УДК 681.03

К.М. Лавріщева, А.Л. Колесник

КОНЦЕПТУАЛЬНІ МОДЕЛІ РОЗПОДІЛЕНИХ КОМПОНЕНТНИХ СИСТЕМ

Запропоновані формальні моделі програмних систем (ПС) і їх сімейств (СПС) для їх використання у розподіленому середовищі. Їх проектування починається з об'єктної моделі предметної області (ПрО) і закінчується побудовою відповідних компонентних моделей. Основу ПС і СПС складають моделі інтероперабельності і варіабельності, які визначають принцип функціонування систем між собою. Зв'язуючою ланкою між ПС і СПС є інтерфейс, який передає дані між системами і середовищами для виконання і формування нових варіантів програмного продукту. Значна частина описаних моделей і операцій їх реалізацій уявлена на сайті інструментально-технологічного комплексу ІТК ІПС НАНУ.

Вступ

Процес побудови моделі ПрО або домену, орієнтованої на її розуміння людиною, називається концептуальним моделюванням. Проблема, яку моделює ПрО, формулюється у просторі її понять або на перехресті декількох пов'язаних між собою областей [1 – 3].

Кожна ПрО має притаманну їй систему понять, свої характерні властивості, відношення та правила поведінки. Концептуальна модель є базою знань для майбутніх користувачів. Ступінь формалізації моделі має бути достатньою, щоб забезпечувати точність та однозначність трактування базових понять і відношень між ними. Різними носіями інтересів у розробці, відповідної ПС та водночас не надмірною, щоб бути доступною для розуміння не тільки спеціалістом за фахом та спроможною відобразити деталі фахових проблем для різних верств майбутніх користувачів.

Найпершим аспектом моделювання домену для усіх відомих підходів може вважатися його понятійна база, тобто система понять, за допомогою якої формулюються усі аспекти проблеми. Понятійна база визначає не тільки термінологію, якою мають користуватися носії інтересів, що приймають участь у процесі аналізу вимог, а також суттєві відношення між поняттями та їх інтерпретація.

У ментальній діяльності людини застосовується ряд відношень, які дозволяють будувати похідні поняття та встановлювати між ними відношення та зв'язки. Серед таких відношень назвемо найпоширеніші:

– узагальнення як звуження істотних ознак поняття, при цьому розширюється коло охоплених поняттям об'єктів, тобто його обсяг;

– конкретизація як додавання істотних ознак, завдяки чому зміст поняття розширюється, а обсяг поняття звужується;

– агрегація як об'єднання ряду понять у нове поняття, істотні ознаки нового поняття при цьому можуть бути або сумою ознак компонент або суттєво новими;

– асоціація як найбільш загальне відношення, що стверджує наявність зв'язку між поняттями, не уточнюючи залежність їх змісту та обсягів.

Для окремих доменів чи предметних областей можуть використовуватися специфічні для них відношення. Ці відношення фіксуються в структурі понятійного базису проблемної області. При цьому сукупність термінології, понять, характерних для них відношень та парадигми їхньої інтерпретації у межах домену чи ПрО прийнято називати онтологією доменних знань.

Підходами для побудови моделі ПрО є об'єктний і компонентний. Відповідно з об'єктної парадигми ПрО будується з об'єктів і їх відношень, а компонентної парадигми проектування моделі ПрО базується на понятті компонента (component – C) з реалізованими

методами чи функціями, які можна використовувати, як повторні в майбутній архітектурі ПС чи СПС.

Об’єктний підхід до проектування ПрО

Об’єктно-орієнтований підхід (ООП) визначає стратегію побудови об’єктної моделі (ОМ) ПрО згідно теорії Г. Буча, відповідно якої увесь «світ» складається з об’єктів [1].

Визначення об'єкта, як базисного поняття ОМ, застосовується на двох припущеннях:

– для кожного об'єкта однозначно визначається його відповідність до сутності ПрО, включаючи відображення необхідних зв'язків та особливостей поведінки;

– засіб подання ПрО за об'єктним підходом визначає міру повноти відображення сутності та поведінки як об'єкта, або міру адекватності його сприйняття замовником.

Повнота відображення сутності, що задається об'єктом, відповідає мірі абстракції та її відображенню у час його опису. Для одночасного урахування обох вимог у запропонованої концепції запроваджується понятійна структура, що відповідає поняттю трикутника Фреге (рисунок) згідно якої об'єкт є *денотатом*. Символ використовується для ідентифікації об'єкта, а денотат відповідає рівню знань виконавця про сутність модельованого світу, що відбивається об'єктом.

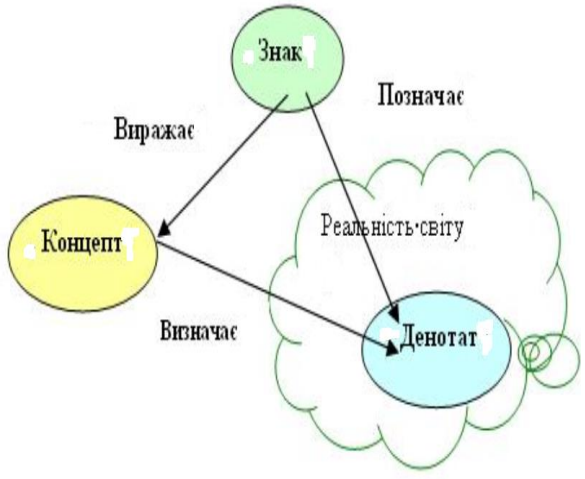

Рисунок. Трикутник Фреге

Природно, що денотат можна ідентифікувати різним чином, використовуючи для цього обрані алфавіти, та одному об'єкту можуть відповідати декілька концептів, відображаючи обраний рівень абстракції. Кожному трикутнику Фреге в обраній логічній системі буде відповідати певний об'єкт з власним ім'ям та певним змістом.

Тобто о**б’єкт** – це іменована частина дійсної реальності з певним рівнем абстракції відносно вибраної ПрО та з цілком обумовленої поведінкою. Поданням об’єкта є понятійна структура у вигляді трикутника Фреге.

При розгляду об’єктів ПрО використовується принцип відміни кожного окремого елемента ПрО з припущенням, що він визначається через його особливість – зовнішню і внутрішню.

Об'єкт, як певна сутність ПрО перебуває у різних станах і має певний набір операцій, які надають іншим об’єктам послуги (сервіси) для виконання методів. Об'єкти поділяються по класам за їх особистими властивостями і в суперкласи за загальними характеристиками (feature). ОМ включає самі об'єкти та їх інтерфейси, які відображають зв'язки між ними. Кожний об'єкт має власний стан і набір операцій, які впливають на стан інших об'єктів. Об'єкти приховують інформацію про значення станів, операцій і обмежують доступ до них.

До базових концепцій проектування ОМ належать такі:

– концепція узагальнення понятійній структурі ПрО, що відповідає формальному поняттю трикутника Фреге;

– теоретико-множинна концепція з операціями над об’єктами (об’єднання, різниці, декартового добутку тощо);

– логіко-алгебраїчна концепція для опису множини об’єктів як алгебраїчної системи з визначеною сукупністю предикатів певної сигнатури;

– концепція поведінки об’єктів у залежності від терміну їх функціонування та сформованих ними станів, що фіксуються в діаграми переходів станів і використовуються при виконанні об’єктів.

Ці концепції реалізуються на рівнях проектування ОМ з забезпеченням її повноти та коректності. Головною рисою рівнів проектування є зовнішні та внутрішні характеристики об'єктів та їх опис на заданій множині об'єктів з встановленням типів взаємовідношень: спеціалізація, агрегація, класифікація, асоціація, екземпляризація тощо.

Будь яка ПрО подає собою сукупність понять, пов'язаних деякою множиною відношень, яка визначає її поведінку на протязі часу.

Базові поняття мають наступний вигляд:

об'єктна орієнтація = об'єкти +
+ успадкування.

Кожне поняття ПрО, разом з його властивостями та особливостями поведінки у середовищі, є окремий об'єкт, а вся ПрО – це сукупність об'єктів зі зв'язками, які встановлюються на базі відношень між цими об'єктами. У ролі об'єкта виступають як абстрактні образи, так і конкретні фізичні предмети або групи предметів з зазначеною підмножиною їх характеристик та функцій.

Модель предметної області (ПрО)

Предметна область – це сукупність точних визначень понять, концептів об'єктів реального простору предметної області з їхніми характеристиками, а також множини синонімів і класифікаційних логічних взаємозв'язків між цими поняттями. Об'єкти як абстракції реального світу і понятєва структура відображають функції об'єктів і відношення між ними. Виділені в ПрО об'єкти структурно впорядковуються за допомогою теоретично множинних операцій (селекції, композиції, перетину тощо) та об'єднуються за загальними ознаками у класи й суперкласи. Модель ПрО подамо у вигляді

$$M_{ПрО} = \{Mo,\ Minc,\ Mcx, M_{ПС},\ P,\ D\},$$

де *Mo* – множина об'єктів і відношень між ними та відомостей у змісті функцій і даних об'єктів, з якими вони функціонують;

Minc – *модель* взаємозв'язку об'єктів між собою через інтерфейс;

Mcx – множина загальних характеристик пов'язаних об'єктів через дані та характеристики внутрішнього типу, що притаманні кожному об'єкту і входять до схеми композицій об'єктів задач ПрО;

$M_{ПС}$ – модель програмної системи чи $M_{СПС}$, які реалізують завдання та функції предметної області ПрО;

P – множина предикатів з порядку і умов виконання об'єктів за їх функціональними і не функціональними характеристиками моделі властивостей (featute) та взаємозв'язку об'єктів моделі *Mo*, методи яких забезпечують їх програмну реалізацію в ПС;

D – множина даних предметної області, які необхідні для виконання окремих компонентів та ПС у цілому та які можуть зберігатися у базах даних СУБД.

Підхід до подання формальних компонентних ПС

Парадигма компонентного програмування формулюється таким чином, щоб перебудувати об'єктну модель ОМ в компонентну, яка визначається на відповідній множині компонентів, які реалізують методи об'єктів моделі ПрО за їх інтерфейсами і властивостями, здатними до змін деяких елементів та механізмів взаємодії між ними і середовищем [3].

Тобто практично формується множина компонентів, адекватна множині функцій (методів) об'єктів ОМ. Створюється множина реальних компонентів С, з яких необхідно сконструювати ПС за певними функціональними і нефункціональними вимогами, що були сформульовані до об'єктної моделі. Завдання полягає у тому, щоб представити ПС з об'єктної моделі у вигляді компонентної моделі СМ за умови, що для будь-якого елемента об'єктної моделі існував програмний компонент із множини $C = (C1, C2, \ldots, CN)$ ОМ або він міг бути отриманий з іншої множини, як готовий ресурс з раніше розробленої ПС.

На практиці такі умови можуть виконуватися не завжди (наприклад, розроблена ПС має принципово нову функціона-

льність) або її функції удосконалені, але вони відповідають функціям об'єктів моделі О. У таких випадках реалізуються додаткові компоненти, множина *C* розширюється або зменшується за рахунок непотрібних. Це ініціює конфігурування нового варіанта ПС з урахуванням умов чи потреб замовника.

Таким чином, ПС створюються, як правило, з компонентів, що розробляються для реалізації функцій ПрО, чи відбираються готові компоненти повторного використання (КПВ), що відповідають новим функціям, накопиченим у сучасних бібліотеках чи репозиторіях.

Побудова ПС з компонентів виконується за такими принципами:

– композиційність складних систем з компонентів;

– інженерність компонентної діяльності з побудови реально існуючих об'єктів ПрО, відповідних множині компонентів і уніфікованих КПВ, каталогізованих у деяких сховищах для відбору нових, необхідних для ПС;

– інтероперабельність КПВ і ПС за їх інтерфейсами і правилами взаємодії компонентів між собою та іншими системами у різних середовищах їх функціонування;

– варіантність як здатність КПВ і компонентних ПС до змін шляхом заміни чи додавання нових функціональних КПВ у конфігураційної структурі ПС чи СПС;

– продуктивність ПС при обчисленні в сучасних гетерогенних середовищах з використанням особистих даних та даних, що накопичені у сучасних віртуальних сховищах даних [1 – 3].

Ці принципи є основополагаючими для побудови розподілених ПС. Значна частина побудови елементів моделей ПС і СПС з компонентів і КПВ програмно реалізовані в інструментально-технологічному комплексі (ІТК) [4] і на фабрики програм КНУ імені Тараса Шевченка [5].

Для побудови ПС розроблено формальну моделі ПрО, на основі якої будуються моделі ПС і СПС. Вони подаються не тільки формально, але і сучасними засобами типу патернів та workflow з метою їх реалізації в операційному середовищі для отримання готового програмного продукту ПС компонентного типу [6].

Далі розглядається низка формальних моделей подання різних моделей ПС і СПС, які сформовані і вдосконалені в рамках фундаментальних проектів ІПС НАНУ (2001 – 2012) на основі методу ОКМ [6, 7]. В ньому запропоновано метод проектування моделі ПрО із об'єктів за теоретико-множинними операціями для відображення об'єктів у моделі з рішення задач ПрО, метод трансформації абстрактних описів об'єктів у форму програмних компонентів шляхом їх реалізації і занурювання в операційне середовище з метою накопичення деяких готових КПВ та практичного виконання в ньому. ОКМ поданий сукупністю простих технологій реалізацій окремих частин цих систем і зборки їх методом конфігурації у нові ПС із готових компонентів – КПВ, сервісів і продуктів [5, 8, 9].

Модель програмної системи – $M_{ПС}$. Програмна система – сукупність окремих компонентів, які реалізовані з об'єктів ПрО, з установленими інтерфейсами, взаємопов'язаними з функціями деякої ПрО у заданому середовищі. Програмний компонент з загальної точки зору – це самостійний продукт (код) методу чи функції, що підтримує об'єктну модель, реалізує частину або усю окрему предметну область і уміє взаємодіяти з іншими компонентами системи через інтерфейси. Модель ПС дамо у такому визначенні:

$$M_{ПС} = \{L,\ Mf,\ Ms,\ Mi,\ Md\},$$

де $L = \{L1,\ L2,\ \ldots,\ Lk\}$ – мовні засоби опису функцій чи задач предметної області, одна з мов якої включає набір операцій (actions) алгебри, щодо дій з реалізації і виконання відповідних програмних компонентів;

Mf – множина функцій моделі ПрО, яка є по сутності моделлю об'єктів ПрО $Mf = \{O1,\ O2,\ldots,\ Or\}$ цієї ПрО, кожний об'єкт якої трансформується до компонентного коду цієї функції відповідної ПС ПрО;

$Ms = \{Ms^{in}, Ms^{out}, \ldots, Ms^{inout}\}$ – множина сервісів, що базується на моделі зв'язку і включає моделі передачі вхідних даних Ms^{in}, вихідних даних Ms^{out} та серверних даних Ms^{inout} ПС на сервері Application, базованих на множині системних сервісів (Common Facility Services) операційного середовища, які виконуються спеціальним брокером чи монітором сервісів тощо;

Mi – множина інтерфейсів у мові IDL з описом вхідних (in) і вихідних параметрів (out) та (inout) КПВ об'єднаного типу для пари зв'язних компонентів у структурі ПС. З практичної точки зору усі загальні типи даних специфікуються як зовнішні дані в паспорті моделі ПС з об'єктів ПрО;

Md – множина даних та метаданих предметної області ПС, які специфіковані в КПВ, як примітивні чи складні фундаментальні типи даних, що є в кожній мові програмування. ПС , а також в модулях посередниках (stub, skeleton) у мові IDL.

Множини *Minc, Ms* та *Mi* пов'язані з інтерфейсними та загальними даними і мають перетин за даними, що віднесені до in, out, inout, і входять до складу зовнішніх типів даних, якими обмінюються по мережі один компонентний об'єкт з іншим, що розташованій на сервері Application. За переданими даними виконуються компоненти і відправлять результати компоненту, що звернувся з запитом чи протоколом до серверного компонента.

Модель сімейства систем. Сімейство систем СПС – це сукупність ПС, які визначаються спільною множиною понять та множиною спеціальних понять і характеристик, що притаманні кожному члену цього сімейства з реалізації деякої ПрО.

Поняття сімейства програм ввів Дейкстра (1970), яке основується на тому, що програми мають спільну родинну мету, з якої можуть породжуватися різні варіанти інших програм, що потребують необхідного коректування чи замінювання некоректних або недостатньо визначених функцій, або для уточнення загальної постановки задач сімейства у цілому.

Модель СПС включає моделі ПС, задамо у вигляді:

$M_{СПС} = \{\{M_{ПС1}, M_{ПС2}, \ldots, M_{ПСk}\}, \{Mgf, Msf = \{Msf1, Msf2, \ldots, Msfk\}, Mi, Md\}\}$,

де $M_{ПС1}, M_{ПС2}, \ldots, M_{ПСk}$ – множина членів сімейства ПС;

Mgf – множина зовнішніх характеристик предметної області чи властивості загального типу, що визначають загальну термінологію та властивості усіх ПС сімейства;

$Msf = \{Msf1, Msf2, \ldots, Msfr\}$ – множина внутрішніх характеристик кожного окремого члена сімейства, поданого моделлю ПС ($M_{ПСi}$) тобто $Msf1$ для $M_{ПС1}, \ldots, Msfr$ для $M_{ПСk}$;

Mi – множина загальних інтерфейсів СПС, що подаються мовою IDL;

Minc – модель взаємозв'язку ПС через елементи множини *Mi ;*

$Md = \{Md1, Md2, \ldots, Mdk\}\}$ – множина даних, що визначають кожного члена ПС.

Монітор чи брокер запитів керує функціями ПрО, відправляє за протоколами типу RPC, RMI, Icontract дані, задані у *Minc* і специфіковані мовою інтерфейса у IDL чи API моделі клієнт-сервер і який розуміє інтерфейсний посередник (stub, skeleton).

Компонентна модель ПС – СМ. Компоненти які використовуються з множини С типизовані таким чином [1, 2]:

– прості компоненти, що реалізують деяку функцію ПрО;

– готові компоненти (КПВ, beans–компоненти у мові Lava тощо);

– складні компоненти (каркас, патерни тощо) зі схемою групування чи конфігурування компонентів;

– інтерфейсні посередники, що містять опис параметрів з забезпечення зв'язків між різними функціональними компонентами між собою;

– сервісні елементи, що підтримують різні системні дії з організації передачі параметрів і даних, представлених в інтерфейсах чи контрактах, а також керування

виконанням компонентів;

– засоби розгортання компонентів через конфігураційний файл у відповідному гетерогенному середовищі.

Таким чином, можливо сказати, що компонентне програмування є самостійним стилем програмування зі своєю концептуальною базою, теорією, методологією та інструментальними засобами. Він доповнює об'єктну парадигму засобами збирання чи композиції готових КПВ, що значно спрощує процеси розробки і супроводу ПС за рахунок:

– формалізації та впорядкованості процесу розробки окремих компонентів та систем з КПВ зі зменшенням часу розробки і збільшення якості та надійності ПС з компонентів;

– механізмів стандартного опису інтерфейсів компонентів для передачі даних компонентам, що будуть виконуватися в інтегрованому середовищі за допомогою відповідних загальносистемних сервісів та ін.;

– спрощення процесів побудови ПС за формалізованими моделями зв'язку, інтерфейсів та компонентної ПС та використання методів конфігурування для збирання компонентів з множини C у різні ПС з варіантами їх виконання.

Компонентна модель ПС є абстрактним уявленням ПС, що конструюється з компонентів, які є елементами цього уявлення, якому зіставлені реальні функції чи методи об'єктів ПрО для забезпечення реалізації функціональності ПС та дотримання не функціональних вимог щодо ПрО. Для однієї і тієї ж ПС може існувати множина компонентних моделей у залежності від концепцій проектування і використовуваного компонентного підходу. Модель компонентної моделі СМ відповідає цілком і повністю структура моделей $М_{ПС}$ або однієї $М_{ПС}$ з тою різницею, що тут використовуються готові програмні компоненти (КПВ), як ресурси різного призначення.

На загальному рівні подання моделі СМ ставиться завдання щодо представлення її функціональності згідно вимог у вигляді сукупності компонентів і їх інтерфейсів або контрактів для забезпечення взаємодії між собою окремих програмних компонентів, з яких складається модель.

$$i = 1$$

Нехай $\prod A_i$ – множина інтерфейсів за контрактами компонентів чи програм, що визначають її функціональність. Кожному A_i описує контракт як клієнт-серверну взаємодію з відповідними методами і структурами даних. Кожний інтерфейс має опис інтерфейсних даних, які вже раніш перераховані, а саме – In, Out і Inout. Їх будемо називати відповідно визначальним поданням інтерфейсу (поняття – вхідний, вихідний та проміжний інтерфейс). In_i визначає умови і мету контракту з боку клієнта, Out_i задає реалізацію компоненту з боку сервера, а Inout – властивості інтерфейсів $In_{i\,i}$ і Out_i

Визначення інтерфейсів Ai для компонентів моделі СМ можуть групуватися у різні поєднання з урахуванням семантики характеристик загального і внутрішнього типів, що визначені для моделі системи та компонентів і входять до складу компонентної моделі.

Довільна сукупність інтерфейсів In_i, Out_j, $Inout_{i\,j}$, де і ≅ J, може не входить в кожну таку сукупність і одночасно визначати пари програмних компонентів при їх збиранні в складну структуру.

Для кожної отриманої сукупності пар $Ci \cup Iv = \{Ci,\ Iv,\ OutJ,\ Inout_{ij}\}$ створюється модель компонента або шаблон, який містить деяку множину параметрів, які визначають і реалізують уявлення інтерфейсів. За цими уявленнями надалі здійснюється зіставлення шаблонів і інтерфейсів реальних компонентів.

Внаслідок операцій групування виходить множина пар $Ci \cup Iv$, що буде зватися контрактом компонентної моделі ПС. Для однієї і тієї ж програми існує множина таких моделей, які визначаються множинами контрактів (або інтерфейсів) зі способами розбиття на шаблони компонентів для послідовного їх виконання.

Загальне визначення алгебри компонентів ПС

Нехай X – множина об'єктів, де $x_i \in X$ – довільний об'єкт деякої ПрО, який задає функцію ПрО і пов'язаний з іншими об'єктами відношенням зв'язку. Цій множині відповідає множина програмних компонентів C ($C \in X$). Кожен компонент реалізує функцію і характеризується деякими притаманними йому властивостями (наприклад, варіантність, змінюваність тощо), які задаються в їх інтерфейсах.

Властивості чи характеристики компонентів задаються загальними і особистими типами за предикатами $P_1, P_2, ..., P_k$, виходячи з умов виконання компонентів на даних з множини xi ∈ X деякими операціями, результати виконання яких також будуть елементами множини X. Тривіальними прикладами таких операцій можуть бути – об'єднання кількох об'єктів в один або декомпозиція деякого програмного компоненту на окремі елементи.

Нехай $\prod R_i$ – множина операцій над об'єктами з X. Кожна операція має власну арність у залежності від її семантики. В цілому існують операції, які задаються на підмножинах з X, а також можна створювати підмножину операцій (наприклад, попередні операції об'єднання та декомпозиції).

Серед множини операцій розглядаються ті, які зберігають властивості компонентів і їх результатів. Ці операції $R_1, R_2, ..., R_m$ виконуються над будь-якими компонентами $c_1, c_2, ..., c_m \in C$ з урахуванням арності операції R_i та результату двох видів: $c_1, c_2, ..., c_m \in C$ та $c_1, c_2, ..., c_m \notin C$).

Множина C та операції $R_1, R_2, ..., R_m$ визначатимуть алгебру компонентів. Прикладами реальних операцій над компонентами можуть бути операції розширення інтерфейсів і факторингу для сукупності нових компонентів за іншими інтерфейсами, але з такою ж функціональністю. Для кожної алгебри відповідає властивостям компонентів Pi та операцій Ri, формально визначених для множини компонентів і які застосовуються в конкретної теорії компонентного програмування щодо конструювання програм і систем з готових КПВ.

Визначення загальної множини компонентів. Вищерозглянута множина компонентів C є математичною абстракцією. У реальному випадку мається на увазі деяка множина реальних компонентів з $c_i \in C$. Застосувавши до цієї множини операції $R_1, R_2., .., R_m$ можна отримати множину, яку будемо називати замиканням множини С'. Цей термін обраний для того, щоб підкреслити, що С' є максимально можливою множиною компонентів, які можна отримати на основі вибраних компонентів з множини С, відповідно до $c_i \in C$. У реальних процесах конструювання програм розглядається множина компонентів C (або C'), але при цьому враховується, що для нього справедливі всі положення відповідної компонентної алгебри при конкретній побудові компонентних ПС, виконаної практично в ІТК [5].

Моделі взаємодії і варіабельності ПС в сучасних середовищах

Під *взаємодією* розуміють сумісність двох і більше об'єктів (програм і середовищ між собою), яка пов'язана з обміном інформацією і використанням її для організації обчислень у сучасних операційних середовищах. Для завдання взаємодії програм у мовах програмування (МП) використовується апарат зв'язку типу CALL, а для розподілених програм – RPC, RMI, ORB (stub, skeleton), IContract тощо. Взаємодія здійснюється інтерфейсом, який специфікується мовою IDL (Interface Definition Language) допомогою якої на загальному рівні передаються дані компонентам, що знаходяться в різних гетерогенних середовищах [7]. Для опису взаємодії різнорідних компонентів нами розроблено модель взаємодії. Головне її призначення – забезпечувати взаємодію різнорідних компонентів, що побудовані в інтегрованому середовищі

ІТК з Eclipse, MS.Net, CORBA, Protégé тощо [4, 5, 8 – 12].

Модель взаємодії M_{inter} має такий загальний вигляд:

$$M_{inter} = \{M_{pro}, M_{sys}, M_{env}\},$$

де $M_{pro} = \{Com, Int, Pro\}$ – модель програми, *Com* – компонент, *Int* – інтерфейс, *Pro* – програма;

$M_{sys} = \{PS, Int, Prot\}$ – модель програмної системи *PS*, *Int* – інтерфейс, *Prot* – протокол зв'язку для передачі даних; через мережу;

$M_{env} = \{Envir, Int, Pro\}$ – модель середовища, в якому *Int, Pro* відображають сукупність зовнішніх інтерфейсів, викликів та протоколів, що забезпечують передачу даних між програмами або системи через мережу відповідно.

Фактично *Minter* щодо стандартної семирівневої моделі відкритих систем OSI, є моделлю верхнього рівня.

Програмні компоненти і системи й інтерфейси специфікуються відповідною МП, інтерфейсом IDL, протоколами з XDL, RDF, що зберігаються в бібліотеках і репозиторіях ІТК. Новим способом взаємодії між програмами типу клієнт і сервер є інтерфейс типу *Icontract* системи WCF [ ], який призначений для опису атрибутів та операцій для передачі даних від сервісного об'єкта (*Service consumer*) до клієнтського (*Service provider*) за контрактами Icontracts.

Базисом інструментального середовища ІТК є Eclipse, як засіб керування репозиторієм КПВ і його використання для зборки в складну структуру. В процесі взаємодії програм і систем через інтерфейс при обчисленнях у тому чи іншому середовищі застосовуються різні механізми перебудови типів даних з різних форматів даних їх опису і формою трансформації типів даних у системах програмування з МП та використання бібліотек функцій і процедур з примітивами перетворення різних даних.

Практичним рішенням задач взаємодії в ІТК є моделі розподілених систем:

– *Visual Studio.Ne↔ Eclipse* реалізує взаємодію програм у мові C# на основі опису специфікації інтерфейсу, перенесення готового продукту в репозиторій системи Eclipse, що відображає зв'язок з даним середовищем програм через плагини і конфігураційний файл з параметрами і операціями оброблення даних у даному середовищі;

– *Corba↔Java↔MS.Net* забезпечує встановлення зв'язків між цими програмними середовищами шляхом розміщення програм в МП у репозиторії ІТК та надання доступу до них інших програм;

– *IBM ↔Eclipse* орієнтована на нові програми в МП з використання інтерфейсу, що допускаються у цьому середовищі та у віртуальному варіанті VSphere.

Тобто при переході в середовище Eclipse використовуються вихідні файли програми, *dll*–бібліотки VS та файли ресурсів (*.resx*). Коли необхідно знов перейти із цього середовище в Visual Studio, вся проектна програма імпортується туди і зворотньо. Обчислення програми та її змінювання можна виконувати як у середовищі Eclipse, так і у Visual Studio. Модель IBM Eclipse буде продовжено надалі у випадку використання сервісів для взаємодії програм, як обчислень у розширеному середовищі ІТК.

Модель варіабельності ПС і СПС.

Варіабельність це властивість деякого програмного об'єкта до розширення, змінювання, пристосування або конфігурування з метою використання у визначеному контексті та забезпечення подальшого еволюціонування. Програмування ПС як членів СПС з КПВ у парадигмі СПС стала важливим засобом для забезпечення еволюційності ПС за рахунок побудови різних варіантів продукту в залежності від зміни деяких функцій деяких окремих компонентів [4, 5, 11, 12].

СПС із ПС має спільну множину характеристик, відповідних потребам певних функціональних варіантів ПрО, розрізняються способами втілення цих характеристик в окремі ПС і розробляються з використанням готових ресурсів. ПС створюються не «з нуля», а породжуються за моделлю СПС або обираються для неї го-

тові КПВ з репозиторіїв, після чого адаптуються і збираються у нову структуру ПС [ ].

Модель СПС пов'язана з об'єктною і її компонентною моделлю ПрО і доповнена точками варіантності, створив розширену модель сімейства варіантної системи (СВС) за таким формальним виглядом:

$$M_{CBC} = (Mnc, \text{Int}, C),\ C = CC \cup \cup BC;$$

$$Bc = \langle ib, rb, \varnothing \rangle BC \Leftrightarrow \exists! o_3^{**} \in$$

$$\in L_3 \mid ib = < \langle met(o_3^{**}),$$

$$dat(o_3^{**}) \rangle,\ \{rb\} = im(ib);$$

$$CC = \langle ic, rc \rangle \in CC \Leftrightarrow \exists! o_3^{*} \in O_3 \setminus L_3 \mid ic =$$

$$= \langle met(o_3^{*}), dat(o_3^{*}) \rangle,$$

$$BC = \{ \langle met(o_{3u}), dat(o_{3u}) \rangle,$$

$$\text{Part\_of}\ (o_3^{*}, o_{3u})\},\ \{rc\} = im(ic),$$

де $bc \in BC$ і CC – унікальні імена КПВ, названих *базовими* та *складними*;

ib, rb – вхідний інтерфейс і реалізація базового КПВ, яка підтримує інтерфейс базового КПВ: $\forall (bc_1, bc_2) \in BC\ \ im(bc_1) \neq$

$\neq im(bc_2)$;

ic, rc – вхідний s вихідні інтерфейси та реалізація складного КПВ;

im(·) – функція, що надає множину реалізацій інтерфейсу;

$cr_{ij} = (c_i, c_j) \in Int$ – відношення Part_of: $(c_i, c_j) \in NC \Leftrightarrow \text{Part\_of}(o_{3i}, o_{3j})$, що пов'язує КПВ, прототипи яких o_{3i}, o_{3j} в об'єктній моделі.

Компонентна модель програмного КПВ входить в модель *CBC*. Для забезпечення властивостей варіантності й змінності ПС з СПС проведено узагальнення наведених моделей для послаблення відношення Part_of.

Нехай C_t: $O_t \to \{0;1\}$ і D_t: $\{vp_t \in O_t \mid C(vp_t) = 1\} \otimes 2^{Ot} \to \{0;1\}$, $t = 1,...,4$ – деякі предикати, які для об'єкта $o_{tj} \in O_t(C_t, D_t)$ декомпозує *варіант* об'єкта $o_{ti} \in O_t$ $\left(VP\left(C_t, D_t; o_{ti}, o_{tj}\right)\right)$, якщо реалізація в певній ПС об'єкта тягне за собою реалізацію нового об'єкта тоді й тільки тоді, коли

$$C_t(o_{ti}) = 1, \quad \exists\ SO_{ti} \subseteq O_t\{o_{ti}\} \mid o_{tj} \in$$

$$\in SO_{ti},\ D_t(o_{ti}, SO_{ti}) = 1,\ t = 1,...,4.$$

Об'єкти o_{ti} і o_{tj} у даному виразі будемо звати *об'єктною точкою варіантності* та *варіантом* підпорядкованого об'єкта для o_{ti}, а предикати C_t і D_t – *обмеженням* на точки варіантності й *залежністю* між варіантами.

Процес розроблення СПС як програмного продукту розроблено на Вебсайті, у вигляді декількох спрощених технологій.

Висновок

Наведені формальні моделі ПС і СПС та моделі взаємодії і варіабельності з використанням КПВ пройшли апробацію в ІТК і відображені в ряді наведених публікацій. Вони є нові й оригінальні, розроблені за участю аспіранта даної роботи та старшого наукового співробітника Слабоспицкої О.О. при виконанні фундаментального проекту ІПС НАНУ «Розробка теоретичного фундаменту генеруючого програмування та інструментальних засобів його підтримки"(2007 – 2011). Основні результати проекта опубліковані в заключному звіті та в е-монографії, дисертаційної роботі та реалізовані на Веб-сайті http://sestudy.edu-ua/com.

1. *Лавріщева К.М.* Взаємодія програм, систем й операційних середовищ // Проблеми програмування. – 2011. – № 3. – С. 13–24.
2. *Лавріщева К.М., Слабоспицька О.О., Коваль Г.І., Колесник А.О.* Теоретичні аспекти керування варіабельністю в сімействах програмних систем. – Вісник КНУ, серія фіз.-мат. наук. – 2011. – № 1. – С. 151 – 158.
3. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. Основы индустрии програмных продуктов. – Киев: Наук. думка, 2009. – 377 с.
4. *Колесник А.Л.* Механізми забезпечення варіабельності в сімействах програмних систем // Проблеми програмування. – 2010. – № 1. – С. 35–44.

5. *Колесник А.Л.* Підходи до забезпечення варіабельності схем баз даних у сімействах програмних систем // Молодь і наука: сучасний стан, проблеми та перспективи національного відродження України: Матеріали II Міжвузівської студентської науково-практичної конференції. – К.: МНТУ, 2010. – С. 323–324.
6. *Лавріщева К.М., Коваль Г.І., Бабенко Л.П. та ін.* Нові теоретичні засади технології виробництва сімейств програмних систем у контексті ГП – Електронна монографія, ДРНТІ. – № 67–УК–2011 від 5.10.11. – 377 с.
7. *Лаврищева К.М.* Компонентне програмування. Теорія і реалізація // Проблеми програмування. – 2012. – № 4. – С. 3–18.
8. *Лавріщева Е.М., Зинькович В.М., Колесник А.Л. та ін.* Інструментально-технологічний комплекс розробки та навчання прийомам виробництва програмних систем. – Державна служба інтелектуальної власності України. – Свідоцтво про реєстрацію авторського права на твір. – № 45292, від 27.08.2012.
9. *Lavrischeva E., Ostrovski A., Radetskyi I.* Approach to E–Learning Fundamental Aspects of Software Engineering // 8-th International Conf. ICTERI – 2012 "ICT in Education, Research and Industrial Applications". – Publ. springer – sbm.com/ocs/home/ ICTERI–2012.
10. *Lavrischeva E., Ostrovski A.* General Disciplines and Tools for E– Learning Software Engineering. – http://senldogo0039.springer-sbm.com/ocs/
11. *Kolesnyk A., Slabospitskaya O.* Tested Approach for Variability Management Enhancing in Software Product Line. – In: Ermolayev V., Mayr H.C., Nikitchenko M. et al. (eds.): ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer. Proc. // 8-th Int. Conf. ICTERI 2012, Kherson, Ukraine, June 6-10, 2012, CEUR-WS.org/Vol: 848, – P. 125–133.
12. *Slabospitskaya O., Kolesnyk A.* The Model for Enhanced Variability Management Process in Software Product Line – Preproc // of 4th International United Information Systems Conference (UNISCON 2012), Crimean State Humanitarian University, Yalta, Ukraine, June 1 – 3, 2012 – Available at http://www.uniscon.org/images/ uniscon/downloads/UNISCONPapers.zip

Одержано 10.10.2012

Про авторів:

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

Колесник Андрій Леонідович,
молодший науковий співробітник.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, Київ–187,
проспект Академіка Глушкова, 40.
Тел.: 526 3470.