DOI: https://doi.org/10.26565/2304-6201-2025-67-02

УДК (UDC) 004.4

Гаврилюк Єгор Андрійович
аспірант кафедри математичного моделювання та аналізу даних
Харківський національний університет імені В.Н. Каразіна,
м. Харків, 61022, Україна, м. Харків, майдан Свободи, 4
e-mail: yehor.havryliuk@karazin.ua
https://orcid.org/0000-0002-4392-2000

Коробчинський Кирил Петрович
Доцент, доцент кафедри математичного моделювання та штучного інтелекту, Національний аерокосмічний університет «Харківський авіаційний інститут», 61070, Україна, м. Харків, вул. Манька Вадима, 17
e-mail: k.korobchinskiy@khai.edu
https://orcid.org/0000-0002-3676-6070

UML-орієнтована інформаційна технологія для неперервних задач максимального покриття з об’єктами довільної форми

Актуальність. Неперервні задачі максимального покриття з об’єктами довільної форми відіграють важливу роль у геоінформаційних системах, моніторингових платформах, логістичних сервісах, системах безпеки, аналізі просторових даних та рішеннях підтримки прийняття рішень. Зростання обсягів даних, динамічність середовищ і висока складність моделей потребують створення формалізованих, модульних і масштабованих інформаційних технологій. UML, як стандарт моделювання, дозволяє формально описати архітектуру програмних рішень, забезпечуючи надійність, повторюваність та прозорість програмної реалізації.
Мета. Розробити UML-орієнтовану інформаційну технологію розв’язання неперервних задач максимального покриття, що включає архітектурну модель, структуру даних, інформаційні потоки, функціональні компоненти та UML-специфікації модулів для реалізації систем покриття.
Методи дослідження. Застосовано методи об’єктно-орієнтованого та структурного моделювання, UML-діаграмування (Use Case, Class, Activity, Sequence, Component, Composite Structure, State Machine, Deployment), методи архітектурного проєктування, принципи модульності, інверсії залежностей, компонентної декомпозиції та підходи до побудови масштабованих інформаційних систем.
Результати. Побудовано повну UML-специфікацію архітектури інформаційної технології для задач максимального покриття: визначено зовнішні сценарії взаємодії, класи, компоненти, послідовності операцій, логіку поведінки і станів системи, інфраструктурні зв’язки та структуру розгортання. Сформовано інтегровану трирівневу архітектуру (рівень представлення, прикладної логіки, даних). Описано принципи формування модулів просторової аналітики, оптимізації, обчислення критерію покриття, управління сценаріями покриття, візуалізації та інтерфейсів даних. UML-моделі забезпечують формалізовану структуру, що дозволяє розробляти масштабовані й відтворювані IT-рішення для задач покриття.
Висновки. Створена інформаційна технологія забезпечує структурну, поведінкову та архітектурну формалізацію системи максимального покриття. UML-орієнтоване моделювання дозволяє підвищити прозорість архітектури, зменшити ризики інтеграційних помилок, забезпечити масштабованість і повторне використання компонентів. Отримані UML-моделі можуть слугувати методологічною основою для побудови інтелектуальних GIS-платформ, оптимізаційних сервісів, систем моніторингу та аналітичних рішень у реальному масштабі.
Ключові слова: *UML, інформаційна технологія, максимальне покриття, архітектура програмних систем, просторові дані, моделювання, оптимізація, GIS.*

Як цитувати: Гаврилюк Є. А., Коробчинський К. П. UML-орієнтована інформаційна технологія для неперервних задач максимального покриття з об’єктами довільної форми. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2025. вип. 67. С.18-34. https://doi.org/10.26565/2304-6201-2025-67-02

How to quote: Y. Havryliuk, K. Korobchynskyi, “UML-oriented information technology for continuous maximum coverage problems with arbitrary-shaped objects”, *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 67, pp. 18-34, 2025. https://doi.org/10.26565/2304-6201-2025-67-02 [in Ukrainian]

Вступ

Задачі максимального покриття відіграють ключову роль у широкому спектрі сучасних інформаційних технологій. До типових застосувань належать оптимізація розміщення сенсорів у моніторингових системах, побудова інфраструктурних об’єктів (медичних, транспортних, сервісних), оптимізація роботи безпілотних апаратів, проектування систем відеоспостереження, планування ресурсів та аналіз просторових даних у GIS-системах. Незважаючи на велику кількість

робіт, присвячених математичним методам розв'язання подібних задач, питання проектування інформаційних технологій, архітектурних моделей і програмних систем, здатних практично масштабувати такі рішення, залишаються недостатньо висвітленими.

Створення інформаційної технології охоплює алгоритмічні, архітектурні, інженерні та інтеграційні аспекти, що вимагає застосування стандартизованих методів моделювання. Серед них особливе місце займає UML-моделювання як універсальний засіб проєктування програмних систем. На відміну від математичних формалізацій, UML дозволяє формально описати структуру, поведінку, компоненти та сценарії взаємодії у вигляді стандартизованих діаграм, що значно підвищує надійність і передбачуваність процесу розробки.

У роботі запропоновано комплексну інформаційну технологію розв'язання задач максимального покриття, яка базується на архітектурі модульного типу, формально змодельована за допомогою UML, підтримує інтеграцію просторових даних, містить спеціалізований модуль оптимізації, включає інтелектуальні механізми керування та аналізу. Запропонована технологія може бути використана як основа для створення практичних IT-рішень у сферах телекомунікацій, логістики, міського планування, військової аналітики, кібербезпеки та ін.

Огляд публікацій з тематики дослідження.

Розроблення інформаційних технологій для задач максимального покриття перебуває на перетині програмної інженерії, геоінформаційних систем та оптимізаційного моделювання. У програмній інженерії основу для побудови архітектурних моделей становлять фундаментальні роботи з проектування програмних систем. У класичних джерелах [1–3] описано принципи модульності, інверсії залежностей, абстракції та багатоетапної архітектурної декомпозиції, що формують базу для створення масштабованих IT-рішень. Методи архітектурного дизайну та проєктування високорівневих компонентів, що використовуються у розподілених системах, викладено в сучасних працях з архітектури програмного забезпечення [4–6], які визначають рекомендації щодо структуризації компонентів, сценаріїв взаємодії та управління складністю систем.

UML як міжнародний стандарт моделювання, регламентований OMG, відіграє центральну роль у формалізації програмних систем. У базових та прикладних роботах [7–10] детально описано нотації структурних (Class, Component, Deployment), поведінкових (Activity, State Machine, Sequence) та інтеграційних (Composite Structure, Use Case) діаграм, що забезпечують можливість уніфікованого опису архітектури IT-рішень. Ці джерела формують методологічну основу для моделювання інформаційних технологій, подібних до тієї, що розглядається у цій статті.

Сучасні підходи до роботи з просторовими даними та геоінформаційними системами представлені у фундаментальних джерелах [11, 12], де наведено принципи аналізу геометричних об'єктів, індексації, операцій над геометричними формами та методів обчислення метричних характеристик просторових областей. Ці роботи є базовими для підсистеми просторової аналітики, що є частиною запропонованої інформаційної технології.

Алгоритмічні та прикладні аспекти задач покриття представлені в інтенсивно розвиненій групі робіт з оптимізації, мультимодальних моделей та евристичних методів. Роботи [13–15] містять загальні підходи до оптимізації покриття та низку конкретних алгоритмічних технік, зокрема градієнтні та комбінаторні методи. У системах розподіленої обробки просторових даних широко застосовуються архітектури високопродуктивних обчислень [16, 17], що дозволяють масштабувати обчислення для великих геометричних областей та складних конфігурацій покриття.

Блок досліджень, пов'язаний з математичним моделюванням задач максимального покриття, сформував окрему наукову лінію. У роботах [18–20] представлено математичні моделі неперервного максимального покриття з довільною формою областей, методи оптимізації та дослідження властивостей отриманих оптимальних конфігурацій. Ці праці формують теоретичну основу, на якій базується функціональність оптимізаційного модуля запропонованої інформаційної технології.

Окрему підгрупу становлять дослідження, присвячені застосуванню програмних бібліотек обчислювальної геометрії для практичної реалізації задач покриття. У публікаціях [21, 22] детально описано використання бібліотеки Shapely для моделювання, об'єднання, перетину та метричних оцінок геометричних областей. Ці роботи мають прикладне значення для формування підсистеми геометричних операцій нашої інформаційної технології.

У практично орієнтованих задачах покриття широко використовуються методи прогнозування, аналітики та застосування покриття в реальних сервісах. Такі аспекти висвітлено у роботі [23], де оптимізація розміщення мобільних сервісів інтегрується з прогнозними моделями у кризових сценаріях. Це актуалізує використання покриття в задачах планування, логістики та моніторингу.

Остання група джерел пов'язана з надійністю сенсорних мереж і гібридних систем моніторингу, які також базуються на принципах покриття. Дослідження [24, 25] розглядають архітектуру та надійність конфігурацій сенсорних мереж для екологічного та кризового моніторингу. Вони демонструють, що задачі покриття є ключовими не лише в оптимізації, але і в забезпеченні стійкості систем.

Таким чином, огляд літератури показує, що хоча математичні, алгоритмічні та геометричні аспекти задач максимального покриття досліджені досить широко, архітектурні рішення, побудовані на UML-моделюванні, практично не представлені. Це визначає наукову новизну та практичну значущість запропонованої інформаційної технології.

Формальна постановка неперервної задачі максимального покриття.

У загальному вигляді неперервна задача максимального покриття полягає у знаходженні такої конфігурації геометричних об'єктів (агентів), яка забезпечує максимальне покриття заданої області при виконанні певної системи обмежень на розташування покриваючих об'єктів.

Нехай:

$\Omega \subset R^2$ – компактна область покриття (довільна вимірна множина);

n – кількість покриваючих об'єктів;

$\{S_1,...,S_n\}$ - набір компактних покриваючих об'єктів (агентів) довільної форми;

$p_i = (x_i, y_i)$ – координати трансляції об'єкта S_i;

θ_i – кут повороту об'єкта S_i;

$A(\theta_i)$ – матриця повороту;

$S_i'(p_i, \theta_i) = A(\theta_i)S_i + p_i$ – трансформований об'єкт;

k – кількість заборонених зон Z_j, $j = 1,...,k$ для координат $p_i = (x_i, y_i), i = 1,...,n$.

Потрібно знайти такі $(p_1, \theta_1,..., p_n, \theta_n)$, щоб максимізувати площу покриття області Ω:

$$F(p_1, \theta_1,..., p_n, \theta_n) = area\left(\Omega \cap \bigcup_{i=1}^{n} S_i'(p_i, \theta_i)\right) \to \max \quad (1)$$

за умови

$$p_i \notin \bigcup_{j=1}^{k} Z_j,\ i = 1,...,n.$$

Обчислення значення функції (1) є нетривіальною геометричною операцією. У роботі передбачається використання п'яти підходів, кожен із яких має власний компроміс між точністю та швидкістю:

- *Сітковий метод (Монте–Карло)* - Оцінка покриття за часткою точок, що потрапили у область покриття. Гарантує універсальність та високу швидкодію, але має стохастичну похибку $O(1/N)$.
- *Метод подвійних перетинів (обмежений Inclusion–Exclusion)* - Ураховує площі поодиноких об'єктів та попарні перетини. Для кругів та еліпсів доступні точні формули. Ігнорує перетини третього порядку і вище.
- *Точні геометричні обчислення (Shapely/sympy.geometry)* - Дає аналітично точний результат і дозволяє працювати з об'єктами довільної форми. Недолік — квадратична складність) $O(N^2)$ непридатність на ранніх етапах оптимізації.
- *Метод тріангуляції області покриття* - Розбиття області $\Omega \subset R^2$ на трикутники з точним обчисленням перетинів. Висока точність, низька швидкість.
- *Метод обмежувальних рамок (bounding boxes)* - Дуже швидкий грубий метод для попередніх оцінок на стартових ітераціях оптимізації.

З практичної точки зору технологія повинна дозволяти динамічно перемикати методи обчислення площі залежно від етапу оптимізації, складності конфігурації та потрібної точності.

Зазначимо, що оптимізаційна задача (1) є нелінійною, багатовимірною, багатоекстремальною, з геометричними обчисленнями всередині функції. Тому для її розвязування застосовуються стійкі підходи глобальної оптимізації: метод штрафних функцій для врахування обмежень; ройові алгоритми (PSO, ACO, FA тощо); генетичні алгоритми (GA); меметичні алгоритми (hybrid GA + локальний пошук); локальні методи (Nelder–Mead, Powell, Quasi-Newton) у комбінованих схемах.

У практичних застосуваннях особливо важливо забезпечити: перехід від швидких грубих методів площі до точних повільних методів на фінальних етапах; адаптацію параметрів алгоритмів залежно від поточної конфігурації; стійкість до складної геометрії (неправильні полігони, багаточисельні перетини, вузькі заборонені зони).

Попри наявність значної кількості публікацій щодо методів оптимізації, більшість із них не описують архітектуру ПЗ для реалізації таких моделей, не пропонують цілісних технологічних конвеєрів, не враховують різні методи обчислення площі та перемикання між ними, не формалізують структури даних і модулі системи, не описують UML-моделі, необхідні для практичної реалізації.

У зв’язку з цим у даній статті основна увага приділяється розробленню UML-орієнтованої інформаційної технології, яка формалізує представлення геометричних об’єктів різних типів, інтеграцію методів обчислення покритої площі, керування ройовими, генетичними та меметичними оптимізаторами, підтримку масштабованої та розширюваної архітектури, можливість застосувань у GIS, моніторингу, логістиці та системах безпеки.

Архітектура інформаційної технології розв’язання задач максимального покриття.

Інформаційна технологія розв’язання неперервних задач максимального покриття з об’єктами довільної форми має бути орієнтована на підтримку повного циклу: від формулювання математичної постановки до отримання верифікованого рішення, придатного для практичного використання у ГІС-системах, системах моніторингу та підтримки прийняття рішень.

На основі постановки задачі, наведеної у попередньому розділі, інформаційна технологія повинна забезпечувати:

- завантаження, зберігання та попередню обробку просторових даних (область покриття, заборонені зони, початкові розташування покриваючих об’єктів);
- підтримку різних типів геометричних об’єктів для області, покриваючих об’єктів та зон заборони (багатокутники, кола, еліпси й інші композитні фігури);
- конфігуровані обмеження на розташування покриваючих об’єктів (геометричні обмеження, заборонені зони);
- наявність модуля обчислення критерію покриття, який підтримує щонайменше методи оцінювання площі покритої частини області (Монте-Карло-сітковий, обмежене включення–виключення, бібліотеки обчислювальної геометрії, триангуляція, апроксимація обмежувальними рамками);
- адаптивний вибір методу обчислення покриття залежно від етапу оптимізації, розмірності задачі та доступних обчислювальних ресурсів;
- універсальний оптимізаційний модуль, який підтримує як глобальні метаевристики (ройові алгоритми, генетичні, меметичні алгоритми), так і локальні методи покращення (градієнтоподібні, локальний пошук, hill-climbing);
- механізми зв’язку між вибором оптимізаційного алгоритму та методом обчислення площі, що дозволяють застосовувати швидкі наближені оцінки на ранніх етапах і точні обчислення на фінальних;
- візуалізацію проміжних і фінальних рішень, формування звітів, експорт результатів у ГІС-формати;
- модульну, розширювану та масштабовану архітектуру з чітко визначеними інтерфейсами.

Архітектура реалізується у вигляді класичної *трирівневої моделі:*

1. *Рівень представлення (Presentation Layer***)** – графічний або веб-інтерфейс аналітика, модулі візуалізації покриття, засоби налаштування задачі.

2. *Рівень прикладної логіки (Application / Logic Layer)* – підсистеми геометричного моделювання, оцінювання покриття, оптимізації, керування сценаріями, адаптивного вибору стратегій.

3. *Рівень даних (Data Layer)* – сховище геоданих, параметрів сценаріїв, історії конфігурацій, логів.

UML-діаграми, що описуються далі, послідовно розкривають ці рівні з різних точок зору: функціональної (Use Case), структурної (Class, Component, Composite Structure), поведінкової (Activity, Sequence, State Machine) та фізичної (Deployment).

Діаграма варіантів використання (Use Case Diagram)

UML-діаграма варіантів використання (Рис.1) формалізує зовнішню функціональну специфікацію інформаційної технології з точки зору користувачів та зовнішніх систем. Вона визначає, які задачі може виконувати аналітик, які сервіси надає система та як відбувається інтеграція з ГІС і обчислювальною інфраструктурою.

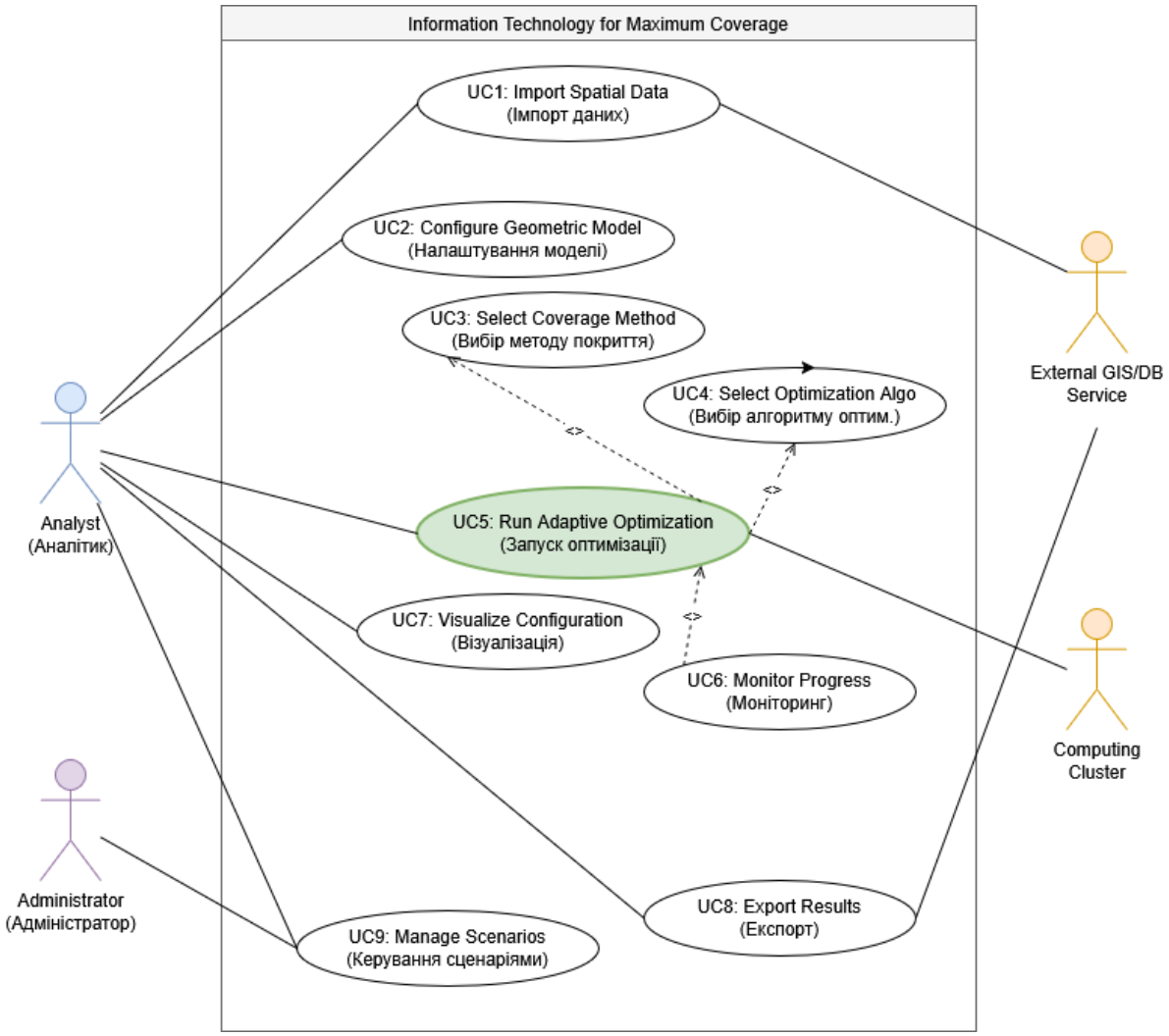

Рис. 1 – Діаграма варіантів використання UML-орієнтованої інформаційної технології для неперервних задач максимального покриття з об'єктами довільної форми
Use Case Diagram of the UML-oriented information technology for continuous maximum coverage problems with arbitrary-shaped objects.

Основні актори:

- Аналітик – формулює задачу, обирає типи геометричних об'єктів, методи обчислення покриття та оптимізації, запускає обчислення, аналізує результати.
- Адміністратор системи – керує конфігураціями, правами доступу, обчислювальними ресурсами.
- Зовнішній GIS/DB-сервіс – постачає та приймає просторові дані (геометрія області, зон заборони, довідкова інфраструктура).
- Обчислювальний кластер покриття – виконує ресурсомісткі операції з оцінювання покриття та оптимізації (може бути абстрагований як зовнішній сервіс).

Ключові варіанти використання:

- UC1 – Імпорт просторових даних (область, заборонені зони, інфраструктура).

- UC2 – Налаштування геометричної моделі (вибір типів об'єктів: багатокутник, коло, еліпс; параметрів форми).
- UC3 – Вибір методу обчислення покриття (доступних стратегій та їх комбінації).
- UC4 – Вибір алгоритмів оптимізації (ройові, генетичні, меметичні, локальний пошук).
- UC5 – Запуск адаптивної оптимізації (з автоматичним перемиканням між грубими та точними методами оцінювання).
- UC6 – Моніторинг прогресу та якості покриття (графіки збіжності, показники якості).
- UC7 – Візуалізація конфігурації покриття (карта покриття, зони прогалин, порушення обмежень).
- UC8 – Експорт рішень та звітів (карти, таблиці, журнали роботи).
- UC9 – Керування шаблонами сценаріїв (збереження й повторне використання постановок задачі).

Між UC3, UC4 та UC5 задаються залежності типу *include/extend*: вибір методів покриття та оптимізації є обов'язковими етапами перед запуском адаптивної оптимізації; деталізовані режими моніторингу та експорту реалізуються як розширення.

На діаграмі наведено основних акторів (аналітик, адміністратор, зовнішні сервіси) та ключові варіанти використання, включаючи вибір геометричної моделі, методів обчислення покриття та алгоритмів оптимізації.

Діаграма класів (Class Diagram)

UML-діаграма класів (Рис.2) задає статичну структуру інформаційної технології: сутності предметної області, їх атрибути, методи та зв'язки. Для нашої задачі виділено чотири взаємопов'язані підсистеми: геометричну, підсистему оцінювання покриття, оптимізаційну та інфраструктурну.

Геометрична підсистема відповідає за представлення області покриття, покриваючих об'єктів та зон заборони з використанням різних типів геометрії. Основні класи:

- AbstractGeometry – абстрактний базовий клас для геометричних об'єктів.
  Атрибути: id, geometryType.
  Методи: area(), contains(point), intersect(another), transform(transformParams).
- Region (наслідує AbstractGeometry) – область покриття $\Omega \subset R^2$.
  Атрибути: boundary : GeometryShape, holes : List<GeometryShape>.
  Методи: clip(geom), toPolygonalApprox().
- ForbiddenRegion (наслідує AbstractGeometry) – заборонені зони.
  Атрибути: severityLevel, penaltyWeight.
  Методи: violatedBy(configuration).
- CoverageObject (наслідує AbstractGeometry) – покриваючий об'єкт.
  Атрибути: shapeType (polygon/circle/ellipse), baseShape : GeometryShape, orientation, params.
  Методи: placeAt(position, orientation), getFootprint().
- PolygonShape, CircleShape, EllipseShape – конкретні класи форм, що інкапсулюють параметри (вершини, радіус, півосі тощо) та реалізують спеціалізовані методи обчислення площі, перетинів тощо (часто через геометричну бібліотеку).

Підсистема оцінювання покриття реалізує методи обчислення площі покритої частини області, організованих за шаблоном Strategy.

- CoverageEstimator – інтерфейс (або абстрактний клас).
  Методи:
  estimateCoverage(region : Region, objects : List<CoverageObject>) : CoverageResult.
- MonteCarloEstimator – сітковий/монте-карло метод.
  Атрибути: numSamples, samplingScheme.
  Особливості: застосовується на ранніх етапах оптимізації як швидкий наближений метод.
- PairwiseInclusionExclusionEstimator – метод з урахуванням подвійних перетинів.
  Атрибути: maxPairs, useCircleIntersectionFormula.
  Підходить для задач з переважно круговими/еліптичними об'єктами.
- GeometryLibraryEstimator – точне обчислення через бібліотеку обчислювальної геометрії (наприклад, Shapely).
  Атрибути: tolerance, backendType.
  Використовується на фінальних етапах, коли критична висока точність.

- TriangulationEstimator – метод на основі тріангуляції області.
  Атрибути: numTriangles, refinementLevel.
  Застосовується для складних форм області $\Omega \subset R^2$, коли потрібна детальна локальна оцінка.
- BoundingBoxEstimator – швидка груба оцінка через обмежувальні рамки.
  Використовується як допоміжний метод для первинного відбору конфігурацій.
- CoverageResult – результат оцінювання.
  Атрибути: coveredArea, coverageRatio, uncoveredRatio, penaltyValue.
  Методи: isFeasible(), toReportMetrics().

Оптимізаційна підсистема керує процесом пошуку конфігурації, що максимізує покриття, з урахуванням обмежень. Основні класи:

- Configuration – конфігурація покриваючих об’єктів.
  Атрибути: objects : List<CoverageObject>, decisionVector, coverageResult, metaInfo.
  Методи: evaluate(estimator), isFeasible().
- ProblemDefinition – постановка задачі.
  Атрибути: region : Region, forbiddenRegions : List<ForbiddenRegion>,
  numObjects, allowedShapeTypes, constraints.
  Методи: generateInitialConfiguration(), repairConfiguration().
- OptimizationAlgorithm – абстрактний базовий клас.
  Методи: initialize(problem), iterate(), getBestSolution().
- PSOAlgorithm, GAAlgorithm, MemeticAlgorithm, LocalSearchAlgorithm, HybridAlgorithm – конкретні реалізації.
  Кожен клас використовує різні схеми генерації, відбору та локального покращення конфігурацій.
- CoverageStrategyManager – керування вибором методу оцінювання покриття.
  Атрибути: currentEstimator, coarseEstimatorList, preciseEstimatorList, switchCriteria.
  Методи: selectEstimator(iteration, stagnationLevel, problemSize).
- OptimizationEngine – фасад для запуску оптимізації.
  Атрибути: algorithm : OptimizationAlgorithm,
  strategyManager : CoverageStrategyManager,
  penaltyManager : PenaltyManager,
  history : OptimizationHistory.
  Методи: run(problemDefinition), step(), refineBestSolution().
- PenaltyManager – реалізує штрафні функції для обмежень (заборонені зони, ліміти кількості об’єктів, мінімальні відстані).
  Методи: computePenalty(configuration), updatePenaltyCoefficients().

У Табл.1 наведено приклад основних класів та методів.

Таблиця 1 – Приклад основних класів та методів / Example of main classes and methods

Клас	*Основні атрибути*	*Основні методи*
Region	id, boundary, holes	area(), clip(), toPolygonalApprox()
CoverageObject	id, shapeType, baseShape, orientation, params	placeAt(), getFootprint()
CoverageEstimator	estimatorType	estimateCoverage()
MonteCarloEstimator	numSamples, samplingScheme	estimateCoverage()
GeometryLibraryEstimator	backendType, tolerance	estimateCoverage()
Configuration	objects, decisionVector, coverageResult	evaluate(), isFeasible()
OptimizationEngine	algorithm, strategyManager, penaltyManager	run(), step(), refineBestSolution()
CoverageStrategyManager	currentEstimator, switchCriteria	selectEstimator()

Інфраструктурна підсистема

- DataImporter / GISAdapter – імпорт даних області, зон заборони та довідкових шарів.

- ScenarioManager – керування сценаріями експериментів.
- ResultExporter – експорт карт, звітів, конфігурацій.
- VisualizationService – візуалізація конфігурацій та динаміки збіжності.
- SystemLogger / MonitoringService – логування та моніторинг.

Зв'язки між класами включають:

- узагальнення (generalization) між AbstractGeometry та його нащадками;
- композицію (composition) між Configuration та CoverageObject;
- агрегацію (aggregation) між Region і ForbiddenRegion;
- асоціацію між OptimizationEngine, CoverageStrategyManager та реалізаціями CoverageEstimator;
- використання шаблонів *Strategy, Factory, Facade.*

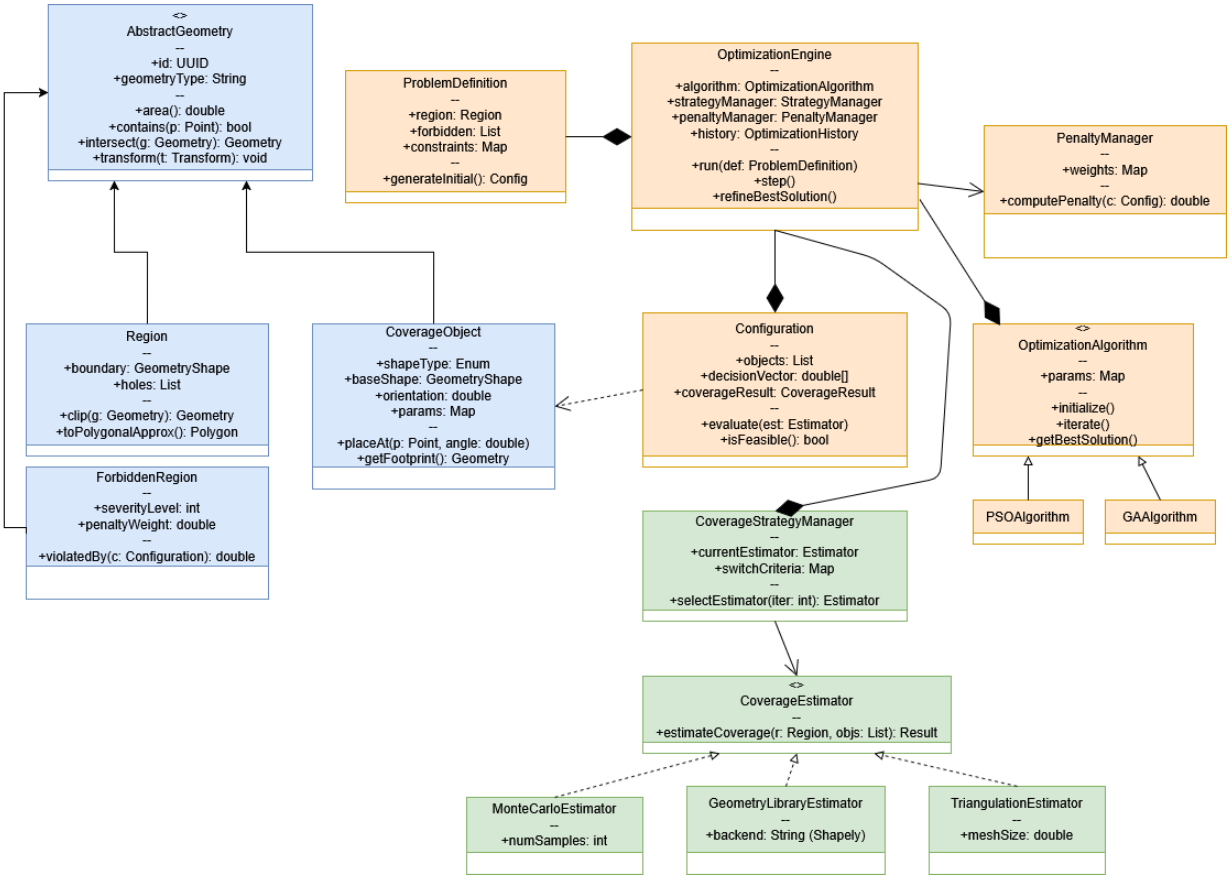

Рис. 2 – Діаграма класів UML-орієнтованої інформаційної технології для неперервних задач максимального покриття з об'єктами довільної форми.

Class Diagram of the UML-oriented information technology for continuous maximum coverage problems with arbitrary-shaped objects.

На діаграмі відображено основні класи геометричної підсистеми, підсистеми оцінювання покриття, оптимізаційного ядра та інфраструктури, а також їхні зв'язки та узагальнення.

Діаграма компонентів (Component Diagram)

Діаграма компонентів, представлена у дослідженні (Рис. 3), відображає високорівневу модульну архітектуру інформаційної технології та визначає інтерфейси взаємодії між її складовими частинами. Дана модель узгоджує логічну структуру системи з класичною трирівневою архітектурою, забезпечуючи чітку декомпозицію на рівень представлення, рівень прикладної логіки та рівень даних.

1. Рівень представлення (Presentation Layer)

Цей рівень відповідає за взаємодію з кінцевим користувачем (аналітиком) та візуалізацію результатів. До його складу входить *Visualization & Reporting Component*, що реалізує інтерфейси IVisualization та IReporting. Цей компонент відповідає за графічне представлення конфігурацій покриття, побудову карт та формування аналітичних звітів .

2. Рівень прикладної логіки (Application Logic Layer)

Це ядро системи, де зосереджена основна обчислювальна логіка. Рівень включає *Optimization Engine Component* - *ц*ентральний керуючий модуль, що реалізує інтерфейс *IOptimizationService*. Він відповідає за запуск глобальних та локальних алгоритмів оптимізації та взаємодіє з модулем оцінювання через адаптивний менеджер стратегій .

3. Рівень даних (Data Layer)

Рівень забезпечує персистентність даних та інтеграцію із зовнішнім середовищем. Рівень включає *Data Access / GIS Integration Component:* Реалізує інтерфейси IDataImport, IDataExport та IGISAdapter. Забезпечує обмін даними із зовнішніми геоінформаційними системами (GIS), базами даних та файловими сховищами .

Міжкомпонентна взаємодія

Взаємодія між рівнями та компонентами реалізується через чітко визначені інтерфейси, що забезпечує слабку зв'язність (low coupling) системи. Наприклад, Optimization Engine використовує Coverage Evaluation для оцінки рішень, який, у свою чергу, делегує геометричні обчислення компоненту Geometry Core. Така архітектура дозволяє масштабувати систему та замінювати окремі модулі без впливу на загальну функціональність.

Ця структурна організація, представлена на діаграмі компонентів (Рис. 3), є основою для побудови масштабованих та відтворюваних IT-рішень у сфері оптимізації покриття .

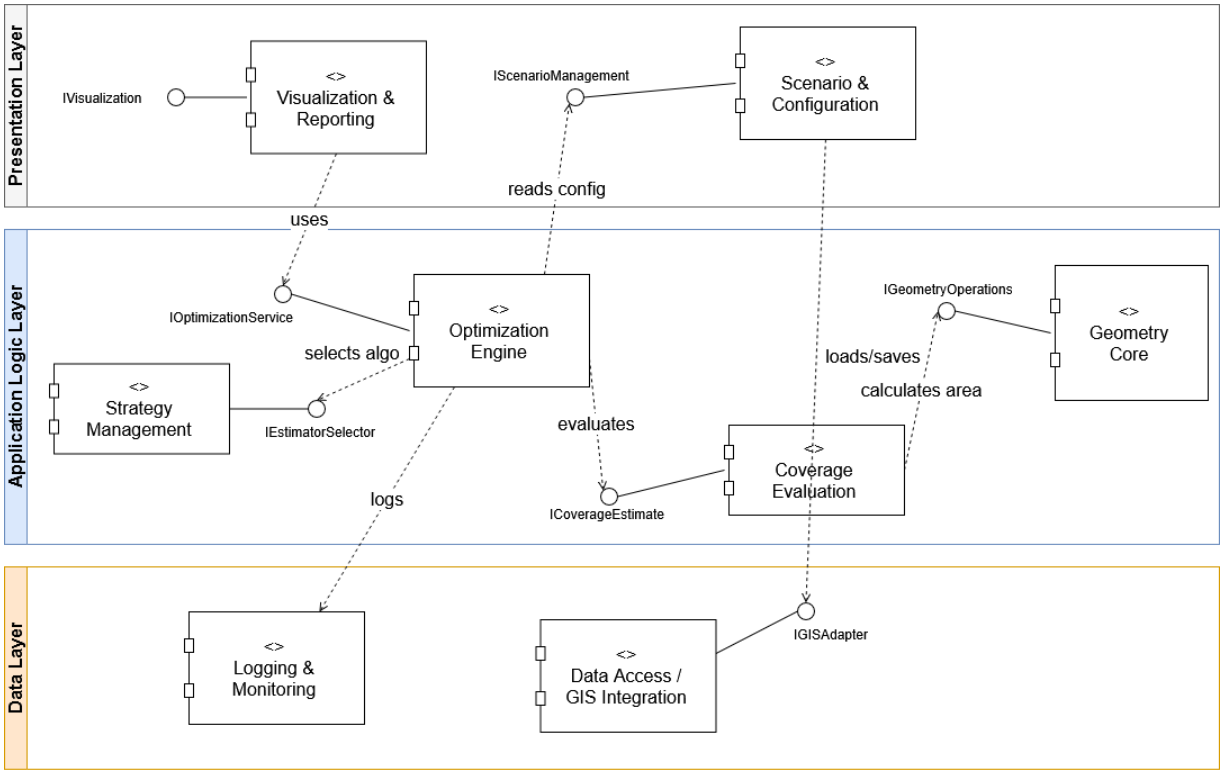

Рис. 3 – Діаграма компонентів UML-орієнтованої інформаційної технології для неперервних задач максимального покриття.

Component Diagram of the UML-oriented information technology for continuous maximum coverage problems.

Діаграма діяльності (Activity Diagram)

Діаграма діяльності (Рис.4) описує робочий процес інформаційної технології від моменту постановки задачі до отримання остаточного рішення.

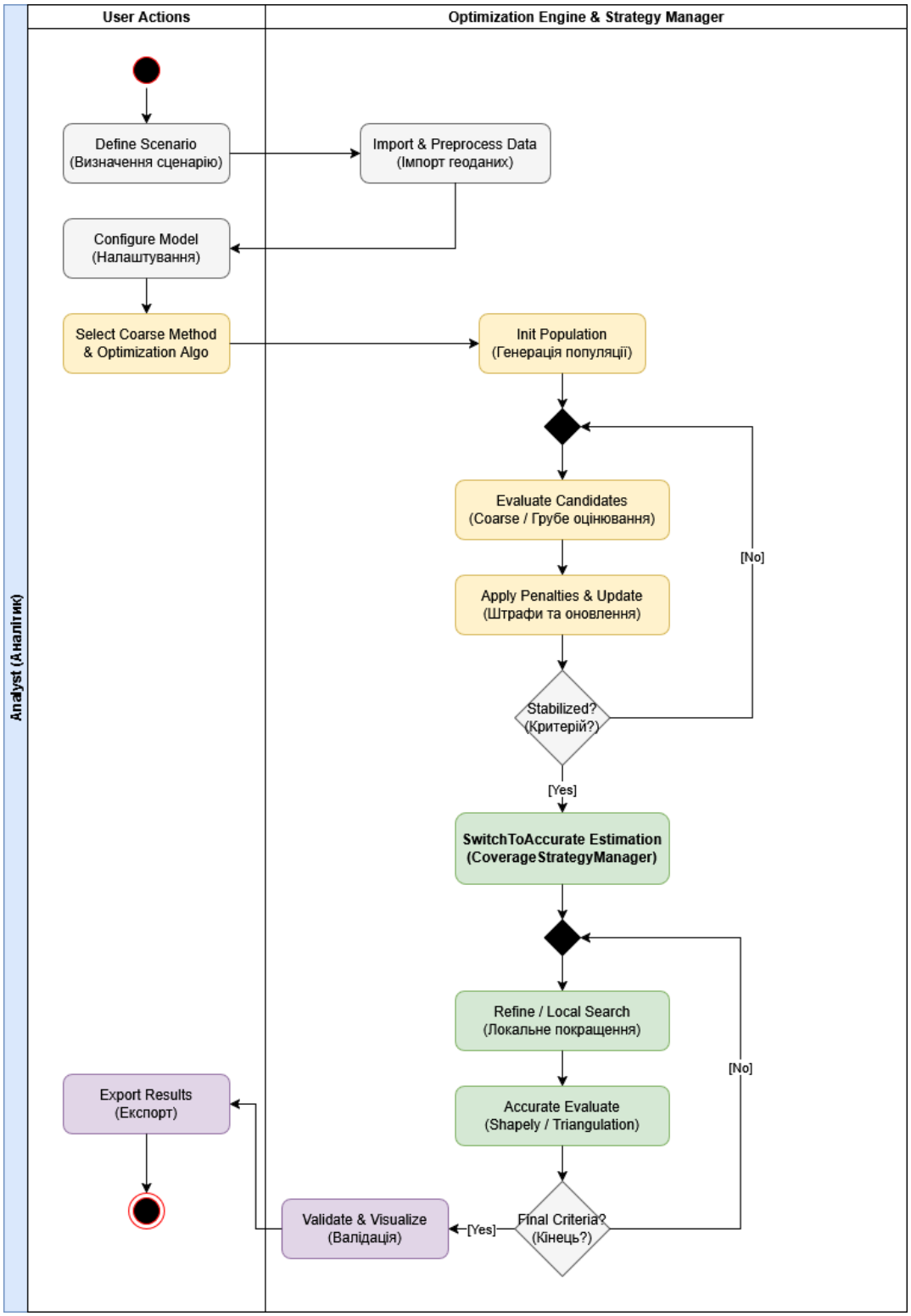

Рис. 4 – Діаграма діяльності процесу адаптивної оптимізації максимального покриття з використанням кількох методів обчислення площі та різних алгоритмів оптимізації.
Activity Diagram of the adaptive maximum coverage optimization process using multiple area-calculation methods and different optimization algorithms.

Основні етапи процесу:

- *Визначення сценарію* - Аналітик задає область покриття, типи об’єктів, обмеження, цільові функції та вимоги до точності.

- *Імпорт та попередня обробка геоданих* - Виклик дій LoadRegion, LoadForbiddenRegions, PreprocessGeometry.
- *Налаштування геометричної моделі* - Вибір типів форм (багатокутник, коло, еліпс), параметрів об’єктів та способів задання заборонених зон.
- *Вибір методу обчислення покриття* - Початково обираються швидкі наближені методи (BoundingBoxEstimator, MonteCarloEstimator) для грубої оцінки рішень.
- *Вибір алгоритмів оптимізації* - Задання глобальних методів (PSO, GA, меметичні) та, за потреби, локального пошуку.
- *Генерація початкової популяції / конфігурацій* - Діяльність InitPopulation або GenerateInitialConfigurations.
- *Основний цикл глобальної оптимізації* – EvaluateCandidates (оцінка покриття для всієї популяції за допомогою обраного грубого методу); ApplyPenalties (врахування порушень обмежень), SelectAndUpdate (відбір кращих рішень і побудова нових конфігурацій), CheckCoarseStoppingCriteria (перевірка критеріїв зупинки для грубої фази).
- *Перемикання на точні методи оцінювання* - Якщо досягнуто стабілізації покриття, активується діяльність SwitchToAccurateEstimation, де CoverageStrategyManager обирає GeometryLibraryEstimator або TriangulationEstimator.
- *Фаза уточнення рішення RefineBestConfigurations* (застосування локального пошуку і більш точних методів оцінювання), AccurateEvaluate (переоцінка кандидатів точним методом), CheckFinalCriteria (контроль фінальних критеріїв: точність, обмеження).
- *Валідація та формування остаточного рішення* - Діяльність ValidateFinalSolution використовує найточніший доступний метод, перевіряючи всі обмеження.
- *Візуалізація та експорт* - Діяльності VisualizeCoverage, GenerateReport, ExportToGIS.
- *Паралельність* - обчислення покриття для різних конфігурацій можуть виконуватися паралельно;

Діаграма відображає послідовність кроків від завдання сценарію до валідації та експорту рішення, включно з перемиканням між грубими та точними методами оцінювання.

Діаграма послідовності (Sequence Diagram)

Діаграма послідовності (Рис. 5) формалізує часову динаміку взаємодії між ключовими архітектурними компонентами інформаційної технології під час виконання адаптивної оптимізації задачі максимального покриття. Діаграма деталізує потік керування та обмін повідомленнями, необхідні для реалізації гібридної стратегії обчислення, яка поєднує грубі та точні методи оцінювання.

Учасники взаємодії (Lifelines) У процесі беруть участь такі активні об'єкти системи :

ScenarioController: Ініціює процес виконання сценарію та керує постановкою задачі.

OptimizationEngine: Виступає центральним координатором, що організовує ітераційний цикл оптимізації.

StrategyManager: Відповідає за адаптивний вибір методу обчислення покриття (Strategy Pattern) залежно від етапу оптимізації.

CoverageEstimator: Абстракція обчислювача, яка делегує виконання конкретним реалізаціям (наприклад, BoundingBoxEstimator або GeometryLibraryEstimator).

GeometryCore: Виконує низькорівневі геометричні операції (перетин, обчислення площі).

Допоміжні сервіси: PenaltyManager (розрахунок штрафів), ResultRepository (збереження рішень) та VisualizationService (відображення результатів).

Алгоритмічна логіка процесу Взаємодія компонентів реалізується у такій хронологічній послідовності:

Ініціалізація та вибір початкової стратегії: Процес розпочинається із запиту аналітика (startOptimization), після чого ScenarioController конфігурує оптимізаційне ядро (configure) . На початковій ітерації (iter=0) OptimizationEngine звертається до StrategyManager, який повертає метод грубої оцінки — BoundingBoxEstimator, що дозволяє швидко обробляти велику кількість конфігурацій .

Ітераційний цикл оцінювання (Coarse Evaluation): Після генерації початкової популяції (generateInitialConfigurations) система входить у цикл оцінювання. Для кожної конфігурації викликається метод estimateCoverage. При цьому CoverageEstimator звертається до GeometryCore для виконання операцій intersect() та area(), повертаючи об'єкт CoverageResult .

Обробка обмежень та збереження: Отримані результати передаються до PenaltyManager для обчислення штрафних функцій (computePenalty), після чого найкращі рішення зберігаються у репозиторії (saveBest).

Адаптивне уточнення (Refinement Phase): Ключовою особливістю алгоритму є динамічна зміна стратегії. При досягненні критеріїв стагнації або певної кількості ітерацій, OptimizationEngine повторно запитує стратегію у StrategyManager. На цьому етапі активується точний метод оцінювання — GeometryLibraryEstimator (на базі бібліотеки Shapely). Це ініціює цикл локального покращення та точного перерахунку метрик (Refinement Loop).

Завершення та візуалізація: Після отримання фінального розв'язку OptimizationEngine повертає результат контролеру, який ініціює його візуалізацію через виклик showFinalCoverage у компоненті VisualizationService.

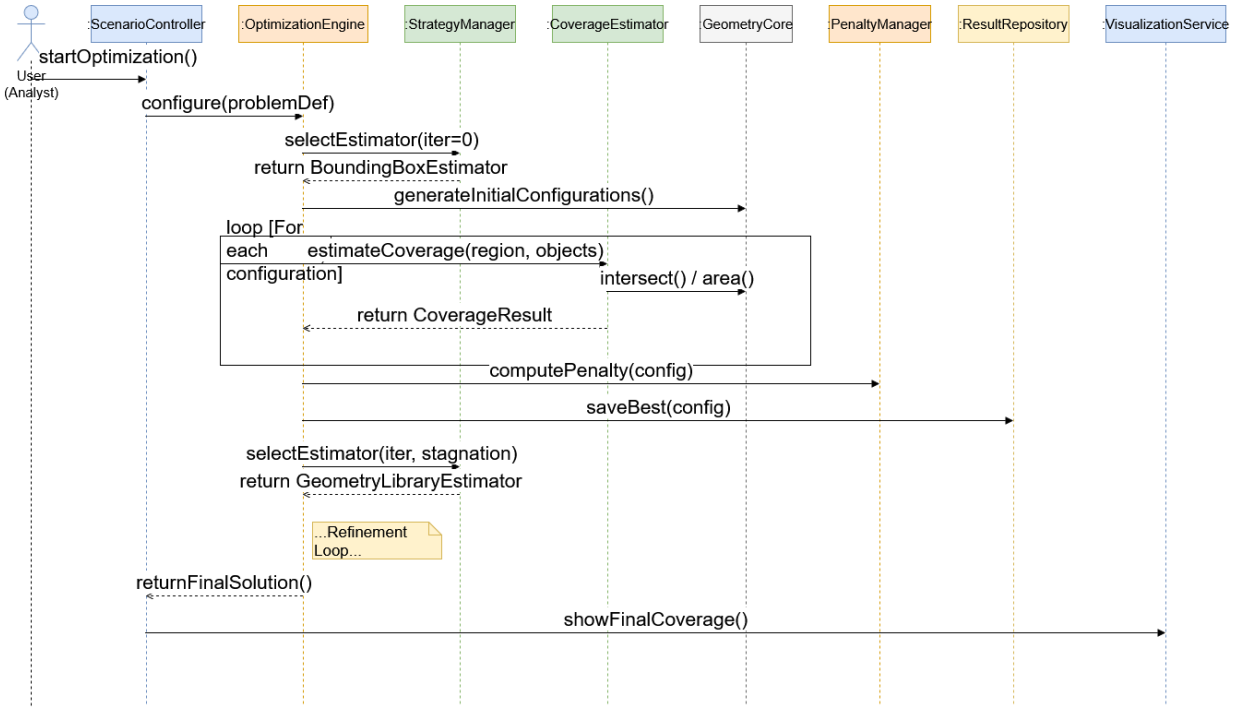

Рис. 5 – Діаграма послідовності взаємодії компонентів під час адаптивної оптимізації неперервної задачі максимального покриття.

Sequence Diagram of the interaction between components during the adaptive optimization of the continuous maximum coverage problem.

Діаграма демонструє динамічний обмін повідомленнями між користувачем, керуючим сценарієм, оптимізаційним ядром, менеджером стратегій покриття, геометричним ядром та сервісом візуалізації.

Діаграма станів (State Machine Diagram)

На Рис.6 представлена загальна характеристика діаграми станів яка формалізує життєвий цикл окремої конфігурації покриваючих об'єктів у процесі роботи адаптивного оптимізаційного алгоритму. Вона визначає логіку переходів між етапами генерації, оцінювання, фільтрації та покращення розв'язків.

Можна виділити наступні етапи життєвого циклу:

Ініціалізація та грубе оцінювання. Життєвий цикл розпочинається зі стану Generated (Згенеровано), в який об'єкт переходить після виклику події init() (випадкова генерація або створення на основі евристик). Далі ініціюється процес швидкого попереднього аналізу (подія coarseEvaluate()), що переводить систему у стан CoarseEvaluating (Грубе оцінювання). Завершення цього процесу фіксується у стані CoarseEvaluated, де конфігурація отримує наближену оцінку якості та штрафів.

Селекція та фільтрація. Ключовим етапом є перевірка перспективності рішення (guard condition isPromising?).

Негативний сценарій: Якщо конфігурація має низькі показники якості або критичні порушення обмежень, вона переходить у стан RejectedByCoarse (Відхилено). Такі об'єкти згодом переміщуються до архіву (Archived) для збереження історії пошуку.

Позитивний сценарій: Перспективні конфігурації переходять у стан SelectedForRefinement (Відібрано), що є вхідною точкою для ресурсомістких обчислень.

Точне оцінювання та локальне покращення. Для відібраних кандидатів запускається процедура точного розрахунку критеріїв (наприклад, на базі геометричних бібліотек або триангуляції), що відповідає стану AccurateEvaluating (Точне оцінювання). Після отримання точних метрик (AccurateEvaluated) до конфігурації застосовуються методи локального пошуку (applyLocalSearch()), переводячи її у стан LocalImprovement (Локальне покращення).

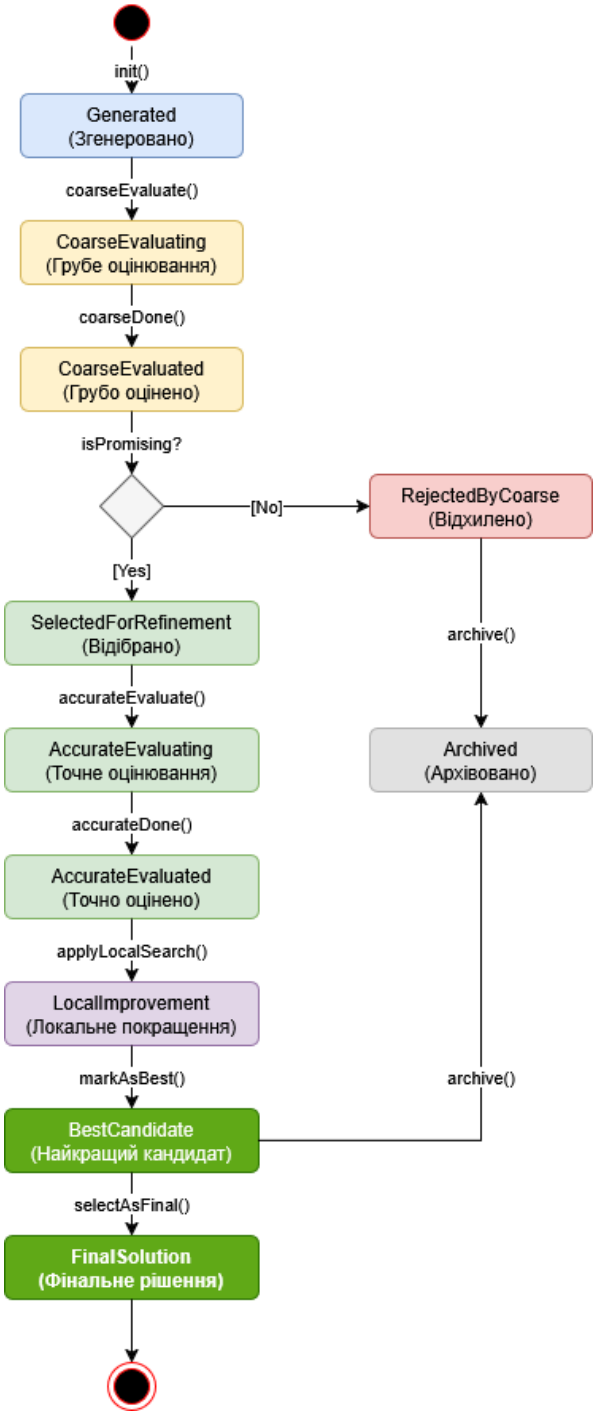

Рис. 6 – Діаграма станів конфігурації покриваючих об'єктів у процесі адаптивної оптимізації задачі максимального покриття.

State Machine Diagram of the configuration of coverage objects during the adaptive optimization of the maximum coverage problem.

Фіналізація рішення. Оптимізовані конфігурації, що продемонстрували найвищі показники ефективності, набувають статусу BestCandidate (Найкращий кандидат). З цієї множини, згідно з критеріями зупинки алгоритму, обирається єдине оптимальне рішення, яке переходить у фінальний стан FinalSolution (Фінальне рішення). Інші кандидати з множини найкращих також підлягають архівації.

Таким чином запропонована модель станів забезпечує ефективне керування обчислювальними ресурсами, дозволяючи відсіювати неперспективні рішення на етапі грубої оцінки та концентрувати обчислювальну потужність на уточненні найбільш якісних конфігурацій. Діаграма відображає переходи між станами генерації, грубого та точного оцінювання, локального покращення, відбору найкращих рішень та формування фінального результату.

Діаграма композитної структури (Composite Structure Diagram)

На Рис. 7 представлена діаграма композитної структури яка деталізує внутрішню архітектурну організацію компонента OptimizationEngine. Вона розкриває інкапсульовану логіку роботи оптимізаційного ядра, демонструючи взаємозв'язки між підмодулями, що відповідають за генерацію рішень, керування популяцією, виконання глобального та локального пошуку, а також адаптивне перемикання стратегій оцінювання .

Структурна декомпозиція компонента Внутрішня архітектура OptimizationEngine складається з наступних функціональних частин (parts):

Модулі пошуку та генерації:

GlobalSearchModule: Реалізує метаевристичні алгоритми глобальної оптимізації (PSO, GA, меметичні алгоритми) для дослідження простору рішень.

CandidateGenerator: Відповідає за процедурну генерацію початкових та проміжних геометричних конфігурацій.

LocalSearchModule: Забезпечує локальне покращення (fine-tuning) відібраних перспективних рішень.

Модулі керування даними:

PopulationManager: Виконує функцію сховища для поточної популяції кандидатів та реалізує логіку їх відбору (selection).

HistoryManager: Накопичує історію обчислень (значення функції пристосованості, метрики покриття) та використовується для детекції стагнації оптимізаційного процесу.

Модулі координації та стратегії:

EvaluationCoordinator: Виступає центральним вузлом для запитів на оцінювання конфігурацій, взаємодіючи із зовнішніми сервісами через порти.

StrategySelector: Реалізує алгоритмічну логіку динамічного перемикання між "грубими" та "точними" методами оцінювання на основі даних про хід оптимізації.

Логіка взаємодії та інформаційні потоки. Зв'язки між внутрішніми компонентами визначають ключові процеси системи:

Генерація та оновлення: GlobalSearchModule взаємодіє з CandidateGenerator та PopulationManager для створення нових особин та оновлення популяції.

Уточнення (Refinement): PopulationManager передає перспективні рішення до LocalSearchModule через конектор refinement для їх локальної оптимізації.

Оцінювання та логування: EvaluationCoordinator приймає запити на оцінку (requests eval), спрямовує їх через CoverageStrategyPort до відповідного оцінювача, а результати передає (logs result) до HistoryManager.

Адаптивне керування: HistoryManager аналізує динаміку збіжності та передає сигнал про стагнацію (stagnation info) до StrategySelector, який, у свою чергу, коригує параметри оцінювання через керуючий вплив (controls) на координатора.

Інтерфейси Взаємодія із зовнішнім середовищем здійснюється через спеціалізовані порти:

CoverageStrategyPort: Забезпечує доступ до поточного реалізатора стратегії оцінювання покриття (Strategy Pattern).

LogPort: Використовується для експорту даних логування.

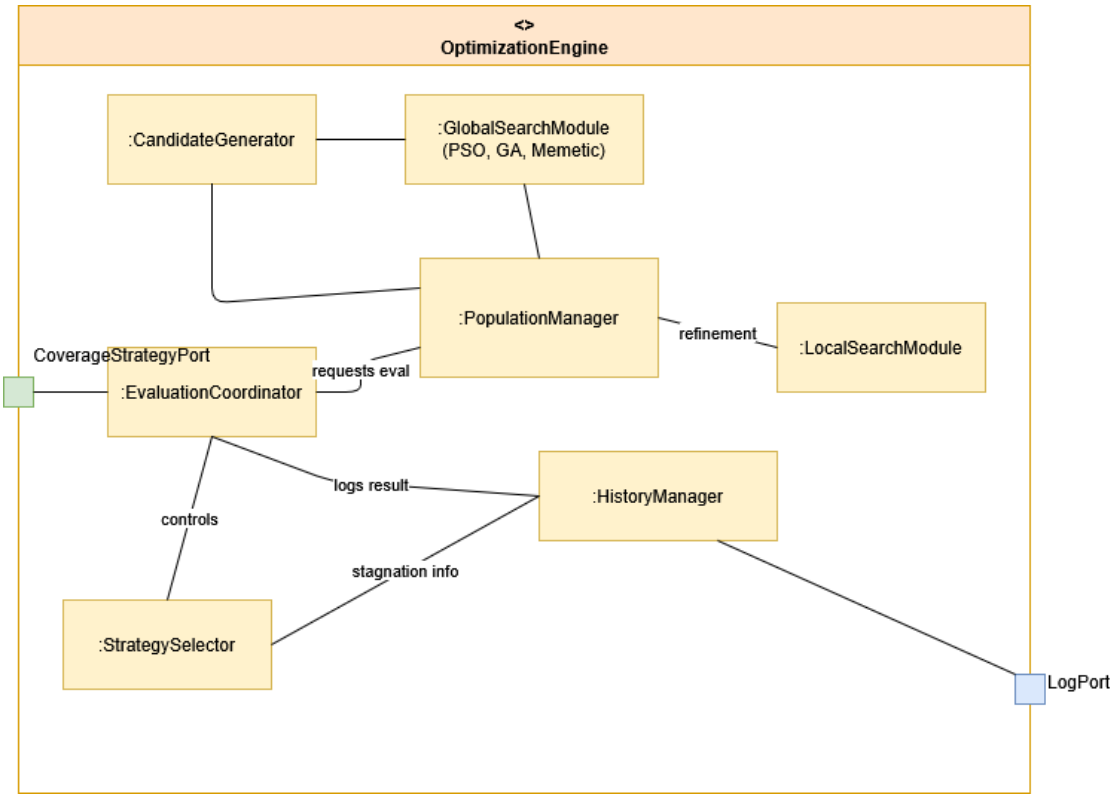

Рис. 7 – Діаграма композитної структури компонента OptimizationEngine в UML-орієнтованій інформаційній технології максимального покриття.
Composite Structure Diagram of the OptimizationEngine *component in the UML-oriented information technology for maximum coverage.*

На діаграмі показано внутрішні частини рушія оптимізації, порти та з'єднання між модулями глобального й локального пошуку, координації оцінювання, вибору стратегій та керування історією.

Висновки

У статті представлено цілісну UML-орієнтовану інформаційну технологію розв'язання неперервних задач максимального покриття з об'єктами довільної форми. Технологія формалізує архітектуру, структури даних, інформаційні потоки та алгоритмічні компоненти системи, забезпечуючи відтворюваність, масштабованість і прозорість процесу проєктування програмних рішень для задач покриття. Розроблений комплекс UML-діаграм охоплює структурні, поведінкові та інтеграційні аспекти системи, що дає змогу однозначно описати її логіку та підтримувати розширюваність.

Наукова новизна виконаного дослідження полягає у створенні уніфікованої UML-орієнтованої інформаційної технології, призначеної для розв'язання неперервних задач максимального покриття з геометричними об'єктами довільної форми.

У роботі запропоновано повну архітектурну формалізацію задач максимального покриття засобами UML. Побудовано комплекс діаграм (Use Case, Class, Component, Activity, Sequence, State Machine, Composite Structure), який формує стандартизований архітектурний каркас системи. На відміну від існуючих досліджень, що концентруються лише на алгоритмах, технологія враховує повний життєвий цикл програмної системи. Введено універсальний метамодуль оцінювання площі покриття, що об'єднує п'ять незалежних стратегій обчислення та підтримує їх адаптивне перемикання. Така інтеграція вперше формалізована UML-моделями та дозволяє ефективно працювати зі складними геометріями та великими конфігураціями. Розроблено новий підхід до адаптивної оптимізації, який поєднує грубі й точні методи оцінювання, глобальні й локальні алгоритми, а також механізм динамічного вибору стратегії. Це створює гібридну оптимізаційну платформу, яка є науково унікальною. Запропоновано оригінальну модель життєвого циклу конфігурації у вигляді діаграми станів, яка формалізує переходи між станами оцінювання, покращення та архівації кандидатних рішень. Сформовано композитну структуру

оптимізаційного ядра, що визначає внутрішню організацію глобального пошуку, локального покращення, моніторингу та стратегічного керування. Це забезпечує масштабованість і відтворюваність архітектурних рішень.

Таким чином, робота формує клас інформаційних технологій, у яких UML виступає не просто засобом документування, а фундаментом архітектурного проєктування систем оптимізації покриття.

Подальші дослідження можуть бути спрямовані на інтеграцію методів машинного навчання для автоматичного вибору стратегій оптимізації, розроблення розподіленої та паралельної реалізації оптимізаційного ядра, підтримку динамічних задач покриття зі змінними у часі геометричними об’єктами, інтеграцію з потоковими сенсорними мережами реального часу.

REFERENCES

1. Object Management Group, “Unified Modeling Language (UML), Version 2.5.1,” formal/17-12-05, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1

2. J. Arlow and I. Neustadt, *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*, 2nd ed. Boston, MA, USA: Addison-Wesley, 2005, p. 624.

3. P. Clements, F. Bachmann, L. Bass et al., *Documenting Software Architectures: Views and Beyond*, 2nd ed. Boston, MA, USA: Addison-Wesley, 2010, p. 624.

4. L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Boston, MA, USA: Addison-Wesley, 2012, p. 624.

5. R. N. Taylor, N. Medvidović, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice*. Hoboken, NJ, USA: Wiley, 2009, p. 736.

6. I. Rauf, M. Z. Iqbal, and Z. I. Malik, “UML based modeling of web service composition—A survey,” *Int. J. Comput. Appl.*, vol. 1, no. 6, pp. 301–307, 2008. doi: 10.5120/324-524.

7. P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind, *Geographic Information Systems and Science*, 3rd ed. Chichester, U.K.: Wiley, 2011, p. 560.

8. S. Gillies, *Shapely: Computational Geometry Library*, ver. 2.0.0. Zenodo, 2021. doi: 10.5281/zenodo.7428463.

9. K. Jordahl et al., “GeoPandas: Python tools for geographic data,” *J. Open Source Softw.*, vol. 9, no. 1083, Art. no. 5660, Mar. 2023. doi: 10.21105/joss.05660.

10. J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. IEEE Int. Conf. Neural Netw. (ICNN'95)*, Perth, WA, Australia, 1995, vol. 4, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.

11. C. J. A. Bastos-Filho et al., “A novel search algorithm based on fish-school behavior,” *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 39, no. 2, pp. 237–252, Apr. 2009. doi: 10.1109/TSMCC.2009.2030235.

12. X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed. Beckington, U.K.: Luniver Press, 2010, p. 148.

13. J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006, p. 664. doi: 10.1007/978-0-387-40065-5.

14. W. E. Hart, N. Krasnogor, and J. E. Smith, Eds., *Recent Advances in Memetic Algorithms*, vol. 166. Berlin, Germany: Springer, 2005. doi: 10.1007/3-540-32363-5.

15. A. Calvagna, A. Gargantini, and E. Viganò, “An adaptive penalty based parallel tabu search for constrained covering array generation,” *Inf. Softw. Technol.*, vol. 138, Art. no. 106768, Oct. 2021. doi: 10.1016/j.infsof.2021.106768.

16. J. Kallrath, “Cutting circles and polygons from area-minimizing rectangles,” *J. Glob. Optim.*, vol. 43, no. 2–3, pp. 267–298, Jun. 2009. doi: 10.1007/s10898-007-9251-2.

17. Y. Shi, H.-Z. Huang, Y. Liu, Y.-F. Li, and X.-Y. Xiao, “A new reliability analysis method based on the efficient Latin hypercube sampling,” *Struct. Multidiscip. Optim.*, vol. 58, no. 6, pp. 2371–2386, Dec. 2018. doi: 10.1007/s00158-018-1978-3.

18. S. V. Yakovlev, “The concept of modeling packing and covering problems using modern computational geometry software,” *Cybern. Syst. Anal.*, vol. 59, no. 1, pp. 108–119, Jan. 2023. doi: 10.1007/s10559-023-00547-5.

19. S. Yakovlev, O. Kartashov, and A. Mumrienko, “Formalization and solution of the maximum area coverage problem using library Shapely for territory monitoring,” *Radioelectron. Comput. Syst.*, vol. 2, pp. 35–48, 2022. Available: http://nti.khai.edu/ojs/index.php/reks/article/view/reks.2022.2.03.

20. S. Yakovlev, O. Kartashov, and D. Podzeha, “Mathematical models and nonlinear optimization in continuous maximum coverage location problem,” *Computation*, vol. 10, no. 7, Art. no. 119, Jul. 2022. doi: 10.3390/computation10070119.

21. S. Yakovlev, O. Kiseleva, D. Chumachenko, and D. Podzeha, “Maximum service coverage in business site selection using computer geometry software,” *Electronics*, vol. 12, no. 10, Art. no. 2329, May 2023. doi: 10.3390/electronics12102329.

22. S. Yakovlev et al., “Continuous maximum coverage location problem with arbitrary shape of service areas and regional demand,” *Symmetry*, vol. 17, no. 5, Art. no. 676, 2025. doi: 10.3390/sym17050676.

23. S. Yakovlev et al., “Optimization of mobile medical service locations based on predictive analytics in crisis scenarios,” in *Proc. IADIS Inf. Syst. E-Soc.*, 2025, pp. 538–541.

24. K. Leichenko et al., “Assessment of the reliability of wireless sensor networks for forest fire monitoring systems considering fatal combinations of multiple sensor failures,” *Cybern. Syst. Anal.*, vol. 61, no. 1, pp. 137–147, Jan. 2025. doi: 10.1007/s10559-025-00722-w.

25. S. Skorobohatko et al., “Architecture and reliability models of hybrid sensor networks for environmental and emergency monitoring systems,” *Cybern. Syst. Anal.*, vol. 60, no. 2, pp. 293–304, Mar. 2024. doi: 10.1007/s10559-024-00670-x.

Havryliuk Yehor Andriyovych *PhD student of the Department of Mathematical Modeling and Data Analysis Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine, 61022 e-mail: yehor.havryliuk@karazin.ua* **https://orcid.org/0000-0002-4392-2000**

Korobchynskyi Kyryl Petrovych *Associate Professor of the Department of Mathematical Modeling and Artificial Intelligence, National Aerospace University "Kharkiv Aviation Institute", Manka Vadyma St. 17, Kharkiv, Ukraine. 61070 e-mail: k.korobchinskiy@khai.edu* **https://orcid.org/0000-0002-3676-6070**

UML-Oriented Information Technology for Continuous Maximum Coverage Problems with Arbitrary-Shaped Objects

Relevance. Continuous maximum coverage problems with arbitrary-shaped objects play a crucial role in geographic information systems, monitoring platforms, logistics services, security systems, spatial data analysis, and decision-support solutions. The growing volume of data, dynamic environments, and high model complexity require formalized, modular, and scalable information technologies. UML, as a modeling standard, enables formal architectural descriptions of software solutions, ensuring reliability, reproducibility, and transparency of implementation.

Purpose. To develop a UML-oriented information technology for solving continuous maximum coverage problems that incorporates an architectural model, data structures, information flows, functional components, and UML specifications of modules supporting coverage-based systems.

Methods. The study employs object-oriented and structural modeling techniques, UML diagramming (Use Case, Class, Activity, Sequence, Component, Composite Structure, State Machine, Deployment), architectural design methods, principles of modularity, dependency inversion, component decomposition, and approaches used in building scalable information systems.

Results. A complete UML specification of the architecture of an information technology for maximum coverage problems has been constructed: external interaction scenarios, classes, components, operation sequences, system behavior and state logic, infrastructural links, and deployment structure have been defined. An integrated three-tier architecture (presentation, application logic, and data layers) has been formed. Principles for constructing modules for spatial analytics, optimization, coverage criterion computation, scenario management, visualization, and data interfaces have been described. The UML models provide a formalized structure that enables the development of scalable and reproducible IT solutions for coverage problems.

Conclusions. The developed information technology provides structural, behavioral, and architectural formalization of a maximum coverage system. UML-oriented modeling improves architectural transparency, reduces risks of integration errors, and ensures scalability and reusability of components. The obtained UML models may serve as a methodological foundation for building intelligent GIS platforms, optimization services, monitoring systems, and real-time analytical solutions.

Keywords: *UML, information technology, maximum coverage, software architecture, spatial data, modeling, optimization, GIS.*

Надійшла у першій редакції 27.08.2025, в останній - 29.09.2025.
The first version has been received on 27.08.2025, the final version - on 29.09.2025.