УДК 004.9:504

Є.О. ЄВДІН

РОЗРОБКА АРХІТЕКТУРИ КРОСПЛАТФОРМНИХ РОЗПОДІЛЕНИХ СИСТЕМ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, ОСНОВАНИХ НА МАТЕМАТИЧНИХ МОДЕЛЯХ

Анотація. *Представлено архітектуру системи підтримки прийняття рішень, основаної на математичних моделях. Описані структури даних, що дозволяють уніфікувати обмін даними, стандартизувати процес інтеграції моделей та поєднання їх у ланцюги. Моделі організовані у формі плагинів, які містять усю залежну від них функціональність. Архітектура системи передбачає наявність трьох компонентів, кожен з яких може бути розташований на окремому комп'ютері в мережі: управляючий сервер для забезпечення взаємодії інших компонентів, клієнтська частина з графічним інтерфейсом для збору та візуалізації даних, обчислювач для запуску на виконання математичних моделей та сервери баз даних.*
Ключові слова: *підтримка прийняття рішень, плагин, розподілена система.*

Аннотация. *Представлена архитектура системы поддержки принятия решений, основанной на математических моделях. Описаны структуры данных, которые позволяют унифицировать обмен данными, стандартизировать процесс интеграции моделей и объединения их в цепочки. Модели организованы в форме плагинов, которые содержат всю зависимую от них функциональность. Архитектура системы предполагает наличие трёх компонентов, каждый из которых может исполняться на отдельном компьютере в сети: управляющий сервер, обеспечивающий взаимодействие остальных компонентов, клиентская часть с графическим интерфейсом для сбора и визуализации данных, вычислитель для запуска математических моделей на исполнение и серверы баз данных.*
Ключевые слова: *поддержка принятия решений, плагин, распределённая система.*

Abstract. *Architecture of a decision support system based on mathematical models is introduced. Article describes data structures that allow unifying the data communication, and standardizing process of integration of models and combining them into chains. Models are organized in the form of plug-ins which contain all of their dependent functionality. The system architecture assumes the existence of three components, each of which can be deployed on a single computer in the network: the management server, providing the interaction of the other components, client-side with graphical user interface for data collection and data visualization, computational engine for mathematical models execution and database servers.*
Key words: *decision support, plugin, distributed system.*

1. Вступ

У роботі описані основні структури даних та загальна архітектура системи підтримки прийняття рішень (СППР) на основі результатів розрахунків математичних моделей. Так як якість прогнозних та діагностичних розрахунків залежить від інтегрованих до системи моделей, особлива увага приділяється інтеграції зовнішніх математичних моделей, поєднанню їх до ланцюга моделей та реалізації в ньому потоку даних. Водночас графічний інтерфейс забезпечує зручне представлення результатів з можливістю додавання нових засобів візуального представлення даних за необхідністю. Спеціалізація представленої системи підтримки прийняття рішень визначається інтегрованими моделями та базами даних для збереження моніторингової та прогнозної інформації.

Приклади архітектурних рішень і реалізацій систем підтримки прийняття рішень, заснованих на моделях, представлені в [1–4]. Розвиток сучасних інформаційних технологій

дає змогу закладати в архітектуру СППР можливість кросплатформних та розподілених реалізацій.

Кросплатформність і адаптованість системи до наявного у користувача апаратного та програмного забезпечення дозволяє налаштовувати систему під відповідні обмеження. Водночас розподілена архітектура, використання в ході роботи додаткових процесів, паралельного обчислення дозволяють ефективно задіяти сучасні багатопроцесорні машини та можливості мережі.

2. Вимоги

Згідно з уніфікованим процесом [5] розробки програмного забезпечення, першим кроком у побудові програмного комплексу є встановлення вимог. Вимоги до сучасного програмного комплексу сформульовані у [6] та [7], останнє спеціалізує вимоги до предметної області оцінки та прогнозування наслідків радіаційних аварій, для якої реалізований програмний комплекс на основі представленої нижче архітектури.

Основними функціональними, тобто такими, що описують поведінку, яку має пропонувати система, є такі вимоги:

- інтерфейс користувача повинен бути зручний, звичний та зрозумілий;
- система повинна підтримувати мінімально необхідний набір функцій при роботі з геоприв'язаними об'єктами (прямий імпорт растрових та векторних даних у стандартних форматах до геоінформаційної підсистеми, агрегація результатів, робота з шарами даних);
- легка адаптація до національних та об'єктових умов;
- система повинна підтримувати багато користувачів;
- легка інтеграція зовнішніх обчислювальних моделей;
- використання сучасних систем управління базами даних для зберігання інформації.

Нефункціональними вимогами, тобто особливими можливостями та обмеженнями, що накладаються на систему, є:

- кросплатформність, система повинна працювати в усіх відомих операційних системах;
- забезпечення можливості віддаленого доступу до комп'ютера, на якому буде встановлена система;
- забезпечення можливості розподілення програмного комплексу по декількох комп'ютерах.

Основним прецедентом використання системи є створення користувачем ланцюга моделей, введення вхідних параметрів для кожної з моделей, запуск на обрахунок, отримання результатів та прийняття рішення за допомогою аналізу вихідних значень моделі.

3. Структури даних

Аналіз наведених вимог і прецедентів показує, що система буде мати справу з числовими даними, вхідними та вихідними параметрами від різних моделей і повинна забезпечувати обмін даними між моделями. Одним із шляхів поєднання моделей є стандарт OpenMI, що визначає інтерфейс, який дозволяє просторово-часовим моделям обмінюватися інформацією під час виконання [8]. OpenMI визначає набір інтерфейсів, які повинна реалізувати модель, щоб її можна було поєднати з іншою OpenMI сумісною моделлю.

Розглядувана система використовує схожий принцип: для представлення даних використовується структура класів датаітемів (DataItem) [9], які зберігають власне чисельні дані, метадані (розмірність, одиниці виміру, субстанцію) та відносини між ними. Таке рішення уніфікує діалог системи та різних моделей, дозволяє відображати вхідні параметри

й результати розрахунку моделей за допомогою типових візуалізаторів, спрощує процес поєднання моделей, замінюючи встановлення відповідності виходів попередньої моделі входам наступної моделі, операціями з датаітемами.

Класи датаітемів реалізують композитний шаблон програмування, який об'єднує об'єкти у деревовидну структуру та дозволяє системі звертатися до окремих об'єктів і до групи об'єктів однаково [10]. Таким чином, розглядувана система класів представляє собою ієрархічну структуру, листами якої є датаітеми з конкретними чисельними даними, а проміжними вершинами – комплексні датаітеми.

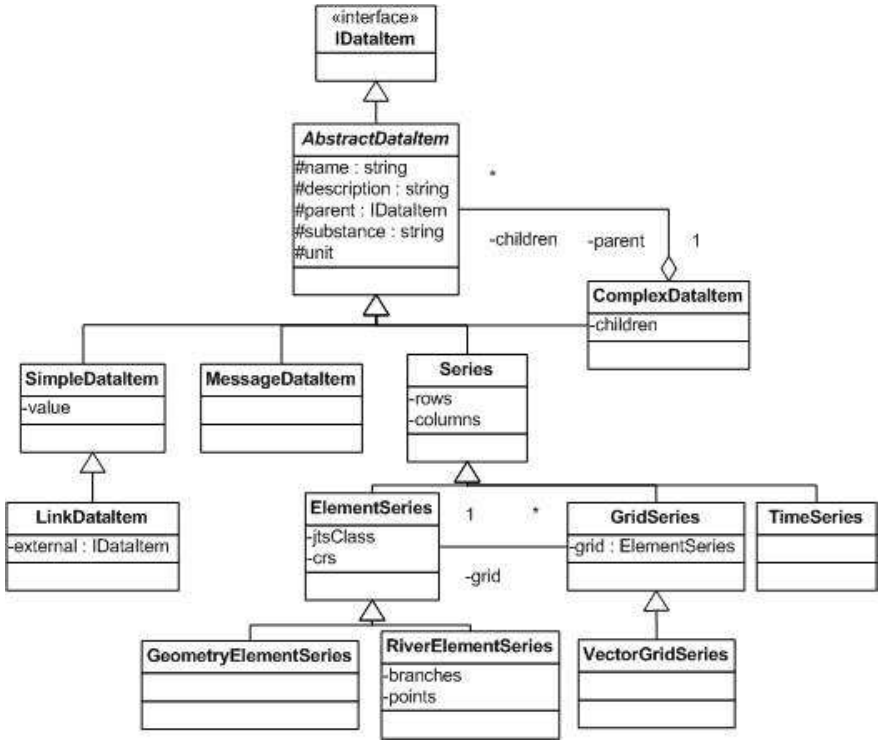

Рис. 1. Структура класів датаітемів

На рис. 1 наведена UML-діаграма даної структури класів. Якщо існуючих класів недостатньо для опису предметної області, можна додавати нові типи датаітемів, не змінюючи існуючу структуру. Кожен датаітем реалізує інтерфейс IDataItem, що визначає операції, підтримувані всіма датаітемами. Абстрактний клас AbstractDataItem реалізує спільні функції інтерфейсу, які, при необхідності, можуть бути перевизначені в його нащадках.

Основні операції можна розділити на дві групи:

• операції, що підтримують маніпулювання семантикою даних, наприклад, встановлення та повернення імені, опису, розмірності, значення, субстанції тощо;

• операції для маніпулювання ієрархічністю структури даних: додавання, пошук, видалення безпосереднього нащадка, встановлення та повернення батька даного датаітема, визначення шляху від кореня та ін.

Комплексний датаітем (ComplexDataItem) служить для організації датаітемів у деревовидну структуру, містить у собі своїх безпосередніх нащадків і не призначений для зберігання чисельних даних. Проте може мати інші спільні для всіх нащадків характеристики, такі як опис, субстанція, одиниці виміру.

Простий датаітем (SimpleDataItem) служить для збереження чисельних даних як єдиного цілого і не передбачає операцій з їх частиною (лише встановлення та повернення усіх даних одним об'єктом). Найчастіше зберігає скалярні величини, але в загальному випадку призначений для зберігання будь-якого типу даних.

Датаітем-повідомлень (MessageDataItem) призначений для збереження повідомлень від моделі із зазначенням часу, рівня (інформаційне, попередження, помилка) та тексту повідомлення.

Таблиця, послідовність (Series) зберігає табличні дані, містить у собі структури рядків та стовпчиків, кожен стовпчик може містити дані лише одного, але довільного типу.

Часова таблиця, часова послідовність (TimeSeries) представляють собою таблицю, в якій перша колонка завжди містить дату та час.

Таблиця елементів ElementSeries наслідує клас Series і призначена для збереження регулярної просторової області (сітки), а також містить клас просторових даних (точка, полігон, лінія та ін.) та тип системи координат. Кожен рядок таблиці відповідає одній комірці, колонки визначають координати центру комірки, орієнтацію та лінійні розміри.

RiverElementnSeries, GeomertyElementSeries наслідують ElementSeries і реалізують відповідно річкову сітку та просторову структуру з комірками довільної форми.

Сіткова таблиця (GridSeries) визначає чисельні дані для спільної просторової області, визначеної у ElementSeries, і посилання на яку вона містить. Порядок даних повинен відповідати порядку комірок у просторовій області. Таке обмеження дозволяє зменшити об'єм пам'яті, необхідний для зберігання просторово-часових даних. Кожна колонка таблиці відповідає певному моменту часу.

Датаітем-посилання (LinkDataItem) – це простий датаітем, що як значення містить інший датаітем (найчастіше це комплексний датаітем) і перевизначає ієрархічні операції й операції повернення значення на відповідні функції з датаітемом-значенням. В основному використовується при поєднанні моделей у ланцюг для відповідності вхідних даних однієї моделі вихідним даним іншої.

Одиницею даних у роботі системи є Задача (Task), яка є реалізацією обчислювальної моделі з конкретними входами, виходами, поточним станом тощо. Датаітеми, що представляють всі вхідні та вихідні параметри, ієрархічно об'єднуються у дерево, кореневий елемент якого належить задачі. Декілька задач об'єднуються у Проект (Project), який є реалізацією ланцюга моделей і містить, окрім задач, дані про власника проекту, час його створення та зміни, назву ланцюга моделей, яку він реалізує. Задачу не можна переносити з одного проекту до іншого, але їх можна копіювати в середині самого проекту, створюючи декілька гілок обчислень для порівняння результатів при незначній зміні вхідних параметрів. Структура проектів, задач наведена в UML-діаграмі на рис. 2.

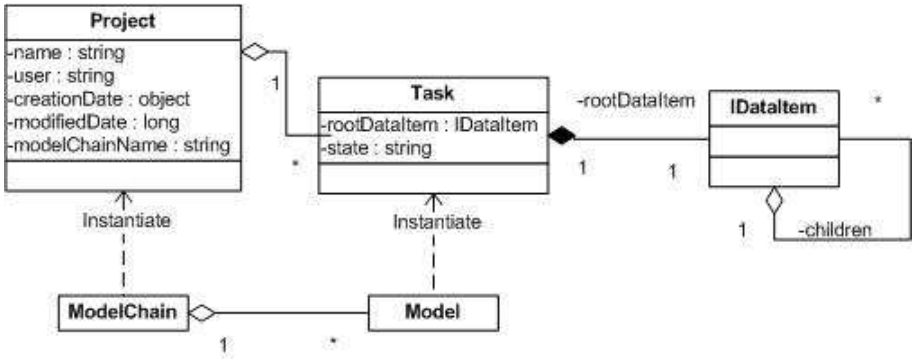

Рис. 2. Структура проектів та задач

4. Загальна архітектура системи

Абстрактна архітектура системи підтримки прийняття рішення, що реалізує зазначені вище вимоги та прецеденти, складається з окремих компонентів (рис. 3), кожен з яких може бути розміщений як на окремому, так і на одному комп'ютері: Управляючий сервер або Менеджер, Обчислювальний компонент, Клієнт, Сервери баз даних. Різні користувачі системи можуть приєднуватися до Менеджера за допомогою Клієнта, кожному користувачу виділяється окремий Обчислювальний компонент.

Рис. 3. Загальна архітектура систем зі схемою потоків даних

4.1. Управляючий сервер (менеджер)

Управляючий сервер (Менеджер) є основним компонентом системи, який забезпечує взаємодію між собою усіх інших компонентів, контролює та направляє потоки даних між частинами системи. Це єдиний компонент системи, який має зв'язок з усією рештою компонентів, які, у свою чергу, контактують тільки з ним. Менеджер відсилає запити до баз даних, видає команди на запуск обчислювальних моделей, надає їм вхідні дані з різних джерел, наприклад, з графічного інтерфейсу клієнта, бази даних, попередньо обчислених моделей, слідкує за станом задач та реагує на їх зміну, передає вихідну інформацію відповідним адресатам (графічному інтерфейсу користувача, геоінформаційному модулю клієнта, базі даних).

Менеджер містить у собі ізольовані один від одного робочі області користувачів (англ. User workspace) та спільний планувальник завдань (англ. Job scheduler). Кожному користувачу, що приєднується до Менеджера, виділяється автономна робоча область, яка містить усі необхідні об’єкти для роботи, основними з яких є посилання на віддалені об’єкти відповідних користувачу компонентів Клієнта та Обчислювача. Віддалені виклики методів з Менеджера на Клієнт і Обчислювач ставляться у чергу, операції з якою синхронізовані й безпечні для багатопотокового виконання. У разі втрати з’єднання між Менеджером та цільовим компонентом віддаленого виклику, заявки на виклик зберігаються у черзі доти, доки з’єднання не відновиться або черга не буде скинута.

Планувальник завдань зберігає завдання (англ. Job) та активує їх виконання. Час, періодичність та кількість виконань визначаються тригерами, які надсилаються до планувальника разом із завданням. В залежності від наявних ресурсів (тактова частота процесора, кількість ядер, розмір оперативної пам’яті) можна регулювати кількість потоків обслуговування черги завдань у зовнішньому конфігураційному файлі. Завдання надходять до Менеджера з інших компонентів системи, Клієнтів та Обчислювачів і можуть містити різноманітні роботи: створення, збереження, завантаження проектів, ініціалізацію моделей,

запуск обчислень, виконання запитів до бази даних тощо. По закінченні виконання завдання генерується заявка на віддалений виклик методу з результатом виконаного завдання та повідомленням про його закінчення, що стає у відповідну чергу віддалених викликів у робочій області користувача. Окрім цього, Менеджеру можна поставити до виконання статичні автоматичні завдання, тобто такі, які не потребують прив'язки до конкретних Клієнтів чи результатів обчислення. Серед таких завдань можна виділити імпорт різноманітних даних до бази даних із зовнішніх джерел (прогнозу погоди, результатів моніторингу у реальному часі тощо), очищення баз даних від зайвої інформації (наприклад, старих результатів проектів, видалених завдань, неактуальних даних), резервне копіювання та ін. Автоматичні завдання разом зі своїми тригерами визначаються у відповідному конфігураційному файлі.

4.2. Клієнт

Клієнтська частина системи – це орієнтований на користувача компонент, основним завданням якого є представлення інформації користувачу, відображення його дій та підготовка завдань для планувальника задач Менеджера. Клієнт містить у собі графічний інтерфейс, який складається з таких модулів: основний модуль, модуль інтерфейсів моделей, геоінформаційний модуль, модуль звітів, модуль візуалізаторів.

Модуль інтерфейсів моделей відображає інтерфейси обчислювальних моделей і призначений для збору вхідних параметрів від користувача у зручній для нього формі, валідації введених параметрів, перетворення їх на датаітеми, надання рекомендацій та довідок щодо необхідних вхідних даних. Вхідні дані можуть бути як конкретними скалярними величинами, так і посиланням на джерело даних (наприклад, бази даних, результатів попередньо обчисленої моделі у ланцюгу моделей), визначення просторового, часового, логічного фільтру.

Геоінформаційний модуль призначений для зручного відображення просторово-часових та тематичних даних, які складаються з цифрових моделей місцевості (рівнів висот, категорій землекористування, ґрунтів, річок та озер, кількості населення, карти доріг, населених пунктів, кордонів адміністративних регіонів тощо), карт розташування об'єктів, що представляють інтерес (вимірюючих станцій, блоків атомних станцій, будинків та ін.), результатів обчислення та інформації з бази даних (карти прогнозу погоди, вимірів різноманітних характеристик).

Геоінформаційний модуль Клієнта відображає просторово-часові дані у вигляді множини шарів. Шари, які відповідають цифровим моделям місцевості (картам-підкладкам) та картам статичних об'єктів, відображають фіксовану (тобто таку, на яку система не впливає) інформацію, джерелом якої є спеціальні файли у сучасних растрових та векторних форматах (shp, tiff), геоінформаційна база даних, WFS, WMS-сервери тощо. Шари, що відповідають результатам обчислень, оперативній інформації з бази даних, формуються ГІС-модулем Клієнта в оперативній пам'яті на основі чисельних даних, що у вигляді датаітемів надходять з Менеджера. Модуль дозволяє анімувати такі шари. Ці шари, накладені на статичну інформацію, дають змогу оцінити ситуацію та полегшити процес прийняття рішення.

Модуль також містить мінімально необхідний набір засобів для роботи з шарами даних, серед яких є можливість масштабування, перетягування, вимірювання відстаней, виділення об'єктів, перегляду атрибутивної інформації, експорту даних у векторний формат. Інтерфейси обчислювальних моделей, що потребують роботу з геоприв'язаними даними, можуть надавати спеціалізовані засоби для роботи із відповідними додатковими шарами.

Модуль візуалізаторів зображує значення датаітема у зручній для користувача формі: простого тексту, масиву, таблиці, гістограми. Просторово-часові дані зображуються за допомогою геоінформаційного модуля. Візуалізатори призначені лише для відображення інформації у датаітемі і не передбачають можливості її зміни.

Модуль звітів дозволяє будувати звітні документи на основі отриманих результатів та вхідних параметрів, використовуючи карти, графіки, табличні дані, текст. Цей модуль представляє собою редактор звітів (точніше схеми звіту) та движок, що створює звіт у внутрішньому форматі за відповідною схемою та даними. Звіт надалі можна експортувати до стандартних, зручних форматів (pdf, xls, rtf та ін.) або одразу ж відсилати на друк. Модуль звітів підтримує як інтерактивне, так і автоматичне створення звітів після розрахунку обчислювальної моделі.

Основний модуль керує усіма вищезазначеними модулями й призначений для забезпечення користувачу доступу до відповідної функціональності, а також відображення поточного стану проектів, завдань та датаітемів, завантажених у систему. Серед основних дій, які можуть бути виконані користувачем, слід виділити створення, копіювання, видалення проектів; копіювання, видалення завдань у середині проекту, виклик інтерфейсу користувача для моделі, виклик візуалізатора для обраного датаітема, візуалізацію інформації з бази даних; групу завдань, що відправляється на Менеджер: ініціалізацію, запуск на виконання завдань та отримання результатів.

4.3. Обчислювальний компонент

Обчислювальний компонент відповідає за ініціалізацію, запуск моделей на виконання та отримання результатів. Він зберігає задачі (екземпляри класу Task) і разом з ними вхідну та вихідну інформацію по кожному проекту, з яким працює користувач на даний момент. Цей компонент за командою з Менеджера запускає на виконання обчислювальне ядро моделі, перетворює вхідну інформацію з датаітемів до формату, необхідного кожній моделі, забирає та зберігає результати обчислень, трансформуючи їх до датаітемів. Вхідні дані можуть бути отримані через Менеджера з бази даних або інтерфейсу користувача моделей, з попередньо обчислених моделей, а також з атрибутивної інформації геоінформаційних даних. Під час роботи моделі Обчислювач направляє поточні вихідні результати, повідомлення, попередження та виключні ситуації Менеджеру для їх обробки.

Для підвищення загальної стабільності системи кожна модель запускається в окремому процесі. Такий підхід дозволяє ефективно використовувати можливості сучасних багатоядерних процесорів, а також захищає основну систему від помилок у моделі, таких як витоки пам’яті, неперехоплені виключення, критичні помилки та непередбачені закінчення виконання процесу обчислення.

4.4. Сервер баз даних

Необхідні системі бази даних можуть знаходитися як на одному сервері баз даних, так і на декількох окремих. Серед основних баз даних можна виділити базу даних проектів, що зберігає проекти і задачі разом зі своїми вхідними та вихідними даними, геоінформаційну базу даних, що містить геодані, розмір яких не дозволяє завантажувати їх одразу до оперативної пам’яті, базу фіксованих даних, яка містить таку інформацію, що не змінюється часто, а також бази зовнішніх даних, необхідних для роботи системи (наприклад, дані прогнозу погоди, моніторингової інформації). Пізніше до системи можуть бути встановлені додаткові бази даних, в яких інтегровані моделі зберігають необхідну інформацію, параметри, бібліотечні дані. Інформація, необхідна для встановлення зв’язку з базами даних, прописується у відповідному конфігураційному файлі Менеджера. Доступ до даних, їх онов-

лення та видалення відбувається через завдання, які надходять до Менеджера з Клієнта, Обчислювача або є статичними автоматичними завданнями самого Менеджера (наприклад, імпорт даних, при їх наявності у відповідному місці (папки на жорсткому диску, фтп-сервери тощо), видалення даних за визначеним фільтром, резервне копіювання.

5. Інтеграція обчислювальних моделей

Моделі організовані у формі плагинів, тобто необов'язкових програмних компонентів, що незалежно компілюються та динамічно підключаються до основної системи для розширення її функціональності [11]. Вся залежна від моделей функціональність винесена до плагинів так, щоб інтеграція кожної нової моделі або зміни в уже існуючі не призводила б до необхідності вносити зміни в уже існуючі компоненти та перевіряти на цілісність систему, тобто підтримувала б можливості незалежного розширення [6].

Обчислювальні моделі в абсолютній більшості випадків розроблялися окремо від системи, їх обчислювальне ядро написане на іншій мові програмування, ніж сама система. Тому модель компілюється у динамічні бібліотеки. Для інтеграції код моделі модифікується лише у функціях, процедурах вводу-виводу даних для того, щоб направити ці потоки інформації до системи. Реалізація моделей у вигляді динамічних бібліотек, а не автономних програмних проектів, дозволяє скористатися перевагами концепцій обміну даними за допомогою датаітемів, підтримувати зворотній зв'язок з моделями (англ. callback) та істотно скоротити обсяг оброблюваної інформації в порівнянні з використанням файлової системи.

Плагин складається з пакетів, кожен з яких виконується на окремому компоненті системи: пакет з інтерфейсом користувача, пакет з інтерфейсом доступу до бази даних та завданнями для Менеджера, пакет з інтерфейсом до обчислювальної моделі. Окрім програмного коду, плагин містить також динамічні бібліотеки, ресурсні файли, стилі відображення тощо (рис. 4).

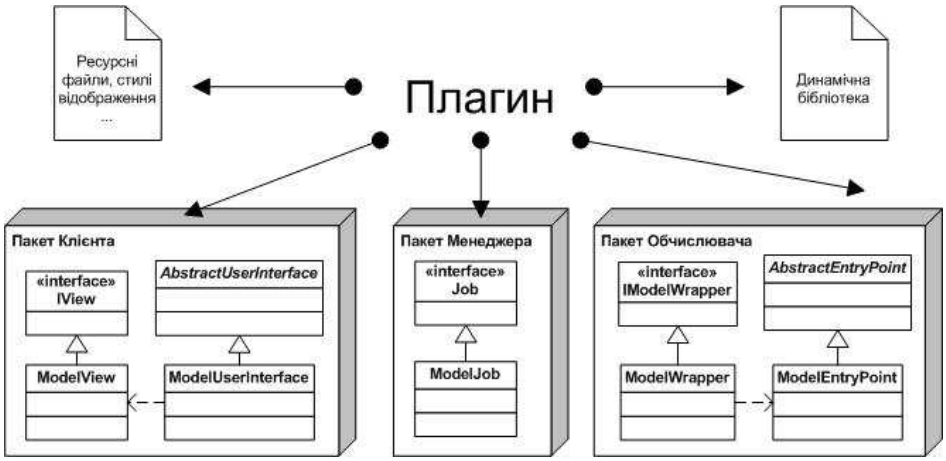

Рис. 4. Структурна схема плагину

Інтерфейс користувача моделі є необов'язковим пакетом, який виконується на Клієнті у модулі інтерфейсів моделей. У складі цього пакета є два основні класи:

• клас ModelView, що реалізує інтерфейс IView та призначений для збору вхідних параметрів від користувача у зручній для нього формі, валідації введених параметрів, надання рекомендації та довідок щодо необхідних вхідних даних;

• клас ModelUserInterface, що уточнює AbstractUserInterface та призначений для перетворення датаітемів на чисельні дані й навпаки, для виклику функцій Клієнта, створення об'єктів класу ModelView, надання та збору з нього чисельних даних.

Пакет із завданнями Менеджеру, інтерфейсом доступу створюється у тому випадку, якщо для моделі існує спеціальна база даних, що містить, наприклад, параметри, бібліоте-

чні дані або інформацію, збереження якої не передбачено основними базами даних системи.

Пакет з інтерфейсом до обчислювальної моделі є єдиним обов'язковим пакетом плагину, який реалізує власне обчислювальну функціональність. Подібно до пакета інтерфейсу користувача, цей пакет також містить два основних класи:

• клас ModelWrapper, що реалізує інтерфейс IModelWrapper та призначений для роботи з деревом датаітемів, тобто перетворення вхідних датаітемів на чисельні дані, необхідні власне обчислюваному ядру, створення вихідних датаітемів на основі отриманих результатів, формування ієрархічної структури результатів, генерування повідомлень та змін стану завдання. Цей клас є атрибутом класу Task і тому зберігається й виконується на Обчислювальному компоненті;

• клас ModelEntryPoint, який уточнює абстрактний клас AbstractEntryPoint, виконується в окремому процесі і відповідає за зв'язок з ModelWrapper та обмін з ним чисельними даними, а також за роботу з динамічною бібліотекою, яка є власне обчислювальним ядром. Робота полягає у пошуку відповідної бібліотеки, завантаженні її в оперативну пам'ять та виклику відповідних її зовнішніх функцій.

6. Забезпечення кросплатформності системи

Для забезпечення мультиплатформності системи підтримки прийняття рішень її розробка відбувається на мові Java. Програмний код компілюється у байт-код, який виконується віртуальною машиною Java. Таким чином, система може функціонувати у будь-якому середовищі, для якого існує імплементація віртуальної машини. На сьогоднішній час це більшість відомих операційних систем, включаючи Windows, різноманітні Linux, MacOS. Зв'язок компонентів системи встановлюється за протоколом TCP/IP за допомогою технології віддалених викликів RMI.

Обчислювальні моделі, у свою чергу, компілюються у динамічні бібліотеки окремо для кожної платформи, на якій плануються використовуватися. Якщо код обчислювальних моделей написаний якісно, без використання (або з мінімальним використанням) нестандартних функцій та можливостей компілятора, то процес портування моделі на різні платформи є простим і включатиме у себе лише компіляцію одного і того програмного коду під різні платформи.

7. Висновки

У роботі представлено архітектуру системи підтримки прийняття рішень, основаної на математичних моделях. Описана система підтримує можливість пристосовуватися до різного програмного та апаратного середовища, вона є розподіленою для ефективного розміщення ресурсоємних компонент на більш потужних комп'ютерах. Водночас з цим, усі компоненти системи можуть функціонувати на одному сучасному персональному комп'ютері. Система є кросплатформною, тобто існує можливість виконання системи на різних операційних системах, а також виконання різних компонентів у різних середовищах.

Інтеграція моделей у вигляді плагинів з можливістю виконання додаткової функціональності усіма компонентами дозволяє адаптувати систему до розв'язання нових задач без модифікації існуючого коду, тобто робить її розширюваною. Запуск обчислювальної моделі в окремому процесі забезпечує стабільність системи, її незалежність від помилок моделей, витоку пам'яті, непередбачуваних переривань виконання. Водночас застосування динамічних бібліотек з обгорткою з системного класу дозволяє скористатися перевагами концепцій обміну даними за допомогою датаітемів, підтримувати зворотний зв'язок з мо-

делями та істотно скоротити обсяг оброблюваної інформації в порівнянні з використанням автономних програмних проектів.

На основі вимог та описаної абстрактної архітектури реалізована перепроектована система підтримки прийняття рішень при аварії на ядерних об'єктах у Європі – Родос [12].

СПИСОК ЛІТЕРАТУРИ

1. Гофман Д.С. Применение программно-инструментальной системы LIANA для интеграции прикладных задач, ГИС и баз данных в системы поддержки принятия решений, основанные на моделях / Д.С. Гофман // Математические машины и системы. – 1998. – № 1. – С. 75 – 88.
2. Морозов А.А. Системы принятия решения: проблемы и перспективы / А.А. Морозов // УСиМ. – 1995. – № 1. – С. 13 – 21.
3. Elbir T. A GIS based decision support system for estimation, visualization and analysis of air pollution for large Turkish cities / T. Elbir // Atmospheric Environment. – 2004. – Vol. 38. – P. 4509 – 4517.
4. STEEDS: A strategic transport–energy–environment decision support / C. Brand, M. Mattarelli, D. Moon [et al.] // European Journal of Operational Research. – 2002. – Vol. 139. – P. 416 – 435.
5. Arlow J. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Edition) / J. Arlow, I. Neustadt. – Addison-Wesley Professional, 2005. – 624 p.
6. Массель Л.В. Моделирование и разработка современных программных комплексов для исследований энергетики / Л.В. Массель, Е.А. Болдырев // Вычислительные технологии. – 2002. – Т. 7, № 4. – С. 59 – 70.
7. User requirements for the re-design of RODOS in phase 2 of the EURANOS-project, EURANOS(CAT2)-TN(06)-09 / F. Gering, B. Gerich, T. Duranova [et al.]. – Karlsruhe, 2006. – 22 p.
8. Gregersen, J.B. OpenMI: Open Modelling Interface / J.B. Gregersen, P.J.A. Gijsbers, S.J.P. Westen // Journal of Hydroinformatics. – 2007. – Vol. 9 (3). – P. 175 – 191.
9. Donchyts G. Object-Oriented Framework for Modelling of Pollutant Transport in River Network / G. Donchyts, M. Zheleznyak // Computational Science – ICCS 2003, Lecture Notes in Computational Science. – 2003. – Vol. 2657, 0302-9743. – P. 35 – 44.
10. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, H. Richard, R. Johnson, J. Vlissides. – Addison-Wesley Professional, 1994. – 416 p.
11. Predictable Dynamic Plugin Systems / R. Chatley, S. Eisenbach, J. Kramer [et al.] // Fundamental Approaches to Software Engineering (FASE 2004), Lecture Notes in Computer Science. – 2004. –Vol. 2984. – P. 129 – 143.
12. Ievdin Ie. Distributed architecture of RODOS – decision support system for nuclear emergency management in Europe / Ie. Ievdin, D. Treebushny, M. Zheleznyak // Системи підтримки прийняття рішень. Теорія і практика: зб. доп. наук.-прак. конф. з міжнар. участю. «СППР'2010». – Київ: СП «Інтертехнодрук», 2009. – С. 181 – 184.

Стаття надійшла до редакції 15.04.2010