УДК 681.03

К.М. Лавріщева, А.Ю. Стеняшин

ФОРМАЛІЗМИ ОБ'ЄКТНОГО ПРОЕКТУВАННЯ І ТЕСТУВАННЯ РОЗПОДІЛЕНИХ ПРОГРАМНИХ СИСТЕМ

Розглядається підхід до проектування розподілених програмних систем формальним поданням об'єктів функціонального і інтерфейсного типів. Дається визначення об'єктів та операцій проекції, взаємодії та паралельного виконання. Дано опис мови подання інтерфейсів об'єктів IDL (stub, skeleton) і оброблення їх брокером ORB системи CORBA. Запропоновано процес тестування об'єктних структур програм та перетворення даних різнорідних об'єктів для забезпечення їх взаємодії у операційному середовищі.

Вступ

Проектування розподілених програмних систем (РПС) виконується різними методами і підходами. Нині набувають чинності підходи проектування з використанням об'єктно-орієнтованого підходу (ООП), компонентного та сервісного програмування.

При проектуванні РПС з застосуванням ООП [1–5] і концепції розподілу об'єктів за різними вузлами середовища виникає проблема забезпечення їх взаємозв'язку, яка вирішується за допомогою інтерфейсів подібно зв'язку через stub, skeleton системи Corba [4]. В ній дані механізми створення об'єктної моделі (ОМ), яка орієнтована на функціонування в гетерогених середовищах. Інтерфейсні об'єкти ОМ описуються новою мовою IDL, яка робить цю модель узгодженою, чітко організованою з сучасними умовами середовищ. Метамодель ОМА системи CORBA підтримується різними сучасними платформами, у тому числі мейнфреймами і мінікомп'ютерами, Unix-системами тощо.

Міжнародною групою (Object Managment Group – OMG) розроблено проект стандарту об'єктної й еталонної моделей ОМА для проектування нових застосувань. Для об'єкта визначаються властивості, характеристики і типи даних. Об'єкти, що володіють загальними властивостями, групуються в класи. Тип примірника класу розглядається як відношення підтип/супертип. Кожному об'єкту відповідає одна або декілька операцій, що завдають виклику його методів. Операції визначають дії за об'єктами і їхній поведінці. Після виконання операції об'єкт набуває деякий стан, що впливає на його поведінку. Основними компонентами еталонної моделі ОМА є:

– мова IDL і транслятор інтерфейсу компонент застосувань (Application Interface); загальний об'єктний сервіс (Common Object Services), що забезпечує керування подіями, транзакціями, інтерфейсами, запитами й ін.;

– загальні засоби (Common facilities), необхідні для груп компонентів і застосувань (електронна пошта, телекомунікація, керування інформацією, емулятор програм й ін.);

– брокер об'єктних запитів (брокер ORB) для організації взаємозв'язку різнорідних об'єктів.

Таким чином, брокер ORB – це головний «зв'язок» об'єктів і компонентів із різних застосувань і середовищ. Є його опис на багатьох мовах програмування (МП) для поширеного використання у інших середовищах. Наприклад, у сучасних середовищі Grid, Cloud Computing [1].

Далі пропонується підхід до проектування РПС з використання функціональних й інтерфейсних об'єктів.

1. Засоби опису об'єктів РПС

Пропонується використання ООП для формального завдання функціональних й інтерфейсних об'єктів з метою їх виконання у сучасних гетерогенних середовищах. В ООП розглядаються програмні

об’єкти двох типів, які розрізняються між собою семантикою [5, 6].

Під об’єктами 1-го типу розумітимемо об’єкти, які відповідають прикладним функціям ПрО і забезпечують рішення задач предметної області. До об'єктів 2-го типу відносимо об’єкти-інтерфейси, що включають операції виклику методів об'єктів 1-го типу і грають роль посередників для забезпечення взаємозв’язку між функціональними об’єктами 1-го типу, віддаленими один від одного за різними середовищами. Тобто, інтерфейсні об’єкти виконують зв’язок об’єктів 1-типу клієнта з сервісними функціями – віддаленими процедурами мережі, розташованими у серверній частині РПС. Його загальними функціями є: підготовка зовнішніх даних клієнта (параметрів), набір викликів цих процедур або сервісу до сервера, обробка різних помилок, повернення даних від сервера до клієнта тощо.

Операції взаємодії і відповідні для них інтерфейси визначаються для об’єктів 1-го типу. Самі об’єкти описуються сучасними МП (наприклад, C++, Java, Паскаль), а їх інтерфейси в мові IDL.

Модель взаємозв’язку можна розглядати як узагальнення машини Тюрінга. В загальному випадку в ній реалізується модель обчислення алгоритму. Узагальнену модель цієї машини будемо називати розподіленою інтерактивною машиною з моделлю взаємодії об’єктів чи компонентів. Ця машина розширює машину Тюрінга вхідними і виходними (input, output) діями (actions), виробляючими динамічну генерацію і просування потенційно нескінченного потоку даних.

Розподілена інтерактивна машина включає машину Тюрінга для проведення обчислень, тобто її модель допускає обчислення й обмін повідомленнями в заданий інтервал часу, тоді як у машині Тюрінга на обчислення алгоритмів чинник часу не впливає. Інша важлива відмінність між цими машинами полягає у тому, що машині Тюрінга відповідає вхідна стрічка з кінцевим числом символів, а розподіленій інтерактивній машині – необмежений вхідний потік (запитів, пакетів).

Таким чином, інтерпретація моделі взаємодії за допомогою розподіленої інтерактивної машини ґрунтується на діях і станах загальної (розділеній) пам'яті взаємодіючих об'єктів.

Специфікація об’єктів 2-го типу

Структура об’єкта 2-типу, чи посередника незалежно від мови у цілому однакова. Це дає змогу стандартизувати структуру посередника та визначити її за допомогою однієї з зазначених мов. РПС складається із двох частин, кожна з яких виконується у різних процесах, а взаємодіють вони шляхом виклику інтерфейсних функцій. Перша частина – серверна програма, а друга – клієнтська.

Пропонуємо фрагмент мови опису інтерфейсів IDL у формі правил Бекуса–Наура [4]:

<інтерфейс об’єкта> ::= **Object** <ім’я_Об’єкта> **:{**<множина вихідних інтерфейсів>}; {<множина вхідних інтерфейсів>} **end;**

<множина вхідних інтерфейсів> ::= <множина інтерфейсів>;

<множина вихідних інтерфейсів> ::= <множина інтерфейсів>;

<множина інтерфейсів> ::= ∅ | <інтерфейс>; <множина інтерфейсів>;

<інтерфейс>::= **Interface <**ім’я_ інтерфейсу>**:** {<множина функцій>} **end**

<множина функцій> ::= ∅ | <функція>; <множина функцій>;

<функція> ::= **function <**ім’я_ функції>**:** <множина параметрів> **end;**

<множина параметрів> ::= <параметр> |<параметр>, <множина параметрів>

<параметр> ::= <тип> (<вид параметру>);

<вид параметра> ::= **in | out | inout.**

Тип описує множину типів, які підтримуються мовами програмування (C++, Pascal, т. п.) та забезпечують взаємодію між процесами, а <вид параметра> це: **in –** вхідний параметр, **out –** вихідний параметр, **inout –** сумісний параметр.

Множини вхідних та вихідних інтерфейсів мають різну семантику, але

однаковий синтаксис. Взаємодію між об'єктами будемо описувати за допомогою операції конкатенації (·).

Формальне визначення базових понять у термінах об'єктів за допомогою запропонованої мови специфікації [6]:

F – множина функцій,

O – множина об'єктів,

I – множина інтерфейсів об'єктів,

$O_k \in O,\quad In\left(O_k\right)$ – множина вхідних інтерфейсів (як клієнтських),

$O_k \in O,\quad Out\left(O_k\right)$ – множина вихідних інтерфейсів (серверних).

Визначення взаємодій об'єкта (взаємодія першого типу). Результатом взаємодії двох об'єктів буде об'єкт, у якого множина вхідних інтерфейсів збігається з множиною вхідних інтерфейсів об'єкта-сервера, а множина вихідних інтерфейсів – з множиною вихідних інтерфейсів об'єкта-клієнта:

$$O_k = \left(Out\left(O_k\right),\ In\left(O_k\right)\right),$$

$$O_l = \left(Out\left(O_l\right),\ In\left(O_l\right)\right),$$

$$O_k \cdot O_l = \left(Out\left(O_k\right), In\left(O_l\right)\right).$$

Не всі об'єкти можуть взаємодіяти один з одним, тому накладемо певні умови. Взаємодія об'єктів $O_k \cdot O_l$ є коректною, якщо об'єкт-сервер повністю забезпечує сервіс, необхідний об'єкту-клієнту

$$\forall I_m \in In\left(O_k\right) \Rightarrow \exists I_n \in Out\left(O_l\right) \wedge I_m = I_n.$$

Інтерфейс буде вхідним для об'єкта, якщо об'єкт отримує сервіс за допомогою цього інтерфейсу. І вихідним, якщо за допомогою цього інтерфейсу об'єкт надає сервіс об'єктам.

Визначення проекції об'єкта. Результатом проекції об'єкта на інтерфейс буде об'єкт, у якого множина вхідних інтерфейсів містить один елемент – цей інтерфейс, а множина вихідних інтерфейсів містить тільки ті інтерфейси, які необхідні для надання сервісу цього інтерфейсу.

$$O_k = \left(Out\left(O_k\right), In\left(O_k\right)\right),$$

$$O_k\left[I_m\right] = if\ \left(I_m \in Out\left(O_k\right)\right)$$

$$then\left(\left\{I_m\right\}, \begin{Bmatrix} I_n : I_n \in In\left(O_k\right) \wedge \\ exec\left(I_m, I_n\right) \end{Bmatrix}\right) else(\{\ \}, \{\ \}),$$

де $exec\left(I_m, I_n\right)$ – предикат, що вказує на необхідність інтерфейсу I_n для виконання I_m інтерфейсу.

Визначимо проекцію об'єкта на множину інтерфейсів:

$$O_k\left[\left\{I_1, I_2, \ldots, I_M\right\}\right] = \begin{pmatrix} \bigcup_{m=1}^{M} Out\left(O_k\left[I_m\right]\right) \\ \bigcup_{m=1}^{M} In\left(O_k\left[I_m\right]\right) \end{pmatrix}.$$

Результатом проекції об'єкта на об'єкт є проекція об'єкта на множину вхідних його інтерфейсів:

$$O_k\left[O_l\right] = O_k\left[In\left(O_l\right)\right].$$

З операцій проекції та взаємодії об'єктів випливає рівність об'єктів:

$$O_k \cdot O_l = O_k \cdot O_l\left[O_k\right].$$

Операція проекції об'єкта має вищий пріоритет, ніж операція взаємодії. Вона позначається конкатенацією:

$$\left(O_k \cdot O_l\right)\left[I_m\right] = O_k\left[I_m\right] \cdot O_l.$$

Визначення операції успадкування (взаємодія другого типу). Успадкування серед об'єктів РПС відбувається на рівні інтерфейсів. Так, якщо об'єкт успадковує інтерфейси іншого об'єкта ($O_k \leftarrow O_l$), то він надає сервіс всієї множини вихідних інтерфейсів цього об'єкта:

$$O_k \leftarrow O_l \Rightarrow Out\left(O_k\right) \subseteq Out\left(O_l\right).$$

Це положення випливає із ООП програмування РПС, де виконання програми може змінюватися динамічно, а не тільки статично.

Отже, успадкування об'єктом іншого об'єкта – це об'єкт, у якого множина вихідних інтерфейсів містить всі інтерфейси як першого, так і другого об'єкта, а множина вхідних інтерфейсів містить

тільки ті інтерфейси, які необхідні для надання сервісу цих інтерфейсів.

$$O_k O_l = \begin{pmatrix} Out(O_k) \cup Out(O_l), \\ \{ I_m : (I_m \in In(O_k) \cup In(O_l)) \wedge \\ \exists I_n \in Out(O_k \leftarrow O_l) : exec(I_n, I_m) \} \end{pmatrix}.$$

Операція успадкування об'єкта делегує іншому об'єктові свої інтерфейси.

З визначення операції успадкування інтерфейсів випливають такі властивості:

транзитивності

$$\forall O_{1,2,3} \in O : O_1 \leftarrow O_2, O_2 \leftarrow O_3 \Rightarrow O_1 \leftarrow O_3;$$

симетричності

$$\forall O_k \in O \Rightarrow O_k \leftarrow O_k.$$

Розкриття проекції над об'єктами, між якими існує взаємодія другого типу:

$$(O_1 \leftarrow O_2)[\ ] = O_1[\ ] \leftarrow O_2[\ ].$$

Опис розподілених РПС. Основне призначення мови специфікації РПС полягає у тому, щоб описати основні її властивості та функції для функціонування у розподіленому середовищі. Мова специфікації об'єктів середовища близька до мови опису РПС.

Для послідовного опису об'єктів середовища використовуються мови опису різних програми, що виконуються в цьому середовищі. Основною відмінністю між послідовним та розподіленим середовищем є можливість паралельного виконання об'єктів і програм. Ця можливість задається відповідними операціями опису паралельного виконання. Ці операції дають можливість розширити клас систем шляхом додавання інших ПС, що не увійшли до цього класу.

Надалі елемент множини РПС (об'єкт чи ПС) позначимо Р та дамо визначення операцій над системами РПС.

Визначення паралельного виконання РПС. Результатом паралельного виконання двох РПС буде середовище: $P_o || P_r$.

Якщо у розподіленому середовищі РС виконується не лише два РПС, а декілька, тоді розподілене середовище описується: $P_o || ... || P_r$.

Операцію паралельного виконання можна розширити і перенести на множину об'єктів і ПС, оскільки РПС є об'єктом складної структури. Але зауважимо, що результатом паралельного виконання об'єктів буде не об'єкт, а середовище, в якому між об'єктами не виникає взаємодії ні першого, ні другого типу $O_k \| \ldots \ldots \| O_l$.

Розширимо раніше визначені операції над об'єктами щодо середовища.

Визначення взаємодії об'єкта і середовища (взаємодія першого роду). Результатом взаємодії об'єкта і середовища буде об'єкт, у якого множина вхідних інтерфейсів збігається з множиною вхідних інтерфейсів об'єкта-сервера, а множина вихідних інтерфейсів – об'єднання множин вихідних інтерфейсів об'єктів середовища:

$$O_k \cdot (O_{l_1} \| \ldots \| O_{l_n}) = \left(Out(O_k), \bigcup_{j=1}^{n} In(O_{l_j}) \right).$$

Для подальшого використання вважаємо, що множина вхідних інтерфейсів при такій взаємодії є порожньою (накладемо таке обмеження). Оскільки при взаємодії об'єкта $O_k \cdot (O_{l_1} \| \ldots \| O_{l_n})$ виникає не детермінованість взаємодії (який із об'єктів середовища взаємодіє з об'єктом-сервером), тобто

$$O_k \cdot (O_{l_1} \| \ldots \| O_{l_n}) = (Out(O_k), \{\ \}).$$

Взаємодія об'єкта і середовища $O_k \cdot (O_{l_1} \| \ldots \| O_{l_n})$ є коректною, якщо виконується умова: середовище повністю забезпечує сервіс, необхідний об'єкту-клієнту

$$\forall I_m \in In(O_k) \Rightarrow \exists I_n \in \bigcup_{j=1}^{n} Out(O_{l_j}) \wedge I_m = I_n.$$

Визначення проекції середовища. Результатом проекції середовища на інтерфейс буде середовище, у якого всі об'єкти є проекціями об'єктів середовища.

$$(O_k \| \ldots \| O_l)[I_m] = O_k[I_m] \| \ldots \| O_l[I_m].$$

Аналогічно визначаємо проекцію середовища на множину інтерфейсів та на об'єкт. З визначення операцій проекції об'єкта та взаємодії між об'єктами випливає рівність:

$$O_k\left(O_{l_1} \| \ldots \| O_{l_n}\right) = O_k\left(O_{l_1} \| \ldots \| O_{l_n}\right)[O_k].$$

Визначення операції успадкування (взаємодія другого роду). Успадкування об'єктом інтерфейсів від РПС є об'єкт, що успадковує інтерфейси всіх об'єктів середовища. Дана операція в ООП також називається як множинне успадкування. Надалі будемо використовувати цей термін як синонім.

При успадкуванні інтерфейсів від середовища виникає недетерміновність коли існують два (або декілька) об'єкти, що надають сервіс по одному інтерфейсу. Для виключення не детермінованості запропоновано вважати, що операція паралельного виконання (в даному випадку) не симетрична, тобто:

$$O_k \leftarrow \left(O_{l_1} \| O_{l_2}\right) \neq O_k \leftarrow \left(O_{l_2} \| O_{l_1}\right).$$

Класи РПС, що описуються за допомогою мови специфікації. Описавши операції паралельного виконання РПС та об'єктів розширимо множину ПС, що описуються мовою специфікації. Тепер об'єкт може отримувати сервіс не тільки від одного, але і від багатьох об'єктів. Всі запити щодо інтерфейсу та сервіс об'єкт O_{k_1} отримує тільки від об'єкта O_{k_2}, який в свою чергу, якщо виникає необхідність, отримує сервіси від $O_{l_1} \ldots O_{l_n}$. Тобто між об'єктами O_{k_1} та O_{k_2} існує взаємодія першого типу, а об'єкти $O_{k_1} \ldots O_{k_n}$ виконуються паралельно, і визначають ПС класу $O_{k_1} \cdot O_{k_2} \cdot \left(O_{l_1} \| \ldots \| O_{l_n}\right)$.

Запити щодо інтерфейсу та сервіс об'єкт O_{k_1} отримує від об'єкта O_{k_2}, якщо цей об'єкт надає сервіс по цьому інтерфейсу, в противному разі від першого із об'єктів $O_{l_1} \ldots O_{l_n}$, який надає цей сервіс. Тобто об'єкт O_{k_2} наслідує інтерфейси від (взаємодія другого типу).

2. Тестування об'єктів 1-го і 2-го типів

Під перевіркою правильності інтерфейсів віддалених об'єктів розуміється визначення коректності завдання операцій виклику в stub-об'єкта і динамічної перевірки проходження протоколу повідомлення від локального об'єкта до віддаленого й обернено [6–8].

Результатом досліджень взаємодії об'єктів є розроблений оригінальний підхід для перевірки коректності уявлення об'єктів і засобів передачі параметрів віддаленого виклику через мережне середовище. Суть підходу полягає у вбудовуванні механізмів спостереження того, в що перевіряється об'єкт, у stub-об'єкт та у повідомлення для виконання методу об'єкта. Далі розглядаються основні його положення.

РПС складається з об'єктів 1-, 2- і 3-го типів. *Об'єкти 1-го типу* відображають прикладні функції застосування і забезпечують рішення задач кінцевого користувача. До *об'єктів 2-го типу* ставляться об'єкти-інтерфейси, що включають опис зовнішніх змінних і операції віддаленого виклику, що реалізують взаємодії об'єктів 1-го типу, коли ці операції у виді запитів потрапляють у розподілене середовище. Об'єкти 3-го типу – це повідомлення від одного об'єкта до іншого.

Підхід до тестування орієнтований на перевірку об'єктів 1-го – 3-го типів РПС окремо, їхніх взаємодій між собою і роботи об'єктів у комплексі. Об'єкти 1–го типу можна тестувати і традиційними методами, вставляючи заглушки на місцях звернення до посередників [6].

Тестування об'єктів 2-го типу полягає у перевірці правильності й описів інтерфейсів взаємодіючих об'єктів 1-го типу та операцій викликів методів у локальному режимі без виходу в мережу. При мережному варіанті об'єкти ПС їде перевірка на наявність або відсутність посередників у репозиторії, а також на перевірку переданих параметрів, маршалінг даних і шляхи проходження запитів від клієнта до сервера й зворотно.

Тестування об'єктів 1-го типу

Кожний об'єкт 1-го типу включає базові структури керування (БСК) обчисленнями: умови, ітерацію і послідовне виконання. Ці структури визначають шлях обчислень об'єктів 1-го типу в залежності від значень виражень, що утримуються в них , і умов, які впливають на керування. До БСК об'єктів 1-го типу саме і застосовується метод вбудовування механізмів тестування для перевірки правильності їхнього виконання.

Об'єкт 1-го типу тестується з вмонтованим механізмом тестування, що дозволяє переходити в режим тестування для перевірки правильності виконання БСК.

Кероване тестування об'єктів 1-го типу – це спроможність механізму управляти (контролювати) поводженням об'єкта під час тестування за допомогою вибору шляху виконання в його керуючих структурах (гілках програми) для виявлення структурних і семантичних помилок.

Спостережність тестування об'єктів 1-го типу – це спроможність механізму переглядати значення будь-яких змінних програмного об'єкта, отримуваних під час обчислень.

Метрики тестування об'єктів 1-го типу. Метод тестування об'єктів 1-го типу включає метрики, засновані на двох основних властивостях установлення ступеня тестування: контрольованістю (КТ) тестування БСК і спостережністю за тестуванням (НТ).

Контрольованість БСК – це властивість незалежного присвоєння керуючим змінним або вираженням цих структур таких значень, щоб будь-який шлях обчислень об'єктів 1-го типу був контролюємо доступний. КТ будь-якої БСК кількісно визначимо у такий спосіб:

КТБСК = 1, якщо БСК – незалежні один від одного,

КТБСК = 0, якщо БСК залежить від шляху виконання або взаємозалежностей виражень.

Контрольованість тестування КТ окремого об'єкта опишемо формулою:

$$КТ = \frac{1}{n}\sum_{i=1}^{n} КТ_{БСК_i}, \qquad (1)$$

де $КТ_{БСк_i}$ – контрольованість усіх БСК об'єкта.

Область визначення КТ – [0; 1]. Дана властивість інформує про ступінь перевірки слушності об'єкта, якщо

$КТ = 1$, то об'єкт цілком проконтрольований;

$КТ = 0$, то об'єкт не про контрольований.

$КТ < 0 < 1$, то об'єкт частково проконтрольований.

Під спостережністю тестування (СТ) об'єкта 1-го типу будемо розуміти властивість перегляду значень, прийнятих для будь-якої змінної усередині шляху тестування.

Область визначення СТ збігається з областю КТ. Якщо у об'єкта $CT = 1$ або $CT = 0$, то це означає, що об'єкт 1-го типу досліджено або ні. Одним із засобів реалізації дослідження об'єкта є, наприклад, додавання в програму таких операторів, як *write* або *print,* тоді можна досягти СТ ≡ 1.

Очевидно, чим більше КТ і СТ, тим більший ступінь повноти тестування.

Тестовність об'єкта 1-го типу (ТО1) можна визначити як функцію від конрольованості КТ і *спостереженості* СТ за тестуванням:

$$ТО1 = f(КТ, СТ) = КТ * СТ. \qquad (2)$$

Тут: ТО1 = 1, якщо КТ і СТ рівні 1.

ТО1 = 0, якщо хоча б один або обидва рівні 0.

Область значень ТО1 ∈ [0;1], при КТ ∈ [0;1] і СТ ∈ [0;1]. У зв'язку з тим, що отримується СТ≡1 просто, маємо формулу:

$$ТО1 = КТ \times 1 = \frac{1}{n}\sum_{i=1}^{n} КТ_{БСК_i}. \qquad (3)$$

Очевидно, ТО1 значно менше 1, що припускає повну тестовність об'єкта 1-го типу.

Таким чином, замінюючи БСК об'єктів 1-го типу відповідними БСК з вмонтованим механізмом тестування,

отримуємо повну керовність тестуванням об’єктів.

Тестування об'єктів 2-го типу. Об'єкти 2-го типу є посередниками stub/skeleton і генеруються у вигляді самостійних об’єктів-інтерфейсів для об’єктів 1-го типу. Таким чином, існують об’єкти інтерфейсу, що використовуються для взаємодії об’єктів 1-го типу.

Під тестуванням об'єктів 2-го типу розуміється *тестування інтерфейсів* взаємодіючих об'єктів і є самостійною процедурою, що не включає питання передачі повідомлень, які спеціально підтримуються мережними засобами (IPX, IP, TCP й інші).

Перевірка коректності здійснюється за допомогою спеціальної системної компоненти, названої Контролер інтерфейсів. Ця компонента є елемент РПС, що розташовується в мережному середовищі, як віддалений системний об'єкт. Для проходження через Контролер інтерфейсу у вихідний опис stub-об'єкта включаються спеціальні вмонтовані властивості спостереження за процесом обертання до віддаленого методу об'єкта і його перевірки до сервера і після його виконання.

Фактично зазначені дії аналізованого підходу забезпечують перевірку правильності інтепретації переданих і/або отриманих даних від взаємодіючих об'єктів мережі.

Для проведення тестування у вихідний опис об'єкта 2-го типу включаються спеціальні механізми, створюючи за своєю суттю вмонтовані властивості.

Метрики тестування об'єктів 2-го типу. Розглянемо метрики тестування об'єктів 2-го типу для систематичного і кількісного оцінювання результатів тестування за допомогою запропонованих моделей метрик для них. Основою тестування тут є інтерфейс.

Керування тестуванням інтерфейсів об'єкта 2-го типу – це контроль над поводженням об'єкта 1-го типу при взаємодії через заданий інтерфейс. Результатом керованого тестування інтерфейсу може бути: втрата запиту з віддаленим викликом (переданого або отриманого), відсутність об'єкта для взаємодії, перевірка поводження об'єкта на різноманітних етапах його життєвого циклу та ін.

Таким чином, керування інтерфейсами об'єкта розподіленого середовища ($У_И$) – це властивість керування об'єктом при його взаємодії з іншими об'єктами середовища умикання механізмів тестування в процесі виконання.

Тестовність об'єктів 2-го типу – це властивість незалежного керування всіма інтерфейсами об'єкта. Фізичний зміст $УТ_О$ – це властивість примушувати об'єкт виконувати визначені інтерфейси в режимі тестування, спонукати об'єкт до визначеного поводження при взаємодії з іншими об'єктами 1-го типу. $УТ_О$ опишемо за формулою:

$$УТ_О = \frac{1}{n}\sum_{i=1}^{n} У_{И_i}\ , \qquad (4)$$

де $У_{И_i}$ – керування j–інтерфейсом об'єкта 2-го типу РПС.

Область визначення $УТ_О$ – [0; 1]. Якщо в об'єкта $УТ_О = 1$ то він цілком керуючий; якщо УТ = 0 об'єкт – не керуючий. У інших випадках – об'єкт частково керуючий.

Спостереженість за тестуванням інтерфейсу об'єкта 2-го типу (СІ) – це властивість перегляду значень стимулювання запиту або відповіді після взаємодії об'єктів. *Спостереження тестування об'єкта 2-го типу* ($СТ_О$) – це властивість незалежного спостереження за всіма інтерфейсами об'єкта 1-го типу за допомогою вмонтованих механізмів:

$$СТ_О = \frac{1}{n}\sum_{i=1}^{n} Н_{И_i}\ , \qquad (5)$$

де $Н_{И_i}$ – можливість спостереження за *i*–інтерфейсом об'єкта.

Кількісні значення СІ і $СТ_О$ такі:

НІ = 1, якщо значення стимулювання або відповіді інтерфейсу можна спостерігати;

НІ = 0, якщо в інтерфейсу значення не можна спостерігати.

Область визначення $СТ_О$ збігається з областю НІ. Якщо в об'єкта $СТ_О = 1$

або $СТ_О = 0$, то він являє собою цілком досліджуваний або цілком не досліджуваний об'єкт. Реалізувати повну спостережність не так просто, як це можна зробити в нерозподіленому застосуванні, оскільки на тестовність ресурсів не накладається обмежень, пов'язаних із точками виконання.

Очевидно, що чим більше значення $УТ_О$ і $СТ_О$, тим легше провести повне тестування об'єкта 2-го типу РПС. Базуючись на визначеннях $УТ_О$ і $СТ_О$, отримуємо, що *тестовність об'єкта 2-го типу* (ТО2) – це функція від тестовності та спостережності даного об'єкта:

$$TO2 = f(УТ_О, НТ_О) = УТ_О * НТ_О. \quad (6)$$

Дана формула дає уявлення про значення ОТ2, якщо:

ТО2 = 1, якщо й $УТ_О$ і $НТ_О$ рівні 1.

ТО2 = 0, якщо хоча б $УТ_О$ або $СТ_О$ рівні 0.

Областю значень ТО2 $\in$ [0;1], при $УТ_О \in [0;1]$ і $СТ_О \in [0;1]$. Розкриваючи дану формулу, отримуємо:

$$TO2 = УТ_О \times НТ_О = \frac{1}{n}\sum_{i=1}^{n} У_{И_i} \times \frac{1}{n}\sum_{i=1}^{n} Н_{И_i}. \quad (7)$$

Зазначимо, що при тестуванні об'єктів 2-го типу важливу роль відіграє тестовність об'єкта 1-го типу, в яких відбувається взаємодія, тобто для повного тестування інтерфейсу необхідно отримати тестовність двох об'єктів 1-го типу, що дає можливість провести порівняльний аналіз інтепретацій інтерфейсів системним Контролером.

Тестування об'єктних РПС. Після тестування об'єктів 1-го і 2-го типу та встановлення ступеня їх тестовності, починається тестування окремих ПС у комплексі з використанням усіх механізмів тестування об'єктів. Тестовність РПС (Тоорп) *визначимо через тестовність об'єктів 1-го і 2-го типу ПС:*

$$T_{ООРП} = \frac{1}{m}\sum_{j=1}^{m}\left(TO1_j \times TO2_j\right), \quad (8)$$

де $ТО1_m$ – тестовність об'єкта 1-го типу;

ТО2m – відповідного йому об'єкта 2-го типу;

m – кількість розподілених об'єктів у об'єктно-орієнтованих ПС. Підставимо замість цих розмірів відповідні формули, отримуємо:

$$T_{ООРП} = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{n_j}\sum_{i=1}^{n_j} КТ_{БСК_i} \times \right.$$

$$\left. \times \frac{1}{k_j}\sum_{i=1}^{k_j} У_{И_i} \times \frac{1}{k_j}\sum_{i=1}^{k_j} Н_{И_i}\right), \quad (9)$$

де n_j – кількість БСК j-го об'єкта 1-го типу у складі об'єктно-орієнтованого ПС,

k_j – кількість інтерфейсів j-го об'єкта 2-го типу.

Дана формула показує, що якщо для всіх об'єктів ПС змінна ТО1 = 1 і ТО2 = 1, тестовність РПС приймає значення 1 і окреме ПС є цілком тестовано. У противному випадку тестовність РПС має бути поліпшена за допомогою інших методів. Базуючись на метричних моделях тестовність РПС на рівні керуючих структур БСК і спостережність за тестуванням об'єктів 1–го типу або прикладної частини додатка зробимо такий висновок.

Твердження. РПС розподілене застосування є цілком тестовним, якщо:

а) Тоопр = 1 або

б) ТО1j = 1, ТО2j = 1, j = 1, 2, ..., m.

Дане твердження і формули для ТО1 И ТО2, показують, що поліпшення тестовності РПС можна досягти на рівні БСК об'єктів 1-го типу і керуючих змінних об'єктів 2-го типу.

Запропоновано тестування РПС провадити за чотирма етапами:

1) тестування об'єктів 1-го типу (можливо традиційними методами тестування) незалежно від їхніх інтерфейсів;

2) тестування об'єктів 2-го типу на момент побудови як мінімум двох об'єктів РПС для перевірки їхньої взаємодії через інтерфейс;

3) тестування об'єктів 3-го типу, при проходженні повідомлень по мережі;

4) тестування РПС у комплексі з усіх об'єктів, коли перевірені функції об'єктів, їхні інтерфейси і треба перевірити протоколи-інтерфейси в динаміці їх виконання.

3. Операції взаємодії різнорідних об'єктів

Аналіз опису програм з використанням різних ТД показує, що головною проблемою взаємодії програм є інтерфейс, в якому дається набір параметрів з описом їх типів даних. Для забезпечення інформаційного зв'язування пари різномовних модулів склалися три класи операцій [2, 9]:

1) P – операції перетворення ТД модулів, що зв'язуються між собою;

2) S – операції селектора компонентів складного ТД;

3) C – операції конструювання структурних типів.

Операції класу 1 дозволяють здійснювати безпосереднє перетворення типу даних T_a^t у T_β^t без додаткових операцій зміни рівня структурування [2, 8]. Операцію перетворення ТД запишемо у вигляді: $P^{tq}_{a\beta} = (T_a^t , T^q_\beta)$.

Тут дані ТД T_a^t перетворяться в T^q_β, a і β відповідають мовам l_a і l_β. Множина ТД кожної МП впорядкована й індекси t і q визначають конкретні елементи цієї множини. Для деяких МП t і q будуть функціями від інших індексів і упорядкованість їх типів може визначатися конструюванням нового типу t з типів у яких індекси не більше t. Новий тип буде мати індекс, що функціонально залежить від індексів визначених типів і конкретних операцій конструювання.

Операції класу 2 слугують вибору зі структурного типу його окремих компонентів з меншим рівнем структурування. Механізми реалізації цих операцій відмінні від аналогічних, наявних у МП, тому що вони не повинні змінювати структури даних, тих, що безпосередньо обробляються в модулях.

Операції класу 3 є зворотними стосовно операцій селектора. Їхні механізми конструювання відмінні від аналогічних операцій, що є в МП.

Множина розглянутих операцій класу 1, 2, 3 охоплює перетворення ТД для МП, функції конструювання структурних типів і вибору окремих компонентів. Детальний розгляд показує, що для даної множини операцій повнота відсутня. Причини цього розглядаються далі.

Виходячи з наведених визначень постановка задачі перетворення ТД за заданим набором операцій класів 1–3 має такий вигляд.

Нехай дано клас $L = \{l_1, l_2, ..., l_n\}$ і для кожної з окремої мови відомі множини ТД і операції конструювання нових типів.

Необхідно: побудувати операції перетворення ТД $P = \{ P^{tq}_{\alpha\beta} \}$, операції селектора S і конструювання C.

Для рішення поставленої задачі проводимо розбивку множин фактичних і формальних параметрів і будування відображень ТД за допомогою операцій P, S і C. Якщо відображення одного типу до релевантного типу іншого не вдається, то це означає, зв'язування пари модулів не можливо або воно може бути реалізоване з порушенням деяких властивостей, що розглядалися.

Визначимо апарат опису ТД МП. Кожен ТД характеризується множиною значень, що можуть приймати змінні цього типу, і множини операцій над цими змінними.

Тому найбільш підходящим методом є опис ТД як алгебраїчних систем.

Операції перетворення типів $P^{tq}_{\alpha\beta}$ мають забезпечувати не тільки однозначну відповідність множини значень перетворюваних типів даних у модулях, які викликають, та тих модулях що викликаються, але й однакову інтерпретацію операцій над даними в цих модулях. При цьому має здійснюватися як пряме перетворення даних викликаючого до викликаного модуля, так і зворотне. При такому підході операції перетворення $P^{tq}_{\alpha\beta}$ відповідають ізоморфним відображенням однієї алгебраїчної системи в іншу. Тобто ми маємо справу з ізоморфним відображенням множин фактичних і формальних параметрів.

Висновок

У роботі розглянуто формальні підходи до проектування РПС, які базуються на ООП. Розроблено клас функціональних і інтерфейсних об'єктів, які моделюються при аналізі предметної області. Дано опис мови IDL для подання даних функціональних і інтерфейсних об'єктів, що забезпечують їхню взаємодію у межах сучасних середовищ. Запропоновано методику тестування цих об'єктів окремо і разом в комплексної структурі РС. Наведено формальні механізми перетворення з подання простих і складних типів даних МП. Обґрунтовано місце цього напряму дослідження при проектуванні і тестуванні РПС з різномовних програм й забезпечення їх взаємодії у сучасних середовищах типу Grid та Cloud Computing.

1. *Лаврищева Е.М.* Проблема интероперабельности разнородных объектов, компонентов и систем. Подходы к ее решению // Мат. VII Міжнар. конф. З програмування "Укрпрог-2008." –2008. – № 2–3. – С. 28–41.
2. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. Основы индустрии программных продуктов. – К.: Наук. думка, 2009. – 371 с.
3. *Siegel J.* CORBA Fundamentals and Programming, Wiley Co. Publ. Group, John Wiley & Sons, Inc. – USA. – 1996. – 694 p.
4. *Эммерих В.* Конструирование распределенных объектов // Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft COM и Java RMI. – М.: Мир, 2002. – 510 с.
5. *Андон Ф.И., Лаврищева Е.М*. Методы инженерии распределенных компьютерных приложений – К., Наук. думка, 1997. – С. 228
6. *Лаврищева Е.М.* Методы програмирования // Теория, инженерия, практика. – К.: Наук. думка. – С. 451.
7. *Лаврищева Е.М*. Сборочное программирование. Теория и практика // Кибернетика и системный анализ. – 2009. – № 6. – С. 3–12.
8. *Лаврищева Е.М.* Програмна інженерія. – Підручник.– К.: Академперіодика, 2008. – 317 с.

Одержано 24.02.2013

Про авторів:

Лавріщева Катерина Михайлівна,
доктор фізико-математичних наук,
професор,
завідувачка відділу програмна інженерія,

Стеняшин Адрій Юрійович,
аспірант Інституту програмних систем НАН України.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 4579.
E-mail: lavryscheva@gmail.com
andrey.stenyashin@gmail.com