НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри інформаційних систем і технологій**
_____ **Ігор БОЛБОТ** (підпис)	_____ **Михайло ШВИДЕНКО** (підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмний застосунок проєктування електронних компонентів»

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми д.е.н., професор	_____ (підпис)	**Роман РУДЕНСЬКИЙ**
Керівник бакалаврської кваліфікаційної роботи Професор , Доктор технічних наук	_____ (підпис)	**Вікторія Смолій**
Виконав	_____ (підпис)	**Василь Казанецький**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем і технологій

к.е.н., доцент _____ **Михайло ШВИДЕНКО**
(підпис)

« 06 » січень 2026 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

Казацецькому Василю Олеговичу
(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Програмний застосунок проєктування електронних компонентів» затверджена наказом ректора НУБІП України від 06.01.2026 №46"С"
2. Термін подання завершеної роботи на кафедру 2026.05.25
(рік, місяць, число)
3. Вихідні дані до роботи: спосіб відображення та створення умовних графічних позначень електронних компонентів.
4. Перелік питань, що розглядаються:
   - Аналіз існуючих стандартів та норм для відображення та створення умовних графічних позначень електронних компонентів.
   - Проєктування та розробка інформаційного забезпечення системи для проєктування електронних компонентів.
   - Проєктування та розробка програмного забезпечення системи для проєктування електронних компонентів.
   - Впровадження та експлуатація системи.

Дата видачі завдання " 06 " січень 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Вікторія Смолій**

Завдання взяв до виконання _____ (підпис) **Василь Казанецький**

ЗМІСТ

ВСТУП

Метою розробки програмного додатку є створення практичного інструменту для автоматизації базових процесів проєктування електронних компонентів та схем, спрямованого на підвищення продуктивності роботи. Основні завдання включають: спрощення процесу розробки електронних схем шляхом надання інструментів створення шаблонів стандартних компонентів та їх використання для проектування; підвищення точності проєктування за рахунок автоматичного малювання та редагування компонентів на схемах; забезпечення графічного інтерфейсу, який дозволяє користувачам створювати та редагувати електронні схеми; оптимізація часових витрат на проєктування шляхом автоматизації рутинних операцій; налагодження можливості експорту схем для подальшого використання.

Актуальність. Проєктування електронних пристроїв залишається однією з найбільш трудомістких і відповідальних операцій у сучасній технічній практиці. Традиційні методи розробки електронних схем потребують значних часових витрат: розробник повинен вручну вибирати компоненти, креслити схему. Ці процеси часто виконуються послідовно, що приводить до затягування циклу розробки та збільшення кількості помилок на ранніх етапах.

Розроблення доступного програмного застосунку, орієнтованого на спрощення та прискорення проєктування, дозволить залучити більшу кількість осіб до роботи з електронними схемами та сприяти розвитку у сфері електроніки. Практична цінність такого інструменту визначається його спроможністю скоротити часові витрати на розробку схем, зменшити кількість проектних помилок.

Методи та технології розробки програмного застосунку проєктування електронних компонентів базуються на сучасних підходах до розробки програмного забезпечення та принципах інженерного проєктування.

Основною методологією розробки є об'єктно-орієнтоване програмування, яке забезпечує структурованість кода, його переносимість та можливість подальшого розширення функціональності. За допомогою об'єктно-орієнтованого підходу електронні компоненти моделюються як окремі класи з чітко визначеними властивостями та методами, що дозволяє організувати кодову базу логічною та ієрархічною структурою.

Для реалізації графічного інтерфейсу користувача застосовуються мова java та бібліотеки для побудови інтерфейсу, такі як Awt та Swing. Ці технології дозволяють створювати багатофункціональні інтерфейси, через які користувачі можуть взаємодіяти зі схемами та компонентами. Awt забезпечує можливості для роботи з графікою, що є необхідним для візуалізації електронних схем.

Для збереження та завантаження проектів використовуються формат обміну даними XML. Ці формат забезпечує зберіганні структурованої інформації про компоненти схеми та їх взаємозв'язки.

структура записки: кількість сторінок 61, використаних джерел 13, додатків 5.

Структура роботи організована таким чином: спочатку розглядаються теоретичні основи електроніки та методи проєктування, далі описується архітектура застосунку, реалізація окремих модулів, та результати тестування, демонструється інсталяційний пакет, надаються висновки по роботі.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання

Електрична схема - це схема, на якій за допомогою умовних познак і зображень показано електричні складові частини виробу і зв’язки між ними[12].

Електронний компонент - це дискретний фізичний пристрій, що використовується в електронних схемах для управління, впливу або управління потоком електричної енергії або сигналів. Електронні компоненти класифікуються на основі того, як вони взаємодіють з електричними сигналами та потужністю всередині ланцюга. Дві первинні класифікації - це пасивні компоненти та активні компоненти[13].

Електронні компоненти мають ряд електричних виводів або проводів. Ці виводи з'єднуються з іншими електричними компонентами за допомогою дроту для створення електронної схеми з певною функцією.

Система автоматизованого проєктування (САП або САПР) або автоматизована система проєктування (АСП) - автоматизована система, призначена для автоматизації технологічного процесу проєктування виробу, результатом якого є комплект проєктно-конструкторської документації, достатньої для виготовлення та подальшої експлуатації об'єкта проєктування[11].

У реальному світі електронний компонент є матеріальним об’єктом, який має явну форму. Однак, під час проєктування електронних схем та друкованих плат за допомогою САПР із використанням комп’ютерів, електронні компоненти абстрагуються у віртуальному світі у набір моделей, які по-різному представляють компонент у різних областях проєктування: на схемі – це умовне графічне позначення (або символ), на платі – посадкове місце тощо[10, с. 14].

У роботі з САПР електронні компоненти проєктуються не самі по собі, а так звані компонентні модулі або бібліотечні компоненти. Компонентний

модуль - це набір взаємопов'язаних конструкторських, технологічних та схемотехнічних даних про електронний компонент.

На електросхемі позначаться як умовні графічні позначення, що використовуються для відображення електронних компонентів електричних схемах. Це стандартизовані графічні елементи, що описують функціонал компонента та точки підключення, необхідні автоматизованого проектування друкованих плат.

Автоматизація проектування електронних пристроїв - тип систем автоматизованого проектування (САПР) для проектування електронних пристроїв, друкованих вузлів (плат) або мікросхем.

САПР дозволяє створити принципову електричну схему проектованого пристрою за допомогою графічного інтерфейсу, створювати і модифікувати базу радіоелектронних компонентів, перевіряти цілісність сигналів на ній. Сучасні САПР дозволяють обмежено виконати автоматичне розміщення елементів і автоматично трасувати доріжки в моделі друкованої плати, з'єднуючи тим самим виводи радіоелектронних компонентів відповідно до принципової схеми. Створена через редактор електрична схема передається за допомогою проміжного файлу зв'язків і завантажується в редактор плати для подальшого проектування.

Системи автоматизації проектування електроніки можуть мати можливість моделювання розроблюваного пристрою і дослідження його роботи до того, як він буде втілений в апаратуру.

Виходячи з предметної області можна зробити наступні висновки. Проєкти та схеми є основними об'єктами, які обробляються та зберігаються в системі. Кожен проєкт має унікальний назву, опис. Схеми, що входять до проєктів, містять графічне представлення та інформацію про з'єднання між компонентами.

Для забезпечення повноцінної роботи необхідно обробляти та зберігати різноманітну інформацію, яка включає: дані про компоненти, схеми.

Для забезпечення ефективної роботи необхідно автоматизувати наступні операції: автоматичне розміщення компонентів на схемі, автоматичне підключення компонентів між собою, створення бібліотеки умовних позначень, генерація звітів.

Характер використання інформації у системі відноситься до Інформаційно-обчислювальні системи у якій здійснюють усі операції перероблення інформації за певним алгоритмом.

Програмний застосунок проєктування електронних компонентів можна віднести до класу систем Автоматизації проектування електронних пристроїв EDA або ECAD автоматизованого проектування (САПР) для проектування електронних пристроїв, друкованих вузлів (плат) або мікросхем.

За масштабом система відноситься до однокористувацьких ІС. Однокористувацькі інформаційні системи призначені для роботи на одному робочому місці. Вони характеризуються обмеженим функціоналом та орієнтацією на індивідуального користувача.

За автоматизацію система відноситься до класу автоматизованих інформаційних систем, у які передбачають активну взаємодію людини та технічних засобів. Функціональні вимоги до системи включають: система повинна надавати інструменти для візуального проектування електричних схем; система повинна надавати можливості для автоматичного розміщення компонентів та автоматичного трасування доріжок; система повинна надавати можливості для автоматичної генерації звітів про проєкт; система повинна надавати можливості для керування проєктами. Нефункціональні вимоги: система має не порушувати цілісність документа проектування при аварійному завершенні; система повинна забезпечувати 30 кадрів в секунду; система повинна бути сумісною з windows 10,11.

1.2 Огляд інформаційних джерел та існуючих рішень

Системи автоматизованого проєктування електронних компонентів характеризується складною екосистемою спеціалізованого програмного забезпечення.

Altium Designer (AD) - це програмний пакет для автоматизації проектування друкованих плат (PCB) та електронних пристроїв. Він розроблений американською компанією Altium Limited. Раніше Altium Designer випускався під брендом Protel. Зараз Altium Designer більше не продається як окремий продукт, а входить до складу платформ Altium Develop та Altium Agile.

Приклад створення PCB Schematic (рис. 1.1).

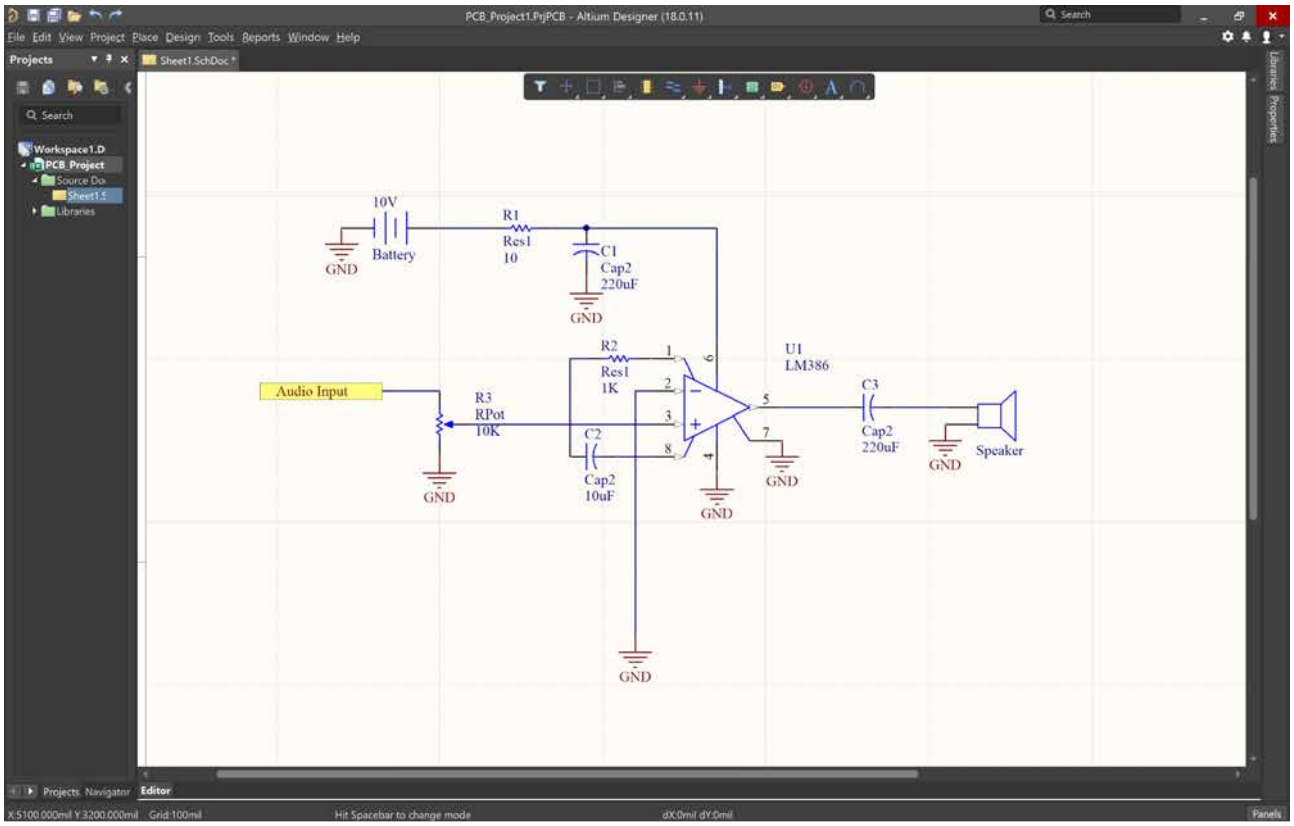

Рис. 1.1 - Робоча область створення Altium Develop

Демонстрація повного циклу розробки електронних плат в Altium Designer (рис. 1.2).

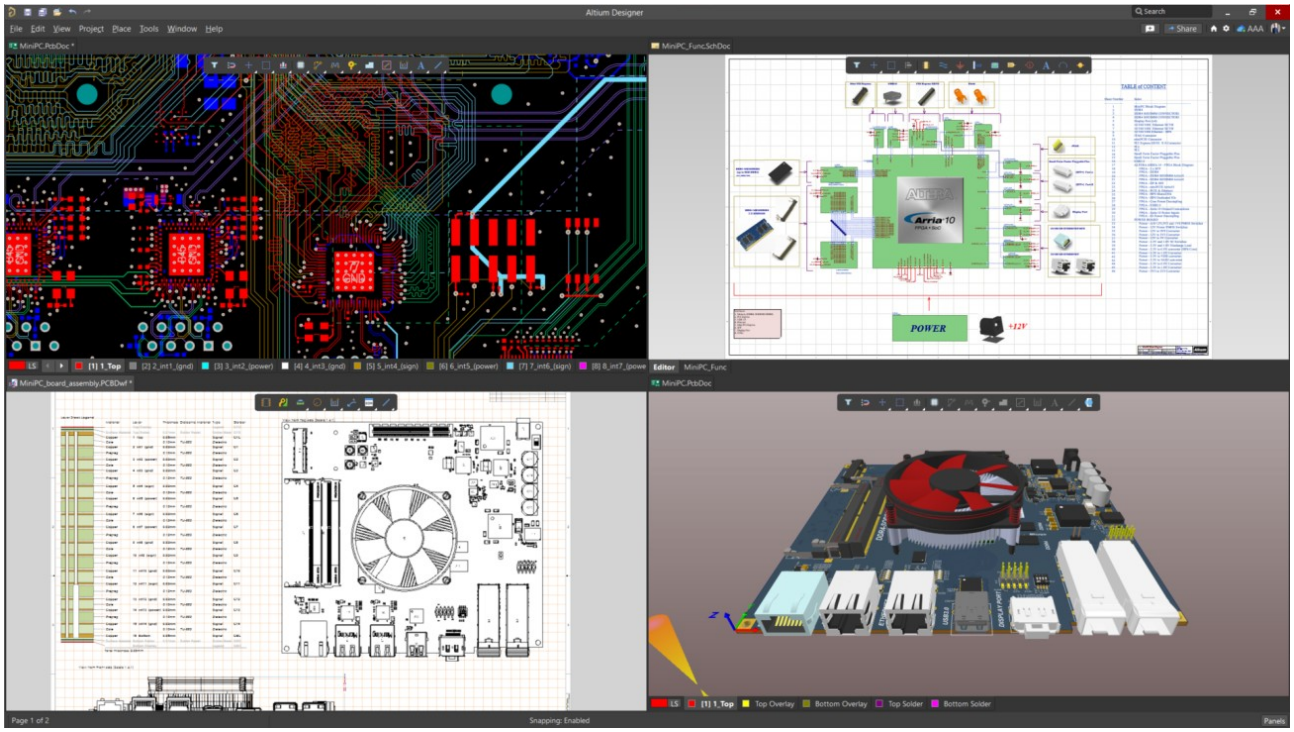

Рис. 1.2 - Цикл розробки електронних плат

Переваги:

- Повний цикл проєктування друкованих плат;
- Глибока інтеграція схемотехнічного та топологічного проєктування;
- Розширені бібліотеки електронних компонентів.

Недоліки:

- Висока вартість ліцензування;
- Складність освоєння для початківців;
- Значні системні вимоги до апаратного забезпечення.

KiCad - це безкоштовний програмний пакет для автоматизації електронного проектування (EDA). Він полегшує проектування та моделювання електронного обладнання для виробництва друкованих плат.

Він має інтегроване середовище для схематичного введення, компонування друкованих плат, перегляду виробничих файлів, моделювання SPICE за допомогою ngspice та інженерних розрахунків.

Робоча область створення KiCad (рис. 1.3).

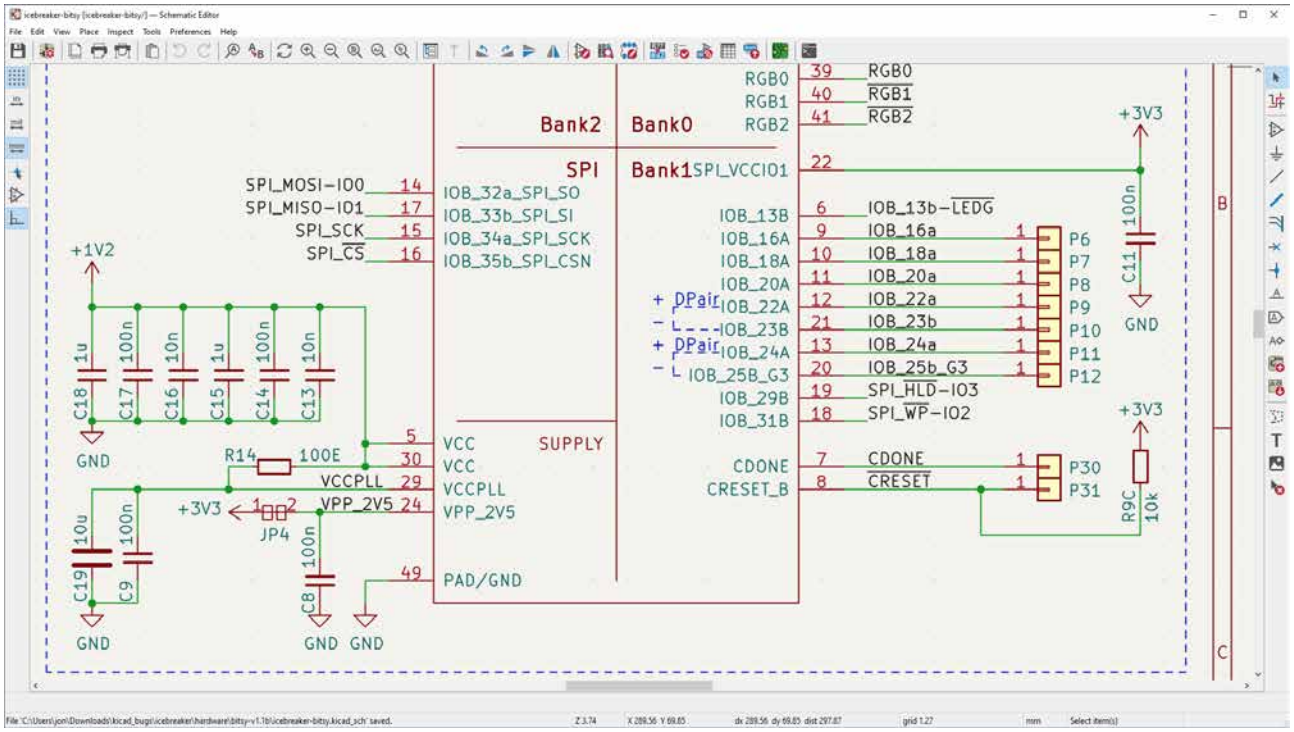

Рис. 1.3 - Робоча область

Приклад відображення конструкції друкованих плат в KiCad (рис. 1.4) та (рис. 1.5).

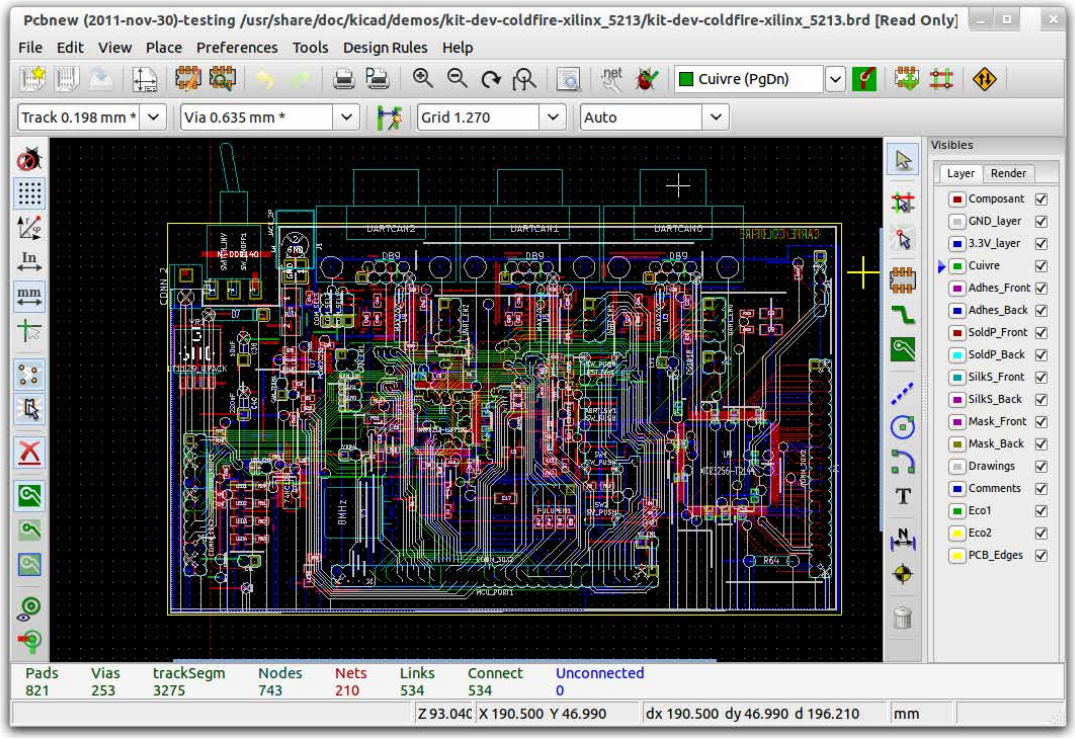

Рис. 1.4 - конструкція друкованої плати режим перегляду перший

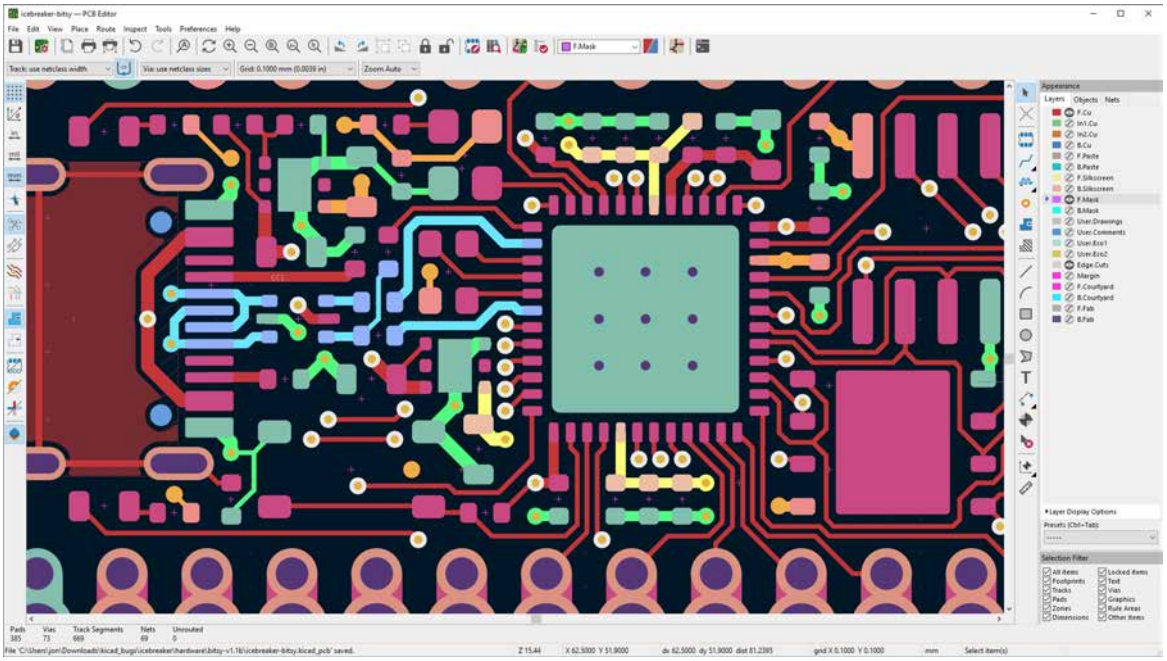

Рис. 1.5 - конструкція друкованої плати режим перегляду другий

Приклад відображення 3д моделі плати в KiCad (рис. 1.6).

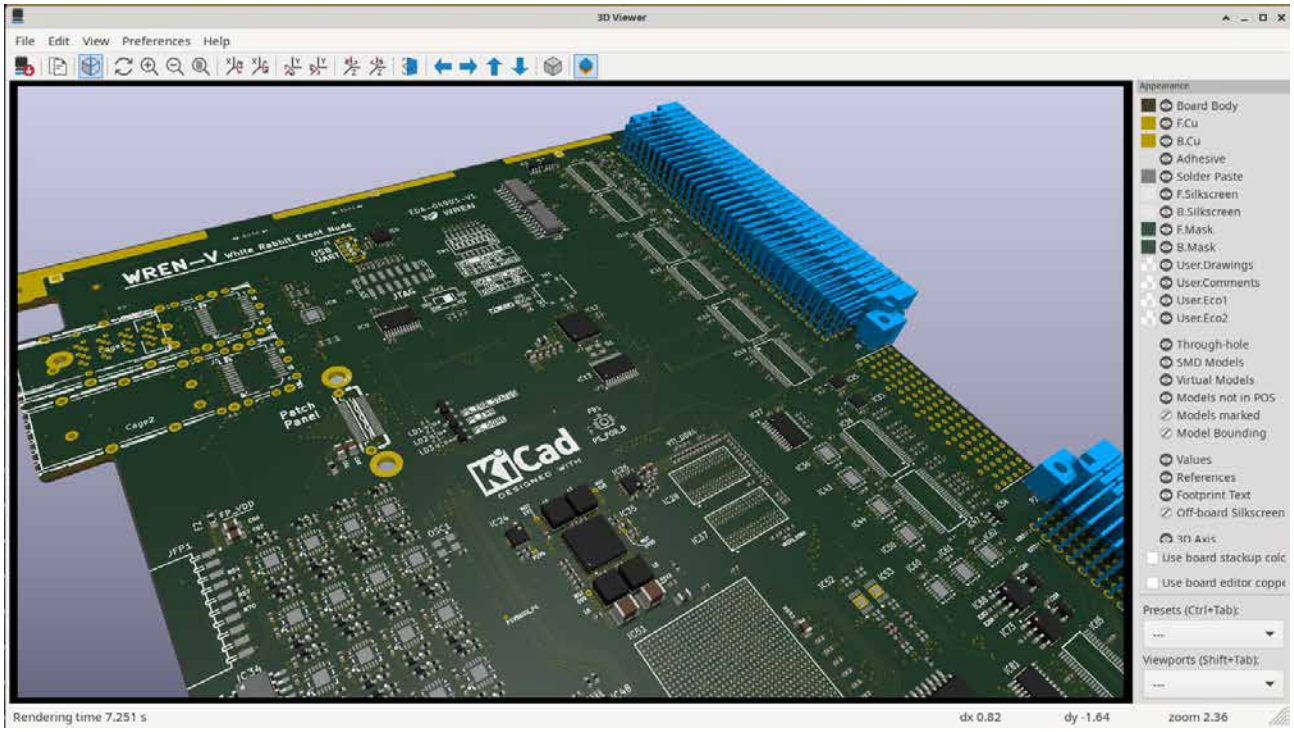

Рис. 1.6 - 3д модель

Переваги:

- Безкоштовне програмне забезпечення;
- Відкритий вихідний код;
- Кросплатформна архітектура.

Недоліки:

- Обмежений функціонал порівняно з комерційними аналогами;
- Менш розвинені бібліотеки компонентів;
- Вільна модель підтримки.

EAGLE - це скриптове програмне забезпечення для автоматизації електронного проектування з функціями схематичного введення даних, компонування друкованих плат, автоматичного маршрутизації та комп'ютерного управління виробництвом.

EAGLE розшифровується як Easily Applicable Graphical Layout Editor і розробляється компанією CadSoft Computer GmbH.

Робоча область створення EAGLE (рис. 1.7).

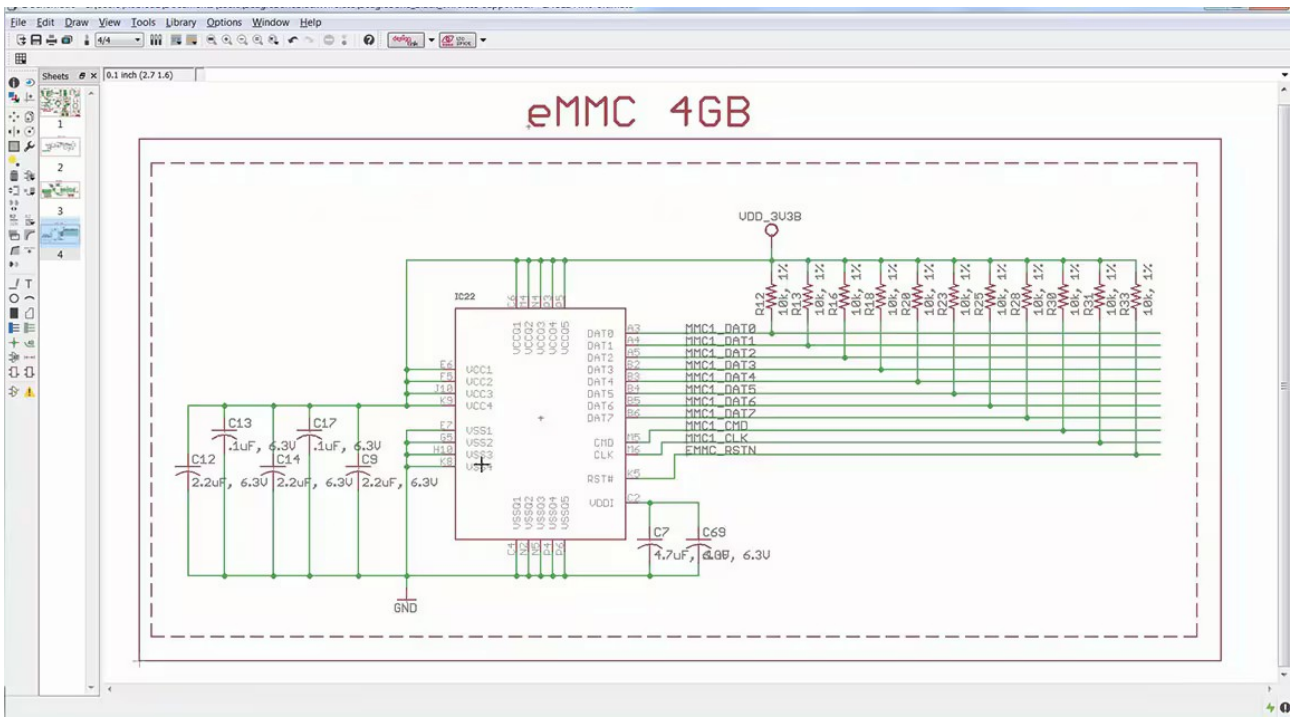

Рис. 1.7 - Робоча область

Приклад відображення конструкції друкованих плат в EAGLE (рис. 1.8).

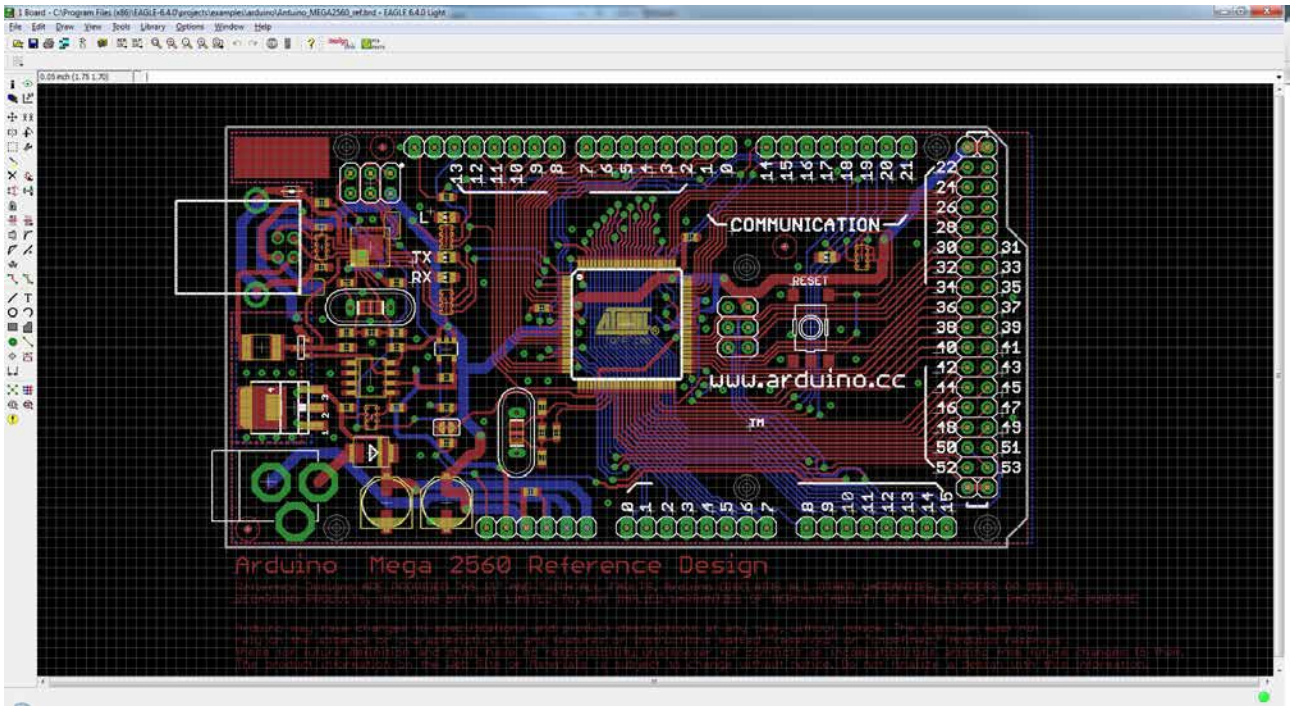

Рис. 1.8 - конструкція друкованої плати в EAGLE

Переваги:

- Зручний інтерфейс;
- Помірна вартість;
- Широка база бібліотек.

Недоліки:

- Обмеження безкоштовної версії;
- Складності з великими проєктами;
- Застарілий графічний інтерфейс.

1.3 Моделювання предметної області

Діаграма прецедентів для програмного застосунку проєктування електронних компонентів (рис. 1.9) є важливим інструментом для візуалізації взаємодії між користувачем та системою. Ця діаграма ілюструє основні функції та можливості застосування, а також показує, як актор взаємодіє з системою для досягнення своїх цілей.

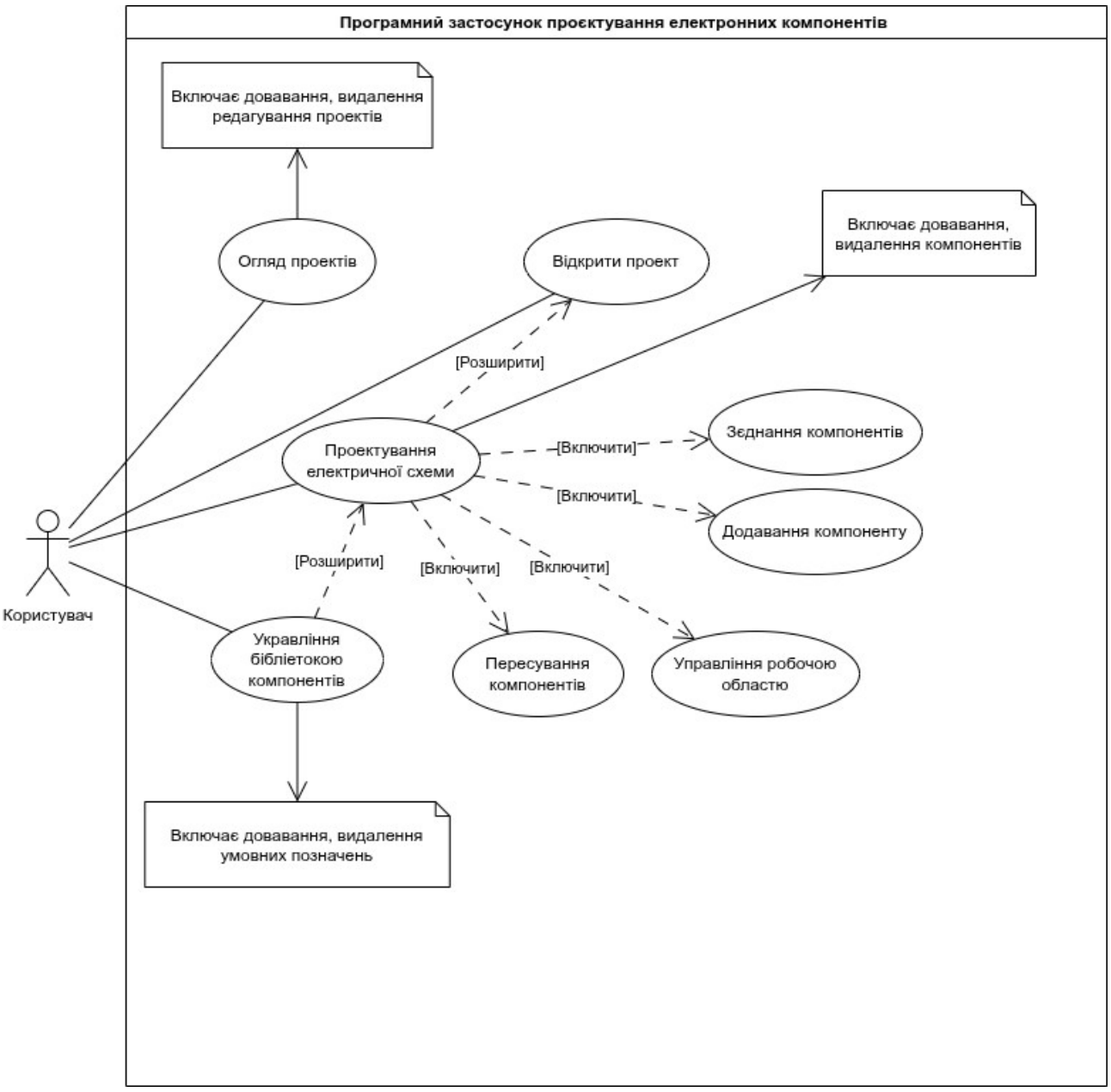

Рис. 1.9 - Діаграма прецедентів

Діаграма послідовностей ілюструє взаємодію об'єктів у системі в хронологічному порядку, показуючи послідовність повідомлень між ними (рис. 1.10). Для програмного застосунку проєктування електронних компонентів, допомагає візуалізувати взаємодію між модулями.

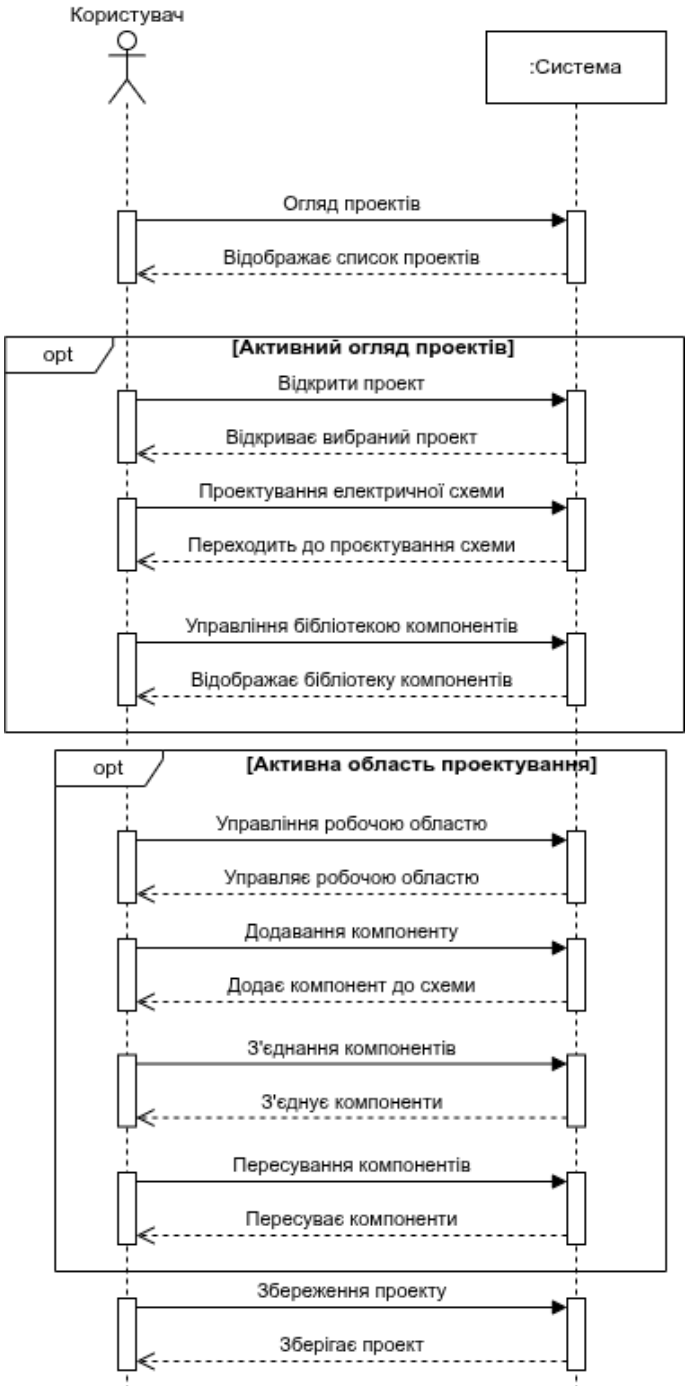

Рис. 1.10 - Діаграма послідовностей

Діаграма активності ілюструє послідовність дій і станів у процесі розробки електронних компонентів з використанням програмного забезпечення (рис. 1.11). Вона включає початкові та кінцеві стани, конкретні дії, переходи між ними та проміжні стани. Діаграма допомагає візуалізувати та оптимізувати процес проектування, покращуючи комунікацію та управління проектами.

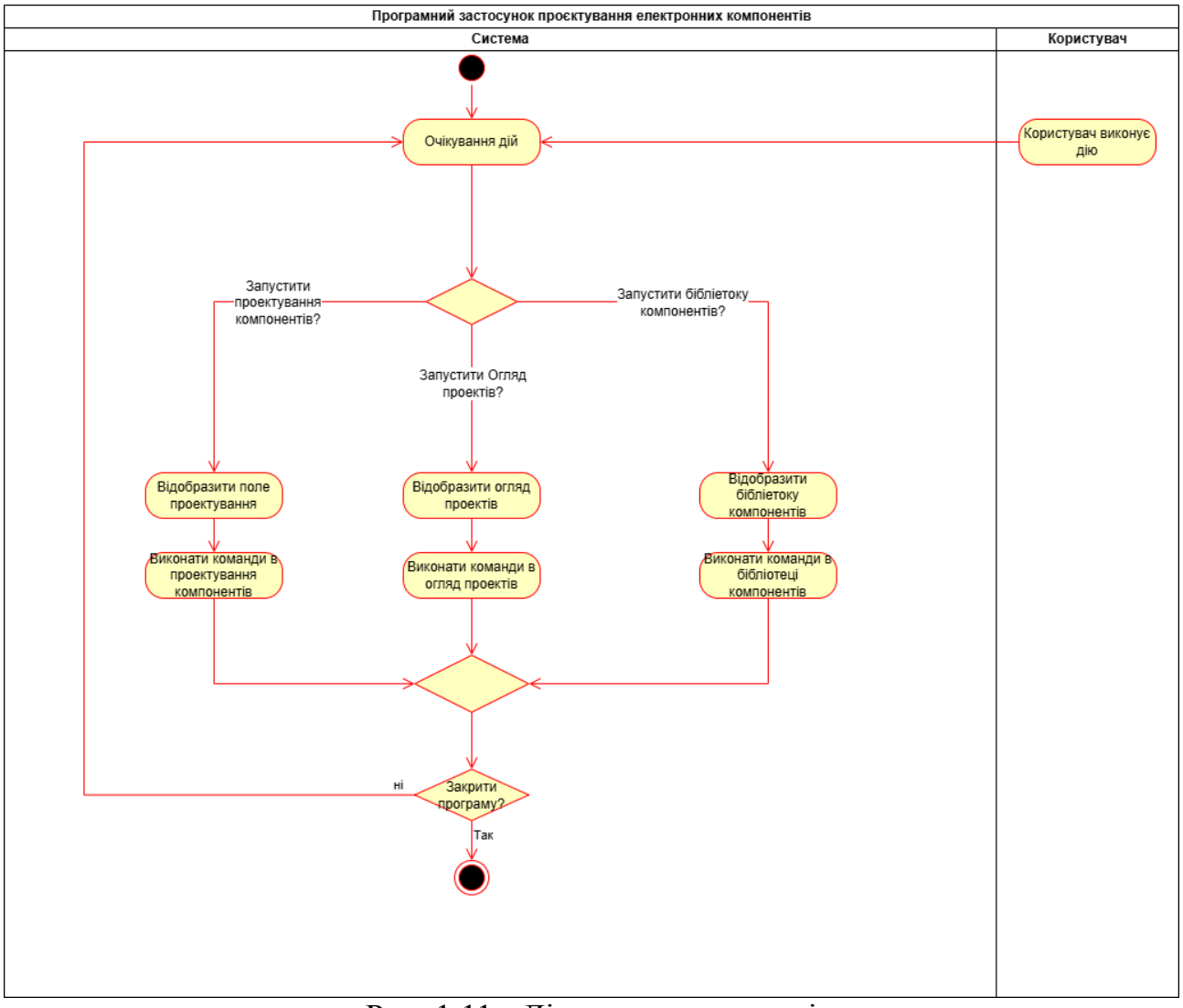

Рис. 1.11 - Діаграма активності

1.4 Вибір інструментарію для створення ППЗ

Java є оптимальним вибором для розробки програмного застосування проектування електронних компонентів з низки об'єктивних причин. Мова Java забезпечує повну кросплатформність завдяки концепції віртуальної машини (рис. 1.12). Для програми проектування, яка має бути доступна широкому колу інженерів і проектувальників, ця властивість має критичне значення, оскільки дозволяє значно знизити витрати на підтримку та адаптацію.

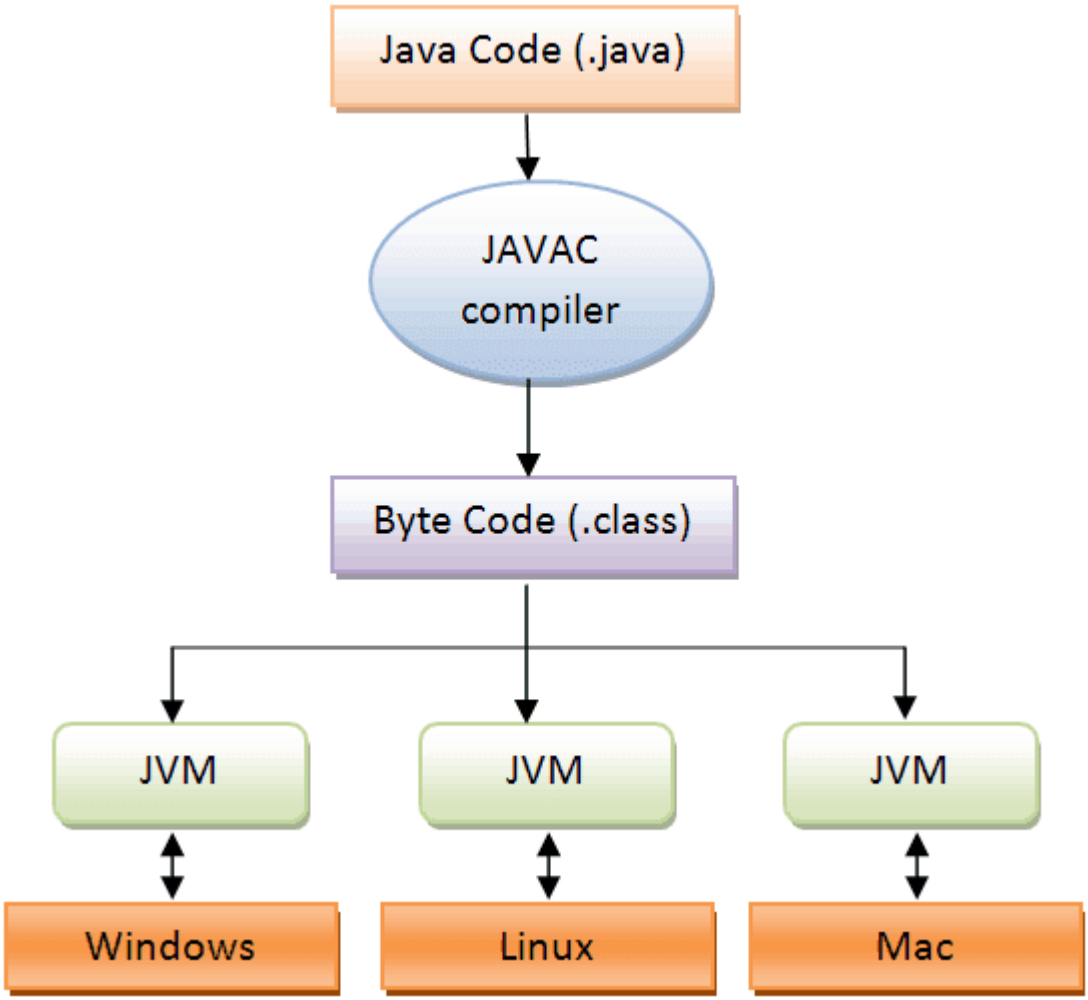

Рис. 1.12 - Архітектура java

Має зарекомендовані фреймворки для розробки графічних інтерфейсів. Фреймворк Swing дозволяє створювати професійні інтерфейси з підтримкою малювання векторної графіки, що необхідно візуалізації схем електронних компонентів. Ця бібліотека містять вбудовані компоненти для роботи з двовимірною графікою, обробки подій миші та клавіатури, управління шарами та трансформаціями об'єктів (рис. 1.13). Така функціональність дозволяє зосередитися на логіці програми, а чи не на низькорівневих деталях графічного виводу.

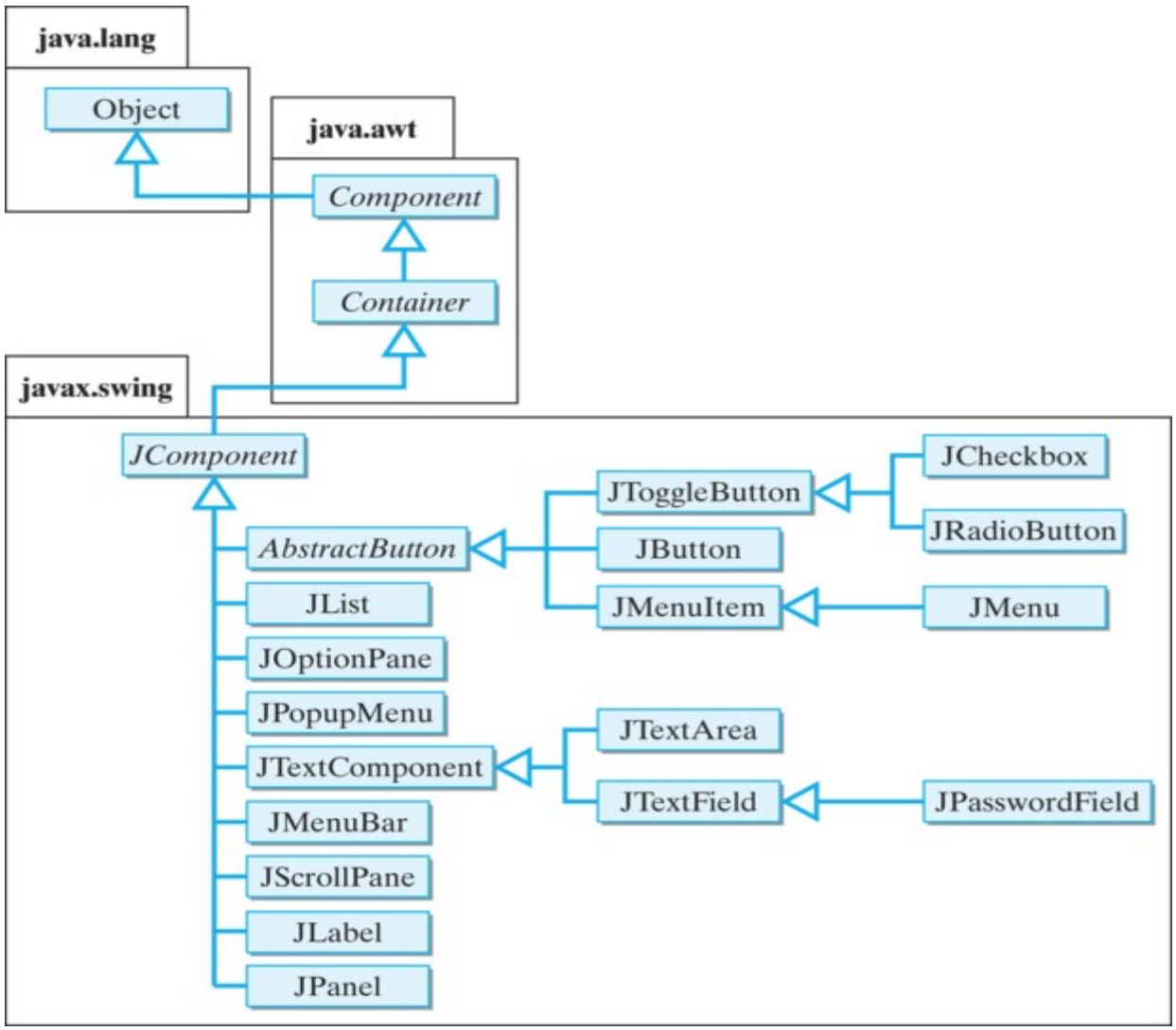

Рис. 1.13 - Архітектура Swing

Має вбудовану та добре документовану підтримку роботи з форматом XML. Цей формат є стандартом для зберігання структурованих даних у обчислювальній техніці та широко застосовується в електронній інженерії для опису конфігурацій компонентів та їх зв'язків. Java надає кілька підходів до роботи з XML. Модель DOM (Document Object Model) дозволяє повністю завантажувати документ до пам’яті та модифікувати його структуру, що зручно для інтерактивного редагування схем (рис. 1.14).

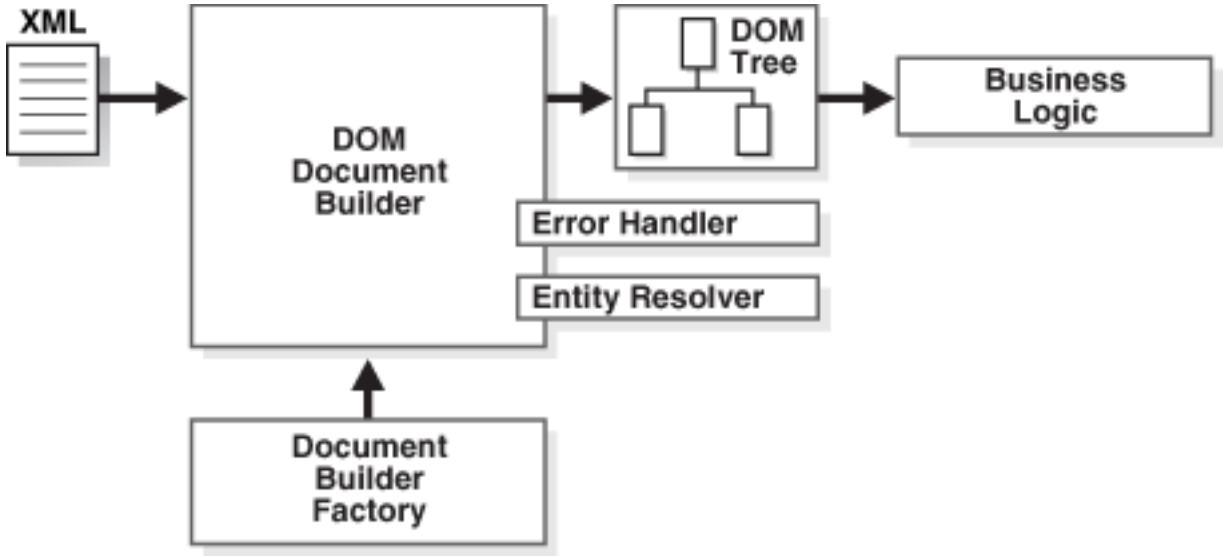

Рис. 1.14 - Створення XML документа в java

Java є статично типізованою мовою з автоматичним складанням сміття. Це підвищує надійність програми, що розробляється, так як багато помилок виявляються на етапі компіляції, а не під час виконання. Для складного застосування проектування, де потрібна обробка безлічі взаємозалежних об'єктів і операцій, статична типізація забезпечує додатковий рівень контролю якості коду і знижує ймовірність виникнення логічних помилок, що важко виловити.

Java забезпечує хорошу продуктивність додатків такого класу. Java поступається компілюваним мовам низького рівня в абсолютних показниках швидкості виконання, для інтерактивного застосування проектування електронних компонентів продуктивність сучасної Java-машини є достатньою.

Таким чином, вибір Java для розробки програми проектування електронних компонентів обґрунтований кросплатформністю, наявністю потужних інструментів для графічного програмування, вбудованою підтримкою XML, надійністю, зрілістю екосистеми та адекватною продуктивністю.

Eclipse є оптимальним вибором для розробки програмного застосунку проєктування електронних компонентів на мові Java. Eclipse представляє собою модульне середовище розробки з відкритим вихідним кодом, що поширюється на умовах ліцензії Eclipse Public License. Модульна архітектура Eclipse базується на системі плагінів, що дозволяє розширювати функціональність середовища додатковими інструментами без модифікації ядра (рис. 1.15).

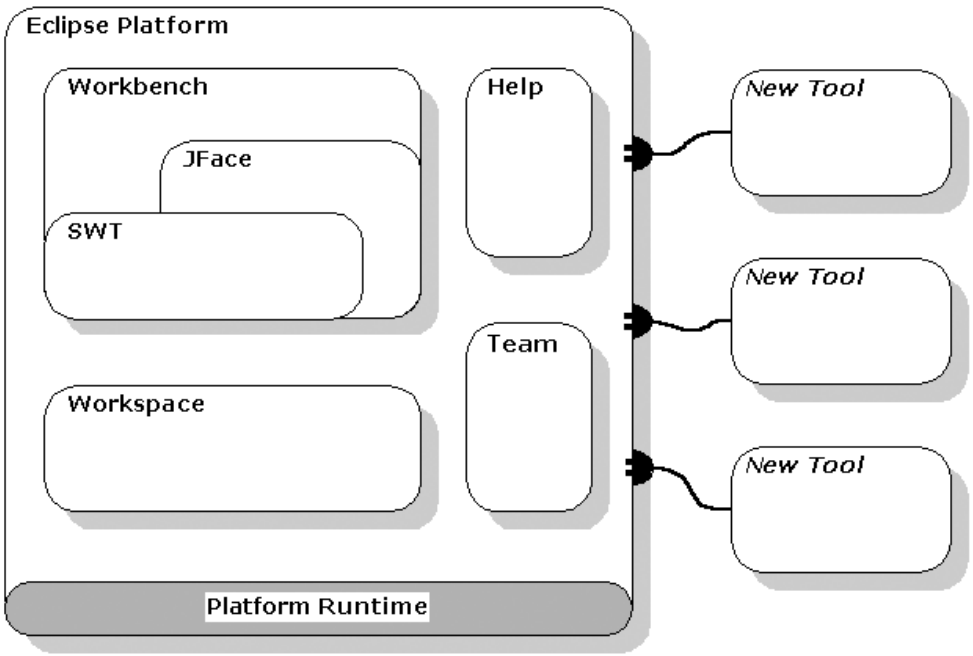

Рис. 1.15 - Архітектура Eclipse

Eclipse забезпечує комплексну підтримку мови Java на всіх етапах розробки. Вбудований редактор коду надає автоматичне доповнення синтаксису, аналіз помилок у реальному часі, рефакторинг коду та навігацію за класами та методами (рис. 1.16) та (рис. 1.17).

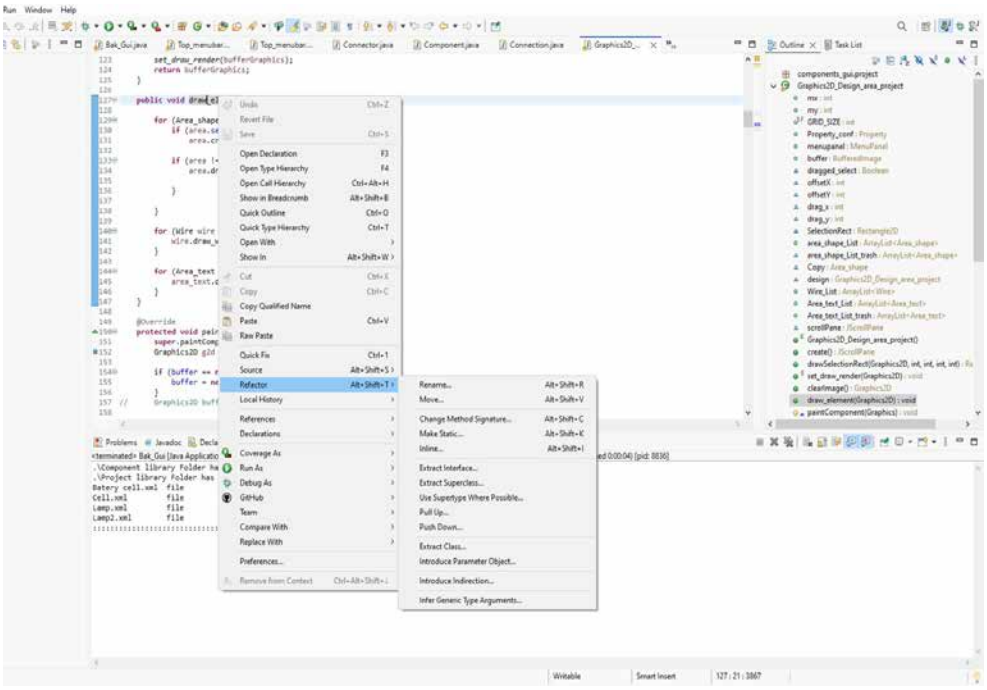

Рис. 1.16 - Функції рефакторингу

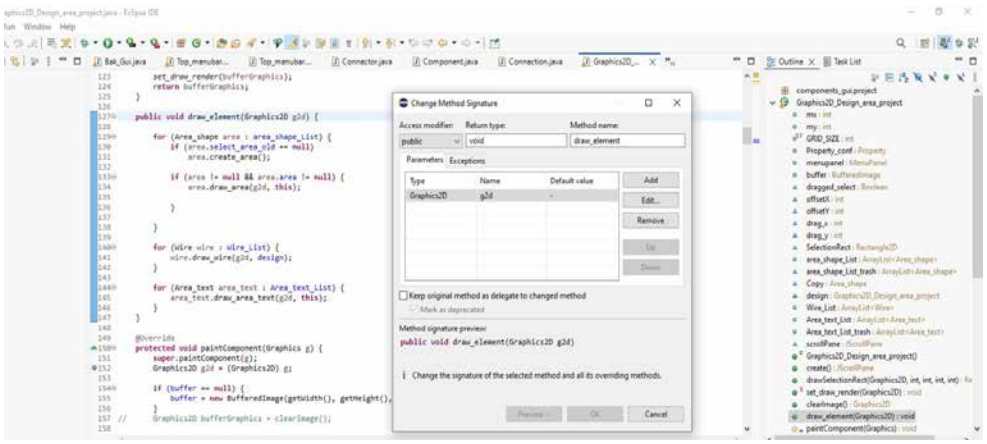

Рис. 1.17 - Рефакторинг методу

Ці можливості значно прискорюють процес розробки та зменшують кількість синтаксичних помилок.

Система відладження Eclipse (рис. 1.18) дозволяє встановлювати точки зупинення, переглядати стан змінних, виконувати код покроково та аналізувати стеки викликів, що є необхідною функціональністю для налагодження складних логічних помилок у застосунку.

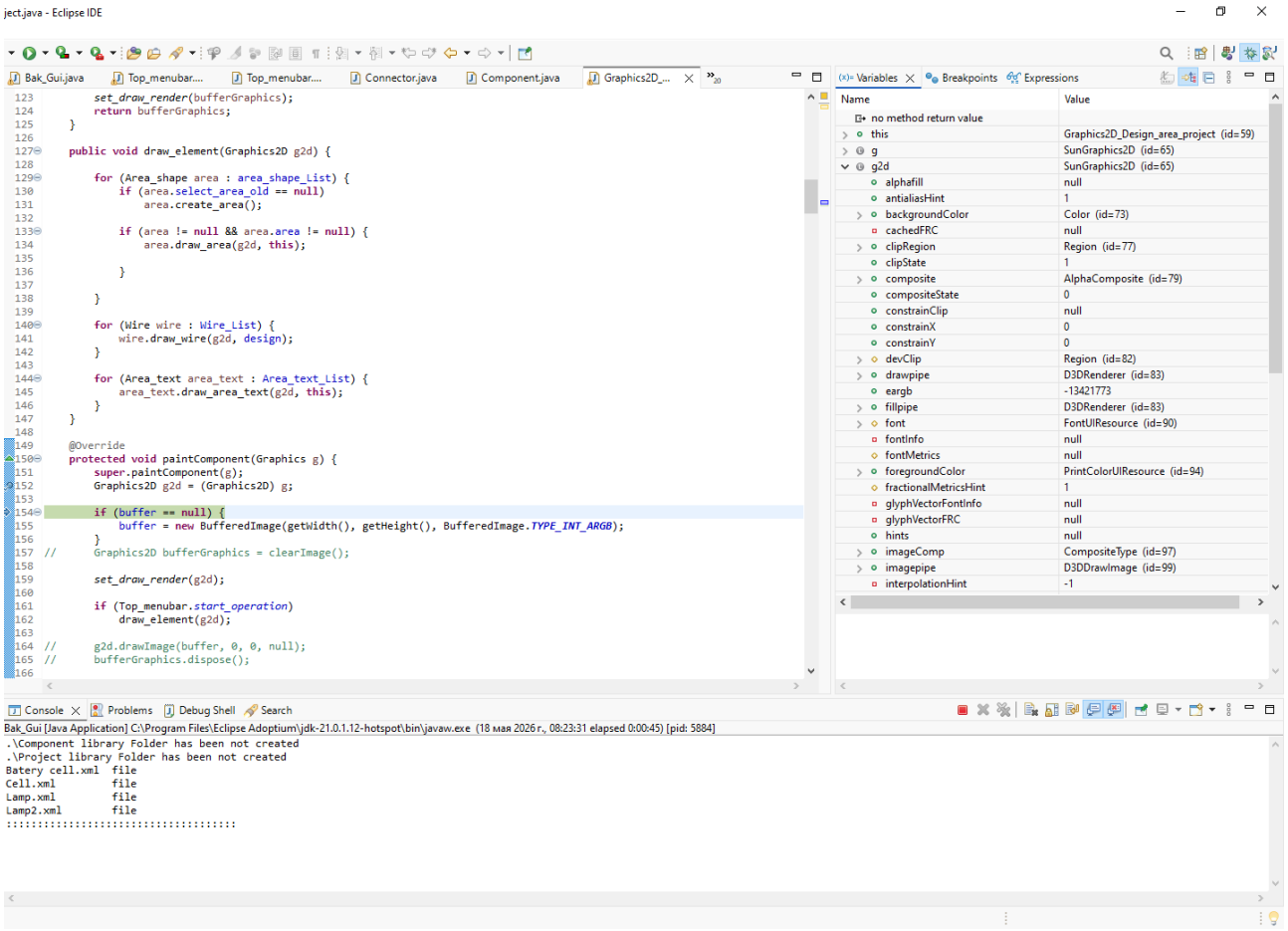

Рис. 1.18 - Режим відладження Eclipse

Eclipse постачається з вбудованим конструктором форм (рис. 1.19), що дозволяє розробляти графічний інтерфейс. Eclipse є достатньо функціональний для створення базового інтерфейсу користувача програми проєктування. Конструктор генерує Java-код, який можна далі коригувати вручну для досягнення необхідної функціональності.

Eclipse є повністю безплатним та має невеликий розмір встановлення порівняно з іншими середовищами розробки. Це дозволяє розпочати розробку без затрат та встановити середовище навіть на комп'ютерах з обмеженими ресурсами. Незважаючи на невеликий розмір порівняно з іншими ide, Eclipse забезпечує достатній набір функцій для ефективної розробки середньомасштабних Java-проектів.

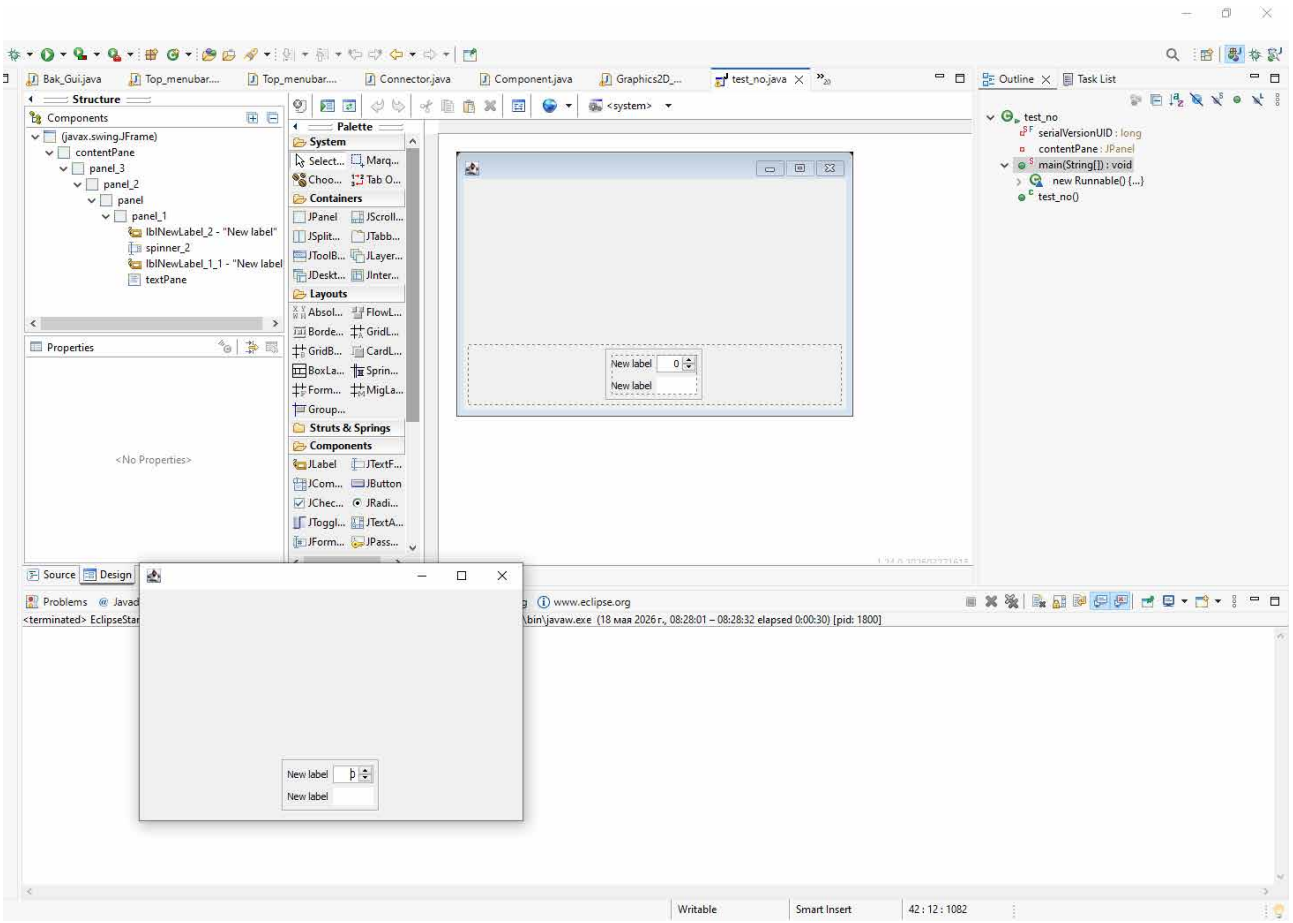

Рис. 1.19 - Конструктор форм

Обґрунтування вибору засобів для розробки інтерфейсу з базою даних.

Пакет org.w3c надає стандартизовані інструменти для роботи з документами XML відповідно до офіційних специфікацій Консорціуму Всесвітньої павутини.

Використання org.w3c дозволяє розробнику створювати XML структури, які відповідають міжнародним стандартам і можуть бути прочитані та оброблені іншими програмами та системами. Коли візуальні компоненти зберігаються в XML файл з використанням org.w3c, кожен компонент може бути представлений як окремий елемент XML з атрибутами, що описують його властивості та стан.

Наприклад, прямокутник може бути збережений як елемент з атрибутами координат X і Y, ширини, висоти та кольору, що дозволяє повністю відновити його зовнішній вигляд при завантаженні з файлу.

Наприклад, щоб зберегти лінію в xml, потрібно створити гілку ShapeCGS для збереження фігури (рис. 1.20).

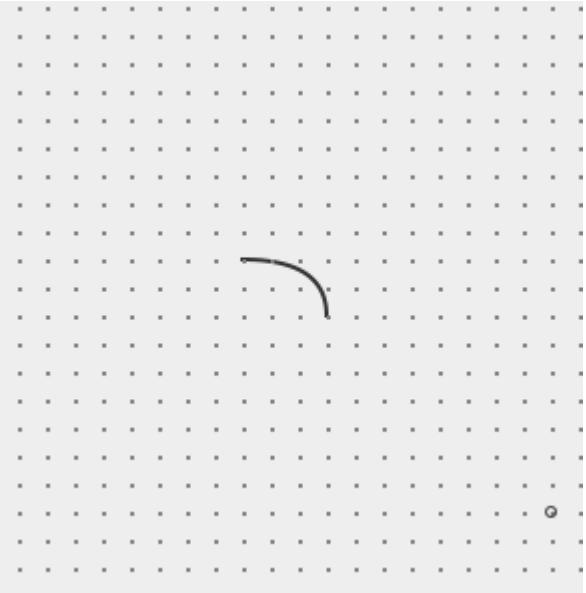

Рис. 1.20 - Крива

<ShapeCGS BASICSTROKE_WIDTH="2.0" height="0" id="0" rotate="0" scale_x="1.0" scale_y="1.0" width="0">
<X>153.0</X>
<Y>363.0</Y>
<Type>GeneralPath</Type>
<Description/>
<Point_paths>
<PointPath mode="Line" x="-21.0" x2="-175.0" y="-14.0" y2="-378.0"/>
<PointPath mode="Quad" x="21.0" x2="21.0" y="14.0" y2="-14.0"/>
</Point_paths>
</ShapeCGS>

Для збереження Rectangle (рис. 1.21) достатньо наступного xml.

<ShapeCGS BASICSTROKE_WIDTH="-1.0" height="40" id="1" rotate="0" scale_x="0.0" scale_y="0.0" width="40">
<X>196.0</X>
<Y>448.0</Y>
<Type>Rectangle</Type>
<Description/>
<Point_paths/></ShapeCGS>

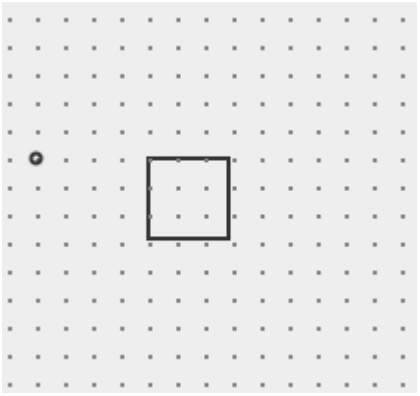

Рис. 1.21 - Rectangle

Пакет javax.xml.xpath відіграє важливу роль в обробці та видаленні даних із збережених XML файлів. XPath є мовою запитів, яка дозволяє знаходити та вибирати потрібні елементи у XML документі без необхідності повної його обробки.

Коли програма завантажує збережений XML файл із графічними компонентами, javax.xml.xpath дозволяє розробнику писати точні запити для пошуку конкретних компонентів або групи компонентів на основі різних критеріїв.

Комбінація org.w3c та javax.xml.xpath забезпечує повний цикл роботи зі збереженими компонентами від створення до відновлення. Коли користувач малює компоненти в Java програмі, їх властивості та параметри можуть бути перетворені на XML елементи та збережені у файл з використанням API з пакету org.w3c.

При наступному запуску програми XML файл завантажується, і javax.xml.xpath використовується для отримання даних про кожен компонент. На основі цих вилучених даних програма може відновити візуальне подання компонентів на екрані, відтворивши їх точне розташування, розміри, кольори та інші візуальні властивості.

Обґрунтування вибору засобів для розробки інтерфейсу користувача.

Swing побудований на принципі ієрархії компонентів, де кожен елемент інтерфейсу - це об'єкт, що має власний стан і поведінку. Ця архітектура має низку архітектурних переваг. Кожен компонент - це закінчена одиниця, що приховує внутрішню складність, і розробник взаємодіє з інтерфейсом, не переймаючись деталями реалізації. Завдяки такій інкапсуляції компоненти легко перевикористовувати в різних місцях програми або навіть у різних додатках.

Проблема адаптації інтерфейсу до різних розмірів екрана та роздільної здатності – одна з проблем у розробці UI. Swing вирішує цю проблему через систему Layout Managers. Кожен контейнер може мати свій менеджер розташування, який автоматично управляє позиціонуванням та розмірами компонентів. BorderLayout розбиває простір на п'ять зон (північ, південь, схід, захід, центр) (рис. 1.22).

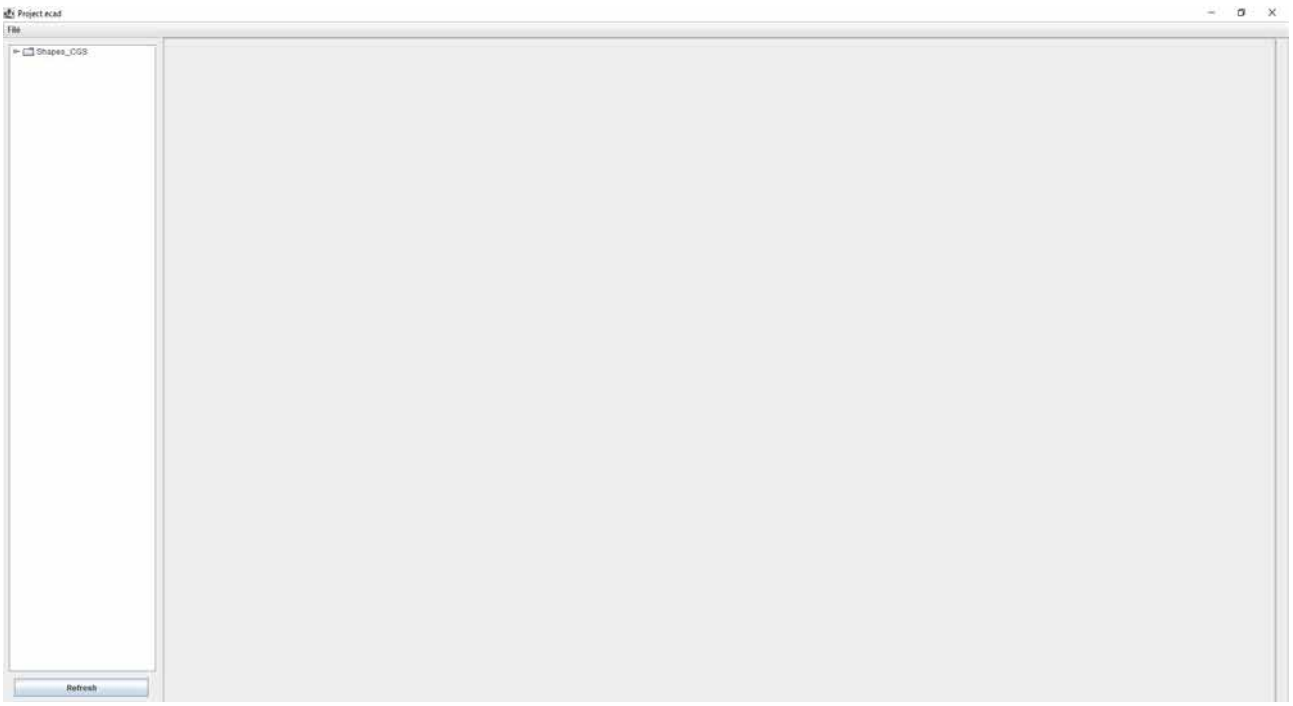

Рис. 1.22 - Демонстрація використання BorderLayout

GridLayout створює рівномірну сітку, FlowLayout розміщує елементи в рядок з перенесенням, GridBagLayout надає гнучку сітку з підтримкою розтягування компонентів, а BoxLayout вибудовує елементи (рис. 1.23).

Завдяки цьому підходу інтерфейс автоматично адаптується до зміни розміру вікна, програма виглядає прийнятно на екранах різних розмірів, і розробник не повинен вручну перераховувати координати при кожній зміні розміру.

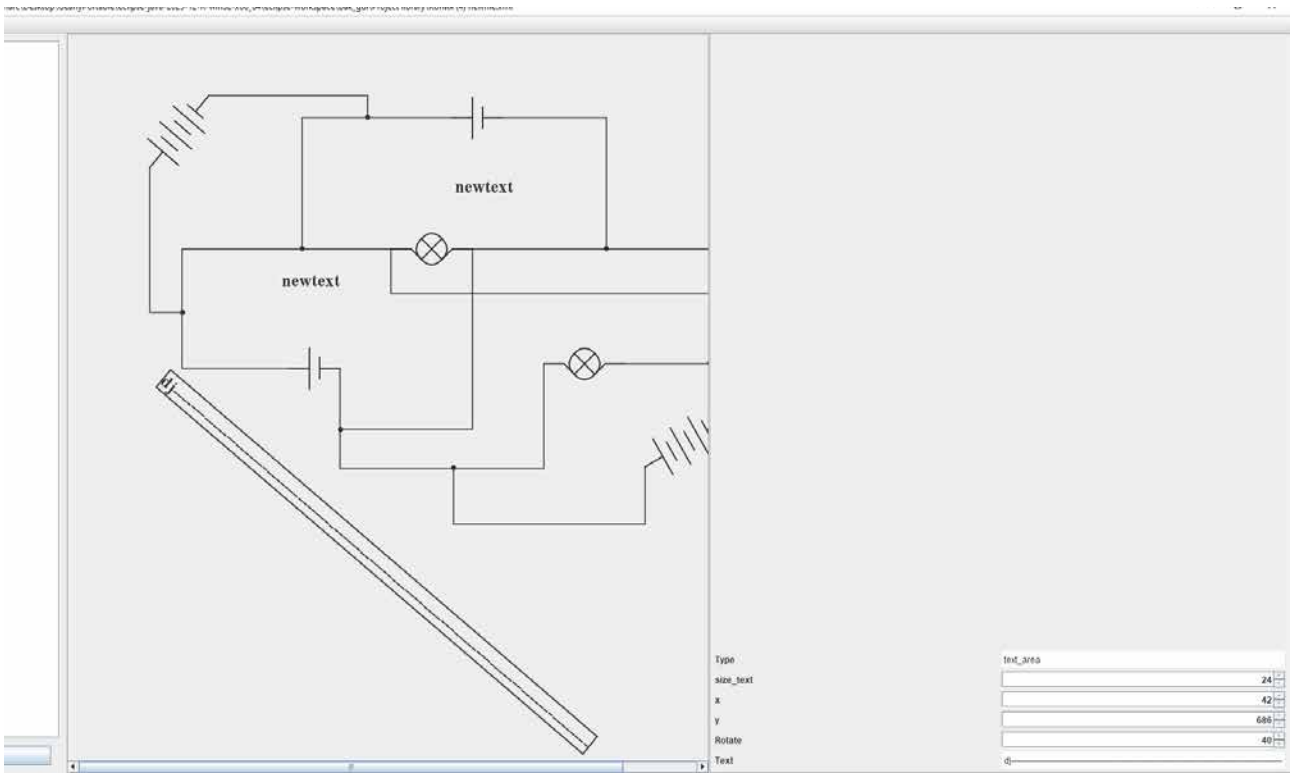

Рис. 1.23 - Демонстрація адаптації інтерфейсу

Swing повставляється із розширеним набором готових компонентів. До простих компонентів відносяться JButton, JLabel, JTextField, JCheckBox та JRadioButton. До складних компонентів належать JTable для роботи з таблицями, JTree для відображення деревоподібних структур (рис. 1.24) та JList для списків. Контейнери представлені JFrame, JDialog, JPanel та JTabbedPane.

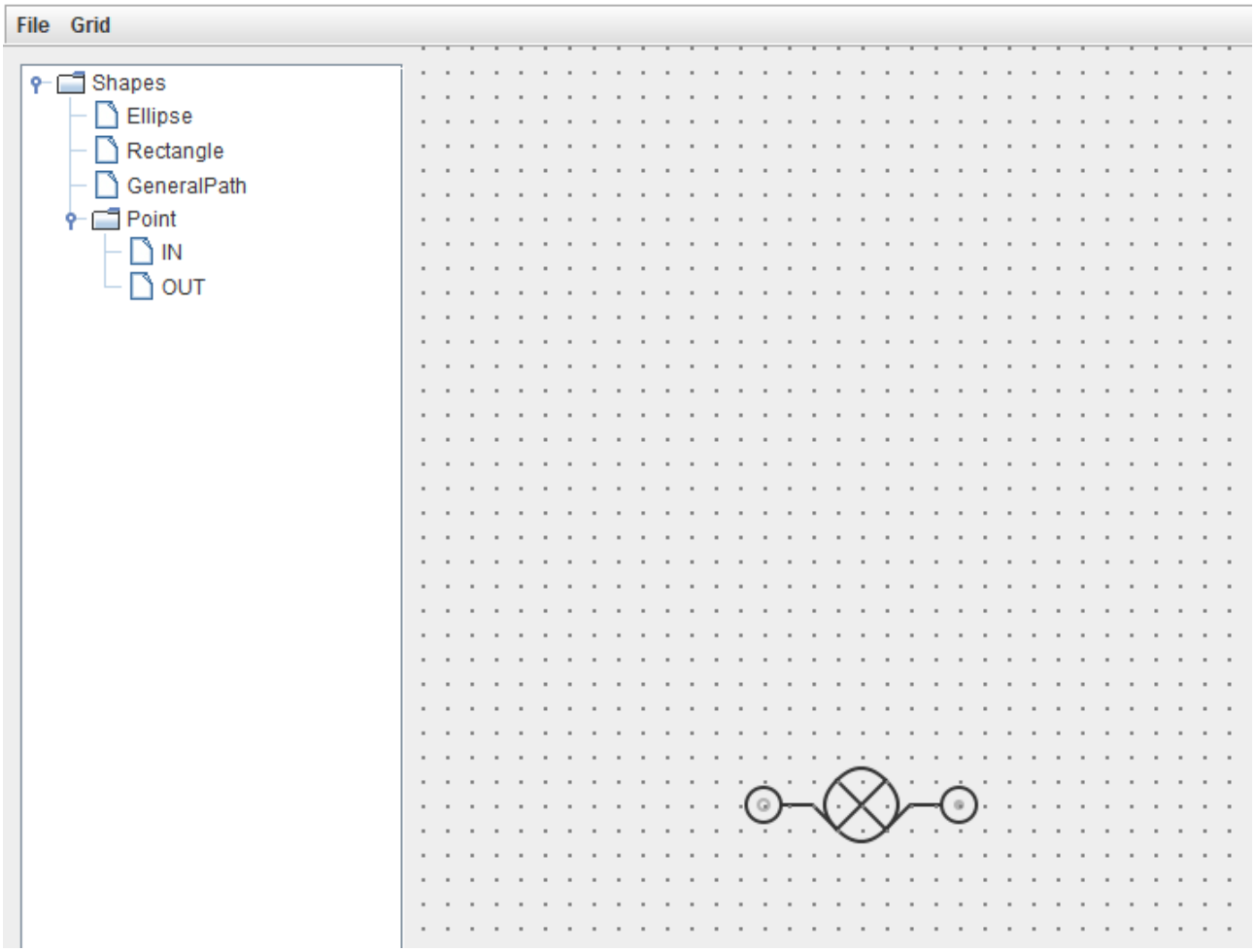

Рис. 1.24 - Демонстрація JTree

Вбудовані діалоги включають JFileChooser (рис. 1.25) для відкриття файлів. Наявність таких вбудованих компонентів означає, що більшість типових завдань UI можуть бути вирішені без залучення зовнішніх бібліотек, що спрощує розгортання та знижує кількість залежностей.

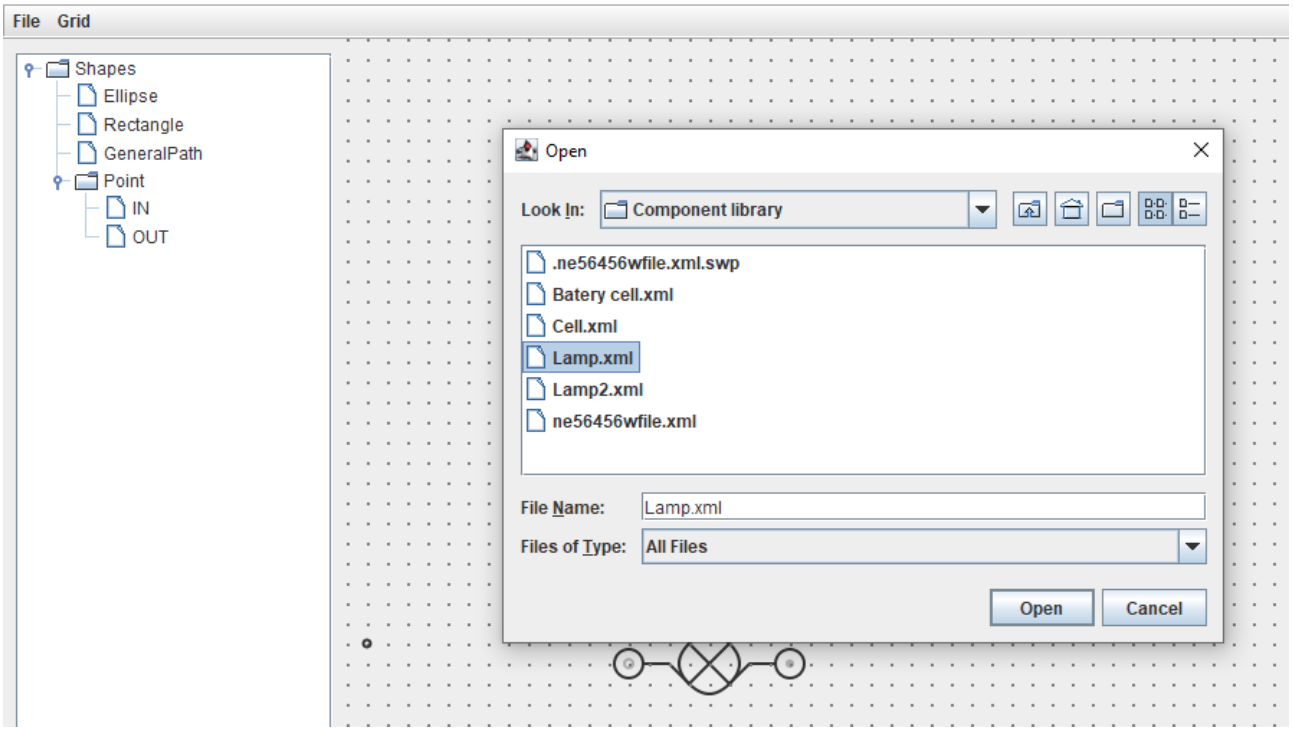

Рис. 1.25 - Демонстрація JFileChooser

Swing побудований на основі архітектури, керованої подіями. Це означає, що програма реагує на дії користувача: натискання кнопки, введення тексту, рух миші тощо.

Система подій у Swing характеризується типізованістю - кожна подія представлена об'єктом певного класу (ActionEvent, MouseEvent, KeyEvent тощо), що містить всю необхідну інформацію про цю подію. Розробник може зареєструвати слухача на компоненті, і цей слухач буде викликаний у разі виникнення події.

Обґрунтування вибору засобів для формування звітів.

Для формування звітів у вигляді зображень у Java використовуються класи java.awt.Graphics2D і javax.imageio.ImageIO. Обидва ці засоби мають свої особливості та переваги, які роблять їх підходящими для цих завдань.

Graphics2D - це клас з пакета java.awt, який надає розширені можливості для малювання двовимірної графіки. Він є підкласом Graphics і надає більше контролю над процесом малювання. Graphics2D дозволяє застосовувати трансформації, такі як масштабування, обертання та перенесення, що дозволяє легко змінювати позицію та розмір малюваних об'єктів. Graphics2D підтримує антиаліасинг, що дозволяє зменшити ступені та зробити малювання більш плавним і високоякісним. Graphics2D надає розширені можливості для малювання тексту, включаючи підтримку різних шрифтів та стилів тексту. Graphics2D підтримує альфа-змішування, що дозволяє створювати прозорі ефекти та накладати один малюнок на інший з різним рівнем прозорості.

ImageIO - це клас з пакета javax.imageio, який надає можливості для читання та запису зображень у різних форматах. ImageIO надає методи для читання та запису зображень, що спрощує процес роботи з зображеннями. ImageIO дозволяє читати та записувати метадані зображень, що може бути корисно для збереження додаткової інформації про звіт. ImageIO може працювати з потоками вводу/виводу, що дозволяє читати та записувати зображення.

Вибір Graphics2D та ImageIO для формування звітів у вигляді зображень обґрунтований їхніми можливостями та перевагами. Graphics2D надає можливості для малювання та трансформації графіки, що дозволяє створювати звіти (рис. 1.26). ImageIO дозволяє зберігати звіти у форматах зображень (рис. 1.27). Обидва класи надають методи для малювання та запису зображень, що спрощує процес розробки звітів.

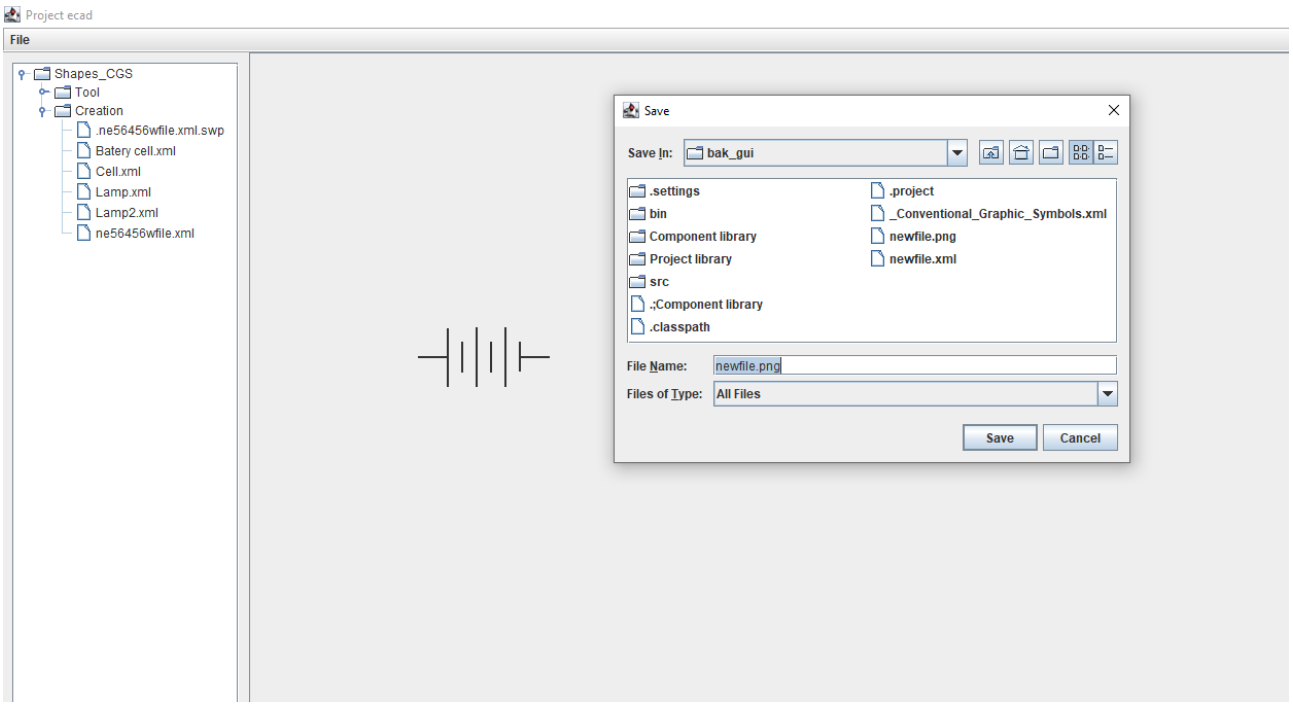

Рис. 1.26 - Створення звіту

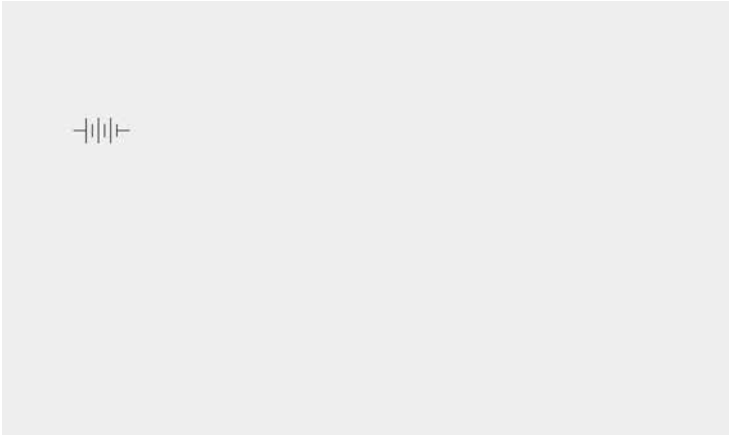

Рис. 1.27 - Файл звіту в png

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Логічна модель даних для програмного застосунку проєктування електронних компонентів описує структуру та взаємозв'язки даних, необхідних для управління процесами проєктування (рис. 2.1).

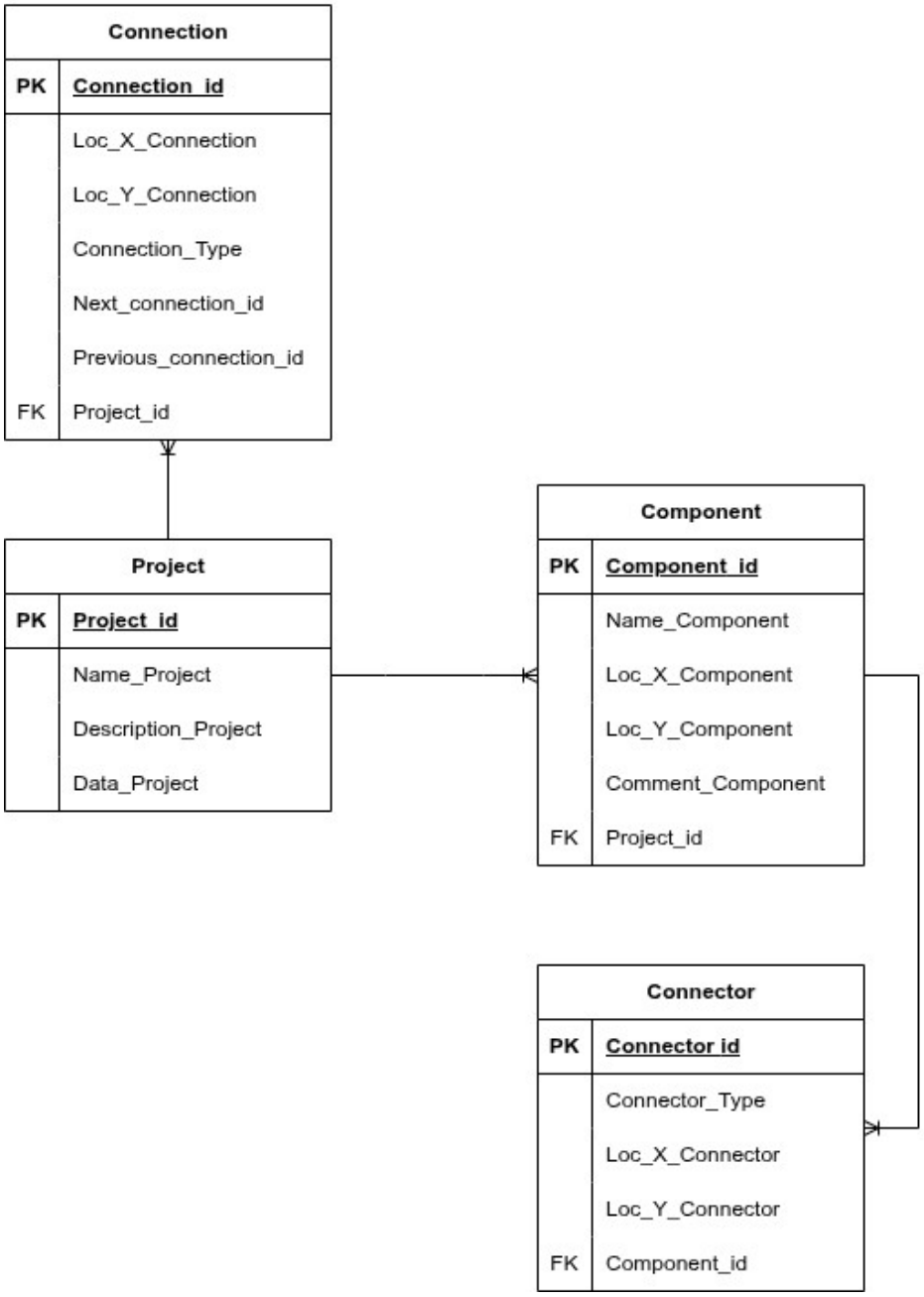

Рис. 2.1 - Логічна модель даних Project

project зберігає інформацію про проект, який створюються користувачем в програмному забезпеченні для проектування електронних компонентів.

- Name_project зберігає назву проекту, яку користувач задає при створенні нового проекту.
- Description_project містить додаткову інформацію про проект.
- Data_project зберігає мітку часу, коли проект був створений.

component зберігає інформацію про компоненти, які належать до проекту. Компоненти можуть бути різними електронними елементами, такими як резистори, конденсатори, транзистори тощо.

- Name_component зберігає ім'я компонента, яке користувач задає при створені нового компонента в бібліотеці.
- Loc_x_component зберігає координату x, що вказує положення компонента на схемі.
- Loc_y_component зберігає координату y, що вказує положення компонента на схемі.
- Comment_component містить додаткові коментарі або описи.

Connector зберігає інформацію про конектори, які належать компонентам. Конектори являють собою точки з'єднання між компонентами.

- Connector_type зберігає тип конектора, наприклад, вхідний, вихідний, загальний тощо.
- Loc_x_connector зберігає координату x, що вказує положення конектора на схемі.
- Loc_y_connector зберігає координату y, що вказує положення конектора на схемі.

connection зберігає інформацію про з'єднання між компонентами в рамках проекту. З'єднання являють собою лінії або шляхи, які з'єднують конектори різних компонентів.

- Loc_x_connection зберігає координату x, що вказує положення з'єднання на схемі.
- Loc_y_connection зберігає координату y, що вказує положення з'єднання на схемі.
- Connection_type зберігає тип з'єднання, наприклад, прямий, перехресний тощо.
- Next_connection_id дозволяє пов'язати з'єднання в ланцюжок з'єднань.
- Previous_connection_id також дозволяє пов'язати з'єднання в ланцюжок.

Зв'язки між project і component - один проект може містити безліч компонентів. Кожен компонент належить одному проекту.

Component і connector - один компонент може містити безліч конекторів. Кожен конектор належить одному компоненту.

Project і connection - один проект може містити безліч з'єднань. Кожне з'єднання належить одному проекту.

Логічна модель даних описує структуру та взаємозв'язки даних, необхідних для управління процесами проєктування у створені умовних позначень для компонентів (рис. 2.2).

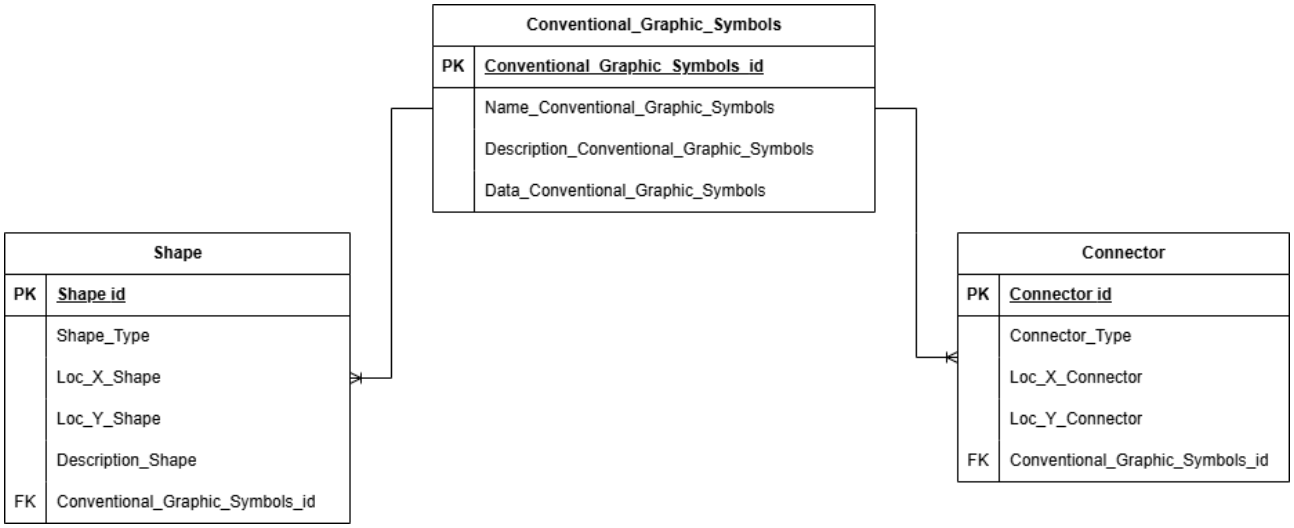

Рис. 2.2 - Логічна модель даних Conventional_Graphic_Symbols

shape зберігає інформацію про форми, що використовуються в проектуванні електронного компонента. Кожна форма має певний тип, розташування та опис, а також пов'язана з графічним символом.

- Shape_type - вказує тип форми (наприклад, прямокутник, коло, багатокутник).
- Loc_x_shape - координата x розташування форми на робочому полі.
- Loc_y_shape - координата y розташування форми на робочому полі.
- Description_shape - детальний опис форми, включаючи її призначення та будь-які специфічні атрибути.

conventional graphic symbols зберігає інформацію про графічний символ, що використовуються в проектуванні електронних компонентів. Кожен символ має унікальний ідентифікатор, додаткові дані, назву та опис.

- Data_conventional_graphic_symbols - додаткові дані, пов'язані з графічним символом, такі як його властивості або атрибути.
- Name_conventional_graphic_symbols - назва графічного символу.
- Description_conventional_graphic_symbols - детальний опис графічного символу, включаючи його використання та значення в проектуванні електронних компонентів.

connector зберігає інформацію про з'єднувачі, що використовуються в проектуванні електронних компонентів. Кожен з'єднувач має певний тип, розташування і пов'язується з графічним символом.

- Connector_type - вказує тип з'єднувача (наприклад, штир, майданчик, провід).
- Loc_x_connector - координата x розташування з'єднувача на робочому полі.
- Loc_y_connector - координата y розташування з'єднувача на робочому полі.

Один графічний символ може містити безліч фігур з яких він складається. Кожна фігура належить одному графічному символу.

Один графічний символ може містити безліч конекторів. Кожен конектор належить одному графічному символу.

2.2 Вибір системи управління інформаційною базою

Вибір XML як основного формату зберігання та обробки даних для електронних компонентів ґрунтується на аналізі функціональних вимог проекту, характеристик формату та його придатності для предметної області.

XML - це універсальний мовний стандарт для представлення структурованих даних у текстовому форматі. Основною характеристикою XML є його ієрархічна структура, яка відображає взаємозв'язки між елементами системи. Кожен елемент може містити атрибути та вкладені елементи, що дозволяє описати складні залежності без необхідності нормалізації даних, притаманної реляційним базам даних.

Електронні компоненти мають ієрархічну організацію: кожен компонент характеризується набором параметрів, які, у свою чергу, мають атрибути. XML дозволяє представити цю ієрархію без перетворень.

XML-файли не залежать від операційної системи, архітектури процесора чи конкретного програмного забезпечення. Це означає, що дані, збережені в XML, можуть бути легко передані між різними системами проєктування, обробленими на різних платформах та інтегровані з іншими інструментами.

Така портативність особливо важлива в контексті електроніки, де часто необхідна взаємодія між різними розробниками електронних схем.

Одна з найважливіших переваг XML - це можливість легко додавати нові елементи та атрибути без необхідності перестроювання всієї системи.

XML забезпечує оптимальний баланс між простотою розробки, функціональністю та розширюваністю.

Структурована природа XML відповідає ієрархії електронних компонентів та схем, а портативність формату забезпечує можливість інтеграції з іншими промисловими інструментами.

Побудовано модель даних фізичного рівня системи xml.

Файл Project.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Project>
<Name/>
<Description/>
<Data/>
<Components>
 ...
</Components>
<Connections>
 ...
</Connections>
</Project>
```

Файл Conventional_Graphic_Symbols.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Conventional_Graphic_Symbols>
<Name/>
<Description/>
<Data/>
<Shapes>
 ...
</Shapes>
<Connectors>
 ...
</Connectors>
</Conventional_Graphic_Symbols>
```

Подано структуру бази даних у вигляді схеми xml.

Файл Project (рис. 2.3).

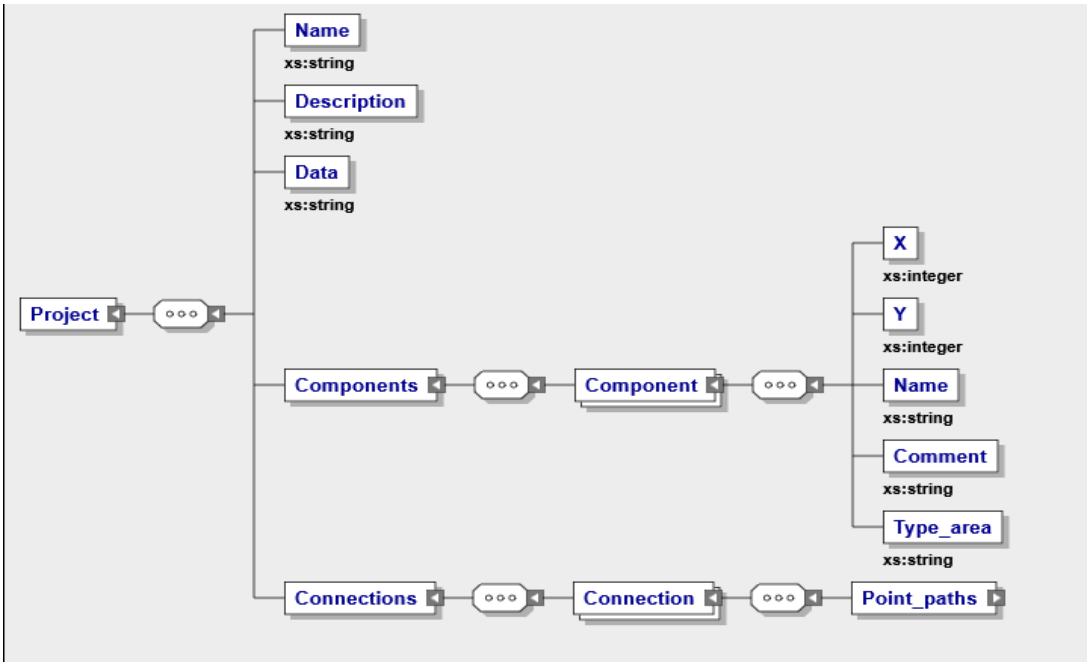

Рис. 2.3 - Project

Файл Conventional_Graphic_Symbols (рис. 2.4).

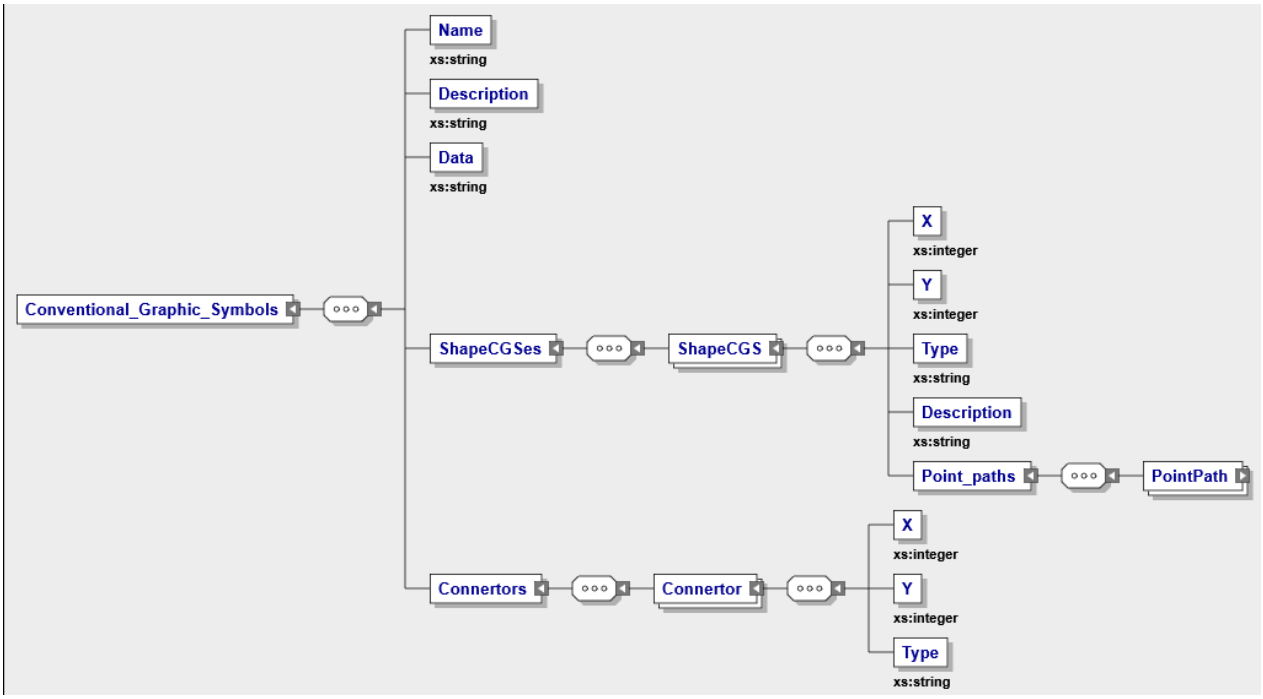

Рис. 2.4 - Conventional_Graphic_Symbols

2.3 Створення інформаційної бази

Для створення бази даних умовних позначень потрібно викликати наступний код.

```
Conventional_Graphic_Symbols _conventional_Graphic_Symbols = new Conventional_Graphic_Symbols(select_file.getName(),"","");

try {

_conventional_Graphic_Symbols.Conventional_Graphic_Symbols_Save_File(select_file.toString());
} catch (ParserConfigurationException | TransformerException e1) {
e1.printStackTrace();
}
```

У цьому коді створюється клас Conventional_Graphic_Symbols, який дозволяє викликати метод Conventional_Graphic_Symbols_Save_File.

Метод збереження дозволяє виконати збереження сворених елементів проектування. Опис методу наданий в (додаток А).

З самого коду видно, що метод збереження викликає метод save для себе самого та елементів проектування. Нижче вказані методи збереження для класів у такому порядку: Conventional_Graphic_Symbols (додаток А), ShapeCGS (додаток Б), Connector (додаток В).

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Змодельовано архітектуру системи програмного застосунку проєктування електронних компонентів у вигляді діаграми розгортання (рис. 3.1).

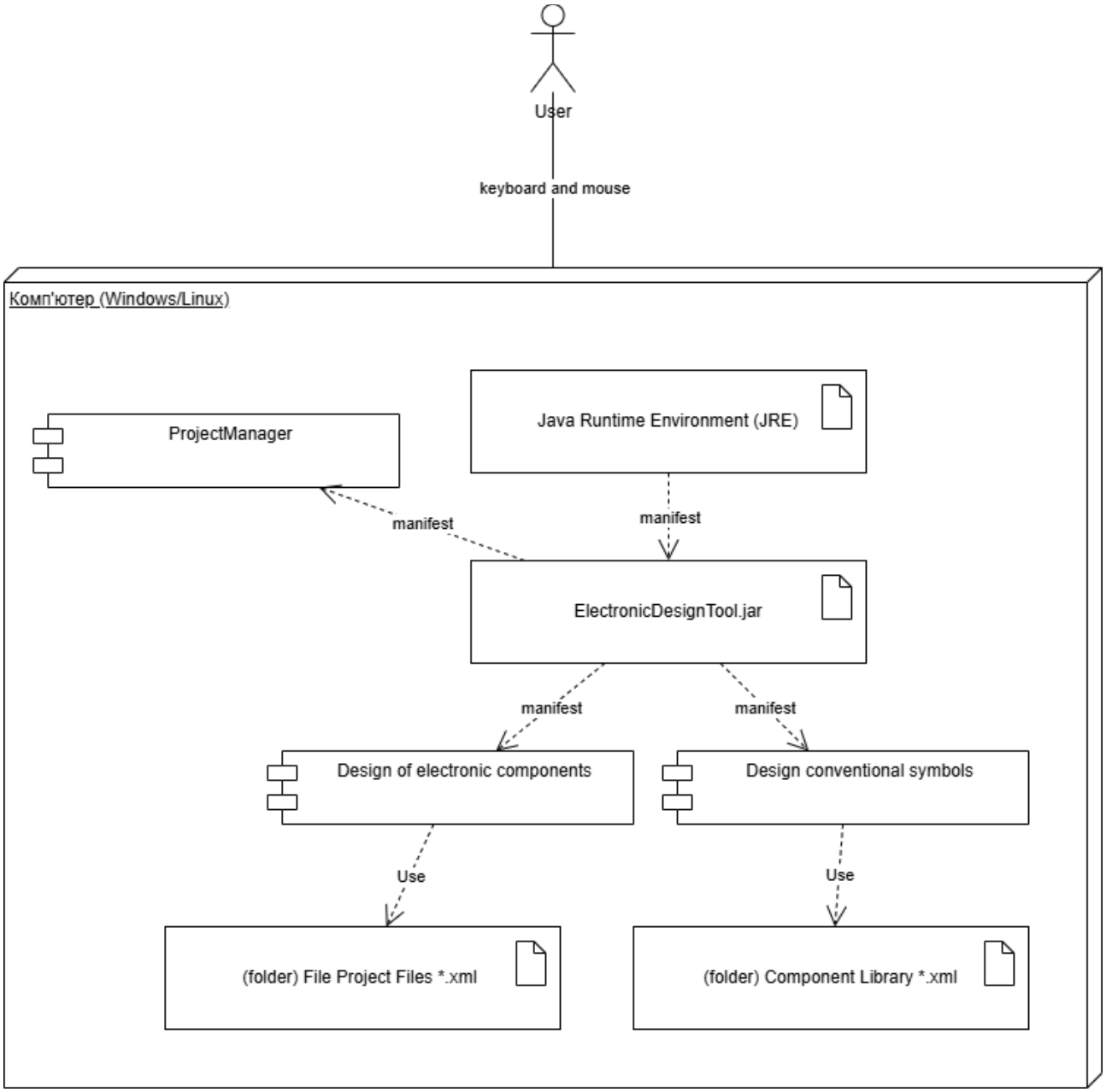

Рис. 3.1 - Діаграма розгортання

Змодельовано організаційну структуру програмного забезпечення програмного застосунку проєктування електронних компонентів у вигляді діаграми пакетів (рис. 3.2).

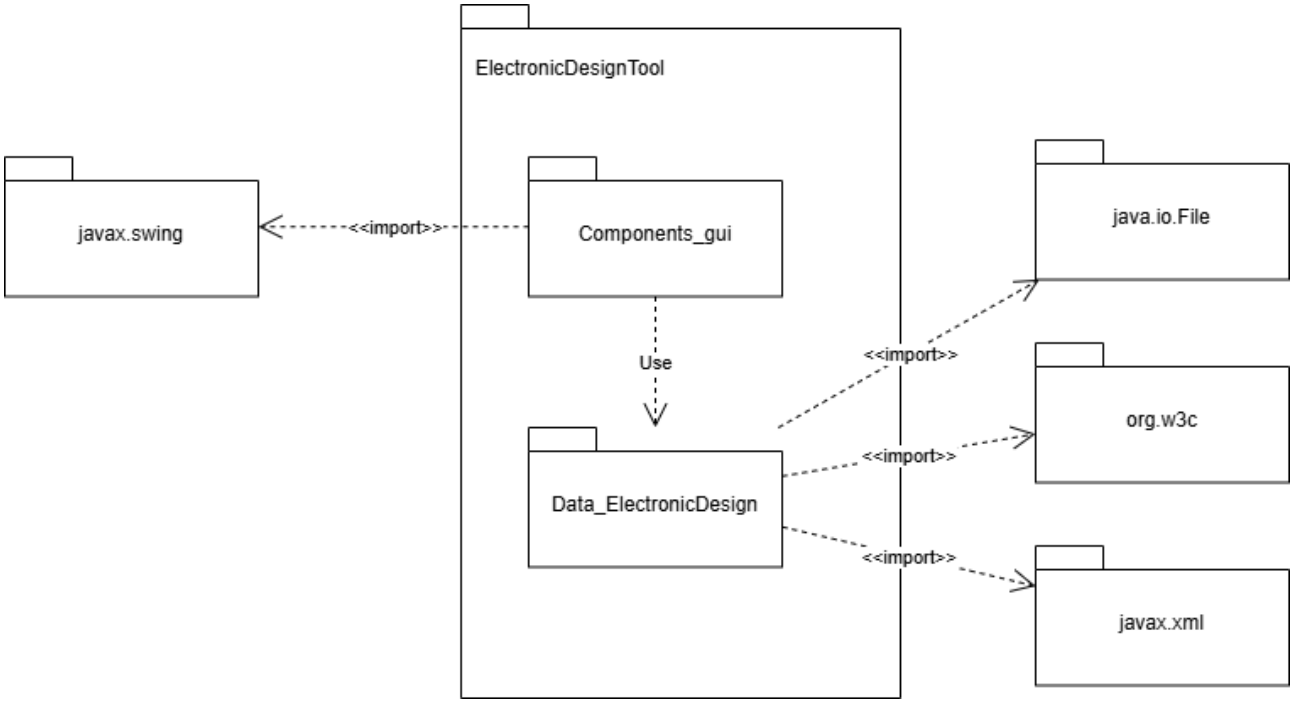

Рис. 3.2 - Діаграма пакетів

Діаграма показує, що система розподіляється на пакет компонентів інтерфейсу користувача, який використовую графічний пакет swing, та пакет data, який організує збереження на завантаження даних про проектування.

3.2. Алгоритмізація та програмування програмних модулів

Алгоритми збереження даних у вигляді блок-схеми (рис. 3.3).

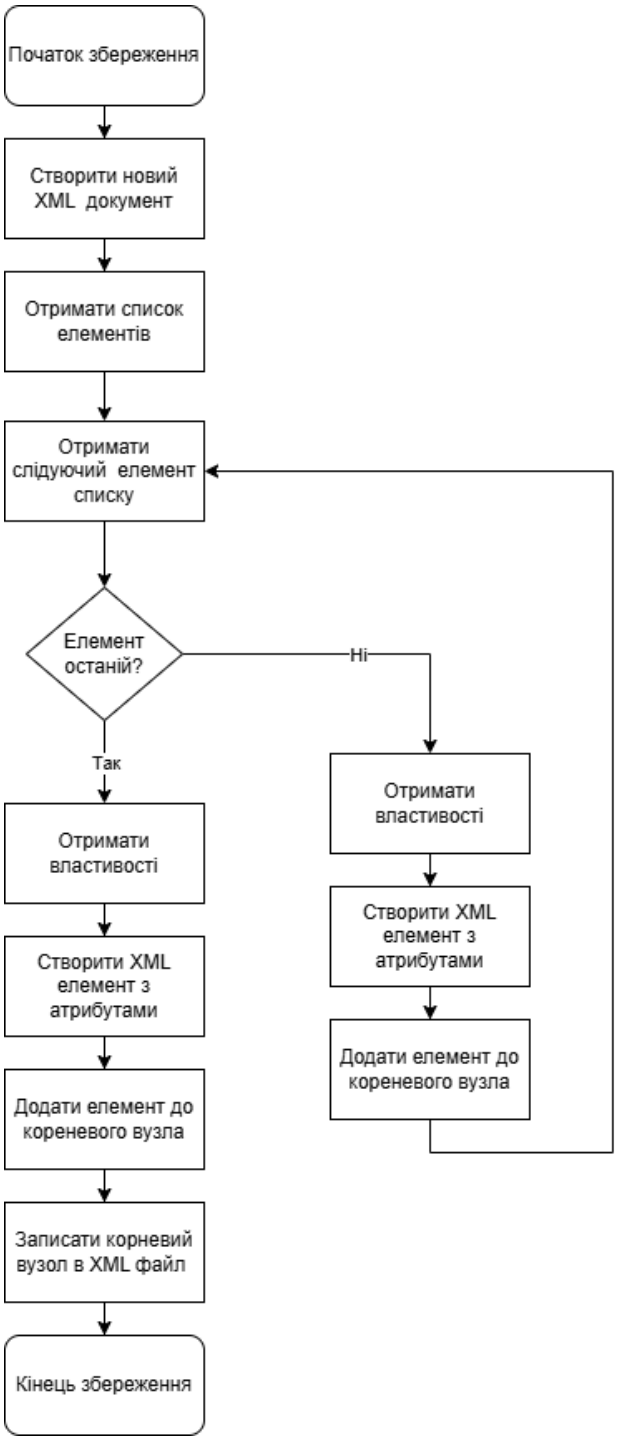

Рис. 3.3 - Алгоритми збереження даних

Сам код збереження наведений в (додаток А).

Алгоритми завантаження даних у вигляді блок-схеми (рис. 3.4).

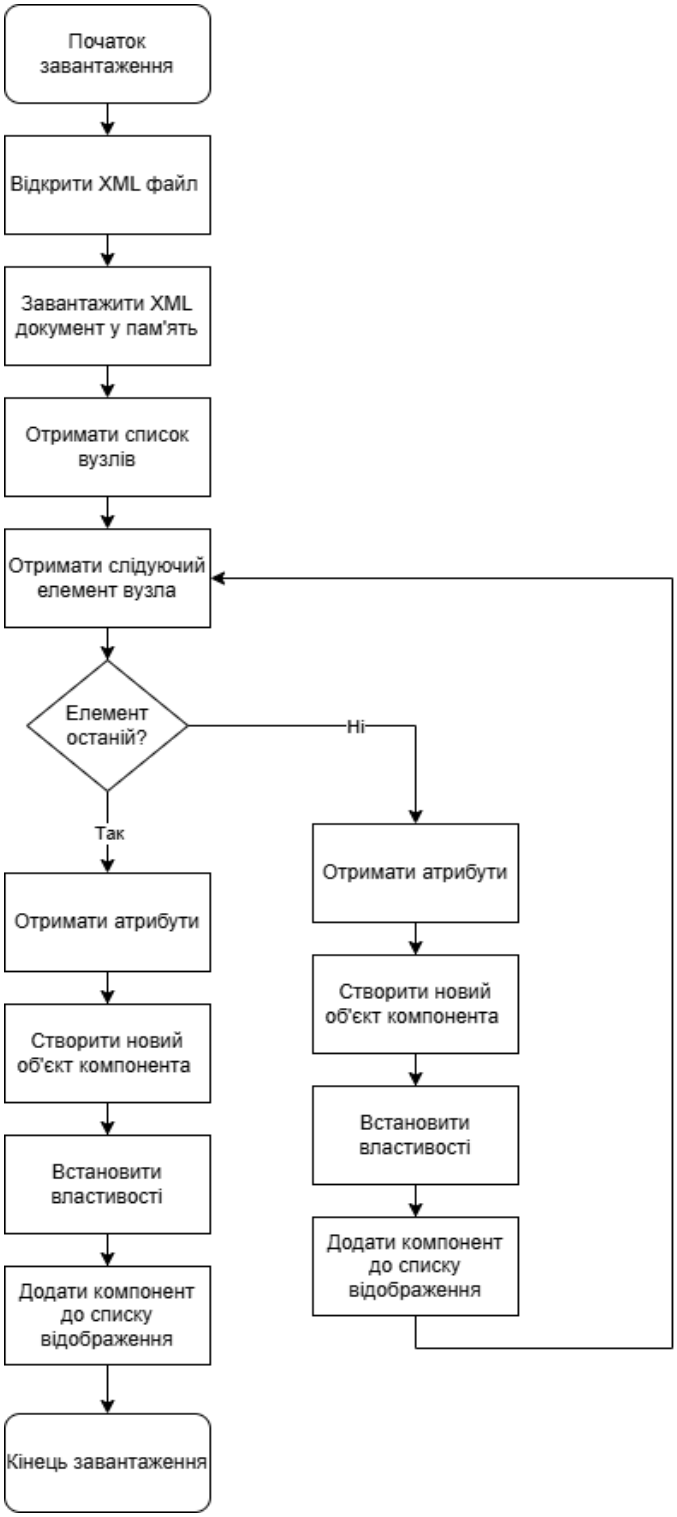

Рис. 3.4 - Алгоритми завантаження даних

Код завантаження проекту наведений в (додаток Д).

Далі наведено алгоритми відбору даних для формування звітно-статистичної інформації у вигляді блок-схеми (рис. 3.5).

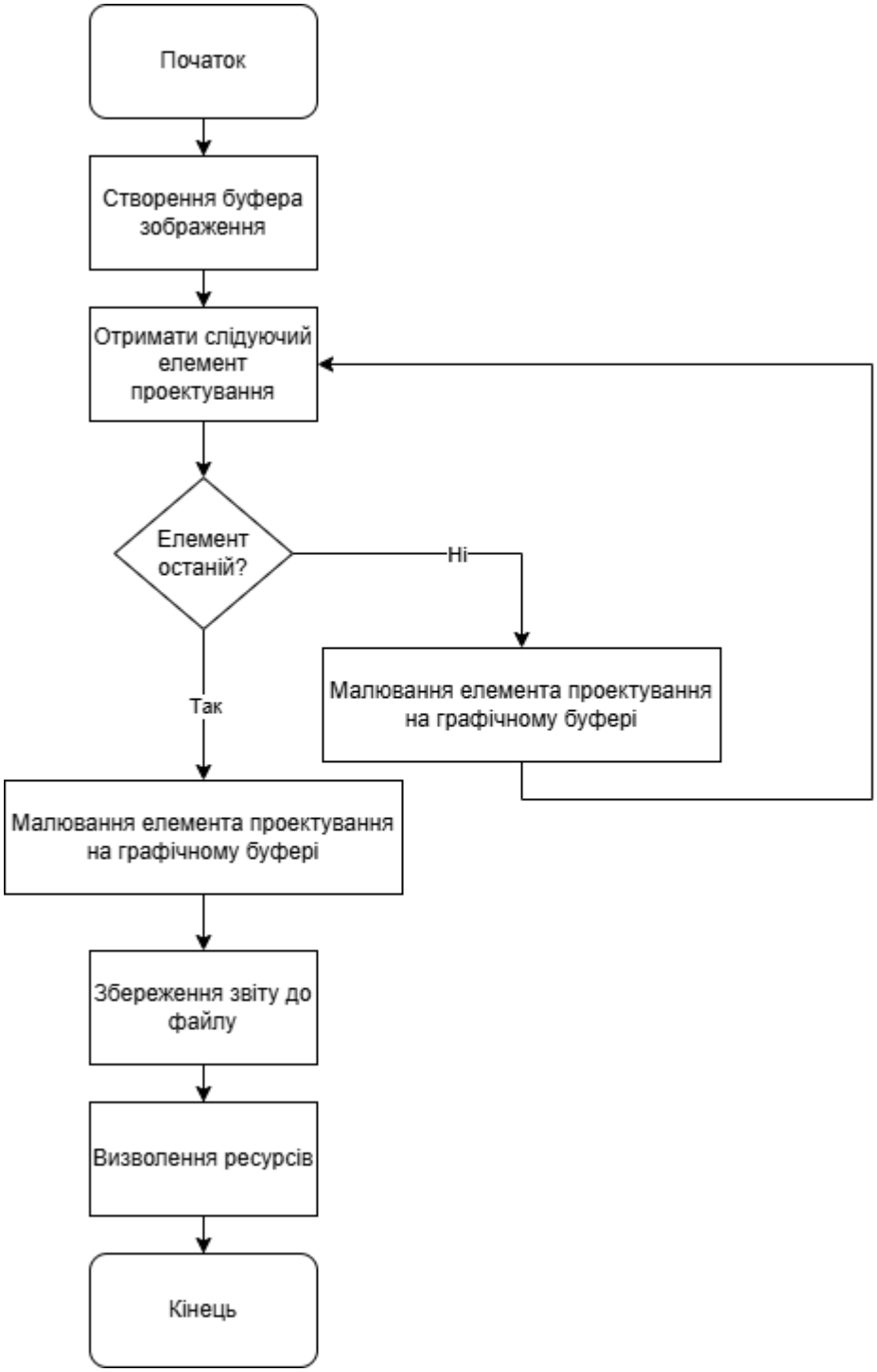

Рис. 3.5 - Алгоритми формування звітно-статистичної інформації

Код побудови звітів вказаний в (додаток Е).

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Тестування програмного забезпечення – техніка контролю якості, що перевіряє відповідність між реальною і очікуваною поведінкою програми завдяки кінцевому набору тестів, які обираються певним чином. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки[9, с. 11].

Тестування програмного забезпечення – це ключовий етап у розробці, який гарантує, що продукт відповідає очікуванням.

Оцінка може базуватися на відповідності вимогам розробників, коректності роботи з усіма вхідними даними, прийнятної швидкості виконання функцій, зручності використання, сумісності з ПЗ та операційними системами, а також відповідності потреб замовника.

Якість не є абсолютною, це суб’єктивне поняття. Тому тестування, як процес своєчасного виявлення помилок та дефектів, не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукта зі специфікацією. При цьому треба розрізняти тестування програмного забезпечення і забезпечення якості програмного забезпечення, до якого належать усі складові ділового процесу, а не тільки тестування[9, с. 11].

Якість – це суб’єктивна категорія, тому тестування лише порівнює продукт зі специфікацією.

Таким чином, можна виділити два крайніх випадки: у першому випадку ми володіємо абсолютно повною інформацією про систему, що тестується; у другому – система, що тестується, виступає в ролі «чорного ящика», тобто ми володіємо лише знанням про входи системи і можемо фіксувати одержувані виходи (результати внутрішньої активності системи)[9, с. 15].

Існує два підходи до тестування: аналіз "білої скриньки" (відома внутрішня структура) та "чорної скриньки" (відомі лише вхід/вихід).

При тестуванні системи було виявлено проблеми, наведені нижче.

Перша проблема полягає в тому, що при створенні проєкту в полях опису умовного позначення вказується шлях до файлу відповідно до операційної системи. Проблема виникає при переносі збережених даних між різними операційними системами. Код проблемної обробки надано нижче.

```
public void open_cgs(String filename) {

        File file = new File(filename);

        if (!file.exists())
                return;

        Conventional_Graphic_Symbols conventional_Graphic_Symbols = new
Conventional_Graphic_Symbols();
        try {


conventional_Graphic_Symbols.File_Load_Conventional_Graphic_Symbols(filenam
e);
        } catch (ParserConfigurationException | SAXException | IOException
e1) {
                e1.printStackTrace();
                conventional_Graphic_Symbols = null;
    }
```

Для того щоб виправити помилкове завантаження, потрібно замінити аргумент у методі File_Load_Conventional_Graphic_Symbols з filename на file.toString()
Проблема друга полягає в тому, що на різних екранах виділення працює не коректно (рис. 4.1).

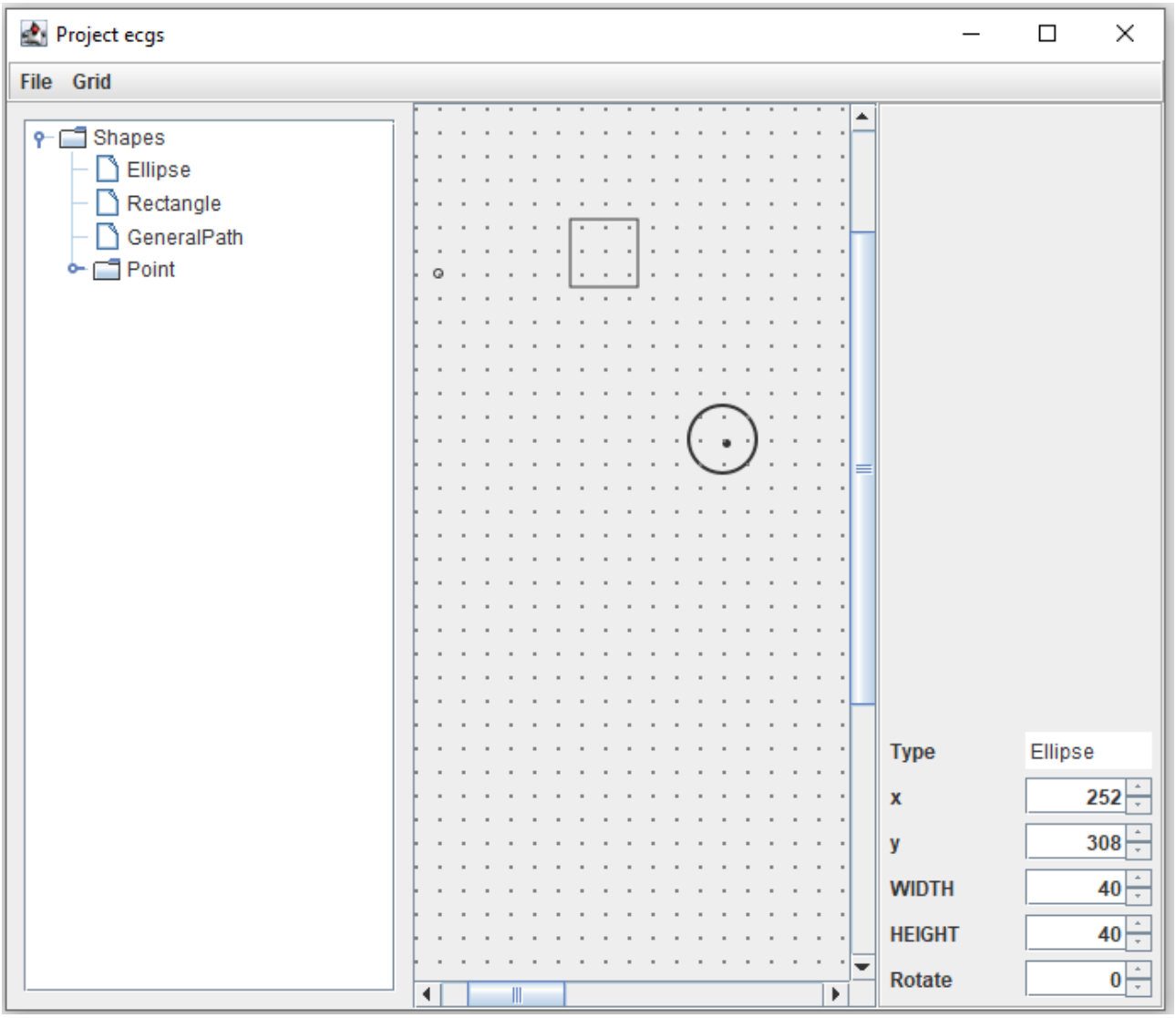

Рис. 4.1 - Помилкове відображення виділення фігури

Проблемна частина коду.

```
public void recreate_select_area(Graphics2D g2d) {

    Rectangle2D rectangle = shape2d.getBounds2D();
```

```
            AffineTransform _transform = g2d.getTransform();
            Shape rotatedShape = _transform.createTransformedShape(shape2d);
            rectangle = rotatedShape.getBounds2D();
            select_area = rectangle;
            _design.repaint();
 }
```

Вирішення полягає у зміні створення _transform.

```
            AffineTransform _transform = new AffineTransform();
      _transform.rotate(Math.toRadians(rotate), rectangle.getCenterX(),
rectangle.getCenterY());
```

Проблема третя полягає в тому, що область виділення фігур працює не коректно (рис. 4.2).

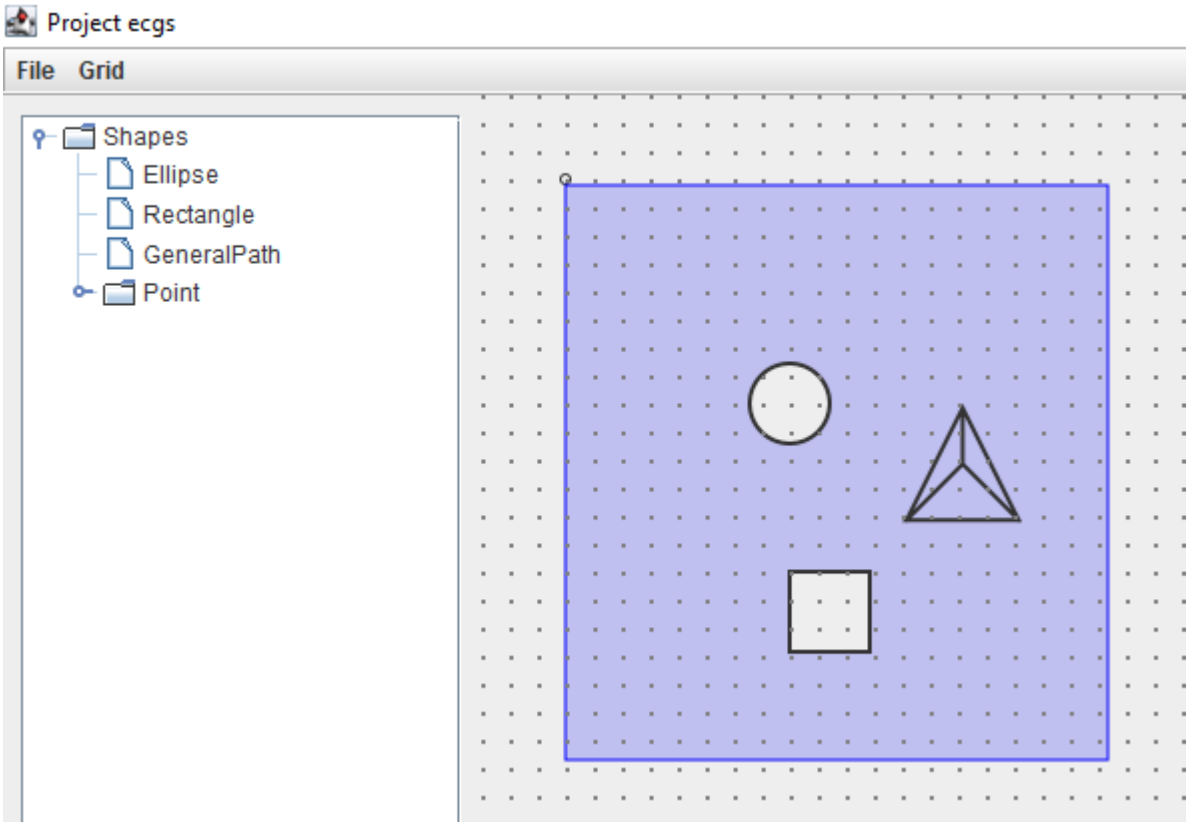

Рис. 4.2 - Помилкова область виділення

Вирішення полягає у зміні порядку малювання виділення в останню чергу.

4.2 Вимоги до апаратного та програмного забезпечення

Побудовано діаграму розміщення, на якій видно, що для повноцінної роботи програми потрібні дві папки, які містять елементи проектування або самі проекти (рис. 4.3).

A

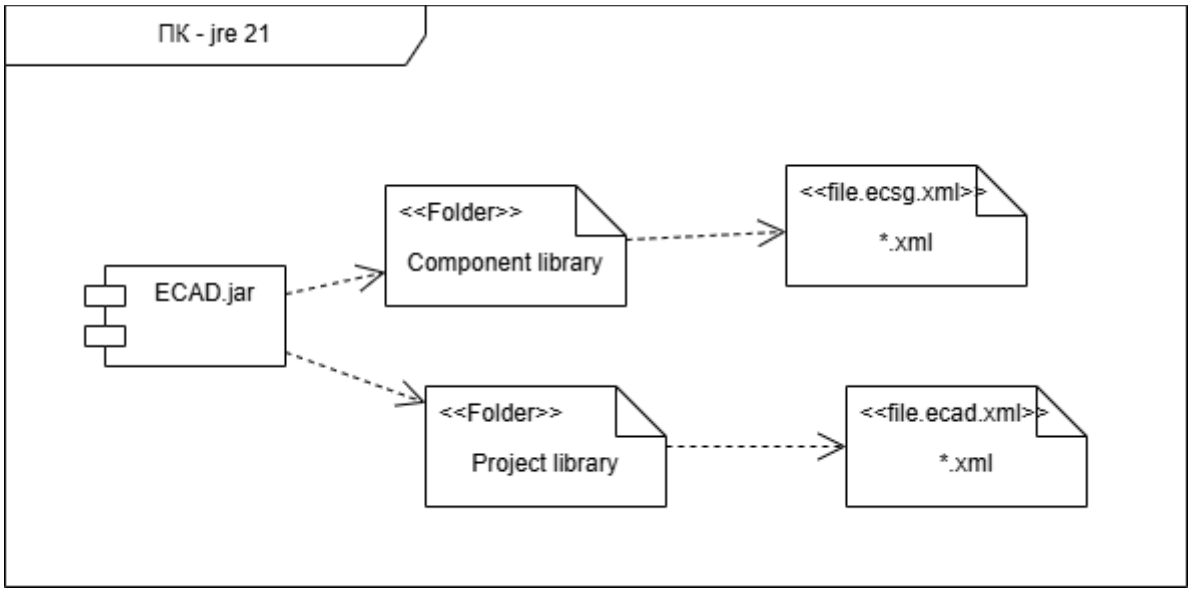

Рис. 4.3 - Діаграма розміщення

для запуску програми потрібно java версії Java Runtime Environment (JRE) 21. Продемонстровано важливі моменти роботи програмної системи у вигляді моментальних знімків.

Якщо папка Component library пуста, то проектувати електроні компоненти не вийде (рис. 4.4).

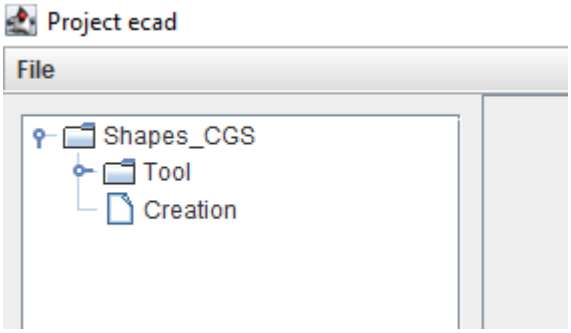

Рис. 4.4 - Вигляд відсутніх компонентів

Для того щоб проектувати електроні компоненти, потрібно створити умовні графічні позначення (рис. 4.5).

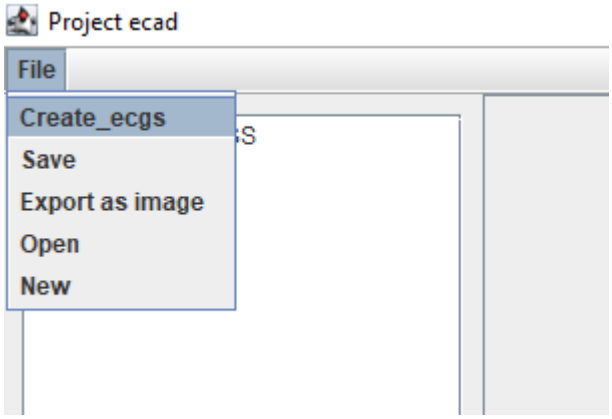

Рис. 4.5 - Створення умовного позначення

Та зберегти умовне позначення у .папку Component library (рис. 4.6).

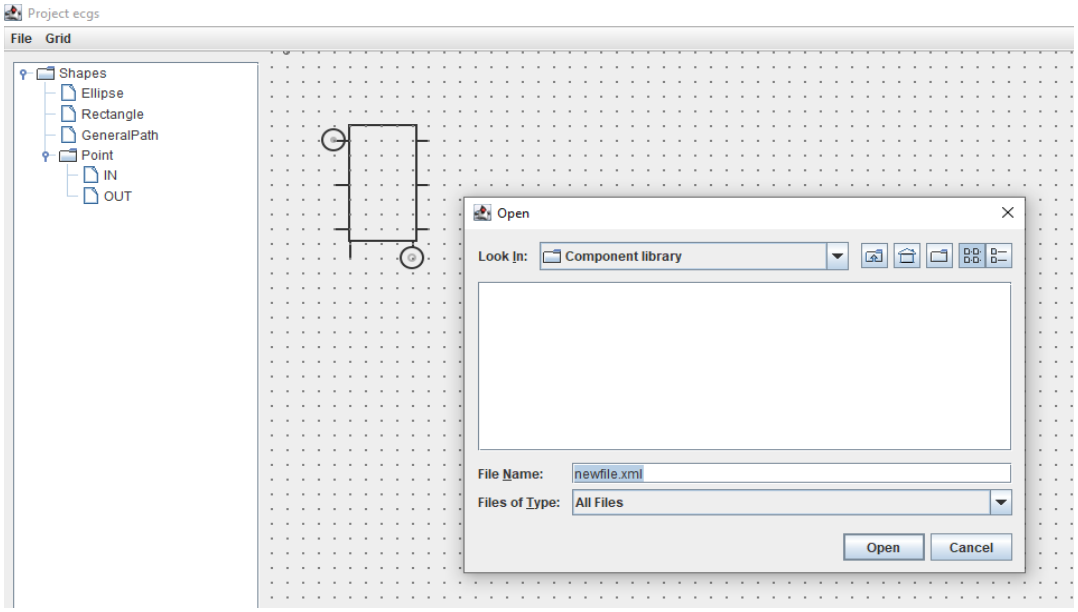

Рис. 4.6 - Збереження умовного позначення

Далі потрібно повернутись до проекту та натиснути кнопку оновлення (рис. 4.7).

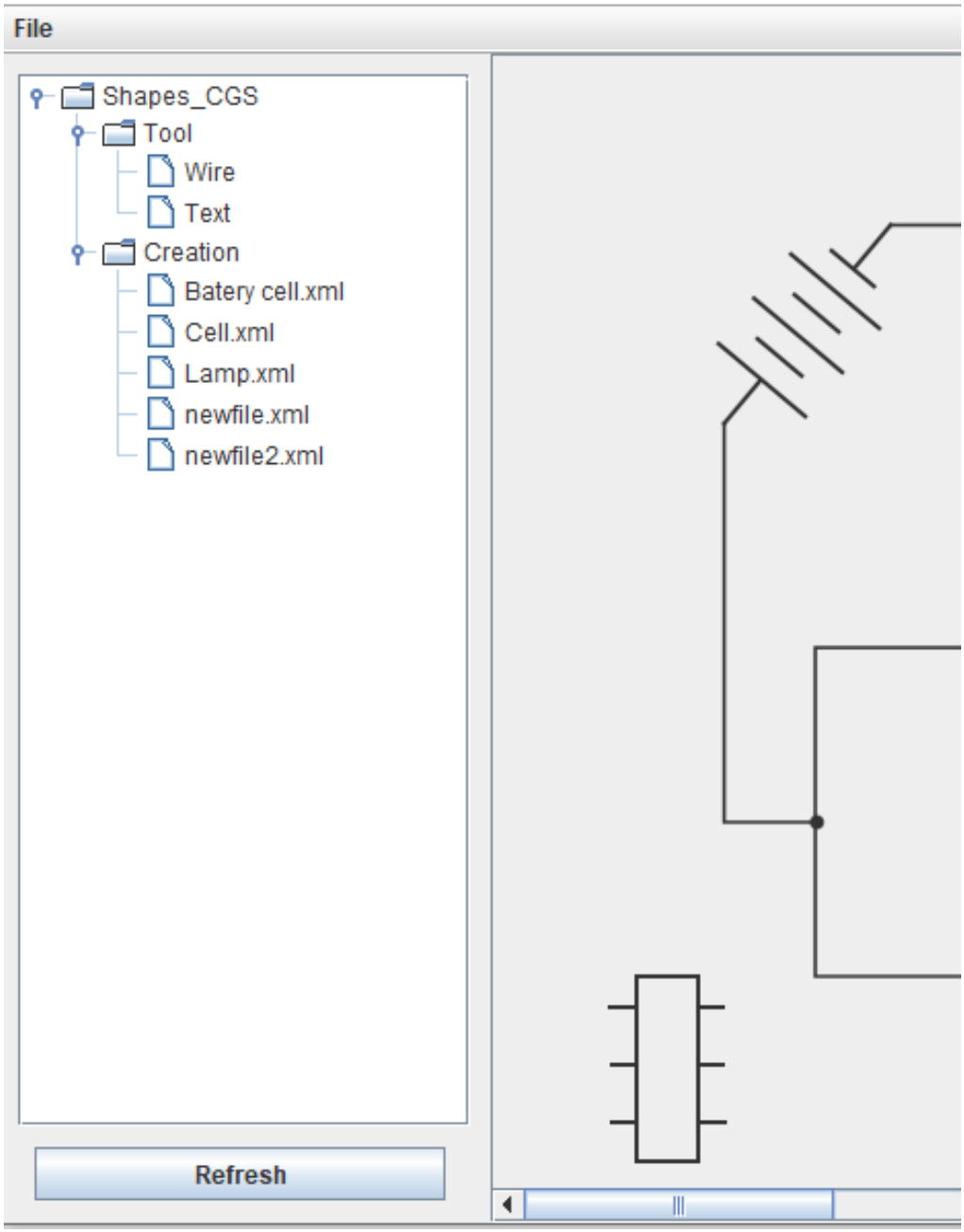

Рис. 4.7 - Заповнена панель умовних позначень

Визначено апаратні та програмні вимоги до роботи системи.

Програмні вимоги.

Версія Java Runtime Environment (JRE) 21. JRE постачається разом з інсталяційним пакетом.

Операційна система: Windows, Linux.

Апаратні вимоги.

Процесор:

Мінімально: Двоядерний процесор з тактовою частотою 2 ГГц.

Рекомендовано: Чотириядерний процесор з тактовою частотою 3 ГГц або вище.

Оперативна пам'ять (RAM):

Мінімально: 1 ГБ.

Рекомендовано: 4 ГБ або більше.

Вільне місце на диску:

Мінімально: 1 ГБ.

Рекомендовано: 2 ГБ або більше.

Відеокарта:

Мінімально: Вбудована відеокарта з підтримкою OpenGL 2.0.

Рекомендовано: Дискретна відеокарта з підтримкою OpenGL 3.0 або вище.

Пристрої керування:

Мишка та клавіатура.

4.3 Склад інсталяційного пакету

Інсталяційний пакет містить скрипт запуску програми за допомогою jre (рис. 4.8).

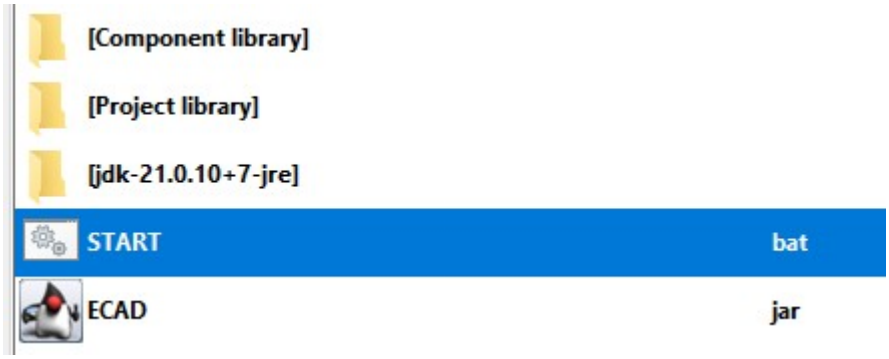

Рис. 4.8 - інсталяційний пакет

Для запуску файл START.bat виконує команду, наведену та поясненну далі.

Команда .\jdk-21.0.10+7-jre\bin\java -jar ECAD.jar складається з декількох компонентів:

Шлях до виконуваного файлу Java

.\jdk-21.0.10+7-jre\bin\java - це шлях до Java Runtime Environment (JRE):

.\ - поточна директорія (відносний шлях)

jdk-21.0.10+7-jre - папка з встановленою версією Java Development Kit версії 21.0.10

bin — підпапка, що містить виконувані файли Java

java - основний виконуваний файл, який запускає Java Virtual Machine (JVM)

Опція запуску -jar - аргумент, який вказує JVM на те, що потрібно запустити JAR-файл (Java Archive). Наступний аргумент - це JAR-архів.

ВИСНОВКИ

Постановка завдання включає детальний опис інформації, яку необхідно оброблювати і зберігати, операцій, які необхідно автоматизувати.

Для розділу огляд інформаційних джерел та існуючих рішень проведено аудит систем: Altium Designer, KiCad, EAGLE за схемою переваги та недоліки існуючих рішень.

У розділі моделювання предметної області розроблено комплекс моделей, які моделюють предметну область системи програмного застосунку проєктування електронних компонентів.

У розділі інформаційне забезпечення Було створено логічну модель даних, обґрунтовано вибір системи управління інформаційною базою та реалізовано її структуру.

У розділі організаційна структура програмного забезпечення розглянуто організаційна структура ППЗ у вигляді діаграми пакетів та діаграми розгортання.

Для розділу вибір інструментарію для створення ППЗ обґрунтовано вибір мови програмування java, середовища розробки eclipse, бібліотек графіки, xml.

У розділі алгоритмізація та програмування програмних модулів показано алгоритми для модулів збереження та завантаження xml-документів, а також програмного модуля формування звітно-статистичної інформації. Ці алгоритми представлені у вигляді блок-схем. Продемонстровано окремі частини коду.

У розділі тестуванні системи визначено, що таке тестування та як воно відбувається, проілюстровано процес проведення тестування розробленої системи та виправлення проблем із відображенням компонентів, збереженням їх на різних системах.

У розділі вимоги до апаратного та програмного забезпечення побудовано діаграму розміщення системи, на якій визначено вузли системи та

місцезнаходження компонентів системи відносно вузлів. Відповідно представленій топології вказано вимоги до апаратного, програмного забезпечення.

У розділі склад інсталяційного пакету показано, що інсталяційний пакет містить скрипт запуску програми, папку компонентів та проектів, JRE.

У результаті виконаної бакалаврської кваліфікаційної роботи було створено програмний застосунок для проєктування електронних компонентів, який відповідає поставленим завданням та забезпечує автоматизацію ключових процесів збереження, завантаження і обробки даних.

Проведений аналіз сучасних аналогів, таких як Altium Designer, KiCad та EAGLE, показав, що розроблена система має менший набір функцій. Система відрізняється простотою використання.

З техніко-економічної точки зору застосунок є ефективним завдяки використанню безкоштовних інструментів розробки, невисоким вимогам до апаратного забезпечення та простоті інсталяції. Інструмент придатний для малювання електросхем.

Негативним результатом є обмежений функціонал системи та відсутність інтеграції з зовнішніми бібліотеками компонентів, що знижує її конкурентоспроможність у порівнянні з професійними рішеннями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Eclipse Platform Overview [Електронний ресурс]. – Режим доступу: https://help.eclipse.org/2026-03/index.jsp (дата звернення: 10.04.2026).
2. Lesson: Document Object Model (The Java™ Tutorials > Java API for XML Processing (JAXP)) [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/javase/tutorial/jaxp/dom/index.html (дата звернення: 10.04.2026).
3. Lesson: Getting Started with Graphics (The Java™ Tutorials > 2D Graphics) [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/javase/tutorial/2d/basic2d/index.html (дата звернення: 10.04.2026).
4. Lesson: Overview of the Java 2D API Concepts (The Java™ Tutorials > 2D Graphics) [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/javase/tutorial/2d/overview/index.html (дата звернення: 10.04.2026).
5. Lesson: Using Swing Components (The Java™ Tutorials > Creating a GUI With Swing) [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/javase/tutorial/uiswing/components/index.html (дата звернення: 10.04.2026).
6. Trail: 2D Graphics (The Java™ Tutorials) [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/javase/tutorial/2d/index.html (дата звернення: 10.04.2026).
7. Trail: Creating a GUI With Swing (The Java™ Tutorials) [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/javase/tutorial/uiswing/ (дата звернення: 10.04.2026).

8. What is Java technology and why do I need it? [Електронний ресурс]. – Режим доступу: https://www.java.com/download/help/whatis_java.html (дата звернення: 10.04.2026).
9. Авраменко А. С., Авраменко В. С., Косенюк Г. В. Тестування програмного забезпечення: навч. Посіб. – 2017.
10. Галаган Р. М. Комп'ютерне проєктування електронних схем //Комп’ютерний практикум: навчальний посібник. Київ: КПІ ім. Ігоря Сікорського. – 2023.
11. ДСТУ 2226-93 Автоматизовані системи. Терміни і визначення.
12. ДСТУ 3321:2003 Система конструкторської документації. Терміни та визначення основних понять.
13. Що таке електронні компоненти? Типи, символи, тестування та нові тенденції [Електронний ресурс]. – Режим доступу: https://ua.y-ic.com/blog/understand-basic-electronic-components-resistors,capacitors,diodes,transistors,inductors,and-digital.html (дата звернення: 10.04.2026).

Додаток А

Метод збереження для Conventional_Graphic_Symbols

```
public void Conventional_Graphic_Symbols_Save_File(String name)
		throws ParserConfigurationException, TransformerException {
	DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
	DocumentBuilder builder = null;
	builder = factory.newDocumentBuilder();
	Document document = builder.newDocument();
	Element shapes = document.createElement(ShapeCGS._Node_Container);
	Element connectors = document.createElement(Connector._Node_Container);
	Element root = this.save(document);

	if (ShapeCGS_List != null)
		for (ShapeCGS component : ShapeCGS_List) {
			shapes.appendChild(component.save(document));
		}

	if (Connertor_List != null)
		for (Connector connertor : Connertor_List) {
			connectors.appendChild(connertor.save(document));
		}

	root.appendChild(shapes);
	root.appendChild(connectors);
	TransformerFactory transformerFactory = TransformerFactory.newInstance();
	Transformer transformer = null;

	transformer = transformerFactory.newTransformer();
	DOMSource source = new DOMSource(document);
	StreamResult result = new StreamResult(name);
	transformer.setOutputProperty(OutputKeys.INDENT, "yes");
	transformer.transform(source, result);

	}

public Element save(Document document) {
	Element root = document.createElement(_Node_Name);
	document.appendChild(root);
	Element Name =
document.createElement(OBJECT_FIELDS_Conventional_Graphic_Symbols.Name.toString());
	Name.appendChild(document.createTextNode(this.Name));
	Element Description =
document.createElement(OBJECT_FIELDS_Conventional_Graphic_Symbols.Description.toString());
	Description.appendChild(document.createTextNode(this.Description));
	Element Data =
document.createElement(OBJECT_FIELDS_Conventional_Graphic_Symbols.Data.toString());
	Data.appendChild(document.createTextNode(this.Data));
	root.appendChild(Name);
	root.appendChild(Description);
	root.appendChild(Data);

	return root;

	}
```

Додаток Б

Метод збереження для ShapeCGS

```
public Element save(Document document) {
	Element root = document.createElement(_Node_Name);

	Map<OBJECT_FIELDS_ShapeCGS_Attribute, String> field_attribute = new LinkedHashMap<>();
	field_attribute.put(OBJECT_FIELDS_ShapeCGS_Attribute.id, Integer.toString(this.id));
	field_attribute.put(OBJECT_FIELDS_ShapeCGS_Attribute.rotate, Integer.toString(this.rotate));
	field_attribute.put(OBJECT_FIELDS_ShapeCGS_Attribute.height, Integer.toString(this.height));
	field_attribute.put(OBJECT_FIELDS_ShapeCGS_Attribute.width, Integer.toString(this.width));
	field_attribute.put(OBJECT_FIELDS_ShapeCGS_Attribute.scale_x, Float.toString(this.scale_x));
	field_attribute.put(OBJECT_FIELDS_ShapeCGS_Attribute.scale_y, Float.toString(this.scale_y));
	field_attribute.put(OBJECT_FIELDS_ShapeCGS_Attribute.BASICSTROKE_WIDTH,
Float.toString(this.basicstroke_width));

	field_attribute.forEach((field, value) -> {
		root.setAttribute(field.toString(), value);
	});

	Map<OBJECT_FIELDS_ShapeCGS, String> fields = new LinkedHashMap<>();
	fields.put(OBJECT_FIELDS_ShapeCGS.X, String.valueOf(this.X));
	fields.put(OBJECT_FIELDS_ShapeCGS.Y, String.valueOf(this.Y));
	fields.put(OBJECT_FIELDS_ShapeCGS.Type, this.Type);
	fields.put(OBJECT_FIELDS_ShapeCGS.Description, this.Description);

	fields.forEach((field, value) -> {
		Element element = document.createElement(field.toString());
		element.setTextContent(value);
		root.appendChild(element);
	});

	Element element_point_path = document.createElement("Point_paths");

	if (_Path != null)
		for (Point_path point_path : _Path) {
			element_point_path.appendChild(this.savePointPath(document, point_path));
		}

	root.appendChild(element_point_path);

	return root;
	}
```

Додаток В

Метод збереження для Connertor

```
public Element save(Document document) {
        Element root = document.createElement(_Node_Name);

        Map<OBJECT_FIELDS_ShapeCGS_Attribute, String> field_attribute = new LinkedHashMap<>();
        field_attribute.put(OBJECT_FIELDS_ShapeCGS_Attribute.id, Integer.toString(this.id));

        field_attribute.forEach((field, value) -> {
                root.setAttribute(field.toString(), value);
        });

        Map<OBJECT_FIELDS_Connertor, String> fields = new LinkedHashMap<>();
        fields.put(OBJECT_FIELDS_Connertor.X, String.valueOf(this.X));
        fields.put(OBJECT_FIELDS_Connertor.Y, String.valueOf(this.Y));
        fields.put(OBJECT_FIELDS_Connertor.Type, this.Type);

        fields.forEach((field, value) -> {
                Element element = document.createElement(field.toString());
                element.setTextContent(value);
                root.appendChild(element);
        });
        return root;
        }
```

Додаток Д

Код завантаження проекту

```
	public void File_Load_Project(String File_name) throws ParserConfigurationException, SAXException,
IOException {

		File xmlFile = new File(File_name);
		DocumentBuilder builder = null;
		builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();

		Document document = null;
		document = builder.parse(xmlFile);
		document.getDocumentElement().normalize();

		Boolean result = this.Load(document);

		if (result) {
			Component_List = Component.LoadArray(document);
			Connection_List = Connection.LoadArray(document);
		}
	}

	public Boolean Load(Document document) {
		try {
			XPath xpath = XPathFactory.newInstance().newXPath();
			Node first = (Node) xpath.evaluate("//" + _Node_Name, document, XPathConstants.NODE);

			if (first == null) {
				System.err.println("Warning: '" + _Node_Name + "' node not found in document");
				return false;
			}

			Name = getXPathValue(xpath, first, OBJECT_FIELDS_Project.Name.toString());
			Description = getXPathValue(xpath, first,
					OBJECT_FIELDS_Project.Description.toString());
			Data = getXPathValue(xpath, first, OBJECT_FIELDS_Project.Data.toString());

			return true;
		} catch (XPathExpressionException e) {
			System.err.println("Error parsing XML: " + e.getMessage());
			return false;
		}
	}

	public static ArrayList<Component> LoadArray(Document document) {
		ArrayList<Component> list = new ArrayList<>();

		try {
			XPath xpath = XPathFactory.newInstance().newXPath();
			NodeList nodeList = (NodeList) xpath.evaluate("//" + _Node_Container + "/*", document,
					XPathConstants.NODESET);

			for (int i = 0; i < nodeList.getLength(); i++) {
				Node node = nodeList.item(i);

				NamedNodeMap attribute = node.getAttributes();
```

```
                                int _id = ungetAttributeDef(attribute,
OBJECT_FIELDS_Component_Attribute.id.toString(), -1,
                                                Integer::parseInt);
                                int _rotate = ungetAttributeDef(attribute,
OBJECT_FIELDS_Component_Attribute.rotate.toString(), -1,
                                                Integer::parseInt);

                                String _area_text = ungetAttributeDef(attribute,
OBJECT_FIELDS_Component_Attribute.area_text.toString(), "none",
                                                String::toString);

                                int _size_text = ungetAttributeDef(attribute,
OBJECT_FIELDS_Component_Attribute.size_text.toString(), 24,
                                                Integer::parseInt);

                                double x =
java.lang.Double.parseDouble(xpath.evaluate(OBJECT_FIELDS_ShapeCGS.X + "/text()", node));
                                double y =
java.lang.Double.parseDouble(xpath.evaluate(OBJECT_FIELDS_ShapeCGS.Y + "/text()", node));
                                String type = xpath.evaluate(OBJECT_FIELDS_Component.Name + "/text()",
node);
                                String description = xpath.evaluate(OBJECT_FIELDS_Component.Comment +
"/text()", node);

                                OBJECT_FIELDS_Component_type Type_area =
OBJECT_FIELDS_Component_type
                                                .valueOf(xpath.evaluate(OBJECT_FIELDS_Component.Type_are
a + "/text()", node));

                                switch (Type_area) {
                                case Area_shape:
                                        list.add(new Component(_id, x, y, _rotate, type, description));
                                        break;
                                case Area_text:
                                        list.add(new Component(x, y, _rotate, _area_text, _size_text));
                                        break;

                                default:
                                        break;
                                }

                        }
                } catch (XPathExpressionException e) {
                        e.printStackTrace();
                }

                return list;
        }

        public static ArrayList<Connection> LoadArray(Document document) {
                ArrayList<Connection> list = new ArrayList<>();
                try {
                        XPath xpath = XPathFactory.newInstance().newXPath();
                        NodeList nodeList = (NodeList) xpath.evaluate("//" + _Node_Container + "/*", document,
                                        XPathConstants.NODESET);

                        for (int i = 0; i < nodeList.getLength(); i++) {
```

```
                                        Node node = nodeList.item(i);
                                        NamedNodeMap attribute = node.getAttributes();
                                        int _id = ungetAttributeDef(attribute,
OBJECT_FIELDS_Connection_Attribute.id.toString(), -1,
                                                        Integer::parseInt);

                                        int _Next_connection_id = ungetAttributeDef(attribute,

        OBJECT_FIELDS_Connection_Attribute.Next_connection_id.toString(), -1, Integer::parseInt);
                                        int _Previous_connection_id = ungetAttributeDef(attribute,

        OBJECT_FIELDS_Connection_Attribute.Previous_connection_id.toString(), -1, Integer::parseInt);
                                        int _in_connector_id = ungetAttributeDef(attribute,

        OBJECT_FIELDS_Connection_Attribute.in_connector_id.toString(), -1, Integer::parseInt);
                                        int _out_connector_id = ungetAttributeDef(attribute,

        OBJECT_FIELDS_Connection_Attribute.out_connector_id.toString(), -1, Integer::parseInt);
                                        int _wire_connection_id = ungetAttributeDef(attribute,

        OBJECT_FIELDS_Connection_Attribute.wire_connection_id.toString(), -1, Integer::parseInt);

                                        List<Wire_Point> _Path_ = new ArrayList<>();

                                        NodeList nodeList_path = (NodeList) xpath.evaluate("Point_paths/PointPath", node,
                                                        XPathConstants.NODESET);

                                        for (int i1 = 0; i1 < nodeList_path.getLength(); i1++) {
                                                Node node1 = nodeList_path.item(i1);
                                                if (node1.getNodeType() == Node.ELEMENT_NODE) {
                                                        Element element = (Element) node1;
                                                        _Path_.add(loadPointPath_wire(element));
                                                }
                                        }
                                        list.add(new Connection(_id, _Next_connection_id, _Previous_connection_id,
_in_connector_id,
                                                        _out_connector_id, _wire_connection_id, _Path_));
                                }

                        } catch (XPathExpressionException e) {
                                e.printStackTrace();
                        }
                        return list;
 }
        public static <T> T ungetAttributeDef(NamedNodeMap attributes, String attributeName, T defaultValue,
Function<String, T> parser) {
            String strValue = null;
            try {
               strValue = attributes.getNamedItem(attributeName).getTextContent();
            } catch (NullPointerException e) {
               return defaultValue;
            }
            if (strValue == null || strValue.isEmpty()) {
               return defaultValue;
            }
            return parser.apply(strValue);
 }
```

Додаток Е

Код побудови звітів

```
public void draw_element(Graphics2D g2d) {
	for (Wire wire : Wire_List) {
		wire.draw_wire(g2d, design);
	}
	for (Area_shape area : area_shape_List) {
		if (area.select_area == null)
			area.create_area();
		if (area != null && area.area != null) {
			area.draw_area(g2d, this);
		}
	}
	for (Area_text area_text : Area_text_List) {
		area_text.draw_area_text(g2d, this);
	}
}
```

```
		case EXPORT_AS_IMAGE:
			Graphics2D bufferGraphics = designArea.clearImage();

			designArea.draw_element(bufferGraphics);

			try {
				File output = new File(select_file.toString());
				ImageIO.write(designArea.buffer, "png", output);

			} catch (IOException e) {
				e.printStackTrace();
			}
break;
```

Додаток Ж

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Казанецький Василь Олегович здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Програмний застосунок проєктування електронних компонентів» є результатом моєї особистої праці.

Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

[ ] інструменти генеративного ШІ не використовувалися.

[+] інструменти ШІ Claude, DeepL були використані для:

[+] перекладу іншомовних джерел;

[+] стилістичного редагування та перевірки граматики;

[+] допомоги у написанні/відлагодженні програмного коду;

[ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Василь Казанецький**
(підпис)