УДК 681.3

ПІДХОДИ ІНЖЕНЕРІЇ ЯКОСТІ СІМЕЙСТВ ПРОГРАМНИХ СИСТЕМ

К.М. Лавріщева, Г.І. Коваль, Т.М. Коротун

Інститут програмних систем НАН України
03187, Київ, проспект Академіка Глушкова, 40.
Тел.: (044) 526 3470, 526 4579.

Визначено напрямки розвитку інженерії якості програмних систем (ПС), які розробляються у новій парадигмі програмування – генеруюче програмування. Охарактеризовано актуальні проблеми забезпечення якості сімейств систем, що складають базис для побудови окремих ПС - членів сімейств – з повторно використовуваних компонентів. Сформульовано задачі інженерії якості сімейств та підходи до їх вирішення, зокрема задач верифікації, тестування, моделювання та оцінювання якості ПС у новій парадигмі.

Directions to improvement of software system quality engineering are considered in the report. This software systems (SS) are creating in a new programming paradigm – generating programming. The actual issue of providing quality of a system families are described, because system families make a basis of constructing a set of systems (members of family) from the reused components. A great number of tasks for the quality engineering of system family, as well as approaches to their decision, are defined, in particular the tasks of verification, testing, modeling and evaluation of SS quality in the new paradigm.

Вступ

За сучасних темпів розвитку індустрії програмного забезпечення квінтесенцією здешевлення та підвищення якості програмних систем (ПС) у різних проблемних областях (ПрО) є їх побудова у новій парадигмі *генеруючого програмування* (ГП) з множини різнотипних повторно використовуваних програмних компонентів, а також постійне накопичення, узагальнення та використання знань стосовно ПрО та перспектив їх еволюції.

Інженерію ПрО ототожнюють з інженерією *сімейств ПС (СПС)* або *лінійок продуктів* у ПрО, які будуються в *інтегрованому середовищі розроблення* (IDE, Integrated development environment) на загальній архітектурній платформі. Ця платформа окрім постійної складової, що відповідає функційним потребам вирішення задач у ПрО, повинна мати *динамічну складову*, яка забезпечуватиме *варіабельність архітектури* з урахуванням ресурсних (технічних, фінансових та ін.) обмежень середовищ застосування конкретних згенерованих ПС (членів сімейства), а також моделей якості, визначених для цих ПС.

Існує чимало підходів інженерії якості прикладних ПС (переважно вбудованих або мережних нерозподілених), побудованих у парадигмах структурного та об'єктно-орієнтованого програмування [1]. Для ПС нової генерації, що будуються у сучасних нових парадигмах (зокрема, генеруючого програмування), такі підходи до тепер не сформовані.

Розроблення теоретичного та методологічного підґрунтя забезпечення якості СПС, а також підтримки прийняття й обґрунтування рішень стосовно розроблення СПС і генерації ПС з необхідними показниками якості та врахуванням вимог до їх складності, вартості та інших нефункційних характеристик, є надзвичайно актуальною проблемою. Вирішення цієї проблеми розпочато в рамках досліджень за проектом ІПС НАН України ІІІ-1-07 «Розробка теоретичного фундаменту генеруючого програмування та інструментальних засобів його підтримки».

Створення підходів інженерії якості СПС ґрунтується на аналізі теоретичних і прикладних досягнень у об'єктному, компонентному, аспектному, сервісно-орієнтованому, агентному та іншому програмуванні з позицій верифікації, тестування, забезпечення та оцінювання якості розроблюваних та повторно використовуваних елементів сімейств/лінійок програмних систем.

У доповіді сформульовано задачі інженерії якості ПС, які будуються у парадигмі ГП, та загальні підходи до їх вирішення.

1. Процесний підхід до розроблення ПС у парадигмі генеруючого програмування

На сучасному етапі розвитку інформаційних технологій архітектурно-функціональний спектр типів програмних компонентів (від утиліт для мейнфреймів до вбудованих, автономних (клієнтських) та розподілених мережних ПС і Web-застосувань, сервісів, систем) є досить широким. Побудова з них програмних систем з необхідними властивостями потребує вирішення задач класифікації і типізації, ідентифікації архітектури, специфікації функцій і інтерфейсів, зберігання, пошуку, вибору, адаптації, інтеграції (конфігурування) компонентів і, нарешті, власне генерації ПС та накопичення отриманого досвіду. Існуючі методи розроблення таких ПС не передбачають автоматизації. Їх автоматизована побудова можлива в рамках нової парадигми в програмній інженерії – парадигми генеруючого (породжуючого) програмування .

В основі парадигми ГП лежить процес розроблення ПС, двома складовими якого є *інженерія ПрО* (domain engineering) і *інженерія програмних застосувань* (application engineering) (рис. 1). Ці складові взаємодіють через інформаційні структури IDE, а "ініціатором" взаємодії виступає підпроцес інженерії програмних застосувань, який розпочинається з аналізу вимог до розроблюваної ПС.

Головним цільовим об'єктом програмування в парадигмі ГП є не унікальний програмний продукт, а *сімейство* (лінійка) продуктів у певній ПрО, кожний елемент (член) якого не створюється "з нуля", а генерується на основі загальної генеруючої моделі ПрО - моделі сімейства – шляхом її конкретизації відповідно до специфікації програмної системи, вибору і композиції повторно використовуваних компонентів (ПВК), які містяться у репозиторії інтегрованого середовища розроблення СПС.

Рис. 1. Модель процесу розроблення ПС у парадигмі генеруючого програмування

На теперішній час існує декілька цілісних методологій розроблення, які могли б бути покладені в основу базового процесу побудови сімейств ПС та їх членів, зокрема [2]: DRACO, CFRP (Conceptual Framework for Reuse Process), ROSE Process Model, ODM (Organization Domain Modeling), RSEB (Reuse-Driven Software Engineering Business), FeatureRSEB (інтеграція у RSEB діаграм властивостей з FODAcom), FORM (Feature-oriented Reuse Method), PuLSE (Product Line Software Engineering), KobrA, CoPAM (Component-oriented Platform Architecture Method), PECOS (Pervasive Component Systems), RiZE (Reuse in Software Engineering), GenVoca та інші. Але загальною проблемою застосування цих методологій є відсутність у них засобів моделювання та оцінювання якості СПС та ПС під час розроблення і, отже, вони не дозволяють робити проекцію тих чи інших рішень, що приймаються, на якість майбутніх ПС.

Пропонується будувати базовий процес розроблення сімейств ПС з процесів життєвого циклу (ЖЦ), присутніх в еталонній моделі процесів в ISO/IEC 12207 [3]. Застосовно до потреб генеруючого програмування процеси ЖЦ, що спочатку призначалися для розроблення одиничних унікальних ПС, можуть бути «розтиражовані» та адаптовані для сімейства ПС (табл. 1). Виконувачами цих процесів є спеціалісти з розроблення *сімейств ПС* (а не окремої ПС).

Серед множини процесів, поданих у табл. 1, слід виділити процеси 3.5 – 3.7, поява яких у вихідній моделі процесів (за версією ISO/IEC 12207 2002 року) засвідчила важливість проблеми повторного використання напрацювань організацій-розробників програмного забезпечення у окремих ПрО. Ці організаційні процеси не потребують адаптації. Ефективне виконання інших процесів ЖЦ СПС має грунтуватися на методах та технологічних модулях, пристосованих для функціювання в інформаційно-програмному IDE генеруючого програмування.

Таблиця 1. Модель адаптованих процесів ЖЦ для побудови сімейств ПС[1]

№ п/п	Найменування процесу (підпроцесу)
	Процеси з категорії «Основні процеси»
1.3	*Розроблення*
1.3.1	Виявлення вимог до сімейства систем у ПрО
1.3.2	Аналіз вимог до сімейства систем
1.3.3	Проектування архітектури сімейства систем
1.3.4	Аналіз вимог до компонентної архітектури сімейства
1.3.5	Проектування сімейства систем (проблемно-орієнтованої мови ПрО - DSL-мови, інформаційних структур для сімейства)
1.3.6	Формування ПВК (розроблення, придбання готових ринкових COTS-компонентів, реінженіринг успадкованих систем)
1.3.7	Реалізація активів ПрО та їх зберігання у репозитарії IDE для ПрО
1.3.8	Тестування активів ПрО
1.3.9	Композиція активів ПрО (у репозитарії)
1.3.10	Тестування цілісності та простежуваності взаємопов’язаних активів в репозитарії
1.3.11	Фіксація сімейства в репозитарії
1.5	*Супровід* (еволюція) ПрО
	Процеси з категорії «Підтримувальні процеси»
2.1	Документування ПрО і активів
2.2	Керування конфігурацією активів у ПрО
2.3	Гарантування якості активів ПрО
2.4	Верифікація активів ПрО
2.10	Оцінювання сімейства ПС
	Процеси з категорії «Організаційні процеси»
3.1.3	Керування проектом розроблення сімейства ПС
3.1.4	Керування якістю сімейства ПС
3.1.5	Керування ризиком розроблення сімейства ПС
3.1.6	Вимірювання активів сімейства ПС
3.5	Керування активами (надбанням організації-розробника)
3.6	Керування програмою повторного використовування
3.7	Доменна інженерія

Відповідно до процесного підходу головним завданням формування засад інженерії якості сімейств систем є побудова *профілю* базового процесу генеруючого програмування, схематично поданого на рис. 2.

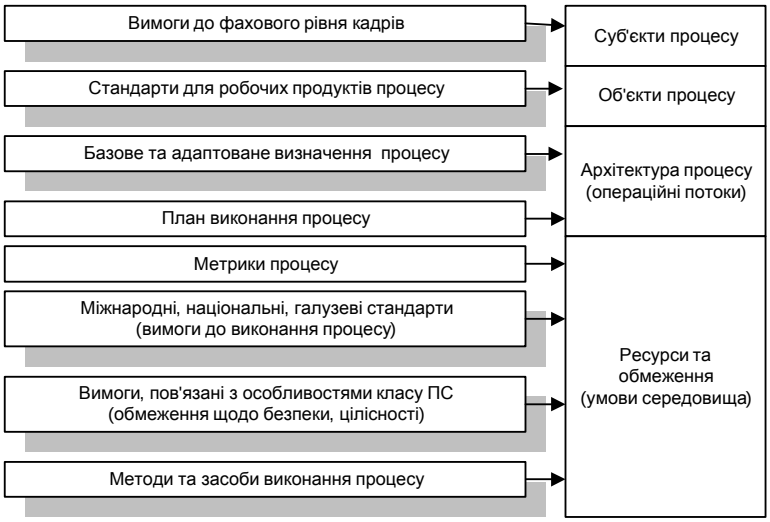

Рис. 2. Елементи профілю процесу

Профіль базового процесу складатиметься з множини профілів процесів ЖЦ, поданих у табл. 1.

[1] Нумерація процесів, включених у модель, відповідає ISO/IEC 12207:2002.

2. Актуальні проблеми інженерії якості сімейств ПС

Інженерія якості сімейств ПС перш за все потребує визначення елементів відповідної ПрО, а саме:

- множини вимірюваних атрибутів тих артефактів *сімейства* ПС, стосовно яких оцінюються показники якості сімейства;

- множини вимірюваних атрибутів тих *робочих продуктів* ПС, *розроблених компонентів*, *композицій компонентів*, стосовно яких прогнозується якість розроблюваної ПС (члена сімейства);

- множини *метрик*, *моделей*, *методів* і *методик*, використовуваних для підтримки ухвалення і обґрунтування рішень розробником сімейства ПС (з одного боку) і розробником системи (з другого боку) щодо створення ПС із заданими (або найкращими відомими для ПрО) показниками якості;

- множини *елементів інформаційно-програмної підтримки* застосування метрик, моделей і методів в інтегрованому середовищі розроблення сімейств ПС.

Крім проблеми "самовизначення" виникають й інші проблеми, пов'язані з розробленням ПС у парадигмі генеруючого програмування, зокрема такі:

1) *моделювання якості сімейства ПС* та оцінювання якості генерованих артефактів на кожній стадії інженерії ПрО та інженерії застосувань. Проблема полягає у відсутності методів оцінювання якості таких артефактів СПС, як, наприклад, модель ПрО, модель архітектури сімейства (при модельно-орієнтованому підході до побудови СПС), генераторів тощо;

2) *вибір компонентів, які щонайкраще відповідають вимогам до цільової ПС*. Проблема полягає у визначенні критеріїв оцінки *рівня довіри* до компонентів;

3) верифікація активів ПрО. Проблема полягає у відсутності методів перевірки відповідності вибраних (розроблених) активів потребам ПрО, зокрема доведення коректності моделей, компонентів, архітектурних композицій компонентів;

4) *тестування компонентів нової генерації*. Проблема полягає в необхідності пристосовування існуючих і розробленні нових методів он-лайнового (оперативного) тестування вибраних компонентів для генерації ПС, зокрема, Web-компонентів;

5) *розроблення інструментів тестування*. Видається, що інструменти тестування компонентів ПС та їх композицій, як специфічний вид ПС у ПрО тестування, самі могли б утворювати сімейства, що полегшило б їх пристосування до програмної архітектури середовища IDE, в якому розробляються цільові ПС;

6) *сертифікація компонентів – затвердження їх відповідності загальноприйнятим стандартам, а також адекватності заданій множині вимог*. Проблема полягає у тому, що початковий код вибираних компонентів (COTS-продуктів або продуктів з Інтернет) недоступний для аналізу і верифікації відомими методами. Необхідно комбінувати різні стратегії сертифікаційного тестування, включаючи тестування не тільки розробниками, але і постачальниками, і споживачами цих компонентів;

7) *визначення оперативної програмної архітектури, що не протиріччитиме вимогам до якості ПС*. Проблема полягає у тому, що програмна архітектура ПС, яка вибудовується з архітектури сімейства шляхом композиції ПВК, має формуватися з урахуванням не лише функційних вимог до ПС, але й вимог до якості. Ця проблема безпосередньо зумовлює наступну проблему, а саме

8) *керування ризиком в ході всього ЖЦ генерації системи*. Проблема полягає в необхідності аналізу компромісів при виборі тієї або іншої архітектурної композиції, стратегії генерації ПС, тестування та оцінювання якості. Її вирішення пов'язано з визначенням таксономії ризиків і регулярним оцінюванням ризиків на кожному кроці генерації ПС.

Пошук напрямків вирішення цих проблем також є завданням формування засад інженерії якості СПС.

3. Задачі інженерії якості СПС та підходи до їх вирішення

Задачі, які вирішуються у ПрО інженерії якості СПС, пропонується класифікувати за трьома категоріями:

- задачі, вирішувані *інженером з якості сімейства ПС* на стадії інженерії ПрО (*MDi*);

- задачі, вирішувані *інженером з якості прикладної ПС* у певній ПрО (*MPi*) та

- задачі стосовно підтримки прийняття рішень з керування варіантністю у прикладній ПрО (*MMi*) та забезпечення якості власне інтегрованого середовища ГП (*MSi*). Ці задачі мають вирішуватися адміністратором IDE.

Задачі у категоріях *MDi, MPi, MMi* та *MSi*, схематично показані на рис. 3, що фактично доповнює рис. 2 (модель процесу генеруючого ГП).

3.1. Задачі та підходи інженерії якості на стадії розроблення сімейства ПС. На етапі аналізу ПрО вирішуються задачі *MD1* та *MD2*.

Задача MD1. Моделювання набору сумісних нефункційних вимог до сімейства ПС і їх специфікація. При моделюванні нефункційних вимог до сімейства ПС необхідно враховувати, що окремі показники якості можуть конфліктувати, наприклад: висока надійність → низька ефективність, висока ефективність → низька модифіковуваність, висока якість → високі витрати.

Для специфікації конфліктуючих вимог до якості можна будувати «матрицю конфліктів» або «дах будинку якості» і вказувати, який вплив одна на одну роблять окремі характеристики – позитивний, негативний або жодного впливу [1]. Можна також застосовувати метод аналізу ієрархій для впорядкування, зважування характеристик і оптимізації їх цільових значень (з використанням цільової функції).

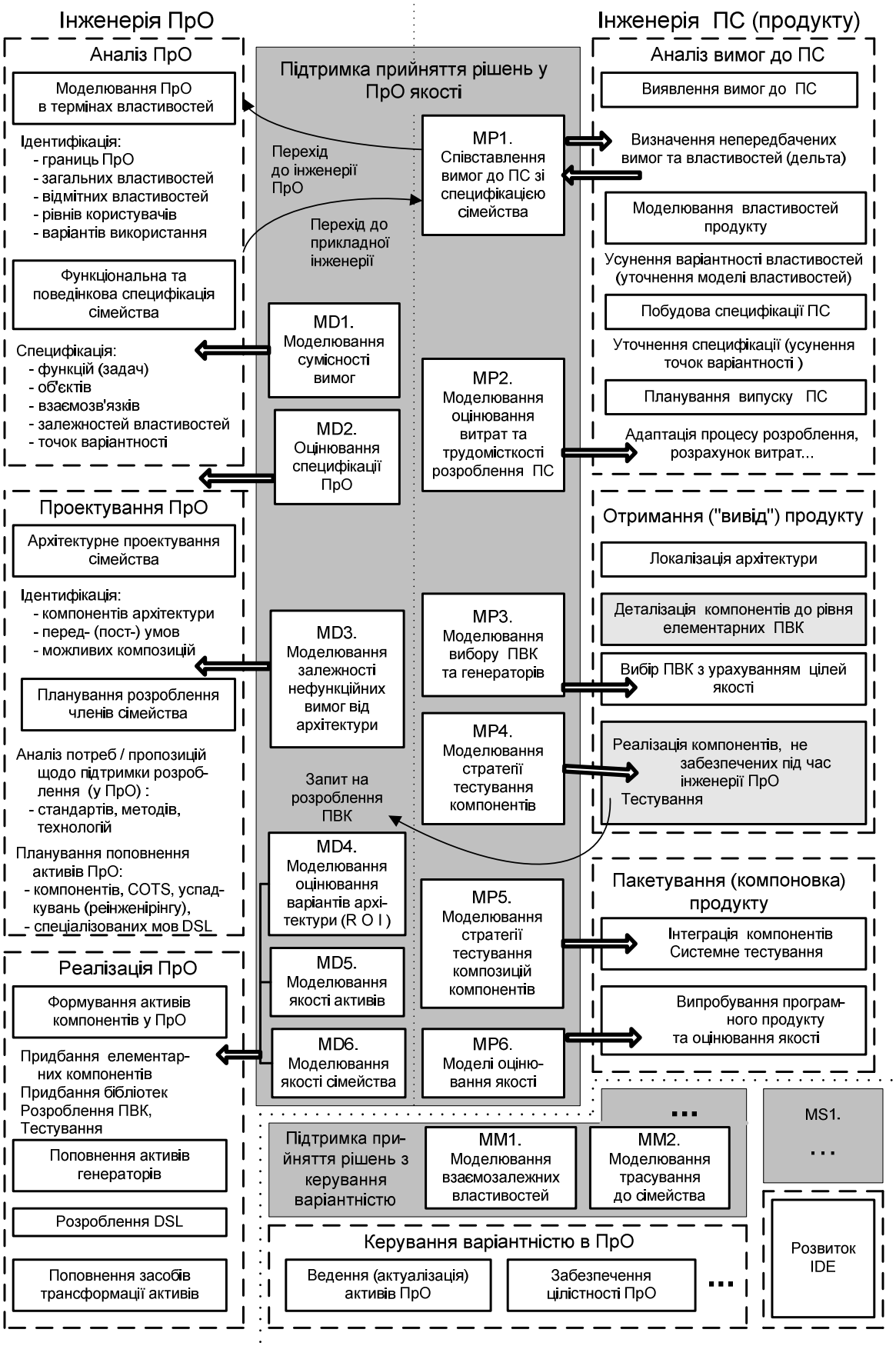

Рис. 3. Групи задач у ПрО інженерії якості сімейства ПС та згенерованих ПС

У разі застосування аспектно-орієнтованого моделювання (АОМ) для побудови сімейств ПС можна було б розширити модель властивостей характеристиками якості і далі використати АОМ для побудови наскрізної моделі якості за кожною характеристикою. Наскрізна трирівнева модель якості пов’язує метрики внутрішньої і зовнішньої якості та якості під час використовування [4].

Задача MD2. Оцінювання специфікації ПрО. Оцінювання специфікації ПрО виконується після верифікації її опису і наступної класифікації та аналізу дефектів. Оцінюватися мають повнота, точність, реалізованість специфікації тощо. Зокрема, треба аналізувати, чи використовуються у специфікації поняття, що виходять за межі ПрО, чи є терміни, не визначені для ПрО тощо.

Верифікація може виконуватися методом формальної інспекції, прототипування і побудови сценаріїв тестування – TestCases [5]. Інспекція проводиться вручну шляхом читання досвідченим експертом (або експертами) з використанням опитувальних листів і заповнення форм даними вимірювань. Аналіз може виконуватися безпосередньо по репозиторію, а певні метрики розраховуватися автоматично (як це робиться, наприклад, у відомих CASE-засобах, зокрема Oracle Designer). Якщо існують XML-шаблони специфікації вимог і інших артефактів ПрО, можна легко моделювати залежності між ними за допомогою XML-links і використовувати їх для трасування артефактів сімейства до вимог до сімейства [5].

На етапі проектування ПрО вирішуються задачі *MD3* та *MD4*.

Задача MD3. Моделювання залежності нефункційних вимог до сімейства ПС від архітектурних особливостей ПрО. Аналітичне моделювання таких залежностей практично неможливе. Пропонується використовувати графічне моделювання. Найбільш перспективним видається клас графічних моделей, заснованих на байєсівських мережах [6]. Загалом, за допомогою байєсівської мережі можна формулювати припущення про існування залежності між різними змінними, що представляють атрибути сутностей, а потім послідовно “поширювати” отримувані об'єктивні дані спостережень по мережі. Відповідним чином можна встановити умовну залежність характеристик якості від атрибутів архітектури і використовувати побудовані мережі при проведенні інспекції архітектури (як, наприклад, у проекті SAABNet) [7].

Задача MD4. Моделювання оцінювання варіантів архітектури ПС (аналіз компромісів з позицій якості, витрат, ресурсів). Стосовно сімейств ПС ще важливішою задачею інженерії якості є оцінювання варіантів архітектури, що виникають в ході еволюції сімейства (у зв'язку з появою нових варіантів у точках варіантності). Під архітектурою може розумітися не лише програмна архітектура, але й системна архітектура, яка включає програмне забезпечення проміжного рівня (middleware) і технічні засоби (hardware). Наприклад, при побудові диспетчерських систем (служб «01», «02», «03») вибір конфігурації обумовлюється масштабом служби, її територіальною розподіленістю, рівнем застосованих сучасних технічних засобів та інформаційних технологій (наприклад, тим, які модеми використовуються – якої швидкості, за якою ціною тощо). У цьому випадку слід враховувати співвідношення ціни-якості і їх залежність від архітектури. Для досягнення компромісів також можуть застосовуватися байєсівські мережі та побудовані на їх основі діаграми корисності (utility diagram). Задача у такому формулюванні є задачею моделювання ризику архітектури.

На етапі реалізації ПрО вирішуються задачі *MD5* та *MD6*.

Задача MD5. Моделювання оцінювання якості активів у репозиторії, що безпосередньо формують сімейства ПС. Вибір засобів моделювання обумовлюється видом активів у репозиторії, а саме: програмних компонентів різних типів (модулів та програм на мовах програмування (наприклад, генераторів), компонентів-сервісів тощо), їх композицій, компонентних каркасів або компонентних потоків робіт (workflow).

Для оцінювання компонентної архітектури і окремих компонентів можна використовувати:

- методи аналізу дерев помилок (FTA) або дерев подій (ETA);
- методи сценарного підходу (методи SAAM, ATAM, CDAV, FAAM) [8], відповідно до яких архітектура має покривати всі сценарії її можливого використання, а також мати високі показники якості за характеристиками інтероперабельність (interoperability), розширюваність (extensibility), модифіковуваність (modifiability) тощо.

Для оцінювання будь-яких активів можна також використовувати підхід RAS, розвинений засобами AOM [9].

Задача MD6. Моделювання оцінювання якості сімейства ПС. Для оцінювання якості сімейства ПС, побудованого в інтегрованому середовищі, необхідно насамперед визначити характеристики якості сімейства. Це може бути, наприклад, повнота (універсальність) відповідно до ПрО, придатність до поповнення, придатність до відторгнення ПС (відокремлення частки сімейства, що формуватиме конкретну ПС), трасовність сімейства, супроводжуваність сімейства тощо.

3.2. Задачі та підходи інженерії якості на стадії створення ПС як члена сімейства. На етапі аналізу вимог до ПС вирішуються задачі *MP1* та *MP2*.

Задача MP1. Зіставлення вимог до ПС з вимогами до сімейства. Вибір підходу до вирішення цієї задачі обумовлений тим, який підхід був застосований для аналізу ПрО і специфікації сімейства. Якщо, наприклад, застосовувався онтологічний підхід, задача зводиться до аналізу онтології сімейства і визначення її повноти та достатності для проектування ПС на базі сімейства. У найпростішому варіанті користувачеві можна було б пропонувати переліки функційних і нефункційних характеристик сімейства та надавати можливість вибирати потрібні характеристики для їх включення у новий перелік характеристик майбутньої системи, а також добавляти нові характеристики. Зіставляються описи функцій сімейства та системи, яку треба розробити, нефункційні характеристики сімейства та майбутньої системи, функційні та технічні обмеження (пов’язані, наприклад, з потребами розмежування доступу, врахування особливостей цільової платформи тощо).

У разі, якщо в специфікації сімейства відсутні належні характеристики, процес специфікації системи має розглядатися у контексті *процесу інженерії ПрО* (рис. 1), оскільки може виникнути потреба доповнення опису ПрО, архітектури тощо. Процес інженерії ПрО у цьому випадку виконується, починаючи з того етапу, на якому виявлена нестача. Прийняті рішення трасуються «вниз» до реалізації ПрО та «вгору» до моделі ПрО, після чого оцінюється цілісність інтегрованого середовища (перш за все репозиторію).

Задача МР2. Моделювання оцінювання витрат і трудомісткості розроблення ПС. Це типова задача інженерії якості ПС [1]. Особливість її вирішення для ПС, які розробляються у парадигмі генеруючого програмування, полягає в тому, що додатково треба моделювати трудомісткість «усунення дельти», величини розбіжностей характеристик майбутньої системи і сімейства, а також враховувати кваліфікацію розробників системи (з огляду на потреби розширення сімейства для ПрО). Оскільки на цей час не існує відповідних регресійних або мультиплікативних аналітичних моделей, пропонується для вирішення поставленої задачі також використовувати апарат байєсівських мереж та розробляти графічні моделі.

На етапі генерації програмного продукту вирішуються задачі *MP3* та *MP4*.

Задача МР3. Моделювання вибору повторно використовуваних компонентів і генераторів. Оскільки вибір ПВК (генераторів) за наявності альтернатив повинен виконуватися з урахуванням множини взаємопов'язаних чинників (якості ПВК, витрат на придбання, розроблення тощо) і не можна точно оцінити, який з ПВК (генераторів) буде найкращим, для вирішення поставленої задачі також можна розробити графічні байєсівські моделі аналізу окремих компонентів та генераторів, які визначатимуть ймовірність, з якою компонент (генератор) вважається придатним для використання у системі. Вибиратиметься компонент (генератор) з вищим значенням ймовірності. Можливе апріорі встановлення рівня довіри до ПВК (генераторів).

Задача МР4. Моделювання стратегії тестування компонентів в середовищі реалізації ПС. Вибір стратегії тестування обумовлений природою компонентів, з яких складається система, і спектром різнотипних компонентів у системі. Існують методології тестування об'єктно-орієнтованого програмного забезпечення, Web-застосувань, сервісів тощо. Крім того, оскільки в більшості випадків початковий код вибираних компонентів недоступний для аналізу і верифікації відомими методами, доцільно у загальній стратегії тестування передбачати додаткові кроки - сертифікаційне тестування, включаючи тестування ПВК (генераторів) розробниками сімейства.

На етапі компонування програмного продукту вирішуються задачі *MP5* та *MP6*.

Задача МР5. Моделювання стратегії тестування композицій компонентів. Стратегія тестування ПС на етапі компонування готового продукту будується з урахуванням того, що можуть тестуватися компоненти у різних композиційних конфігураціях - *каркаси*, які включають різнотипні компоненти, *потоки робіт*, у склад яких входять (у загальному випадку) не лише компоненти або Web-сервіси, але й Grid-сервіси. Для тестування використовуються інструменти тестування, які є частиною інтегрованого середовища ГП.

Задача МР6. Оцінювання якості ПС по моделях. Оцінювання якості ПС виконується за встановленими у специфікації показниками якості (за їх наявності). В разі відсутності аналітичних моделей для оцінювання відповідних характеристик якості (наприклад, моделей зростання надійності) можуть розроблятися байєсівські графічні моделі, які використовуватимуться для обґрунтування досягнення потрібного рівня якості за певними характеристиками [6].

3.3 Задачі інженерії якості стосовно підтримки прийняття рішень з розвитку ПрО. Оскільки варіабельність (варіантність) є невід'ємною характеристикою сімейства ПС, окремою задачею є трасування властивостей сімейства за всією вертикалью та аналіз цілісності сімейства (ПрО) після його доповнення новими артефактами, поява яких обумовлена долученням нового члена до сімейства.

Задачі моделювання взаємозалежних властивостей ПрО (*MM1*), моделювання стратегії трасування одних проектних рішень до інших (*MM2*) тощо, також можна (побічно) віднести до задач інженерії якості, які перегукуються із задачами *MD1-MD6*.

Крім цих задач може бути поставлена задача моделювання якості інтегрованого середовища ГП та періодичного оцінювання якості IDE в зв'язку із його розвитком (зарезервовані задачі *MS1*… на рис. 3).

4. Підходи до верифікації і тестування компонентів сімейства ПС

Відповідно до множини сформульованих задач інженерії якості ПС можна виділити такі категорії задач верифікації та тестування:

- верифікація моделей вимог до ПС, представлених у різних мовах (XML, UML тощо);
- верифікація та тестування окремих активів та композицій активів сімейств ПС, розміщених у репозиторії (специфікацій властивостей ПрО, специфікацій архітектури ПрО, доменних мов (доведення коректності DSL), придбаних COTS-компонентів, генераторів тощо;
- верифікація та тестування робочих продуктів ПС (специфікацій, опису архітектури, компонентів та їх композицій, ПС загалом).

Методи вирішення цих задач суттєво залежать від концептуальних рішень стосовно застосовуваних підходів у інженерії ПрО та інженерії застосувань та архітектурних рішень щодо інтегрованого середовища

генеруючого програмування (зокрема, складу об'єктів IDE, структури репозиторію, вимог до збережуваних артефактів).

Визначення придатних для використання методів верифікації та тестування таких активів, як програмні компоненти СПС, є першочерговою задачею, оскільки вони використовуватимуться при побудові багатьох конкретних ПС різних конфігураціях. Надійність ПВК і їх комбінацій має забезпечуватися незалежно від встановлених вихідних вимог до якості кожної цільової ПС – члена сімейства.

Хоча верифікація ПС, як процес ЖЦ, загалом може виконуватися методами інспекції (перегляду), формального доведення коректності або тестування ПС, перевага в проблематиці генеруючого програмування надається двом останнім підходам, з огляду на специфіку об'єктів, що верифікуються – компонентів з чітко описаними функціями і інтерфейсами [10].

4.1. Підходи до верифікації компонентів. За наявності початкових кодів компонентів найбільш дієвими є методи доведення коректності за формальними специфікаціями. Методи формальних специфікацій і доказового програмування продовжують активно розвиватися, починаючи з 70-х років 20 століття та досить широко представлені в літературі, зокрема, в [11–17].

Основними підходами до формальної верифікації програмних компонентів є дедуктивний аналіз (верифікація на основі логічного виведення) і верифікація на моделі (model checking).

Недолік дедуктивного аналізу полягає в тому, що він вимагає для застосування великого обсягу ручної роботи і високої кваліфікації фахівців. До того ж складно наперед визначити, скільки часу знадобиться для доведення твердження або для його спростування. Тому дедуктивний аналіз застосовується рідко і лише для тих компонентів, які є функціонально критичними (покликані забезпечувати безпеку функціювання або інформації).

Верифікація на моделі є практично повністю автоматичним методом для перевірки властивостей систем з кінцевим числом станів [11]. Як випливає з назви методу, він працює не з реальною системою, а з її моделлю. Для системи, що перевіряється, спочатку будується формальна модель, що описує її поведінку. Потім для неї формулюється специфікація – твердження, істинність яких необхідно перевірити. Після цього виконується власне автоматична верифікація, внаслідок якої або доводиться, що модель задовольняє специфікації, або це спростовується. Спростування являє собою набір дій над моделлю, які призводять до порушення специфікації.

Верификация на моделі краще за тестування, по-перше, тим, що перевіряє не просто деякий набір пар вхід-вихід, а всі можливі варіанти вхідних даних. Перевага методу верифікації на моделі в порівнянні з тестуванням полягає в тому, що у разі невиконання специфікації виводиться сценарій її порушення, аналізуючи який простіше знайти причину помилки. Порушення сценарію може виникати не тільки у разі помилкової моделі системи, але також у разі невірної специфікації. Обидва типи помилок легко знаходяться за допомогою аналізу сценарію.

Хоча верифікація на моделі поступається дедуктивному аналізу тим, що застосовна тільки до моделей з кінцевою кількістю станів, для ПВК це обмеження не суттєве. Перевага методу верифікації на моделі полягає в тому, що перевірка проходить повністю автоматично і для її виконання не потрібно особливих знань і часу.

Тим не менш існують проблеми використання методу верифікації на моделі на всіх етапах його виконання, а саме:

1) *побудова моделі ПВК*. Проблема виконання цього етапу полягає в тому, що досить важко побудувати модель, яка була б адекватна верифіковуваній програмі. Крім того, отримана модель повинна мати невелике число станів, що є умовою її ефективної перевірки. Проте, існуючі методи побудови моделей для програм, написаних традиційним способом, дають дуже велику кількість станів, оскільки в них зазвичай не розділяються керуючі й обчислювальні стани;

2) *формулювання вимог в термінах вибраної темпоральної логіки*. Виконання цього етапу також стикається з проблемами, оскільки вимоги мають формулюватися в термінах моделі і необхідне забезпечення їх адекватності початковій специфікації програми;

3) *верифікація моделі з метою перевірки дотримання формалізованих вимог*. У порівнянні з першими двома етапами, цей етап достатньо добре автоматизується. За допомогою верифікаторів, наприклад, SPIN, CPN Tools або Bogor.

4.2. Підходи до тестування компонентів. Основні особливості процесу тестування компонентних ПС обумовлені процесом їх розроблення, а саме:

- процес розроблення ПС з компонентів відокремлений від процесу розроблення самих компонентів;
- розроблення ПС включає пошук (придбання) потрібних компонентів; їх адаптацію і тестування;
- основні зусилля в ЖЦ зосереджені на інтеграції («вбудовуванні») компонентів в ПС (тобто тестуванні архітектури ПС).

Відповідно, тестування компонентної ПС проходить на декількох рівнях:

1) *тестування компоненту в ізоляції (автономне)*. Проблема виконання цього тестування полягає в тому, що початковий код компонента недоступний, а взаємодія компонента з ПС відбувається тільки через його інтерфейси;

2) *тестування в процесі інтеграції*. На цьому етапі в найбільшій мірі проявляється специфіка компонентного тестування, яка полягає у виборі об'єктів тестування:

- тестування операторів взаємодії з акцентом на тестування пов'язуючого коду, який взаємодіє з компонентами (через інтерфейси компонентів);

- тестування послідовності взаємодії із спостереженням послідовності операторів в пов'язуючому коді, який взаємодіє з компонентами;

- тестування параметрів з акцентом на взаємодію між пов'язуючим кодом і компонентами шляхом спостереження параметрів кожної взаємодії;

- тестування потоку інформації з акцентом на потік інформації в пов'язуючому коді в контексті взаємодії з компонентами шляхом спостереження послідовності дій взаємодії і їх параметрів для кожного виконання тесту.

3) *системне тестування* – інтегрованої системи в різних призначених середовищах експлуатації. Виконується традиційними методами тестування за сценаріями використання з акцентом на нефункційні характеристики ПС.

4.3. Підходи та особливості тестування сімейств програмних продуктів. Тестування сімейств ПП є частиною інженерії ПрО й інженерії застосувань. Воно базується на принципах і методах тестування компонентів і охоплює фази: модульне (автономне) тестування, інтеграційне і системне.

Тестування починається на етапі аналізу вимог, представлених у вигляді графічної моделі. У якості інструментів моделювання та аналізу моделі вимог, а також як основа для генерації тестів, можуть застосовуватися промислові інструменти, зокрема, діаграми розширених use-cases на UML (Rational Rose).

Підходи до тестування в інженерії ПрО відрізняються від підходів до тестування окремих компонентів.

Одна з цілей тестування в інженерії ПрО – повторне використання продуктів тестування. У поєднанні з раннім тестуванням в процесі розроблення це може надавати ті самі переваги в сімействах ПС, як і саме повторне використання ПС.

Продукти тестування (тестові плани і набори тестів) мають бути придатні для інших ПС і є частиною повторно використовуваних компонентів сімейства ПС.

Рекомендації щодо тестування сімейств ПС, які наведені в літературі, традиційні – відображення програмної архітектури на тестову архітектуру.

Одним з підходів, який може застосовуватися для тестування сімейств ПС, є підхід, базований на сценаріях (Scenario-based test derivation) [8], за яким набори тестів генеруються в інженерії ПрО за допомогою use-case моделей і use-case специфікацій, розширюваних специфічними характеристиками.

За допомогою цих специфічних характеристик можна вивести набори тестів з тестових артефактів при створенні застосувань, оскільки всі можливі варіанти вже моделюються в цих наборах тестів.

Через велику кількість можливостей комбінування і інтеграції компонентів сімейства ПС, пропонуються комбінаторні підходи до розробки тестів [18].

Через специфіку об'єкта тестування найбільш прийнятними підходами до тестування є комбінаторні, оскільки вони враховують різні комбінації характеристик членів сімейства і дозволяють автоматизувати генерацію тестів.

На сьогодні є кілька методологій тестування сімейств ПС, одна з них – PLUTO (Product Lines Use case Test Optimization), яка охоплює задачі тестування від вимог до випуску [19].

Основні підходи до тестування та об’єкти тестування сімейств ПС підсумовано в таб. 2.

Таблиця 2. Основні підходи до тестування сімейств ПС

Підходи до тестування	**Об'єкти тестування**
Архітектура тестування відображає програмну	Всі продукти сімейства, окремі продукти, окремі компоненти
Набір тестів сімейства адаптується як набір тестів конкретного продукту	Всі продукти сімейства, окремі продукти сімейства
Тестування базується на вимогах	Окремі продукти сімейства
Самотестування	Окремі компоненти
Застосування метаданих	Окремі компоненти
«Тестопридатність компонентів» "testable beans" (для підвищення продуктивності)	Окремі компоненти
Підходи, засновані на якості сервісів і надійності (для веб-застосувань)	Окремі компоненти
Автоматична генерація тестів по специфікаціях в булевій формі	Всі продукти сімейства, окремі продукти сімейства
Застосування дерев помилок	Всі продукти сімейства, окремі продукти сімейства

Висновки

Формування цілісної концепції інженерії якості ПС у парадигмі генеруючого програмування потребує аналізу застосовності існуючих досягнень, їх узгодження, розроблення нових методів та практичних прийомів забезпечення якості створюваних сімейств ПС.

На поточному етапі досліджень зроблено лише перший, але важливий, крок у даній проблематиці – визначено множину задач, проблем та підходів в інженерії якості ПС нової генерації. На часі їх конкретизація з урахуванням потреб інтеграції з загальною методологією генеруючого програмування ПС і розроблення моделей та методів верифікації, тестування, прогнозування, забезпечення та оцінювання нефункційних властивостей компонентів та сімейств ПС, призначених для вирішення задач інженерії якості ПС на всіх стадіях ЖЦ їх розроблення за парадигмою генеруючого програмування.

1. *Основы* инженерии качества программных систем / Ф.И. Андон, Г.И. Коваль, Т.М. Коротун, Е.М. Лаврищева, В.Ю. Суслов // 2-е изд. – Киев: Академпериодика, 2007. – 672 с.
2. *de Almeida E.S., Alvaro A.* et al. A Survey on Software Reuse Processes // http://www.cesar.org.br/pdf/ASurveyonSoftwareReuseProcesses.pdf
3. *ISO/*IEC 12207:1995 (Amd 1:2002, Amd 2:2004). Information Technology – Software life cycle processes. – ISO. – 106 p.
4. *Лаврищева Е.М., Коваль Г.И., Коротун Т.М* Подход к управлению качеством программных систем обработки данных // Кибернетика и системный анализ. – 2006. – №5. – С. 174-185.
5. *Requirements* engineering for software product lines // http://www.enel.ucalgary.ca/People/eberlein/publications/ProdLine_ICSSEA2002.pdf
6. *Коваль Г.І.* Байєсівські мережі як засіб оцінювання та прогнозування якості програмного забезпечення // Проблеми програмування. – 2005. – № 2. – С. 15–23.
7. *Van Gurp J*. Variability in Software Systems The Key to Software Reuse http//publications.jillesvangurp.com/Lic/ licentiatethesis.pdf
8. *Scenario-Based* Software Architecture Evaluation Methods: An Overview // http://www.win.tue.nl/oas/architecting/aimes/papers/Scenario-Based SWA Evaluation Methods.pdf
9. *Huang G.,* Mei H., Wang Q. Towards Software Architecture at Runtime // Software Engineering Notes Vol. 28, N 2, March 2003 // http://portal.acm.org/citation.cfm?id=638780/2003-2.pdf
10. *Грищенко В.Н., Лаврищева Е.М.* Методы и средства компонентного программирования // Кибернетика и системный анализ. – 2003. – № 1. – С. 39 – 55.
11. *Кларк Э., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking. М.: МЦНМО. 2002. 416 с.
12. *Hoare C.A.R.* Prof of correctness of data representation // Acta Informatica, 1(4) /-271-287. – 1972. – P. 214–224.
13. *Андерсон Р.* Доказательство правильности программ. – М.: Мир, 1982. – 165 с.
14. *Abrial I.R., Meyer B.* Spesification Language Z. – Boston: Massachusetts Computer Associates Inc. – 1979.–378 p.
15. *Петренко А.К.* Віденський метод розробки программ // Програмування, 2001. – №1. – С. 3–23.
16. *Вудкок Д.* Первые шаги к решению проблемы верификации программ. Открытые системы, 2006. – №8. – С. 36–43.
17. *Hoare T., Misra J.* Verified software: Theories, Tools, Experiments. Vision of Grant Challenge project. –Microsoft Research Ltd and the University of Texas at Austin. – 2005. – 1 – 43 c.
18. *Metzger A.* Model-based Testing of Software Product Lines //http://www.andreas-metzger.net/pub/MPR06.pdf
19. *Bertolino A., Gnesi S.* PLUTO: A Test Methodology for Product Families //http://fmt.isti.cnr.it/WEBPAPER/23-BertGnesi.pdf