УДК 681.3

ОСОБЛИВОСТІ ЗАБЕЗПЕЧЕННЯ ЖИТТЄЗДАТНОСТІ ПРОГРАМНИХ СИСТЕМ В УМОВАХ ГЕНЕРУЮЧОГО ПРОГРАМУВАННЯ

П.П. Ігнатенко,В.М. Бистров

Інститут програмних систем НАН України,
03187, Київ, проспект Академіка Глушкова, 40.
Тел.:+380 (44) 526 3309, факс: 380 (44) 526 6263,
e-mail: ignat@isofts.kiev.ua

Розглянуті особливості застосування модельно-орієнтованого підходу до розробки життєздатних прикладних програмних систем у рамках технології породжуючого програмування.

Рассмотрены особенности использования модельно-ориентированного подхода к разработке жизнеспособных прикладных программных систем в рамках технологии порождающего программирования.

Вступ

Поняття життєздатності, як важливої властивості складних організаційних систем, уперше застосував С. Бір [1]. У результаті аналізу діяльності організацій ним було запропоновано практичний підхід до забезпечення їх життєздатності. У його основу покладено запропоновану автором формалізовану модель життєздатної складної системи організаційного керування (VSM, Viable System Model). У складі такої життєздатної системи має бути сукупність визначених автором п'яти типів підсистем, які забезпечують її життєздатність. С. Бір у своїй моделі зазначив шлях створення життєздатних систем у промисловості та суспільстві, які б найкращим чином відповідали закону необхідної різноманітності Р. Ешбі [2] та детально проаналізував механізми зменшення (atttnuation) і збільшення (amplification) різноманітності в життєздатних системах для вирішення можливих протиріч у них. Важливою властивістю моделі життєздатної системи С. Біра є те, що основна мета її типових підсистем більш високого порядку - не "заважати" підпорядкованим підсистемам, а основна мета їх "втручання" (intervention) є забезпечення гомеостатичного регулювання більш високого порядку. Як альтернативна для моделі VSM, може розглядатися теорія живучих систем Міллера (Living System Theory, LST). Теорія LST є моделлю системи-систем біологічних організмів і організацій. Вона припускає концептуальний поділ на вісім взаємодіючих рівнів. На кожному рівні, всі системи складені з 20 критичних підсистем: це фундаментальні інваріанти LST. Підсистеми згруповані за принципом того, як вони обробляють матерію-енергію (matter-energy) або інформацію. Вісім підсистем обробляють матерію-енергію, десять – інформацію процесу, і дві оперують обома групами. Застосування LST для розробки життєздатних систем (зокрема, інформаційних) підтримується в рамках інтегрованого концептуального підходу, який називається Living Systems Process Analysis (LSPA) [3].

Інваріанти моделей VSM та LST утворюють певний теоретичний базис, на основі якого можуть розроблюватись життєздатні програмні системи. Відповідність існуючих програмних систем моделям VSM та LST може розглядатись як критерій їх життєздатності.

Застосування моделі LST для розробки життєздатних ПС заслуговує на увагу з того погляду, що на її основі можуть бути визначена архітектура системи в рамках застосування шаблонів проектування на різних структурних рівнях організації ПС. Так, в [3] визначені утворююча роль парадигми керування, яка представлена архітектурним стилями модель-представлення-оброблювач (Model-View-Controller) та багаторівнева система (Layers), а також визначено множину зразків GoF [4], які формують структуру життєздатної ієрархічної системи. Розрізнюючи поняття організація та структура системи, пропонується розглядати створення ПС на основі LST як підхід на основі **структурних інваріантів архітектури** програмної системи. Забезпечення життєздатності складних ієрархічних систем на основі моделі LST відбувається за рахунок використання структурних інваріантів системи із застосуванням мови зразка для формування каркасу життєздатної багаторівневої складної системи. Такий об’єктно-орієнтований каркас може бути використано в рамках інженерії ПрО та реалізовано, наприклад на основі моделі Gen-Voca [5, 6].

Проблема застосування моделі VSM для створення життєздатного програмного забезпечення розглянута Ч. Херінгом [7]. Застосування VSM для створення життєздатних ПС можна розглядати як підхід на основі **організаційних інваріантів архітектури ПС**. Такий висновок випливає із аналізу матеріалів роботи Ч. Херінга

“Життєздатне програмне забезпечення. Інтелектуальна парадигма керування для адаптуємих та адаптивних архітектур” (Viable software. The intelligent control paradigm for adaptable and adaptive architecture), яка ще не стала предметом широкого обговорення, однак становить особливий інтерес для фахівців в області архітектури та інженерії життєздатних ПС. У цій роботі сформульовані принципи, за якими мають будуватися ПС для того, щоб розвиватися в умовах, коли постійно змінюються вимоги до них. Узагальненням цих принципів стала **Інтелектуальна Парадигма Керування** (Intellectual Control Paradigm) для архітектури ПС. Актуальність роботи полягає в дослідженні ролі архітектурних зразків при розробці життєздатних ПС. У роботі запропонований принципово новий підхід до архітектури ПС на основі розширення архітектурного стилю парадигма керування (Control Paradigm style) в контексті моделі життєздатних систем (Viable System Model). Такий підхід може розглядатися як новий напрямок в області розробки життєздатних ПС. Характерною та принциповою особливістю запропонованих у роботі дев'яти зразків, представлених у вигляді мови зразка, є їхня постійна присутність в архітектурі системи для досягнення її життєздатності. Наявність такого інваріанта визначає можливість міграції навиків (skills) керування в програмну систему доти, поки система не досягне необхідного і достатнього рівня автономності. На наш погляд, запропонований у роботі підхід формує новий напрямок у розробці ПС на основі архітектурних інваріантів, що визначають життєздатність ПС. Оскільки в основі підходу лежить архітектурний стиль, що визначає організацію ПС в цілому й підтримує її еволюцію, такий підхід може розглядатися як підхід до розробки життєздатних ПС на основі організаційних інваріантів архітектури ПС.

Реалізація запропонованого в роботі Ч. Херінга архітектурного стилю приводить до створення рекурсивного компонентного каркасу ПС, що може бути адаптований для використання в конкретній предметній області та підтримувати еволюцію ПС за рахунок інтелектуальних механізмів підтримки адаптуємості та адаптивності ПС. Підхід до розробки життєздатних ПС може бути реалізовано на основі лінійки програмних продуктів (Software Product Line, SPL) в рамках предметно-орієнтованої програмної архітектури (Domain-Specific Software Architecture, DSSA). При цьому еталонна модель для сімейства програмних систем відповідає моделі життєздатних систем С. Біра.

Зазначимо, що запропонований Ч. Херінгом підхід розвиває поняття життєздатності як самоадаптивності ПС без врахування її життєвого циклу, людино-машинної сутності створення та функціонування ПС, а також економічних аспектів його забезпечення, що є важливими аспектами програмної інженерії [4]. Модель життєздатності ПС на основі LST та, що суттєво, можливість її надання у вигляді системи зразків на різних структурних рівнях організації системи є, на наш погляд, більш придатною, ніж VSM, для застосування у підході до створення життєздатних ПС, який розроблюється авторським колективом. Цей підхід ми розглядаємо як модельно-орієнтований та такий, що оснований на архітектурі (architecture-centric), базується на шаблонах проектування (pattern-based) відповідно до всіх рівнів структурної організації ПС (архітектура, проектування, реалізація), інтеграції систем та процесу розробки, розміщення та супроводження ПС. Найбільш придатним для розробки життєздатних ПС та підтримки їх еволюції ми вважаємо процес розробки, що керується моделями (Model-Driven Development, MDD), суттєвою характеристикою якого є застосування шаблонів при перетворенні моделей на різних рівнях абстракції. Застосування цього підходу для сімейства систем полягає у розвитку моделі життєздатності ПС та, зокрема, її складової – моделі простежування змін [8–10], у той її частині, яка стосується моделювання спільності (commonality), мінливості (variability) та залежності (dependency) для сімейства програмних продуктів та розробки механізмів, що підтримують еволюцію сімейства ПС у контексті інженерії предметної області та концепції забезпечення життєздатності ПС. Моделі, методи та технології породжуючого програмування ми відносимо до засобів забезпечення життєздатності сімейства програмних систем.

Описані підходи підвищують життєздатність складних організаційних систем шляхом підвищення їх самоадаптивності і напряму не застосовні до ПС, які мають свої суттєві особливості.

Виходячи із сутності програмної інженерії, нами було запропоновано модельно-орієнтований підхід до забезпечення життєздатності ПС. Модельно-орієнтований підхід розвинуто в роботах [8–14]. В контексті сучасних напрямків програмної інженерії такий підхід до розробки ПС може розглядатись як підхід на **основі архітектури, що керується моделями** (Model-Driven Architecture, MDA) [15]. При цьому необхідними складовими підходу є середовище розробки, модель ПС на основі мови UML, модель і механізми внесення змін для досягнення необхідних характеристик якості та функціональності ПС відповідно до моделі якості та застосування шаблонів проектування, засоби трансформації та інтеграції моделей.

У рамках запропонованого підходу властивість життєздатності (viability) ПС розглядається як комплексна характеристика ПС, яка включає певні характеристики та атрибути якості, що відповідають стандарту ISO 9126. ПС є життєздатною, якщо вона в процесі життєвого циклу може бути модифікована розроблювачем системи для досягнення необхідних функціональних та нефункціональних вимог. Така можливість забезпечується наявністю моделі ПС і засобів її підтримки, які імплементовані у середовище розробки й функціонування ПС. Необхідною складовою частиною моделі життєздатності ПС є модель простежування змін, побудована на основі сценарного підходу і моделі оцінки ПС на основі метрик внутрішньої й зовнішньої якості ПС. Залучення зразків проектування до процесу проектування та побудови ПС із заданими атрибутами якості може бути забезпе-

чено за допомогою каркасу, який підтримує модель оцінки якості на основі шаблонів. Як приклад, можна навести каркас на основі шаблонів GRASP. Передумовами до застосування та розвитку такого підходу є наявність певного зв'язку між зразками та атрибутами якості ПС, відповідно до якого можуть бути побудовані певні оптимізаційні моделі. Отже, в рамках модельно-орієнтованого підходу за допомогою методів та засобів програмної інженерії, а також відповідних спеціалістів еволюційно забезпечується адаптивність ПС до змін вимог користувача та навколишнього середовища.

У зв'язку з подальшим розвитком підходу компонентного програмування [6] (в рамках якого і було запропоновано наш підхід до забезпечення життєздатності прикладних ПС) та формування на його основі підходу генеруючого програмування, що розробляється в рамках відомчої теми Інституту "Розробка теоретичних основ генеруючого програмування, прикладних та інструментальних засобів його підтримки" (науковий керівник теми д-р. фіз.-мат. наук, професор. К.М. Лавріщева) актуальними є задачі визначення особливостей модельно-орієнтованого підходу стосовно забезпечення життєздатності прикладних ПС в генеруючому програмуванні та його подальшого розвитку.

1. Особливості модельно-орієнтованого підходу до забезпечення життєздатності ПС в середовищі генеруючого програмування

Генеруюче (породжуюче) програмування (generative programming) [5] – це методи і засоби генерації сімейств систем і додатків з окремих компонентів, повторно використовуваних компонентів (ПВК), каркасів тощо. Базисом цього програмування є об'єктно-орієнтований підхід, доповнений механізмами генерації багаторазових елементів і ПВК, а також властивостями їх мінливості, взаємодії тощо. У ньому використовуються різні методи програмування для підтримки інженерії предметної області, як дисципліни інженерного проектування сімейства прикладних ПС із різних раніше виготовлених продуктів. Головний продукт інженерії предметної області – це сімейство ПС, що генерується на основі породжуючої доменної моделі (Generative Domain Model, GDM), що включає в себе засоби визначення членів (представників) сімейства, а також методи генерації, зборки членів сімейства і базу конфігурації для розгортання сімейства в операційному середовищі.

Із існуючих підходів до створення прикладних ПС найбільш адекватним для реалізації модельно-орієнтованого підходу до забезпечення їх життєздатності є підхід генеруючого програмування. Важливою для нас особливістю підходу генеруючого програмування є співпадіння загальної схеми одержання кінцевого програмного продукту через моделювання предметної області, моделювання та генерацію конкретної ПС зі схемою розробки життєздатних прикладних ПС.

Згідно підходу до вирішення проблеми створення життєздатних автоматизованих систем організаційного керування [13] ПС є *життєздатними*, *якщо* вони забезпечені *засобами пристосування* до змін на стадії їх функціонування (супроводження). Такі засоби можна розглядати як засоби підтримки еволюції ПС в процесі їх життєвого циклу.

Так визначене поняття життєздатності ПС не вимагає від них мати властивості самоадаптивності при змінах в предметній області чи вимогах користувача, а лише наявності такої технології їх розробки та супроводження, яка налаштована на врахування змін, а отже на постійне здійснення зворотного зв'язку між навколишнім середовищем, ПС та генеруючого програмування (ГП) при їх супроводженні.

Схема моделювання та одержання варіанта життєздатної ПС включає:

- проведення моделювання ПС в основних (необхідних) аспектах та вибір кращого варіанта моделі ПС для подальшої реалізації на основі вхідних даних від предметної області, створеної UML -моделі, вимог користувача;
- одержання оцінок характеристик життєздатності ПС та їх аналіз в аспекті виконання нефункціональних вимог користувача, а за їх прийнятності – перехід до одержання оцінок економічних характеристик;
- одержання оцінок економічних характеристик ПС та їх аналіз в аспекті виконання вимог користувача, а за їх прийнятності – вироблення за моделлю ПС проекту ПС;
- реалізація проекту прикладної ПС;
- передача прикладної ПС в експлуатацію ((а) передача лише ПС, б) передача ПС разом із деякими засобами її моделювання).

Модель життєздатності ПС, схема моделювання та одержання в рамках цієї моделі варіанта проекту ПС, який має оцінки, що задовольняють вимоги користувача, описана в [12].

У рамках використання та розвитку модельно-орієнтованого підходу в ГП життєздатність ПС підтримується наявністю наступних засобів:

- модель ПрО;
- модель життєздатної ПС на основі інтеграції системи зразків проектування, специфікованих з використанням мови UML;
- засоби одержання оцінок характеристик життєздатності та економічних характеристик ПС та їх аналізу в аспекті виконання вимог користувача.

Модель ПС. Пропонований нами підхід передбачає класифікацію предметних областей, виділення для них

типів ПС та створення для цих типів ПС їх UML-моделей. UML-модель ПС підтримує процес керування життєздатністю та процес її забезпечення. Вона визначає атрибути життєздатності ПС, відповідно до нефункціональних вимог до системи, внутрішні та зовнішні атрибути життєздатності ПС, а також зв'язок між ними.

Наприклад, модель UML класу програмних систем CRM (Customer Relationship Management – керування взаємовідносинами з клієнтами [14]) включає ***описи типових класів*** та ***описи типових діаграм взаємодій*** для *підсистеми продажів*; *підсистеми контролю роботи персоналу*; *підсистеми "Органайзер"*. Названі підсистеми складають типову CRM-систему "початкового рівня", яка тим не менш є цілісною системою, аналоги якої є на ринку України. Всі програмні системи для класу CRM-систем, що будуть створюватися в ближчий час в Україні, в основному, описуються цією UML–моделлю за винятком деяких особливостей, які виявляються при обстеженні конкретного об'єкта автоматизації.

Метрики та методи оцінки показників життєздатності ПС. В модельно-орієнтованому підході метрики використовуються для попередження невірних проектних рішень на ранній стадії життєвого циклу ПС. Знайдені невідповідності або дефекти можуть бути модифіковані та попереджені з меншими затратами коштів та зусиль ніж на пізніших етапах проектування або на стадії супроводу.

Метрики забезпечують розробнику швидкий зворотній зв'язок – шляхом аналізу зібраних даних можна прогнозувати життєздатність ПС. При відповідному використанні метрик можливе значне зменшення вартості всієї розробки та покращення якості кінцевого програмного продукту, що в свою чергу веде до зменшення витрат на його підтримку. Однак інформація, доступна на ранній стадії проектування часто є неточною та неповною. Через це інформація про життєздатність ПС, отримана у середовищі ГП вимагає уточнення та доповнення на пізніших етапах життєвого циклу ПС. В модельно-орієнтованому підході пропонується також застосовувати метрики в контексті сценаріїв, які реалізуються за допомогою представлення процесу (діаграм взаємодії в середовищі Rational Rose). Це дозволить оцінювати параметри системи не загалом, а в рамках деяких задач або процесів, що цікавлять нас особливо. Таким чином метрики будуть застосовуватися не до всіх класів та їх методів а лише до тих, що пов'язані спільним сценарієм.

Розвиток модельно-орієнтованого підходу полягає у розробці на основі MDA засобів підтримки формування моделі ПС із моделі ПрО, засобів підтримки одержання необхідних оцінок для прийняття рішень, прийняття рішень щодо їх відповідності вимогам користувача та засобів підтримки формування нової моделі ПС з оцінками що задовольняють вимоги користувача.

Процес інженерії ПрО в рамках парадигми генеруючого програмування та концепції забезпечення життєздатності ПС. Такий процес ми розглядаємо як інженерію ПрО сімейства життєздатних програмних систем. Цей процес, який представлено на рисунку включає:

- прикладну інженерію (Application Engineering) життєздатних ПС або інакше – процес розробки (виробництва) окремих життєздатних систем на основі результатів інженерії ПрО життєздатних ПС. Цей процес характеризується наявністю зворотного зв'язку із процесом інженерії ПрО. Життєздатні ПС будуються за допомогою засобів заказу додатків на тому рівні автоматизації, який відповідає автоматичній зборці кінцевого продукту та засобів забезпечення життєздатності ПС відповідно до певних критеріїв життєздатності;
- інженерію ПрО (Domain Engineering) життєздатних ПС або інакше – процес, проектування та реалізації породжуючої доменної моделі сімейства життєздатних систем на основі інженерії ПрО в рамках парадигми генеруючого програмування та концепції забезпечення життєздатності ПС. Породжуюча доменна модель забезпечує можливість автоматичної генерації членів сімейства на основі абстрактних специфікацій та вміщує засоби специфікації членів сімейства, компоненти реалізації, знання про конфігурації та критерії життєздатності ПС. Критерії життєздатності відповідають моделям життєздатності, які застосовуються при створенні архітектури сімейства життєздатних систем та компонентів реалізації для повторного використання. Конфігурування, оптимізація та забезпечення життєздатності кінцевих продуктів здійснюється на основі моделі прийняття рішень (decision model);
- інженерія засобів забезпечення життєздатності сімейства систем або інакше – процес створення моделей та засобів пристосування (адаптації) інженерії ПрО до цілей та вимог породжуючого програмування, моделей та засобів забезпечення життєздатності програмних систем відповідно до модельно-орієнтованого підходу до створення ПС, а також технологій реалізації породжуючого програмування для сімейства життєздатних ПС.

Курсивом на рисунку виділені ті складові процесу, які стосуються безпосередньо забезпечення життєздатності сімейства ПС.

Відповідно до застосування парадигми породжуючого програмування та концепції забезпечення життєздатності програмних систем для інженерії ПрО життєздатна програмна система, яка належить сімейству систем, може розглядатись як така, що **по-перше**, пристосована до внесення змін (здатна змінюватись) за участю розробника системи відповідно до зміни зовнішнього середовища (*предметної області* та вимог користувача) у *визначених* межах, **по-друге**, володіє засобами пристосування до змін на основі певної *інфраструктури повторного використання*, *засобів генеруючого програмування та засобів підтримки життєздатності ПС* в рамках *процесу еволюційної розробки*, **по-третє**, розроблюється та супроводжується за наявністю обмеженості ресурсів (зокрема, часових, фінансових, робочих) відповідно до певних економічних моделей.

Предметна область в інженерії ПрО [5,6] – це не тільки набір понять та термінологія, які застосовуються спеціалістами у даній області, а ще знання у області розробки програмних систем або їх складових. Відповідно до концепції забезпечення життєздатності ПС, це також знання щодо принципів та підходів до створення життєздатних систем. *Процес еволюційної розробки* охоплює усі стадії та процеси життєвого циклу системи, на яких запроваджуються моделі та механізми, що відповідають засобам пристосування до змін, процес керування проектом, а також процеси формування та узгодження ІТ інфраструктури системи та інфраструктури проблемної області. Моделі та механізми, що відповідають засобам пристосування до змін визначаються моделлю життєздатності, складовою частиною якою є модель спільності (commonality), мінливості (variability) та залежності (dependency).

Критичним фактором щодо забезпечення життєздатності ПС в рамках модельно-орієнтованого підходу є встановлення зв'язку між артефактами програмної системи та характеристиками предметної області. Такий зв'язок досягається на основі застосування шаблонів проектування. в рамках Domain-driven approach. Зокрема, це Product-line Architeture Design Approach та Pattern-driven Architecture Design Approach, які видобувають абстракції архітектурного проектування із моделі предметної області (domain model). Найбільш придатним для підтримки PLA з точки зору концепції забезпечення життєздатності є Domain-Specific Software Architecture (DSSA) approach. Відповідність системи зразків проектування (design patterns) та проблемно-залежних зразків проектування (domain-specific design patterns) зразкам предметної області (domain patterns) має відображатись у предметно-орієнтованій мові конфігурування та мові конфігурування компонентів реалізації відповідно до простору задач та простору рішень. За допомогою генератора може бути здійснено переклад з однієї мови на іншу [5].

Відповідно до застосування парадигми породжуючого програмування та концепції забезпечення життєздатності програмних систем для інженерії ПрО життєздатна програмна система може розглядатись як така, що **по-перше**, пристосована до внесення змін (здатна змінюватись) за участю розробника системи відповідно до зміни зовнішнього середовища (предметної області та вимог користувача) у *визначених* межах, **по-друге**, володіє засобами пристосування до змін на основі певної *інфраструктури повторного використання*, *засобів генеруючого програмування та засобів підтримки життєздатності ПС* в рамках *процесу еволюційної розробки*, **по-третє**, розроблюється та супроводжується за наявністю обмеженості ресурсів (зокрема, часових, фінансових, робочих) відповідно до певних економічних моделей.

2. Особливості модельно-орієнтованого підходу до забезпечення життєздатності в середовищі супроводження прикладних ПС

Розроблена в середовищі ГП і передана користувачу ПС на протязі свого життєвого циклу постійно еволюціонує.

Згідно пропонованого підходу для забезпечення життєздатності прикладних ПС користувачу має бути передана не лише сгенерована ПС, але і також деякі засоби підтримки її життєздатності.

Звичайно, при змінах в "ядрі" системи необхідна перегенерація ПС з допомогою середовища ГП. Але такі зміни відбуваються рідко, або взагалі не відбуваються протягом життєвого циклу ПС. Проводити ж перегенерацію прикладної ПС при всіх її змінах недоцільно. Зміни функціональності ПС відбуваються часто і потребують, як правило, незначних змін спеціального програмного забезпечення. Ці зміни можуть бути розроблені безпосередньо колективом супроводження ПС, або за технічними вимогами цього колективу в середовищі ГП. Після цього компоненти в ПС, що змінюються, заміняються на нові.

Тобто, в середовище супроводження прикладних ПС має передаватися:

- модель конкретної прикладної ПС;
- засоби визначення компонентів ПС, що змінюються при виникаючих змінах (моделі простежування змін;
- засоби заміни версій компонентів в прикладній ПС.

При цьому інструментальні засоби модифікації компонентів ПС, що зазнають змін, можуть бути передані в середовище супроводження ПС.

Механізми простежування змін в ПС. Відповідно до архітектури процесів життєвого циклу ПС *модель простежування змін* підтримує процес керування конфігурацією та, зокрема, процес внесення змін. Вона визначає зв'язок між артефактами прикладної ПС відповідно до різних етапів життєвого циклу та процесу розробки, а також механізми, що зумовлюють процес перетворень архітектури ПС. У відповідності з цим, для класу ПС створюються *моделі простежування змін* та основні *сценарії* їх функціонування.

Концепція сценарного підходу розглядає *сценарій* як базовий артефакт, що підтримує варіантність ППС та визначає механізми, що забезпечують варіантність проектних рішень, артефакти, що підлягають оцінюванню, конкретизує модель життєздатності на рівні мікропроцесу та макропроцесу об'єктно-орієнтованої розробки в рамках ітеративного життєвого циклу.

Для оцінки сценарію на відповідність, наприклад, шаблонам GRASP кожному шаблону необхідно співставити певні параметри оцінки сценарію та спеціальні ситуації, створення яких супроводжується обробкою цих параметрів на етапі функціонування ПС.

Аналіз та оцінка базових артефактів ПС, зокрема варіантів сценаріїв, яким відповідають певні діаграми взаємодій, має завершуватись встановленням зв'язку між їхніми оцінками (на основі внутрішніх метрик ПС) та зовнішніми атри-

бутами ПС (функціональними та нефункціональними). Це приводить до створення інтегральних оцінок сценарію та ПС у цілому.

Актуальними задачами вирішення цієї проблеми для об'єктно-орієнтованих ПС є створення засобів одержання таких оцінок на основі аналізу відповідності ПС шаблонам проектування із використанням діаграм UML-моделей та відповідних інструментальних засобів, таких, як Rational Rose [16].

У загальному випадку інструментальні засоби модифікації компонентів ПС, що зазнають змін знаходяться в середовищі ГП, де генеруються нові компоненти за замовленням групи супроводження ПС. Але при супроводженні ПС можуть бути також застосовані прості засоби модифікації компонент без звернення до середовища ГП.

3. Забезпечення варіантності рішень щодо життєздатності ПС

Проблема забезпечення варіантності рішень щодо ПС в модельно-орієнтованому підході базується на концепції сценарного підходу. Відповідно до цієї концепції поняття сценарію охоплює як функціональний аспект розробки ПС, так і аспект розробки, що 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 дихотомії "клас/об'єкт". 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 варіанта 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" та розглядається у контексті сценарію та певних механізмів проектування. При цьому сценарії розкривають певні функціональні точки системи та визначають інтерфейси системи, а поведінка абстракцій, що належать сценарію, реалізується та ілюструються діаграмами об'єктів та діаграмами взаємодій. Утворення варіантів сценаріїв здійснюється шляхом перерозподілу обов'язків між об'єктами та визначенням 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 підтримуються 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 та утворюється архітектура системи. Функціональні точки розкриваються у вигляді сценаріїв відповідно функціональним вимогам 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 проектування GRASP. Архітектура системи визначається шляхом групування та розподілення функціональних точок за шарами та розділами архітектури із урахуванням їх варіантності та визначенням відповідних інтерфейсів за допомогою інших шаблонів проектування, зокрема, шаблонів GoF [4]. 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 релізи та еволюційні релізи ПС відповідно до варіантності, яку закладено до систем. Внесення змін до ПС та підтримка варіантності здійснюється в рамках певних моделей простежування та процесу керування змінами. Рішення щодо реінжинірингу ПС приймається за умови, коли вимоги щодо 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, одержуємо модель 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.

4. Критерії прийняття рішень щодо забезпечення життєздатності ПС

Основним критерієм забезпечення життєздатності ПС є максимізація функції корисності, яка будується для класу ПС і яка може бути дуже складною і включати багато факторів.

Для більшості ПС, що автоматизують системи організаційного управління в бізнес-сфері, основними критеріями прийняття рішень при забезпеченні життєздатності є критерії економічної ефективності їх автоматизації.

Безумовно, при застосуванні пропонованого підходу до створення та супроводження ПС повинно бути забезпечено **вимоги користувача** щодо якості ПС. До якісних показників ПС за класифікацією характеристик якості ПС, наведеної в ISO/IEC 9126-1, як відомо, належать [12]: функціональність, надійність, зручність використання, ефективність, супроводжуваність, переносимість.

Останні дві характеристики якості характеризують життєздатність ПС. При цьому показники супроводжуваності обумовлюють можливості ефективної модифікації ПС, включаючи корегування, удосконалення, або адаптацію до зміни середовища, вимог та функціональних специфікацій, а показники *переносимості о*бумовлюють здатність ПС бути перенесеними з одного середовища до іншого.

Подальша конкретизація показників якості ПС, що забезпечують необхідну життєздатність при змінах їх функціональних та нефункціональних вимог, зокрема, включає [12]: аналізованість, адаптованість (модернізованість), стабільність, тестованість.

Для забезпечення процесу керування якісними показниками ПС модельно-орієнтований підхід має побудовані відповідні метрики та алгоритми одержання оцінки цих показників.

У програмній інженерії метрики та алгоритми одержання оцінки традиційно використовуються для попередження невірного проектування на ранній стадії життєвого циклу ПС. Знайдені невідповідності та дефекти можуть бути модифіковані та попереджені з меншими затратами коштів та зусиль, ніж на пізніших етапах проектування або на стадії супроводу.

Це забезпечує розробнику швидкий зворотній зв'язок - шляхом аналізу зібраних даних можна прогнозувати якісні показники ПС. При відповідному використанні цих засобів можливе значне зменшення вартості всієї розробки та покращення якості ПС, що в свою чергу, веде до зменшення витрат на їх підтримку. Однак інформація, доступна на ранній стадії проектування часто є неточною та неповною. Через це інформація щодо якісних показників ПС, отримана на стадії проектування вимагає уточнення та доповнення на пізніших етапах їх життєвого циклу.

Пропонується також застосовувати метрики в контексті сценаріїв, які реалізуються за допомогою представлення процесу (діаграм взаємодії у середовищі Rational Rose). Це дозволяє оцінювати параметри системи не загалом, а в рамках деяких задач або процесів, що цікавлять нас особливо. Таким чином метрики можуть застосовуватися не до всіх класів та їх методів а лише до тих, що пов'язані спільним сценарієм.

Для **розробника** ПС найбільш важливими економічними характеристиками проекту *є розмір проекту, трудомісткість, час, необхідний для виконання та вартість.*Серед цих характеристик (для традиційних підходів до розробки) три останні у великій мірі залежать від першої характеристики -*розміру* ПС.

В області управління проектами для традиційних підходів до розробки ПС гостро стоїть проблема оцінювання проектів з розробки, модифікації чи реінжинірингу ПС в аспекті необхідних для їх виконання ресурсів, бюджету, та строків їх виконання. Оскільки з чотирьох основних економічних характеристик проекту ПС *(розмір проекту, трудомісткість, час, необхідний для виконання та вартість)* три останні у великій мірі залежать від першої характеристики – *розміру* ПС, оцінювання цих характеристик можна розбити на два етапи: оцінювання розміру;оцінювання трудомісткості, вартості та тривалості з використанням вже отриманого значення розміру.

Для розрахунку оцінок трьох останніх характеристик добре підходить модель COCOMO [17], оскільки вона розроблялася саме з метою оцінки бюджету, вартості та тривалості проектів та є нині найбільш широко застосовуваною моделлю. Існуючі методи експертних оцінок також можуть бути включені в схему, оскільки застосування більш формалізованих методів у певних випадках є неможливим або невигідним.

Для організації-замовника ці показники також є важливими, але їх недостатньо для оцінки економічного ефекту від впровадження ПС та визначення доцільності проекту.

Отже, мінімізація цих показників не може бути критеріями вирішення задач із забезпечення життєздатності ПС, оскільки вони не враховують ефективність ПС для користувача.

Впровадження прикладної ПС у організації можна розглядати як інвестиційний проект [18], в ході якого замовник витрачає певні кошти на придбання активу (ПС), який дозволяє зменшити видатки через оптимізацію бізнес-процесів або збільшити надходження, запропонувавши ринку нові продукти.

Прикладна ПС є ефективною, якщо грошові потоки - **Cash Flow** (CF), що генеруються її проектом, забезпечують дві умови:

- покриття початкової інвестиції;
- віддачу на вкладені гроші (дохід інвестора).

Найбільш вживаними (основними) показниками ефективності інвестиційних проектів є такі:

- ***чиста приведена вартість –*** відображає різницю між видатками та прибутками, коли всі кошти приведені у часі до теперішнього моменту часу;
- ***внутрішня норма прибутку*** – так називають ставку дисконтування, за якої величина чиста приведена вартість стає нульовою;
- ***період окупності*** – час, необхідний для того, щоб сума, інвестована в той чи інший проект, повністю повернулася за рахунок коштів, одержаних в результаті основної діяльності за даним проектом;
- ***приведений період окупності*** враховує „часову вартість" грошей. Через такі фактори, як інфляція, банківський процент або інші привабливі проекти, одна гривня сьогодні коштує більше, ніж одна гривня завтра;
- ***коефіцієнт прибутковості інвестицій*** прибуток до виплати процентів і податків, поділений на суму довготермінових зобов'язань плюс акції.

Безумовно, на економічні показники (а отже життєздатність) прикладних ПС впливають показники ефективності генеруючого середовища, з допомогою якого створювалася прикладна ПС.

Вартість створення прикладної ПС залежить від вартості створення генеруючого середовища, вартості генерації прикладної ПС та кількості продаж прикладних ПС. Тобто можна говорити про розробку та створення генеруючого середовища як про інвестиційний проект, який має бути успішним. Для його успішності необхідно, щоб грошові потоки – **Cash Flow** (CF) цього проекту також забезпечували покриття початкової інвестиції та віддачу на вкладені гроші (дохід інвестора).

Це можливо у випадку, якщо створене генеруюче середовище генерує життєздатні прикладні ПС і, як наслідок, кількість продаж прикладних ПС дозволяє покрити початкову інвестицію та одержати дохід інвестора.

Висновки

Парадигма породжуючого (генеруючого) програмування є найбільш адекватною для розвитку модельно-орієнтованого підходу до забезпечення життєздатності ПС.

Методи та засоби започаткованого нами підходу до забезпечення життєздатності ПС потребують подальшого розвитку для їх імплементації в середовище породжуючого (генеруючого) програмування.

1. *Бир С.* Мозг Фирмы. – М.: Радио и связь, 1993. – 218 с.
2. *Эшби У.Р.* Введение в кибернетику. – М.: Наука, 1975. – 427 с.
3. *Miller J.G.* Living Systems. – Niwot, Colorado: University Press of Colorado, 1995. – 1102 p.
4. *Ларман К.* Применение UML и шаблонов проектирования. – М.: Изд. дом "Вильямс", 2001. – 496 с.
5. *Чернецки К., Айзенекер У.* Порождающее программирование. Методы, инструменты, применение. – Изд. дом «Питер».– С-Пет.– 2005. – 730 с.
6. *Лаврищева Е.М.* Методи програмування. Теорія, інженерія, практика. – К.: Наук. думка, 2006. – 452 с.
7. *Herring C.* Viable software. The intelligent control paradigm for adaptable and adaptive architecture. – 2002. – 343 p. http://charles-herring.com/public/ViableSoftware.pdf
8. *Ігнатенко П.П., Неумоїн В.М., Бистров В.М.* Про забезпечення ефективного реінжинірингу прикладних програмних систем // Проблемы программирования. – 2001. – № 1-2. – С. 42–52.
9. *Ігнатенко П.П., Бистров В.М., Заєць І.О., Ігнатенко О.П.* Підхід до забезпечення реінжинірингу об'єктно-орієнтованих програмних систем // Проблемы программирования. – 2002. – № 1-2. – С. 98–108.
10. *Ігнатенко П.П.* Проблеми забезпечення життєздатності програмних систем та підходи до їх вирішення // Проблемы Программирования. – 2002. – № 3-4. – С. 58–73.
11. *Ігнатенко П.П., Бистров В.М., Ігнатенко О.П., Ткаченко В.М.* Задачі та засоби моделювання й оцінювання життєздатних програмних систем // Проблемы программирования. – 2003. – № 3. – С. 59–70.
12. *Ігнатенко П.П., Стрєлов І.А, Ткаченко В.М., Дуднік Р.О.* Концепція створення моделі прикладної програмної системи з розвинутою функцією життєздатності // Проблемы программирования. – 2004. – № 2–3. – С. 163–172.
13. *Ігнатенко П.П.* Життєздатні програмні системи. Концептуалізація підходу до автоматизації систем організаційного керування // Проблемы программирования. – 2006. – № 3. – С. 33–44.
14. *Игнатенко П.П., Ткаченко В.Н., Стрелов И.А., Дудник Р.А.* Подход к моделированию и проектированию CRM-систем // УСИМ – 2005. № 2. – С. 57 – 65.
15. *Кузнецов М.* MDA – новая концепция интеграции приложений // Открытые системы. – 2003. – № 9. http://www.osp.ru/os/2003/09/048.htm
16. *Боггс У., Боггс М.* UML и Rational Rose. – М.: Лори. – 2000. – 582 с.
17. *Стрєлов І.А., Ігнатенко П.П.* Підхід до оцінювання економічних характеристик проектних рішень при розробці, модифікації та реінжинірингу програмних систем // Проблемы программирования. – 2004. – № 1. – С. 38–51.
18. *Стрєлов І.А., Ігнатенко П.П.* Підхід до прогнозування основних характеристик економічної ефективності ППС // Проблемы программирования. – 2007. – № 4. – С. 13–20.