УДК 681.518

М. І. БЕЗМЕНОВ, канд. техн. наук, професор НТУ «ХПІ»,
О. М. ЛАНСЬКИХ, магістрант НТУ «ХПІ»,
В. Г. БОРИСОВ, канд. техн. наук, доцент НТУ «ХПІ»

МЕТРИКИ ЯК ОЦІНКА МОДЕЛЕЙ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МЕДИЧНОГО ОБЛАДНАННЯ

Розглянуто види моделей якості програмного забезпечення. Обговорена важливість оцінки моделі якості програмного забезпечення у розробці медичних обладнань. Розглянуті метрики програмного забезпечення як кількісні показники оцінки моделі якості. Наведений приклад найбільш поширеної метрики з коду. Зроблено висновки.

Рассмотрены виды моделей качества программного обеспечения. Обсуждена важность оценки моделей качества программного обеспечения в разработке медицинских приборов. Рассмотрены метрики программного обеспечения как качественные показатели оценки модели качества. Рассмотрен пример наиболее распространенной метрики по коду. Сделаны выводы.

Software quality models have been considered. The importance of software quality models evaluation in medical devices development has been discussed. Software metrics have been considered as qualitative indicators of software quality model. Conclusions are made.

Вступ

Технологія розробки медичних приладів змінювалась упродовж десятиріч. Найбільший крок був зроблений під час 1960–1970 років, коли ставили якість та ефективність на все більш значуще місце. Від тоді і до сих пір вчені працюють над методами покращення якості та ефективності.

Існує три класи медичних приладів:

Клас 1 – несе мінімальну потенційну шкоду хворій людині.

Клас 2 – несе шкоду хворій людині, але не впливає на її життя, загальні заходи контролю недостатні для забезпечення безпеки та ефективності.

Клас 3 – використання медичного прилада, у випадку неякісної його розробки, може вбити людину.

Успішний розвиток медичного обладнання найбільш за все залежить від вдосконаленості та якості вбудованого в нього програмного забезпечення (ПЗ). Якість медичного прилада найбільш за все залежить від рішень, прийнятих на ранніх ступенях розробки програмного забезпечення. Модель якості повинна постійно бути у циклічному процесі збору даних, аналізу, ідентифікації ризику, приймання та виконання коригувальних дій на протязі усіх етапів розробки програмного забезпечення: аналіз вимог, планування, кодування, тестування та випуск [1].

1. Використання моделі якості

Якість ПЗ повинна оцінюватися з використанням моделі якості. Модель якості [2] визначають після розроблення вимог до якості готового ПЗ та проміжного продукту, – вона є сукупністю атрибутів якості ПЗ, класифікованих в ієрархічну деревоподібну структуру характеристик і підхарактеристик. Верхній рівень цієї структури складається з характеристик якості, а нижній – з атрибутів якості ПЗ. Ця ієрархія не є строгою, тому що деякі атрибути можуть входити більш ніж в одну характеристику. Частіше за все ця ієрархічна структура використовується у вигляді UML-діаграм (*англ.*: Unified Modeling Language – Уніфікована Мова Моделювання).

Розглянемо поняття моделі якості [3] як модель внутрішньої [4] та зовнішньої [5]. Визначають шість основних характеристик якості ПЗ: функціональність, надійність, зручність використання, раціональність, супроводжуваність і переносимість (табл. 1).

Таблиця 1

Основні характеристики моделі якості ПЗ

Внутрішня та зовнішня якість					
Функціональність	Надійність	Зручність використання	Раціональність	Супроводжуваність	Переносимість
1) Функціональна повнота 2) Правильність 3) Здатність до взаємодії 4) Захищеність 5) Узгодженість функціональності	1) Безвідмов-ність 2) Стійкість до відхилень 3) Відновлю-ваність 4) Узгодженість надійності	1) Зрозумілість 2) Придатність до вивчення 3) Зручність інтерфейсу 4) Привабливість 5) Узгодженість зручності використання	1) Часова раціональність 2) Використо-ваність ресурсів 3) Узгодженість раціональності	1) Аналізованість 2) Змінюваність 3) Стабільність 4) Тесто-придатність 5) Узгодженість супроводжуваності	1) Адаптованість 2) Налагоджуваність 3) Сумісність 4) Взаємозамінність 5) Узгодженість переносимості

Розглянемо докладніше кожну з характеристик моделі якості програмного забезпечення.

Функціональність – здатність ПЗ забезпечити функції, які виконують заявлені потреби та потреби, що маються на увазі при використанні ПЗ в заданих умовах. Вона включає такі функції:

- функціональна повнота – здатність ПЗ забезпечити відповідний набір функцій для заданих завдань та цілей користувача;
- правильність – здатність ПЗ забезпечувати вірні або припустимі результати чи дії з необхідним ступенем точності;
- здатність до взаємодії – здатність ПЗ взаємодіяти із зазначеними системами;
- захищеність – здатність ПЗ забезпечити захист інформації та даних (при їхньому зберіганні і передачі) таким чином, що неавторизовані користувачі та системи не будуть мати можливості до їх читання і модифікації, у той час як авторизованим користувачам і системам не буде відмовлено у доступі до них;
- узгодженість функціональності – здатність ПЗ до дотримання відповідних стандартів, угод, положень законів або подібних рекомендацій, що стосуються функціональності.

Надійність – група властивостей, що обумовлює здатність програмного забезпечення зберігати працездатність та перетворювати вихідні дані в очікуваний результат у заданих умовах за встановлений час. Ця властивість включає такі підхарактеристики:

- безвідмовність – здатність ПЗ уникати відмов (функціонування без відмов), які є результатом наявності дефектів у ПЗ;
- стійкість до відхилень – здатність ПЗ підтримувати необхідний рівень працездатності у випадках прояву програмних дефектів або порушення його інтерфейсу;
- відновлюваність – здатність ПЗ відновлювати заданий рівень працездатності, а також відновлювати дані, які безпосередньо ушкоджені у випадку виникнення відмови, після перезапуску ПЗ (автоматичного або оператором);
- узгодженість надійності – здатність ПЗ дотримуватись відповідних стандартів, угод, положень законів або подібних рекомендацій, що стосуються надійності.

Зручність використання – здатність ПЗ бути зрозумілим, придатним до вивчання і привабливим для користувача при його використанні в заданих умовах. Вона включає такі підхарактеристики:

- зрозумілість – здатність ПЗ забезпечити користувачеві зрозумілість того, чи може ПЗ бути використано і яким саме чином, для конкретних завдань і умов застосування;
- придатність до вивчення – здатність ПЗ до надання користувачеві можливості його вивчення;
- зручність інтерфейсу для користування – здатність ПЗ до надання користувачеві можливості управління та контролю за його роботою;
- привабливість – здатність ПЗ бути привабливим для користувача (відноситься до графічного інтерфейсу);
- узгодженість використання – здатність ПЗ до дотримання відповідних стандартів, угод, положень законів або подібних рекомендацій, що стосуються використання.

Раціональність – здатність ПЗ забезпечувати відповідну (допустиму) продуктивність із урахуванням займаних ресурсів у заданих умовах. Її під-характеристиками є:

- часова раціональність – здатність ПЗ забезпечити відповідний (допустимий) час відгуку, обробку та пропускну здатність при виконанні його функцій у заданих умовах;
- використовуваність ресурсів – здатність ПЗ використовувати відповідну (допустиму) кількість і тип ресурсів при виконанні його функцій у заданих умовах;
- узгодженість раціональності – здатність ПЗ до дотримання відповідних стандартів або подібних рекомендацій, що стосуються ефективності.

Супроводжуваність – здатність ПЗ до модифікації, що може містити у собі виправлення, поліпшення або адаптацію ПЗ до змін середовища, вимог або функціональних специфікацій. Вона включає такі підхарактеристики:

- аналізованість – здатність ПЗ до діагностування дефектів або причин відмов ПЗ, а також ідентифікації частин програми, які мають потребу в корекції;
- змінюваність – здатність ПЗ до надання можливості для виконання заданої модифікації його коду, структури та алгоритмів або програмної документації;
- стабільність – здатність ПЗ до запобігання неочікуваних ефектів від модифікацій;
- тестопридатність – здатність ПЗ до валідації його змін (модифікацій);
- узгодженість супроводжуваності – здатність ПЗ до дотримання відповідних стандартів або подібних рекомендацій, що стосуються супроводжуваності.

Переносимість – здатність ПЗ бути перенесеним з одного організаційного, апаратного або програмного середовища в інше, що включає такі підхарактеристики:

- адаптованість – здатність ПЗ бути адаптованим до різних середовищ без застосування дій або засобів, відмінних від тих, що передбачено для цієї мети в розглянутому ПЗ, адаптованість містить у собі масштабованість внутрішніх елементів ПЗ (екранних полів, таблиць, форматів звітів тощо);
- налагоджуваність – здатність ПЗ до інсталяції в заданому середовищі;
- сумісність (безконфліктність) – здатність ПЗ без конфліктів співіснувати з іншим незалежним програмним забезпеченням у загальному середовищі, що спільно використовує загальні ресурси;
- взаємозамінність – здатність ПЗ до використання замість іншого заданого ПЗ з тією ж метою та у тому ж середовищі;
- узгодженість переносимості – здатність ПЗ до дотримання відповідних стандартів або подібних рекомендацій, що стосуються переносимості.

Модель якості у використанні визначає чотири характеристики, наведені на рис. 1.

Розглянемо характеристики моделі якості у використанні більш детальніше.

Ефективність – це здатність програмного забезпечення дозволяти користувачеві досягати зазначених цілей з точністю та повнотою в заданому контексті використання.

Продуктивність – це здатність програмного забезпечення дозволяти користувачеві витрачати відповідну кількість ресурсів щодо ефективності, досягнутої в заданому контексті використання.

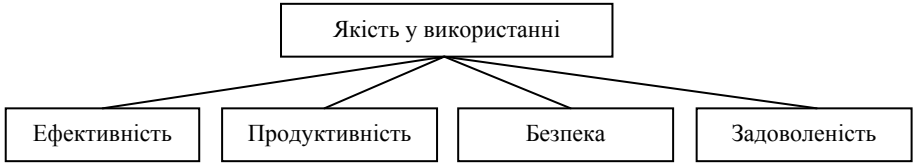

Рис. 1. Модель якості у використанні

Безпека – це здатність програмного забезпечення досягати прийнятних рівнів ризику шкоди людям, бізнесу, ПЗ, майну або навколишньому середовищу в заданому контексті використання.

Задоволеність – це здатність програмного забезпечення задовольняти користувачів у заданому контексті використання.

2. Застосування характеристик якості програмного забезпечення

Для ефективного вибору та застосування характеристик (показників) якості ПЗ [7], що представлені у попередньому розділі, їх необхідно класифікувати за типами на:

- описові – вони описують набір засобів і загальні характеристики об'єкта, його функції, безпеку, захищеність і важливість;
- кількісні – їх можна виміряти та чисельно зіставити з вимогами;
- якісні – ті, що визначають експертним методом.

Кількісний метод, як оцінка якості ПЗ, є одним з найпоширеніших. Кількісні величини повинні вибиратися та фіксуватися і супроводжуватися методикою їх чисельних вимірів при випробуваннях. Для кожної з них повинен бути встановлений ряд допустимих чисельних значень та шкали характеристик надійності. Одним із найефективніших засобів оцінки якості кількісним методом є використання метрик.

3. Метрики якості програмного забезпечення

Метрика програмного забезпечення [8] – це міра, що дозволяє отримати чисельне значення деякої властивості програмного забезпечення або його специфікації. Процес вибору та встановлення метрик є найважливішим.

3.1. Процеси вибору та встановлення метрик

На першому етапі проводиться вибір та обґрунтування вихідних даних, що характеризують загальні особливості і етапи життєвого циклу проекту ПЗ та його споживачів, кожний з яких впливає на певні характеристики якості комплексу програм. Далі проводиться вибір, встановлення та затвердження конкретних метрик і шкал вимірювання показників якості проекту для їх наступного оцінювання і зіставлення з вимогами у процесі кваліфікаційних випробувань або сертифікації на певних етапах життєвого циклу програмного забезпечення.

Для якісних характеристик визначають та зафіксовують у специфікаціях опис умов, за яких необхідно вважати, що дана характеристика реалізується в ПЗ. Результати аналізу і вибору номенклатури та метрик характеристик якості проекту ПЗ повинні бути задокументовані в специфікаціях вимог, погоджені з їх споживачами та затверджені замовником проекту.

3.2. Вибір і використання метрик якості програмного забезпечення

Засоби вибору та використання метрик містять у собі послідовні реалізації наступних процедур:

1) визначення цільових характеристик якості ПЗ;
2) визначення складу метрик, що оцінюють цільові характеристики якості ПЗ;
3) підготовка до оцінювання;
4) задавання цільових значень і шкал допусків:
   a) визначення необхідної програмної документації та складу вхідних параметрів для розрахунку метрик;
   b) уточнення фактичного складу метрик, що використовуються;
   c) визначення того, на якому етапі життєвого циклу програмного забезпечення використання яких метрик буде найбільш корисним;
   d) оцінювання вхідних параметрів і розрахунок метрик;
5) аналіз результатів оцінювання;
6) узагальнення результатів оцінювання, побудова і згортка метричних діаграм;
7) порівняння розрахованих значень із заданими цільовими значеннями;
8) аналіз фактичної повноти та імовірності оцінювання;
9) формування експертного висновку (існуючих ризиків) за результатами оцінювання;
10) створення плану дій з метою рішення проблем та видалення ризиків, а також оцінювання термінів отримання результатів після виконаних корегуючих дій.

Зазвичай, на етапі підготовки оцінювання, при визначенні шкал допусків, використовують три зони які далі використовуються на етапі аналізу:

1) жовта – говорить про наявність допустимих значень, тобто та частина програмного забезпечення, на яку були накладені метрики, має задовільні результати та може бути виключена з наступного кроку аналізу якості;
2) зелена – знаходиться на границі між зеленою та червоною зонами в залежності від складності проекту та може бути включена на наступному кроці аналізу;
3) червона – вказує або на наявність проблем і ризиків у тій частині програмного забезпечення, на яку були накладені метрики (у цьому випадку обов’язково розглядається на етапі аналізу), або на роботу з невірними даними.

Існує величезна кількість метрик для оцінювання якості програмного забезпечення, серед яких можна виділити такі:

1) метрики коду;
2) метрики дефектів;
3) метрики надійності;
4) метрики стійкості до відхилень;
5) метрики зрозумілості;
6) метрики придатності до вивчання;
7) метрики захищеності,
8) метрики переносимості та багато інших.

Задачею аналітика є вибір лише тих метрик, які найбільш за все підходитимуть до конкретної специфіки проекту. Звісно, найбільша увага завжди приділяється якості написання коду програмного забезпечення.

Одна з найпоширеніших метрик з коду є цикломатична складність коду.

Цикломатична складність (1) частини вихідного коду [8] – це кількість лінійно незалежних шляхів всередині коду. Наприклад, якщо вихідний код не містить точок прийняття рішень (наприклад, оператор IF або цикл FOR), складність дорівнюватиме одиниці, тому що існує тільки один можливий шлях крізь програмний код. Якщо програмний код має один оператор IF, який містить тільки одну умову, програма буде мати два можливі шляхи крізь код, а саме, перший шлях, де оператор IF оцінюється як TRUE, та другий шлях, де оператор IF оцінюється як FALSE.

Цикломатична складність коду, як правило, розраховується шляхом створення графа ресурсного коду. Вершинам цього графа відповідають рядки коду, а ребрами є можливі шляхи коду.

Цикломатична складність розраховується так:

$$M = E - N + P + R, \qquad (1)$$

де M – цикломатична складність;

E – кількість вершин графа;

N – кількість ребер графа;

P – кількість з'єднаних компонентів;

R – кількість зворотних стверджень.

Частіше за все ця метрика збирається автоматично за допомогою спеціальних програм, наприклад, таких як Source Monitor. Наступним кроком буде визначення шкал допусків для подальшого аналізу даних (табл. 2).

Таблиця 2

Шкали допусків

Жовта зона	Зелена зона	Червона зона
Менше 3	Більш 3 та менше 6	Більш 6

Розглянемо графік значень отриманої метрики, які збиралися протягом п'яти місяців упродовж постійного процесу розробки програмного забезпечення (рис. 2).

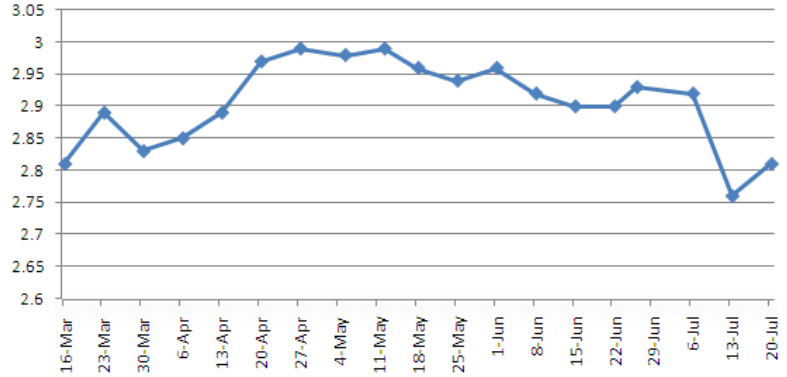

Рис. 2. Графік цикломатичної складності програмного коду

Загальна тенденція значення цикломатичної складності, як це показано на графіку, не переходить у зелену зону. Але у період з березня по травень відбувався видимий зріст цикломатичної складності, значення якої наближувалось до зеленої зони. Відповідні міри були прийняті відразу ж, як тільки ця тенденція була виявлена. Був проведений процес Code Review, у якому більш досвідчені програмісти проглядали та аналізували програмний код тих програмістів, які займались розробкою ПЗ. Після цього було складено план з Code Refactoring (процес зміни програмного коду без зміни його загальної поведінки), де програмний код був перебудований та покращений. У результаті цих вжитих дій, починаючи з червня, спостерігається тенденція до спаду цикломатичної складності, як показано на графіку, що говорить про нормалізацію даних та про відсутність необхідності перебудовувати програмний код. Тим самим, була попереджена якась частина дефектів ПЗ, адже ті класи чи функції, у яких значення цикломатичної складності перевищує порогові, є найбільш вірогідними джерелами дефектів.

Висновки

Успішний розвиток медичного обладнання найбільш за все залежить від вдосконаленості та якості вбудованого в нього ПЗ. Оскільки кількісні оцінки виявилися настільки потужними в інших науках, у галузі інформатики було витрачено немало часу та зусиль на те, щоб розвити аналогічні підходи у процесі розробки ПЗ. Том Де Марко, один із науковців, який перший описав метрики ПЗ як метод оцінки якості, якось заявив: «Ви не зможете контролювати те, що ви не можете виміряти» [1]. Ці слова підтверджують важливість використання метрик, як кількісний метод оцінки якості ПЗ.

Розглянутий приклад використання цикломатичної складності як метрики свідчить про необхідність аналізу процесу розробки програмного коду та його рефакторингу на протязі усіх етапів розробки програмного забезпечення, а саме під час аналізу вимог, планування, кодування, тестування та випуску. Приклад підтверджує можливість виявити аномалії та відхилення у програмному продукті ще у процесі кодування, а не вже на етапі тестування або навіть випуску. Це, в свою чергу, попереджає значний відсоток дефектів ПЗ у вбудованих медичних системах, адже дефекти ПЗ можуть призвести до неадекватної поведінки медичного прилада, що в свою чергу може негативно вплинути на життя людини.

Список літератури: ***1.*** *DeMarco Tom*, Controlling Software Projects: Management, Measurement and Estimation 2002. – 279 p. ***2.*** *McCabe,* A Complexity Measure 1976. – 154 p. ***3.*** *ISO/IEC 9126-1:2001* Software Engineering – Product Quality – Part 1: Quality model (Інжиніринг програмного забезпечення – Якість продукту – Частина 1: Модель якості). ***4.*** *ISO/IEC TR 9126-3:2003* Software Engineering – Product Quality – Part 3: Internal metrics (Інжиніринг програмного забезпечення – Якість продукту – Частина 3: Внутрішні метрики). ***5.*** *ISO/IEC TR 9126-2:2003* Software Engineering – Product Quality – Part 2: External metrics (Інжиніринг програмного забезпечення – Якість продукту – Частина 2: Зовнішні метрики). ***6.*** *ISO 9001:2008* Quality Management System – requirements (Системи менеджменту якості – вимоги). ***7.*** *Carnegie Mellon University, Mark C. Paulk, Bill Curtis*. The Capability Maturity Model: Guideline for Improving the Software Process 2001. – 434 p. ***8.*** *ISO/IEC TR 9126-4:2004* Software Engineering – Product Quality – Part 4: Quality in use metric (Інжиніринг програмного забезпечення – Якість продукту – Частина 4: Метрики якості у використанні).

Надійшло у редколегію 29.11.09