УДК 519.86:574

В.В. Войтенко, асист., аспір.
А.О. Гладишевський, магістрант
Житомирський інженерно технологічний інститут

РОЗПОДІЛЕНА АРХІТЕКТУРА СЕРЕДОВИЩА ОБ'ЄКТНИХ МОДЕЛЕЙ ТА ЇЇ ЗАСТОСУВАННЯ ДЛЯ КОМП'ЮТЕРНОЇ РЕГІОНАЛЬНОЇ ЕКОЛОГІЧНОЇ СИСТЕМИ

Розглянуто проектування розподіленої архітектури середовища об'єктних моделей OMEGA (Object Models Environment General Architecture) та можливість її застосування для комп'ютерної регіональної екологічної системи. Головною особливістю архітектури OMEGA є забезпечення існування та взаємодії об'єктів у єдиному об'єктному просторі.

Екосистема: математична модель

На найнижчому рівні структуризації екосистема – двовимірний простір, кожна точка якого характеризується множиною властивостей (параметрів) та відношень між ними. Різні групи властивостей можуть об'єднуватись, утворюючі підсистеми. Для властивостей, що змінюються в просторі, існує поняття територіального розподілу. Є змінні (ті, що динамічно змінюються в процесі моделювання) та постійні властивості. Крім того, для деяких властивостей з територіальним розподілом характерною є незмінність на деяких обмежених частинах території – регіонах.

Відношення визначають динамічну зміну властивостей. Серед відношень розрізняють горизонтальні та вертикальні. Горизонтальне – це відношення на просторі однієї властивості з територіальним розподілом. Горизонтальні відношення визначають зміну територіального розподілу властивості у часі незалежно від інших властивостей. Вертикальне відношення – це відношення між властивостями однієї точки простору екосистеми. Воно може мати місце на всій території екосистеми або на території окремих регіонів.

Математична модель екосистеми має такий вигляд:

$$S = (t,\ C,\ L,\ M,\ E),$$

де t – це вільний параметр, значення якого відповідає номеру поточного кроку моделювання. Множина значень параметра – множина цілих чисел ($t \in Z$). В процесі моделювання його значення послідовно змінюється в діапазоні $[t^0; t^{N-1}]$, де N – кількість кроків моделювання. "Послідовно змінюється" означає, що $t^{k+1} = t^k + 1$.

$C = \{c_0 \dots c_{N_c-1}\}$ – множина параметрів, що не змінюються в просторі, тобто не мають територіального розподілу. Ця множина завжди містить щонайменше два параметри – розміри території екосистеми D_w та D_h.

$L = \{L_0 \dots L_{N_l-1}\}$ – це множина матриць розподілу – шарів. Кожна матриця L_k визначає територіальний розподіл одного з параметрів, що змінюється в просторі. Елементами матриць є дійсні числа ($l_{ij}^k \in R$).

$M = \{M_0 \dots M_{N_m-1}\}$ – множина карт. Карта – це матриця, що визначає деякий регіональний розподіл території. Кожна клітинка матриці M_k (m_{ij}^k) містить позитивне ціле значення в діапазоні $[0 : N_k - 1]$, що є індексом регіону та визначає, до якого регіону належить відповідна точка простору екосистеми. Таким чином, карта M_k визначає множину регіонів $\{R_k^0 \dots R_k^{N_k-1}\}$. Регіон – це множина пар вигляду $(i,\ j)$, тобто деяка підмножина точок території. Регіон визначається як $R_k^s = \{(i,\ j) : m_{ij}^k = s\}$.

Усі матриці з множин L та M мають однакові розміри: D_h рядків та D_w стовпців.

$E = \{E_1 \dots E_{N_e}\}$ – множина відношень. В найзагальнішому вигляді вона визначає відображення $E^t : \{S^0 \dots S^t\} \to S^{t+1}$, тобто залежність стану моделі від її стану на попередніх кроках. Зауважимо, що множина відношень в загальному випадку є динамічною, тобто може змінюватись в процесі моделювання. Відношення можна розділити на декілька основних груп:

1. Відношення, що визначають пряму залежність параметра від кроку: $t \to \left(C^t,\ L^t\right)$.
2. Горизонтальні відношення: $\left(C^t,\ L_k^t\right) \to \left(l_{ij}^k\right)^{t+1}$. Такі відношення задаються у формі матричних фільтрів.
3. Вертикальні відношення: $\left(C^t,\ H_{ij}^t\right) \to \left(l_{ij}^k\right)^{t+1}$. Тут H_{ij} – це множина значень клітинок усіх матриць L, що мають координати $(i,\ j)$, тобто $H_{ij} = \left\{l_{ij}^k,\ k \in [1 : N_l]\right\}$.

Область дії будь-якого з цих відношень може територіально обмежуватись деяким регіоном R_s^k або розповсюджуватись на усю територію екосистеми. Крім того, область дії відношення може обмежуватись деяким періодом в просторі параметра t, тобто відношення може діяти при $t \in [t_1;\ t_2]$, де t_1, t_2 – відповідно початковий та кінцевий кроки періоду.

Реалізація математичної моделі на основі матричного процесора

Вище була описана математична модель екосистеми. Для її реалізації в системі екологічного моделювання Ecos 2.0 використовується об'єкт, що представляє собою спеціалізований матричний процесор. Матричний процесор (МП) – це система, що складається з таких складових (об'єктів):

1. Змінна t, що містить номер поточного кроку моделювання.
2. Набір простих параметрів (відповідає множині C). Сюди входять усі параметри, що не мають територіального розподілу. На логічному рівні ці параметри можуть утворювати масиви (одно- та багатовимірні).
3. Набір матриць територіального розподілу (відповідає множині L). Кожна матриця називається *шар* (layer).
4. Набір карт (відповідає множині M).
5. Набір алгебраїчних виразів (відповідає множині E). Вирази представлені у формі зворотного польського запису і можуть виконуватись матричним процесором.

Кожний параметр, шар, карта або вираз адресується індексом, унікальним в межах кожного класу об'єктів.

Інтерфейс матричного процесора дозволяє виконувати такі дії:

1. Створювати загальну структуру екологічної моделі, тобто задавати розмір території екосистеми, кількість параметрів, шарів, карт та відношень.
2. Задавати початковий стан екосистеми: значення параметрів, конфігурацію регіонів, встановлювати відношення у вигляді алгебраїчних виразів.
3. Обчислювати вирази на кожному кроці моделювання, що призводить до зміни стану системи.
4. Отримувати інформацію про стан екосистеми в будь-який момент в процесі моделювання.

Екологічна модель

Екологічна модель в Ecos 2.0 є системою об'єктів, і матричний процесор – ядро цієї системи. Структура повної екологічної моделі на основі МП показана на рис. 1.

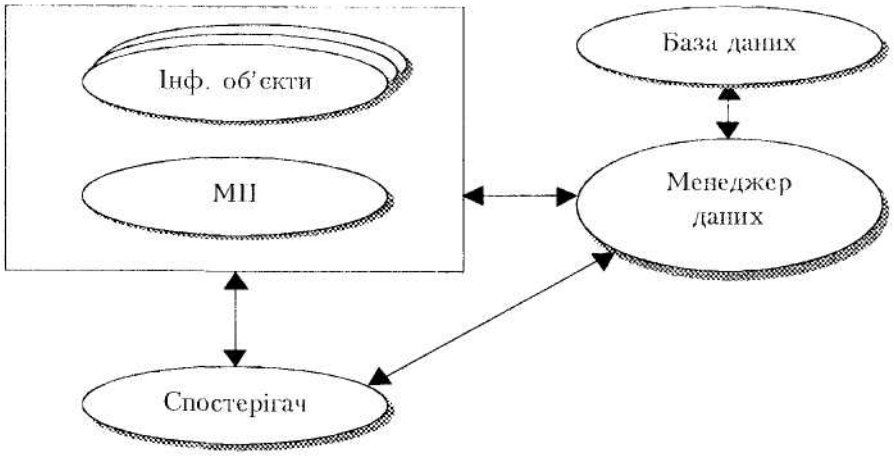

Рис. 1

Як можна побачити, в модель, крім МП, входять декілька нових об'єктів. По-перше, модель власне екосистеми розширена з використанням *інформаційних об'єктів*. Вони містять дані, що не беруть участі в обчисленнях, але є важливими для сприйняття стану моделі. Такою інформацією є, наприклад, назви регіонів, окремих параметрів (одиниці виміру, назви характеристик) тощо.

База даних зберігає початкові дані (матриці розподілу, карти тощо) і результати моделювання.

Менеджер даних – це об'єкт, що виконує передачу даних з *Бази даних* в *Матричний процесор* і навпаки. Інтерфейс менеджера даних дозволяє вказати, які саме дані потрібні для тієї чи іншої конкретної моделі.

Важливим кроком було введення до складу моделі об'єкта *Спостерігач*. Він є абстрактною моделлю людини або програми. *Спостерігач* є ініціатором процесу моделювання, він може отримувати інформацію про стан моделі в процесі імітації, а також впливати на хід процесу.

З'ясувалося, що створені нами раніше системи екологічного моделювання (KEIC, Ecos 1.0) не можуть бути використані для вирішення поставленої задачі. Так, з їх допомогою можна приблизно реалізувати описану вище структуру моделі. Але проблема полягає в тому, що ці системи розраховані на одного користувача та роботу на локальній машині.

Система моделювання, придатна для вирішення поставленої задачі, повинна, як мінімум, забезпечувати інтерфейс з Web-сервером. Це дасть можливість користувачу отримувати доступ до моделі через свій Web-браузер.

Однак треба враховувати те, що Web-інтерфейс дає досить обмежені можливості для взаємодії з моделлю. Тому система моделювання повинна забезпечувати програмний інтерфейс, що дасть можливість власним програмним засобам користувача отримувати прямий доступ до моделі через мережу Інтернет. Крім того, звичайно, система має бути здатна обслуговувати одночасно більше ніж одного користувача.

Нижче описана архітектура, яка дозволяє створити систему, що відповідає усім поставленим вимогам.

Архітектура OMEGA

OMEGA (Object Models Environment General Architecture) представляє собою загальну архітектуру середовища об'єктних моделей. Головною метою архітектури OMEGA є забезпечення існування та взаємодії об'єктів у єдиному об'єктному просторі.

Основними принципами архітектури є:

1. Відкритість.

   Програмний інтерфейс системи дозволяє іншим системам (невідомим під час створення) використовувати її можливості.

2. Розподіленість.

   Можливість розташування різних об'єктів на різних машинах у локальній мережі або мережі Інтернет.

3. Прозорість.

   Механізм взаємодії об'єктів не залежить від їх реального місцеперебування.

4. Універсальність.

   Можливість створення систем будь-якого ступеня складності.

Архітектура OMEGA визначає також модель *метаядра* – базової системи, що реалізує описані вище цілі та принципи.

Основні поняття

Центральним поняттям архітектури OMEGA є об'єкт – підсистема, що характеризується деяким інтерфейсом. Взагалі, об'єкт може мати декілька інтерфейсів, але при розгляданні факту взаємодії двох об'єктів в певний момент часу завжди має значення лише один з інтерфейсів – той, через який відбувається взаємодія. Можна сказати, що один об'єкт "бачить" інший одночасно лише з однієї "сторони", хоча "сторін" може бути й більше. Тому надалі поняття *об'єкт* та *інтерфейс* часто можуть мати синонімічне значення.

Перед тим як визначити, що таке інтерфейс, необхідно розкрити механізм взаємодії об'єктів. Об'єкти взаємодіють між собою шляхом передачі повідомлень. Повідомлення є послідовністю значень різних типів (базові типи стандартизовані). В OMEGA прийнятий

клієнт-серверний механізм взаємодії. Це означає, що в одному сеансі взаємодії беруть участь два об'єкти: *клієнт* та *сервер*. Клієнт передає серверу повідомлення-запит, після чого сервер передає клієнту повідомлення-відповідь. В сеансі взаємодії використовується один з інтерфейсів об'єкта-сервера.

Тепер можно дати визначення інтерфейсу. *Інтерфейс* – це, з одного боку, "точка взаємодії" з об'єктом-сервером, а з іншого – угода, що регламентує синтаксис та семантику запиту та відповіді.

Метаядро

На рис. 2 схематично зображено найбільш абстрактну модель взаємодії об'єктів в OMEGA.

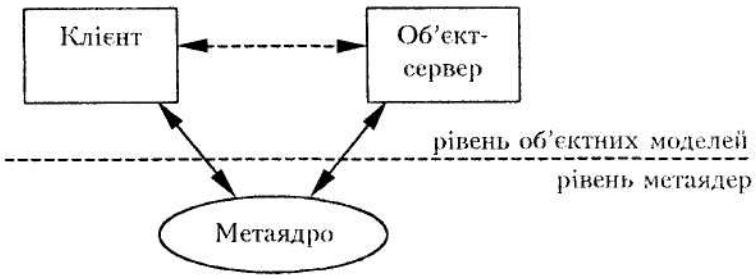

Рис. 2

На рівні об'єктних моделей об'єкти взаємодіють між собою безпосередньо та реальне місцезнаходження клієнта і серверу не має значення. Це відповідає принципу прозорості. Насправді об'єкти взаємодіють не безпосередньо, а через метаядро. *Метаядро* – це абстракція, що відома також під назвою ORB (Object Request Broker – брокер об'єктних запитів). OMEGA визначає метаядро як підсистему, що забезпечує існування та взаємодію об'єктів згідно з принципами архітектури.

Нижче викладаються принципи та технології, що використовуються для створення метаядра згідно з архітектурою OMEGA.

Об'єктний простір. Ідентифікатори

Об'єктний простір в OMEGA – це множина усіх об'єктів у системі. Йому відповідає *ідентифікаційний простір* – одновимірний простір об'єктних ідентифікаторів. Ідентифікатор представляє собою 32-розрядне значення, що однозначно визначає інтерфейс деякого об'єкта. Метаядро здійснює відображення об'єктів у простір ідентифікаторів. Значення ідентифікатора ніяк не визначає фізичного положення об'єкта у системі. Ідентифікатор є лише точкою призначення для повідомлень, що передаються відповідному об'єкту.

Віртуальні канали

Для реалізації принципу прозорості при взаємодії об'єктів в OMEGA введено поняття *віртуального каналу взаємодії*. Віртуальний канал – це об'єкт, якому відповідає ідентифікатор в ідентифікаційному просторі. Цей об'єкт створюється клієнтом під час сеансу взаємодії з об'єктом-сервером. Його завдання – забезпечити передачу повідомлень між клієнтом та сервером. Віртуальний канал реалізує двоспрямований потік значень: до одного з кінців потоку має доступ клієнт, до іншого – сервер. Цей канал є "персональною лінією" – одночасно ним можуть користуватися тільки один клієнт і один сервер.

Модульність. Платформа реалізації об'єктів IMP

Модульна структура системи є необхідною умовою для реалізації принципів відкритості та розподіленості. Концепція модульності в OMEGA грунтується на поняттях об'єкта та об'єктного простору. Модуль – це завершена неподільна структурна одиниця системи. Модуль реалізує деяку підмножину (підсистему) об'єктів складної системи. Взаємодія між модулями можлива лише через інтерфейси, що вони реалізують. Форма взаємодії – двоспрямовані байтові потоки. Один модуль може реалізовувати декілька інтерфейсів. Таким чином, модуль – це об'єкт у просторі операційної системи: програма, процес тощо. Аналогом віртуальних

каналів тут є реальні канали міжпроцесорного та міжмашинного зв'язку, що реалізуються операційною системою.

В архітектурі OMEGA введено поняття *IMP* (*object IMplementing Platform* – платформа реалізації об'єктів). Тоді як модуль є об'єктом фізичної структури системи (на рівні операційної системи), IMP є логічною абстракцією. IMP розкладається при розділі системи на дві частини (рис. 3).

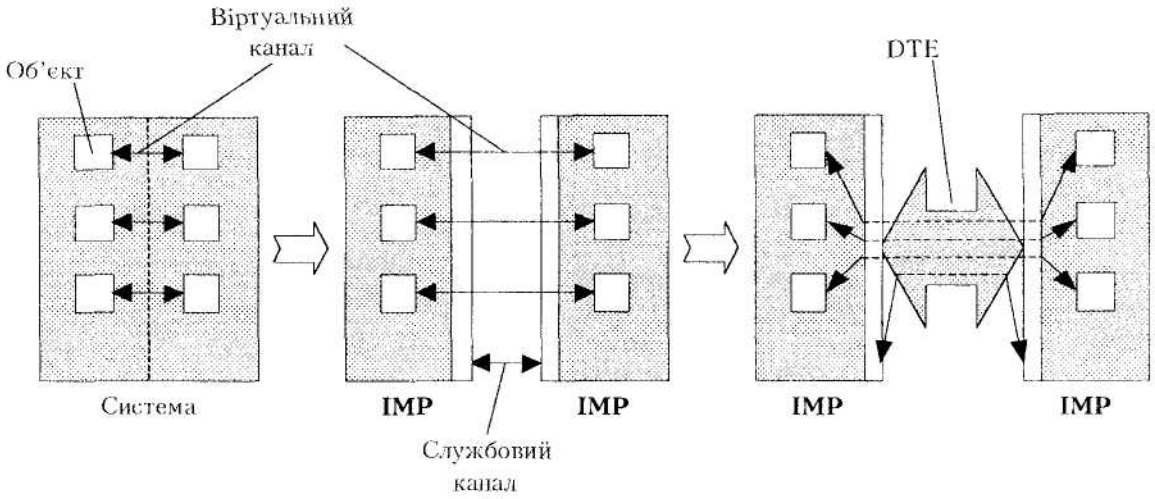

Рис. 3

На рис. 3 видно, що після поділу системи частина об'єктів знаходиться в межах одного IMP, а частина – в межах іншого. Віртуальні канали, що їх з'єднують, перетинають межі підсистем. *Службовий канал* створюється для взаємодії двох IMP через інтерфейс *iMapper* (про це згодом). Віртуальні канали об'єднуються шляхом розбиття віртуальних потоків на пакети (добре відома технологія), що передаються через *транспортне середовище* (*DTE – Data Transport Environment*). Транспортне середовище – це абстракція того ж рівня, що й IMP. Архітектура OMEGA визначає його як двоспрямований потік пакетів довільного розміру. Вимоги до DTE та формат пакетів визначаються протоколом *ICP* (*IMP Communication Protocol*).

В цілому IMP представляє собою об'єкт зі складним інтерфейсом, який складається з інтерфейсу iMapper та інтерфейсів усіх об'єктів, що знаходяться в межах даного IMP. Цей складний інтерфейс має назву *інтерфейс ICP*.

Отже, будь-яку систему в OMEGA можна подати як два IMP, що взаємодіють через DTE. Це *двостороння симетрична модель розподіленої системи*. Згодом буде показано, як ця модель використовується для побудови систем будь-якого ступеня складності.

Симетрична модель вимагає введення деяких нових понять. *Протилежним* називається IMP, що бере участь у сеансі взаємодії з даним IMP. *Локальний об'єкт* – цей об'єкт, що знаходиться в межах даного IMP. *Зовнішній об'єкт* – це об'єкт в межах протилежного IMP. Коли мова йде про конкретний об'єкт, тоді IMP, в межах якого він знаходиться, називається *сервером*, а протилежний IMP – *клієнтом*.

Подвійна ідентифікація

Кожний IMP утворює свій окремий ідентифікаційний простір. З першого погляду, це протирічить основній цілі OMEGA: створити єдиний об'єктний простір. Але насправді це не так.

Основна ідея подвійної ідентифікації полягає в тому, що кожний IMP по-своєму здійснює відображення об'єктів у простір ідентифікаторів. Причому відображаються не тільки локальні, але й зовнішні об'єкти (частина або усі). Таким чином, один об'єкт може існувати одночасно в двох ідентифікаційних просторах, де йому можуть відповідати різні ідентифікатори (рис. 4,*а*).

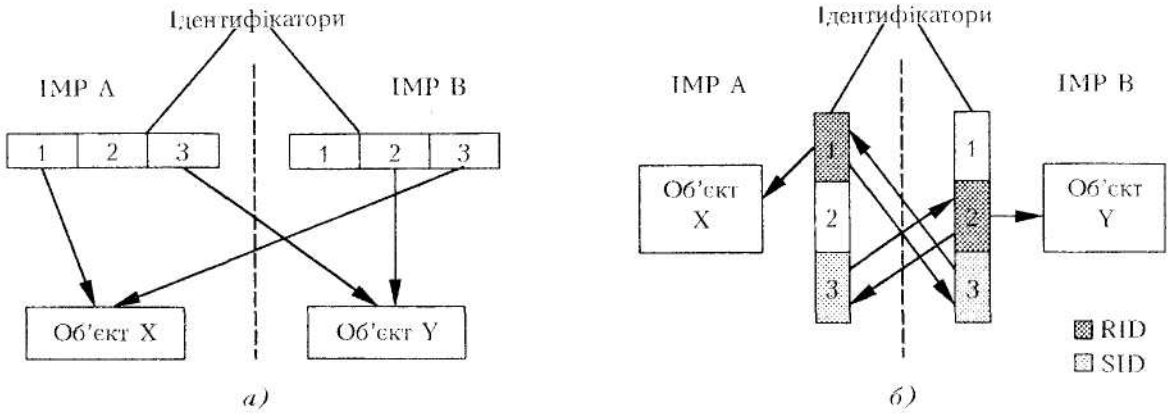

Рис. 4

Для однозначної ідентифікації всіх об'єктів в системі, тобто утворення єдиного об'єктного простору, здійснюється взаємне відображення ідентифікаційних просторів. Кожний IMP визначає відображення зі свого боку (рис. 4,*б*).

В результаті всі ідентифікатори поділяються на два типи: ті, що відповідають локальним об'єктам (*real* ідентифікатори, *RID*) і ті, що відповідають зовнішнім об'єктам (*stub*-ідентифікатори, *SID*).

Коли відбувається передача пакета, той IMP, що відправляє пакет, здійснює в ньому перекодування всіх ідентифікаторів у відповідні ідентифікатори протилежного IMP.

Експорт об'єктів. Інтерфейс iMapper

Відображення локального об'єкта в простір протилежного IMP називається *експортом* об'єкта. В результаті експорту в об'єктному просторі з'являються ідентифікатори SID. Під час експорту об'єкта IMP, що його містить, фігурує як *сервер*, а протилежний IMP – як *клієнт*.

Процес експорту об'єкта складається з таких кроків:

1) сервер посилає клієнту повідомлення, в якому вказує ідентифікатор (RID) експортованого об'єкта;
2) клієнт, отримавши повідомлення, створює у своєму просторі ідентифікатор SID і встановлює відповідність SID → RID;
3) клієнт передає серверу відповідне повідомлення, в якому вказує SID;
4) сервер, отримавши відповідне повідомлення, встановлює відповідність RID → SID. Процес завершено.

OMEGA визначає інтерфейс iMapper, що формально регламентує описаний вище процес експорту об'єктів. Повідомлення згідно з цим інтерфейсом передаються через той самий *службовий канал* (див. вище).

Побудова складних систем. Спеціальні модулі майстер та шлюз

Архітектура OMEGA визначає два стандартних спеціальних модуля, що служать для побудови складних розподілених систем: *майстер* і *шлюз*.

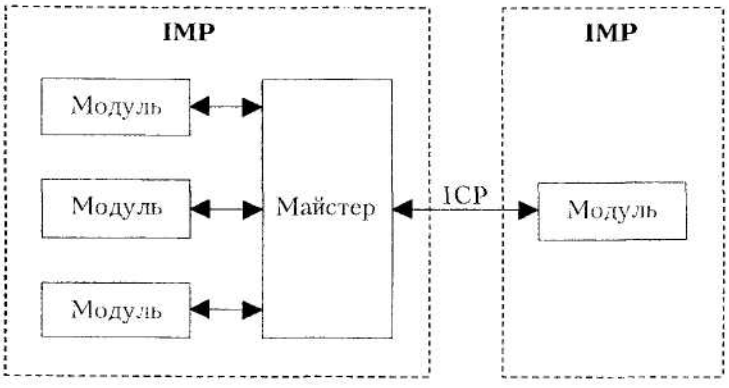

Рис. 5

Майстер – це модуль, що має логічно необмежену кількість інтерфейсів ICP. Це означає, що одночасно майстер може взаємодіяти з декількома іншими модулями.

На рис. 5 зображено як система, побудована на основі майстра, узгоджується з двосторонньою симетричною моделлю. *Будь-яку* пару модуль-майстер можна розглядати як систему з двох IMP. Один IMP – це модуль, інший – майстер і всі інші підключені модулі.

Функції майстра мінімальні: він є маршрутизатором пакетів. Пакети, що отримує від модулів, він відправляє іншим модулям. Для однозначної ідентифікації об'єктів майстер реалізує простір *централізованих ідентифікаторів* (*CID*). Для кожного підключеного модуля майстер реалізує окреме відображення простору CID у простір модуля (рис. 6).

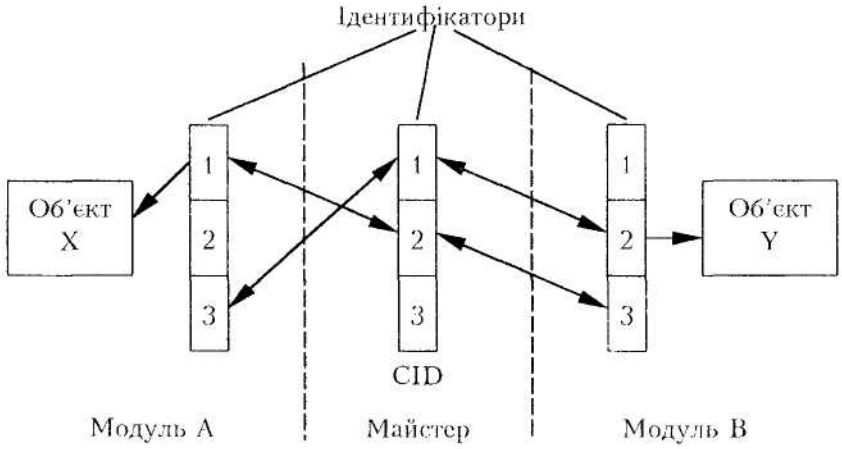

Рис. 6

Процес маршрутизації можна розглянути на прикладі передачі пакета з модуля А об'єкту Y, що знаходиться в модулі В.

1) Пакет створено в модулі А, точка призначення дорівнює SID об'єкта Y (SID = 3).
2) Пакет передається майстру. Перед цим відбувається перекодування ідентифікаторів, після чого точка призначення дорівнює CID об'єкта Y (CID = 1).
3) Майстер, отримавши пакет, визначає, який модуль є сервером об'єкта, що вказаний в пакеті як точка призначення. Це модуль В.
4) Майстер здійснює перекодування ідентифікаторів у простір модуля В і передає пакет цьому модулю.

Шлюз – це модуль з двома інтерфейсами ICP. Шлюзи використовуються для побудови систем з більш ніж одним майстром. Річ у тім, що майстер – пасивний модуль і не вступає у взаємодію з іншими модулями по своїй ініціативі. Шлюз організує прозору взаємодію між двома майстрами (рис. 7).

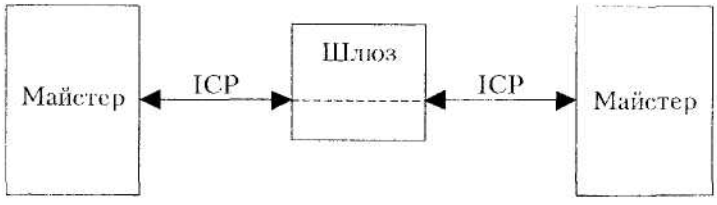

Рис. 7

Основна функція шлюзу – передавати без змін одному майстру пакети, отримані від іншого, і навпаки. Шлюз не має свого об'єктного простору, отже, не здійснює перекодування ідентифікаторів.

Використовуючи спеціальні модулі, можна формувати системи з необмеженою структурною складністю. На рис. 8 зображено систему з двома майстрами та чотирма звичайними модулями.

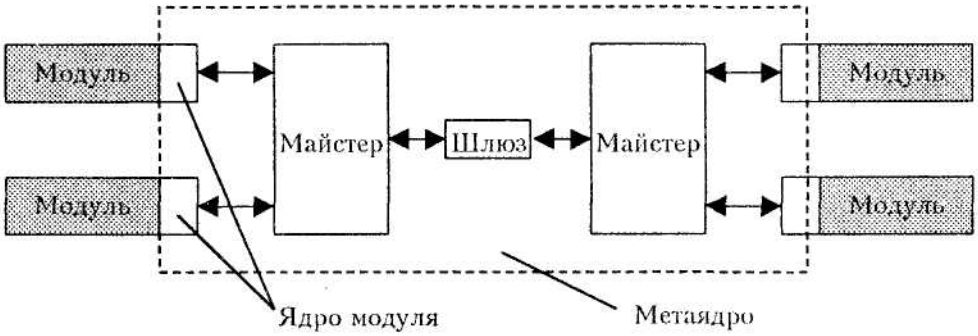

Рис. 8

Пунктирною лінією на рисунку позначене метаядро. Можна побачити, що частина кожного модуля входить до складу метаядра. Цей компонент метаядра має назву *ядро модуля*. Він реалізується у вигляді бібліотеки. Наприклад, у даній реалізації для Win32 ядро модуля реалізоване у вигляді DLL. Інтерфейс між ядром та рештою модуля не є об'єктом стандартизації OMEGA і визначається виключно реалізацією ядра.

Реалізація екологічної моделі

Повернемося до нашої екологічної моделі. Враховуючи усі висунуті вимоги, система виглядає так, як зображено на рис. 9.

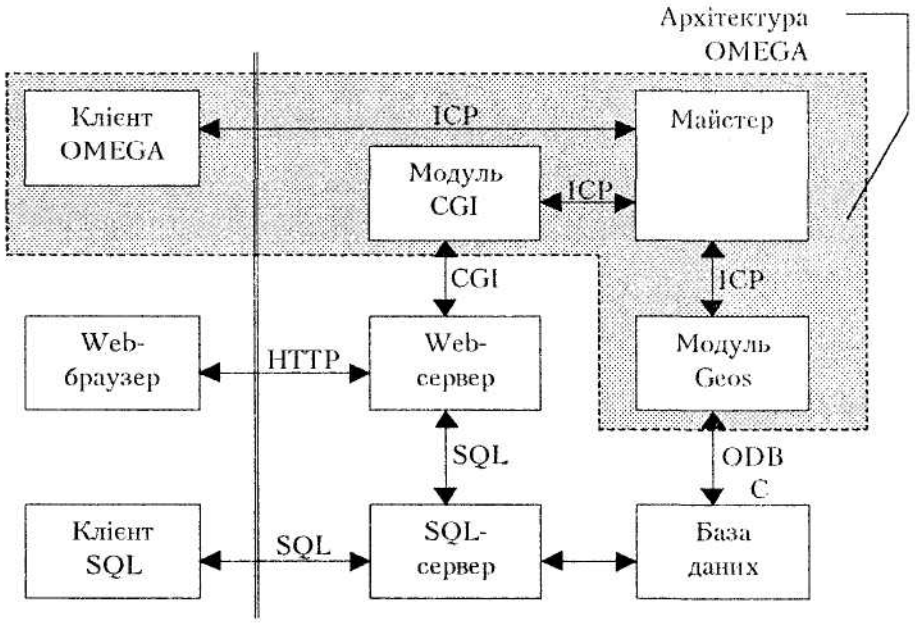

Рис. 9

Модуль Geos реалізує основну частину екологічної моделі: матричний процесор та менеджер даних.

Модуль CGI виконує такі функції:

1) створює одну з конкретних моделей екосистеми на основі МП, що міститься в модулі Geos;
2) здійснює керування процесом моделювання та отримує дані про стан моделі згідно з командами користувача, що поступають через Web-інтерфейс, тобто реалізує об'єкт *Спостерігач*;
3) перетворює дані про стан моделі у форму, зручну для представлення засобами Web (текст, карти, графіки тощо).

Модуль CGI запускається Web-сервером як CGI-програма. Таким чином, у системі одночасно може бути декілька таких модулів і склад їх постійно змінюється.

Клієнт SQL використовується користувачем сумісно з Web-браузером. Він необхідний, якщо користувачу потрібно внести в базу початкові дані для моделювання, що не можуть бути передані засобами Web-браузера (наприклад, карти). Також SQL використовується для отримання результатів моделювання в чисельному вигляді.

Клієнт OMEGA – це OMEGA-підсистема (IMP) на стороні користувача. Вона використовується, якщо можливостей Web-інтерфейсу не вистачає для вирішення поставлених задач. Наприклад, Web-інтерфейс забезпечує доступ до моделей, що реалізуються існуючими CGI-модулями, але не дає засобів для створення нових моделей. У цьому випадку користувач може застосувати клієнт OMEGA для створення власної екологічної моделі на основі МП.

ЛІТЕРАТУРА:

1. *Бейко І.В., Войтенко В.В., Загородній Ю.В.* Об'єктно-орієнтований метод проектування складних екологічних систем // Вісник ЖІТІ, 1997. – № 5. – С. 74–77.
2. *Войтенко В.В., Загородній Ю.В.* Створення та використання комп'ютерної екологічної інтелектуальної системи КЕІС // Вісник ЖІТІ, 1997. – № 6. – С. 160–163.
3. *Войтенко В.В., Загородній Ю.В.* Використання екологічної системи КЕІС у дослідженні впливу фітовірусів на рослинний організм в умовах екологічної нестійкості на території північних районів Житомирської області // Вісник ЖІТІ, 1998. – № 8. – С. 163–169.
4. *Войтенко В.В.* Об'єктні технології. Реалізація методу об'єктно-орієнтованого проектування на прикладі рішення екстремальних задач під MS Windows // Вісник ЖІТІ, 1996. – № 4. – С. 95–98.
5. *Пененко В.В., Алоян А.Е.* Модели и методы для задач охраны окружающей среды. – М.: Наука, 1985. – 256 с.
6. *Пухов Г.Е.* Теория и применение моделирующих систем. – К.: Наукова думка, 1986. – 279 с.
7. *Робертс Ф.С.* Дискретные математические модели с применениями к социальным, биологическим и экологическим задачам. – М.: Наука, 1986. – 496 с.
8. *Сергеев Ю.Н.* Методические аспекты конструирования имитационных моделей экологических систем. – Л.: ИЛУ, 1980. – 20 с.
9. *Zagorodni Yu., Voytenko V., Boyko A.* Studying of the influence of phytoviruses on Plants' organism in conditions of ecological instability and its estimation by mathematical modelling in CEIS system // Process modelling, Berlin, Springer, 1999. – P. 86–98.

ВОЙТЕНКО Володимир Володимирович – асистент кафедри ПЗОТ Житомирського інженерно-технологічного інституту, аспірант Київського національного університету імені Тараса Шевченка.

Наукові інтереси:

– об'єктно-орієнтоване проектування та програмування;

– застосування об'єктно-орієнтованої методології до розробки складних екологічних моделей;

– математичне моделювання в екології.

ГЛАДИШЕВСЬКИЙ Антон Олександрович – магістрант 5-го курсу Житомирського інженерно-технологічного інституту факультету інформаційно-комп'ютерних технологій.

Наукові інтереси:

– об'єктно-орієнтоване проектування та програмування;

– застосування об'єктно-орієнтованої методології при розробці обчислювальних моделей;

– моделювання в екології.

Подано 7.12.1999.