ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АСКП — автоматизована система контролю польотів
БД — база даних
ДНТП — державна науково-технічна програма
ДКНТ — Державний комітет України з питань науки і техніки
ЖЦ — життєвий цикл
ІС — інформаційна система
ІТ — інформаційні технології
ЛА — літальний апарат
ООП — об’єктно-орієнтований підхід
ПЗ — програмне забезпечення
ПП — програмний продукт
ПС — програмна система
ТЗ — технічне завдання
ТНД — тестовий набір даних

ВСТУП

Сучасний рівень розвитку комп'ютерних технологій проектування та подальшого виготовлення інформаційних систем (ІС) характеризується зростаючою складністю не тільки програмних компонентів, але й концепцій та ідей, що лежать в основі цих технологій. У цих умовах особливо важливим є об'єктивне оцінювання якості як кінцевого програмного продукту (ПП), що входить до складу ІС, так і його оцінювання на кожному етапі життєвого циклу (ЖЦ).

Навчальний посібник розроблено відповідно до робочої навчальної програми дисципліни «Стандартизація та сертифікація інформаційних управляючих систем», яка призначена для спеціалістів та магістрів спеціальності 7/8.05010101 «Інформаційні управляючі системи та технології». Мета створення посібника — надання студентам знань у сфері сучасних наукових концепцій, методів і технологій забезпечення якості програмних систем (ПС), що реалізується шляхом упровадження вимог та рекомендацій національних і міжнародних стандартів у процеси розроблення, атестації та сертифікації ПС.

Посібник розроблено відповідно до вимог кредитно-модульної системи оцінювання знань. У процесі вивчення першого модуля студенти здобувають знання про методи формування вимог до властивостей ПС, складу класів вимог до оцінюваних ПС та методів формалізованого опису вимог. Студенти вивчають структуру і процедури побудови узагальненої та частинних моделей якості ПС (зовнішньої, внутрішньої та експлуатаційної), які відповідають сформованим вимогам до ПС.

У процесі вивчення другого модуля студенти ознайомлюються з методами та засобами визначення досягнутих значень показників якості відповідно до побудованої моделі якості ПС. Студенти вивчають методи оцінювання атрибутів характеристик якості, користуючись відповідними метриками, що запропоновані в міжнародних стандартах. Крім того, студенти вчаться оцінювати процеси ЖЦ ПС та зрілість організацій-розробників ПП, а також ознайомлюються із процесом сертифікації ПС.

1. ІНЖЕНЕРІЯ ВИМОГ ДО ПРОГРАМНИХ СИСТЕМ. ПОБУДОВА ТА ДОСЛІДЖЕННЯ МОДЕЛЕЙ ЯКОСТІ ПРОГАМНИХ СИСТЕМ

1.1. Процеси життєвого циклу програмного забезпечення

Будь-яка програмна система є компонентою деякої комп'ютерної (інформаційної) системи, яка в свою чергу є складовою деякої кінцевої системи (бізнес-системи). Зв'язок між системами показано на рис.1.1.

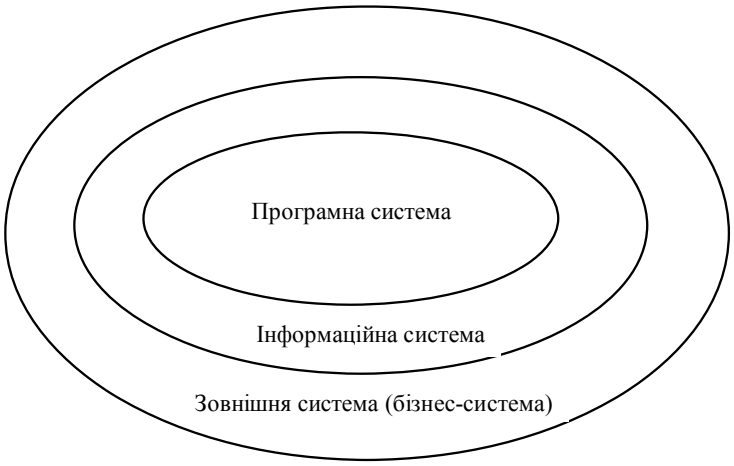

Рис. 1.1. Зв'язок між системами

Програмна система — це група інтегрованих програмних засобів, створених для вирішення множини завдань, специфікованих у межах заданого домену (предметної галузі).

Інформаційна система — це персонал та програмна система, що функціонує в межах домену на апаратних платформах з операційними системами (середовищами).

Бізнес-система — це ІС, розвинена сукупністю бізнес-застосувань. Бізнес-системи використовуються на підприємствах, які виробляють продукцію, у торговельних фірмах, магазинах, банках тощо.

У посібнику основну увагу приділено розгляду саме *програмних систем*, які є ядром кожної ***інформаційної системи***.

Мета посібника — розкриття наукових концепцій, понять і методів забезпечення якості ПС шляхом упровадження в процеси ЖЦ ПС вимог і рекомендацій національних та міжнародних стандартів у сфері програмних засобів.

Кожний програмний засіб за час свого існування проходить фази, які називають *етапами життєвого циклу*. На кожній із цих фаз виконується певна сукупність процесів, котрі породжують деякий продукт. Побудова певного продукту закінчується створенням його опису. Етапи ЖЦ визначені стандартом ISO/IEC 12207:1995 — *Information Technology — Software life cycle processes* (процеси ЖЦ ПЗ) [1]. Процеси ЖЦ поділяють на три групи:

- головні;
- допоміжні;
- організаційні.

Головні процеси: процес купівлі (ініціація ЖЦ ПС і визначення організації, що ініціює розроблення/купівлю); процес розроблення (дії організації розробника); процес постачання (передача ПС покупцю); процес експлуатації; процес супроводження (керування модифікаціями ПС та інсталяція нових версій) [2].

Процес розроблення включає:

1) інженерію вимог до системи;

2) проектування ПС;

3) кодування й тестування ПС.

Допоміжні процеси — процеси, які забезпечують якість ПС (приведення ПС у відповідність до вимог та рекомендацій стандартів).

Якість ПС — це сукупність властивостей ПС, які забезпечують її здатність задовольняти вимоги замовників та користувачів.

Організаційні процеси:

- менеджмент розроблення;
- навчання персоналу;
- визначення обов'язків кожного з учасників процесів ЖЦ.

Вітчизняний стандарт ДСТУ 3918 [2] гармонізований зі стандартом [1]. Зміни та поправки до стандарту ISO/IEC 12207 викладено у стандартах [3; 4]. Ці зміни в основному стосуються розширення складу процесів ЖЦ процесами *придбавання* та *постачання* ПЗ. Керівництво із застосування стандарту [1] наведено в стандарті ISO/IEC 15271:1998 *Guide for* ISO/IEC 12207 *Software Life Cycle Processes* [5], а процеси ЖЦ комп'ютерних систем описано в стандарті ISO/IEC 15288:2002 [6].

Стандарт ДСТУ ISO/IEC 15288:2005 [7] гармонізований зі стандартом [6], а нова редакція стандарту ISO/IEC 15288 вийшла у 2008 р. [8].

Організація зі стандартизації IEEE (*Institute of Electrical and Electronics Engineering*) одразу після виходу стандарту ISO/IEC 12207 видала відповідний стандарт IEEE/EIA Std. 12207.0:1996 [9], а також стандарт IEEE/EIA Std. 12207.1:1997, що стосується даних ПЗ у ЖЦ [10].

Відповідно до стандарту [1] процес розроблення ПС складається з низки характерних робіт, які показано на рис. 1.2.

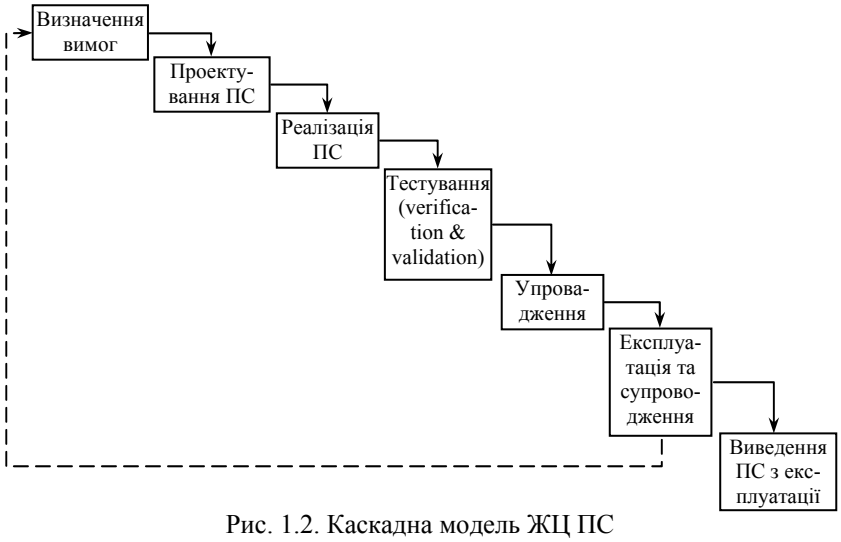

Рис. 1.2. Каскадна модель ЖЦ ПС

Визначення вимог — збирання і аналіз вимог замовника та подання їх у нотації, зрозумілій як замовнику, так і розробнику.

Проектування ПС — перетворення вимог до ПС у послідовність проектних рішень, формування загальної структури та архітектури ПС і узгодження її з предметною галуззю.

Реалізація ПС — перетворення проектних рішень у завершену ПС.

Тестування (verification) — тестування ПС у цілому — перевірка всіх модулів ПС і способів їх інтеграції. ***Тестування (validation)*** — тестування відповідності функцій ПС специфікаціям вимог.

Упровадження — сукупність дій з уведення ПС у робочий стан та дії з передачі її замовнику для експлуатації.

Експлуатація та супроводження ПС — сукупність дій з використання готової ПС замовником та дії із забезпечення її роботи розробником. Це дії із внесення розробником змін у випадку виявлення помилок у процесі експлуатації, а також дії з адаптації ПС до нового середовища функціонування та дії з підвищення продуктивності або інших характеристик ПС.

Виведення ПС з експлуатації — сукупність дій з припинення функціонування ПС та дії з її утилізації.

З огляду на невідворотність еволюції на базі каскадної моделі побудовано спіральну модель ЖЦ, яку показано на рис. 1.3, де v.1, v.2, v.3 і т. д. — версії ПС.

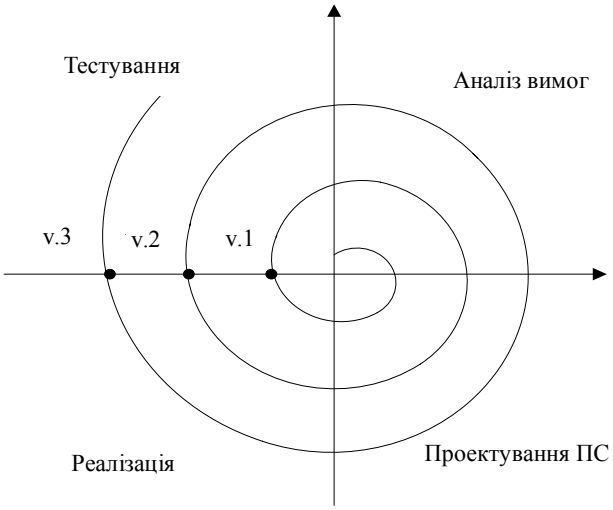

Рис. 1.3. Спіральна модель ЖЦ

1.2. Інженерія вимог

Програмна система вводиться в реальний світ для того, щоб впливати на процеси реального життя. Ті частини реального світу, які впливають на систему або зазнають її впливу, складають ***домен прикладної предметної галузі*** або ***домен використання.***

Вимоги до ПС стосуються тих властивостей, які повинна мати система, якщо вона адекватно виконує свої функції.

Фаза визначення вимог до ПС є найважливішою для якості ПС і мінімізації вартості робіт.

Ціна помилок і неоднозначностей на цьому етапі дуже висока. Статистика показує, що відсоток помилок у постановці вимог перевищує відсоток помилок у реалізації ПС [11].

Діючими персонами в процесі формулювання вимог є:

1) носії інтересів замовника (часто декілька професійних груп);

2) оператори, які здійснюють обслуговування ПС;

3) розробник ПС.

1.2.1. Збирання вимог

Джерела вимог:

1. Мета і завдання системи, які формулюються замовником. Існує небезпека неоднозначного розуміння вимог.

2. Діюча система або колектив, який виконує її функції. Досить часто необхідно замінювати новою системою попередню систему, яка потребує оновлення *(реінжиніринг)*. Тоді процес складання вимог проводять за три кроки:

1) вивчення фізичної структури діючої автоматизованої системи;

2) здійснення логічного узагальнення з виділенням тих функцій, які відображають нові проблеми, які потребують вирішення;

3) визначення логічних розширень функцій, виявлених на другому кроці, що відповідають потребам нової системи, формулювання визначених функцій як вимог до нової системи.

3. Загальні знання про проблемну галузь замовника (використання галузевих стандартів, якщо такі є).

Методи збирання вимог:

– інтерв'ю з носіями інтересів замовника й операторами;

– спостереження за роботою діючої системи з метою відділення її проблемних властивостей від тих, які обумовлені структурою кадрів;

– сценарії (приклади) можливих випадків виконання її функцій, а також ролей осіб, що запускають ці сценарії або взаємодіють із системою під час її функціонування.

Продукт процесу збирання вимог — неформалізований їх опис (або технічне завдання (ТЗ)) — основа контракту на розроблення між замовником і виконавцем розроблення системи. Цей опис фактично стає контрактом на розроблення ПС між замовником і розробником.

Базовим стандартом, який визначає порядок і умови збирання та постановки вимог до ПС, є міжнародний стандарт IEEE Std 830 — *Recommended Practice for Software Requirements Specifications* [12].

Цей стандарт складається з таких розділів:

1. Вступ:
- мета;
- галузь застосування;
- терміни;
- посилання.

2. Основна частина. Загальний опис.

2.1. Перспективи програмного продукту:
- системні інтерфейси;
- інтерфейси користувачів;
- операційні інтерфейси;
- програмні інтерфейси;
- комунікаційні інтерфейси;
- використання пам’яті та операцій.

2.2. Функції продукту.

2.3. Користувацькі характеристики.

2.4. Обмеження.

2.5. Припущення й залежності.

2.6. Розподіл вимог:
- *C*-вимоги — вимоги користувача (*customer*);
- *D*-вимоги — вимоги розробника (*developer*).

3. Конкретні вимоги.

3.1. Вимоги до зовнішнього, користувацького, програмного, апаратного та комунікаційного інтерфейсів.

3.2. Класи й об’єкти (функціональні вимоги).

3.3. Вимоги продуктивності (нефункціональні вимоги).

3.4. Обмеження на проектування.

3.5. Атрибути ПС.

3.6. Інші вимоги.

C-вимоги — неформалізовані, а *D*-вимоги — детальні вимоги, які можуть бути функціональними та нефункціональними. Функціональні вимоги належать до функціональних характеристик системи, а нефункціональні — до якісних властивостей, які безпосередньо не стосуються функціональності ПС (характеристики роботи ПС) [13].

Відповідно до праці [12] специфікація вимоги до ПЗ (SRS) має бути:

- несуперечливою;
- з можливістю контролю;
- тестованою;
- узгодженою;
- повною.

D-вимоги складаються за допомогою функціональних специфікацій, що містять повний перелік усіх властивостей і функцій, які повинна мати система. Ці вимоги мають відповідати *C*-вимогам.

Із питань розроблення специфікацій вимог до ***комп'ютерних систем*** видано також стандарт IEEE Std. 1233:1998 [14]. Крім того, у праці [15] детально розглянуто технології формування специфікацій вимог до ПЗ, зокрема шаблони вимог.

1.2.2. Аналіз вимог

1. Класифікація вимог. Множина вимог поділяється на дві категорії:

- функціональні;
- нефункціональні.

Існує кілька класів нефункціональних вимог, важливих для більшості ПС. Вони виражають ті обмеження, які актуальні для більшості предметних галузей:

- вимоги до конфіденційності;
- вимоги до відмовостійкості;
- вимоги до кількості клієнтів, що одночасно мають доступ до системи;
- вимоги до безпеки;
- вимоги до часу очікування відповіді від системи;
- вимоги до властивостей системи під час виконання її функцій (обмеження на ресурси пам'яті або процесора, швидкість реагування на звернення до ПС тощо).

Для більшості цих обмежень може бути зафіксований спектр характерних понять — *дескрипторів*, які використовують для найменування та розкриття змістовної назви. Склад дескрипторів для ряду нефункціональних вимог зафіксований у відповідних міжнародних, національних і галузевих (відомчих) стандартах, що дозволяє уникнути неоднозначності їх тлумачення.

Функціональні вимоги пов’язані із семантичною особливістю предметної галузі, а тому проблема термінологічних розбіжностей для них є чинником ускладнення.

2. Установлення пріоритетності, оскільки вимоги висунуті різними носіями інтересів і можуть конфліктувати між собою. Крім того, для реалізації кожної вимоги необхідний ресурс.

3. Передбачення можливих змін у зібраних вимогах і можливості подальшого внесення змін у ПС без істотного її перегляду.

4. Необхідність можливості перевірки правдивості вимог і їх відповідності інтересам замовника.

Продуктом процесу аналізу вимог є побудована модель проблеми, орієнтована на її розуміння виконавцем до початку проектування системи [11].

1.3. Концептуальне моделювання проблеми та об’єктно-орієнтована інженерія вимог

Процес побудови моделі проблеми називається *концептуальним моделюванням*. Кожна предметна галузь (домен) має систему понять, яка відома тільки професіоналам. Тут є система замовчування, характерні властивості домену, відношення між об’єктами і правила поведінки.

Роль концептуальної моделі полягає у посередництві між професіоналами, які обізнані з різними доменами (наприклад, фахівцями з бухгалтерського обліку і програмістами).

1.3.1. Онтологія домену

Сукупність термінології, понять, характерних відношень, а також парадигми їх інтерпретації в межах ***домену*** називається *онтологією домену*. Серед відношень між об’єктами домену найбільш відомі:

- узагальнення;
- конкретизація;
- агрегація;
- асоціація.

Онтологія дозволяє утримувати користувачів у максимально можливому просторі визначених наперед можливостей, зміст яких зафіксований і зрозумілий як замовнику, так і розробнику.

1.3.2. Моделі динамічних властивостей доменів

Розглянуті вище властивості належать до статичних властивостей домену. У більшості завдань ПС, які вирішуються за допомогою комп'ютерів, використовують динамічні процеси. Для динамічних доменів істотними поняттями є:

1) стан (домену, системи, об'єкта) — фіксація певних властивостей об'єктів на певний момент (чи інтервал) часу;

2) інтервал стабільності — інтервал часу, протягом якого не змінюється стан;

3) подія — явище, що провокує зміну станів.

Серед динамічних доменів існують такі різновиди:

– інертні — стан не змінюється до настання впливу зовнішніх агентів;

– реактивні — зміна стану як відповідь на певну зовнішню подію;

– активні — перехід зі стану в стан без зовнішніх стимулів.

Найбільш відомою моделлю поведінки динамічних явищ є ***модель переходів у стани (МПС)***, яка базується на моделі скінченного автомата (найчастіше автоматів Мілі або Мура).

1.3.3. Об'єктно-орієнтована інженерія вимог

Основними елементами об'єктно-орієнтованого підходу (ООП) є *сутності*, для яких визначаються *стани*. Взагалі архітектуру системи становлять компоненти (сутності) і правила їх композиції. Відомі такі моделі, що визначають архітектуру ПС:

– модель функції-дані;

– об'єктна модель.

Модель функції-дані історично склалася першою. Відповідно до цієї моделі проблема декомпозується на послідовність функцій і даних, що обробляються за допомогою цих функцій. Елементами композиції є дані та функції їх оброблення.

Для розроблення ПС об'єктна модель є більш потужною порівняно з моделлю функції-дані. Основні концепції ООП:

– світ складають об'єкти, які взаємодіють між собою;

– кожний об'єкт має певний набір властивостей (атрибутів);

– об'єкти можуть вступати між собою у відношення, які можуть змінюватися у часі;

– сукупність значень атрибутів об'єкта у певний момент часу зумовлює його стан;

– сукупність станів усіх об’єктів визначає стан предметної галузі в цілому;

– у певні моменти часу виникають події, які викликають наступні події або зміни станів об’єктів;

– дії, які виконує об’єкт, називають *операцією* (функцією, методом);

– сукупність дій об’єкта називають його *поведінкою*;

– об’єкти можуть бути складними і взаємодіють між собою через обмін повідомленнями.

Об’єкт характеризується:

– інкапсуляцією;

– успадкуванням (чи спадковістю);

– поліморфізмом.

Згідно з ООП концептуальне моделювання проблеми полягає у вираженні взаємодії об’єктів:

1) онтологія домену визначає склад об’єктів домену, їх атрибутів і відношень, а також операцій;

2) модель поведінки визначає можливі стани об’єкта та інциденти, які ініціюють переходи з одного стану в інший, а також повідомлення, які надсилаються;

3) модель процесів визначає дії, які виконують об’єкти.

Відомі такі ООП опису вимог [11]:

– метод Шлеєр і Меллора ;

– метод сценаріїв (метод Джекобсона);

– методологія UML.

1.4. Метод інженерії вимог С. Шлеєр і С. Меллора

Програмна система згідно з цим методом створюється як сукупність певної множини доменів предметних галузей, кожний з яких являє собою окремий світ зі своїми об’єктами. При цьому кожний такий домен аналізується незалежно від інших.

Продуктом аналізу домену є три моделі:

– інформаційна модель системи (онтологія домену);

– модель станів об’єктів, визначена у складі інформаційної моделі (або онтології);

– модель процесів, що забезпечують переходи з одного стану об’єкта в інший.

Відповідно до методу Шлеєр і Меллора результатами аналізу вимог з метою створення ПС є такі продукти інженерії вимог [16]:

1. Інформаційна модель системи (онтологія) у формі:
- діаграми сутність—зв'язок;
- описи об'єктів та їх атрибутів;
- описи зв'язків між об'єктами.

2. Модель поведінки об'єктів системи у формі:
- діаграм і таблиць переходів у стани;
- описів дій та подій діаграм і таблиць переходів у стани.

3. Модель процесів для станів об'єктів у вигляді:
- діаграм потоків даних дій;
- таблиць процесів станів;
- описів процесів (природною мовою).

Сукупність перерахованих продуктів вважається достатньою для переходу до проектування ПС.

1.5. Метод інженерії вимог І. Джекобсона

Метод Джекобсона — це єдиний метод, що вказує послідовний, достатньо формалізований підхід до виявлення об'єктів, істотних для будь-якої предметної галузі.

Метод Джекобсона ґрунтується на варіантах використання системи (сценаріях). На першому кроці складна система декомпозується на більш прості складові. Розроблення системи починається з осмислення мети системи, тобто для кого і для чого створюється ПС. Складність загальної мети виражається через окремі складові мети. Складові мети можуть відповідати функціональним і нефункціональним вимогам, а також проектним рішенням.

Функціональні вимоги — це вимоги до функцій системи.

Нефункціональні вимоги — це вимоги, наприклад, до надійності, безпеки, ефективності, тестованості тощо [17; 18].

Складові мети є джерелом вимог до системи і класифікуються на обов'язкові й бажані. Складові мети можуть перебувати в певних відношеннях (узгодженість, конфлікт, кооперація, залежність тощо).

На другому кроці визначаються носії інтересів, яким відповідає кожна складова мети, і можливі сценарії [11; 16]. Відбувається послідовна декомпозиція проблеми:

1) складні проблеми трансформуються в сукупність складових мети;

2) кожна із складових мети трансформується в сукупність можливих прикладів використання системи (сценаріїв);

3) сценарії трансформуються в процесі аналізу в сукупність взаємодійних об'єктів.

У такий спосіб будується ланцюжок трансформації: проблема — складові мети — сценарії — об'єкти. Проведена трансформація відображається у термінах базових понять предметної галузі [17].

Послідовний підхід до виявлення об'єктів, істотних для предметної галузі, складається з таких кроків [11]:

1. Множина можливих об'єктів визначається на етапі побудови інформаційної моделі предметної галузі (*E-R* діаграми домену).

2. Склад об'єктів уточнюється після побудови комплекту діаграм сценаріїв для моделювання вимог до проектованої ПС (можуть появитися нові об'єкти та скоротитися склад об'єктів з п. 1, якщо вони не використовуються в сценаріях).

3. Остаточно склад об'єктів визначається після побудови діаграм аналізу вимог для кожного сценарію (можуть появитися нові об'єкти і скоротитися склад об'єктів з п. 2, якщо вони не використовуються у відповідних діаграмах взаємодії).

Продуктами інженерії вимог згідно з методом Джекобсона є:

1. Онтологія домену (для нотації предметної галузі можна скористатися діаграмами Чена, тобто ERD).

2. Моделі сценаріїв.

3. Неформальний опис сценаріїв і акторів.

4. Опис інтерфейсів, сценаріїв і акторів.

5. Діаграми взаємодії об'єктів у межах сценаріїв (будуються на основі аналізу кожної вимоги).

Подальша деталізація вимог відбувається на етапах ЖЦ ПС, при цьому зберігається трасування вимог (відстеження відповідних об'єктів протягом ЖЦ, принаймні на всіх етапах розроблення ПС) [11].

1.6. Метод UML — потенційний стандарт засобів моделювання в програмній інженерії

Уніфіковану мову моделювання (*Unified Modeling Language* — UML) розробили Г. Буч, Дж. Рамбо та І. Джекобсон. Метод UML став базовим методом для розроблення ПЗ і є стандартом де-факто

як метод моделювання ПП на всіх стадіях ЖЦ ПС. Автори визначили UML як мову для специфікації, візуалізації, конструювання та документування ПС, а також мову моделювання бізнес-застосувань.

В основу методу покладено парадигму об'єктного підходу, за якого концептуальне моделювання проблеми виконується в термінах взаємодії об'єктів. При цьому:

– онтологія домену визначає склад класів об'єктів домену, їх атрибутів і взаємозв'язків, а також послуг (операцій), які можуть виконувати об'єкти класів;

– модель поведінки визначає можливі стани об'єктів, інциденти, що ініціюють переходи з одного стану в інший, а також повідомлення, якими обмінюються об'єкти;

– модель процесів визначає дії, які виконують об'єкти.

В UML концептуальна модель вимог розглядається як сукупність нотацій — діаграм, які візуалізують основні елементи структури системи.

Діаграми UML версії 1.0:

1) діаграма сценаріїв (варіанти використання);

2) діаграма класів;

3) діаграми поведінки:

– діаграма станів;

– діаграма активності;

4) діаграми взаємодії:

– діаграма послідовності;

– діаграма кооперації;

5) діаграми реалізації:

– діаграма компонентів;

– діаграма розміщення.

Відмінності мови UML 2.0 полягають в обов'язковому використанні основних пакетів метамоделі, а також діаграм композитної структури і додаткових діаграм структури (діаграми пакетів та об'єктів). Крім того, застосовуються допоміжні діаграми взаємодії (діаграма комунікації, огляду взаємодії та часова діаграма), а також діаграма скінченного автомата (*State machine diagram*).

Кожен вид діаграм відображає різні аспекти бачення і розуміння проблеми [11; 13; 17; 19]. Вимоги до системи задаються в основному у вигляді перших двох діаграм (сценаріїв та класів).

1.7. Проектування, реалізація та супроводження програмної системи

Проектування — це етап ЖЦ ПС — наступний після інженерії вимог. Завданням цього етапу є перетворення вимог замовника у проектні рішення, що забезпечать побудову ПС з характеристиками, які відповідають вимогам.

Класифікацію програмних засобів наведено у стандарті ДСТУ ISO/IEC 12182-2004 [20], а словники з розроблення систем та фундаментальних термінів подано у стандартах [21; 22].

Під час проектування відбувається трансформація множини вимог у простір проектних рішень. При цьому можна виділити процеси, які є відносно незалежними один від одного і які можна виконувати як послідовно, так і одночасно командами розробників.

Це такі процеси:

1) ***концептуальне проектування***, яке складається з уточнення розуміння вимог й узгодження деталей вимог;

2) ***архітектурне проектування***, що полягає у визначенні головних структурних особливостей створюваної системи;

3) ***технічне проектування***, що полягає у відображенні вимог середовища функціонування і платформи розроблення системи у проектні рішення та визначення конструкцій у вигляді компонентів системи або їх композиції;

4) ***детальне проектування***, що слугує для визначення деталей та подробиць функціонування і зв'язків для всіх компонентів системи.

В основу проектування ПП покладено принцип декомпозиції складного завдання на окремі більш прості складові (компоненти) [11].

Базовим стандартом опису проектування ПЗ є стандарт IEEE Std. 1016:1998 — IEEE *Recommended Practice for Software Design Descriptions* [23]. Він містить керівництво зі складання документації проекту розроблення, у тому числі має розділи: мета і опис проекту, описи декомпозиції (модульна декомпозиція, декомпозиція на паралельні процеси, декомпозиція даних); описи залежностей (міжмодульні, міжпроцесні, залежності між даними); описи інтерфейсів (модульні та процесів) і опис детального проектування (модулів, даних). У свою чергу, стандарт ISO 13407:1999 [24] присвячено опису процесів проектування людино-орієнтованих інтерактивних систем.

1.7.1. Концептуальне проектування

Уточнення даних — уточнення інформації, яку система обробляє як дані. Зокрема [23]:

– визначаються джерела надходження даних і сторона, яка відповідає за їх достовірність;

– уточнюються атрибути даних;

– визначаються способи матеріалізації зв'язків між об'єктами у формі відповідної організації даних (використовують ERD або класи).

Уточнення інтерфейсів з користувачами системи на ранніх стадіях ЖЦ полягає у такому:

– допомогти користувачам перевірити їх розуміння системи та отримати від них схвалення функцій ПС;

– дозволити розробнику переконатися, що запропоновані правила взаємодії користувачів та системи задовольняють обидві сторони (замовників та розробників) [23, 24].

Організація інтерфейсів ґрунтується на певних ключових елементах:

1) метафори, які використовують у домені користувача (значущі терміни, образи й поняття);

2) ментальна модель організації та подання даних, функцій і ролей;

3) правила навігації (перегляду) даних, функцій і ролей;

4) методи взаємодії, які прогнозуються такими, щоб вони були зручними користувачам [24].

Уточнення функцій оброблення даних. Для зафіксованих у моделях об'єктів уточнюється склад і зміст їх операцій та методів і уточнюється схема взаємодії об'єктів. У свою чергу взаємодія об'єктів реалізується за допомогою обміну повідомлень. У відповідь на ці повідомлення об'єкт виконує відповідні операції і змінює свій стан.

Уточнення нефункціональних вимог. Нефункціональні вимоги відображають обмеження, які накладаються організацією або середовищем використання системи. Бажано створювати для них конкретні специфікації [13].

1.7.2. Архітектурне проектування

Архітектурне проектування полягає у визначенні головних структурних особливостей ПС [13; 18; 20; 23]: складу компонентів, способів їх композиції та обмеження на зв'язки між ними.

Сучасна ПС — це складна композиція різноманітних функцій, кожній з яких відповідає програмний модуль. Водночас є тисячі готових програмних модулів, які можна включати в ПС.

Порівневу архітектуру напрацювань в інженерії розроблення ПС показано на рис. 1.4.

4	Прикладні системи
3	Компоненти бізнес-застосувань
2	Загальносистемні компоненти (інтерфейс з універсальними системами)
1	Системні компоненти (інтерфейс з обладнанням)

Рис. 1.4. Порівнева архітектура ПС

До першого рівня належать системні компоненти, які взаємодіють з периферійними пристроями (принтери, клавіатура, сканери, маніпулятори). Вони використовуються для побудови операційних систем і не потрапляють у поле зору розробників прикладних застосувань.

Другий рівень охоплює загальносистемні компоненти. Вони забезпечують взаємодію з універсальними сервісними системами, такими, як операційні системи, системи баз даних і знань, системи керування мережами тощо.

До третього рівня належать специфічні для проблемної галузі компоненти, які можуть бути використані для побудови бізнес-застосувань у межах заданої прикладної галузі.

Четвертий рівень містить прикладні ПС, побудовані для вирішення конкретних завдань окремих груп споживачів інформації, для яких і створюються компоненти нижніх рівнів.

Основні принципи декомпозиції системи на компоненти [13; 23]:

1) для компонентів має бути чітко визначена мета;

2) для компонентів мають бути чітко визначені всі входи і виходи;

3) компоненти потрібно організовувати у вигляді ієрархії, де кожний рівень ієрархії відповідає рівню абстракції системи, що дозволяє приховувати деякі деталі, які відпрацьовуються на наступних рівнях;

4) робота над компонентами ведеться окремими розробниками з використанням CASE-засобів, що загалом істотно впливає на ефективність розроблення;

5) сукупності об'єктів, поєднаних загальними зв'язками і взаємодією, доцільно об'єднати в підсистеми, причому:

- кожна створювана підсистема повинна бути асоційована з певним продуктом інженерії вимог (актор, сценарій та ін.);
- доцільно часто змінювані функції, а також необов'язкові функції виділяти у вигляді підсистем;
- інтерфейс підсистеми тим більш прозорий, чим менше вона має зв'язків з іншими підсистемами.

У різних джерелах подаються різні визначення терміна «архітектура ПС». Сучасний погляд на архітектуру полягає у визначенні її як системи конструкторських рішень, які в процесі подальшого розроблення ПС змінюються найменше. При цьому наголошується, що одна і та сама архітектура може бути описана різними способами, кожен з яких виражає бачення проектованої системи певною групою учасників розроблення — замовників, архітекторів, менеджерів тощо. Очевидно, що ці описи відображають пріоритети кожної з груп, що неминуче спричиняє конфлікти інтересів. Роль архітектора у цьому випадку — досягнути таких компромісів, щоб рівень задоволення кожної вимоги був максимальним. Таке визначення добре відображає роль архітектури для подальшого процесу створення ПС і є абстрактним поданням системи, але в цьому разі неможливо виміряти якісні характеристики конструкторських рішень.

Утім вважається, що архітектура визначає зовнішнє зображення елементів системи, їх властивостей та зв'язків між ними. Архітектура системи може бути виражена по-різному залежно від інтересів групи зацікавлених осіб через різні моделі архітектури.

Робиться також акцент на незмінності певних елементів розроблення, тобто на рішення, що відображають базову концепцію роботи майбутньої системи у предметній галузі.

У такому сенсі йдеться вже про певні елементи та зв'язки між ними. Це дає змогу формувати характеристики якості архітектури на основі відповідних характеристик якості її елементів та зв'язків між ними. Таке розуміння архітектури ПС відображає моделювання системи на рівні структурної чи функціональної схеми.

Е. Брауде у праці [13] розглядає архітектуру як проект системи на найвищому рівні абстрагування. Усі подальші кроки, на його думку, — це детальне проектування. Але і тут автор робить наголос на незмінності прийнятих рішень як основи для всіх наступних етапів створення ПС. Це визначення теж є проблемним з погляду створення для архітектури моделі якості, оскільки є дуже загальним через високий рівень абстрагування.

На підставі цього аналізу можна констатувати, що визначення не заперечують один одного, а доповнюють. Тобто архітектура ПС є репрезентацією ПС на певному рівні абстрагування з визначенням основних елементів цієї системи та зв'язків між ними. Архітектуру кожної системи можна розглядати з різних точок зору з використанням сукупності відповідних моделей.

Визначення архітектури ПС, яке акумулює наведені вище особливості, наведені у стандарті ISO 42010 *Systems and software engineering — Architecture description*, який регламентує методику опису архітектури. Згідно з цим стандартом архітектура ПС — це фундаментальні концепції властивостей системи, втілені в елементах, зв'язках і принципах дизайну та розвитку. У такому визначенні не вказується перелік концепцій, а тому можна передбачати врахування якості під час проектування архітектури.

Стандарт ISO 42010 містить інформацію про загальні принципи опису архітектури ПС, а тому його слід застосовувати у процесі проектування ПС на відповідному етапі ЖЦ. Він передбачає встановлення переліку зацікавлених сторін (*stakeholders*) під час проектування ПС. Як правило, кожна із зацікавлених сторін має свій набір інтересів (*concerns*) стосовно системи, і ці інтереси у процесі проектування потрібно максимально задовольнити. Для задоволення кожного інтересу мають бути створені описи (*views*) системи. Кожен з описів відображає систему з певної точки зору, а набір описів утворює повний опис системи. Кожен опис здійснюється відповідно до методу опису (*viewpoint*). Методи опису задають мови опису, нотації та способи моделювання чи аналізу моделей кожного опису. Мови та прийоми застосовують для отримання результатів стосовно конкретних інтересів і зацікавлених сторін. Таким чином, кожен опис є набором вимог різного роду.

Наприклад, для розробників інтерес становлять описи процесів, які моделює і реалізує система. Для спеціалістів з управління проектами цінність становлять описи системи з точки зору її логістики.

У загальному випадку опис архітектури може бути поданий у вигляді схеми, яка містить згадані вище сутності (рис. 1.5). Ця схема є спрощеним варіантом аналогічної схеми зі стандарту ISO 42010. Функціональність системи реалізується у вигляді її місій (*missions*). Архітектура системи будується на основі її обґрунтування (*rationale*). Архітектурні моделі (*models*) є основою для описів архітектури.

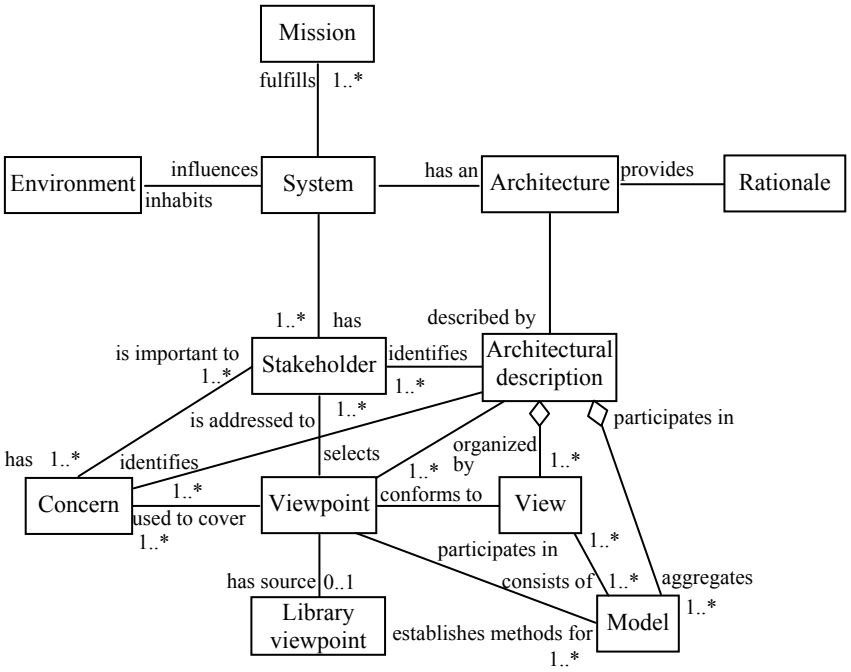

Рис. 1.5. Концептуальна модель опису архітектури ПС відповідно до стандарту ISO 42010

Стандарт ISO 42010 є основою для документування проектів архітектур і не містить установленого переліку концепцій, які мають бути описані у цьому проекті. Однією з концепцій проектування і опису архітектури є її якість. Але, як випливає зі стандарту ISO 42010, якість архітектури, а також якість її опису є нестандартизованими. Унаслідок цього автори публікацій з проектування архітектури ПС використовують різні набори характеристик якості архітектури.

Тому дослідження з уніфікації показників якості архітектури ПС і побудови на їх основі моделі якості є актуальними. Поняття моделі для опису архітектури, використане у стандарті ISO 42010, можна трактувати як абстрактне подання системи без додаткових пояснень деталей цієї системи. Відповідно кожна модель має певний набір характеристик якості, які найкраще реалізуються в її середовищі. Набір цих характеристик може бути виражений у термінах стандарту ISO/IEC 25010 (див. п. 1.12.3).

Визначення. *Архітектура ПС розуміється як набір компонентів, які реалізують логіку роботи ПС, а також зв'язків між цими компонентами, що забезпечують їх взаємодію та визначають конфігурацію компонентів ПС.*

Архітектура ПС забезпечує абстрактну модель високого рівня для подання структури і ключових властивостей ПС і створює передумови забезпечення якості ПС.

1.7.3. Технічне проектування

Технічне проектування полягає у відображенні вимог середовища функціонування та визначенні всіх конструкцій у вигляді композицій компонентів. На цьому етапі проект налаштовується відповідно до технічних особливостей платформи реалізації: СКБД, ОС, організації комунікацій, наявності фактора реального часу і швидкості реагування системи на зовнішні стимули [18; 19].

Об'єкти моделі аналізу вимог узгоджуються з урахуванням перерахованих вище особливостей, формалізуються всі стимули, які посилають або отримують об'єкти, а також усі операції, які є відповіддю на зазначені стимули.

Третій крок технічного проектування — це визначення рівнів властивостей, які належать до показників якості (надійність, безпечність, тестованість, переносимість і т. ін.) [17].

1.7.4. Трансформація проекту в програмну систему

Цей процес має кілька назв: *конструювання, кодування, програмування, реалізація.*

Кожен з них відображає певний аспект процесу трансформації. Дійсно, необхідно виконати конструювання, визначивши модулі, запрограмувати модулі обраною мовою програмування, для чого

перевести нотацію проекту в коди, після чого провести тестування й верифікацію отриманого коду ПС. Тому в цілому необхідно перетворити проект у систему, документуючи достатнім чином всі етапи трансформації, тобто реалізувати ПС [11; 13; 18].

Реалізація є одним з визначальних етапів ЖЦ розроблення ПС. Після реалізації ПС проходить тестування й оцінювання.

Міркування та рекомендації щодо реалізації ПС опубліковано в стандарті IEEE/EIA Std. 12207.2:1997 [25]. У процесі реалізації для оцінювання параметрів системи необхідно виконати відповідні вимірювання. Для цього, з-поміж інших, послуговуються стандартами [26; 27]. Спеціалізовані базові метрики розміру (обсягу) ПЗ можна знайти в стандарті [26].

Конче необхідно для реалізації ПЗ застосовувати CASE-засоби.

Відповідний процес регламентується у стандартах: ISO/IEC 14102:1995 і ISO/IEC 14471:1999 [28; 29].

У цих стандартах розглянуто питання оцінювання, відбору та адаптації CASE-інструментів для розроблення ПС. Вітчизняний стандарт ДСТУ-3919-1999 [30] гармонізований зі стандартом [28].

1.7.5. Експлуатація та супроводження ПС

Експлуатація — це застосування замовником інстальованої ПС після її придбання з метою вирішення сукупності специфікованих завдань у межах його предметної галузі. Під час експлуатації, як правило, виникає необхідність дій з боку розробника із забезпечення ефективної роботи ПС, тобто дій з її *супроводження*.

Супроводження визначається як процес модифікації ПС або її компонентів після постачання ПС замовнику з метою виправлення помилок, підвищення продуктивності чи збільшення функціональних можливостей, а також адаптації ПС до змінених умов. Незважаючи на те, що супроводження не пов’язано зі значними змінами в архітектурі ПС, за різними оцінками його вартість становить від 50 до 90 % від вартості всього ЖЦ ПС. Рекомендації щодо придбання ПЗ надає стандарт IEEE Std. 1062:1998. IEEE *Recommended Practice for Software Acquisition* [31].

Дії із супроводження ПС регламентовані стандартом IEEE Std. 1219:1998 [32]. Згодом був затверджений базовий стандарт із супроводження ISO/IEC 14764:1999 *Information technologies — Software maintenance* [33].

Вітчизняний стандарт ДСТУ ISO/IEC 14764:2002 [34] гармонізований зі стандартом [33]. Основні рубрики стандарту [34]: вступ, мета і сфера застосування, нормативні посилання і визначення понять, засоби супроводження ПЗ та його документування, передавання супроводження, стратегія супроводження, планування процесів супроводження (реалізування процесу, аналізування та реалізація модифікувань, переглядання, перенесення і вилучання ПЗ).

Стандарт із супроводження [34] має посилання на стандарти [35; 36], у яких визначається термінологія з розроблення систем і словник з управління якістю та забезпечення якості ПЗ. Важливу роль у супроводженні відіграє документація на ПП.

Принципи створення документації мають відповідати стандарту ISO/IEC 15910:1999 [37] щодо процесу підготовки документації користувача, стандарту ISO/IEC 9294:2005 *Guidelines for the management of software documentation* [38], а також стандарту про документацію комп'ютерних прикладних систем ISO/IEC 6592:2000 *Guidelines for the documentation of computer-based application systems* [39].

Продуктивність ПЗ під час його експлуатації рекомендується вимірювати з урахуванням норми стандарту ISO/IEC 14756:1999 [40], який визначає метрики ефективності в часі. Якщо користувач потребує поліпшення ефективності, розробник має виконати відповідні модифікації ПЗ.

Для відстеження частин проекту визначають їх межі. Такі частини проекту називаються елементами конфігурації (CI — *Configuration Items*). Ці елементи мають відстежуватися системою керування конфігураціями з метою організації чітких процедур доступу до елементів для їх модифікації, додавання нових елементів, ведення документації та авторизації доступу.

Керування конфігураціями треба починати одразу після початку створення проекту. Однак особливої ваги керування конфігураціями набуває в процесах реалізації і супроводження, оскільки багато елементів конфігурації (модулів, програм, інших компонентів) можуть потребувати різних модифікацій.

IEEE розробив стандарт із планування керування конфігураціями, який має назву IEEE Std. 828:1998 IEEE *Standard for Software Configuration Management Plans* [41]. Він має такий зміст:

1. *Вступ.*

2. *Керування конфігураціями.*

2.1. Організація.
2.2. Відповідальність за керування конфігураціями.
2.3. Політики, директиви, процедури.
3. *Види діяльності.*
3.1. Визначення конфігурації.
– визначення та найменування елементів;
– отримання елементів конфігурації.
3.2. Контроль конфігурації:
– запит на зміни;
– оцінка змін (схвалення чи несхвалення);
– реалізація модифікацій.
3.3. Визначення статусу конфігурації.
3.4. Аудити конфігурації.
3.5. Контроль інтерфейсу.
3.6. Контроль постачальників.
4. *Розклад.*
5. *Ресурси.*
6. *Супроводження.*

Існує достатня кількість інструментальних засобів керування конфігураціями. Крім стандарту [41], у 1998 р. вийшов також стандарт ISO/IEC 15846 — *Software life cycle processes — Configuration Management* [42], який теж успішно й ефективно застосовується для керування конфігураціями ПЗ.

1.8. Якість програмної системи та аспекти її вимірювання

У процесі оцінювання ПС виділяють рівні оцінювання різних характеристик якості залежно від рівня цілісності ПС (та її компонентів). Чим вищий встановлений рівень цілісності, тим вищі вимоги до рівня значень характеристик якості та методів оцінювання якості. У 1998 р. був уведений в дію стандарт ISO/IEC 15026 — *System and software integrity levels* [43], який визначає рівні цілісності систем і ПЗ.

Стандарт пропонує визначати рівень оцінювання якості від *A* (вищого) до *D* (нижчого) з урахуванням ризиків у галузі: *безпеки функціонування* (*A* — загроза життю людей), *захисту інформації* (*B* — загроза втрати інформації), *економіки* (*C* — загроза фінансових збитків), *середовища функціонування* (*D* — загроза руйнування середовища експлуатації ПС).

Крім того, стандарт пропонує встановлення *пріоритетів важливості* для характеристик якості, що досягається введенням *коефіцієнтів вагомості*. Це дозволяє більше уваги приділяти характеристикам з найбільшою вагою.

Вітчизняний стандарт ДСТУ 2850–94 «Показники та методи оцінювання якості» [44] рекомендує порядкову шкалу з п'яти значень: 5 — «украй важливо» (найвище значення ваги характеристики), 4 — «дуже важливо», 3 — «важливо», 2 — «бажано», 1 — «неважливо».

Наприклад, для високоцілісних систем найважливішою характеристикою є *надійність,* для якої показник середнього часу між відмовами ПС у період експлуатації має бути не меншим ніж шість місяців. Водночас для систем з низьким рівнем цілісності найважливішою характеристикою є *функціональність*, підхарактеристика *точність*, для якої числове значення отриманих точних результатів має бути не меншим ніж 95 % від загального значення отриманих результатів.

Для оцінювання якості ПС необхідно виконувати вимірювання.

Вимірювання — це отримання об'єктивних даних про стан продуктів, процесів та ресурсів розроблення ПС. Вимірювання виконують для побудови прогнозних і оцінних моделей, які надалі застосовуються для керування проектом та вдосконалення процесів розроблення організацією-розробником [17].

Вимірювання в проекті — це багатокроковий процес:

1) *визначення мети вимірювання* — для систематизації процесів вимірювання;

2) *побудова власне процесу вимірювання* — модель вимірювання є цілеорієнтованою і ґрунтується на декомпозиції мети;

3) *вибір базових підходів до вимірювання мір ПС* — обираються так, щоб результат вимірювання робочих продуктів, процесів та ресурсів протягом ЖЦ дозволяв оцінити їх відповідність цілям проекту. Основні міри:

– розмір і складність ПС (на основі цього можна оцінити трудомісткість і вартість розроблення);

– продуктивність (роботи фахівця та колективу в цілому);

– міри вимірювання атрибутів якості.

4) *організація збирання даних для виконання вимірювань* (форми, тести, запитальники тощо).

5) *застосування моделей у процесі вимірювання.*

1.8.1. Визначення якості програмної системи

Визначення якості ПС сформульовано в ДСТУ 2844–94 «Програмні засоби ЕОМ. Забезпечення якості. Терміни та визначення» [45]:

Якість — це сукупність властивостей ПС, які забезпечують її здатність задовольняти встановлені або передбачувані потреби відповідно до призначення ПС.

Для остаточного оцінювання ПС застосовують два процеси:

1) атестацію ПС;

2) сертифікацію ПС.

Атестація проводиться організацією-розробником для перевірки відповідності ПС заданим вимогам. У процесі атестації широко використовують *тестування*.

Сертифікація проводиться третьою стороною — спеціальним *органом*, ліцензованим державою для таких дій.

Сертифікація — це оцінка відповідності властивостей ПС вимогам до неї [46]. Для проведення *сертифікаційних випробувань* використовуються *лабораторії сертифікації*, які призначаються *органом із сертифікації*, що регламентовано в стандартах [46; 47; 48].

Основною властивістю ПС є ***якість***. Визначаючи вимоги до якості, зазвичай указують *зовнішні характеристики якості*, які відображають вимоги до продукту [17; 44; 45]. Кожна характеристика якості складається з ***підхарактеристик***. Для кількісного визначення критеріїв якості специфікують зовнішні вимірні властивості (***зовнішні атрибути***) і пов'язані з ними ***метрики***, які являють собою моделі оцінювання атрибутів (рис. 1.6).

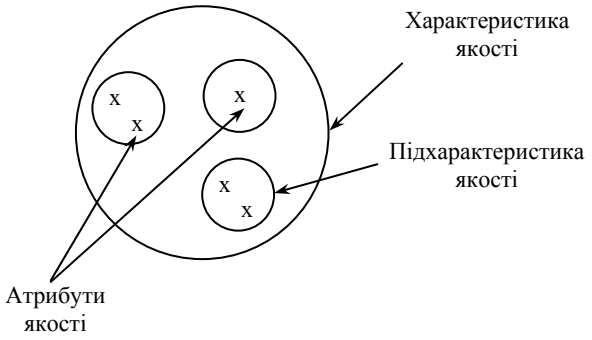

Рис. 1.6. Структура характеристик якості ПС

Зовнішні метрики — це метрики, визначення і застосування яких можливі тільки для ПЗ, яке працює на комп'ютері та перебуває на стадії тестування або функціонування [17]. Після визначення вимог до зовнішніх характеристик якості специфікуються внутрішні характеристики і підхарактеристики якості та внутрішні атрибути.

Внутрішні атрибути використовуються для планування потрібного рівня зовнішніх характеристик якості.

Визначаються також *внутрішні метрики* для вимірювання *внутрішніх характеристик якості*, тобто ***атрибути*** й ***метрики***, пов'язані з непрацюючими на комп'ютері ПП (документація на ПС, тексти програм, тестові набори даних тощо). Ці продукти утворюються на стадії розроблення, що передує тестуванню.

Зовнішні та внутрішні характеристики якості належать до власних властивостей ПС і відображають погляд замовника і розробника ПС на якість. Однак існує ще один погляд на якість — погляд кінцевого користувача, який очікує максимальної ефективності від використання ПС — підвищення продуктивності та загальної задоволеності. Такий підхід визначає ***якість у використанні*** **(*quality in use*)**, або ***експлуатаційну якість***.

Експлуатаційна якість — це сукупний ефект характеристик якості для кінцевого користувача. Її вимірюють за результатами використання ПС, але не за властивостями самої ПС.

Згідно зі стандартом ISO/IEC 9126 існують три види якості ПЗ:

1) *external quality* (зовнішня якість);

2) *internal quality* (внутрішня якість);

3) *quality in use* (якість у використанні).

На експлуатаційну якість (якість у використанні) може впливати будь-яка характеристика зовнішньої якості ПС. Для оцінювання і вимірювання якості ПС використовують моделі якості [17].

Модель якості — це множина взаємозалежних характеристик якості, що утворює базис для специфікації вимог до якості й оцінювання якості [49].

Моделі якості, а також метрики якості розглянуто в серії стандартів ISO/IEC 9126 (parts 1–4) 2001–2004 pp. [49; 50; 51; 52]:

- ISO/IEC 9126-1 — модель якості ПС;
- ISO/IEC 9126-2 — метрики зовнішньої якості ПС;
- ISO/IEC 9126-3 — метрики внутрішньої якості ПС;
- ISO/IEC 9126-4 — якість у використанні.

1.8.2. Фактори, що впливають на якість програмної системи

На якість ПС, а також на вибір моделі, впливають причини, які пов'язані із проблемною галуззю, у якій використовується ПС:

1. Прикладна галузь і категорії користувачів. Модель якості складається з множини характеристик, але кожна з них має різну вагу (значущість) залежно від прикладної галузі.

2. Мета моделювання якості. Залежно від мети модель якості різною мірою відображає погляд на якість різних категорій процесу розроблення й експлуатації ПС.

3. Стадія ЖЦ. На етапах ЖЦ погляд на якість може змінюватися. ISO/IEC 9126 рекомендує варіювати погляди на якість ПС за стадіями ЖЦ у такий спосіб:

– цільова якість (відображає потреби користувача);

– установлена якість ПС — рівень значень характеристик якості, що заявлений у специфікаціях вимог до ПС;

– прогнозована якість ПП — рівень, що передбачається як якість кінцевого продукту;

– якість продукту, що поставляється;

– експлуатаційна якість — якість, вимірювана за результатами використання ПС, але не за її властивостями.

На якість ПС також впливають такі фактори:

– розмір і складність ПС;

– методологія проектування ПС;

– наявність CASE-інструментів;

– рівень зрілості організації-розробника ПС;

– історичний досвід організації-розробника.

У процесі розроблення можуть виникати конфлікти вимог: елементи множин функціональних і нефункціональних вимог до ПС (у тому числі вимог до якості) можуть вступати в конфлікти, які потрібно вирішувати [11; 13; 17; 18].

1.9. Концепція підвищення якості програмної системи та ядро інженерії якості

1.9.1. Методи підвищення якості

Перший крок — це впровадження в практику організації-розробника спеціальних підтримувальних процесів ЖЦ ПС, які регламентуються в стандарті ISO/IEC 12207, а саме:

- процес гарантування якості ПЗ (SQA);
- процес верифікації;
- процес валідації;
- процеси спільних перевірок.

Процес SQA (Software Quality Assurance) — застосування організацією-розробником методів і засобів контролю якості на всіх етапах ЖЦ ПС. Цей процес передбачає впровадження стандартів якості і відповідних процедур у розроблення ПС, а також оцінювання етапу розроблення на відповідність цим стандартам і процедурам. Процес SQA детально регламентований у стандартах IEEE Std. 730:1998 — IEEE *Standard for Software Quality Assurance Plans* та IEEE Std. 730-1:1995 — *IEEE Guide for Software Quality Assurance Planning* [53; 54], а також у стандартах NASA Std. 2201:1993 — NASA *Software Assurance Standard* і NASA GB A201:1995 — NASA *Software Assurance Guidebook* [55; 56].

Відповідно до стандарту IEEE Std. 730 [53] план забезпечення гарантій якості ПС складається з таких частин: вступу (призначення, галузь застосування, посилання), управління SQA (організація, завдання, відповідальність), документації, стандартів і метрик, описів обстежень, аудиторських перевірок, випробувань (тестувань), повідомлень про проблеми; коригувальних дії, інструментів та технологій, контролю коду; носія та постачальника, збирання та ведення звітів, навчання, управління ризиком.

Верифікація — це дослідження трансформації вхідних робочих продуктів у вихідні; при цьому перевіряється, чи правильно розробляється ПП (наприклад, чи правильний код програми відповідно до вхідних специфікацій цієї програми).

Валідація — це дослідження сукупності робочих продуктів на певному етапі процесу розроблення, щоб упевнитися, чи правильно вони розроблені (чи відповідають призначенню та специфікованим вихідним вимогам до ПП).

Процеси верифікації і валідації (*Verification and Validation* — V&V) регламентуються стандартами IEEE Std. 1059:1993 — IEEE *Guide for Software Verification and Validation Plans* та IEEE Std. 1012:1998 — IEEE *Standard for Software Verification and Validation* [57; 58].

У процесах V&V широко застосовується *тестування*.

Структура та зміст плану верифікації та валідації на всіх етапах ЖЦ ПЗ містяться у стандарті IEEE Std. 1012 [58]:

1. *Вступ, посилання, визначення.*
2. *Організація роботи за V&V.*
2.1. Організація та графік робіт.
2.2. Схема призначення рівнів цілісності.
2.3. Розподіл ресурсів і відповідальності.
2.4. Інструменти, методології та методи.
3. *Процеси та дії за V&V.*
3.1. Процес: *Управління.*
Дія: *Управління* V&V.
3.2. Процес: *Придбання.*
Дія: V&V підтримання придбання.
3.3. Процес: *Постачання.*
Дія: V&V *Планування*.
3.4. Процес: *Розроблення.*
Дії: V&V концепції, вимог, проекту, реалізації, випробувань, уведення в експлуатацію.
3.5. Процес: *Експлуатація.*
Дія: V&V експлуатації.
3.6. Процес: *Супроводження.*
Дія: V&V супроводження.
4. *Вимоги до звітних документів V&V.*
5. *Вимоги до адміністративних процедур.*
6. *Вимоги до документування V&V.*

Процеси спільних колективних перевірок — це перевірка *робочих продуктів ПС* (здебільшого описів вимог, описів проекту, текстів програм системи, експлуатаційної та іншої документації тощо) *групою осіб* та прийняття узгодженого спільного рішення.

Проведення формальних спільних перевірок регламентується стандартом IEEE Std. 1028:1997 — IEEE *Standard for Software Reviews* [59]. У цьому стандарті за єдиною схемою описано п’ять видів перевірок: технічний огляд, управлінський огляд, формальна інспекція, наскрізний перегляд, аудиторська перевірка.

Технічний огляд — систематичне оцінювання придатності продукту для використання за призначенням та ідентифікація розбіжностей зі специфікаціями і рекомендаціями стандартів та керівництв. Технічний огляд виконує група проекту виготовлення ПП з необов’язковим запрошенням замовників і користувачів.

Управлінський огляд — систематичне оцінювання стану процесів ЖЦ з метою контролю просування проекту, визначення планів і графіків, розподілу ресурсів і додержання стандартів. Управлінський огляд виконує група проекту, керівництво проекту і необов'язково замовники та користувачі.

Формальна інспекція — систематична перевірка ПП з метою виявлення та ідентифікації проблем з урахуванням дефектів та відхилення від специфікацій і стандартів. Формальну інспекцію виконує група спеціально навчених осіб (3–6 осіб), які володіють технікою інспекції. У групу не можуть входити безпосередні розробники ПП, окрім автора проекту, який дає пояснення щодо ПП та усуває дефекти. Формальна інспекція вказується в плані проекту на розроблення ПП і в плані V&V. Крім того, інспекції виконуються за ініціативи керівництва проекту або групи SQA.

Наскрізний перегляд — запланований систематичний контроль перебігу розроблення ПП і оцінювання його стану. Мета — пошук дефектів, покращання продукту, оцінювання відповідності специфікаціям і стандартам. Наскрізний перегляд виконує група контролерів (3–6 осіб), які володіють технікою перегляду. Додаткові перегляди виконують за запитами керівництва проекту чи групи SQA.

Аудиторська перевірка — аналіз та *незалежна* перевірка продукту або процесу *третьою стороною* з метою оцінювання відповідності стандартам, керівництвам, планам, умовам договору, або іншим критеріям. Аудиторську перевірку виконує група зовнішніх аудиторів (1–5 осіб). Відповідальність за прийняття рішення про необхідність аудита бере на себе *ініціатор.* Він може бути керівником організації-розробника, або представником замовника чи користувачів.

Об'єктами *аудиторської перевірки* можуть бути всі види планів, тексти договорів, рекламації замовників чи користувачів, звіти різного роду, ПП на всіх стадіях ЖЦ, стандарти та інші нормативні документи.

Другий крок підвищення якості — це досягнення організацією-розробником високого рівня зрілості. Основними стандартами якості продукту є стандарти ISO 9000. Ця серія ґрунтується на методології CMM (*Capability Maturity Models* — модель зрілості можливостей). Методологія CMM та шляхи її впровадження в діяльність організації-розробника ПП детально розглянуто в підрозд. 2.5.

Стандарт ISO 9000 [60] рекомендує кожній організації-розробнику мати свою власну систему якості [17]. Сім'ю стандартів ISO 9000 розглянуто в підрозд. 1.12, 2.1 і 2.4, зокрема змістовно у пп. 2.1.1 і 2.4.1.

Система якості ПС (за стандартом ДСТУ 2844) — це сукупність організаційної структури, відповідальності, процедур, процесів і ресурсів, спрямованих на реалізацію керування якістю ПС [45].

1.9.2. Ядро професійних знань інженерії якості

Для того щоб керувати якістю продукції, яка формується на кожному етапі ЖЦ, необхідно мати знання, що становлять ядро професійних знань у сфері якості ПС (рис. 1.7).

Ядро інженерії якості формується на основі *елементів* ядра знань програмної інженерії SWEBOK (S*oftware Engineering Body of Knowledge*) та ядра знань управління проектами створення промислової продукції PMBOK (*Project Management Body of Knowledge*), як показано на рис. 1.7.

У 1990-х роках світове комп'ютерне співтовариство почало систематизацію знань у різних сферах комп'ютерних наук та інформатики з метою зафіксувати їх у вигляді *ядер знань*.

Для створення ядра знань з програмної інженерії зусиллями ACM (*Association for Computing Machinery*) та IEEE *Computer Society* був створений координаційний комітет з програмної інженерії SWECC (*SoftWare Engineering Coordinating Committee*), у який увійшли спеціалісти світового рівня в сфері розроблення ПЗ.

Визначення необхідного ядра знань, рекомендованих прийомів діяльності та процесів програмної інженерії завершилось у 2001 р. першим виданням IEEE Computer Society SWEBOK [61]. Згодом воно було доопрацьоване і вийшло в остаточній редакції підкомітету *Software and System Engineering* ISO/IEC у 2004 р. під назвою *Software Engineering Body of Knowledge* (SWEBOK)/ISO/IEC JTC1/SC7 N2517 Software & System Engineering Secretariat [62].

Керівництво із SWEBOK [63] було видано згодом у формі стандарту ISO/IEC 19759. *Software Engineering — Guide to the Software Engineering Body of Knowledge* — SWEBOK [64].

Сфери знань SWEBOK і підходи до вивчення цієї дисципліни викладено в праці [65].

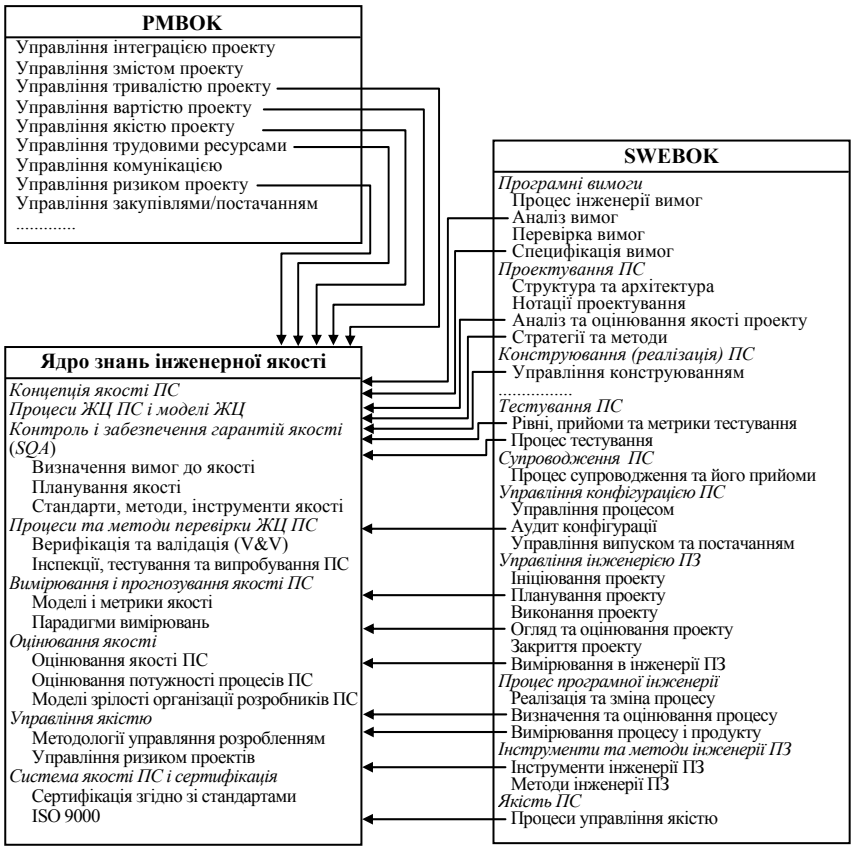

Рис. 1.7. Зв'язок ядра знань інженерії якості
з елементами SWEBOK і PMBOK

Керівництво з ядра знань у сфері управління проектами розроблено організацією PMI (*Project Management Institute*). Третя версія PMBOK містить опис *процесів* і ключових сфер знань з управління проектами в промисловості та створення ПЗ [66; 67].

Установлено рубрики (розділи) як для SWEBOK, так і для PMBOK, а ядро знань інженерії якості синтезовано на їх базі. Таким чином, у ядро знань інженерії якості ввійшли такі розділи:

- концепції якості ПС;
- процеси ЖЦ ПС і моделі ЖЦ;
- контроль і забезпечення гарантій якості (*SQA*);

– процеси та методи перевірки в ЖЦ ПС;
– вимірювання і прогнозування якості ПС;
– оцінювання якості ПС;
– управління якістю;
– система якості ПС і сертифікація.

Концепції якості ПС, процеси ЖЦ ПС і верифікація та валідація розглянуто в підрозд. 1.8 і 1.9, етап тестування — у підрозд. 1.13.

У SQA ввійшли:
– визначення вимог до якості;
– планування якості;
– стандарти, методи й інструменти контролю якості.

Процеси та методи перевірки в ЖЦ ПС містять:
– процеси V&V;
– інспекції, тестування і випробування ПС.

Вимірювання і прогнозування якості:
– моделі й метрики якості;
– вимірювання продуктів ПС.

Оцінювання якості:
– оцінювання якості ПП;
– оцінювання потужності процесів ЖЦ ПС;
– оцінювання зрілості організації розробника в цілому.

Управління якістю ПС:
– управління ризиками під час розроблення ПС;
– удосконалення процесів ЖЦ ПС.

Система якості ПС і сертифікація пов’язані з контролем якості та процесом сертифікації ПС [17] (розглянуто у підрозд. 1.12 і 2.4).

В Україні організацією, що відповідає за стандартизацію та сертифікацію продукції, є УкрСЕПРО, а органом із сертифікації ПЗ — УкрСертСофт.

1.10. Інженерія якості. Метрики та міри якості програмної системи

1.10.1. Концепція інженерії якості програмної системи

Під *інженерією якості ПС* розуміють керований процес інкорпорації в ПС на кожній стадії ЖЦ певних властивостей, які називають *характеристиками якості*. Наявність і рівень досягнення цих властивостей вимірюють, прогнозують, оцінюють та регулюють за допомогою сукупності стандартів, процесів, методів і засобів.

Процес інженерії якості має гарантувати таке:

1) вимоги до якості враховують погляд замовників, користувачів і розробників ПС;

2) усі вимоги до якості визначаються таким чином, щоб можна було виміряти кількісно або верифікувати;

3) процеси ЖЦ містять процедури, орієнтовані на виявлення вимог до якості й аналіз досягнення властивостей якості;

4) стандарти у сфері якості визначені й дотримуються;

5) розробник ПС розуміє цілі якості і вбудовує в ПП властивості, які забезпечують зовнішню, внутрішню та експлуатаційну якість;

6) група якості на підприємстві проводить вимірювання й оцінювання якості ПП;

7) результати вимірювання використовуються для управління програмним проектом;

8) виконуються процеси V&V і тестування ПС з метою перевірки відповідності ПС вимогам до якості.

Стандарт ISO/IEC 12207 [1] не виділяє інженерію якості у вигляді окремого процесу ЖЦ, однак з наявних у ньому процесів безпосередньо стосуються інженерії якості такі процеси:

- управління проектом;
- управління якістю;
- управління ризиками;
- гарантування якості (SQA);
- процеси V&V;
- аналіз вимог до ПС;
- вимірювання;
- удосконалення процесів ЖЦ.

1.10.2. Метрики якості

Атрибути ПС, які характеризують якість, вимірюються за допомогою ***метрик якості.***

Метрика — це комбінація конкретного *методу вимірювання* атрибута сутності та *шкали вимірювання*.

Метод вимірювання визначає спосіб отримання значень атрибута якості певної сутності. *Шкала* використовується для структурування отриманих значень. ***Метрика*** визначає *міру* атрибута, тобто змінну, якій присвоюється значення в результаті вимірювання.

Наприклад, сутність — це звіт про виявлення дефектів у ПС, атрибут — список дефектів, метод вимірювання — підрахунок кількості дефектів у списку, шкала — цілі числа більші за 0, міра атрибута — загальна кількість дефектів, ім'я метрики (таке саме як ім'я міри) — загальна кількість дефектів.

Міра атрибута є безпосередньою, якщо вона не залежить від мір інших атрибутів, або побічною, якщо утворюється на основі мір інших атрибутів. Побудову моделі якості та застосування метрик розглянуто в стандарті ISO/IEC 9126 (рис. 1.8).

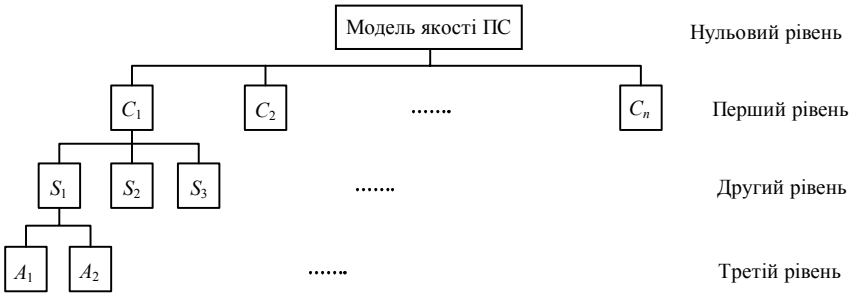

Рис. 1.8. Модель якості відповідно до стандарту ISO/IEC 9126-1

В ієрархічній моделі якості за рис. 1.8: C_i — характеристики, S_i — підхарактеристики, A_i — атрибути якості ПС.

Оскільки метрика якості ПС — це модель вимірювання атрибута, то метрики є індикатором одного або декількох атрибутів, що показано на рис. 1.9. Метрика називається *базовою*, якщо в її основі лежить елементарний метод вимірювання атрибута [17].

Стандарт ISO/IEC 9126-2 рекомендує застосовувати п'ять видів шкал вимірювання значень:

1) *номінальну шкалу* (класифікаційну шкалу);

2) *порядкову шкалу* — дозволяє впорядковувати властивості за зростанням–зменшенням шляхом порівняння з базовим значенням;

3) *інтервальну шкалу* — відзначає дистанцію між властивостями об'єкта (використовується в арифметичних операціях та операціях порівняння);

4) *відносну шкалу* — значення розрізняються відносно обраної одиниці вимірювання (вважається найбільш пріоритетною шкалою);

5) *абсолютну шкалу* — спеціальний випадок відносної шкали, де вказується абсолютне значення величини.

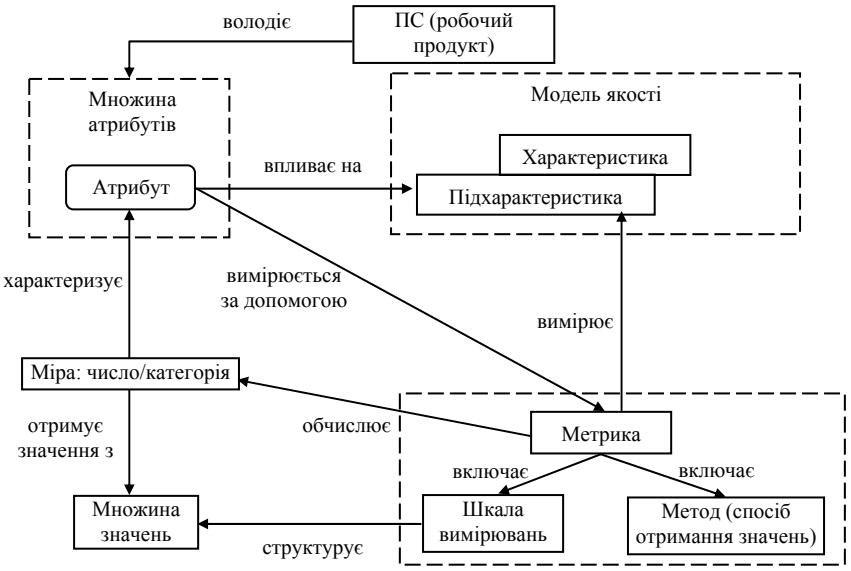

Рис. 1.9. Метрика в системі вимірювання якості

Обчислені значення метрики самі по собі не містять інформації про рівень задоволення вимог до якості. Для цих цілей шкалу, як правило, ділять на ділянки (ранги), які відповідають різним ступеням задоволеності вимог. Наприклад, можна поділити шкалу на чотири категорії, як показано на рис. 1.10.

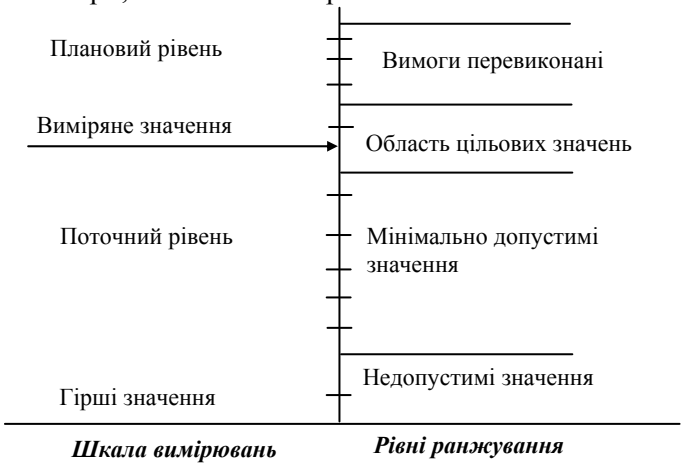

Рис. 1.10. Рівні ранжування метрик

1.10.3. Класифікація мір та метрик якості

Стандарт ISO/IEC 9126-2 визначає такі типи мір:

1) *міру розміру* — розмір ПС в одиницях вимірювання (функціональний розмір, кількість функцій або модулів, кількість рядків у програмі, ємність дискової пам'яті та ін.);

2) *міру часу* — періоди реального часу (у секундах, хвилинах, годинах процесорного часу, час функціонування системи);

3) *міру зусиль* — корисний час, пов'язаний з певним завданням (продуктивність праці, трудомісткість тощо);

4) *міру інтервалів між подіями* (наприклад, час між відмовами);

5) *рахункові міри* — статичні лічильники для обліку певних елементів у робочих продуктах ПС (текстів програм та документації), або динамічні лічильники (вимірюють кількість помилок, кількість відмов, відповідей системи на запити тощо).

Метрики зазвичай класифікуються таким чином:

1. *Об'єктивна/суб'єктивна*. Об'єктивна включає підрахунок елементів, які можуть бути незалежно перевірені (кількість операторів коду, кількість помилок); суб'єктивна ґрунтується на індивідуальному розумінні певних властивостей (рівень складності проблем, модулів).

2. *Примітивні/обчислювальні.* Примітивні — це базові метрики, які безпосередньо спостерігаються (розмір програми, кількість дефектів); обчислювальні — обчислюються на основі примітивних метрик (трудомісткість).

3. *Динамічні/статичні.* Динамічним метрикам властивий компонент часу (кількість помилок за місяць); статичні метрики не залежать від часу (загальна кількість виявлених дефектів).

4. *Передбачувані (прогнозні)/пояснювальні.* Значення прогнозних метрик може бути отримане заздалегідь (прогнозована кількість відмов); значення пояснювальних метрик — постфактум (реальна інтенсивність відмов).

Стосовно об'єкта вимірювання міри та метрики поділяються на *зовнішні*, *внутрішні* та *метрики використання ПС*.

Зовнішні метрики використовують міри ПП, який працює на комп'ютері. Ці міри отримують у результаті вимірювання поведінки ПС у ході тестування та функціонування. Відповідні метрики розробляються і використовуються для демонстрації якості ПП та підтвердження того, що ПП задовольняє зовнішні вимоги до якості [17; 50].

Розроблення зовнішніх метрик ґрунтується на виконанні вимірювань:

– поведінки ПС під час тестування і функціонування;
– поведінки користувача (сценарії використання ПС).

Приклади зовнішніх метрик для характеристики *надійність*: кількість усунутих дефектів, середній час між відмовами.

Внутрішні метрики дають змогу оцінити якість ПС безпосередньо за її властивостями (ємність коду, число цикломатичності тощо). Внутрішні метрики відображають якість проміжного і кінцевого ПП за тими характеристиками, які визначені в моделі. Ці метрики використовують для верифікації того, що проміжний та кінцевий продукти задовольняють вимоги до внутрішньої якості. Розроблення внутрішніх метрик ґрунтується на виконанні вимірювань статичних атрибутів, які визначають й оцінюють за текстом вихідного коду, а також за графічним зображенням потоків керування, чи документами на ПС [51]. Приклади для характеристики *надійність*: кількість помилок, знайдених під час інспекції коду, кількість усунутих дефектів під час інспекції.

Метрики якості у використанні (експлуатаційні метрики). Вони вимірюють ступінь, у якому ПП задовольняє потреби користувачів в ефективності, продуктивному й безпечному вирішенні завдань. Ці метрики оцінюють видимі результати експлуатації ПС [52], наприклад, повноту досягнення цілей користувачів, точність результатів, продуктивність праці, думку користувачів.

Отже, інженерія якості ґрунтується на стандартах: модель якості — ISO/IEC 9126-1 [49]; зовнішні метрики — ISO/IEC 9126-2 [50]; внутрішні метрики — ISO/IEC 9126-3 [51]; експлуатаційні метрики — ISO/IEC 9126-4 [52].

1.11. Модель якості програмної системи

Модель якості ПС являє собою множину взаємозалежних характеристик, які утворюють базис для специфікації вимог до якості та оцінювання якості ПС. Еталоном є стандарт ISO/IEC 9126 (1–3).

1.11.1. Модель зовнішньої і внутрішньої якості

Зовнішні і внутрішні метрики використовують для вимірювання атрибутів підхарактеристик моделі [49]. Модель зовнішньої і внутрішньої якості показано на рис. 1.11.

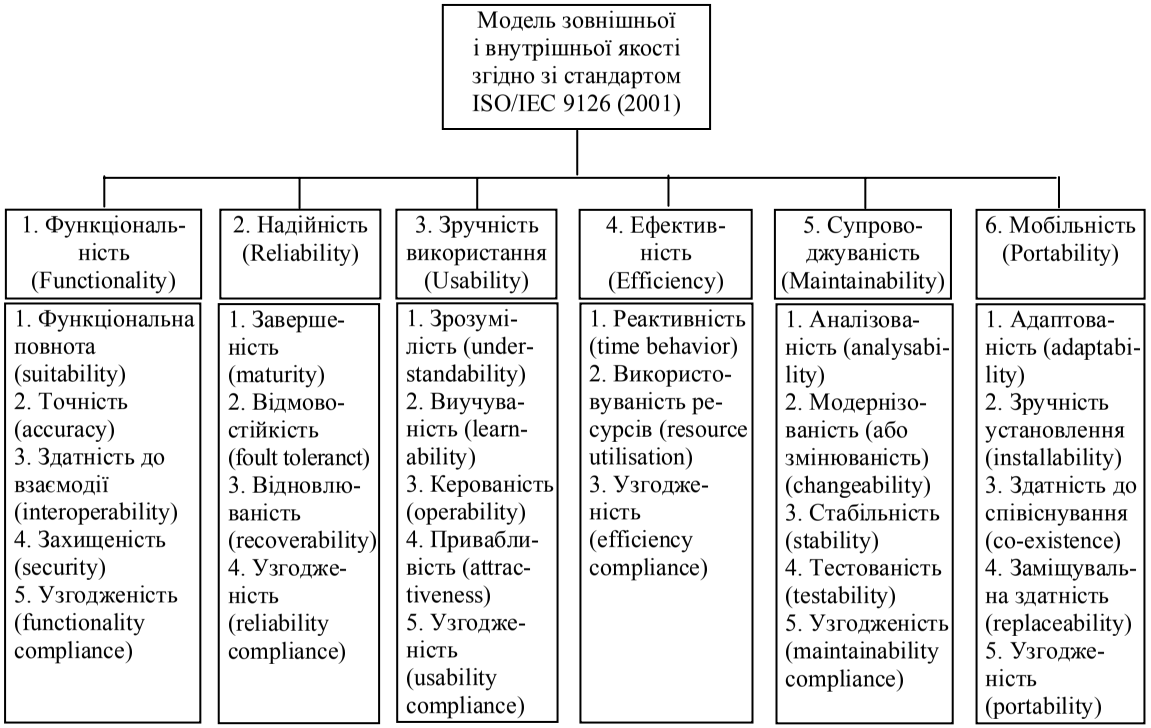

Рис. 1.11. Модель якості ПС