УДК 681.3

П.П. Ігнатенко, В.М. Бистров, О.П. Ігнатенко, В.М. Ткаченко

ЗАДАЧІ ТА ЗАСОБИ МОДЕЛЮВАННЯ Й ОЦІНЮВАННЯ ЖИТТЄЗДАТНИХ ПРОГРАМНИХ СИСТЕМ

Розглянуті задачі та засоби моделювання й оцінювання життєздатних об'єктно-орієнтованих програмних систем (ПС). Проблема вирішується в рамках створення моделі ПС і моделі простежування змін для забезпечення вимог до системи. Підхід до оцінювання артефактів об'єктно-орієнтованих ПС заснований на аналізі проекту системи, специфікованому у вигляді діаграм UML. Для оцінки атрибутів якості ПС пропонується використовувати метрики, які орієнтовані на шаблони проектування та розглядаються у контексті сценарного підходу. Зв'язок між внутрішніми та зовнішніми атрибутами ПС проектується на архітектуру ПС та визначає напрямок їх перетворень.

У роботі [1] розглянуто підхід до створення *життєздатних (viable)* ПС, який базується на передбаченні можливих змін у системі залежно від її типу, на прогнозуванні наслідків змін, прийнятті рішень щодо їх доцільності, ефективності та оптимальності. Згідно цього підходу життєздатність ПС повинна забезпечуватись спеціальними засобами, імплементованими в середовище розробки та функціонування ПС. На стадії супроводження ПС також виникає необхідність в адаптації, модифікації чи реінжинірингу ПС згідно виникаючих змін, а, отже, в отриманні нових оцінок атрибутів якості ПС, таких, як *функціональність, надійність, зручність використання, ефективність, супроводжуваність та переносність* [2, 3]. Таким чином, в середовище розробки та супроводження ПС повинні входити засоби моделювання та оцінювання ПС, які б використовувались при внесенні змін до ПС або їх адаптації. У даній статті розглядаються питання розробки та застосування таких засобів для об'єктно-орієнтованих ПС.

Однією з основних властивостей *життєздатних* програмних систем є їх здатність до адаптації, ефективних модифікацій та реінжинірингу при зміні функціональних і нефункціональних вимог до ПС протягом життєвого циклу [1—3]. Така властивість відповідає, зокрема, наявності варіантів проектних рішень (варіантності ПС) та можливості їх оцінювання на основі моделі ПС. Задачі, що відносяться до проблеми забезпечення варіантності ПС, можуть розглядатись як у контексті передбачення вимог до системи, з'ясування їх варіантності, специфікації та конкретизації [2], так і у контексті варіантності артефактів, специфікації та конкретизації механізмів, що підтримують таку варіантність і забезпечують різні атрибути якості артефактів. У першому випадку задача стосується проблеми повторного використання та забезпечення адаптивних властивостей систем та більшою мірою відноситься до макропроцесу розробки ПС, у другому — проблеми створення внутрішніх архітектурних та програмних релізів системи із заданими властивостями і більшою мірою відноситься до мікропроцесу розробки. Окрема задача полягає у забезпеченні можливості оцінки артефактів на основі аналізу проекту системи, який специфіковано відповідно до певних засобів моделювання. У даній статті розглядаються задачі моделювання, що виникають на рівні мікропроцесу розробки та стосуються забезпечення варіантності артефактів та їх оцінювання. При цьому мікропроцес об'єктно-орієнтованої розробки розглядається як такий, що зумовлюється артефактами та архітектурними продуктами, що породжуються та послідовно уточнюються на рівні макропроцесу [4].

Як засіб моделювання ПС передбачається уніфікована мова моделювання (Unified Modeling Language, UML) [5], базована на діаграмах та призначена для моделювання систем на основі об'єктно-орієнтованого підходу. Загальні механізми мови UML —

прийняті розподіли (Common divisions), механізми розширення (Extensibility mechanisms) — підтримують варіантність артефактів і забезпечують можливість керування їхніми версіями. UML також володіє засобами моделювання механізмів проектування систем з урахуванням представлень їхньої архітектури [4, 6]. Відокремлення представлення архітектури з точки зору прецедентів (use case view) від інших видів та розгляд сценаріїв — екземплярів прецедентів (instances of use cases) як артефактів, що реалізують варіантність архітектури ПС, надає можливість моделювання варіантності ПС засобами UML та її оцінювання на рівні мікропроцесу розробки. Можливості використання UML в аспекті представлення варіантності вимог до ПС на рівні макропроцесу розробки розглянуті у [7].

Як механізми проектування ПС, що підтримують створення ПС із заданими властивостями, розглядаються архітектурний стиль (architectural style), архітектурні зразки (architectural patterns) або каркаси (frameworks), зразки або шаблони проектування (design patterns). Їх систематизоване та послідовне використання на рівні мікро- та макропроцесу об'єктно-орієнтованої розробки в рамках ітеративного життєвого циклу сприяє визначенню та стабілізації архітектури ПС, а також утворенню архітектурних та програмних релізів, що відповідають необхідним функціональним та нефункціональним вимогам до ПС. Архітектурний стиль спрямовує та визначає організацію системи у цілому: статичні та динамічні елементи, їх інтерфейси, кооперації та способи об'єднання. Архітектурні зразки моделюють певну інфраструктуру, яку у подальшому планується повторно використовувати або адаптувати до конкретного контексту. Зразки проектування надають типове вирішення типової проблеми у даному контексті. Вони використовуються для моделювання типових ситуацій, що виникають при проектуванні. Використання архітектурного стилю та архітектурних зразків пов'язано із архітектурним плануванням відповідно до макропроцесу розробки, а зразків проектування — із тактичним проектуванням відповідно до мікро- та макропроцесу розробки [8]. Серед зразків проектування слід відокремити шаблони GRASP (General Responsibility Assignment Software Patterns — загальні шаблони розподілення обов'язків у програмних системах) [9]. Вони більшою мірою відповідають мікропроцесу розробки, дозволяють встановити зв'язок із макропроцесом та забезпечують варіантність проектних рішень щодо утворення артефактів із різними атрибутами якості.

У даній статті висвітлені наступні питання:

- запропоновано концепцію моделювання та оцінювання об'єктно-орієнтованих ПС при їх адаптації, модифікації та реінжинірингу протягом життєвого циклу;
- наведено класифікацію можливих змін в об'єктно-орієнтованих ПС відповідно до архітектурних перетворень системи з метою досягнення певних нефункціональних вимог;
- запропоновано підхід до оцінювання об'єктно-орієнтованих ПС на основі застосування метрик, що орієнтуються на шаблони проектування та розглядаються у контексті сценарію, висвітлено технологічні аспекти його реалізації.

Стаття відображає основні результати, які отримані в рамках теми "Розробка теоретичного підґрунтя та засобів забезпечення супроводження прикладних програмних систем" згідно плану фундаментальних досліджень ІПС НАН України.

1. Концепція моделювання та оцінювання життєздатних ПС

Основними задачами моделювання та оцінювання життєздатних ПС є:

- задачі моделювання та забезпечення варіантності проектних рішень при необхідності адаптації, модифікації та реінжинірингу ПС;
- задачі визначення для основних характеристик якості ПС сукупно-

сті зовнішніх та внутрішніх показників або атрибутів ПС, що характеризують якість ПС, визначення відповідних метрик для їх виміру та алгоритму їх інтеграції відповідно до зовнішніх характеристик ПС.

Задачі моделювання та оцінювання ПС розглядаються в рамках моделі простежування змін та концепції сценарного підходу [10, 11]. Відповідно до архітектури процесів життєвого циклу [2, 3] модель простежування змін підтримує процеси розробки, керування конфігурацією та, зокрема, процес внесення змін. Вона визначає зв'язок між артефактами ПС відповідно до різних етапів життєвого циклу та процесу розробки, а також механізми, що зумовлюють процес перетворень архітектури ПС. Концепція сценарного підходу розглядає сценарій як базовий артефакт, що підтримує варіантність ПС та визначає механізми забезпечення варіантності проектних рішень, артефакти, що підлягають оцінюванню, конкретизує модель якості на рівні мікро- та макропроцесу об'єктно-орієнтованої розробки в рамках ітеративного життєвого циклу. Модель якості ПС підтримує процес керування якістю та процес забезпечення гарантії якості. Вона визначає атрибути якості ПС відповідно до нефункціональних вимог до системи, внутрішні та зовнішні атрибути ПС, а також зв'язок між ними, внутрішні та зовнішні метрики для вимірювання атрибутів ПС тощо. Відповідно до концепції сценарного підходу модель якості повинна враховувати варіантність проектних рішень, наявність певних шаблонів проектування, що приводять до створення артефактів із заданими атрибутами якості, надавати можливість оцінювати системи на рівні мікро- та макропроцесу розробки.

Пропонований підхід до оцінювання ПС включає:

- розробку засобів та методів оцінювання ПС на основі використання моделі ПС у вигляді діаграм UML;
- розробку та застосування метрик, що орієнтовані на шаблони проектування та відповідають різним поглядам на архітектуру ПС;
- встановлення базових артефактів ПС, розробка та оцінювання яких визначають якість ПС у цілому.

Для вирішення задач *моделювання та оцінювання ПС* пропонується побудувати *моделі ПС*, *варіанти сценаріїв для підтримки варіантності ПС* та *моделі простежування змін*.

1.1. Концепція сценарного підходу та узагальнення поняття сценарію. Проблема забезпечення варіантності ПС, що розглядається у даній статті, базується на концепції сценарного підходу. Відповідно до цієї концепції поняття сценарію охоплює як функціональний аспект розробки ПС, так і аспект розробки, пов'язаний із забезпеченням нефункціональних вимог до ПС, об'єднує мікро- та макропроцес розробки. Узагальнення поняття сценарію в рамках сценарного підходу приводить до розгляду наступних аспектів варіантності ПС.

1. Варіантність функціональності ПС на рівні мікро- та макропроцесу розробки, яка відповідає діхотомії "клас/об'єкт" [12]. При цьому сценарій розглядається як екземпляр прецеденту та являє собою конкретну послідовність подій, що ілюструє та виражає деякий аспект поведінки системи. Сценарій абстрагує вимоги користувача та ідентифікує основні абстракції (такі, як класи, механізми, процеси, підсистеми). Шляхом розширення сценарію (утворення нового варіанту прецеденту) протягом ітеративного та інкрементного процесу розробки ПС нарощується функціональність системи. На кожній ітерації відбувається вимірювання та оцінка системи.

2. Варіантність нефункціональних вимог на рівні мікропроцесу розробки, яка відповідає діхотомії "інтерфейс/реалізація" [12] і розглядається у контексті сценарію та певних механізмів проектування. При цьому сценарії розкривають певні функціональні точки системи, визначають інтерфейси системи, а поведінка абстракцій, що належать сценарію, реалізується та

ілюструється діаграмами об'єктів і діаграмами взаємодій. Утворення варіантів сценаріїв здійснюється шляхом перерозподілу обов'язків між об'єктами та визначення інтерфейсів, що забезпечують взаємодію між сценаріями. Такі перетворення відбуваються при незмінній функціональності сценаріїв. При кожному перетворенні відбувається вимірювання та оцінка артефактів у контексті сценарію.

3. Варіантність нефункціональних вимог на рівні макропроцесу розробки, яка підтримується розширенням поняття сценарію щодо варіантів реалізації нефункціональних вимог під час архітектурних перетворень системи на основі певних механізмів проектування. При цьому нефункціональні вимоги розкриваються у вигляді послідовності сценаріїв, за допомогою яких відбувається специфікація змін у системі для досягнення певних атрибутів якості.

Варіантність проектних рішень закладається на етапі аналізу та проектування, коли визначаються основні функціональні точки системи [4] та утворюється її архітектура. Функціональні точки розкриваються у вигляді сценаріїв згідно функціональних вимог до системи. Семантика сценаріїв при специфікації моделі системи у вигляді діаграм UML пояснюється та ілюструється діаграмами об'єктів та взаємодій. Варіанти сценаріїв породжуються відповідно до нефункціональних вимог шляхом послідовного використання шаблонів GRASP. Архітектура системи визначається групуванням та розподіленням функціональних точок по шарах та розділах архітектури з урахуванням їх варіантності та встановленням відповідних інтерфейсів за допомогою інших шаблонів проектування, зокрема шаблонів GoF [12]. При цьому архітектура системи базується на певних архітектурних зразках або каркасах, в рамках яких вона еволюціонує та супроводжується. На етапі еволюції та супроводження утворюються архітектурні та еволюційні релізи ПС відповідно до варіантності, яку закладено до систем. Внесення змін до ПС та підтримка варіантності здійснюються в рамках певних моделей простежування та процесу керування змінами. Рішення щодо реінжинірингу ПС приймається за умови, коли вимоги до адаптивності ПС та ефективності модифікацій не задовольняються або вимагають витрат, що перевищують витрати на їх реінжиніринг.

Таким чином, запропонована концепція сценарного підходу узагальнює поняття сценарію відповідно до мікро- та макропроцесу проектування ПС у контексті досягнення певних функціональних та нефункціональних вимог до ПС. Розглядаючи функціональні точки (сценарії) як точки деякого фазового простору, а шаблони проектування як певні керуючі дії, що спрямовують процес розробки та супроводження ПС, можна побудувати імітаційну модель оптимального керування еволюцією ПС. Координатами фазового простору є певні міри, що відповідають різним атрибутам якості ПС. Функціональні точки, що належать різним варіантам сценаріїв, утворюють реліз системи, який відповідає потрібному рівню атрибутів якості ПС. Траєкторії функціональних точок визначають напрямок еволюції ПС, який забезпечує нефункціональні вимоги до них. Архітектурний стиль та архітектурні шаблони (а взагалі, і обмеження на розробку ПС) та відповідні зовнішні атрибути ПС утворюють обмеження для тих архітектурних перетворень, що відбуваються у ПС на рівні мікропроцесу розробки.

1.2. Оцінювання ПС. Проблема оцінювання ПС щодо їх відповідності певним функціональним та нефункціональним вимогам повинна розглядатися в контексті наступних аспектів, що мають визначальний для неї характер: *технологія розробки системи; шаблон процесу розробки; мова та засоби моделювання; підхід до оцінювання атрибутів якості (quality attributs, QA) ПС; зовнішні та внутрішні метрики відповідно до шаблонів проектування та поглядів на архітектуру ПС*.

Як атрибути якості ПС можна розглядати характеристики та підхарактеристики якості ПС [2, 3], головні з них — *відмовостійкість, зрозумілість, реактивність*, *зручність супроводження*, *можливість повторного використання* тощо, які відповідають нефункціональним вимогам до процесу розробки (development non-functional requirements, development NFR), а також *надійність* (*reliability*), *продуктивність* (*performance*) тощо, які відповідають нефункціональним вимогам до експлуатації ПС (*operational NFR*).

Зазначимо, що основною та найбільш складною задачею з тих, що розглядаються, є встановлення зв'язків між внутрішніми та зовнішніми атрибутами ПС. При цьому внутрішні атрибути, такі, як зв'язність, зачепленість тощо, звичайно стосуються окремих логічних та фізичних структурних сутностей ПС різного рівня складності (клас, кооперація, компонент тощо). Отримання оцінок для зовнішніх атрибутів ПС на основі залучення існуючих внутрішніх метрик або розробки нових відповідно до усіх складових елементів системи складає досить важку задачу. Тому конструктивним підходом при розробці та оцінюванні ПС є вибір такого артефакту, проектування та оцінка якого є найбільш доцільним та важливим з погляду одержання якісної ПС в цілому. Такими артефактами ми пропонуємо вважати сценарії та відповідні діаграми об'єктів та діаграми взаємодій. При цьому внутрішні атрибути системи будемо відносити до тих артефактів, які складають сценарій, а зовнішні — власне до сценарію. Відповідно до цього положення і повинні розроблятися та використовуватися внутрішні та зовнішні метрики. Тобто, проблема оцінювання проектних рішень ПС розглядається у контексті вибраного сценарію.

1.3. Теоретичний аспект. У теоретичному аспекті проблема оцінювання у процесі внесення змін до артефактів ПС з метою досягнення певних значень атрибутів якості системи не має формальних методів вирішення. Це пов'язано з так званим *інваріантом розробки ПС,* який полягає у наявності таких властивостей ПС, як складність, піддатливість, мінливість і невідчутність [13], завдяки яким процеси розробки, модифікації чи реінжинірингу ПС, що гарантують досягнення певних значень атрибутів якості системи, не можуть бути строго детермінованими. Це зумовлено також тим, що перетворення архітектури ПС під час ітеративного процесу розробки системи є неізоморфними. На кожній ітерації можуть з'являтися нові архітектурні сутності та сутності предметної області, змінюватися (зокрема, ускладнюватися) їхня структура та зв'язки. Отримані оцінки на поточній ітерації щодо значень атрибутів якості системи не гарантують тієї ж якості на новій ітерації. Це, зокрема, означає, що задача побудови повної та несуперечливої системи оцінок артефактів ПС з метою досягнення у процесі ітеративної розробки системи необхідних значень атрибутів її якості нині не має строгого вирішення.

1.4. Практичне вирішення. Для практичного вирішення проблеми пропонується побудова *моделей простежування змін ПС.* Визначимо основні їх аспекти:

- к*атегорії змін*, що відбуваються у системі протягом життєвого циклу;
- *зв'язки між артефактами* ПС, які забезпечують внесення змін до системи відповідно до шаблону процесу розробки та етапів життєвого циклу;
- *напрямок внесення змін* під час здійснення *перетворень* у системі згідно певних шаблонів проектування для забезпечення необхідної функціональності та відповідності нефункціональним вимогам.

Категорії змін відповідають певним фазам життєвого циклу системи, рівням архітектурних перетворень та типам артефактів, які підлягають змінам. *Зв'язок між артефактами* забезпечує необхідну послідовність внесення змін для певного шаблону процесу розробки. *Направленість змін* визначається результатом вимірювання та отримання оцінок щодо внутрішніх та

зовнішніх атрибутів системи на основі певних методів та засобів оцінювання в рамках моделі якості.

Важливим аспектом даної проблеми є розробка механізмів підтримки моделі простежування змін, які б забезпечували можливість доступу до моделі системи з метою оцінювання ПС. Ця задача повинна вирішуватися на основі створення прикладного програмного інтерфейсу (Application Programming Interface, API) та застосування компонентної технології для взаємодії механізмів підтримки моделі простежування у середовищі розробки та функціонування системи. У [5] розглянута задача побудови таких механізмів для оцінювання артефактів ПС на відповідність шаблонам GRASP і отримання інтегральних оцінок, що визначають атрибути якості системи. Запропоновані механізми забезпечують доступ до функціонального коду системи та за допомогою створювання та обробки так званих *спеціальних ситуацій* на етапі функціонування ПС дозволяють накопичувати певним чином специфіковану інформацію у репозиторії. Інший підхід полягає у побудові таких механізмів, які дозволяють проводити оцінювання системи та підтримувати процес внесення змін на різних етапах життєвого циклу на основі моделі системи у вигляді діаграм UML.

Загальна схема моделювання та оцінювання рішень щодо змін в об'єктно-орієнтованих ПС представлена на рисунку.

1.5. Вимоги до середовища створення та функціонування ПС. Опишемо основні висхідні умови до створення та функціонування ПС, у контексті яких вирішується проблема оцінювання, що розглядається у даній статті.

Розглянемо *об'єктно-орієнтовану технологію розробки ПС*, яка базується

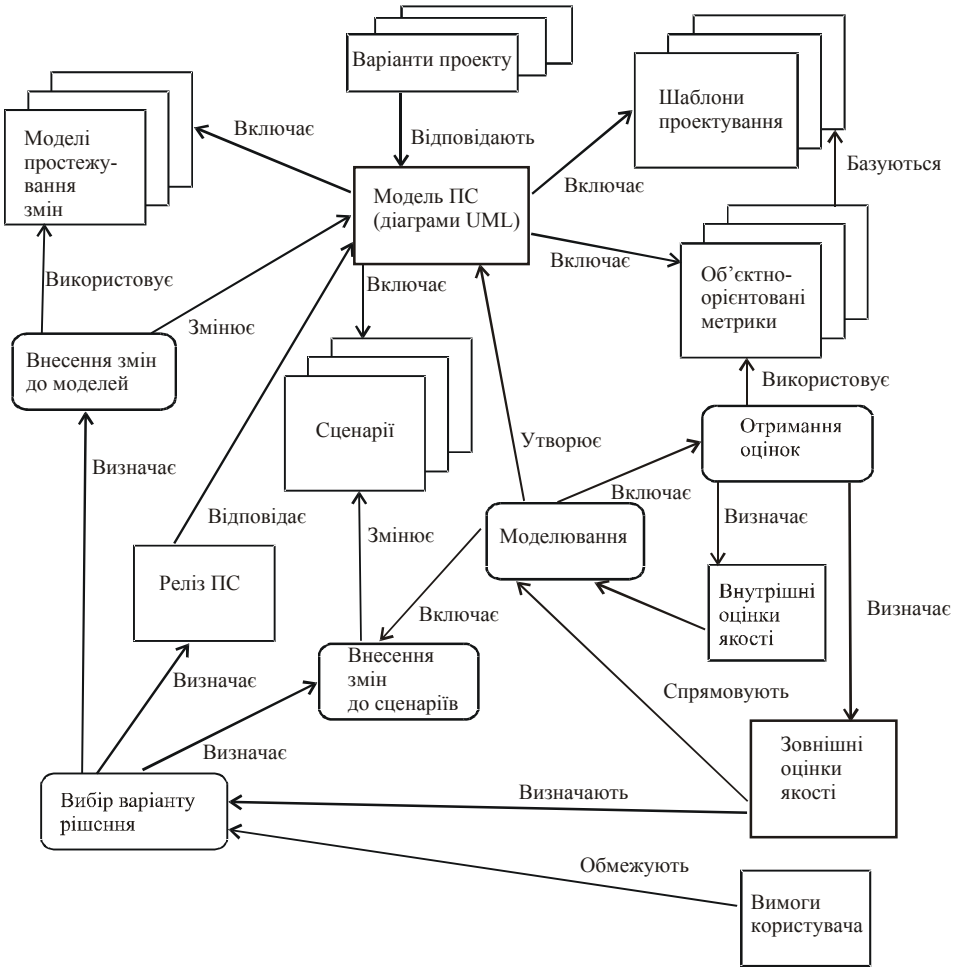

Рисунок. Схема моделювання та оцінювання об'єктно-орієнтованих ПС при їх створенні, модифікіції та реінжинірингу

на об'єктній моделі [4]. Використання такої моделі є найбільш сприятливим для створення ПС, що відповідають вимогам масштабованості (scale adjustment), можливості повторного використання (reusability) та зручності супроводження (maintainability). Вибір об'єктно-орієнтованої технології зумовлює вибір відповідного шаблону процесу розробки, мови та засобів моделювання, а також об'єктно-орієнтованих метрик для підтримки вимірювань протягом життєвого циклу ПС.

У пропонованому підході як *шаблон процесу розробки ПС* розглядається *раціональний уніфікований процес* (*Rational Unified Process*) [5]. Цей процес розробки ПС заснований на архітектурі, керується прецедентами використання, є ітеративним та інкрементним, включає аналіз якості і керування ризиками. Одне з положень методології Rational Unified Process звучить у такий спосіб: "Ітеративний процес — це процес, спрямований на керування потоком версій програмного забезпечення, що виконуються. Процес з нарощуванням можливостей (incremental) — це процес, спрямований на безперервну інтеграцію системної архітектури для створення (виробництва) цих версій, при цьому кожна версія містить в собі удосконалені можливості у порівнянні з попередніми" [9]. Тобто використання цього шаблону процесу розробки відповідає такому оцінюванню артефактів ПС, яке враховує перетворення логічного рівня, коли з кожною ітерацією нарощується функціональність системи (за рахунок ускладнення версій прецедентів) та певні архітектурні перетворення, що приводять до зміни атрибутів якості у напрямку нефункціональних вимог до ПС.

Як уже зазначалося, *мовою моделювання ПС* вибирається *уніфікована мова моделювання UML.* Мова UML добре узгоджується з раціональним уніфікованим процесом та призначена для візуалізації, специфікації, конструювання та документування артефактів ПС [12]. Оцінювання ПС на *фазі проектування (design phase)* може відбуватися на основі аналізу діаграм UML і тих сутностей (*класів*, *інтерфейсів*, *кооперацій* тощо) та відношень (*залежностей*, *асоціацій* тощо), що їх утворюють. Таке оцінювання може проводитися з використанням як простих об'єктно-орієнтованих метрик (наприклад, кількість класів, атрибутів, асоціацій, глибина дерева наслідування), так і метрик, що базуються на певних зразках (наприклад, кількість Singleton зразків [13] та відповідних екземплярів у діаграмі). Крім того, для оцінки архітектури системи, представленої як набір діаграм UML, може бути використано результат витягання зразків (pattern mining results) або антизразків на основі аналізу цих діаграм. Також важливою особливістю UML, яка стосується оцінювання ПС, є можливість відображення системної архітектури із різних точок зору та їх взаємодії. Це погляди з точки зору: *проектування* (Design view), *процесів* (Process view), *реалізації* (Implementation view), *розгортання* (Deployment view). Статичні аспекти Design view передаються діаграмами класів та об'єктів, а динамічні — діаграмами взаємодій, станів та дій. Статичні та динамічні аспекти Process view візуалізуються тими ж діаграмами, що і Design view, але при цьому особлива увага приділяється активним класам, які представляють нитки (thread) та процеси. Статичні аспекти Implementation view передаються діаграмами компонентів, а динамічні — діаграмами взаємодій, станів та дій. Статичні аспекти Deployment view описуються діаграмами розгортання, а динамічні — діаграмами взаємодій, станів та дій. Погляд з точки зору прецедентів (Use case view) об'єднує наведені погляди на архітектуру системи.

Як *засіб розробки ПС* розглядаються CASE-засіб Rational Rose компанії Rational Software [5]. Це інструмент аналізу і проектування об'єктно-орієнтованих ПС на основі мови UML. Він дозволяє створювати та супроводжувати UML-діаграми, здійснювати пряме та зворотне проектування тощо. На даний час цей інструмент є найбільш

потужним та широко уживаним серед існуючих засобів розробки. Розробка засобів доступу до UML-діаграм, що зберігаються у моделях Rational Rose, для обчислення метрик програмного забезпечення до початку реалізації проектних рішень складає важливу задачу оцінювання ПС.

2. Архітектурні перетворення ПС

Можна виділити наступні види змін, що відбуваються у ПС протягом життєвого циклу.

- Зміни, що пов'язані з розробкою нової системи та зумовлюються необхідністю виконання функціональних та нефункціональних вимог користувача за наявності певних обмежень на розробку. Розробка ПС здійснюється відповідно до шаблону процесу розробки — раціонального уніфікованого процесу, який об'єднує мікро- та макропроцес.
- Зміни, що пов'язані з етапом супроводження ПС. Відповідно до змін ПС розглядаються наступні стадії [13]: *підтримка експлуатації*, *адаптивне супроводження*, *поліпшуюче супроводження*.
- Зміни, що відповідають наслідуваній системі для реконструкції її проекту та часткової реалізації у новому вигляді (реінжиніринг).

При змінюванні нефункціональних вимог до системи протягом її життєвого циклу архітектура системи також змінюється для досягнення нових атрибутів якості. Такі зміни розглядаються як архітектурні перетворення. Наступні п'ять категорій змін визначаються тими факторами, які зумовлюють перетворення архітектури відповідно до нефункціональних атрибутів ПС.

Перетворення архітектури відповідно до обраного архітектурного стилю. Існує декілька архітектурних стилів, запровадження яких дозволяє досягти певних атрибутів якості ПС, тобто їх застосування приводить до покращення або погіршення тих чи інших нефункціональних вимог до ПС. Певний стиль, наприклад, такий, що відповідає багаторівневій архітектурі, збільшує гнучкість (flexibility) ПС, але зменшує її продуктивність. Придатний стиль необхідно вибирати залежно від того, досягнення яких нефункціональних вимог є пріоритетним. При цьому треба зважати на те, що зміна архітектурного стилю зажадає суттєвої чи навіть кардинальної переробки ПС. Тобто такі рішення повинні прийматися до стадії планування та аналізу ПС або на перших ітераціях розробки системи.

Перетворення архітектури відповідно до архітектурних зразків, що використовуються. Архітектурний зразок відрізняється від архітектурного стилю тим, що він не є переважним щодо впливу на архітектуру ПС. Він також відрізняється від зразку проектування тим, що впливає на архітектуру ПС у цілому або принаймні на значну її частину. Архітектурний зразок нав'язує скоріше певне правило створення проекту ПС, яке специфікує поведінку системи в одному аспекті, наприклад паралельність (concurrency) або стійкість (persistency).

Перетворення архітектури відповідно до застосування зразків проектування. Вплив зразків проектування на архітектуру ПС є найменш значним. Наприклад, для багаторівневої архітектури такі зразки можуть визначати структуру певного рівня щодо перерозподілу обов'язків між прикладними або програмними сутностями, які належать цьому рівню. Вони також можуть зажадати появи нової структурної сутності на іншому рівні. Такі перетворення відповідають, зокрема, шаблонам GRASP.

Зміни, пов'язані з переведенням (converse) нефункціональних вимог до функціональних. Таке перетворення розширює архітектуру ПС за рахунок внесення певної функціональності, яка не має відношення до проблемної області. Наприклад, застосування механізму обробки виключень під час реалізації певної функції (не змінюючи призначення функції) приводить до поліпшення відмовостійкості (fault-tolerance) певного компонента ПС.

Зміни, пов'язані з розподілом (або перерозподілом) вимог. При цьому певний системний рівень розподіляється між підсистемами або компонентами та для кожного такого компонента визначаються нефункціональні вимоги. Тобто початкова нефункціональна вимога розглядається як певна композиція вимог до компонентів для деякого рівня архітектури.

Перелічені фактори, що відповідають наведеним категоріям, також визначають направленість змін у системі для досягнення певних значень атрибутів якості.

3. Підхід до оцінювання об'єктно-орієнтованих ПС

Розглянемо основні особливості вимірювання об'єктно-орієнтованих ПС у пропонованому підході. Для оцінювання атрибутів якості довільних ПС можуть бути застосовані наступні основні підходи.

1. *Оцінки, що базуються на сценаріях.* Для того щоб оцінити деякий атрибут якості (наприклад, maintainability QA), розроблюється певна кількість сценаріїв, які конкретизують даний атрибут. Кожний такий сценарій визначає деяку низку змін у системі, яка стосується прикладних та архітектурних сутностей ПС і прийнята до розгляду у контексті даного сценарію. Далі проводиться та підраховується кількість перетворень у системі, необхідних для її адаптації відповідно до кожного сценарію. Кінцевий результат аналізується на предмет кількості сценаріїв, що вдалося реалізувати шляхом перетворень різного типу, зокрема із використанням шаблонів проектування. Цей результат визначає оцінку певного атрибута якості.

2. *Імітація.* Застосовуються засоби та методи імітаційного моделювання для розгляду та оцінювання необхідних перетворень у ПС. При цьому використовується інформація, що характеризує вже реалізовані компоненти.

3. *Математичне моделювання.* Базується на розробці та застосуванні математичних моделей або метрик, особливо для оцінювання експлуатаційних нефункціональних вимог.

4. *Об'єктивна аргументація.* Цей підхід ґрунтується на логічній аргументації прийнятих проектних рішень.

Основою пропонованого підходу до оцінювання атрибутів якості як ПС, так і проектів змін до ПС є використання метрик ПС, які засновані на шаблонах проектування та враховують повноту їх застосування. При цьому стандартом середовища для проектування систем пропонується середовище Rational Rose, де логічному представленню відповідають діаграми класів та взаємодії, представленню процесу — діаграми процесів, представленню розробки — діаграми розробки, фізичному представленню — діаграми компонентів і, нарешті, сценарному — діаграми способів використання.

Представлення процесу враховує такі нефункціональні вимоги, як продуктивність (performance) та придатність (availability) системи. Це відповідає властивостям паралелізму (concurrency), розподіленості (distribution), цілісності (integrity) системи та відмовостійкості (fault-tolerance). Представлення розробки бере до уваги такі внутрішні вимоги, як легкість розробки, керування розробкою, повторне використання або загальність та обмеження, що накладаються на систему інструментальними засобами розробки. Фізичне представлення враховує такі нефункціональні атрибути, як придатність, надійність (відмовостійкість), продуктивність та масштабованість. Логічне представлення взагалі підтримує функціональні вимоги — сервіси, які система надає користувачу. У контексті сценарного підходу логічне представлення з'ясовує архітектурні елементи системи у процесі архітектурного проектування та ідентифікує механізми та загальні елементи системи.

Для одержання оцінок рішень підхід повинен об'єднувати, наприклад, метрики логічного представлення та метрики представлення процесу. В підході основною ланкою, що зв'язує всі представлення між собою, є сценарії.

Тому і метрики представлень повинні взаємодіяти зі сценарними метриками. Сценарні метрики вимірюють характеристики конкретних задач, які повинна виконувати система, тому вони інваріантні відносно рівня реалізації. Наприклад, час очікування виконання запиту можна оцінити на рівні діаграми класів (кількість класів запиту тощо), діаграми взаємодії (кількість елементарних взаємодій між об'єктами), діаграми компонентів (кількість та потужність процесорів, що задіяні) або інших.

Дослідження та розробка методів і засобів оцінювання можливих змін ПС спирається в підході на моделі простежування змін. На основі цих моделей передбачається оцінювати якість сценаріїв, що визначають найбільш важливі та критичні властивості і функції проекту ПС. У рамках підходу сценарії найбільш природним образом забезпечують варіантність проектних рішень і використовують такий важливий стандарт для об'єктно-орієнтованого проектування і розробки ПС, як UML.

Як критерії оцінки сценаріїв пропонується розглядати їх відповідність шаблонам GRASP. Ці шаблони використовуються в процесі створення діаграм взаємодій при розподілі обов'язків між об'єктами, розробці способу їхньої взаємодії. Такі діаграми безпосередньо пов'язані зі сценаріями, і їхня успішна реалізація визначає успіх усього проекту.

На основі інтегральних оцінок сценаріїв пропонується формувати альтернативні сценарії, визначати найкращий сценарій і вибирати кращу реалізацію. Головна задача полягає у виявленні механізму, що дозволяє оцінювати сценарії по шаблонах. Для самих шаблонів відомий їхній зв'язок із властивостями компонентів ПС, що повинні задовольняти необхідним вимогам.

Кожен шаблон характеризується своїми відмінними рисами, що забезпечують [9]:

- підтримку інкапсуляції, надійність і пристосовність до обслуговування (Expert, Low Coupling);
- простоту розуміння і підтримку визначень класів, спрощення підтримки і внесення змін (Expert, High Cohesion);
- зменшення витрат на супровід і можливості повторного використання (Creator, Low Coupling);
- поліпшення умов повторного використання компонентів стосовно логіки процесів предметної області на рівні реалізації, спрощення підтримки і доробки (Controller, High Cohesion);
- можливість контролю стану прецеденту для виконання системних операцій у визначеній послідовності (Controller, Expert, High Cohesion).

Аналіз шаблонів GRASP показує, що застосування їх під час створення діаграм взаємодій приводить до задачі оптимізації використання шаблонів відповідно до існуючих вимог й обмежень на розробку як компонентів ПС, так і всієї системи.

При оцінюванні об'єктно-орієнтованих ПС у контексті сценарію та шаблонів розподілення обов'язків можуть розглядатись наступні атрибути ПС:

— розмір програми, що відповідає класам, які утворюють сценарій;

— зв'язність (coupling) або ступінь глибини зв'язків між класами та об'єктами;

— зачепленість (cohesion) між класами та об'єктами для реалізації функціональності;

— спадкування (inheritance) відповідно до зв'язку підкласів та суперкласів;

— повторне використання (reuse) класів та об'єктів сценаріями, що утворюють різні релізи ПС.

Для вимірювання цих атрибутів можуть бути запропоновані наступні метрики:

— кількість методів класу, що підтримують його функціональність в рамках сценарію, та кількість методів класу, що підтримують інтерфейс сценарію;

— розмір коду класів, що належать сценарію;

— зв'язність між об'єктами або кількість інших класів сценарію, з

якими зв'язаний даний об'єкт у складі сценарію з (без) урахуванням ієрархії спадкування;

— відношення кількості методів класу, які безпосередньо використовуються у контексті сценарію, до їхньої загальної кількості;

— кількість атрибутів класу, які використовуються методами даного класу, по відношенню до загальної кількості атрибутів, що підтримують функціональність класу;

— глибина дерева ієрархії класів та кількість нащадків класу відповідно до певного рівня ієрархії;

— кількість керувань, які надаються даному класу для підтримки функціональності сценарію та його інтерфейсу.

Відповідно до сценарного підходу та узагальненого поняття сценарію зовнішнім атрибутом ПС, що визначає їх якість, може бути перелік шаблонів, які охоплюють сутності ПС, зокрема класи та об'єкти, а зовнішньою метрикою — кількість класів та об'єктів, охоплених екземплярами шаблонів, у порівнянні з їхньою загальною кількістю в ПС. Витяг шаблонів може бути здійснений як на основі аналізу діаграм UML (зокрема, діаграм класів), так і програмного коду системи. Інші зовнішні атрибути ПС можуть бути визначені шляхом поширення сценарного підходу на макропроцес проектування. При цьому може бути сформована послідовність сценаріїв у вигляді вимог до системи, реалізація яких шляхом архітектурних перетворень ПС зумовлює досягнення певного рівня атрибута якості ПС. Таким чином підтримується варіантність ПС на рівні макропроцесу проектування ПС. Кількість реалізованих сценаріїв із загальної кількості є мірою певного атрибута якості ПС.

Інформація про шляхи еволюції (track evolution) ПС та результати вимірювання за допомогою певних метрик повинна накопичуватися у репозиторії (history data repsitory). Надалі вона може використовуватися для аналізу (зокрема, методами статистичного аналізу) та вибору оптимальних шляхів створення нових версій ПС. При цьому, безумовно, важливу роль відіграє логічна аргументація з боку розробника системи щодо створення варіанту сценарію у контексті забезпечення необхідних нефункціональних вимог до ПС.

Подальшим напрямком розробки проблеми оцінювання ПС є обґрунтування повної та мінімальної системи метрик, яка б давала оцінку поточним атрибутам системи в контексті заданого сценарію, аналіз зв'язку вимог до ПС та зовнішніх атрибутів системи, створення інструментаріїв керування процесом верифікації якості ПС.

Висновки

Проблема моделювання й оцінювання життєздатних ПС при їх модифікації та реінжинірингу на стадії розробки і супроводження ПС відноситься до основних проблем програмної інженерії.

Перспективним підходом до її вирішення є застосування технології розробки та створення життєздатних ПС, яка базується на запропонованих засобах підтримки життєздатності: *моделі ПС, специфікованої у вигляді діаграм UML*; варіантності артефактів на основі сценарного підходу; *моделі простежування змін в ПС*; *системі оцінювання ПС на основі метрик, що базуються на шаблонах проектування ПС та розглядаються у контексті сценарію*; *CASE-засобах Rational Rose* компанії Rational Software на основі мови UML.

Розробка технології створення життєздатних ПС має велике теоретичне і практичне значення для інформатизації змінних (evolutional) систем з підвищеними вимогами до стійкості функціонування, зокрема систем спеціального призначення.

1. *Ігнатенко П.П.* Проблеми забезпечення життєздатності програмних систем та підходи до їх вирішення // Пробл. программирования — 2002. — №3—4. — С. 58—73.
2. *Бабенко Л.П., Лаврищева Е.М.* Основи програмної інженерії. — Київ: Знання, 2001. — 269 с.
3. *Основы* инженерии качества программных систем / Ф.И. Андон, Г.И. Коваль, Т.М. Ко-

ротун, В.Ю. Суслов. — Киев: Изд. дом «Академпериодика» НАН України, 2002. — 502 с.
4. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на С++: 2-е изд., пер. с англ. — М.: Бином, 2000. — 560 с.
5. *Боггс У., Боггс М.* UML и Rational Rose. — М.: Лори, 2000. — 582 с.
6. *Kruchten Ph.* The 4+1 View Model of Architecture: Current Practice // IEEE Software. — 1995. — **12**(6), Nov. — P. 42–50.
7. *Бабенко Л.П., Поляничко С.Л.* Підхід до атестації властивостей компонентів програмних систем засобами UML // Пробл. программирования. — 2001. — №1–2. — С. 35–41.
8. *Буч Г., Рамбо Д., Джекобсон А.* Язык UML: Руководство пользователя. — М.: ДМК, 2000. — 432 с.
9. *Ларман К.* Применение UML и шаблонов проектирования. — М.: Вильямс, 2001. — 496 с.
10. *Підхід* до забезпечення реінжинірингу об'єктно-орієнтованих програмних систем / П.П. Ігнатенко, В.М.Бистров, І.О. Заєць, О.П. Ігнатенко // Пробл. программирования. — 2002. — №1–2. — С.98–108.
11. *Scenarios* in System Development: Current Practice / K. Weidenhaupt, K. Polh, M. Jarke, P. Haumer // IEEE Software. — 1998. — March/April. — P. 34–35.
12. *Design* Patterns, Elements of Reusable Object-oriented Software / E. Gamma, R. Helm, R. Johnson, J. Vlissides. — N.–Y.: Addison–Wesley, 1995. — 345 p.
13. *Мацяшек Л.* Анализ требований и проектирование систем. Разработка информационных систем с использованием UML: Пер. с англ. — М.: Изд. дом "Вильямс", 2002. — 432 с.

Отримано 25.04.03

Про авторів

Ігнатенко Петро Петрович
кандидат технічних наук, завідувач відділом

Бистров Віктор Михайлович
кандидат фізико-математичних наук, ст. наук. співробітник

Ігнатенко Олексій Петрович
аспірант

Ткаченко Володимир Миколайович
аспірант

Місце роботи авторів:
Інститут програмних систем НАН України
просп. Академіка Глушкова, 40,
Київ-187, 03680, Україна
Тел. (044) 266 1540