УДК 004.9:504

Є.О. ЄВДІН, М.Й. ЖЕЛЕЗНЯК, Д.М. ТРИБУШНИЙ

РОЗРОБКА КРОСПЛАТФОРМЕНОЇ ВЕРСІЇ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ПРИ РАДІАЦІЙНИХ АВАРІЯХ JRODOS

Анотація. *Представлено реалізацію системи підтримки прийняття рішень з реагування на ядерні аварії JRODOS на основі новітніх інформаційних технологій з використанням програмного інструментарію у відкритих кодах. Детально описано архітектуру та інформаційні технології, реалізовані при розробці системи. Представлені методика та програмні засоби інтеграції зовнішніх обчислювальних моделей до системи на основі уніфікованого типу даних у вигляді незалежних програмних компонент – плагінів. При розробці системи запропоновано декілька підходів, що зменшують обсяг використаної оперативної пам'яті, пришвидшують роботу та забезпечують стабільність, відмовостійкість системи.*
Ключові слова: *підтримка прийняття рішень, плагін, розподілена система.*

Аннотация. *Представлена реализация системы поддержки принятия решений по реагированию на ядерные аварии JRODOS на основе новейших информационных технологий с использованием программного инструментария в открытых кодах. Детально описаны архитектура и информационные технологии, реализованные при разработке системы. Представлены методика и программные средства интеграции внешних вычислительных моделей к системе на основе унифицированного типа данных в виде независимых программных компонент – плагинов. При разработке системы предложено несколько подходов, которые уменьшают объем использованной оперативной памяти, ускоряют работу и обеспечивают стабильность, отказоустойчивость системы.*
Ключевые слова: *поддержка принятия решений, плагин, распределённая система.*

Abstract. *Implementation of a decision support system on nuclear emergency respond JRODOS based on the latest information technologies using software in open codes is introduced. The article describes architecture and information technologies which were implemented during system design. The methods and software tools of external computational models integration on base of unified data types are described. Models are integrated as plugins. In system implementation several approaches, which reduce the amount of used memory, speed up and ensure the stability and resilience of the system are proposed.*
Keywords: *decision support, plugin, distributed system.*

1. Вступ

Останнім часом значна увага приділяється створенню та розвитку комп'ютерних систем, призначених для оцінки та прогнозування наслідків різних екологічних аварій. Сучасні системи підтримки прийняття рішень (СППР) з питань екологічної безпеки засновані на моделях прогнозування впливу надзвичайних ситуацій (НС) на довкілля та населення, що працює й мешкає в середовищі, на яке впливає НС, та, у відповідності з цими прогнозами, розраховують ефективність різних можливих контрзаходів, які повинні ліквідувати шкідливі наслідки НС для населення та довкілля. З урахуванням такого визначення, СППР з екологічної безпеки, які працюють у реальному часі, можна віднести до Автоматизованих систем управління (АСУ) в тому сенсі, що, отримуючи інформацію від систем моніторингу, такі системи розраховують і рекомендують найбільш ефективні заходи з управління ліквідацією наслідків НС.

У рамках наукового напряму «Математичне і програмне забезпечення обчислювальних машин» задача розробки сучасних СППР (АСУ) з екологічної безпеки приводить до необхідності розробки комплексних програмних систем, які повинні об'єднувати в кросплатформеному та багатомовному середовищі підсистеми обчислювальних кодів математичних моделей, баз даних реального часу, баз даних інформації періодичного оновлювання, баз даних ГІС (цифрових карт), числові інтерфейси з зовнішніми джерелами прогностич-

ISSN 1028-9763. Математичні машини і системи, 2012, № 1

ної інформації (наприклад, метеорологічної), підсистеми обробки вхідної інформації для її представлення в обчислювальні модулі, підсистеми обробки вихідної інформації для передачі у графічний інтерфейс користувача з елементами ГІС, управляючий модуль усієї програмної системи.

Основною вимогою до таких СППР є простота інтеграції нових математичних обчислювальних моделей до системи, що представляє собою комплексне завдання по забезпеченню такої функціональності: завантаження моделі до системи, надання можливості введення вхідних даних користувачем, наповнення вхідних даних, запуск розрахунку моделі, повернення та обробка вихідних даних, об'єднання моделей в обчислювальні ланцюжки та забезпечення потоку даних з однієї моделі до іншої. При цьому математичні моделі розробляються та тестуються незалежно від системи, використовуючи різні мови програмування, формати вхідних та вихідних даних.

Прикладом такої систем підтримки реагування на екологічні аварії є датська система ARGOS (http://www.pdc.dk/Argos/decision.asp), що набула широкого розповсюдження у країнах Балтії, Скандинавії, Канаді та ін. Система включає моделі атмосферного переносу (на короткі, середні та далекі відстані, перенос в умовах забудови), моделі розрахунку доз опромінення у продуктах харчування, модулі сільськогосподарських контрзаходів та заходів у населених пунктах. ARGOS здатний проводити розрахунки наслідків хімічних аварій, містить модель оцінки джерела для хімічних викидів. Система працює в операційній системі Windows, є клієнтською оболонкою з єдиним центральним сервером баз даних під керуванням Microsoft SQL Server.

У Росії науково-виробничим об'єднанням «Тайфун» Росгідромету розроблена система РЕКАСС [1, 2]. Основним призначенням сучасної її версії є оцінка та прогноз наслідків аварійних викидів, скидів, розливів, забруднюючих речовин у навколишнє середовище. Система розроблялась для моделювання наслідків аварійних викидів в атмосферу. Лише в останні роки розпочалась розробка гідрологічного модуля для оцінки забруднення поверхневих вод, а також розвивається модуль оцінки ефективності після аварійних контрзаходів.

В Україні на Рівненській АЕС впроваджено систему прогнозування наслідків аварійних атмосферних викидів КАДО [3], яка дозволяє розраховувати за заданим напрямком вітру дози в 30 км зоні АЕС за рахунок прямого опромінювання, інгаляції та внутрішнього опромінювання за межами зони за рахунок вживання забруднених продуктів.

Європейська СППР RODOS розробляється з 1992 року в рамках наукових програм Європейської комісії [4]. Ця система проектувалась як система реального часу, що використовує прогностичні поля метеоелементів на різних висотах, і включає в себе математичні моделі розповсюдження радіонуклідів у всіх компартментах навколишнього середовища та бази даних реального часу для прогнозування й оцінки наслідків можливих радіаційних аварій, а також планування невідкладних і довгострокових контрзаходів. Математичні моделі RODOS розроблені більш, ніж в 20 європейських інститутах, використовують інформацію систем станційного радіологічного моніторингу, оперативних метеорологічних прогнозів або сценаріїв розвитку гідрометеорологічної ситуації для прогнозування міграції радіонуклідів у повітряному середовищі, випадінь на водозбори, транспорту радіонуклідів водними потоками та їх міграції в сільськогосподарській продукції, в природних екосистемах для розрахунку доз внутрішнього і зовнішнього опромінення персоналу та населення в зоні аварії.

Спочатку СППР RODOS була спроектована для RISC-серверів і робочих станцій компанії Hewlett Packard, які працюють під управлінням операційної системи UNIX. Система мала реалізацію геоінформаційного модуля, який підтримував лише власний формат даних, складну процедуру інтеграції нових моделей та була важкою при налаштуванні. В міру розвитку інформаційних технологій користувачами системи була поставлена задача створити нову кросплатформену, розподілену систему, яка повинна мати сучасний інтер-

фейс користувача, містити функціональність геоінформаційної системи (ГІС), яка буде працювати із загальнопоширеними форматами ГІС-даних, легко адаптуватися до національних та об'єктових умов, підтримувати роботу з декількома користувачами, пропонувати легкий шлях до інтеграції зовнішніх математичних моделей та ін. Крім того, в систему мали бути інтегровані всі обчислювальні моделі старої системи RODOS [5].

Для СППР, заснованих на математичних моделях, було розвинуто підхід до інтеграції моделей на основі формалізації опису входів та виходів моделей та їх взаємодії з інтерфейсом [6, 7]. При цьому моделі запускаються як автономні програми, обмін інформацією між ними відбувається через файли і можливий тільки до початку або після закінчення роботи моделі. Перевагою цього методу є прискорення розробки системи шляхом розділення роботи між розробниками системи та розробниками моделей, відсутність необхідності будь-яким чином модифікувати код моделі, автоматизована побудова інтерфейсу. Водночас недоліками такого методу інтеграції є практична проблема вивчення нової формалізованої мови розробником моделі тільки для інтеграції моделі до системи, орієнтованість на одноманітний інтерфейс користувача, при якому внесення додаткових компонентів призводить до модифікації формалізованої мови та її інтерпретатора, комунікація моделей та системи тільки через файли чи з використанням платформозалежних засобів (спільна пам'ять, shared memory UNIX), необхідність створювати та читати файли різних форматів.

Шлях інтеграції моделей у сучасних СППР екологічної безпеки на основі новітніх інформаційних технологій з використанням програмного інструментарію у відкритих кодах запропоновано в [8] і реалізовано при розробці системи JRODOS. Загальний опис функціональних властивостей JRODOS, у порівнянні з попередньою версією RODOS, подано в [9]. В представленій статті більш детально описуються архітектура нової системи й інформаційні технології, реалізовані при розробці JRODOS.

2. Архітектура системи

Система підтримки прийняття рішень JRODOS реалізує архітектуру автоматизованих розподілених СППР, заснованих на математичних моделях [8], що спирається на аналіз вимог і прецедентів [10] до нової системи RODOS. Ця архітектура використовує уніфікований тип даних (датаітем) для уніфікації діалогу системи та різних моделей, об'єднання моделей в обчислювальні ланцюги, відображення чисельних даних різного походження (вхідні дані, результати розрахунку моделей, дані моніторингу, прогнозу погоди тощо). Датаітеми – це структура класів, що зберігають чисельні дані різних типів (скалярні величини, масиви, таблиці, службову інформацію), метадані (розмірність, одиниці виміру, субстанцію, проекцію) та відносини між ними [9, 12]. Вони реалізують композитний шаблон програмування, який дозволяє їх об'єднувати у деревовидну структуру та дає можливість системі звертатися до окремих об'єктів і до групи об'єктів однаково.

Система складається з таких компонент: управляючий сервер або менеджер, обчислювальний компонент, клієнт, сервер баз даних. Різні користувачі системи можуть приєднуватися до менеджера за допомогою власної інсталяції клієнта, при цьому кожному користувачу виділяється окремий обчислювальний компонент. Компоненти фізично можуть бути розміщені як на окремих комп'ютерах, так і на одній машині.

Управляючий сервер 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), слід-

кує за станом задач та реагує на їх зміну, передає вихідну інформацію відповідним адресатам (графічному інтерфейсу користувача, геоінформаційному модулю клієнта, базі даних).

Клієнтська частина системи – це орієнтований на користувача компонент, основним завданням якого є представлення інформації користувачу, відображення його дій, збір даних для запитів до менеджера. Клієнт містить графічний інтерфейс, в якому відображаються інтерфейси обчислювальних моделей для збору вхідних параметрів від користувача у зручній для нього формі, валідації введених параметрів, надання рекомендацій та довідок щодо необхідних вхідних даних. Геоінформаційний модуль клієнта призначений для зручного відображення просторово-часових та тематичних даних, які складаються з цифрових моделей місцевості (рівнів висот, категорій землекористування, грунтів, річок та озер, кількості населення, карти доріг, населених пунктів, кордонів адміністративних регіонів тощо), карт розташування об'єктів, що представляють інтерес (станції моніторингу, блоків атомних станцій, будинків та інших), датаітемів з географічною прив'язкою (результатів обчислення, інформації з бази даних, карт прогнозу погоди, вимірів метеорологічних, радіологічних величин). Окрім цього, клієнт містить засоби для побудови звітів та візуалізації різних типів результатів у зручній для користувача формі: простого тексту, масиву, таблиці, гістограми, карти.

Обчислювальний компонент відповідає за ініціалізацію, запуск моделей на виконання та отримання результатів. Він зберігає задачі та разом з ними вхідну і 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 уніфікованих типів даних (датаітемів) до формату, необхідного кожній моделі, забирає та зберігає результати обчислень, трансформуючи їх у датаітеми.

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, над якими можна буде проводити ГІС-операції (перетин декількох географічних об'єктів, визначення лінійних розмірів та ін.), базу фіксованих даних, яка містить таку інформацію, що не змінюється часто (наприклад, характеристики атомних станцій, критичні рівні, параметри радіоекологічних регіонів), а також бази зовнішніх даних, необхідних для роботи системи (наприклад, дані прогнозу погоди, моніторингова інформація). Пізніше до системи можуть бути встановлені додаткові бази даних, в яких інтегровані моделі зберігають необхідну інформацію, параметри, бібліотечні дані.

Зовнішні розрахункові моделі організовані у формі плагінів, 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 плагінів так, щоб інтеграція кожної нової моделі чи внесення змін до вже існуючих моделей не призводила до необхідності коригувати інші компоненти та перевіряти на цілісність систему, тобто підтримувала можливість незалежного розширення.

3. Реалізація СППР JRODOS

Для забезпечення кросплатформеності 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 день це більшість відомих операційних систем, включаючи Windows, різноманітні Linux, MacOS. Крім того, світове Java-товариство забезпечує розробників величезною кількістю бібліотек у відкритих кодах з відповідною ліцензією, які можуть бути ефективно використані у процесі

створення системи. Відкриті коди забезпечують доступність, швидке оновлення та виправлення помилок, можливість модифікації бібліотеки під потреби системи.

Програмна система СППР JRODOS реалізована у вигляді двох основних програмних додатків: JRODOS-сервера та JRODOS-клієнта. JRODOS-сервер об'єднує у собі реалізацію обчислювача та менеджера, JRODOS-клієнт реалізує клієнта СППР. Кожен з програмних додатків JRODOS постачається разом із Java-машиною для відповідної операційної системи, щоб уникнути можливих проблем, таких як конфлікт версій, нестандартні налаштування при використанні системної Java-машини.

Крім основних програмних додатків, реалізовано декілька важливих повторно використовуваних компонентів (ПВК). Ці компоненти входять до складу основних додатків, а також використовуються при створенні плагінів. Основними ПВК є такі:

• Бібліотека уніфікованих типів даних (датаітемів). Датаітеми – це структура класів, що зберігають чисельні дані різних типів, метадані та відносини між ними. За допомогою датаітемів уніфікується діалог системи та різних моделей, стандартизуються способи відображення вхідних параметрів і результатів розрахунку моделей. Ця бібліотека використовується в JRODOS-клієнті для відображення вхідних даних, результатів розрахунку моделей, а також для перетворення вхідних даних різноманітних типів на дерево датаітемів. JRODOS-сервер використовує бібліотеку для процесу поєднання моделей в обчислювальні ланцюги, замінюючи встановлення відповідності виходів попередньої моделі входам наступної моделі операціями з датаітемами. Крім того, в моделях, інтегрованих до системи як плагіни, бібліотека використовується для перетворення уніфікованих типів даних від різних моделей на відповідні параметри вхідних функцій, а також навпаки – вихідні параметри на датаітеми.

• Бібліотека основних абстрактних класів (JRodos-core). Ця бібліотека містить абстрактні класи, що використовуються для побудови плагіна в цілому й основних його частин (графічного інтерфейсу, оболонок обчислювальних моделей, службові класи плагіна) та забезпечення інтегрованості плагіна до системи. Тобто бібліотека постачає абстрактні класи, які в процесі створення плагіна необхідно реалізувати. Таким чином, завдяки механізму наслідування об'єктно-орієнтованої мови програмування Java, функціональність моделі інтегрується до JRODOS-системи. Також бібліотека містить класи-інтерфейси клієнта, менеджера та обчислювача (IClient, IServer та IEngine відповідно), які повинні бути реалізовані у відповідних компонентах системи. Наприклад, будь-який клас, що реалізує інтерфейс IClient, може бути використаний для створення об'єкта, який буде розглядатися менеджером як JRODOS-клієнт. Таким чином, підтримується можливість створення декількох типів клієнтів: традиційний з графічним інтерфейсом користувача, веб-клієнт, клієнт без графічного інтерфейсу (англ. headless).

• Бібліотека функцій доступу до баз даних (DBAccess). Ця бібліотека містить класи для даних, що зберігаються в базі даних, використовуючи технологію об'єктно-реляційної проекції за допомогою бібліотеки Hibernate (http://www.hibernate.org). Окрім класів, зберігається власне інформація по відображенню, а також класи з функціями доступу до об'єктів, збережених у базах даних (data access functions). Методи цих класів формують SQL-запити до бази даних і повертають результат у вигляді екземплярів класів бібліотеки. Бібліотека DBAccess використовується менеджером для організації доступу до баз даних, а решта компонентів системи використовує екземпляри класів даних для різних власних цілей.

• Реалізація основних об'єктів графічного інтерфейсу користувача (GuiTools). Це бібліотека стандартних компонентів графічного інтерфейсу користувача, що можуть бути використані при побудові інтерфейсу моделі. Враховуючи предметну область та досвід інтеграції основних моделей JRODOS, сформовано набір з декількох компонент, що використовуються найчастіше. Бібліотека застосовується при побудові плагіна.

• Реалізація ГІС-функціональності. ГІС-функціональність клієнта [13] винесена в окремий модуль. Таким чином забезпечується можливість відображення географічно прив'язаних даних окремо від JRODOS-клієнта, наприклад, при створенні звітів на сервері в автоматичному режимі.

4. Інтеграція обчислювальних моделей

Запуск зовнішніх обчислювальних моделей є основною вимогою до автоматизованих систем, заснованих на математичних моделях. Необхідно забезпечувати збір, обробку вхідних даних, запуск розрахунку моделі на цих даних, обробку вихідних даних, контроль ходу виконання роботи моделі, перехоплення та трансляцію стандартних потоків виводу в систему. Таким чином, СППР фактично є фреймворком (каркасом [10]) для роботи з різними математичними моделями і повинна забезпечувати чіткий і простий шлях для інтеграції нових моделей і надавати відповідні інтерфейси. При цьому математичні моделі розробляються та тестуються незалежно від системи, використовуючи різні мови програмування, формати вхідних та вихідних даних.

Для системи JRODOS розроблено методику та програмно-інструментальні засоби інтеграції моделей, що дозволяють полегшити процес інтеграції і зробити його більш гнучким. Обчислювальну модель можна інтегрувати як у вигляді динамічних бібліотек, так і у вигляді автономних програм.

При компіляції моделей у вигляді динамічних бібліотек потрібно модифікувати лише функції введення вхідних і виведення вихідних даних таким чином, щоб система і модель спілкувалися безпосередньо, оминаючи файлову систему. Реалізація моделей у вигляді динамічних бібліотек, а не автономних програмних проектів, дозволяє скористатися перевагами концепцій обміну даними за допомогою датаітемів, підтримувати зворотний зв'язок з моделями, контролювати хід виконання розрахунку (наприклад, підрахований крок), отримувати покрокові результати та істотно скоротити обсяг оброблюваної інформації в порівнянні з використанням файлової системи. Однак іноді єдиним шляхом інтеграції є використання автономних проектів, наприклад, якщо вихідні коди і їх розробники недоступні або модель має відповідну ліцензію.

Моделі інтегруються до системи у вигляді плагінів. Плагін розробляється у вигляді окремого Java-проекту на основі шаблону, з використанням створених, повторно використовуваних компонентів (бібліотек) та сторонніх бібліотек у відкритих кодах. Плагін складається з трьох частин, кожна з яких представлена абстрактним класом, що є точкою розширення системи [8].

Клас ModelWrapper (оболонка моделі) реалізує абстрактний клас AbstractModelWrapper і є власне інтерфейсом [14] між системою та обчислювальною моделлю. Він відповідає за комунікацію моделі та обчислювача на загальному, високому рівні. ModelWrapper оперує лише з уніфікованими типами даних. При спілкуванні з системою він генерує повідомлення, змінює стан задачі, приймає вхідний датаітем та формує вихідний. При роботі з моделлю ініціює, стартує та перериває її розрахунок, запитує результати, доповнює їх властивостями, зв'язує з розрахунковою сіткою і стандартними стилями відображення й остаточно формує дерево вихідного датаітема.

За безпосередній зв'язок з обчислювальним ядром моделі відповідає клас EntryPoint, що реалізує абстрактний клас AbstractEntryPoint з ПВК JRodos-core. Об'єкт цього класу створюється в оболонці моделі та відповідає за перетворення вхідних даних з форми датаітемів у прості типи, структури або файли, готові для передачі в модель, організацію запиту даних з додаткових джерел (бази даних, файлової системи), що направляються до ModelWrapper, повернення результатів розрахунку і перетворення їх в датаітеми. EntryPoint відповідальний за виклик функцій з моделі (динамічної бібліотеки) та організа-

цію зворотного зв'язку. В разі інтеграції моделі як автономної програми ця частина плагіна запускає зовнішній процес, перехоплює його потоки виводу та контролює виконання.

Для інтерфейсу користувача необхідно реалізувати методи класу AbstractUserInterface, які будуть обробляти значення з вхідного датаітема (результати розрахунку попередньої моделі ланцюга, значення, збережені під час попереднього запуску, значення за замовчуванням), створювати графічний інтерфейс, обробляти введені в нього дані та відправляти їх у систему у вигляді датаітема. Графічний інтерфейс призначений для збору вхідних параметрів від користувача у зручній для нього формі, перевірки введених параметрів, надання рекомендації та довідок. Верхній контейнер графічного інтерфейсу повинен реалізовувати інтерфейс IView. Він буде відображатися у вигляді закладки в центральній панелі клієнта. Інтегратору моделі доступний ПВК GuiTools, який містить набір компонентів для побудови інтерфейсу. Для більшості моделей послідовності форм достатньо для ефективного збору вхідних даних, однак деякі комплексні моделі використовують геоінформаційні дані від користувача. Для цього розроблений набір часто використовуваних інструментів, доступний у GuiTools. Розширюваність набору інструментів досягнута за рахунок включення абстрактного класу AbstractCustomMapTool, успадкування та спеціалізація якого дозволяє отримати інструмент з необхідною функціональністю.

Поділ оболонки моделей на дві сутності (ModelWrapper і EntryPoint) обґрунтовано вимогою загальної стабільності системи. Якщо під час виконання коду математичної моделі, інтегрованої як динамічна бібліотека, виникнуть не перехоплені виключення, критичні помилки, гілки коду, що призводять до передчасного завершення виконання, то аварійно завершить роботу вся віртуальна машина Java (JVM), в рамках якої виконувалося завантаження та керування бібліотекою. Бібліотека може бути вивантажена з пам’яті лише тоді, коли об’єкт, що завантажує класи (classloader), буде видалено збирачем сміття (garbage collector). Однак витоки пам'яті в моделі накопичуються і не звільняються навіть при вивантаженні бібліотеки.

Таким чином, працювати з моделлю в рамках однієї і тієї ж віртуальної машини, що й основна система, не можна. Для підвищення загальної стабільності системи модель слід запускати у вигляді окремого процесу в окремій віртуальній машині. До того ж такий підхід дозволяє ефективно використовувати сучасні багатоядерні та багатопроцесорні машини, так як розрахунок моделі найчастіше дуже ресурсомісткий та його можна виконувати на іншому ядрі процесора, ніж основна система.

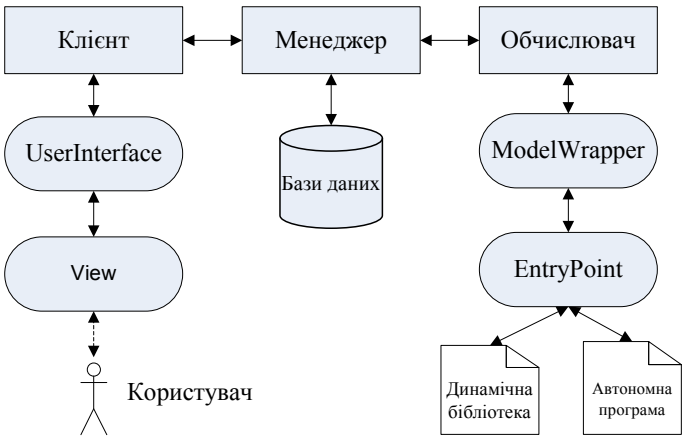

Рис. 1. Схема завантаження та комунікації елементів плагіна на різних компонентах системи. Елементи плагіна (ModelWrapper, EntryPoint, UserInterface, View) реалізують відповідні абстрактні класи та інтерфейси (AbstractModelWrapper, AbstractEntryPoint, Abstract ModelWrapper, IView)

Саме такий порядок виконання є основною причиною поділу Java-оболонки моделі на дві сутності: ModelWrapper та EntryPoint. ModelWrapper виконуються в системній віртуальній машині, а EntryPoint – у новій, розрахунковій JVM. ModelWrapper запускає розрахункову JVM, підтримує з нею комунікацію і, зберігаючи зв'язок з системою, виконує транзитні функції передачі даних до моделі. На рис. 1 наведена схема комунікації системи та елементів плагіна.

Також плагін повинен містити основний клас, який має реалізовувати інтерфейс PluginEntry. Цей інтерфейс описується в бібліотеці основних абстрактних класів і тому відомий як плагіну, так і системі. Інтерфейс PluginEntry містить три функції, що повертають відповідно AbstractModelWrapper, AbstractEntryPoint та AbstractUserInterface. Діаграма класів плагіна з основними методами та атрибути представлена на рис. 2.

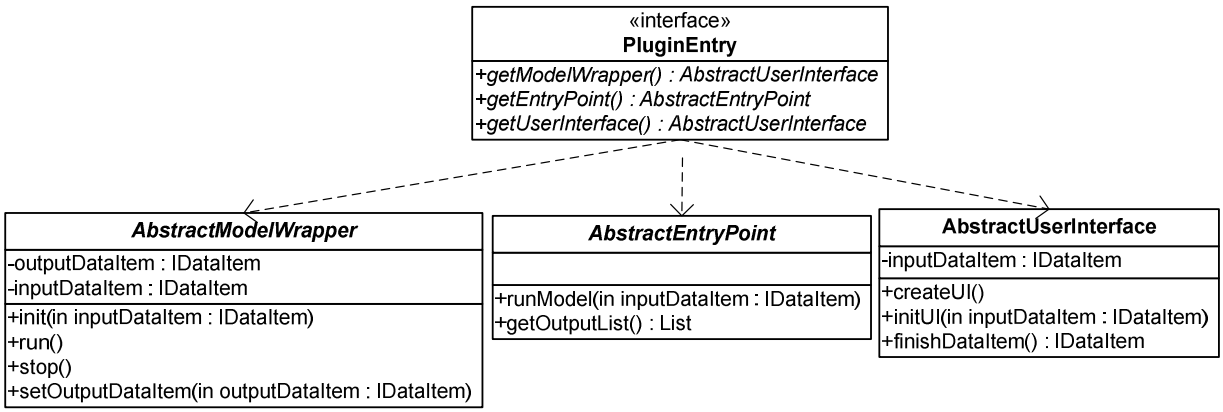

Рис. 2. Діаграма класів плагіна

Щоб система знайшла та правильно розпізнала плагін, вихідний jar-файл, в який компілюється Java-проект, повинен містити спеціальний XML-файл-дескриптор з описом моделі. В файлі повинні зберігатися назва моделі, версія, повна назва основного класу, дерево вхідних даних, що містять значення за замовчуванням та на основі яких будується вхідний датаітем. Зауважимо, що це має бути опис входів оболонки (інтерфейсу) моделі, а не входів моделі. Дерево вхідних даних необхідне для того, щоб при створенні задачі (task) автоматично було створене піддерево вхідних даних. Піддерево вихідних даних формується в класі ModelWrapper.

Плагіни постачаються у вигляді файлових каталогів, які повинні розташовуватись у певному місці відносно кореневого каталогу менеджера. Каталог в обов'язковому порядку має містити jar-файл з файлом-дескриптором, який включає в себе основну функціональність плагіна. Також плагін може містити й необов'язкові компоненти: математичні моделі у вигляді автономних програм чи динамічних бібліотек, локалізаційні файли, стилі відображення, статичні дані-параметри.

5. Менеджер системи

Обчислювач та менеджер реалізовані в одному програмному додатку – JRODOS-сервері. Об'єднання компонент в один додаток зумовлено необхідністю частих комунікацій та передачі великих об'ємів даних саме між цими компонентами. Однак компоненти реалізовані як два окремі незалежні пакети, тобто витримана логічна структура архітектури. Тому далі будемо описувати реалізацію кожної компоненти окремо.

Менеджер системи JRODOS, представлений класом ManagementServer, реалізує інтерфейс IServer з ПВК JRodos-core і складається з планувальника завдань та робочих областей.

Кожному клієнту, що приєднується до системи, виділяється ізольована робоча область (екземпляр класу ClientWorkspace), в якій створюється окремий екземпляр класу обчислювача. Окрім обчислювача, робоча область також містить віддалене посилання на екземпляр класу клієнта, завантажені проекти, чергу викликів методів клієнта та інші службові об'єкти. Якщо до системи приєднається інший клієнт з тими ж параметрами авторизації, то попередній клієнт отримує повідомлення про приєднання нового клієнта і відключається, а вся активність робочої області переключається на нового клієнта. Такий підхід до-

зволяє користувачу за необхідністю працювати з довготривалими розрахунками послідовно з різних робочих місць. Також передбачена можливість приєднання клієнта до активної робочої області без скидання попереднього клієнта та з обмеженою функціональністю, наприклад, клієнт не може реагувати на віддалені виклики (бо посилання на клієнта в робочій області зайнято попереднім клієнтом). Такий спосіб приєднання має сенс при доступі до даних робочої області в режимі тільки для читання.

Клієнт може від'єднатися від системи. В такому випадку всі процеси, пов'язані з робочою областю (обслуговування черги віддалених викликів, розрахунки моделей), перериваються і об'єкт класу ClientWorkspace видаляється. Окрім цього, клієнт може тимчасово від'єднатися від сервера, наприклад, при виконанні будь-якої довготривалої операції. Тоді посилання на об'єкт класу ClientWorkspace залишається активним, і всі процеси продовжуються, крім зворотних викликів методів клієнта. У випадку, коли з клієнтом тимчасово обривається зв'язок, процеси робочої області продовжують виконуватися, лише черга зворотних викликів росте до тих пір, поки не перевищить встановлений ліміт, після чого перші виклики видаляються.

Планувальник завдань використовує бібліотеку Quartz для реалізації черги завдань. Кожне завдання має реалізувати інтерфейс Job. Час, періодичність та кількість виконань визначаються тригерами (класами, що реалізують абстрактний клас Trigger), які надсилаються до планувальника разом із завданням. У зовнішньому конфігураційному файлі можна регулювати кількість потоків обслуговування черги завдань у залежності від наявних ресурсів (тактова частота процесора, кількість ядер, розмір оперативної пам'яті). Завдання надходять до менеджера від клієнта, обчислювача чи з конфігураційного файла при старті системи (системні завдання) і можуть містити різноманітні роботи: створення, збереження, завантаження проектів, ініціалізацію моделей, запуск обчислень, автоматичне завантаження даних вимірювань чи чисельного прогнозу погоди, обслуговування бази даних тощо. Після виконання завдання планувальником завдань виконується виклик методу jobMessage на компоненті системи, з якого надійшло завдання, з відповідними параметрами (результатом та виключенням, якщо воно було викинуто). На обчислювачі jobMessage викликається безпосередньо, а для виклику на клієнті відповідний запит ставиться в чергу викликів методів клієнта в його робочій області.

Менеджер відповідає також за зв'язок з базами даних. Доступ до баз даних організовано за допомогою бібліотеки Hibernate, що реалізує технологію об'єктно-реляційної проекції. Ця технологія зв'язує об'єкти реляційної моделі (відношення, строки та атрибути) та об'єктно-орієнтованої мови програмування (класи, екземпляри, методи, атрибути). Для розробника системи база даних виглядає як сховище об'єктів, і всі операції з базами даних перетворюються на операції з екземплярами Java-класів. Бібліотека створює додатковий шар програмного інтерфейсу (API) між системою та базою даних, який створює схему бази даних, генерує SQL-запити, організовує транзакції. Hibernate звільняє розробника системи від написання великої кількості однотипного коду, в якому є можливість помилитися, чим прискорює процес розробки системи. Водночас додатковий шар іноді генерує неефективні SQL-запити, чим спричиняє повільніше виконання та використання більшого об'єму пам'яті. У випадках, коли сповільнення роботи є критичним для роботи системи (наприклад, при операціях з вимірюваннями), в транзакціях жорстко прописується код SQL-запитів, що передбачено API-бібліотеки.

Власне сама модель об'єктно-реляційної проекції прописана в ПВК DBAccess. Менеджер лише налаштовує Hibernate, зчитуючи інформацію з конфігураційного файла (назва, адреса бази даних, JDBC-драйвер тощо), та перенаправляє запити до баз даних до бібліотеки DBAccess.

6. Обчислювальний компонент системи

Обчислювач системи JRODOS, представлений класом Engine, реалізує інтерфейс IEngine з ПВК JRodos-core. Він містить список завантажених задач. Задача (Task) є реалізацією обчислювальної моделі з конкретними входами, виходами, поточним станом тощо. Датаітеми, що представляють всі вхідні та вихідні параметри, ієрархічно об'єднуються в дерево, кореневий елемент якого належить задачі (рис. 3). Датаітеми зберігають чисельні дані різних типів (скалярні величини, масиви, таблиці, службову інформацію).

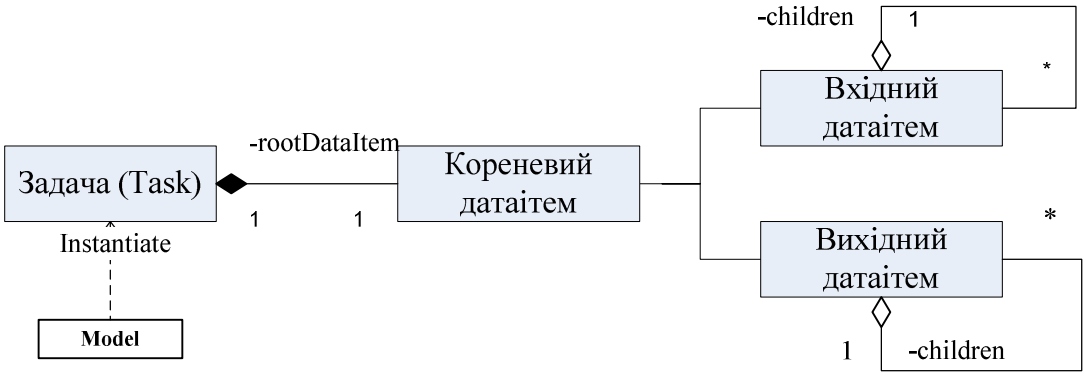

Рис. 3. Структура і відношення задачі та датаітемів

Серед результатів обчислення моделей системи JRODOS значну частину складають табличні дані. При цьому результати на обчислювальній сітці (наприклад, поля концентрації, потужності дози) також представляють собою таблицю, де рядки відповідають обчислювальним коміркам, а стовпчики – моментам часу. Зазвичай більшість комірок обчислювальної сітки не підпадають під забруднення, тому такі таблиці часто є розрідженими. Збереження всієї цієї інформації вимагає великого об'єму пам'яті, при цьому самі дані постійно користувачами не запитуються. Тому для ефективної організації збереження інформації, зменшення використання оперативної пам'яті і для підвищення відмовостійкості системи при одночасному розрахунку декількох моделей застосовуються такі підходи (один з них чи обидва одночасно):

• Дані для окремих моментів часу зберігаються як масив значень, порядок в якому відповідає нумерації комірок сітки. При цьому масив значень стискається як послідовність байтів за алгоритмом ZIP і зберігається в таблиці.

• Стиснений масив значень у вигляді масиву байтів зберігається не в оперативній пам'яті, а в базі даних SQLite (http://www.sqlite.org). SQLite не використовує парадигму клієнт-сервер, тобто не вимагає створення окремого процесу, а надає бібліотеку, з якою програма компілюється, а СУБД стає складовою частиною програми. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному файлі. Кілька процесів або потоків можуть одночасно читати дані з однієї бази, а запис до бази можна здійснити тільки в тому випадку, коли жодних інших запитів у даний момент не виконується. Вибірка та запис даних до SQLite-файла виконується швидко, однак видалення даних може зайняти певний час, крім того, дані фактично не видаляються, а лише помічаються як видалені. Це зумовлює особливості застосування SQLite-файлів для збереження результатів розрахунку.

Кожен SQLite-файл відповідає певній гілці дерева датаітемів, тобто певному комплексному датаітему та всім його нащадкам. У кореневого датаітема задачі завжди є відповідний SQLite-файл. При цьому у файлі зберігаються лише вихідні датаітеми. При ініціалізації та розрахунку моделі старі результати, якщо вони були, зазвичай видаляються. Таким чином, найефективнішим способом очищення SQLite-файла є видалення старого та

створення порожнього файла для збереження нових результатів. Для моделей, які за цикл розрахунку змінюють лише деякі вихідні датаітеми, SQLite-файл створюється для кожного комплексного датаітема, що оновлюється під час розрахунку.

Для операцій з SQLite-файлами розроблений клас RowConnectionPool. Для кожної завантаженої задачі створюється екземпляр цього класу, який зберігається в обчислювачі. При збереженні задачі в базі даних SQLite-файли також поміщаються в базу даних проектів.

Кожній завантаженій до обчислювача задачі виділяється робочий каталог, в який повністю копіюється каталог плагіна. Далі обчислювач шукає jar-файл плагіна і завантажує його класи за допомогою об'єкта класу URLClassLoader. Після цього система має можливість створити об'єкти класів, розташованих у цьому jar-файлі. За допомогою файла-дескриптора з jar-файла визначається основний клас, який повинен реалізувати інтерфейс PluginEntry. Цей інтерфейс описується в бібліотеці основних абстрактних класів і тому відомий як плагіну, так і системі. Отримавши об'єкт цього класу, обчислювач створює об'єкт класу ModelWrapper (оболонка моделі), що уточнює абстрактний клас AbstractModelWrapper та здійснює комунікацію системи й моделі на високому рівні. За безпосередній зв'язок з обчислювальним ядром моделі відповідає клас EntryPoint, який уточнює абстрактний клас AbstractEntryPoint з ПВК JRodos-core.

Для підвищення загальної стабільності системи ModelWrapper виконується в системній віртуальній машині, а EntryPoint – у новій, розрахунковій віртуальній машині Java (JVM). ModelWrapper запускає розрахункову JVM, підтримує з нею комунікацію і, зберігаючи зв'язок з системою, виконує транзитні функції передачі даних до моделі. Таким чином, якщо під час виконання коду математичної моделі, інтегрованої як динамічна бібліотека, виникнуть не перехоплені виключення, критичні помилки, гілки коду, що призводять до передчасного завершення виконання, то аварійно завершить роботу лише розрахункова JVM, в рамках якої виконувалося завантаження та керування бібліотекою, а не системна бібліотека, як було б у протилежному випадку. До того ж такий підхід дозволяє ефективно використовувати сучасні багатоядерні та багатопроцесорні машини, так як розрахунок моделі, найчастіше дуже ресурсомісткий, можна виконувати на ядрі процесора, який не задіяний під основну систему.

Спілкування двох JVM здійснюється за допомогою технології віддаленого виклику методів RMI. Для організації зв'язку використовується бібліотека у відкритих кодах Cajo (http://java.net/projects/cajo/pages/Home). Програмний об'єкт Cajo – це невелика бібліотека, яка організовує ефективне спілкування між віртуальними машинами як на одному фізичному комп'ютері, так і по мережі. Вона надає легкий у використанні і зрозумілий фреймворк, який дозволяє віддаленим JVM працювати разом як єдине ціле. Бібліотека Cajo забезпечує легкий доступ до функціональності багатоадресної розсилки (multicast). По-перше, вона дозволяє об'єктам "транслювати" віддалене посилання на себе мережею. По-друге, вона дозволяє об'єктам "прослуховувати", уловлювати ці посилання й викликати відкриті методи об'єкта, що транслював посилання. Multicast дозволяє об'єктам виявляти і звертатися один до одного спонтанно під час виконання без необхідності статичної реєстрації.

7. Клієнт системи

Клієнт JRODOS системи повинен реалізувати інтерфейс IClient з ПВК JRodos-core. Таким чином, підтримується можливість створення декількох типів клієнтів: традиційний з графічним інтерфейсом користувача, веб-клієнт, клієнт без графічного інтерфейсу (англ. headless). Веб-клієнт надає можливість обмеженого керування системою з вікна браузера. Зокрема, веб-клієнт підтримує візуалізацію дерева проектів, розрахованих даних, повідомлень. Також він підтримує ініціалізацію та запуск деяких моделей на виконання. Headless-

клієнт можна використовувати для тестування, автоматичного розрахунку моделі, створення звітів у фоновому режимі.

Основним у використанні клієнтом є програмний додаток JRODOS-клієнт – повноцінний десктоп-клієнт з графічним інтерфейсом користувача, призначений для представлення інформації користувачу, відображення його дій, підготовки завдань для планувальника завдань менеджера з JRODOS-сервера. JRODOS-клієнт – це Java-програма, що використовує стандартну бібліотеку swing для реалізації графічного інтерфейсу, бібліотеку у відкритих кодах GeoTools (http://geotools.org) для маніпулювання геопросторовими даними та реалізації ГІС-модуля системи.

Комунікація сервера з клієнтом відбувається за технологією RMI, реалізованою у бібліотеці Cajo. Серверний об’єкт "транслює" віддалене посилання на себе мережею. При цьому сервер має бути налаштований таким чином, щоб транслювати посилання саме в ту мережу, в якій будуть запущені клієнти. При старті клієнта користувач вводить параметри авторизації та IP-адресу машини, на якій виконується сервер. Клієнт намагається вловити віддалене посилання на сервер і викликає методи запиту на авторизацію. Далі він надсилає посилання сервера, віддалене посилання на себе, після чого між сервером і клієнтом встановлюється двосторонній зв'язок.

Основне вікно інтерфейсу користувача клієнта складається з таких компонент: меню, панелі швидкого доступу, лівої та правої панелей, центральної панелі з вкладками та панелі повідомлень. Ліва панель вікна використовується для відображення дерева проектів, задач та датаітемів, виклику меню доступних операцій з ними. Центральна панель з вкладками призначена для відображення карти, чисельного значення датаітема, інтерфейсу моделі. Кожен з цих елементів відображається в окремій вкладці центральної панелі. Всі вкладки, що з’являються в центральній панелі, імплементують інтерфейс IView з ПВК JRodos-core. Права панель також представляє собою панель із вкладками для зображення схем звітів, легенди карти, значень атрибутів виділених на карті об’єктів. У панелі повідомлень відображаються будь-які повідомлення, що надходять до системи з клієнта, оболонок моделі, JRODOS-сервера. Вигляд головного вікна JRODOS-клієнта показано на рис. 4.

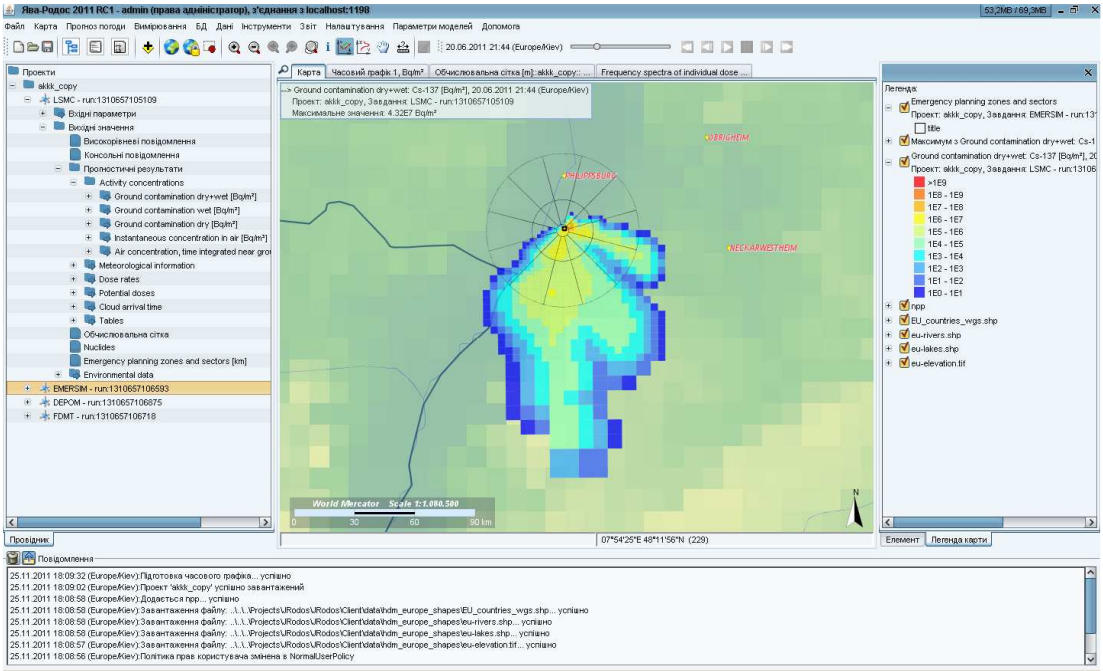

Рис. 4. Головне вікно JRODOS-клієнта із завантаженим проектом, розрахованою картою випадіння Cs-137 при гіпотетичному сценарії аварії на ядерному об’єкті FZK

У лівій панелі розташоване дерево проектів, які користувач створив або завантажив до JRODOS-сервера з бази даних. Для побудови дерева проектів клієнт з менеджера отримує лише метаінформацію по кожному з вузлів дерева (проект, задача, датаітем). Ця метаінформація містить назву вузла, його тип, кількість дочірніх вузлів та ін. Реальні значення датаітема надходять до клієнта і відображаються за запитом користувача.

Кожен датаітем може мати один або декілька візуалізаторів – класів, що генерують компонент з графічним відображенням збережених в датаітемі чисельних значень. При цьому один з них призначається візуалізатором за замовчуванням, а решта – доступні через меню операцій з датаітемами. Ці класи реалізують інтерфейс IView і повертають swing компоненти, що відображається в центральній панелі головного вікна клієнта. Візуалізатори призначені лише для перегляду значень датаітемів і не підтримують функції її редагування.

Редагування значень датаітемів відбувається через виклик інтерфейсу моделі. При завантаженні проекту до менеджера jar-файли плагіна відповідних моделей надсилаються до клієнта, де він, аналогічно до дій обчислювача, завантажує його класи за допомогою об’єкта класу URLClassLoader. Отримавши об’єкт класу PluginEntry, обчислювач створює об’єкт класу UserInterface, що уточнює абстрактний клас AbstractUserInterface.

ГІС-модуль системи JRODOS [13] розділений на дві частини. Клієнтська частина відповідає за представлення даних користувачеві, полегшення процесу прийняття рішення при аварійних ситуаціях або аналіз результатів моделі при використанні системи в дослідницьких цілях. Друга частина, розташована на обчислювальному сервері, дозволяє передавати оброблені ГІС-дані на вхід математичним моделям.

Геоінформаційний модуль JRODOS клієнта призначений для зручного відображення для користувача просторово-часових і тематичних даних, серед яких:

• Цифрові моделі місцевості, що використовуються як карти-підкладки. У систему необхідно завантажити карти висот, категорій землекористування, ґрунтів, населення, адміністративних регіонів, населених пунктів і доріг.

• Карти розташувань об'єктів, що представляють інтерес, таких як блоків АЕС, детекторів, метеостанцій і т.д.

• Карти метеорологічної та радіологічної обстановки, результатів вимірювань і прогнозу погоди.

У JRODOS можна використовувати такі джерела для шарів:

• Файли в форматі ESRI. shp або geoTIFF. Такі файли завантажуються і відображаються цілком.

• Результати просторового запиту до геоінформаційної бази даних PostGIS / PostgreSQL.

• Результати запиту до WMS-сервера, сервера Google Maps, OpenStreetMap, що відправляється при зміні видимої області, масштабу.

• Результати розрахунків моделей, даних вимірювань або чисельного прогнозу погоди. У цьому випадку шари формуються в оперативній пам'яті на основі об'єктів, які містять чисельні дані, і так само метаінформацію про результат (назва, джерело, субстанція).

Розгорнуті в часі дані можна анімувати. В ході анімації змінюються значення атрибутів відповідного шару (наприклад, при зміні значень у клітинках сітки) або орієнтації просторових об'єктів (наприклад, стрілок при вимірі напрямку вітру), не створюючи сам шар заново, що значно покращує швидкодію системи.

ГІС-модуль також має необхідний набір засобів для роботи з шарами, серед яких є можливість зміни масштабу, прокручування видимої області, зміни проекції, вимірювання відстані, виділення об'єктів, перегляду атрибутивної інформації, експорту у векторний формат. Кожен шар у ГІС-модулі системи JRODOS являє собою об'єкт певного класу, що визначає набір операцій, які можливо проводити з шаром. Вбудований редактор стилів до-

зволяє редагувати стилі для векторних шарів, експортувати та імпортувати їх в .sld файли. ГІС-модуль відображається як вкладка «Карта» в центральній панелі клієнта.

8. Бази даних системи

Для збереження інформації система використовує базу даних PostgreSQL (http://www.postgresql.org) зі встановленим геопросторовим розширенням PostGIS (http://postgis.refractions.net). Бази даних можуть бути встановлені як на одній машині з JRODOS-сервером, так і на різних. При першому запуску сервера в СУБД PostgreSQL реєструється користувач jrodos з правами на створення баз даних, таблиць, виконання запитів і т.д., від імені якого в подальшому будуть виконуватися всі дії. Далі автоматично створюються бази даних, якщо вони ще не існують за схемою з ПВК DBAccess, описаною засобами Hibernate, наповнюються початковими даними при необхідності.

Просторове розширення PostGIS дозволяє зберігати в базі даних ГІС-об’єкти, виконувати геопросторову фільтрацію, зберігати результати розрахунків в базі геоданих для прямого доступу до них сторонніми програмами.

Основними базами даних є бази даних проектів, фіксованих даних, прогнозу погоди, вимірювань та користувачів. База даних проектів зберігає інформацію про проекти, задачі та датаітеми (чисельні дані, метадані та відносини між ними), що належать задачам. Враховуючи те, що при видаленні/заміні інформації в комірках таблиці реальна інформація не стирається, а лише помічається як видалена, а також час пошуку по таблиці зростає з кількістю записів в ній, для кожної задачі створюється окрема таблиця для збереження датаітемів. Таким чином, розміри таблиці для задачі обмежені максимальною кількістю датаітемів, що можуть бути створені для конкретної моделі (найчастіше це константа), при перезаписі результатів чи видаленні задачі відбувається видалення таблиці і створення нової за необхідністю. Операція видалення таблиці і створення нової відбувається значно швидше, ніж видалення всіх рядків однієї таблиці. Крім того, при такому підході дані з бази даних фізично видаляються і немає необхідності викликати команду-прибирання (vacuum в PostgreSQL).

9. Висновки

На основі архітектури автоматизованих систем підтримки прийняття рішень [8], заснованих на математичних моделях з використанням розроблених уніфікованого типу даних та програмного інструментарію реалізації, наявного у відкритих кодах, реалізовано нову версію системи Європейського союзу з питань ядерного реагування JRODOS. Впровадження нових інформаційних технологій дозволило у короткий термін розробити оновлену гнучку кросплатформену систему, яка задовольняє усім вимогам Європейської групи користувачів RODOS, що висувались до оновленої версії системи [5]. Система підтримує можливість пристосовуватися до різного апаратного середовища, є розширюваною, кросплатформеною та розподіленою для ефективного розміщення ресурсоємних компонент на більш потужних комп’ютерах.

Розроблено методику та програмні засоби (шаблон проекту, бібліотеки класів) інтеграції зовнішніх обчислювальних моделей до системи на основі уніфікованого типу даних у вигляді незалежних програмних компонент – плагінів. Досягнута більш тісна інтеграція за рахунок використання динамічних бібліотек замість автономних програм, що дозволило підтримувати зворотний зв'язок з моделями, контролювати хід виконання розрахунку, отримувати покрокові результати та істотно скоротити обсяг оброблюваної інформації. За запропонованою технологією на момент написання статті до системи інтегровано 25 різних моделей.

При розробці системи запропоновано декілька підходів, що зменшують обсяг використаної оперативної пам'яті, пришвидшують роботу та забезпечують стабільність, відмовостійкість системи. Апробовано ефективний спосіб комунікації декількох віртуальних Java-програм, що виконуються в одній локальній мережі за допомогою технології RMI з використанням бібліотеки Cajo.

СПИСОК ЛІТЕРАТУРИ

1. Shershakov V.M. Radioecological Analysis Support System (RECASS) / V.M. Shershakov, V.S. Kosykh, R.V. Borodin // Radiation Protection Dosimetry. – 1993. – Vol. 50. – P. 181 – 184.
2. Косых В.С. Интеграция системы поддержки принятия решений в случае радиационных аварий RECASS NT с аналогичными системами, действующими в других странах / В.С. Косых, А.В. Крылова // Проблемы безопасности и чрезвычайных ситуаций. – 2009. – № 2. – С. 128 – 147.
3. Бончук Ю.В. Программный комплекс анализа дозиметрической обстановки при аварийных выбросах АЭС Украины / Ю.В. Бончук, Н.Н. Талерко, А.Г. Кузьменко // Проблеми безпеки атомних електростанцій і Чорнобиля. – 2009. – № 12. – С. 30 – 39.
4. Raskob W. European approach to nuclear and radiological emergency management and rehabilitation strategies (EURANOS) / W. Raskob // Kerntechnik. – 2007. – Vol. 72, N 4. – P. 172 – 175.
5. 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.
6. Гофман Д.С. Застосування програмно-інструментальної системи LIANA для інтеграції прикладних задач, ГІС і баз даних у системи підтримки прийняття рішень, заснованих на моделях / Д.С. Гофман // Математичні машини і системи. – 1998. – № 1. – С. 75 – 88.
7. Integration of external programs in RODOS [Електронний ресурс] / G. Benz, A. Lorenz, M. Rafat [et al.] // Report RODOS TN(1)9601. – Germany, Karlsruhe, Forshungszentrum Karlsruhe, IKET, 2001. – 91 p. – Режим доступу: http://www.rodos.fzk.de.
8. Євдін Є.О. Розробка архітектури кросплатформних розподілених систем підтримки прийняття рішень, основаних на математичних моделях / Є.О. Євдін // Математичні машини і системи. – 2011. – № 1. – С. 72 – 81.
9. RODOS reengineering: aims and implementation details / Ie. Ievdin, D. Trybushnyi, M. Zheleznyak [et al.] // Radioprotection. – 2010. – Vol. 45, N5. – P. 181 – 189.
10. Об'єктно-орієнтоване моделювання при проектуванні вбудованих систем і систем реального часу: [навч. посібник] / В.В. Литвинов, С.В. Голуб, К.М. Григор'єв [та ін.]. – Черкаси: Вид. від. ЧНУ ім. Богдана Хмельницького, 2011. – 376 с.
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. 2984/2004. – P. 129 – 143.
12. 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. 2657/2003, 0302-9743. – P. 35 – 44.
13. Евдин Е. Разработка ГИС модуля Европейской системы поддержки принятия решений при радиационных авариях РОДОС / Е. Евдин, Д. Трибушный, М. Железняк // Ученые записки Таврического национального университета им. В.И. Вернадского. – (Серия: География). – 2010. – Т. 23 (62), № 2. – С. 66 – 71.
14. Лаврищева Е.М. Интерфейс в программировании / Е.М. Лаврищева // Проблеми програмування. – 2006. – № 2. – С. 126 – 139.

Стаття надійшла до редакції 27.12.2011