Стандартизація та сертифікація інформаційних управляючих систем

доцент Райчев Ігор Едуардович

Мета дисципліни – розкриття наукових концепцій, понять і методів забезпечення якості програмних систем (ПС) шляхом впровадження в процеси життєвого циклу ПС вимог і рекомендацій національних і міжнародних стандартів в області інформаційних технологій та програмних засобів.

Основы инженерии качества программных систем / Ф.И.Андон, Г.И.Коваль., Т.М. Коротун, Е.М.Лаврищева, В.Ю.Суслов. –К.: Академпериодика, 2007. – 672 с.

Бабенко Л.П., Лавріщева К.М. Основи програмної інженерії. Навч. посіб. –К.: Т-во “Знання”, КОО, 2001. – 269 с.

Шлеер С., Меллор С. Объектно-ориентированный анализ: моделирование мира в состояниях : пер. с англ. –К.: Диалектика, 1993. – 240с.

Канер Сэм, Фолк Дж., Нгуен Енг Кен. Тестирование программного обеспечения : Пер. с англ. –К.: Диасофт, 2001. – 544 с.

Лавріщева К.М. Програмна інженерія. Підручн. –К.: Академперіодика, 2008. – 320 с.

Майерс Г. Искусство тестирования программ : –М.: Мир, 1982. – 176 с.

Дастин Э., Рэшка Д., Пол Дж. Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация : Пер. с англ. –М.: Изд. “Лори”, 2003. – 567с.

Брауде Э. Технология разработки программного обеспечения. – СПб.: Питер, 2004. – 655 с.

Соммервилл И. Инженерия программного обеспечения. –М.: Изд. дом Вильямс, 2002. – 624 с.

Райчев І.Е., Харченко О.Г. Концепція побудови сертифікаційної моделі якості програмних систем // Проблемы программирования. –2006. –№2-3. – С. 275–281.

Стандартизація та сертифікація інформаційних управляючих систем

Зв'язок між системами

Будь-яка **програмна система** (ПС) є компонентою деякої комп'ютерної (інформаційної) системи, яка в свою чергу є складовою деякої кінцевої системи (бізнес-системи).

Програмна система – це група інтегрованих програмних засобів, створених для автоматизації вирішення множини задач, специфікованої в межах заданого домена (проблемної галузі).

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

Бізнес-система (БС) – це *ІС*, яка розвинута сукупністю бізнес-застосувань. БС використовуються на підприємствах, які виробляють продукцію, в торгівельних фірмах, магазинах, банках тощо.

ISO/IEC 12207:1995. Information technologies – Software life cycle processes.

ДСТУ 3918-1999. (відповідає ISO/IEC 12207:1995). Інформаційні технології. Процеси життєвого циклу програмного забезпечення. Держстандарт України.

ISO/IEC 12207:1995 / Amd.1:2002. IT – Software life cycle processes.
ISO/IEC 12207:1995 / Amd.2:2004. IT – Software life cycle processes.

ISO/IEC 15271:1998. IT – Guide for ISO/IEC 12207 Software Life Cycle Processes.

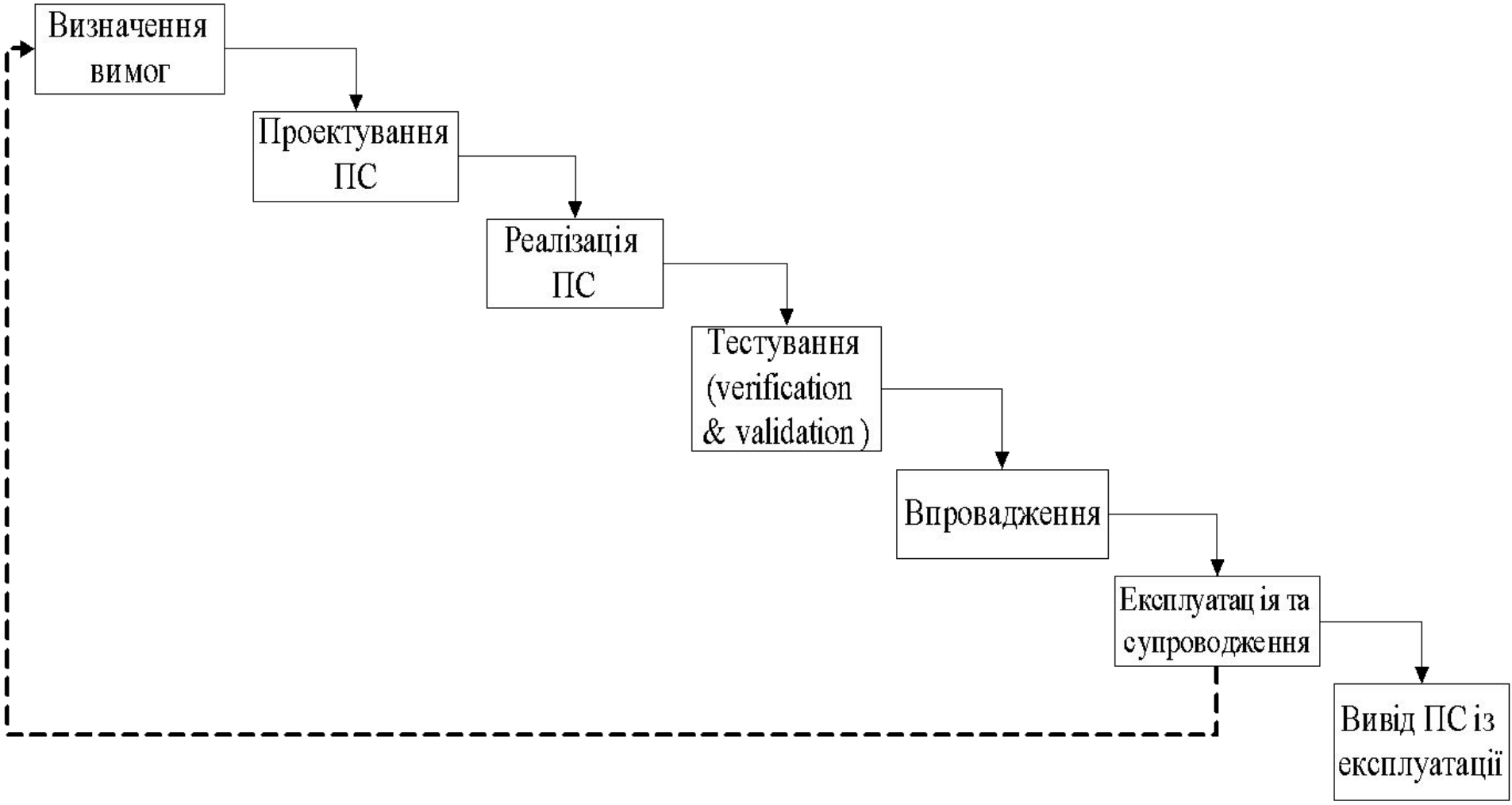
ISO/IEC 15288:2002. Systems engineering – System life cycle processes.
ДСТУ ISO/IEC 15288:2005. Інженерія систем – Процеси життєвого циклу системи.
ISO/IEC 15288:2008. Systems and software engineering – System life cycle processes.

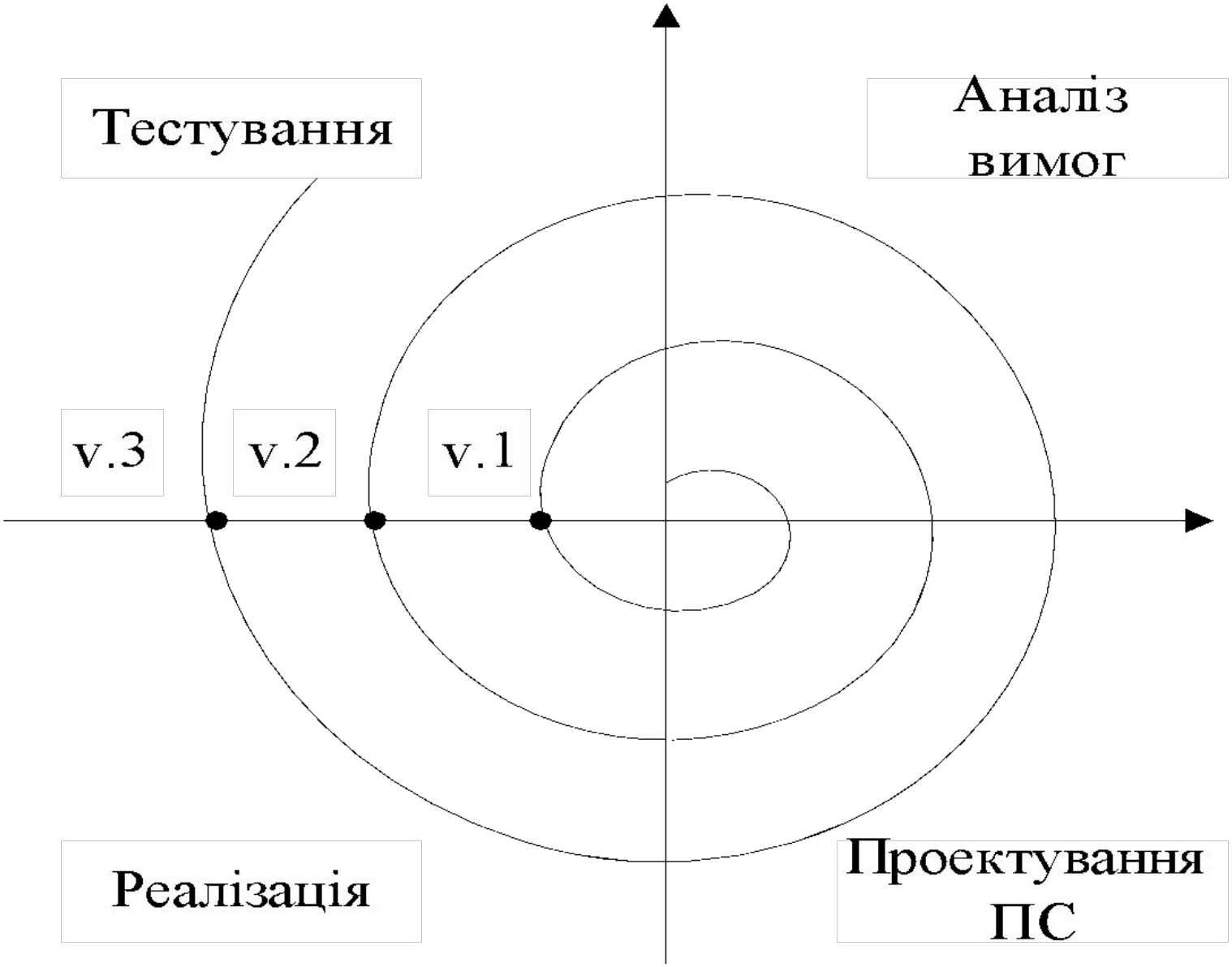
IEEE/EIA Std. 12207.0:1996. Software life cycle processes.
IEEE/EIA Std. 12207.1:1997. Software life cycle processes – Life cycle data.

IEEE Std 830-1993. Recommended practice for software requirements specification.

IEEE Std. 1233:1998. IEEE Guide for Developing System Requirements Specifications.

Якість – це сукупність властивостей ПС, які забезпечують її здатність задовольняти встановлені або передбачувані потреби відповідно до призначення ПС (ДСТУ 2844-1994. Програмні засоби ЕОМ. Забезпечення якості. Терміни та визначення.)




Тестування
Аналіз вимог
v.3
v.2
v.1
Реалізація
Проектування ПС

ЖЦ ПЗ визначений стандартом ISO/IEC 12207:1995 – Information Technology – Software life cycle processes.

Процеси ЖЦ діляться на 3 групи: головні; допоміжні; організаційні.

Головні процеси: процес покупки (ініціація ЖЦ ПС і визначення організації, що ініціює розробку/покупку); процес розробки (дії організації розробника (ОР) – визначення та аналіз вимог, проектування ПС, реалізація ПС, тестування); процес постачання (передача ПС покупцеві); процес експлуатації; процес супроводження (керування модифікаціями ПС та інсталяція нових версій).

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

Організаційні процеси: менеджмент розробки; навчання персоналу; визначення обов'язків учасників процесів ЖЦ.

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


Причини провалу проекту
Неповнота вимог – 13,1%
Недостатня кількість залучення користувачів – 12,4%
Недостатня кількість ресурсів – 10,6%
Не відповідність потребам – 9,9%
Недостатня підтримка від керівництва – 9,3%
Зміна вимог специфікації – 8,7%
Недостатнє планування – 8,1%
Втрата необхідності – 7,5%

Інженерія вимог

ПС вводиться в реальний світ для того, щоб впливати на процеси реального життя. Ті частини реального світу, які впливають на систему або піддаються її впливу, складають ***домен прикладної галузі (області) або домен використання.***

Вимоги до ПС стосуються тих властивостей, які повинні бути у системи, якщо вона адекватно виконує свої функції.

Діючі персони в процесі формулювання вимог: носії інтересів замовника (часто декілька профгруп); оператори, які здійснюють обслуговування ПС; розробник ПС.

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 збору вимог – неформалізований їхній опис (або технічне завдання – ТЗ) – основа контракту на розробку між замовником і виконавцем розробки системи (*ГОСТ 34.602-89.* Информационная технология – Комплекс стандартов на автоматизированые системы. Техническое задание на создание автоматизированой системы).

Базовим стандартом, який визначає порядок і умови збору та постановки вимог до ПС, є міжнародний стандарт **IEEE Std 830 – Recommended Practice for Software Requirements Specifications**. Стандарт складається з розділів:

1. Вступ: мета; область 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 requirements); D-вимоги – вимоги розробника (Developer requirements).

3. Конкретні вимоги.

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

3.2. Класи й об'єкти (відносять до функціональних вимог).

3.3. Вимоги продуктивності (відносять до нефункціональних вимог).

3.4. Обмеження на проектування. 3.5. Атрибути ПС. 3.6. Інші вимоги.

C-вимоги – неформалізовані, а D-вимоги – детальні вимоги, які поділяються на функціональні та нефункціональні. **Функціональні вимоги** відносяться до функціональних характеристик системи (можливості, які повинна забезпечувати система), а **нефункціональні вимоги** відносяться до якісних властивостей, котрі не мають прямого відношення до функціональності ПС (обмеження, пов'язані з характеристиками функціонування ПС).

D-вимоги складаються за допомогою функціональних специфікацій, що містять у собі повний список всіх властивостей і функцій, якими повинна володіти система. Ці вимоги повинні відповідати **C-вимогам**. Специфікація вимоги до ПЗ (SRS) має бути: несуперечливою; з можливістю контролю; тестованою; погодженою; повною.

Мається кілька класів *нефункціональних вимог*, які є суттєвими для більшості ПС. Це обмеження, які актуальні для більшості проблемних областей:

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 при виконанні її функцій (обмеження на ресурси пам'яті або процесору, швидкість реакції на звернення до ПС тощо).

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

Процес побудови моделі проблеми називається ***концептуальним моделюванням.*** Роль концептуальної моделі полягає в тому, щоб бути посередником між професіоналами, які знаються на різних доменах (наприклад, фахівцями з бухгалтерського обліку і програмістами).

Сукупність термінології, понять, характерних відносин, а також парадигми їхньої інтерпретації в рамках ***домену*** називається ***онтологією домену.*** Онтологія утримує користувачів у максимально можливому просторі визначених наперед можливостей, зміст яких зафіксований і зрозумілий як замовнику, так і розробнику. Серед існуючих відношень між об'єктами **статичних доменів** найбільш відомі: узагальнення; конкретизація; агрегація; асоціація.

Для **динамічних доменів** істотними поняттями є: **стан** (домену, області, об'єкта) – фіксація певних властивостей об'єктів на певний момент (чи інтервал) часу; **інтервал стабільності** – інтервал часу протягом, якого не змінюється стан; **подія** – явище, що провокує зміну станів. Серед динамічних доменів існують наступні різновиди: *інертні* – стан не змінюється до впливу зовнішніх агентів; *реактивні* – зміна стану як відповідь на певну зовнішню подію; *активні* – перехід зі стану в стан без зовнішніх стимулів.

Об'єктно-орієнтована інженерія вимог

Відомі наступні моделі, що визначають архітектуру ПС: модель функції-дані; об'єктна модель. Основні концепції ООП:

- світ складають об'єкти, які взаємодіють між собою;
- кожний об'єкт має певний набір властивостей (атрибутів);
- об'єкти вступають між собою у відношення, які можуть змінюватися у часі;
- сукупність значень атрибутів об'єкта у певний момент часу обумовлює його стан -> сукупність станів всіх об'єктів визначає стан предметної області 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 називається його ***поведінкою***;
- об'єкти можуть бути складними і взаємодіють між собою шляхом обміну повідомленнями. Об'єкти характеризуються: інкапсуляцією; успадкуванням (спадковістю); поліморфізмом.

Відомі такі об'єктно-орієнтовані підходи: **метод Шлеєр і Меллора; метод сценаріїв (метод Джекобсона); методологія UML**.

Метод інженерії вимог С.Шлеєр і С.Меллора

Продуктом аналізу домену є 3 моделі: інформаційна модель системи (онтологія домену); модель станів об’єктів, визначеної у складі інформаційної моделі; модель процесів, що забезпечують переходи об’єкту із одного стану в інший.

Інформаційна модель домену

Приклад об’єкта з атрибутами

3. Співробітник

* ім'я
* прізвище
* посада
* табельний номер
* телефон

Мається 3 види зв’язків

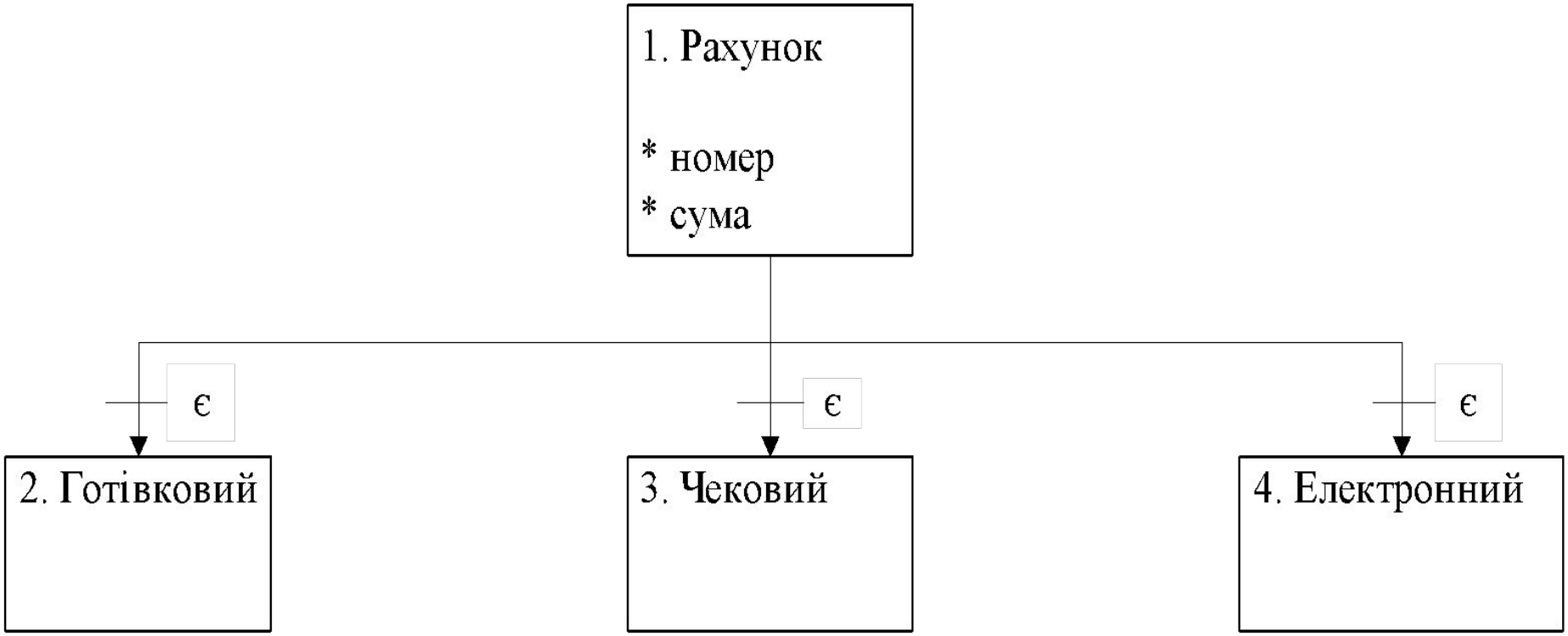
1:1 

1:n 

m:n 

Фрагмент та приклад інформаційної моделі

Приклад відношення спадкування



Для фіксації динамічних аспектів вимог визначені дві нотації: **діаграма переходів у стани (ДПС); таблиця переходів у стани (ТПС)**. Обидві нотації базуються на автоматі Мура.

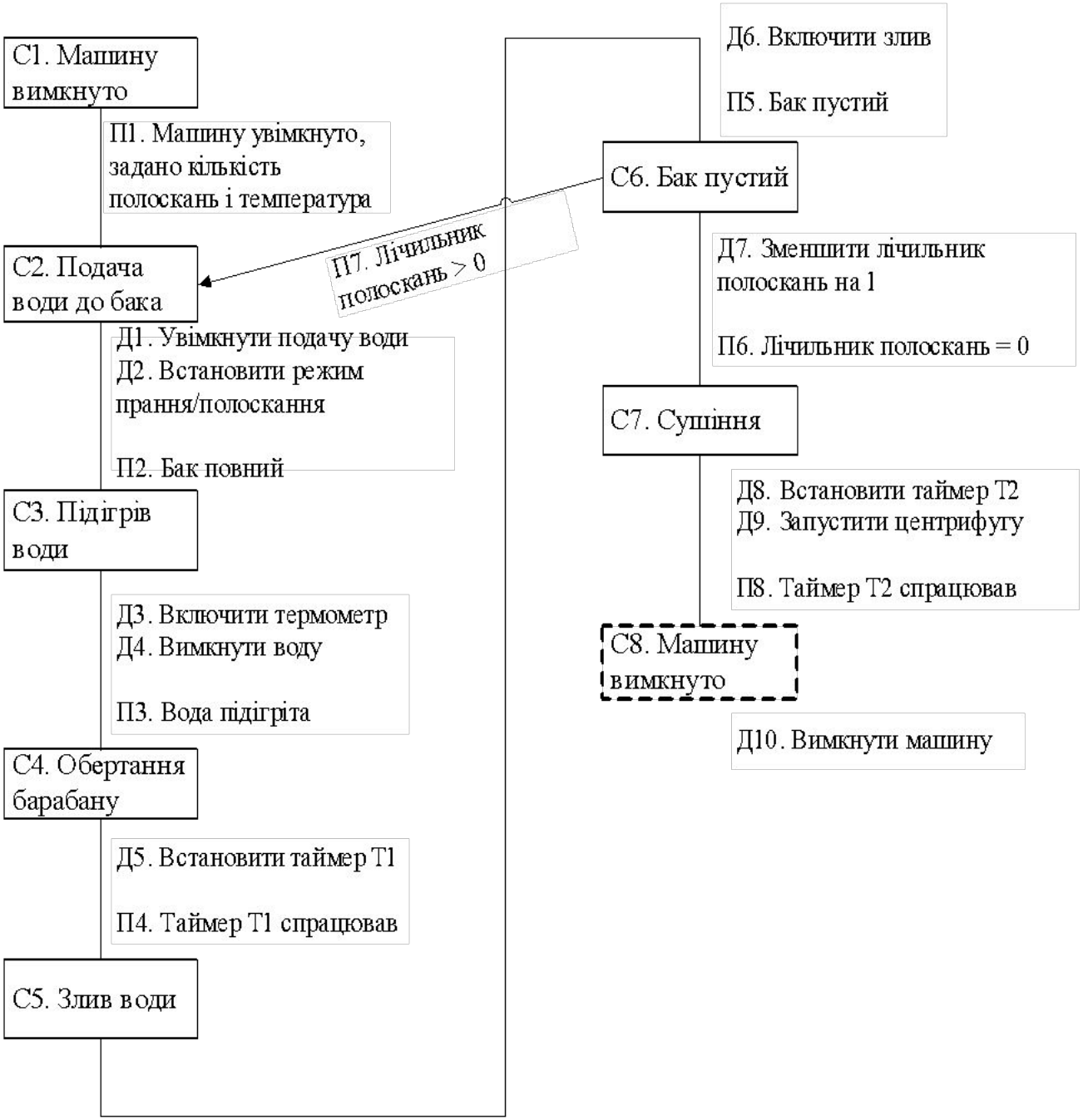
Для відображення поведінки об'єктів, у вимогах на розробку ПС потрібно:
- визначити множину станів;
- визначити множину інцидентів або подій, які спонукують екземпляри класів змінювати свій стан;
- визначити правила переходу для кожного із зафіксованих станів, які вказують у який новий стан переходить екземпляр класу;
- визначити процеси, що виконуються при переході у новий стан.

Використовуються спеціальні системні об'єкти – ***таймери*** як механізми виміру інтервалів часу (атрибути таймеру: унікальний ідентифікатор таймера; інтервал часу, через який подається сигнал про настання певної події; мітка події, яка наступить, коли залишок часу буде дорівнювати 0).

ТПС складається з рядків і стовпців, причому кожний з можливих станів об'єкта представляється рядком, а кожна з можливих подій – стовпцем. Клітина ТПС визначає стан у який переходить об'єкт.
Перевагою **ДПС** є наочність і визначення дій, однак **ТПС** дозволяє зафіксувати всі можливі комбінації стан-подія. За допомогою побудови ТПС забезпечують повноту та несуперечність представлення вимог.

Поведінка системи в цілому представляється як **схема взаємодіючих ДПС**. На схемі кожна з них має ім'я, що зображується в овалі.

ДПС автоматичної пральної машини



ТПС автоматичної пральної машини

	П1	П2	П3	П4	П5	П6	П7	П8
С1	2							
С2		3						
С3			4					
С4				5				
С5					6			
С6						7	2	
С7								1(8)

Дії є алгоритмами, які виконуються системою, як реакція на зовнішні події.

Набір таких дій визначає поведінку системи, її **функціональність**. Для подолання складності розуміння дій, виконують їх *декомпозицію* на окремі складові, які називають **процесами**. Послідовність процесів визначає **потік керування**.

При цьому процеси обмінюються даними, які утворюють **потік даних**.

Для представлення моделі алгоритмів дій системи використовуються **діаграми потоків даних дій (ДПДД)**.

Кожному стану з **ДПС** може відповідати тільки одна **ДПДД**. **Процес** на **ДПДД** зображується у вигляді овалу (усередині дається назва), потоки даних зображуються стрілками (на котрих вказуються ідентифікатори даних, напрямок до овалу - вхідні, а від овалу - вихідні дані), джерела даних зображуються прямокутниками або рамками з відкритими сторонами.

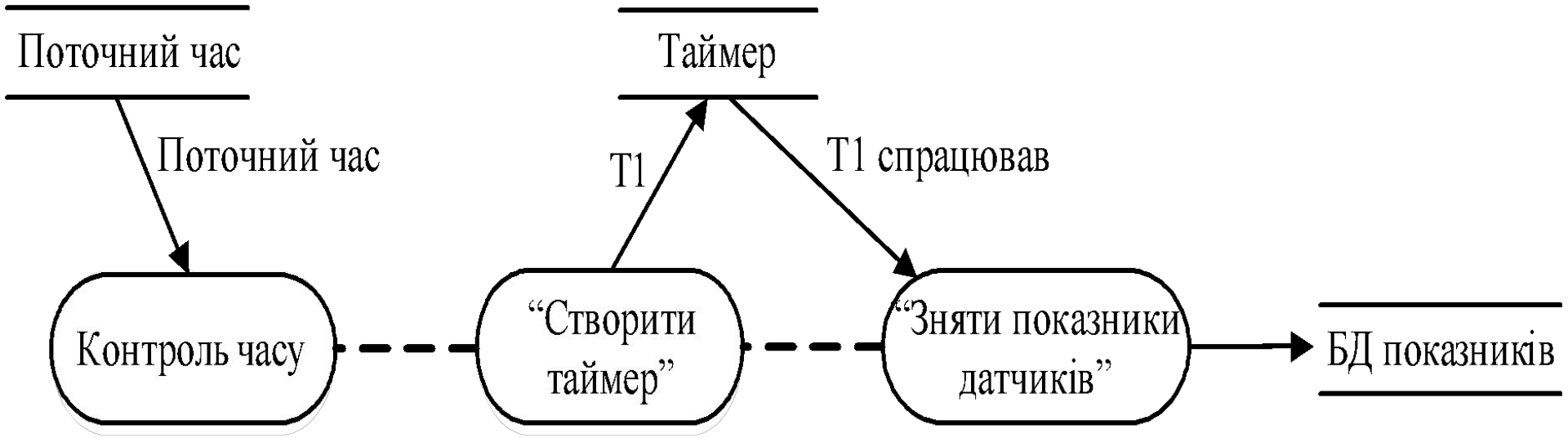
Як **джерела даних** допускаються: атрибути об'єктів (зберігаються у файлах або базах даних); системні годинники й таймери; дані подій і повідомлення.

Після побудови **ДПДД** варто побудувати **загальну таблицю процесів** із такими колонками: ідентифікатор процесу; тип процесу (аксесор – доступ до архівів даних, генератор подій, процес-обчислення, перевірка умов); повна назва процесу; назва стану, у якому визначений процес; назва дії стану;

За допомогою цієї таблиці перевіряється:

- несуперечність назв та ідентифікаторів процесів;
- повнота подій та процесів;
- можливість виявити спільні процеси для різних дій чи станів.

Приклад діаграми потоків даних дій (ДПДД) для стану “Запис показників датчиків”



Продукти інженерії вимог за методом Шлеєр і Меллора

Відповідно до методу, результатами аналізу вимог до створення ПС є наступні продукти:

Інформаційна модель системи (онтологія) у формі:

Діаграми сутність-зв'язок;
Описи об'єктів та їхніх атрибутів;
Описи зв'язків між об'єктами.

Модель поведінки об'єктів системи у формі:

ДПС і ТПС;
Описи дій для ДПС;
Описи подій для ДПС та ТПС.

Модель процесів для станів об'єктів у вигляді:

ДПДД;
Таблиці процесів станів;
Описи процесів (природною мовою).

Метод інженерії вимог І.Джекобсона (модель сценаріїв)

Метод Джекобсона – це єдиний метод, що вказує послідовний, достатньо формалізований підхід до виявлення об'єктів, суттєвих для даної предметної області.

Відбувається послідовна декомпозиція проблеми:

1) складні проблеми трансформуються в сукупність складових мети;

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

3) кожний сценарій трансформується в процесі аналізу в сукупність взаємодіючих об'єктів.

У такий спосіб будується ланцюжок трансформації:

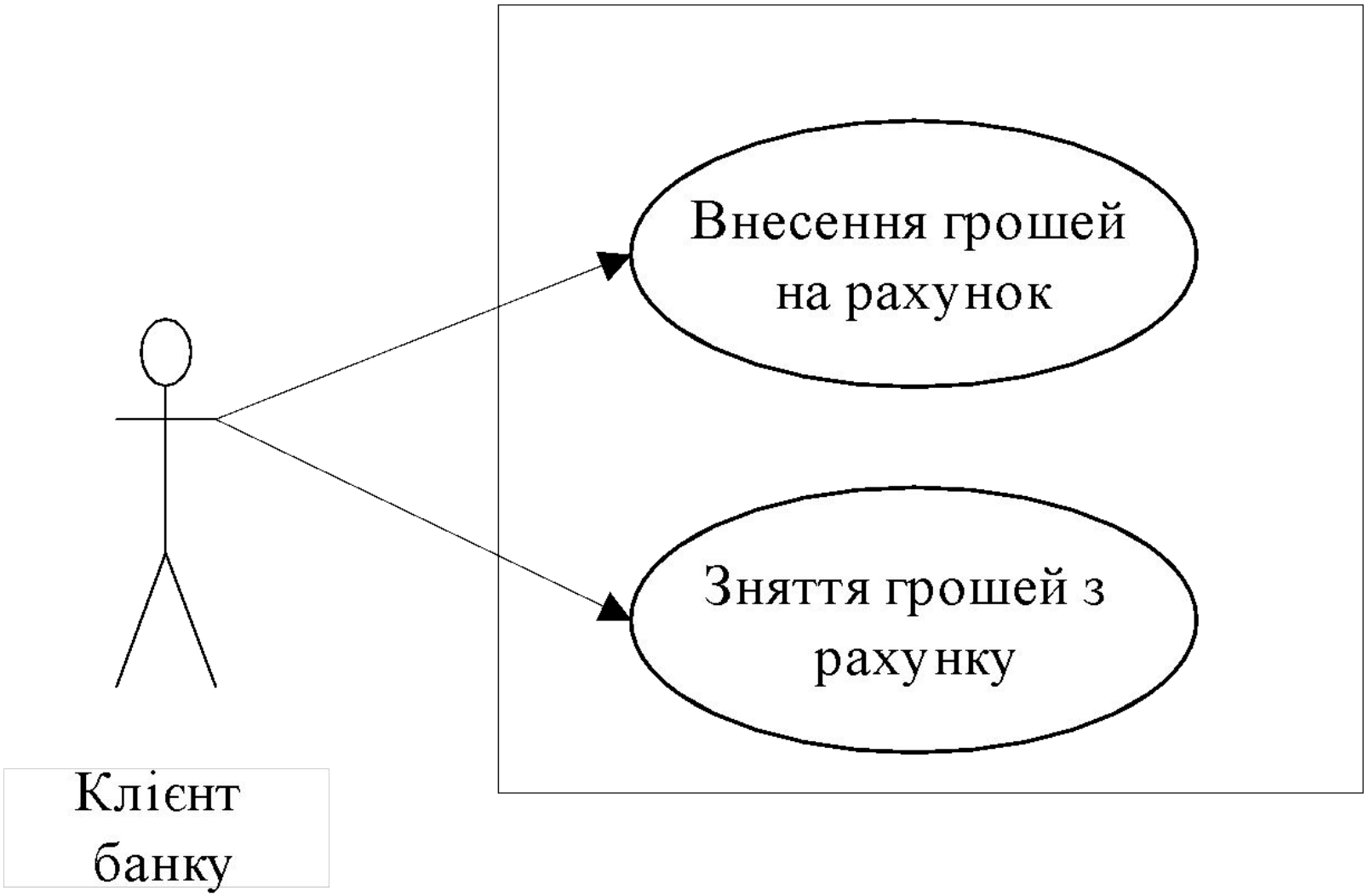
проблема – складові мети – сценарії – об'єкти.

Проведена трансформація відображається у термінах базових понять проблемної області. Кожний сценарій запускається користувачем (актором), який є зовнішнім чинником системи. Кожний сценарій запускає один актор.

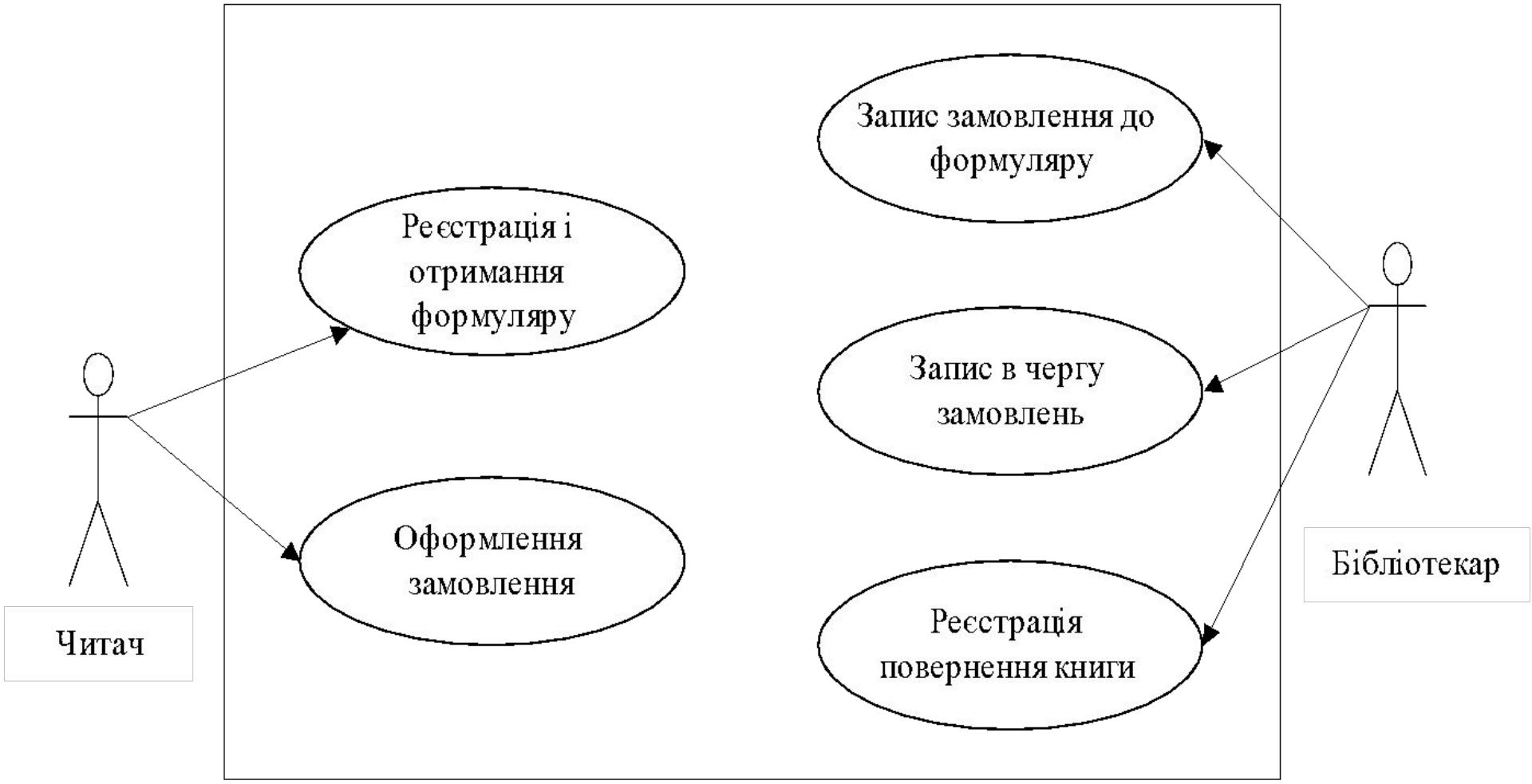
Складові мети є джерелом вимог до системи і класифікуються на обов'язкові й бажані. Складові мети можуть перебувати в певних відношеннях (узгодженість, конфлікт, кооперація, залежність).

Сценарій складається з ряду операцій і має стани й поведінку. ***Сценарій –*** це повний ланцюжок подій, ініційованих актором, і специфікація реакції на них. **Сукупність сценаріїв** визначає всі можливі шляхи використання системи.

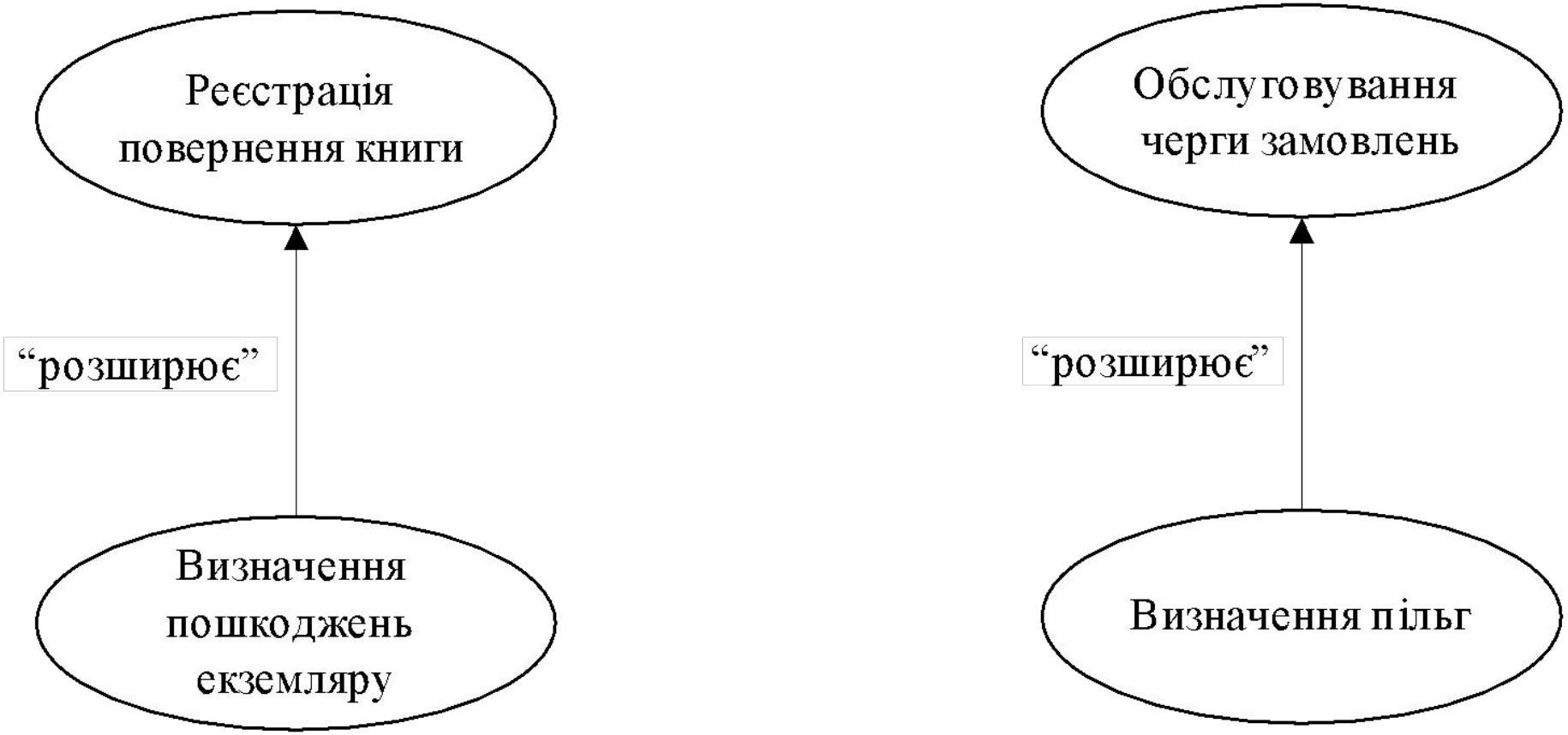
Діаграма сценаріїв для клієнта банку



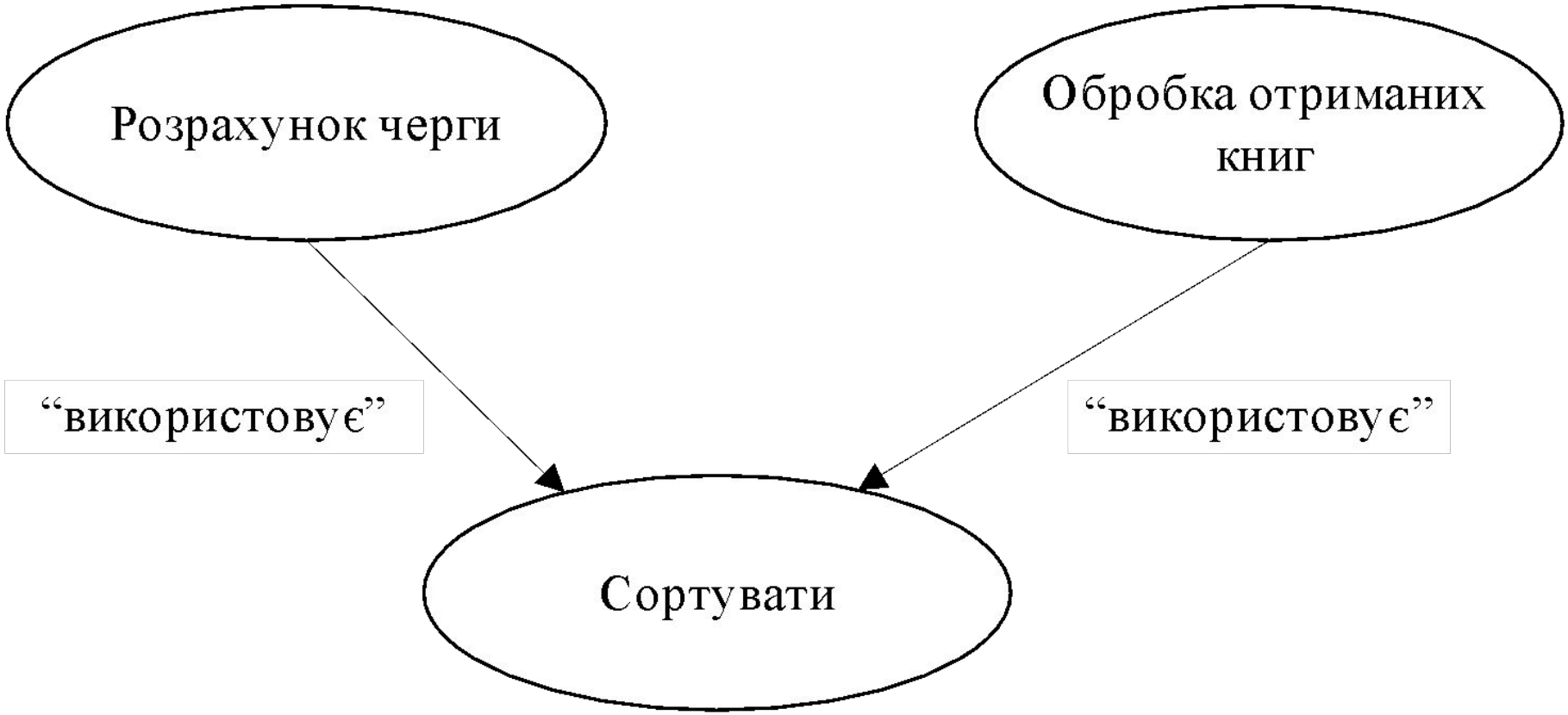
Діаграма сценаріїв бібліотеки



Приклади розширення сценаріїв



Приклад відношення “використовує”



Модель сценаріїв супроводжується неформальним описом кожного сценарію: назва; анотація; актори; опис всіх аспектів дій акторів по взаємодії з системою; передумови; функції, які виконуються в сценарії; нестандартні ситуації й реакція на них; умови завершення сценарію і постумови.

Модель вимог піддається аналізу з метою подальшого структурування проблеми, що вирішує дана ПС. Сукупність об'єктів знаходимо шляхом послідовної декомпозиції сценаріїв на об'єкти.

Вищенаведені принципи приводять до того, що простір, у якому функціонує система, потрібно структурувати в ***трьох*** вимірах: 1) інформація, що обробляє система (її внутрішній стан); 2) поведінка системи; 3) презентація системи (інтерфейси ПС з користувачем та іншими системами).

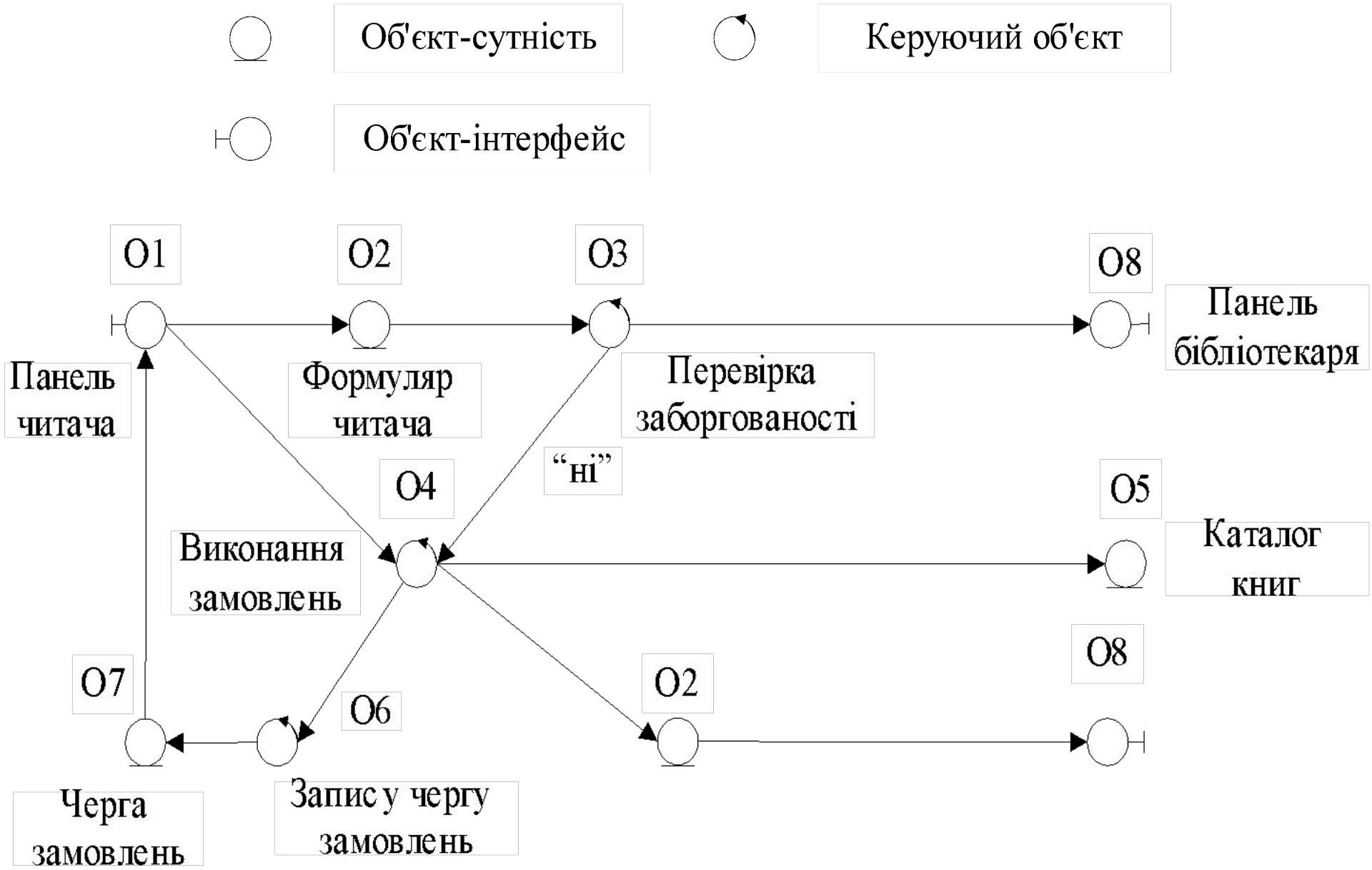
Звідси виділяємо **3 види об'єктів**:

1) ***об'єкт-сутність; 2) об'єкт-інтерфейс; 3) керуючий об'єкт.***

Об'єкт-сутність *моделює довгоживучу інформацію, яка зберігається після виконання сценаріїв.* ***Об'єкти-інтерфейси*** містять функціональності, які залежать безпосередньо від оточення системи. Ці об'єкти визначаються шляхом аналізу опису інтерфейсу сценарію з моделі вимог. Завдання об'єкта – трансляція інформації, яку вводить актор, у події, на які реагує система та зворотня трансляція інформації системи. ***Керуючі об'єкти*** – це об'єкти, які перетворюють інформацію, введену об'єктами інтерфейсу й представлену об' єктами-сутностями, в інформацію, що виводиться інтерфейсними об'єктами. Керуючі об'єкти відповідають функціям обробки інформації (перетворювачі).

Діаграма взаємодії об'єктів в рамках сценарію будується на основі аналізу вимог до сценарію. Приклад такої діаграми показано для сценарію бібліотеки.

Модель аналізу вимог: асоціації між об’єктами бібліотечної системи



Послідовний підхід до виявлення об'єктів, суттєвих для даної предметної області, складається з таких кроків:

1) Множина можливих об'єктів визначається на етапі побудови інформаційної моделі проблемної галузі (E-R діаграми домену).
2) Склад об'єктів уточнюється після побудови комплекту діаграм сценаріїв для моделювання вимог до проектованої ПС (можуть з'явитися 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 з'явитися нові об'єкти та скоротитися склад об'єктів з п.2, якщо вони не використовуються у відповідних діаграмах взаємодії).

Продукти інженерії вимог по методу Джекобсона

1) Онтологія домену (для нотації проблемної 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 в рамках сценаріїв (будуються на основі аналізу вимог до функцій ПС).

Unified Modeling Language (UML) розробили Г.Буч, Дж.Рамбо та І.Джекобсон. UML стала базовим методом при розробці ПЗ і є стандартом де-факто, як метод моделювання ПП на всіх стадіях ЖЦ. Автори визначили UML, як мову для специфікації, візуалізації, конструювання й документування ПС, а також мову моделювання бізнес-застосувань. В UML концептуальна модель вимог розглядається як сукупність нотацій – діаграм, які візуалізують основні елементи структури системи. (Вимоги до ПС задаються в основному діаграмами 1 і 2).

Діаграми UML

1) діаграма сценаріїв (варіанти використання) – use case diagram
2) діаграма класів – class diagram
3) діаграми поведінки:
   - діаграма станів – statechart diagram
   - діаграма активності – activity diagram
4) діаграми взаємодії:
   - діаграма послідовності – sequence diagram
   - діаграма кооперації – collaboration diagram
5) діаграми реалізації:
   - діаграма компонентів – component diagram
   - діаграма розміщення – deployment diagram

Відмінності **UML 2.0** полягають в обов'язковому використанні основних пакетів метамоделі та діаграм **композитних**(складених) **структур** і додаткових діаграм структур (**діаграм пакетів** та **об'єктів**). Застосовуються допоміжні ***діаграми взаємодії*** (**діаграма комунікації, огляду взаємодій та часова діаграма**), а також діаграма скінченного автомата (**State machine diagram**).

Проектування – це етап ЖЦ ПС, що йде наступним після інженерії вимог. Завдання етапу **–** перетворення вимог у проектні рішення, що мають забезпечити побудову ПС з характеристиками, які відповідають вимогам (відбувається трансформація множини вимог у простір проектних рішень).

Процеси, які є відносно незалежними один від одного (їх можна виконувати як послідовно, так і паралельно командами розробників) :

1) ***Концептуальне проектування***, яке складається з уточнення розуміння вимог й узгодження деталей вимог. Виконується: ***уточнення даних; уточнення інтерфейсів***; ***уточнення функцій обробки даних*** (для зафіксованих об'єктів уточнюється склад і зміст операцій та схема взаємодії об'єктів); ***уточнення нефункціональних вимог*** (нефункціональні вимоги відображають обмеження-накладаються організацією та середовищем використання системи).

2) ***Архітектурне проектування***, що полягає у визначенні головних структурних особливостей створюваної системи.

3) ***Технічне проектування***, що полягає у відображенні вимог середовища функціонування і платформи розробки системи у проектні рішення та визначення конструкцій у вигляді компонентів системи або їх композиції (прив' язка проекту до технічних особливостей платформи реалізації).

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

Базовим стандартом опису проектування ПЗ є стандарт *IEEE Std. 1016:1998* **–** IEEE Recommended Practice for Software Design Descriptions.

Стандарт *ISO 13407:1999.* Human-centred design processes for interactive systems - опис процесів проектування людино-орієнтованих інтерактивних систем.

Архітектурне проектування: порівнева архітектура ПС

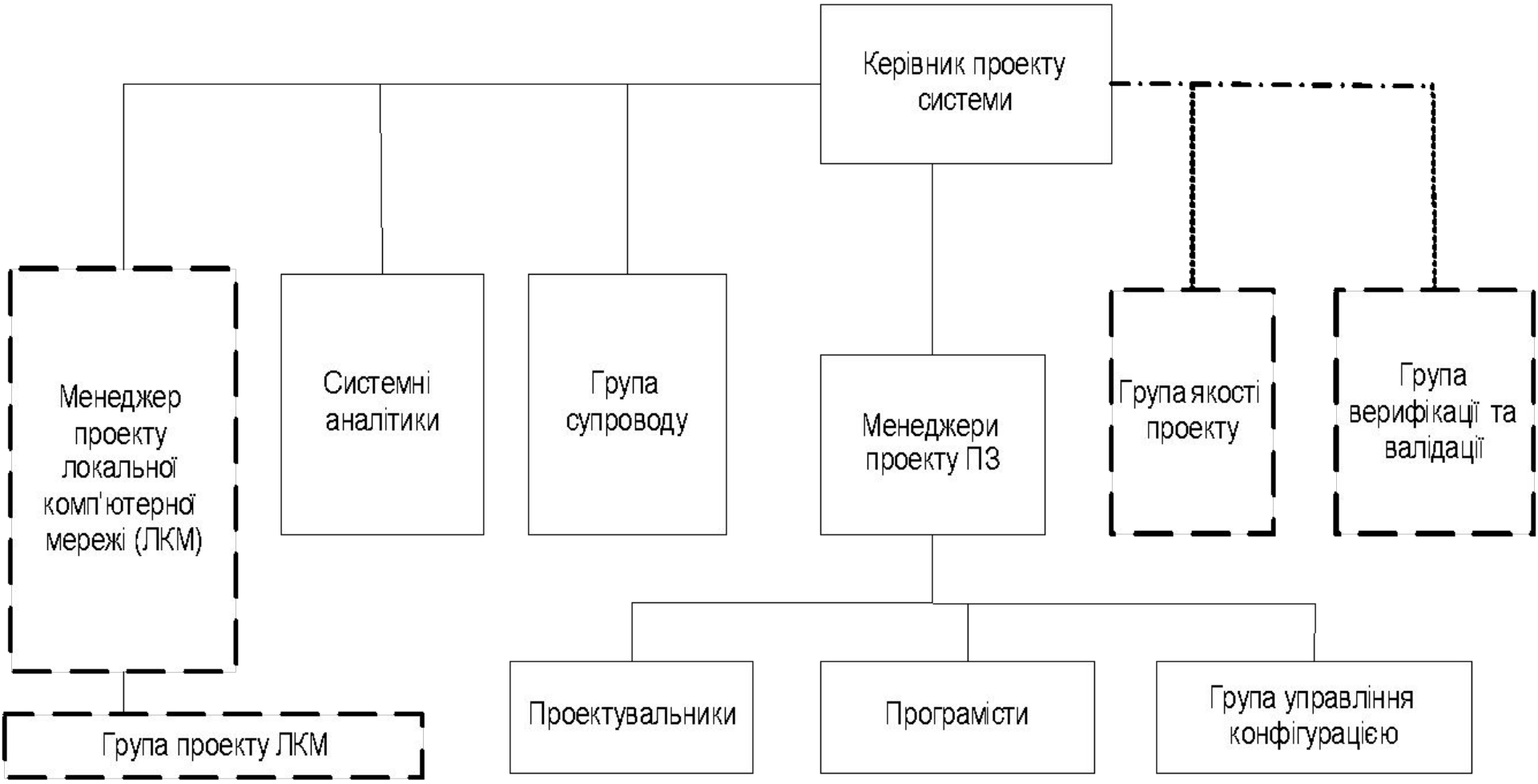
4	Прикладні системи
3	Компоненти бізнес-застосувань
2	Загальносистемні компоненти (інтерфейс з універсальними системами)
1	Системні компоненти (інтерфейс з обладнанням)

Архітектурне проектування полягає у визначенні: складу компонент; способів їхньої композиції; обмеження на їх зв'язки та сполучення.

Трансформація проекту в програмну систему

Процес має кілька назв: ***конструювання,кодування, програмування,реалізація***
Необхідно виконати конструювання, визначивши модулі, потім запрограмувати модулі обраною мовою програмування, для чого переводять нотацію проекту в коди, після чого проводиться тестування й верифікація отриманого коду ПС. В цілому, необхідно перетворити проект у систему, документуючи достатнім чином всі етапи трасформації, тобто реалізувати ПС. Конче необхідно при реалізації ПЗ застосовувувати CASE-засоби.

Рівень проекту розробки системи



Якість ПС та аспекти її вимірювання

У 1998 році був введений в дію стандарт **ISO/IEC 15026 – System and software integrity levels**, який визначає рівні цілісності систем і ПЗ.

Стандарт пропонує визначати рівень оцінювання якості від А (вищого) до D (нижчого), з врахуванням ризиків у сфері: *безпеки функціонування* (А – загроза життю людей); *захисту інформації* (В – загроза втрати інформації); *економіки* (С – загроза фінансових збитків); *середовища функціонування* (D – загроза руйнування середовища експлуатації ПС). Стандарт пропонує *пріорітети важливості* для характеристик якості, що досягається введенням *коефіцієнтів вагомості*.

Вітчизняний стандарт **ДСТУ 2850–94 «Показники та методи оцінювання якості»** рекомендує порядкову шкалу з п'яти значень: 5 – "вкрай важливо", 4 – "дуже важливо", 3 – "важливо", 2 – "добре було б", 1 – "неважливо".

Вимірювання – це одержання об'єктивних даних про стан продуктів, процесів та ресурсів розробки ПС. Вимірювання проводять із метою побудови прогнозних і оціночних моделей, які надалі застосовуються для керування проектом та удосконалення процесів розробки ОР.

Вимірювання в проекті – це багатокроковий процес: 1) *визначення мети вимірювання* – служить 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, які обираються таким чином, щоб результат вимірювання робочих продуктів, процесів та ресурсів протягом ЖЦ дозволяв оцінити їхню відповідність цілям проекту.

Основні міри: -розмір і складність ПС (на основі цього можна оцінити трудомісткість і вартість розробки);
- продуктивність (роботи фахівця й колективу в цілому);
- міри вимірювання атрибутів якості.

4) організація збору даних для виконання вимірювань (форми,тести,запитальники).

5) застосування моделей у процесі вимірювання.

Для оцінки ПС використовують два процеси: **атестація ПС; сертифікація ПС.**

Атестація проводиться ОР для перевірки того, чи відповідає ПС заданим вимогам. У процесі атестації використовуєть *тестування*.

Сертифікація проводиться третьою стороною **–** спеціальним *органом*, ліцензованим державою для таких дій. ***Сертифікація –*** це оцінка відповідності властивостей ПС вимогам до неї. *Сертифікаційні випробування* проводяться *лабораторіїями сертифікації*, які призначаються *органом сертифікації.*

Фактори, що впливають на якість ПС

1) Прикладна область і категорії користувачей. 2) Мета моделювання якості.
3) Стадія ЖЦ. Із плином етапів ЖЦ погляд на якість може змінюватися.

Цільова якість ПС (відображає потреби користувача);

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

Прогнозована якість програмного продукту – рівень, що передбачається як якість кінцевого програмного продукту; *Якість продукту*, що поставляється;

Експлуатаційна якість – якість вимірювана в термінах результату використання ПС, але не її властивостей.

Для високоцілісних систем найважливішою характеристикою є ***надійність,*** для якої показник середнього часу між відмовами ПС в період експлуатації має бути не меншим ніж 6 місяців. В той же час для систем з низьким рівнем цілісності найважливішою характеристикою є ***функціональність***, підхарактеристика ***точність***, для якої значення числа отриманих точних результатів має бути не меншим від 95% загального числа отриманих результатів.

Галузь знань ІПЗ "Якість ПЗ"

1) Основи якості ПЗ - Software Quality Fundametals.
2) Процеси керування якістю - Software Quality Management Processes.
3) Практичні міркування - Practical Considerations.

1) Питання якості повинні розглядатися в контексті *вимог до ПП* => Вибір характеристик якості. Вибір методів кількісної оцінки показників якості. Метрики якості. Модель якості. Базовий стандарт якості **ISO/IEC 9126 (parts 1-4).**

Інженери з групи забезпечення якості мають сприймати питання якості як частину власної *професійної культури*. Досягти *компромісу* між *цінністю* характеристик якості для замовника і *вартістю* забезпечення необхідного їх рівня.

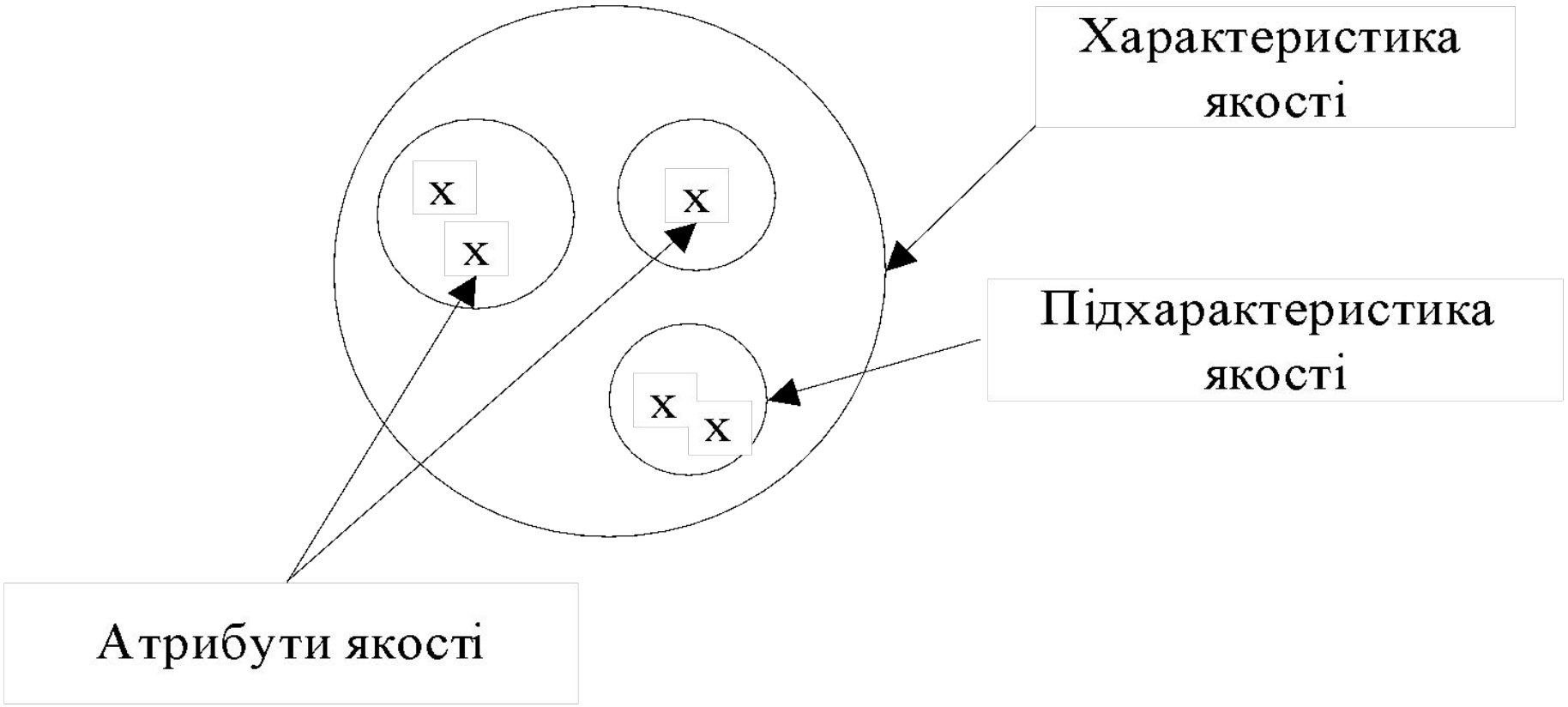
2) Процеси **SQM** – контроль і оцінка всіх процесів, продуктів та ресурсів програмної інженерії на етапах ЖЦ з точки зору покращання їх якості.
CMM – Capability Maturity Model (Модель потужності, зрілості можливостей ОР).
ISO/IEC 15504:*2004.* Information technology – Software Process Assessment.

3) *ISO 9000-3:1997.* Guidelines for the application of ISO 9001:1994 to the development, supply, installation and maintenance of computer software.

Критичність області застосування (відмовостійкість, безпека експлуатації, захищеність, зручність використання, рівень цілісності ПС).

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

Інженерія якості ПС - керований процес інкорпорації в ПС на кожній стадії ЖЦ певних властивостей, які називають характеристиками якості.



Згідно стандарта ISO/IEC 9126 існують три види якості ПЗ:

External quality (зовнішня якість);
Internal quality (внутрішня якість);
Quality in use (якість у використанні).

Модель якості – це множина взаємозалежних характеристик якості, що утворює базис для специфікації вимог до якості й оцінювання якості.

Модель якості, а також метрики якості розглянуті в серії стандартів ISO/IEC 9126 (parts 1-4) 2001-2004рр. :

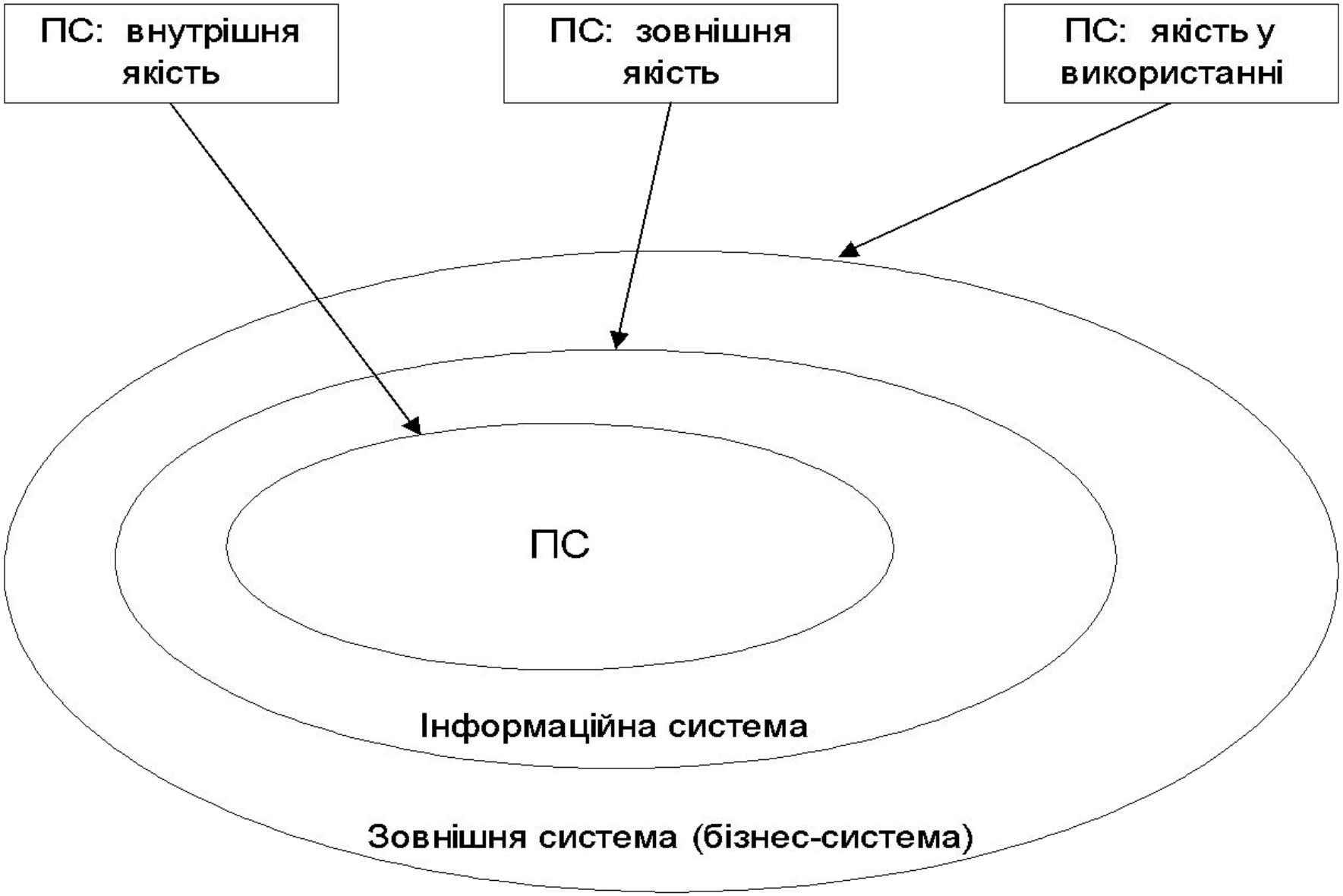
ISO/IEC 9126-1 – Модель якості ПС;

ISO/IEC 9126-2 – Метрики зовнішньої якості ПС;

ISO/IEC 9126-3 – Метрики внутрішньої якості ПС;

ISO/IEC 9126-4 – Якість у використанні.

Зв’язок між системами та типи якості



Зовнішні характеристики якості відображають вимоги до продукту. Кожна характеристика якості складається з ***підхарактеристик***. Аби кількісно визначити критерії якості, специфікують зовнішні вимірні властивості (***зовнішні атрибути***) і пов'язані з ними ***метрики***, які являють собою моделі оцінювання атрибутів.
Зовнішні метрики – це метрики застосування яких можливо тільки для ПЗ, яке працює на комп'ютері та перебуває на стадії тестування або функціонування.
Внутрішні атрибути використовуються для планування досягнення 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 програмними продуктами (документація на ПС, тексти програм, тестові набори даних тощо). Ці продукти утворюються на стадії розробки, що передує тестуванню. **Зовнішні** й **внутрішні** характеристики якості відносяться до власних властивостей ПС і відображають погляд замовника й розробника на якість. Існує ще й третій погляд на якість – погляд кінцевого користувача, який очікує максимальної ефективності від використання ПС – підвищення продуктивності й загальної задоволеності. Такий підхід визначає ***якість у використанні*** **(*quality in use*)**, або ***експлуатаційну якість***.
Експлуатаційна якість – це сукупний ефект характеристик якості для кінцевого користувача. Вона виміряється в термінах результату використання ПС, але не у властивостях самої ПС.
У процесі розробки 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.

Концепція підвищення якості ПС. Методи підвищення якості

Перший крок – це впровадження в практику організації-розробника (ОР) спеціальних підтримуючих 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 ОР методів і засобів контролю якості на всіх процесах і етапах ЖЦ ПС. Процес SQA регламентований у стандарті IEEE Std. 730:1998 **–** IEEE Standard for Software Quality Assurance Plans.

Верифікація – це дослідження трансформації вхідних робочих продуктів у вихідні; при цьому перевіряється, чи правильно розробляється програмний продукт (наприклад, чи правильний код програми по відношенню до вхідних специфікацій).

Валідація – це дослідження сукупності робочих продуктів на певному етапі процесу розробки на предмет, аби 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 V&V.

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

Проведення *формальних спільних перевірок* регламентується стандартом IEEE Std. 1028:1997 – IEEE Standard for Software Reviews. (5 видів перевірок).

План верифікації і валідації ПС

1. Призначення
2. Посилання
3. Визначення
4. Організація роботи по V&V
   - 4.1. Організація
   - 4.2. Графік робіт
   - 4.2 Схема призначення рівнів цілісності ПО
   - 4.4. Розподіл ресурсів
   - 4.5Відповідальності
   - 4.6. Інструменти, методології і методи
5. Процеси і дії з V&V
   - 5.1. Процес: Управління
     - 5.1.1. Дія: Управління V&V
   - 5.2. Процес: Придбання
     - 5.2.1. Дія: V&V підтримки придбання
   - 5.3. Процес: Постачання
     - 5.3.1. Дія: V&V планування
   - 5.4. Процес: Розробки
     - 5.4.1. Дія: V&V Концепції
     - 5.4.2. Дія: V&V Вимог
     - 5.4.3. Дія: V&V Проекту
     - 5.4.4. Дія: V&V Реалізації
     - 5.4.5. Дія: V&V Випробувань
     - 5.4.6. Дія: V&V Введення в дію
   - 5.5. Процес: Експлуатація
     - 5.5.1. Дія: V&V Експлуатації
   - 5.6. Процес: Супровід
     - 5.6.1. Дія: V&V Супроводу
6. Вимоги до звітних документів V&V
7. Вимоги до адміністративних процедур
   - 7.1.Реєстрація і ухвалення рішень по аномаліях
   - 7.2. Політика ітерації завдань
   - 7.3. Регулювання відхилень від плану
   - 7.4. Процедури контролю
   - 7.5. Стандарти, практичне керівництво і угоди
8. Вимоги до документування V&V

Входи V&V
- Опис концепції системи
- Плани та графіки розробки
- Потреби користувача
- Потреби замовника
- Призначення рівнів цілісності
- Звіт про аналіз загроз
- Результати розв'язання задач V&V

Входи V&V
- Документи прийнятої концепції
- Специфікація вимог до ПЗ
- Специфікація вимог до інтерфейсів
- Звіт про критичність задачі
- Документація користувача
- План випробувань системи
- план приймочних випробувань
- Документація процесу управління конфігурацією ПЗ
- Звіт про аналіз загроз
- Плани і графіки розробки
- Результат рішення задач V&V

Входи V&V
- Специфікація вимог до ПЗ
- Опис проекту ПЗ
- Специфікація вимог до інтерфейсів
- Документ проекту інтерфейсу
- Стандарти проектування
- Документація концепції
- Звіт про критичність задачі
- Проекти та плани випробувань
- Документація користувача
- Звіт про аналіз загроз
- Плани та графік розробки
- Результат вирішення задач V&V

Входи V&V
- Опис проекту ПЗ
- Документ проекту інтерфейсу
- Початковий код
- Стандарти кодування
- Документація користувача
- Документація концепції
- Звіт про критичність задачі
- Проекти/ набори тестів
- Процедури тестування
- Результат тестування компонентів
- Звіт про аналіз загроз
- Плани та графік розробки
- Результат вирішення задач V&V

Входи V&V
- Плани, проекти, набори і процедури тестування
- Опис проекту ПЗ
- Документ проекту інтерфейсу
- Початковий та кінцевий код
- Результат тестування компонентів
- Документація користувача
- Звіт про аналіз загроз
- Плани та графік розробки
- Результат вирішення задач V&V

Входи V&V
- Пакет інсталяції
- Документація користувача
- Звіт про аналіз загроз
- Плани та графік розробки
- Результат вирішення задач V&V
- Підсумковий звіт про дії V&V

Дії по V&V **концепції**

Дії по V&V **вимог**

Дії по V&V **проекту**

Дії по V&V **реалізації**

Дії по V&V **випробувань**

Дії по V&V **інсталяції пуску**

Задачі V&V
- Оцінка документу концепції
- Аналіз критичності
- Аналіз розподілення вимог (до ТО/ ПЗ)
- Аналіз трасованості
- Аналіз загроз
- Аналіз ризику

Задачі V&V
- Аналіз трасованості
- Оцінка вимог до ПЗ
- Аналіз інтерфейсів
- Аналіз критичності
- Створення/верифікація плану тестування для V&V системи(С)
- Створення/верифікація плану тестування для V&V приймання(П)
- Оцінка стану управління конфігурацією
- Аналіз загроз
- Аналіз ризику

Задачі V&V
- Аналіз трасованості
- Оцінка проекту ПЗ
- Аналіз інтерфейсів
- Аналіз критичності
- Створення/верифікація плану тестування для V&V компонентів (К)
- Створення/верифікація плану тестування для V&V інтеграції (І)
- Створення/верифікація проекту тестування для V&V
- Аналіз загроз
- Аналіз ризику

Задачі V&V
- Аналіз трасованості
- Аналіз початкового коду та документації початкового коду
- Аналіз інтерфейсів
- Аналіз критичності
- Створення/верифікація тестових наборів для V&V
- Створення/верифікація тестових процедур для V&V
- Створення/верифікація тестів компонентів
- Аналіз загроз
- Аналіз ризику

Задачі V&V
- Аналіз трасованості
- Створення/верифікація тестових процедур для V&V приймання
- Виконання/верифікація тестів для V&V інтеграції
- Виконання/верифікація тестів для V&V системи
- Виконання/верифікація тестів для V&V приймання
- Аналіз загроз
- Аналіз ризику

Задачі V&V
- Аудит інсталяційної конфігурації
- Пробний пуск
- Аналіз загроз
- Аналіз ризику
- Генерація заключного звіту V&V

Виходи V&V
- Звіти про задачі
- Звіти про аномалій

Виходи V&V
- Звіти про задачі
- Звіти про аномалій
- Плани тестування для V&V системи/ приймання

Виходи V&V
- Звіти про задачі
- Звіти про аномалій
- Плани тестування для V&V КЛ
- Проекти тестів для V&V КЛ/С/П

Виходи V&V
- Звіти про задачі
- Звіти про аномалій
- Набори тестів для V&V КЛ/С/П
- Процедури тестування для V&V КЛ/С

Виходи V&V
- Звіти про задачі
- Звіти про аномалій
- Процедури тестування для V&V приймання

Виходи V&V
- Звіти про задачі
- Звіти про аномалій
- Заключний звіт про V&V

Технічний огляд - це систематична оцінка придатності ПП для використання по призначенню та ідентифікація розбіжностей зі специфікаціями і рекомендаціями стандартів та керівництв.
Управлінський огляд - систематична оцінка стану процесів ЖЦ з метою контролю просування проекту, визначення планів, розподілу ресурсів і додержання стандартів.
Формальна інспекція - це систематична перевірка ПП з метою виявлення та ідентифікації проблем, включаючи дефекти та відхилення від специфікацій і стандартів.
Наскрізний перегляд - це запланований систематичний контроль перебігу розробки ПП і оцінка його стану. Мета – пошук дефектів, покращання продукту, оцінка відповідності специфікаціям і стандартам.
Аудиторська перевірка - це аналіз та незалежна перевірка продукту або процесу третьою стороною з метою оцінювання відповідності стандартам, керівництвам, планам, умовам договору. Вона виконується групою зовнішніх аудиторів (1-5 чоловік).

Другий крок по підвищенню якості – це досягнення ОР високого рівня зрілості.

Основними стандартами в області якості продукту є стандарти серії **ISO 9000**.

Серія заснована на методології CMM (Capability Maturity Models – модель зрілості можливостей). Методологія CMM та шляхи її впровадження в діяльність організації-розробника ПП докладно розглянуті в стандартах серії **ISO 9000**.

Стандарт ISO 9000 рекомендує кожній ОР мати свою власну систему якості.

Система якості ПС (за стандартом ДСТУ 2844) – це сукупність організаційної структури, відповідальності, процедур, процесів і ресурсів, які спрямовані на реалізацію керування якістю ПС.

Ядро інженерії якості базується на *елементах* SWEBOK (SoftWare Engineering Body Of Knowledge) та PMBOK (Project Management Body Of Knowledge)

Ядро професійних знань інженерії якості

У 90-х роках світове комп'ютерне співтовариство почало систематизацію знань у різних галузях комп'ютерних наук та інформатики з метою зафіксувати їх у вигляді ***ядер знань***. Для створення ядра знань по програмній інженерії зусиллями ACM (Association for Computing Machinery) та IEEE Computer Society був створений координаційний комітет по програмній інженерії SWECC (SoftWare Engineering Coordinating Committee), в який увійшли спеціалісти світового рівня.

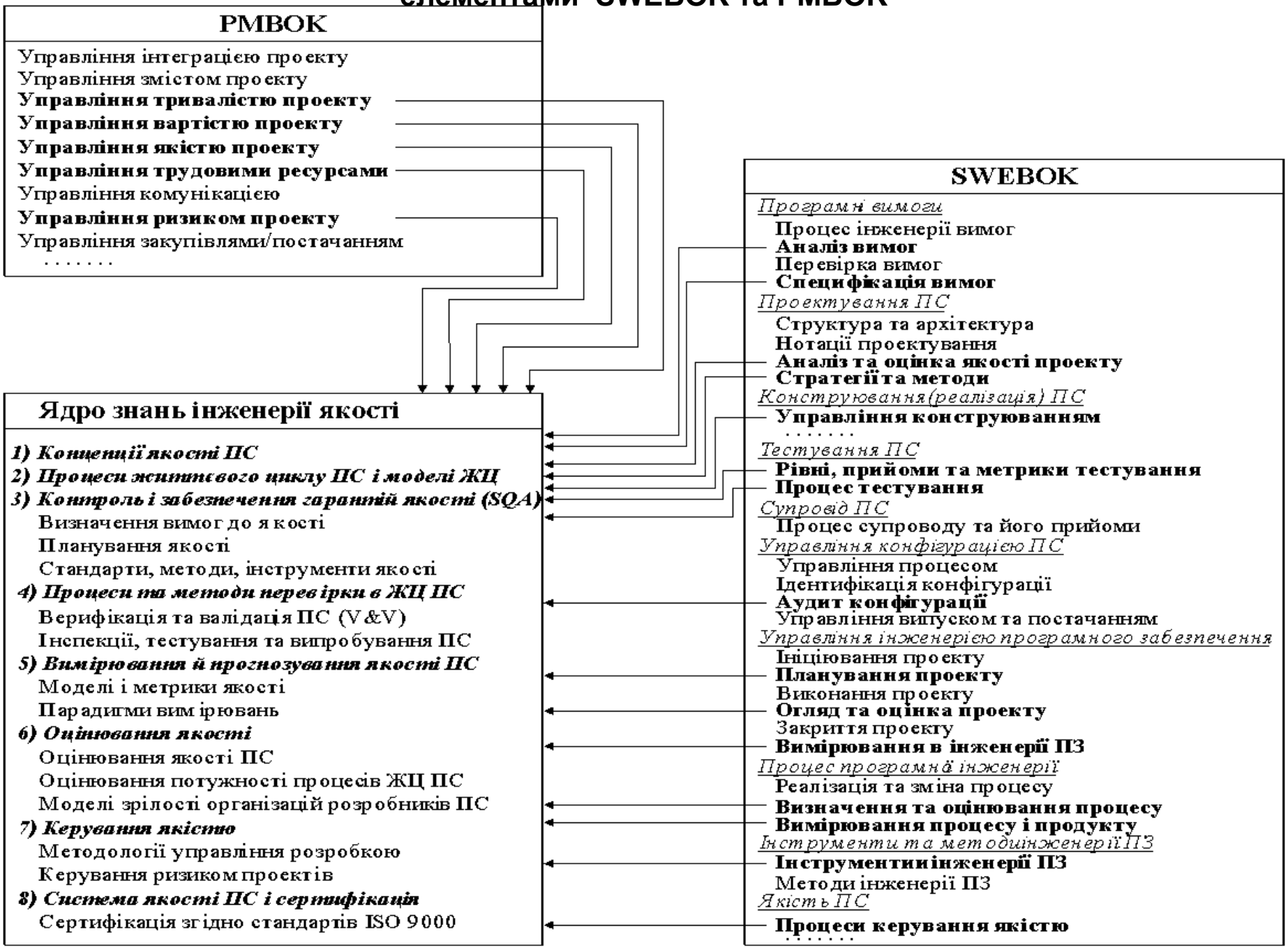
Ядро знань по програмній інженерії у остаточній редакції підкомітету "Software and System Engineering" ISO/IEC вийшло у 2004 році під назвою: Software Engineering Body of Knowledge (SWEBOK) Керівництво по SWEBOK було видане згодом у формі стандарту *ISO/IEC 19759.* Software Engineering – Guide to SWEBOK.

Керівництво по ядру знань в галузі управління проектами було розроблено організацією PMI (Project Management Institute). Третя версія PMBOK містить опис *процесів* та ключових галузей знань з управління проектами в промисловості та створення ПЗ. Встановлено рубрики (розділи) як для SWEBOK, так і для PMBOK, а ядро знань інженерії якості було синтезовано на їх базі.

У **ядро знань інженерії якості** увійшли такі розділи: концепція якості ПС; процеси ЖЦ ПС і моделі ЖЦ; контроль і забезпечення гарантій якості (SQA); процеси та методи перевірки в ЖЦ ПС; вимірювання й прогнозування якості ПС; оцінювання якості ПС; керування якістю; система якості ПС і сертифікація.

Під ***<u>інженерією якості ПС</u>*** розуміють керований процес інкорпорації в ПС на кожній стадії ЖЦ певних властивостей, які називають характеристиками якості. Наявність і рівень досягнення цих властивостей вимірюється, прогнозується, оцінюється й регулюється за допомогою сукупності стандартів, процесів і методів.

Ядро професійних знань інженерії якості. Зв'язок ядра знань інженерії якості з елементами SWEBOK та PMBOK



Процес інженерії якості має гарантувати, що:

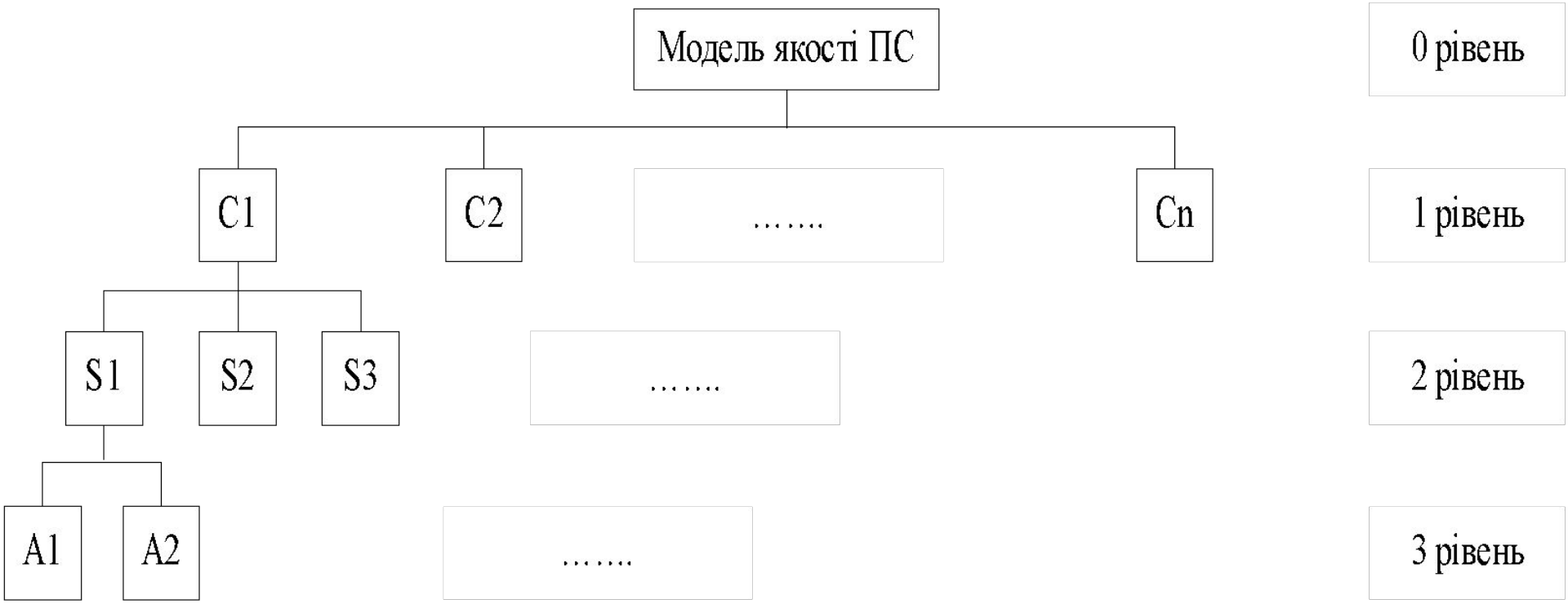
- вимоги до якості враховують погляд замовників, користувачів і розробників ПС;
- всі вимоги до якості можуть бути виміряні кількісно або верифіковані;
- процеси ЖЦ містять процедури, орієнтовані на виявлення вимог до якості й аналіз досягнення властивостей якості;
- стандарти в області якості визначені й дотримуються;
- розробник ПС розуміє цілі якості й вбудовує в ПП властивості, які забезпечують зовнішню, внутрішню й експлуатаційну якість;
- група якості на підприємстві проводить виміри й оцінювання якості ПП;
- результати вимірів використовуються для керування програмним проектом;
- виконуються процеси V&V та тестування ПС з метою перевірки відповідності ПС вимогам до якості.

Стандарт *ISO/IEC 12207* не виділяє **інженерію якості** у вигляді окремого процесу ЖЦ, однак із представлених в ньому процесів, безпосередньо інженерії якості стосуються наступні процеси:

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

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

Метрика – це комбінація конкретного *методу вимірювання* атрибута сутності й *шкали вимірювання*.



***C**haracteristic, **S**ubcharacteristic, **A**ttribute*

Метод вимірювання визначає спосіб одержання значень атрибута якості певної сутності. *Шкала* використовується для структурування отриманих значень.

Метрика визначає ***міру*** атрибута, тобто змінну, котрій присвоюється значення в результаті вимірювання. Наприклад, *сутність* – це звіт про виявлення дефектів у ПС, *атрибут* – список дефектів, *метод вимірювання* – підрахунок кількості дефектів, *шкала* – цілі числа більше 0, *міра атрибута* – загальне число дефектів, *ім'я метрики* (однойменне *мірі*) – загальне число дефектів.

Міра атрибута є безпосередньою, якщо вона не залежить від мір інших атрибутів, або побічною, якщо утворюється на основі мір інших атрибутів.

В основі ***базової метрики*** лежить елементарний метод вимірювання атрибута.

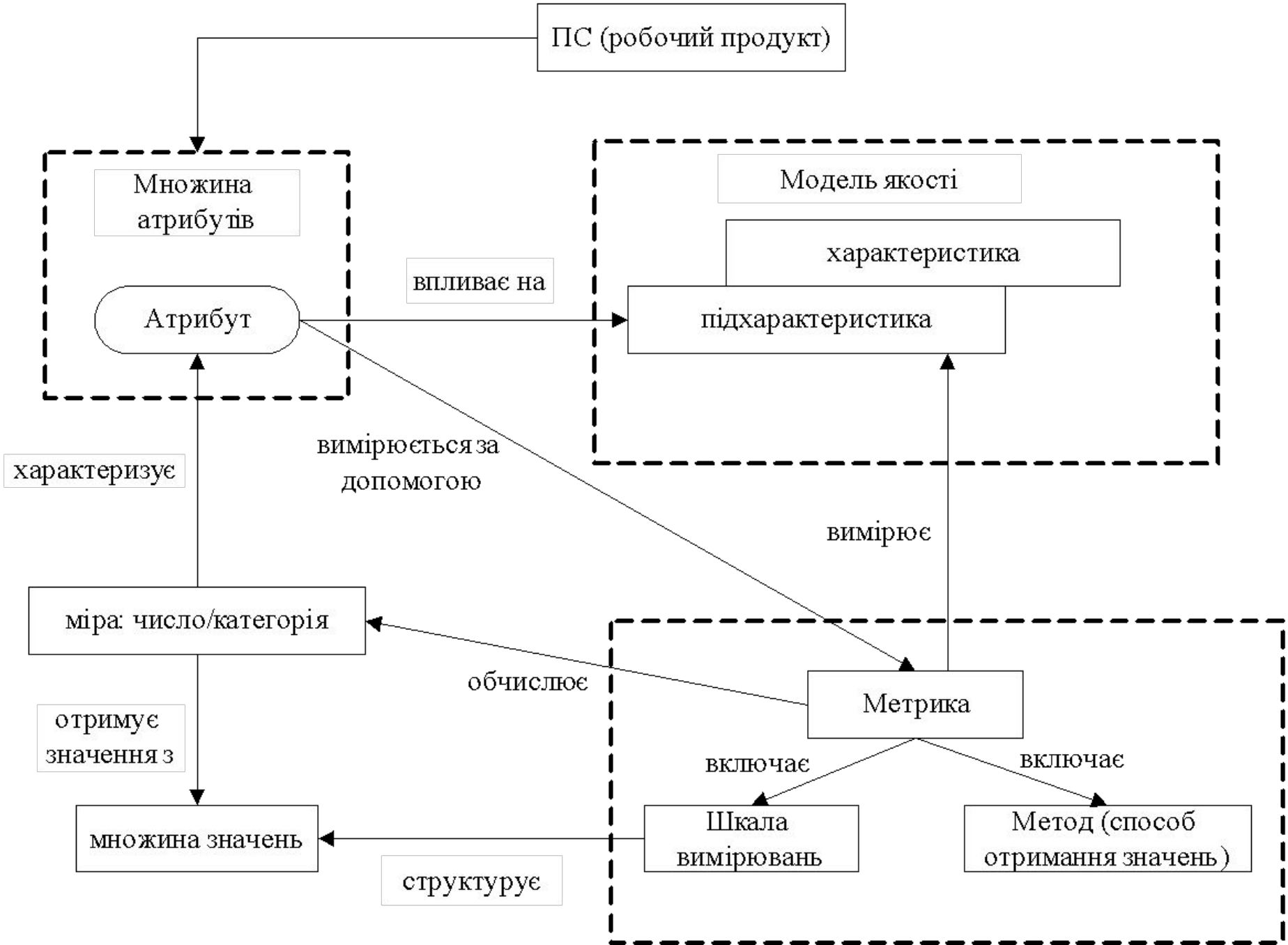
Стандарт ISO/IEC 9126-2 рекомендує застосовувати *5 видів шкал виміру значень*:

номінальна шкала (класифікаційна шкала); ***порядкова шкала***, яка дозволяє впорядковувати властивості по зростанню/зменшенню, шляхом порівняння з базовим значенням; ***інтервальна шкала*** – відзначає дистанцію між властивостями об'єкта (арифметичні операції та операції порівняння);

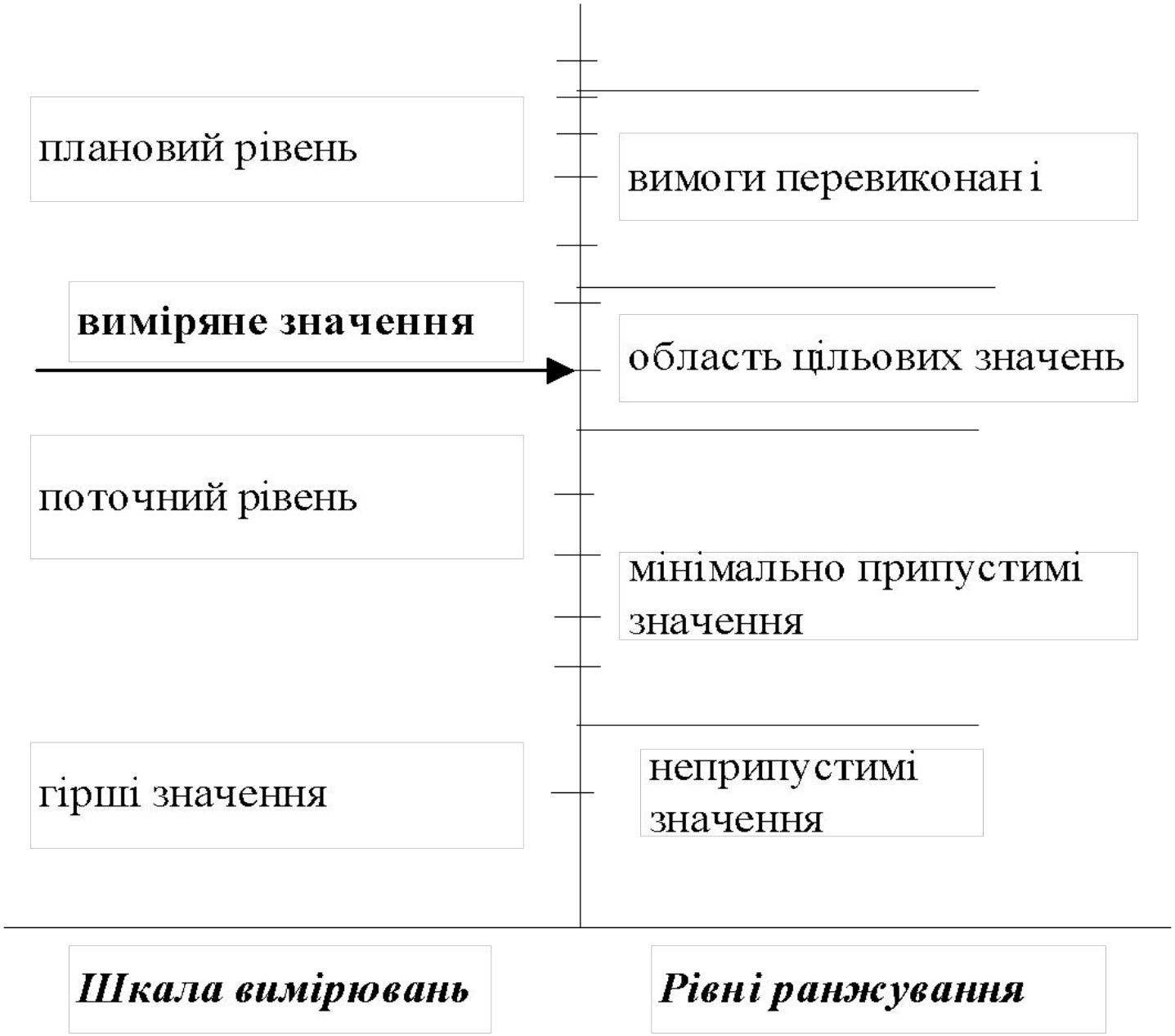
відносна шкала – значення розрізняються відносно обраної одиниці вимірювання (найбільш пріорітетна шкала); ***абсолютна шкала*** є спеціальним випадком відносної шкали, де вказується абсолютне значення величини.

Інформацію про рівень задоволення вимог до якості задають **області (ранги)** на **шкалі**, які відповідають різним ступеням задоволеності вимог.

Метрика в системі вимірювання якості



Рівні ранжування метрик



Класифікація мір та метрик якості

Стандарт ISO/IEC 9126-2 визначає наступні **типи мір**:

міра розміру – представляє розмір ПС в одиницях вимірювання (функціон. розмір, число функцій або модулів, кількість рядків у програмах, розмір дискової пам'яті та ін.); ***міра часу*** – періоди реального часу (у секундах, хвилинах, годинах процесорного часу, час функціонування системи); ***міра зусиль*** –корисний час пов' язаний з певним завданням (продуктивність праці, трудомісткість);

міра інтервалів між подіями (наприклад, час між відмовами);

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

Метрики зазвичай ***класифікуються*** наступним чином:

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

Примітивні/такі що обчислюються. Примітивні – це базові метрики, які безпосередньо спостерігаються (розмір програми, число дефектів); *такі що обчислюються* – обчислюються на основі примітивних метрик (трудомісткість);

Динамічні/статичні. Динамічним метрикам властивий компонент часу (число помилок за місяць); *статичні метрики* не залежать від часу (число виявлених загалом дефектів); *Такі що передбачаються (прогнозуючі)/пояснюючі.* Значення прогнозуючих метрик отримують заздалегідь (прогнозована кількість відмов); значення пояснюючих з'являються постфактум (реальна інтенсивність відмов).

Зовнішні метрики використовують ***міри ПП***, який працює на комп'ютері.

Такі міри отримують в результаті вимірювання поведінки ПС у ході тестування й функціонування відповідно до сценаріїв використання ПС.

Відповідні їм метрики використовують для демонстрації якості ПП,

а також для підтвердження того, що ПП задовольняє зовнішнім вимогам до якості.

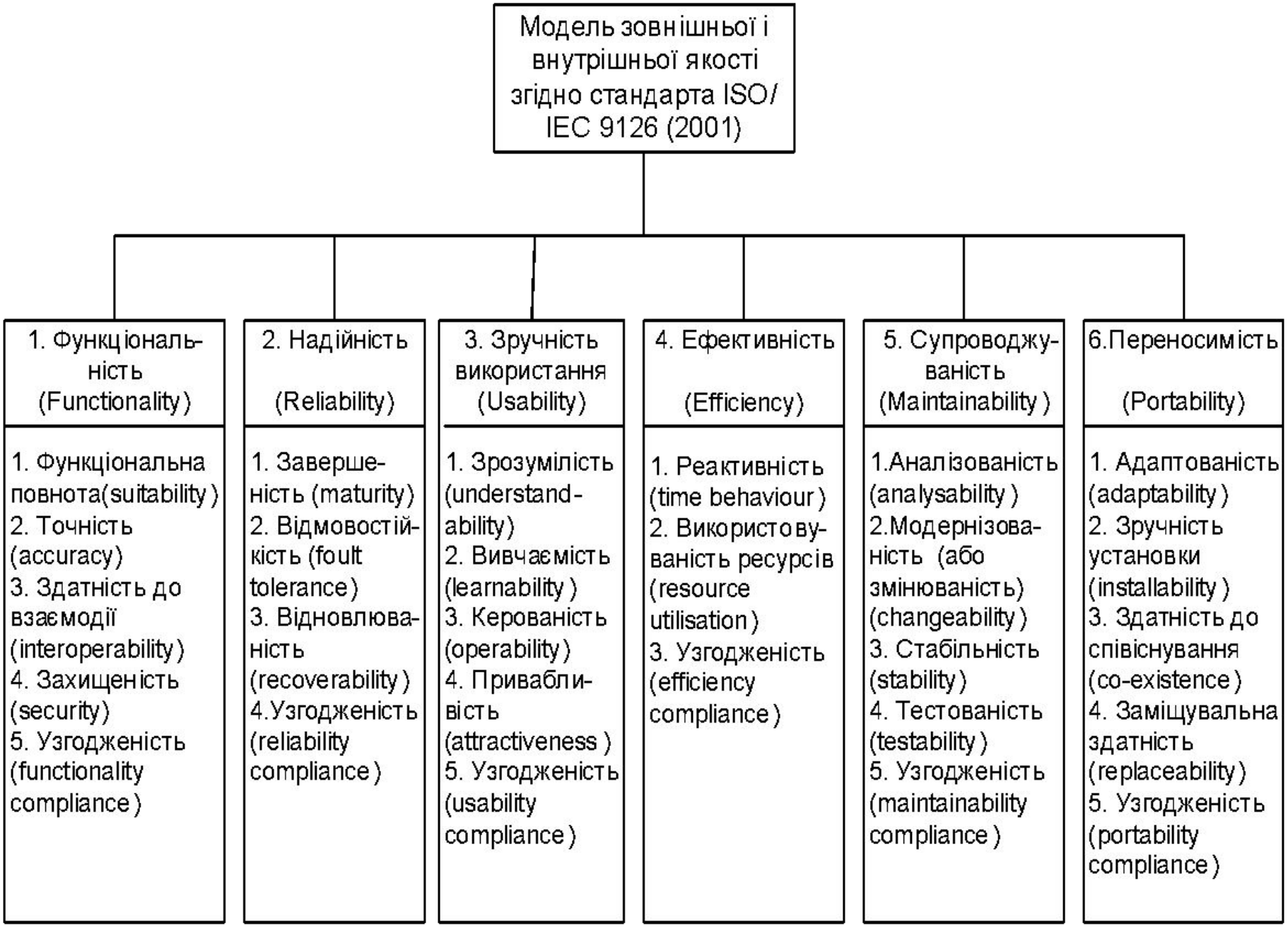
Приклади зовнішніх метрик для характеристики надійність – число усунутих дефектів, середній час між відмовами.

Внутрішні метрики дають можливість оцінити якість ПС безпосередньо по її властивостях (об'єм коду, число цикломатичності тощо).

Внутрішні метрики відображають якість проміжного й кінцевого ПП по тим характеристикам, які визначені в моделі. Ці метрики використовують для верифікації того, що проміжний та кінцевий продукти задовольняють вимогам до внутрішньої якості. Розробка внутрішніх метрик ґрунтується на виконанні вимірювань статичних атрибутів, які визначаються й оцінюються по тексту вихідного коду, а також по графічому представленню потоків керування, чи документам на ПС. Приклади для характеристики надійність – число помилок знайдених при інспекції коду, число усунутих дефектів при інспекції.

Метрики якості у використанні (експлуатаційні метрики) вимірюють ступінь, у якій ПП задовольняє потреби користувачів в ефективності, продуктивному й безпечному рішенні завдань. Ці метрики оцінюють видимі результати експлуатації ПС. Приклади – повнота досягнення цілей користувачів,

точність результатів, продуктивність праці, думка користувачів.


Модель зовнішньої і внутрішньої якості згідно стандарта ISO/ IEC 9126 (2001)
1. Функціональність (Functionality)
1. Функціональна повнота(suitability)
2. Точність (accuracy)
3. Здатність до взаємодії (interoperability)
4. Захищеність (security)
5. Узгодженість (functionality compliance)
2. Надійність (Reliability)
1. Завершеність (maturity)
2. Відмовостійкість (foult tolerance)
3. Відновлюваність (recoverability)
4.Узгодженість (reliability compliance)
3. Зручність використання (Usability)
1. Зрозумілість (understandability)
2. Вивчаємість (learnability)
3. Керованість (operability)
4. Привабливість (attractiveness)
5. Узгодженість (usability compliance)
4. Ефективність (Efficiency)
1. Реактивність (time behaviour)
2. Використовуваність ресурсів (resource utilisation)
3. Узгодженість (efficiency compliance)
5. Супроводжуваність (Maintainability)
1.Аналізованість (analysability)
2.Модернізованість (або змінюваність) (changeability)
3. Стабільність (stability)
4. Тестованість (testability)
5. Узгодженість (maintainability compliance)
6.Переносимість (Portability)
1. Адаптованість (adaptability)
2. Зручність установки (installability)
3. Здатність до співіснування (co-existence)
4. Заміщувальна здатність (replaceability)
5. Узгодженість (portability compliance)

Функціональність (functionality) – властивість ПС виконувати функції у відповідності встановленим і очікуваним потребам при використанні у вказаних умовах. *Містить підхарактеристики:*
- функціональна повнота (suitability, іноді – функціональна придатність) – властивість ПС надавати належну множину функцій для вирішення специфікованих задач і досягнення цілей користувача;
- точність (accuracy) – властивість ПС забезпечувати правильні й погоджені результати чи впливи з необхідним ступенем точності;
- здатність до взаємодії (interoperability) – властивість ПС взаємодіяти з однією чи більш специфікованими системами;
- захищеність (security) – здатність ПС забезпечувати захист інформації від несанкціонованого доступу осіб або систем на читання або модифікацію, та доступність інформації для осіб і систем, що володіють правами доступу;
-узгодженість **функціональності** *із нормативними документами (compliance).*

Надійність (reliability) – властивість ПС зберігати рівень функціонування при роботі в зазначених умовах. *Підхарактеристики:*
- завершеність (maturity, іноді –безвідмовність) – властивість ПС уникати відмов через дефекти, що містяться в ній;
- відмовостійкість(fault tolerance, або стійкість до відмов) – властивість ПС підтримувати встановлений рівень функціонування в умовах прояву дефектів, а також помилок у даних або порушень специфікованого інтерфейсу;
- відновлювальність (recoverability) – властивість ПС відновлювати функціонування на заданому рівні та відновлювати пошкоджені програми й дані;
- узгодженість характеристики **reliability** *із нормативними документами.*

Зручність використання (usability) – властивість ПС бути зрозумілою, освоюваною, зручною і привабливою для користувачів, при використанні в зазначених умовах. *Підхарактеристики:*

- зрозумілість (understandability) – властивість ПС, яка дає користувачу зрозуміти чи дійсно ПС може задовольнити його потреби, та як вона може використовуватися для вирішення визначених задач і які умови її використання;

- вивчаємість (learnability, іноді – *освоюваність)* – властивість ПС, що забезпечує користувачів можливістю освоїти прийоми її застосування;

- керованість (operability) – властивість ПС, яка надає можливість користувачу керувати або контролювати її дії;

- привабливість (attractiveness) – властивість ПС, яка забезпечує її привабливість для користувача;

- узгодженість характеристики ***usability*** *із нормативними документами.*

Ефективність (efficiency) – властивість ПС забезпечувати раціональне використання виділених ресурсів при роботі у встановлених умовах.

Містить такі підхарактеристики:

- реактивність (time behavior, або ефективність у часі) – властивість ПС забезпечувати належний час відповіді (відгуку) та обробки завдань, а також рівень пропускної спроможності при виконанні функцій ПС у встановлених умовах застосування;

- використовуваність ресурсів (resource utilization) – властивість ПС використовувати належні ресурси в потрібні періоди часу при виконанні своїх функцій у встановлених умовах застосування;

- узгодженість характеристики ***efficiency*** *з нормативними документами.*

Супроводжуваність (maintainability) – властивість ПС забезпечувати можливість її ефективної модифікації. Модифікація може включати коригування,вдосконалення або адаптацію ПС до змін середовища, вимог або функціональних специфікацій.

- аналізованість (analyzability) – властивість ПС, яка дає можливість діагностувати її недоліки або причини відмов, а також ідентифікувати частини для модифікації; модернізованість (changeability, або змінюваність) – властивість ПС, що дає можливість виконувати встановлені види модифікацій;

- *стабільність (stability)* – властивість ПС мінімізувати несподівані ефекти модифікацій;

- *тестованість (testability)* – властивість сприяти перевірці модифікованого ПЗ;

- *узгодженість* характеристики ***maintainability*** із нормативними документами.

Переносність (portability) – властивість ПС бути перенесеною з одного середовища до іншого.

- адаптованість (adaptability) – властивість ПС адаптуватися для застосування в різних специфікованих середовищах без використання дій і засобів відмінних від тих, які спеціально призначені для цих цілей;

- зручність установки (installability, іноді – настроюваність) – властивість ПС, що зумовлює її здатність до інсталяції в специфікованому середовищі;

- здатність до співіснування (co-existence, або сумісність) – властивість ПС співіснувати з іншими незалежними ПС у загальному середовищі, розділяючи загальні ресурси;

- заміщувальна здатність (replaceability) – властивість ПС бути використаною замість інших специфікованих ПС у середовищі їхнього застосування;

-*узгодженість* характеристики ***portability*** *з нормативними документами.*

Для кожної ***підхарактеристики якості*** існує декілька різних метрик якості, а тому в моделі якості запроваджують **атрибути**. Кожний атрибут вимірюють за допомогою відповідної метрики.

У формулі (1): ***Q*** – множина взаємозалежних характеристик (властивостей) якості, *Ci* – *i*-та характеристика якості, *Sij* – *j*-та підхарактеристика *i*-ї характеристики якості, *Aijk* – *k*-й атрибут *j*-ї підхарактеристики *i*-ї характеристики якості, *Mijk* – метрика *k*-го атрибута *j*-ї підхарактеристики *i*-ї характеристики якості, *Wijk* – коефіцієнт ваги (значимості) *k*-го атрибута *j*-ї підхарактеристики *i*-ї характеристики якості. Крім того: *i, j, k* – індекси, *I* – загальна кількість характеристик якості в моделі якості, *Ji* – загальна кількість підхарактеристик *i*-ї характеристики якості, *Kij* – загальна кількість атрибутів *j*-ї підхарактеристики *i*-ї характеристики якості.

Загальна формула моделі якості

$$Q = \left\{ C_i \left\{ S_{ij} \left\{ A_{ijk} \left(M_{ijk}, W_{ijk} \right) \right\}_{k=1}^{K_{ij}} \right\}_{j=1}^{J_i} \right\}_{i=1}^{I} \qquad \textbf{(1)}$$

Інтегральна формула моделі якості (інтегральний рівень якості)

$$U_q = \sum_{i=1}^{I} W_i \sum_{j=1}^{J_i} W_{ij} \sum_{k=1}^{K_{ij}} W_{ijk} Q_{ijk} \qquad \textbf{(2)}$$

- Після визначення всіх елементів моделі **(1)** можна перейти до випробувань, в ході яких обчислюються фактичні значення атрибутів підхарактеристик якості. Ці значення порівнюються із обмеженнями, заданими у вимогах.
- Якщо отримані фактичні значення показників якості відповідають вимогам, то подальшу оцінку можна провести, використовуючи інтегральний показник якості. Інтегральний показник якості **(2)** можна застосувати для вибору кращого ПЗ із декількох конкуруючих в даній області (галузі), при умові узгодження конфліктуючих атрибутів якості (у формулі **(2)** ***Qijk –*** *відносний показник якості* ***k****-го атрибута,* ***j****-ї підхарактеристики* ***i****-ї характеристики моделі якості*). Але, оскільки в моделі є показники з різними метриками (неперервні числові, бальні, номінальні та інші), необхідно попередньо провести узгодження та нормування метрик, що роблять шляхом уведення шкал для якісних та категорійних критеріїв і заданням вагових множників.
- Таким чином, враховуючи **(1),** інтегральний (узагальнений) рівень якості ПС U_q можна обчислити як середньозважений показник по формулі **(2).**
- Необхідно провести узгодження конфліктуючих атрибутів якості.
- Необхідно провести нормування метрик атрибутів якості.

Фрагмент матрицы конфликтов характеристик качества

Характеристики, на которые оказывается влияние / Характеристики, по которым определяется рост качества		Функциональность				Надежность			Удобство использования				Эффективность	
		Функциональная полнота	Точность	Способность к взаимодействию	Защищенность	Завершенность	Устойчивость к аномалиям	Восстанавливаемость	Понятность	Осваиваемость	Управляемость	Привлекательность	Реактивность	Используемость ресурсов
Функциональность	Функциональная полнота								+		+			
	Точность	+												
	Способность к взаимодействию					-							-	-
	Защищенность			-						-	-		-	
Надежность	Завершенность										+		-	-
	Устойчивость к аномалиям					-					+		-	-
	Восстанавливаемость			-										-

Приклад обчислення рівня якості *атрибута*, який вимірюється відповідно до метрики *functional implementation completeness (закінченість, завершеність функціональної реалізації)* підхарактеристики *suitability* (*функціональна повнота*) характеристики *функціональність*.

Це буде перший атрибут першої підхарактеристики першої характеристики якості моделі, тобто: Q_{111}. Міра цього атрибута вимірюється відповідно до метрики *X*. Рівень якості цього атрибута будемо обчислювати по формулі:

$$Q_{111} = X = 1 - A / B \qquad \textbf{(3)}$$

A – кількість нереалізованих функцій; *B* – кількість функцій, котрі мають бути реалізовані відповідно до специфікацій та описів ПС.

Приклад для ПрО “Діяльність поліклініки”

Інтервали значення для метрики ***Fault density*** визначаємо на основі припущення, що максимальним допустимим значенням є 1 помилка на 100 рядків коду. Кількість в 100 рядків визначена на основі середнього розміру методу класу в вихідному коді ПС. Значення метрики ***Available co-existence*** оцінюємо, припускаючи, що максимальною допустимою кількістю помилок при використанні стороннього ПЗ може бути 1 помилка за всю тривалість робочого дня (8 годин). В кращому випадку можна розраховувати на 1 помилку за робочий тиждень(5 робочих днів). Метрику ***Help frequency*** унормовуємо виходячи з припущення, що в оптимальному випадку для використання невідомої функції системи користувачу достатньо 2 звернень до довідки, для ознайомлення та перевірки результатів, а відповідно трьохкратне перевищення даного значення є неприпустимим (6–“2”, 5 –“3”,4 –“3”,3 –“4”, 2–“5”).

Відносний показник якості визначається зі співвідношення $Q_{ijk} = \frac{P_{ijk}}{PB_{ijk}}$

Pijk - рівень якості *ijk*-го елемента показника якості; *PBijk* - базове значення *ijk*-го елемента. Але оцінювання системи на основі різнорідних числових значень рівня якості атрибутів не буде достовірним, тому доцільно їх нормувати та привести значення метрик Fault density, Help frequency, Available co-existence до вигляду:

$$Q_{ijk} = 1 - \frac{P_{ijk}}{PB_{ijk}}; Q_{ijk} \in [0,1] \qquad \textbf{(4)}$$

- 2.Надійність. 2.1. Завершеність.
- Fault density $P_{211} = \frac{A}{B} = \frac{91}{215} = 0{,}42[Добре]; Q_{211} = 1 - \frac{0{,}42}{1{,}0} = 0{,}58$

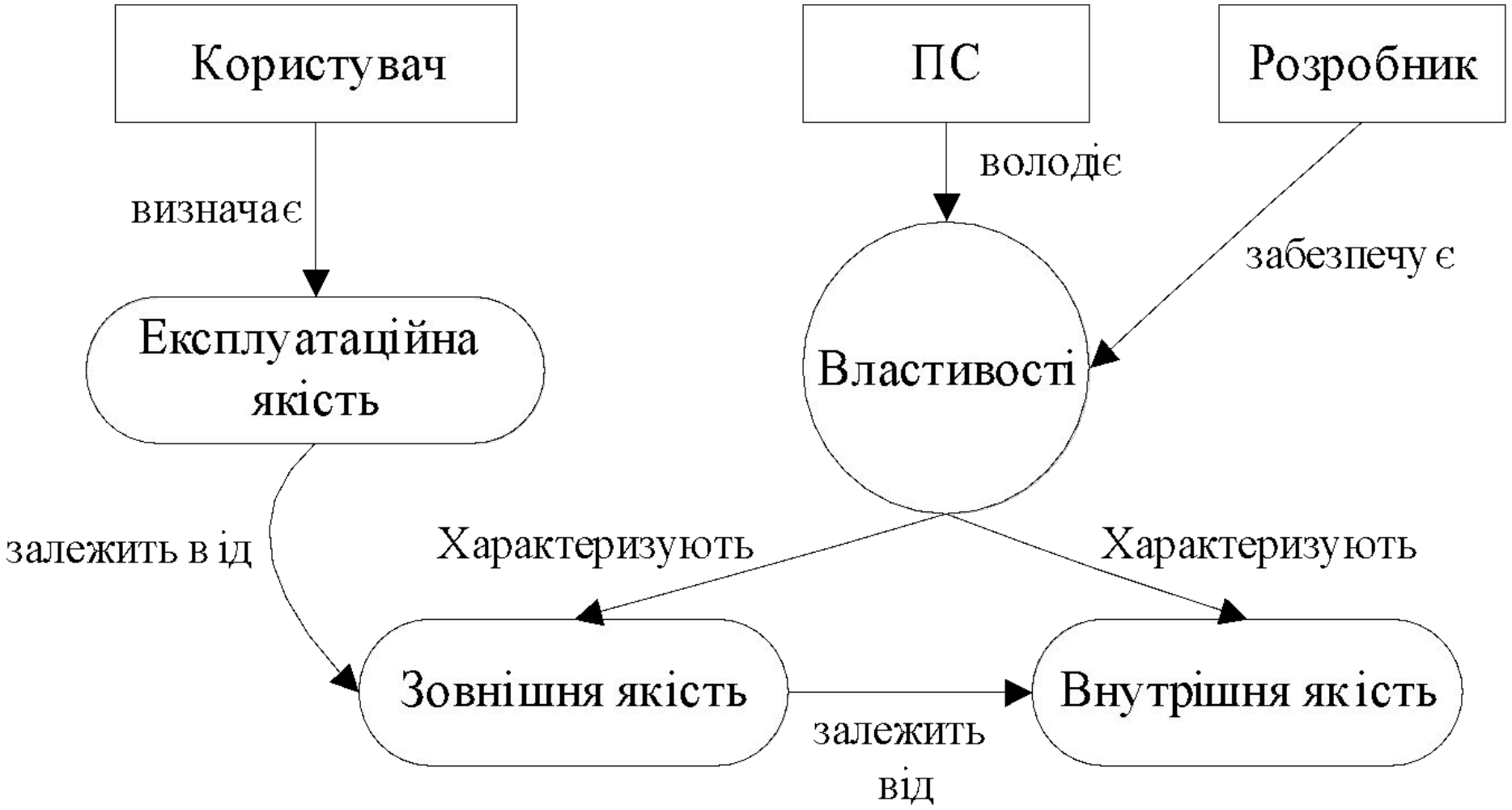
  - 3.Зручність використання. 3.2. Вивчаємість.
  - Help frequency

$$P_{321} = A = 5[Задовільно]; Q_{321} = 1 - \frac{5}{6} = 0{,}16$$

- 4.Переносимість. 4.3.Здатність до співіснування.
  - Available co-existence

$$P_{421} = \frac{A}{T} = \frac{4}{5} = 0{,}8[Задовільно]; Q_{421} = 1 - \frac{0{,}8}{1{,}0} = 0{,}2$$

Взаємозв’язок понять при оцінюванні якості



Експлуатаційна якість

На відміну від моделі зовнішньої і внутрішньої якості, для опису експлуатаційної якості служить однорівнева модель, яка розподіляє атрибути експлуатаційної якості по чотирьом характеристикам.

Характеристики експлуатаційної якості:

- *Результативність (іноді – ефективність, effectiveness)* – ступінь у якій користувач досягає заданих цілей по точності й повноті вирішення задач у встановленому контексті використання ПС.
- *Продуктивність (productivity)* – ступінь у якій витрачаються ресурси на досягнення користувачем заданої ефективності у встановленому контексті використання ПС.
- *Безпечність (safety)* – рівень ризику завдання матеріальних і моральних збитків, тобто шкоди здоров'ю людей, бізнесу, майну, навколишньому середовищу при використанні ПС у встановленому контексті.
- *Задоволеність (satisfaction)* – ступінь у якій користувач задоволений ПС у певному контексті її використання.

Таблица 2.4. **Характеристики эксплуатационного качества ПС**

№	Наименование характеристики	Определение характеристики
1	**Результативность** (effectiveness)	Степень, в которой пользователями достигаются заданные цели по точности и полноте решения задач в установленном контексте использования ПС
2	**Продуктивность** (productivity)	Степень, в которой расходуются ресурсы на достижение пользователями заданной эффективности решения задач в установленном контексте использования ПС
3	**Безопасность** (safety)	Уровень риска нанесения ущерба (материального и морального) - вреда здоровью людей, бизнесу, имуществу, окружающей среде при использовании ПС в установленном контексте
4	**Удовлетворенность** (satisfaction)	Степень, в которой пользователь удовлетворен ПС в определенном контексте ее использования. На степень удовлетворенности влияют отношение пользователя к свойствам ПС, представленным характеристиками качества для внешних и внутренних атрибутов ПС, а также его отношение к эффективности, продуктивности и безопасности ПС

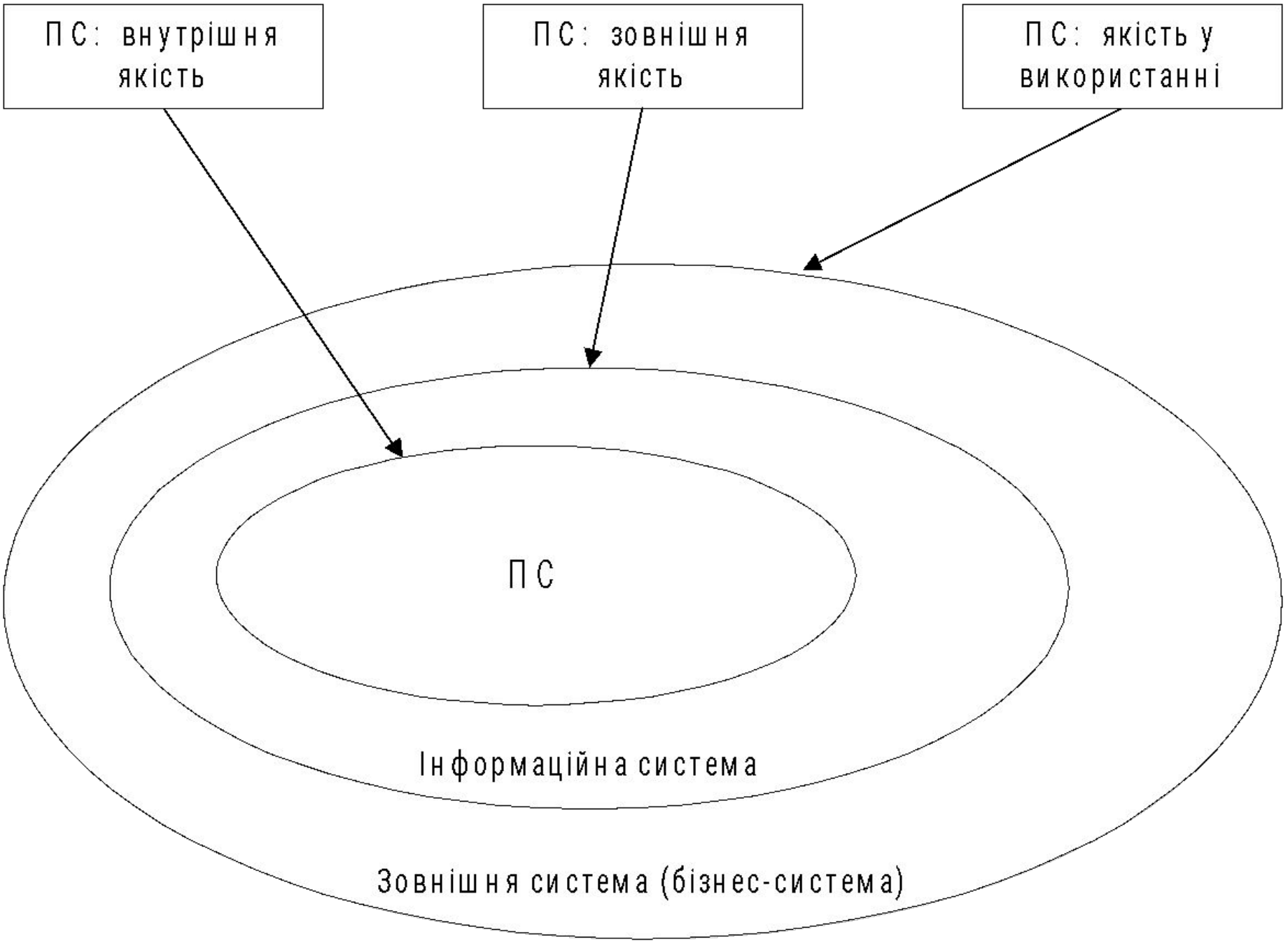
Метрики в узагальненій моделі якості

Метрики в узагальненій моделі класифікуються по підхарактеристикам внутрішньої і зовнішньої якості, а також по характеристикам експлуатаційної якості, що описані у 2-й, 3-й і 4-й частинах стандарту **ISO/IEC 9126**.

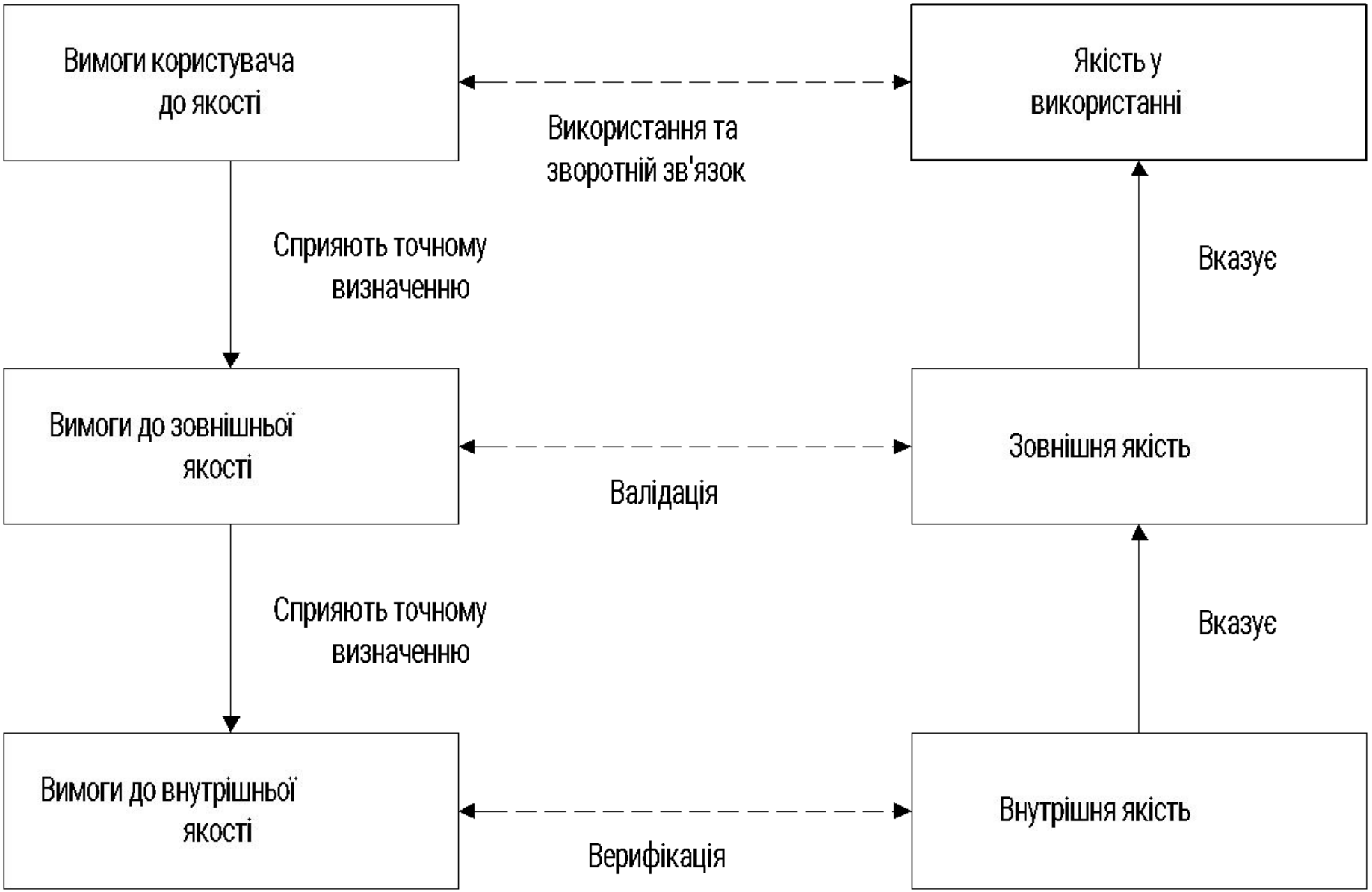
Опис метрик виконаний за єдиною схемою із зазначенням наступної інформації:

- *ім'я метрики.* Відповідні метрики внутрішньої і зовнішньої якості мають однакові імена;
- *призначення метрики.* Формулюється у вигляді питання, на яке дає відповідь метрика, що застосовується;
- *метод застосування*. Містить правила одержання даних і схему їхнього використання в метриці;
- *формула й елементи даних*. Вказується формула обчислення й пояснюються елементи даних, які беруть в них участь;
- *інтерпретація виміряних даних*. Діапазон значень і найліпше значення;
- *тип шкали метрики*, *тип міри*;
- *вихідні дані.* Джерело даних, що використовується у вимірі.
- *процес ЖЦ*. Вказується процес у якому рекомендується застосування цієї метрики для виміру відповідних мір атрибутів якості.

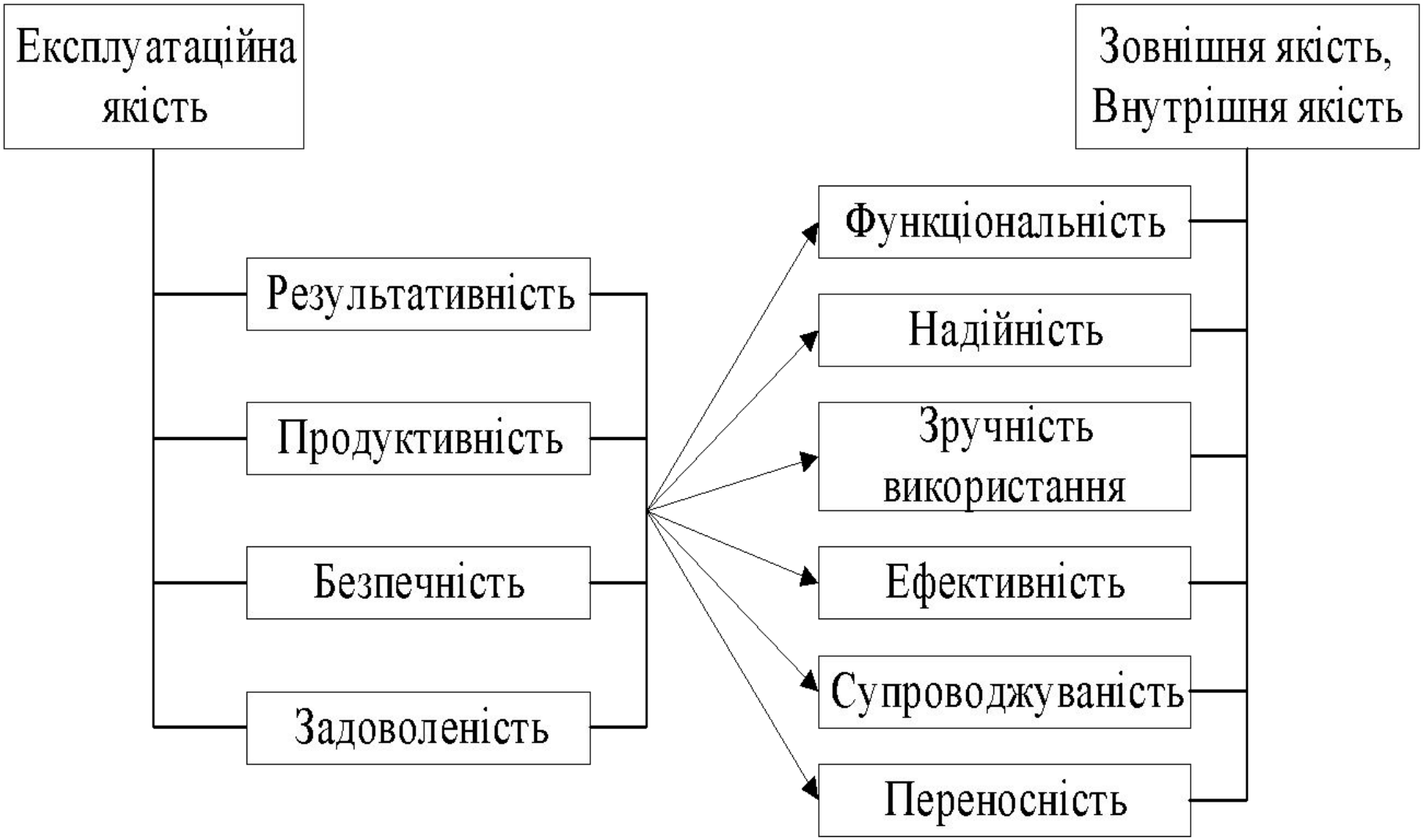
Зв’язок між системами



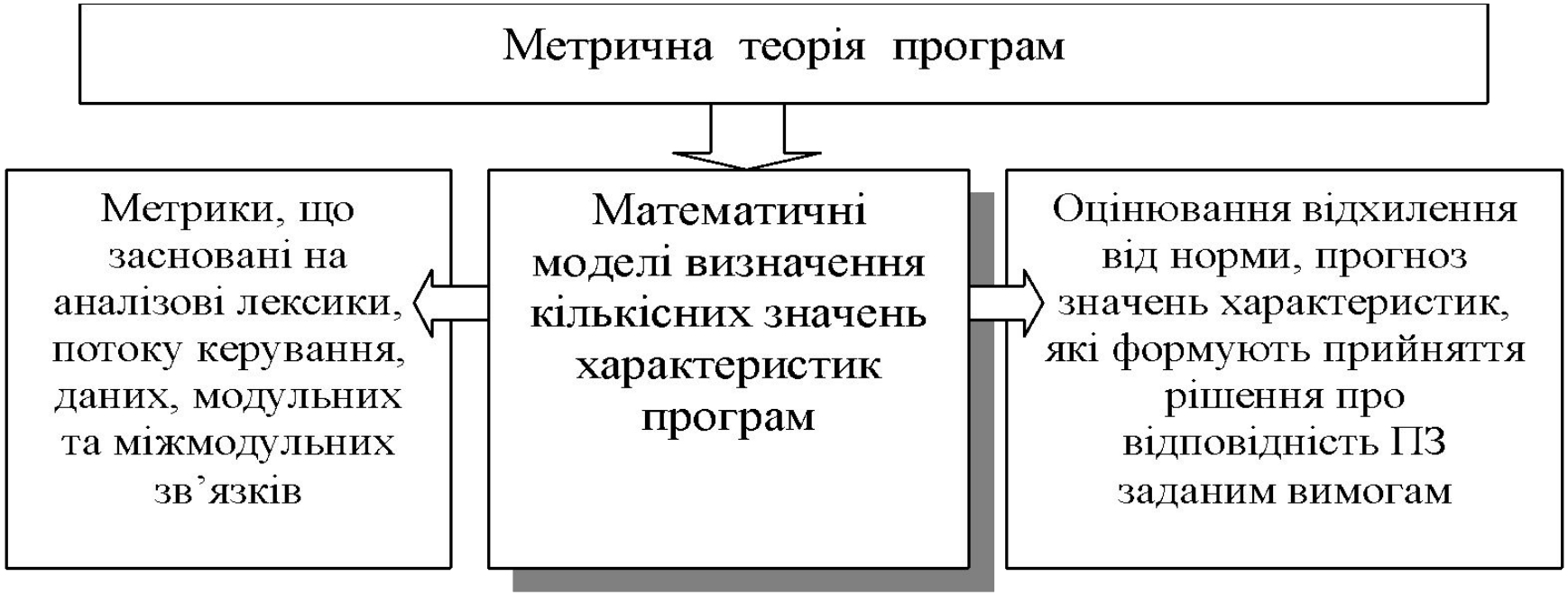
Якість у життєвому циклі ПС



Взаємозвязок між моделями якості стандарту ISO/IEC 9126



Оцінювання внутрішньої якості програмної ситеми на основі стандарту ISO/IEC 9126-3. Застосування системи метрик М.Холстеда та Т.Маккейба



Метрики Холстеда відображають лексичний підхід до вимірювання характеристик ПЗ, заснований на вимірних властивостях алгоритмів. Властивості будь-якого опису алгоритму (або програми) можуть бути виміряні чи обчислені на основі таких метричних характеристик:

n1 – кількість різних (відмінних один від одного) операторів програми;

n2 – кількість різних (відмінних один від одного) операндів програми;

N1 – загальна кількість операторів програми;

N2 – загальна кількість операндів програми.

На цій основі Холстед визначає такі метрики:

Словник програми (в умовних одиницях) $n = n1+n2$ (5.1)

Довжина реалізації (в умовних одиницях) $N = N1+N2$ (5.2)

Довжина програми (в умовних одиницях) $\tilde{N} = (n1 \times \log_2 n1)+(n2\times\log_2 n2)$ (5.3)

Обсяг програми (у бітах) $V = (N1+N2) \times \log_2(n1+n2)$ (5.4)

Потенційний обсяг програми $V^* = (n2^*+2) \times \log_2(n2^*+2)$, (5.5)

де ***n2* –*** загальна кількість вхідних і вихідних параметрів.

Припускаємо, що в програмах, ідеальних з погляду економії витрат пам'яті, 1) оператори та операнди не повторюються; 2) всі операнди є або вхідними, або вихідними параметрами; 3) для запису тексту програми досить двох операторів (опису заголовка процедури-функції і присвоювання). Використаємо вираз (5.4) для обсягу програми, за умови, що $N1=n1=2$ і $N2=n2=n2^*$.

Рівень програми (в умовних одиницях) $L = V^*/ V \cong (2\times n2)/(n1\times N2)$, (5.6)

якщо $n2^*=2$.

Рівень мови $\lambda=L\times V^*$, (5.7)

Інтелектуальний зміст програми (в умовних одиницях)

$I=L\times V\cong(2\times n2/n1\times N2)\times(N1+N2)\times\log_2(n1+n2)$ (5.8)

Робота з програмування (в умовних одиницях)

$E = V / L = V^2 / V^*$ (5.9)

Час на програмування (в умовних одиницях)

$T = E/S$, (5.10)

або $T\cong(n1\times N2\times\log_2 n\times(n1\times\log_2 n1+n2\times\log_2 n2))/(2\times n2\times S)$, (5.11)

де ***S*** – число Страуда (***5<S<20***).

Число Страуда* S** визначається як число «страудовських моментів» за секунду. «Страудовський момент» – це час, необхідний людині для виконання елементарного розрізнення об'єктів, подібно розрізненню кадрів фільму. Страуд винайшов, що людина здатна розрізняти від 5 до 20 об'єктів за секунду. *Потенційний обсяг програми* є мірою мінімально необхідного обсягу програми із заданим словником. Потенційний обсяг не залежить від мови реалізації. При перекладі програми з однієї мови на іншу потенційний обсяг не змінюється, але дійсний обсяг V чи збільшується, чи зменшується залежно від мови реалізації. Використовуючи вираз потенційного обсягу програми, отримані наступні метрики: *Рівень програми* L≤1 характеризує ефективність реалізації алгоритму щодо витрат пам'яті. Тільки для найбільш компактної форми реалізації алгоритму (V=V*) рівень програми дорівнює 1. Усім іншим варіантам реалізації відповідають значення L<1. *Рівень мови* λ – це коефіцієнт пропорційності зміни обсягу програми при перекладі з однієї мови на іншу так, що добуток рівня програми на потенційний обсяг залишається незмінним. *Інтелектуальний зміст програми* характеризує міру «сказаного» у програмі, чи її «інформативність». Інтелектуальний зміст (рівень) програми корелює з потенційним обсягом (**I≈V) і не залежить від мови реалізації. *Робота з програмування* (рівняння розумової роботи) характеризує величину розумової роботи, зв'язаної із написанням програмного коду. Рівняння роботи дає підставу для розбиття програми на складові частини – модулі. Модульність знижує обсяг програмування. Найбільш продуктивною є ситуація, за якої для отримання одного результату використовується не більше **5** об'єктів. У прикладному сенсі цей результат називають гіпотезою про **«6 об'єктів»**. Для визначення кількості модулів **M** у програмі Холстед рекомендує використовувати вираз:

$M = n2^*/6$, (5.12)

де n2* – загальна кількість вхідних і вихідних змінних у програмі.

З рівняння роботи отримаємо таке *рівняння помилок:*

$B = L \times E / E0$, (5.13)

де B – кількість помилок у програмі, E0 – середня кількість елементарних відмінностей між помилками програмування.

Використовуючи перетворене рівняння роботи:

$E = (V^*)^3 / \lambda^2$, (5.14)

а також значення рівня англійської мови (λ=2,16), як аналог мови програмування, і гіпотезу про «шість об'єктів» ідеальної за витратами пам'яті програми (n1=n1*=2, n2=n2*=6), Холстед вивів таке рівняння для прогнозу кількості помилок у програмі:

$B = E^{2/3} / 3000$,(5.15) або $B = V / 3000$, (5.16)

де V – обсяг програми (5.4).

Якщо розрахунки довжини програми і довжини реалізації відрізняються більш ніж на *десять відсотків*, то це свідчить про можливу наявність у програмі таких *6 класів недосконалостей:*

1). Наявність послідовності операторів, що доповнюють один одного до того ж самого операнда, наприклад, A+C–A. 2). Наявність неоднозначних операндів, наприклад, A=D і A=C.

3). Наявність операндів-синонімів, наприклад, A=B и T=B. 4). Наявність загальних підвиразів: (A+B)×C + D×(A+B). 5). Непотрібне присвоювання, наприклад C=A+B, якщо змінна C використовується в програмі тільки один раз.

6). Наявність виразів, що не подані в згорнутому вигляді як добуток множників, наприклад X×X+2×X×Y+Y×Y не подається як (X+Y)×(X+Y).

Добуток рівня програми на обсяг є постійною величиною, що дорівнює потенційному обсягу реалізації даного алгоритму: $L \times V = V^* = const$.

Якщо мова не змінюється, а змінюється тільки алгоритм, то для будь-якої мови добуток потенційного обсягу на рівень програми залишається сталою величиною і дорівнює рівню мови: $L \times V^* = \lambda = const$.

Холстед ввел следующие основные обозначения:

n_1 - количество разных операторов
n_2 - количество разных операндов
N_1 - общее количество операторов
N_2 - общее количество операндов.

Основные метрики:

Размер (длина) программы: $N=N_1+N_2$

Размер словаря: $n=n_1+n_2$

Предсказанный размер: $\hat{N} = n_1 \bullet \log_2 n_1 + n_2 \bullet \log_2 n_2$

Объем программы: $V = N \bullet \log_2 n$

Дополнительные метрики:

Трудоемкость разработки:

$$E = \frac{n_1 \bullet N_2 \bullet N \bullet \log_2 n}{2 \bullet n_2}$$

Время, требуемое на разработку: $T=E/18$ секунд.

Количество ошибок: $B = V / 3000$

Рис. 4.11. **Метрики Холстеда**

Метрика Маккейба

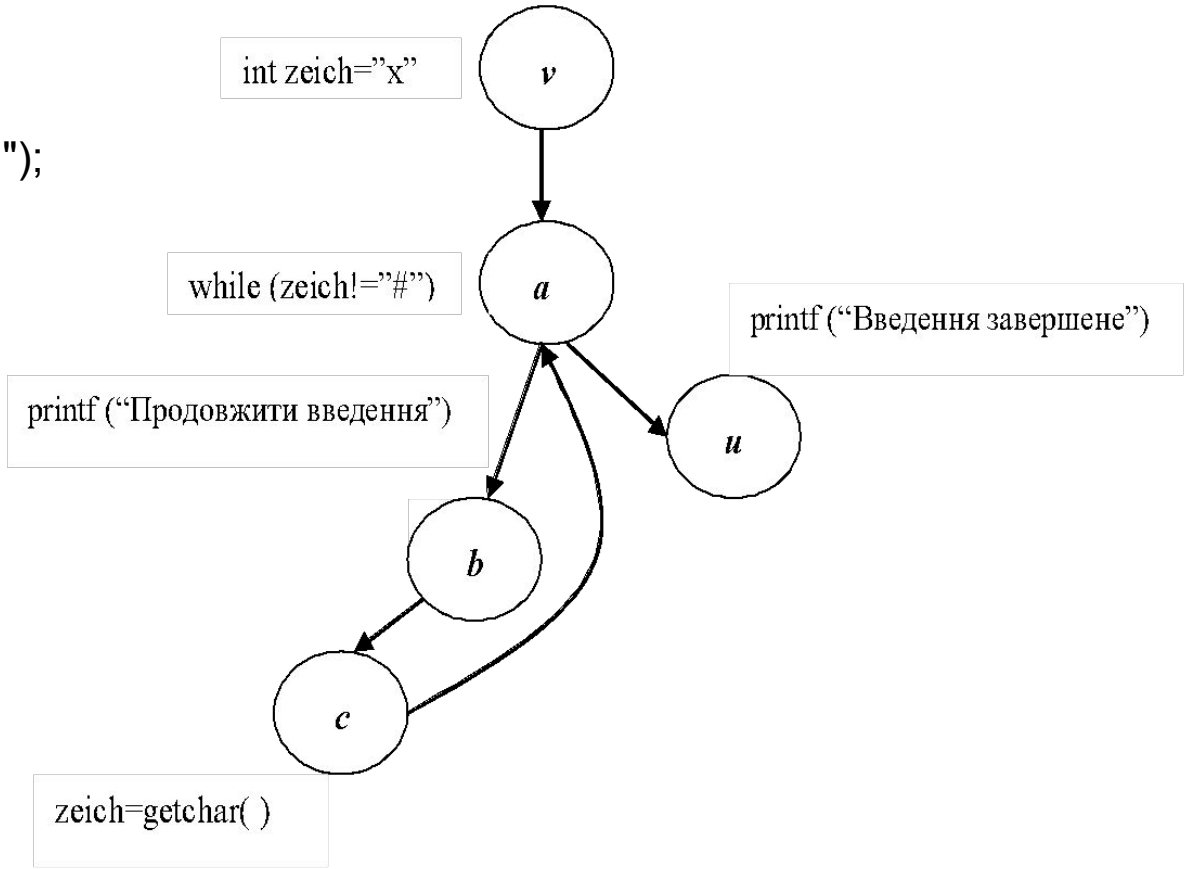
Метрика Маккейба ґрунтується на аналізі потоку передавання керування від одного оператора до іншого. Це дозволяє, на відміну від метрик Холстеда, врахувати логіку програми при оцінюванні її складності.

Програма (алгоритм, специфікація) має бути подана у вигляді управляючого орієнтованого графу G = (V, E) із V вершинами і E дугами, де вершини відповідають операторам, а дуги – переходам від одного оператора до іншого.

Граф, що описує програму у вигляді вершин-операторів і дуг-переходів, називають ***графом керування*** або *управляючим графом* програми.

Приклад.

```
main()
 { int zeich = 'x';
    while(zeich != '#')
     {  printf(" Продовжити введення ");
        zeich = getchar();
     }
    printf("Введення завершене "); }
```



Метрика Маккейба є цикломатичним числом графу передач керування програми і визначається виразом: $M = m – n + 2$, (5.17)

де ***m*** – кількість ребер графу; ***n*** – кількість вершин графу. Величину M, розраховану за формулою (5.17), називають ***цикломатичним числом Маккейба***.

Теоретичною базою визначення цикломатичного числа Маккейба є теорія графів. У теорії графів цикломатичне число орієнтованого графа визначається виразом:

$$M = m – n + 2p \quad (5.18)$$

де ***m*** – кількість ребер; ***n*** – кількість вершин; ***p*** – кількість компонентів зв'язності графу.

Кількість компонентів зв'язності ***p*** можна розглядати як мінімально необхідну кількість ребер, які потрібно додати до графу, щоб зробити його повнозв'язним. Повнозв'язним вважають граф, у якого існує шлях з будь-якої вершини графу в будь-яку іншу вершину графу. Для графів керування програм повнозв'язність забезпечується додаванням однієї фіктивної дуги з кінцевої вершини в початкову, тобто у розглядуваному прикладі із вершини **u** у вершину **v**. Тому вважаємо, що для будь-якого графу керування програми кількість компонентів зв'язності дорівнює одиниці. Підстановка ***p=1*** у формулу (5.18) дає цикломатичне число Маккейба (5.17). Визначимо цикломатичне число Маккейба для графу керування програми, зображеного на рисунку. Кількість ребер графу дорівнює п'яти, кількість вершин теж дорівнює п'яти, тому цикломатичне число дорівнює:

$M = 5 – 5 + 2 = 2$.

Фізичний зміст цикломатического числа. Контур – це площина, обмежена циклічним шляхом, у якій початкова і кінцева вершина графу збігаються. Кожному контуру відповідає шлях, що обмежує його, який веде з початкової вершини графа в кінцеву. Цикломатичне число визначає кількість незалежних контурів у повнозв'язному графі і, як наслідок, кількість різних шляхів, що ведуть з початкової вершини в кінцеву. У практичному аспекті цикломатичне число є мірою складності програми і визначає кількість тестів, достатніх для тестування за критерієм покриття всіх гілок програми.

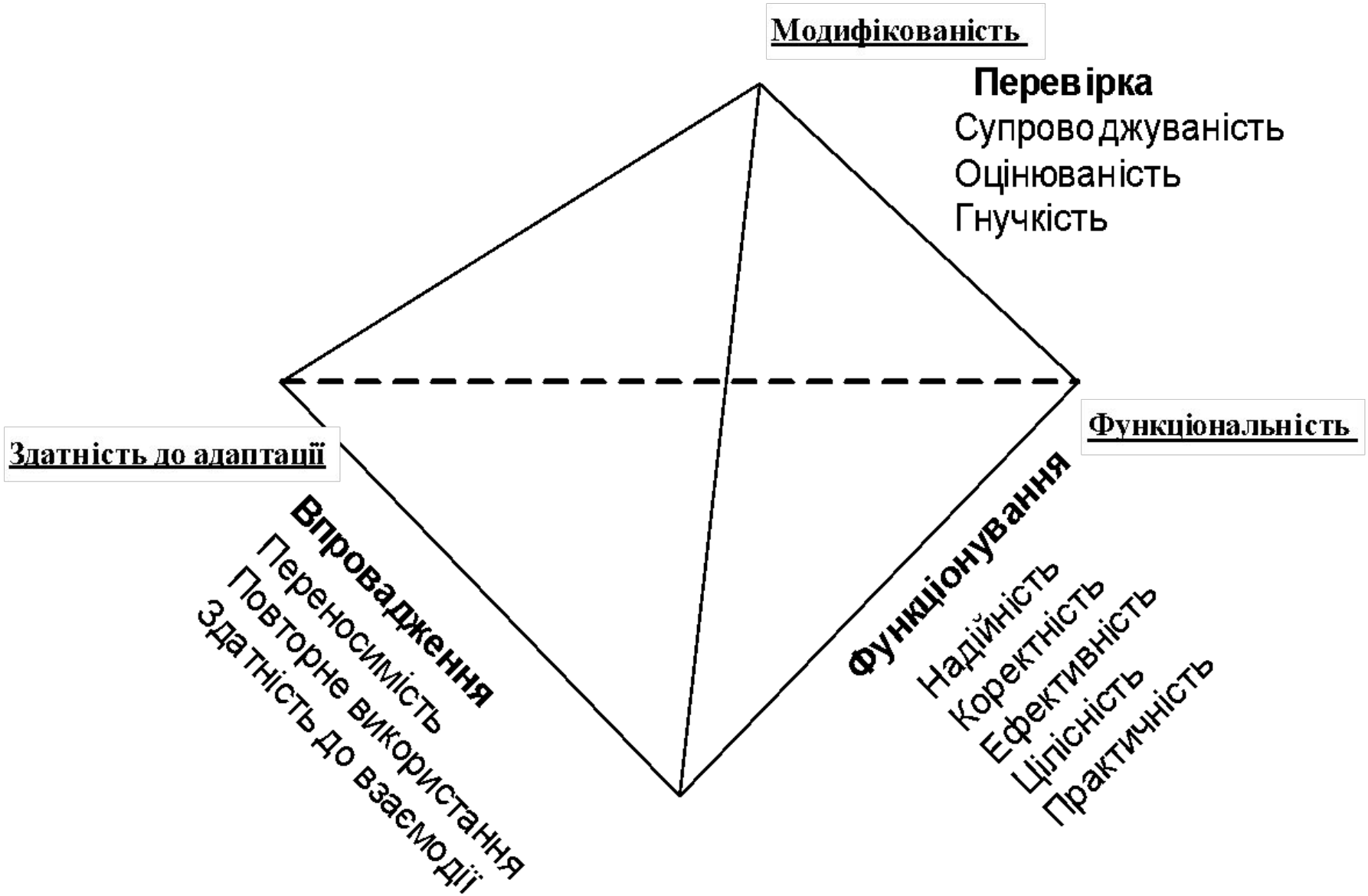
Моделі якості ПС

Модель МакКола (1977). Три аспекта якості: функціональність, модифікованість, здатність до адаптації . Аспекти поділяються на *11 факторів якості* і далі на *23 критерія якості*, які не обов'язково пов'язані тільки з одним *фактором якості* (ієрархічно-мережева структура). Критерії формують цілі проекту. Третя група характеристик якості - *20 метрик*, на основі яких здійснюється кількісне оцінювання показників із двох інших груп. **Модель Боема (1978).** Перша ієрархічна модель, на основі якої згодом була сформована модель ISO/IEC 9126 (1991). Початкові рівні ієрархії схожі з МакКолом, але в моделі Боема модифікованість і здатність до адаптації віднесені не до верхнього рівня моделі, а до проміжного та нижнього. По Боему *якість ПЗ* складається з двох аспектів - *функціональності (людський фактор, ефективність, надійність, мобільність) і зручності експлуатації (оцінюваність, зрозумілість, модифікованість).* Наведені 7 характеристик якості пов'язані з *12 основними критеріями*, які знаходяться рівнем нижче. Більшість із критеріїв якості пов'язані один з одним, через що можуть виникати конфлікти та неоднозначності при оцінці якості програмного продукту (ПП). **Модель Ватса (1987)** вдосконалює модель МакКола, а **Модель Дьюча і Вілліса (1988)** задає *15 користувацьких та 27 технічних атрибутів якості*, які можуть пов'язуватися з *факторами якості, або критеріями якості* відповідно.

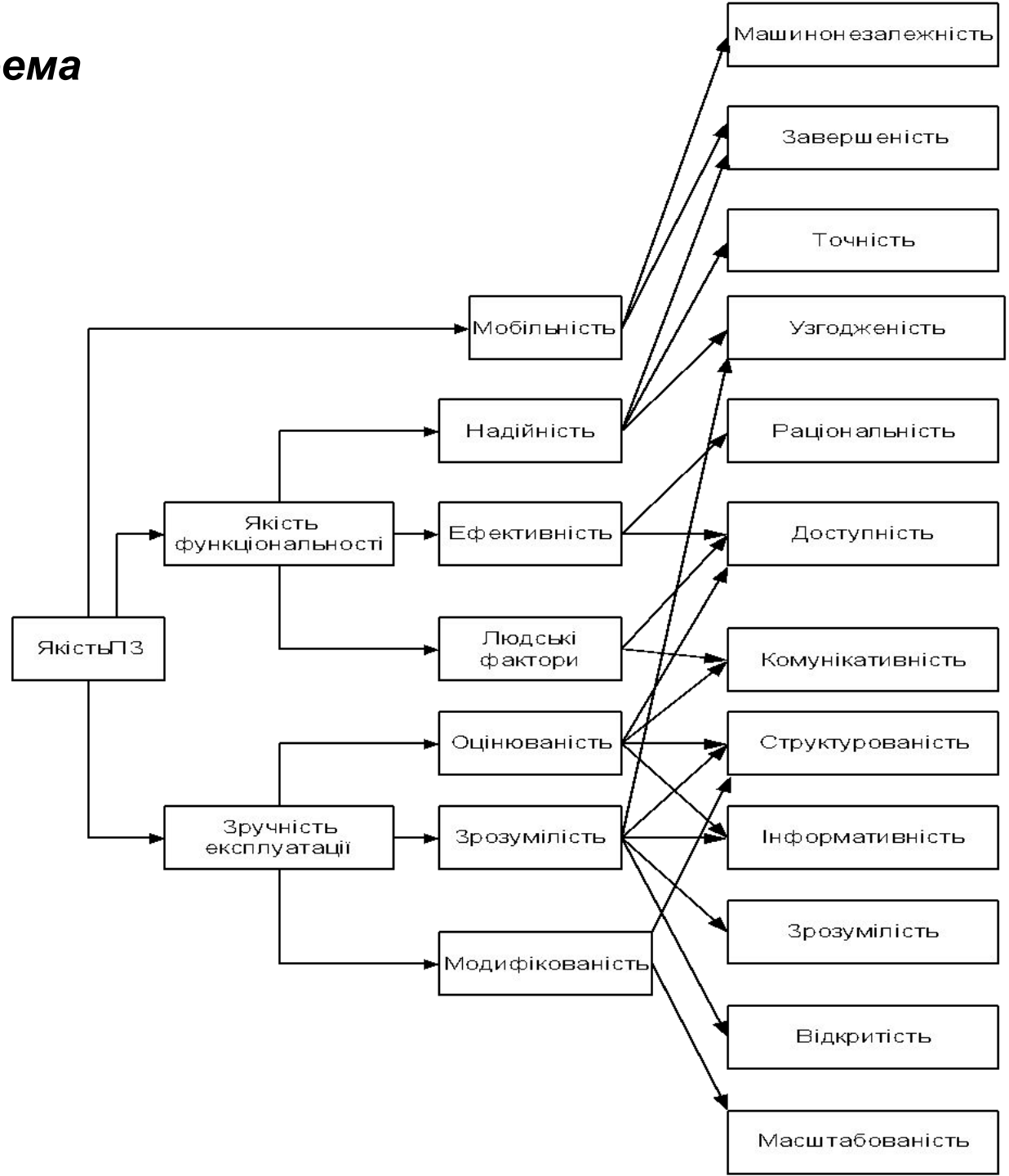
Модель ISO/IEC 9126 (1991) складається з *6 характеристик якості (функціональність, надійність, ефективність, зручність використання, супроводжуваність, переносимість).* Модель є строго ієрархічною у відмінності від попередніх моделей. Недоліком моделі є те, що не всі підхарактеристики якості можуть бути визначені чисельно. Це питання вирішено у стандарті *ISO/IEC 9126 (2001)* шляхом впровадження відповідних множин внутрішніх та зовнішніх метрик.

Альтернативою класичним ієрархічним моделям є **модель Джилба**, яка теж базується на ієрархії характеристик та підхарактеристик якості, але *характеристики ресурсів проекту розробки* також увійшли у модель. Крім того, модель **Джилба** пов'язана з *еволюційною моделлю розробки ПС*. *Характеристики якості – працездатність, готовність, адаптованість і зручність використання, а ресурсів – час, гроші, люди та інструменти.* У **1996р. Дромей** запропонував гнучку модель якості, яка дозволяє враховувати особливості ПС, що розробляється. Це досягається за допомогою аналізу *архітектури компонентів ПС та властивостей тих компонентів*, які впливають на *атрибути якості кінцевого ПП*. Встановлюється зв' язок між атрибутами якості та властивосттями компонентів і оцінюється правильність моделі (слабкі місця усуваються). Біля **2000р. Боєм** запропонував своєрідну, але в наш час дуже популярну та корисну модель COCOMO (COnstructive COst MOdel). Модель орієнтована на досягнення цілей якості у сполученнні з оцінюванням вартості та трудозатратності виготовлення ПС і є дуже важливою як для аналітиків та менеджерів, які керують проектом, так і для розробників ПП, дозволяючи визначити "ціну помилки".

Модель МакКола



Модель Боема



Результати порівняльного аналізу моделей якості ПС

Для проведення порівняльного аналізу моделей якості ПС визначимо категорії критеріїв. Моделі якості повинні містити сукупність характеристик якості, набори метрик для проведення оцінювання реалізованих у ПС властивостей та забезпечувати здатність до адаптації на стадіях ЖЦ.

Для досягнення задоволення користувацьких потреб розробники серійного ПС використовують моделі якості **FURPS** (functionality, usability, realibility, perfomance, service) та **CUPRIMDSO** (capability, usability, perfomance, realibility, installability, maintainability, documentation / information, service, overall).

Структура моделі **FURPS** наслідувана з моделей МакКола та Боема. При цьому здійснити комунікацію вимог якості та провести їх оцінювання складно, оскільки необхідно враховувати пріоритетність атрибутів різних характеристик моделі FURPS. Модель **CUPRIMDSO** є більш повною по відношенню до FURPS, оскільки має ширший набір характеристик якості, що дозволяє точніше виявити рівень задоволення реалізованих у ПС властивостей.

Спільним для цих двох моделей є те, що вони дозволяють виражати якість ПС як з точки зору кінцевого користувача так і збоку самої ПС. Тому якість ПС, виходячи із застосування цих моделей, можна поділити на три категорії: якість продукту; внутрішня якість процесів; якість підтримки.

У результати аналізу і порівняння включимо і **модель Дромея**. Вона практично не відрізняється від моделі якості стандарту ISO 9126, який був прийнятий в 1991р. Перевагою цієї моделі є те, що її використання дозволяє контролювати якість програмного продукту на етапах ЖЦ. Однак характеристики якості моделі Дромея є не уніфікованими, не стандартизованими і не набули широкого застосування.

Вимоги до моделей	Модель МакКола	Модель Боема	Модель FURPS	Модель CUPRI MDSO	Модель Дромея	Моделі ISO 9126
Повнота визначення характеристик якості	+/-	+/-	+/-	+/-	+	+
Наявність метрик для кількісного вираження якості	-	+	+	+	+	+
Формалізованість та узгодженість із стандартами	-	-	-	+	-	+
Здатність до трансформації	-	-	+	-	+	+
Можливість комунікації вимог на стадіях ЖЦ	-	-	-	-	-	+
Ідентифікація вимог до якості	-	-	+	+	+	+
Структурованість та адаптація до предметної області	+/-	+/-	+	+	+	+
Ідентифікація цілей проекту	-	-	+	+	+	+

Розробка стандартів здійснюється міжнародними й національними органами стандартизації:

ISO – International Standard Organization
IEC – International Electrotechnical Commission
EOQ – European Organization for Quality
WSSN – World Standards Services Network
ANSI – American National Standards Institute
BSI – British Standards Institution
IEEE – Institute of Electrical and Electronics Engineering
NASA – National Aeronautics Space Administration
NIST – National Institute of Standards and Technology
SCC – Standards Council of Canada
ДСТУ – Державний комітет по стандартизації, метрології та сертифікації України
ГОСТ – Государственный комитет РФ по стандартизации и метрологии

Міжнародні стандарти в області інформаційних технологій (**IT**) розробляються *об'єднаним технічним комітетом* **JTC1 (Joint Technical Committee №1)** "Information Technology", заснованим ISO та IEC. Він включає 36 підкомітетів SC (**SubCommittie**), а за розробку міжнародних стандартів по програмній інженерії (Software Engineering – SE) відповідає робочий підкомітет ISO/IEC **SC7** "Software and System Engineering". Для роботи над проектами стандартів у підкомітеті SC7 створені робочі групи (WG, Working Group).

Базовими стандартами по керуванню якістю продукту і системах управління якістю ОР є міжнародні стандарти серії ISO 9000. Стандарт ISO 9000-3 регламентує управління якістю під час розробки, поставки та супроводу ПЗ.

Базові стандарти по керуванню якістю ПЗ

ДСТУ ISO 9000-3 – Стандарти з управління якістю та забезпечення якості. Частина 3. Настанови щодо застосування ДСТУ ISO 9001-95 під час розроблення, постачання та супроводження програмних засобів.

Основні діючі стандарти в області інженерії якості

ДСТУ ISO 9000-2001. Системи управління якістю. Основні положення та словник.
ДСТУ ISO 9001-2001. Системи управління якістю. Вимоги.
ДСТУ ISO 9004-2001. Системи управління якістю. Настанови щодо поліпшення діяльності.
ГОСТ 34.602-89. Информационная технология – Комплекс стандартов на АС. Техническое задание на создание автоматизированой системы.
ГОСТ РФ 28195-89. Оценка качества программных средств. Общие положения.

Основні діючі стандарти в області інженерії якості (продовження)

ISO/IEC 12119 (1994) – IT – Пакети програм, вимоги до якості й тестування.
ISO/IEC 12207 (1995) – IT – Software life cycle processes.
Група стандартів з оцінки ПП – ISO/IEC 14598 :
- 14598-1 (1999) – IT – Оцінка продукту – Частина 1. Загальний огляд.
- 14598-2 (2000) – SE – Оцінка продукту – Частина 2. Планування й керування.
- 14598-3 (2000) – SE – Оцінка продукту – Частина 3. Процес для розробників.
- 14598-4 (2000) – SE – Оцінка продукту – Частина 4. Процес для замовників.
- 14598-5 (1999) – IT – Оцінка продукту – Частина 5. Процес для оцінювачів.
- 14598-6 (1999) – IT – Оцінка продукту – Частина 6. Документація модулів оцінювання.

ISO/IEC 14764 (1999) – IT – Software maintenance.
ISO/IEC 9126-1 (2001) Software engineering – Product quality – Part 1: Quality model.
ISO/IEC TR 9126-2 (2003) Software engineering – Product quality – Part 2: External metrics.
ISO/IEC TR 9126-3 (2003) Software engineering – Product quality – Part 3: Internal metrics.
ISO/IEC TR 9126-4 (2004) Software engineering – Product quality – Part 4: Quality in use metrics.
IEEE std. 610.12 (1990) – Глосарій термінології по програмній інженерії.
IEEE std. 12207.0 (1996) – Процеси ЖЦ ПЗ.
IEEE std. 12207.1 (1997) – Дані ЖЦ ПЗ.
IEEE std. 730 (1998) – Плани забезпечення гарантії якості ПЗ (підходи відповідно до SQA).
IEEE std. 830 (1993) – Recommended Practice for Software Requirements Specifications.
NASA std. 8719.13A (1997). Безпека фукціонування ПЗ.
ДСТУ 2844 (1994). Програмні засоби ЕОМ. Забезпечення якості. Терміни та визначення.
ДСТУ 2850 (1994). Програмні засоби ЕОМ. Показники та методи оцінювання якості.
ДСТУ 2851 (1994). Програмні засоби ЕОМ. Документування результатів випробовувань.
ДСТУ 2853 (1994). Програмні засоби ЕОМ. Підготовка та проведення випробувань.
ДСТУ 2462 (1994). Сертифікація. Основні поняття. Терміни та визначення.
ДСТУ 3918 (1999) – Інформаційні технології. Процеси ЖЦ ПЗ.

Група стандартів *SPICE* - *S*oftware *P*rocess *I*mprovement and *C*apability d*E*termination – WG10 “Оцінювання процесу” – “Process Assessment”

- *ДСТУ ISO/IEC 15504-1:2002.* Інформаційні технології – Оцінювання процесів життєвого циклу програмних засобів. Частина 1. Концепції та вступна настанова.
- *ДСТУ ISO/IEC 15504-2:2002.* Інформаційні технології – Оцінювання процесів життєвого циклу програмних засобів. Частина 2. Еталонна модель процесів та потужності процесу.
- *ДСТУ ISO/IEC 15504-3:2002.* Інформаційні технології – Оцінювання процесів життєвого циклу програмних засобів. Частина 3. Виконання оцінювання.
- *ДСТУ ISO/IEC 15504-4:2002.* Інформаційні технології – Оцінювання процесів життєвого циклу програмних засобів. Частина 4. Настанови з виконання оцінювання.
- *ДСТУ ISO/IEC 15504-5:2002.* Інформаційні технології – Оцінювання процесів життєвого циклу програмних засобів. Частина 5. Модель оцінювання та настанови щодо показників.
- *ДСТУ ISO/IEC 15504-6:2003.* Інформаційні технології – Оцінювання процесів життєвого циклу програмних засобів. Частина 6. Настанови з визначення компетенції оцінювачів.
- *ДСТУ ISO/IEC 15504-7:2003.* Інформаційні технології – Оцінювання процесів життєвого циклу програмних засобів. Частина 7. Настанови з удосконалення процесу.
- *ДСТУ ISO/IEC 15504-8:2003.* Інформаційні технології – Оцінювання процесів життєвого циклу програмних засобів. Частина 8. Настанови з визначання потужності процесу постачальника.
- *ДСТУ ISO/IEC 15504-9:2003.* Інформаційні технології – Оцінювання процесів життєвого циклу програмних засобів. Частина 9. Словник термінів.

Робочі групи підкомітету SC7 “Software and System Engineering”

WG2: Документація системних ПП;
WG4: Інструменти і CASE-засоби;
WG6: Оцінювання ПП і метрики;
WG7: Управління життєвим циклом;
WG8: Підтримка процесів ЖЦ;
WG9: Забезпечення гарантій і цілісність ПЗ;
WG10: Оцінювання і контроль процесів ЖЦ;
WG11: Подання й визначення даних в інженерії ПЗ;
WG12: Функціональні вимірювання ПЗ;
WG19: Мови моделювання, метадані, схема та компоненти відкритої розподіленої обробки;
WG20: Ядро знань в області програмної інженерії;
WG21: Процес управління наробітками в галузі ПЗ;
WG23: Управління якістю систем;
WG24: Життєві цикли ПЗ для малих підприємств.

SQuaRE ***(Software Quality Requirements and Evaluation) - ISO/IEC9126 + ISO/IEC14598.*** **SQuaRE** ***містить 5 груп стандартів:***

2500n – Управління якістю ПП:
- огляд, структура і довідник по SQuaRE (25000);
- планування й керування оцінкою якості продукту.

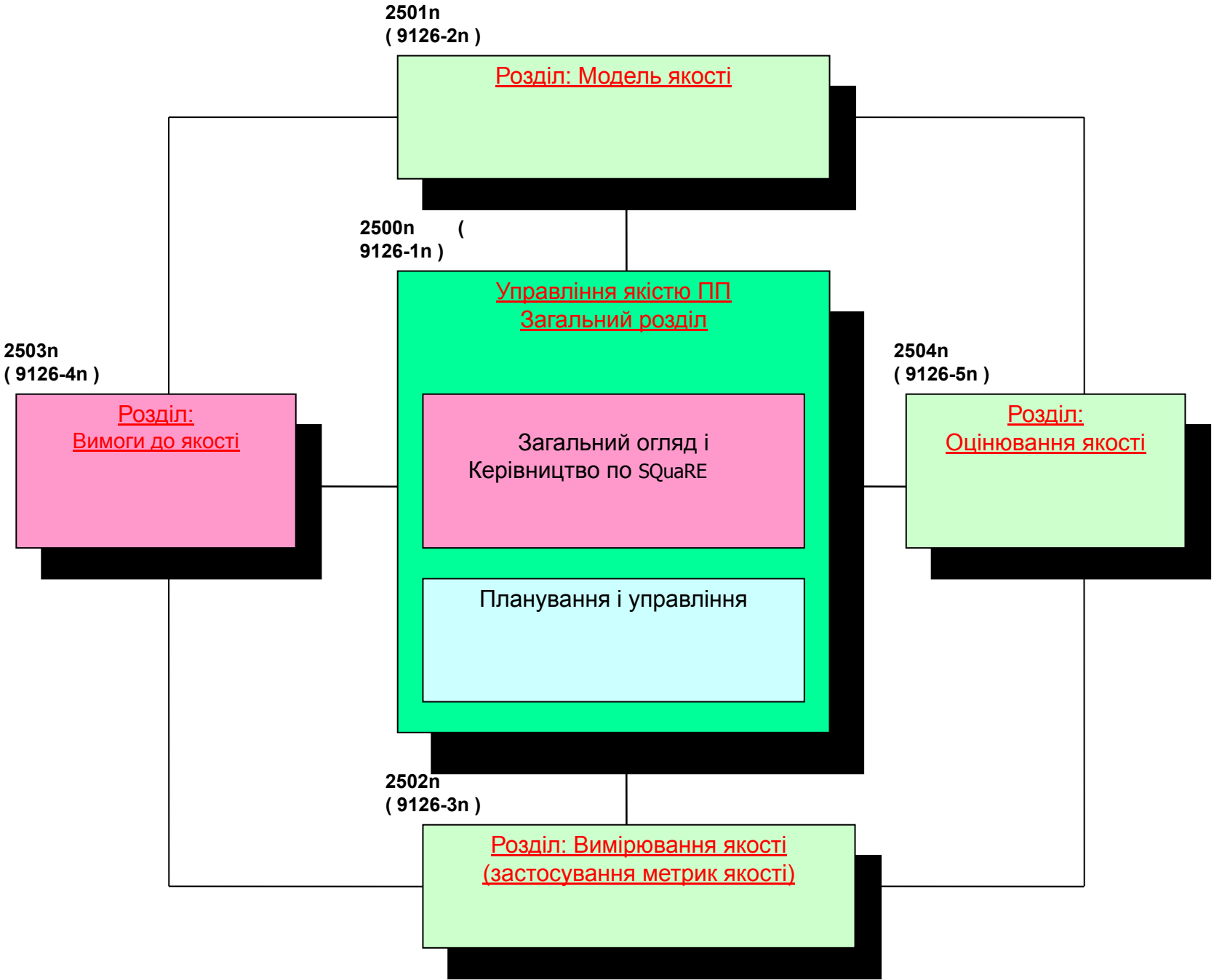
2501n – Модель якості:
- модель якості ПЗ (25010);
- модель якості даних (структури даних ПС, БД) (25012);
- вказівки по застосуванню моделей якості.

2502n – Вимірювання якості:
- еталонна модель вимірювання й вказівки по застосуванню;
- базові і похідні метрики якості (примітиви вимірювання);
- внутрішні метрики якості;
- зовнішні метрики якості;
- метрики якості у використанні;

2503n – Вимоги до якості:
- вимоги до якості ПЗ;
- вказівки по визначенню якості.

2504n – Оцінювання якості продукту:
- огляд процесів оцінювання;
- процес для розробників;
- процес для споживачів (замовників);
- процес для оцінювачів.
- документування модулів оцінювання.

Архітектура стандартів SQuaRE



Зв’язок між поточними стандартами і SQuaRE

Поточні розділи	SQuaRE
9126 : Якість продукту	**2500n: Управління якістю**
-1 Модель якості	0:Загальний огляд і керівництво
-2 Зовнішні метрики	1: Планування і управління
-3 Внутрішні метрики	**2501n: Модель якості**
-4 Якість у використанні	0:Модель якості і керівництво
	2502n: Вимірювання якості
Нові пропозиції	0 : Загальні вимоги і керівництво
Керівництво по використанню 9126&14598	1: Базові метрики
Основні метрики (модель)	2: Внутрішні метрики
Вимоги до якості	3: Зовнішні метрики
	4: Метрики якості у використанні
14598 : Оцінювання продукту	5: Документація модулів оцінки
-1 Загальний огляд	**2503n: Вимоги до якості**
-2 Планування і управління	0: Вимоги до якості і керівництво
-3 Процес для Розробників	**2504n: Оцінювання якості**
-4 Процес для Покупців	0: Огляд оцінки якості і керівництво
-5 Процес для Оцінювачів	1: Процес для Розробників
-6 Документація модулів оцінки	2: Процес для Споживачів
	3: Процес для Оцінювачів