ІНФОРМАТИКА, ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА АВТОМАТИЗАЦІЯ. КІБЕРНЕТИКА

УДК 004.891.3: 004.3

О.В. ПОМОРОВА, Т.О. ГОВОРУЩЕНКО, С.Я. ТАРАСЕК
Хмельницький національний університет

АНАЛІЗ ТА ОПРАЦЮВАННЯ МЕТРИК ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ЕТАПІ ПРОЕКТУВАННЯ

У статті проведено огляд стандартів в галузі забезпечення якості програмного забезпечення (ПЗ) та моделей якості ПЗ. Проаналізовано метрики якості програмного забезпечення з точки зору можливості їх застосування на етапі проектування ПЗ з одержанням точного або прогнозованого значення. Розглянуто методи вимірювання показників якості з точки зору їх застосовності на етапі проектування. Виділено основні задачі в галузі метричного аналізу та опрацювання метрик якості на етапі проектування ПЗ.

In article review of standards in the area software quality guaranteeing and software quality models were surveyed. Software quality metrics were analyzed in respect to theirs adaptability on the software design stage with receiving of exact or predictable value. The measuring techniques of software quality metrics were viewed in respect to theirs adaptability on the software design stage. Basic tasks in the area software metrical analyze and software quality metrics on the developing stage processing were emphasized.

Ключові слова: якість ПЗ, стандарти якості, моделі якості, метрики якості ПЗ, методи вимірювання якості ПЗ, методи опрацювання метрик.

Вступ

Сучасне програмне забезпечення (ПЗ) є визначальною складовою багатьох систем, серед яких системи критичного застосування та спеціалізовані системи різноманітного призначення. Для зазначених систем проблема наявності помилок в програмному забезпеченні та низька якість ПЗ загрожують катастрофами [1-3] в певних галузях, які призводять до людських жертв та екологічних катаклізмів, або, щонайменше, до значних економічних та часових втрат. Проблема виявлення та усунення помилок загострюється по мірі збільшення складності задач та програм, які їх вирішують. Розвиток сучасних технологій розроблення ПЗ вимагає динамічного розвитку засобів оцінки якості ПЗ, причому вже на етапі проектування (з точки зору економічної та часової доцільності).

Сучасна індустрія ПЗ характеризується високою конкуренцією. Для успішної роботи на цьому ринку софтверна компанія повинна розробляти, впроваджувати та супроводжувати ПЗ швидко, вкладаючись в термін та із задовільною якістю. Тому багато софтверних компаній вкладають кошти в покращення якості процесу розробки ПЗ, пам'ятаючи про те, що таке вкладення коштів обов'язково окупається – аналіз документованих випадків покращення процесів розробки ПЗ показує, що в успішних випадках спостерігається істотне покращення продуктивності та якості з середнім рівнем повернення вкладень від 5: 1 до 8: 1 [4].

На етапі проектування важливо закласти цілу низку вимог по якості: вимоги до структури програмної системи (ПС); вимоги до навігації по ПС; вимоги до дизайну інтерфейсів користувача; вимоги до мультимедіа-компонентів ПС; вимоги по зручності (usability); технічні вимоги. На цьому етапі формується відповідь на питання "Яким чином програмна система буде реалізовувати висунуті до неї вимоги?" Інформаційні потоки етапу проектування ПЗ: вимоги до ПЗ, представлені інформаційною, функційною та поведінковою моделями аналізу. Інформаційна модель описує інформацію, які повинна обробляти ПЗ на думку замовника. Функційна модель визначає перелік функцій обробки інформації та перелік модулів програмної системи. Поведінкова модель фіксує бажану динаміку системи (режими її роботи). На виході етапу проектування – розробка даних, розробка архітектури та процедурна розробка ПЗ.

Огляд стандартів та основних термінів в області забезпечення якості програмного забезпечення

Якість ПЗ – це характеристика ПЗ, яка відображає ступінь його відповідності вимогам. При цьому вимоги можуть трактуватись досить широко, що породжує цілий ряд незалежних визначень поняття якості. Згідно з визначенням ISO [5], якість – це ступінь відповідності присутніх характеристик вимогам. Згідно з [6], якість – це повнота властивостей і характеристик продукту, процесу або послуги, які забезпечують здатність задовольняти оголошеним або передбачуваним потребам. Згідно з [7], якість ПЗ – це ступінь, в якому воно володіє потрібною комбінацією властивостей. Згідно з [8], якість програмного засобу – це сукупність властивостей програмного засобу, які обумовлюють його придатність задовільняти задані або передбачувані потреби у відповідності до його призначення. Стандарт [9] дає наступне визначення якості ПЗ – весь обсяг ознак та характеристик програмної продукції, який належить до її здатності задовольняти встановлені або передбачувані потреби.

У відповідності до стандартів [10], забезпечення якості – це сукупність планованих та систематичних заходів, необхідних для впевненості в тому, що продукція або процеси задовольняють певним вимогам до якості.

Критерій оцінки якості програмного засобу – сукупність прийнятих у встановленому порядку правил і умов, за допомогою яких встановлюється прийнятність в цілому якості програмного засобу [8]. Згідно з [9], критерій оцінки якості ПЗ – набір визначених та задокументованих правил і умов, які

використовуються для висновку про прийнятність загальної якості конкретної програмної продукції. Характеристика якості програмного засобу – набір властивостей програмного засобу, за допомогою яких описується та оцінюється його якість [8]. Згідно з [9], характеристика якості ПЗ – набір властивостей (атрибутів) програмної продукції, за якими її якість описується і оцінюється. Показник якості програмного засобу – характеристика якості програмного засобу, яка має кількісне значення [8]. Метрика якості ПЗ – кількісний масштаб і метод, які можуть бути використані для визначення значення ознаки, прийнятої для конкретної програмної продукції [9]. Згідно з [11], метрика визначається як міра ступеня володіння властивістю, яка має числове значення. Взагалі, метрика ПЗ – це міра, яка дозволяє одержати числове значення деякої властивості ПЗ або його специфікацій.

Основним стандартом якості в області інженерії ПЗ є стандарт [12], який визначає номенклатуру, атрибути і метрики вимог якості ПЗ. Цей стандарт є одним з визначальних факторів при моделюванні якості ПЗ. На додаток до нього випущено набір стандартів [13], які регламентують способи оцінки цих характеристик. В сукупності вони утворюють модель якості, відому під назвою SQuaRE (Software Quality Requirements and Evaluation).

Стандарти серії ISO 9000 складаються з частин [14-18]. Стандарти [14, 18] – довідники, стандартами відповідності є [15-17]. Крім серії ISO 9000 завтерджені й інші системи стандартів, які поки що не настільки популярні, але впевнено набирають силу [19] – похідний стандарт від ISO 9000, розроблений асоціацією QuEST Forum. В нього включено додаткові вимоги до надійності, до управління життєвим циклом ПЗ, а також питання інсталяції та налагодження ПЗ [20] – модель зрілості процесів – стандарт, розроблений американським інститутом SEI і призначений переважно для програмної індустрії. Згідно з моделлю [20], софтверне підприємство повинно довести, що його продукція не просто задовільняє вимоги абстрактного замовника, але й ідеально підходить конкретній групі клієнтів.

Стандарт [8] встановлює терміни та визначення понять в області якості програмних засобів. Стандарт [21] встановлює загальні положення по оцінці якості програмних засобів, а також номенклатуру та застосовність показників якості ПЗ. Стандарт [9] визначає 6 характеристик, які оцінюють якість програмного забезпечення: функційні можливості, надійність, практичність, ефективність, супроводжуваність, мобільність. Дані характеристики утворюють основу для подальшого уточнення та опису якості ПЗ. Керівництва описують використання характеристик якості для оцінки якості ПЗ. Цей стандарт не визначає методи вимірювання та оцінки якості ПЗ. Визначення характеристик та відповідна модель процесу оцінки якості стандарту [9] застосовні, коли визначено вимоги до ПЗ та оцінюється якість ПЗ в процесі життєвого циклу.

Моделі якості програмного забезпечення

Загальний підхід до моделювання якості ПЗ полягає в тому, щоб спочатку ідентифікувати невеликий набір атрибутів якості найвищого рівня абстракції, а потім в напрямку "згори-донизу" розбити ці атрибути на набори підлеглих атрибутів. Стандарт [12] є типовим прикладом такого підходу. В межах моделі SquaRE виділяються наступні 6 характеристик якості [22]:

1) функційність – функційні вимоги формулюються у вигляді тверджень в імперативній модальності, які описують поведінку ПЗ;

2) надійність – показники надійності характеризують поведінку ПЗ при виході за межі штатних значень параметрів функціювання за причини збою в оточенні або в самому ПЗ;

3) зручність – відповідність ПЗ вимогам до зручності надзвичайно важко піддається оцінці – запропоновані підходи базуються на вимірах витрат нормативних одиниць праці (нормогодин) на опанування програмного забезпечення користувачем, а також на проведенні та аналізі експертних оцінок, в тому числі із застосуванням методів нечіткої логіки;

4) ефективність (за ресурсами і часом) – атрибути ефективності традиційно належать до найважливіших кількісних показників якості ПЗ; для вимірювання значень атрибутів ефективності створено певний високорозвинений інструментарій, причому розроблені методики прогнозування інтегральних значень показників ефективності ПЗ, виходячи із значень цих показників для компонентів ПЗ та його оточення;

5) супроводжуваність – вимоги до супроводжуваності спрямовані в першу чергу на мінімізацію зусиль по супроводу та модернізації ПЗ, докладених експлуатаційним персоналом; для їх оцінки використовуються різні методики прогнозування витрат на виконання типових процедур супроводу;

6) можливість переносу ПЗ – характеризує ступінь свободи у виборі компонентів системного оточення, необхідних для його функціювання. Оцінка можливості переносу ускладнюється принциповою незавершеністю, динамічністю списку можливих варіантів оточення, обумовленою швидким прогресом в сфері інформаційних технологій.

Модель якості, створена в межах стандарту [12], визначається загальними характеристиками продукту. Характеристики можуть розбиватись на підхарактеристики якості. Нижній рівень ієрархії представляють атрибути ПЗ, які підлягають точному опису та виміру. Вимоги якості можуть бути представлені як обмеження на модель якості. Оцінка якості ПЗ відбувається наступним чином. Спочатку оцінюються атрибути ПЗ; обирається метрика, градуюється шкала оцінки залежно від можливих ступенів відповідності атрибуту накладеним обмеженням. Набір "виміряних" атрибутів представляє собою критерій для оцінки підхарактеристики та характеристик.

Існують десятки різних підходів до забезпечення якості ПЗ, і в кожного є свої переваги. Однією з

перших моделей якості став стандарт ISO серії 9000. Сертифікати ISO серії 9000 зберігають популярність та визнаються в усьому світі, але методики, покладені в основу стандартів серії ISO 9000, поступово застарівають. Недоліки стандартів серії ISO 9000 [23]:

1) недостатня деталізованість стандарту, можливість різних його трактувань залежно від уявлень аудитора;

2) неточність оцінки якості процесів, задіяних при створенні та впровадженні ПЗ;

3) відсутність в стандарті механізмів, які сприяють покращенню існуючих процесів.

Перераховані проблеми змусили експертів розробляти більш досконалі рішення в галузі забезпечення якості ПЗ, що призвело до створення цілої низки нових стандартів та методологій (Capability Maturity Model (CMM), ISO/IEC 15504 (SPICE), Bootstrap, Trillium, ISO 12207). Найбільш вдалі та змістовні стандарти – Capability Maturity Model (CMM) та ISO/IEC 15504 (SPICE).

Для визначення загального рівня розвитку технологічних процесів в програмних організаціях розробили спеціальну систему оцінки зрілості технологічних процесів в софтверних організаціях – модель Capability Maturity Model (CMM), засновану на так званих рівнях зрілості (maturity levels) [24]. Таких рівнів зрілості в CMM п’ять, і кожен з них характеризує певний ступінь якості програмних продуктів [24]:

1) початковий (initial) – відсутнє стабільне середовище розробки і супроводження; не витримуються терміни випуску продуктів; всі сили спрямовано на кодування і тестування програми (75 % софтверних організацій);

2) повторюваний (repeatable) – жорстке керування, планування і контроль; акцент робиться на початкові вимоги, методи оцінки і конфігураційний менеджмент (15 % софтверних організацій);

3) фіксований (defined) – процеси повністю документовані, стандартизовані та інтегровані в єдиний технологічний потік (8 % софтверних організацій);

4) керований (managed) – намагання оцінити якість процесів і готового продукту кількісно; для контролю над процесами використовуються метрики (1.5 % софтверних організацій);

5) оптимізований (optimizable) – намагання покращення роботи, керуючись кількісними критеріями якості; основна мета – випуск бездефектних продуктів, в яких помилки усунені ще на стадії внутрішнього тестування (0.5 % софтверних організацій).

6) 5-й рівень CMM вимагає постійного самостійного покращення процесу. Для покращення процесу керування проектом його ефективність вимірюється за допомогою метрик процесу [25]. Ці метрики дозволяють виміряти ефективність організації процесу, а також аналізу вимог, проектування, програмування і тестування.

Використання CMM ускладнюють наступні проблеми:

1) стандарт CMM є власністю Software Engineering Institute і не є загальнодоступним, відтак подальша розробка стандарту ведеться самим інститутом без залучення іншої частини програмістської спільноти;

2) оцінка якості процесів організацій може проводитись лише спеціалістами, які пройшли спеціальне навчання та акредитовані SEI;

3) стандарт орієнтований на застосування у відносно великих софтверних компаніях.

Для оцінки рівня компанії за шкалою CMM розроблено декілька програмних пакетів: CMM Live – електронний довідник та експерт-радник по технологіях CMM; SoftGuide – орієнтований на розробників ПЗ, які намагаються покращити якість своїх процесів; SCOPE*PROCEPT [24] – охоплює інформаційні моделі процесів створення ПЗ, включаючи методи оцінки якості на основі метрик.

Стандарт SPICE (скорочення від Software Process Improvement and Capability dEtermination) [26] створено ISO як єдиний стандарт оцінки програмних процесів. SPICE нагадує CMM. Основною задачею софтверної організації теж є постійне покращення процесу розробки ПЗ, також використовується схема з різними рівнями можливостей (в SPICE визначено 6 різних рівнів), але ці рівні застосовуються не лише до організації в цілому, але й до окремо взятих процесів. Рівні можливостей моделі SPICE [23]:

1) рівень 0– процес не виконується;

2) рівень 1– виконуваний процес (1.1 – вимірювання продуктивності процесу);

3) рівень 2– керований процес (2.1– керування продуктивністю, 2.2– керування створенням продуктів);

4) рівень 3– встановлений процес (3.1– документування процесу, 3.2– відстежування ресурсів процесу);

5) рівень 4– передбачуваний процес (4.1– вимірювання процесу, 4.2– керування процесом);

6) рівень 5– оптимізуючий процес (5.1– зміна процесу, 5.2– постійне вдосконалення).

Однією з найважливіших переваг SPICE є його вільне поширення на офіційному сайті SPICE [27]. SPICE являє собою більш повний набір засобів по забезпеченню якості та покращення процесів, ніж стандарти серії ISO 9000. Використовувати SPICE можна й в невеликих софтверних компаніях (менше 50 чоловік), причому, як показують результати роботи проекту SPIRE [28], навіть при невеликих вкладеннях в маленьких компаніях можна досягти істотного збільшення продуктивності праці та покращення якості розробленого ПЗ. В зв’язку з вільним поширенням стандарт SPICE набуває все більшої популярності.

В останні роки запропонований та розвивається так званий "мовно-орієнтовний підхід (МОП) до вимірювань та оцінки ПЗ", який дозволяє створювати автоматичні, засновані на знаннях, програмні технології оцінки ПС, що базуються на однозначному, "прозорому" та семантично коректному описі понять прикладної області "якість програмного забезпечення" [29]. МОП до оцінки ПС полягає у використанні трьох рівнів узгоджених формальних мов, призначених для [29]:

1) точного визначення множини властивостей базового набору символів і конструкцій кожної використовуваної мови опису програмних продуктів, виходячи з визначень семантики і синтаксису цієї,

точно визначеної мови. Ці властивості представляються в "вимірювальній моделі мови";

2) формального визначення метрик програмного продукту в термінах вимірювальних моделей мов опису цих продуктів;

3) формального визначення знань прикладної області "якість програмного забезпечення", специфікацій вимог до якості конкретних програмних продуктів в термінах формально визначених метрик і методів вимірювань програм, а також теорії вимірювань.

МОП підтримує всі відомі стандарти, моделі і метрики програмного забезпечення.

Аналіз метрик якості програмного забезпечення з точки зору можливості їх застосування на етапі проектування ПЗ з одержанням точного або прогнозованого значення

Показники оцінки складності проекту є дуже важливими для одержання об'єктивних оцінок щодо проекту. Як правило, дані показники не можуть бути обчислені на ранніх стадіях роботи над проектом, оскільки вимагають, як мінімум, детального проектування. Однак ці показники важливі для одержання прогнозованих оцінок тривалості та вартості проекту, оскільки безпосередньо визначають його трудомісткість.

Метрики складності програм прийнято розділяти на 3 основні групи [30]:

- метрики розміру програм;
- метрики складності потоку управління програм;
- метрики складності потоків даних програм.

Метрики розміру програм базуються на визначенні кількісних характеристик, пов'язаних з розміром програми, і відрізняються відносною простотою. Метрики цієї групи орієнтовані на аналіз вихідного тексту програм, тому вони можуть використовуватись для оцінки складності проміжних продуктів розробки. До найбільш відомих метрик даної групи відносять кількість операторів програми, кількість рядків вихідного тексту, набір метрик Холстеда [30, 1].

Метрики складності потоку управління програм базуються на аналізі керуючого графу програми. Метрики другої групи теж можуть застосовуватись для оцінки складності проміжних продуктів розробки. Представником другої групи є метрика Маккейба [30].

Метрики складності потоків данпх програм базуються на оцінці використання, конфігурації та розташування даних в програмі. В першу чергу це стосується глобальних змінних. До даної групи відносять метрики Чепіна [30].

Розмірно-орієнтовані метрики (показники оцінки обсягу) прямо вимірюють програмний продукт і процес його розробки. Базуються такі метрики на LOC-оцінках (LOC – Lines of Code). Кількість рядків вихідного коду (Lines of Code – LOC, Source Lines of Code – SLOC) є найбільш простим і поширеним способом оцінки обсягу робіт по проекту. Ці метрики не універсальні, особливо це відноситься до показника LOC, який істотно залежить від використовуваної мови програмування.

Залежно від того, яким чином враховується вихідний код, виділяють 2 основних показники LOC [30, 1]: кількість «фізичних» рядків коду (LOC, SLOC, KLOC, KSLOC, DSLOC) – визначається як загальна кількість рядків вихідного коду, включаючи коментарі і порожні рядки; кількість «логічних» рядків коду (LSI, DSI, KDSI, де SI-source instructions) – визначається як кількість команд і залежить від використовуваної мови програмування. В тому випадку, якщо мова не допускає розташування декількох команд на одному рядку, то кількість "логічних" LOC буде відповідати кількості "фізичних", за винятком кількості порожніх рядків і рядків коментарів.

Метрика Холстеда належить до метрик, які обчислюються на основі аналізу кількості рядків та синтаксичних елементів вихідного коду програми.

Основу метрики Холстеда складають 4 вимірювані характеристики програми [30, 1]: ***NUOprtr*** (Number of Unique Operators) – кількість унікальних операторів програми, включаючи символи-роздільники, імена процедур і знаки операцій (словник операторів); ***NUOprnd*** (Number of Unique Operands) – кількість унікальних операндів програми (словник операндів); ***NOprtr*** (Number of Operators) – загальна кількість операторів в програмі; ***NOpnd*** (Number of Operands) – загальна кількість операндів в програмі.

На основі цих характеристик розраховуються оцінки: словник програми, довжина програми, обсяг програми, оцінка її реалізації, складність її розуміння, трудомісткість кодування, інформаційний вміст, оптимальна модульність, складність програми ($HDiff = (NUOprtr/2)\times(NOprnd/NUOprnd)$) [30].

Показник цикломатичної складності Маккейба є одним з найбільш розповсюджених показників оцінки складності програмних проектів. Цикломатичне число Маккейба показує необхідну кількість проходів для покриття всіх контурів сильнозв'язаного графа або кількість тестових прогонів програми, необхідних для вичерпного тестування за принципом «працює кожна гілка програми». Показник цикломатичної складності розраховується для модуля, метода та інших структурних одиниць програми.

Показник цикломатичної складності дозволяє не лише провести оцінку трудомісткості реалізації окремих елементів програмного проекту і скоригувати загальні показники оцінки тривалості та вартості проекту, але й оцінити зв'язані ризики і прийняти необхідні управлінські рішення.

Існує декілька модифікацій *метрик Чепіна.* Суть найпростішого з точки зору практичного використання і досить ефективного методу полягає в оцінці інформаційної міцності окремо взятого програмного модуля за допомогою аналізу характеру використання змінних зі списку введення-виведення. Вся множина змінних, яка складає список введення-виведення, розбивається на 4 функційні групи [30]:

множина "*P*" – змінні, що вводяться для розрахунків та для забезпечення виведення; множина "*M*" – модифіковані або створювані всередині програми змінні; множина "*C*" – змінні, які приймають участь в управлінні роботою програмного модуля (керуючі змінні); множина "*T*" – не використовувані в програмі ("паразитні") змінні.

Оскільки кожна змінна може виконувати одночасно декілька функцій, необхідно її враховувати в кожній відповідній функційній групі.

Значення метрики Чепіна: $Q = P + 2 * M + 3 * C + 0.5 * T$.

Для оцінки складності програми використовуються й інші метрики [31]: метрики Джилба – кількість операторів циклу, кількість операторів умови, кількість модулів або підсистем, відношення кількості зв'язків між модулями до кількості модулів; метрика Шнадевида – кількість шляхів в графі керування; метрика Майєрса – інтервальна міра; метрика Хансена – пара (цикломатична кількість, кількість операторів); метрика Чена – топологічна міра; метрика Вудворда – кількість вузлів передач управління; метрика Кулика – кількість найпростіших циклів; метрика Хура – цикломатична кількість мереж Петрі; метрики Вітворфа, Зулевського – міра складності потоку керування, міра складності потоку даних; метрика Петерсона – кількість багатовходових циклів; метрики Харрісона, Мейджела – функційна кількість, функційне відношення, регулярні вирази; метрика Пивоварського – модифікована цкломатична міра складності; метрика Пратта – тестуюча міра; метрика Кантоне – характеристичні числа поліномів графу програми; метрика Мак-Клура – міра складності, заснована на кількості можливих шляхів виконання програми, кількості керуючих конструкцій та змінних; метрика Кафура – міра на основі концепції інформаційних потоків; метрика Схуттса, Моханті – ентропійні міри; метрика Коллофело – міра логічної стабільності програм; метрика Зольновського, Сімонса, Тейєра – зважена сума різних індикаторів; метрика Берлінгера – інформаційна міра; метрика Шумана – складність з позиції статистичної теорії мови; метрика Янгера – логічна складність з врахуванням історії обчислень; метрика Тая – покращення метрики Маккейба; метрика Кокола – комплексна метрика, заснована на більш простих; метрики зв'язності (зчеплення) – ступінь залежності кожного модуля від кожного з інших модулів за даними, за зразком, за управлінням, за зовнішніми посиланнями, за загальною областю, за змістом (кількісний показник – ступінь зчеплення).

Для оцінки складності програми на етапі розробки специфікації вимог до програми використовується метрика прогнозованої кількості операторів Nпрогн програми [30]: $N_{прогн} = NF * N_{од}$, де NF – кількість функцій або вимог в специфікації вимог до розроблюваної програми; $N_{од}$ – одиничне значення кількості операторів (середня кількість операторів, які доводяться на одну середню функцію або вимогу).

Оцінка складності програми на етапі визначення архітектури [30]: $C = \frac{NI}{NF * NI_{од} * K_{скл}}$, де NI – загальна кількість змінних, які передаються по інтерфейсах між компонентами програми (статистична); $NI_{од}$ – одиничне значення кількості змінних, які передаються по інтерфейсах між компонентами (середня кількість змінних, які передаються по інтерфейсах, що доводяться на одну середню функцію або вимогу); $K_{скл}$ – коефіцієнт складності розроблюваної програми, враховує ріст одиничної складності програми (складності, що доводиться на одну функцію або вимогу специфікації вимог до програми) для великих та складних програм в порівнянні з середнім показником складності.

Враховуючи результати досліджень, одержаних в [32], створимо таблицю метрик (табл. 1) процесу проектування ПЗ з огляду на точність або прогнозованість їх значень.

Різні метрики відображають різні аспекти складності ПС. Для всебічного врахування даних аспектів при оцінці ПС застосовується не одна метрика, а їх сукупність. Якщо було одержано декілька метрик, то кожне значення метрики множиться на відповідний ваговий коефіцієнт, встановлений експертним шляхом з огляду на домінуючі критерії якості відповідно до принципових задач, особливостей, функціонального призначення та властивостей ПС, а потім додаються всі показники для одержання комплексної оцінки рівня якості ПЗ або якості процесу проектування ПЗ. Найбільш актуально застосовувати досить великі сукупності метрик складності на етапі проектування, а в наступних етапах значення метрик фактично уточнюються.

Щодо *метрик процесу розроблення ПЗ*, то однією з найпростіших метрик є ступінь виявлення дефектів для заданої фази виявлення і заданої фази появи [25]. Наприклад, "ступінь виявлення дефектів, рівний 0,2 дефекта на 100 вимог на стадії реалізації" означає, що на стадії реалізації 500 вимог, і в одній з них було виявлено дефект. Коли ступені виявлення дефектів стають рівними нормам організації, відбувається оцінка всього процесу в цілому, а не лише конкретного проекту. Результат такої метрики очевидний: більше дефектів виявляється на тому етапі, коли вони були допущені, а на більш пізніх етапах виявляється менше дефектів. Оскільки чим пізніше виявляється і виправляється дефект, тим дорожче це обходиться, даний факт може означати, що проект і процес розроблення виконаний краще.

Таблиця 1

Метрики процесу проектування ПЗ

Метрики:	
з точним значенням на етапі проектування	**з прогнозованим значенням на етапі проектування**
1) метрики Чепіна – аналізують характер використання змінних зі списку введення, тобто оброблювану інформацію; 2) метрики зв'язності (зчеплення) – чим вище зв'язність модуля з іншими модулями, тим вища чутливість до внесення змін, тобто високий ступінь зв'язності (зчеплення) модуля з іншими модулями – це суттєвий недолік; 3) метрика Джилба – на етапі проектування можна тільки підрахувати кількість модулів та відношення кількості зв'язків між модулями до кількості модулів; 4) метрика Мак-Клура – метрика, спрямована на вимірювання архітектури системи; 5) метрика Кафура – метрика, заснована на врахуванні даних; 6) метрика Зольновського, Сіммонса, Тейєра – складова метрики (структура, взаємодія, обсяг, дані); 7) метрика звертання до глобальних змінних – пара (модуль, глобальна змінна); 8) метрика Тая – складність програми визначається вимірюванням інформаційного потоку даних; 9) метрика Вітворфа, Зулевського – складова метрики "міра складності потоку даних"; 10) час модифікації моделей; 11) кількість знайдених помилок при інспектуванні.	1) очікувана LOC-оцінка – за кожною функцією експерти надають краще, гірше та імовірне значення, тоді очікувана LOC-оцінка обчислюється за формулою: $LOC_{оч_i} = (LOC_{кращ_i} + LOC_{гірш_i} + 4 \times LOC_{імов_i}) / 6$ є вільно поширювані інструменти очікуваної LOC-оцінки [33]; 2) метрики Холстеда – міра довжини модуля ($N = n_1 \log_2(n_1) + n_2 \log_2(n_2)$, де n_1 – кількість різних операторів, n_2 – кількість різних операндів) та обсяг модуля як кількість символів для запису всіх операторів і операндів тексту програми ($V = N \times \log_2(n_1 + n_2)$); 3) метрики Маккейба – для оцінки складності ПС, виходячи з топології внутрішніх зв'язків, Маккейб розробив метрику цикломатичної складності $V(G) = E - N + 2$, де E – кількість дуг, а N – кількість вершин в керуючому графі ПЗ; 4) метрика Хансена; 5) метрика Кокола – комплексна метрика, яка, як правило, враховує значенн метрик Холстеда, Маккейба і LOC; 6) загальний час розробки і окремий час для кожної стадії; 7) час виконання робіт в процесі; 8) прогнозована кількість операторів програми; 9) прогнозована оцінка складності програми; 10) очікувана вартість розробки кожної функції: $ВАРТІСТЬ_i = LOC_{оч_i} \times ВАРТІСТЬ_{рядка_i}$; 11) прогнозована вартість перевірки якості; 12) прогнозована вартість процесу розробки; 13) прогнозована продуктивність розробки кожної функції (на основі продуктивності аналогічних функцій в аналогічних програмних продуктах): $ПРОДУКТ_i = ПРОДУКТ_{ан_i} \times (LOC_{ан_i} / LOC_{оч_i})$; 14) прогнозовані витрати на розробку кожної функції: $ВИТРАТИ_i = (LOC_{оч_i} / ПРОДУКТ_i)$; 15) прогнозований функційний розмір FP – для обчислення функційного розміру: ідентифікуються очікувані від програмного додатку функції за критеріями International Function Point Users Group (IFPUG) [35]; для кожної виділеної функції порахувати кількість зовнішніх входів, кількість зовнішніх виходів, кількість зовнішніх запитів, кількість внутрішніх логічних файлів, кількість зовнішніх логічних файлів; кожен з факторів, визначених на попередньому кроці, множиться на коефіцієнт складності даного фактору в програмному додатку [34], такі добутки додаються за кожним з факторів; додати одержані для кожної функції суми (наближений функційний розмір); визначити ваги для 14 загальних характеристик проекту [36, 25] від 0 до 5 та знайти їх суму; обчислити уточнений функційний розмір за формулою: $Уточн.функц.розмір = Наближений\_функц\_розмір \times$ $\times [0.65 + 0.01 \times (Сума\_загальних\_характеристик]$. Функційний розмір використовується як відносна метрика для порівняння з попередніми проектами, за його допомогою можна обчислити кількість рядків коду, що дозволяє визначити загальну трудомісткість та терміни проекту; є вільно поширювані інструменти для обчислення функційного розміру [37]; 16) прогнозована оцінка трудовитрат та тривалості проекту – за моделлю Боема [25, 38] трудовитрати на розробку програмних додатків зростають швидше, ніж розмір додатків, для представлення даного співвідношення використовується експоненційна функція зі значенням показника, близьким до 1.12; тривалість проекту за моделлю Боема зростає експоненційно разом з докладеними до проекту зусиллями, однак в цьому випадку значення експоненти менше 1 і складає близько 0.35.

Інші метрики процесу розроблення ПЗ [25]: кількість дефектів на тисячу рядків програмного коду, виявлених протягом 12 тижнів після завершення проекту; відхилення в розкладі на кожній фазі: $\frac{(Фактична\_тривалість - Планова\_тривалість)}{Планова\_тривалість}$; відхилення у вартості: $\frac{(Фактична\_вартість - Планова\_вартість)}{Планова\_вартість}$; відношення загального часу проектування до загального часу програмування повинно бути не менше 50 %; ступені появи і виявлення дефектів на деякій фазі; ймовірна норма дефектів, що залишились – відношення кількості дефектів на 100 рядків коду до норми організації кількості дефектів на 100 рядків коду; залишковий ступінь дефектності – кількість дефектних детальних вимог на 100 вимог після завершення фази детальних вимог.

Процес розроблення ПЗ можна виміряти лише в порівнянні з даними по організації або по галузі, а самі по собі одержані значення не є інформативними [25].

Загальне покращення процесу розроблення ПЗ, в першу чергу, вимагає класифікації типів робіт і процесів, яка цілком залежить від компанії. Для кожного типу робіт визначається середня кількість дефектів на тисячу рядків вихідного коду до моменту завершення на етапах вимог, архітектури, детального проектування, реалізації для різних моделей життєвого циклу ПЗ – наприклад, каскадної, спіральної на 2-4 ітерації та спіральної на 5-10 ітерацій [25]. Для покращення процесу розроблення ПЗ використовуються кращі частини різних моделей.

В результаті аналізу метрик ПЗ можна визначити *основні проблеми метрології ПЗ* [32]: відсутність загальноприйнятої номенклатури показників якості; неможливість проведення натурних випробувань програм на всій множині початкових даних; низька достовірність та недостатність інформації для одержання оцінок показників якості; недостатність засобів вимірювання метрик програми; відсутність обґрунтованих вимог до метричної інформації, виражених в числовому вигляді, які могли б підлягати перевірці; відсутність можливості інтерпретації одержуваних метрик і оцінок показників якості програм.

Отже, методи оцінки якості ПС, особливо на етапі проектування, на сьогодні є суб'єктивно залежними, оскільки використовуються експертні вагові коефіцієнти для метрик; порівняння значень метрик поточних проектів з попередніми (постає проблема, що робити, якщо проект принципово новий); немає загальноприйнятої номенклатури метрик; відсутні точні значення метрик, з якими можна було б порівняти поточні одержані значення.

Методи вимірювання показників якості з точки зору їх застосовності на етапі проектування

Для одержання оцінки значень показників якості за стандартом [21] використовуються такі методи:

1) вимірювальний – базується на використанні інструментальних, вимірювальних та спеціальних програмних засобів для одержання інформації про властивості та характеристики ПЗ (обсяг ПЗ, кількість рядків коду, кількість операторів, кількість гілок в програмі, кількість точок входу/виходу та ін);

2) реєстраційний – заснований на одержанні інформації під час випробувань або функціювання ПЗ, коли реєструються або підраховуються певні події (час і кількість збоїв та відмов, час передачі керування від одного модуля до іншого, час початку і завершення роботи ПЗ);

3) органолептичний – заснований на використанні інформації, одержаної в результаті аналізу сприйняття органів чуття, і застосовується для визначення таких показників як зручність застосування, ефективність;

4) розрахунковий – базується на використанні теоретичних та емпіричних залежностей (на ранніх етапах розробки), статистичних даних, зібраних при проведенні випробувань, експлуатації та супроводженні ПЗ. Розрахунковими методами оцінюються показники надійності, точності, стійкості, час реакції, необхідні ресурси та ін.;

5) експертний – здійснюється групою експертів. Їх оцінка базується на досвіді та інтуїції, а не на безпосередніх результатах розрахунків або експериментів. Цей метод реалізується шляхом перегляду програм, кодів, супровідних документів, описів вимог до ПЗ групою експертів. Для цього встановлюються контрольовані ознаки, корельовані з одним або декількома показниками якості і включені в карти опитування експертів [36]. Метод застосовується при оцінці таких показників, як наочність, повнота та доступність програмної документації, легкість засвоєння, аналізованість, документованість, структурованість ПЗ та ін.

З опису методів вимірювання показників (метрик) якості зрозуміло, що на етапі проектування ПЗ неможливо виміряти жодної характеристики ще не розробленого ПЗ, неможливо реєструвати моменти процесу виконання ще не існуючого ПЗ і неможливо сприйняти органами чуття інформацію щодо нерозробленого ПЗ. Отже, на етапі проектування є можливість визначати якість ПЗ лише із застосуванням розрахункових та експертних методів.

Опрацювання результатів метричного аналізу

Порівняння якості програмних продуктів без кількісних оцінок є неможливим, тому доцільність застосування кількісних методів оцінки якості (метрик) очевидна. Актуальним є питання, як трактувати та опрацьовувати значення метрик, оскільки однією з проблем аналізу метрик вихідного коду є складність інтерпретації обчислених величин.

Статистичний аналіз метричної інформації передбачає:

1) Аналіз метрик за релізами: накопичення статистичної інформації метрик складності і якості ПЗ служить основою для управління складністю і якістю ПЗ в наступних проектах. Так, метрики довжини і

обсягу програми дають інформацію про збільшення чи зменшення обсягу програми в часі. Метрика цикломатичної складності показує, чи зростає складність від релізу до релізу. Метрика кількості рядків на реалізацію вимоги попереджає про виявлення збільшення кількості рядків під час виконання типового запиту чи під час реалізації типової вимоги та дозволяє спрогнозувати кількість рядків коду при використанні типових вимог і функцій на ранніх етапах. Глибокий аналіз змін по релізах інших кількісних метрик дає можливість виявити вузьке місце в програмі – блок коду, що інтенсивно змінюється, як потенційне місце виникнення помилок.

2) Аналіз метрик за замовниками: слід зберігати інформацію про всі зміни коду для всіх замовників.

3) Вибірка проектних даних за певними критеріями – проектами, програмами, замовниками і т.п.

Для визначення задовільності чи незадовільності досягнутого рівня складності ПС використовується відношення розрахункового значення метрики до базового (статистичного) значення метрики, взятого зі статистичних даних попередніх проектів. Якщо таке відношення близьке до 1 або менше 1, то можна зробити висновок про задовільний рівень складності ПС за аналізованою метрикою. При значенні відношення, яке значно перевищує 1, можливо, слід виконати попередні роботи заново. При цьому слід враховувати змістовний вміст незадовільної метрики, щоб коригувати відповідні аспекти програмного продукту.

Для визначення впливу конкретних значень окремих метрик на загальну складність ПС виконується інтегральна оцінка складності [39]. Можливі 2 варіанти інтегральної оцінки: 1) інтегральну відносну складність ПС можна визначити як середнє арифметичне відносних результатів (відношень розрахункового значення метрики до базового) всіх метрик, якщо відсутня апріорна інформація про вплив результатів конкретної метрики на загальну складність проміжного або кінцевого продукту; 2) інтегральна складність ПС визначається як середньозважена сума одержаних значень метрик (сума ваг всіх метрик дорівнює 1), якщо наявна статистична інформація про ступінь впливу значень конкретних метрик на інтегральну складність ПС.

При статистичному аналізі метричної інформації основною проблемою є неможливість опрацювання метрик для принципово нового проекту.

Важливість якості ПЗ збільшує інтерес до нових методів, які використовуються в побудові моделей якості ПЗ для прогнозування атрибутів якості. Одним з таких методів є метод, що базується на штучній нейронній мережі (ШНМ). В [40] показано використання ШНМ для прогнозу якості ПЗ на основі об'єктно-орієнтованих метрик. Залежною змінною в такому досліді були роботи по технічному обслуговуванню. Незалежними змінними були головні компоненти 8 об'єктно-орієнтованих метрик. Результат показав, що середня абсолютна величина відносної похибки був 0.265 для моделі з ШНМ. Тому можна зробити висновок, що ШНМ можуть бути корисні в побудові моделі якості ПЗ. В [41] доведено застосовність байєсових мереж в якості інструменту підтримки для числової оцінки ПЗ в реальному масштабі часу, особливо для додатків з особливими вимогами щодо безпеки. Байєсові мережі можуть також відігравати важливу роль при прийнятті рішення про сертифікацію ПЗ.

Проблема оцінювання результатів метричного аналізу програмних продуктів з використанням інтелектуальних методів (зокрема з використанням ШНМ) досліджувалась мало, причому лише для конкретних видів метрик (об'єктно-орієнтовані метрики) і для конкретних типів програмного забезпечення (об'єктно-орієнтоване ПЗ та ПЗ з особливими вимогами щодо безпеки). Отже, актуальними є подальші дослідження на предмет можливості використання ШНМ для прогнозу якості ПЗ на основі аналізу інших типів метричної інформації для різних типів ПЗ.

Висновки

Незважаючи на дослідження програмних метрик, в галузі забезпечення якості ПЗ є *ряд невирішених питань:*

1) лише 1.5 % софтверних організацій намагаються оцінити якість процесів і готового продукту кількісно, за допомогою метрик і лише 0.5 % софтверних організацій намагаються покращити роботу, керуючись кількісними критеріями якості з метою випуску бездефектних продуктів [42];

2) технологія вимірювання якості ще не досягла зрілості, оскільки лише 0.5 % софтверних організацій знаходяться на оптимізованому, зрілому рівні моделі CMM;

3) відсутні єдині стандарти на метрики, створено більше тисячі метрик [24], тому кожен постачальник "вимірювальної" системи пропонує власні способи оцінки якості і відповідно метрики;

4) існує проблема складності інтерпретації величин метрик.

Саме через невирішеність цих питань поки що неможливо створити бездефектне високоякісне ПЗ.

Приймаючи до уваги результати аналізу стандартів, моделей та метрик якості ПЗ, а також методів вимірювання показників якості ПЗ, можна зробити висновок, що перспективним напрямком досліджень є розроблення інтелектуальних систем, які: 1) обчислюватимуть розрахунковими та експертними методами точні або прогнозовані значення метрик програмного забезпечення вже на етапі проектування; 2) не лише будуватимуть метрики, але й аналізуватимуть і опрацьовуватимуть результати метричних оцінок, на основі чого надаватимуть рекомендації, висновки і прогнози про розроблюване програмне забезпечення.

Література

1. Скляр В.В. Оценка качества и экспертиза программного обеспечения: Лекционный материал. –

Харьков: НАУ "ХАИ", 2008. – 204 с.

2. Майерс Г. Надежность программного обеспечения: Пер. с англ. – М.: Мир, 1980. – 360 с.

3. Myers G.J. The Art of Software Testing. – New York: John Wiley and Sons, 1979. – 312 pp.

4. . Paulk M. C., Curtis B., Chrissis M. B., Weber C. V. Capability Maturity Model, Version 1.1 // IEEE Software, July 1993, pp. 18-27

5. ISO 9001: 1994 Quality systems – Model for quality assurance in design, development, production, installation and servicing

6. ISO 8402: 1994 Quality management and quality assurance

7. 1061-1998 IEEE Standard for Software Quality Metrics Methodology

8. ГОСТ 28806-90. Качество программных средств. Термины и определения.

9. ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению.

10. Сборник действующих международных стандартов ИСО серии 9000: Т.1, 2, 3. – М.: ВНИИКИ, 1998.

11. IEEE Standard Glossary of Software Engineering Terminology /IEEE Std 610.12-1990

12. ISO/IEC 9126 Software engineering – Product quality

13. ISO/IEC 14598 Information technology – Software product evaluation

14. ISO 9000 Quality systems – Fundamentals and vocabulary

15. ISO 9001 Quality systems – Requirements

16. ISO 9002 Quality systems – Model for quality assurance in production, installation and servicing

17. ISO 9003 Quality systems – Model for quality assurance in final inspection and test

18. ISO 9004 Quality management systems – Guidelines for performance improvements

19. TL 9000 A Two-part Quality Management System for Telecommunications

20. CMM Capability maturity model for Software

21. ГОСТ 28195-89. Оценка качества программных средств. Общие положения

22. http://softwarequality.narod.ru/qualitymodelstandards.html

23. http://www.interface.ru/fset.asp?Url=/misc/qs.htm

24. http://www.rol.ru/news/it/press/cwm/25_96/teh.htm

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

26. ISO/IEC 15504: Information Technology – Software Process Assessment

27. http://www.sqi.gu.edu.au/spice

28. http://www.cse.dcu.ie/spire

29. Коган Б.И. Автоматизация оценивания качества программного обеспечения // http://www.febras.ru/~conf/seminar/kogan.html

30. Новичков А., Шамрай А., Черников А. Метрики кода и их практическая реализация в Subversion и ClearCase // http://cmcons.com/articles/CC_CQ/dev_metrics/mertics_part_1/

31. http://kapustin_andrey.boom.ru/Materials/Metrics2.htm

32. Поморова О.В., Говорущенко Т.О. Аналіз методів та засобів оцінки якості програмних систем // Радіоелектронні і комп'ютерні системи. – Харків: НАУ "ХАІ", 2009. – № 6. С.148-158.

33. http://www.construx.com

34. Орлов С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем: Учебник для ВУЗов. – СПб.: Питер, 2004. – 527 с.

35. International Function Point User's Group, http://www.ifpug.org/

36. Петрухин В.А., Лаврищева Е.М. Методы и средства инженерии программного обеспечения // http://www.intuit.ru/department/se/swebok/10/1.html

37. International Function Point User's Group reference to function point spread-sheets, http://ifpug.org/home/docs/freebies.html

38. Boehm B. Software Engineering Economics – NJ: Prentice Hall, 1981. – 392 p.

39. Бахтизин В.В., Глухова В.А. Применение метрик сложности при разработке программных средств // http://www.giac.unibel.by/docs/pdf/1-2005/s10-1-2005.pdf

40. Aggarwal K. K., Yogesh Singh, Arvinder Kaur, and Ruchika Malhotra. Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics // PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY VOLUME 15 OCTOBER 2006 ISSN 1307-6884 // http://www.waset.org/pwaset/v15/v15-52.pdf

41. Janusz Zalewski, Andrew J. Kornecki, Henry L. Pfister. Numerical Assessment of Software Development Tools in RealTime Safety Critical Systems Using Bayesian Belief Networks // http://www.proceedings2006.imcsit.org/pliks/194.pdf

42. Липаев В.В. Выбор и оценивание характеристик качества программных средств: Методы и стандарты. – М.: Синтег, 2001. – 224 с.

Надійшла 13.12.2009 р.