УДК 681.3.07
Я.І. Соколовський[1,2], О.П. Сикала[1], В.Я. Семенюк[2], А.П. Здолбіцький[2]
[1]Національний лісотехнічний університет України,
[2]Луцький національний технічний університет

ПРОЕКТУВАННЯ CASE-ЗАСОБАМИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АВТОМАТИЗОВАНОГО ЗБИРАННЯ ДАНИХ У ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ

У роботі з використанням CASE-технології (Computer Aided Software Engineering) здійснено проектування програмної системи автоматизованого збирання даних. Для створення моделі використано середовище Rational Rose, яке надає можливість проектувати системи за допомогою уніфікованої мови моделювання UML. Структуру системи подано у вигляді діаграм та у словесній формі. Для автоматичного генерування програмного коду обрано об'єктно-орієнтовану мову програмування JAVA.

Ключові слова: *CASE-технології, Rational Rose, проектування, інформаційна система, UML-діаграми, процес.*

Аналіз останніх досліджень і публікацій. Інтенсивний розвиток комп'ютерних та інформаційних технологій створення програмних систем тісно пов'язаний з використанням методів об'єктно-орієнтованого аналізу та CASE-засобів для автоматизації процесів аналізу і проектування програмного забезпечення.

Починаючи з середини 70-х років минулого сторіччя і до недавнього часу для проектування програмних систем широко використовувалася структурна методологія. В її основі покладено принцип функціональної або алгоритмічної декомпозиції, за якої структура системи описується в ієрархії її функцій та передавання інформації між окремими функціональними елементами, а поведінка системи – у термінах послідовності виконання функцій. Ця технологія дозволяє штучно розбивати системи на підсистеми, а також встановлювати слабкий взаємозв'язок процесів і даних, які описують саму систему.

Інший об'єктно-орієнтований підхід аналізу та проектування програмних систем використовує декомпозицію, для якої структура системи описується у термінах об'єктів і зв'язків між ними, а поведінка системи описується у термінах обміну повідомленнями між об'єктами. Незважаючи на явну перевагу об'єктно-орієнтованих технологій, їх поширення у 70-х – початок 90-х років минулого сторіччя було незначним, оскільки кожний метод добре описував одну або декілька сторін реальної системи, залишаючи поза увагою множину інших, не менш важливих сторін. Окрім цього відсутність єдиного стандарту й автоматизованих засобів розробки не сприяли широкому поширенню об'єктно-орієнтованих методів для проектування програмного забезпечення.

Тому існувала об'єктивна реальність появи автоматизованих програмних засобів для проектування інформаційних систем. Ними стали засоби, які реалізують CASE-технологію створення та супроводження інформаційно-програмних систем. Для опису, візуалізації і документування об'єктно-орієнтованих систем з орієнтуванням на використання CASE-технологій для розроблення програмного забезпечення використовується UML (Unified Modeling Language).

Першочерговими областями застосування CASE-засобів були проектування баз даних, особливо тих, які вимагали значних зусиль для розробки своїх концептуальних схем. Реалізація можливостей автоматичного генерування програмного забезпечення на основі розроблених концептуальних схем виявилась настільки конструктивною, що поняття CASE на теперішній час охоплює процес розробки складних інформаційних систем уцілому. Тепер під терміном CASE розуміють програмні засоби, які підтримують процеси створення і супроводження інформаційних систем, включаючи аналіз та формулювання вимог, проектування прикладного програмного забезпечення і баз даних, генерування коду, тестування, документування, конфігураційне керування, та керування проектом уцілому. Тобто, CASE-засоби разом із системними засобами створюють повне середовище розробки інформаційних систем.

Постановка проблеми. За допомогою Case-засобів необхідно розробити програмне забезпечення для інформаційної системи збирання даних, яка б забезпечувала виведення наступних параметрів для автоматичного моніторингу технологічного процесу: швидкість потоку повітря, температуру, барометричний тиск, вологість повітря.

Система також повинна вираховувати деякі похідні параметри, у число яких входять: коефіцієнт жорсткості, точка роси, відносна зміна температури, відносна зміна барометричного тиску. В інформаційній системі повинно бути передбачено можливість визначення поточного часу та дати, які будуть використовуватись при генерації повідомлень про максимальне і мінімальне значення первинних показників за останні 24 години.

Інформаційна система повинна забезпечувати постійний вивід на дисплей всіх первинних і похідних параметрів, а також поточного часу та дати. Користувач повинен мати можливість побачити максимальне та мінімальне значення будь-якого з первинних параметрів.

Для вирішення даного завдання вибрано CASE-технологію Rational Rose[4] з наступним генеруванням програмного коду на JAVA. Моделювання предметної області здійснено у UML 2.0.

АНАЛІЗ системи за допомогою UML. Приступивши до безпосереднього проектування системи необхідно визначити, які основні абстракції фігуруватимуть у майбутній системі.

Робота автоматизованої системи базується на наступному: первинних даних, температури, атмосферного тиску і вологості, отриманих від давачів; усі дані прив'язані до дійсного часу. Користувач має можливість побачити на дисплеї будь-які дані.

Відповідальність за роботу системи було поділено між групами абстракцій: класи давачів для отримання і оброблення первинних даних (TemperatureSensor, PressureSensor, HumiditySensor, WindSpeedSensor, RainFallSensor, WindDirectionSensor, WindChill, DewPoint, TrendSensor, HistoricalSensor, CalibrationSensor, Sensor, Sensors), класи відповідальні за надання інформації про час та дату (TimeDate, Timer), абстракції для реалізації введення та виведення даних (InputManager, DisplayManager, LCDDevice, Keypad, Form) та керуючий клас системи (Sampler) для забезпечення взаємодії усіх механізмів.

Для прикладу, клас давача температури, змодельований на UML [2,3] з атрибутами і операція, має наступний вигляд:

TemperatureSensor
temperature
currentTemprature() setLowTemperature() setHightTemperature()

У загальному, після аналізу було розроблено 21 клас. Всі вони взаємодіють між собою. Зокрема, взаємодія абстракцій системи для виведення даних представлена на рис.1.

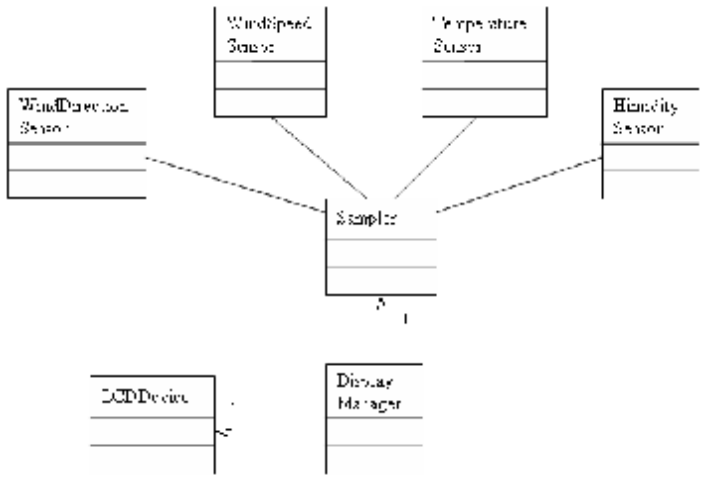

Рис. 1. Класи, відповідальні за виведення даних

Проектування системи з UML. Аналізуючи роботу системи, перш за все необхідно з'ясувати, як вона працюватиме з точки зору користувача, та які можливості йому надаватиме. Для цього слід розглянути сценарії поведінки системи. У Rational Rose ці можливості надає діаграма варіантів використання, яка була спроектована для даної системи (рис. 2).

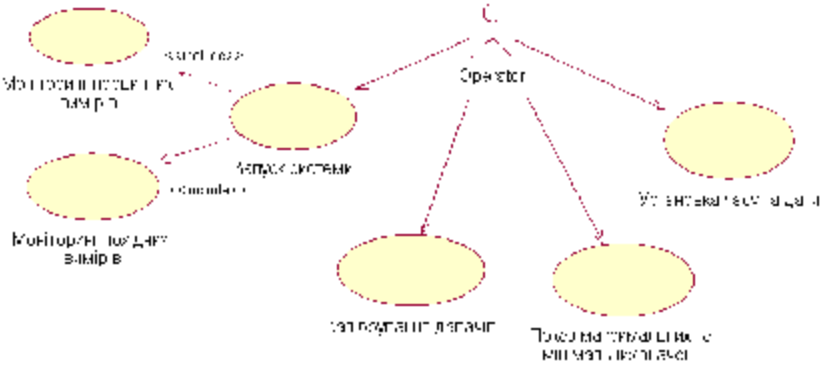

Рис. 2. Діаграма варіантів використання

Для подальшого проектування проведено опис кожного варіанту використання.

Наступним кроком у проектуванні системи є розробка її логічної моделі у вигляді діаграм класів і відношень між ними. Були розроблені чотири діаграми класів: ієрархії давачів, виведення даних, кадрової обробки, структури системи. У даній статті розміщена одна з чотирьох діаграм класів системи, а саме діаграма класів структури системи «StructureClassDiagram» (рис. 3)

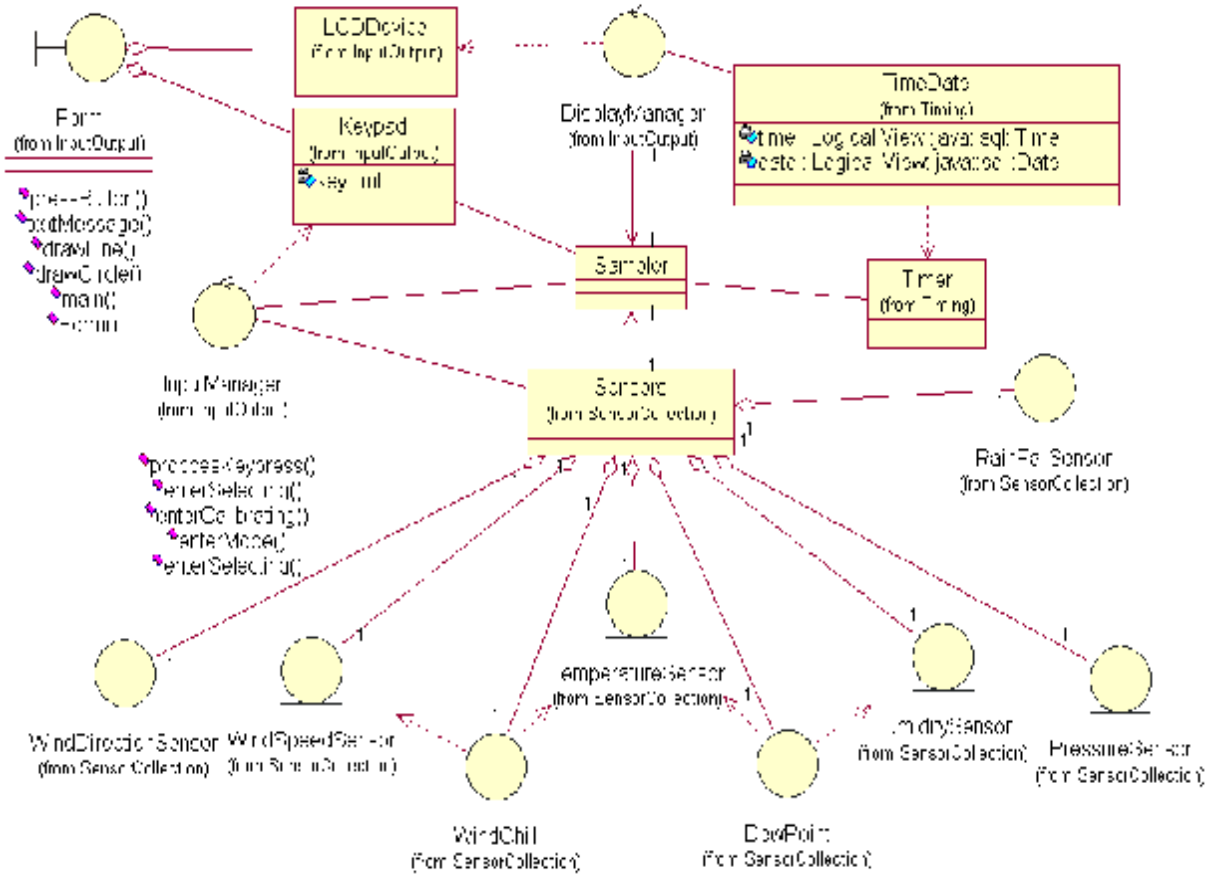

Рис. 3. Діаграма класів «StructureClassDiagram»

На діаграмі класів «StructureClassDiagram» показано контролюючий клас. Це InputManager, який відповідальний за введення даних. Цей клас використовує Keypad для введення даних, тобто між InputManager та Keypad існує зв'язок залежності. InputManager має двосторонні асоціації з класами Sampler та Sensors.

Так як дана система буде реалізуватися програмно, то доцільно виділити ще один клас – клас форми Form. Форма складатиметься з кнопок, які реалізовуватимуться класом Keypad, та графічного зображення, яке формуватиметься за допомогою класу LCDDevice. Отже Keypad та LCDDevice є частинами цілого, тому між ними та Form існує агрегація. Класу Form присвоєно стереотип граничного класу (boundary class), тобто такого, який розміщений на межі системи з оточуючим середовищем.

На діаграмі також можна побачити зв'язки класу часу та дати TimeDate. Він залежить від класу Timer та має зв'язок асоціації з класом DisplayManager.

Наступним етапом проектування системи є створення діаграм взаємодії, які включають діаграму послідовності і діаграму кооперації.

Діаграма послідовності дозволяє представити часову взаємодію між елементами моделі програмної системи у термінології ліній життя об'єктів і повідомлень між ними. Діаграма кооперації відображає внутрішню структуру абстракцій та їх взаємодію.

Для кожного варіанту використання розроблені діаграми послідовності та кооперації. На рис. 4 та рис. 5 наведені приклади таких діаграм, розроблених для даної системи.

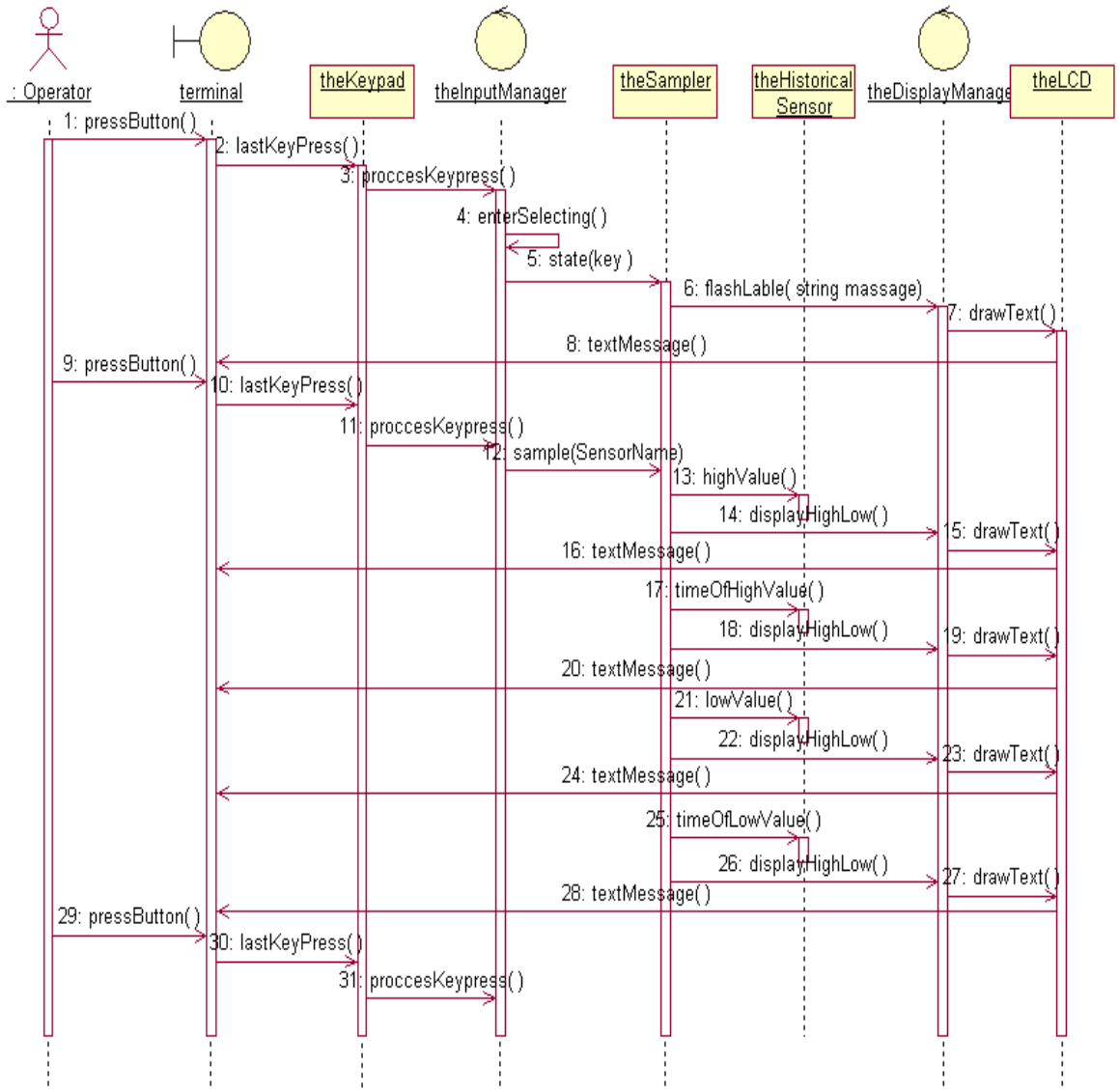

Рис. 4 Діаграма послідовності «SelectingMinMax»

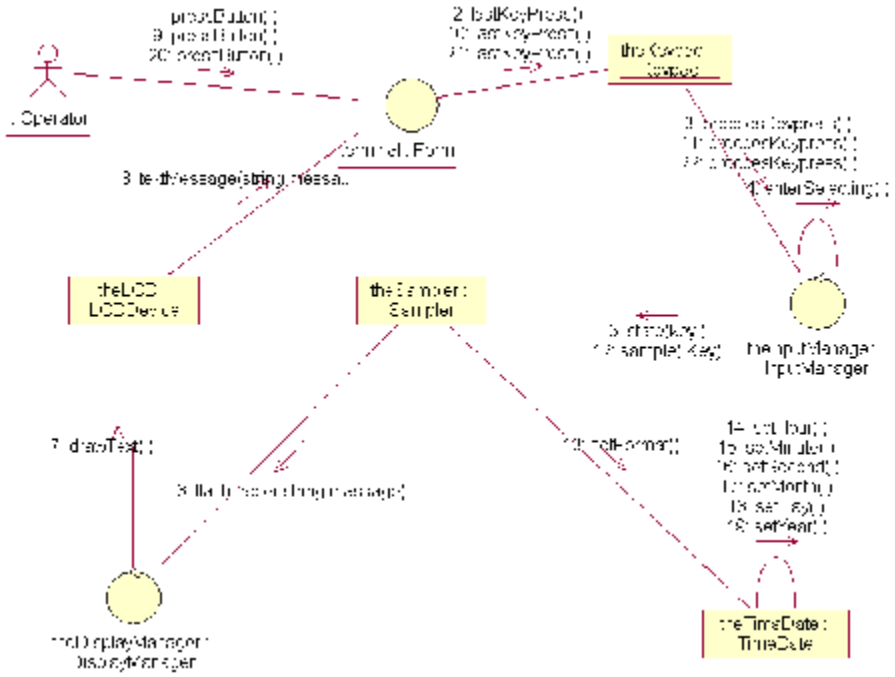

Рис. 5 Діаграма кооперації «Selecting»

Діаграма станів розглядає життєвий цикл об'єкту від створення до знищення. Ця діаграма використовується для представлення дискретної поведінки системи з скінченим числом станів. Вона є графом, який описується скінченим автоматом. У даній інформаційній системі побудована діаграма станів для об'єкта класу InputManager (рис. 6).

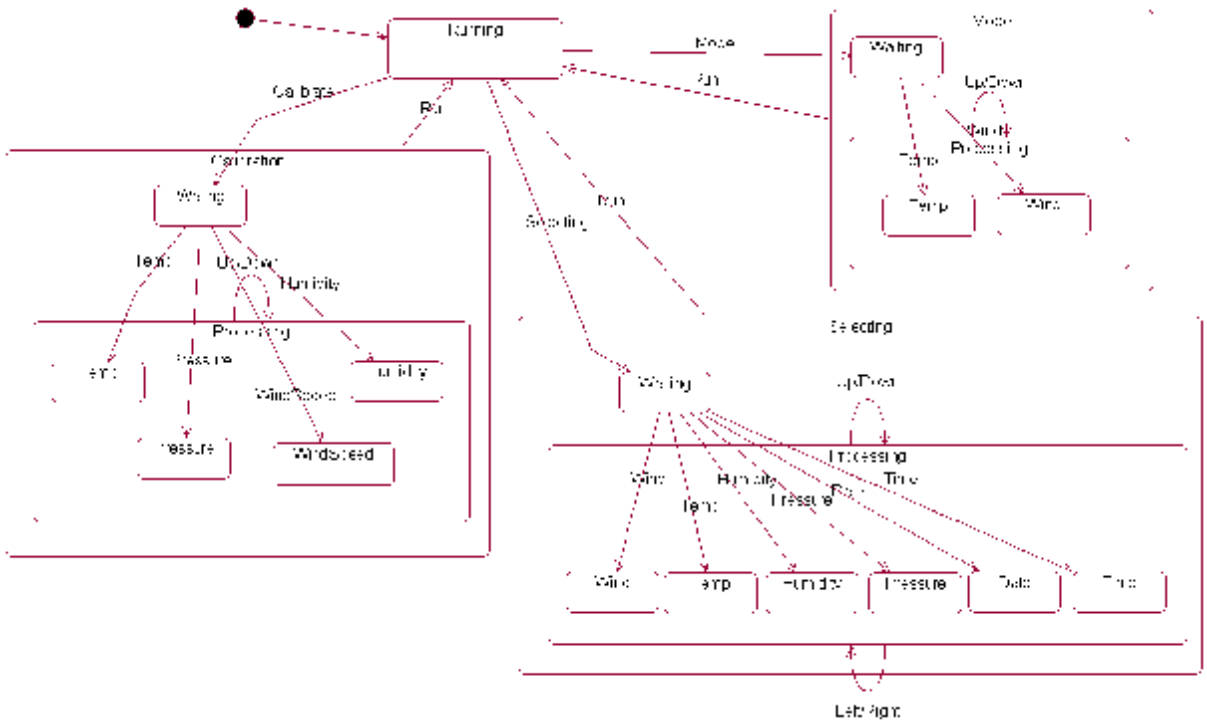

Рис. 6. Діаграма станів об'єкту класу InputManager

Усі класи інформаційної системи, окрім класу-агенту. Sampler, були розділені на три пакети: InputOutput, SensorCollection, Timing. Пакет формує основний спосіб організації моделі з UML. До пакету InputOutput ввійшли класи, відповідальні за введення і виведення даних, а саме: DisplayManager, InputManager, Keypad, LSDDevice та Form. До пакету SensorCollection увійшли усі класи, пов'язані з давачами: Sensor, Sensors, HistoricalSensor, CalibrationSensor, TrendSensor,

DewPoint, WindChill, WindSpeedSensor, WindDirectionSensor, TemperatureSensor, HimiditySensor, PressureSensor, RainFallSensor. До пакету Timing увійшло два класи: Timing та TimeDate.

У системі між пакетами можливий лише один тип відношення – це залежність. Таке відношення означає, що деякий клас з одного пакету зв'язаний однонаправленим відношенням з деяким класом з іншого пакету. На даній діаграмі безпосередньо відображена залежність між двома пакетами InputOutput та Timing (рис. 7).

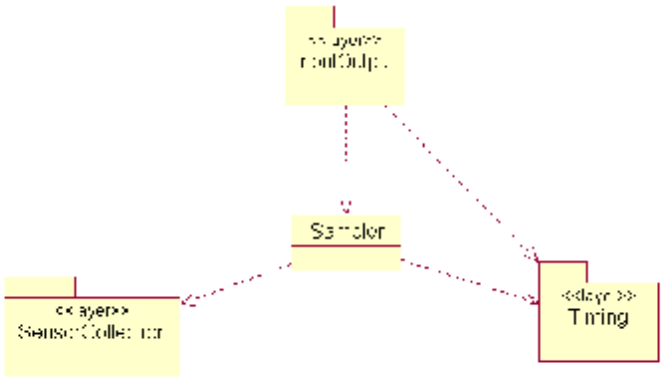

Рис. 7. Діаграма пакетів

Наступною діаграмою, яку було побудовано в процесі проектування є діаграма компонентів (рис. 8). Ця діаграма фізичного рівня, яка використовується для представлення програмних компонентів і залежностей між ними. Перед початком автоматичної генерації коду необхідно співвіднести клас з компонентою. В JAVA кожен клас відповідає одній компоненті – файлу з розширенням .java.

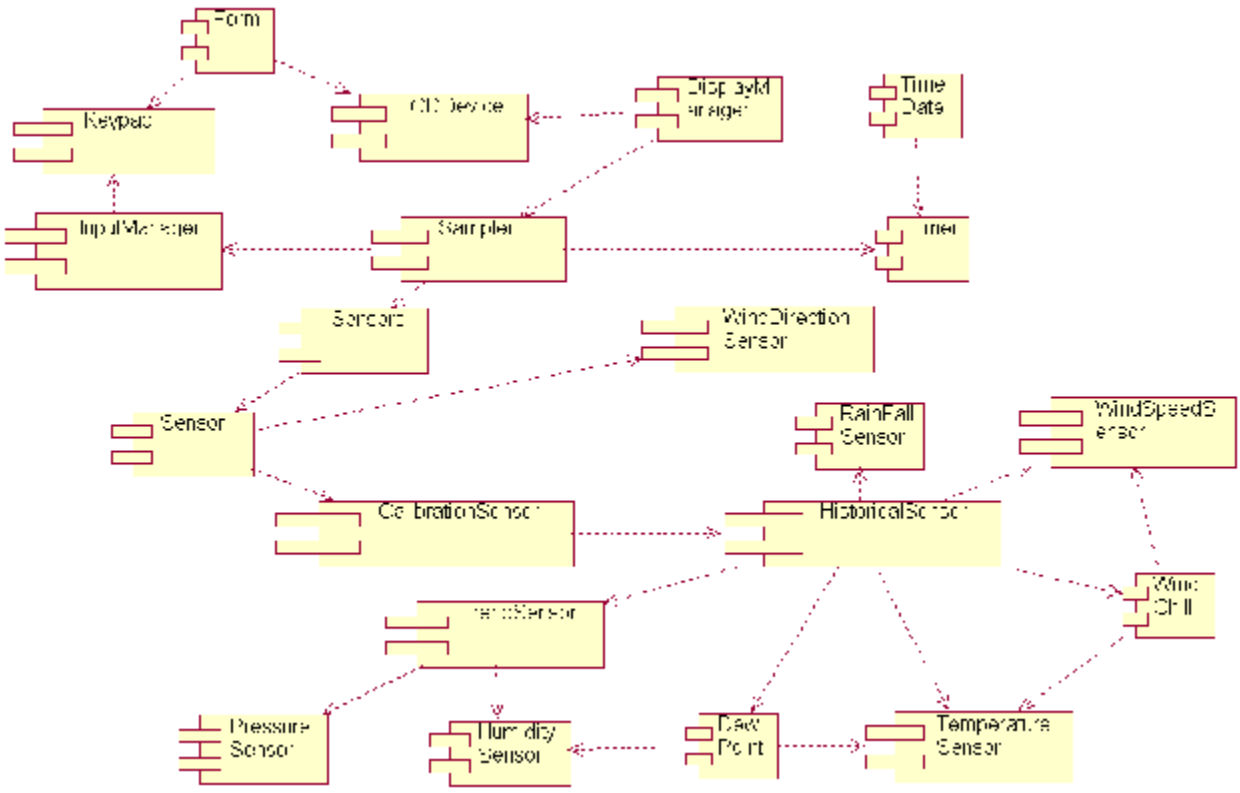

Рис. 8. Діаграма компонентів

Для автоматичного генерування програмного коду у Rational Rose використовується інформація з компоненти для створення бібліотеки коду. Після створення компонентів їх розміщують на діаграмі компонентів і проводять зв'язки.

Останньою побудованою діаграмою стала діаграма розгортання (рис. 9), за допомогою якої моделюється конфігурація програмної системи.

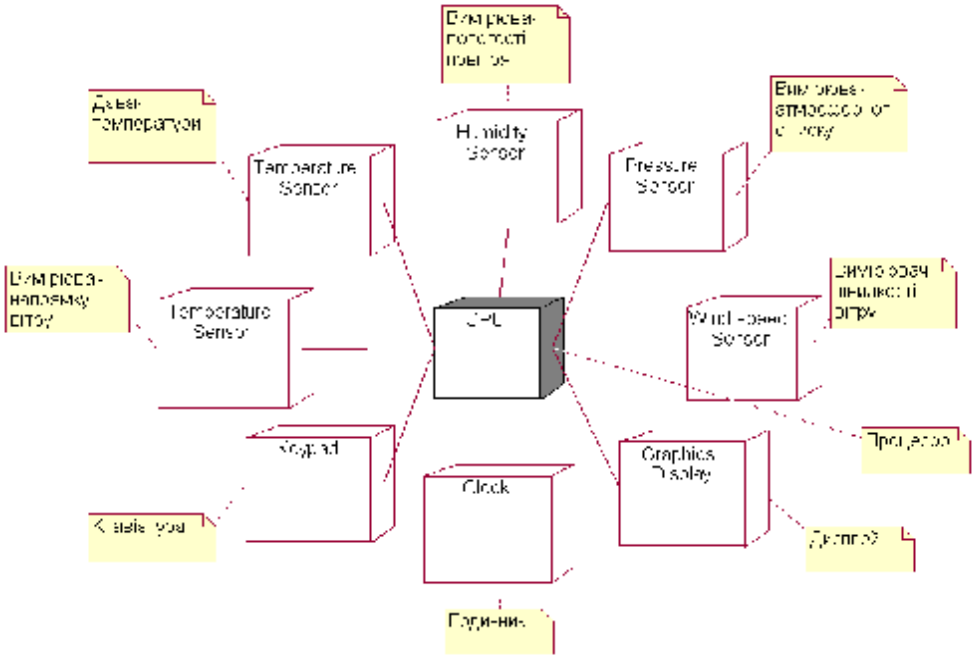

Рис. 9. Діаграма розгортання

Після побудови усіх необхідних діаграм можна приступати до автоматичної генерації коду програми. У результаті генерування було автоматично створено опис усіх класів на мові Java.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.*;
class Form extends JFrame
{
  public LCDDevice LCD;
   public DisplayManager display;
   public MTimer timer;

// Конструктор класу
    public Form()
    {
    super("Метеорологічна станція");
    display = new DisplayManager();
    timer = new MTimer();
    setDefaultCloseOperation
(JFrame.EXIT_ON_CLOSE);

    setSize(500,300);
    setLocation(400,100);
    setVisible(true);
    }
public DisplayManager getDisplayManager()
  { return this.display;}
public LCDDevice getLCD()
  {return this.LCD=LCD;}
public void
setDisplayManager(DisplayManager display)
  { this.display=display;}
public void setLCD(LCDDevice LCD)
  { this.LCD=LCD;}
public void paint(Graphics g)
{
  display.drawStaticItem(g);
  display.displayTime(g);
  display.displayDate(g);
}
//головна програма
public static void main(String arg[])
    {
      Form theForm=new Form();
    }
}
```

Висновки. У роботі з використанням CASE-технології Rational Rose спроектована інформаційна система автоматизованого збирання даних. Проектування класів та діаграм класів

мовою UML допомогло виділити основні абстракції. Проектування станів системи та сценаріїв її поведінки максимально наблизило до розуміння природи системи, як з точки зору програміста, так і користувача.

У програмному забезпеченні наведені діаграми з моделі системи в середовищі Rational Rose. Усі створені діаграми класів відображають можливі зв'язки між певними класами, а також різні механізми функціонування системи на окремих діаграмах. За допомогою діаграм послідовності вибудувана логіка послідовності настання подій у певному потоці. Діаграми кооперації забезпечують взаємодію між об'єктами, відділяючись від послідовності виконання. Для структурування самої моделі створена діаграма пакетів. Діаграма компонентів забезпечує генерування коректного програмного коду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Г. Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. 2-е изд./Пер. с англ. – М.: «Издательство Бином», СПб.: «Невский диалект», 2001. – 560 с.: ил.
2. Ларман, Крэг. Применение UML и шаблонов проектирования. 2-е издание./Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 624 с.: ил. – Парал. тит. англ.
3. А. Леоненков. Самоучитель UML. 2-е издание./П.: «БХВ», 2004. – 158 с.
4. С.А. Трофимов. CASE – технологии. Практическая робота в Rational Rose. М.: «Издательство Бином», 2001. –547 с.