної підтримки користувачів різного статусу в орієнтуванні в будівлях складної, багаторівневої й розгалуженої структури та топології (офісні й культурні приміщення, комерційні центри, культурні об'єкти, паркувальні майданчики тощо).

Окремої уваги потребує сьогодні швидка й надійна навігація в університетському кампусі та його корпусах, адже приміщення сучасного університету зазвичай мають складну структуру з великою кількістю поверхів, аудиторій, кабінетів і локацій різноманітного призначення. Університети сьогодні є важливими навчальними, науковими та культурними осередками, які щоденно відвідує значна кількість людей із різними цілями. Зокрема, потреба у зручній навігації у приміщеннях університету виникає в абітурієнтів, студентів, учасників змагань, нових викладачів і відвідувачів університету, незнайомих із розташуванням його корпусів та їхньою топологією, що актуалізує проєктування й розроблення відповідної інформаційно-навігаційної системи (далі ІНС).

Відповідно до джерел [23; 45], під ІНС розуміють електронну систему, що забезпечує навігацію. ІНС вирішують питання визначення місцезнаходження користувача, прокладання маршруту з однієї точки в іншу, а також доповнюючи вказаний раніше функціонал певною корисною інформацію, яка може бути важливою користувачу для полегшення орієнтування.

Для побудови систем, що використовують інформацію про місцезнаходження користувача всередині приміщень, застосовують досить широкий спектр інформаційно-комунікаційних технологій. У контексті орієнтування у приміщеннях університету, залучення вебтехнологій для вирішення завдань навігації має переваги, адже вебзастосунки не потребують заздалегідь установленого програмного забезпечення, окрім браузерів, які зазвичай є доступними в кожному сучасному пристрої,

не потребують установлення на пристрій та мають можливість використовувати кеш браузера та технологію PWA для роботи офлайн.

Аналіз науково-прикладних джерел засвідчує наявність різноманітних технологічних підходів до вирішення завдань місцезнаходження та орієнтування [29; 43]. Зосередимося на аналізі ІНС (або подібних застосунків), що забезпечують саме навігацію у приміщенні. Такі системи не можуть використовувати повною мірою можливості GPS-навігації (як це роблять ІНС місцевості за типом Google Maps, 2Gis, Open Street Maps тощо) через те, що навігація відбувається в невеликих масштабах і зазвичай у закритих приміщеннях. Отже, залучення технологій GPS-навігації призведе до значних помилок. У зв'язку із цим більшість ІНС для орієнтування у приміщеннях використовують indoor-навігацію.

Під *indoor-навігацією* розуміють виявлення об'єктів (пристроїв або людей) та орієнтування всередині будівлі за допомогою радіохвиль, інфрачервоних сенсорів, магнітного поля, акустичних сигналів, безпровідних мереж (Wi-Fi), технології Bluetooth або інших технологій [23]. Проте проблема такої навігації в тому, що кожна будівля, де треба використовувати таку навігацію, мусить мати спеціальні пристрої, налаштовані на роботу з ІНС, що робить розроблення такої системи дорогим та довготривалим, а також додає певних труднощів до її тестування, супроводження та розширення її можливостей.

Крім цього, дослідники зазначають, що технології indoor-навігації, що ґрунтуються на опрацюванні візуальних зображень, а також QR-кодів є недостатньо вживаними й невиправдано забутими, хоча потребують значно менше асигнувань і мають значні перспективи щодо їхнього залучення до розроблення мобільних та веборієнтованих застосунків [30].

Для аналізу можливостей було вибрано декілька сервісів і застосунків, що забезпечують орієнтування у приміщенні, використовуючи indoor-навігацію, і реалізують такі загальні функції: візуалізацію схеми будівлі поверху; віртуальне пересування по поверху (перегляд частин приміщення); забезпечення користувача візуальною інформацією для спрощення сприйняття схеми будівлі та інтуїтивної навігації.

У процесі здійснення аналізу зосередилися на оцінюванні аналогів ІНС за такими основними критеріями, як-от: 1) функціонал прокладання маршруту; 2) адаптивність інтерфейсу; 3) зручність та ергономічність інтерфейсу; 4) якість візуалізації графічного контенту; 5) естетичність дизайну; 6) ціна; 7) технологічні можливості щодо застосування в девайсах різних типів; 8) можливості нарощування функціоналу та доопрацювання.

Зокрема, за такими критеріями було оцінено низку застосунків, спираючись на такі джерела [23; 29; 29]:

- Mapsindoors (розширення платформи MapsPeople, побудованої за технологією Google Maps, що забезпечує перехід від зовнішньої до внутрішньої навігації та її швидке впровадження);
- Situm Mapping Tool (сервіс indoor-навігації, що забезпечує супровід відвідувачів конкретної будівлі для визначення шляху в реальному часі);
- AnyPlace (безкоштовний відкритий сервіс indoor-навігації, що забезпечує визначення місцезнаходження та пошук усередині будівель за допомогою смартфонів);
- BSB Navigator (застосунок для смартфонів, що є навігатором для бібліотеки на Людвіґштрассе в Мюнхені та використовує технологію маяків із застосуванням Bluetooth смартфона);
- AAU Map (програмний продукт для навігації в будівлях, корпусах, приміщеннях на території Ольборзького університету (Данія).

Здійснений аналіз за означеними критеріями засвідчив, що, незважаючи на значні функціональні можливості цих сервісів indoor-навігації, які є аналогами ІНС університету, вони мають такі недоліки й обмеження. Більшість із проаналізованих аналогів (1) є здатними забезпечити навігацію всередині конкретної будівлі, проте вони не є придатними для використання в інших приміщеннях. Водночас вони не мають достатньо розвиненого функціоналу розширення й доопрацювання; (2) не забезпечують користувача мовною локалізацію; (3) є або вебсервісами, або мобільними застосунками.

Указані обмеження актуалізують пошук відповідних алгоритмічних, інтерфейсних і технологічних рішень у процесі проєктування ІНС університету. Їх також доцільно взяти до уваги під час визначення функціональних і нефункціональних вимог до означеної системи, а також логічного формулювання завдання її проєктування.

2.2. Мета та формулювання завдання

Метою цієї роботи є розроблення і висвітлення алгоритмічних та інтерфейсних рішень для проєктування ІНС ХНЕУ ім. С. Кузнеця на основі застосування вебтехнологій.

Для досягнення поставленої мети необхідно розробити специфікацію вимог до означеної ІНС; уточнити формулювання завдання проєктування;

висвітлити основні етапи проєктування та розроблення інформаційно-навігаційної системи в контексті запропонованих рішень; проаналізувати результати апробації системи в освітній практиці ХНЕУ ім. С. Кузнеця.

2.3. Виклад основного матеріалу

У процесі пошуку алгоритмічних та інтерфейсних рішень було проведено специфікацію вимог до всієї ІНС шляхом побудови діаграми використання систем користувачем, який є будь-якою людиною, що має на смартфоні встановлену ІНС як вебзастосунок або вебпосилання на нього.

Передбачені варіанти використання системи користувачем є такі:

перегляд інструкцій із застосування;

перегляд налаштувань (включно зі зміною мови інтерфейсу);

визначення власного місцезнаходження;

вибір стартової точки (точки, яка відповідає координатам розміщення фізичного QR-коду відносно схеми будівлі);

вибір кінцевої точки (будь-якої локації у приміщенні університету, нанесеної на інтерактивну мапу, для якої користувач хоче визначити маршрут);

визначення прокладеного маршруту між стартовою та кінцевою точками;

здобуття анімованої візуалізації прокладеного маршруту; використання сканера QR-кодів;

вільний перегляд схеми поточного поверху;

зміна масштабу мапи;

зміна поточного поверху.

Приклади специфікації деяких варіантів використання наведено в табл. 2.1 – 2.3.

Таблиця 2.1

Варіант використання "Вибір кінцевої та/або стартової точки"

Характеристики	Значення
1	2
Контекст використання	Дозволено користувачу у формі пошуку ввести стартову точку та/або бажану точку, яку треба визначити або прокласти до неї маршрут
Дійові особи	Будь-який користувач ІНС

Закінчення табл. 2.1

1	2
Передумова	Немає
Тригер	Візуалізація стартової точки на інтерактивній мапі; прокладання маршруту; візуалізація кінцевої точки на інтерактивні мапі
Сценарій	1) увести або вибрати зі списку назву точки; 2) натиснути кнопку з іконкою збільшувальної лупи
Постумова	Автоматичне позначення на мапі відповідної точки або точки за прокладеним маршрутом за умови визначення обох точок (стартової та кінцевої, відповідно)

Таблиця 2.2

Варіант використання "Використання сканера QR-кодів"

Характеристики	Значення
Контекст використання	Відкриття діалогового вікна із запитом на камеру девайса користувача та активуванням сканера QR-кодів
Дійові особи	Будь-який користувач сайта з наявною камерою у його пристрої
Передумова	Користувач натиснув відповідну кнопку з іконкою сканера
Тригер	Необхідність у використанні сканера для визначення місцезнаходження користувача на основі фізичного QR-коду
Сценарій	Натискання відповідної кнопки
Постумова	Користувач зможе використовувати сканер та дістати після сканування певну інформацію, наприклад стартову точку

Таблиця 2.3

Варіант використання "Визначення прокладеного маршруту між стартовою та кінцевою точками"

Характеристики	Значення
1	2
Контекст використання	Дозволено будь-якому користувачеві визначити побудований системою шлях від стартової до кінцевої точки
Дійові особи	Будь-який користувач сайта
Передумова	Користувач вибрав стартову та кінцеву точки

Закінчення табл. 2.3

1	2
Тригер	Необхідність прокласти маршрут між визначеними точками
Сценарій	1. Вибір стартової точки, де розміщено QR-код, або вибір її у списку, або сканування QR-коду за допомогою сканера. 2. Натискання кнопки пошуку у відповідному полі введення. 3. Уведення назви потрібної локації або вибір її зі списку. 4. Натискання кнопки пошуку у відповідному полі введення
Постумова	Користувач зможе побачити побудований ІНС маршрут від стартової до кінцевої точки

Аналіз варіантів використання дозволив здійснити детальну специфікацію функціональних вимог до ІНС, що розробляють, із значенням їхнього пріоритету та складності.

До них слід зарахувати такі вимоги, як:

- налаштування мовної локалізації;
- вибір наявних на мапі локацій;
- визначення місцезнаходження користувача;
- пошук оптимального маршруту від стартової до кінцевої точок;
- візуалізація визначеного маршруту;
- огляд схеми поверху;
- вільна зміна поверхів;
- зміна масштабу мапи;
- надання підказок щодо типів локацій;
- використання спливних вікон для розміщення певного функціоналу;
- адаптивність інтерфейсу під різні розміри пристроїв та їхні графічні налаштування.

З огляду на визначені функціональні вимоги, було обґрунтовано технологічні підходи до проєктування та розроблення ІНС університету.

Крім цього, було взято до уваги необхідність у подоланні обмежень, притаманних наявним аналогам ІНС, зазначеним раніше (відсутність можливості розширення функціоналу системи; адаптації до навігації в іншому приміщенні, а також мовної локалізації; жорстка залежність від платформи функціонування).

Отже, було визнано необхідність у залученні сукупності таких технологій:

1) Angular як платформи для розроблення складних односторінкових застосунків, що використовує діалект TypeScript мови JavaScript для опису компонентів інтерфейсу з використанням принципів об'єктно-орієнтованого програмування;

2) реактивного програмування за допомогою бібліотеки RxJs;

3) бібліотеки SVG.js, яка застосовує можливості HTML і JavaScript для динамічної генерації, опрацювання та анімації векторних документів, що допоможе у створенні інтерактивної мапи.

Далі було здійснено специфікацію нефункціональних вимог до ІНС, яких було дотримано у процесі розроблення інтерфейсних та алгоритмічних рішень.

Специфікацію окремих суттєвих функціональних вимог до системи наведено в табл. 2.4 і 2.5.

Таблиця 2.4

Специфікація нефункціональних вимог (загальні вимоги)

Ідентифікатори вимог	Назви вимог	Атрибути вимог	
		пріоритет	складність
NFR-1-01	Час завантаження сторінки – не більше ніж 5 с	Обов'язковий	Висока
NFR-1-02	Відображення схеми поверху – не більше ніж 2 с	Обов'язковий	Середня
NFR-1-03	Швидкість роботи інтернету – 100 Мбіт/с	Рекомендований	Середня

Таблиця 2.5

Специфікація нефункціональних вимог (окремі вимоги за групами)

Ідентифікатори вимог	Назви вимог	Атрибути вимог	
		пріоритет	складність
1	2	3	4
NFR-2-01	Безвідмовна робота – півроку	Обов'язковий	Висока
NFR-2-02	Відсутність помилок у роботі системи	Обов'язковий	Висока

Закінчення табл. 2.5

1	2	3	4
NFR-3-01	Час завантаження сканера коду – не більше ніж 5 с	Обов'язковий	Висока
NFR-4-01	Дотримання форматування програмного коду та якісної організації файлів	Обов'язковий	Середня
NFR-4-02	Дотримання принципів SOLID, KISS, DRY	Обов'язковий	Середня
NFR-4-03	Коректне використання GoF-патернів проєктування	Обов'язковий	Висока
NFR-5-01	Фреймворк розроблення Angular	Обов'язковий	Середня
NFR-5-02	Використання готових бібліотек для роботи з векторною графікою	Обов'язковий	Середня
NFR-5-03	Використання готових бібліотек для сканування QR-коду	Обов'язковий	Середня
NFR-5-04	Програмний продукт має бути легко розширюваним та розумілим новим розробникам	Обов'язковий	Висока
NFR-6-01	Виведення повідомлень про помилки	Рекомендований	Низька
NFR-7-01	Однаковий стиль інтерфейсу та кольорові рішення	Обов'язковий	Середня
NFR-7-02	Зручний та зрозумілий інтерфейс сайта	Обов'язковий	Середня
NFR-7-03	Інтернет-адаптер 100 Мбіт/с	Не обов'язковий	Низька
NFR-7-05	Постійне підключення до інтернету	Не обов'язковий	Низька
NFR-7-06	Постійна взаємодія системи із сервером	Не обов'язковий	Низька

З огляду на сформульовані функціональні та нефункціональні вимоги, було визначено архітектуру ІНС, показану на рис. 2.1.

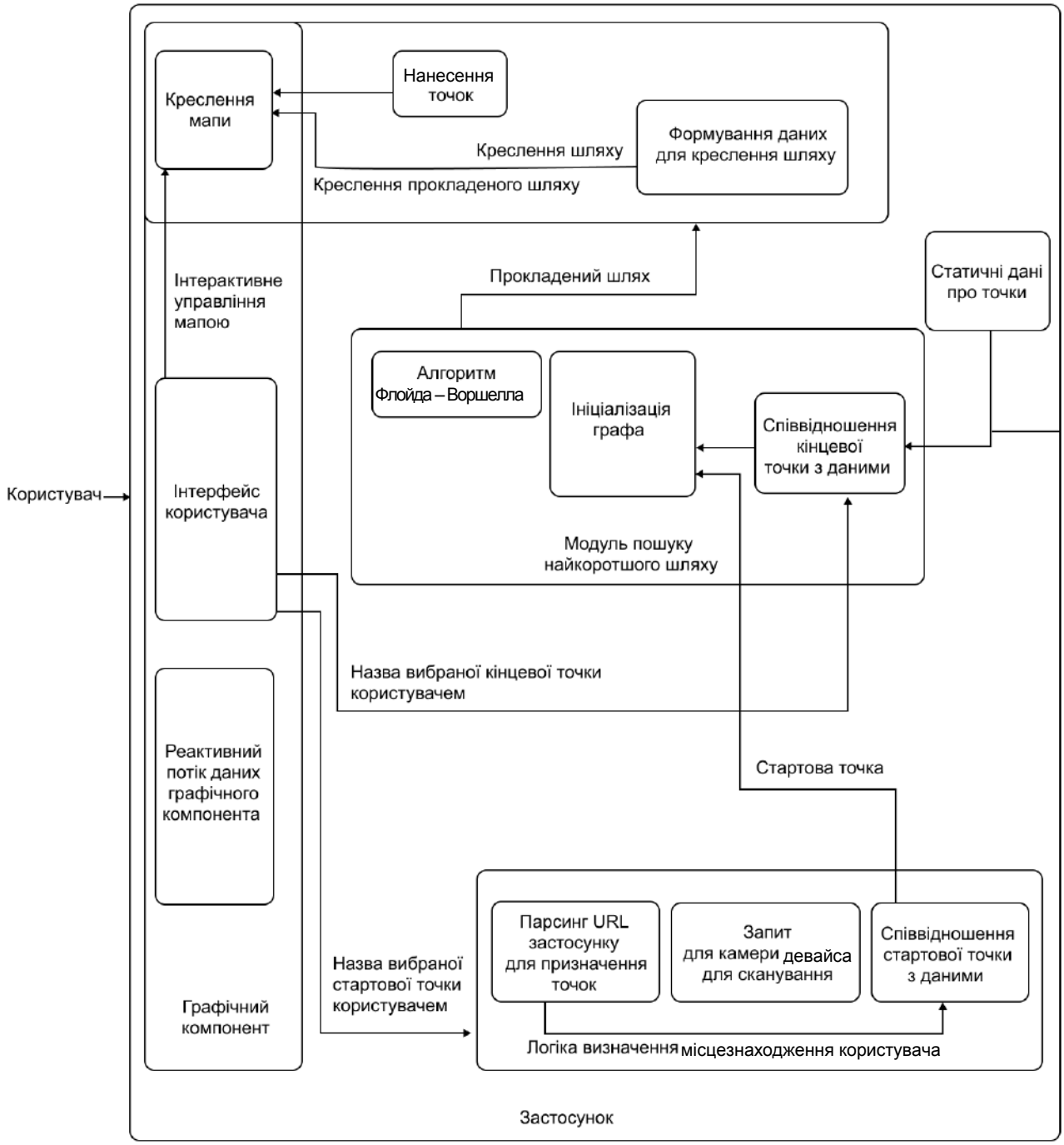


Рис. 2.1. **Архітектура ІНС університету як сукупність взаємопов'язаних модулів**

Архітектура ІНС є сукупністю взаємопов'язаних модулів (див. рис. 2.1).

У контексті інтерфейсних рішень, реалізованих в ІНС, ключове місце в архітектурі посідає *графічний компонент*, що є посередником між користувачем та всією бізнес-логікою застосунку від опрацювання даних до виконання складних алгоритмів пошуку місцезнаходження користувача

та побудови оптимального маршруту від стартової до кінцевої точки. Його роль полягає в наданні користувачеві зручного та зрозумілого інтерфейсу для взаємодії з функціоналом застосунку, а також візуалізації мапи та навігаційної інформації (назв конкретних локацій, додаткових позначок тощо). Проте, аналізуючи місце графічного компонента в архітектурі ІНС, слід підкреслити, що він забезпечує не тільки зовнішній вигляд застосунку разом із засобами взаємодії з користувачем, а й логіку взаємодії даних, які належать до візуальної частини ІНС (коефіцієнт масштабу мапи, позиція відносно екрана, вибрані користувачем налаштування тощо).

Загалом графічний компонент становить сукупність елементів інтерфейсу та програмного коду, який дозволяє користувачу взаємодіяти із системою, впливати на її стан, а також діставати певну інформацію. Крім цього, графічний компонент реалізує візуалізацію мапи приміщення та маршруту користувача для його орієнтування. Важливою складовою графічного компонента є інтерактивна мапа поточного поверху, для якої означений компонент забезпечує можливість зміни положення екрана відносно мапи, а також її масштабу за допомогою тачпаду пристрою.

Під час проєктування графічного компонента було враховано його зв'язки з іншими модулями ІНС та особливості обміну даними між ними. Отже, головним завданням стала реалізація такого управління даними, що відповідають за інтерфейс та стан інтерактивної мапи (коефіцієнт масштабу, позиція мапи відносно екрана, поточний поверх тощо), яке забезпечить коректне функціонування всієї системи та її подальше розширення. Відповідно, у процесі розроблення графічного компонента ІНС університету було реалізовано такий підхід, як:

- реалізація компонентів елементів управління інтерфейсу у вигляді шаблону, стилів та контролеру, html-, css- і ts-файлів компонентів;
- реалізація адаптивного та гнучкого інтерфейсу користувача як групування всіх елементів управління з додатковою логікою відображення;
- організація бізнес-логіки інтерфейсу у вигляді сервісів, що зберігають у собі дані та засоби їхнього опрацювання, використовуючи реактивне програмування за допомогою RxJs;
- організація утилітарних класів та методів, що дозволять винести повторювані частини опрацювання, які безпосередньо не належать до логіки роботи всього графічного компонента.

Відповідно до запропонованого підходу та визначених раніше функціональних вимог до ІНС, було реалізовано його проєктування як складової архітектури ІНС університету, що відповідає за інтерфейс та генерацію й опрацювання даних інтерактивної мапи.

Розроблено мокапи та прототипи інтерфейсу ІНС, а також дизайн схем будівлі для кожного окремого поверху. Мокапи було спроєктовано для максимальної ергономічності користування. Для цього екран було розподілено на три інтерактивні зони, які, залежно від розмірів екрана пристрою користувача, масштабують, змінюють відступи внутрішніх елементів та адаптують під зручне користування й досяжність. Схарактеризуємо виділені зони екрана.

Верхня зона – зона пошуку, де містяться елементи інтерфейсу, які забезпечують установлення стартової та кінцевої точок. Однією з найважливіших функцій ІНС є визначення місцезнаходження та пункту призначення для пошуку оптимального маршруту або орієнтування у приміщенні, адже застосунок не відстежує місцезнаходження користувача в реальному часі, прив'язане не до його реального місцезнаходження, а до стартових точок, що є фізичними QR-кодами. Отже, виникає потреба в самостійному виборі користувачем стартової й кінцевої точок для пошуку та візуалізації маршруту (або тільки кінцевої точки), для чого інтерфейсом ІНМ передбачено верхню зону екрана, яка містить дві форми для пошуку. Кожна форма складається з таких елементів управління (рис. 2.2):

1) іконка, що відображає призначення форми пошуку;

2) поле введення, куди користувач може вводити назву певної локації, де розташовано фізичний QR-код;

3) випадний список підказок за введеним користувачем текстом (рис. 2.3);

4) кнопки пошуку з іконкою збільшувальної лупи, яка виставляє введену точку як стартову або кінцеву за поточну, якщо така є, та запускає алгоритм пошуку маршруту, якщо обидві точки визначено.

Рис. 2.2. **Зона пошуку**

Рис. 2.3. **Випадний список із підказками**

Для того щоб користувач міг повноцінно використовувати ІНС, графічним компонентом передбачено елементи управління для зміни позиції мапи, її масштабу тощо. Такі елементи розміщено в лівій зоні екрана, щоб не заважати користувачеві розглядати мапу, забезпечувати ергономічне розташування без випадкового натискання на непотрібні елементи фалангою пальця під час тримання мобільного девайса. Ліва зона інтерфейсу користувача угруповує:

1) елементи управління масштабом мапи;

2) кнопку центрування мапи;

3) стрілки для перемикання поверхів та номером поточного поверху.

Указані елементи інтерфейсу зони управління мапою показано на рис. 2.4.

Рис. 2.4. **Елементи інтерфейсу зони управління мапою**

Інтерфейс графічного компонента ІНС також передбачає додаткові функції та можливості, які не є обов'язковими та належать до загальних елементів управління застосунком. Такі елементи управління було винесено у праву зону, де розміщено:

1) кнопку довідника з іконкою літери "і", яка відкриває вікно з інформацією про користування, підказками, тощо;

2) кнопку налаштувань з іконкою шестерні, яка відкриває вікно з перемиканням мови інтерфейсу;

3) кнопку з іконкою сканера, що дозволяє відкрити сканер QR-кодів та визначити точку, якій відповідає успішно відсканований код.

Далі наведено дизайн правої зони перелічених раніше елементів інтерфейсу користувача (рис. 2.5).

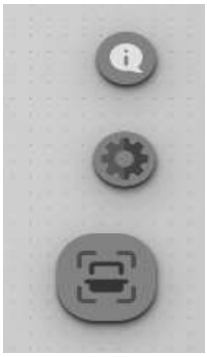


Рис. 2.5. **Елементи інтерфейсу користувача, розташовані у правій зоні екрана**

Кольорову гаму інтерфейсу ІНС було вибрано, з огляду на загальні фізичні умови використання застосунку (у добре освітлених приміщеннях університету), отже, зображення на екрані має бути досить яскравим та контрастним. Основним кольором інтерфейсу користувача було вибрано світло-блакитний (#358DFF), білий і відтінки сірого кольорів із використанням плавних та легких градієнтів. Застосовано м'які форми елементів управління та різноманітні іконки для спрощення сприйняття користувачем.

До інтерфейсних рішень, реалізованих графічним компонентом ІНС, слід зарахувати також розроблення дизайну інтерактивної мапи шляхом оцифрування схем поверхів приміщень університету за допомогою редактора векторних зображень Adobe Illustrator. Результат оцифрування схем деяких поверхів головного корпусу університету та їхні

інтерактивні мапи, що опрацьовують графічним компонентом ІНС, показано на рис. 2.6.

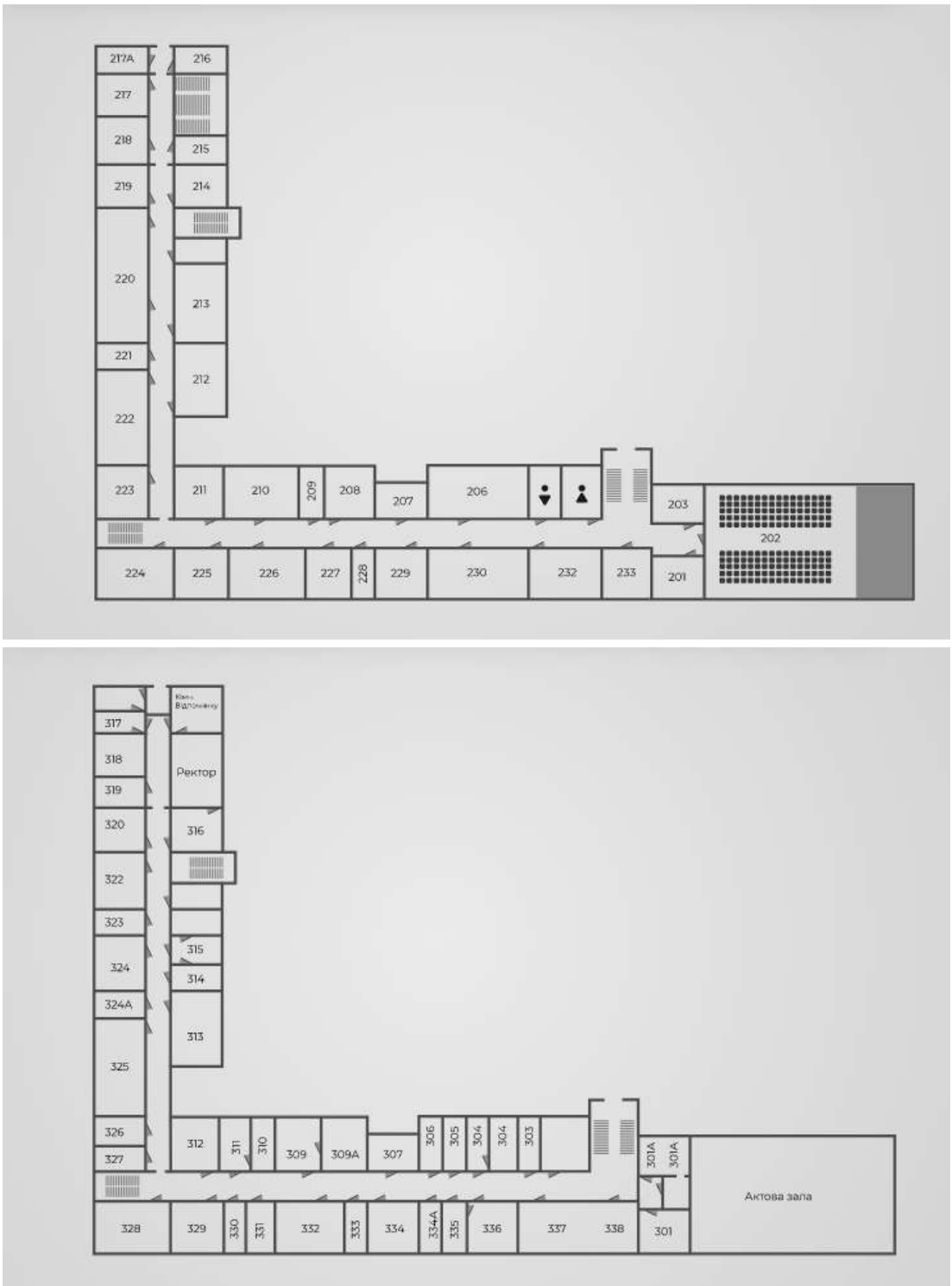


Рис. 2.6. **Результат оцифрування схем деяких поверхів головного корпусу університету та їхні інтерактивні мапи, що опрацьовують графічним компонентом ІНС**

Завдяки застосуванню саме векторної графіки, для кожного поверху було розроблено мапу у вигляді документа зі своїм синтаксисом і правилами, який описує відповідні графічні елементи зображення за допомогою атрибутів та формул. Це забезпечує динамічну взаємодію з таким документом за допомогою програмного коду, що дозволить надалі адаптувати вхідні дані графічного компонента для розширення можливостей ІНС щодо її застосування в інших приміщеннях. Крім цього, застосування оцифрованих мап, підготовлених засобами векторної графіки, дає перевагу перед растровими аналогами щодо швидкості завантаження та якості зображення.

Інтерфейсні рішення, застосовані під час розроблення графічного компонента, забезпечують адаптивність та гнучкість усій ІНС, завдяки чому її можна застосовувати як на комп'ютерах і ноутбуках, так і на мобільних пристроях. Водночас інтерфейс користувача відображається на екрані пристрою коректно. Адаптивний дизайн було розроблено шляхом урахування геометричних розмірів девайса, а також щільності віртуальних пікселів щодо фізичних, згідно з [23]. Також було враховано можливі налаштування масштабу відображення браузера та пристрою користувача.

На рис. 2.7 і 2.8 наведено варіанти відображення інтерфейсу користувача ІНС університету на різних пристроях. Як можна побачити, на відносно великих розмірах екрана застосунок має досить зручний та інтуїтивно розподілений інтерфейс, зберігаючи ті самі властивості й на екрані мобільного пристрою.

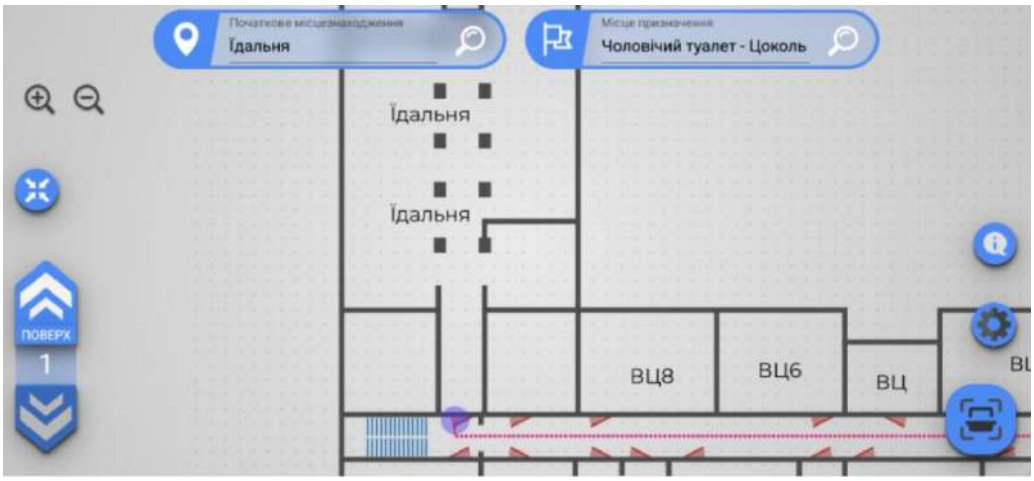


Рис. 2.7. **Вигляд інтерфейсу ІНС на ноутбуці з роздільною здатністю екрана 1366×768**

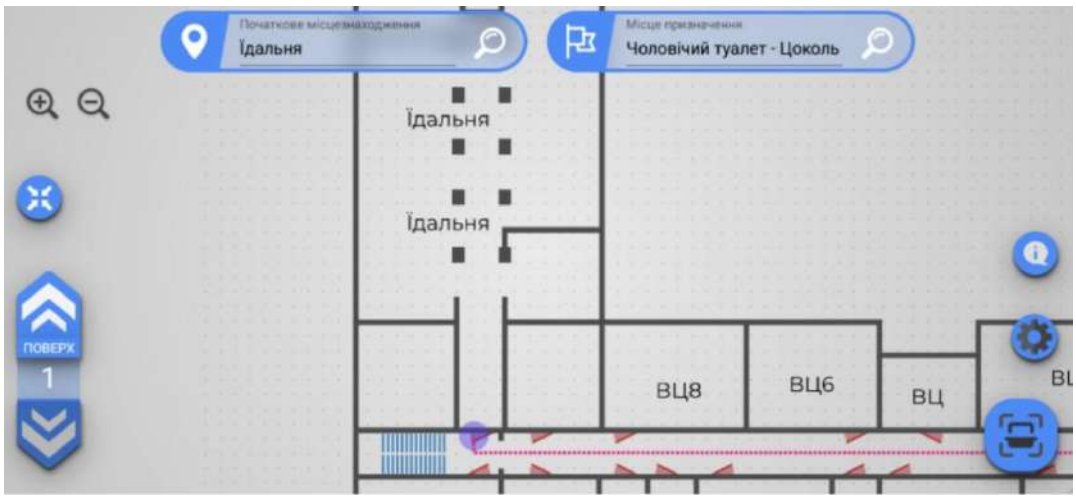


Рис. 2.8. **Інтерфейс користувача на смартфоні Galaxy A30 в альбомному режимі**

На мобільних пристроях із невеликими розмірами екрана, низькою щільністю пікселів або в портретному режимі змінюються відступи від країв екрана та між елементами управління, а також орієнтація розташування елементів управління масштабом мапи, щоб залишити достатньо місця для відображення елементів верхньої зони, кнопок центрування мапи та перемикача поверхів (рис. 2.9).

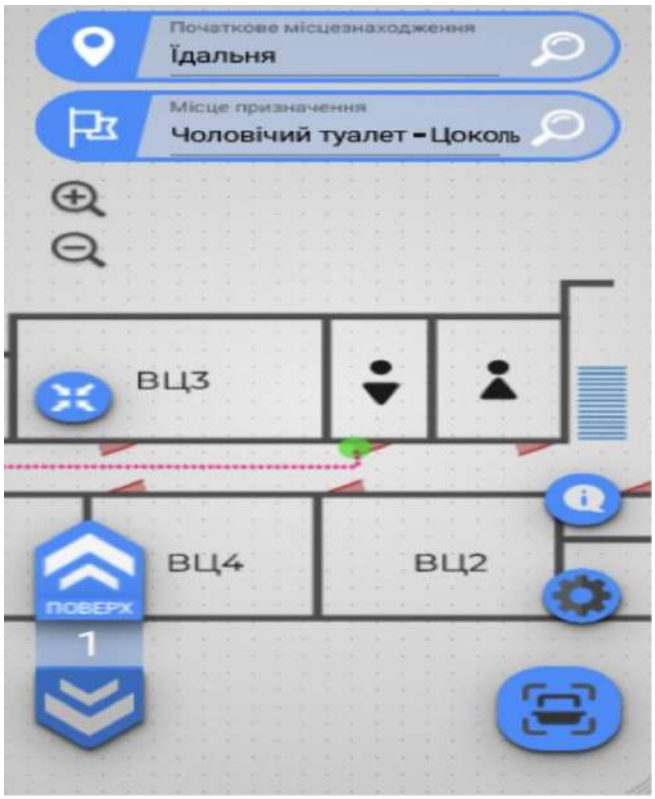


Рис. 2.9. **Інтерфейс користувача на смартфоні IPhone 5/SE в портретному режимі**

Інтерфейс добре адаптовано під старі моделі мобільних пристроїв, продукти компанії Apple, а також має підтримку для старих версій браузерів.

Однією із важливих інтерфейсних можливостей ІНС сучасного університету є її реалізована мовна локалізація, яка дозволяє легко змінити мову всіх інтерфейсних елементів. Приклад вибору англійської мовної локалізації системи показано на рис. 2.10.

Рис. 2.10. **Вибір англомовної локалізації із заміною мови для всіх елементів інтерфейсу ІНС**

Отже, інтерфейсні рішення, що реалізують графічним компонентом ІНС, такі:

- уможливлюють взаємодію з усіма підсистемами ІНС;
- дозволяють візуалізувати інтерактивну мапу приміщення з маршрутом користувача, побудованим відповідним модулем ІНС;
- забезпечують користувача гнучким та адаптивним інтерфейсом.

Формування алгоритмічних рішень проєктування ІНС здійснювали в кілька етапів. На етапі проєктування структури бази даних було здійснено концептуальне інфологічне проєктування, проєктування логічної та фізичної баз даних. Схарактеризуймо їх, адже саме структура даних дозволяє визначити, із яких основних класів та об'єктів буде складатися система, чим будуть оперувати алгоритми пошуку тощо.

Уся логіка роботи ІНС університету не потребує регулярного та динамічного створення, оновлення або видалення даних, проте потребує певного сховища статичних даних, які описують інформацію, необхідну для роботи навігаційної системи, щоб визначати шлях, показувати точки тощо. Використання повноцінних СУБД та сервера, який потребує регулярного інтернет-з'єднання, є недоцільним через те, що система має повноцінно працювати офлайн у режимі PWA.

У зв'язку із цим було ухвалено рішення використовувати локальний файл із серіалізованими даними формату JSON. Легкий формат обміну даними (JSON) побудовано на двох універсальних структурах даних [4]:

- колекція пар "ім'я – значення". Відповідно до мов програмування його реалізують як об'єкт, запис, словник, структура, хеш-таблицю, список ключів тощо;
- упорядкований список значень. У більшості мов це реалізують як масив, список або послідовність, вектор тощо.

Такий формат було вибрано також тому, що він є досить компактним і валідним описом сутностей мови JavaScript, що дозволяє легко читати такий файл, який є зручним для сприйняття людиною. Таке рішення дозволяє зберігати всі необхідні дані для роботи пошукових алгоритмів локально на сервері, який є хостингом застосунку. Під час звернення браузером на адресу застосунку ці дані буде автоматично завантажено паралельно із самим вебзастосунком, а під час використання PWA такі дані будуть повноцінно зберігати на пристрої користувача. Крім цього, гнучкий та ієрархічний характер документа бази даних дозволить розвиватися, відповідно до розвитку застосунку [23].

Вхідну та вихідну інформацію для ІНС подано документами, що надають для опрацювання та дістають у його результаті, відповідно. Список цих документів наведено в табл. 2.6.

Таблиця 2.6

Перелік вхідних та вихідних документів

№ з/п	Зміст документів	Вхідний/вихідний
1	Дані про місцезнаходження та/або кінцеву точку	Вхідний
2	Відображене місцезнаходження точки та/або прокладений маршрут	Вихідний

На основі аналізу вхідних та вихідних документів будують модель відображення множини реквізитів вихідних і вхідних документів на множину елементів даних, що підлягають збереженню в базі даних, далі виконують приведення зібраної інформації до вигляду, зручного для проєктування. Для цього складено словник даних (табл. 2.7).

Таблиця 2.7

Словник даних

№ з/п	Назва елементів	Ідентифікатори	Типи	Призначення елементів
1	2	3	4	5
1	Ідентифікатор точки	id	Рядковий	Унікальний номер кожної точки, незалежно від категорії та поверху (фактичний)
2	Назва точки	name	Рядковий	Назва точки (для категорії приміщень або міток)(фактичний)
3	Категорія точки	category	Рядковий	Категорія точки (приміщення, коридор, сходи, мітка) (фактичний)
4	Координата по горизонталі відносно схеми	x	Кількісний	X-координата точки на схемі поверху будівлі (фактичний)

Закінчення табл. 2.7

1	2	3	4	5
5	Координата по вертикалі відносно схеми	y	Кількісний	Y-координата точки на схемі поверху будівлі (фактичний)
6	Номер поверху	floor	Кількісний	Номер поверху, на якому розміщено відповідну точку (фактичний)
7	Ідентифікатор точки з категорією коридору	corridor	Рядковий	Код або коди точок коридорів або сходів, пов'язані з відповідною точкою (фактичний)

Для всіх сутностей розроблюваної системи було здійснено специфікацію обмежень цілісності, а саме: до обмеження атрибутів сутностей для коректності роботи з даними, а також визначення формату та властивостей. Специфікацію обмеження атрибутів сутностей наведено в табл. 2.8.

Таблиця 2.8

Обмеження атрибутів сутностей

№ з/п	Назви атрибутів	Межі / допустимі значення	Структури (формати)	Умови	Значення за замовчуванням
1	2	3	4	5	6
1	Ідентифікатор точки	Для категорії приміщень, коридорів та сходів код буде мати вигляд цілого числа >= 1, для категорії QR-кодів міток додатково надають префікс qr-	Рядковий тип, N/qr-N	Загальні	NOT NULL
2	Назва точки	Є тільки для категорії приміщень або QR-кодів міток	Рядковий тип, "Їдальня"	Загальні	NOT NULL

Закінчення табл. 2.8

1	2	3	4	5	6
3	Категорія точки	Один із: room, corridor stairs qr-code	Рядковий тип, room	Загальні	NOT NULL
4	Координата по горизонталі відносно схеми	0 <= n <= 3 500	Кількісний тип, NNNN	Загальні	NOT NULL
5	Координата по вертикалі відносно схеми	0 <= n <= 2 550	Кількісний тип, NNNN	Загальні	NOT NULL
6	Номер поверху	1 <= n <= 8	Кількісний тип, N	Загальні	NOT NULL
7	Ідентифікатор точки з категорією коридору	Одне значення для категорії приміщень та міток або масив значень для сходів і коридорів	Рядковий тип, N/ [N, N]	Загальні	NOT NULL

Отже, можна узагальнити обмеження в такий спосіб. Кожен код точки має бути унікальним. Кожній точці категорії room або qr-code буде належати назва. Кожному коридору або сходам може відповідати одне або кілька значень коду коридору або сходів. Кожне приміщення або мітка може бути пов'язано лише з одним кодом коридору.

У процесі проєктування логічної моделі даних було враховано те, що дані не оновлюються користувачами, не піддаються регулярним запитам, не підлягають паралельним запитам та їхнє зберігання відбувається на сервері самого застосунку, тому вага даних безпосередньо впливає на швидкість первинного завантаження застосунку. Отже, дані мають зберігати в компактному форматі, який буде легко опрацювати застосунком.

Також до уваги було взято те, що під час розподілу основної сутності точок за категоріями в разі коридорів та сходів будуть зв'язки між сутностями однієї категорії (кожна точка коридору має зв'язки з іншими точками коридору, кожна точка сходів має зв'язки з іншими точками

сходів). У результаті було побудовано діаграму "сутність – зв'язок" для логічної моделі (рис. 2.11).

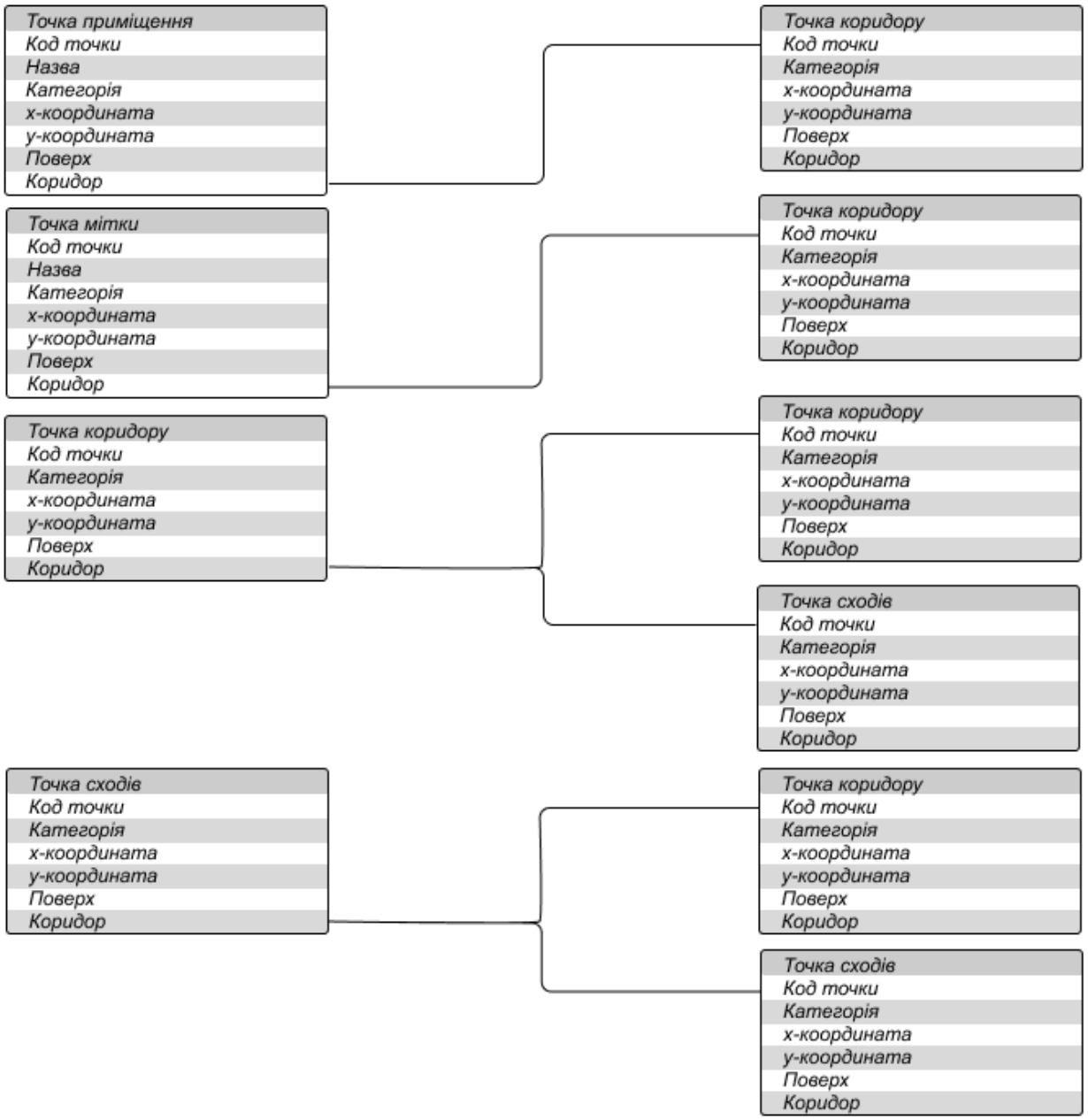


Рис. 2.11. **Діаграма "сутність – зв'язок" для логічної моделі**

Для більш конкретного розгляду структури даних у вигляді сутностей та відношень між ними, відповідно до способу їхнього збереження, спроєктовано фізичну модель даних, показану на рис. 2.12.

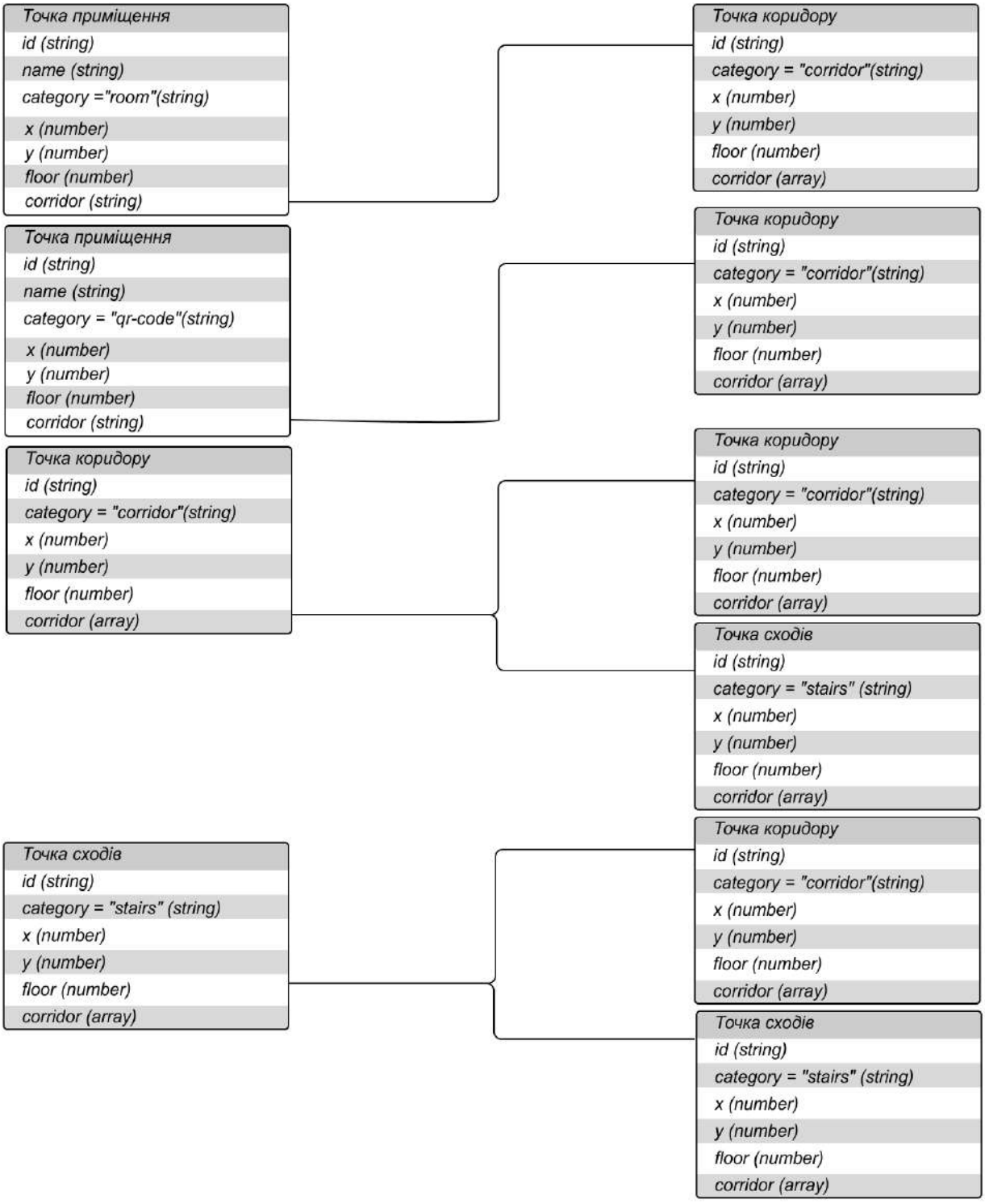


Рис. 2.12. **Діаграма "сутність – зв'язок" для сутностей фізичної моделі**

На наступному етапі відбулося моделювання загальної структури ієрархії класів системи та їхньої семантики за допомогою UML-діаграми класів із дотриманням основних принципів архітектури ІНС університету:

- усю бізнес-логіку системи, включно з тією, що належить до компонентів інтерфейсу користувача, має бути розміщено в сервісах;
- для уникнення дублювання коду, що загалом не належить до бізнес-логіки сервісів, створюють утилітарні класи та методи;
- усі дані не примітивного типу мусять мати інтерфейси, що їх описують.

Діаграми класів, які моделюють структуру ієрархії класів системи та їхню семантику з дотриманням указаних принципів, показано на рис. 2.13 – 2.16.

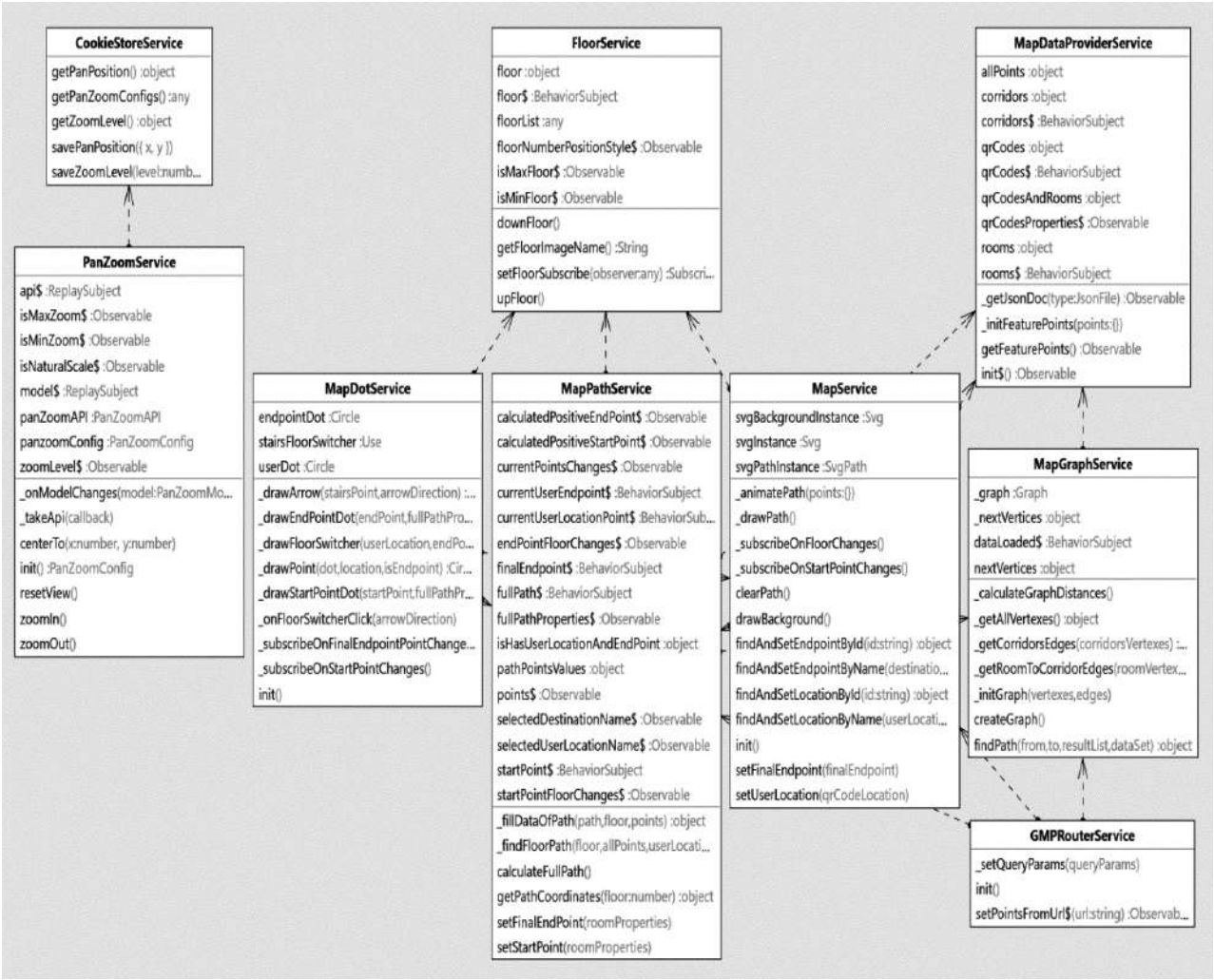


Рис. 2.13. **UML-діаграма класів сервісів**

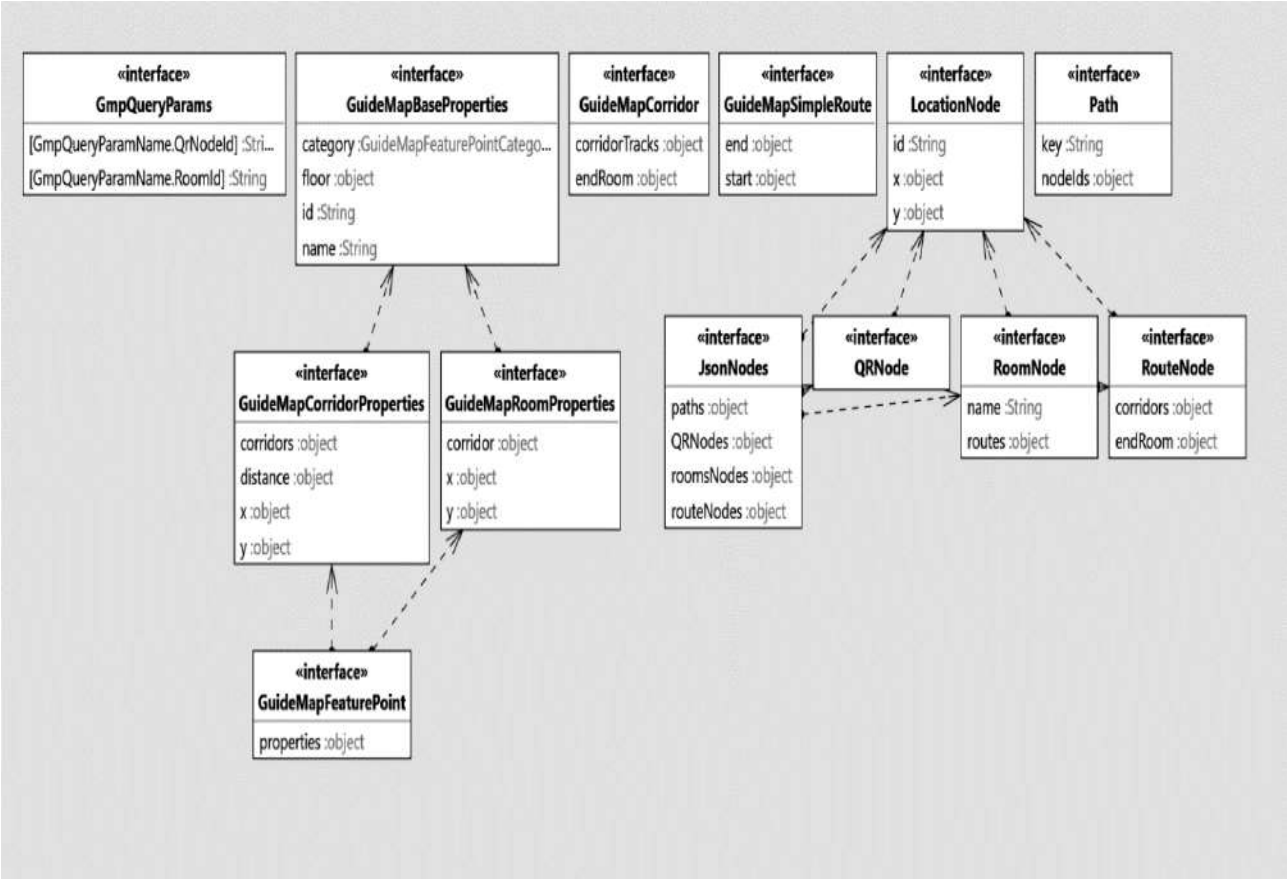


Рис. 2.14. **UML-діаграма класів інтерфейсів**

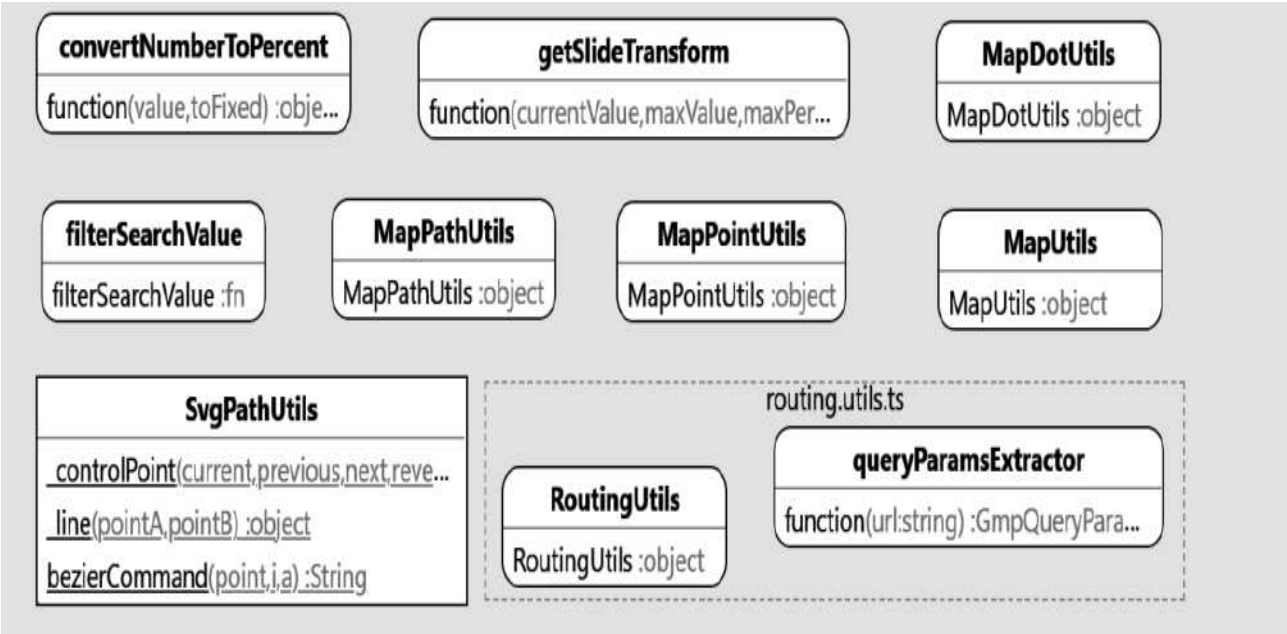


Рис. 2.15. **UML-діаграма класів утилітарних методів та функцій**

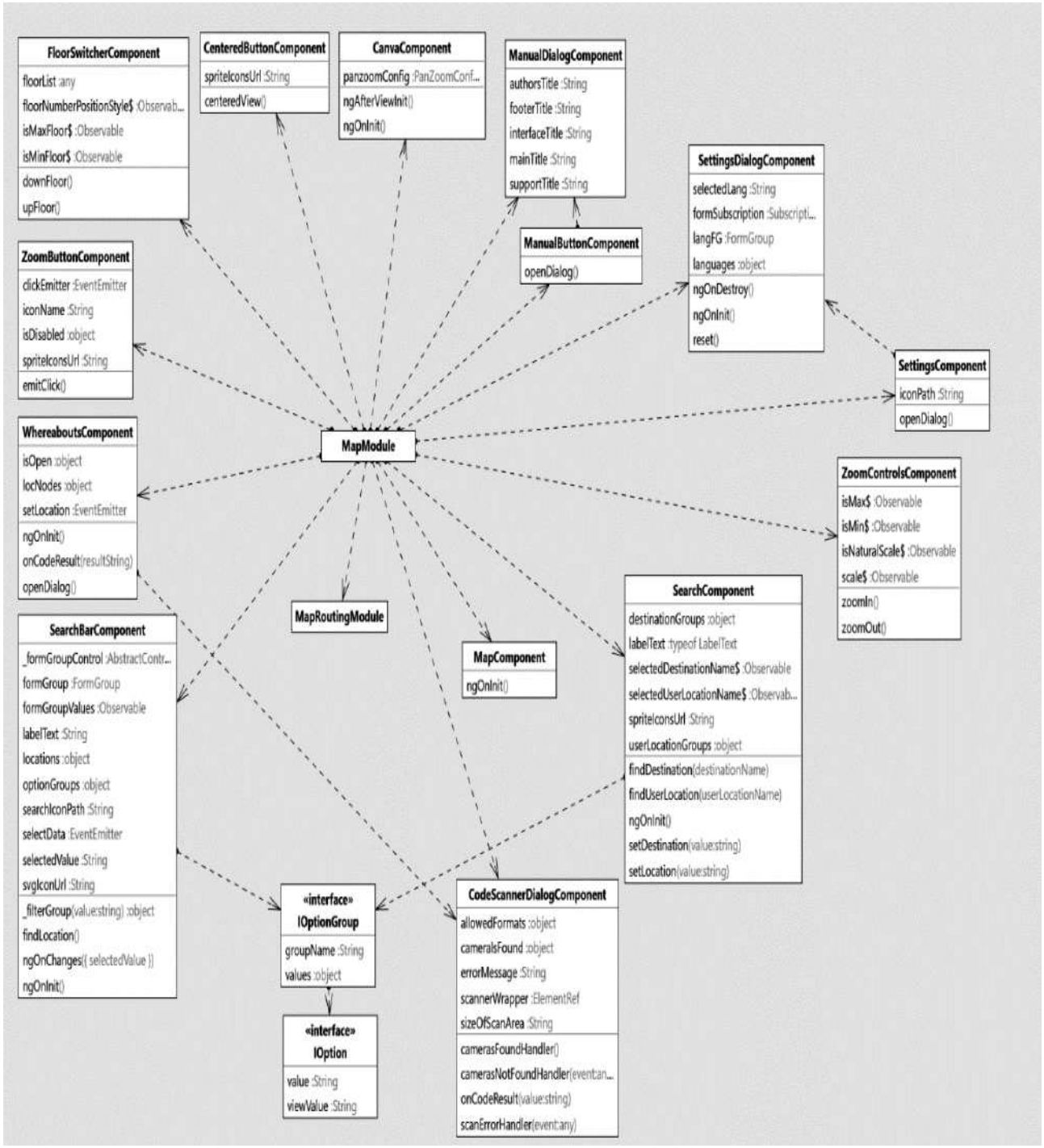


Рис. 2.16. **UML-діаграма класів компонентів графічного інтерфейсу**

У контексті формування алгоритмічних рішень у процесі проєктування ІНС суттєвого значення набуває реалізація модуля пошуку найкоротшого шляху як складової цілісної архітектури всієї системи, вхідними даними якого є місце знаходження (стартова точка) та місце призначення (кінцева точка).

Сформулюймо математичне формулювання завдання, яке виконує модуль пошуку найкоротшого шляху. Нехай задано граф:

$$G = (V,E), \tag{2.1}$$

де V – множина вершин;

E – множина ребер із заданими вартостями c_{ij} на кожному ребрі (i, j).

У термінах нашої предметної галузі вершинами графа є різноманітні локації приміщення та мітки QR-кодів, ребрами графа є фізичні зв'язки між вершинами, які дозволяють до них дістатися.

Потрібно визначити найкоротший шлях між вибраними вершинами. Це означає, що потрібно визначити такий шлях P між вершинами ν_1 та ν_2 $P(\nu_1,\nu_2,...,\nu_n)$, щоб

$$\sum_{i=1}^{n} C_{ij} \to \min. \tag{2.2}$$

Ця проблема в англомовних наукових джерелах також має назву The Shortest Path Problem (SPP) [2; 5].

Є кілька різних, але пов'язаних між собою завдань, у яких необхідно визначити найкоротші шляхи у графі:

1. Найкоротший шлях s – t: визначити найкоротший шлях від s до t (однієї вибраної пари вершин).

2. Пошук найкоротших шляхів з одним джерелом у графі: визначити найкоротший шлях від s до всіх вузлів.

3. Пошук для всіх пар вершин найкоротших шляхів: визначити SPP між кожною парою вузлів.

Крім цього, можна розрізняти також завдання пошуку найкоротших шляхів, залежно від типу графів, заданих як вхідні дані.

Для вирішення конкретного практичного завдання пошуку найкоротшого шляху для навігації у приміщенні було здійснено відповідний аналіз щодо вибору адекватного алгоритму пошуку серед таких відомих алгоритмів, як Дейкстри, Беллмана – Форда, Флойда – Воршелла, Джонсона та ін. [43]. Основними критеріями вибору для цього завдання є властивості графа, а також час виконання алгоритму. У результаті було вибрано два найбільш популярні алгоритми: алгоритм Дейкстри та Флойда – Воршелла.

Під час вибору між ними було взято до уваги такі їхні характеристики. Найбільша різниця між алгоритмами полягає в тому, що алгоритм Флойда визначає найкоротший шлях між усіма вершинами, а алгоритм

Дейкстри – найкоротший шлях між окремою вершиною та усіма іншими вершинами. Водночас витрати на алгоритм Дейкстри є значно вищими, ніж на алгоритм Флойда. Якщо запустити алгоритм Дейкстри *n* разів на *n* різних вершинах теоретичні труднощі в часі становлять $O(n \cdot n^2) = O(n^3)$. Інакше кажучи, у разі використання алгоритму Дейкстри для визначення шляхів від кожної вершини до будь-якої іншої вершини, маємо таку саму ефективність і результат, як і в разі використання алгоритму Флойда. Звичайно, конкретне завдання може бути вирішене шляхом багатократного застосування алгоритму Дейкстри з послідовним вибором кожної вершини графа як початкової вершини. Проте алгоритм Флойда є більш ефективним, ніж багатократне повторення алгоритму Дейкстри. Водночас алгоритм Флойда є набагато простішим у реалізації. У зв'язку із цим вибір було зроблено на користь алгоритму застосування Флойда – Воршелла.

Схарактеризуємо його характеристики та особливості. Хоча цей алгоритм має назву Флойда – Воршелла і його появу в інформаційному просторі датовано початком 1960-х рр., він насправді містить доробки раніше опублікованих праць Б. Роя (1959 р.) та тісно пов'язаний з алгоритмом Кліні, опублікованим 1956 р., для перетворення дестрокового кінцевого автомата на регулярний вираз. Сучасне формулювання алгоритму у вигляді трьох укладених циклів *for* було вперше здійснено П. Інґерманом 1962 р. Алгоритм Флойда – Воршелла у своїй основі дотримується методу динамічного програмування, що визнано в роботах Р. Флойда [43].

Це дозволяє використовувати його для розв'язання багатьох проблем, як-от:

оптимальна маршрутизація;

визначення найкоротших шляхів у графах;

транзитивне замикання графів;

обчислення подібності між графіками та ін.

Як зазначалося раніше, алгоритм Флойда – Воршелла ґрунтується на використанні методу динамічного програмування, що є альтернативою вирішення завдання методом грубої сили або жадібними алгоритмами. У загальній інтерпретації цей метод складається з таких етапів:

1. Розподіл завдання на підзавдання меншого розміру.

2. Визначення оптимального рішення підзавдань рекурсивним методом.

3. Використання визначеного рішення підзавдань для конструювання рішення вихідного завдання.

Покрокове виконання алгоритму Флойда – Воршелла на основі використання методу динамічного програмування наведено на рис. 2.17 [2].

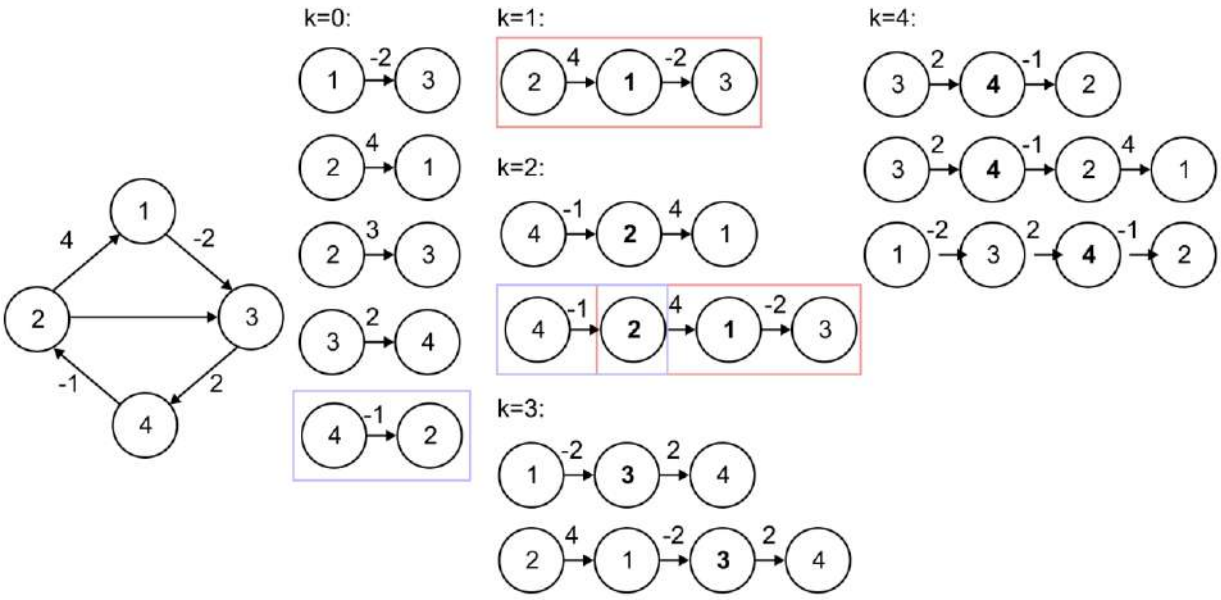


Рис. 2.17. **Покрокове виконання алгоритму Флойда – Воршелла**

Характеризуючи це алгоритмічне рішення, зауважимо, що для визначення найкоротших шляхів між усіма вершинами графа використовують не перебір усіх можливостей, що призведе до значного часу роботи алгоритму та займе більше пам'яті, а висхідне динамічне програмування. Отже, усі підзавдання, які згодом знадобляться для вирішення вихідного завдання, виконують заздалегідь і потім використовують їхні рішення.

Математичну інтерпретацію алгоритму Флойда – Воршелла можна подати в такий спосіб [45]. Нехай вершини графа послідовно пронумеровано від 1 до n. Алгоритм використовує матрицю V розміру $n \times n$, у якій обчислюють довжини найкоротших шляхів.

Спочатку елемент (i, j) дорівнює відстані d_{ij} від вузла i до вузла j, якщо є ребро (i, j), (якщо ні, він дорівнює нескінченності). Кожен діагональний елемент матриці V дорівнює нулю. Над матрицею V виконують n ітерацій.

Після k-ї ітерації елемент $V[i, j]$ має значення найкоротшого шляху з вершини i до вершини j, що не проходить через вершини з номером більшим ніж k, що означає, що між кінцевими вершинами i та j можуть

бути лише вершини, номери яких менші або дорівнюють k. На k-й ітерації для обчислення елементів матриці V використовують таку формулу:

$$V_k[i, j] = \min\{V_{k-1}[i, j], V_{k-1}[i, k] + V_{k-1}[k, j]\}, \quad (2.3)$$

яку інтерпретують таким способом.

Нехай є три вузли i, j і k та задані відстані між ними: $V_{k-1}[i, j], V_{k-1}[j,k]$ та $V_{k-1}[i,k]$, відповідно.

Для обчислення $V_k[i, j]$, $A_k[i, j]$ виконують порівняння величини $V_{k-1}[i, j]$ (тобто вартість шляху від вершини i до вершини j без участі вершини k) із величиною (тобто сума вартості шляхів між іншими двома вершинами). Якщо шлях через проміжну вершину k має меншу вартість, ніж $V_{k-1}[i, k]+V_{k-1}[k, j]$, то величину $A_k[i, j]$ замінюють.

Як видно із наведеного на рис. 2.18 псевдокоду [23], що відтворює алгоритм Флойда – Воршелла, його реалізують у три цикли *for*. Зовнішній цикл вибирає вершину, через яку пробуймо визначити оптимальний шлях, внутрішні два цикли перебирають пари вершин, між якими визначаємо оптимальний шлях. У внутрішньому циклі, якщо між вершинами є шлях і він не містить пересадок (прямий шлях), то зберігаймо його й переходимо до наступної кінцевої вершини, інакше відбувається розрахунок труднощів маршрутів та їхнього порівняння.

```
let dist be a |V| × |V| // масив мінімальних відстаней
for each edge (u, v) do
                    dist[u][v] ← w(u, v)  // вага ребра (u,v)
                    for each vertex v do
                        dist[v][v] ← 0
                    for k from 1 to |V|
                    for i from 1 to |V|
                    for j from 1 to |V|
                    if dist[i][j] > dist[i][k] + dist[k][j]
                    dist[i][j] ← dist[i][k] + dist[k][j]
                    end if
```

Рис. 2.18. **Псевдокод, що відтворює алгоритм Флойда – Воршелла**

Якщо новий шлях через проміжну вершину за труднощами є таким самим, як і наявний, то додаймо його як новий варіант шляху; якщо він

є кращим, то замінюймо ним наявний шлях; якщо є гіршим, то залишаймо попередньо визначений шлях. Якщо наявного шляху між двома розглянутими вершинами немає, то створюймо новий шлях через проміжну вершину та зберігаймо його в матрицю.

Отже, алгоритм Флойда – Воршелла розв'язує проблему пошуку найкоротшого шляху для всіх пар вузлів у графі, навіть для графів із негативними вагами ребер. Асимптотичні труднощі такої процедури становлять $\Theta\left(|V|^3\right)$. Відповідно до джерел, якщо використати структуру даних фібоначчієвої купи, то труднощі можна оптимізувати до $O(VE\log(V))$, де V – кількість вершин, а E – кількість ребр ("O"-велике) [45].

Для програмного розроблення модуля пошуку найкоротшого шляху вибраним алгоритмом (як складової архітектури ІНС університету), було створено відповідну структуру даних, що відтворює граф, на якому здійснюють пошук та який може легко розширюватися, коли додають нові вершини та ребра.

Із цією метою побудовано клас Graph, його методи та необхідні властивості, що дозволяє додавати нові вершини, видаляти та коригувати їх.

Структуру класу Graph з описом призначення його властивостей та методів наведено в табл. 2.9.

Таблиця 2.9

Структура класу Graph з описом призначення його властивостей та методів

Назви властивостей, методів	Призначення
Vertices	Використовують для зберігання вершин графа
Edges	Використовують для зберігання ребер графа
AddVertex	Метод додає вершину графові
GetVertexByKey	Клас ребра графа
GetNeighbors	Клас вершини графа
GetAllVertices	Клас пов'язаного списку
GetAllEdges	Клас пов'язаних вузлів
AddEdge	Метод додавання ребра
DeleteEdge	Метод видалення ребра
FindEdge	Метод пошуку ребра
FindVertex	Метод пошуку вершини

Вхідну та вихідну інформацію для модуля пошуку найкоротшого шляху подано документами, що надають для опрацювання й дістають у його результаті, відповідно.

Список цих документів наведено в табл. 2.10.

Таблиця 2.10

Інформаційний перелік документів

№ з/п	Назви документів	Вхідний/вихідний
1	Координати стартової (місцезнаходження користувача) та кінцевої (місце призначення) точок	Вхідний
2	Оптимальний маршрут	Вихідний

Отже, архітектуру модуля пошуку найкоротшого шляху та його структуру, показану на рис. 2.19 і 2.20, було побудовано в такий спосіб.

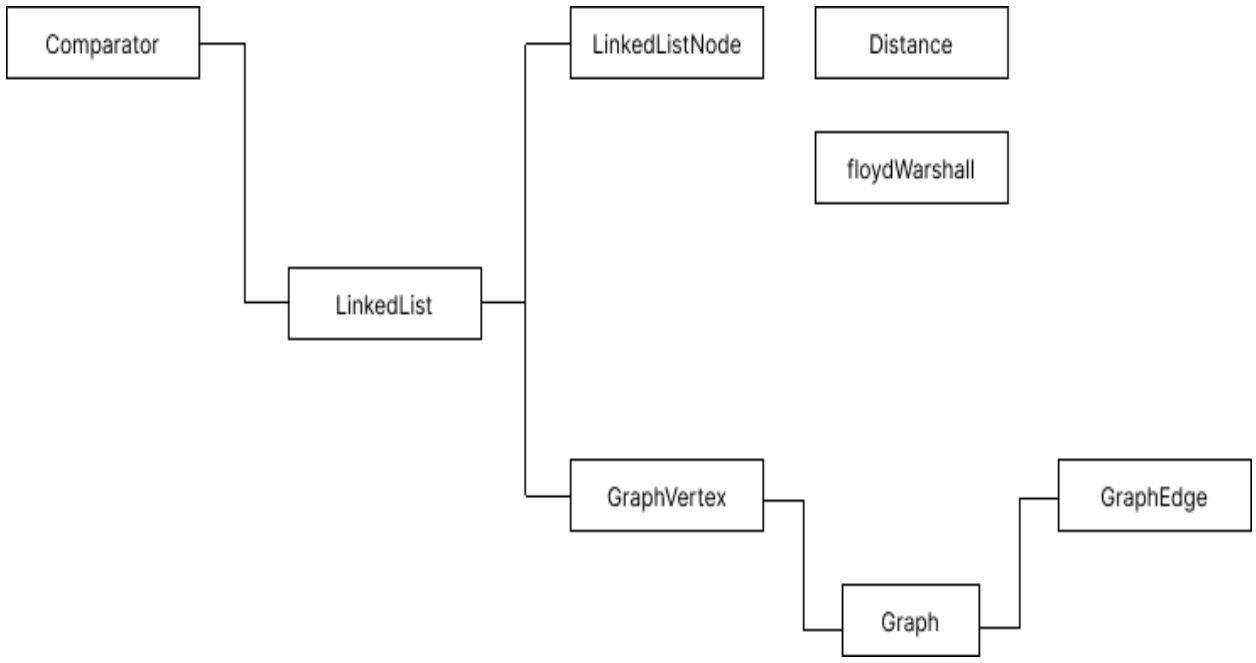


Рис. 2.19. **Архітектура модуля пошуку найкоротшого шляху**

Рис. 2.20. **Структура модуля пошуку найкоротшого шляху**

Розгляньмо призначення всіх складових модуля та допоміжних утиліт, наведених у табл. 2.11.

Таблиця 2.11

Призначення складових модуля пошуку найкоротшого шляху та допоміжних утиліт

Назви модулів, папок	Призначення
Comparator.ts	Використовують для порівняння двох вершин
Distance.util.ts	Утиліта для вимірювання дистанції між двома вершинами графа
FloydWarshall.ts	Функція з реалізацією алгоритму Флойда – Воршелла
Graph.ts	Клас із головною структурою графа
GraphEdge.ts	Клас ребра графа
GraphVertex.ts	Клас вершини графа
LinkedList.ts	Клас пов'язаного списку
LinkedListNode.ts	Клас пов'язаних вузлів
Graph.service.ts	Сервіс для зберігання графа та логіки пошуку маршруту

Зупинімося більш детально на характеристиці сервісу Graph.service.ts, призначеному для зберігання графа та алгоритму пошуку маршруту.

У ньому міститься логіка роботи з даними, що завантажують із JSON-файла, який зберігають у клієнтський частині програми. Структуру сервісу Graph.service.ts з описом призначення його властивостей та методів наведено в табл. 2.12.

Таблиця 2.12

Структура сервісу Graph.service.ts із описом призначення його властивостей та методів

Назви властивостей, методів	Призначення
_graph	Клас для зберігання графа
_nextVertices	Матриця з найкоротшими маршрутами між кожною вершиною
createGraph	Використовують для створення графа
getAllVertexes	Метод для визначення всіх імовірних вершин
_initGraph	Метод ініціалізації графа
_getRoomToCorridorEdges	Метод будує зв'язок між кімнатами та коридорами
_getCorridorsEdges	Метод будує зв'язок між коридорами
_calcualteGraphDistances	Метод, який виконує розрахунок усіх дистанцій та коротких маршрутів

Схарактеризуймо роботу основних методів сервісу Graph.service.ts.

Спочатку застосовують метод createGraph, який визначає всі координати та створює для кожної координати свою вершину за допомогою унікального номера.

Далі застосовують метод _getCorridorEdges, який будує всі зв'язки між коридорами та сходами. Алгоритм роботи цього методу полягає в тому, щоб циклічно перебрати всі вершини коридорів, визначити точки, які можуть бути поєднаними (мати ребро графа), та створити цей зв'язок із назвою oneToOneCorridorEdge за допомогою класу GraphEdge. Крім цього, під час створення ребра графа виконують розрахунок дистанції між двома вершинам. Далі побудоване ребро додають до масиву з новими ребрами.

Наступним застосовують метод _getRoomCorridorEdges, який утворює зв'язки (ребра графа) між аудиторіями, qr-кодами (або іншими

приміщеннями). Алгоритм роботи цього методу є схожим до логіки, реалізованої методом _getCorridorEdges, проте тут визначають за всіма локаціям та коридорам, до яких належать локації.

Далі метод _initGraph ініціалізує граф, тобто заповнює його ребрами та вершинами, визначеними методами getCorridorEdges та _getRoomCorridorEdges.

Коли граф ініціалізовано, застосовують алгоритм Флойда – Воршелла, який розраховує найкоротші шляхи для всіх пар вершин у графі.

Після створення графа та застосування алгоритму Флойда – Воршелла (функція floydWarshall.ts) стороння програма, код чи сервіс може запросити найкоротший маршрут між вказаними вершинами графа, передавши два аргументи, from і to, що є назвами стартової та кінцевої точок пошуку, відповідно.

Із класу Graph визначають необхідні вершини та їхні індекси. Далі здійснюють перевірку наявності маршруту в матриці, визначеній, завдяки реалізації алгоритму Флойда – Воршелла. Якщо маршрут наявний, перевіряють, чи наступна вершина є кінцевою вершиною маршруту (тоді рекурсію мають зупинити). Якщо в матриці ще є вершини, рекурсію продовжують. Під час роботи рекурсії заповнюють масив з оптимальним маршрутом. Якщо маршруту в матриці немає, повертається порожній масив, тобто маршрут між вказаними вершинами не визначено.

У результаті визначають масив GmpSimpleRoute, який має інтерфейс, зображений на рис. 2.21.

```
export interface GmpSimpleRoute {
readc`ly start: staring;
readc`lu end: staring;
}
```

Рис. 2.21. **Інтерфейс масиву GmpSimpleRoute з визначеним маршрутом**

Наведімо деякі приклади результатів визначення найкоротшого маршруту. Результат пошуку та відображення оптимального шляху від початкового місцезнаходження до шуканої локації, розміщених на одному поверсі, показано на рис. 2.22. Червоною пунктирною лінією зображено

визначений оптимальний шлях з однієї вершини (їдальні) до іншої (аудиторії обчислювального центру).

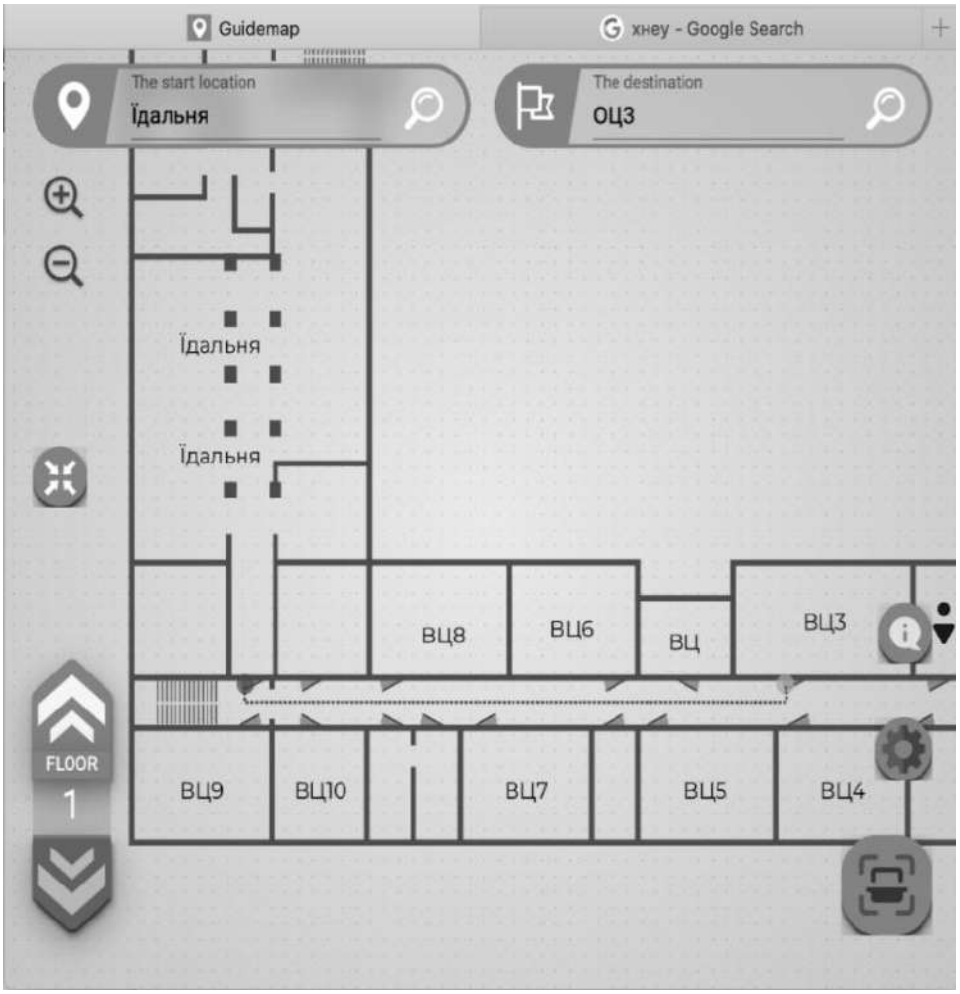


Рис. 2.22. **Результат пошуку та відображення оптимального шляху від початкового місцезнаходження до шуканої локації, розміщених на одному поверсі**

Крім цього, реалізовано можливість пошуку та відображення маршрутів між двома локаціями, розміщеними на різних поверхах.

На рис. 2.23 і 2.24 червоною пунктирною лінією візуалізовано оптимальний маршрут зі стартової локації (чоловічого туалету, розміщеного на цокольному поверсі) до сходів на другий поверх (див. рис. 2.23 та зі сходів – кінцевої локації (лекційної аудиторії 222, розміщеної на другому поверсі головного навчального корпусу університету) (див. рис. 2.24).

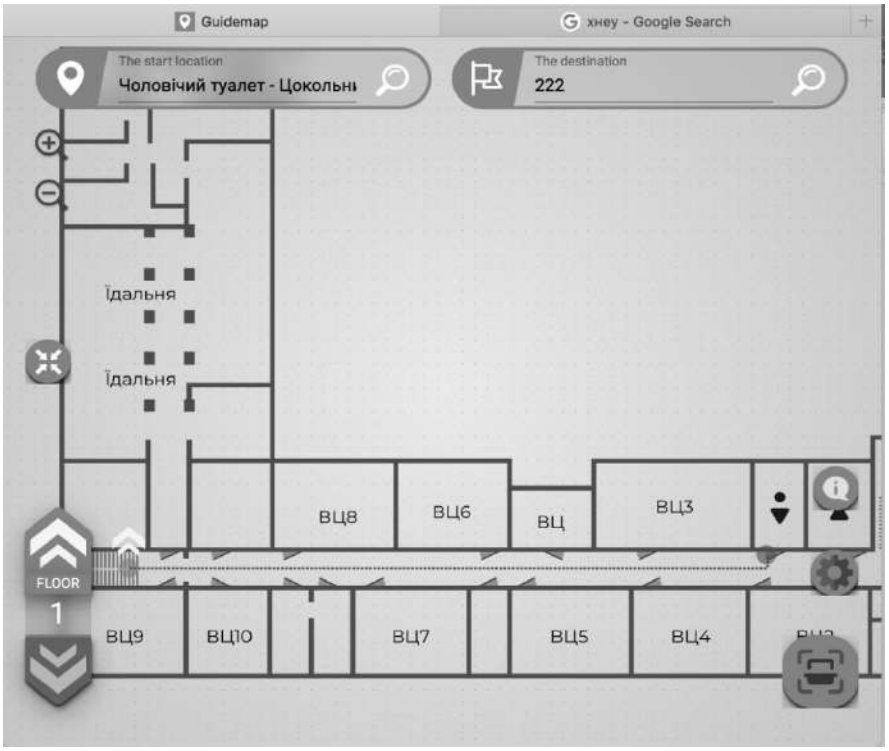


Рис. 2.23. **Оптимальний шлях зі стартової локації на цокольному поверсі до сходів на другий поверх**

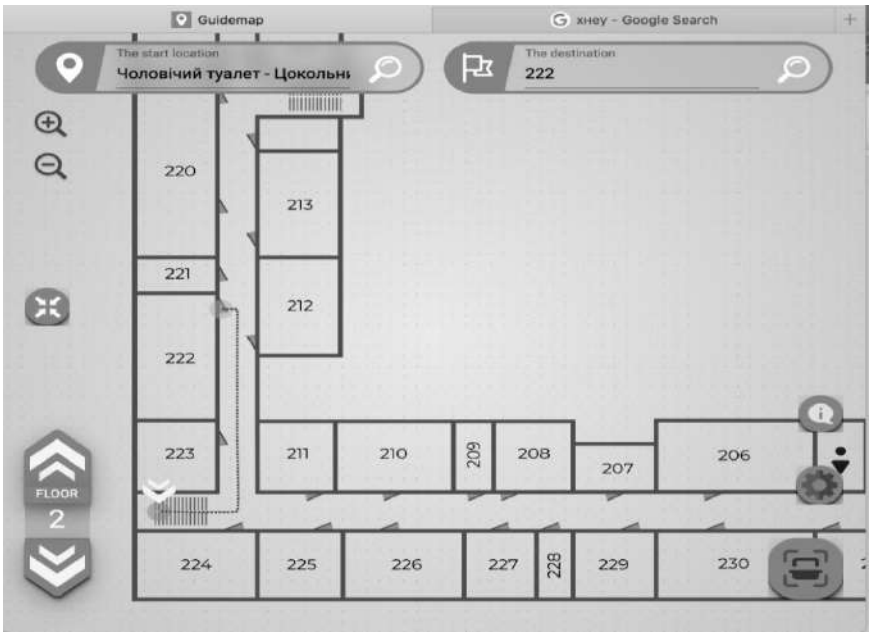


Рис. 2.24. **Візуалізований оптимальний шлях зі сходів на другому поверсі до кінцевої локації**

Реалізації як інтерфейсних, так і алгоритмічних рішень, висвітлених раніше, сприяє модуль визначення місцезнаходження користувача, що є посередником між основними компонентами ІНС, відповідно до її архітектури.

Відповідно до діаграми варіантів використання, за допомогою ІНС користувач може визначити своє місцезнаходження двома шляхами: вибрати його зі списку доступних або дістати його контекстно у приміщенні університету шляхом сканування QR-коду мітки, розташованої біля певних локацій.

Для того щоб вибрати місцезнаходження зі списку, користувач має здобути одним зі способів доступ до застосунку та на його головному екрані визначити відповідне поле під назвою "Початкове місцезнаходження". Після натискання на це поле користувач зможе вибрати місцезнаходження, виконавши пошук за назвою в цьому полі або визначивши необхідний пункт із потрібною локацією у випадному списку, що відкрився. Визначення місцезнаходження шляхом вибору зі списку застосовують у разі якщо:

користувач бажає завчасно спланувати свої переміщення та перебуває поза територією закладу;

користувач точно знає своє місцезнаходження та не потребує додаткової допомоги в позиціонуванні.

Цей спосіб реалізують через взаємодію користувача із графічним інтерфейсом застосунку.

Для того щоб визначити місцезнаходження за допомогою QR-коду мітки, також необхідно здобути доступ до застосунку одним зі способів та натиснути відповідну кнопку на головному екрані. Після натискання на цю кнопку відкриється вікно сканера QR-кодів. На цьому етапі ІНС може відправити запит на здобуття дозволу для доступу до камери електронного пристрою користувача, якщо його не було надано раніше. Після здобуття дозволу у вікні сканера користувач буде бачити зображення з камери свого пристрою та має навести її на розташовану в закладі мітку з QR-кодом. Сканер виконає розшифрування коду та опрацювання даних. Позиціонування відбувається відносно мітки із QR-кодом, тобто місцезнаходженням користувача вважають фізичне розташування самої мітки.

Позиціонування шляхом сканування QR-коду застосовують у разі, якщо користувач перебуває на території закладу, але точно не знає свого місцезнаходження та потребує допомоги у його визначенні. Доцільність використання такого способу пояснено тим, що серед наявних методів позиціонування (глобальної системи позиціонування, позиціонування відносно радіо- або Bluetooth-міток) визначення місцезнаходження

за QR-кодами міток має найбільшу точність у поєднанні з найбільш простою реалізацією та найменшою вартістю впровадження [30].

Визначення місцезнаходження реалізують через опрацювання визначених із QR-коду даних. Кожна мітка містить посилання на застосунок із певним кодом збереженої точки. Отже, сторонній користувач, який не має посилання, може здобути доступ до застосунку, відсканувавши код будь-яким сканером, які часто постачають на електронні пристрої разом з операційними системами [30]. Якщо QR-код відскановано в самому застосунку, із його вмісту буде визначено код збереженої точки та виконано пошук серед збережених об'єктів точок.

Результатом позиціонування користувача за допомогою цього модуля ІНС будь-яким із цих способів є визначений у сховищі збережений об'єкт точки, що містить координати, за якими візуалізують місцезнаходження на схемі поверху приміщення та які будуть використовувати надалі як відправну точку маршруту іншими модулями застосунку для навігації у приміщеннях університету.

Для забезпечення роботи модуля визначення місцезнаходження користувача у складі ІНС, на електронному пристрої користувача необхідно зберігати певні відомості, а саме детальний план будівель та опис приміщень закладу. Він має містити координати точок коридорів, сходів, міток та локацій, а також стислі дані про них та їхнє розташування відносно один одного. Кожен збережений об'єкт точки буде містити унікальний код, координати на плані поверху будівлі, номер поверху та код точки найближчого коридору або сходів, із яким цей об'єкт пов'язаний. Усі збережені об'єкти буде розподілено за кількома вказаними категоріями. Об'єкти точок приміщень та міток із QR-кодами будуть мати унікальну назву, яку може бути використано користувачем для визначення під час вибору місцезнаходження зі списку. Усі дані, які будуть використовувати у процесі позиціонування, для безперервної й безпомилкової роботи модуля мають бути доступними постійно. Для завантаження застосунку детальний план будівель та опис приміщень мають зберігати в певному вигляді на вебсервері та бути готовими для передавання через інтернет-з'єднання. Після завантаження всі дані будуть зберігати у внутрішній пам'яті електронного пристрою користувача задля використання у процесі позиціонування. Отже, для роботи модуля визначення місцезнаходження користувача необхідно застосовувати множину всіх елементів даних, наведених у словнику ІНС та описаних раніше в табл. 2.7.

Модуль визначення місцезнаходження користувача як складова архітектури ІНС університету взаємодіє із графічним компонентом системи (який надає вхідні дані для роботи модуля) та з модулем пошуку найкоротшого шляху (для якого модуль визначення місцезнаходження постачає вхідні дані). Отже, модуль, що розглядають, має досить складну та специфічну об'єктно-орієнтовану структуру, модель якої у вигляді діаграми класів показано на рис. 2.25.

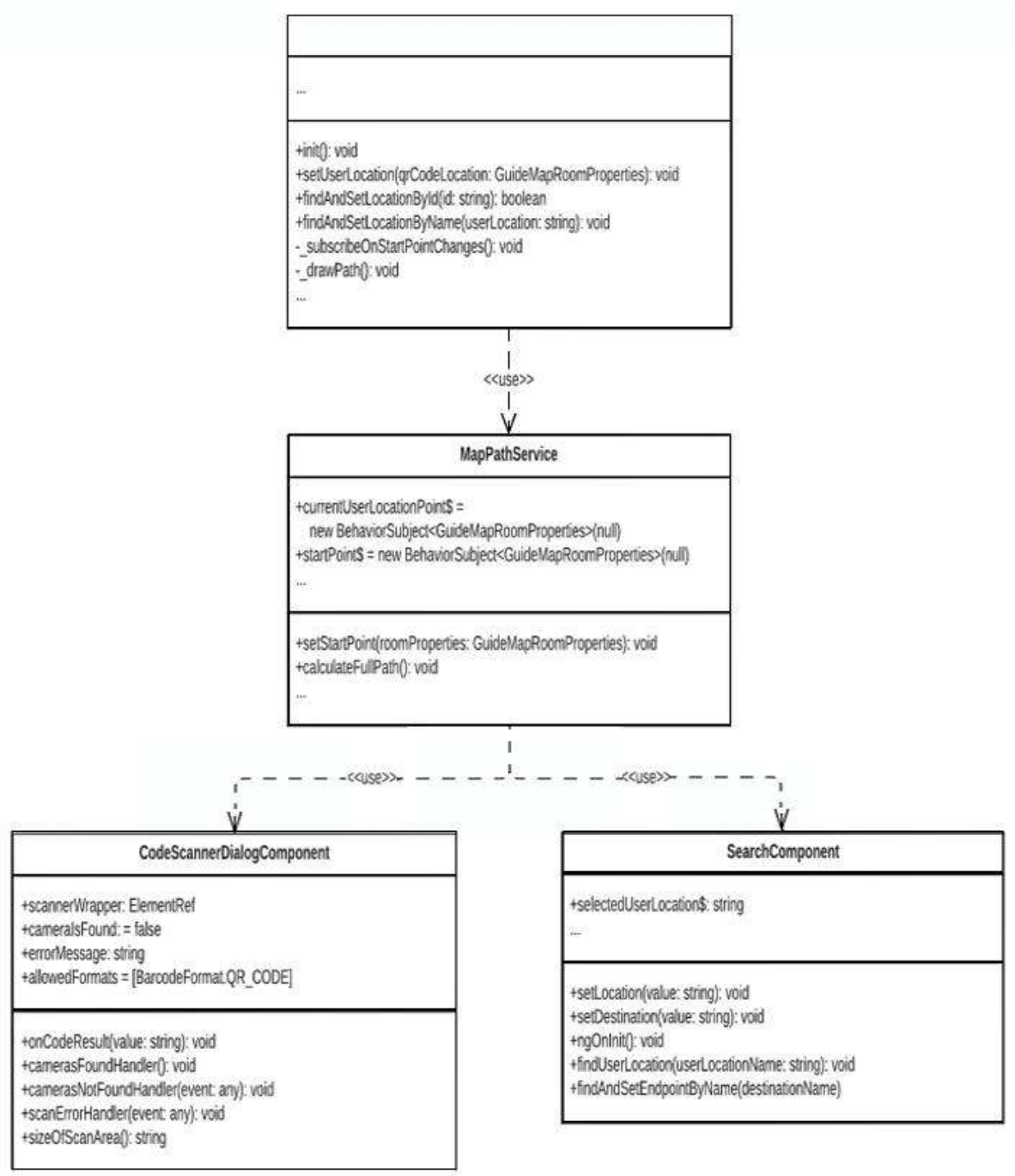


Рис. 2.25. **Діаграма класів модуля визначення місцезнаходження користувача**

Схарактеризуймо основне призначення та логіку роботи окремих класів модуля визначення місцезнаходження користувача. Клас MapService забезпечує введення місцезнаходження, пошук за назвою, а також пошук за кодом позиції користувача, відстеження змін місцезнаходження та його візуалізації на схемі певного поверху будівлі.

Для здобуття даних про місцезнаходження та використання їх для навігації слугує клас MapPathService, який задає початкове місцезнаходження як стартову точку маршруту та розраховує маршрут. Клас MapPathService використовує дані роботи класу SearchComponent, що відповідає за вибір та пошук місцезнаходження зі списку та CodeScannerDialogComponent, який виконує опрацювання даних із камери електронного пристрою користувача та розшифрування QR-кодів міток, відносно яких і визначають місцезнаходження користувача.

Для формального опису інтерфейсу користувача та відстеження взаємодії з ним на рис. 2.26 показано діаграму станів, що пов'язує й розкриває зміст подій і станів системи.

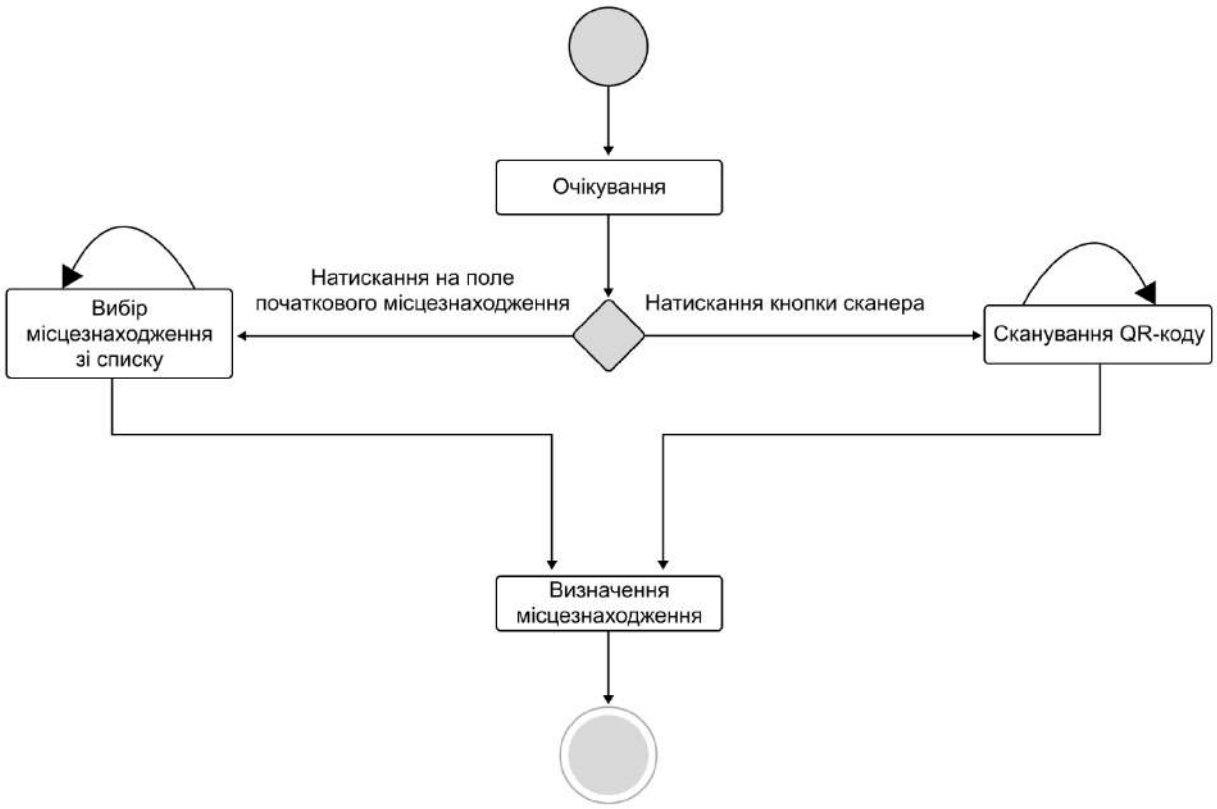


Рис. 2.26. **Діаграма станів модуля визначення місцезнаходження користувача**

Отже, означений модуль забезпечує необхідну посередницьку дію між графічним компонентом та модулем пошуку найкоротшого шляху. Результатом натискання на головному екрані поля під назвою "Початкове місцезнаходження" користувачем є відкриття випадного списку з доступними місцезнаходженнями (рис. 2.27).

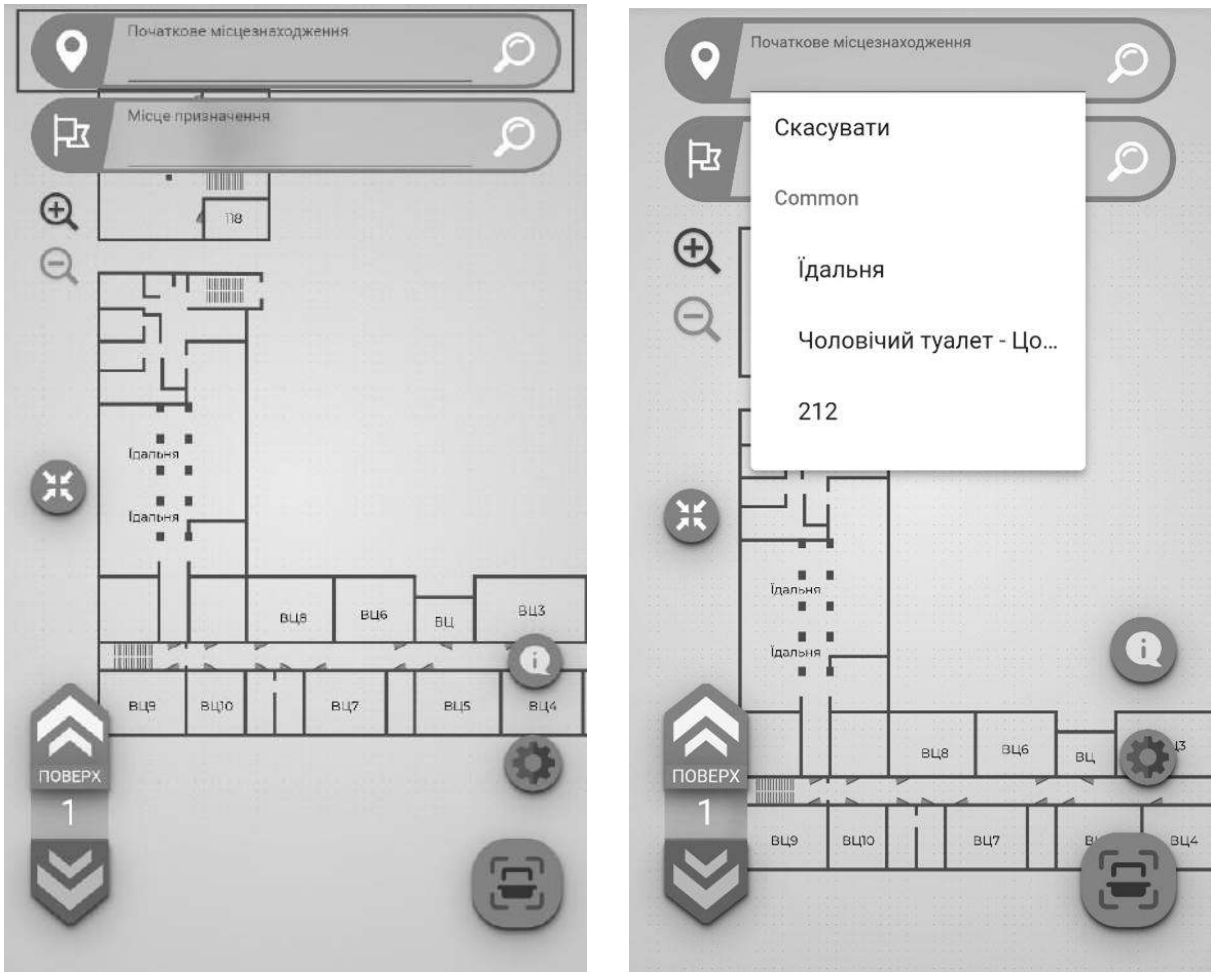


Рис. 2.27. **Вибір початкового місцезнаходження з випадного списку у відповідному полі на головному екрані**

Після вибору зі списку в полі початкового місцезнаходження виводиться назва вибраної локації. Натискання відповідної кнопки поруч із полем для відправлення запиту на позиціонування користувача приводить до візуалізації на схемі поверху будівлі вибраного місцезнаходження (рис. 2.28).

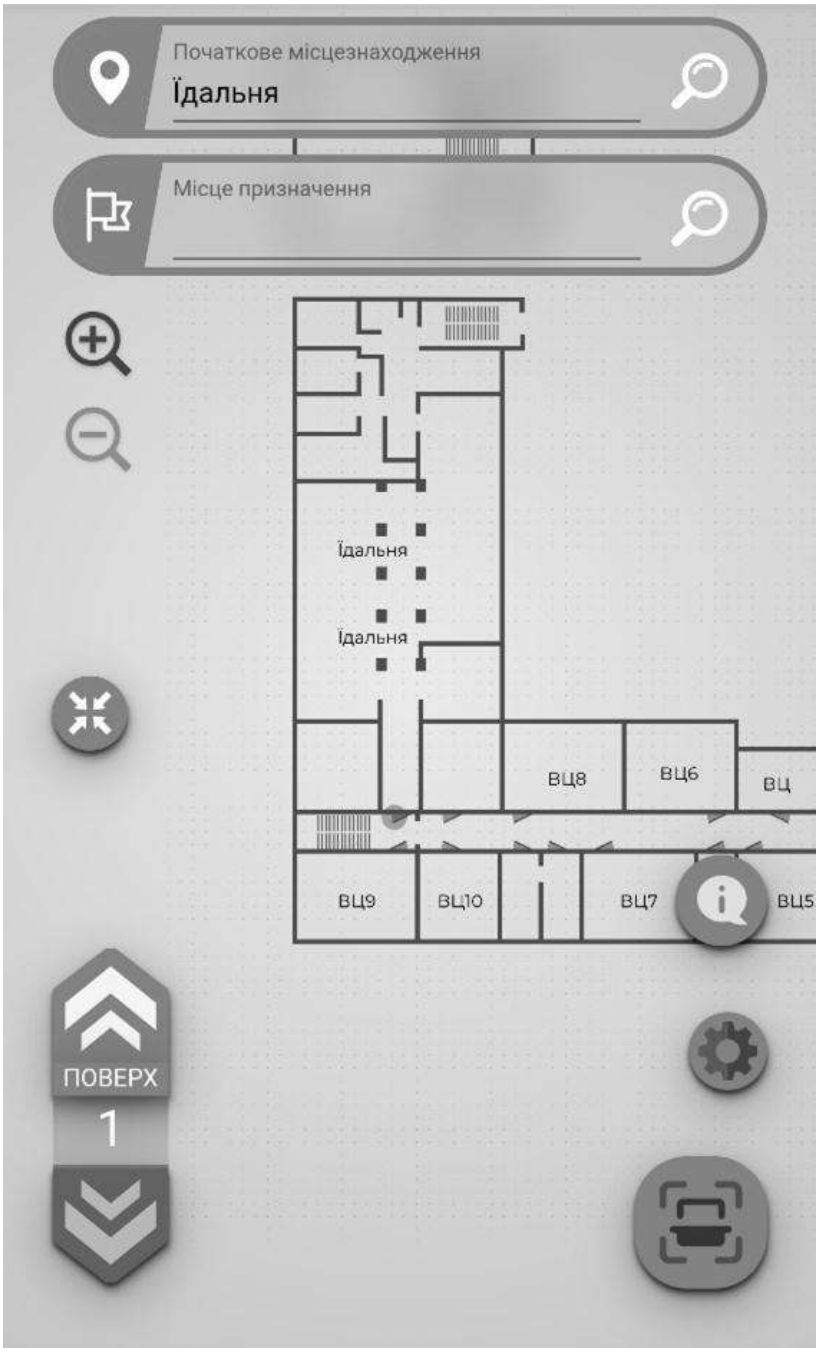


Рис. 2.28. **Позначене на мапі поверху вибране місцезнаходження**

Для визначення місцезнаходження користувача шляхом сканування QR-коду мітки, модулем реалізовано запуск сканера QR-кодів за допомогою камери мобільного пристрою користувача. Натискання на головному екрані кнопки запуску сканера, показаної на рис. 2.29, викликає модальне вікно зі сканером.

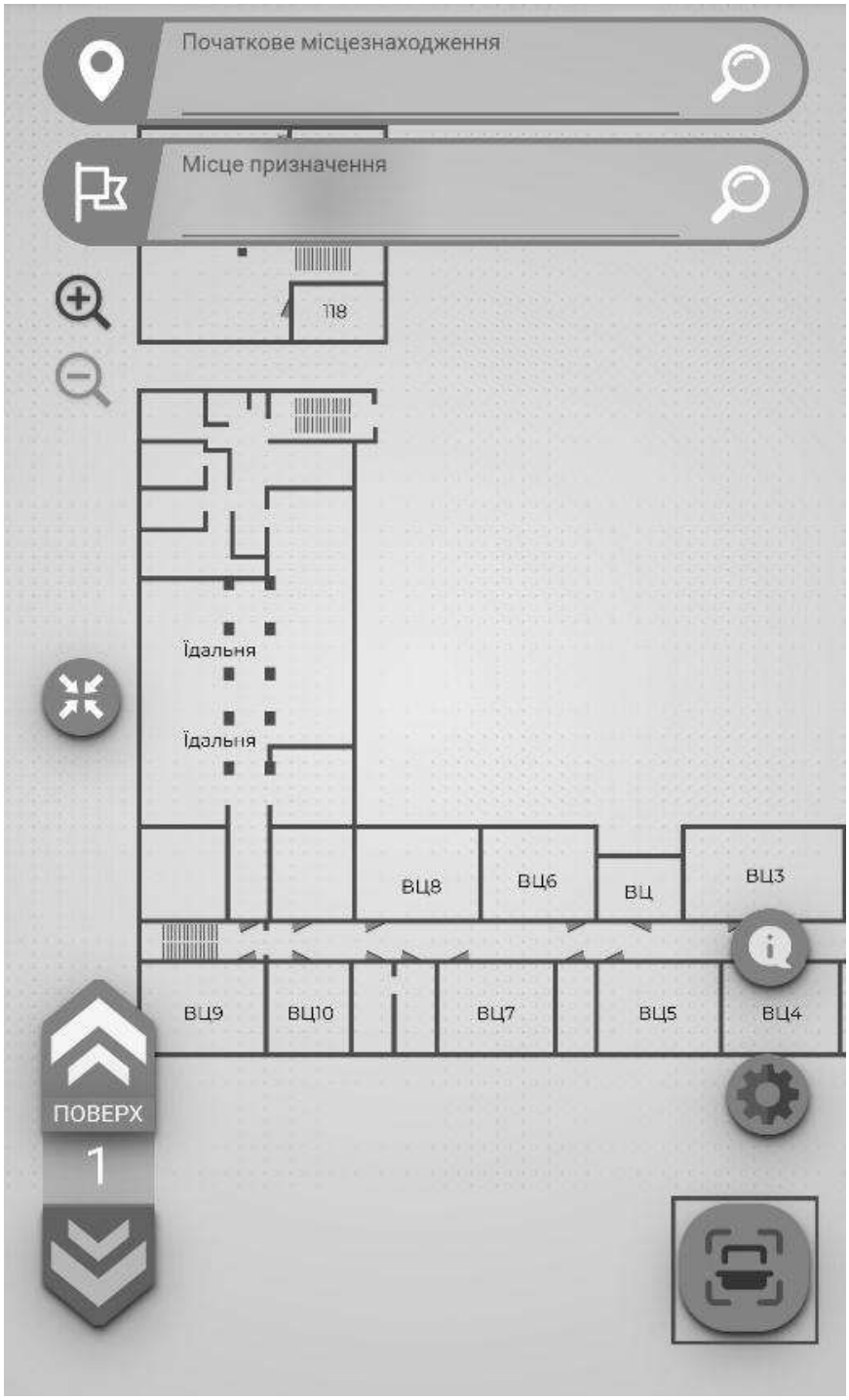


Рис. 2.29. **Кнопка запуску сканера на головному екрані ІНС**

Під час сканування QR-коду мітки шляхом наведення камери електронного пристрою модальне вікно закривається, визначене із QR-коду місцезнаходження візуалізують на схемі поверху будівлі та записують у поле "Початкове місцезнаходження".

Деякі приклади роботи та результати роботи сканера показано на рис. 2.30.

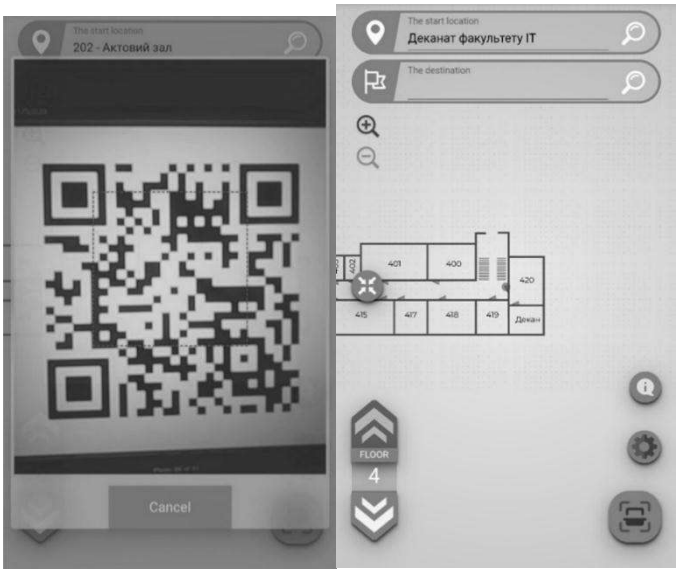


Рис. 2.30. **Деякі результати роботи сканера QR-коду міток та місцезнаходження користувача, визначене із QR-коду мітки**

Сканування випадкового QR-коду приводить до повідомлення про помилку та продовження подальшого сканування (рис. 2.31).

Рис. 2.31. **Попередження під час сканування випадкового QR-коду**

2.4. Результати та обговорення

Характеризуючи функціонал розробленої ІНС університету, спроєктованої на основі запропонованих інтерфейсних та алгоритмічних рішень,

слід зазначити низку її реалізованих можливостей. Розроблена ІНС університету, яка має як мобільну, так і вебверсії, забезпечує будь-якому користувачеві виконання таких основних функцій:

вільний перегляд схеми приміщення університету на інтерактивній мапі;

вільна зміна поточного поверху приміщення;

вільна зміна масштабу мапи;

прокладання найкоротшого маршруту від стартової точки, що відповідає фізично розташованому QR-коду до будь-якої доступної аудиторії або іншого приміщення;

візуальні підказки про місцезнаходження всіх доступних локацій;

зміна поверху натисканням на іконку сходів, якщо шлях пролягає через них;

сканування QR-кодів безпосередньо засобами ІНС;

відцентровування мапи;

локалізація інтерфейсу користувача на трьох мовах (українській, російській, англійській);

перегляд інструкції з використання ІНС та додаткової інформації.

Окрім цього, розроблена ІНС забезпечує низку додаткових функцій. Зокрема системою реалізовано можливість для користувачів ділитися посиланням на застосунок із уже побудованим маршрутом від найближчої точки із QR-кодом до точки потрібної локації. Це може бути корисним під час організації масових заходів, адже забезпечує швидке та надійне орієнтування одночасно значної кількості відвідувачів університету.

Також реалізовано можливість сканувати QR-коди стороннім програмним забезпеченням для переходу на ІНС. У QR-код, крім місцезнаходження самого QR-коду, також може бути закладено будь-яку кінцеву точку або цілий маршрут, що значно розширює сфери використання QR-кодів для користувачів ІНС.

Розроблену систему може бути встановлено на мобільний пристрій та за потреби використовувати офлайн. Шляхом застосування векторної графіки для генерації та опрацювання інтерактивних мап приміщень, а також, завдяки запропонованим технологічним підходам, розроблена ІНС має достатній функціонал для розширення й доопрацювання та є придатною для адаптації до застосування також в інших приміщеннях.

Отже, завдяки запропонованим алгоритмічним, інтерфейсним та технологічним рішенням, у процесі проєктування вдалося подолати основні

обмеження, притаманні аналогічним системам, що реалізують навігацію у приміщенні.

ІНС університету у її тестовій версії було впроваджено у практику освітньої діяльності ХНЕУ ім. С. Кузнеця. Зокрема, у відповідних локаціях університету було розміщено QR-коди, які дозволяють будь-якому відвідувачеві університету за допомогою смартфона здобути доступ до ІНС і скористатися описаним раніше функціоналом (рис. 2.32).

Рис. 2.32. **QR-коди, розміщені в певних локаціях ХНЕУ ім. С. Кузнеця, які дозволяють за допомогою смартфона здобути доступ до ІНС та її функціоналу**

У процесі апробації було зібрано та проаналізовано відгуки користувачів системи саме щодо її інтерфейсних та алгоритмічних можливостей. Користувачам було запропоновано заповнити форму, за допомогою якої їм необхідно було оцінити якість реалізації як функціональних, так і нефункціональних вимог до ІНС за п'ятибальною шкалою.

Загалом було отримано схвальні відгуки щодо якості інтерфейсу, швидкості завантаження інтерактивної мапи та зручності роботи з нею, зрозумілості візуалізації стартового місцезнаходження користувача й кінцевої точки, а також коректності запропонованого системою маршруту. Позитивне враження справила апробована користувачами можливість ділитися посиланням на застосунок із уже побудованим маршрутом.

Серед побажань щодо вдосконалення роботи ІНС було запропоновано підвищити якість анімованої візуалізації прокладеного маршруту між локаціями на різних поверхах, що, на думку деяких користувачів, є недостатньо чітко промальованою.

Отримані під час апробації відгуки від користувачів засвідчують доцільність розроблення та застосування системи для навігації в університеті. Побажання буде враховано у процесі розширення та удосконалення ІНС університету, що належить до перспектив нашої роботи.

2.5. Висновки

У роботі висвітлено основні алгоритмічні й інтерфейсні рішення, запропоновані для проєктування ІНС ХНЕУ ім. С. Кузнеця на основі застосування вебтехнологій.

У процесі пошуку алгоритмічних та інтерфейсних рішень було проаналізовано особливості науково-прикладного завдання навігації у приміщенні, а також оцінено можливості окремих аналогів ІНС університетів, що реалізують подібні функції цієї предметної області. Установлено, що проаналізовані аналоги, незважаючи на значний функціонал, мають такі обмеження:

1) є здатними забезпечити навігацію всередині конкретної будівлі, проте вони не є придатними до використання в інших приміщеннях;

2) не мають достатньо розвиненого функціоналу розширення та доопрацювання;

3) не забезпечують користувача мовною локалізацію;

4) є або вебсервісами, або мобільними застосунками.

Здійснено специфікацію функціональних та нефункціональних вимог до ІНС університету, визначено її архітектуру як сукупність взаємопов'язаних модулів, для реалізації яких запропоновано й розкрито відповідні інтерфейсні та алгоритмічні рішення.

Висвітлено основні етапи проєктування та розроблення ІНС університету в контексті запропонованих рішень.

Схарактеризовано функціонал реалізованої системи.

Установлено, що у процесі проєктування вдалося подолати основні обмеження, притаманні аналогічним системам, що реалізують навігацію у приміщенні.

Проаналізовано результати апробації системи в освітній практиці ХНЕУ ім. С. Кузнеця. Отримані під час апробації відгуки від користувачів засвідчують доцільність розроблення та застосування системи для навігації в університеті.

Сформульовано перспективи подальшої роботи.

Розділ 3
Захист інформації в інфокомунікаційній мережі за віддаленої роботи установи

3.1. Вступ і формулювання завдання

За період пандемії, коли багато організацій вимушені організовувати віддалену роботу, питання захисту інформації в інформаційно-телекомунікаційних мережах стало актуальним як ніколи. Масштабна кібератака 14 січня 2022 року, коли хакери атакували й частково зламали понад 70 державних вебсайтів України, зайвий раз довела слабкість кіберзахисту більшості установ і нехтування рекомендаціями CERT-UA – урядової команди реагування на комп'ютерні надзвичайні події України, яка функціонує у складі Державного центру кіберзахисту Державної служби спеціального зв'язку та захисту інформації України.

Причиною вдалої атаки вебсайтів імовірно стала вразливість системи управління вмістом сайтів October CMS, яку виявили ще у травні 2021 року. За сім місяців ніхто з відповідальних за кібербезпеку в установах, які зазнали атаки, не спромігся оновити програмне забезпечення.

Для складання рекомендацій для безпечної віддаленої роботи організації насамперед потрібно розглянути наявні загрози інфокомунікаційній мережі (кіберінциденти); потім дослідити сучасне шкідливе програмне забезпечення, яке застосовують зловмисники для кібератак і з'ясувати, які сучасні методи захисту використовують для його знешкодження.

Окремо слід дослідити питання захищеності вебсерверів: виявити найнебезпечніші кібератаки та вразливості, які вони використовують, запропонувати методи захисту, згідно з рекомендаціями CERT-UA [35].

3.2. Категорії кіберінцидентів

Закон України "Про основні засади забезпечення кібербезпеки України" визначає кіберінцидент (інцидент кібербезпеки) як подію або ряд несприятливих подій ненавмисного характеру (природного, технічного, технологічного, помилкового, зокрема внаслідок дії людського фактора) та/або таких, що мають ознаки можливої (потенційної) кібератаки, які становлять загрозу безпеці систем електронних комунікацій, систем управління технологічними процесами, створюють імовірність порушення

штатного режиму функціонування таких систем (зокрема зриву та/або блокування роботи системи, та/або несанкціонованого управління її ресурсами), ставлять під загрозу безпеку (захищеність) електронних інформаційних ресурсів.

Перелік категорій кіберінцидентів було розроблено CERT-UA [18], згідно з рекомендаціями Європейської агенції з кібербезпеки (ENISA Reference Incident Classification Taxonomy) і спільного документа ENISA та Європейського центру боротьби з кіберзлочинністю Європолу (Common Taxonomy for Law Enforcement and The National Network of CSIRTs). Усі відомі типи інцидентів було розподілено на 10 категорій. Кожній категорії й типу інциденту було надано двозначний код.

Категорія 01. "Шкідливий (образливий) вміст" (Abusive content), складається з одного типу інциденту – 01. Спам (Spam) – надсилання небажаних повідомлень або великої кількості повідомлень (флуд).

Категорія 02. "Шкідливий програмний код" (Malicious Code) містить такі типи інцидентів: 01. Зараження шкідливим програмним забезпеченням (Malware) infection – у системі виявлено шкідливе програмне забезпечення (ШПЗ); 02. Розповсюдження ШПЗ (Malware distribution) – розповсюдження ШПЗ, наприклад шляхом розсилання повідомлень електронної пошти, що містять укладення із ШПЗ або посилання на його завантаження; 03. Командно-контрольний центр (C2) (Command & Control (C2)) – система, яку використовують як точку управління ботнетом та/або слугує точкою для збирання інформації, викраденої ботнетами; 04. Шкідливе підключення (Malicious connection) – спроби з'єднання від/до IP/URL-адреси, пов'язаної з відомим ШПЗ, наприклад C2C або ресурсом розповсюдження компонентів, пов'язаних з активністю певної ботмережі.

Категорія 03. "Збирання інформації зловмисником" (Information Gathering) складається із трьох типів інцидентів: 01. Сканування (Scanning) – збирання інформації про системи або мережі; 02. Сніфінг (Sniffing) – несанкціоноване перехоплення (логічне або фізичне) та аналіз мережевого трафіка. Несанкціонований моніторинг та зчитування мережевого трафіка; 03. Фішинг (Phishing) – спроба збирання інформації про користувача чи систему за допомогою методів соціальної інженерії (масове розсилання електронною поштою, спрямоване на збирання даних, може містити посилання на фішингові сайти).

Категорія 04. "Спроби втручання" (Intrusion Attempts) містить два типи інцидентів: 01. Спроба експлуатації вразливості (Vulnerability exploitation

attempt) – спроба вторгнення з використанням вразливості в системі, компоненті чи мережі; 02. Спроби авторизації/входу в систему (Login attempts) – спроба входу до служб або механізмів автентифікації / доступу. Невдала спроба підбору автентифікаційних даних чи використання раніше скомпрометованих, уже неактуальних даних.

Категорія 05. "Утручання" (Intrusion) об'єднує два типи інцидентів: 01. Компрометація облікового запису (Account compromise) – фактичне вторгнення в систему, компонент або мережу шляхом компрометації облікового запису користувача або адміністратора; 02. Компрометація системи (System compromise) – фактичне вторгнення в систему чи її компонент, сервіс, застосунок через використання вразливості в компоненті або мережі. Несанкціонований доступ до системи або компонента в обхід системи контролю за доступом.

Категорія 06. "Порушення доступності" (Availability) складається із трьох типів інцидентів: 01. Атака на відмову в обслуговуванні (DoS/DDoS) – вплив на нормальне функціонування системи чи сервісу, що досягають спрямуванням з одного чи багатьох джерел до цільового ресурсу запитів для перенасичення пропускної спроможності чи системних ресурсів; 02. Саботаж / шкідливі дії (Sabotage) – дії (навмисні або ненавмисні), спрямовані на пошкодження системи, переривання процесів, зміну або видалення інформації тощо; 03. Перебій (Outage, no malice) – перебій у роботі системи чи її компонента без зловмисного втручання.

Категорія 07. "Порушення властивостей інформації" (Information Content Security) містить два типи інцидентів: 01. Несанкціонований доступ до інформації (Unauthorised access to information) – несанкціонований доступ до інформації й несанкціонований обмін конкретним набором інформації; 02. Несанкціонована модифікація (Unauthorised modification of information) – несанкціонована зміна або видалення певного набору інформації.

Категорія 08. "Шахрайство" (Fraud) складається з одного типу інцидентів – 01. Шахрайський сайт (Fraudulent site) – створення фішингових сайтів для збирання автентифікаційних чи інших даних користувачів. Використання ресурсів установи для цілей, відмінних від передбачуваних.

Категорія 09. "Відома вразливість" (Vulnerable) поєднує два типи інцидентів: 01. Уразливість (Vulnerability) – наявність у системі чи її компонентах відомих уразливостей, відкритих для експлуатації; 02. Некоректна конфігурація (Misconfiguration) – недоліки в налаштуваннях, що може бути використано зловмисником (налаштування за замовчуванням тощо).

Категорія 10. "Інше" (Other) складається з одного типу інцидентів – 01. Невизначений інцидент (Undetermined incident) – недостатньо даних для опрацювання інциденту.

Якщо код інциденту 02.04 – то це тип інциденту Malicious connection ("Шкідливе підключення").

Якщо інцидент зараховано до певної категорії, проте не визначено його тип, використовують код 00. Наприклад, код інциденту: 01.00; тип інциденту: не визначено; категорія: Abusive content.

3.3. Шкідливе програмне забезпечення

Шкідливе програмне забезпечення (ШПЗ) – це програмне забезпечення, яке за умови запуску може заподіяти шкоду різними способами, зокрема: призвести до блокування пристрою та його непридатності для використання; крадіжки, видалення або шифрування даних; використання пристроїв користувача для атак на інші організації; здобуття облікових даних, які дозволяють дістати доступ до систем або служб, якими користується суб'єкт атаки; майнінг криптовалюти; використання платних послуг на основі даних суб'єкта атаки (наприклад, телефонні дзвінки преміум-класу).

ШПЗ часто охоплює кілька категорій. Наприклад, програма може одночасно містити кейлогер, збирати паролі й бути "хробаком" для розсилання спаму. Далі наведено категорії, на які розподіляють більшість шкідливого програмного забезпечення:

Бекдор (backdoor) – шкідливий програмний код, який установлюють у систему, щоб надати зловмиснику віддалений доступ. Бекдори зазвичай дозволяють підключитися до комп'ютера з мінімальною автентифікацією або зовсім без такої та виконувати команди в локальній системі.

Завантажувач (downloader) – ШПЗ, єдиною метою якого є завантаження іншого шкідливого програмного коду. Зловмисники зазвичай установлюють завантажувачі за першого доступу до системи.

Викрадач інформації (stealer) – ШПЗ, яке збирає інформацію на комп'ютері жертви та, зазвичай, відправляє її зловмисникові. Як приклад можна навести програми, що збирають хеші паролів, перехоплювачі та кейлогери. Це ШПЗ зазвичай використовують для здобуття доступу до облікових записів інтернет-застосунків, як-от електронна пошта або інтернет-банкінг.

Руткіт (rootkit) – ШПЗ, що приховує наявність іншого коду. Руткіти зазвичай застосовують у поєднанні з іншим ШПЗ, як-от бекдор, що дозволяє

їм відкрити зловмисникові доступ до системи й ускладнити виявлення коду.

Залякувальне ПЗ (scareware) – створене для залякування атакованого користувача та спонукання його до купівлі чого-небудь. Зазвичай має графічний інтерфейс, схожий з антивірусом або іншою програмою, що забезпечує безпеку. Воно повідомляє користувачеві про наявність у його системі шкідливого коду й переконує його в тому, що єдиним виходом із ситуації є купівля певного програмного забезпечення.

Програма для розсилання спаму (spam-sending malware) – ШПЗ, яке заражає комп'ютер користувача й потім із його допомогою розсилає спам. Цей тип програм генерує дохід для зловмисників, дозволяючи їм продавати послуги з розсилання спаму.

Вірус-вимагач (ransomware) – тип шкідливого програмного забезпечення, що блокує доступ до системи або унеможливлює роботу з файлами (часто за допомогою методів шифрування), після чого вимагає від жертви викуп для відновлення вихідного стану.

Кейлоггер (keylogger) – програмне забезпечення, що реєструє кожну дію користувача, наприклад із пристроїв уведення (рух комп'ютерної мишки, натискання кнопок клавіатури). Дозволяє заволодіти даними користувача, що були введені після його встановлення.

Зазвичай зловмисники просять здійснити платіж (часто вимагають у криптовалюті), щоб розблокувати комп'ютер жертви. Однак у разі оплати немає гарантії, що жертва дістане доступ до своїх файлів. Беручи до уваги сказане раніше, важливо, щоб у режимі офлайн, завжди зберігали резервні копії найважливіших файлів і даних. CERT-UA рекомендує ігнорувати подібні вимоги, не переказувати кошти шахраям і повідомити про інцидент правоохоронні органи та CERT-UA.

3.4. Методи й засоби захисту від шкідливого програмного забезпечення

Основними методами захисту мережі від шкідливого програмного забезпечення є такі: резервне копіювання; запобігання розповсюдженню ШПЗ у мережі; запобігання запуску ШПЗ на пристроях; обмеження впливу ШПЗ; коректна робота під час ураження мережі ШПЗ.

Розгляньмо їх докладніше.

Резервне копіювання даних. Ключові дії, які потрібно вжити, щоб знизити рівень шкоди, яку може заподіяти шкідливе програмне забезпечення, – це забезпечити наявність актуальних резервних копій важливих файлів, за їхньої наявності можливо відновити свої дані, ігноруючи вимоги зловмисників. Резервне копіювання є ефективним заходом зниження ризиків від впливу ransomware. Слід регулярно здійснювати резервне копіювання даних, зберігати резервні копії на зовнішніх носіях інформації (SDD, HDD тощо) та налаштовувати функцію "відновлення системи". Потрібно періодично перевіряти можливість відновлення даних із резервних копій.

Хмарні сервіси (cloud service), що використовують синхронізацію (наприклад, Dropbox, OneDrive та SharePoint або Google Drive) не слід використовувати як єдине середовище для збереження резервних копій. Недоліком цих систем є те, що вони можуть автоматично синхронізуватися відразу після зараження файлів, і тоді можливо втратити й резервні копії також.

Здебільшого зашифровані файли не можуть бути розшифрованими ким-небудь. Не варто витрачати свій час чи гроші на послуги, які обіцяють це зробити. У деяких випадках фахівці з кібербезпеки можуть надати програми, які можуть розшифровувати файли через недоліки шкідливого програмного забезпечення. Рекомендуємо не використовувати програми для дешифрування даних із неперевірених джерел.

Можливо знизити ймовірність розповсюдження шкідливого програмного забезпечення в мережі за допомогою:

створення політик, що дозволить завантаження лише файли тих типів, які мають надходити (наприклад заборонити отримання чи передавання .EXE-файлів);

блокування вебсайтів, які є шкідливими;

перевірки антивірусними програмами файлів, що викликають підозру, у разі відсутності ліцензійного антивірусу рекомендуємо використовувати безкоштовний сервіс VirusTotal чи Cuckoo sandbox;

використання сигнатур для блокування відомого шкідливого коду.

Зазвичай названі раніше функції формуються системами на кшталт мережевих екранів, а не пристроями користувачів. Як приклад:

фільтрація пошти (у поєднанні з фільтруванням спаму), яка може блокувати шкідливі повідомлення електронної пошти та видаляти підозрілі вкладення;

використання засобів, які блокують відомі шкідливі вебсайти за відповідними списками;

використання засобів із функціями інформаційної безпеки, які можуть перевіряти вміст даних щодо відомих зловмисних програм;

під час використання віддаленого доступу дозволити підключення лише визначеним користувачам за допомогою білого списку (IP whitelisting).

Також слід ужити заходів для запобігання запуску ШПЗ. Необхідні кроки можуть бути різними для кожного типу пристроїв та операційних систем, але слід звернути увагу на такі методи захисту:

централізоване управління пристроями підприємства, щоб: дозволяти встановлювати лише те програмне забезпечення, яким довіряє організація (як приклад використання AppLocker); дозволяти запускати програми лише з надійних джерел чи ті, що мають відповідні сертифікати розробників;

використання антивірусного програмного забезпечення з технологією евристичного аналізу та вчасне оновлення його бази сигнатур;

унеможливлення підключення флеш-пристроїв та зовнішніх дисків, якщо не має повної довіри до їхнього джерела;

вимкнення або обмеження використання макросів (використовують у багатьох офісних продуктах, наприклад Microsoft Office, CorelDRAW, Notepad++).

Слід регулярно підтримувати безпечне налаштування та своєчасно встановлювати оновлення пристроїв, дотримуючись таких рекомендацій: установлювати оновлення безпеки, як тільки вони стануть доступними, щоб виправити недоліки, що використовують на ваших пристроях; увімкнути автоматичні оновлення для операційних систем, програм та мікропрограмного забезпечення, за можливості використовувати найновіші версії операційних систем та застосунків, щоб скористатися найновішими функціями безпеки.

Виконання таких заходів забезпечить швидке реагування та відновлення системи:

Використовувати двофакторну автентифікацію (2FA) для автентифікації користувачів всюди, де це можливо. Якщо облікові дані викрадено шкідливим програмним забезпеченням, це ускладнить можливість їхнього несанкціонованого використання.

За необхідності використання установою застарілих платформ (операційних систем і застосунків), рекомендуємо належним чином відокремити їх від основної частини мережі.

Не слід зберігати дані для автентифікації в легкодоступних місцях (наприклад, на робочому столі).

Потрібно використовувати для зберігання паролів менеджери паролів (наприклад KeePass, LastPass) і стійкі парольні фрази.

Потрібно регулярно переглядати та перевизначати права користувачів, щоб обмежити можливість розповсюдження ШПЗ. Шкідливі програми можна розповсюдити лише в ті місця мережі, до яких мають доступ облікові записи заражених користувачів.

Потрібно налаштовувати відповідні політики мережі, щоб використовували тільки необхідні порти, інтерфейси; використовувати програмні міжмережеві екрани (брандмауери) та штатні засоби захисту ОС від шкідливого програмного забезпечення; установлювати стабільні версії оновлень.

Якщо інфокомунікаційна мережа організації вже заражена шкідливим програмним забезпеченням, ці кроки можуть допомогти обмежити вплив вірусу: у певних випадках може бути необхідним негайне відключення заражених комп'ютерів, ноутбуків чи планшетів від усіх мережевих підключень, незалежно від проводового чи безпроводового, проте не вимикати сам пристрій; повідомити правоохоронні органи та CERT-UA.

Із метою збереження доказів несанкціонованого впливу, лише після завершення дій правоохоронних органів: слід змінити облікові дані, включно з паролями (особливо для адміністраторів); перш ніж відновити дані з резервної копії, потрібно переконатися, що копію створено до факту інфікування; за необхідності потрібно перевстановити операційні системи; слід оновити та виконати запуск антивірусного програмного забезпечення; здійснити перевстановлення операційної системи та застосунків, включно з базами даних програмного забезпечення та їхніми сигнатурами, має відбуватися в довіреному сегменті мережі; потрібно постійно відстежувати мережевий трафік щодо підозрілої мережевої активності.

3.5. Безпека вебресурсів

Є багато авторитетних ресурсів з рекомендаціями щодо захисту та коректного налаштування вебресурсів [28; 31; 32; 33 35; 39]. На авторів думку, найкраще ризики для вебресурсів та рекомендації щодо їхнього усунення описано у проєкті OWASP Top-10 [39], який роз'яснює

розробникам, проєктувальникам, архітекторам, менеджерам та організаціям наслідки, до яких призводять найвразливіші місця безпеки вебресурсів. У десятці найкращих подано основні методи захисту від цих проблем високого ступеня ризику, а також рекомендації щодо подальших дій із їхнього усунення.

Злам типу "Вставлення інструкцій", наприклад вставлення інструкцій SQL, ОС та LDAP, відбувається, коли ненадійні дані відправляють на інтерпретатор даних як частину команди, або запиту. Ворожі дані зловмисника можуть призвести до того, що інтерпретатор почне виконувати довільні команди, або зловмисник здобуде доступ до даних без належної авторизації.

Найкращий спосіб визначити, чи уразливий застосунок до вставлення інструкцій – це перевірити, чи всі інтерпретатори, що використовують, чітко відокремлюють сумнівні дані від команд або запитів. Для звернень SQL це означає використання присвоєних змінних у всіх підготовлених операторах та збережених процедурах, а також уникнення динамічних запитів.

Перевірка коду – це швидкий та надійний спосіб визначити, чи безпечно використовує застосунок "інтерпретатори". Інструменти аналізу кодів можуть допомогти аналітику безпеки визначити спосіб використання інтерпретаторів та відстежити опрацювання даних у застосунку. Спеціаліст із проникнення в системи може перевірити ці питання шляхом моделювання вторгнення, що підтвердить уразливість.

Автоматичне динамічне сканування застосунку може показати, чи є можливість вставлення інструкцій шляхом, придатним для використання. Сканери не завжди доходять до інтерпретаторів, і їм важко виявити, чи була атака успішною. Неналежне опрацювання помилок спрощує виявлення вставки інструкцій.

Процес запобігання вставленню інструкцій передбачає відокремлення сумнівних даних від команд та запитів. Найкращий варіант – використовувати безпечний ІПП (інтерфейс прикладного програмування), який узагалі не використовує інтерпретатор або забезпечує інтерфейс із заданими параметрами. Будьте обережними з такими ІПП, як збережені процедури, що є налаштованими, проте все ще можуть приховано виконати вставлення інструкцій. Якщо ІПП із заданими параметрами є не доступним, користувачу слід уникати певних символів шляхом використання спеціального синтаксису уникнення для інтерпретатора.

Позитивна перевірка даних, що вводять, або білий перелік, також рекомендують, проте не є повним захистом, оскільки багато застосунків потребують спеціальних символів під час уведення.

Некоректна автентифікація та управління сеансами – функції застосунку, пов'язані з автентифікацією та управлінням сеансами, часто є некоректно впровадженими, що дозволяє зловмисникам обходити паролі, ключі чи сеансові ідентифікатори або використовувати інші типи зламів для здобуття інших ідентифікаторів користувачів.

Ресурси управління сеансами, як-от облікові дані користувачів та ідентифікатори сеансу, можуть бути уразливими, якщо: облікові дані користувача для автентифікації не захищено під час збереження за допомогою хешування або шифрування; облікові дані можна підібрати або перезаписати в разі слабких функцій управління обліковими записами (наприклад, створення облікового запису, зміни пароля, відновлення пароля, слабких індивідуальних номерів (ІН) сеансів зв'язку); незахищені ІН сеанси зв'язку у URL (наприклад, переписування URL; ІН сеансів є уразливими для атаки; ІН сеансів є не обмеженими за часом або ідентифікатори користувачів чи автентифікації, особливо ідентифікатори "Технології єдиного входу до системи (SSO)", не перевіряють належним чином під час реєстрації; ІН сеансів не змінюють після успішного входу до системи; паролі, ІН сеансів та інші облікові дані відправляють незашифрованим з'єднанням.

Основна рекомендація для організацій для захисту від атаки "некоректна автентифікація й управління ресурсами" – це надати розробникам єдиний набір елементів сильного контролю за автентифікацією та управлінням сеансами. Такі елементи контролю мають відповідати всім вимогам до автентифікації й управління сеансами, визначеним у стандартах; мати простий інтерфейс для розробників. Задля емуляції, використання або як основи гарним прикладом є автентикатор та ІПП користувача ESAPI. Крім того, необхідно докладати всіх зусиль задля запобігання атакам XSS, що використовують для крадіжки індивідуальних номерів сесій.

Міжсайтове виконання сценаріїв (XSS) – атаки XSS відбуваються, коли застосунок здобуває ворожі дані та відправляє їх до веббраузера без належної перевірки. Атаки XSS дозволяють зловмисникам виконувати сценарії в браузері жертви, у результаті яких вони можуть перехоплювати сеанси користувача, видозмінювати вебсайти або переспрямовувати користувачів на інші шкідливі сайти.

Сервіс буде вразливим, якщо власник не забезпечить, щоб усі дані, що вводять користувачі, належним чином фільтрували, або якщо власник не перевіряє їх щодо безпеки за допомогою перевірки даних, що вводять, перед тим, як додати такі дані на сторінку, що випадає. Якщо Ajax використовують для динамічного оновлення сторінки, чи власники впевнені, що використовуєте безпечні ІПП JavaScript. Для небезпечних ІПП JavaScript також необхідно використовувати шифрування або перевірку.

Автоматизовані інструменти можуть виявляти деякі проблеми XSS автоматично. Однак кожний застосунок по-різному створює вихідні сторінки та використовує різні інтерпретатори браузера, JavaScript, ActiveX, Flash та Silverlight, що ускладнює автоматичне виявлення. Відповідно, повний захист потребує поєднання ручного перегляду коду та тестування на проникнення на застосунок до автоматичних підходів. Вебтехнології 2.0, Ajax, ускладнюють виявлення атак XSS за допомогою автоматизованих інструментів.

Запобігання атакам XSS потребує відокремлення сумнівних даних від умісту активного браузера. Найкращим варіантом є фільтрування всіх сумнівних даних, основаних на контексті HTML (тіло, атрибути, JavaScript, CSS або URL), у який будуть уносити дані. Позитивна перевірка даних, що вводять, або білий перелік, також рекомендують, проте вона не є повним захистом, оскільки багато застосунків потребують спеціальних символів під час уведення. Така перевірка має, наскільки це можливо, містити перевірку довжини даних, символів, формату та правил дій щодо таких даних перед тим, як приймати дані, що вводять.

Небезпечні прямі посилання на об'єкти – пряме посилання на об'єкт відбувається, коли розробник залишає незахищеним посилання на внутрішній об'єкт застосунка, як-от файл, каталог або ключ до бази даних. Без перевірки прав доступу або іншого захисту зловмисники можуть маніпулювати такими посиланнями, із метою несанкціонованого доступу до даних.

Найкращий шлях виявлення, чи є застосунок уразливим для небезпечних прямих посилань на об'єкти – це перевірити, чи всі посилання на об'єкти належним чином захищено. Для цього для прямих посилань на обмежені ресурси слід перевірити, чи застосунок виконав перевірку прав доступу користувача саме до ресурсу, щодо якого подано запит. Якщо посилання є непрямим, перевірити чи прив'язування до прямого посилання виконало обмеження значень, на які має право поточний користувач. Аналіз коду застосунка може швидко перевірити, чи безпечно

впроваджено кожен зі шляхів. Тестування також є ефективним для визначення прямих посилань на об'єкти та того, чи є вони безпечними. Автоматизовані інструменти, зазвичай, не визначають таких недоліків, оскільки вони не можуть розпізнати, що потребує захисту або є безпечним чи небезпечним.

Запобігання небезпечним прямим посиланням на об'єкти потребує вибору підходу до захисту кожного об'єкта, до якого має доступ користувач (наприклад, номер об'єкта, назва файлу). По-перше, використання непрямих посилань на об'єкти для користувача або сеансу. Це запобігає спробам зловмисника націлюватися безпосередньо на недозволені ресурси. Наприклад, замість використання ключа до бази даних ресурсу, у контекстному переліку шести ресурсів, дозволених для поточного користувача, використовуються числа від 1 до 6, щоб указати, яке значення вибрав користувач. Застосунок має перетворити непряме посилання для користувача назад у реальний ключ до бази даних на сервері. Перевірка доступу передбачає, що кожне використання прямого посилання на об'єкт із сумнівного джерела має містити перевірку контролю за доступом, щоб переконатися, що користувач має право доступу до запитаного об'єкта.

Небезпечна конфігурація оточення – належна безпека потребує визначення та використання безпечної конфігурації для застосунків, середовища розроблення, сервера застосунку, вебсервера, сервера бази даних та платформи. Необхідно визначати, упроваджувати та підтримувати безпечні налаштування, оскільки типові налаштування є, зазвичай, небезпечними. Крім того, ПЗ має бути оновленим.

Для того щоб перевірити, чи мають усі частини стека застосунку належну безпеку, насамперед слід перевірити таке: чи використовують будь-яке застаріле програмне забезпечення (таке програмне забезпечення містить ОС), вебсервер/сервер застосунку, СУБД, застосунки та всі бібліотеки коду; з'ясувати, чи є активованими або встановленими будь-які непотрібні елементи (наприклад, порти, сервіси, сторінки, облікові записи, привілеї); чи є активованими та незміненими стандартні облікові записи та паролі до них; чи ваша система опрацювання помилок відображає користувачам послідовність викликів функцій (трасу стека) або іншу надмірну інформацію в повідомленнях про помилки; чи налаштування безпеки у ваших інструментах розроблення застосунків (наприклад, Struts, Spring, ASP.NET) та бібліотеках установлено на безпечні значення.

Без забезпечення погодженого, постійного процесу безпеки конфігурації застосунку, системи перебувають під високим ризиком.

Основні рекомендації із захисту передбачають постійний процес посилення безпеки, що забезпечує швидке та просте розгортання іншого, належним чином заблокованого середовища. Середовище розроблення, контролю за якістю та експлуатацією мають бути однаково налаштованими (із різними паролями в кожному середовищі). Такий процес має бути автоматизованим для мінімізації зусиль, необхідних для підготовки нового, безпечного середовища.

Витік критичних даних – багато вебзастосунків неналежним чином захищають такі критичні дані, як дані платіжних карток, індивідуальні податкові номери й облікові дані для перевірки автентичності. Зловмисники можуть вкрасти або змінити такі слабо захищені дані та здійснити шахрайські операції із платіжними картками, украсти особисті дані або вчинити інші кримінальні правопорушення. Критичні дані слід додатково захищати шляхом шифрування під час збереження або передавання, а також необхідно дотримуватися певних застережень під час обміну такими даними із браузером.

Перше, що необхідно зробити – це визначити, які саме дані є критичними та потребують додаткового захисту. Наприклад, паролі, номери платіжних карток, медичні картки й особисті дані слід захищати. Для всіх таких даних слід з'ясувати таке: чи будь-які такі дані зберігають у вигляді незашифрованого тексту впродовж тривалого часу, включно з резервними копіями таких даних; чи будь-які такі дані передають у вигляді незашифрованого тексту, внутрішньо або зовнішньо (інтернет-трафік становить особливу загрозу); чи використовують будь-які старі/слабкі криптографічні алгоритми; чи генерують слабкі криптографічні ключі; чи є належним управління ключами; чи є ротація; чи є будь-які відсутні директиви безпеки або головні мітки браузера під час надання/відправлення чутливих даних до браузера.

Повний перелік ризиків, пов'язаних із небезпечною криптографією, використанням SSL та захистом даних не входить до обсягу десятки найбільших загроз вебсайтам. Тобто всі рекомендації, указані далі, є мінімально необхідними діями щодо критичних даних:

упевнитися, що шифруються всі критичні дані, що зберігають та передають, відповідно до визначених загроз;

не зберігати непотрібні критичні дані, видаляти їх так швидко, як це можливо, бо не можна вкрасти дані, яких немає;

забезпечити використання стійких стандартних алгоритмів і ключів, а також належне управління ключами, зважаючи на стандарт FIPS-140;

переконатися, що паролі зберігають за допомогою алгоритмів, призначених спеціально для захисту паролів, наприклад, bcrypt, PBKDF2 або scrypt;

відключити автоматичне заповнення форм, що збирають критичні дані та кешування для сторінок, що містять критичні дані.

Відсутність контролю за доступом до функціонального рівня: більшість вебзастосунків перевіряють права доступу до функціонального рівня перед тим, як відображати відповідну функцію в інтерфейсі користувача. Однак застосункам необхідно виконувати аналогічні перевірки контролю за доступом на сервері, коли здійснюють доступ до кожної функції. Якщо запити не перевіряють, зловмисники можуть підробляти їх для доступу до функцій без відповідної авторизації.

Найкращий шлях визначити, чи контролює застосунок доступ до функціонального рівня – це перевірити кожну функцію застосунку: чи відображає інтерфейс користувача посилання на недозволені функції; чи є перевірки автентифікації або авторизації з боку сервера; чи перевірки з боку сервера здійснюють виключно на основі інформації, що надає зловмисник.

За допомогою програми-посередника перегляньте ваш застосунок із привілейованою роллю. Потім повторно зайдіть на обмежені сторінки з менш привілейованою роллю. Якщо реакція сервера є однаковою, ви, напевно, є уразливими. Деякі програми-посередники для тестування безпосередньо підтримують такий тип аналізу.

Можна перевірити контроль за доступом у коді. Спробуйте відстежити один привілейований запит у коді та перевірити механізм авторизації. Потім визначте базу кодів, щоб виявити, де механізм не дотримується. Автоматизовані інструменти є не схильними до виявлення таких проблем.

Застосунок мусить мати постійний та легкий для аналізу механізм авторизації, що викликають з усіх функцій. Часто такий захист забезпечено одним або кількома компонентами, що є зовнішніми щодо коду застосунку. Слід проаналізувати процес управління дозволами та упевнитися, що ви можете легко оновлювати та контролювати його. Не перевантажуйте код. Механізм(и) виконання має відхиляти всі запити на стандартний доступ, вимагати чіткий дозвіл на доступ до кожної функції, відповідно до конкретної ролі. Якщо функція є залученою до послідовності дій,

що виконують, перевірте та впевніться, що всі умови для доступу зазначено належним чином.

Більшість вебзастосунків не відображають посилання та кнопки до недозволених функцій, однак такий "контроль за доступом на рівні відображення" не забезпечує реального захисту. Вам також необхідно впровадити перевірки в логічну частину контролера або програмний код, що реалізує функціональність застосунку.

Підроблення міжсайтових запитів (CSRF) – атака CSRF змушує підключений до системи браузер жертви автоматично відправляти підроблені запити HTTP, включно із фрагментом даних (кукізом) сеансу жертви та іншу інформацію щодо автентифікації до уразливого веб-застосунку. Це дає зловмисникам змогу змусити браузер жертви створювати запити, які уразливий застосунок уважає правомірними запитами жертви.

Для перевірки того, чи уразливий ваш застосунок, перевірте, чи всі посилання та форми мають непередбачувані ключі CSRF. Без таких ключів зловмисники можуть підробляти шкідливі запити. Альтернативний спосіб захисту – це вимагати від користувачів підтвердження наміру подати запит шляхом повторної автентифікації або будь-яким іншим шляхом підтвердження того, що вони є справжніми користувачами (наприклад, CAPTCHA). Слід звернути особливу увагу на посилання та форми, що викликають функції зміни стану, оскільки вони є найважливішими цілями CSRF. Потрібно перевіряти багатоетапні транзакції, оскільки вони не є захищеними як такі. Зловмисники можуть легко підробити ряд запитів за допомогою складних тегів. Зауважте, що фрагменти даних сеансів, IP-адреси джерел та інша інформація, що автоматично відправляють браузером, не забезпечують захист від CSRF, оскільки такі дані також додають до підроблених запитів.

Запобігання CSRF, зазвичай, потребує включення непередбачуваних ключів CSRF у кожний запит HTTP. Такі ключі мають бути, як мінімум, унікальними для кожного сеансу. Найкращий варіант – уставити унікальний ключ CSRF у приховане поле. Це приводить до того, що значення відправляється в тілі HTTP-запиту, запобігаючи його включенню до URL, який є більш схильним до розкриття. Унікальний ключ CSRF можна також додати в сам URL або параметр URL. Однак таке розміщення підвищує ризик розкриття URL зловмиснику, ставлячи під загрозу секретність ключа. Захист від CSRF можна також забезпечити шляхом повторної автентифікації або підтвердження особи користувача (наприклад, за допомогою CAPTCHA).

Використання компонентів із відомими вразливостями – такі компоненти, як бібліотеки, середовища розроблення та інші модулі програмного забезпечення майже завжди працюють із повними привілеями. Якщо використовують уразливий компонент, така атака може сприяти втраті критичних даних або підміні сервера. Застосунки, що використовують компоненти з відомими вразливостями, можуть знизити рівень захисту та сприяти різноманітним атакам і наслідкам.

Теоретично має бути легко виявити, чи використовуєте будь-які вразливі компоненти або бібліотеки. На жаль, звіти про вразливості в комерційному або відкритому програмному забезпеченні не завжди чітко зазначають, яка саме версія компонента є вразливою. Крім того, не всі бібліотеки використовують зрозумілу систему нумерації версій. Найгіршим є те, що не про всі вразливості повідомляють до центру обміну інформацією, у якому легко здійснити пошук; стає легше визначити такі сайти, як CVE та NVD. Для визначення того, чи є сервіси вразливими, необхідно шукати такі бази даних, а також стежити за переліком розсилань проєкту та оголошеннями про будь-що, що може бути вразливим. Якщо один із ваших компонентів має вразливість, вам слід ретельно оцінити, чи й справді ви є вразливими, шляхом перевірки, чи ваш код використовує частину компонента із вразливістю та чи може така вразливість вплинути на вас.

Один із варіантів захисту від використання компонентів із відомими вразливостями – це не використовувати компоненти, які ви не писали. Однак це майже нереально. Більшість проєктів зі створення компонентів не створюють вставки для вразливостей для старих версій. Замість цього, вони просто виправляють проблему в наступній версії. Отже, оновлення до таких нових версій є просто критичним.

Проєкти з розроблення програмного забезпечення мають забезпечити процес визначення всіх компонентів і версій, що ви використовуєте, включно зі всіма залежностями (наприклад, додаткових модулів для версій); контролю за безпекою таких компонентів у публічних базах даних, реєстрах розсилання проєкту та реєстрах розсилання безпеки й забезпечення їхнього оновлення; створення політики безпеки, яка буде регулювати використання компонентів, наприклад, потребувати певних методів розроблення програмного забезпечення, проходження тестів на безпеку та прийнятні ліцензії; де доречно, необхідно зважити на додавання безпечних обгорток, щоб відключити непотрібні функції та/або захистити слабкі або вразливі частини компонентів.

Небезпечні переадресування – вебзастосунки часто переспрямовують користувачів на інші сторінки та вебсайти, а також використовують сумнівні дані для визначення цільової сторінки. Без належної перевірки зловмисники можуть переспрямувати жертв до фальшивих чи шкідливих сайтів або використовувати переадресування для доступу до несанкціонованих сторінок.

Найкращий спосіб виявити, чи має ваш застосунок будь-які небезпечні переадресування, це:

аналізувати код для всіх переадресувань (називають передаванням у .NET). Для кожного разу використання визначте, чи включений цільовий URL у будь-які значення параметра. Якщо це так і якщо цільовий URL не перевіряється в білому переліку – ви є вразливими;

крім того, слід індексувати сайт, щоб виявити, чи він не створює переадресування (HTTP-коди відповіді 300 – 307, зазвичай, 302). Погляньте параметри, визначені до переадресування, та перевірте, чи є вони цільовим URL або частиною такого URL. Якщо так, то змініть цільовий URL і подивіться, чи сайт переадресує вас до іншої цілі;

якщо код є недоступним, перевірте всі параметри, щоб виявити, чи вони мають вигляд частини переадресування з URL-призначенням, та перевірте відповідність дій.

Крім того адміністратори вебресурсів мають забезпечувати:

управління оновленнями програмного забезпечення – необхідно постійно стежити за версіями операційної системи, системи управління контентом (CMS), менеджера пакетів, фреймворків або іншого ПЗ, що забезпечують роботу вебресурсу, та регулярно оновлювати їх. Водночас краще використовувати тільки LTS-версії;

використання HTTPS – використання протоколу HTTPS гарантує цілісність і конфіденційність взаємодії із сервером, захищає дані користувачів під час передавання в мережі "Інтернет". Сертифікат має бути виданим центром сертифікації. Необхідно використовувати TLS останньої версії (SSL має недоліки та вразливості й не є прийнятним для безпечного зв'язку). Гарною практикою є налаштування механізму HSTS (HTTP Strict Transport Security) для примусового використання HTTPS, навіть у разі переходу за посиланнями з явним зазначенням протоколу HTTP;

моніторинг журнальних файлів (логів) на підозрілі події – якщо у вебресурсу відсутня своя система журналювання (що описано в пункті A10 OWASP Top-10 2017), потрібно відстежувати журнальні файли

вебсервера (access.log). У журнальних файлах потрібно звертати увагу на POST-запити та код відповіді сервера на них. Особливу увагу варто приділяти POST-запитам до сторінок, які не мають приймати ніякі дані або яких взагалі не має бути. Це може свідчити про несанкціоновані дії з вебресурсом;

періодична перевірка директорій на сервері вебресурсу, із метою виявлення підозрілих файлів (пошук вебшелів), – зазвичай після зламу вебресурсів зловмисники залишають на сервері бекдори (вебшели) для віддаленого доступу до сервера сайта. Рекомендуємо періодично переглядати директорії вебзастосунку для пошуку таких бекдорів. Для цього можна використовувати спеціальні скрипти або банально перевіряти наявність нових файлів у директоріях. Виявлення створеного сторонніми особами файлу буде свідчити про злам вебресурсу та дає можливості для подальших дій із пошуку вразливостей, які було використано;

управління правами доступу – налаштуйте дозволи для файлів і каталогів. Розподіляйте права доступу до файлів на сервері й окремих розділів сайта, відповідно до завдань користувачів. Доцільно розмежувати розташування скриптів і програм, даних, призначених тільки для читання, та даних, призначених для зміни відвідувачами;

уникайте вразливих конфігурацій вебсервера – злам сайта починається зі збирання інформації про сервер. Приховування версій використовуваного ПЗ – це один з елементів убезпечення вебсервера. Знання версій цих програм може полегшити завдання зловмисника з пошуку відомих для цієї версії вразливостей і, як наслідок, досягнення основної мети – проникнення. Тому необхідно приховувати службові сторінки (наприклад phpinfo.php, temp.php, test.php.) і службову інформацію, що виводять у повідомленнях про помилки. Вимкніть непотрібні сервіси. Заблокуйте порти, що не використовують, налаштуйте міжмережевий екран та/або Web Application Firewall (WAF). Обмежте доступ до панелі адміністратора з мережі "Інтернет" і мереж загального користування. Регулярно змінюйте паролі доступу до сайта та сервера. Використовуйте захищені методи доступу до сервера для передавання файлів та управління ним (SFTP, SSH та ін.). Налаштуйте фільтрацію вхідних даних у вебформах. Регулярно здійснюйте резервне копіювання сайта та БД (якщо така наявна);

розмежування вебзастосунків – досить часто фахівці CERT-UA спостерігають розташування на одній віртуальній машині кількох вебресурсів, які не належать один до одного. Наприклад, вебсайт і стара версія

вебсайта або нова тестова версія. Стара версія не підтримується і має старі вразливості, тестова версія є недопрацьованою і також вразливою, водночас до них є доступ із мережі "Інтернет". Через уразливості цих вебресурсів зловмисники здобувають несанкціонований доступ до основного вебресурсу.

3.6. Рекомендації з безпечної організації доступу віддаленого до інформаційних ресурсів організації

Часто віддалена робота передбачає доступ працівників до інформаційних ресурсів організацій, розміщених у внутрішній мережі. Для цього використовують віртуальні приватні мережі VPN, програмне забезпечення для віддаленого доступу (AnyDesk, TeamViewer, RDP). Фахівці не рекомендують організовувати віддалену роботу за допомогою RDP без використання VPN із шифруванням.

Якщо в організації немає технічної можливості організувати віддалене підключення з використанням VPN із шифруванням необхідно дотримуватися таких правил:

Пароль до RDP має бути стійким.

Слід фільтрувати доступ до RDP. Визначити IP-адреси, із яких працівники організації працюють віддалено. Відфільтрувати доступ до комп'ютера із RDP за віддаленими IP-адресами працівників організації. Доступ з усіх інших IP-адрес слід заборонити. Це можливо реалізувати за допомогою брандмауера Windows.

Не рекомендовано надавати до директорій спільний доступ із мережі "Інтернет". Слід або фільтрувати доступ або взагалі заборонити. Працівники організації можуть здобути до неї доступ після підключення до RDP із внутрішньої мережі. Це можливо реалізувати за допомогою брандмауера Windows.

Потрібне постійне журналювання та моніторинг RDP-з'єднань. Працівники, відповідальні за інформаційну безпеку, мають періодично переглядати журнальні файли на наявність підозрілих записів (наприклад з'єднання працівників уночі).

Рекомендованим методом організації віддаленої роботи є використання віртуальних приватних мереж (VPN) із шифруванням. Є велика кількість комерційних і безкоштовних рішень. Одним із найпопулярніших є OpenVPN. Організація такого типу віддаленого доступу передбачає

наявність сервера, до якого приєднуються клієнти (працівники) за допомогою спеціально згенерованих сертифікатів, після чого їхній трафік переспрямовують до внутрішніх інформаційних систем.

Під час організації віддаленої роботи за допомогою VPN потрібно дотримуватися таких правил: налаштувати автентифікацію на VPN-сервері за допомогою сертифіката та пароля; VPN-сертифікати та паролі до них мають зберігати в захищеному середовищі. Якщо зловмисники здобудуть до них доступ, вони здобудуть доступ до мережі організації.

Здійснювати постійне журналювання та моніторинг з'єднань до VPN-сервера. Усі популярні VPN-сервіси мають функціонал журналювання подій. Працівники, відповідальні за інформаційну безпеку, мають періодично переглядати журнальні файли на наявність підозрілих записів (наприклад з'єднання працівників уночі).

Фільтрація доступу до VPN-серверу. Слід визначити IP-адреси, із яких працівники організації працюють віддалено. Потрібно відфільтрувати доступ до VPN-сервера за віддаленими IP-адресами працівників організації. Доступ з усіх інших IP-адрес заборонити.

Розмежування доступу працівників до внутрішніх ресурсів. Це можливо реалізувати за допомогою фаєрволів, мережевих екранів, віртуальних мереж.

Для розгортання VPN слід використовувати оновлене ліцензійне програмне забезпечення, завантажене з офіційних ресурсів.

Особливу увагу слід приділити безпеці домашніх персональних комп'ютерів працівників організації. Шкідливе програмне забезпечення часто розповсюджується з використанням фішингу та методів соціальної інженерії, що знижує пильність користувачів. Антивірусне програмне забезпечення може захистити персональний комп'ютер від вірусів та інших видів шкідливого програмного забезпечення, а також небажаного контенту в мережі "Інтернет". З огляду на це, працівникам варто дотримуватися кількох порад, щоб повною мірою скористатися перевагами антивірусного програмного забезпечення:

Слід увімкнути захист у режимі реального часу. Захист у режимі реального часу – це функція багатьох типів антивірусного програмного забезпечення, що відповідає своїй назві. Він автоматично сканує дані під час завантаження та в разі виявлення підозрілого вмісту блокує їх.

Потрібно сканувати зовнішні пристрої на наявність шкідливого програмного забезпечення. Багато користувачів використовують антивірусне програмне забезпечення лише для сканування фізичних дисків свого

персонального комп'ютера. Радимо сканувати всі флеш-носії на наявність шкідливого програмного забезпечення.

Як і все програмне забезпечення, антивірусне програмне забезпечення потребує своєчасних оновлень. Усі типи антивірусного програмного забезпечення працюють, використовуючи бази даних, які складаються з відомих загроз та їхніх відповідних індикаторів. У разі несвоєчасного оновлення ефективність виявлення актуальних загроз значно знижується.

Ще однією функцією, яку слід увімкнути, – це ведення журналу. Під час своєї роботи антивірусне програмне забезпечення автоматично буде реєструвати всі події – від звичайного сканування до виявлення шкідливого навантаження. Ця інформація може бути корисною в разі виявлення вектора зараження чи побудови схеми зараження. Журналювання може допомогти вам ідентифікувати шкідливе програмне забезпечення та його переміщення на вашому комп'ютері.

Деякі антивірусні програмні забезпечення більш ефективно захищають від загроз, ніж інші. Використовуйте антивірусне програмне забезпечення, що розповсюджується відомими вендорами у сфері кібербезпеки та яким довіряєте. Використовуйте ліцензійні рішення. Якщо не має змоги встановити антивірус на домашній комп'ютер, використовуйте вбудовані в ОС Windows функції захисту Windows Security та Microsoft Security Essentials.

Головна поширена проблема, яку допускають користувачі, – слабкий пароль адміністратора, який дозволяє здобути доступ до налаштувань домашнього WiFi. Саме використання встановленого за замовчуванням розробниками паролю може дати зловмисникам безпосередній контроль над WiFi-роутером. Приклад паролів за замовчуванням: 1111, root, user, admin тощо.

З огляду на попередній пункт, важливо розуміти, що WiFi без пароля робить вас уразливими. У такому разі необхідно встановити WPA2 шифрування та використовувати надійний пароль.

Так званий Broadcast SSID можна приховати в налаштуваннях роутера. Так ідентифікатор клієнтської мережі не буде видно стороннім особам. Це ускладнює можливий злам зловмисниками. Водночас під час підключення клієнту необхідно буде кожен раз уводити цей ідентифікатор.

Сучасні роутери можуть підтримувати різні протоколи, які використовують "розумні" пристрої. Так користувач стає потенційною жертвою зловмисників. Адже в цих пристроях можуть бути активними відкриті

вразливості. Якщо користувач не використовує цю можливість – відключіть UPnP.

Використовуйте оновлення версії вбудованого програмного забезпечення. Саме оновлення позбавляє користувачів від уразливостей, що стали відомими розробнику. Завдяки оновленням, розробник виправляє помилки, які дають можливість зловмиснику дістати дані із клієнтської мережі, здобути безпосередній доступ та управління.

Відключення функції WPS. Ця функція дозволяє без уведення пароля швидко підключитися до безпроводової мережі, тому її слід вимкнути.

3.7. Висновки

У дослідженні описано 20 типів кіберінцидентів, які, згідно з методиками CERT-UA [18], було об'єднано в 10 категорій (шкідливий уміст; шкідливий програмний код; збирання інформації зловмисником; спроби утручання; втручання; порушення доступності; порушення властивостей інформації; шахрайство; відома вразливість; інше), що допоможе користувачам виявити актуальні кіберзагрози.

Досліджено вісім видів найактуальнішого шкідливого програмного забезпечення й запропоновано методи захисту від кожного з них: резервне копіювання; запобігання розповсюдженню ШПЗ у мережі; запобігання запуску ШПЗ на пристроях; обмеження впливу ШПЗ; коректна робота під час ураження мережі ШПЗ. На підставі цього аналізу слід вибирати програмні методи захисту мережі.

Розглянуто 10 типів атак на вебресурси: уставлення інструкцій; некоректна автентифікація та управління сеансами; міжсайтове виконання сценаріїв (XSS); небезпечні прямі посилання на об'єкти; небезпечна конфігурація оточення; витік критичних даних; відсутність контролю за доступом до функціонального рівня; підроблення міжсайтових запитів (CSRF); використання компонентів із відомими вразливостями; небезпечні переадресування. Для кожного типу атак з'ясовано вразливості, що дозволяють їх здійснити, та запропоновано програмно-апаратні методи захисту.

Було розроблено рекомендації щодо захисту інформаційно-інфокомунікаційних мереж за віддаленої роботи установи: організації безпечного віддаленого доступу до інформаційних ресурсів компанії з робочого місця співробітника; ефективного використання антивірусного на домашніх персональних комп'ютерах; збереження чутливих даних/паролів; безпечного використання домашніх роутерів.

Розділ 4
Застосування штучного інтелекту в дослідженнях з управління проєктами

4.1. Вступ і формулювання завдання

Останніми десятиліттями проєкти мали тенденцію до підвищення труднощів до такої міри, що вони перетворилися на мегапроєкти, водночас супутнє промислове зростання призвело до вищого рівня невизначеностей під час здійснення таких проєктів із погляду контролю та розвитку. Застосування певних методологій управління проєктами (наприклад, PMI, IPMA і PRINCE) дозволяє управляти початком і розвитком проєкту найбільш оптимальним способом, контролюючи та реагуючи на будь-які проблеми, які виникають під час проєкту, сприяти їхньому завершенню та затвердженню до виникнення будь-яких подальших ризиків. Однак цих методологій, можливо, недостатньо, оскільки процеси мають бути чітко структурованими з повним і чітким контролем над проєктом в усіх відповідних сферах. Метою має бути поліпшення досвіду керівника проєкту під час роботи з різними несприятливими ситуаціями, які можуть виникнути під час розроблення проєкту, одночасно запобігаючи помилкам через відсутність планування чи управління, наприклад в управлінні портфелем [3]. Зараз методологію управління проєктами (project management methodology – PMP) широко впроваджують, що дозволяє краще управляти проєктом. Як зазначено раніше, процеси мають бути чітко структурованими та всі сфери проєкту мають бути жорстко контрольованими, зокрема з погляду інформаційних систем [16].

Сучасні методології є значною мірою недостатніми, оскільки менеджерові проєкту, зазвичай, залишається займатися ухваленням рішень. Менеджер, ґрунтуючись на своєму професійному досвіді, має ухвалювати "інтуїтивні" рішення на основі попередніх випадків, коли натрапляє на проблему з нескінченними змінними та можливостями. Тут практично неможливо зіткнутися з усіма проблемами та викликами, які тягнуть за собою сучасні проєкти. Насправді є низка різних причин, чому проєкти, зазвичай, зазнають невдачі. Ґрунтуючись на вивченні досвіду професіоналів, можна виділити такі:

- цілі, які не є чітко визначеними;
- відсутність протоколу зв'язку;

- відсутність визначення ролей і відповідальності;
- управління очікуваннями;
- сферу корупції;
- ігнорування ризиків проєкту;
- відсутність участі учасників;
- відсутність формального планування;
- оцінені помилки / нереальні;
- відсутність методик, шаблонів і документації;
- брак ресурсів;
- відсутність доказів або недостатня зосередженість на якості;
- недостатньо формалізований процес модифікації;
- відсутність підготовки;
- невелику адресну підтримку або її відсутність.

Із розвитком комерційної складової ІТ-галузі стало очевидно, що для якісного та своєчасного розроблення програмних продуктів потрібен штат фахівців із розробників і тестувальників, а також керівники ІТ-проєктів, які будуть вести комунікацію із замовниками й управляти командами розробників та тестувальників. Процес розроблення програмного забезпечення ускладнено обмеженнями у строках та необхідністю в дотриманні меж бюджету проєкту, а також великою кількістю основних та допоміжних процесів, які потребують планувати, управляти, робити звіти, розподіляти ресурси, мотивувати, стежити за строками та багато інших завдань, що в сучасних умовах може бути вирішено засобами штучного інтелекту. Використання таких технологій для управління ІТ-проєктами може привести до зміни багатьох правил і принципів в управлінні проєктами.

Штучний інтелект (ШІ) може розв'язувати багато проблем в управлінні проєктами, які можна автоматизувати, є приклади використання штучного інтелекту як ті, що самонавчаються, так і боти, які дотримуються чіткого алгоритму. За допомогою таких засобів стає можливим уникати проблем із ризик-менеджментом, розподілом ресурсів, створенням звітів для клієнта та за статусом виконання різних робіт. Водночас, як показує досвід, ШІ буде мати значний вплив на ефективність роботи проєктної команди та результати самого проєкту. Менеджери проєктів та організації, які на ранніх стадіях проєктної діяльності застосовують інструменти штучного інтелекту, безумовно, випереджають конкурентів та значно

підвищують свою ефективність щодо значного поліпшення результатів проєктної діяльності.

Отже, першочерговим завданням під час створення сучасних інтегрованих ІТ з інтелектуальною підтримкою для управління проєктами є оптимізація витрат на такі розроблення та інтеграцію, перетворення ІТ з обслуговчих на елемент генерації цінності.

4.2. Основна частина

Ефективне стратегічне планування проєкту має важливе значення для довгострокового успіху та невдачі для будь-якої організації, що працює в умовах цифрової економіки, незалежно від того, чи ця організація є бізнес-структурою, компанією з державною участю або органом влади. Водночас стратегічне планування проєкту, що забезпечує стійка конкурентна перевага організації, має враховувати організаційні можливості, як-от взаємодія між членами проєктної команди, керівництвом та співробітниками з різних бізнес-підрозділів, а також ролі різних зацікавлених сторін, корпоративну структуру та культуру. Водночас перспективи розвитку кар'єри для всіх керівників проєктів в організації відіграють роль у високій ефективності компанії в галузі стратегічного управління проєктами.

Стратегічне управління проєктами – це комплексний підхід до досягнення стійкої конкурентної переваги, що досягають шляхом погодження стратегічних бізнес-цілей зі стратегією управління проєктами, що "забезпечує поєднання інституційної, соціальної, економічної, екологічної та фінансової стійкості компанії" [3].

Отже, стратегічне управління проєктами є життєво важливим для організації способом поєднання стратегічного й оперативного управління, оскільки воно визначає процеси, за допомогою яких організація адаптується до середовища, що постійно змінюється. Реалізація стратегії слідує за ухваленням організаційних політик та практик, які відповідають корпоративній стратегії. Крім того, розроблення стратегії, яка має бути дієвою, містить таке:

визначення організаційних структур, процесів та відносин для досягнення видатних результатів у бізнесі;

досягнення успіху, завдяки перевазі в таких ресурсних галузях організації, як людські ресурси, інформаційні технології, фінанси тощо.

Під *штучним інтелектом (ШІ)* розуміють сукупність когнітивних технологій на основі комп'ютерного зору, машинного навчання, опрацювання природної мови та робототехніки. ШІ управління проєктами можна розглядати як убудовану систему, здатну управляти проєктом без допомоги людини. Використання можливостей ШІ не тільки тягне за собою автоматизацію розв'язання рутинних управлінських завдань, а передбачає розроблення висновків на основі різних уявлень, вироблення рекомендацій щодо процесу, ухвалення рішень, пов'язаних із проєктом, та розкриття ідей команди.

Управління IT-проєктами – це процес планування, організації та розмежування відповідальності за досягнення конкретних цілей організації в галузі інформаційних технологій (IT). Процес управління містить нагляд за проєктами з розроблення програмного забезпечення, установлення апаратного забезпечення, оновлення мережі, розгортання хмарних обчислень і віртуалізації, бізнес-аналітики та проєктів управління даними, а також упровадження IT-послуг. На додаток до звичайних проблем, які можуть призвести до невдачі проєкту, фактори, які можуть негативно вплинути на успіх IT-проєкту, мають прогрес у технології під час виконання проєкту, зміни інфраструктури, які впливають на безпеку й управління даними, а також невідомі залежні відносини між апаратним і програмним забезпеченням, мережеву інфраструктуру та дані [41].

Управління проєктом можна розподілити на п'ять груп процесів, які становлять життєвий цикл проєкту та є універсальними для всіх проєктів. Однак конкретні фази проєкту є унікальними для кожного проєкту та презентують життєвий цикл проєкту.

Початок. Визначено мету проєкту, потребу чи проблему. За проєктом призначають менеджера проєкту та укладають статут проєкту.

Планування. Керівник та команда проєкту працюють разом, щоб спланувати всі необхідні кроки для досягнення успішного завершення проєкту. Процеси планування проєкту мають ітераційний характер, та очікують, що планування буде відбуватися часто протягом усього проєкту.

Виконання. Після створення плану проєкту, команда розпочинає його виконання, щоб досягти результатів проєкту. Команда може перейти до планування проєкту в міру необхідності протягом усього його виконання.

Моніторинг і контроль. Оскільки проєкт виконує команда проєкту, керівник проєкту контролює роботу щодо часу, вартості, обсягу, якості, ризику та інших факторів проєкту.

Закриття. Наприкінці кожної фази та наприкінці всього проєкту відбувається закриття проєкту, щоб переконатися, що всю роботу було завершено, схвалено, та в кінцевому підсумку передають право власності від команди проєкту до операцій.

Не менш важливим, ніж фази проєкту є галузі управління знаннями проєкту. Є 10 галузей знань з управління проєктами, що сегментують різні дії, які виконує керівник проєкту протягом усього проєкту.

Управління обсягом проєкту: обсяг проєкту визначено, задокументовано та затверджено. Обсяг проєкту є захищеним від несанкціонованих змін, відредагованим із затвердженими змінами та підтвердженим зацікавленими сторонами проєкту для його ухвалення.

Управління розкладом проєкту: графік проєкту спочатку визначають його робочими годинами, будь-якими етапами та, зрештою, кінцевим терміном виконання. Доступність команди протягом усього проєкту документують та планують відповідним чином. Менеджер проєкту буде працювати з його командою, щоб визначити завдання проєкту та оцінити його тривалість, щоб створити графік проєкту.

Управління витратами на проєкт: витрати на проєкт оцінюють так, щоб можна було призначити його бюджет. Витрати на проєкт містять матеріали, послуги, устаткування, ліцензії на програмне забезпечення та інші витрати, пов'язані безпосередньо із цим проєктом.

Управління якістю проєкту: те, що є якістю у проєкті, визначають у конкретних показниках і погоджують між зацікавленими сторонами якомога раніше в межах проєкту. Програми та політика забезпечення якості спрямовують роботу над проєктом, тоді як контроль за якістю перевіряє проєктну роботу, щоб підтвердити, що якості було досягнуто в роботі.

Управління людськими ресурсами проєкту: менеджер проєкту співпрацює з його командою, щоб переконатися, що кожен член команди виконує свої завдання, добре працює з іншими та що їхня участь і результати роботи повідомляють їхнім відповідним менеджерам.

Управління комунікаціями проєкту: зацікавленим сторонам знадобиться інформація від керівника проєкту, необхідно буде надавати йому

інформацію протягом усього життєвого циклу проєкту. Ця галузь знань створює план управління комунікаціями, який визначає кому яка інформація буде потрібною, коли вона є потрібною, і найкращий спосіб комунікації.

Управління ризиками проєкту: ризики – це ситуації, події, умови, які можуть загрожувати, а іноді й давати користь цілям ІТ-проєкту. Ризики мають бути ідентифікованими, проаналізованими та дати відповіді на подію ризику. Імовірність та вплив кожної ризикової події оцінюють для аналізу ризику, щоб виправдати витрати, необхідні для управління ризиковою подією.

Управління закупівлями проєкту: якщо для проєкту потрібно буде придбати товари чи послуги, необхідно здійснити офіційний процес закупівлі. План має стосуватися вибору для проєкту типу контракту, адміністрування контракту, аудиту закупівель та закриття контракту. Багато керівників проєктів не управляють закупівлями, а довіряють централізованому відділу закупівель або процесу закупівель організації.

Управління зацікавленими сторонами проєкту: зацікавлені сторони – це будь-яка особа, яка має власний інтерес до проєкту. Управління зацікавленими сторонами – це ідентифікація, включення та спілкування із групами зацікавлених сторін проєкту. Воно усуває тривоги та занепокоєння зацікавлених сторін щодо роботи над проєктом.

Управління проєктною інтеграцією: ця спеціальна галузь знань є координацією подій у всіх інших галузях знань. Наскільки добре керівник проєкту працює в одній галузі знань, безпосередньо впливає на ефективність інших галузях знань. Управління інтеграцією проєкту вивчає взаємодію та непередбачені ситуації між галузями знань, щоб гарантувати, що його належним чином сплановано, виконано, контрольовано та закрито.

Цими 10 галузями знань необхідно управляти ітераційно протягом усього проєкту.

Кожен проєкт має свою унікальність, та життєвий цикл кожного проєкту відрізняється один від одного. Є кілька різних підходів до управління ІТ-проєктом, які впливають на його життєвий цикл. Організації можуть вибрати один із цих популярних підходів, щоб знизити ризик дорогого перероблення, ризики від швидко змінюваних технологій або масштабного планування під час запуску проєкту. Життєвий цикл типового ІТ-проєкту

проходить через ітерації планування, виконання та контролю, поки його остаточно не буде закрито й переведено в експлуатацію. Життєві цикли управління ІТ-проєктами такі:

Прогнозний життєвий цикл: це найбільш поширений і традиційний життєвий цикл проєкту для ІТ-проєктів. У цьому підході керівник та команда проєкту спочатку визначають його обсяг, графік та очікувані витрати до початку його виконання. У межах планування проєкту типовим є визначення його фаз (кожна фаза виконує певний тип проєктної роботи). Для того щоб проєкт рухався від початку до закриття, кожну його фазу має бути розпочато та завершено в певному порядку, як планували. Цей тип підходу іноді називають підходом водоспаду, оскільки етапи йдуть один за одним і ніяк інакше.

Ітеративний життєвий цикл: цей підхід до управління ІТ-проєктами потребує, щоб управління цим проєктом було визначено на його початку, але оцінювання витрат і тривалість діяльності планують на більш високому рівні на початку проєкту. У міру виконання проєкту формують оцінки витрат і тривалості для найближчої роботи за допомогою ітерацій планування. Ітеративний життєвий цикл також планує ітерації переваг, наданих організації. Наприклад, ітеративний життєвий цикл може створювати нове програмне забезпечення з більшими можливостями з кожним новим випуском у межах проєкту.

Адаптивний життєвий цикл: цей життєвий цикл проєкту також використовує ітерацію планування та виконання, але планування зазвичай триває два тижні. Цей підхід використовує хвилю планування та виконання через короткі секції планування та виконання. Очікують зміни в цьому підході до ІТ-проєкту, і він ідеально підходить для проєкту розроблення програмного забезпечення. Agile-менеджмент проєктів і Scrum є прикладами адаптивного життєвого циклу.

Усі ці життєві цикли використовують концепцію фаз для просування роботи над проєктом. Фаза описує тип роботи, яку будуть виконувати в цій частині проєкту. Керівник проєкту, організаційні вимоги та навіть вимоги замовника можуть впливати на тип життєвого циклу цього проєкту, який менеджер проєкту адаптує в ньому [18].

Упродовж дослідження фаз, галузей та життєвого циклу проєктів стало зрозумілим, що головною фігурою у проєкті є її менеджер або керівник проєкту. Менеджер ІТ-проєктів відповідає за нагляд за ІТ-відділом

організації й управління командами для виконання IT-проєктів вчасно та в межах бюджету. Деякі з обов'язків менеджера IT-проєктів містять:

формулювання цілей проєкту та створення планів їхнього досягнення;

ведення графіка та бюджету проєкту, складання звітів про стан;

управління ресурсами, включно з командою, устаткуванням тощо;

призначення завдань учасникам команди;

розроблення стратегії для виконання проєктів вчасно та в межах бюджету;

використання інструментів управління IT-проєктами для відстеження прогресу та ефективності;

оцінювання ризику та адекватне реагування;

проведення регулярних зустрічей із командою та зацікавленими сторонами.

Зазвичай у різних компаніях деякі деталі в розкладі робочого дня керівників проєктів можуть відрізнятися, але основні функції мають такий вигляд. Очікують, що керівники IT-проєктів володіють розширеними знаннями комп'ютерів, операційних систем, адміністрування мережі та служби підтримки. Вони також мають бути гарними комунікаторами та вміти чітко пояснювати складні технічні питання. Інші необхідні навички містять досвід планування, бюджетування та планування ресурсів.

Ця роль також потребує стратегічного ділового мислення, уміння будувати команду та розв'язувати конфлікти, а також досвід управління змінами, серед інших ключових навичок, які користуються великим попитом. На базовому рівні керівники проєктів мають виявляти лідерство, уміти мотивувати членів команди, розставляти пріоритети та розв'язувати проблеми. Адаптивність – це ще одна ключова нетехнічна навичка, яку має набути менеджер проєктів, щоб досягти успіху. Для того щоб бути високоефективним менеджером проєктів, людина має бути стратегічним бізнес-партнером, повністю покладеним на успіх організації, а також має бути в змозі подолати неминучі невдачі. У поєднанні з необхідними технічними навичками певні атрибути підвищать попит на керівника проєкту, забезпечуючи міцну основу, яка дозволить адаптуватися до постійно мінливої динаміки проєкту, ставлячи потреби зацікавлених сторін на перше місце [25].

Американський інститут управління проєктами (PMI) провів глобальне дослідження Pules of the proffesions 2018 [40]. В опитуванні взяли участь

понад 4 000 професіоналів з управління проєктами з усього світу. Одним із запитань було таке: "Із тих проєктів, які розпочалися у вашій організації за останні 12 місяців, які вважали невдалими, які основні причини цих провалів?" Результати цього опитування можна побачити на рис. 4.1.

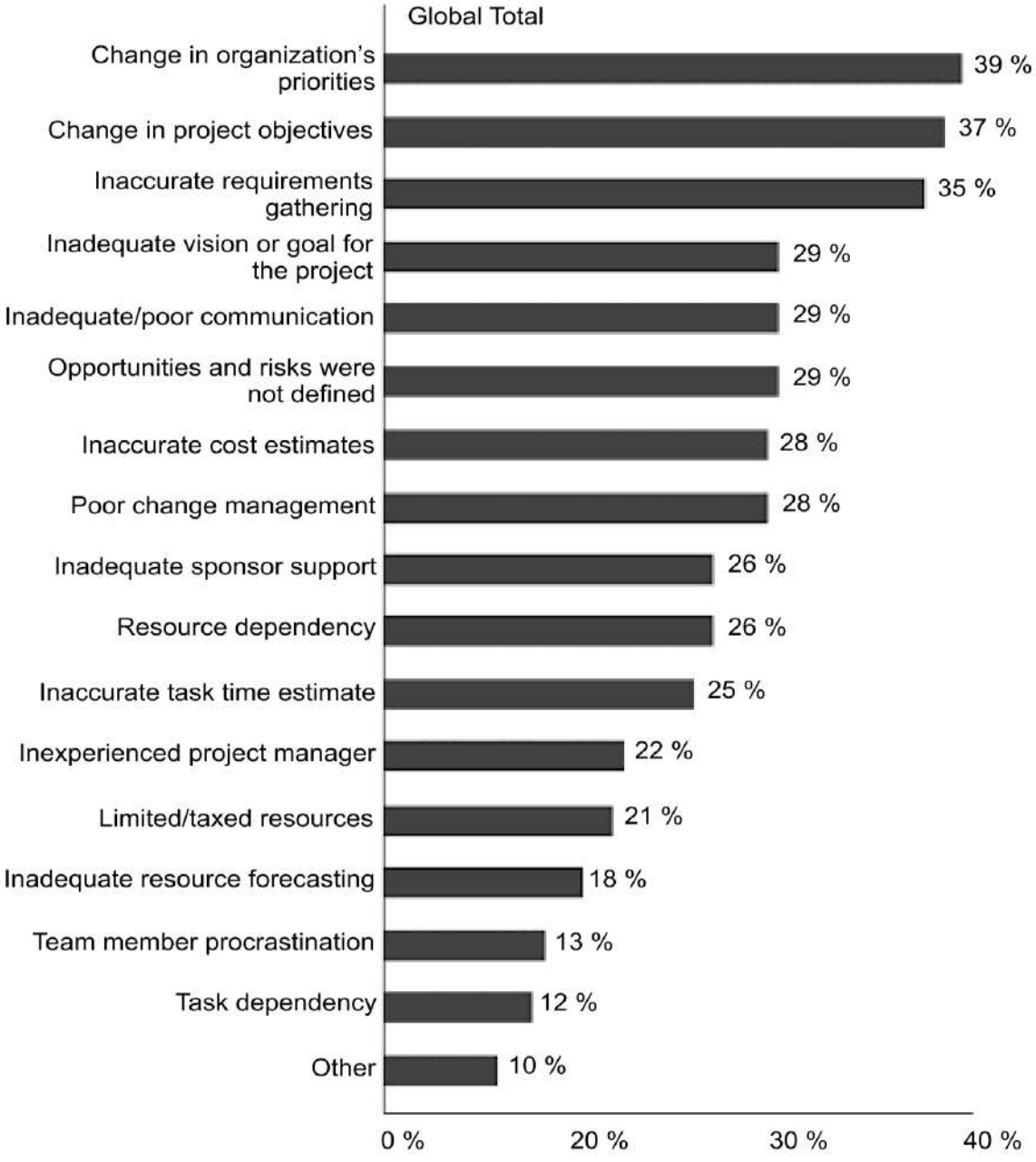


Рис. 4.1. **Основні причини провалів IT-проєктів**

Найбільший вплив мають такі фактори: зміна пріоритетів організації – 39 %; зміна цілей проєкту – 37 %; неправильне збирання вимог – 35 %; середні показники впливу: недостатнє бачення або мета проєкту – 29 %; недостатня/незадовільна комунікація – 29 %; невизначені можливості та ризики – 29 %; неточні оцінки – 28 %; незадовільне управління змінами – 28 %; недостатня підтримка спонсора – 26 %; залежність від ресурсів – 26 %; неточна оцінка часу виконання завдання – 25 %; недосвідчений керівник проєкту – 22 %; обмежені ресурси – 21 %; найменше постраждали: неадекватне прогнозування ресурсів – 18 %; неефективна робота в команді – 13 %; залежність від завдань – 12 % та інші – 10 % [40].

За результатами опитувань, проведених Harvard Business Review [41], 54 % робочого часу керівника проєкту відведено на адміністративні завдання. Відповідно, залишається менше часу на координацію завдань управління проєктом. Більше того, більшість адміністративних завдань можна оптимально виконувати за допомогою сучасних технологій штучного інтелекту. Такий підхід допоможе керівникам проєктів зосередитися на більш ефективних процесах створення цінності проєкту та забезпечити ухвалення обґрунтованих управлінських рішень щодо проєктів та розвитку організації загалом.

Для автоматизації рутинних завдань та інтеграції штучного інтелекту в управління проєктами можна використовувати трикутну модель на рис. 4.2, яка за допомогою інтеграції та розширення можливостей основних методологій управління проєктами в поєднанні з сучасним управлінням ІТ-проєктами дозволяє передавати рішення значущих даних потоків, розв'язання рутинних проблем проєктних конфліктів та обмежень і тим самим розвантажити керівника проєкту для вирішення стратегічних питань, фіксують позитивні результати проєктної діяльності. Як видно з рис. 4.2 особливу увагу в ході реалізації моделі слід приділити інтеграції інструментів управління проєктами із сучасними технологіями штучного інтелекту. Перші забезпечено інформаційними компетентностями керівника проєкту та команди проєкту, а другі – додатковими інтелектуальними (допоміжними, творчими) компетентностями.

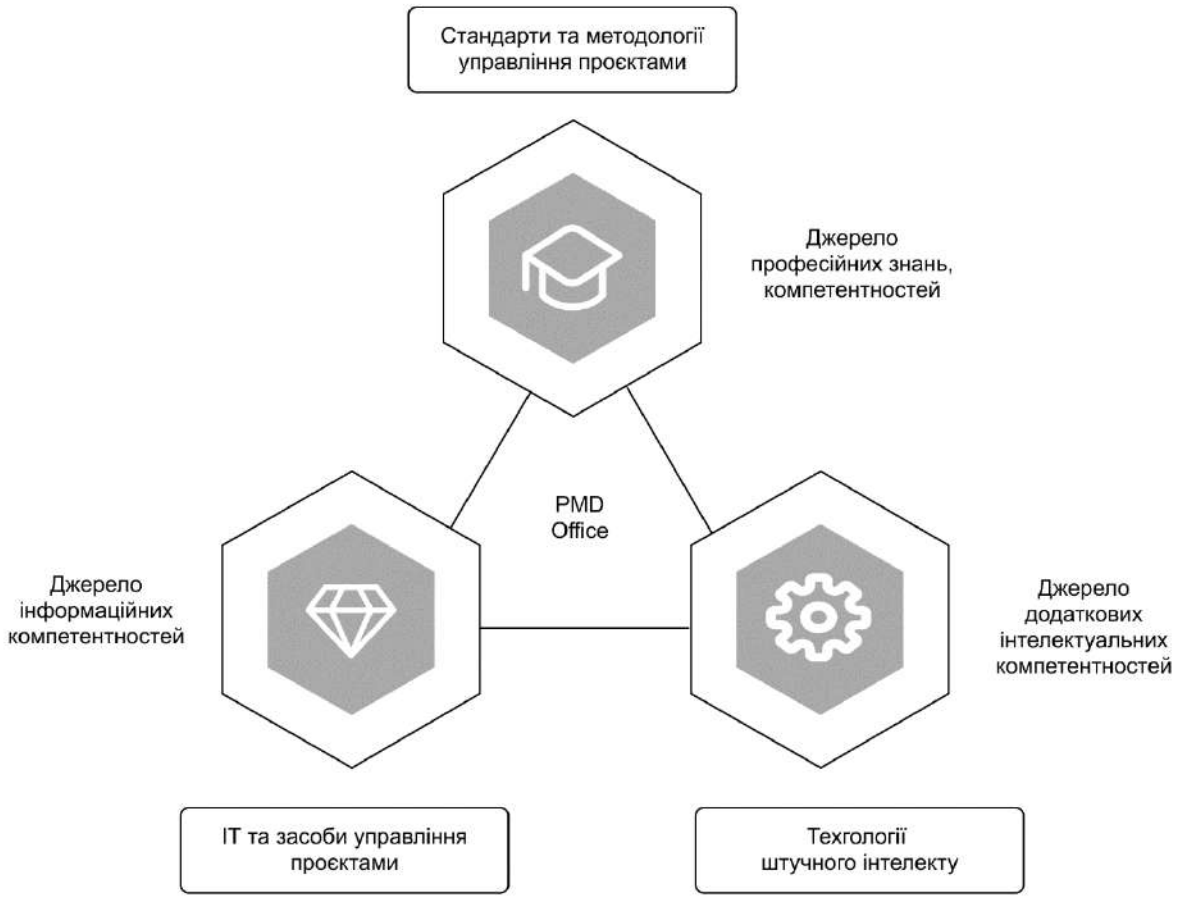


Рис. 4.2. **Модель інтегрованої системи управління проєктами**

Глибоке навчання використовує потужні обчислювальні ресурси нейронних мереж та обчислювальних пристроїв на різних рівнях для пошуку закономірностей у великих наборах даних (наприклад, для виявлення зображень). Ці моделі, які належать до блоку машинного навчання, навчаються на наданих їм даних, тому для досягнення ефективності та точності даних має бути багато.

Нейронна мережа як тип машинного навчання складається із взаємопов'язаних блоків, які опрацьовують інформацію на основі зовнішніх даних і передають цю інформацію між блоками. Блоки працюють разом як нейрони. А нейронна мережа становить обчислювальну систему, що складається із простих, взаємопов'язаних елементів, які опрацьовують інформацію, динамічно реагуючи на зовнішні вхідні сигнали. Треба зазначити, що лише останніми роками виникло достатньо обчислювальних ресурсів для просування вперед розвитку нейронних мереж [25].

У нейронах високого порядку вихідні сигнали нейронів повертаються до того самого нейрона або до нейронів попередніх шарів, як показано на рис. 4.3. Сигнали передають у прямому та зворотному напрямках. Штучні нейронні мережі високого порядку переважно ґрунтуються на моделі Гопфілда.

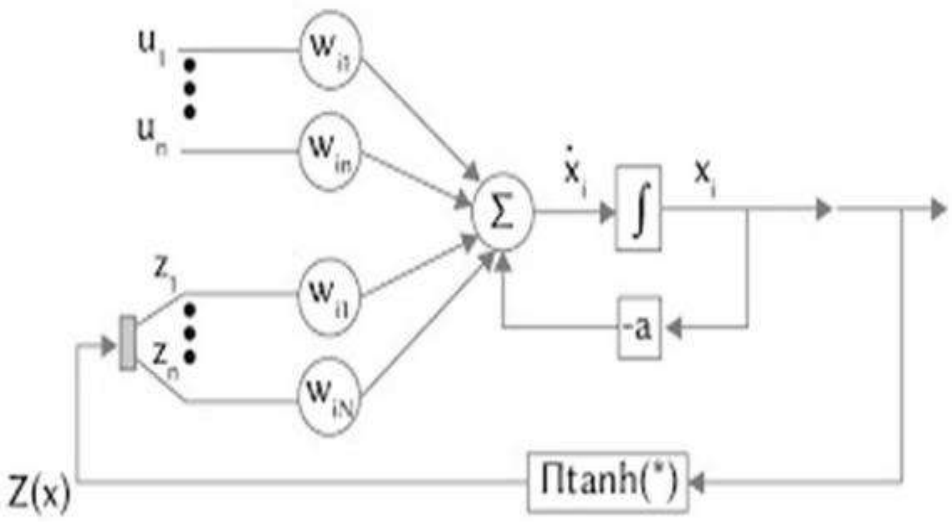


Рис. 4.3. **Нейронна мережа високого порядку**

Нейронна мережа Гопфілда (HNN) є формою штучної нейронної мережі високого порядку з одним шаром повністю пов'язаних нейронів і надає метод для вирішення завдань комбінаторної оптимізації. HNN гарантовано зведено до локального мінімуму, якщо проблему можна описати як функцію енергії з мінімумом, що відповідає оптимальному рішенню. Кожен нейрон дістає на вхід сигнали з усіх інших нейронів, далі розсилає сигнал усім іншим нейронам у мережі. Матриця зв'язків мережі HNN має бути симетричною, а елементи головної діагоналі – нулями. Ці нулі означають, що сигнал нейрона не буде передаватися йому самому на вхід. Ці умови мають забезпечити стійкість системи.

Нечітка логіка Fuzzy Logic (FL) є інструментом для опису невизначеності та неточності, імітує режим вищого порядку, у якому людський мозок ухвалює рішення, забезпечує ефективний спосіб для опису складних автоматизованих систем, а також недостатньо визначених або

складних для аналізу предметів та явищ. FL складається з фазифікатора, бази правил, механізму висновку та дефазифікатора.

Підхід FL містить низку проблем, наприклад, конфігурацію функції належності, визначення оператора композиції та здобуття специфічних нечітких правил до програми. Хоча параметри FL можна визначити, використовуючи досвід і знання експертів, визначення цих параметрів за відсутності таких експертів залишається складним, особливо з погляду складних питань.

Нечіткі когнітивні карти (DCM) презентують розширення когнітивних мап і становлять нечітку графічну структуру, яку використовують для подання причинно-наслідкових зв'язків. Їхнє застосування рекомендовано для галузей, де поняття та відносини є принципово нечіткими, як-от політика, історія та стратегічне планування (проєкти). На діаграмі, показаній на рис. 4.4, кожен вузол презентує нечітку множину або подію, яка відбувається певною мірою. Вузли є причинними поняттями й можуть моделювати події, дії, цінності, цілі чи процеси. Використання такої техніки також забезпечує переваги візуального моделювання, моделювання та прогнозування. Аналіз сценарію сприяє визначенню різних альтернатив для досягнення майбутнього стану, що становить гнучкий метод стратегічного планування, який часто використовують в управлінні технологіями.

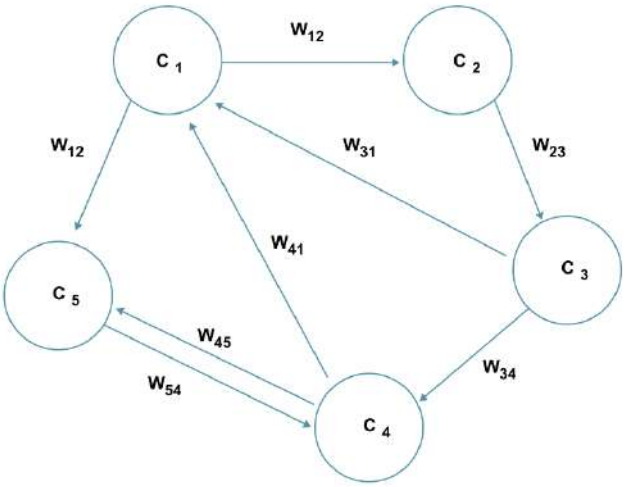


Рис. 4.4. **Топологія дифузної когнітивної мапи**

Незважаючи на те що DCM використовували для аналізу сценаріїв, не вистачає методологій та інструментів, які б уможливили повністю ефективний кількісний аналіз створених сценаріїв. У сфері управління інформаційними технологіями виділяють моделювання проєктів розроблення програмного забезпечення й аналіз ризиків під час обслуговування систем планування ресурсів підприємства (Enterprise Resource Planning System – ERP), хоча використання DCM було запропоновано для інтеграції стратегічного планування щодо інформаційних систем і процесів.

Генетичні алгоритми Genetic Algorithms (GA) презентують адаптивні методи, які може бути використано для вирішення завдань пошуку й оптимізації та засновано на генетичному процесі живих організмів. Сила генетичних алгоритмів полягає в тому, що вони становлять надійну техніку й можуть успішно справлятися із широким спектром проблем у різних галузях, включно з тими, де інші методи стикаються із труднощами. Хоча застосування генетичних алгоритмів не гарантує, що визначить оптимальне рішення для конкретної проблеми, емпіричні дані свідчать про те, що рішення прийнятного рівня можна визначити своєчасно, порівняно з іншими комбінаторними алгоритмами оптимізації. Широке застосування генетичних алгоритмів пов'язано із проблемами, для розв'язання яких немає спеціалізованих методик. Загалом застосування GA для планування кількох проєктів, які мають виконувати одночасно, дало гарні результати. У деяких дослідженнях було прийнято метод, заснований на штрафах, оскільки важко визначити повністю правильні рішення через труднощі завдання оптимізації. Хоча визначені рішення загалом були гарними, важливо підкреслити, що в деяких випадках рішення лежать за межами алгоритмів, оскільки найкращі рішення не завжди відповідають усім обмеженням проблеми.

Генетичний алгоритм швидкого безладдя Fast Messy Genetic Algorithm (FmGA) може ефективно визначати оптимальні рішення завдань із великою кількістю перестановок. Цей тип алгоритму відомий своєю гнучкістю через його здатність поєднуватися з іншими методологіями для досягнення кращих результатів. Різницю між цим та іншими генетичними алгоритмами засновано на можливості модифікації будівельних блоків, щоб визначити найкращі часткові рішення, які допоможуть нам зосередитися на швидшому глобальному рішенні. Алгоритм використовують у багатьох програмах, особливо у сфері управління ресурсами в управлінні проєктами та цивільному будівництві.

Метод опорних векторів Support Vector Machine (SVM) презентує нову форму навчання, яка є більш потужною за традиційні засоби навчання. Цей метод також можна використовувати для розв'язання проблем регресії даних і категоризації. Як і нейронні мережі, SVM потребує навчання та тестування з використанням навчального набору даних. Функції SVM дозволяють краще опрацьовувати невідомі дані, і ця техніка, зазвичай, має певні переваги перед нейронними мережами, які часто успішно застосовують до вартості та управління проєктами. У межах класифікації SVM належить до категорії лінійних класифікаторів, оскільки він індукує лінійні, або гіперплоскі сепаратори; у вихідному просторі вхідних прикладів – роздільний, або квазіроздільний (шум).

Техніка завантаження Bootstrap Technique (BT) – це статистичний метод, який використовують для оцінювання кількостей у певній сукупності шляхом усереднення оцінок із кількох невеликих вибірок даних. Важливо, що вибірки складають шляхом відбору спостережень із великої вибірки даних по одному перед поверненням їх до вибірки даних після їхнього вибору. Це дозволяє це спостереження додати в невелику вибірку понад один раз. Цей підхід до вибірки відомий як вибірка заміни. Метод початкового завантаження можна використовувати для оцінювання розміру цієї сукупності. Цього досягають шляхом багаторазового відбору малих вибірок, обчислення статистичних даних, а потім вилучення середнього. Техніка bootstrap є широко застосованим і надзвичайно потужним статистичним інструментом, який можна використовувати для кількісного оцінювання невизначеності, пов'язаного із цією оцінкою або статистичним методом навчання (наприклад, визначення ймовірності того, що проєкт буде успішним). Цього досягають шляхом навчання моделі з вибіркою та оцінювання ємності моделі щодо вибірок, не включених до основної вибірки. Корисна функція методу завантаження полягає в тому, що вибірка, складена в результаті оцінювання, часто формує гауссівський розподіл. Цю техніку використовують у різних галузях, включно з медициною, фінансовим менеджментом та управлінням проєктами.

Засоби K-групування (K-means) становлять простий підхід до створення груп даних із випадкових наборів даних. Групування K-середніх, яке містить евристики, такі, наприклад, як алгоритм Ллойда, легко реалізувати, навіть із погляду великих наборів даних, і тому широко використовують у багатьох галузях, як-от сегментація ринку, комп'ютерний зір,

геостатистика, астрономія й інтелектуальний аналіз даних у сільському господарстві. Метод також використовують для попереднього опрацювання в інших алгоритмах, зокрема з погляду визначення початкової конфігурації. Хоча головна проблема полягає в тому, що він не може гарантувати оптимальну конвергенцію, він залишається широко використовуваним, завдяки своїй простоті. Багато алгоритмів можуть ідентифікувати конкретні домени. K-середнє зазвичай збігається у практичних застосуваннях, особливо у проблемах розпізнавання образів. Кластеризацію K-середніх також широко й часто використовують, завдяки своїй простоті, хоча вона має певні, притаманні їй недоліки, зокрема наявність фіксованої конфігурації для оптимального рішення є досить трудомісткою.

Найважливішим аспектом штучного інтелекту (ШІ) у його застосуванні для управління проєктами є машинне навчання та використання нейронних мереж, співвідношення яких показано на рис. 4.5.

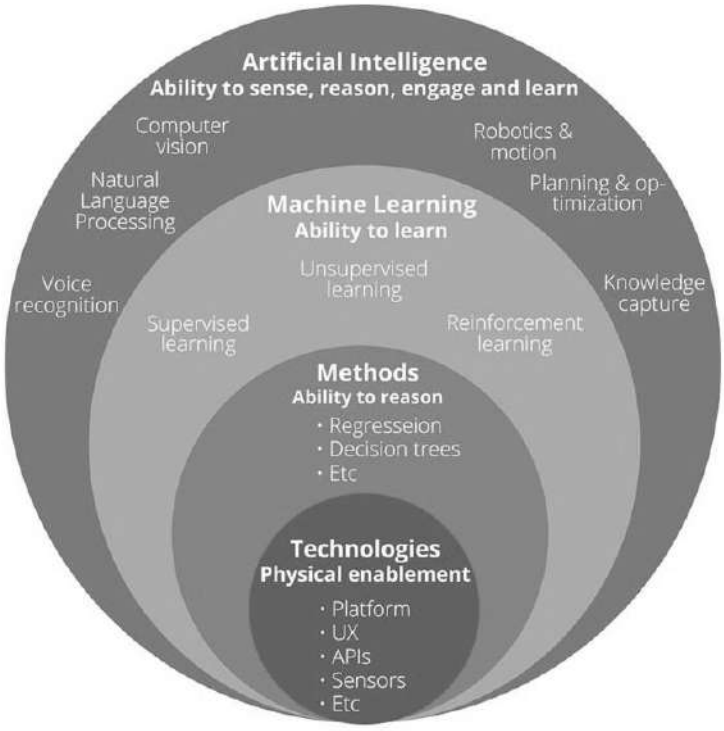


Рис. 4.5. **Структурна модель технологій ШІ**

Розглядаючи технології ШІ з підмножини машинного навчання, яка переважно є математичним алгоритмом, і на основі аналізу великих наборів даних необхідно враховувати динаміку змін у часі та їхній вплив на систему управління проєктами. Це означає, що на основі цих наборів

даних технології ШІ здатні зберігати й опрацьовувати обсяги даних і візуалізувати результати та надавати рекомендації особі, яка ухвалює управлінські рішення.

Технології ШІ дозволяють опрацювати великий обсяг неструктурованих даних, систематизувати їх, проаналізувати та виявити закономірності там, де людський мозок ніколи б їх не помітив. Це відкриває нові можливості для використання даних.

KDD (Knowledge Discovery in Databases). Термін Knowledge Discovery in Databases, або скорочено KDD, належить до широкого процесу пошуку знань даних і підкреслює високорівневе застосування конкретних методів інтелектуального аналізу даних. Він становить інтерес для дослідників у галузі машинного навчання, розпізнавання образів, баз даних, статистики, штучного інтелекту, збирання даних для експертних систем та візуалізації даних.

Об'єднувальною рисою процесу KDD є здобуття знань із даних у контексті великих баз даних. Він робить це за допомогою методів інтелектуального аналізу даних (алгоритмів) для визначення (ідентифікації) того, що вважають знанням, відповідно до специфікацій заходів та порогів, використовуючи базу даних разом із будь-яким необхідним попереднім опрацюванням, підрахунком та перетвореннями цієї бази даних.

Загальний процес пошуку та інтерпретації патернів на основі даних охоплює повторне застосування таких кроків:

розвиток розуміння;

створення цільового набору даних: вибір набору даних або фокусування на підмножині змінних або зразках даних, на яких мають виконувати пошук знань;

очищення та попереднє опрацювання даних;

зведення та прогнозування даних;

вибір завдання інтелектуального аналізу даних;

вибір алгоритму або алгоритмів інтелектуального аналізу даних – інтелектуальний аналіз даних;

інтерпретація здобутих зразків;

консолідація відкритих знань.

CRISP-DM (Cross-industry standard process for data mining). Міжгалузевий стандарт для data mining (інтелектуального аналізу даних),

відомий як CRISP-DM, є моделлю життєвого циклу дослідницького проєкту, яка описує загальні підходи, що використовують експерти з ШІ. Цю аналітичну модель використовують найбільш широко.

CRISP-DM є, можливо, найвідомішою платформою для реалізації проєктів машинного навчання, завдяки своїм перевагам, які розв'язують проблеми в галузях інтелектуального аналізу даних, але він не розкриває процеси, що виникають після впровадження.

CRISP-DM складається із шести фаз: розуміння бізнес-цілей (Business Understanding), вивчення даних (Data Understanding), підготовки даних (Data preparation), моделювання (Modeling), оцінювання (Evaluation), упровадження (Deployment). Передбачено можливість повернутися до попередніх фаз.

ASUM. Багато фахівців з інтелектуального аналізу даних використовують CRISP-DM, але IBM є основною корпорацією, яка зараз використовує цю модель. IBM додав деякі зі старих документів CRISP-DM у свій продукт SPSS Modeler і випустив доповнену версію за назвою ASUM.

Уніфікований метод аналітичних рішень (ASUM) є покроковим посібником зі здійснення повного життєвого циклу реалізації для рішень (завдань) IBM Analytics. Його було створено, щоб скоротити час, що витрачають на виконання деяких процесів, оцінити та знизити ризик, установивши послідовні підходи, що підвищує ефективність загалом. Він містить структуровані етапи, заходи щодо розвитку, ролі та обов'язки, шаблони та принципи. ASUM призначено для створення успішних і відтворюваних IBM Analytics упроваджень продуктів та технологій.

ASUM подано п'ятьма чітко визначеними фазами, наведеними в табл. 4.1.

Таблиця 4.1

Фази ASUM

Фази	Короткий опис
1	2
Аналіз	Визначає, що рішення має містити як у разі ознак, так і разі нефункціональних атрибутів (продуктивність, зручність використання тощо). Досягають згоди між усіма сторонами щодо цих вимог

Закінчення табл. 4.1

1	2
Розроблення/ проєктування/ задум	Визначає всі компоненти рішення та їхні залежності, виконавців та встановлює середовище розроблення. Спринти ітеративного прототипування використовують, якщо застосовують для уточнення вимог
Налагодження та складання	Налагодження, складання та інтеграція компонентів на основі ітеративного й інкрементного підходу. Використовує тестування та перевірки кількох середовищ на основі V-подібної моделі
Упровадження	Створення плану запуску й обслуговування рішення, включно із графіком підтримки. Міграція у виробниче середовище, за необхідності налагодження та взаємодії з аудиторією бізнес-користувачів
Управління та оптимізація	Використовують рішенням IBM Analytics. Робота містить завдання з обслуговування й моніторингу після впровадження, що сприяє успішному використанню рішення та його підтримці
Управління проєктом	Складається із процесів, які допомагають в управлінні й моніторингу ходу та технічного обслуговування проєкту

Кожна фаза є контрольованим потоком управління проєктами, що забезпечує послідовні та скоординовані комунікації та співробітництво.

4.3. Огляд програмного забезпечення для управління IT-проєктами, яке має елементи технології штучного інтелекту

Програмне рішення Aurora – це програмне рішення для інтелектуального планування, яке використовує передовий штучний інтелект, було розроблено, щоб допомогти NASA розв'язати складні, критичні проблеми планування зі складними обмеженнями, включно з рішенням та досвідом експертів із планування. Aurora є особливо ефективною під час використання у великих проєктах зі складними обмеженнями та вимогами до ресурсів.

Протягом останніх 25 років усе більше й більше організацій вибирають Aurora для більш ефективного управління операціями, ніж будь-яку іншу систему, із якою її коли-небудь порівнювали, а для управління

портфелем проєктів (PPM) Aurora виявилася набагато потужнішою, ніж будь-яке інше рішення, включно з Microsoft Project і Primavera P6. Aurora перевершує звичайне програмне забезпечення тим, що використовує технологію штучного інтелекту для кодування та застосування обширних знань і правил планування для автоматичного планування ресурсів у подальшому управлінні. Згідно з дослідженням Boeing, Aurora управляла ресурсами більш ефективно, ніж програмне забезпечення, розроблене та оновлене Boeing протягом майже двох десятиліть, спеціально розроблене для їхнього використання.

Порівняно із Primavera, Aurora скоротила час виконання на 20 %. Порівняно з Microsoft Project графіки Аврори були на 30 % більш короткими. Aurora не тільки є більш ефективною, але й надає багато функцій, недоступних в інших програмах управління проєктами, як-от Microsoft Project і Primavera P3/P6. Більшість інформаційних систем використовують прості правила для вибору та планування подальшої діяльності та розподілу ресурсів для їхньої реалізації. Часто такі графіки є далекими від оптимальних.

Aurora більш ефективно розв'язує складні проблеми, кодуючи й застосовуючи складні знання із планування та правила ухвалення рішень, а також складні обмеження та вимоги до ресурсів. Aurora кодує атрибути окремих завдань, груп завдань, ресурсів, їхніх наборів та обмежень. Убудовані та визначені користувачем правила ухвалення рішень забезпечують кращі розклади, беручи до уваги ці атрибути у ключових моментах планування рішень, наприклад, визначення того, яке завдання планувати наступним, призначення найкращого часу та ресурсів для кожного завдання, щоб оптимізувати загальний розклад і врегулювати ситуації, де деякі ресурси є недоступними, коли завдання заплановано.

На додаток до швидкого та легкого планування, Aurora розв'язує проблеми, які багато систем планування ігнорують. Зосереджуючись на потребах у ресурсах і плануванні часу, Aurora розглядає ці різні аспекти планування разом. Це особливо важливо в галузях, де є ряд потреб у ресурсах. Крім того, це програмне забезпечення надає багато функцій, недоступних в інших програмах управління проєктами, як-от Microsoft Project і Primavera P3/P6, або в іншому програмному забезпеченні для планування обмеженої потужності (рис. 4.6).

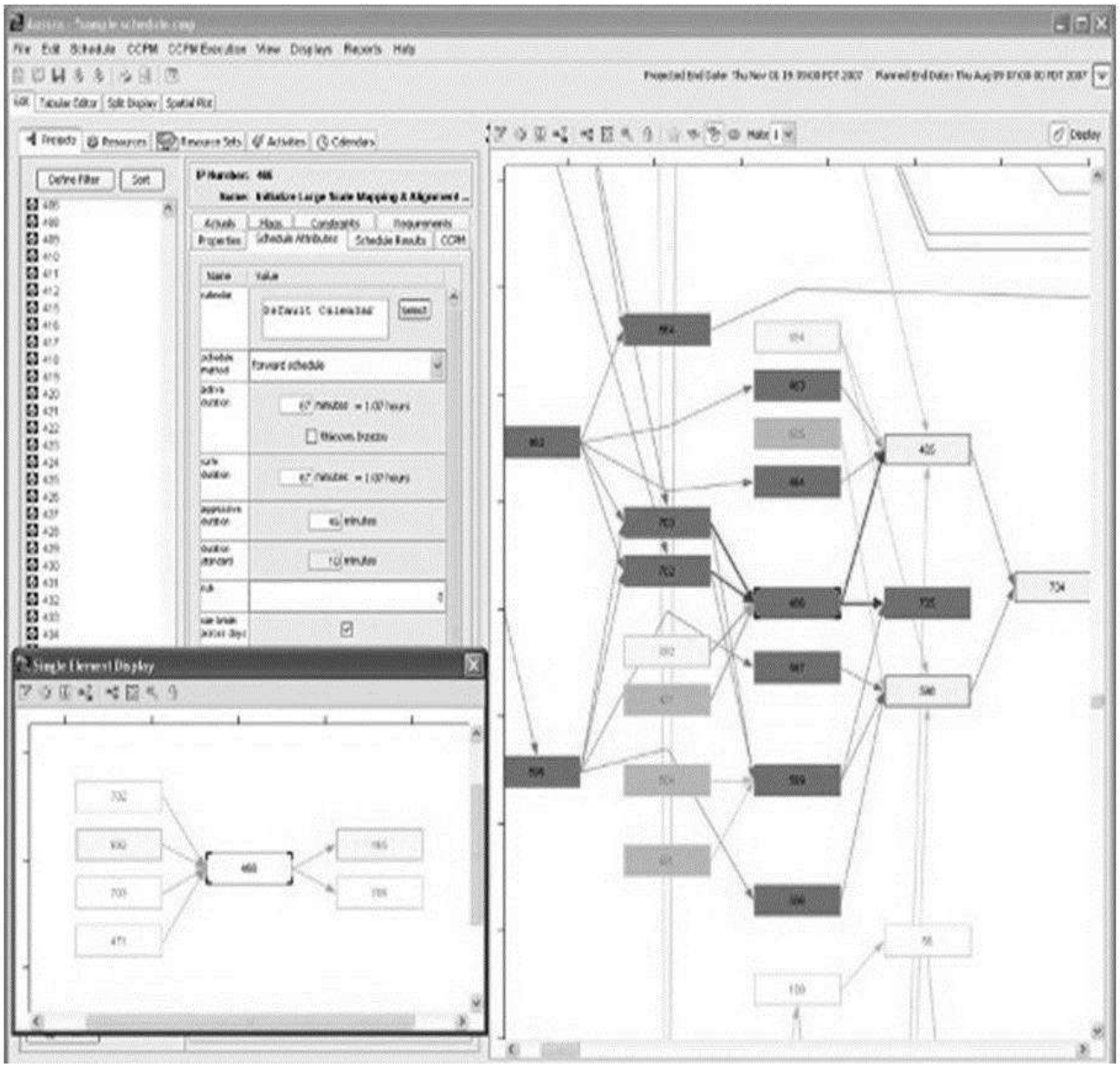


Рис. 4.6. **Дисплей завдань PERT**
програмного рішення Aurora

Особливості Aurora:

управління портфелем проєктів (PPM) / оптимізація планування доступних ресурсів підприємства (ERPO): Aurora оптимізує планування всіх ресурсів підприємства всіх проєктів, які перебувають у роботі;

економія часу для керівників, які будують плани вручну, автоматизуючи їхню роботу;

Aurora може працювати з такими планувальниками, як Primavera P6, Microsoft Project, Siemens TeamCenter, Artemis, PSS, Open Plan, PS8,

Oracle та іншими корпоративними застосунками, робити імпорт, аналіз, будувати й експортувати графіки;

щодо кожного спланованого графіка Aurora має чітке пояснення, чому саме так було розподілено ресурси та чому саме так розташовано кожне завдання у плануванні;

багатошарове інтелектуальне планування з обмеженими ресурсами, тобто інтелектуальний засіб вирівнювання всіх ресурсів приводить до значно коротших графіків проєктів, ніж варіант одношарового вирівнювання ресурсів, який передбачений у Microsoft Project та Primavera P6;

підтримка багатоплатформенного використання, тобто можливість використовувати Aurora саме так, як цього потребує проєкт: вебверсія, можливість використовувати її на таких ОС, як Windows або Linux.

Aurora використовує такі потужні структури та компанії, як NASA, U.S. Air Force, The Boeing Company, Mitsubishi Heavy Industries (MHI), Korea Aerospace Industries Ltd (KAI).

Програмне рішення Forecast – є лідером на ринку управління проєктами. Саме Forecast має найбільшу кількість інтеграцій на сьогодні. Це програмне забезпечення було створено спеціально для управління проєктами. Платформа забезпечує інтуїтивно зрозумілий інтерфейс для управління проєктами, фінансами та витратами, а також скорочує вдвічі рутинну роботу в будь-якій організації за допомогою потужної автоматизації. Власний штучний інтелект програмного засобу полягає в тому, щоб мінімізувати форму завдання, створюючи точні прогнози часу, бюджету та прибутку. Компанії, які хочуть зробити свої міжфункціональні й розподілені команди будь-якого розміру більш організованими та продуктивними, можуть мати потужну двосторонню інтеграцію з іншими інструментами, тому менеджерам не потрібно нав'язувати нові команди інтерфейсу й робочого процесу. Загальну схему роботи програмного рішення Forecast показано на рис. 4.7.

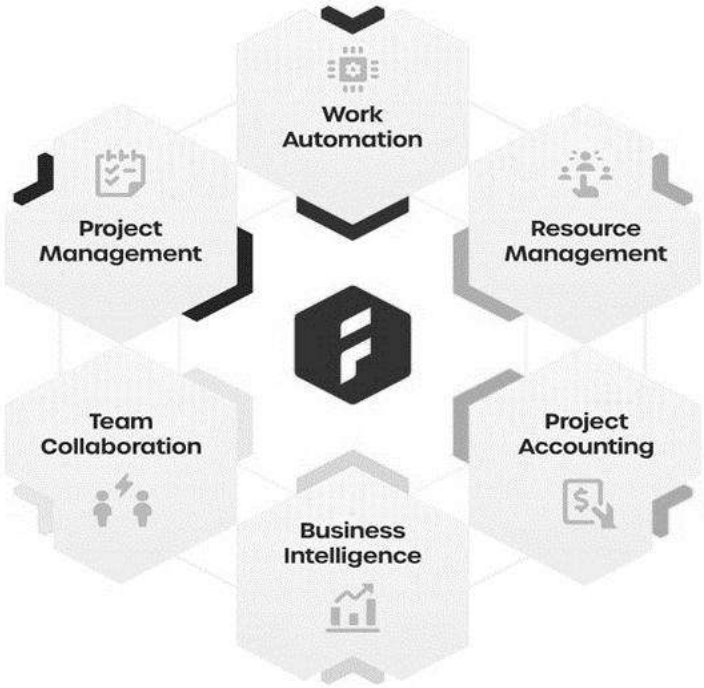


Рис. 4.7. **Структура програмного забезпечення Forecast**

Forecast має власний вбудований штучний інтелект для планування щоденних завдань. Автоматичне створення щоденного графіка дає керівникам проєктів можливість відразу створювати більш вигідні плани. Потужний штучний інтелект вивчає навички та компетентності вашої команди та використовує їх особисто, ураховуючи особливості й індивідуальність кожного співробітника проєкту.

Forecast також має штучний інтелект для відстеження часу, щоб мінімізувати фінансові ризики. Співробітники команди можуть записувати час, витрачений на внутрішню діяльність, поза межами проєктних завдань, наприклад на зустрічі, тренінги, лікарняні тощо.

Forecast має близько 35 інтеграцій, наприклад, месенджери типу Slack і комп'ютерні програми типу Excel.

Програмний сервіс Onebar допомагає швидко дістати потрібну інформацію в корпоративному месенджері, зберегти потрібний файл або створити якісь інструкції, наприклад для початківців.

Artage Risk Burndown Tool (RBd) – це програмне забезпечення, яке допомагає поліпшити управління ризиками за допомогою найсучаснішого прогнозування та машинного навчання для Agile-проєктів. Працює із програмним забезпеченням, JIRA, Version One і CA Agile. Це програмне

забезпечення застосовує всі командні дії для підвищення точності кінцевого результату, забезпечує візуальний аналіз швидкості й тенденцій ризику, використовує передові алгоритми машинного навчання для надання подальших рекомендацій і підтримує різні профілі інноваційного ризику, відповідно до різних проєктів. Керівник IT-проєкту може використовувати інтегрований Artage для допомоги в управлінні ризиками. Програмне забезпечення стежить за розвитком проєкту, його основними напрямами й може на основі свого досвіду звернути увагу менеджера на передбачення критичних точок.

Табл. 4.2 містить порівняльний аналіз програмних систем Aurora Software та Forecast за певними критеріями.

Таблиця 4.2

Порівняльний аналіз програмних систем Aurora Software та Forecast

Можливості програмного забезпечення	Aurora Software	Forecast
1	2	3
Автоматичне створення календаря	так	так
Трекінг неробочого часу	ні	так
План проєкту	так	так
Бюджет проєкту	так	так
Обґрунтування розкладу	так	ні
Пошук доступного співробітника	ні	так
Планування ресурсів на кожне завдання окремо	так	так
Створювання звітів	так	так

Закінчення табл. 4.2

1	2	3
Потенційні конфлікти в розкладі	так	ні
Трирівневе планування	так	ні

Розглядаючи ризики інтегрованої стратегічної моделі управління проєктами з використанням ШІ, слід зазначити такі пастки (логічні помилки, що порушують взаємодії між технічними та соціальними системами):

кадрову – нездатність змоделювати всю систему, щодо якої будуть застосовувати такий соціальний критерій, як справедливість;

портативності – нездатність зрозуміти, як повторне використання алгоритмічних рішень, розрахованих на один соціальний контекст, може ввести в оману, неточним або іншим чином заподіяти шкоду застосуванню в різних умовах;

формалізму – нездатність ШІ врахувати весь сенс соціальних понять, наприклад, справедливість;

хвильового ефекту – нездатність зрозуміти, як використання технології в наявній соціальній системі змінює поведінку та вбудовані цінності вже наявної системи;

солюціонізму – нездатність визнати можливість того, що найкраще розв'язання проблеми може й не містити технологію.

Мінімізувати наслідки цих пасток доцільно на основі використання інституційно-синергетичного підходу з урахуванням поєднання суб'єктивного й об'єктивного початку.

4.4. Висновки

Можливість успіху проєкту – це галузь досліджень, у якій інтенсивно працюють дослідники. Тут початкові підходи ґрунтувалися на статистичних моделях, які не відповідали потребам управління проєктами. У сфері штучного інтелекту дослідники визначили алгоритми й інструменти, які найкраще справляються з різними змінними проєкту та складними середовищами, із специфічними алгоритмами, розробленими для розв'язання

конкретних проблем проєкту. Основні висновки, зроблені з розглянутих робіт, містять те, що інструменти штучного інтелекту є більш точними, ніж традиційні інструменти, але нині вони залишаються доповнювальними до традиційних підходів.

Інструменти штучного інтелекту є дуже корисними для менеджера проєкту з погляду контролю за проєктом та його моніторингу; однак багато з розглянутих моделей мають слабкі сторони та обмеження, що вказує на те, що керівники проєктів мають продовжувати використовувати свій досвід під час оцінювання за результатами. Тенденція до об'єднання різних інструментів штучного інтелекту продовжує панувати, коли сильні сторони одного інструменту можуть компенсувати слабкі сторони іншого. Справді, такий підхід дає найкращі результати, і саме за ним майбутнє. У цій роботі вивчено доступні методи ШІ та можливі застосування у сфері управління проєктами. Може бути запропоновано гібридну обчислювальну модель, яка могла б повністю визначити потенціал ШІ у сфері управління проєктами. Однак автономна система управління проєктами також має враховувати й повністю контролювати середовище проєкту, зокрема з погляду статусу замовників або зацікавлених сторін проєкту. Таку систему можна використовувати для застосування алгоритмів ШІ для психологічного й емоційного аналізу, щоб оцінити як роботу команди, так і задоволеність клієнтів. З огляду на майбутнє, цілком імовірно, що буде штучний інтелект, здатний управляти всім проєктом.

Повільний прогрес ШІ у сфері управління проєктами значною мірою пояснено відсутністю інвестицій із боку приватних компаній. У майбутньому ШІ буде ухвалювати всі рішення та управляти ресурсами оптимально та своєчасно, а менеджер проєкту візьме на себе роль спеціаліста з даних, який буде працювати як частина команди із ШІ, щоб інтерпретувати дані та рішення, тобто менеджери проєктів будуть продовжувати відігравати вирішальну роль, навіть за широкого застосування систем управління з елементами ШІ.

Розділ 5
Інтелектуальна система підтримки ухвалення клінічних рішень на основі мультиагентного підходу та міркувань за прецедентами

5.1. Вступ і формулювання завдання

Нині актуальною є проблема створення систем підтримки ухвалення рішень (СПУР), які все більш широко застосовують під час вирішення складних, важко формалізованих завдань, якими є завдання автоматизації проєктування, робототехніки, управління, діагностики, інформаційного моніторингу та, зокрема, завдання діагностики різних захворювань у медицині [11; 20]. Тобто, основною галуззю застосування СПУР є нестандартні ситуації та слабкоструктуровані проблеми, у яких є необхідними збирання й аналіз великої кількості інформації, потрібної для ухвалення правильного та зваженого рішення та врахування великої кількості різних факторів. Для них є характерним високий ступінь невизначеності, що утруднює ухвалення об'єктивного рішення. Процедура ухвалення рішень у таких ситуаціях потребує механізму визначення системи переваг особі, що ухвалює рішення (ОУР), і більш глибокого порівняльного аналізу альтернативних варіантів. Використання СПУР полегшує роботу керівника, яка є сполучною ланкою між користувачем (ОУР) і джерелами інформації, та підвищує ефективність роботи персоналу, вивільняючи додаткові ресурси.

Під час визначення діагнозу захворювання велике значення має необхідність у врахуванні взаємного впливу різних параметрів один на одного, достовірності вихідних даних, повноти інформації про досліджуваних пацієнтів. Водночас невизначеність тут виникає не тільки через труднощі створення самої моделі об'єкта, а й із-за відсутності в розробника чіткого уявлення про взаємозв'язок можливих рішень, якості використовуваних евристик, а також іншими важкообумовленими факторами. Чим вищим є ступінь невизначеності про ефективність можливих альтернативних рішень і важливості різних критеріїв, які оцінюють ці альтернативи, тим більшого значення набуває експертна оцінка фахівця, зроблена ним на основі досвіду й інтуїції. І оскільки часом експертам

неможливо визначити діагноз у великої кількості людей, допомогу може надати застосування математичних методів, що застосовують у сучасних системах підтримки ухвалення клінічних рішень (СПУКР) [11].

СПУКР призначено для поліпшення надання медичної допомоги шляхом модернізації медичних рішень за допомогою цілеспрямованих клінічних знань, інформації про пацієнтів та іншої медичної інформації. Традиційна СПУКР складається із програмного забезпечення, призначеного для безпосередньої допомоги під час ухвалення клінічних рішень, у якому характеристики окремого пацієнта зіставляють із комп'ютеризованою базою клінічних знань, а потім для лікаря передаються клінічні оцінки або рекомендації для конкретного пацієнта [11]. Сфера функцій, що надають СПУКР, є широкою, включно з діагностикою, системою сигналізації, лікування захворювань, призначення ліків, приймання ліків та багато іншого. Їх можуть виявляти у вигляді комп'ютеризованих попереджень і нагадувань, комп'ютеризованих посібників, наборів замовлень, звітів про пацієнтів, шаблонів документації та інструментів клінічного робочого процесу тощо.

Є декілька підходів до побудови СПУКР [11; 20; 28]:

1) надання лікарю релевантних інформаційних джерел, що допомагають йому самостійно ухвалити рішення;

2) використання клінічних шляхів, так звані технологічні карти, які є прескриптивними моделями стандартних процедур охорони здоров'я, що має бути зроблено для конкретної вибірки пацієнтів;

3) розроблення широкого кола приватних вузькопрофільних СПУКР, зокрема SkyChain, яка пропонує навчені нейронні мережі для аналізу діагностичних зображень різних захворювань;

4) побудова когнітивної системи, здатної до самонавчання та засвоєння знань безпосередньо з текстових неформалізованих джерел, наприклад, система IBM Watson, заснована на явних знаннях;

5) побудова СПУКР із використанням прецедентного підходу, який ґрунтується на неявних емпіричних знаннях і згідно з яким пропонують формувати банки клінічних даних, визначити в них випадки – прецеденти, схожі на поточний, і рекомендувати лікувально-діагностичні заходи на основі визначених прецедентів.

Вибір необхідного підходу до побудови СПУКР впливає на якість і строки ухвалення рішень із діагностики захворювання та надання медичної допомоги пацієнту.

Для аналізу та вироблення пропозицій СПУКР можуть використовувати різні методи: інформаційний пошук, інтелектуальний аналіз даних, пошук знань у базах даних, міркування на основі прецедентів, імітаційне моделювання, еволюційні обчислення й генетичні алгоритми, нейронні мережі, ситуаційний аналіз, когнітивне моделювання та ін. Основні обмеження відомих методів і технологій, що використовують нині для вирішення важкоформалізованих завдань, обумовлені недостатньою ефективністю розв'язання в них проблем навчання, налаштування й адаптації до проблемної області, опрацювання неповної та неточної вихідної інформації, інтерпретації даних і накопичення знань експертів, однакового подання інформації, що надходить із різних джерел, тощо. Ці обмеження може бути зняти, використовуючи синергізм між мультиагентними системами (MAS – Multi-Agent System) і міркуваннями на основі прецедентів (CBR – Case-Based Reasoning) [28].

Міркування на основі прецедентів є підходом, що дозволяє вирішити нове, невідоме завдання, використовуючи або адаптуючи рішення вже відомого завдання, тобто використовуючи вже накопичений досвід вирішення подібних завдань. Основна мета використання апарату прецедентів у межах СПУКР полягає у видачі готового рішення оператору (ОУР) для поточної ситуації на основі прецедентів, які вже мали місце в минулому в разі управління даними або подібним об'єктом (системою).

В основі мультиагентного підходу лежить поняття інтелектуального агента, який перебуває в середовищі та функціонує як самостійна комп'ютерна програма, а також здатен на взаємодію з ним для досягнення своїх цілей. Основною ідеєю мультиагентних технологій є принципово новий підхід до вирішення поставленого завдання. Так за традиційного підходу розв'язання проблеми відбувається за заздалегідь визначеного алгоритму, який забезпечує розв'язання конкретної проблеми, тоді як у мультиагентних технологіях рішення досягають у результаті множинної взаємодії автономних програмних систем, так званих агентів, у яких немає чіткого алгоритму взаємодії.

У мультиагентних системах кожній сутності системи ставлять у відповідність такий самий програмний агент, у завдання якого входить подання її інтересів. Агенти схожі на членів однієї команди, які мають можливість змагатися один з одним або допомагати один одному у процесі пошуку рішення. Їх важливою особливістю є динаміка й непередбачуваність процесу пошуку рішень. На практиці це дає такий ефект, що ухвалення рішення досягають за допомогою сотень і тисяч міжагентних

взаємодій, які дуже важко відстежити в реальному часі. Але слід зазначити, що це й не потрібно, оскільки агентам дають завдання, які вони ставлять собі за цілі, яких вони мають досягати. Водночас не визначають алгоритм або сценарій, за яким вони мають рухатися для досягнення цих цілей. Ці сценарії агенти формують самостійно, після чого вони їх і виконують. Під час роботи агенти виконують моніторинг системи і реагують на події, які в ній відбуваються.

Отже, дослідження СПУКР на основі MAS, що використовують CBR, для підвищення ефективності реалізації в них механізмів навчання й адаптації до особливостей проблемної середовища є важливими та актуальними.

У роботі поставлено завдання розроблення СПУКР із використанням мультиагентних технологій, яке полягає в дослідженні методів, спрямованих на інтеграцію агентів у СПУКР, а також використання алгоритмів міркувань на основі прецедентів для підвищення здатності агентів до ухвалення рішень. Результатом дослідження є створення архітектури програмного забезпечення, що підвищує ефективність процесів опрацювання інформації у СПУКР і дозволяє використовувати розроблені методи та програмні системи, що відповідають конкретним застосункам. Запропоновані методи й алгоритми може бути застосовано під час вирішення конкретних завдань ухвалення рішень.

Запропоновано підхід, який передбачає організацію пошуку діагнозу з використовуванням міркувань на основі прецедентів у поєднанні з мультиагентною архітектурою, яка забезпечує гнучкість та адаптивність системи, що значно розширює сферу її застосування, сприяє більш глибокому впровадженню сучасних методів і засобів опрацювання інформації, де рішення ухвалюють на основі прогнозування, аналізу й оцінювання.

5.2. СПУКР на основі MAS, що використовує CBR

5.2.1. СПУКР на базі мультиагентного підходу

Загальну структуру мультиагентної СПУКР для діагностування захворювань показано на рис. 5.1, яка складається із трьох модулів високого рівня: модуля інтерфейсу; модуля виконання; модуля знань. Зазначені модулі містять таких агентів: 1) агента з інтерфейсу; 2) агента-координатора; 3) агента з ухвалення рішень; 4) агента зі звітності; 5) агента з бази даних.

Рис. 5.1. **Структура СПУКР на основі мультиагентного підходу**

Модуль інтерфейсу є загальнодоступним для інших агентів та користувачів. Він забезпечує механізми взаємодії з агентом та підтримує міжагентне спілкування та співпрацю. До цього рівня належать агенти з інтерфейсу, які забезпечують доступ до агентів з ухвалення рішень. Доступ до модуля виконання та модуля знань обмежено лише агентами, які там містяться, тобто інші агенти чи користувачі не можуть безпосередньо маніпулювати вмістом цих модулів без прав доступу.

Модуль виконання містить методи й евристики, які реалізують різноманітні функції та процеси, за допомогою яких агент може реагувати на запити від інших агентів або користувачів. Отже, модуль процесу переважно забезпечує послуги й обчислення, які можуть бути необхідними для розв'язання конкретної проблеми. До цього рівня належать: агент-координатор, агент з ухвалення рішень, а також агент зі звітності.

Модуль знань містить домени та незалежні від домену знання, що стосуються розв'язання проблем. На цьому рівні міститься агент із бази даних, який надає дані, необхідні для інших агентів.

Агенти є інтелектуальними та становлять складні програмні системи, які мають здатності до підлаштування під конкретну обставину, що викликає зміну їхньої поведінки або характеристик для забезпечення здатності до адаптації та в підсумку розв'язання поставленої перед ними проблеми. Інтелектуального агента можна описати у вигляді набору [28]:

$$AG = (S, A, env, I, refine, action), \qquad (5.1)$$

де S – непорожня кінцева множина станів зовнішнього середовища;

A – непорожня кінцева множина дій агента;

env – функція поведінки (стану) зовнішнього середовища;

I – непорожня кінцева множина внутрішніх станів агента;

refine – функція оновлення стану, зіставляє попередній внутрішній стан із новим станом зовнішнього середовища та впливає на новий внутрішній стан агента;

action – функція ухвалення рішення, що зіставляє поточний внутрішній стан агента з деякою дією.

Отже, **агент** – це самостійна програмна система, яка має можливість приймати вплив із зовнішнього світу; визначає свою реакцію на цей вплив і формує відповідну дію; змінює свою поведінку із плином часу, залежно від накопиченої інформації та вилучених із неї знань; володіє мотивацією та здатна після делегування повноважень користувачем поставити себе на його місце й ухвалити рішення, відповідне ситуації.

5.2.2. MAS із використанням CBR

Мультиагентну технологію використовують із метою визначення, повторного використання й адаптації прецедентів у системі CBR. Розгляньмо структуру системи, яка об'єднує технології MAS і CBR для підтримання динамічного обміну знаннями, показана на рис. 5.2.

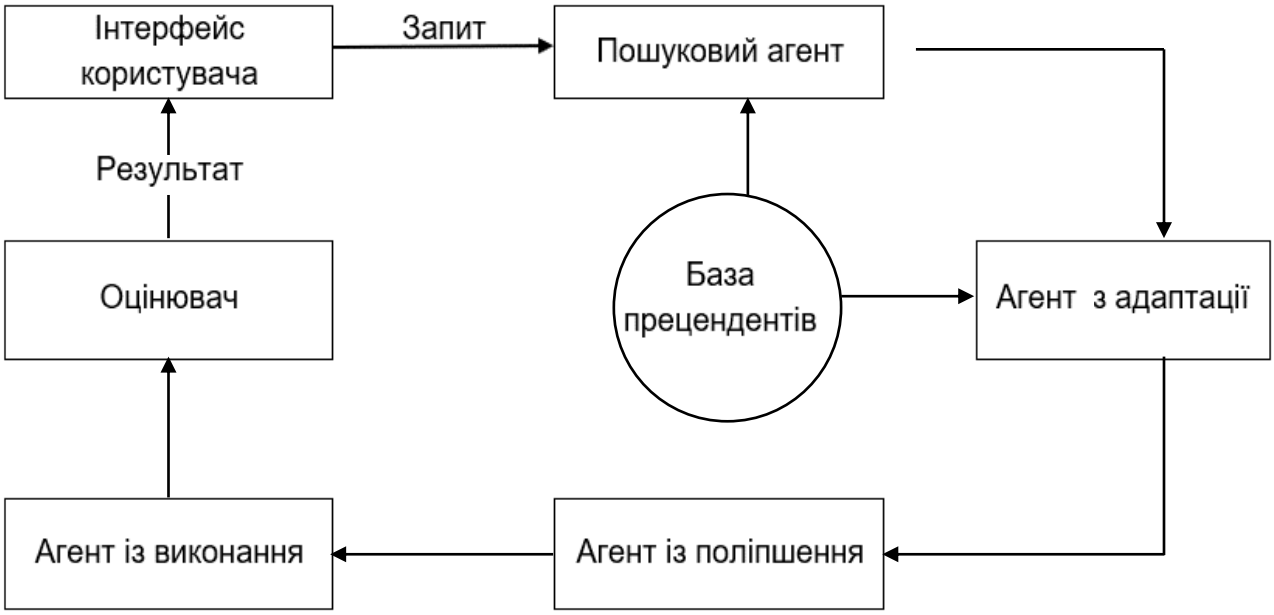


Рис. 5.2. **Структура системи CBR із використанням мультиагентних технологій**

Структура системи характеризується диференціальними рівнями обміну знань, залежно від застосунків, а також створенням нових знань і повторним використанням на основі попередніх знань у проблемній галузі. Вона також забезпечує новий інструмент для управління міжорганізаційними знаннями. Система CBR вилучає з бази прецедентів випадки, пов'язані з наявною проблемною ситуацією, та вирішує питання розв'язання поточної проблеми на підставі результатів попередніх справ. Система CBR із використанням мультиагентних технологій складається з таких основних агентів (див. рис. 5.2):

1) *пошуковий агент:* якщо нову проблему вводять до системи, пошуковий агент виконує функції з пошуку випадку найбільш схожого до проблеми. Пошук завершують, коли агент знайшов індекси всіх випадків, прийнятних до проблеми;

2) *агент з адаптації:* визначає різницю між вибраними випадками та проблемою, і якщо потрібно, застосовує набір необхідних правил для того, щоб старе рішення як найкраще підходило до нової проблеми;

3) *агент із поліпшення:* він критикує адаптоване рішення проти попередніх результатів. Один із способів зробити це – порівняти його з аналогічними рішеннями попередніх випадків. Якщо наявна відома помилка для похідного рішення, система потім вирішує, чи є подібність достатньою, щоб підозрювати, що нове рішення буде невдалим;

4) *агент із виконання:* після того як рішення критикують, виконавець застосовує вишукане рішення до поточної проблеми;

5) *оцінювач:* якщо результати є такими, як очікували, подальший аналіз не здійснювали, а випадки та їхнє рішення зберігають або використовують для розв'язання майбутніх проблем, якщо ні, рішення буде переглянутим.

Пошуковий агент прецеденту займається процесом пошуку прецеденту, схожого на нове завдання, який може бути орієнтовано на різні цілі, наприклад: визначити такий прецедент, щоб результівний діагноз виявився найбільш надійним; визначити прецедент із діагнозом, який має мінімальний час виконання; визначити прецедент, для якого час вибору нового рішення виявиться мінімальним (мінімум модифікацій); визначити прецедент, який відображає найбільш сучасний (пізній) досвід тощо. Зазвичай, завдання пошуку прецеденту розподіляють на три підзавдання:

1) вибір властивостей для зіставлення (feature identification);

2) зіставлення (matching);

3) вибір рішення (selection або ranking).

Перше підзавдання має на увазі вибір властивостей, які має бути взято до уваги під час пошуку кращого прецеденту. Зазвичай системи діагностування, засновані на прецедентах, використовують для цього мету, початковий стан та іноді опис труднощів (failures), які можуть виникнути під час вирішення завдання.

Підзавдання зіставлення полягає в пошуку одного або більше діагнозів, які (частково) збігаються з вибраним властивостями з поточною проблемою. Зіставлення можна виконувати декількома різними способами. Найбільш поширеними є повне зіставлення та використання заходів подібності (іноді, метрик подібності).

У разі повного зіставлення виконують пошук прецедентів, у яких вибрані властивості точно збігаються з відповідними властивостями нового завдання. Якщо система не визначає потрібного рішення, то здійснюють узагальнення вибраних властивостей нового завдання й потім знову роблять спробу визначити прецедент.

Міра подібності (similarity measure) – це функція, яка обчислює ступінь подібності заданого прецеденту та нової проблеми. Найпростішу міру подібності можна визначити як кількість загальних підцілей. Більш гнучкі заходи подібності обчислюють зважену суму загальних підцілей і ступінь подібності початкових станів. Ваги слугують для балансування важливості й корисності підзавдань і їх можна визначати константами для всіх прецедентів, але в більш складних предметних галузях кожен прецедент має свій власний вектор ваги.

У результаті роботи агента пошуку прецеденту може бути визначено декілька відповідних прецедентів. У цьому разі планувальник має вибрати один або декілька з них. Під час використання заходів подібності це підзавдання вирішується автоматично. Але можна використовувати й інші методи, засновані, наприклад, на ступені узагальненості прецеденту (чим більш конкретним є прецедент, тим менше буде потрібно зусиль на адаптацію).

Розгляньмо способи подання та вилучення прецедентів. На першому етапі CBR-циклу – здобуванні прецедентів – виконують визначення ступеня подібності поточної ситуації із прецедентами з бази прецедентів (БП) і подальше їхнє вилучення, із метою розв'язання цієї нової проблемної ситуації. Для успішної реалізації міркувань на основі прецедентів, необхідно забезпечити коректне вилучення прецедентів із БП. Вибір

методу визначення прецедентів безпосередньо пов'язано зі способом подання прецедентів і відповідно до організації БП.

БП може входити до складу бази знань інтелектуальної СПУКР, але може бути самостійним компонентом системи. Структура БП істотно впливає на різні показники роботи системи і, зокрема, на час пошуку та вилучення прецедентів. Є різні способи подання та зберігання прецедентів – від простих (лінійних) до складних (ієрархічних). Слід зазначити, що прості способи, які ґрунтуються, зазвичай, на технології реляційних баз даних, потребують значно нижчих витрат на реалізацію, а також на підтримання та супровід БП, ніж складні. Однак часу для пошуку рішення в разі простого подання прецедентів може знадобитися значно більше, порівняно з іншими більш складними способами подання та збереження прецедентів.

Прецедент у загальному випадку може містити такі компоненти:

опис завдання (проблемної ситуації);

рішення завдання (діагностування ситуації та рекомендації ОУР);

результат (або прогноз) застосування рішення.

Результат може містити список виконаних дій, додаткові коментарі та посилання на інші прецеденти. Прецедент може мати як позитивний, так і негативний результат застосування рішення, також у деяких випадках може здійснювати обґрунтування вибору запропонованого рішення та можливі альтернативи.

Основні способи подання прецедентів можна розподілити на такі групи:

параметричні;

об'єктно-орієнтовані;

спеціальні (у вигляді дерев, графів, логічних формул тощо).

Здебільшого для подання прецедентів досить простого параметричного уявлення, тобто уявлення прецеденту у вигляді набору параметрів із конкретними значеннями та рішенням (діагнозом і рекомендаціями ОУР):

$$CASE = (x_1, \ldots, x_n, R), \qquad (5.2)$$

де x_1, ..., x_n – параметри ситуації, яка описує цей прецедент; $x_1 \in X_1$, ..., $x_n \in X_n$;

n – кількість параметрів прецеденту;

X_1, ..., X_n – межі допустимих значень відповідних параметрів;

R – діагноз і рекомендації ОУР.

Додатково може бути наявним опис результату застосування визначеного рішення та додаткові коментарі.

Є цілий ряд методів вилучення прецедентів і їхніх модифікацій, найбільш поширеними з яких є такі [5; 6]:

найближчого сусіда (NN – Nearest Neighbor);

дерев рішень;

на основі знань;

штучних нейронних мереж тощо, які можуть використовувати різні метрики.

Метод найближчого сусіда – найбільш використовуваний метод порівняння та вилучення прецедентів. Він дозволяє досить легко обчислити ступінь подібності поточної проблемної ситуації та прецедентів із БП. Для визначення ступеня подібності на множині параметрів, використовуваних для опису прецедентів і поточної ситуації, уводять деяку метрику. Далі, відповідно до вибраної метрики, визначають відстань від цільової точки, відповідної поточній проблемній ситуації, до точок, що подають прецеденти із БП, і вибирають найближчу точку до цільової. До істотних недоліків методу можна зарахувати труднощі вибору метрики для визначення ступеня подібності та пряму залежність необхідних обчислювальних ресурсів від розміру БП, а також неефективність під час роботи з неповними та неякісно визначеними (так званими зашумленими) вихідними даними.

Метод вилучення прецедентів на основі *дерев рішень* ґрунтується на визначенні необхідних прецедентів шляхом дозволу вершин дерева рішень. Кожна вершина дерева вказує, із якої її гілки слід здійснювати подальший пошук рішення. Вибір гілки здійснюють на основі інформації про поточну проблемну ситуацію. Необхідно дістатися до кінцевої вершини, яка відповідає одному або декільком прецедентам. Такий підхід рекомендовано застосовувати для великих БП, тому що основну частину роботи з вилучення прецедентів виконують заздалегідь на етапі побудови дерева рішень, що значно скорочує час пошуку рішення.

Метод вилучення прецедентів *на основі знань* дозволяє врахувати знання експертів (ОУР) із конкретної предметної галузі під час вилучення прецедентів. Під час визначення прецедентів ураховують важливості параметрів прецедентів, задані експертом або ОУР, й інша інформація, що дозволяє врахувати знання про конкретну предметну галузь. Завдяки цьому значно скорочено час пошуку рішення, що є істотною перевагою цього методу.

В основі методів вилучення на основі *застосування прецедентів* лежить той факт, що вилучення прецедентів ґрунтується не тільки на їхній схожості з поточною проблемною ситуацією, але й на тому, наскільки якісну для бажаного результату модель вони становлять. Отже, на вибір здобутих прецедентів впливає можливість їхнього успішного застосування (адаптації) у конкретній ситуації, тобто наявність відомостей про їхню застосовність у ситуації, що склалася. У деяких системах цю проблему розв'язують шляхом збереження прецедентів разом із коментарями щодо їхнього застосування. Використання цього методу дозволяє зробити пошук рішення більш ефективним, заздалегідь відкидаючи частину свідомо неперспективних прецедентів.

Крім розглянутих методів вилучення прецедентів, можна успішно застосовувати й інші методи, наприклад, апарат *штучних нейронних мереж* (ШНМ). Безумовно, добре навчена ШНМ здатна успішно й досить швидко вирішувати завдання класифікації, кластеризації та визначення схожих прецедентів, але проблеми із ШНМ полягають у необхідності у використанні представницької навчальної вибірки для навчання мережі із заданою точністю й істотних утрат часу на навчання ШНМ. Крім того, виникає проблема, пов'язана з розробленням спеціальної топології ШНМ, орієнтованої на конкретну проблемну галузь і розв'язання складних багато-параметричних задач.

У роботі використано метод найближчого сусіда, який дозволяє досить легко обчислити ступінь подібності поточної проблемної ситуації та прецедентів, які зберігають у БП.

Нехай заданий прецедент (С) і поточну проблемну ситуацію (Т) задано в *n*-вимірному просторі ознак (властивостей, якостей). Тоді ступінь подібності або близькості S (C,T) прецеденту С і поточної ситуації Т мож-

на визначити, використовуючи одну з основних метрик, що визначають відстань між двома точками x_i^C і x_i^T, зокрема, евклідову відстань:

$$d_{ст} = \sqrt{\sum_{i=1}^{n}(x_i^C - x_i^T)^2} . \qquad (5.3)$$

Далі, відповідно до вибраної метрики, визначають відстань від цільової точки, що відповідає поточній проблемній ситуації, до точок, що подають прецеденти із БП, і вибирають найближчу точку до цільової.

Для визначення значення ступеня подібності $S(C,T)$ необхідно визначити максимальну відстань d_{max} у вибраній метриці, використовуючи межу діапазонів параметрів для опису початкового $x_{i,in}$ і кінцевого $x_{i,fin}$ прецедентів, $i=1,...,n$. Потім можна обчислити значення ступеня подібності:

$$S(C,T) = 1 - d_{ст} / d_{max}, \qquad (5.4)$$

яке може набирати значення від 0 до 1.

Робота **агента з адаптації** складеться із двох частин: по-перше, у підміні мети й початкових умов вибраного прецеденту метою та початковими умовами нового завдання; по-друге, у забезпеченні коректності діагнозу після підміни. Після підміни мети й початкових умов діагноз може потребувати змін. Деякі кроки діагнозу можуть виявитися непотрібними, оскільки зникла мета, досягненню якої вони слугували. Можуть виникнути нові цілі та, крім того, зміна початкових умов може призвести до непридатності ряду кроків. Коригування діагнозу може виконуватися або автоматично, або користувачем.

Результатом виконання фаз пошуку й адаптації є план для розв'язання поточної проблеми. Щоб замкнути CBR-цикл і поповнити систему знань, необхідно зберегти поточне планування досвіду. Навчання здійснюють на основі спостережень за відповідною реакцією під час виконання плану.

Після того як агент із виконання застосовує вишукане рішення до поточної проблеми, оцінювач аналізує результати та якщо результати

є такими, як очікували, та подальший аналіз не здійснюють, а випадки та їхнє розв'язання зберігають або використовують для розв'язання майбутніх проблем. Якщо ні, рішення буде переглянуте.

Агент з інтерфейсу користувача може бути розподілений на безліч агентів, залежно від потреб системи. У лікарні найбільш вдало використовувати розподіл на два типи користувачів:

1) інженери (цей тип користувачів займається налаштуванням і моніторингом роботи системи ухвалення рішень);

2) лікарі (цей тип користувачів займається аналізом даних пацієнтів).

Кожен користувач має власний агент з інтерфейсу. Зазвичай, агентом з інтерфейсу користувача є вебінтерфейс, який дозволяє користувачам взаємодіяти із СПУКР, але таких інтерфейсів може бути кілька різних типів, наприклад, це може бути настільною програмою або мобільним застосунком. За допомогою цього інтерфейсу користувачі можуть заповнювати онлайнові форми для взаємодії із СПУКР, виконувати завантаження документів та здійснювати аналіз даних. Лікар може надати дані про пацієнтів лікарні для аналізу їх за допомогою СПУКР, а інженер може виконати завантаження додаткових даних. Агент з інтерфейсу користувача відповідає за здобуття даних від користувачів та надання результатів. Він також відстежує налаштування користувача. Може підлаштовувати інтерфейс, залежно від ролі користувача та його цілей, наприклад, агент з інтерфейсу може приховувати деяку інженерну інформацію від лікарів.

Інтерфейсний модуль агента користувача містить методи для міжагентного зв'язку, а також дістання введення від користувача. Функції агента з інтерфейсу забезпечують таке: вебінтерфейс для взаємодії з користувачем; вебсторінку для опису проблем; надання параметрів налаштування; вебсторінку з інформації про статус виконання аналізу, надання відгуків про стан різних процесів; вебсторінку, що містить кінцеві результати; динамічне створення HTML-документів з особливими форматуваннями, залежно від користувача, тощо.

Агент-координатор відповідає за координацію різних завдань, які необхідно виконати під час вирішення кооперативних завдань. Отримавши завдання користувача від агента з інтерфейсу, агент-координатор ідентифікує відповідні критерії, визначає альтернативу, яку потрібно оцінити, і генерує план дій, рейтинг альтернатив. Ці альтернативи можуть містити ідентифікацію відповідних джерел даних, запит послуг інших агенцій

та створення звітів. Інтерфейсний модуль агента-координатора відповідає за міжагентну комунікацію.

Модуль процесу містить методи контролю та погодження різних завдань, а також створення послідовності завдань. Послідовність виконаних завдань створюють за допомогою спеціальної формули, які зберігають у модулі знань, використовуючи підхід, які ґрунтується на правилах. Модуль знань також містить метазнання про можливості інших агентів у федерації, доступних джерелах даних і базах даних. Агент-координатор може шукати послуги групи агентів і синтезувати кінцевий результат.

Функції агента-координатора залежать від уведення користувача, визначають цілі високого рівня, на основі цих цілей визначають завдання, генерують послідовність завдань і делегують дії для відповідних агентів, надають проміжні відгуки користувачеві, синтезують та генерують кінцевий результат.

Після надходження запиту до агента-координатора на ухвалення рішення, він визначає, які дії потрібно зробити для успішного опрацювання заявки від користувача, для цього він може посилати запити до агента з бази даних для пошуку додаткових знань про предметну галузь, а також він посилає запит до агента з ухвалення рішень з інформацією про проблему.

Агент з ухвалення рішень зберігає в модулі знань інформацію про можливих агентів, здатні розв'язати проблему. Використовуючи цю інформацію і залежно від типу заявки, агент має визначити на основі метаданих, які зберігають у модулі знань, той модуль, який найкраще з нею справиться, і відправити до нього запит на розв'язання проблем. У нашій системі такий модуль один, це модуль ухвалення рішень із використанням CBR.

Агент зі звітності перебуває в модулі виконання й надає користувачам сервіси для генерації різних звітів про роботу системи. Це можуть бути як текстові звіти, так і різні діаграми: стовпчасті та лінійчасті діаграми; графіки; секторні діаграми; гістограми; діаграми з галузями; точкові діаграми; поверхневі діаграми; кільцеві діаграми; бульбашкові діаграми тощо. Ці звіти будують на основі інформації, яку агент дістає від агента з бази даних і можуть містити різноманітну інформацію про параметри розподілу даних у базі знань. Наприклад це можуть бути гістограми захворювань серця, показані на рис. 5.3.

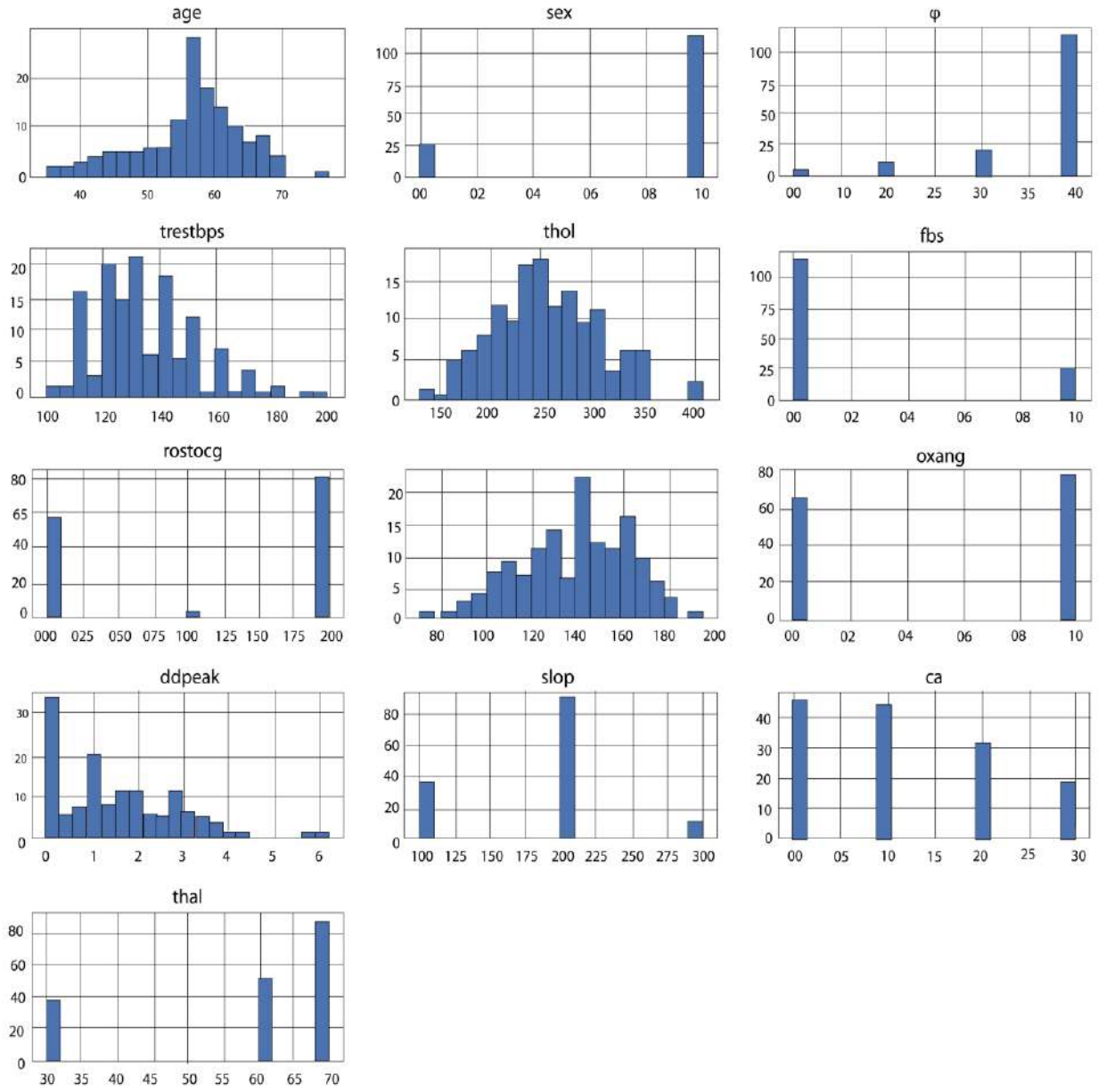


Рис. 5.3. **Гістограми захворювань серця**

Така інформація може бути дуже важливою для аналізу симптомів, які можуть призвести до захворювання серця. Наприклад, якщо побудуємо гістограму розподілу різних атрибутів людей, у яких було виявлено захворювання серця, що показано на рис. 5.3, можемо побачити, що кількість людей із захворюваннями різко збільшується після 50 років. Усе це може бути дуже корисним для подальшої діагностики захворювань серця.

Агент із бази даних відповідає за відстеження того, які дані зберігають у базі даних. Він забезпечує попередньо визначені та спеціальні можливості пошуку, а також бере на себе відповідальність за здобуття необхідних даних, яких потребує агент з опрацювання даних, під час підготовки до конкретної операції з пошуку даних. Агент із бази даних ураховує

неоднорідність баз даних, які можуть бути, і розв'язує конфлікти під час визначення та подання даних.

Інтерфейсний модуль агента з бази даних забезпечує загальнодоступний інтерфейс до наявних баз даних. Це поліпшує взаємозв'язок і дозволяє користувачам дістати доступ до різних джерел даних, які в іншому разі можуть бути недоступними. Модуль процесу надає можливості спеціального та попередньо визначеного пошуку даних.

З огляду на запит користувача, відповідні запити створюють та виконують у сховищі даних. Результати цих запитів передають користувачеві або іншим агентам.

Агент із бази даних може використовувати декілька джерел під час генерації відповіді на запит. Це можуть бути такі: SQL-бази даних; NoSQL-бази даних; текстові файли різних форматів (XML, JSON); зовнішні сервіси даних тощо. Функції агента із бази даних – це взаємодія між агентами, інтерфейс для баз даних, підтримка локальної та глобальної схеми та форматування результатів запитів, залежно від потреб користувачів.

5.2.3. Інтеграція CBR у СПУКР

Для того щоб поєднати агентів СПУКР та агентів CBR, потрібно під'єднати пошукового агента до агента з ухвалення рішень. Архітектуру системи показано на рис. 5.4, а діаграму послідовності на рис. 5.5.

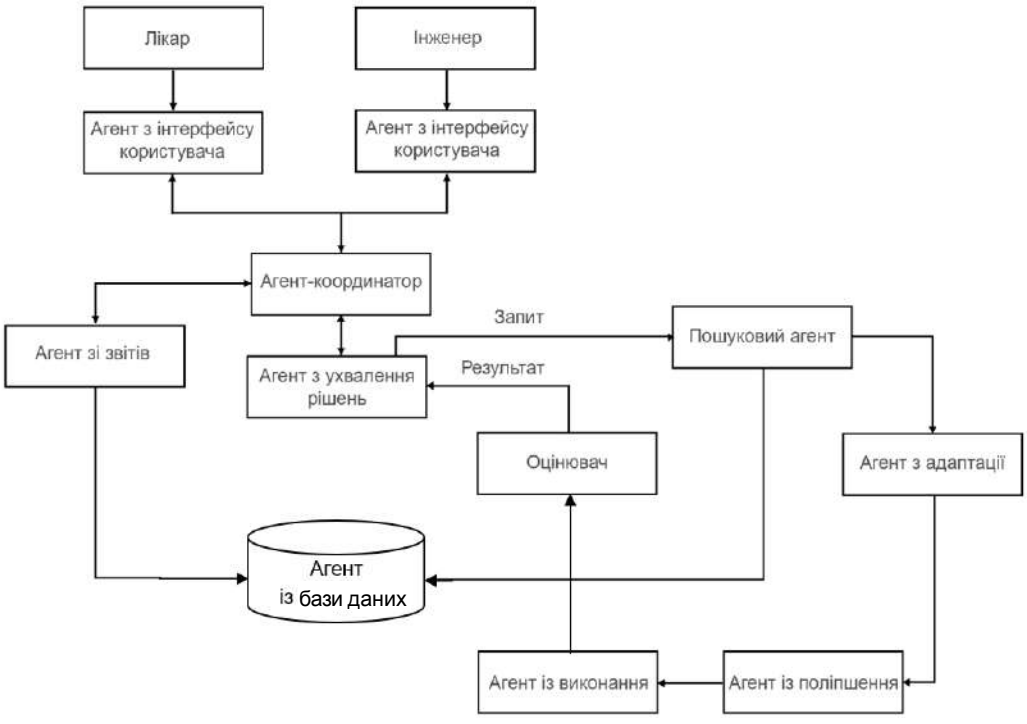


Рис. 5.4. **Архітектура мультиагентної СПУКР із використанням CBR**

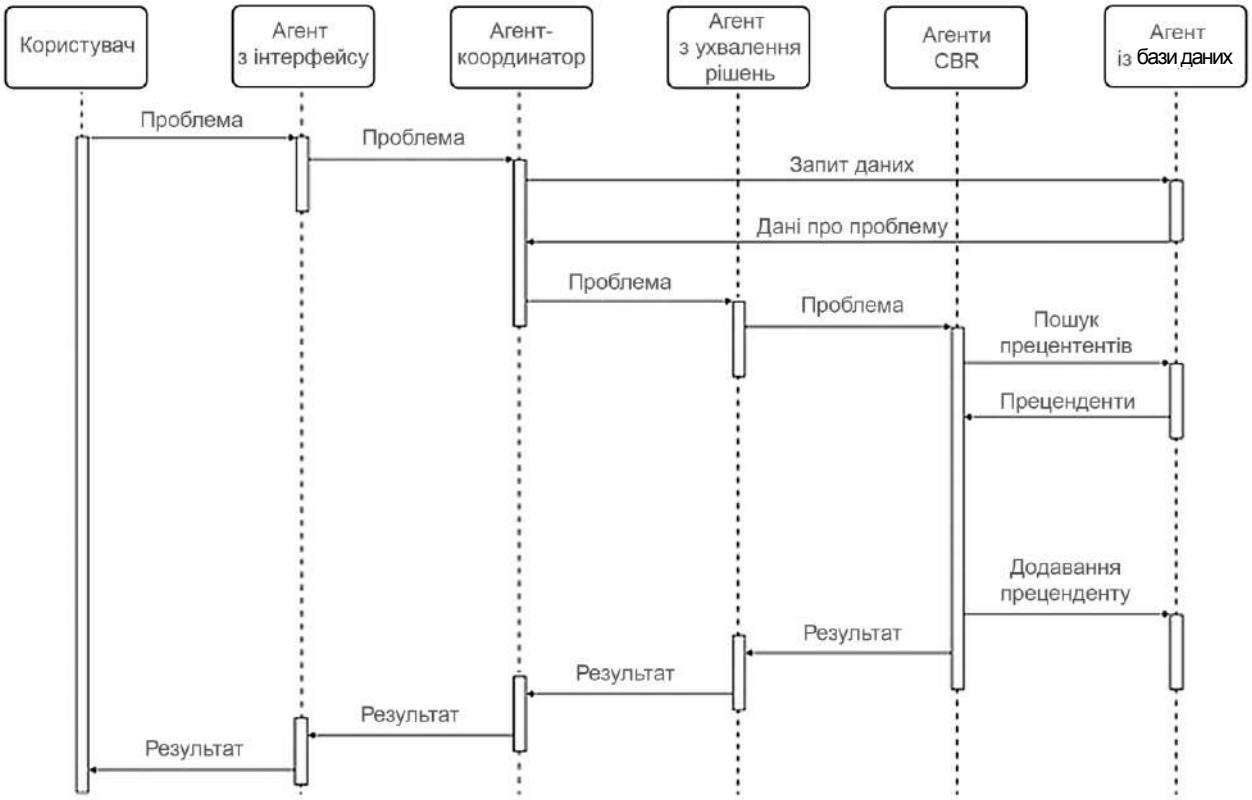


Рис. 5.5. **Загальна діаграма послідовності ухвалення рішень**

Після того як користувач надіслав проблему до системи підтримки ухвалення рішень, агент-координатор бере проблему від клієнта через агента з інтерфейсу користувача та передає її до агента з ухвалення рішень. Зі свого боку, агент з ухвалення рішень інтерпретує проблему та надсилає її до системи CBR. У самій системі CBR заявка спочатку потрапляє до пошукового агента, який виконує функції з пошуку випадків найбільш схожих до проблеми.

Пошуковий агент передає результат до агента з адаптації, який визначає різницю між вибраними випадками та проблемою, і якщо необхідно, застосовує набір необхідних правил, для того щоб старе рішення якнайкраще підходило до нової проблеми.

Після цього агент із поліпшення критикує адаптоване рішення проти попередніх результатів і після того, як рішення критикують, агент із виконання застосовує вишукане рішення до поточної проблеми. У кінці оцінювач зберігає результат до бази прецедентів для подальшого використання

та відправляє результат до агента з ухвалення рішень, який, своєю чергою, через агента-координатора повертає результат до користувача.

5.3. Програмна реалізація системи

Було розроблено прототип програмної системи. Спочатку необхідно визначити стандарт взаємодії агентів у системі. Є різні стандарти взаємодії мультиагентних систем [4]: OMG (Object Management Group), KAoS (Knowledge-able Agent-oriented System), FIPA (Foundation for Intelligent Physical Agents) та ін.

Модель OMG окреслює характеристики агентського середовища, що складається з агентів та агентств як суб'єктів, які співпрацюють за допомогою загальної моделі та політики взаємодії. Агенти в цій моделі характеризуються своїми можливостями, їхньою взаємодією й мобільністю, тоді як агентство визначено для підтримання паралельного виконання агентів, їхньої безпеки виконання та мобільності.

KAoS – це відкрита розподілена архітектура, що описує реалізацію агента, починаючи від поняття простого загального агента до більш конкретних рольових агентів, як-от посередницький агент, та розробляє інтерактивну динаміку обміну повідомленнями "агент – агент" за допомогою політики розмови.

Підхід *FIPA* ґрунтується на основі мінімального фреймворка для управління агентами у відкритому середовищі. Цей фреймворк складається з моделі та платформи агентів. Модель визначає нормативне середовище, у якому є агенти та діють, тоді як агентна платформа визначає інфраструктуру для розгортання та взаємодії агентів. Саме цю специфікацію було вибрано як основу для побудови мультиагентної системи.

Управління агентами передбачає використання нормативного фреймворка, усередині якого є FIPA-агенти та взаємодіють. Він визначає логічні моделі для створення, реєстрації, визначення місцезнаходження, комунікації, міграції та видалення агентів. Сутності, що визначають у цій моделі – це не що інше, як набір якихось можливостей (служб), які зовсім не пов'язані з фізичним конфігурацією.

Модель управління агентами складається з таких логічних компонентів, наведених на рис. 5.6, кожна з яких становить набір певних можливостей.

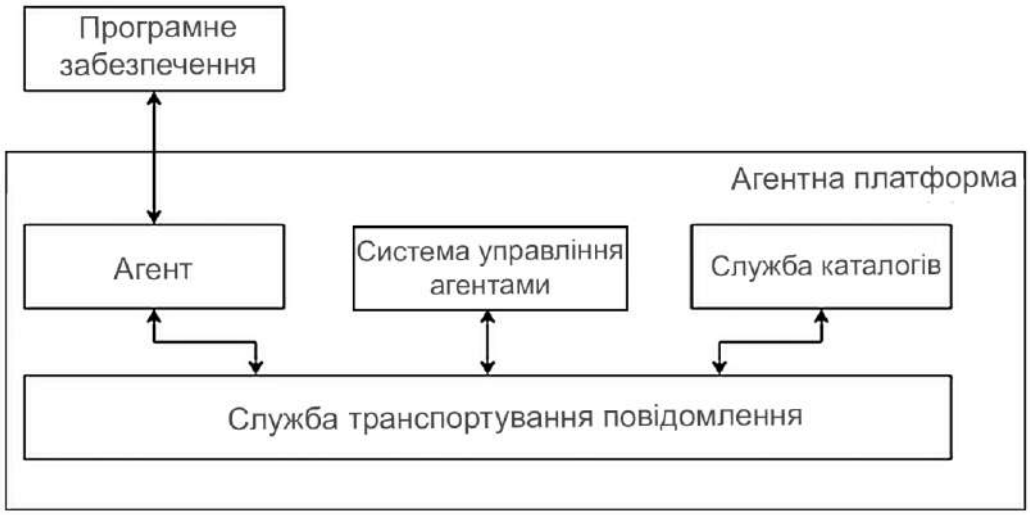


Рис. 5.6. **Модель управління агентами**

Агентна платформа становить фізичну інфраструктуру, у якій може бути розгорнуто агентів, і вона складається з апаратних засобів, операційної системи, програмного забезпечення підтримання агентів, компонентів управління FIPA-агентами (служби каталогу, системи управління агентами та служби передавання повідомлень) і самих агентів.

Необхідно зазначити, що концепція агентної платформи не означає, що агенти мають "розташовуватися" на одному комп'ютері. FIPA передбачає різні варіанти агентних платформ – від простих процесів, що містять "легкі" агентні потоки, до повністю розподілених платформ, побудованих на власних або відкритих міжплатформених стандартах. Для нашої системи цілком достатньо наявності однієї агентної платформи з агентами у процесорних потоках. Крім того, FIPA розглядають тільки з погляду комунікацій між агентами як всередині, так і ззовні для агентної платформи. Агенти вільно можуть обмінюватися повідомленнями будь-якими способами підтримання.

Агент – це обчислювальний процес, який наділений автономністю й комунікативною функціональністю в межах програми та становить комбінацію однієї або безлічі різних сервісних можливостей і операцій усередині об'єднаної й інтегрованої моделі, що виконують.

Служба каталогів – це опціональний компонент агентної платформи, реалізований як спеціальна служба-куратор. Служба каталогу є "жовтими сторінками" можливостей інших агентів. Агенти можуть реєструвати свої можливості за допомогою служби каталогу або запитувати список доступних для виконання дій агентів. Усередині агентної платформи можуть бути множинні служби каталогу і вони можуть об'єднуватися у федерації.

Система управління агентами – це обов'язковий компонент агентного застосунку, який здійснює безпосередній контроль за існуванням і використанням агентної платформи й обслуговує каталог ідентифікаторів агентів, які містять транспортні адреси для зареєстрованих в агентній платформі агентів. Вона становить "білі сторінки" можливостей для інших агентів. Кожен агент має реєструватися в системі управління агентами для здобуття дійсного ідентифікатора.

Служба транспортування повідомлень – це стандартний метод комунікації між агентами агентної платформи. Структуру транспортування повідомлення засновано на стандарті FIPA, що становить формат повідомлення та протокол опрацювання повідомлень для підтримання загальних знань агентів, який може бути подано як складову трьох таких рівнів:

рівня комунікацій, який описує нижній рівень параметрів комунікації, як-от відправник, приймач і комунікаційний ідентифікатор;

рівня повідомлень, який становить протокол взаємодії;

рівня вмісту.

Програмне забезпечення описує не пов'язані з агентами набори виконуваних інструкцій, доступні через агента. Агенти можуть діставати доступ до програмного забезпечення, наприклад, для додавання нових можливостей, запиту нових комунікаційних протоколів, нових протоколів та алгоритмів шифрування, нових протоколів погодження.

5.4. Експериментальні дослідження

Визначення діагнозу захворювання серця є одним із найскладніших завдань, у процесі вирішення якого враховують різні критерії й умови, адже вчасно діагностоване захворювання підвищує шанс на успішне лікування. Завдання щодо підтримання визначення можливих серцевих захворювань потребує ефективно організованої інформації, що відображає досвід і знання фахівців (експертів) у базах знань, прогнозування можливих варіантів рішень, їхнього аналізу та оцінювання, що дозволяє особі, яка ухвалює рішення, більш обґрунтовано вибирати один із можливих діагнозів. Під час визначення діагнозу специфіка баз діагнозів полягає в тому, що вони містять різнорідні знання предметної галузі, знання безлічі різних параметрів (віку, статі, типу болю, кров'яного тиску, кількості цукру у крові, результату електрокардіограми, максимального пульсу тощо), які впливають на результат ухваленого рішення.

Для тестової системи діагнозу захворювання серця використовували набір даних, який налічує понад 300 записів і має 14 атрибутів:

1. Age: вік у роках.

2. Sex: стать (1 = чоловіча; 0 = жіноча).

3. SP: тип болю у грудях:

a) значення 1: типова стенокардія;

b) значення 2: атипова стенокардія;

c) значення 3: неангінальний біль;

d) значення 4: безсимптомно.

4. Trestbps: кров'яний тиск у спокої (у мм рт. ст. під час надходження до лікарні).

5. Chol: холестерин сироватки у мг/дл.

6. Fbs: (цукор у крові > 120 мг/дл):

a) 1 = так;

b) 0 = ні.

7. Restecg: (електрокардіографічний результат):

a) значення 0: нормальне;

b) значення 1: наявність аномалії хвиль ST-T (інверсії T-хвиль та/або підвищення ST, або депресія > 0,05 мВ);

c) значення 2: показ вірогідної або певної гіпертрофії лівого шлуночка за критеріями Естеса.

8. Thalach: досягнута максимальна частота серцевих скорочень.

9. Exang: стенокардія, індукована фізичними вправами:

a) 1 = так;

b) 0 = немає.

10. Oldpeak: ST депресія, викликана вправою щодо спокою.

11. Slope: нахил піку тренування сегмента ST:

a) значення 1: підйом;

b) значення 2: пряма;

c) значення 3: спад.

12. CA: кількість великих судин, пофарбованих флюороскопією.

13. Thal: сканування серця талієм:

a) 3 – нормальний (без холодних плям);

b) 6 – фіксований дефект (холодні плями під час відпочинку та фізичних вправ);

c) 7 – зворотний дефект (коли холодні плями з'являються тільки під час тренування).

14. Pred_attribute: діагностика хвороби серця (стан агіографічної хвороби):

a) значення 0: < 50 % звуження діаметра;

b) значення 1: > 50 % звуження діаметра.

Для початку роботи системи необхідно відправити запит до агента з інтерфейсу. Після чого система починає свою роботу, результатом якої буде вибір чотирьох найбільш відповідних прецедентів у порядку збільшення віддаленості від вихідного. Для ОУР було надано чотири можливі варіанти розв'язання проблеми, серед яких є прецедент із повною схожістю. Тож на основі здобутих даних можна зробити висновок, що хоч і є прецедент, який схожий на початковий із захворюванням серця, але на основі роботи СПУКР, шанс на діагностування захворювання в людини зі вказаними показниками є низьким.

Розподіл часу між агентами на ухвалення рішень показано на рис. 5.7. Для ухвалення одного рішення системі необхідно близько 170 мілісекунд, що є гарним показником, але може бути навіть поліпшеним, якщо до системи додати додаткових агентів з ухвалення рішень, що дозволить горизонтально масштабувати визначену систему.

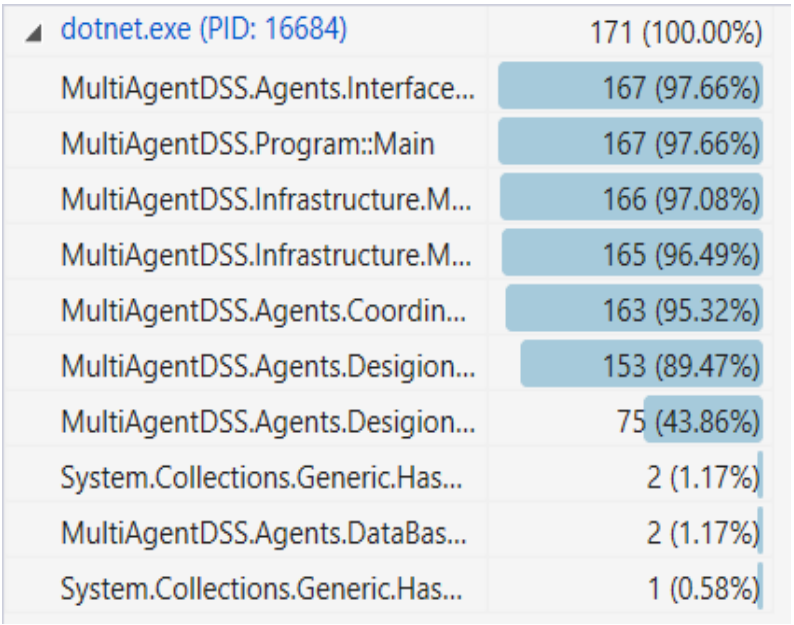


Рис. 5.7. **Розподіл часу між агентами на ухвалення рішень**

Аналізуючи здобуті дані розподілу часу на ухвалення рішень, можна зробити висновок, що агент з ухвалення рішення займає близько 40 % від усіх обчислень у системі. А якщо зазначити, що до недоліків роботи

системи CBR можна зарахувати збільшення часу пошуку найближчих прецедентів, то необхідно здійснити порівняльний аналіз часу пошуку, залежно від розміру бази прецедентів.

Результат тестування продуктивності системи, залежно від кількості прецедентів у базі, показано на рис. 5.8.

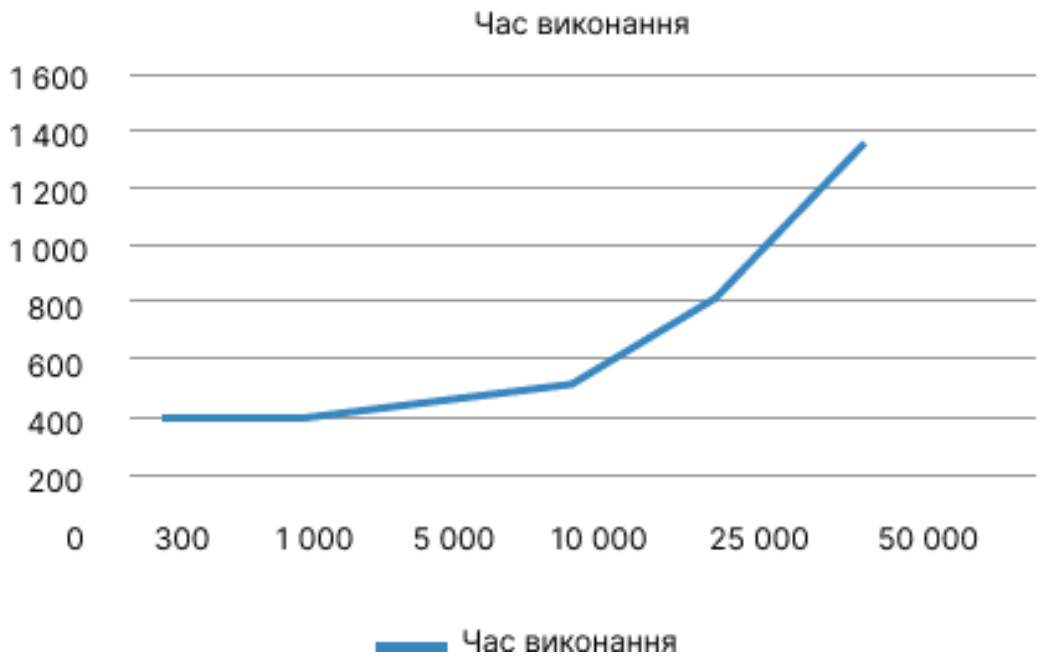


Рис. 5.8. **Графік залежності часу на ухвалення рішення від кількості прецедентів**

У результаті тестування можна сказати, що хоч зі збільшенням кількості прецедентів у базі час виконання дійсно збільшується, але це відбувається плавно й незначно впливає на загальний час роботи. Незважаючи на зазначені недоліки, експеримент показав гарну спроможність системи ухвалювати рішення, так що її може бути використано для ухвалення рішень у реальний умовах.

5.5. Висновки

Запропоновано систему підтримки ухвалення клінічних рішень (СПУКР), призначену для поліпшення надання медичної допомоги шляхом модернізації медичних рішень за допомогою цілеспрямованих клінічних знань, інформації про пацієнтів та іншої медичної інформації. СПУКР

основано на використанні мультиагентного підходу та міркувань на основі прецедентів (CBR), і вона складається з модулів інтерфейсу, виконання та знань, які містять різних агентів. В основі мультиагентного підходу лежить поняття інтелектуального агента, який перебуває в середовищі та функціонує як самостійна комп'ютерна програма, а також здатен на взаємодію з ним для досягнення своїх цілей. Агенти становлять складні програмні системи, які мають здатності до підлаштування під конкретну обставину, що викликає зміну їхньої поведінки або характеристик для забезпечення здатності до адаптації та розв'язання поставленої перед ними проблеми. Мультиагентну технологію використовують, із метою здобуття, повторного використання та адаптації прецедентів у системі CBR.

Міркування на основі прецедентів є підходом, що дозволяє вирішити нове завдання, використовуючи або адаптуючи рішення вже відомого завдання, тобто використовуючи вже накопичений досвід вирішення подібних завдань. Основна мета використання апарату прецедентів у межах СПУКР полягає у видачі готового рішення оператору (ОУР) для поточної ситуації на основі прецедентів, які вже мали місце в минулому під час управління даними або подібним об'єктом (системою). Система CBR вилучає з бази прецедентів випадки, пов'язані з наявним захворюванням, та вирішує питання визначення діагнозу захворювання на підставі результатів попередніх даних про таке захворювання.

Проведені експериментальні дослідження запропонованої СПУКР щодо підтримання визначення можливих серцевих захворювань. Для тестової системи діагнозу захворювання серця використовували набір даних, який налічує понад 300 записів і має 14 атрибутів. Здобуті дані розподілу часу на ухвалення рішень показали, шо агент з ухвалення рішення займає близько 40 % від всіх обчислень у системі. Результати тестування продуктивності системи, залежно від кількості прецедентів, показали, що зі збільшенням кількості прецедентів у базі час виконання збільшується, але це відбувається повільно й незначно впливає на загальний час роботи. Експериментальні дослідження показали спроможність запропонованої СПУКР і можливість її використання для ухвалення рішень у реальних умовах.

Розділ 6
Особливості застосування технологій управління та моніторингу поточного стану цифрових комп'ютерних мереж

6.1. Вступ і формулювання завдання

Нині на основі ухвалених базових концепцій управління комп'ютерними (телекомунікаційними) мережами застосовують прості та ефективні протоколи управління IP-мережами. Концепція OSI передбачає для управління мережами створення мережевих служб NMS (ISO 7498-4, 9595, 9596), які забезпечують оптимальне функціонування мереж, планування, управління та контроль за роботою всіх їхніх компонентів, водночас службу управління мережею утворено сукупністю розподілених у комп'ютерній мережі апаратних і програмних засобів та інформаційних ресурсів, розміщених у всіх її елементах. Елементами мережі є устаткування користувачів (персональні комп'ютери, IP-телефони, факси) комутатори, модеми, сервери, маршрутизатори та інші компоненти.

Модель TMN (Telecommunication Management Networks) подано в рекомендаціях ITU-TM.3010 M.3020, M.3100 й дозволяє будувати систему управління неоднорідною комп'ютерною (телекомунікаційною) мережею, побудованою з використанням різних технологій, устаткування та програмного забезпечення. Ефективне вирішення завдань управління сучасними комп'ютерними мережами можливе тільки тоді, коли чітко подано (синтезовано) детальну адекватну модель мережі на принципах GII [8].

Основне завдання управління телекомунікаційними (комп'ютерними) мережами полягає в реалізації цілеспрямованого впливу (моніторингу, контролю, адміністрування) на устаткування комп'ютерної мережі за допомогою технічних і програмних засобів. Системи управління IP-мережами складаються із програмних, програмно-апаратних засобів, оперативного й адміністративного персоналу, які забезпечують управління та належать до класу систем автоматизованого управління.

Система управління телекомунікаційною мережею має здійснювати аналіз продуктивності та надійності мережі, тобто виконувати завдання, пов'язані з оцінюванням таких параметрів, як час реакції системи, пропускна спроможність реального або віртуального каналу зв'язку,

інтенсивність трафіка в окремих сегментах і каналах мережі, імовірність спотворення даних під час їхнього передавання через мережу, а також коефіцієнт готовності мережі.

Система управління мережею має глобальну мету – забезпечення заданого інформаційного обміну з визначеною якістю обслуговування. Система управління мережею має забезпечувати оперативне й безперервне управління устаткуванням, оптимальне використання ресурсів мережі, автоматичну реєстрацію подій та оперативну зміну конфігурації мережі в масштабі часу, близькому до реального, тощо. Прикладами засобів управління системою є такі продукти, як System Management Server компанії Microsoft або LAN Desk Manager фірми Intel.

Технічною складовою системи управління є підсистема моніторингу, яка виконує завдання збирання, опрацювання, зберігання та відображення повної інформації про стан усіх компонентів комп'ютерної мережі в реальному часі, незалежно від її архітектури й типу устаткування. Система моніторингу має надавати набір рішень, що забезпечують автоматичний моніторинг мереж, реалізованих на базі різних технологій (мережі передавання даних, телефонні мережі, сигналізації тощо) та убудованих на устаткуванні різних виробників. Урахування наведених раніше *факторів* приводить до необхідності в удосконаленні принципів і підходів до мережевого моніторингу й управління [7].

Необхідність у своєчасному прогнозуванні, запобіганні поломкам і перевантаженням, оповіщенні про них, зберіганні інформації про стан устаткування в комп'ютерній мережі та оперативному усуненні виявлених несправностей підтверджує актуальність цієї роботи.

6.2. Основна частина

Для вирішення завдань моніторингу й аналізу функціонування комп'ютерних мереж є велика кількість різного програмного забезпечення, програмних аналізаторів, вимірювальних платформ та ін. від різних виробників. Кожен із цих програмних (програмно-апаратних) засобів призначено для вирішення схожих завдань, проте кожен із них має свої відмінності.

Об'єктами управління в комп'ютерних мережах є такі: локальні мережі; устаткування систем передавання (мультиплексори, крос-конектори, каналоутворювальне устаткування, комутатори, маршрутизатори тощо); операційні системи програмно-керованого устаткування зв'язку; лінії зв'язку (мідні й оптичні кабельні лінії, радіолінії, радіорелейні, супутникові);

системи сигналізації й синхронізації (устаткування та програмне забезпечення); термінали (хости) користувачів, апаратне та програмне забезпечення, необхідне для надання телекомунікаційних послуг тощо.

Розгляньмо варіанти застосування, особливості функціонування, переваги та недоліки найбільш уживаних програмних, програмно-апаратних засобів і мережевих аналізаторів.

1. **Програмне забезпечення PingInfoView** є невеликою утилітою за допомогою якої можна пінгувати вузли за доменними іменами або IP-адресами з можливістю встановлення кількості й інтервалів опитувань. Вона є аналогом стандартної консольної програми ping.exe. Для моніторингу мережі відправляють ICMP-запити, та зворотно присилають ICMP-відповіді, що устаткування є доступним і підключеним у єдину IP-мережу. Результати пінгування можна зберегти в текстовий, HTML- і XML-файл або скопіювати його в буфер обміну. Розробником цієї вільної (безкоштовної) для використання програми є NirSofer. Робоче вікно програми PingInfoView показано на рис. 6.1.

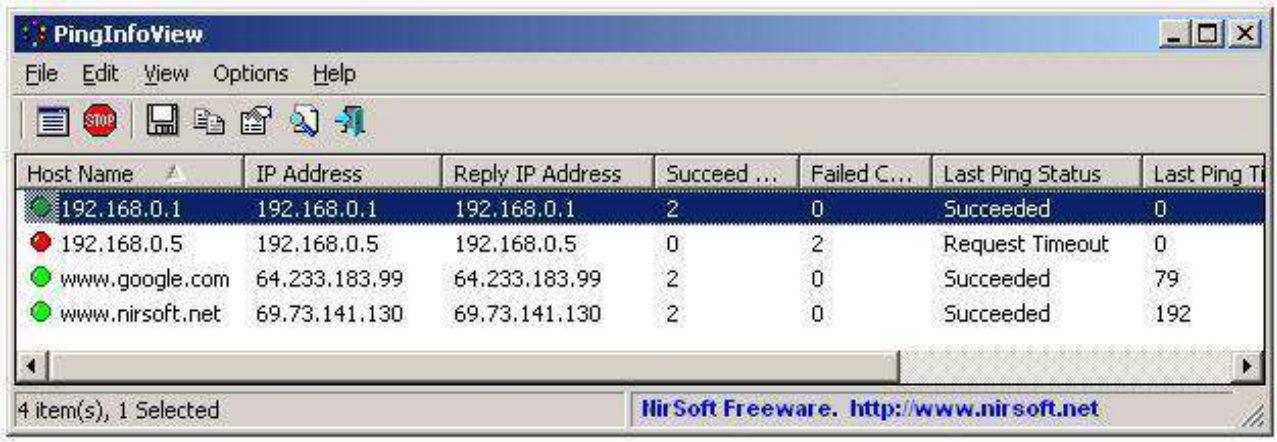


Рис. 6.1. **Робоче вікно програми PingInfoView**

Можливості системи моніторингу PingInfoView такі:

дозволяє перевіряти доступність вузлів мережі в режимі реального часу та працює на основі команди ping протоколу ICMP;

надсилає ping на задану кількість IP-адрес вузлів із заданим інтервалом;

відображає кількість успішних та невдалих перевірок, відсоток утрат, середнє й останнє значення мережевої затримки, часові межі доступності та недоступності вузла комп'ютерної мережі, кількість транзитних вузлів.

Звичайно, такий спосіб є простим, надійним і його часто застосовують ті, хто займаються налагодженням цифрового устаткування IP-мереж, та мережеві адміністратори, але якщо потрібно стежити за значеннями конкретних параметрів мережевого устаткування, відповідністю параметрів і помилок необхідно використовувати більш складні та надійні системи контролю.

2. **Убудовані системи діагностики й управління (Embedded systems) –** система управління та моніторингу устаткування локальних IP-мереж System Center Operations Manager (SCOM), яка забезпечує збирання даних, їхній аналіз та зберігання на сервері баз даних Microsoft SQL Server. Ці системи виконують у вигляді програмно-апаратних модулів, убудованих в операційні системи Windows, Unix та Linux, та встановлюють у комунікаційне устаткування.

Одним із найбільш зручних, надійних та простих засобів, що дозволяє виконувати більшість завдань управління мережевими пристроями, є протокол для управління пристроями в IP-мережах Simple Network Management Protocol (SNMP).

Основною концепцією протоколу є те, що всю необхідну для управління пристроєм інформацію зберігають на самому пристрої – будь-то сервер, модем або маршрутизатор – у так званій адміністративній базі даних. Ці змінні можуть відображати такі параметри, як кількість пакетів опрацьованих пристроєм, стан його інтерфейсів, час та особливості функціонування пристрою тощо.

Уніфікований протокол мережевого моніторингу SNMP версії 3 (SNMP v3), завдяки сучасній модульній архітектурі, удосконаленим протоколам управління, включно із протоколом прикладного рівня, схемою баз даних і набором об'єктів даних, можливістю шифрувати трафік, підтримуванням SNMP версії 1 і 2 та поліпшеному віддаленому налаштуванню, широко застосовують для моніторингу й аналізу мережевого устаткування, для виявлення й розв'язання багатьох мережевих проблем. SNMP визначено Інженерною радою інтернету (IETF) як компонент стека TCP/IP [32].

Для запуску SNMP менеджер мережі має встановити програмне забезпечення Network Manager на сервері. **Network Manager** – це програмне забезпечення з відкритим кодом, як-от Solar winds та Cisco IOS. Основним завданням менеджера простого управління мережевим протоколом є запит та отримання даних від агента для управління та моніторингу

елементів мережі. Крім того, є можливість редагування конфігурації, коли це потрібно, відповідно до вимог мережі.

Усю необхідну інформацію протокол SNMP дістає з бази керівної інформації Management Information Base (MIB). MIB становить базу даних стандартизованої структури. База даних має деревоподібну структуру, а всі змінні класифіковано за тематикою. Кожне піддерево містить певну тематичну підгрупу змінних. Найбільш важливі компоненти, що відповідають за роботу мережевих вузлів, об'єднано в підгрупі MIB-II.

Застосовують два типи MIB: стандартні та фірмові. Стандартні MIB визначено комісією з діяльності Internet Activity Board (IAB), а фірмові задає виробник пристрою.

У табл. 6.1 наведено перелік найбільш поширеніших стандартів баз керівної інформації. У базах даних, зазначених у табл. 6.1, є багато змінних, які можуть бути корисними для діагностування комп'ютерної мережі та мережевого устаткування.

Таблиця 6.1

Перелік основних стандартів баз керівної інформації

Бази	Призначення
MIB-II	Задає велику кількість об'єктів, які може бути використано для управління мережевими інтерфейсами
MIB повторювача	Уміщено до підмножини MIB-II. Установлює об'єкти, які можна використовувати для управління повторювачем
MIB мосту	Уміщено до підмножини MIB-II. Задає дані, які можна використовувати для управління мостом
RMON MIB	Указує об'єкти даних, які можна використовувати для управління мережею загалом за допомогою протоколу Remote Network MONitoring (RMON)

Наприклад, використовуючи MIB-II, можна дістати відомості про загальну кількість пакетів, переданих мережевим інтерфейсом, а за допомогою MIB повторювача можна здобути інформацію про кількість колізій порту [32].

У MIB кожен об'єкт має ім'я та тип. Ім'я об'єкта характеризує його становище в дереві MIB. Водночас ім'я дочірнього вузла містить ім'я батьківського вузла та його задано цілим числом. Є три компоненти

SNMP, за допомогою яких він виконує свої основні завдання, показні на рис. 6.2:

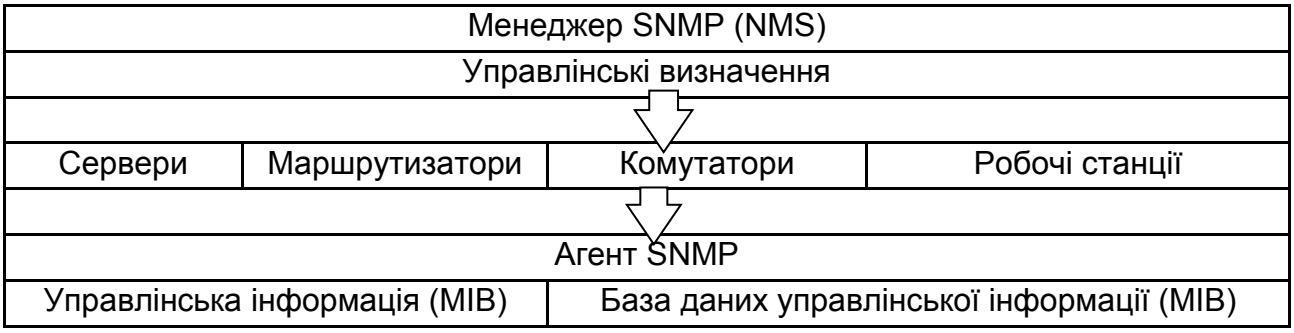


Рис. 6.2. **Структура SNMP**

1) *менеджер SNMP.* Це централізована система вузлів на основі графічного інтерфейсу, яку використовують для моніторингу мережі, її також називають системою управління мережею Network Management System (NMS). Він взаємодіє із двоспрямованим потоком інформації між вузлом NMS та елементами мережі. Елементами мережі можуть бути комутатори, комп'ютери (хости), маршрутизатори, сервери, IP-телефони, модеми, IP-відеокамери тощо;

2) *агент SNMP.* Агент – це модуль програмного забезпечення для управління мережею, який установлюють на мережевому пристрої, персональний комп'ютер, сервер, маршрутизатор тощо. Агент підтримує базу даних про елементи мережі, якими дистанційно можна управляти. Коли NMS запитує будь-яку інформацію, вона повертається разом із даними, які зберігали в базі даних, до NMS. Якщо будь-яку пастку або помилку зустрічає агент на керованому пристрої, він надсилає повідомлення про пастку менеджеру SNMP із зазначенням його стану в реальному часі;

3) *база даних управлінської інформації (MIB).* Кожен з агентів SNMP підтримує інформаційну базу даних для керованих пристроїв, яка пояснює параметри пристроїв. Менеджер SNMP використовує цю базу даних, щоб запитати в агента інформацію про конкретний пристрій для NMS. Отже, ця спільна інформація між агентом та менеджером відома як база даних MIB.

У процесі розроблення SNMP v3 достатньо уваги було приділено безпеці протоколу. Тепер стали підтримувати модель, орієнтовану на користувача User-Based Security Model (USM), завдяки якій стало можливим додавання модулів автентифікації та шифрування без зміни базової архітектури.

Однак у системи управління пристроями в IP-мережах SNMP є ряд недоліків, а саме: система моніторингу охоплює багато загальних показників системи, але не є придатною для стеження за визначеними параметрами устаткування IP-мереж; специфічність та труднощі під час налаштування.

3. **Програмне забезпечення Zabbix** – гнучка система на вільному програмному забезпеченні (opensource), яке використовують для комплексного моніторингу комп'ютерних мереж і відстеження стану різноманітних сервісів устаткування, серверів та устаткування мереж.

Платформу розробляє компанія Zabbixsia (Рига, Латвія). Програмне забезпечення Zabbix дозволяє здійснювати автоматичне виявлення устаткувань, перевірку доступності та продуктивності, аудит і аналіз функціонування, повідомлення й оповіщення в разі виникнення перебоїв у мережі [37].

Складається із сервера моніторингу (Zabbix-сервера), що містить базу даних, які збирають та аналізують, вебінтерфейсу управління і візуалізації, й агентів збирання даних на об'єктах мережі, за якими ведуть спостереження (об'єктах моніторингу). Також до його складу може входити проксі. Усі налаштування, журнали, графіки, мапи та звіти є доступними через графічний вебінтерфейс (рис. 6.3).

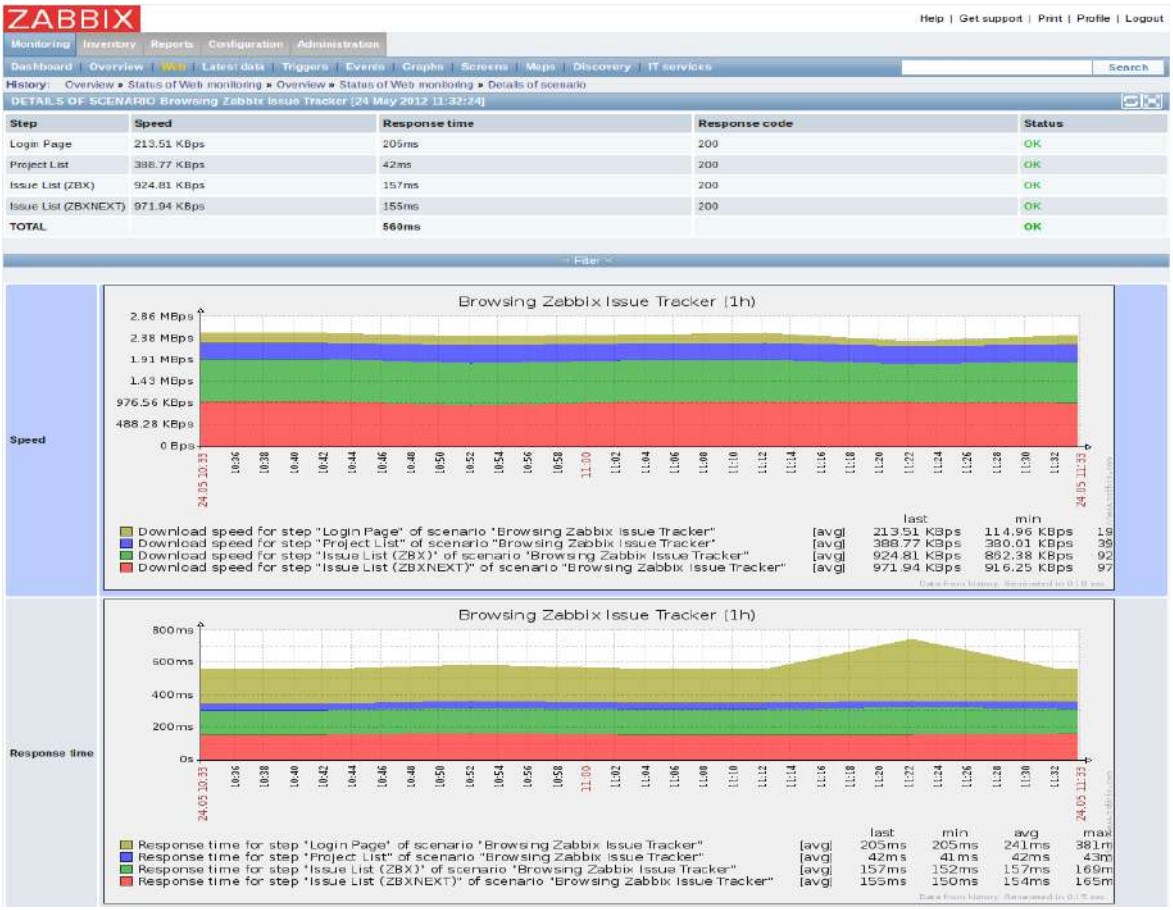


Рис. 6.3. **Робоче вікно програми Zabbix**

Zabbix-проксі (proxy) збирає дані про продуктивність і доступність об'єктів від імені Zabbix-сервера та передає їх на сервер для опрацювання, зберігання й дозволяє перевіряти об'єкти, що перебувають під захистом мережевих екранів, які неможливо відстежувати безпосередньо із сервера моніторингу (Zabbix-сервера). Zabbix-проксі є надійним рішенням для централізованого віддаленого моніторингу місць, філій, мереж, які не мають локальних адміністраторів та автоматичного виявлення за діапазоном IP-адрес.

Zabbix-агент збирає інформацію про стан локальних ресурсів і застосунків, як-от жорсткі диски, пам'ять, статистика процесора та ін., і передає її серверу моніторингу для опрацювання та зберігання. Zabbix-агенти є невидимими для користувача є надзвичайно ефективними через використання власних системних викликів для збирання інформації про фактичний стан устаткування.

Zabbix-сервер – ядро системи моніторингу Zabbix. Сервер може віддалено перевіряти мережеві сервіси, а також зберігає всі конфігураційні, статистичні й оперативні дані. Сервер оповіщає адміністраторів у разі виникнення проблем із будь-яким мережевим устаткуванням. До його складу входять база даних та вебінтерфейс.

Вебінтерфейс – це інтерфейс мовою PHP (гіпертекстовий препроцесор), що працює з базою даних системи моніторингу. Слугує для відображення даних та управління налаштуваннями. Дані можуть зберігатися в системах управління базами даних Mysql, Postgresql, Sqlite та Oracle [37].

Особливістю програмного продукту Zabbix є потреба у значному обсягу місця на жорсткому диску персонального комп'ютера. До його переваг можна зарахувати суттєві можливості щодо збирання даних, налаштування оповіщень, побудови графіків у режимі реального часу, зручної візуалізації та ін.

Система моніторингу Zabbix має специфічні недоліки, як-от громіздкість сервісу, необхідність у встановленні програмного забезпечення на все устаткування мережі, відсутня автоматизована побудова мап мереж та ін.

4. **Система моніторингу на базі Cisco Prime Infrastructure.** Програмне забезпечення Cisco Prime Infrastructure надає управління повним життєвим циклом **проводових** та **безпроводових** комп'ютерних мереж. Вона об'єднує рішення для управління локальною мережею Cisco Prime

LAN Management Solution (LMS) і систему управління мережами Cisco Prime Network Control System (NCS) у єдину систему для спрощення замовлення й управління ліцензіями. Cisco Prime Infrastructure застосовують переважно в корпоративному використанні для управління мережами передавання даних.

Для моніторингу устаткування комп'ютерної мережі система збирає дані SNMPtrap і Syslog із пристроїв, також сама періодично опитує устаткування, генерує аварійні події в разі виходу параметрів за певні граничні значення. Наявні готові шаблони для контролю й моніторингу із заданими визначеними значеннями, а також можна створювати на їхній основі свої власні. Зовнішній вигляд вікна програмного забезпечення показано на рис. 6.4.

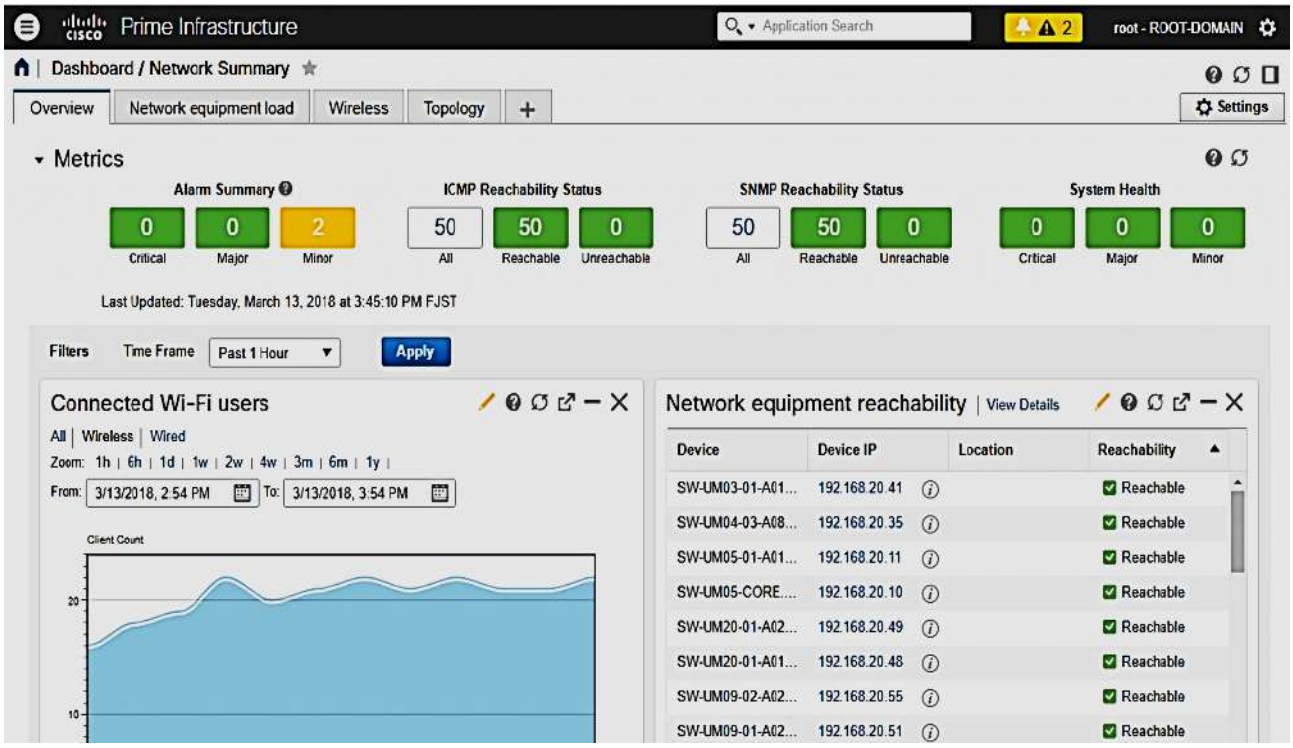


Рис. 6.4. **Робоче вікно програми Cisco Prime Infrastructure**

Основні завдання, які дозволяє виконати Cisco Prime Infrastructure:

підтримання щоденних завдань адміністратора (модуль повного циклу): моніторинг устаткування, резервне копіювання, відновлення, збирання базової статистики, даних інвентаризації, налаштування устаткування (шаблони, готові сценарії та ін.), аудит пристроїв мережі;

контроль за роботою застосунків (модуль Assurance);

автоматичне встановлення устаткування без використання консольного доступу (модуль Plug and Play).

До складу Cisco Prime Infrastructure входять такі програмні компоненти (модулі) як-от Lifecycle, Assurance, Plug and Play.

Lifecycle – це основний модуль, що допомагає адміністраторам у вирішенні таких поточних завдань:

моніторингу інфраструктури, також дає можливість моніторингу й діагностики проблем безпроводової інфраструктури;

гнучкого налаштування фільтрів інцидентів;

базового аудиту мережі (інвентаризації пристроїв);

створення резервних копій (програмного забезпечення мережевих пристроїв, конфігурації);

управління образами програмного забезпечення;

статистики роботи пристроїв (uptime, стан, інциденти тощо);

налаштування устаткування, а також використання шаблонів.

Система моніторингу дозволяє фільтрувати, підтверджувати або видаляти сповіщення, додавати до них різні коментарі. З інтерфейсу управління також є доступними інструменти діагностики устаткування, як-от ping/traceroute, show-commands та ін. Дані про підключене устаткування зберігають у базі даних Cisco Prime Infrastructure та відображають в інтерфейсі за допомогою визначеної структурованої форми. Із цього інтерфейсу є доступними також різні засоби управління. Стандартизовані архіви конфігурацій щодо кожного пристрою застосовують в IP-мережі.

Cisco Prime Infrastructure дозволяє адміністратору мережі стежити за всіма проводовими та безпроводовими пристроями в мережі, каналами передавання даних і надсилати дані в центр обміну даними. Використовуючи визначені дані, адміністратори можуть переглядати характеристики всього устаткування. Крім того, адміністратор може діставати інформацію про один із пристроїв мережі, щоб переглянути навіть складові частини цього пристрою, наприклад, модулі, порти та інтерфейси. Система дозволяє за іменем користувача визначити всі його підключені пристрої та виводити діагностичну інформацію про їхнє підключення. Адміністратор також може переглянути звіт щодо того, наскільки добре працює застосунок, але реальне значення приходить на рівні деталізації. Можливий доступ до відомостей про сеанс, історію, використання застосунків та ідентифікаторів користувача.

Основні переваги використання програми Cisco Prime Infrastructure:

зручний інтерфейс та інтуїтивне налаштування;

візуалізація різної інформації (наприклад, Device 360 View);

можливість адаптувати профіль інтерфейсу (даш-борди);

простота та зручність в експлуатації;

підтримання значного парку пристроїв CiscoSystems. Cisco Prime Infrastructure доступний у вигляді як програмного, так і апаратного рішення.

Недоліком є те, що систему спочатку спрямовано на устаткування CiscoSystems, а вона не має повноцінної мультивендорної підтримки. Підтримання устаткування інших виробників є можливим, але лише за наявності відповідної бази даних управлінської інформації.

5. **Програму "10-Страйк: Схема Сети"** призначено для побудови схем мереж із можливістю сканування топології мережі та визначення всіх підключених у комп'ютерній мережі активних пристроїв.

Програма дозволяє доопрацювати схему за допомогою потужних убудованих засобів редагування, показати зв'язки, нанести написи, окреслити ділянки, залити їх різними кольорами та візерунками. Програма містить велику бібліотеку піктограм мережевих пристроїв.

Робоче вікно програми "10-Страйк: Схема Сети" показано на рис. 6.5.

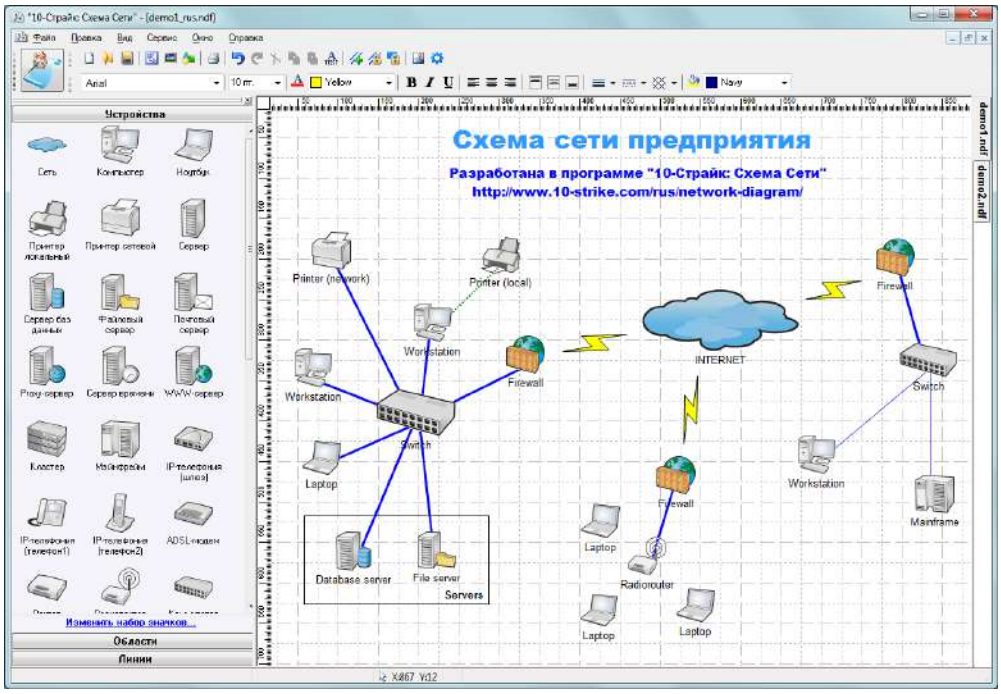


Рис. 6.5. **Робоче вікно програми "10-Страйк: Схема Сети"**

Переваги програми "10-Страйк: Схема Сети" такі:

автоматична побудова зв'язків між вузлами під час здобуття інформації про топологію мережі від керованих комутаторів (за протоколом SNMP);

потужний убудований редактор схем мереж та устаткування;

підтримання сканування структури мережі за протоколом LLDP і трасування маршрутів;

векторне зображення мережевих пристроїв та висока якість зображення визначених схем мереж.

Недоліки програми такі:

використовують тільки для побудови схем комутаційних та кінцевих пристроїв IP-мереж й тільки для Windows;

програмний продукт розповсюджують на платній основі.

6. **Програмне забезпечення FriendlyPinger 5.0.1** – програма для моніторингу, інвентаризації та управління комп'ютерними мережами. Програма дозволяє простежити за доступністю кожного пристрою, переглянути список встановленого програмного забезпечення на віддалених персональних комп'ютерах і нарисувати графічну схему з усіх підключених вузлів. За допомогою інтегрованої системи повідомлень можна дізнатися про зупинку або запуск сервера, виконати пінгування всього устаткування мережі.

Програма FriendlyPinger 5.0.1 дозволяє підтримувати пошук за службами, також передбачено функцію створення дистрибутивів та можливість графічного трасування. На рис. 6.6 показано вікно візуалізації зазначеної програми.

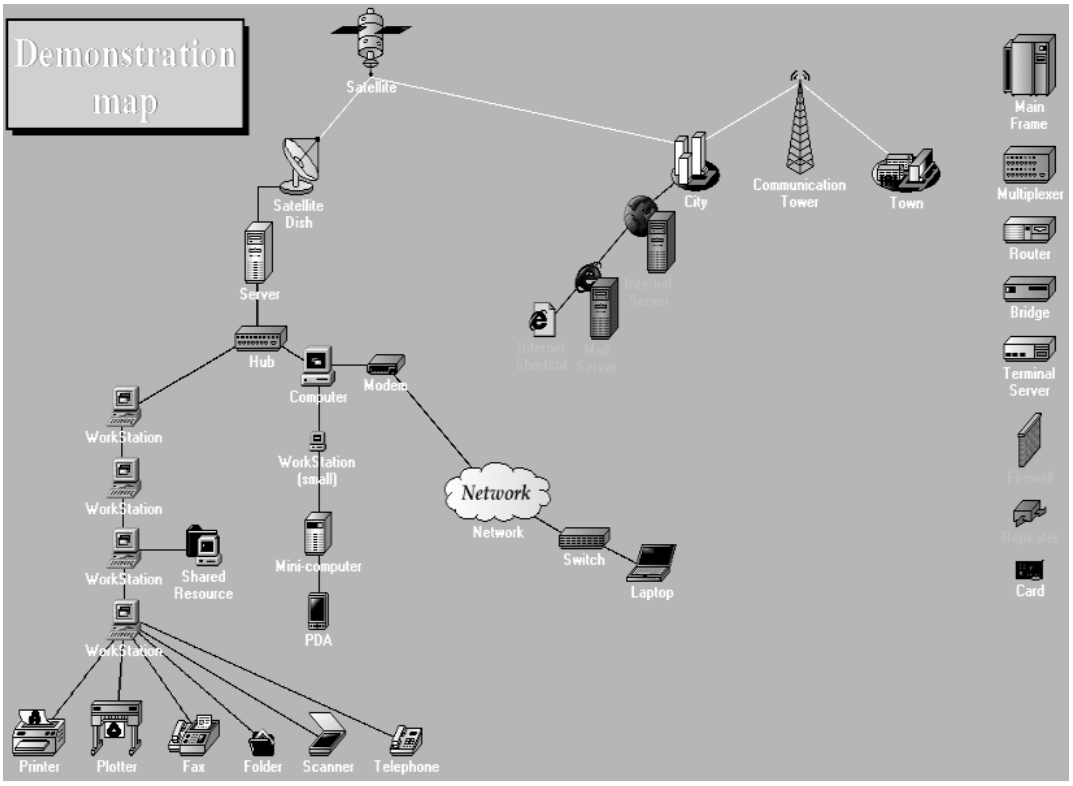


Рис. 6.6. **Робоче вікно програми FriendlyPinger 5.0.1**

Переваги програми FriendlyPinger 5.0.1 такі:

візуалізація мережі в анімаційної формі;

відображення ввімкнених вузлів у режимі реального часу;

виконання команди ping усіх пристроїв за один раз;

інвентаризація й аналіз програмного та апаратного забезпечення всіх комп'ютерів у мережі, графічний TraceRoute;

призначення зовнішніх команд (наприклад, telnet, tracert, net.exe та ін.) для пристроїв на мапі;

пошук HTTP-, FTP-, DNS-, SMTP-серверів та інших мережевих служб і доступність відповідних портів, оповіщення в разі зупинки/запуску серверів;

відображення стану мережі на робочому столі або вебсторінці;

функція "Створити дистрибутив" дозволяє створити полегшену версію із власними мапами та налаштуваннями.

До недоліків програми можна зарахувати тривалу відсутність оновлень та непідтримання протоколу SNMP.

7. **Система моніторингу Cacti** – це програма, яка дозволяє виконувати моніторинг мережі, збирати статистичні дані за певні часові інтервали та відображати їх у графічному вигляді за допомогою RRDtool-утиліти. Її написано в інфраструктурі Apache-PHP-MySql, вона дозволяє налаштовувати збирання та відображення даних моніторингу на основі вебінтерфейсу й заздалегідь складеного набору графіків даних протягом останнього дня, тижня, місяця та року. Також є можливість довільно задавати часовий проміжок, за який буде згенеровано графік, зображений на рис. 6.7.

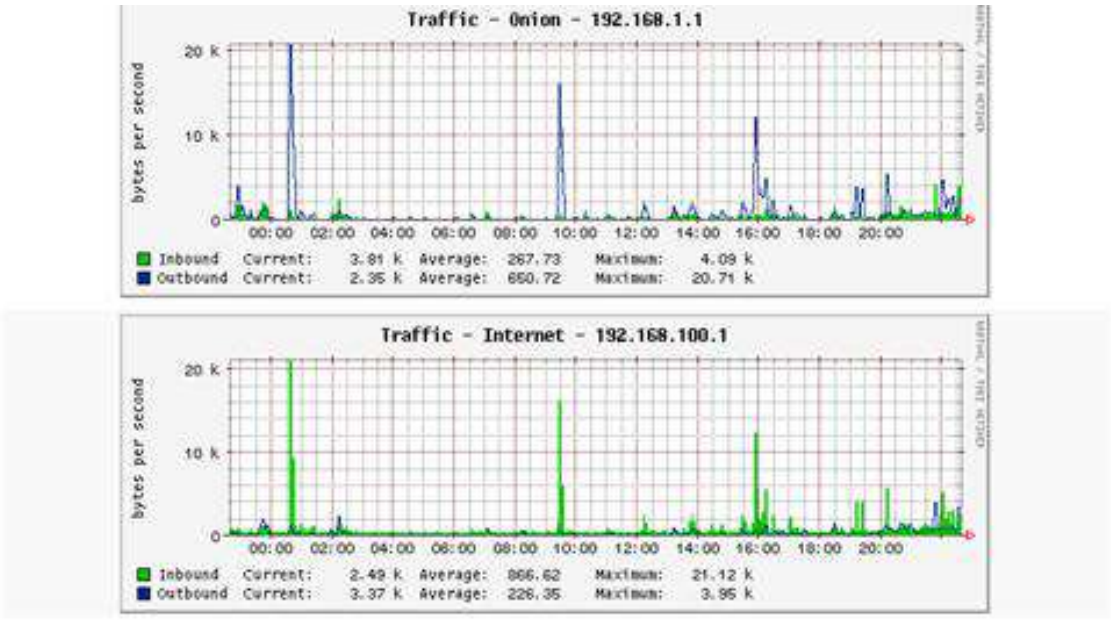


Рис. 6.7. **Графік стану IP-мережі протягом визначеного часу**

Основне використання Cacti – моніторинг мережевого трафіка шляхом опитування мережевого комутатора або інтерфейсу маршрутизатора через протокол SNMP.

До недоліків системи Cacti слід зарахувати досить швидке нарощування кількості однотипних налаштувань у разі великої кількості користувачів і серверів, а також обмежену продуктивність деяких JMX-рішень для Cacti.

8. **Програмне забезпечення 10-Strike LANState (Pro) 8.5r** призначено для адміністрування та моніторингу серверів, комп'ютерів та інших мережевих пристроїв із можливістю спостерігати поточний стан мережі у графічному вигляді, управління серверами й робочими станціями на мапі, моніторингу віддалених пристроїв за допомогою їхнього періодичного опитування. Робоче вікно програми 10-Strike LANState (Pro) 8.5r показано на рис. 6.8.

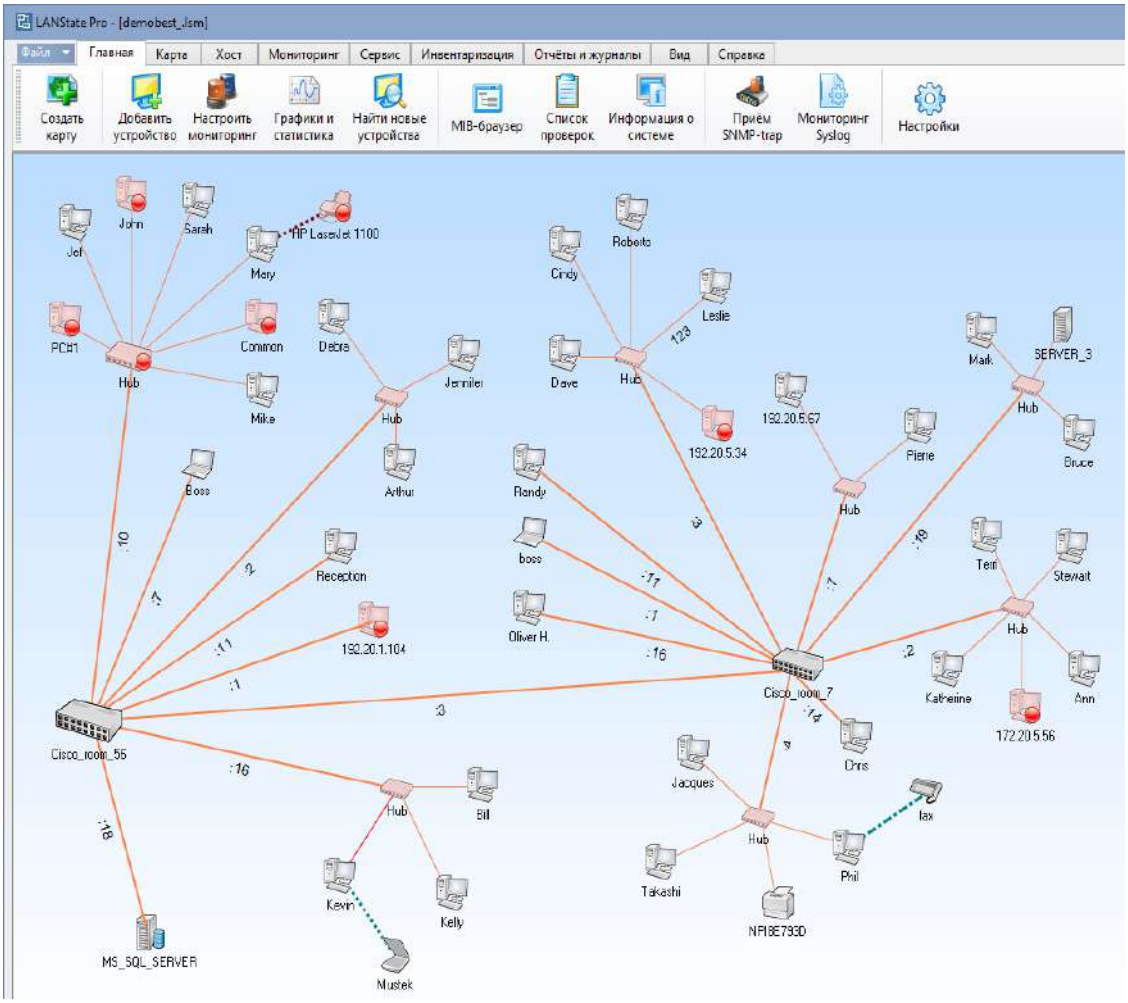


Рис. 6.8. **Робоче вікно програми 10-Strike LANState (Pro) 8.5r**

Особливості та переваги програми 10-Strike LANState (Pro) 8.5r такі:

побудова графічної мапи мережі й наочний перегляд стану комп'ютерів (працює, не працює), збереження та друкування мапи;

підтримання можливості роботи з декількома мапами одночасно;

вимикання, перезавантаження, вмикання віддалених серверів і робочих станцій, програвання звуку, відправлення e-mail або SMS;

перегляд різноманітної інформації про мережу (установлені пристрої та служби, eventlog, реєстр, список запущених процесів);

відправлення повідомлень користувачам домену за допомогою зручного вбудованого месенджера;

відсутня необхідність в установленні будь-яких компонентів на віддалені сервери та робочі станції.

До недоліків програми можна зарахувати те, що її було розроблено в Російській Федерації, тому вона має обмежену сферу застосування.

9. **Програмне забезпечення Algorius Net Viewer** – інструмент для візуалізації, адміністрування, моніторингу й інвентаризації комп'ютерної мережі будь-якого рівня. Робоче вікно програми Algorius Net Viewer показано на рис. 6.9.

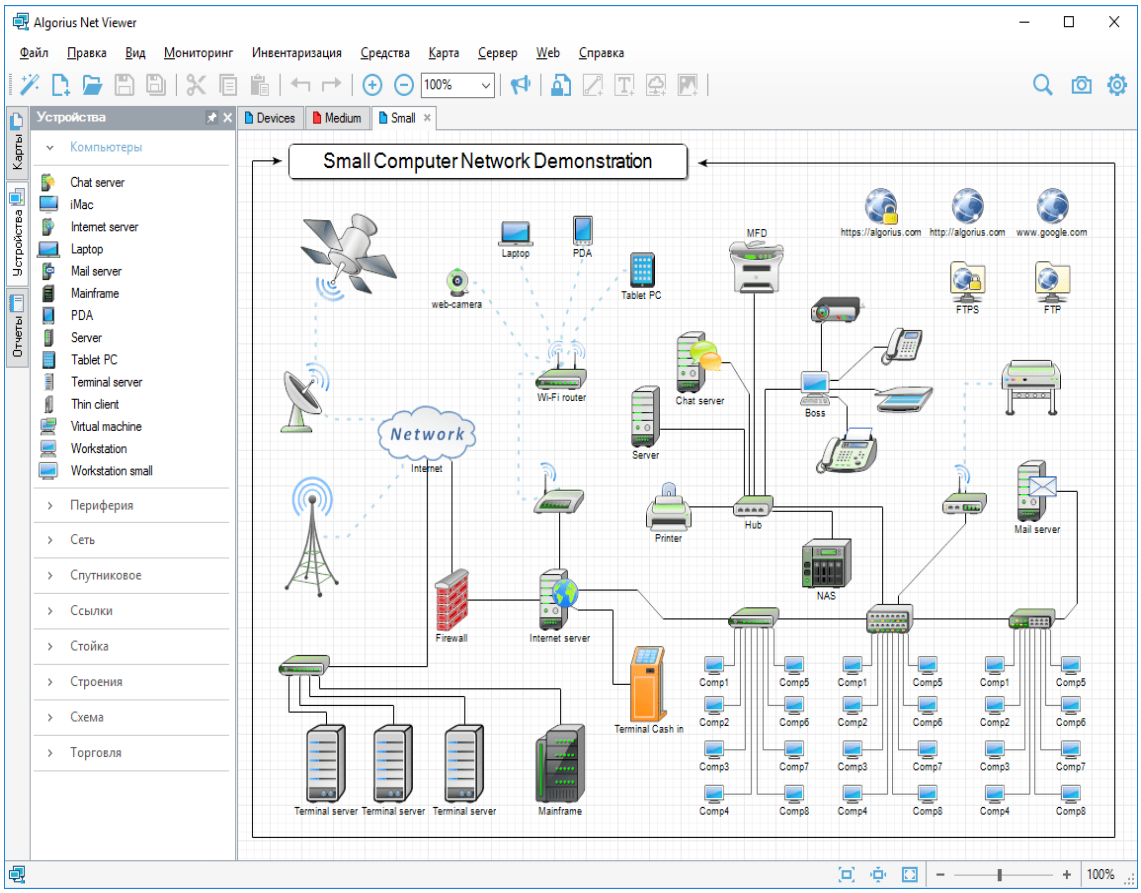


Рис. 6.9. **Робоче вікно програми Algorius Net Viewer**

Особливості застосування та переваги програми Algorius Net Viewer такі:

зручний інтерфейс і побудова мапи мережі за допомогою вбудованого майстра створення мапи мережі;

можливість використання зовнішніх команд, автоматизація їхнього виконання;

підтримання протоколу SNMP;

використання менеджера паролів;

моніторинг мережі в режимі реального часу;

підтримання різних варіантів опитування (підтримання протоколів ICMP, UDP, TCP, SNMP, WMI, ARP, DNS, NETBIOS, HTTP(s), FTP(s));

підтримання баз даних;

можливість підтримання різних варіантів оповіщення про події в мережі.

Недоліки продукту такі:

програма блокує (ускладнює) роботу деяких протоколів (ARP, DNS та ін.), тому продовжують роботу над її вдосконаленням;

програмний продукт розповсюджується на платній основі.

10. **Аналізатори протоколів.** Сучасні аналізатори протоколів мають одночасно декодувати велику кількість протоколів на швидкостях до декількох Гбіт/с. Крім детального аналізу протоколів, мультисервісні прилади мають стежити за мережею та збирати докладну статистику про її роботу. Останнє покоління аналізаторів протоколів надає можливість усебічного стеження як за справною роботою й завантаженістю всієї мережі, так і за функціонуванням окремої програми індивідуального користувача. Принцип роботи аналізатора протоколів відрізняється від принципу роботи засобу моніторингу мережі. Аналізатор мережевих протоколів досліджує весь мережевий трафік, який проходить повз нього. Локальні мережі за своєю природою є широкомовними, тобто кожен кадр від будь-якої станції в межах колізійного домену бачать усі станції цього домену мережі. Підключаючи аналізатор до будь-якої точки колізійного домену мережі, можна бачити весь трафік у цьому домені.

Аналізаторів протоколів розподіляють на дві категорії: програмні й апаратні (програмно-апаратні). *Програмний аналізатор* – це програма, що встановлюють на персональний комп'ютер зі звичайною мережевою платою (рис. 6.10).

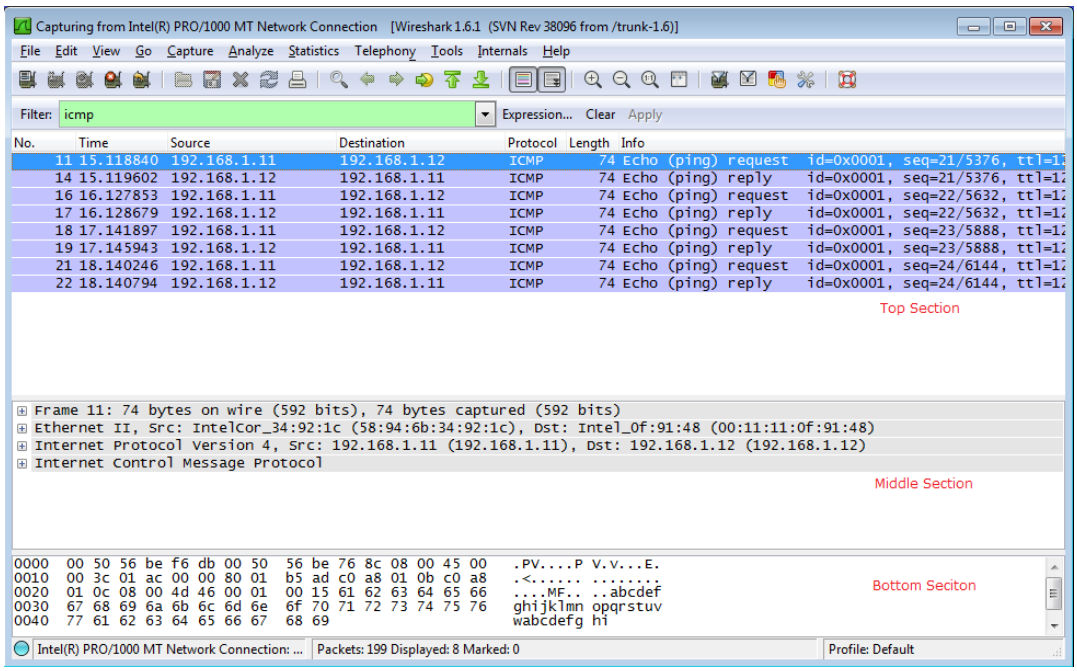


Рис. 6.10. **Робоче вікно аналізатора Wireshark**

Аналізатор мережевих протоколів можна використовувати для:

локалізації складних проблем та генерації трафіка для випробування на вторгнення (penetration test), із метою перевірки системи захисту;

роботи із системами виявлення вторгнень Intrusion Detection System (IDS);

здобуття такої інформації як базової моделі трафіка (baseline traffic patterns) і метрики утилізації мережі;

прослуховування трафіка, тобто локалізації несанкціонованого трафіка з використанням Instant Messaging (IM) або безпроводових точок доступу Access Points (AP) й ідентифікації протоколів, що не використовують.

Маючи великий буфер для збирання та зберігання пакетів, аналізатори протоколів дозволяють швидко локалізувати причину перебою в мережі, наприклад: визначити факт перевантаження конкретного сервера; безслідне зникнення пакетів транспортного рівня на несправних мережевих платах, комутаторах і маршрутизаторах; IP-пакети з неправильною контрольною сумою; дублікати IP-адрес та ін.

Аналізатори протоколів надають можливість збирати дані про роботу протоколів усіх рівнів мережі та здебільшого є здатними робити генерацію тестового трафіка в мережу. Під час проведення діагностики

мережі за допомогою програмного аналізатора протоколів особливе значення мають типи мережевої плати та драйвера, на яких працює програмний аналізатор. Типи мережевої мапи й типи драйвера визначають можливість програмного аналізатора протоколів фіксувати помилки в мережі на канальному рівні.

Аналізатори протоколів мають певні переваги перед іншими технологіями, як-от: аналіз та визначення несправностей у процесі роботи; оцінювання продуктивності під час планування мережі та розв'язування проблем функціонування мережевих застосунків, що є недоступним для систем віддаленого управління комп'ютерною мережею й контролю за нею.

Аналізатори трафіка (аналізатори мережевих пакетів) – це прилади або програмно-апаратні комплекси, які здійснюють захоплення мережевого трафіка з подальшим його поглибленим аналізом – DPI (Deep Packet Inspection), здійснюють мережевий аналіз, проводять тестування пропускної спроможності та продуктивності комп'ютерної мережі, а також різних застосунків.

Платформа NETSCOUT nGeniusONE – це система моніторингу основних застосунків, сервісів та мережевої IT-інфраструктури (NPMD). Вона поєднує оцінювання ситуації в режимі реального часу, а також можливості аналізу статистичних даних і багаторівневого аналізу для ефективного і дієвого управління сервісами в розподілених комп'ютерних мережах.

Платформа VIAVI Observer є комплексним рішенням для моніторингу та діагностики продуктивності мережі (NPMD), що дозволяє надійно підтримувати всі служби та сервіси в комп'ютерній мережі.

11. **Вимірювальна модульна платформа EXFO FTB-500.** Універсальна вимірювальна платформа EXFO FTB-500 дозволяє проводити тестування комп'ютерних мереж, також мереж, що застосовують технології мультисервісних мереж зв'язку на основі концепції Next Generation Network (NGN) та мереж наступного покоління.

Вимірювальна платформа EXFO FTB-500 працює під управлінням операційної системи Windows XP на базі процесора (CPU) Intel Core 2 Duo й забезпечує безпроводовий зв'язок. Завдяки програмному забезпеченню FastReporter, призначеному для опрацювання результатів вимірювання й підготовки звітів, можна виконати повну діагностику і здійснити аналіз мережі, тестування та створення звітної документації [5].

Пристрій дозволяє завантажувати та переглядати довідкові матеріали безпосередньо на екрані платформи. Також є доступ до мережевого журналу EXFO Next-Gen Networking, у якому зберігають інформацію про сучасні методи тестування й докладно описують поширені проблеми, які можуть виникати у процесі експлуатації мережі й алгоритми їхнього усунення. Зовнішній вигляд апаратного аналізатора показано на рис. 6.11.

Рис. 6.11. **Апаратний аналізатор EXFO FTB-500**

До вимірювальної платформи FTB-500 випускають кілька десятків модулів для вимірювання різних параметрів мереж із пакетним передаванням даних та аналізу комунікаційних протоколів версії IPv4 та IPv6 – Transport and Datacom (T&D) і тестування мереж доступу (xDSL, xPON/FTTx, Ethernet тощо). Найбільш уживаними є модулі для діагностування комп'ютерних мереж: Ethernet/IP/MPLS/VPN, IPTV/VoIP, SDH/SONET/OTN, Fibre Channel, PDH/DSn, VDSL2/ADSL1/2/2+, CPRI та ін. Платформа EXFO FTB500 є здатною вмістити від одного до восьми змінних вимірювальних модулів та застосунки, які можуть знадобитися для тестування на кожному кроці життєвого циклу мережі: під час побудови, приймально-здавальних випробувань, активації сервісу, експлуатації, обслуговування, діагностики та усунення несправностей.

Платформа EXFO FTB-500 сумісна з такими модулями для T&D, як:

FTB-8120NGE, FTB-8130NGE – модулі аналізаторів 10G SDH/Ethernet;

FTB-8120, FTB-8130 – модулі аналізаторів 10G SDH/SONET/OTN;

FTB-8510G – модуль аналізатора 10G Ethernet (включно з IPv6);

FTB-8115 – модуль аналізатора SDH/SONET до 2,5 Гбіт/с;

FTB-8510B – модуль аналізатора Fast Ethernet до 1 Гбіт/с;

FTB-8105 – модуль аналізатора PDH/SDH до 155 Мбіт/с.

Платформа EXFO FTB-500 сумісна з такими модулями для тестування оптоволоконних ліній зв'язку, як:

FTB-5240S, FTB-5240BP – модулі аналізаторів оптичного спектра;

FTB-7200D – модуль рефлектометра одномодовий і багатомодовий;

FTB-7300E – модуль рефлектометра для PON FTTx/MDU;

FTB-7400E – модуль рефлектометра для магістралей та CWDM;

TB-7600E – модуль рефлектометра для ультрамагістралей та ін.

Гарантована сумісність з усіма майбутніми модулями для тестування мереж, які будуть підтримувати передавання Ethernet пакетів на швидкості 25, 40 та 100 Гбіт/с (25 GbE 40 GbE, 100 Gigabit Ethernet), згідно з IEEE P802.3ba Ethernet Task Forc. Зовнішній вигляд універсального апаратного аналізатора FTB-8120NGE/8130NGE показано на рис. 6.12.

Рис. 6.12. **Апаратний аналізатор FTB-8120NGE**

Нині розроблено нову модульну вимірювальну платформу EXFO FTB-1v2 Pro. Вона працює під управлінням операційної системи Windows 10 із процесором на 4 ядра. Платформу FTB-1v2 Pro призначено для вимірювання параметрів оптоволоконних ліній (магістральних, міських, мереж доступу) для тестування протоколів Ethernet, OTN, SDH/PDH, CPRI та ін. Інтерфейси: USB 3.0 та 1 G Ethernet, Wi-Fi та Bluetooth (опція RF). Флеш-пам'ять на 128 Гбіт.

12. **Мережеві аналізатори**. Мережеві аналізатори (не слід плутати з аналізаторами протоколів) – це еталонні вимірювальні інструменти для діагностики та сертифікації кабелів і кабельних систем. Як приклад можна навести мережеві аналізатори компанії HewlettPackard – HP 4195A і HP 8510C. Мережеві аналізатори містять високоточний частотний генератор і вузькосмуговий приймач. Передаючи сигнал різних частот

у передавальну пару та вимірюючи сигнал у приймальній парі, можна виміряти затухання, NEXT і діагностику мереж Ethernet (10/100/1000 Мбіт/с).

Мережеві аналізатори – це універсальні й чутливі прилади, призначені для використання в лабораторних умовах (рис. 6.13).

а) LanExpert

б) EtherScope Series II

в) ES2-LAN

г) OptiView™ Series III Integrated Network Analyzer

Рис. 6.13. **Мережеві аналізатори**

Мережеві аналізатори також дозволяють виконувати таке:

здійснення діагностики оптичних кабелів та мереж Wi-Fi-стандартів 802.11a/b/g/n;

здобуття детальної інформації про мережу, перегляд налаштувань будь-якого мережевого пристрою: конфігурації, адресації, статусу;

визначення доступних інтерфейсів, активних портів, MAC та IP-адрес, імен SNMP та швидкості підключення;

виявлення подвійного використання IP-адрес, невідповідності налаштувань устаткування мережі та помилок роботи сервера DHCP;

підтримання технологій VLAN та PoE (Power over Ethernet).

13. **Кабельні тестери.** Кабельний тестер – це пристрій, що дозволяє перевіряти стан кабелю або кабельної лінії (рис. 6.14а). Тестери та кабельні аналізатори дозволяють виконувати вимірювання електричних характеристик кабельної лінії. Нині є три класи приладів: для базової перевірки кабелю, кваліфікації й сертифікації кабельного устаткування.

Найбільш важливими параметрами, що вимірюють під час тестування кабелю, є такі: довжина, схема розведення провідників, загасання, перехресні наведення на ближньому кінці кабелю (NEXT), опір по шлейфу постійного струму (для мідних ліній) і поворотні втрати (Return loss) та ряд інших.

Основне завдання кабельних аналізаторів, за допомогою яких виконують сертифікацію кабельних ліній, – це перевірка кабельної системи на відповідність міжнародним стандартам. Етап сертифікації є невід'ємним під час побудови структурованої кабельної системи. Галузеві стандарти організацій ISO та TIA визначають класи або категорії кабелю. Зовнішній вигляд спеціального кабельного аналізатора показано на рис. 6.14б.

а) кабельний тестер ММР-50

б) кабельний аналізатор CERTILAN C.A. 7040

Рис. 6.14. **Кабельні тестери**

Кабельний тестер виконує повне тестування кабелю та виводить на екран частотні залежності різних параметрів, які відповідають ISO або TIA. Крім того, такий прилад зазвичай має можливість роздрукувати результати вимірювань у стандартизованому вигляді, і надалі цей роздрукований документ використовують під час затвердження та введення лінії зв'язку в експлуатацію.

Будь-яка кабельна лінія мусить мати паспорт, що містить інформацію про технічні дані лінії: тип (марку) кабелю; довжину траси лінії; місця встановлення муфт, поворотів; точки, які використовують для формування ділянок лінії; місця встановлення устаткування тощо.

14. **Кваліфікаційні та мережеві тестери.** *Кваліфікаційний тестер* – це прилад, який дозволяє перевіряти, чи забезпечують наявні кабелі достатню пропускну спроможність для підтримання голосу, стандартів 10/100 Ethernet, VoIP, Gigabit Ethernet та ін.

Багатофункціональні тестери дозволяють виявляти велику кількість несправностей усередині мережі та приводити її до робочого стану. Вони дозволяють з'ясувати таке: активність порту, збіг параметрів дуплексу, точне розташування пошкодження – в мережі або кабелі. Кабельні тестери надають всю необхідну інформацію про розташування аномалії та її причини.

Кабельний тестер FLUKE Networks CIQ-100 призначено для виявлення несправностей та сервісного обслуговування локальної мережі Ethernet. Зовнішній вигляд кваліфікаційного тестера показано на рис. 6.15.

Рис. 6.15. **Кваліфікаційний тестер FLUKE Networks CIQ-100**

Кваліфікаційний кабельний тестер Networks CIQ-100 дозволяє виконувати таке:

обслуговування устаткування кабельної лінії, визначення пошкодження й ізоляцію несправних кабелів (струмопровідних проводів) від мережі;

підключення до кабелю та виведення на екран тестера детальної інформації про його довжину, схему підключення, параметри підключеного пристрою на протилежному кінці кабелю тощо;

визначення наявності короткого замикання, місце розривів струмопровідних проводів та відображення їх у графічному вигляді на екрані;

вимірювання параметрів кабелів типів: UTP, STP, FTP, SSTP, RG6, RG59 – і телефонних кабелів: ТБ, ТБГ, ТГ, ТПП, ТППеп, П-274М, П-296 та ін.;

автоматичне виконання тестування кваліфікації 1000BASE-T, VoIP, 100BASE-TX, 10BASE-T, 1394b S100, TELCO й також коаксіальних кабелів;

перевірку кабелю з використанням цифрових тональних сигналів, аналогових тональних сигналів, виявлення та ідентифікацію портів Ethernet.

Мережеві тестери – це надійні та прості в експлуатації прилади для діагностики кабелю на основі кручених пар (звита пара, англ. twisted pair). Використовують для побудови мереж у багатьох технологіях, наприклад, Ethernet, ARCNet, 1000BASE-T й Token ring. Завдяки своїй дешевизні та зручності встановлення, є найпоширенішою для побудови локальних мереж.

Мережевий тестер характеризується широкими можливостями, зокрема має функцію моніторингу. Він дозволяє виконувати тестування звитої пари й коаксіальних кабелів, визначати довжину кабелю або відстань до пошкодження. Приклади мережевих тестерів показано на рис. 6.16.

а) тестер світлодіодний (RJ45, RJ12, STP і UTP, BNC, USB)

б) мережевий тестер NetTool™ Series II NTS2-PRO

Рис. 6.16. **Мережеві тестери**

Мережеві тестери дозволяють виконувати завдання з діагностики кабельних ліній звитої пари, на основі яких побудовано комп'ютерні локальні (абонентські) мережі. Можливості тестера такі:

тестування екранованого та неекранованого кабелю звитої пари;

тестування кабелю із RJ12, RJ45 та BNC-конекторами;

тестування USB-кабелів;

для проведення тестування віддаленої точки підключення є передбачений виносний блок із RJ12, RJ45 та USB-A рознімачами;

тестування наявності з'єднання між двома точками підключення з дальністю до 100 м (305 футів);

визначення прямого та перехресного з'єднання;

за допомогою світлодіодів показує обрив окремої лінії в кабелі;

надає звуковий сигнал у разі правильного конекту;

має індикатор розряду батареї живлення.

Мережевий кабель звита пара – це вид мережевого кабелю з однією або декількома парами ізольованих провідників, скручених між собою (із визначеною кількістю витків на одиницю довжини для кожної пари) для зменшення взаємних наведень під час передавання сигналу та покритих пластиковою оболонкою. Застосовують кілька категорій кабелю звитої пари, які нумерують від CAT 1 до CAT 8.2.

Зі зміною категорії кабелю підвищується його пропускна спроможність, але й зростають вимоги до екранування. Категорії неекранованої звитої пари описують у стандарті EIA/TIA-568. Дальність передавання даних зі звитої пари обмежено відстанню до 100 м.

Кабель підключають до мережевих пристроїв за допомогою рознімача типу 8P8C і 8P6C, які називають конекторами RJ-45 (рис. 6.17).

Рис. 6.17. **Конектор 8P8C (RJ-45)**

Рознімачі типу 6P6C і 6P4C є конекторами RJ-12. Прозорий корпус конектора забезпечує візуальний контроль за якістю затискання провідників (рис. 6.18).

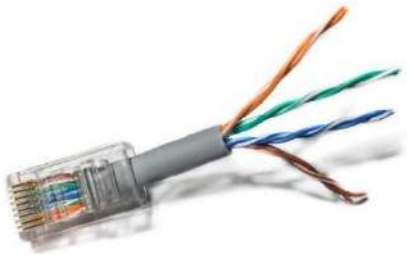


Рис. 6.18. **Звита пара з конектором 8P8C**

Для захисту від зовнішніх і перехресних завад між крученими парами в кабелях звитої пари можна застосовувати екранування. Екранування застосовують як до окремих кручених пар, які обертають в алюмінієву фольгу, так і до кабелю загалом у вигляді загального екрана з фольги та/або обплетення з мідного дроту.

Для під'єднання конекторів до кабелю необхідно дотримуватися використання певної послідовності підключення мідних дротів, які в кабелі позначено різними кольорами.

Прямий кабель, обтягнутий з обох боків за однаковим стандартом, використовують для з'єднання різнотипного устаткування ("комп'ютер – комутатор", "комутатор – маршрутизатор" тощо).

Перехресний кабель, обтягнутий з обох боків за різними стандартами, використовують для з'єднання однотипного устаткування ("комп'ютер – комп'ютер", "комутатор – комутатор", "маршрутизатор – маршрутизатор" тощо).

6.3. Висновки

У роботі здійснено детальний аналіз сучасних способів моніторингу устаткування комп'ютерних мереж та контролю за ним, розглянуто особливості функціонування, показано переваги й недоліки основних програмних та апаратних засобів і мережевих аналізаторів, які використовують для моніторингу та діагностування устаткування IP-мереж.

Системи моніторингу комп'ютерних мереж реалізовано як гнучкі й адаптивні системи, побудовані на принципах автоматизації й інтелектуалізації, та їх використовують для оперативного управління всіма компонентами мережі й раціонального (оптимального) використання устаткування мережі. Тенденція до подальшого розвитку систем управління полягає в інтегруванні кількох підходів під час забезпечення сумісності нових технологій із наявними програмними засобами.

Технологія TMN буде домінувати на транспортних мережах на рівні управління елементами. Технологія CORBA підтримує складні об'єктивно-орієнтовані прикладні системи та є надійним рішенням для рівнів управління послугами й діяльністю. Система діагностики та управління SNMP буде й надалі застосовувати в управлінні мережами передавання даних.

Технологія SDN дозволяє будувати інфраструктурні хмарні сервіси, створювати віртуальні вузли та виділяти віртуальні мережеві ресурси для них. Водночас для управління мережами використовують мережеві служби, які здійснюють планування, управління й контроль за роботою всіх її компонентів та інформаційних ресурсів.

Розділ 7
Завдання контролю за процесами обміну даними в комп'ютерних мережах й управління ними

7.1. Вступ і формулювання завдання

Сучасні тенденції розвитку та впровадження програмного забезпечення в системах передавання й опрацювання даних потребують удосконалення адаптації способів та алгоритмів рішення завдань до умов роботи, що постійно змінюються, у наочній галузі. Такі можливості важко забезпечити на основі розроблення апаратних засобів, які потребують тривалого часу проєктування та впровадження. Найбільш зручним є розроблення програмних продуктів, що дозволяють швидко адаптуватися до умов, що змінюються. В умовах зростальних вимог до достовірності передавання повідомлень і часу їхнього доставлення є актуальним завдання контролювання й управління процесом передавання даних, а також оперативного доведення інформації про стан каналів зв'язку та іншого устаткування в наочному вигляді до оператора.

Мета управління полягає в забезпеченні заданого рівня якості функціонування комп'ютерних мереж і надання послуг. Основне завдання управління мережею полягає в реалізації цілеспрямованого впливу (моніторингу, контролю, адміністрування) на устаткування КМ за допомогою засобів автоматизації та інформатизації.

В основі організації управління лежать такі принципи:

інтеграція функціональних, фізичних та інформаційних структур управління;

створення гнучкої архітектури управління на основі методології відкритих систем, що забезпечує можливість реконфігурації та розвитку автоматизованої системи управління мережами;

стандартизація компонентів системи управління;

підвищення рівня автоматизації процесів управління.

Дедалі більшого поширення набувають нові технології, пов'язані з опрацюванням і передаванням не тільки комп'ютерних даних, але й голосової та відеоінформації. Інтеграція різних видів інформаційного обслуговування в межах однієї мережі є закономірним наслідком розвитку цифрових технологій. Розподілений характер великої мережі зі складною

структурою унеможливлює підтримання її роботи на належному рівні без системи управління, яка також є складною системою [8].

Нині система управління працює в автоматизованому режимі, виконуючи найпростіші дії з управління мережею автоматично, а складні рішення надаючи ухвалювати людині на основі підготовленої системою інформації [8]. Система управління мережею має забезпечити, з одного боку, підтримання в робочому стані як мережі загалом, так і окремих її складових, для того щоб вона могла виконувати свої функції; а з іншого – розподіл і доставляння інформаційних повідомлень за адресами з дотриманням різних вимог користувачів [9].

Досвід розроблення й експлуатації складних систем, успіх їхньої оптимізації залежить не тільки від адекватності моделі процесу її функціонування та досконалості використовуваного математичного апарату для досягнення точних і достовірних результатів оцінювання характеристики системи, а й від вибраного критерію ефективності системи.

На основі стандартної моделі взаємодії відкритих систем виділимо групи характеристик, які визначають ефективність КМ. До таких характеристик належать:

часові (час доставляння, час опрацювання повідомлень);

наведені цифри щодо (пропускна спроможність, завантаженість елементів);

зв'язності (тимчасова, просторова, протокольна);

цілісності (стійкість, достовірність, точність передавання);

безпеки;

надійності та живучості.

Нині в Україні немає точної концепції зі створення системи мережевого управління. Тому всі питання, пов'язані з розробленням такої системи, є надзвичайно актуальними. Питанням управління телекомунікаційними мережами присвячено достатню кількість робіт [9; 13].

У [10] висвітлено загальні питання управління мережами зв'язку. Наведено базову інформацію про структуру й особливості протоколів CMIP і SNMP, зроблено огляд платформ і продуктів мережевого управління деяких іноземних компаній.

У [11] дано основи побудови систем управління телекомунікаційними мережами. Висвітлено такі питання динамічного управління, як маршрутизація, принципи управління трафіком. Наведено багаторівневу архітектуру мережевого управління.

Актуальними є питання, пов'язані з оцінюванням ефективності мереж. Зазвичай, ефективність мережі обміну оцінюють за допомогою окремих приватних показників якості, як-от імовірність помилки, час доставляння, без урахування цінності переданої по мережі інформації та зв'язності елементів мережі. У [21] поставлено завдання узагальненого підходу до оптимального проєктування систем управління мережею. Запропоновано введення єдиного векторного показника ефективності. Водночас не вказано його формульне обчислення. Не враховано можливу невизначеність стану елементів мережі.

У роботі [40] розглянуто завдання синтезу оптимальної системи управління мережею за допомогою мінімаксного методу, яка компромісно оптимізує тільки обмеження на вхідні дані та спектр наявних умов.

У [41] проаналізовано можливість створення автоматичної системи управління мережею. Однак як ухвалювати рішення, ця система управління не розглядає.

У галузі теоретичних досліджень є значна кількість робіт, присвячених питанням підвищення ефективності комп'ютерних мереж. Частина з них є орієнтованою на дослідження інформаційних мереж, розроблення моделей управління ними, методів їхнього аналізу та реалізованих у них протоколів [18].

Питанням розподілу мережевих ресурсів присвячено роботу [26]. У наявних роботах у галузі динамічного управління в інформаційних мережах вирішують завдання управління потоками мовних повідомлень [10], розроблено методи підвищення продуктивності мереж із мультимедійними послугами [30], методи підвищення ефективності протоколів доступу обчислювальних мереж [45], управління пропускною спроможністю мереж зв'язку [32], динамічного управління потоками інформації [7], а також оптимізації та динамічного регулювання навантаження інформаційних мереж [9]. Кожну із цих робіт присвячено вирішенню приватного конкретного завдання, але її не спрямовано на розроблення комплексного підходу, що забезпечує необхідну ефективність функціонування всієї системи динамічного управління.

Під час оцінювання ефективності роботи мережі у процесі управління необхідно враховувати якість обслуговування та пріоритетність передавання інформації. У [40; 41] описано деякі питання реалізації заданої якості обслуговування в мережах, однак у них не враховано той факт, що інформація має свою цінність, причому зі збільшенням часу затримки

в передаванні або обслуговуванні цінність інформації знижується, тобто відбувається її старіння.

Низька ефективність застосовуваних методів збирання інформації, ухвалення рішення та контролю за виконанням цього рішення знижує якість обслуговування користувачів. Тому для виконання вимог щодо якості обслуговування необхідно вдосконалення методів контролю за процесом обміну даними в мережах із комутацією пакетів та управління ним шляхом моделювання мультиоб'єктного управління ресурсами адаптивних комп'ютерних мереж.

7.2. Контроль за процесом обміну даними в мережах із комутацією пакетів та управління ним

Завдання управління мережею зв'язку підрозділяють на завдання планування, оперативного контролю й управління.

До завдань планування належать прогнозування стану мережі, розроблення заходів щодо профілактики й ремонту техніки тощо. Оперативний контроль та управління передбачають збирання, опрацювання, документування та відображення інформації про стан мережі, ухвалення рішень, доведення рішень до відома виконавчого органа й контроль за виконанням ухвалених рішень, складання зведень і звітних документів, що характеризують стан мережі зв'язку [7].

Завдання контролю й оперативного управління розподіляють на інформаційні, розрахункові, які формалізують і не формалізують. Інформаційні завдання (збирання й опрацювання) здебільшого є логічними й характеризуються більшим обсягом даних, що опрацьовують.

Під час вирішення розрахункових завдань обсяг інформації, яку опрацьовують, не є великим, однак застосовувані методи рішення можуть бути досить складними. Особливістю їх є те, що в результаті вирішення визначають нові дані.

Ці завдання можна розподілити на дві підгрупи. Одна підгрупа містить завдання, пов'язані із прямими розрахунками, а інша – із визначенням оптимального або раціонального варіанта [9].

Розрахункові завдання, які формалізують, можуть бути з успіхом вирішені за допомогою засобів автоматизації, які не формалізують і непередбачені заздалегідь завдання може бути вирішено тільки людиною-оператором.

Усе це переконує в необхідності в застосуванні автоматизованих систем управління мережею (АСУМ). Водночас автоматизації, що входять до АСУМ, дозволяють швидко й із високою точністю виконувати необхідні розрахунки, вирішувати завдання опрацювання даних, здійснювати контроль за станом елементів мережі, виробляють інформацію черговому операторові для ухвалення рішення. Тому під час впровадження АСУМ виникає проблема розподілу завдань між людиною та технічними засобами [10]. Розв'язання цієї проблеми пов'язано з необхідністю враховувати, що технічні засоби й людина виконують функції управління з неоднаковим ефектом. Так, людина (черговий оператор) має обмежену можливість з опрацювання інформації, стомлюється, розрахунки виконує повільно й неточно. Однак він здатний реагувати на несподівані ситуації й перетворювати уривки повідомлень на взаємозалежне ціле. Обчислювальний засіб має високу пропускну спроможність і точність, може працювати тривалий час, але не може вирішувати завдання в непередбаченій ситуації.

Роботу чергового оператора пов'язано зі здобуттям та опрацюванням великого обсягу інформації, часто виникає необхідність у дістанні різного роду довідкового матеріалу. Тому оперативний контроль за мережею й управління нею вкрай утруднено без впровадження автоматизованих робочих місць (АРМ) операторів.

Автоматизована система управління мережею вирішує такі завдання: збирання й опрацювання даних про стан елементів мережі (вузлів, каналів зв'язку); видачу інформації черговому операторові; формування службових повідомлень під час аварійних ситуацій; накопичення статистичних даних про несправності та проходження пакетів різної категорії важливості; формування розпоряджень на основі ухваленого рішення; управління процесами резервування каналів і ліній зв'язку; видачу варіантів рішень із використання резерву; оцінювання технічного стану й ремонт елементів мережі [10].

Для вирішення цих завдань автоматизована система управління мережею містить системи динамічного й адміністративного управління, а також систему технічної експлуатації. У разі виходу з ладу елементів мережі потрібно визначати ступінь руйнування й можливість відновлення елементів, можливість дістання додаткових ресурсів із каналів і трактів первинної мережі, використання резервних центрів комутації й ліній зв'язку (тропосферних, радіорелейних та ін.). У разі недоліків ресурсів

мережі має бути передбачено можливість у примусовому відключенні деяких користувачів.

Для забезпечення функціонування АСУМ необхідно передавати службову інформацію (сигналізації й управління). Цю інформацію в мережах із комутацією пакетів передають робочими каналами разом із повідомленнями абонентів, і, зазвичай, вона має вищий пріоритет.

Технічну базу системи адміністративного управління становить сукупність пунктів управління мережею й елементами мережі, сполученими з вузлами комутації.

Автоматизована система технічної експлуатації (АСТЕ) є сукупністю методів управління, програмних і технічних засобів, людських та матеріальних ресурсів, що здійснюють технічну експлуатацію за прийнятими критеріями управління. Основною метою АСТЕ є мінімізація трудових витрат під час забезпечення необхідної якості обслуговування за обмежень на економічні витрати [27].

Вирішення завдань автоматизації управління мережею обумовлено такими особливостями, як необхідністю обліку великої кількості контрольованих параметрів територіально рознесених об'єктів.

Контроль за станом мережі та її елементів розподіляють на програмний, апаратурний і програмно-апаратурний.

Програмний метод засновано на використанні спеціальних тестових програм або робочих програм, що передбачають додаткові операції для перевірки апаратури.

Апаратурний метод засновано на використанні вбудованих засобів контролю та спеціальної контрольної апаратури.

Автоматизована система контролю, залежно від особливостей контрольованого об'єкта, може бути централізованою, із частковою централізацією й децентралізованою [40].

За *централізованої системи* контролю на об'єктах розташовують тільки пристрої знімання інформації. Класифікатор та інформатор перебувають у центральному обчислювачі. Така структура є доцільною в разі контролю за станом устаткування одного об'єкта з більшою кількістю контрольованих елементів.

За *часткової децентралізації* на основі здобутої інформації на об'єкті вирішують завдання класифікації й систематизації інформації про стан цього об'єкта. На основі здобутої інформації класифікатором мережі визначають її стан та інформатор мережі видає необхідні відомості.

За *повної децентралізації* оцінювання стану мережі здійснюють на кожному об'єкті за власною інформацією й інформацією, здобутою від інших вузлів комутації.

Під час визначення глибини охоплення окремих пристроїв за допомогою вбудованого контролю варто враховувати, що зі збільшенням кількості контрольованих параметрів різко зростає обсяг додаткового устаткування, що може вплинути на надійність елементів мережі.

Залежно від призначення, розрізняють функціональний та автономний контроль.

Функціональний контроль є оперативним безперервним контролем, що дозволяє оцінити стан мережі у процесі передавання інформації. Реалізують цей контроль шляхом використання спеціальних убудованих пристроїв або датчиків.

Автономний контроль призначено для автоматичної перевірки працездатності апаратури під час відбудовних і профілактичних робіт, і його здійснюють шляхом тестування окремих пристроїв [9].

Мережа з контрольними пунктами становить складну відбудовну систему цілодобового використання. Із метою локалізації відмов у різних блоках у реальному масштабі часу, здійснюють функціональну діагностику елементів мережі, що містить сигнатурний аналіз і параметричну діагностику. Крім того, застосовують тестову діагностику, що дозволяє локалізувати відмови з більшою, ніж функціональна діагностика, роздільною здатністю. Однак потрібно додатково використовувати ресурси мережі для передавання тестів та здобуття результатів реакцій на них.

Звичайно ці два види діагностики використовують спільно. У цьому разі контроль за функціонуванням елементів мережі здійснюють за сукупністю параметрів, що характеризують їхню працездатність. Правильність функціонування на виході пристрою захисту від помилок (ПЗП) передавача визначають рівень сигналу, тривалість, логіку обміну; на вході ПЗП приймача перевіряють первинні параметри сигналу, а на виході – кількість виявлених помилок [10]. Вихід чисельних параметрів за припустимі межі сигналізує про порушення працездатності елементів мережі. Якість функціонування каналів зв'язку контролюють за рівнем сигналу й заводу точці приймання, частотою та тривалістю короткочасних перерв, зрушенню частоти, відхиленням амплітудно-частотних (АЧХ) і фазо-частотних (ФЧХ) характеристик, первинних параметрів сигналу.

Контроль за функціонуванням двополюсної мережі здійснюють за правильністю передавання даних, часу доставляння повідомлень,

імовірністю переповнення буферних запам'ятовувальних пристроїв (БЗУ), кількістю перезапитів перекручених пакетів. Водночас абоненти обмінюються тестовими пакетами та ведуть спостереження за довжиною черги на вузлах комутації (ВК) і часом затримки пакетів у різних фазах. Якість закритих телефонних каналів оцінюють за проходженням зашифрованих спеціальних сигналів. Контроль за технічним станом ділянки мережі здійснюють під час передавання службового пакета й повернення його передавальному абонентові. Після оцінювання ймовірносно-тимчасових характеристик ухвалюють рішення про стан контрольованої ділянки.

Оцінювання стану напрямку зв'язку здійснюють за мінімально необхідною кількістю каналів для передавання потоку інформації. Справним є такий напрямок, у якому кількість каналів заданого виду є більшою за припустиму. Стан мережі зв'язку вважають нормальним, коли всі напрямки є справними. Мережу вважають несправною (аварія), якщо її ефективність знизилася на величину, більшу за припустиму.

Відмітною рисою вузлів комутації пакетів є їхня гнучкість щодо відмов елементів. Відмова однієї або кількох груп елементів (процесора, модуля пам'яті та ін.) трохи погіршує характеристики вузла, не виводячи цей вузол із ладу. Для виявлення несправностей періодично проводять програмне тестування. Під час визначення елементів, що відмовили, використовують передбачену апаратну надмірність і здійснюють зміну конфігурації комплексу, у якому елемент, що відмовив, виводять із контуру взаємодії та замінюють резервним [8]. Після ремонту цей елемент знову вводять у комплекс.

Вимірювання затримки у вузлах здійснюють шляхом підрахунку кількості пакетів у черзі на видачу в канал. У кожному вузлі зберігають мережеву таблицю затримок. Усі вузли мережі передають значення мінімальної затримки до будь-якого іншого вузла. Після додавання часу затримки на цьому вузлі формують скориговану таблицю маршрутизації.

Основними факторами, що впливають на вибір того або того методу підвищення правильності передавання інформації, є такі: необхідна правильність передавання інформації, вид використовуваних каналів зв'язку (симплексна або дуплексні) та їхня якість, інтенсивність і закон розподілу помилок, економічні міркування.

За відсутності зворотного зв'язку в симплексних каналах зв'язку може бути використано багаторазове повторення або коди з виправленням помилок. Якщо помилки в каналі зв'язку є незалежними, то ймовірність виникнення помилок великої кратності є низькою й у цих умовах

можна використовувати коди з виправленням помилок. Якщо ж помилки, що виникають, групують, то пристрої для виправлення помилок, що кодують і декодують, є складними. Крім того, важко передбачити можливу тривалість групової помилки. У цих умовах більш доцільним є використовувати системи з багаторазовим повторенням, які є менш чутливими до зміни кратності помилок, що виникають.

Під час використання дуплексних каналів зв'язку доцільно застосовувати системи зі зворотним зв'язком. За однакового прямого та зворотного каналів системи з вирішальним зворотним зв'язком та інформаційним зворотним зв'язком забезпечують практично однакову правильність передавання інформації. Однак у системах з інформаційним зворотнім зв'язком зворотний канал є більше завантаженим передаванням інформації для підвищення правильності, ніж у системах із вирішальним зворотним зв'язком, а в системах із вирішальним зворотним зв'язком прямий канал є більше завантаженим передаванням надлишкової інформації, ніж у системах з інформаційним зворотним зв'язком.

За задовільного стану обох каналів (прямого та зворотного) системи передавання даних зі зворотним зв'язком є найбільш ефективними. Однак із підвищенням рівня перешкод характеристики систем передавання даних зі зворотним зв'язком (правильність, швидкість і затримки передавання інформації) погіршуються. У цих умовах може виявитися, що передавання інформації без використання зворотного зв'язка є більш ефективним.

Зокрема, у системах із вирішальним зворотним зв'язком (ВЗЗ), алгоритми функціонування яких передбачають однакову реакцію системи на факт виявлення помилок та здобуття сигналу перезапиту, із погіршенням стану одного з каналів (прямого або зворотного) практично припиняють передавання інформації в обох напрямках (по обох напрямках безупинно передають сигнали перезапиту й повторні повідомлення із запам'ятовувальних пристроїв). Тому у процесі побудови системи передавання даних доцільно використовувати комплексний метод підвищення правильності передавання інформації. У разі гарного стану обох каналів система має використовувати вирішальний зворотний зв'язок, а з погіршенням стану одного з каналів її мають вимикати й передавання інформації здійснювати без зворотного зв'язку.

Одним із можливих рішень завдання зниження частоти перезапитів із підвищенням інтенсивності завад є спільне використання в системах

із вирішальним зворотним зв'язком кодів із виправленням і кодів із виявленням помилок. Код із виправленням помилок забезпечує виправлення найбільш інтенсивних помилок, а код із виявленням помилок забезпечує перепопит тих комбінацій, помилки у яких є не виправленими.

У системах із вирішальним зворотним зв'язком повідомлення, які передають, кодують завадостійким кодом, котрий забезпечує виявлення помилок. Приймач, який прийняв повідомлення, виділяє пакети, котрі містять помилки, і посилає по зворотному каналу сигнал перезапиту (вирішальний сигнал), у результаті чого здійснюють повторне передавання інформації.

Проведені дослідження показали, що системи з вирішальним зворотним зв'язком, котрі використовують канали виявлення помилок, є більш ефективним від систем без зворотнього зв'язку, котрі використовують коди з виправленням помилок.

Це пояснено тим, що в системах із вирішальним зворотним зв'язком повторення передавання здійснюють лише під час виявлення помилок у переданому повідомленні, тоді як у системах з автоматичною корекцією помилок на приймальній стороні, незалежно від наявності викривлень, постійно вводять надлишок, котрий є потрібним для виправлення помилок. Крім того, основною перевагою систем із вирішальним зворотним зв'язком (ВВЗ) є постійність інформаційного надлишку під час використання завадостійкого кодування та висока достовірність передавання інформації.

Широке використання методів контролю за передаванням інформації й управління нею обмежено відсутністю простих та ефективних методік виявлення помилок, які дозволяють значно підвищити швидкодію опрацювання даних, не знижуючи достовірності приймання повідомлень, та ефективність обміну даними за автоматизації управління в мережах із комутацією пакетів. Тому в умовах зростальних вимог до достовірності передавання повідомлень і часу їхнього доставлення є актуальним завданням розроблення ефективних засобів опрацювання інформації в системах обміну даними як із часовим, так і адресним перезапитом.

Аналіз відомих методів підвищення достовірності передавання даних показує [10], що сьогодні застосовують переважно системи безперервного передавання інформації із часовим перезапитом, котрі відрізняються найбільшою простотою реалізації. Основним недоліком цього способу передавання даних є відносна велика кількість та обсяг повторних передавань. Дійсно, у разі виявлення помилки в повідомленні повторному

передаванню підлягає все повідомлення. Отже, ціле повідомлення можуть повторювати кілька разів, водночас суттєво збільшується час його доставляння та знижується ефективність системи обміну даними загалом.

Цього недоліку не мають аналогічні системи з адресним перезапитом, у котрих передбачають не тільки виявлення помилок у повідомленні, але й локалізацію їх із точністю до пакета. Водночас, у разі виявлення помилки повторно буде передано тільки пакети, котрі містять ці помилки.

Тут потрібно зазначити, що за постійного інформаційного надлишку повідомлення меншої довжини мають менший час затримки у тракті передавання інформації та їх передають із більшою достовірністю. Отже, знижено обсяг і кількість повторних передавань, що, своєю чергою, приводить до підвищення ефективності систем обміну даними.

У багатьох сучасних роботах, присвячених передаванню дискретної інформації, є думка, що системи із ВЗЗ, котрі використовують коди з виявленням помилок, є набагато більш раціональним застосовувати для передавання інформації в каналі зв'язку із завадами, ніж займатися виправленням помилок. Більш того, є думка, що використання систем із ВЗЗ повністю розв'язує проблему надійності зв'язку по будь-яких каналах і виключає необхідність у завадостійкому кодуванні [10]. Але відмова від виправлення помилок часто призводить до неефективного використання потенціальних можливостей прямого та зворотного каналів.

Забезпечення інформаційного обміну в сучасних АСУ із заданими вимогами за достовірністю та швидкістю є можливим тільки на базі цифрової мережі інтегрального обслуговування. Поетапне впровадження такої мережі забезпечує суттєве підвищення показників використання мережевих ресурсів наявної мережі, а в подальшому, доступ користувачів до широкого набору послуг (передавання даних, телефонії, телеграфії, факсимільного зв'язку) на основі єдиного цифрового методу. Одними з основних характеристик цього процесу є пропускна спроможність системи та достовірність передавання інформації. Завдання їхнього підвищення завжди була, є і буде актуальною.

У системах із вирішальним зворотним зв'язком повідомлення, які передають, кодують завадостійким кодом, котрий забезпечує виявлення помилок. Приймач, що прийняв повідомлення, виділяє пакети, котрі містять помилки, і посилає зворотним каналом сигнал перезапиту (вирішальний сигнал), у результаті чого здійснюють повторне передавання інформації.

У таких системах під час вибору коригувального коду зазвичай орієнтуються на декотрі середньостатистичні дані про канал зв'язку. Використовувані в цих системах декодери, які виправляють помилки, відрізняються своєю складністю, котра зростає зі збільшенням довжини кодової комбінації та кратності помилок, що виправляють. Суттєвим недоліком описувальних систем є низька пропускна спроможність і неефективне використання введеного структурного або інформаційного надлишку.

На вибір завадостійкого коду великий вплив робить стан каналів зв'язку. Розрізняють кілька порогових значень стану каналів зв'язку. Нормальний стан відповідає інтенсивності потоку в каналі не більше ніж 0,7 від максимально можливої. Стан запобігання перевантаженню відповідає інтенсивності потоку 0,7...0,8. Сигнал про перевантаження відбувається за інтенсивності потоку більше ніж 0,8. Вузли комутації обмінюються між собою інформацією про інтенсивність потоку. Можливі різні варіанти ухвалення рішень за цією інформацією. Наприклад, якщо канал перебуває у стані запобігання перевантаженню, то прийнятий пакет ставлять у чергу до цього каналу, а відправнику видають сигнал про припинення передавання пакетів. Якщо саме в каналі перевантаження, то прийнятий пакет стирається.

Усі вказані раніше методи підвищення достовірності передавання даних може бути реалізовано як апаратно, так і програмно. В останньому випадку обчислювальний комплекс, нарівні з вирішенням основних завдань, має вирішувати комплекси завдань, пов'язані з підвищенням достовірності передавання інформації, діагностикою та виправленням помилок, збереженням інформації, веденням статистики помилок у каналах зв'язку.

7.3. Завдання й алгоритми маршрутизації пакетів у мережах

Складність проблеми контролю за передаванням даних та управління ними полягає в тому, що під час її розв'язання немає повної інформації про вимоги споживачів на обслуговування їх нині. У результаті рішення ухвалюють із застарілої інформації. Тому розроблення простих та ефективних засобів маршрутизації пакетів і повідомлень у процесі передавання інформації, а також використання нового програмного забезпечення устаткування в системах обміну даними є актуальним завданням.

Під *алгоритмом маршрутизації* часто мають на увазі протокол мережевого рівня, який управляє пакетами під час їхнього руху мережею зв'язку до необхідного місця призначення. Моменти часу, коли ухвалюють рішення про вибір маршруту, залежать від того, чи використовує мережа дейтаграмне передавання інформації або віртуальні з'єднання.

У *дейтаграмному* з'єднанні два послідовні пакети однієї пари користувачів можуть проходити різними маршрутами та вибирати маршрут, необхідний для кожного пакета.

У мережі з *віртуальними* з'єднаннями маршрут вибирають під час установлення кожного віртуального з'єднання. Алгоритм маршрутизації використовують для вибору шляху по мережі для цього віртуального з'єднання. Усі пакети послідовно використовують цей шлях до моменту, коли це віртуальне з'єднання закінчує своє існування або коли для цього з'єднання з якихось причин вибирають інший маршрут. Зазвичай для вибору маршруту використовують досить складний набір алгоритмів, які працюють більш-менш незалежно, хоча й обмінюються інформацією. Його складність обумовлено низкою причин: по-перше, маршрутизація потребує координації роботи всіх вузлів підмережі, а не лише однієї пари модулів, як, наприклад, у протоколах рівнів лінії передавання даних та транспортного протоколу; по-друге, система маршрутизації має справлятися з виходами з ладу ліній або вузлів шляхом переспрямування трафіка й оновлення баз даних, що використовують системою; по-третє, задля досягнення найкращих характеристик алгоритм маршрутизації може змінити маршрути, коли деякі ділянки мережі стають перевантаженими [16]. Кожен із відомих типів протоколів маршрутизації Link-State та Distance-Vector має свої позитивні та негативні властивості. До переваг алгоритмів маршрутизації Distance-Vector, безсумнівно, можна зарахувати простоту їхньої реалізації, тоді як алгоритми Link-State забезпечують кращі експлуатаційні характеристики. Недоліком протоколів маршрутизації Distance-Vector, є неефективне використання пропускної спроможності каналів під час передавання маршрутної інформації, алгоритми Link-State є погано пристосованими для побудови маршрутів у мережах із нестабільними каналами. Гібридні протоколи маршрутизації, що належить EIGRP, поєднують у собі позитивні властивості протоколів Link-State та Distance-Vector і максимально вільними від своїх недоліків. Основні проблеми будь-якого протоколу маршрутизації пов'язано з нестабільністю маршрутів. У будь-якій системі завжди є хоча б один

елемент, що нестабільно працює, або елемент, що виходить із ладу в найбільш невідповідний момент. Чим більшою є система, тим вищою ймовірність відмови чи перебою. Відмови та перебої в системах передавання даних призводять до зникнення маршрутів. Завдання алгоритму маршрутизації наразі полягає в тому, щоб якнайшвидше визначити обхідний маршрут, замість того, що вийшов із ладу. Саме в такі моменти деякі алгоритми маршрутизації виявляють себе не найкращим чином – пошук резервного маршруту й перемикання на нього може виконуватися недозволено довго, у гіршому разі несправність каналу може взагалі призвести до виникнення циклічних маршрутів, унаслідок чого частину переданої через мережу інформації буде безповоротно втраченою. Залежно від місця, де виконують маршрутні обчислення, та виду необхідної інформації про стан мережі, розрізняють розподілені, централізовані та ієрархічні стратегії, а також кооперовані й ізольовані.

У разі *ізольованої стратегії* маршрутні обчислення виконують кожним вузлом незалежно на основі наявної локальної інформації. Між вузлами не здійснюють обміну маршрутною інформацією та інформацією про стан мережі. Необхідною інформацією для вузла можуть слугувати довжини його локальних черг. Проста стратегія, спрямована на мінімізацію затримки, полягає у виборі маршрутів із найкоротшою чергою (шляхом). Оскільки в мережі з комутацією пакетів середня затримка пакета залежить від інтенсивності навантаження, яке, своєю чергою, визначено вибраними маршрутами, то завдання маршрутизації є набагато складнішим за завдання вибору найкоротшого шляху.

За *централізованої кооперованої стратегії* мережевий маршрутний центр збирає глобальну інформацію про стан мережі, обчислює маршрути мінімальної затримки та розподіляє маршрутні таблиці всім вузлам.

Розподілена кооперована стратегія передбачає, що маршрутні обчислення виконують ряд вузлів на основі інформації про стан мережі, якими вони обмінюються.

Маршрутизація взаємодіє з управлінням потоками пакетів у мережі. Алгоритм управління в разі перевантаження мережі за сигналами алгоритму маршрутизації обмежує доступ пакетів у мережу. Якщо алгоритм маршрутизації забезпечує меншу затримку пакетів, то алгоритм управління дає доступ у мережу більшому потоку пакетів. Отже, процес маршрутизації робить вплив на такі важливі характеристики мережі, як вірогідність утрати й доставляння пакетів не за адресою, час затримки пакета та пропускну спроможність.

Основні вимоги до алгоритмів маршрутизації формують у такий спосіб, щоб забезпечити мінімум середнього часу доставляння пакетів, швидко адаптуватися до змін навантаження й топології, перешкоджати зацикленню пакетів, бути достатньо простим і потребувати мінімальних апаратних та програмних витрат.

Теоретичною базою для побудови системи управління мережею ухвалено концепцію побудови мережі управління TMN. Цей підхід надає методологічну основу для реалізації інтегрованого управління мережами, які відрізняються за структурою, складом та обсягом переданої інформації.

Загалом під системою управління мережею розуміють систему, що автоматизовано виконує функції з управління мережею на основі комплексу інформаційних технологій із планування, технічного обслуговування, експлуатації, оперативного й адміністративного управління мережами та послугами, які надають.

Організаційно кожна система управління мережами є територіально рознесеною ієрархічною структурою. Топологію мереж управління в межах зони відповідальності оператора, розміщення центрів управління, кількість рівнів ієрархії сукупно визначають, відповідно до особливостей керованих мереж, їхнього призначенням, розмірів, розгалуженості, організації технічних засобів.

Отже, структура управління мережею становить складну багаторівневу структуру з різноманітними функціональними зв'язками й може містити від двох до чотирьох рівнів.

Кількість рівнів і конкретний зміст функцій центрів управління на всіх рівнях залежать від багатьох факторів і їх мають визначати у процесі розроблення системи управління.

7.4. Моделювання процесу мультиоб'єктного управління ресурсами адаптивної комп'ютерної мережі

Найбільш ефективне управління мережею забезпечено шляхом адаптації під час вирішення різних завдань, тобто використання адаптивних систем управління (СУ).

Необхідність у використанні таких систем виникає у зв'язку з ускладненням вирішуваних завдань, відсутністю практичної можливості детального вивчення процесів у керованих об'єктах. *Адаптивними* називають систе-

ми, у яких брак апріорної інформації заповнюють шляхом повного (порівняно зі статичними системами) використання поточної інформації.

Ефекти налаштування до змін умов зовнішнього середовища в адаптивних системах досягають через те, що частина функцій зі здобуття, опрацювання й аналізу інформації, якої бракує, про керований процес здійснюють не на попередній стадії, а самою системою у процесі роботи. Це сприяє більш повному використанню інформації про поточний стан об'єкта управління.

Залежно від способів реалізації контрольованих змін у процесі нормальної експлуатації системи, можна навести таку класифікацію адаптивних систем: самоналагоджувальні системи, системи з адаптацією в особливих умовах і системи з навчанням (рис. 7.1).

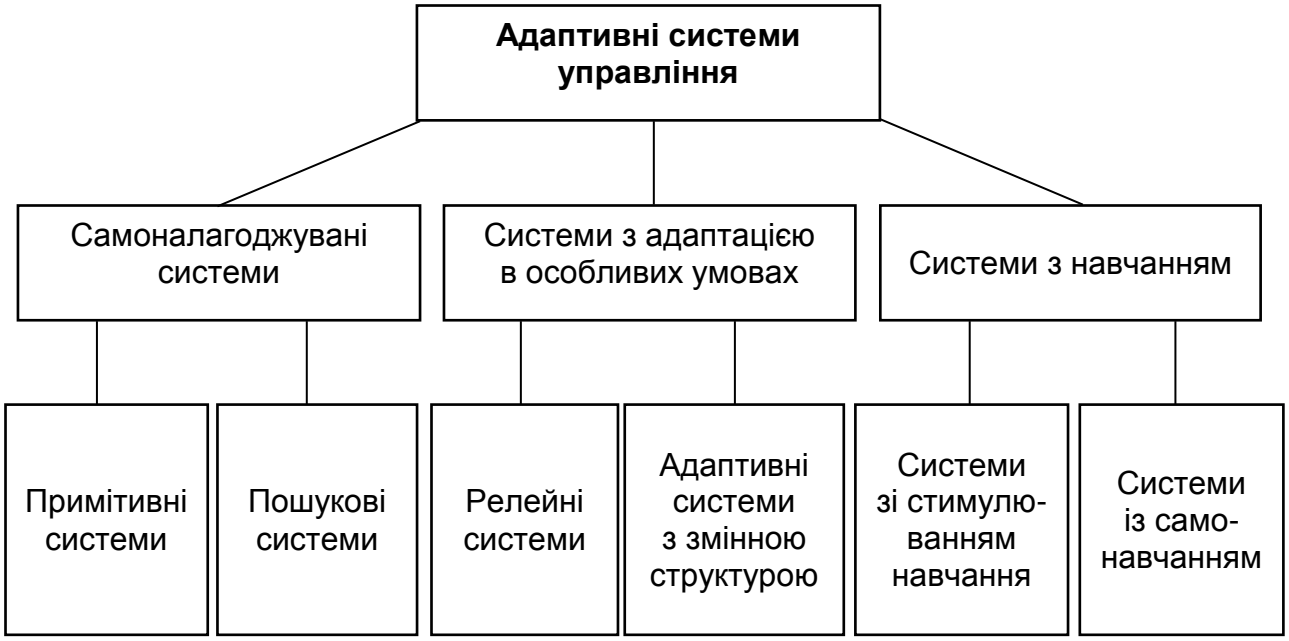


Рис. 7.1. **Класифікація адаптивних СУ**

Самоналагоджувані системи характеризуються наявністю спеціальних контурів самоналагоджування, за допомогою яких оцінюють динамічні та статичні властивості системи та формують такі контрольовані впливи, що система мимовільно наближається до певного стандарту, їх часто задано математично у вигляді критерію якості функціонування. Водночас контур самоналагоджування слугує для зміни параметрів або структури основного контуру, із метою забезпечення заданого критерію якості управління.

Основне завдання самоналагоджуваних систем полягає в підтриманні заданого у вигляді функціоналу заходу якості системи поблизу екстремального значення в разі змін у процесі функціонування системи вхідних керівних впливів, а також динамічних характеристик об'єкта.

Системи з навчанням характеризуються наявністю спеціальних процесів навчання, які будуть полягати в поступовому накопиченні, запам'ятовуванні й аналізі інформації про поведінку системи та зміну законів функціонування, залежно від набутого досвіду. До процесу навчання доводиться вдаватися тоді, коли не тільки малий обсяг апріорних відомостей про об'єкт, але й відсутня можливість установлення детальних причинно-наслідкових зв'язків у структурі самої системи через її труднощі. Системи з навчанням є найбільш складними.

Аналіз наведених алгоритмів адаптації в комп'ютерних мережах (КМ) та особливостей структури системи управління цими мережами показує, що адаптивні алгоритми у КМ може бути використано як під час управління мережею, так і конкретними окремими процесами. З огляду на особливості алгоритмів адаптації, слід зазначити, що під час управління КМ можна використовувати як самоналагоджувані алгоритми, так і алгоритми систем з навчанням. Кількість процесів у КМ, для управління якими може бути використано зазначені алгоритми, є досить великою. Крім того, часто виникає необхідність і в оптимізації розподілу ресурсів за кількома критеріями, які можуть бути суперечливими. Отже, у процесі управління інформаційним обміном потрібно вирішення великої кількості складних багатокритеріальних завдань.

Як показано раніше, складність завдання динамічного управління призводить до необхідності у її розподіл на окремі завдання. Такий розподіл є можливим тому, що ряд процедур у процесі управління можна виконувати паралельно. Так, наприклад, завдання контролю за станом мережі можна виконувати паралельно із завданням маршрутизації. Це дозволить скоротити час ухвалення рішення та спростити цей процес шляхом скорочення простору пошуку найкращого варіанта управління. Необхідність у паралельному рішенні завдань управління виникає у процесі управління багатьма об'єктами (за мультиоб'єктного управління).

З огляду на перелічені раніше особливості процесу інформаційного обміну в мережах, а також аналіз особливостей, що виникають у процесі управління цим процесом, можна зробити висновок про те, що ефективний

розподіл ресурсів мережі є незадовільно формалізованим завданням вибору найкращого варіанта рішення з управління об'єктом з урахуванням безлічі різних факторів в умовах невизначеності.

У зв'язку з можливістю та доцільністю розподілу загального завдання динамічного управління на окремі підзавдання, про що зазначено раніше, властивістю автономності (здатністю функціонувати самостійно без прямих керівних впливів) елементи системи динамічного управління мають самостійно й паралельно вирішувати свої приватні завдання. Отже, узагальнюючи сказане раніше, необхідно зазначити такі особливості системи управління адаптивними комп'ютерними мережами:

ця система має бути розподіленою у просторі;

у процесі функціонування система має бути адаптованою з використанням методів штучного інтелекту;

система загалом, а також її елементи мають бути автономними.

Зазначеним вимогам відповідає перспективна адаптивна багатоагентна система (МАС), що становить сукупність взаємопов'язаних агентів, що спільно функціонують.

Технологія МАС нині бурхливо розвивається та претендує на одну із провідних ролей у межах інтелектуальних інформаційних технологій. Використання МАС дозволяє таке:

подати завдання управління у вигляді набору підзавдань і в такий спосіб приховати її складність;

розподілити програмні компоненти, відповідно до вимог користувачів;

забезпечити незалежність програмних компонентів.

Розподіл агентів для вирішення різних завдань мають здійснювати, відповідно до таких принципів:

кількість агентів не має бути великою;

кожен агент має вирішувати конкретне завдання;

інтерфейс між агентами має бути простим;

зміни, що вносять у програмний компонент одного агента, не мають впливати на програмні компоненти інших агентів;

забезпечення можливості паралельного вирішення деяких завдань різними агентами;

легкість розв'язання між агентами можливих конфліктів;

розподіл за завданнями мають забезпечувати на стандартній моделі взаємодії відкритих систем.

Відомі деякі варіанти побудови МАС для інших предметних галузей [44; 46]. Однак конструктивної загальної методики розроблення таких систем сьогодні немає. У кожному індивідуальному випадку МАС розробляють, з огляду на особливості предметної галузі.

Є різні класифікації МАС: гомо- і гетерогенні, із комунікацією між агентами та без неї. Під *гетерогенною* розуміють таку МАС, у якій агенти мають різні цілі, способи сприйняття та впливу на об'єкт управління. Оскільки управління в мережах зв'язку засновано на ієрархічній моделі протоколів, то агенти на кожному рівні вирішують різні завдання.

Тому для вирішення завдання динамічного управління адаптивними КМ структура МАС має становити ієрархічну гетерогенную систему взаємодійних між собою агентів.

У цьому разі МАС можна уявити як сукупність агентів, що мають спільну мету (Ö). Тоді модель МАС можна подати в такому вигляді:

$$M_{(MAC)} = \left\{ \ddot{O}, `A, `Y, F \right\}, \qquad (7.1)$$

де `A – сукупність агентів;

`Y – критерій якості функціонування МАС;

F – механізм взаємодії.

Модель агента можна подати у такому вигляді:

$$M_{(A_i)} = \left\{ \ddot{O}_i, I_i, U_i, `Y_{O'A_i}, Z_i \right\}, \qquad (7.2)$$

де $\ddot{O}_i$ – мета функціонування і-го агента;

I_i – інформація, яку сприймає агент A_i;

U_i – дії, що управляють, які агент може виконати;

$`Y_{O'A_i}$ – критерій управління, яким керується агент під час вибору керівного впливу;

Z_i – знання агента про середовище, у якому він міститься.

Технічною базою системи динамічного управління є центри (вузли) комутації. Тому елементи МАС – агенти – мають розташовуватися на центрах комутації та брати участь в управлінні відповідними ресурсами й елементами мережі (NE). Насправді, завдання агентів полягає не в тому,

щоб вони замінили наявні алгоритми розподілу ресурсів мережі, а в тому, що вони мають управляти цими алгоритмами, із метою підвищення їхньої ефективності, тобто здійснювати такі функції, які нині в системах динамічного управління наявних мереж не реалізовано. З огляду на зазначене раніше, пропонують використовувати багаторівневу ієрархічну структуру багатоагентної системи динамічного управління (рис. 7.2).

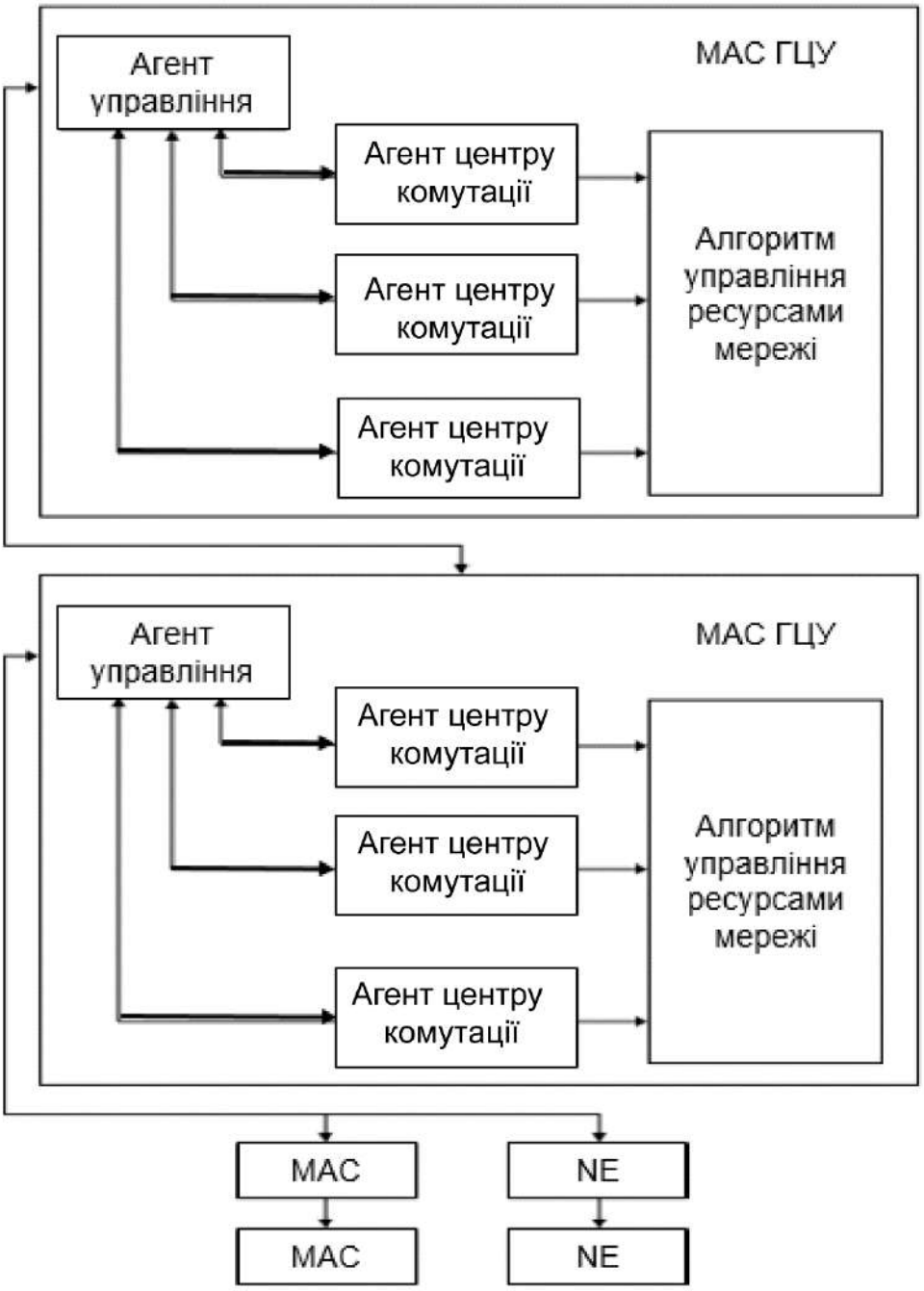


Рис. 7.2. **Структура багатоагентної системи динамічного управління**

На головному центрі управління (ГЦУ) розташовується агент, що управляє мережею. Його завдання полягає в тому, що він установлює найбільш важливі вимоги до якості обслуговування тієї чи тієї інформації в мережі, у разі необхідності за допомогою керівних впливів коригує дії інших агентів, а також усуває конфліктні ситуації, що виникають у системі.

На інших вузлах розташовуються агенти, що управляють відповідними регіональними центрами управління (РЦУ). Ці агенти вирішують завдання управління розподілом ресурсів у центрах комутації. На кожному центрі комутації, крім керівного агента, розташовуються інші агенти, зазначені на розглянутій схемі як агенти. Вони вирішують приватні завдання та беруть участь в управлінні розподілом окремих видів ресурсів. Функції агента, що управляє ЦК, насправді є аналогічними функціям агента, що управляє мережею, але тільки поширюються в масштабах вузла.

Отже, на кожному центрі комутації розташовано сукупність взаємопов'язаних агентів, що утворюють багатоагентну систему динамічного управління центром комутації – МАС ЦК.

Багатоагентні системи окремих ЦК є підсистемами багатоагентної системи динамічного управління всією мережею.

Отже, одним зі способів підвищення ефективності управління в адаптивній комп'ютерній мережі є використання адаптивної багатоагентної системи динамічного управління з елементами штучного інтелекту. У цьому разі загальне завдання динамічного управління розподіляють на окремі підзавдання управління ресурсами мережі, для вирішення яких можна використовувати оптимальні адаптивні алгоритми. Кожне підзавдання характеризується своєю моделлю.

Відповідно до зазначених раніше особливостей адаптивної комп'ютерної мережі та сукупності її моделей інформаційних процесів, слід зазначити, що комп'ютерна мережа є досить складною інформаційною структурою, яка забезпечує взаємодію користувачів на основі єдиної системної ідеології, що реалізують у мережевих протоколах. Функціонування такої мережі, як і будь-якої складної системи управління, характеризується взаємодією зовнішнього та внутрішнього середовища (див. рис. 7.2).

Вплив зовнішнього середовища відображено взаємодією вхідного інформаційного потоку $\sum\lambda_{вх_i}(t)$ і випадкових факторів, що заважають $\sum_{i=1}^{N} U_{\pi_i}(t)$. Внутрішнє мережеве середовище містить системи внутрішньої та зовнішньої взаємодії, а також систему опрацювання й передавання (ОП).

Внутрішню взаємодію може бути описано за допомогою квадратної матриці зв'язності абонентів (вузлів комутації) розміром $N \times N$ (N – кількість абонентів, що є джерелами та споживачами інформації).

Зовнішню взаємодію здійснюють за допомогою комплексів засобів доступу абонентів (КЗДА) через $N_{вх.}$ входів і $N_{вих.}$ виходів. Система опрацювання та передавання описує основні структурні елементи мережі. Такими елементами є вузли комутації (вузли зв'язку), концентратори, абонентські пункти, шлюзи, тракти передавання тощо. Конфігурацію мережі з її топологією задано таблицею зв'язності із сусідніми вузлами та маршрутними таблицями.

Система управління мережею містить засоби контролю (К) та управління (У), які здійснюють збирання інформації про стан мережі, опрацювання цієї інформації, вирішення завдання управління, відображення та контроль за приведенням рішення у виконання на основі застосовуваних протоколів.

Система управління може впливати як на засоби доступу (U_1), так і систему опрацювання й передавання (U_2), шляхом впливу на параметри протоколів. Керівні взаємодії залежать від важливості (цінності) інформації про стан мережі та окремих її елементів $I_0(t)$, яка надходить на засоби управління та відображення від засобів контролю. Деяка додаткова інформація може надходити й від інших зовнішніх джерел $\left(I(t)\right)$.

Вихідний ефект мережі $\left(\lambda_{вих.}(t)\right)$ характеризує потік інформації, що надходить користувачам, із певними параметрами. Тоді структура мережі як об'єкта управління буде мати вигляд, показаний на рис. 7.3.

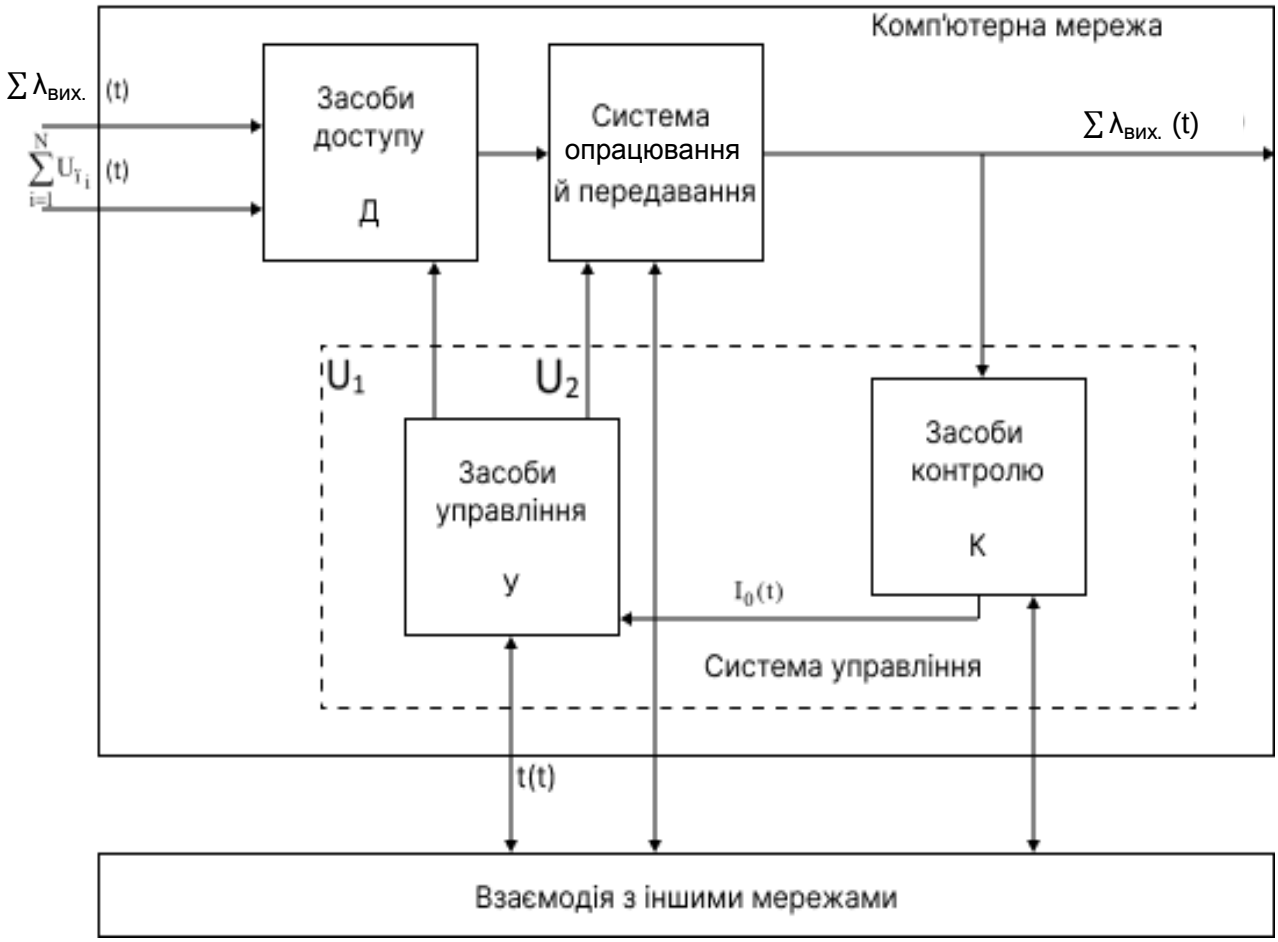


Рис. 7.3. **Загальна структура комп'ютерної мережі як об'єкта управління**

У результаті вихідний ефект мережі може бути записано в такому вигляді:

$$\lambda_{вих.}(t) = F\left[\lambda_{вх.}(t), U_{ї}(t), U_1(I_0(t), I(t)), U_2(I_0(t), I(t))\right], \qquad (7.3)$$

де F – оператор досліджуваної мережі, визначений алгоритмами передавання, опрацювання інформації та ухвалення рішення у процесі управління мережею.

Метою управління є забезпечення інтенсивності вихідного інформаційного потоку $\lambda_{вих.}(t)$, відповідно до вхідних $\lambda_{вх.}(t)$ і необхідних імовірнісно-часових характеристик.

У загальному випадку засоби взаємодії, передавання та опрацювання загалом характеризуються своєю архітектурою, під якою розуміють

сукупність функціональної, топологічної, інформаційної, організаційної, алгоритмічної, програмної та технічної структур.

Ці структури містять повний набір структурно упорядкованих мережевих елементів і систему мережевих протоколів. Сукупність протоколів однозначно визначає правило спільної роботи (взаємодії) мережевих елементів, із метою реалізації різних режимів функціонування, забезпечених цією мережею.

Уведений раніше оператор F може бути досить складним і зазда-легідь невідомим. Із метою деякого спрощення врахуймо, що мережу синтезовано з таким розрахунком, щоб один і той самий вплив зовніш-нього середовища не порушував функціонування кількох двополюсних мереж. У цьому разі вихідний ефект в операторній формі може бути оцінено за таким виразом:

$$\lambda_{вих.}(t) = \sum_{i=1}^{\gamma} F_i' \left[\lambda_{вх.}(t), U_{ï_i}(t), U_{1_i}(I_0(t), I(t)), U_{2_i}(I_0(t), I(t)) \right], \qquad (7.4)$$

де γ – кількість двополюсних мереж.

Управління засобами доступу, опрацювання й передавання можна здійснювати незалежно. Процес управління засобами доступу має бути залежним від зовнішніх факторів, що заважають. У цьому разі вираз (7.4) може бути записано в такому вигляді:

$$\lambda_{вих.}(t) = \sum_{i=1}^{\gamma} F_i' \left[\lambda_{вх.}(t), U_{ï_i}(t), U_{1_i}(I_0(t), I(t)) \right] + \sum_{i=1}^{\gamma} F_i'' \left[\lambda_{вх.}(t), U_{ï_i}(t), U_{2_i}(I_0(t), I(t)) \right]. \qquad (7.5)$$

Відповідно до зазначеного раніше, оператора F_i' визначають ефектом управління з урахуванням матриці зв'язності абонентів (внутрішньої взаємодії) та ефектом управління, що впливає на входи та виходи мережі (зовнішнього управління). Операторів F_i' визначають ефектом управління двополюсними мережами, вузлами комутації тощо.

Для визначення керівних впливів, а також операторів і є необхідною база знань, реалізована за допомогою різних моделей як мережі загалом, так і її елементів.

Модель – це деякий образ оригіналу, що частково відображає його сутність. Оригінал зазвичай має безліч різних властивостей (атрибутів). Часто вони тісно пов'язані один з одним і цим можуть маскувати сутність

оригіналу. Тому модель мають розробляти та використовувати такий математичний апарат, щоб виділити найбільш головні особливості оригіналу й абстрагуватися від другорядних деталей.

Це дає можливість, досліджуючи модель, аналізувати основні характеристики оригіналу. Істотне значення під час моделювання має цільове призначення моделі, що розробляють.

Для різних цілей вибирають різні атрибути й різний математичний апарат. Водночас ставлять і різні вимоги до математичного апарату. Однак є й загальні вимоги до моделей та апарату моделювання.

Такими вимогами є такі:

повнота моделі й апарату моделювання має надавати користувачеві можливість здобуття необхідного набору оцінок характеристик системи з необхідною адекватністю, точністю та достовірністю;

інформаційне забезпечення має давати можливість ефективного функціонування моделі з базою даних;

структура моделі має допускати можливість заміни, додавання та вилучення деяких її частин без перероблення всієї моделі;

гнучкість моделі має давати можливість аналізувати різні ситуації в разі зміни структури, алгоритмів і параметрів системи;

тривалість розроблення й реалізації моделі має бути по можливості мінімальною;

програмні та технічні засоби мають забезпечувати ефективну реалізацію моделі.

З огляду на особливості досліджуваної мережі, до моделі й математичного апарату моделювання ставлять такі додаткові вимоги:

адекватно відображати процеси, що відбуваються під час вирішення завдань управління, передавання в адаптивних розподілених телекомунікаційних мережах з урахуванням їхніх особливостей;

модель має забезпечувати можливість оцінювати такі основні ймовірнісно-часові характеристики, як середнє значення та дисперсія часу рішення завдання, імовірність виконання завдання з помилкою, залежно від умов та особливостей застосовуваного методу управління;

давати користувачеві можливість досліджувати процеси з урахуванням цінності переданої інформації та її старіння;

забезпечувати оцінювання та вивчення впливу різних етапів технології управління;

визначати ймовірнісно-часові характеристики у процесі мультиоб'єктного управління;

забезпечувати вирішення завдання управління як за наявності повної інформації про стан мережі, так і в умовах невизначеності;

давати можливість користувачу в умовах невизначеності визначати переважну альтернативу вибору варіанта ухваленого рішення.

З огляду на зазначені вимоги й результати порівняльного аналізу математичного апарату моделювання, можна вважати, що основним апаратом моделювання слід вибрати ймовірнісно-часові графи та продуктивні функції.

7.5. Висновки

Здійснено аналіз особливостей функціонування системи управління сучасних інформаційних мереж.

Показано, що функціонування цієї системи ґрунтується на застосуванні великого арсеналу методів динамічного управління, водночас розглянуто організацію процесу контролю за обміном даними в мережах із комутацією пакетів й управління ним, переваги та недоліки алгоритмів маршрутизації пакетів у мережах, а також можливості ефективного моделювання процесу мультиоб'єктного управління ресурсами адаптивної комп'ютерної мережі.

Запропоновано загальний підхід до моделювання, визначеного ефектом управління, з урахуванням матриці зв'язності абонентів (внутрішньої взаємодії об'єктів у мережі) й ефектом управління, що впливає на входи та виходи мережі (зовнішнього управління).

Водночас досліджено, що ефективність роботи мережі характеризується потоком інформації, що надходить користувачам, із певними характеристиками: сукупністю функціональної, топологічної, інформаційної, організаційної, алгоритмічної, програмної та технічної структур. Ці структури містять повний набір структурно упорядкованих мережевих елементів і систему мережевих протоколів. Сукупність протоколів однозначно визначає правило спільної роботи (взаємодії) мережевих елементів, із метою реалізації різних режимів функціонування, забезпечених цією мережею.

Розділ 8
Алгоритми асимптотично оптимальної кусково-лінійної інтерполяції плоских параметричних кривих

8.1. Вступ і формулювання завдання

Відтворення безперервних геометричних об'єктів, як-от криві лінії та поверхні, на комп'ютерах, верстатах чи принтерах потребує дискретизації, тобто подання у вигляді кінцевої множини точок (вузлів). Ця множина надалі перетворюється на пікселі, або з'єднується простішими лініями (найчастіше, прямими) [7]. Звичайно, водночас виникає проблема оптимізації кількості вузлів та їхнього розташування вздовж кривої чи поверхні, що відтворюється. Отже, розроблення ефективних алгоритмів відтворення кривих ламаними з урахуванням точності апроксимації й оптимізації кількості вузлів ланок становить актуальну наукову проблему.

Метою цієї роботи є вдосконалення методів використовуваних алгоритмом асимптотично оптимального алгоритму кусково-лінійної інтерполяції плоских параметричних кривих на основі дослідження впливу параметрів під час моделювання похибок апроксимації реальних кривих ліній.

Розробленню алгоритмів кусково-лінійного відтворення кривих присвячено численну множину досліджень. Ці дослідження можна розподілити на роботи з інтерполяції кривих, коли вузли ламаної розміщено на первісній кривій, і дослідження з апроксимації кривих, під якими будемо розуміти розташування цих вузлів і самої ламаної в межах допуску, що визначено точністю відтворення.

Загальні алгоритми асимптотичної оптимальної інтерполяції й апроксимації кривих технологічними лініями (зокрема ламаними) було розглянуто в роботі [12]. Особливістю такого підходу є те, що він дає ламану з найменшою кількістю ланок. Проаналізувавши ці дослідження, можна виділити ряд невирішених у цих дослідженнях питань, а саме: обґрунтування кількості точок попередньої дискретизації кривої, вибір методу чисельного інтегрування під час обчислення значень функції регулятора

вузлів і методу інтерполяції значень цієї функції, обґрунтування обчислювальних формул дискретних аналогів похідних під час обчислення значень інтегральної функції та ін.

8.2. Основна частина

8.2.1. Асимптотично оптимальні моделі інтерполяції кривих ламаними в гаусдорфовій метриці

Теоретичною базою досліджень, викладених у роботі, є монографія [7], де у формі теорем було доведено асимптотичну оптимальність моделей для загальних випадків подання кривих.

Далі будемо вважати, що криві, які відтворюються, мають подання в параметричній векторній формі у вигляді єдиної дуги:

$$p = p(t)\ (x = x(t), y = y(t)) \tag{8.1}$$

або складаються з декількох дуг зі спільною параметризацією, наприклад, мають форму сплайна:

$$S(t) = \sum_{j=1}^{k} a_j B_j(t)\,. \tag{8.2}$$

Асимптотично оптимальною в гаусдорфовій метриці, згідно з доведеним у [7], уважають інтерполяцію плоскої кривої ламаною, що ґрунтується на виборі вузлів на основі репараметризації кривої за такою схемою:

1) визначають кількість ланок інтерполяції за такою формулою:

$$m = \left[\frac{1}{\sqrt{8\varepsilon}} \int_0^T \Phi(t)dt \right] + 1, \tag{8.3}$$

де $[a]$ – ціла частина числа a;

ε – похибка інтерполяції, якій відповідає допустима гаусдорфова відстань між кривою та ламаною;

t – параметр кривої з областю зміни $[0;T]$;

$\Phi(t)$ – функція розподілу, що має такий вигляд:

$$\Phi(t) = \sqrt{\frac{|x''(t)y'(t) - x'(t)y''(t)|}{\sqrt{x'(t)^2 + y'(t)^2}}}; \qquad (8.4)$$

2) значення параметра кривої t_i, яким відповідають вузли інтерполяції, визначають із таких умов:

за $\Phi(t) \neq 0, t \in [0;T]$

$$\int_0^{t_j} \Phi(t)dt = \frac{j}{m}\int_0^T \Phi(t)dt, (j = 0,1,...,m); \qquad (8.5)$$

за наявності нульових значень функції $\Phi(t)$ до неї додають величину m^{-a} і рівняння розподілу перетворюється на таке:

$$\int_0^{t_j} (\Phi(t) + m^{-\alpha})dt = \frac{j}{m}\int_0^T (\Phi(t) + m^{-\alpha})dt, (j = 0,1,...,m), \qquad (8.6)$$

де α – стала, що згідно із [10] може набирати значення в межах від 0 до 2/3.

Оскільки права частина рівняння (8.5) або (8.6) становить монотонну зростальну дискретну послідовність значень, то у граничному випадку цю послідовність можна розглядати як змінну s, а вираз (8.5) або (8.6) – як рівняння репараметризації. Якщо припустити, наприклад:

$$\psi(t) = \int_0^t (\Phi(u) + m^{-a})du, \qquad (8.7)$$

то залежність між параметрами буде мати такий вигляд:

$$t = \psi^{-1}(s), \qquad (8.8)$$

де ψ^{-1} – функція, зворотна до ψ.

Природно, такий підхід є можливим тільки, коли ψ монотонно зростає. До того ж найчастіше інтеграл правої частини (8.7) не має кінцевого виразу. Це призводить до необхідності в дискретизації значень інтегральної функції на основі чисельного інтегрування правої частини (8.7) та подальшої інтерполяції цих значень.

8.2.2. Визначення вузлів асимптотично оптимального розподілу кривої

Для інтерполяції значень інтегральної функції розподілу може бути запропоновано такий алгоритм.

Найперше, визначмо кількість інтервалів первісної дискретизації інтеграла правої частини рівнянь (8.5) або (8.6), яку позначмо через N.

Далі визначмо крок рівномірної сітки дискретизації $h = \frac{T}{N}$ та набір точок кривої

$$P_i = p\left(\frac{i}{N}\right), (i = 0,1,...,N), \tag{8.9}$$

що відповідають дискретному ряду значень параметра кривої $\Delta_N^t \left\{ t_i = 0, \frac{1}{N}, \frac{2}{N}, ..., T \right\}$.

Для обчислення значень послідовності вузлів $\Delta_m^t = \left\{ t_{j,m}^* \right\} (j = 0,1,...,m)$ асимптотично оптимального розподілу кривої одним із відомих чисельних методів інтегрування визначмо послідовність значень $s_i = \Psi(t_i)$, $t_i \in \Delta_N^t, (i = 0,1,...,N)$ функції (8.7) у вузлах первісної рівномірної сітки по $t - \Delta_N$. І далі, інтерполюючи ці значення між вузлами, обчислімо значення зворотної функції $t_j = \Psi^{-1}(s_j)$ за рівномірного вже, згідно із правою частиною (8.5) або (8.6), розподілу значень s_j від 0 до $s_N = \Psi(t_N)$.

Водночас кількість ланок m асимптотично оптимального розподілу, згідно з (8.3), набуде такого значення:

$$m = \left[\frac{S_N}{\sqrt{8\varepsilon}} \right] + 1. \tag{8.10}$$

Значення вузлів рівномірної сітки по s – $\Delta_m^S = \{S_j, j = 0,1,...,m\}$ обчислімо за таким виразом:

$$S_j = \frac{jS_N}{m}. \tag{8.11}$$

Обчислімо значення параметра кривої $t_{j,m}^*$, які утворюють асимптотично оптимальну послідовність вузлів. Найпростішим таким підходом є лінійна інтерполяції функції $s = \Psi(t)$ за її значеннями на первісній рівномірній сітці Δ_N^t. Визначаючи перетин ліній сітки $s = s_j$ із побудованим на сітці Δ_N графіком $\Psi(t)$, маємо відповідні значення послідовності Δ_m^t із такого виразу оберненої залежності:

$$t_j^* = t_i + \frac{T(S_j - S_i)}{N(S_{i+1} - S_i)} \text{ за } S_i \le S_j \le S_{i+1}\left(t_i \in \Delta_N^t\right). \tag{8.12}$$

Альтернативний підхід, який можна запропонувати, – це інтерполяція між вузлами сітки Δ_N^t лінійним раціональним B-сплайном. Засади такого підходу було запропоновано в роботі [46]. Цей підхід дозволяє додатково забезпечити перший порядок гладкості у вузлах інтерполяції. Пропонований метод кожному інтервалу інтерполяції $[t_i, t_{i+1}]$ ставить у відповідність LRBS криву з відкритим двоінтервальним вузловим вектором: $\left[t_i, t_i, t_{i+1/2}, t_{i+1}, t_{i+1}\right]$, де $t_{i+1/2}$ – деяке внутрішнє значення параметра, якому можна зіставити значення коефіцієнта $\lambda_{i,i+1}$, тоді $t_{i+1/2} = (1-\lambda_{i,i+1})t_i + \lambda_{i,i+1}t_{i+1}$. Монотонні зростальні лінійні раціональні функції, що інтерполюють функцію між значеннями $s_i = \Psi(t_i)$ та $s_{i+1} = \Psi(t_{i+1})$ на кінцях інтервалу, будуть мати такий вигляд:

за $t_i \le t < t_{i+1/2}$

$$S_{i,i+1/2} = \frac{W_i S_i\left(t_{i+1/2} - t\right) + W_{i+1/2} S_{i+1/2}\left(t - t_i\right)}{W_i\left(t_{i+1/2} - t\right) + W_{i+1/2}\left(t - t_i\right)}; \tag{8.13}$$

за $t_{i+1/2} \le t < t_{i+1}$

$$S_{i+1/2,i+1} = \frac{W_{i+1/2} S_{i+1/2} (t_{i+1} - t) + W_{i+1} S_{i+1} \left(t - t_{i+1/2}\right)}{W_{i+1/2} (t_{i+1} - t) + W_{i+1} \left(t - t_{i+1/2}\right)}, \qquad (8.14)$$

де $S_{i+1/2}$ – коефіцієнт, що визначає значення функції у проміжній точці $t_{i+1/2}$.

Маючи значення інтерпольованої функції та її похідних (S'_i, S'_{i+1}) на кінцях інтервалу, коефіцієнти LRB-сплайна визначають у такій послідовності:

значення одного з вагових коефіцієнтів, згідно із [29], можна взяти довільно: він буде відігравати роль множника масштабу для інших вагових коефіцієнтів, задля визначеності призначмо для довільного вибору перший коефіцієнт – W_i;

тоді останній ваговий коефіцієнт маємо з такого виразу [29]:

$$W_{i+1} = \sqrt{\frac{S'_i}{S'_{i+1}}} W_i; \qquad (8.15)$$

значення коефіцієнта λ, який буде визначати внутрішнє значення параметра $t_{i+1/2}$, можна визначити, згідно з таким виразом:

$$\lambda_{i,i+1} = \begin{cases} \dfrac{S'_i - \left(S'_i S'_{i+1}\right)^{1/2}}{S'_i - S'_{i+1}} \text{ за } \quad S'_i \neq S'_{i+1}; \\ \dfrac{1}{2} \text{ за } \quad S'_i = S'_{i+1}; \end{cases} \qquad (8.16)$$

значення $S_{i+1/2}$ функції у проміжній точці, визначають з умови гладкості сплайна [29]:

$$S_{i+1/2} = \frac{\left(1 - \lambda_{i,i+1}\right) W_i S_i + \lambda_{i,i+1} W_{i+1} S_{i+1}}{\left(1 - \lambda_{i,i+1}\right) W_i + \lambda_{i,i+1} W_{i+1}}; \qquad (8.17)$$

вираз для визначення проміжного вагового коефіцієнта має такий вигляд:

$$W_{i+1/2} = \left[\left(1-\lambda_{i,i+1}\right)W_{i+1}S'_{i+1} + \lambda_{i,i+1}W_iS'_i\right]\frac{t_{i+1}-t_i}{S_{i+1}-S_i}. \qquad (8.18)$$

Розв'язуючи рівняння інтерполяції щодо t та підставивши замість правої частини значення s_j послідовності Δ_m^S, маємо таке:

за $S_i \le S_j \le S_{i+1/2}$

$$t_j^* = \frac{W_it_{i+1/2}\left(S_j-S_i\right) + W_{i+1/2}t_i\left(S_{i+1/2}-S_j\right)}{W_i\left(S_j-S_i\right) + W_{i+1/2}\left(S_{i+1/2}-S_j\right)}; \qquad (8.19)$$

за $S_{i+1/2} \le S_j \le S_{i+1}$

$$t_j^* = \frac{W_{i+1/2}t_{i+1}\left(S_j-S_{i+1/2}\right) + W_{i+1}t_{i+1/2}\left(S_{i+1}-S_j\right)}{W_{i+1/2}\left(S_j-S_{i+1/2}\right) + W_{i+1}\left(S_{i+1}-S_j\right)}. \qquad (8.20)$$

8.2.3. Оцінювання результатів відтворення кривих

Перед тим як перейти до обговорення результатів побудови апроксимальних ламаних для конкретних кривих ліній за різними варіантами алгоритму, розгляньмо питання оцінювання забезпечення необхідної точності наближення ламаної до кривої.

Для можливості оцінювання результатів відтворення та порівняння їх із максимально допустимою відстанню ε між кривою та ламаною, що її апроксимує, може бути застосованим метод еквідистант до кривої [7]. Сутність цього методу полягає в побудові разом із ламаною внутрішньої та зовнішньої еквідистант до відтворюваної кривої за їхніми загальними параметричними рівняннями.

Метод еквідистант дає можливості візуального контролю за поведінкою ламаною щодо меж ε – допустимого коридору вихідної кривої. Однак суттєвими недоліками такого підходу є відсутність кількісної оцінки якості наближення, складність такого контролю зі зменшенням похибки і, як наслідок, зі зростанням кількості ланок апроксимальної ламаної.

Щоб усунути вказані недоліки, розгляньмо можливості оцінювання на основі кількісної характеристики якості відтворення. Природно за таку характеристику взяти максимальну відстань між ланкою ламаної l_j та точками відповідної їй дуги кривої – $\Gamma_{t_{j-1}^*}^{l_j^*}$, яку апроксимує ланка, та порівнювати її із заданою похибкою наближення. Отже, склавши послідовність таких відстаней для всієї ламаної $\left\{d_{j,m}^{max}\right\}(j=1,...,m)$, де $d_j^{max}=\max\left\{d\left(\Gamma_{t_{j-1}^*}^{l_j^*},l_j\right)\right\}$ можна подати похибки відтворення кривої для всіх ланок ламаної графічно у вигляді точкової або стовпчикової діаграми, де по осі абсцис відкладати номери ланок ламаної, а по осі ординат – відповідну відстань від кривої до ламаної.

У разі інтерполяційної ламаної початкова та кінцеві точки її довільної ланки збігаються з точками вхідної кривої за $t=t_{j-1}^*$ та $t=t_j^*$. Отже, дуга кривої, яку інтерполює поточна ланка, визначено інтервалом зміни параметра кривої – $\left(t_{j-1}^*,t_j^*\right)$.

Відстань між прямою j-ї ланки інтерполяційної ламаної та точкою $M\left(x_M(t),y_M(t)\right)$ за $t\in\left(t_{j-1}^*,t_j^*\right)$ кривої, визначено такою рівністю:

$$d_j(t)=\left|\frac{A_jx_M+B_jy_M+C_j}{\sqrt{A_j^2+B_j^2}}\right|, \qquad (8.21)$$

де коефіцієнти $A_j=y_{j-1}-y_j, B_j=x_j-x_{j-1}, C=y_jx_{j-1}-x_jy_{j-1}$ залежать від координат кінцевих точок ланки.

Щоб визначити локальні екстремуми функції (8.21), диференціюймо її праву частину по t та дорівнявши її нулеві, маємо таке:

$$A_jx'(t)+B_jy'(t)=0.$$

Обчисливши корені цього рівняння, що лежать в досліджуваному інтервалі значень, вибираймо з них максимальне значення – d_j^{max}.

У разі, якщо вхідна крива є складеної з декількох елементарних дуг, вузлові точки інтерполяційної ламаної можуть належати до різних складових

дуг. Це означає, що відстань між кривою та ламаною потрібно визначати серед усіх дуг кривої, що містяться в межах вузлів інтерполяції такої ланки. Наприклад, для *B*-сплайна з відкритим вузловим вектором T ($t_i \in [0,T]$) визначаймо такі значення вузлового вектора $t_i \geq t_j$ (за $t_{i-1} < t_j$) та $t_{i-k} \geq t_{j-1}$ (за $t_{i-k-1} < t_{j-1}$), які відповідають j-й ланці ламаної. І далі для кожної дуги сплайна в межах значень вузлового вектора $[t_{i-k-1}, t_i]$ обчислімо відстані за наведеним раніше алгоритмом $\{d_k^{max}\}$. Після чого вибираймо максимальне з них – $d_j^{max} = \max\{d_k^{max}\}$.

8.2.4. Результати моделювання кривих без точок перегину

Моделювання відтворення ламаними за асимптотично оптимальним алгоритмом інтерполяції конкретних кривих ставило за мету дослідити вплив параметрів алгоритму на якість результатів апроксимації.

Перед початком розгляду результатів моделювання зупинімося на питанні їхнього оцінювання, залежно від заданої (яку необхідно забезпечити) точності наближення. Розподіл значень похибок або відстаней від кривої до ланок ламаної зазвичай залежить та його порівнюють із значенням допустимої похибки наближення. Щоб спростити це порівняння та мати змогу аналізувати результати, обчислені за різних значень точності, уведімо поняття відносної похибки наближення ланки ламаної до кривої, за яку візьмімо відношення максимальної відстані від ланки ламаної до дуги кривої, яку вона апроксимує, до значення заданої допустимої похибки наближення. Це означає, що значення відносної похибки в межах від 0 до 1 відповідають заданій точності апроксимації, а значення, більші за одиницю, виходять за її межі.

Обговорення результатів моделювання почнімо з дослідження факторів, пов'язаних з інтерполяцією інтегральної функції та її впливом на точність відтворення кривої. До цих факторів можна зарахувати такі: кількість вузлів первісної сітки дискретизації інтерпольованої інтегральної функції – N, похибку обчислення значень функції Ψ у вузлах цієї сітки – δ^{Ψ}.

Спочатку зосередьмося на вивченні впливу на результати відтворення кількісної характеристики первісної дискретизації функції (8.6) N

за сталого δ^{Ψ}. Водночас інтерполяцію обчислених значень інтегральної функції будемо виконувати за лінійним алгоритмом – ф-ли (8.12) та за допомогою LRB-сплайна – ф-ли (8.20). Це дозволить порівняти результати апроксимації та визначити раціональне значення параметра N.

У роботі [7] під час визначення значення N використовували таку залежність:

$$N = \left[\frac{1}{\varepsilon}\right] + 1, \tag{8.22}$$

яка враховує тільки допустиму похибку відтворення та не залежить від властивостей самої кривої. До того ж цей вираз не залежить від того, як відбувається інтерполяція значень інтегральної функції на проміжках між вузлами її дискретизації – прямими лініями, як у цій роботі, або іншими методами. Із метою подальшого порівняння, наведімо ряд значень похибок апроксимації та відповідних їм значень кількості вузлів дискретизації, які зведімо до табл. 8.1.

Таблиця 8.1

Ряд значень кількості вузлів дискретизації, обчислений за виразом (8.22)

Похибки ε	0,000 1	0,001	0,005	0,007 5	0,01	0,025	0,05	0,1
Кількість вузлів N	10 001	1 001	201	134	101	41	21	11

Для визначення необхідної кількості вузлів за розглядуваним методом інтерполяції й подальшого порівняння з результатами табл. 8.1, візьмімо за основу оцінку похибки лінійної інтерполяції та, відштовхуючись від неї, будемо визначати кількість точок дискретизації, одночасно фіксуючи вплив на розподіл похибок відтворення. Згідно із [13; 18; 22; 23] оцінка максимальної похибки лінійної інтерполяції на відрізку $\left[t_{i-1}, t_i\right]$ має такий вигляд:

$$\delta = \max_{[t_{i-1}, t_i]} \left|f(t) - L_1(t)\right| \le h^2 \frac{M_{2,i}}{8}, \tag{8.23}$$

де $M_{2,i} = \max_{[t_{i-1}, t_i]} \left|f''(t)\right|$.

Ураховуючи, що за сталого кроку дискретизації $h = \frac{T}{N}$ та поширюючи оцінку на всі інтервали $M_2 = \max\limits_{T} \left|f''(t)\right|$, обчислюймо таке:

$$M_2 = N \geq T\sqrt{\frac{M_2}{8\delta}}. \tag{8.24}$$

Для визначеності візьмімо таке:

$$N = \left[T\sqrt{\frac{M_2}{8\delta}}\right] + 1. \tag{8.25}$$

Як випливає з оцінки (8.25) лінійна інтерполяція має порядок $O\left(h^2\right)$, похибку самої лінійної раціональної інтерполяції було оцінено в [4] як $O\left(h^3\right)$, тобто на рівні параболічної інтерполяції. Оцінка параболічної інтерполяції має такий вигляд:

$$\delta_i = \max_{[t_{i-1}, t_i]} \left|f(t) - L_2(t)\right| \leq \frac{\sqrt{3}h^3}{27} M_{3,i}, \tag{8.26}$$

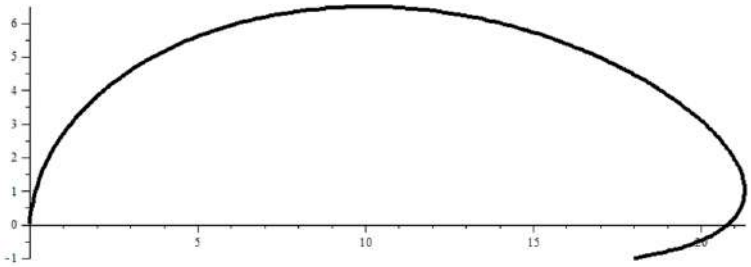
де $M_{3,i} = \max\limits_{[t_{i-1}, t_i]} \left|f'''(t)\right|$.

Відштовхуючись від цієї оцінки, за аналогією з лінійною інтерполяцією будемо мати таке:

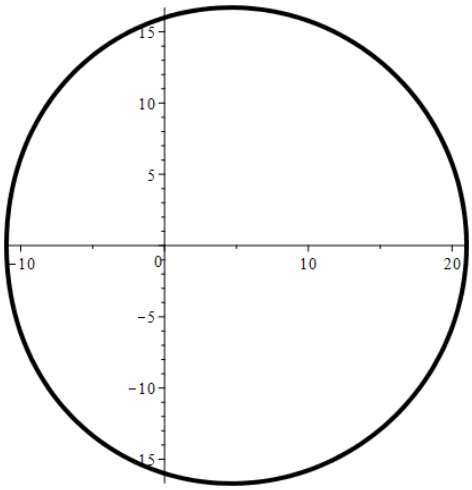
$$N = \left[\frac{T\sqrt[6]{3}}{3}\sqrt[3]{\frac{M_3}{\delta}}\right] + 1. \tag{8.27}$$

Далі постає питання вибору значення похибки інтерполяції інтегральної функції. Оскільки від цього значення залежить обчислена точність наближення кривої, почнемо з оцінки $\delta = \varepsilon$, далі будемо зменшувати цю величину.

Для моделювання відтворення було вибрано дві криві лінії (замкнену та незамкнену) (рис. 8.1).



а



б

Рис. 8.1. **Криві лінії, вибрані для відтворення ламаними: а) крива Безьє, б) равлик Паскаля**

Перша крива (рис. 8.1а) є кривою Безьє шостого порядку з координатами точок опорного полігону (0,0; 0,0), (0,55; 6,9), (11,8; 9,2), (16,5; 6,25), (22,0; 3,2), (24,5; 0,0), (18,0; –1,0).

Друга крива (рис. 8.1б) має назву равлика Паскаля та має такі параметри: радіус вихідного кола – 2,5, зміщення – 14.

Обидві криві мають опуклу форму без точок перегину, що забезпечує відсутність точок розриву першої похідної функції (4), яка буде визначати

значення M_2 в рівнянні (8.25), та виключає необхідність у розгляді питання оптимізації параметра α в рівнянні (8.6).

У табл. 8.2 та 8.3 наведено результати відтворення кривих ламаними за лінійного методу інтерполяції інтегральної функції та сталого значення похибки інтегрування $\delta^{\Psi} = 0{,}5 \cdot 10^{-10}$ методом Кленшоу – Кертіса за обчислення значень ψ_i. Результати подано у вигляді показників розподілів, що обчислювали для послідовності відносних похибок ланок ламаних. Це максимальне та мінімальне значення послідовності $\gamma_{max}, \gamma_{min}$, середнє $\bar{\gamma}$, медіана Me, розмах значень відносних похибок для послідовності ланок ламаної Δ, стандарт (середньоквадратичне відхилення) δ та коефіцієнт варіації c_v. Із поданих результатів випливає, що за значень $\delta = \varepsilon/64, \varepsilon/256$ похибка апроксимації стабілізується, перебуваючи в діапазоні допустимих значень досить близько до його верхньої межі – 0,9...0,98. Збільшення кількості на порядок не робило значного впливу на результати апроксимації.

Таблиця 8.2

Результати відтворення кривої (рис. 8.1а за ε = 0,001, кількість вузлів апроксимації m = 122)

Показники	Значення похибки інтерполяції інтегральної функції, δ						
	ε	$\varepsilon / 4$	$\varepsilon / 8$	$\varepsilon / 32$	$\varepsilon / 64$	$\varepsilon / 256$	ε^2
γ_{max}	0,98939	0,97467	0,97235	0,97148	0,97124	0,97122	0,97120
γ_{min}	0,95361	0,96763	0,96939	0,97040	0,97060	0,97070	0,97072
Δ	0,03578	0,00704	0,00296	0,00108	0,00063	0,00052	0,00047
$\bar{\gamma}$	0,97093	0,97091	0,97092	0,97091	0,97091	0,97091	0,97091
Me	0,97045	0,97083	0,97091	0,97090	0,97090	0,97091	0,97091
σ	0,00731	0,00106	0,00063	0,00020	0,00014	0,00010	9,72105
c_v,%	0,752	0,109	0,065	0,020	0,015	0,010	0,010
N	73	144	203	405	572	1 143	2 257

Таблиця 8.3

Результати відтворення кривої (рис. 8.1б за $\varepsilon = 0{,}01$, кількість вузлів апроксимації $m = 91$)

Показники	Значення похибки інтерполяції інтегральної функції, δ						
	ε	$\varepsilon / 4$	$\varepsilon / 8$	$\varepsilon / 32$	$\varepsilon / 64$	$\varepsilon / 256$	ε^2
γ_{max}	1,03187	0,98851	0,97909	0,97709	0,97643	0,97597	0,97595
γ_{min}	0,91859	0,96454	0,97131	0,97454	0,97506	0,97529	0,97536
Δ	0,11328	0,02397	0,00778	0,00256	0,00137	0,00068	0,00059
$\overline{\gamma}$	0,97588	0,97579	0,97578	0,97578	0,97578	0,97578	0,97578
Me	0,97549	0,97517	0,97604	0,97578	0,97579	0,97579	0,97578
σ	0,01965	0,00537	0,00177	0,00058	0,00026	0,00010	0,91050
N	28	55	78	154	217	433	764

Більш наявно з'ясувати результати можна, безпосередньо порівнявши діаграми розподілів похибок за різних значень δ, зобразивши їх в одному діапазоні значень (рис. 8.2 та 8.3).

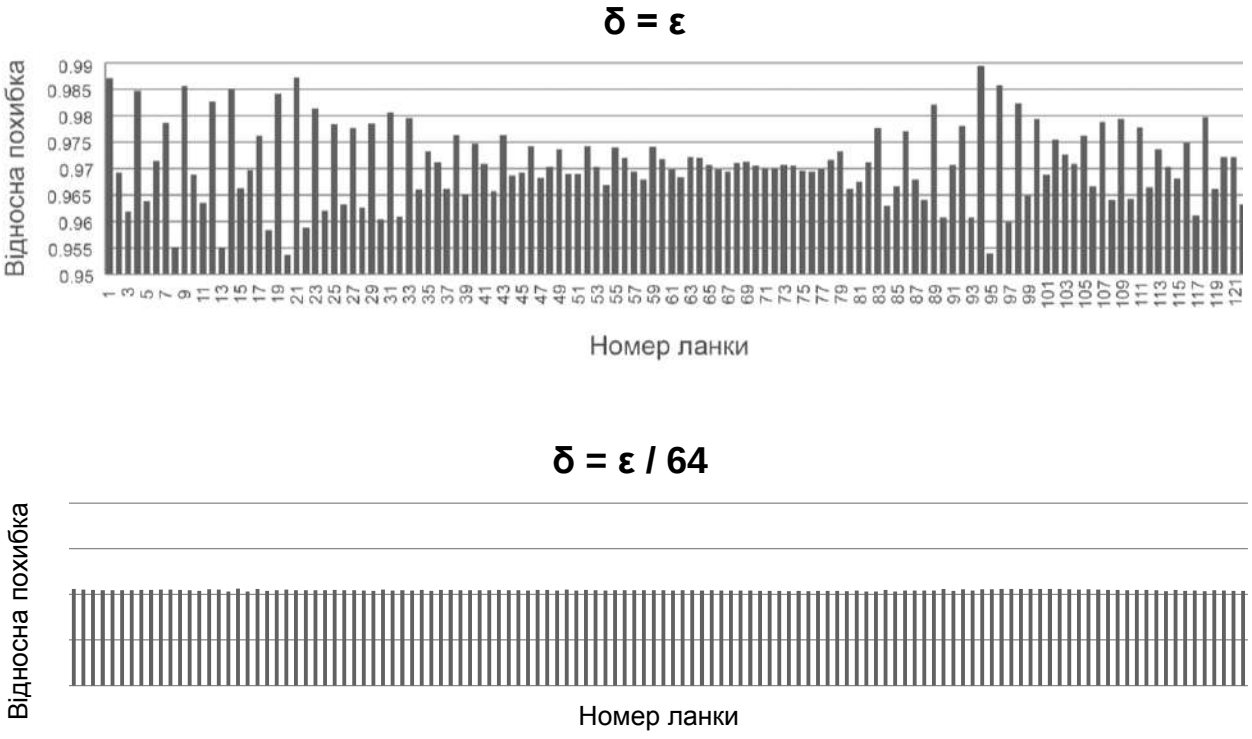


Рис. 8.2. **Розподіли похибок за відтворення кривої** (рис. 8.1а із заданою точністю $\varepsilon = 0{,}001$)

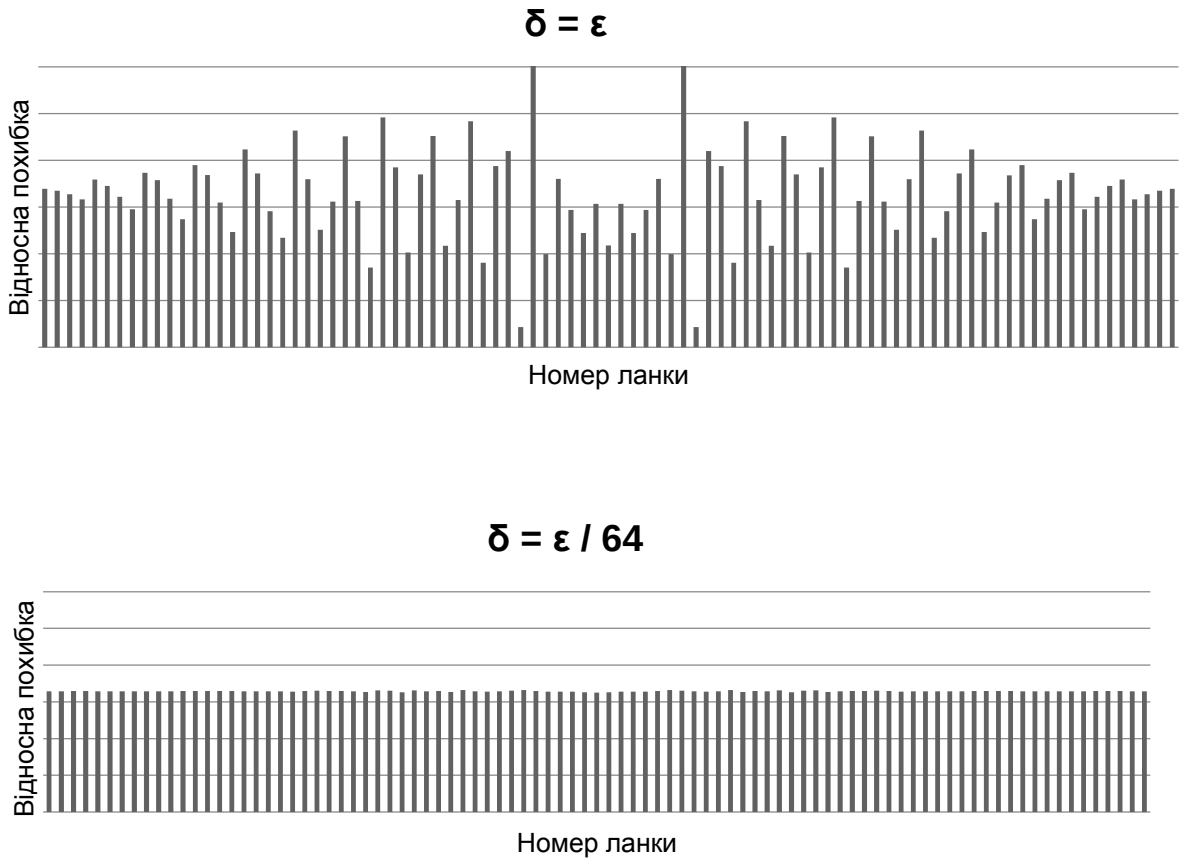


Рис. 8.3. **Розподіли похибок**
за відтворення кривої
(рис. 8.1б із заданою точністю ε = 0,01)

Перейдімо до розгляду впливу інтерполяції функції регулятора вузлів LRB-сплайном та порівняймо їх із уже визначеними результатами за лінійної інтерполяції цієї функції.

Як і за лінійного способу інтерполяції інтегральної функції будемо змінювати величину похибки ε навколо оцінки $\delta = \varepsilon$.

У табл. 8.4, 8.5 наведено результати апроксимації для інтерполяційної схеми відтворення кривих ламаними за методу інтерполяції інтегральної функції лінійним раціональним B-сплайном за аналогічних до лінійного методу умов.

Таблиця 8.4

Результати відтворення кривої (рис. 8.1а за ε = 0,001, кількість вузлів апроксимації m = 122)

Показники	Значення похибки інтерполяції інтегральної функції, δ						
	3ε	ε	$\varepsilon / 9$	$\varepsilon / 27$	$\varepsilon / 81$	$\varepsilon / 243$	ε^3
γ_{max}	0,97310	0,97210	0,97137	0,97130	0,97123	0,97123	0,97120
γ_{min}	0,96703	0,96981	0,97071	0,97073	0,97073	0,97071	0,97073
Δ	0,00607	0,00230	0,00067	0,00057	0,00050	0,00053	0,00047
$\overline{\gamma}$	0,97091	0,97091	0,97091	0,97091	0,97091	0,97091	0,97091
Me	0,97087	0,97091	0,97091	0,97091	0,97091	0,97091	0,97091
σ	0,00070	0,00031	0,00011	0,00010	0,00010	9,83105	9,52105
N	28	40	81	116	167	240	3 817

Таблиця 8.5

Результати відтворення кривої (рис. 8.1б за ε = 0,01, (кількість вузлів апроксимації m = 91)

Показники	Значення похибки інтерполяції інтегральної функції, δ						
	3ε	ε	$\varepsilon / 9$	$\varepsilon / 27$	$\varepsilon / 81$	$\varepsilon / 243$	ε^3
γ_{max}	0,98444	0,98065	0,97606	0,97594	0,97591	0,97591	0,97591
γ_{min}	0,96829	0,97270	0,97530	0,97537	0,97531	0,97537	0,97537
Δ	0,01615	0,00795	0,00072	0,00057	0,00056	0,00054	0,00054
$\overline{\gamma}$	0,97578	0,97578	0,97578	0,97578	0,97578	0,97578	0,97578
Me	0,97612	0,97603	0,97582	0,97579	0,97579	0,975783	0,97578
σ	0,00217	0,00110	0,00020	9,18105	8,62105	8,52105	8,51105
N	14	19	39	55	79	114	389

Відносна похибка апроксимації, згідно з даними таблиць, стабілізувалася за значень $\delta = \varepsilon / 81 ... \varepsilon / 243$ на значеннях близьких до тих, що спостерігали за лінійного методу. Порівняння дисперсій розподілів за критерієм Кохрена, уже починаючи з $\delta = \varepsilon / 9 \ldots \varepsilon / 27$, не виявляє статистичної

значущості. Якщо порівняти також результати відповідних таблиць за лінійного та LRBS-методу інтерполяції інтегральної функції стає зрозумілим те, що, аби досягти тих самих результатів наближення, сплайновому методу потрібно було в 5 – 6 разів меншу кількість вузлів дискретизації, аніж лінійному методу.

Розподіли похибок за відтворення кривих наведено на рис. 8.2 і 8.3.

Щоб детальніше дослідити різницю між розподілами похибок для двох методів інтерполяції інтегральної функції (лінійного та LRBS), порівняймо залежності вибіркових дисперсій σ^2 від ступеня дискретизації інтегральної функції. Щоб знизити вплив заданої допустимої похибки апроксимації ε, будемо визначати цей ступінь через відношення кількості точок дискретизації інтегральної функції N до кількості вузлів апроксимації m.

На рис. 8.4 та 8.5 показано графіки для кривих, що було вибрано для відтворення ламаними. Із цих графіків випливає, що швидкість зниження дисперсії зі зростанням ступеня дискретизації була різною для лінійної та LRBS-інтерполяцій інтегральної функції. Це виявляється в тому, що за LRBS-інтерполяції похибка апроксимації стабілізувалася набагато раніше – за до $1 \div 2\,\mathrm{N/m}$, порівняно зі $5 \div 7\,\mathrm{N/m}$ – для лінійного методу.

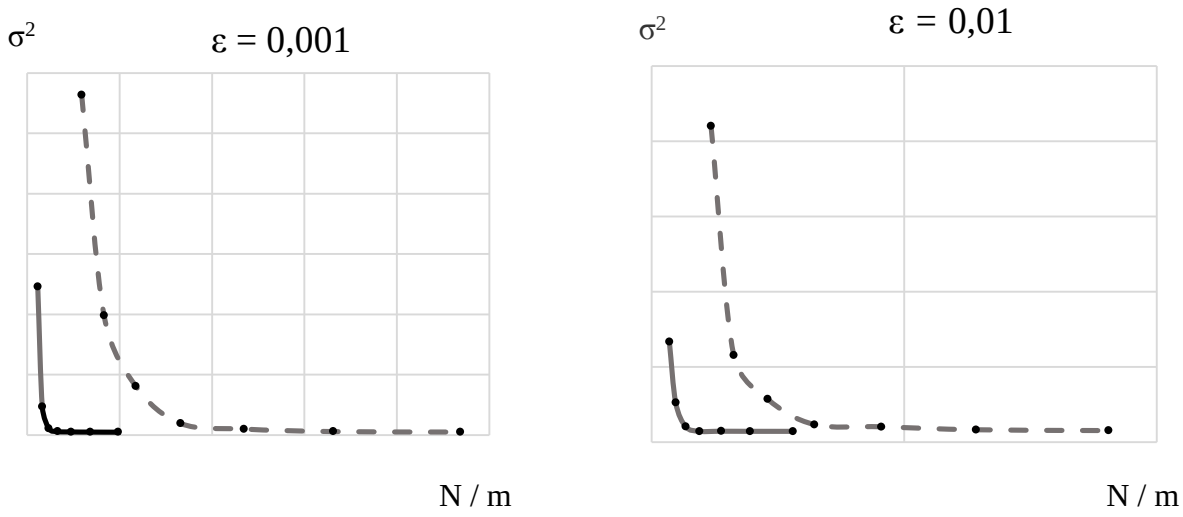


Рис. 8.4. **Графіки значень дисперсії розподілів, залежно від відношення кількості вузлів дискретизації функції до кількості вузлів апроксимації для кривої** (рис. 8.1а)

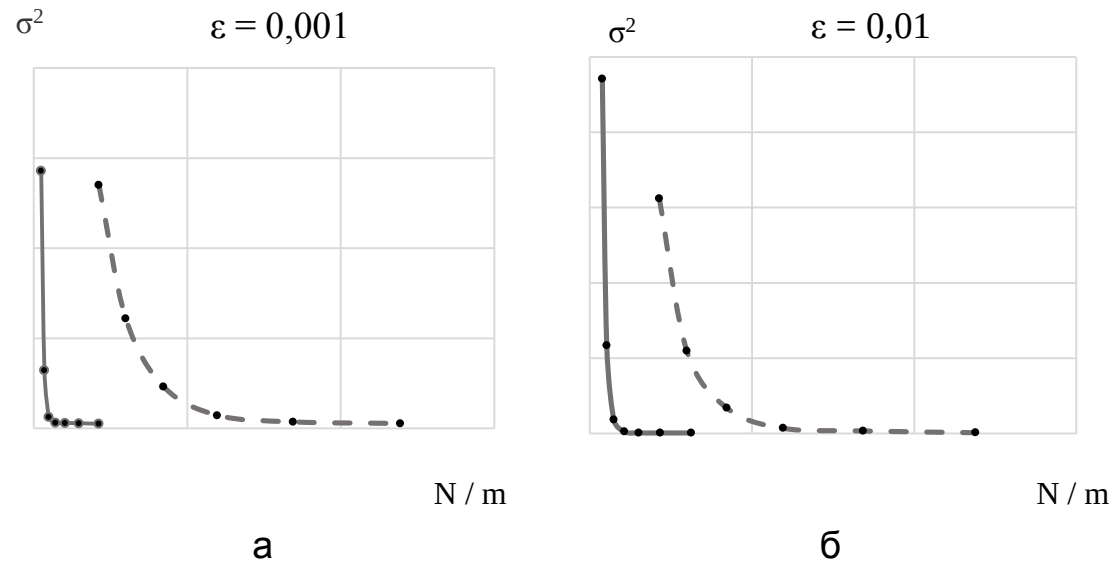


Рис. 8.5. **Графіки значень дисперсії розподілів, залежно від відношення кількості вузлів дискретизації функції до кількості вузлів апроксимації для кривої** (рис. 8.1б)

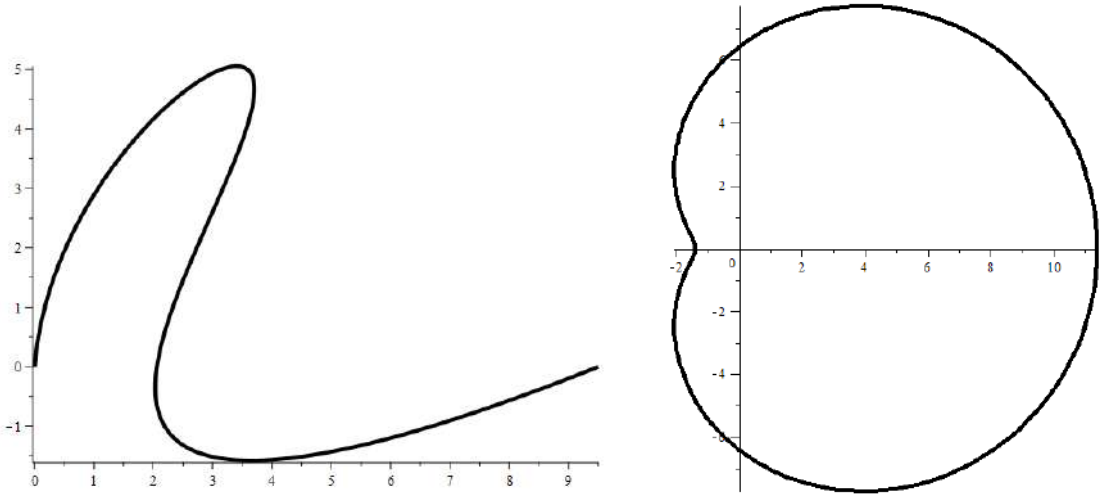
Порівняння кількісних показників дискретизації за виразами (8.22) та (8.25), (8.27) демонструє, що ф-ла (8.22) може давати як збільшену, так і зменшену кількість точок дискретизації щодо значень стабілізації похибки апроксимації (див. табл. 8.2 – 8.5).

Проведені експерименти також показали придатність застосованих розрахункових моделей до відтворення реальних кривих ліній. Водночас похибка апроксимації ланок ламаних стабілізувалася на рівні 90 – 98 % від допустимої межі. Остання обставина свідчить про прагнення асимптотично оптимального алгоритму до верхньої межі ε – допустимого коридору апроксимації та найменшої кількості ланок інтерполяційної ламаної.

8.2.5. Результати моделювання кривих із наявними точками перегину

Розгляньмо результати інтерполяції ламаними за формулою розподілу (8.5) для випадків кривих зображених на рис. 8.6. Перша крива є кривою Безьє п'ятого порядку (рис. 8.6а) та має одну точку перегину, а інша має назву равлика Паскаля (рис. 8.6б) із двома точками перегину. Під час відтворення цих кривих було враховано результати щодо визначення

кількості точок первісної дискретизації функції (8.5). Однак, оскільки для таких кривих друга похідна у виразах (8.25), (8.27) має розриви, брали значення $N = 10m$.



а б

Рис. 8.6. **Криві лінії, вибрані для відтворення ламаними: а) крива Безьє п'ятого порядку з координатами точок опорного полігону (0,0; 0,0), (0,55; 6,9), (11,8; 9,2), (-2,5; -1,25), (-1,0; -4,2), (9,5; 0,0); б) равлик Паскаля: радіус вихідного кола – 2,5, зміщення – 6,4**

Щоб оцінити якість відтворення кривих, скористаймося запропонованою раніше методикою та побудуймо декілька стовпчикових діаграм розподілів похибок за ланками інтерполяційної ламаної. На рис. 8.7 та 8.8 зображено розподіли похибок кривих, де по горизонтальній осі відкладено номери ланок ламаної, що відраховують у бік збільшення параметра кривої, а по вертикальній осі – відносні похибки для відповідної ланки. На цих рисунках можна побачити аномальні мінімуми та максимуми, які пояснено наявністю у кривих точок перегину, де кривина набуває нульового значення та відбувається зміна її знака. Ф-ла (8.5) інтегральної функції не враховує можливу наявність таких ділянок кривої, оскільки від'ємні значення кривини нівельовано наявністю модуля.

Отже, такі значення можна вважати за викиди. Так, для розподілу кривої з однією точкою перегину (рис. 8.7а) значенню максимуму похибки, що відбувається на 18-й ланці, відповідає інтервал зміни параметра t кривої, який містить значення, де підінтегральна функція (8.4) набуває нульового значення. Саме таке значення містить й інтервал 53-ї ланки, де перебуває мінімум розподілу на рис. 8.7б.

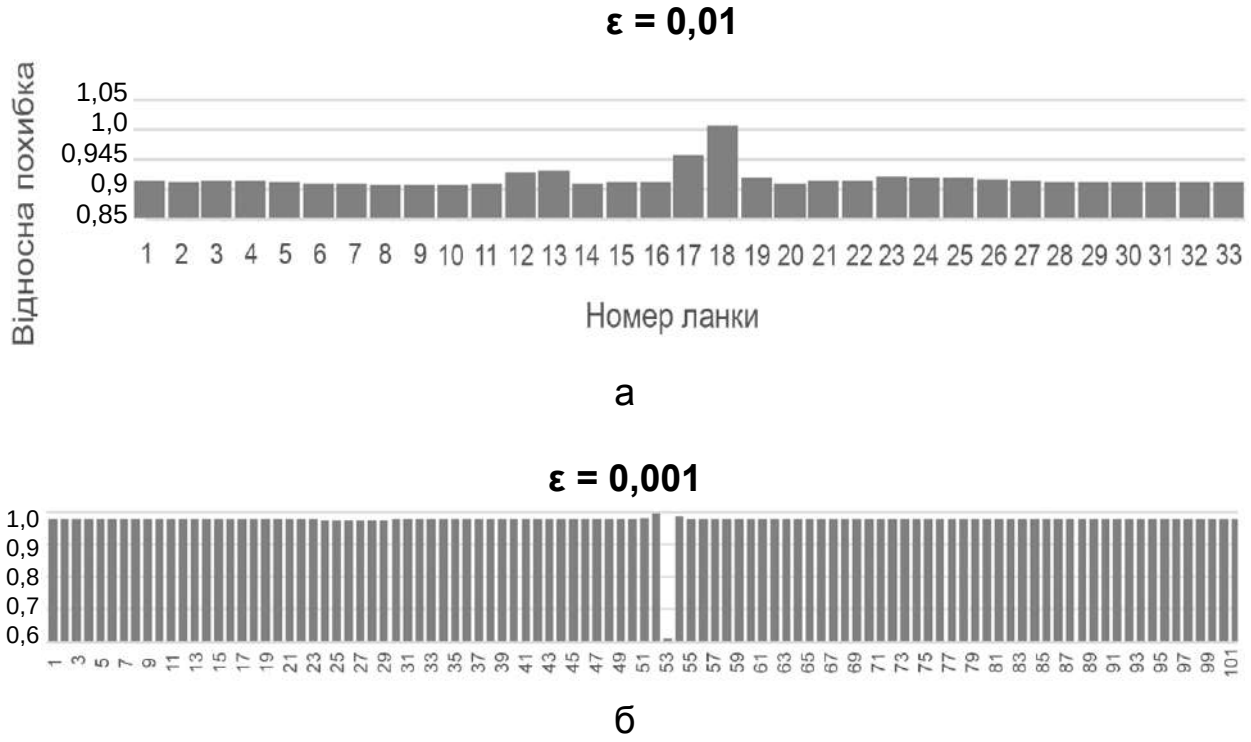


Рис. 8.7. **Результати відтворення кривої Безьє (рис. 8.1а) за визначення вузлів інтерполяції за виразом (8.3) за допустимої похибки: а) ε = 0,01; б) ε = 0,001**

Аналогічні до попереднього випадку тенденції демонструють результати моделювання відтворення іншої кривої (рис. 8.8). Завдяки наявності вже двох симетричних відносно горизонтальної осі точок перегину равлика Паскаля, спостерігають аномалії мінімальних та максимальних значень розподілів, які так само є симетричними.

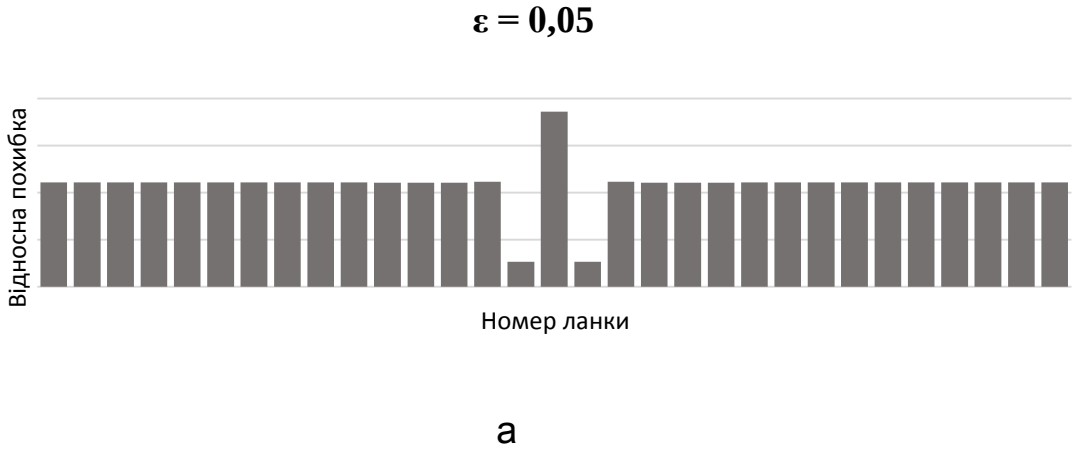
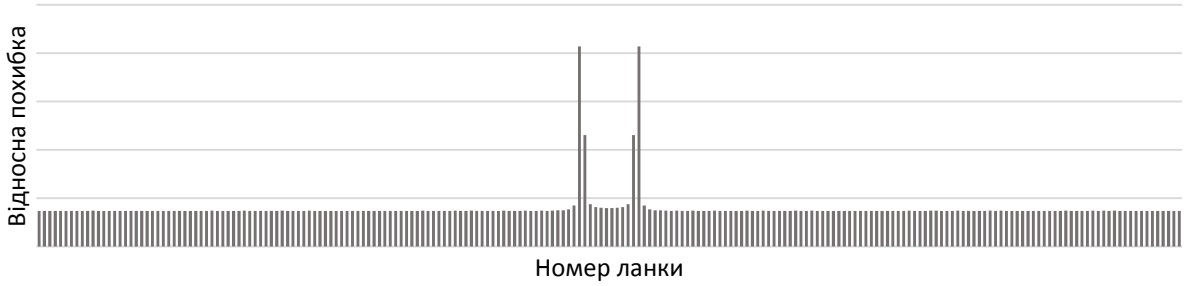


Рис. 8.8. **Результати відтворення равлика Паскаля (рис. 8.1б) за визначення вузлів інтерполяції за виразом (8.3) за допустимої похибки: а) ε = 0,05; б) ε = 0,001**

ε = 0,001



б

Закінчення рис. 8.8

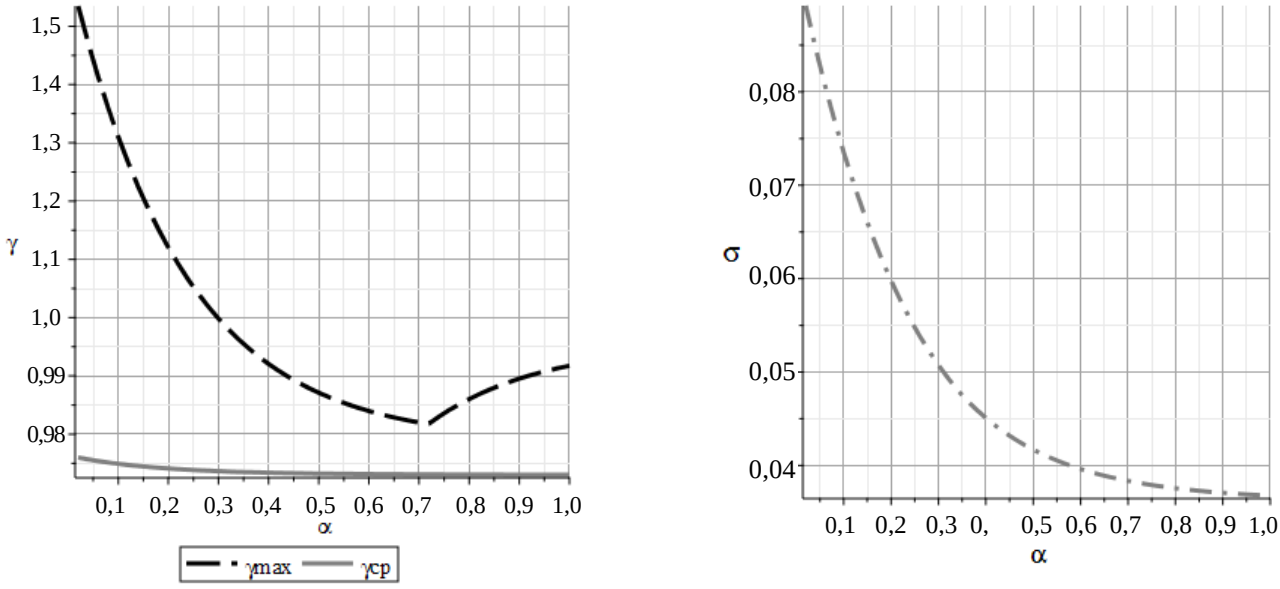
Отже, застосування функції-регулятора вузлів у формі (8.5) до випадку, коли крива, що відтворюють, має точки перегину, дає результати апроксимації з порушенням монотонності на ланках ламаної, які відповідають ділянкам дуги, де відбувається зміна напрямку кривини. Це може призводити до появи аномальних значень похибки апроксимації, які перевищують допустимі значення. Запропоноване в [7] використання ф-ли (8.6) до таких випадків, передбачає наявність доданка, за допомогою якого можна дещо змістити вузлові значення, щоб уникнути занадто великих значень похибки. Але в цій роботі було тільки визначено межі зміни параметра *α* без указівок, за якими критеріями може бути вибрано конкретне значення цього параметра за наявності кривої та необхідної точності її наближення.

Отже, набуває актуальності питання можливості оптимізації значення параметра α під час визначення вузлів інтерполяції плоских кривих, що мають точки перегину.

Обговорення критеріїв, за якими один розподіл похибок апроксимації можна вважати кращим, ніж інший, почнімо з визначення того, які риси може мати оптимальне рішення. Вочевидь, розподіл похибок інтерполяції можна вважати таким, якщо за тієї самої кількості ланок він забезпечує таке: значення послідовності похибок, які не перевищують допустимого значення; значення окремих похибок є близькими між собою, інакше кажучи, забезпечено мінімальне середньоквадратичне відхилення послідовності похибок, або коефіцієнт варіації; немає аномалій, тобто занадто малих або великих значень, що відрізняються від інших. Згідно із цим, за показники послідовностей похибок, що підлягають дослідженню, можна взяти максимальне значення відносної похибки послідовності, її середнє значення, середньоквадратичне відхилення та коефіцієнт варіації.

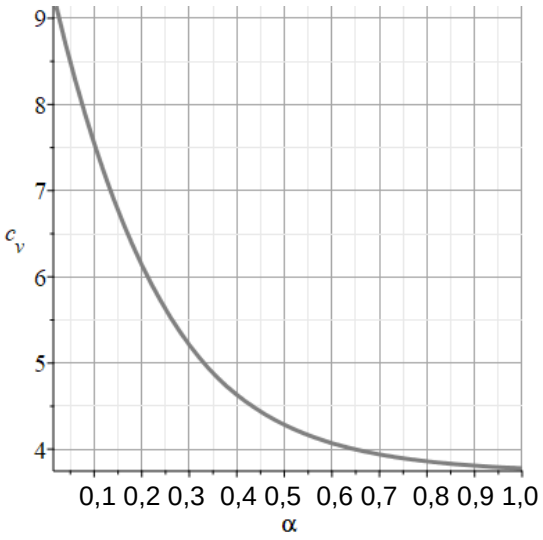
Для вивчення характеру зміни зазначених показників, побудуймо графіки зміни цих величин, залежно від значень параметра α функції-регулятора вузлів. Побудову здійснімо для тих самих кривих (див. рис. 8.5а та 8.5б). Щоб обчислити значення кожного з показників, виконали попередню дискретизацію значень по t функції-регулятора, обчислювали значення інтегральної функції Ψ з урахуванням доданка (8.6), далі визначали вузлові значення параметра для асимптотично оптимального розподілу, координати вузлових точок ламаної інтерполяції й, нарешті, значення відстаней від кривої до кожної ланки ламаної, які брали за абсолютні значення похибок апроксимації. Зіставивши значення ряду абсолютних похибок до значення допустимої похибки ε було визначено послідовності відносних похибок $\{\gamma_j\}$, за значеннями яких дістали статистичні характеристики послідовностей. Значення параметра α функції-регулятора брали, зважаючи на межі, що було визначено в [7], зі збільшенням вправо до 1, тобто в інтервалі [0;1]. Водночас для обчислення значень рядів похибок брали 50 значень рівномірного розподілу цього інтервалу (побудовано 50 інтерполяційних ламаних заданої кривої).

На рис. 8.9 зображено графіки показників розподілів похибок за ε = = 0,001 для кривої Безьє (див. рис. 8.6а), аналогічні ним тенденції виявляють графіки для равлика Паскаля (див. рис. 8.6б).

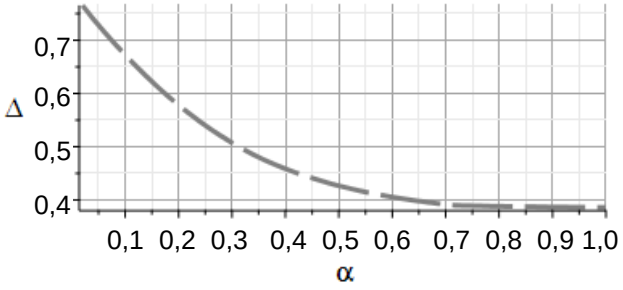


а б

Рис. 8.9. **Графіки статистичних характеристик рядів похибок, залежно від значень параметра α для кривої: а) максимальна та середня похибка; б) середнє квадратичне відхилення; в) коефіцієнт варіації; г) розмах**



в



г

Закінчення рис. 8.9

Проаналізувавши побудовані графіки, можна зазначити таке:

для обох кривих наявні екстремуми (мінімуми) на графіках максимальної похибки;

графіки значень середньої похибки рядів найменше були схильними до змін (для першого випадку в межах зміни параметра графік монотонно спадає, наближаючись до асимптоти; для другого випадку на графіку наявний слабкий мінімум, який за значенням параметра дає максимальну похибку, що перевищує допустиме значення);

порівнявши графіки середньоквадратичного відхилення та коефіцієнта варіації можна зазначити, що вони демонструють однакові тенденції, які можна просто пояснити, адже коефіцієнт варіації залежить від стандартного відхилення та середнього для послідовності похибок, при чому остання величина досить слабо змінюється. Для першої кривої ці графіки не виявляють екстремальних значень, спадаючи зі збільшенням α, наближаючись до горизонтальної асимптоти. Для другої кривої графіки, що розглядають, мають два мінімуми (локальний і глобальний), водночас, якщо значення α для екстремумів порівняти зі значеннями екстремумів для графіків інших характеристик, то можна дослідити, що першому (меншому) мінімуму відповідають значення близькі до мінімуму для середньої похибки послідовності, а другому (глобальному) мінімуму – значення мінімуму для максимальної похибки;

графіки розмаху значень похибок ламаних для обох кривих загалом відображають аналогічні до зміни стандартного відхилення та коефіцієнта варіації тенденції до залежності з більш чітко визначеними у випадку другої кривої максимумами і мінімумами.

Отже, зважаючи на пріоритет забезпечення похибки апроксимації в допустимих межах для всіх ланок ламаної, а також ураховуючи описані раніше тенденції до зміни показників розподілу з урахуванням припущення унімодальності для методів оптимізації [20], можна вибрати за цільову функцію максимальну серед усіх похибок послідовності ланок ламаної. Тоді завдання мінімізації цільової функції можна записати в такому вигляді:

$$f(\alpha) = \max\{\delta_i (i = 1..m)\} \to \min. \qquad (8.28)$$

Оскільки вираз цільової функції не має аналітичного характеру й не може бути визначено її похідну, до пошуку оптимального рішення може бути застосовано методи нульового порядку [20], стратегія яких ґрунтується на використанні інформації тільки про цільову функцію.

Серед цих методів було вибрано методи золотого перетину та більш швидкий метод Брента, який використовує комбінацію методів золотого перетину та квадратичної інтерполяції.

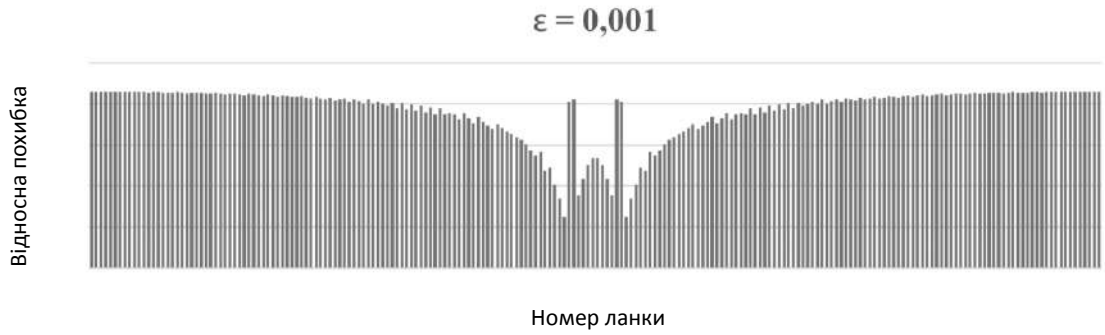
Результати обчислення значення параметра α за вказаним алгоритмом для кривих (див. рис. 8.6) для різної точності наближення наведено в табл. 8.6. Порівнявши результати, наведені в цій таблиці для $\varepsilon = 0{,}001$, із графіками максимальної похибки, пересвідчімося у відповідності результатів обчислень.

Таблиця 8.6

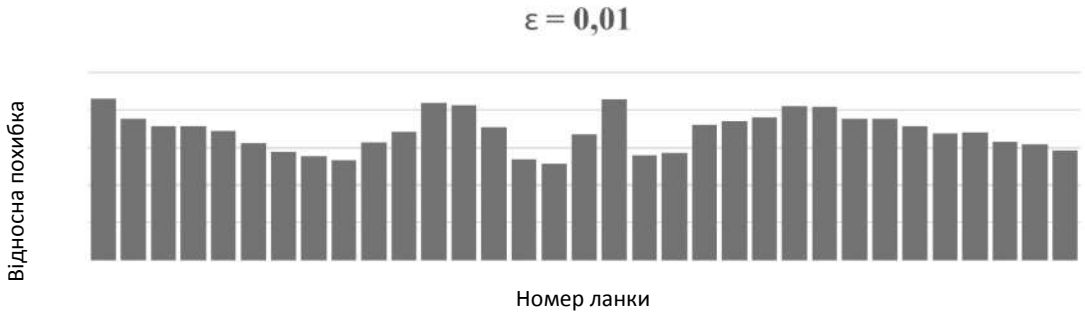
Результати пошуку оптимального значення параметра α

Назва кривої	Значення параметра α за похибки апроксимації ε			
	ε = 0,05	ε = 0,01	ε = 0,005	ε = 0,001
Крива Безьє (рис. 8.6а)	0,99609375	0,47656250	0,61328125	0,71875000
Равлик Паскаля (рис. 8.6б)	0,4296875	0,7187500	0,7343750	0,6171875

На рис. 8.10 у вигляді стовпчикових діаграм показано послідовності похибок ланок ламаних під час визначення їх за оптимізованими значеннями параметра α.



а



б

Рис. 8.10. **Результати відтворення кривих під час визначення вузлів інтерполяції за виразом (8.6) з оптимізацією параметра α: а) крива (рис. 8.5а); б) крива (рис. 8.5б)**

Якщо порівняти ці діаграми з відповідними розподілами, зображеними на рис. 8.7 та 8.8, можна побачити, що визначені в результаті оптимізації послідовності вже не містять значень, які перевищують допустиму похибку.

8.3. Висновки

Моделювання відтворення реальних плоских параметричних кривих опуклої форми ламаними за асимптотично оптимальним алгоритмом із лінійним методом інтерполяції значень інтегральної функції засвідчило цілком прийнятні результати роботи алгоритму без перевищення допустимої похибки апроксимації в разі достатнього ступеня дискретизації значень зазначеної функції.

Порівняння кількісних показників дискретизації за виразами (8.22) (8.25) та (8.27) демонструє, що ф-ла (8.22) може давати як збільшену, так і зменшену кількість точок дискретизації щодо значень стабілізації похибки апроксимації на рівні $\varepsilon / 64$ (див. табл. 8.2 – 8.5).

Результати моделювання також дозволили встановити вплив підвищення якості апроксимації інтегральної функції шляхом збільшення кількості вузлів дискретизації на результати відтворення кривої ламаною. Він полягає в поліпшення показників розподілу похибок апроксимації за ланками апроксимувальної ламаної, а саме зменшення коефіцієнтів варіації та середньоквадратичних відхилень цих рядів, що відображає стабілізацію значень похибок навколо середнього значення.

Застосування до асимптотично оптимального алгоритму LRBS-інтерполяції значень інтегральної функції, дозволило значно знизити (у 5 – 6 разів) кількість точок дискретизації інтегральної функції, порівняно із її лінійною інтерполяцією, та пов'язані із цим розрахунки.

Проведені дослідження з моделювання відтворення плоских кривих, що мають точки перегину, показали прийнятність застосування ф-ли (8.4) з оптимізацією параметра α для регулювання розподілу вузлів лінійної інтерполяції таких кривих за асимптотично оптимальним алгоритмом. Водночас вибору за цільовою функцією максимуму серед похибок послідовності ланок ламаної дозволило забезпечити рівень похибок, нижчий від граничного значення. Із результатів моделювання випливає, що серед значень параметра α, які відповідали оптимальному рішенню для різних умов, були значення, що виходили за вказані в роботі [2] межі.

Отже, це питання має бути досліджено окремо.

Розділ 9
Оцінювання продуктивності кластерів загального призначення Azure під час тестування завдань машинного навчання Apache Spark

9.1. Вступ і формулювання завдання

Розроблення власної системи опрацювання даних із використанням машинного навчання потребує досить великих витрат на створення та підтримання необхідного апаратного забезпечення і здебільшого достатньо великих часових витрат. Розв'язанням цих проблем може стати використання хмарних платформ для машинного навчання, яких тепер є достатньо багато [6; 9; 12; 16; 27; 38; 42]. Відомим терміном для платформ такого роду є MLaaS (Machine learning as a service) – [22; 24; 28; 29; 45]. Під цим визначенням мають на увазі різні хмарні платформи, які охоплюють більшість інфраструктурних питань, як-от попереднє опрацювання даних, навчання, тестування й оцінювання моделей, а також подальше прогнозування. MLaaS допомагає отримати вигоду від машинного навчання без суміжних витрат часу та ризику створити внутрішню команду машинного навчання. Окремо можна виділити платформи штучного інтелекту – набір інструментів для створення комплексної системи машинного навчання. Зазвичай вони дають спрощений інтерфейс для взаємодії та деякі додаткові функції, але натомість користувач є обмеженим у виборі того чи того інструменту. Такі платформи дають можливість створити модель, протестувати та розгорнути її для використання.

До найбільш відомих бібліотек машинного навчання слід зарахувати [17; 21; 27] такі:

TensorFlow – це відкрита програмна бібліотека для машинного навчання, розроблена компанією Google для вирішення завдань побудови та тренування нейронної мережі, із метою автоматичного визначення та класифікації образів, досягаючи якості людського сприйняття. Застосовують як для досліджень, так і для розроблення власних продуктів Google.

PyTorch – це бібліотека машинного навчання для мови Python із відкритим вихідним кодом, створена на базі Torch. Використовують для вирішення різних завдань: комп'ютерного зору, опрацювання природної мови. Розробляють переважно групою штучного інтелекту Facebook.

Apache Spark MLlib – це бібліотека машинного навчання Apache Spark, що складається із загальних алгоритмів навчання та утиліт, включно із класифікацією, регресією, кластеризацією, спільною фільтрацією, зниженням розмірності та основних оптимізацій примітивів [42].

Scikit-learn – це безкоштовна бібліотека машинного навчання програмного забезпечення для мови програмування Python. У ньому подано різні алгоритми класифікації, регресії та кластеризації, включно з підтримувальними векторними машинами, випадковими лісами, підвищенням градієнта, k-засобами та DBSCAN, і розроблено для взаємодії із числовими та науковими бібліотеками Python NumPy та SciPy.

Apache MXNet – це швидкий, гнучкий та надзвичайно масштабований фреймворк глибинного навчання, що підтримує сучасні технології в моделях глибинного навчання, включно зі згортковими нейронними мережами та мережами, які використовують короткострокову пам'ять.

Caffe – це система для глибинного навчання, розроблена в Каліфорнійському університеті в Берклі, яка підтримує багато різних типів архітектур глибинного навчання, орієнтованих на класифікацію та сегментування зображень.

Keras – це відкрита нейромережева бібліотека, написана мовою Python, яка здатна працювати зверху TensorFlow, CNTK або Theano. Вона містить численні реалізації широковживаних будівельних блоків нейронних мереж, як-от шари, цільові та передавальні функції, оптимізатори й безліч інструментів для спрощення роботи із зображеннями та текстом.

Багато з наявних сервісів підходять для випадків, коли потрібно здійснити аналіз невеликого обсягу даних, але не можуть бути ефективними для великих обсягів даних. Для завдань такого класу є ряд інших інструментів і сервісів, оптимізованих для розподіленого опрацювання даних у реальному часі. Одними з найбільш відомих представників сервісів цього класу є Azure HDInsight, Amazon EMR, Google Cloud Dataproc.

Azure HDInsight є хмарним дистрибутивом компонентів Hadoop. Azure HDInsight забезпечує просте, швидке й економічне опрацювання великих обсягів даних. Він дає можливість реалізувати різні сценарії, як-от здобуття, перетворення, завантаження та зберігання даних, інтернет-речей; машинне навчання за допомогою таких платформи з відкритим кодом, як Hadoop, Spark, Hive, LLAP, Kafka, Storm, R та ін. Azure HDInsight дає можливість створити власну конфігурацію кластера, використовуючи різні класи віртуальних машин: початкового рівня, загального призначення, оптимізовані для обчислень та пам'яті.

Apache Spark є програмним забезпеченням із відкритим вихідним кодом для вирішення завдань Bigdata та має різні переваги перед іншими ПЗ, зокрема воно є динамічним за природою, підтримує обчислення в пам'яті RDD та дає можливість повторного використання, толерантності до помилок, опрацювання потоку в реальному часі тощо. Фреймворк Spark – це загальноприйнята платформа обчислювальних кластерів. Він містить API, яке допомагає працівникам, що працюють із даними, виконувати потокове, машинне навчання або робочі високонавантажені запити SQL, які потребують повторного доступу до наборів даних. Spark може виконувати одночасно пакетне опрацювання та опрацювання потоків. *Пакетне опрацювання* належить до опрацювання раніше зібраної роботи у єдиному пакеті, тоді як *потокове опрацювання* означає роботу з потоковими даними Spark. Він може дістати доступ до будь-якого джерела даних Hadoop, а також працювати на кластерах Hadoop. Крім того, Apache Spark поширює можливості Hadoop MapReduce, що містить ітераційні запити та опрацювання потоків. Apache Spark пропонує високорівневі API для розробників на таких мовах програмування, як Java, Scala, Python та R. Порівняно з Hadoop, він у 100 разів є швидшим, ніж Big Data Hadoop, і має в 10 разів швидший доступ до даних із диска.

Отже, використання кластерів **Azure HDInsight** створює можливості для ефективного вирішення завдань машинного навчання шляхом розроблення та дослідження побудови кластерів із різною структурою для оцінювання продуктивності їхнього використання в умовах наявності великих даних.

Мета – експериментально дослідити вплив різних конфігурацій кластерів Azure HDInsight *загального призначення* (на відміну від роботи [43], у якій використано *кластери з оптимізованою пам'яттю*), на вирішення завдань машинного навчання моделей Apache Spark із застосуванням тестового набору даних Spark-Perf.

9.2. Моделі та методи машинного навчання Apache Spark Azure Hdinsight

Опис екземпляра кластера Azure HDInsight

Дослідження продуктивності роботи кластера Apache Spark проводять на платформі Azure HDInsight для віртуальних машин *загального призначення*, а саме версії 3.6. Інформацію про версії супутніх компонентів Azure HDInsight наведено в табл. 9.1.

Таблиця 9.1

Версії компонент HDInsight 3.6

Компоненти ПЗ	Версії ПЗ
Apache Hadoop и YARN	2.7.3
Apache Tez	0.7.0
Apache Pig	0.16.0
Apache Hive	2.1.0, 1.2.1
Apache Tez Hive2	0.8.4
Apache Ranger	0.7.0
Apache HBase	1.1.2
Apache Sqoop	1.4.6
Apache Oozie	4.2.0
Apache Zookeeper	3.4.6
Apache Storm	1.1.0
Apache Mahout	0.9.0 +
Apache Phoenix	4.7.0
Apache Spark	2.3.0, 2.2.0, 2.1.0
Apache Livy	0,4, 0,4, 0,3
Apache Kafka	1.1, 1.0
Apache Ambari	2.6.0
Apache Zeppelin	0.7.0
Mono	4.2.1

У цьому дослідженні використано Azure, який застосовує декілька серій обчислювальних машин *загального призначення*, у кожній із яких надано збалансоване співвідношення ресурсів ЦП і пам'яті. Такий клас є оптимальним рішенням для тестування й розроблення, створення невеликих і середніх (за обсяг) баз даних, а також вебсерверів із малим і середнім обсягом трафіка.

Для тестування вибрано серію віртуальних машин Dv2: машини цієї серії працюють на процесорах Intel® Xeon® 8171M із тактовою частотою 2,1 ГГц (Skylake), або Intel® Xeon® E5-2673 v4 2,3 ГГц (Broadwell), або процесорах Intel® Xeon® E5-2673 v3 2,4 ГГц (Haswell) із підтримкою технології Intel Turbo Boost 2.0.

У табл. 9.2 наведено дані про кількість віртуальних ЦП, дисків даних і мережевих адаптерів, а також про пропускну спроможність сховища

для кожного розміру віртуальної машини серії Dv2, що підтримує Azure HDInsight, а в табл. 9.3 наведено вартість їхнього використання.

Таблиця 9.2

Характеристики екземпляра Dv2

Характеристики	Розміри (**Standard**)		
	D3_v2	D4_v2	D5_v2
Віртуальних ЦП	4	8	16
Пам'ять, ГБ	14	28	56
Тимчасове сховище (SSD), ГБ	200	400	800
Максимальна пропускна спроможність тимчасового сховища: операцій введення – виведення на секунду / швидкість читання (Мбіт/с) / швидкість запису (Мбіт/с)	12 000 / 187 / 93	24 000 / 375 / 187	48 000 / 750 / 375
Максимальна кількість дисків даних	16	32	64
Пропускна спроможність, операцій уведення – виведення на секунду	16 × 500	32 × 500	64 × 500
Максимальна кількість мережевих адаптерів / очікувана пропускна спроможність мережі (Мбіт/с)	4 / 3 000	8 / 6 000	8 / 12 000

Таблиця 9.3

Вартість використання вузлів серії Dv2

Екземпляри	CPU, шт.	Оперативна пам'ять, ГБ	Ціна за ОС, $/год	Ціна, $/год	Повна вартість, $/год
D1_v2	1	3,5	0,057	0,017	0,074
D2_v2	2	7,0	0,114	0,033	0,147
D3_v2	4	14,0	0,229	0,066	0,295
D4_v2	8	28,0	0,458	0,132	0,590
D5_v2	16	56,0	0,916	0,264	1,180

Одними з найважливіших компонент Azure HDInsight є Apache Hadoop та планувальник YARN – підвищення продуктивності роботи напряму – залежить від їхніх конфігурацій. Apache Hadoop складається із двох основних компонент – розподіленої файлової системи (HDFS), яка забезпечує передавання та зберігання даних, і модуля управління ресурсами Hadoop

YARN, який забезпечує опрацювання завдань. Завдяки можливостям зберігання й опрацювання на кластері, можна використовувати програми MapReduce для забезпечення розподіленого опрацювання даних.

Платформа YARN складається із двох основних служб, що виконують як процеси на вузлах кластера:

диспетчера ресурсів (Resource Manager);

диспетчера вузлів (Node Manager).

Диспетчер ресурсів (Resource Manager) надає обчислювальні ресурси кластера процесам. Ці ресурси надано як контейнери, кожен із яких складається з виділених ядер ЦП та обсягу оперативної пам'яті.

Диспетчер вузлів (Node Manager) запускає завдання, із яких складаються програми, а потім повідомляє про хід виконання та стан програми до Application Master. Він, своєю чергою, відправляє звіт про стан програм до Resource Manager, який повертає результати клієнту.

YARN розгорнуто на всіх типах кластерів HDInsight. Для забезпечення високого рівня доступності Resource Manager розгортають у двох екземплярах (первинному та вторинному), що виконують на першому та другому головних вузлах кластера, відповідно. Одночасно може бути активним тільки один екземпляр Resource Manager. Примірники Node Manager виконують на робочих вузлах, доступних у кластері.

У табл. 9.4 наведено основні параметри конфігурації компонент Apache Spark – Hadoop YARN, Hadoop HDFS та Apache Tez, а також конфігурацій самого Apache Spark.

Таблиця 9.4

Склад основних параметрів конфігурації Apache Spark

Назви	Описи	Значення за замовчуванням
1	2	3
Hadoop YARN		
yarn.resourcemanagerscheduler.class	Клас, який слід використовувати як планувальник ресурсів	org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler

Продовження табл. 9.4

1	2	3
yarn.scheduler minimum-allocation-mb	Мінімальний розподіл для кожного запиту контейнера у RM, у МБ. Запити на пам'ять є нижчими, ніж це, будуть кидати InvalidResourceRequestException	1 024
yarn.scheduler. maximum-allocation-mb	Максимальний розподіл для кожного запиту контейнера у RM, у МБ. Запити на пам'ять, що перевищують це, викличуть InvalidResourceRequestException	8 192
yarn.scheduler. minimum-allocation-vcores	Мінімальний розподіл для кожного запиту контейнера у RM у перерахунку на віртуальні ядра CPU. Запити, що є меншими за це, викликають InvalidResourceRequestException	1
yarn.scheduler. maximum-allocation-vcores	Максимальний розподіл для кожного запиту на контейнер у RM у перерахунку на віртуальні ядра CPU. Запити, що перевищують це, призведуть до порушення InvalidResourceRequestException	32
Hadoop HDFS		
dfs.namenode. safemode.threshold-pct	Відсоток блоків, який має задовольняти мінімальну вимогу реплікації, визначену dfs.namenode. replication.min. Значення є меншими або дорівнюють 0 означають, що не потрібно чекати певного відсотка блоків перед виходом у безпечний режим. Значення більше за 1 зробить безпечним режим постійним	0,999f
dfs.datanode.du.reserved	Зарезервований простір у байтах на обсяг. Завжди залишайте цей простір вільним для використання простору без dfs	0
Apache Tez		
tez.am.resource. memory.mb	Обсяг пам'яті у МБ, який має використовувати AppMaster	1 024

Закінчення табл. 9.4

1	2	3
Apache Spark		
spark.driver. memoryOverhead	Обсяг пам'яті, що виділено, для того, щоб бути розподіленим на одного драйвера у кластерному режимі, у МБ, якщо не вказано інше. Це пам'ять, на яку припадають такі речі, як накладні витрати на ВМ, інтерновані рядки, інші накладні витрати тощо. Це має тенденцію до зростання з розміром контейнера (зазвичай 6 – 10 %)	driverMemory * 0,10, мінімум – 384
spark.executor.cores	Кількість ядер, які слід використовувати кожному виконавцеві	1
spark.executor.instances	Кількість виконавців, які мають працювати	
spark.executor. memory	Обсяг пам'яті, що використовують для кожного виконавця, у тому самому форматі, що й рядки пам'яті JVM із суфіксом одиниці розміру ("k", "m", "g" або "t")	1g
spark.executor. memoryOverhead	Обсяг off-heap пам'яті, що виділяють кожному виконавцю, у МБ, якщо не вказано інше. Це пам'ять, яка йде на такі речі, як накладні витрати на VM, інтерновані рядки, інші накладні витрати тощо. Вона має тенденцію до зростання, залежно від розміру виконавця (зазвичай на 6 – 10 %)	executorMemory * 0,10, мінімум – 384

Вибір значень для наведених параметрів впливає на продуктивність кластера.

9.3. Характеристика пакета Machine Learning Library Apache Spark

Машинне навчання є частиною більш широкого поняття – штучного інтелекту. Машинне навчання використовує великі набори даних для того, щоб виявити закономірності та на їхній основі ухвалювати рішення чи робити прогнози, й отже, розробляти моделі. Ці моделі "тренують" для

конкретної проблеми на основі даних, узятих із певної предметної галузі (або кількох галузей). Саме автоматизоване ухвалення рішень робить машинне навчання досить перспективним.

Машинне навчання можна класифікувати на дві категорії (класи), з огляду на підхід: контрольоване (із вчителем) і неконтрольоване (без учителя).

Контрольоване навчання працює з набором даних, який містить як вхідні дані, так і бажані вихідні дані, наприклад, набір даних, що містить різні характеристики об'єкта, а також його вартість, розраховану на їхній основі. Алгоритми, що належать до контрольованого навчання, своєю чергою, можна розподілити на два підкласи:

алгоритми класифікації – алгоритми, що роблять чіткий висновок про належність до певної категорії;

алгоритми регресії – алгоритми, які, замість чіткої відповіді, надають діапазон можливих значень.

Неконтрольоване навчання, своєю чергою, працює з набором даних, у якому є тільки вхідні значення. Алгоритми такого типу намагаються визначити внутрішню структуру вхідних даних, щоб далі робити висновки щодо залежності одних даних від інших.

Apache Spark MLlib – це бібліотека машинного навчання Apache Spark, що складається із загальних алгоритмів навчання, як-от класифікація, регресія, кластеризація, спільна фільтрація, зниження розмірності та основні оптимізації примітивів, а також допоміжних утиліт для лінійної алгебри, статистики, опрацювання даних тощо.

Великий обсяг даних, який часто вимірюють терабайтами або навіть петабайтами, – це те, що робить Spark MLlib досить важливим із погляду ефективного опрацювання даних.

Найбільш поширеними сценаріями використання машинного навчання є такі:

моніторинг безпеки та виявлення шахрайства;

рекомендація продуктів та оптимізація персоналізованого маркетингу;

показ таргетованої реклами;

опрацювання природнього мовлення.

Бібліотека MLlib складається із двох програмних пакетів:

spark.mllib, що містить оригінальний API, побудований зверху RDD;

spark.ml, що дає API більш високого рівня, побудований зверху DataFrames для конструювання конвеєрів машинного навчання, відомих як ML Pipelines.

Нині пакет spark.ml є основним API машинного навчання для Spark, а spark.mllib перебуває в режимі підтримки.

Spark MLlib підтримує кілька типів даних, як локальних, так і розподілених, для подання вхідних даних та відповідних результатів. Типи даних Spark MLlib розподіляють на такі категорії:

локальний вектор (local vector);

розмічену точку (labeled point);

локальну матрицю (local matrix);

розподілену матрицю (distributed matrix).

Локальний вектор (local vector) – це вектор, який зберігають на одному комп'ютері та може містити як цілі значення, так і значення із рухомою точкою. Локальний вектор розподіляють на два види:

щільні вектори (dense vector) – вектори, які становлять звичайний вектор із нецілими значеннями в ролі вхідних значень;

розріджені вектори (sparse vector) – вектор, аргументами якого є кількість ознак, а також два масиви, у першому з яких указано індекси, а у другому – значення.

Розмічена точка (labeled point) – це локальний вектор (щільний або розріджений), пов'язаний із міткою, яка може набирати будь-яке дійсне значення під час використання в завданні регресії та значення, починаючи з 0, для завдань класифікації. У MLlib розмічені точки використовують у контрольованих алгоритмах навчання.

Локальна матриця (local matrix) – матриця із цілочисельними індексами рядків і стовпців та значеннями з рухомою точкою, що зберігають на одному комп'ютері. MLlib підтримує щільні матриці, чиї значення записів зберігають в одному масиві нецілими значеннями, і розріджені матриці, ненульові значення записів яких зберігають у форматі стислих розріджених стовпців.

Розподілена матриця (distributed matrix) – матриця із цілочисельними індексами рядків і стовпців (тип long) та значеннями з рухомою точкою, які зберігають у вигляді однієї або декількох RDD. Є чотири типи розподілених матриць:

RowMatrix – становить розподілену матрицю, орієнтовану на рядки, без значущих індексів. Кожен рядок матриці становить локальний вектор та містить рядок із RDD;

IndexedRowMatrix – те саме, що і RowMatrix, але з різницею в тому, що тут є значущі індекси рядків;

CoordinateMatrix – становить розподілену матрицю, де кожен запис є кортежем, який характеризується індексом рядка, індексом стовпця та значенням запису;

BlockMatrix – становить розподілену матрицю, де кожен запис є кортежем, що характеризується парою індексів блоку та підматрицею.

Spark MLlib надає алгоритми машинного навчання для вирішення завдань таких типів:

базової статистики;

регресії;

класифікації;

рекомендаційної системи;

кластеризації;

зниження розмірності;

визначення характеристик;

оптимізації.

Базова статистика містить основні методи машинного навчання. До них входять:

підсумкова статистика, як-от максимум, мінімум, середнє значення, дисперсія, кількість ненульових значень, а також загальна кількість;

кореляції між декількома рядами за допомогою двох методів – співвідношення Пірсона та Спірмена;

розшарована вибірка, до якої належать методи sampleBykey і sampleByKeyExact;

перевірка гіпотез за допомогою критерію Пірсона хі-квадрат на те, що результат є статистично значущим чи випадковим;

генерація випадкових даних за допомогою методів RandomRDD, Normal і Poisson.

Регресійний аналіз – це статистичний процес оцінювання взаємозв'язків між змінними. Він охоплює безліч методів моделювання й аналізу декількох змінних, коли основну увагу приділяють взаємозв'язку між залежною змінною та однією або декількома незалежними змінними. Регресійний аналіз допомагає зрозуміти, як типове значення залежної змінної змінюється, коли змінюється будь-яка з незалежних змінних, тоді як інші незалежні змінні залишаються фіксованими. Регресійний аналіз широко використовують для прогнозування, а також для аналізу того, які з незалежних змінних пов'язано із залежною змінною та якою є форма цих взаємозв'язків.

Бібліотека MLlib дає реалізацію таких видів регресії:

лінійної регресії (Linear regression);

узагальненої лінійної регресії (Generalized linear regression);

регресії дерева рішень (Decision tree regression);

регресії випадкового лісу (Random forest regression);

регресії дерева градієнтного бустингу (Gradient-boosted tree regression);

регресії виживання (Survival regression);

ізотонічної регресії (Isotonic regression).

Класифікація – це завдання визначення, до якого з набору категорій належить нове спостереження, ґрунтуючись на навчальному наборі даних, що містять спостереження з відомою належністю до певної категорії. Прикладом класифікації є зарахування електронного листа до класу "спам" або "не спам".

До складу бібліотеки MLlib входять такі реалізації алгоритмів класифікації:

логістична регресія (Logistic regression);

біноміальна логістична регресія (Binomial logistic regression);

поліноміальна логістична регресія (Multinomial logistic regression);

класифікатор дерева рішень (Decision tree classifier);

класифікатор випадкового лісу (Random forest classifier);

класифікатор дерева із градієнтним бустингом (Gradient-boosted tree classifier);

багатошаровий перцептрон-класифікатор (Multilayer perceptron classifier);

лінійна опорна векторна машина (Linear Support Vector Machine);

класифікатор "Один проти всіх" (One-vs-Rest classifier);

баєсова класифікація (Naive Bayes).

Рекомендаційна система – це підклас системи фільтрації інформації, який прагне передбачити "рейтинг", який користувачі надають елементу. Рекомендаційні системи зазвичай видають список рекомендацій одним із двох способів – за допомогою колаборативної й контентної фільтрації або особистісного підходу. Колаборативна фільтрація підходить для побудови моделі на основі поведінки користувача в минулому, а також аналогічних рішень, ухвалених іншими користувачами. Потім цю модель використовують для прогнозування елементів або їхніх оцінок, у яких може зацікавитися користувач. Підходи контентної фільтрації

використовують ряд окремих характеристик елементу, щоб рекомендувати додаткові елементи з аналогічними властивостями. Іноді ці підходи об'єднують у гібридні рекомендаційні системи. Для колаборативної фільтрації за допомогою засобів бібліотеки MLlib можна використовувати реалізацію алгоритму чергування найменших квадратів (ALS).

Кластеризація – це завдання групування набору об'єктів у такий спосіб, щоб об'єкти в одній і тій самій групі були більше схожими один на одного, ніж об'єкти в інших групах. Отже, це основне завдання інтелектуального аналізу даних і загальна методика статистичного аналізу даних, яку використовують у багатьох галузях, включно з машинним навчанням, розпізнаванням образів, аналізом зображень, пошуком інформації, біоінформатикою, стисненням даних і комп'ютерною графікою.

До складу бібліотеки MLlib входять такі реалізації алгоритмів кластеризації:

K-середніх (K-Means);

латентне розміщення Діріхле (Latent Dirichlet allocation, LDA);

бісектриса K-середніх (Bisecting K-Means);

модель гауссової суміші (Gaussian Mixture Model, GMM).

Зниження розмірності – це процес зменшення кількості розглянутих випадкових величин шляхом визначення набору головних змінних. Його може бути розділено на вибір ознак і пошук ознак:

вибір ознак – визначення підмножини вихідних змінних (також названих об'єктами або атрибутами);

пошук ознак – перетворення даних у багатовимірному просторі на простір меншого розміру. Перетворення даних може бути лінійним, як в аналізі головних компонентів (PCA), але також є безліч методів нелінійного зниження розмірності.

Процес пошуку ознак починається з початкового набору даних вимірювань і створення похідних ознак, призначених для полегшення наступних етапів навчання й узагальнення.

До складу бібліотеки MLlib входять такі реалізації алгоритмів:

а) алгоритми пошуку ознак:

TF-IDF;

Word2Vec;

CountVectorizer;

FeatureHasher;

б) алгоритми вибору ознак:

VectorSlicer;

RFormula;

ChiSqSelector.

Оптимізація – це вибір кращого елементу з деякого набору доступних альтернатив з урахуванням деякого критерію. У найпростішому випадку завдання оптимізації полягає в максимізації або мінімізації реальної функції шляхом систематичного вибору вхідних значень із допустимого набору й обчислення значення функції. Узагальнення теорії та методів оптимізації на інші формулювання охоплює велику галузь прикладної математики. У більш загальному сенсі, оптимізація містить пошук "найкращих доступних" значень деякої цільової функції для заданої галузі (або вхідних даних), включно з різними типами цільових функцій і різних типів доменів.

Бібліотека MLlib дає реалізацію таких алгоритмів оптимізації, як:

градієнтний спуск (Gradient descent);

стохастичний градієнтний спуск (Stochastic gradient descent, SGD);

алгоритм BFGS з обмеженою пам'яттю (Limited-memory BFGS, L-BFGS).

9.4. Оцінювання продуктивності кластера Apache Spark на основі тестових наборів даних

Щоб вибрати оптимальний розмір (параметри конфігурації) кластера, необхідно вимірювати продуктивність кластера та змінювати його розмір, відповідно до визначених результатів. Для цього необхідно провести тестування продуктивності кластера. Тестування продуктивності – це процес запуску змодельованих робочих навантажень на різних віртуальних машинах для вимірювання того, наскільки ефективно їх буде виконувати в робочих навантаженнях.

Для тестування продуктивності кластера Apache Spark є декілька бенчмарків: Spark-Bench, HiBench, BigDataBench-Spark та Spark-Perf [26].

Spark-Bench – це гнучка система для моделювання, порівняння, тестування та бенчмаркінгу застосунків Spark, а також самої системи Spark, від IBM. Spark-Bench спочатку використовували як набір для бенчмаркінгу для визначення часових характеристик алгоритмів машинного

навчання. Але із часом перетворився на дуже гнучкий фреймворк, придатний для багатьох використань у багатьох класах завдань.

Spark-Bench має чотири категорії тестування навантажень:

застосунки машинного навчання, що підтримують Spark MLLib;

застосунки для обчислення граф, що підтримують Spark GraphX;

SQL-застосунки, що підтримують стандартною службою Spark SQL та Hive;

потокові застосунки, що підтримують Spark DStream.

Тестування бібліотеки машинного навчання проводять за допомогою трьох найбільш поширених алгоритмів для регресії, класифікації та рекомендацій, а саме:

алгоритму логістичної регресії (LR);

методу опорних векторів (SVM);

алгоритму матричної факторизації (MF).

HiBench – це набір бенчмарків опрацювання великих даних, розроблений компанією Intel, який допомагає оцінювати різні фреймворки великих даних із погляду швидкості, пропускної спроможності та використання системних ресурсів. Він містить набір робочих навантажень Hadoop, Spark і навантажень потокового передавання, включно з такими алгоритмами, як Sort, WordCount, TeraSort, Sleep, SQL, PageRank, індексування Nutch, Bayes, K-Means, NWeight, розширений DFSIO та ін.

HiBench застосовує шість категорій тестування навантаження:

мікробенчмарки;

бенчмарки машинного навчання;

бенчмарки роботи з SQL-запитами;

бенчмарки вебпошуку;

бенчмарки роботи із графами;

бенчмарки для оцінювання потокового передавання.

Пакет бенчмарку машинного навчання покриває тестами такі алгоритми:

баєсову класифікацію (Bayesian Classification, Naive Bayes);

кластеризацію K-середніх (K-Means clustering, K-Means);

логістичну регресію (Logistic Regression, LR);

чергування найменших квадратів (Alternating Least Squares, ALS);

градієнтні дерева (Gradient Boosting Trees, GBT);

лінійну регресію (Linear Regression);

прихований розподіл Діріхле (Latent Dirichlet Allocation, LDA);

аналіз основних компонентів (Principal Components Analysis, PCA);
випадковий ліс (Random Forest, RF);
машину опорних векторів (Support Vector Machine, SVM);
розкладання за сингулярними значеннями (Singular Value Decomposition, SVD).

BigDataBench-Spark – це інтегрована частина проєкту пакета бенчмарків великих даних із відкритим сирцевим кодом BigDataBench. До складу пакета входять мікробенчмарки та тести деяких алгоритмів машинного навчання, як-от K-середніх кластерів (K-Means), чергування найменших квадратів (ALS) та баєсова класифікація (Naive Bayes).

Spark-Perf – це бенчмарк для комплексного оцінювання продуктивності кластера, розроблений компанією Databricks, яка займалася розробленням системи Apache Spark. До бенчмарку входять комплекти тестів на перевірку ефективності таких компонент, як ядро Spark, PySpark, Spark Streaming та MLlib. Особливостями бенчмарку Spark-Perf є те, що він має можливість налаштувати потрібну конфігурацію тестів, а також автоматично завантажувати та створювати потрібну конфігурацію кластера Spark.

Spark-Perf має найбільше покриття тестами алгоритмів машинного навчання серед аналогічних пакетів бенчмарків. Далі наведено повний список алгоритмів, для яких у Spark-Perf проводять тестування:

узагальнена модель лінійної регресії (GLM-regression);
узагальнена модель лінійної класифікації (GLM-classification);
баєсова класифікація (Naive Bayes);
баєсова класифікація Бернуллі (Naive-Bayes-bernoulli);
дерево рішень (Decision-tree);
чергування найменших квадратів (ALS);
кластеризація K-середніх (K-Means);
модель гауссової суміші (GMM);
розкладання за сингулярними значеннями (SVD);
аналіз основних компонентів (PCA);
зведена статистика (Summary-statistics);
матричне множення (Block-matrix-mult);
кореляція Пірсона (Pearson);
кореляція Спірмена (Spearman);
розподіл хі-квадрат (chi-sq-feature/gof/mat);
розподілена презентація слів Word2Vec (word2vec);
алгоритм пошуку асоціативних правил (FP-growth).

9.5. Моделі машинного навчання в бенчмарку Spark-Perf

Оскільки Spark-Perf має найбільше покриття тестовими наборами бібліотеки машинного навчання Apache, його було вибрано для здійснення вимірювання продуктивності кластера Apache Spark. У табл. 9.5 наведено детальну інформацію для використання тестів машинного навчання пакета Spark-Perf.

Таблиця 9.5

Статистичні характеристики тестів машинного навчання Spark-Perf

Кодові назви	Назви тестів	Кількість випробувань	Кількість тестових примірників
1	2	3	4
Алгоритми регресії та класифікації			
glm-regression	Generalized Linear Regression Model	10	5 000
glm-classification	Generalized Linear Classification Model	10	5 000
naive-bayes	Naive Bayes	10	5 000
Алгоритми з деревом			
decision-tree	Decision Tree	16 × 10	5 000
Алгоритми рекомендацій			
als	Alternating Least Squares	10	500
Алгоритми кластеризації			
kmeans	K-Means clustering	10	5 000
lda	Latent Dirichlet allocation	10	–
pic	Power Iteration Clustering	10	–
gmm	Gaussian Mixture Model	10	5 000
Статистичні алгоритми			
summary-statistics	Summary Statistics	10	–
pearson	Pearson's Correlation	10	50 000
spearman	Spearman's Correlation	10	50 000
chi-sq-feature/gof/mat	Chi-square Tests	10	100 000
Алгоритми лінійної алгебри			
svd	Singular Value Decomposition	10	–
pca	Principal Component Analysis	10	–
block-matrix-mult	Matrix Multiplication	10	–

Закінчення табл. 9.5

1	2	3	4
Алгоритми пошуку асоціативних правил			
fp-growth	FP-growth frequent item sets	10	–
prefix-span	PrefixSpan	10	–
Алгоритми пошуку ознак			
word2vec	Word2Vec distributed presentation of words	10	–

Указані дані здобуті на основі статистичного опрацювання результатів навчання та тестування.

9.6. Методичне забезпечення організації процесу тестування Spark-кластера

Для того щоб провести тестування кластера Spark, використовують репозитарій Spark-Perf, який покриває тестами ядро Spark та бібліотеку Machine Learning (MLlibrary).

Крок 1. Клонування репозитарія із сирцевим кодом бенчмарку на клієнтську машину. Рекомендовано клонувати модифікований репозитарій (форк), у якому вже є оптимізована конфігурація для тестування кластера HDInsight невеликого розміру та налаштована підтримка служби моніторингу Spark-Measure.

Для цього необхідно виконати таку команду:

git clone https://github.com/HackedBeat/spark-perf.

Першим кроком клонуймо репозитарій бенчмарку (рис. 9.1), виконавши таку команду:

git clone https://github.com/deib-polimi/spark-perf.git

```
vlads@vlads-notebook:~$ mkdir spark-perf
vlads@vlads-notebook:~$ cd spark-perf/
vlads@vlads-notebook:~/spark-perf$ git clone https://github.com/deib-polimi/spark-perf.git
Cloning into 'spark-perf'...
remote: Enumerating objects: 3111, done.
remote: Total 3111 (delta 0), reused 0 (delta 0), pack-reused 3111
Receiving objects: 100% (3111/3111), 3.17 MiB | 4.55 MiB/s, done.
Resolving deltas: 100% (1429/1429), done.
```

Рис. 9.1. **Клонування репозитарія Spark-Perf із тестами**

Крок 2. Необхідно додати сервіс моніторингу й метрики SparkMeasure, для чого потрібно модифікувати скрипт запуску тестів.

Для того щоб **налаштувати SparkMeasure вручну**, необхідно виконати такі кроки:

додати до файлів mllib-tests/project/MLlibTestsBuild.scala та spark-tests/project/SparkTestsBuild.scala два такі рядки, як показано на рис. 9.2 та 9.3:

"org.apache.spark" %% "spark-sql" % "2.0.0" % "provided",
"ch.cern.sparkmeasure" %% "spark-measure" % "0.13"

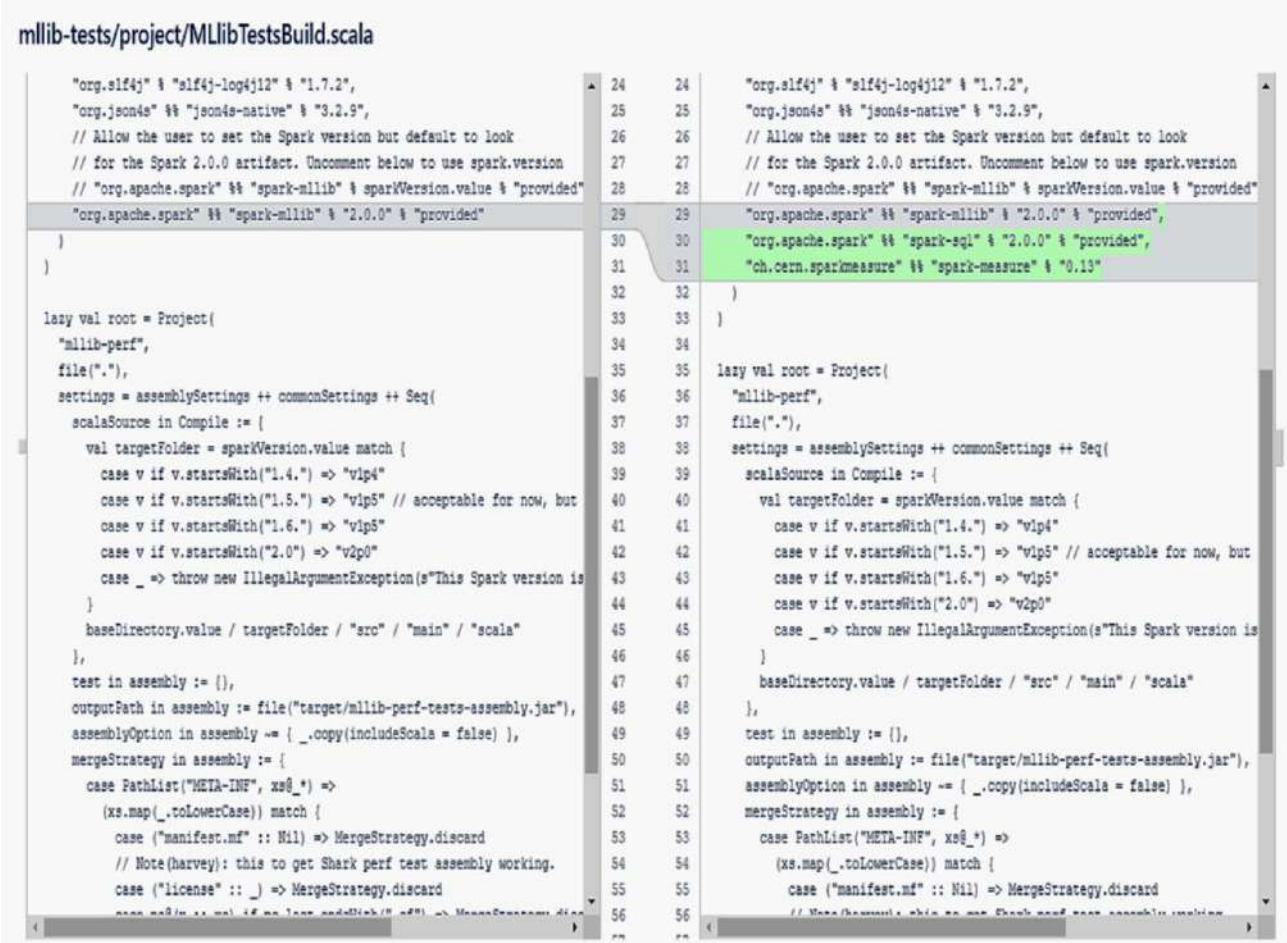


Рис. 9.2. **Підключення необхідних бібліотек до файлу MLlibTestsBuild.scala**

spark-tests/project/SparkTestsBuild.scala

```
import sbt._
import sbt.Keys._
import sbtassembly.Plugin._
import AssemblyKeys._

object SparkTestsBuild extends Build {
  lazy val root = Project(
    "spark-perf",
    file("."),
    settings = assemblySettings ++ Seq(
      organization := "org.spark-project",
      version := "0.1",
      scalaVersion := sys.props.getOrElse("scala.version", default="2.11.8"),
      libraryDependencies ++= Seq(
        "net.sf.jopt-simple" % "jopt-simple" % "4.6",
        "org.scalatest" %% "scalatest" % "2.2.1" % "test",
        "com.google.guava" % "guava" % "14.0.1",
        "org.apache.spark" %% "spark-core" % "2.0.0" % "provided",
        "org.json4s" %% "json4s-native" % "3.2.9",
        "org.apache.commons" % "commons-math3" % "3.5"
      ),
      test in assembly := {},
      outputPath in assembly := file("target/spark-perf-tests-assembly.jar"),
      assemblyOption in assembly ~= { _.copy(includeScala = false) },
      mergeStrategy in assembly := {
        case PathList("META-INF", xs@_*) =>
          (xs.map(_.toLowerCase)) match {
            case ("manifest.mf" :: Nil) => MergeStrategy.discard
            // Note(harvey): this to get Shark perf test assembly working.
            case ("license" :: _) => MergeStrategy.discard
            case ps@(x :: xs) if ps.last.endsWith(".sf") => MergeStrategy.disc
            case _ => MergeStrategy.first
```

```
import sbt._
import sbt.Keys._
import sbtassembly.Plugin._
import AssemblyKeys._

object SparkTestsBuild extends Build {
  lazy val root = Project(
    "spark-perf",
    file("."),
    settings = assemblySettings ++ Seq(
      organization := "org.spark-project",
      version := "0.1",
      scalaVersion := sys.props.getOrElse("scala.version", default="2.11.8"),
      libraryDependencies ++= Seq(
        "net.sf.jopt-simple" % "jopt-simple" % "4.6",
        "org.scalatest" %% "scalatest" % "2.2.1" % "test",
        "com.google.guava" % "guava" % "14.0.1",
        "org.apache.spark" %% "spark-core" % "2.0.0" % "provided",
        "org.apache.spark" %% "spark-sql" % "2.0.0" % "provided",
        "org.json4s" %% "json4s-native" % "3.2.9",
        "org.apache.commons" % "commons-math3" % "3.5",
        "ch.cern.sparkmeasure" %% "spark-measure" % "0.13"
      ),
      test in assembly := {},
      outputPath in assembly := file("target/spark-perf-tests-assembly.jar"),
      assemblyOption in assembly ~= { _.copy(includeScala = false) },
      mergeStrategy in assembly := {
        case PathList("META-INF", xs@_*) =>
          (xs.map(_.toLowerCase)) match {
            case ("manifest.mf" :: Nil) => MergeStrategy.discard
            // Note(harvey): this to get Shark perf test assembly working.
            case ("license" :: _) => MergeStrategy.discard
```

Рис. 9.3. **Підключення необхідних бібліотек до файла SparkTestsBuild.scala**

Крок 3. Далі треба модифікувати код таких файлів, як показано на рис. 9.4 й 9.5.

mllib-tests/v2p0/src/main/scala/mllib/perf/TestRunner.scala
spark-tests/src/main/scala/spark/perf/TestRunner.scala

```
import org.apache.spark.sql.SparkSession
val sparkSession = SparkSession
.builder()
.appName("TestRunner: " + testName)
.getOrCreate();
```

Рис. 9.4. **Додавання Spark Measure до скрипту запуску тестів (частина 1)**

val sc = sparkSession.sparkContext

…

val stageMetrics = ch.cern.sparkmeasure.StageMetrics(sparkSession)

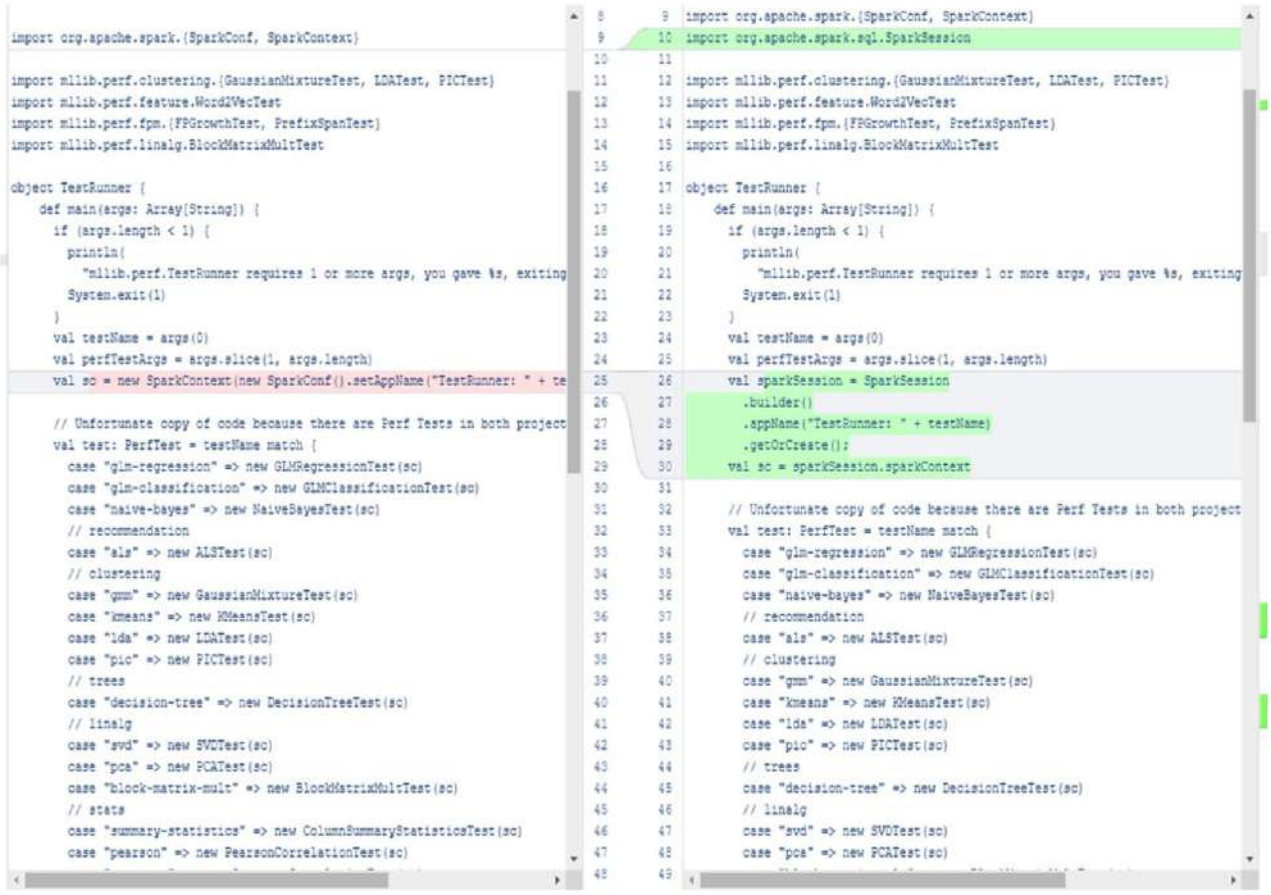
stageMetrics.begin()

…

stageMetrics.end()

stageMetrics.printReport()

mllib-tests/v2p0/src/main/scala/mllib/perf/TestRunner.scala



Закінчення рис. 9.4

mllib-tests/v2p0/src/main/scala/mllib/perf/TestRunner.scala

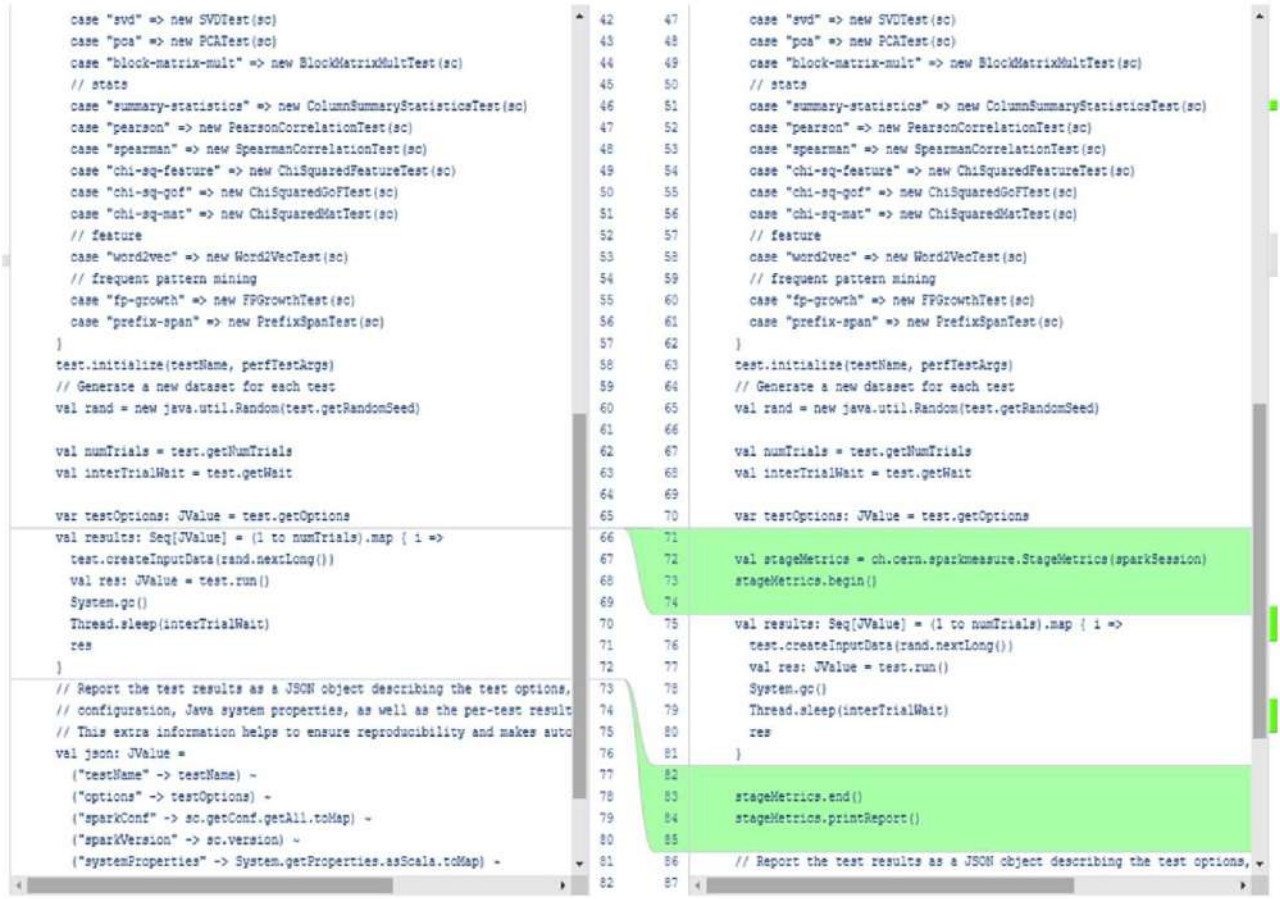


Рис. 9.5. **Додавання Spark Measure до скрипту запуску тестів (частина 2)**

Модифікацію файлу spark-tests/src/main/scala/spark/perf/TestRunner.scala здійснюють аналогічно тому, як показано на рис. 9.4 й 9.5.

Крок 4. Далі потрібно створити конфігурацію для виконуваних тестів. Для цього копіюймо файл config.py.template з назвою config.py (рис. 9.6) за допомогою таких команд:

cd spark-perf/config/
cp config.py.template config.py

```
vlads@vlads-notebook:~/spark-perf$ cd spark-perf/config/
vlads@vlads-notebook:~/spark-perf/spark-perf/config$ cp config.py.template config.py
vlads@vlads-notebook:~/spark-perf/spark-perf/config$
```

Рис. 9.6. **Копіювання файлу налаштувань**

Крок 5. У файлі config.py треба налаштувати конфігурацію для проведення тестування.

SPARK_CLUSTER_URL – URL кластера Spark, який буде використано під час запуску завдань. Має бути задано значення yarn (рис. 9.7), оскільки за замовчуванням кластери Spark на Azure HDInsight використовують саме планувальник Yarn.

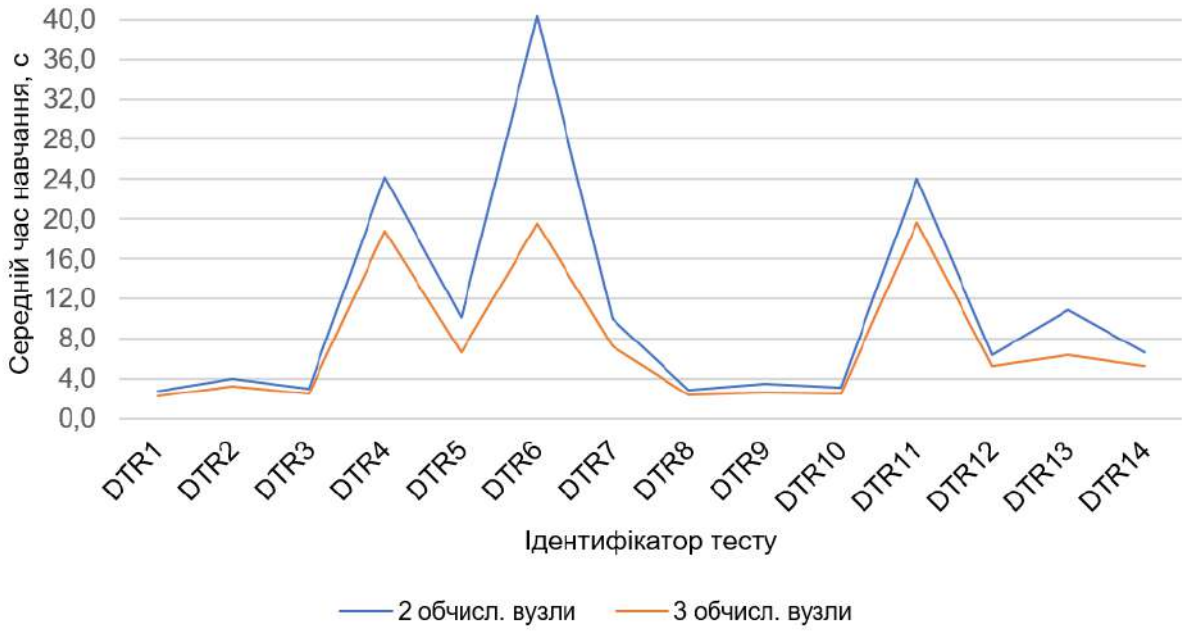


Рис. 9.7. **Порівняльний аналіз часу навчання на гомогенних кластерах із 2 та 3 обчислювальними вузлами**

Секції RUN та PREP, у яких можна вказати, які тести треба запустити та треба підготовляти. Процес підготовки складається із завантаження third-party пакетів і компіляції коду (тільки для Scala).

Доступні такі набори тестів:

SPARK_TESTS – набір тестів Spark Core на Scala;

PYSPARK_TESTS – набір тестів Spark Core на Python;

STREAMING_TESTS – набір тестів Spark Streaming на Scala;

MLLIB_TESTS – набір тестів Spark MlLib на Scala;

PYTHON_MLLIB_TESTS – набір тестів Spark MlLib на Python.

У цьому разі вмикаймо тести SPARK_TESTS і MLLIB_TESTS.

Після того як роботу над конфігурацією файлу config.py завершено, можна завантажити сирцевий код Spark-Perf на master-вузол кластера. Для цього потрібно виконати такі кроки:

1. Заархівувати каталог, який було створено під час клонування репозитарія із GitHub із новою конфігурацією. Для зменшення розміру архіву можна видалити папку .git перед архівацією.

2. Завантажити створений архів на master-вузол за допомогою команди scp:

scp spark-perf.zip sshuser@sparkmeasurehneu-ssh.azurehdinsight.net

9.7. Проведення експериментальних досліджень для кластерів із різною конфігурацією та аналіз визначених результатів

У результаті проведених обчислювальних експериментів було визначено результати для таких конфігурацій кластера Apache Spark (табл. 9.6).

Таблиця 9.6

Конфігурації кластерів для тестування

Типи кластерів	Master-вузли		Worker-вузли	
	кількість, шт.	конфігурації	кількість, шт.	конфігурації
Гомогенний	2	Standard_D3_v2	2	Standard_D3_v2
	2	Standard_D3_v2	3	Standard_D3_v2
Гетерогенний	2	Standard_D3_v2	2	Standard_D4_v2
	2	Standard_D3_v2	3	Standard_D4_v2

Аналіз результатів для гомогенних кластерів

Для тестування продуктивності Apache Spark розгорнуто 2 гомогенні кластери із 2 та 3 робочими вузлами типу Standard_D3_v2. Для кожного з них були розраховано оптимальні конфігурації, залежно від наявного обсягу ресурсів. Характеристики тестових кластерів наведено в табл. 9.7.

Таблиця 9.7

Опис тестової установки для гомогенних кластерів

Назви характеристик	Значення характеристики для кластера	
	2 обчислювальні вузли	3 обчислювальні вузли
1	2	3
Конфігурація master-вузлів	2 x Standard_D3_v2	3 × Standard_D3_v2
Конфігурація worker-вузлів	2 x Standard_D3_v2	3 × Standard_D3_v2
Планувальник ресурсів	Capacity Scheduler	Capacity Scheduler
Ступінь паралелізму	32	40

Закінчення табл. 9.7

1	2	3
Кількість вузлів	4	5
Застосовано ядер	8 + 8 = 16	8 + 12 = 20
Застосовано оперативної пам'яті	28 + 28 = 56	28 + 42 = 70
Процесор на вузлах	4	4
Оперативна пам'ять на вузлах	14	14

Оптимальні параметри параметрів конфігурації кластера Apache Spark наведено в табл. 9.8.

Таблиця 9.8

Оптимальні параметри конфігурації кластера Apache Spark для гомогенних кластерів

Параметри	Значення параметрів	
	2 обчислювальні вузли	3 обчислювальні вузли
spark.executor.instances, шт.	4	5
spark.yarn.executor.memoryOverhead, Гб	1,05	1,05
spark.executor.memory, Гб	9,45	9,45
spark.yarn.driver.memoryOverhead, Гб	1,05	1,05
spark.driver.memory, Гб	9,45	9,45
spark.executor.cores, шт.	3	3
spark.driver.cores, шт.	3	3
spark.default.parallelism, шт.	32	40

За результатами тестування здійснено статистичний аналіз за кожним із тестів для більш точного оцінювання результату та похибки вимірювань.

У табл. 9.9 і 9.10 наведено результати роботи тестового набору MLlib для кластерів із 2 та 3 обчислювальними вузлами, відповідно.

Таблиця 9.9

Результати роботи тестового набору MLlib на кластері із 2 обчислювальними вузлами

Назви тестів	Id	Тестування			Навчання		
		середнє значення, с	медіана, с	СКВ	середнє значення, с	медіана, с	СКВ
1	2	3	4	5	6	7	8
als	ML1	0,5533	0,5385	0,1313	2,8353	2,5715	0,8786
lda-1	ML2	20,9436	20,1795	3,2978	–	–	–

Закінчення табл. 9.9

1	2	3	4	5	6	7	8
lda-2	ML3	10,5780	9,6395	2,2355	–	–	–
svd	ML4	111,1700	110,8680	7,0772	–	–	–
pca	ML5	15,4386	15	1,3197	–	–	–
summary-statistics	ML6	0,9867	0,8115	0,6433	–	–	–
block matrix-mult	ML7	0,6581	0,5475	0,3127	–	–	–
pearson	ML8	9,4788	9,1955	0,9807	–	–	–
spearman	ML9	13,3195	12,486	2,3886	–	–	–
chi-sq-feature	ML10	9,6609	9,6280	0,2590	–	–	–
word2vec	ML11	16,6349	16,3100	1,2276	–	–	–
fp-growth	ML12	141,8274	141,886	2,1599	–	–	–
prefix-span	ML13	1,3820	1,1670	0,6361	–	–	–
glm-regression	ML14	0,1079	0,0965	0,0334	3,9974	3,3115	2,0178

Таблиця 9.10

Результати роботи тестового набору MLlib на кластері із 3 обчислювальними вузлами

Назви тестів	Id	Тестування			Навчання		
		середнє значення, с	медіана, с	СКВ	середнє значення, с	медіана, с	СКВ
1	2	3	4	5	6	7	8
als	ML1	0,5789	0,4905	0,2785	2,8962	2,4275	1,4485
lda-1	ML2	16,1753	14,786	3,4383	–	–	–
lda-2	ML3	9,9092	8,7255	3,1377	–	–	–
svd	ML4	69,5102	68,735	6,0706	–	–	–
pca	ML5	11,9162	11,0675	1,8336	–	–	–
summary-statistics	ML6	0,8025	0,5385	0,8230	–	–	–
block-matrix-mult	ML7	1,6885	2,0535	0,7723	–	–	–
pearson	ML8	5,5047	5,2575	0,7460	–	–	–

Закінчення табл. 9.10

1	2	3	4	5	6	7	8
spearman	ML9	7,9640	7,1115	1,9814	–	–	–
chi-sq-feature	ML10	5,7097	5,5085	0,4905	–	–	–
word2vec	ML11	10,2717	9,8130	1,4620	–	–	–
fp-growth	ML12	76,9748	76,558	2,0421	–	–	–
prefix-span	ML13	1,2251	1,0460	0,6032	–	–	–
glm-regression	ML14	0,1037	0,0945	0,0309	4,1855	3,1605	3,2729
glm-classification-1	ML15	0,0957	0,0885	0,0237	3,5659	2,9620	1,8340
glm-classification-2	ML16	0,1187	0,1095	0,0361	9,6496	8,5010	3,5480

У табл. 9.11 і 9.12 наведено результати роботи тестового набору "Дерево рішень" (Decision-Tree) для кластерів із 2 та 3 обчислювальними вузлами, відповідно.

Як метрику продуктивності роботи кластерів вибрано відношення середнього часу тестування до середнього часу навчання.

Таблиця 9.11

Результати роботи тестового набору "Дерево рішень" на кластері із 2 обчислювальними вузлами

Id тестів	Тестування			Навчання		
	середнє значення, с	медіана, с	СКВ	середнє значення, с	медіана, с	СКВ
1	2	3	4	5	6	7
DTR1	0,1311	0,0955	0,0876	2,7709	2,2875	1,3372
DTR2	0,1355	0,1190	0,0450	4,0486	3,5720	1,4339
DTR3	0,1272	0,0970	0,0640	2,9933	2,6075	1,3249
DTR4	0,1186	0,1045	0,0399	24,1196	23,1500	2,2689
DTR5	0,1748	0,1470	0,0881	10,1216	9,6640	2,1515
DTR6	0,4745	0,4450	0,0672	40,2908	39,5425	2,0521

Закінчення табл. 9.11

1	2	3	4	5	6	7
DTR7	0,1799	0,1505	0,1024	10,0306	9,5495	1,5853
DTR8	0,1144	0,0945	0,0450	2,9029	2,3735	1,3260
DTR9	0,1740	0,1620	0,0539	3,4727	3,0150	1,5447
DTR10	0,1113	0,0925	0,0477	3,0674	2,5880	1,4431
DTR11	0,1345	0,1205	0,0478	24,0404	23,2430	2,5686
DTR12	0,1630	0,1530	0,0387	6,3863	6,0190	1,5512
DTR13	0,2785	0,2560	0,0665	10,8813	10,4595	1,9575
DTR14	0,1626	0,1235	0,0801	6,6291	5,9055	1,8097

Таблиця 9.12

Результати роботи тестового набору "Дерево рішень" на кластері із 3 обчислювальними вузли

Id тестів	Тестування			Навчання		
	середнє значення, с	медіана, с	СКВ	середнє значення, с	медіана, с	СКВ
DTR1	0,1339	0,1075	0,0588	2,3088	1,8535	1,3960
DTR2	0,1555	0,1270	0,0639	3,1862	2,5750	1,6913
DTR3	0,1251	0,1070	0,0585	2,4335	1,8645	1,4422
DTR4	0,1211	0,1055	0,0489	18,7880	17,7005	3,8061
DTR5	0,1750	0,1625	0,0419	6,6728	5,9910	1,9153
DTR6	0,4622	0,4345	0,0867	19,5265	18,4610	2,9926
DTR7	0,1874	0,1770	0,0516	7,3046	6,3210	2,1765
DTR8	0,1342	0,1205	0,0464	2,3615	1,8905	1,2708
DTR9	0,1666	0,1305	0,0800	2,5800	2,1125	1,3805
DTR10	0,1294	0,1095	0,0427	2,5336	2,0185	1,4431
DTR11	0,1227	0,1065	0,0410	19,6367	18,5235	3,0684
DTR12	0,2133	0,1460	0,1728	5,2214	4,4595	1,8546
DTR13	0,2521	0,2370	0,0522	6,3890	5,7840	1,8771
DTR14	0,1640	0,1525	0,0453	5,3003	4,5680	1,8861

Порівняння значень середнього часу навчання на кластерах із 2 та 3 обчислювальними вузлами наведено в табл. 9.13 та на графіках на рис. 9.7. Як метрику для порівняння продуктивності різних кластерів

вибрано відношення середнього часу навчання на кластері із 3 вузлами до середнього часу навчання на кластері із 2 вузлами.

Таблиця 9.13

Порівняння часу навчання на гомогенних кластерах із 2 та 3 обчислювальними вузлами

Id тестів	Середній час навчання, с		Відношення
	2 обчислювальні вузли	3 обчислювальні вузли	
DTR1	2,7709	2,3088	0,8332
DTR2	4,0486	3,1862	0,7870
DTR3	2,9933	2,4335	0,8130
DTR4	24,1196	18,7880	0,7790
DTR5	10,1216	6,6728	0,6593
DTR6	40,2908	19,5265	0,4846
DTR7	10,0306	7,3046	0,7282
DTR8	2,9029	2,3615	0,8135
DTR9	3,4727	2,5800	0,7429
DTR10	3,0674	2,5336	0,8260
DTR11	24,0404	19,6367	0,8168
DTR12	6,3863	5,2214	0,8176
DTR13	10,8813	6,3890	0,5872
DTR14	6,6291	5,3003	0,7996

Отже, із визначених та наведених результатів можна зробити висновок, що середня різниця приросту продуктивності досягає близько 20 %, про що свідчить закон Амдала щодо збільшення обчислювальних ресурсів, що приводить до зменшення часу виконання завдань, які виконують для гомогенних кластерів із різною конфігурацією.

9.8. Висновки

У процесі проведення дослідження було проаналізовано основні інструменти та програмне забезпечення для вирішення завдань машинного навчання, що мають можливість виконувати на розподілених системах Apache Spark.

Розглянуто платформу для розподілених обчислень Apache Spark та її основні компоненти.

Досліджено середовище, у якому працює ця платформа та типові випадки застосування на кластері Azure HDInsihgt.

Розглянуто особливості використання платформи Azure, що дозволяють розгорнути кластери на основі платформи Apache Spark.

Проаналізовано механізми розподілу ресурсів у кластерах під управлінням програмної платформи Apache Spark та досліджено наявні параметри конфігурації, що впливають на продуктивність роботи кластера.

Для оцінювання продуктивності кластерів загального призначення розглянуто наявні тестові набори для проведення експериментальних досліджень, зокрема для тестування моделей машинного навчання. Було вибрано бенчмарк Spark-Perf як тестовий набір із розширеними можливостями щодо моделей та алгоритмів машинного навчання.

Розроблено методичне забезпечення для налаштування програмного забезпечення для різних конфігурацій кластерів.

Проведені експерименти для декількох моделей машинного навчання та здійснений їхній аналіз довели та обґрунтували підвищення продуктивності обчислень зі збільшенням кількості обчислювальних вузлів на прикладі гомогенного кластера.

Використана література

1. Аксак Н. Г. Концепція побудови мультиагентних систем розподіленої нейромережевої обробки великих даних / Н. Г. Аксак // ВІСНИК ХНТУ. – 2018. – Т. 1, № 3 (66). – С. 205–212.

2. Аксак Н. Г. Разработка системы персонализации специализированного веб-портала / Н. Г. Аксак // Радіоелектроніка, інформатика, управління. – 2018. – № 1. – С. 91–99.

3. Аксак Н. Г. Системи штучного інтелекту : навч. посіб. / Н. Г. Аксак. – Харків : ХНУРЕ, 2015. – 148 с.

4. Артеменко О. І. Інформаційні технології в галузі туризму. Аналіз застосувань та результатів досліджень / О. І. Артеменко, В. В. Пасічник, В. В. Єгорова // Вісник Національного університету "Львівська політехніка". – 2015. – № 814. – С. 3–22.

5. Кордяк В. Інформаційна технологія моніторингу та аналізу трафіку у комп'ютерних мережах / В. Кордяк, І. Дронюк, О. Федевич // Вісник Національного університету "Львівська політехніка". – 2015. – № 826. – С. 35–42. – Серія : Комп'ютерні науки та інформаційні технології.

6. Крылов А. В. Проблема извлечения знаний с использованием рассуждений на основе прецедентов / А. В. Крылов // Известия высших учебных заведений. – 2018. – Т. 61, № 11. – С. 956–962. – Серия : Приборостроение.

7. Лигун А. А. Асимптотические методы восстановления кривых / А. А. Лигун. А. А. Шумейко. – Киев : Институт математики НАН Украины, 1997. – 358 с.

8. Лосев Ю. И. Методика анализа эффективности системы динамического управления компьютерной сетью / Ю. И. Лосев, К. М. Руккас // Вісник Харківського національного університету ім. В. Н. Каразіна. – 2010. – № 890, вип. 13. – С. 154–165. – Серія: Математичне моделювання. Інформаційні технології. Автоматизовані системи управління.

9. Лосев Ю. И. Методы и модели обмена информацией в распределенных адаптивных вычислительных сетях с временной параметризацией параллельных процессов : монография / Ю. И. Лосев, С. И. Шматков, К. М. Руккас. – Харьков : ХНУ им. В. Н. Каразина, 2011. – 204 с.

10. Лосєв Ю. І. Моделювання процесу збору інформації в розподілених ієрархічних мережах / Ю. І. Лосєв, М. Ю. Лосєв // Системи обробки інформації. – 2020. – № 1 (160). – С. 59–66.

11. Малых В. Л. Системы поддержки принятия решений в медицине // Программные системы: теория и приложения. – 2019. – Т. 10, вып. 2 (41). – С. 155–184.

12. Мелешко Є. В. Алгоритми та структури даних : навч. посіб. для студентів технічних спеціальностей денної та заочної форми навчання / Є. В. Мелешко, М. С. Якименко, Л. І. Поліщук. – Кропивницький : Видавець Лисенко В. Ф., 2019. – 156 с.

13. Модель технологии информационного обеспечения решения задач управления / С. И. Шматков, Ю. И. Лосев, К. М. Руккас и др. // Научные ведомости БГУ. – Белгород : БГУ, 2013. – № 22 вып. 28/1. – С. 34–42. – Серия: Экономика. Информатика.

14. Пасічник В. Системи баз даних та знань туристичних мобільних путівників / В. В. Пасічник, В. Савчук // Вісник Національного університету "Львівська політехніка". – 2016. – № 43. – С. 154–164. – Серія: Комп'ютерні науки та інформаційні технології.

15. A real-time interpolator for parametric curves / W. Zhong, X. Luo, W. Chang [etc.] // International Journal of Machine Tools and Manufacture. – 2018. – Vol. 125. – P. 133–145.

16. A review on artificial intelligence applications to the optimal design of dedicated and reconfigurable manufacturing systems / C. Arenzi, F. Leali, M. Cavazzuti, A. Andrisano // International Journal of Advanced Manufacturing Technology. – 2014. – Vol. 72, No. 1–4. – P. 403–418.

17. Analyzing performance of Apache Spark Mllib with multinode clusters on Azure Hdinsight: Spark-Perf case study / S. Minukhin, N. Brynza, D. Sitnikov // International Scientific Conference "Intellectual systems of Decision Making and Problem of Computational Intelligence", July 26, 2021, Munich, Germany. – Munich : Springer International Publishing, 2021. – P. 114–134.

18. Arora Rajesh K. Optimization: algorithms and applications / Rajesh K. Arora. – Hoboken : CRC Press, 2015. – 454 p.

19. Axak N. G. Visualization of internet users on basis of selforganizing Kohonen maps / N. G. Axak, Ye. V. Sokolets // Materials of the 5th International Scientific Conference "Information-Management Systems and Technologies". 20 – 22 September, 2016, Odesa National Maritime University, Odesa. – 2016. – P. 226–229.

20. Cardinality constrained portfolio optimization using a hybrid approach based on particle swarm optimization and hopfield neural network // A. N. Sadigh, H. Mokhtari, M. M. Iranpoor, S. M. T. Fatemi Ghomi // Advanced Science Letters. – 2012. – No. 17 (1). – P. 11–20.

21. Experimental research of optimizing the Apache Spark tuning: RDD vs data frames / S. Minukhin, M. Novikov, N. Brynza, D. Sitnikov // CEUR Workshop Proceedings. – International Workshop On Computer Modeling and Intelligent Systems (CMIS), April 27 – May 1, 2020, Zaporizhzhia, Ukraine. – Zaporizhzhia : National University "Zaporizhzhia Polytechnic", 2020. – P. 409–425.

22. Fuhr R. D. Monotone linear rational spline interpolation / R. D. Fuhr, M. Kallay // Computer Aided Geometric Design. – 1992. – Vol. 9, issue 4. – P. 313–319.

23. Hashemi H. Bootstrap technique for risk analysis with interval numbers in bridge construction projects / H. Hashemi, S. M. Mousavi, S. M. H. Mojtahedi // Journal of Construction Engineering and Management. – 2011. – No. 133 (8). – P. 603–608.

24. Idrees A. An efficient indoor navigation technique to find optimal route for blinds using QR codes / Affan Idrees, Zahrid Iqbal, Maria Ishfaq // Proceedings of the 2015 10th IEEE Conference on Industrial Electronics and Applications, ICIEA, 2015, 15–17 June 2015, Auckland, New Zealand. – Auckland : eXpress Conference Publishing, 2015. – P. 690–695.

25. Kaklauskas A. Biometric and Intelligent Decision Making Support / A. Kaklauskas. – Geneva : Springer International Publishing, 2015. – 220 p.

26. Knowledge Warehouse: An Architectural Integration of Knowledge Management / H. R. Nemati, D. Steiger, L. S. Iyer, R. T. Herschel // Decision Support, Artificial Intelligence and Data Warehousing. Decision Support Systems. – 2020. – No. 33 (2). – P. 143–161.

27. Minukhin S. V. Analysis of ways for exchanging data in networks with package commutation / S. V. Minukhin, D. E. Sitnikov, M. Y. Losev // Radio Electronics Computer Science Control. – 2018. – No. 4. – P. 196–204.

28. Rafael V. Exploring Intelligent Decision Support Systems. Current State and New Trends / V. Rafael. – Munich : Springer International Publishing AG, 2018. – 237 p.

29. Shklovets A. V. Visualization of High Dimensional Data Using Two Dimensional Self Organizing Piecewise Smooth Kohonen Maps / A. V. Shklovets, N. G. Axak // Optical Memory and Neural Networks (Information Optics). – 2012. – Vol. 21, No. 4. – P. 227–232.

30. Srinivasan S. Multi-agent Based Decision Support System using Data Mining and Case Based Reasoning / S. Srinivasan, J. Singh, V. Kumar // International Journal of Computer Science. – 2011. – Vol. 8, issue 4, No. 2. – P. 340–349.

31. Загальні рекомендації щодо зменшення наслідків від впливу шкідливого програмного забезпечення [Електронний ресурс]. – Режим доступу : https://cert.gov.ua/recommendation/2502.

32. Засоби моніторингу та аналізу мережі [Електронний ресурс]. – Режим доступу : https://www.arc-it.net/html/archuse/archuse.html.

33. Перелік категорій кіберінцидентів [Електронний ресурс]. – Режим доступу : https://cert.gov.ua/recommendation/16904.

34. Платформа EXFO FTB-500 [Електронний ресурс]. – Режим доступу : https://www.tehencom.com/Companies/EXFO/FTB-500/EXFO_ftb-500.htm.

35. Рекомендації щодо організації віддаленої роботи [Електронний ресурс]. – Режим доступу : https://cert@cert.gov.ua/recommendation/11388.

36. Рекомендації CERT-UA з безпеки вебресурсів [Електронний ресурс]. – Режим доступу : https://cert.gov.ua/recommendation/19.

37. Система моніторингу Zabbix [Електронний ресурс]. – Режим доступу : https://www.zabbix.com/documentation/1.8/ru/manual/about/overview_of_zabbix.

38. Структура та ідентифікація керуючої інформації в мережах на основі стеку протоколів TCP/IP [Електронний ресурс]. – Режим доступу : https://www.rfc-editor.org/info/rfc1155.

39. Топ-10 – 2013. OWASP. Десять найбільш критичних ризиків для безпеки веб-додатків [Електронний ресурс]. – Режим доступу : https://cert.gov.ua/files/pdf/OWASP_Top_10_-_2013_Final_Ukrainian.pdf.

40. Aalborg University: Indoor Navigation by MapsIndoors – MapsPeople website. – Access mode : https://www.mapspeople.com/showcases/aau. – Title from the screen. – Visited: 07.03.2021.

41. Analytics Solutions Unified Method – IBM [Electronic resource] / IBM, 2016. – Access mode : ftp://ftp.software.ibm.com/software/data/sw-library/services/ASUM.pdf.

42. Apache Spark в Azure HDInsight [Electronic resource]. – Access mode : https://docs.microsoft.com/ru-ru/azure/hdinsight/spark/apache-spark-overview.

43. Floyd R. Algorithm 97: Shortest Path [Electronic resource] / R. Floyd // Communications of the ACM. – 1962. – No. 5 (6). – P. 345. – Access mode : https://doi.org.10.1145/367766.368168.

44. MLlib is Apache Spark's scalable machine learning library [Electronic resource]. – Access mode : https://spark.apache.org/mllib.

45. Ramadiani D. Floyd-warshall algorithm to determine the shortest path based on android [Electronic resource] / D. Ramadiani, D. Azainil // 1st International Conference on Tropical Studies and Its Application (ICTROPS), Series: Earth and Environmental Science 144, 2018. – Access mode : https://www.researchgate.net/publication/325017484_Floydwarshall_algorithm_to_determine_the_shortest_path_based_on_android.

46. Spark-Perf – Performance tests for Apache Spark [Electronic resource]. – Access mode : https://github.com/databricks/spark-perf.

Зміст

НАУКОВЕ ВИДАННЯ

Аксак Наталія Георгіївна
Гризун Людмила Едуардівна
Щербаков Олександр Всеволодович та ін.

СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

Монографія

За загальною редакцією
д-ра екон. наук, професора В. С. Пономаренка

Самостійне електронне текстове мережеве видання

Відповідальний за видання *І. О. Ушакова*

Відповідальний редактор *О. С. Вяткіна*

Редактор *О. Г. Доценко*

Коректор *О. Г. Доценко*

План 2022 р. Поз. № 9-ЕНВ. Обсяг 271 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
ДК № 4853 від 20.02.2015 р.