УДК 681.03

К.М. Лавріщева

МОДЕЛІ ВЗАЄМОДІЇ ПРОГРАМ, СИСТЕМ І ОПЕРАЦІЙНИХ СЕРЕДОВИЩ

Розглядаються моделі взаємодії програм, що будуються в різнорідних середовищах, систем, що збираються з готових програм та середовищ, що забезпечують сумісне об'єднання програм, інтегрованих у репозиторію Eclipse. Дано опис реалізації моделей взаємодії програм з репозиторію через інтерфейси в середовищах Visual Studio.Net і CORBA. Визначені декілька конфліктів взаємодіючих програм у середовищах засобами системи WCF, що залежить від різних способів реалізації у них інтерфейсів. Сформульовані головні теоретичні та прикладні шляхи їх подолання у майбутньому проекті.

Вступ

Сьогодні побудова окремих прикладних програмних систем (ПС) виконується масово в основному з використанням компонентного, сервісного та генерувального програмування (ГП). Ці методи підтримуються сучасними системами автоматизації у різних платформах і середовищах. Постійно удосконалюються методи збирання (інтеграційного, композиційного) різнорідних програм і компоненттів, накопичених забагато в інформаційному світі Інтернету. Відповідно проведеного аналізу сучасних фабрик програм [1–4] нами зроблений висновок про те, що кожне діюче операційне середовище або розподілена система загального призначення в основному підтримують розробку і зборку деякої сукупності готових чи знов розроблених програм. Зборка програм забезпечується на рівні інтерфейсного (брокерного), проміжного (middleware) або вихідного коду ПС. У випадку коли кінцевий результат розробки ПС необхідно виконувати за межами середовища його виготовлення, тобто в іншому середовищі, то як, правило, виникають деякі проблеми, що пов'язані з особливостями реалізації моделей взаємодії програм, систем і середовищ [2–5].

Індустрія програмної продукції (ПП) базується на готових і звичайних компонентах (артефактах, reuses, assets і даних) багаторазового використання, накопичених у різних бібліотеках, репозиторіях та нових «хмарних» сховищах Інтернету. Головним базисом індустріального розвитку ПП є програмні компоненти і системи, які створюються у сучасних розподілених середовищах, та метод їх зборки у різні структури ПП з вбудованими механізмами взаємодії і зв'язками, необхідними при рішенні широкого класу наукових задач з e-sciense.

Виходячи з того, що механізми взаємодії у кожному середовищі виробництва ПП відрізняються між собою, то в даній роботі розглядаються моделі взаємодії та вирішується задача інтеграції ПП у репозиторій іншого середовища з доробкою інтерфейсу взаємодії для виконання цього ПП у даному середовищі. Ця можливість підвищує ефективність вироблення програм з готових програмних ресурсів, що розміщені в інтегрованому репозиторію [6–8]. Далі наведено опис різних варіантів моделей взаємодії програм, розподілених систем і операційних середовищ, а також запропонований новий підхід до вирішення складних проблем взаємодії між різними сучасними середовищами, що використовуються для побудови за їх життєвим циклом (ЖЦ) прийнятої технології нових різнорідних програм (C#, Java, Basic). Варіанти цих моделей розроблені в межах фундаментального проекту ІІІ–1–07 [3] і

практично реалізовані для пар середовищ: VS.Net, Eclipse, Corba, Eclipse та VS.Net на прикладі програм у мовах Java і C#. Вони розширюють середовище Eclipse технологічними інструментами з розробки програм та засобами їх інтеграції у репозиторій.

Характеристика моделей взаємодії програм, систем і середовищ

Модель взаємодії призначена для обміну інформацією між різними компонентами при їх об'єднанні й обчисленні [4, 9]. Під *взаємодією* розуміються зв'язки двох і більше об'єктів і відношення між ними.. Даний термін має спеціальне значення для програм, систем і середовищ [6–8]. Його основу містить процес обміну повідомленнями (викликами, запитами, протоколами) між програмами, системами і середовищами для сумісного рішення деякої задачі. В повідомленні задається інтерфейс, який специфікується загально прийнятою мовою IDL (Interface Definition Language) або іншими (APL, SIDL й ін.). На загальному рівні інтерфейс є механізмом забезпечення взаємодії (interconnection) різнорідних програм для сучасних середовищ, що забезпечують розроблення програм і систем.

Нові механізми забезпечення взаємодії програмних компонентів (різномовних, різноплатформенних, різносередовищних) розробляються нами у рамках генерувальної моделі GDM (Generative Domain Model) ПС у середовищі ГП [3]. Сукупність ПС або сімейств систем (СПС) породжується за моделлю GDM і моделлю характеристик їх окремих елементів (компонентів, сервісів, каркасів і т. п.), що входять до складу СПС. Вони відповідають деяким функціям або поняттям предметної області (ПрО), що специфікуються різними мовами програмування (МП) і реалізуються в одному з діючих середовищ. Основу подання моделі ПрО містить об'єктно-орієнтований та компонентний підхід, доповнені механізмами породження готових компонентів повторного використання (КПВ) і засобами забезпечення їхньої взаємодії між собою [4].

Формально під *моделлю взаємодії* розуміється опис процесів для встановлення залежностей між різними параметрами елементів програмних чи інформаційних систем. Тобто така модель відображає систему відношень, що описує процес побудови ПП, як деяке явище. Ці відношення можуть задаватися математичними засобами – абстрактна алгебра, теорія множин тощо.

Параметрами моделі взаємодії є програма (компонент), інтерфейс і повідомлення. Модель має такий загальний вигляд:

$Мвз = \{Мпр, Мсис, Мсеред\}$,

де – $Мпр = \{Com, Int,, Prot\}$ – модель програми, $Мсис = \{CPC, Int,, Prot\}$– модель системи, $Мсеред = \{ENVIR, INT, PROT\}$– модель середовища, в якому *INT, PROT* відображають сукупність інтерфейсів та протоколів стосовно.

Програмний елемент моделі (*Com, CPC*, *ENVIR*) специфікується відповідною МП, інтерфейс – мовою IDL, протокол – мовою XDL, RDF тощо. Програмні елементи й інтерфейси зберігаються в бібліотеках і репозиторіях системи ГП, які використовуються для організації пошуку елемента типу КПВ, аналізу його функції і можливості застосування в нової системі. Розроблення самостійних програмних елементів виконується в інтегрованому середовищі Eclipse, MS.Net, CORBA, Java, COM з використанням взаємозв'язків між складними елементами.

Інтерфейс як самостійний об'єкт сформувався у зв'язку з об'єднанням модулів і програм у великі системи або комплекси програм на mainframes [1, 2] у 80-х роках минулого сторіччя. Інтерфейси як тоді й нині завдають атрибути та набір операцій, які визначають необхідні дії та методи обробки типів даних з отриманням від викликаної програми результату. Він відігравав роль посередника між модулями, що викликаються і викликають. Оператор виклику CALL завдає список фактичних параметрів, який перевіряється на відповідність (кількість і порядок розташування) формальним параметрам описаної процедури чи функції. Якщо типи даних параметрів, виявлялися не релевантними, то проводилося пряме і обернене перетворення даних з урахуванням структури пам'яті комп'ютерів. У розподілених програмах, що розташовані у різних середовищах, використовуються нові засоби взаємодії – RPC, RMI, ORB (stub, skeleton), IContract тощо.

Середовище розроблення програм. Розглядається операційне середовище з системними програмними засобами і інструментами для підтримки процесів розроблення окремих програм у МП і зборки їх у проект. Базисом інструментального середовища ГП обрано Eclipse, як засіб керування репозиторієм КПВ і використання їх при виробленні нових ПС методом зборки. За життєвим циклом середовищ MS.Net, CORBA, Java знову розробляються програми на допустимих МП. Кожне з цих середовищ містить свій специфічний набір засобів і підходів до трансформації описаних програм в МП для вихідного коду, що відрізняються між собою.

Загальна модель розподілених систем мережного середовища Інтернет показана на рис. 1.

Кожне з показаних середовищ CORBA, IBM, VS.Net, Java, базується на своїх інтерфейсах взаємодії і включає загальні методи та засоби доступу до даних мережного середовища.

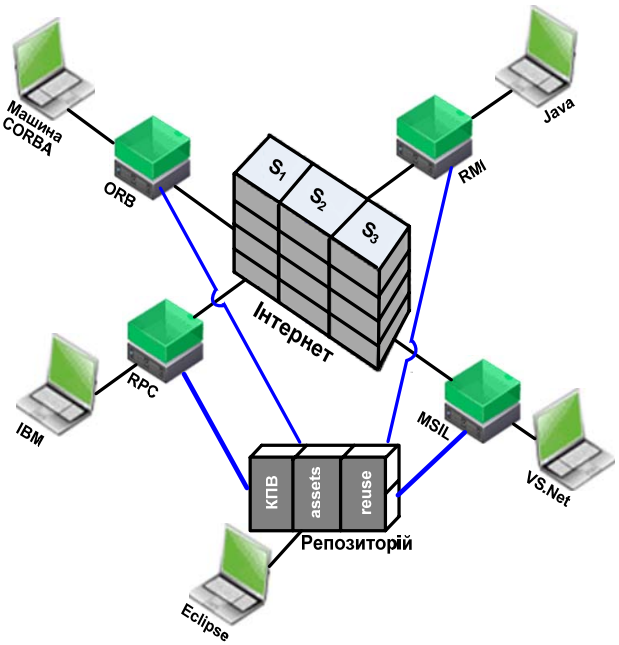

Рис. 1. Загальна модель схеми розробки і взаємодії програм у Інтернет середовищі

Програми, виготовлені в одному з середовищ, можуть бути інтегровані з одного середовища в інше через репозиторій, наприклад Eclipse. У межах проекту ГП здійснено розширення середовища Eclipse за рахунок інтеграції нових програм, вироблених в одному з середовищ, інтегрованих у репозиторій системи ГП [3, 10 – 13]. Ці операційні середовища забезпечують відповідні процеси ЖЦ розроблення різнорідних програм та методи їх об’єднання у різні структури ПП через механізми взаємодії, реалізованому в цьому середовищі. Розглянемо основні методи вироблення, інтеграції і взаємодії програм у парах системних середовищ ГП:

– Visual Studio.Net, Eclipse, на які орієнтується проект Grid і які застосовують технологію розробки окремих програм

у мові C# за стандартними процесами ЖЦ, специфікацію інтерфейсу з паспортними даними і з перенесенням готового продукту в репозиторій системи Eclipse, що відображає зв'язок з даним середовищем розроблення програм через механізм плагинів або конфігураційний файл з параметрами і операціями обробки даних у тому чи іншому середовищі [14];

– CORBA, Java, MS.Net забезпечують розробку програм на МП цього середовища та встановлення зв'язків між цими програмними середовищами в цілях забезпечення розміщення розроблених програм у репозитарії системи ГП та додавання доступу іншим розробникам його програм [15];

– IBM Sphere, Eclipse, де розробляються нові програми з використанням можливих МП, що допускаються у цьому середовищі чи у VSphere віртуального варіанта.

Інтерфейс є головним параметром операційних середовищ, а також моделей взаємодії програм і систем. Дамо опис змісту моделей взаємодії.

Модель взаємодії програм – це схема зв'язків окремих частин програм між собою. Як зв'язки виступають оператори звернення (типу CALL) до процедур і функцій цієї програми з формальними параметрами. Програми, процедури і функції записуються в одній МП. Оператори звернення містять імена об'єктів (процедур і функцій), що викликаються, список фактичних параметрів з завданням їх значень і параметрів, для отримання результатів. Послідовність і число формальних параметрів мають відповідати фактичним параметрам. Виконання функції у програмі на одній МП не викликає проблем, коли типи даних параметрів збігаються.

Моделі взаємодії систем найбільш пов'язані з використанням готових програм у процесі розробки нових ПС методом зборки. Якщо всі готові програми подані в різних МП і розташовані на різних комп'ютерах мережі, то при їх збиранні можуть виникати проблеми неоднорідності типів даних у цих програмах, структурах пам'яті платформ комп'ютерів і операційних середовищ, де вони виконуються. Зібрані ПП враховують формати даних готових програм на платформах сучасних комп'ютерів, особливості структури вихідного коду програм після компіляторів з МП, який залежить від використання спеціальних бібліотек data types і routines або без них. Реально існуючі розходження в апаратній частині платформ і у вихідному коді програм враховуються у конфігураційному файлі ПП, який використовується при організації виконання ПП у сучасних обчислювальних або гетерогенних середовищах. Деяким питанням перебудови загальних типів даних (ЗТД) до фундаментальних типів даних присвячений стандарт ISO/IEC 11404–2007 General Data Types. Він пропонує механізми генерації складних типів даних (контракт, портфель тощо) до більш простих, що є в МП. Але фундаментальні типи даних МП підтримуються спеціальними засобами сучасних операційних середовищ (Sun IBM, Microsofts.Net, CORBA, COM, JAVA й ін.). До них відносяться проміжний посередник типу stub, skeleton брокерного типу, що виконує перебудову типів даних від клієнта для передачі серверу і навпаки, а також вихідної код типу MISL у проміжному коді або EXE для виконання в системах Linux, Windows Server, MS.Net, IBM Web Sphere і т. п. [2].

Модель взаємодії операційних середовищ між собою розглядається нами як послідовність дій з інтеграції програми, створеної у одному середовищі в інше. Загальна схема розробки розподілених систем у середовищі ГП на основі інтерфейсів програм і моделей взаємодії для середовищ IBM, VS.NET, JAVA, CORBA показано на рис. 2.

У центрі цієї схеми розміщена система Eclipse, яка виконує по сутності функцію інтегратора програм у репозиторій з інших середовищ Інтернету та адміністратора цього репозиторію по збереженню, відбору і застосуванню готових програмних ресурсів типу КПВ, а також для виконання ПП, виготовлених у цих середовищах.

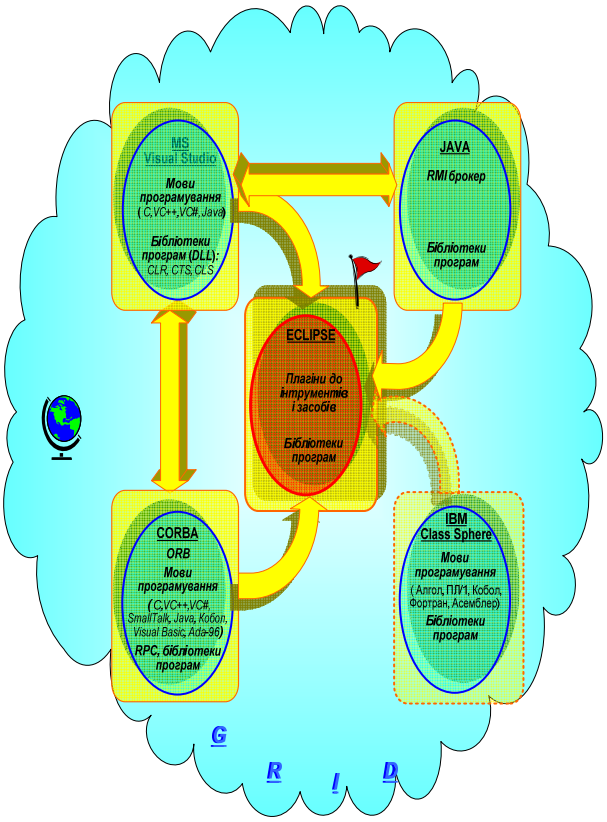

Рис. 2. Модель взаємодії розподілених систем з Eclipse

Деякі середовищі, наприклад, VS.Net і JAVA вже мають прямі зв'язки і створюють одне інтегроване середовище.

У межах фундаментального проекту проведено реалізацію моделей взаємодії пар наступних операційних середовищ Visual Studio ↔ Eclipse та CORBA↔ Visual Studio за участю студентів КНУ імені Тараса Шевченка [14, 15].

Дамо короткий зміст процесів проектування програм і забезпечення їх взаємодії на прикладі вказаних пар середовищ.

Модель Visual Studio ↔ Eclipse [14]. На прикладі побудови програм на мові C# у середовищі Visual Studio, реалізована модель взаємодії програм через розміщення їх у репозиторій середовища Eclipse. Взаємодія програми у мові C# у середовищі Eclipse виконана через спеціальний плагін Emonic та компонент NAnt із платформи (.Net). Створені з них архіви розархівуються при переносі їх в аналогічні папки середовища Eclipse. Для їх виконання в системі Eclipse створюється пустий проект, потім натискується права кнопка миші на назву цього проекту, із контекстного меню вибирається пункт Import → File System, й із файлової системи імпортується вибраний проект. У конфігураційному файлі проекту (.build) змінюється його вміст шляхом завдання вихідних файлів, бібліотеки та файли ресурсів. Тобто при переході в середовище Eclipse використовуються вихідні файли програми, dll-бібліотеки VS.Net та файли ресурсів (.resx). Коли необхідно знов перейти із цього середовищf в Visual Studio, то весь проект імпортується. Обчислення програми і її зміни можуть виконуватися як у середовищі Eclipse, так і Visual Studio. Net.

Модель CORBA↔ Visual Studio [15]. На експериментальному прикладі програми у мові Java пакету Java Development Kit в системі CORBA і відповідної компіляції IDL-опису її інтерфейсу брокер ORB, що створює клієнтський і серверний посередник Stub і Skeleton, продемонстровано перенос цієї програми на платформу Microsoft.Net за допомогою пакета IIOPNet. Тобто реалізація задачі взаємодії прикладних компонентів, розроблених у заданих середовищах Microsoft.Net (клієнтська частина) і Java (серверна частина), виконана через систему CORBA. У платформі MS.Net значення після повернення мають тип

MarshalByRefObject. Це дозволяє трактувати всі звертання до полів і методів об'єкта як вилучені виклики. Забезпечення взаємодії програм між двома середовищами Java і MS.NET виконано наступними процедурами: компіляція серверної частини застосування; створення серверного Skeleton й опис інтерфейсного об'єкта мовою IDL за допомогою утиліти rmic; перетворення отриманого IDL-опису в CLS-сумісну бібліотеку, використовуючи пакет IIOPNet приєднання бібліотеки до клієнтської частини застосування для використання відповідного сервісу.

Усі дані, що обмінюють між собою програми клієнта і сервера, є строкові, тобто є екземплярами класу String. У системах програмування строки зберігаються усередині об'єктів String, як послідовність символів Unicode фіксованої довжини, тобто відповідають стандартному типу даних CORBA wstring і використовуються для передачі даних через систему CORBA.

Розміщення побудованої у мові Java програми у репозиторій Eclipse виконується через плагін. При цьому формується паспорт програми (табл. 1), відповідно шаблону специфікатора програм, який за своїм змістом наближений до специфікаторів (програм, підсистем, систем, проектів, модулів), прийнятих у системі Grid [10].

У межах ГП розроблений набір моделей: взаємодії (програм, систем і середовищ), а також моделей варіабельності (варіантності, змінюваності) [9, 10], живучості (надійності, відмовостійкості, відновлюваності, адаптивності) [11] та життєвого циклу (ТЛ, Product Line), конфігураційні (збіркові), трансформаційні, генераційні моделі [3] тощо. При цьому Eclipse виконує функції інтегратора готових для використання програм у репозиторій та керування інструментами і засобами, що входять до складу інтегрованого середовища ГП.

Таблиця 1. Шаблон специфікатора

Назва параметрів	**Зміст параметрів**
Розробник	Прізвище, ім'я по-батькові автора, власника компонента, що додається
Дата створення	Дата створення КПВ автором (дата кінцевої, атестованої, специфікованої версії продукту)
Дата зміни	Дата внесення змін у КПВ
Версія	Версія компонента повторного використання
Платформа	Платформа, для якої створювався КПВ та на якій перевірена його працездатність
Операційна система	Операційна система, для якої створювався КПВ та на якій перевірена його працездатність
Розмір	Загальний розмір КПВ (продукту, документації тощо)
Опис	Короткий опис КПВ, список внесених змін, системні вимоги, вимоги до користувачів, список необхідних програм для коректної роботи, довідка тощо
Правила використання	Особливий опис КПВ, згідно побажань автора, правил розповсюдження тощо

Зокрема, ця система забезпечує доступ до репозиторію програм і інтерфейсів користувачу ГП, а також до системи PROTÉGÉ для створення онтології деякого домену й отримання інформації про ті домени, які представлені в системі ГП.

Інтерфейс як головний параметр запропонованих моделей взаємодії

До нинішнього часу розроблені інтерфейси різного призначення, які є головним елементом при збиранні складних програм із готових програм типу КПВ [6, 7]. Нині використовуються інтерфейси програм, мов, даних, сервісів, процесів:

– інтерфейс модульний, програмний (RPC, RMI, IDL, API, ISO тощо);

– міжмовний інтерфейс (Java Native Interface, SIDL– Scientifical IDL, Fundamental Data Types, GDT – General Data Types);

– сервісний інтерфейс (Icontract, web, ISO);

– проміжний інтерфейс (Middleware, Virtualware, ISO, міжсистемний, між– середовищний, між клієнтом і сервером тощо);

– технічний інтерфейс (стандарт SEI на інтерфейсну карту МПО–16Е–2).

У залежності від видів сучасних середовищ у табл. 2 наведено їх перелік, інтерфейси і програмні елементи.

Таблиця 2. Типи інтерфейсів

Середовище виготовлення	**Інтерфейс взаємодії**	**Програмні елементи**
IBM Sphere	RPC	Модуль, програма
MS.Net, VSTS.Net	MSIL	Програма
CORBA (Skeleton)	ORB	Об'єкт (stub)
JAVA	RMI	Розподілені ПС, програма
COM	API	Компонент
WCF	Icontact	Програма, система, розподілене ПС
GRID	Transport Connectivity	Компонент, система, модуль, пакет
Eclipse	Plug-in	Системи і інструменти

Наведені інтерфейси застосовуються у середовищах обчислення розроблених для спеціально доменів та готових елементів – КПВ, assets, reuses, які можуть об'єднуватися віртуальним композіром і виконуватися за допомогою нових хмарних засобів (WCloud, Azure, Amazon, Mech, Wapps), побудованих фірмами IBM-VSphere та Microsoft (рис. 3).

На даному рисунку показана технологічна схема розподіленої обробки різнорідних програм у мережному середовищі, базованому на готових програмах і системах сервера "Системи" та сховищах даних сервера "Сховища", що займають центральне місце у мережному середовищі. Виходячи з того що ці сервери підтримують обчислення програм за даними, що знаходяться у сховищах даних, це середовище завдає новий віртуальний прошарок між Internet і засобами Cloud Computing. Цей прошарок надає сервіс з обробки даних великих розмірів у режимі online з віртуальних серверів Web-services, Sky-driven (www.cmswire.com), які забезпечують майбутню віртуальну обробку даних. Такий напрям отримав назву хмарних обчислень задач глобального типу (Http://lenta.ru/articles/2010). Ці засоби орієнтовані на збереження, синхронізацію даних великих розмірів даних та доступ до глобальних сховищ даних on-line, викладених на віддалених серверах тощо.

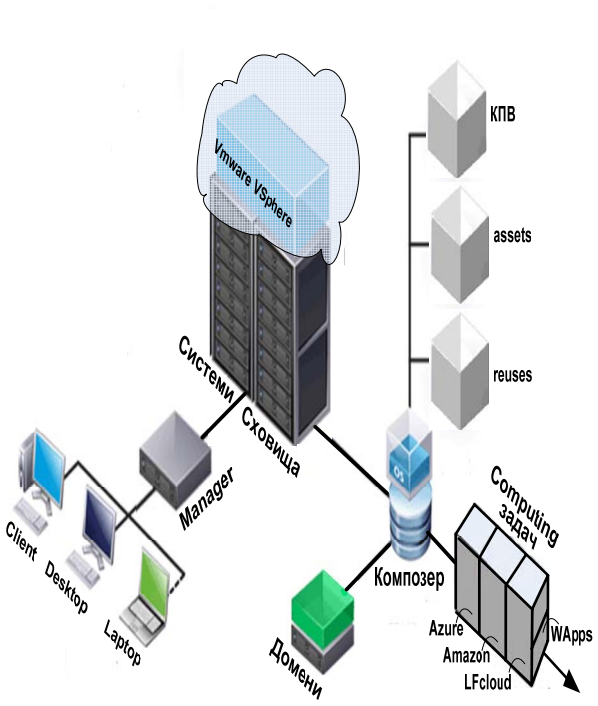

Рис. 3. Схема обчислення програм у віртуальному середовищі VSphere

Організація хмарних обчислень потребує реалізації нових методів взаємодії готових ресурсів і сервісів (наприклад,

через конфігураційний файл для обчислення наукових задач. Наведені сучасні середовища підтримують відповідні принципи взаємодії (див. табл. 2) за наступними типами інтерфейсів:

1) через брокер запитів між клієнтом і сервером, щодо отримання даних від stub клієнта, їх обробку під формати комп'ютера, передачу їх серверу і зворотно від skeleton сервера, що обробляє результати обчислювань до форматів даних клієнта;

2) через проміжний прошарок загальносистемних засобів, сучасних середовищ або віртуальних серверів хмарних обчислень [12, 13];

3) через пряму зборку трансльованих програм за різними МП і їх інтерфейсами у бібліотеках, наприклад, MS.Net [15].

Перший тип інтерфейсу практично забезпечується брокером ORB для класу МП у середовищі CORBA. Зборка компонентів базується на Client class з викликами інших, Stub class з конвертування даних, ORB class з передачі даних; Server class зі створення сервентів та посилання даних ORB; Skeleton class з конвертування форматів даних та породження різних видів серверів [16].

Другий тип інтерфейсу забезпечує взаємодію між компонентами, розміщеними на різних комп'ютерах через проміжний прошарок (middleware) шляхом повідомлень (наприклад, Java Message Queue), віддаленого виклику процедур RPC в MS Windows, ONS IBM, CORBA та виклику методів RMI у Java. Модель ISO/OSI також забезпечує проміжну взаємодію на рівнях (фізичному, канальному, мережному і транспортному) мережної ОС. Верхній рівень моделі (прикладний) забезпечує перетворення даних до транспортного рівня шляхом їх маршалінгу й демаршалінгу за інтерфейсним stub. Сеансовий рівень моделі передає дані іншим рівням через транспортний канал за механізмами посилань.

Третій тип інтерфейсу є проміжним (MSIL), реалізованим в MS.Net за допомогою таких бібліотек системи: CLR (Common Language Runtime), CTS (Common Type System) і CLS (Common Language Specification). CLR забезпечує виявлення і завантаження даних, керування ними у відповідності до завдань розробника програми. CTS визначає принципи взаємодії із іншими, відповідно представлення форматів мета–даних. Збірка різномовних програм виконується засобами CLS бібліотеки. Усі компілятори з МП створюють модулі DLL або EXE у проміжній мові MSIL (Microsoft Intermediate Language) як механізм зборки програми незалежно від платформи, на якій вона буде виконуватися. При її запуску вона конвертується в специфічний код CPU, що виконується на різних архітектурах комп'ютерів [16].

Таким чином, у залежності від цілей вироблення програм в якості середовища вибирається те середовище, що найбільш підходить для організації процесів розроблення і зборки відповідних програмних ресурсів у межах деякої ПрО. Коло проблем звужується, коли різні середовища можуть взаємодіяти між собою, створюючи одне велике, гнучке середовище з поширеними можливостями щодо виробництва кінцевого ПП.

Новий підхід до забезпечення взаємодії різних сучасних середовищ у WCF

Головна ціль системи ГП міститься в виготовленні ПП шляхом:

– побудови нових програм в одному з обраних середовищ – VS.Net –C# і CORBA Java;

– специфікації збудованих програм і заповнення шаблону специфікатора паспорта для розміщення їх репозиторії системи ГП;

– вибору готових КПВ або програм для збирання з них нових ПС для рішення деяких задачі ПрО;

– перевірки інтерфейсів для збирання готових компонентів у ПС, тестування цієї ПС, формування комплексного паспорту з занесенням його разом з готовим продуктом у репозиторій системи Eclipse тощо.

У системі ГП вперше зроблено спробу об'єднати готові продукти, що отримані у різних середовищах, VS.Net – C# і CORBA Java і зберігаються у репозиторії Eclipse. ПП будуються різними мовами. У кожному з обраних середовищ ГП реалізується свій підхід до вирішення проблеми взаємозв'язку різномовних або однакової мови програм. Коли отриманий продукт з одного середовища переноситься в інше, в системі ГП є реалізований механізм взаємодії [17].

Водночас ця проблема вирішувалася в межах WCF (Windows Communication Foundation) [18] іншим способом, а саме, за допомогою нового типу інтерфейсу, так званого контракту (IContract), як складової частини Consumer та Provider (рис. 4), який не звільняє розробників від різного роду конфліктів взаємодії.

Інтерфейс Icontract містить опис атрибутів та операцій з передачі даних від одного сервісного об'єкта клієнта (Service consumer) до іншого (Service provider). Їх опис дається у мові XML. Передачу інтерфейсів за контрактом виконує протокол, в якому задаються атрибути та операції інтерфейсу для передачі відповідному об'єкту розподіленої системи при їх об'єднанні.

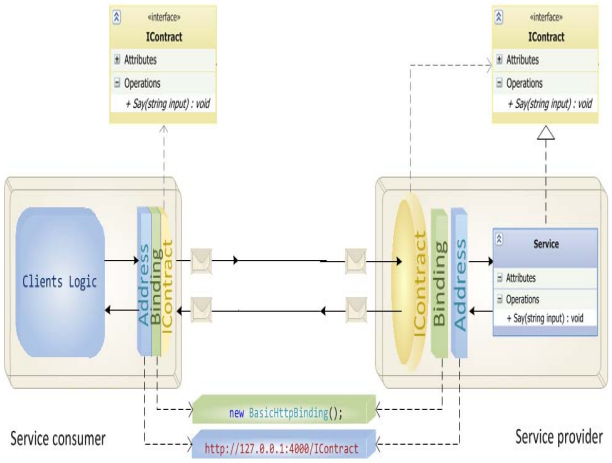

Рис. 4. Зв'язок двох сервісних компонентів в архітектурі WCF

В результаті проведеного дослідження щодо використання нового типу інтерфейсу взаємодії сервісів через контракт у системі WCF для розроблення з наявних сервісів деяких розподілених систем. У межах проекту WCF визначено чотири види інтерфейсів-контрактів для забезпечення взаємозв'язків компонентів між собою:

контракти служб описують операції, що викликаються клієнтом на сервісі;

контракти даних визначають типи даних, що приймаються і передаються службою, а також непрямі контракти убудованих типів (таких як int, float, string і тощо);

контракти помилок визначають помилки, що втримуються службою для передачі їх клієнтам;

контракти повідомлень – механізм прямої взаємодії об'єктів через повідомлення.

Повідомлення специфікується мовою XML і подається протоколом SOAP в XML наступного виду:

```
<?xml version=»1.0» ?>
<env:Envelope xmlns:env=»
http://www.cbsystematics.com»>
<!—Конверт протоколу SOAP →
    <env:Header>
<!— Заголовок протоколу SOAP →
```

</env:Header> <env:Body>

<!—Тело протоколу SOAP →

</env:Body> </env:Envelope>

При взаємодії між користувачем і провайдером можуть виникнути різного роду конфлікти (рис. 5).

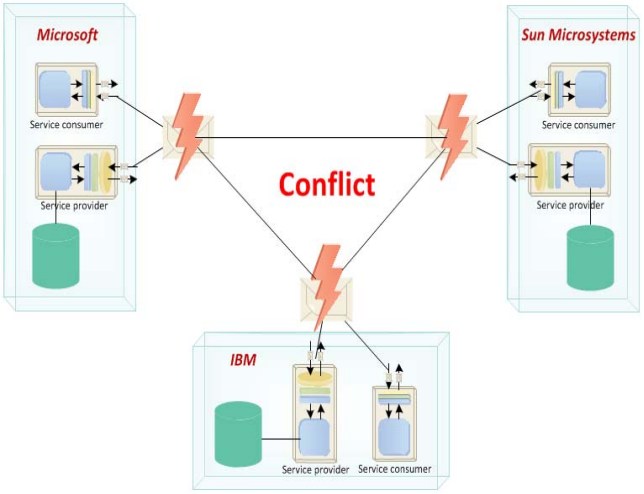

Рис. 5. Схема взаємодії програм середовищ через протоколи і можливі конфлікти

До варіантів конфліктів належать:

– несумісність типів даних, які передаються в інтерфейсах кожного компонента, що об'єднується разом з іншою різномовною програмою. Наприклад, передача даного типу – ціла програму, де цей параметр поданий, як символьний;

– відмінність конфігураційних файлів ПП за структурою та змістом інформації, що необхідно використати при обчисленні тої чи іншої програми з ПС;

– різниця в мовах програмування, що використовуються для розробки окремих програм;

– відмінність порядку в опису параметрів протоколів передачі інформації між різними сервісними об'єктами;

– відмінності в архітектурі платформ об'єктів клієнта і серверу тощо.

Для реалізації деяких конфліктів генеруються відповідні програми чи компоненти з відокремленням конфліктних ситуацій. Іншим способом подання проблем передачі й обробки даних є генерація типів даних відповідно стандарту ISO/IEC-11404.

Надалі у майбутньому проекті, як продовженню фундаментального проекту ІІІ–07–01 для рішення цієї проблеми, будуть прийняті проектні рішення щодо використання сервісів при забезпеченні взаємодії програм у сучасних середовищах для проведення обчислень за готовому ПП.

1. *Андон П.І., Лавріщева К.М.* Розвиток фабрик програм в інформаційному світі // Вісник НАН України. – 2010.– № 10.– С. 15–41.
2. *Лавріщева К.М.* Концепція індустрії наукового софвера і підхід до обчислення наукових задач // Проблеми програмування. – 2011.– № 1. – С. 3–17.
3. *Лавріщева К.М.* Генерувальне програмування програмних систем і сімейств // Проблеми програмування. – 2009.– № 1. – С. 3–16.
4. *Лаврищева Е.М., Грищенко В.М.* Сборочное программирование. Основы индустрии программных продуктов. – Второе издание. – Киев: Академпериодика, 2009.– 371 с.
5. *Анісімов А.В., Лавріщева К.М., Шевченко В.П.* Про індустрію наукового софтвера. – Conf. Theoretical and Applied Aspects of Cybernetics, Kiev, 2011, Ukraine.
6. *Лавріщева К.М.* Програмна інженерія. – К.: Академперіодика. – 2008.–319 с.
7. *Лаврищева Е.М.* Интерфейс в программировании // Проблеми програмування.– 2007. – № 2. – С. 126–139.
8. *Лаврищева Е.М.* Проблема интероперабельности разнородных объектов, компонентов и систем. Подходы к ее решению // Матер. 7-ї Міжнар. конф. з програмування "УкрПрог – 2008". – 2008. – С. 28–41.
9. *Лавріщева К.М., Слабоспицька О.О., Коваль Г.І., Колесник А.О.* Теоретичні аспекти керування варіабельністю в сімействах програмних систем. – Вісник КГУ, серія фіз.-мат. наук. – 2011. – № 1. – С. 151–158.
10. *Колесник А.Л.* Механізми забезпечення варіабельності в сімействах програмних систем // Проблеми програмування. – 2010. – № 1. – С. 35 – 44.

11. *Ігнатенко П.П., Бистров В.М.* Особливості забезпечення життєздатності програмних систем в умовах генеруючого програмування // Проблеми програмування. – 2008. – № 2–3. – С. 270–278.
12. *Таковицкий О.* Технология Grid computing. – С. 1–9.
13. *Ильин В.А.* Сетка с облаками для интернета // В мире науки.– 2010.–С. 83–85.
14. *Основы* инженерии качества программных систем / *Ф.И. Андон, Г.И. Коваль, Т.М. Коротун, Е.М. Лаврищева, В.Ю. Суслов* // 2-е изд. – Киев: Академпериодика, 2007. – 672 с.
15. *Радецький І.О*. Один з підходів до забезпечення взаємодії середовищ MS.NET і ECLIPSE // Проблеми програмування. – 2011. – № 2. – С. 43–49.
16. *Островский А.В.* Подход к обеспечению взаимодействия программных сред JAVA и MS.NET // Проблеми програмування.– 2011. – № 2. – С. 34–42.
17. *Коваль Г.І., Колесник А.Л., Лавріщева К.М., Слабоспицька О.О.* Удосконалення процесу розроблення сімейств програмних систем елементами гнучких методологій // Проблеми програмування (Спецвипуск конф. УкрПрог–2010). – 2010. – № 2–3. – С. 261 – 270.
18. *Слабоспицька О.О.* Технологічна модель процесу автоматизованого виробництва сімейств програмних систем // Проблеми програмування. – 2011. – № 1. – С. 39–48.
19. Windows Communication Foundation –www. edu.cbsycemantic.com.

Отримано 23.04.2011

Про автора:

Лавріщева Катерина Михайлівна,
доктор фізико-математичних наук,
професор, завідуюча відділом.

Місце роботи автора:

Інститут програмних систем
НАН України,
03187, Київ-187,
проспект Академіка Глушкова, 40.
Тел. (044) 526 3470.