**УДК 683.1**

**О.Б. Вовк**
Національний університет “Львівська політехніка”,
кафедра “Інформаційні системи та мережі”

# АНАЛІЗ ТА ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ (ПОНЯТТЯ, ТЕРМІНИ, ОЗНАЧЕННЯ)



***The main terms and notions of quality metrology used in computer technology area are formulized and considered. The definitions are proposed in intelligent information systems context. The present state of the problem is considered and previous researches are reviewed.***

***Сформульовано та розглянуто основні метрологічні терміни та поняття якості в галузі комп’ютерних технологій. Визначення цих понять дано в контексті застосування їх до інтелектуальних інформаційних систем. Розглянуто сучасний стан проблеми та попередні дослідження у цій сфері.***

## ВСТУП

Сьогодні створення високоякісного програмного забезпечення є однією з найважливіших завдань розвитку науки та виробництва. Від того, наскільки вдало зроблено програмне забезпечення системи, залежить в кінцевому результаті її життєздатність.

Проте, в зв’язку з тим, що тільки деякі суттєві властивості програмного забезпечення можуть бути виміряні безпосередньо і оцінені кількісними показниками, якість комп’ютерних технологій набуває першочергового значення.

## ОЦІНКА ЯКОСТІ: ОГЛЯД ЛІТЕРАТУРИ ТА СУЧАСНИЙ СТАН ПРОБЛЕМИ

Розглянемо сучасний стан проблеми оцінки якості програмного забезпечення, а також попередні дослідження у цій галузі досліджень.

Імовірно, перша організована спроба розробки методів оцінки якості програмного забезпечення була почата Рубай і Гартвиком [12]. Метод, запропонований у цій роботі, полягав у тому, щоб визначити набір “властивостей” програми та їхні “вимірники”. Бажаним властивостям давався словесний опис, а як вимірники виступали математичні вирази, у яких аргументами були параметри, що прямо або побічно відбивали конкретні властивості програми. Зокрема, називалися такі властивості, як “$A_г$ – правильність виконання математичних обчислень”, “$A_5$ – зрозумілість програми”, “$A_6$ – простота внесення змін”, потім проводився докладніший аналіз для визначення більш конкретних характеристик цих властивостей, які можна було б вимірювати і тим самим виявляти, у якому ступені та або інша властивість характерна для даної програми (за 100-бальною шкалою). І хоча така деталізація була виконана для кожної із запропонованих авторами характеристик якості, лише деяким з них вдалося поставити у відповідність виміряні показники; крім того, не наводилося жодних прикладів практичного застосування розробленого методу.

У більш пізніх дослідженнях Брауна і Липова [13] було сформульовано систему мір якості програмного забезпечення і показано її використання у межах керованого експерименту для оцінки двох машинних програм, написаних незалежно одна від однієї на

основі однієї і тієї ж специфікації вимог. Однак це дослідження обмежувалося невеликим набором характеристик, головним чином тими, котрі відповідали властивостям $A_5$ і $A_6$, зазначеним вище. Основна увага у цьому дослідженні приділялась аналізу двох різних підходів до забезпечення високої якості програми, що в одному випадку розроблялася програмістом, що намагався написати максимально ефективний варіант, а в іншому – програмістом, що намагався забезпечити простоту програми. Основні результати проведеного дослідження такі: в "ефективній" програмі було виявлено у 10 разів більше помилок, ніж у простій (за 1000 тестових прогонів у тому і в іншому випадку); значення показників якості "простої" програми виявилися значно вищими. Отже, можна зробити висновок, що розглянуті характеристики є справжніми індикаторами функціональної надійності, принаймні у межах цього дослідження.

Одночасно й іншими авторами було визнано необхідність встановлення показників якості програмного забезпечення і використання їх для оцінки конкретних властивостей програм. Вулф [14] запропонував й описав для цієї мети сім важливих і досить незалежних характеристик:

- зручність експлуатації або модифікація;
- життєздатність, чіткість, ефективність;
- вартість;
- мобільність і ступінь обліку людських факторів.

Абернаті й ін. встановили ряд характеристик операційних систем і проаналізували деякі компроміси, що їх об'єднують. Вайнберг [15] здійснив експерименти, у яких декільком групам програмістів давалося однакове завдання, але встановлювалися різні критерії успіху (швидкість розробки програми, кількість операторів, обсяг використовуваної оперативної пам'яті, чіткість вихідних даних і ясність програми). Кожна характеристика досягала свого найвищого значення тоді, коли вона була критерієм успіху групи, що розробляла програму. Усе більше фахівців починають визнавати важливість якості програмних засобів, і намагаються впливати на якісні характеристики програмного забезпечення створенням «практичної допомоги з програмування». Мабуть, найкращим зразком такої літератури є книга Кернигена і Плоужера, присвячена стилеві програмування [16]. Слід зазначити також серію науково-дослідних звітів корпорації CIRAD [17] про роботи у галузі експлуатаційної надійності програмного забезпечення; у цих звітах можна знайти багато корисного з цієї проблеми, зокрема практичні рекомендації з методів програмування і метрик експлуатаційної надійності. Серед методів програмування розглядаються, наприклад, концептуальне групування, модульна побудова; як характеристики програмного забезпечення, що визначають зручність його експлуатації, називають свідомість, однаковість, компактність, зрозумілість та інші. Огляд різних сучасних підходів до забезпечення мобільності програмного забезпечення, таких, як моделювання однієї ЕОМ на іншій, емуляція, інтерпретація, самозавантаження програм, використання мов.

Зроблено певні кроки у напрямку визнання важливості критерію надійності при проектуванні систем програмного забезпечення. З'явилися матеріали, присвячені проблемам створення метрики програмного забезпечення й аналізові його якості. Так, наприклад, книга Майерса стосується питань надійності програмного забезпечення [18], у якій увага зосереджується на методах конструювання високонадійних програм і способах оцінки надійності програмного забезпечення за результатами спостереження за процесами налагодження програм і усунення помилок. Розглядаються й інші показники якості

програмного забезпечення, що стосуються стилю й мови програмування. У працях Холстеда, що стосуються основ теорії програмного забезпечення [19], підводиться підсумок успішних досліджень, спрямованих на встановлення й експериментальну перевірку загальної міри складності програмного забезпечення як функції числа різних операторів і операндів та частоти їх використання.

Особливо слід зазначити роботу Гилба, в якій розглядається метрика програмного забезпечення [20]. Для неї характерним є широке охоплення показників якості програмного забезпечення, а також виразно підкреслюється важливість їхнього використання в практичних цілях на основі формалізованих методів оцінки (по можливості, кількісних).

## ОСНОВНІ МЕТРОЛОГІЧНІ ТЕРМІНИ ТА ПОНЯТТЯ

Згідно з [1] сформулюємо визначення основних понять, які необхідні для подальшого розуміння процесу визначення якості програмного продукту.

**Означення 1.** Метрологія – наука про вимірювання, яка містить як теоретичні, так і практичні аспекти вимірювань у всіх галузях науки і техніки.

Головним поняттям в системі оцінки якості загалом і програмного продукту зокрема, є поняття вимірювання.

**Означення 2.** Вимірювання – відображення фізичних величин їхніми значеннями за допомогою експерименту та обчислень із застосуванням спеціальних технічних засобів.

У контексті програмного забезпечення вимірювання – множина однорідних значень, які відіграють роль індексів. Для певного вимірювання можуть бути визначені певні ієрархії значень. Це дозволяє здійснювати агрегацію показників.[21]

Найбільш раціональний спососіб дій з оцінки якості програмного забезпечення сьогодні полягає в тому, щоб розробити певну систему індикації його недоліків і використовувати цю систему для подальшого вдосконалення програмних засобів. Необхідним при цьому є поняття одиниці вимірювання.

**Означення 3.** Одиниця вимірювання – фізична величина певного розміру, прийнята для кількісного відображення однорідних з нею величин.

Через велику кількість користувачів важко визначитись з будь-якою конкретною універсальною мірою якості програмного забезпечення. Тому необхідно ввести поняття єдності вимірювань.

**Означення 4.** Єдність вимірювань – стан вимірювань, за якого їхні результати наводяться у прийнятих одиницях вимірювань, а похибки (відхилення) вимірювань відомі та із заданою ймовірністю не виходять за встановлені межі.

Ефективна методика оцінки якості показників на основі добре продуманих і чітко визначених умов відповідних характеристик за їх важливістю можлива лише за умов дотримання чітко визначеної методики вимірювань.

**Означення 5.** Методика виконання вимірювань – сукупність процедур і правил, виконання яких забезпечує одерження результатів вимірювань з потрібною точністю.

Будь-яке вимірювання неможливо здійснити без спеціальних засобів, а точніше сказати, засобів вимірювання.

**Означення 6.** Засіб вимірювань – технічний засіб, який застосовується під час вимірювань і має нормовані метрологічні характеристики.

Якщо виникає потреба, для точності вимірюваня , а отже, і оцінки якості загалом, необхідно чітко зазначати тип засобу вимірювань.

**Означення 7.** Тип засобу вимірювання – сукупність засобів вимірювання одного і того ж призначення, які мають один і той же принцип дії, однакову конструкцію та створені за однією і тією ж технологією.

Сформулювавши потяття, які є теоретичною основою будь-якого процесу, пов'язаного з визначенням якості, дамо визначення понять, які безпосередньо пов'язані з комп'ютерними технологіями і програмним забезпеченням зокрема.

**Означення 8.** Під програмним забезпеченням (ПЗ) розуміють прикладні та інструментальні засоби, що розроблені на основі комп'ютерних технологій або засоби, що підтримують такі технології на стадії розробки та/або виконання.

У деяких випадках в процесі вимірювання необхідно вводити таке поняття, як верифікація програмного забезпечення.

**Означення 9.** Під верифікацією програмного забезпечення (ПЗ) розуміють перевірку готового продукту чи його проміжних версій (як це прийнято, наприклад, у технології Cleanroom Software Engineering) на відповідність вихідним вимогам. При цьому мається на увазі не тільки тестування самої програми, але й аудит проекту, користувацької і технічної документації тощо [21].

На ринку існує безліч продуктів, що дозволяють автоматизувати процес верифікації. Верифікація інтелектуальних інформаційних систем має багато спільного з тестуванням будь-яких систем з високими вимогами до надійності. Щоб всебічно оцінити якість системи, необхідно застосовувати різні методи технічного та статистичного аналізу. Запропоновано п'ять типів аналізу системи [22]:

1. Аналіз потоку управління.
2. Аналіз використання даних.
3. Аналіз інтерфейсу.
4. Аналіз потоків даних.
5. Аналіз гілок програмного забезпечення.

Але порівнювати якість програмних продуктів "на пальцях" не надто зручно, тому очевидною є доцільність застосування кількісних методів оцінки якості (метрик). Введемо поняття метрики.

**Означення 10.** Метрика (еталон) – засіб вимірювання, що забезпечує відтворення та (або) зберігання одиниці вимірювань одного чи кількох значень, а також передачу розміру цієї одиниці іншим засобам вимірювання.

Сьогодні існує більше ніж тисяча видів метрик, проте теорія і практика програмних метрик продовжують розвиватися.

Оскільки метрику програмного забезпечення не можна вважати всеохоплюючою, загальний результат такої оцінки варто розглядати як набір певної інформації. Дослідженнями якості програмного продукту згідно з [5] з'ясовано, що важливішою є інформація про те, де і як з'являються дефекти програмного забезпечення, ніж про частоту їх появи. Найбільшу цінність тут являють такі автоматичні засоби аналізу якості програмного забезпечення, які не тільки діагностують програму, а і реєструють конкретні недоліки програми.

Незважаючи на численні дослідження програмних метрик (за цією тематикою опубліковано декілька тисяч статей), у цій галузі, як і раніше, залишається багато невирішених питань.

По-перше, технологія виміру якості ще не досягла зрілості. По-друге, відсутні єдині стандарти на метрики. Найбільш популярний стандарт, що належить до виробництва

надійного ПП (Standard Dictionary of Measures to Product Reliable Software інституту IEEE), так і не став загальноприйнятим. У результаті кожен розробник "вимірювальної" системи пропонує свої власні способи оцінки якості і відповідно метрики.

Такі негативні моменти виникають через те, що немає чітко визначеної методики оцінки якості ПЗ. Для її створення необхідно з'ясувати такі три моменти:

- чи можна дати означення характеристикам якості програмного забезпечення (ПЗ), які б можна було виміряти і забезпечити при цьому достатньо незалежну оцінку якості;
- наскільки точно можна виміряти якість ПЗ загалом і його окремі характеристики;
- як інформацію про характеристики якості ПЗ використати для вдосконалення процесу його експлуатації надалі.

Складність полягає в тому, що існуючи показники якості ПЗ, як правило, неадекватно відображають ті чи інші їх властивості, що визначені потребами і вимогами користувача.

Ефективну методику оцінки якосних показників доцільно робити на основі добре продуманих формулювань відповідних характеристик з врахуванням їх важливості. Оскільки метрику ПЗ не слід вважати абсолютною, загальний результат такої оцінки повинен розглядатись як інформація для проведення аналізу, а не як остаточний висновок.

Три джерела, або три складові частини процесу створення якісного ПЗ – це, по-перше, люди – розробники програм, по-друге, процеси, у вигляді яких організований випуск ПЗ, і, нарешті, технології, відповідно до яких реалізуються ці процеси. Очевидно, що якість продукту на виході технологічного ланцюжка визначається якістю його складових.

Якість одержуваного ПЗ залежить від кваліфікації розробників, склад яких неоднорідний у силу яскраво вираженої спеціалізації. У кожному великому проекті є аналітики, які ставлять задачі, системні програмісти, які готують інструментарій для програмістів прикладних, група тестування ПЗ, технічні фахівці з встановлення і супроводу готових систем. Професійні виробники програмних продуктів давно вже змогли переконатися, що найкращий спосіб поліпшити програму – це удосконалити процеси її створення. За останні десять років розроблено (і реалізовано) безліч концепцій удосконалювання зазначених процесів. Наприклад, американський Software Engineering Institute (SEI) запропонував концепцію "Поліпшення процесів створення ПЗ (Software Process Improvement, SPI), що спирається на статистичні методи контролю технологічних процесів, розроблені в Японії. Пізніше з'явилися й інші концепції: "Наскрізний контроль якості" (Total Quality Management, TQM), "Реінжиніринг ділових процесів" (Business Process Reengineering, BPR), що припускає модернізацію базових бізнесів-процесів організації, "Поступове удосконалювання ділових процесів" (Business Process Improvement, BPI).

В основі всіх цих концепцій лежить загальне розуміння життєвого циклу ПЗ як сукупності фаз, що проходить програмний продукт у процесі свого розвитку. Це:

- вироблення вихідних вимог до ПЗ з боку користувача;
- формулювання загальних вимог до ПЗ з боку розроблювача (системні вимоги);
- проектування архітектури;
- детальна реалізація ПЗ;
- інсталяція ПЗ;
- експлуатація.

Фази можуть перекриватися за часом, а деякі процеси вестися паралельно. Тобто, третя версія програми може знаходитися на етапі проектування, тоді як друга – на стадії інсталяції. Для підтримки життєвого циклу ПЗ [23–25] фірми-розроблювачі організують свою діяльність за декількома ключовими напрямками:

- керування проектом (планування, розподіл ресурсів, контроль виконання і термінів);
- тестування (перевірка відповідності якості готового продукту вихідним вимогам і перевірка функціонування);
- конфігураційний менеджмент (підтримка версій, редакцій, варіантів ПО на рівні вихідного коду, дистрибутивів, документації);
- супровід (встановлення продукту, навчання користувачів, усунення помилок, розвиток функціональних можливостей, постачання upgrade-версій, технічна підтримка).

Користувачу важливо знати, наскільки якісно прорoблені виробничі процеси у тій фірмі, з продукцією якої він має намір працювати.

Проблема полягає в тім, як будувати метрики. В міру росту актуальності програмних метрик на ринку стали з’являтися різні "вимірювальні" програми. Одні з них досліджували характеристики проектів і ПЗ комплексно, інші орієнтувалися на цілком конкретні цілі: аналіз вихідного коду, розмірів і структури окремих модулів тощо.

## МЕТОДОЛОГІЯ ОЦІНКИ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ (ПП)

Розглянемо методологію оцінки якості програмного продукту (ПП). Програми набувають високої якості не стільки в результаті комплексного тестування кінцевого продукту, скільки в процесі його розробки. Якщо методологія створення ПП така, що помилки "виловлюються" на регулярній основі і на всіх стадіях виконання проекту, то на виході "програмного конвеєра" з’являється продукт, у якому практично немає помилок.

Корпорація IBM запропонувала методологію створення складних програмних комплексів, що одержала назву Cleanroom Software Engineering. Вона орієнтована на професіоналів, що бажають удосконалити свої методи розробки ПП.

Cleanroom – це сукупність адміністративно-технологічних процесів, що дозволяють колективам розроблювачів планувати, вимірювати, проектувати, кодувати, тестувати і сертифікувати програмні продукти.

Методологія Cleanroom побудована на трьох концепціях:

- модульному принципі специфікації й проектування;
- математичному доказі правильності застосовуваних алгоритмів;
- використанні статистики за результатами тестування як основи для оцінки надійності програм (сертифікації).

Метод покрокової деталізації з адміністративної точки зору методології Cleanroom полягає у покроковій деталізації проекту, коли кінцева функціональність системи досягається ітераційно. На кожному етапі реалізується визначений рівень функціональних можливостей, що тестується і сертифікується автономно. Такий спосіб розробки має кілька плюсів. З одного боку, видно, як система розвивається, а з іншого – виникають добрі передумови для поліпшення не тільки самого продукту, але і процесів його виробництва – адже на кожному етапі аналізуються джерела виникнення помилок і відбувається їх

усунення. На етапі формування архітектури майбутньої системи процедури тестування проводяться більш ретельно, що дозволяє локалізувати помилки на ранніх стадіях.

Специфікації Cleanroom дають повний і точний опис функцій системи. Вироблення специфікацій сприяє більш глибокому розумінню вимог, пропонованих до кінцевого продукту, і його функцій, а самі специфікації є основою для тестування, сертифікації і подальшого розвитку системи. Відповідно до методології Cleanroom системи будуються за модульним принципом. При цьому розрізняють три категорії модулів: модуль типу “чорна шухляда” (black box), модуль-опис (state box) і “прозорий” модуль (clear box), що відбивають суть технології покрокової деталізації. “Чорна шухляда” являє собою специфікацію всієї системи чи її частини з погляду зовнішнього “користувача” (їм може бути звичайний користувач або об’єкт системи). “Користувач” впливає на "чорну шухляду", що певним чином відгукується (реагує) на ці впливи. Ціль уведення специфікацій “чорної шухляди” у тому й полягає, щоб описати коректне поводження системи у всіх ймовірних ситуаціях.

Процес проектування і верифікації в методології Cleanroom зводиться до формального опису функцій, необхідних для реалізації поводження “чорної шухляди”, тобто до створення модуля-опису. Даний процес нагадує проектування об’єктів в об’єктному програмуванні, коли дані і функції (методи) інкапсулюються в єдиній сутності. “Прозорий” модуль – це програмна реалізація функцій модуля-опису. При написанні програм повинні використовуватися тільки базові конструкції з технології структурного програмування (блоки, розгалуження, цикли). Процес програмування може приводити до створення нових “чорних шухляд”, модулей-описів тощо. Якість програмного коду (відповідність роботи програми закладеним специфікаціям) перевіряється в ході верифікації.

Інструментом тестування надійності автоматизованого тестування й оцінки надійності ПЗ в методології Cleanroom є середовище Cleanroom Certification Assistant, в основі якої покладено ідею використання статистичних результатів тестування для визначення надійності ПЗ математичними методами. Спеціальний компонент Statistical Test Generation Facility (STGF) має власну мову опису тестових даних, що дозволяє запрограмувати сценарій тестування; характер розподілу даних, моменти виникнення критичних подій тощо. У результаті STGF генерує код мовою С, що після компіляції і запуску подає на вхід спробні дані програми, що тестується. Другий компонент – Cleanroom Certification Model – фіксує результати тестування у вигляді показників MTTF (Mean Time To Failure – середній час наробітку на відмовлення), що і використовуються для обчислення метрик надійності.

Одного разу згенерована програма тестування може використовуватися повторно для порівняння надійності і продуктивності різних версій розроблювального ПЗ.

Всі розробки, що орієнтовані на розроблювачів ПЗ, скеровані на покращання якості своїх процесів. Деякі з них мають безліч запитань і за отриманими відповідями оцінюють інтелектуальні системи, що тестуються. Після ідентифікації існуючих проблем для їх вирішення пропонуються конкретні заходи.

Процедура виконується в чотири етапи:

- діагностика проблем;
- визначення пріоритетів для їх вирішення;
- складання підготовчого плану;
- вироблення плану конкретних дій.

Інші моделі якості сприяють появі аналогічних моделей і стандартів якості в деяких розвинених країнах. Але запропоновані варіанти більшою мерою відповідають внутрішнім вимогам конкретної країни.

Сьогодні Німецький національний дослідницький центр з інформаційних технологій (GMD) завершує роботу над проектом SCOPE*PROCEPT, що охоплює інформаційні моделі процесів створення ПЗ, зокрема методи оцінки якості на основі метрик.

До складу SCOPE*PROCEPT входить кілька компонентів, що відповідають різним аспектам діяльності із створення програм:

- специфікація процесів програмування і тестування (ProcePT). Цей компонент призначений для вироблення специфікацій і тестування технологічних процесів, що використовуються у проекті. Розробка процесів ведеться за спадною схемою, починаючи з рівня загальної моделі процесу. Далі шляхом ітераційних уточнень автоматично створюється комплект необхідних документів:
- різноманітні довідники щодо проекту, опису типів діяльності; переліки продуктів, що будуть створені в проекті, відповідні діаграми і схеми;
- інжиніринг моделей якості. Суть його у визначенні моделі якості для даного проекту з використанням методів оцінки якості; передбачає вбудовування цієї моделі в існуючі технологічні і бізнес-процеси;
- вимір характеристик (метрик) ПЗ. Можна подати у вигляді своєрідного іспитового стенда, на якому досліджуються кількісні характеристики програм;
- інтегроване середовище для виміру характеристик компіляторів;
- моделювання процесів виробництва ПЗ. Цей компонент призначений для оцінки якості процесів розробки, причому на основі кількісних показників.

## ЗАСТОСУВАННЯ МЕТРИК ДЛЯ ОЦІНКИ ЯКОСТІ ПРОГРАМНИХ ПРОДУКТІВ НА ПРИКЛАДІ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Розглянемо застосування метрик оцінки якості програмних продуктів на прикладі інтелектуальних інформаційних систем.

Створення наукових основ проектування інтелектуальних інформаційних систем (ІІС) вимагає дослідження проблем кількісної оцінки якості систем на різних стадіях їхнього життєвого циклу. Оскільки невід’ємними компонентами ІІС є база знань (БЗ) і механізм логічного введення рішень, виникає необхідність включення в загальну оцінку ІІС якісних характеристик цих двох інтелектуальних компонентів.

Однією з основних задач проблеми оцінювання якості баз знань (БЗ) є вибір метрик і методів їхнього вимірювання.

Проблема подання знань ускладнюється ще й деякими додатковими проблемами, включаючи подання знань в практичних додатках , таких як видобування і внесення знань, ефективне подання знань, вибір інструментарію схем подання знань, модифікація знань при зміні середовища, оперування знаннями та використання відповідних знань в створенні рішень, що пропонуються.

Під метрикою у цьому випадку розуміють елементарну характеристику, що є характеристикою нижнього рівня, яка не розчленовується і підлягає безпосередньому вимірюванню [4]. Ототожнення понять метрики і елементарної характеристики зроблено з метою погодженості з термінологією, введеною в [2].

Можна припустити, що метрики відповідно до можливих статусів БЗ за аналогіює з базами даних (БД)[2] утворюють чотири групи метрик адекватності БЗ як інформаційної моделі предметної області:

- метрики адекватності концептуальної схеми (КС);
- метрики адекватності логічної схеми (ЛС);
- метрики адекватності схеми бази знань (СхБЗ);
- метрики БЗ.

При виборі метрик і методів їхнього виміру варто керуватися такими положеннями:

- метрики повинні бути коректно визначені, тобто метрики з різними іменами повинні мати різні визначення;
- метрики повинні відбивати безліч елементарних властивостей оцінюваного об'єкта, що мають фундаментальні відмінності один від іншого і групуватися в набори відповідно до умов, необхідних для існування конкретних характеристик програмного рівня;
- метрики повинні, з одного боку, мати практичну цінність, а з іншого – допускати максимальну можливість їхнього кількісного оцінювання;
- метрики повинні брати участь у формуванні якості БЗ як у розрізі окремої інтегральної характеристики, так і загалом, забезпечуючи при цьому можливість використання єдиного, незалежного від номенклатури характеристик, алгоритму оцінювання;
- методи виміру метрик повинні забезпечувати різну точність оцінок якості БЗ відповідно до висунутих до них вимог;
- методи виміру метрик повинні бути ефективними, тобто забезпечувати найбільш економічний вимір метрики.

При цьому для оцінки економічності використовується модифікація шкали, запропонованої в [5]:

АЛ – потрібен автоматичний алгоритм;

НС – потрібен некваліфікований фахівець для перегляду БЗ або приймальна особа без великого досвіду;

КС – потрібен перегляд БЗ кваліфікованим фахівцем;

ЕК – потрібен перегляд БЗ експертом;

ПР – потрібен прогін тестів на ЕОМ.

Запропонований набір метрик і методів не є статичним. Він може бути розширений або змінений залежно від вимог, пред'явлених до повноти і глибини оцінювання якості БЗ.

Уведення строгих кількісних метрик у програмування повинне сприяти розв'язанню таких практичних завдань:

- прогнозувати ймовірне число помилок у системі із самого початку проектування;
- на основі аналізу фази проектування системи прогнозувати рівень складності наступного супроводу;
- на основі аналізу вихідного коду програм прогнозувати рівень складності процесів тестування і відсоток помилок, що залишаються;
- за оцінками складності фази проектування системи визначати кінцевий розмір коду;
- визначати кореляцію окремих характеристик програмного коду з якістю готової системи;

- контролювати стадії розвитку проекту;
- аналізувати явні і приховані дефекти;
- на основі експериментального порівняння виявляти кращі методи і технології.

## ВИСНОВКИ

Розглянуто основні, хоча і далеко не всі, технологічні терміни. Описано і проаналізовано деякі з проблем, що пов'язані зі створенням і застосуванням метрик якості ПЗ загалом і баз знань зокрема. Найбільш ефективним видається спосіб створення деякої системи індикації дефектів і використання цієї системи для визначення конкретних напрямків подальшого розвитку і вдосконалення ПЗ. Якість програмного забезпечення, технологічна зрілість виробників і стандарти повинні бути поставлені користувачем на перше місце.

*1. Закон України "Про метрологію та метрологічну діяльність". – 1993. http://rada.gov.ua. 2. Качество баз данных. Основнные понятия и определения: Методические материалы // ИПС АН Украины. – 1993. – 47 с. 3. Качество баз данных. Обобщенная оценка качества: Методические материалы // ИПС АН Украины. – 1994. – 48 с. 4. Оценка качества баз данных. Метрики адекватности баз данных. Методические материалы // ИПС АН Украины. – 1994. – 57 с. 5. Боэм Б., Браун Дж., Каспар Х. и др. Характеристики качества программного обеспечения. – М.: Мир, 1981. – 206 с. 6. Цикритзис Д., Лоховски Ф. Модели данных. М.: Финансы и статистика, 1985. – 344 с. 7. Захарова О.В. Функциональные зависимости как один из аспектов проверки полноты информационной модели. // Инженерия и инструментальные средства программирования: Тез. докл. АН Украины. – К., 1–4 июня 1992; – К.: Ин-т программных систем АН Украины, 1993. – 116 с. 8. Barket. R. CFSE*Method: Entity-relationship modeling // Addison-Wesley Publ.Comp., Oracle. – 1989. – 695 с. 9. Озкарахан Э. Машины баз данных и управление базами данных. – М.: Финансы и статистика, 1989. – 695 с. 10. Калиниченко Л.А. Методы и средства интеграции неоднородных баз данных. – М.: Наука, 1983. – 424 с. 11. Ульман Дж. Основы систем баз данных; Финансы и статистика, 1989. – 350 с. 12. Rubey R.J., Hartwick R.D. Quantitative Measurement of Program Quality, Proceedings, ACM National Conference. – 1986. – P. 671–677. 13. Brown J.R., Lipow M. The Quantitative Measurement of Software Safety and Reliability, revised from TRW Report No SDP-1776. – 1983. 14. Wulf W.A. Program Methodology, Proceeding of a Symposium on the High Cost of Software, J.Goldberg (ed.), Stanford Research Institute, September 1983. 15. Weinberg G.M. The Psychology of Improved Programmer Performance, Datamation, 82–85 (November 1986). 16. Kernighan B.W., Plauger P.J. The Elements of Programming Style McGraw – Hill, 1984. 17. A Study of Fundamental Factors Underlying Software Maintenance Problems, CIRAD, Inc., December 1981. 18. Myers G.J. Software Reliability: Principles and Practices, Wiley. – New York, 1986. 19. Haltead M.H. Elements of Software Science, Elsevier North-Holland, Inc., 1987. 20. Gilb T., Software Metrics, Studentlitteratur AB, Lund, Sweden and Winthrop, Cambridge, MA, USA, 1996. 21. Когаловский М.Р. Энциклопедия технологій баз даннях. – М.: Финансы и статистика, 2002. – 800 с. 22. Соммервил И. Инженерия программного обеспечения. – 6-е изд.: Пер. с англ. – М.: Издательский дом "Вільямс", 2002. – 624 с. 23. Агапов А.С., Зенин С.В., Михайловский Н.Э., Мкртумян А.А. Оценка и аттестация зрелости процессов создания и сопровождения программных средств и информационных систем (ISO/IEC TR 15504-CMM) / Пер. с англ. – М.: "Книга и бизнес", 2001. 24. Мартин Фаулер. Новые методологии программирования. www.maxkir.com/sd/newmethRUS.html. 25. http://www.citforum.ru/programming/digest/testirovanie/.*