УДК 681.3.048

О.В. Щербаков, Є. С. Луценко, Ю.І. Скорін

Харківський національний економічний університет, Харків

МЕТОДИКА КІЛЬКІСНОЇ ОЦІНКИ ЕФЕКТИВНОСТІ РОБОТИ ІНЖЕНЕРА ІЗ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У статті доводиться значимість оцінки ефективності роботи тестувальників для управління процесом розробки програмного забезпечення. Розглянуто існуючі підходи та технології тестування. Проведено аналіз методів оцінки роботи інженера з забезпечення якості програмного забезпечення. Запропонована методика кількісні оцінки ефективності тестування програмних продуктів на підставі математичної моделі. Зроблені висновки щодо впровадження запропонованої методики в процесі тестування.

Ключові слова: *програмне забезпечення, процес розробки ПЗ, тестування, інженер з забезпечення якості програмного забезпечення, оцінка ефективності.*

Вступ

Тестування програмного забезпечення (ПЗ) відіграє значну роль при його розробці, як комплекс заходів для отримання якісного програмного продукту. Якість ПЗ, створеного розробниками, у значній мірі залежить від роботи тестувальників.

Тестування програмного забезпечення - це перевірка відповідності реальної та очікуваної поведінки програми, що здійснюється на кінцевому наборі тестів, що обрані належним чином [1].

Якість програмного забезпечення – характеристика програмного забезпечення як ступеня його відповідності вимогам [2]. У широкому розумінні процес тестування – це одна з методик контролю якості, яка включає планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) та аналіз отриманих результатів (Test Analysis) [1].

Дефект – це невідповідність продукту встановленим вимогам [3]. В якості встановлених вимог можуть виступати різноманітні джерела: специфікація, здоровий глузд, усна домовленість тощо.

Відповідно до наведених означень метою тестування є пошук дефектів, тобто, фіксація кожної розбіжності між реальною та очікуваною поведінкою програми. Може виникнути заперечення щодо некоректності такого формулювання, при розгляді проблеми у разі, якщо система «не має дефектів». Але, нереальність існування систем з повною відсутністю дефектів [4] є обґрунтуванням коректності такого означення. Загалом доцільно розглядати тестування як порядок фіксації результатів перевірки відповідності реальної та очікуваної поведінки програми (на підставі виявлених дефектів), що передбачає наявність особи, яка буде виконувати пошук дефектів. Майерс, який розглядає тестування, як пошук дефектів, наводить таке означення: тестування – це процес виконання програми з метою виявлення помилок [5].

Таким чином тестування є складним та дуже важливим процесом, що потребує відповідного контролю, здійснення якого значно ускладнює велика кількість суб'єктивних та об'єктивних факторів, наприклад врахування якості роботи програмістів та менеджерів, завантаження тестувальника (його участь у декількох проектах), особливості функціональної області..

Мета статті – проаналізувати існуючі технології тестування ПЗ і підходи до оцінки якості цього процесу та запропонувати нову методику кількісної оцінки ефективності роботи інженера з якості ПЗ.

Основна частина

Існує велика кількість різноманітних технологій тестування [6,7]. Умовно їх можна віднести до статичних або до динамічних.

Статистичне тестування - це процес, який зазвичай асоціюють з аналізом ПЗ. Статичним тестуванням користуються для верифікації практично будь-якого артефакту розробки: програмного коду компонент, вимог, системних специфікацій, функціональних специфікацій, документів проектування та архітектури програмних систем і їх компонентів і т.д. Використання статичних методів тестування - один з найбільш ефективних способів виявлення дефектів на ранніх стадіях розробки ПЗ. Статистичне тестування - це єдиний спосіб тестування без запуску програмного коду програми.

Динамічне тестування - процес тестування, що здійснюється над працюючою системою або підсистемою. Таке тестування не може бути здійснено без запуску програмного коду.

Динамічне тестування складається з:

запуску системи або підсистеми;

виклику необхідних функціональних елементів або модулів;

порівняння через графічний інтерфейс користувача поведінки системи з очікуваним результатом поведінки.

© О.В. Щербаков, Є. С. Луценко, Ю.І. Скорін

При застосуванні різних методів тестування використовуються відповідні технології тестування. Серед методів тестування зазвичай виділяють два найпоширеніших:

метод «чорного ящика» («black-box» testing);

метод "білого ящика» («white-box» або «glass-box» testing).

Різниця між тестуванням «білого ящика" і "чорного ящика" має місце на будь-якому рівні. На перший погляд, тестування внутрішніх компонент є тестуванням "білого ящика" але у той же час, з точки зору розробника, сам компонент може бути протестований як методом «чорного», так і «білого» ящиків.

Тести розділяють за завданнями (які вирішуються з їхньою допомогою) та за використовуваною технікою. Різниця завдань тестування призводить до необхідності використовувати різноманітні типи (види) тестування. Прийнято поділяти тестування на види за наступними категоріями:

- по об'єктах (елементах) тестування (поділ тестування на рівні);

- по глибині тестування (поділ тестових випробувань на типи проводиться в залежності від кількості часу та обсягу тестуємих компонент програмного продукту).

Модульне тестування (Автономне або Unit-тестування). Вхідними вимогами є архітектура компонентів або модель "нижнього рівня" системи (Component Design або Low Level Design);об'єкт тестування - Розроблені компоненти. На даному рівні тестуються окремо невеликі елементи системи, максимально відокремлені від інших елементів і, в той же час, придатні для тестування. Таке тестування проводиться слідом за розробкою кожного з елементів та направлено на оцінку відповідності функціональності кожного з компонентів спроектованої "моделі компонентів".

Комплексне тестування (Складальне тестування, integration testing або interface testing). Вхідними вимогами є архітектура системи або модель "верхнього рівня" системи (System Design або High Level Design);об'єкт тестування - Зібрана з компонентів система чи підсистема. На даному рівні тестуються об'єднані елементи (компоненти або підсистеми) загальної системи (найчастіше деяка взаємодіюча між собою група елементів).

Комплексне тестування спрямоване не на перевірку функціонування кожного з компонентів, а на перевірку взаємодії компонентів відповідно до «Архітектури системи». Тести даного рівня зазвичай перевіряють всі інтерфейси взаємодії між компонентами, які визначені в системній архітектурі, до тих пір, поки усі компоненти не будуть розроблені, налагоджені та протестовані у сукупності.

Системне тестування (system testing). Вхідними вимогами є системні специфікації (System Specification); об'єкт тестування - розроблена система. Після того, як система зібрана з складових компонентів, вона повинна бути протестована на відповідність "Системним специфікаціям", тобто повинна бути здійснена перевірка щодо реалізації всіх функціональних та нефункціональних вимог до розроблюваної системі.

На даному рівні тестується програма або система (один або більше додатків) у цілому.

Приймальне тестування (Acceptance testing). Вхідними вимогами є вимоги (Requirements); об'єкт тестування - розроблена система. На даному рівні завершений додаток (система) тестується замовником, кінцевими користувачами або відповідними уповноваженими з метою визначення відповідності системи "вимогам замовника" та її готовності до впровадження. Приймальні випробування оформлюють процес передачі продукту від розробника замовнику. Залежно від особливостей продукту і від вимог замовника вони можуть проводитися в різній формі. Наприклад, у вигляді альфа-або бета-тестування. Приймальне тестування є подібним системному тестуванню, але має наступні відмінності:

- системне тестування перевіряє, що розроблена система відповідає вимогам специфікації;

- приймальне тестування перевіряє, що розроблена система задовольняє вимогам замовника, роблячи акцент на потреби кінцевих користувачів в даній предметній області.

Операційне тестування (Release Testing). Вхідними вимогами є бізнес-модель (Business Case або Business Model); об'єкт тестування - розроблена система. Навіть якщо система задовольняє усі вимоги, важливо переконатися у тому, що вона задовольняє потребам користувача і виконує свою роль у середовищі своєї експлуатації, як це було визначено в "Бізнес моделі" системи. Слід врахувати, що і бізнес модель може містити помилки. Тому важливо проводити операційне тестування як фінальний крок валідації.

Крім цього, тестування в середовищі експлуатації дозволяє виявити і нефункціональні проблеми, такі як: конфлікт з іншими системами, суміжними в області бізнесу або в програмних та електронних оточеннях; недостатня продуктивність системи в середовищі експлуатації та інше [8].

Очевидно, що знаходження подібних речей на стадії впровадження - критична і високовартісна проблема. Тому важливо проведення не тільки верифікації, а й валідації, починаючи з самих ранніх етапів розробки ПЗ [9].

Таким чином тестування (забезпечення якості) представляє собою складний механізм, який потребує особливого контролю ефективності роботи його виконавців для того щоб у майбутньому можливо було уникнути різноманітних колізій тим самим

підвищуючи ефективність та відповідно якість тестування [2, 4].

Кількісна оцінка якості роботи тестувальника є надзвичайно складною задачею, насамперед, через наявність чималої кількості факторів, що впливають на якість його роботи, більшість яких не піддається кількісному оцінюванню. Крім того, дуже важко кількісно порівняти якість роботи тестувальників, оскільки підходи до визначення важливості різних критеріїв щодо оцінки якості їх роботи є суб'єктивними.

Пропонується нова методика кількісної оцінки роботи тестувальників, яка враховує такі фактори:

1. Кількість дефектів програмного продукту, знайдених тестувальником.

2. Значимість виявлених дефектів на підставі баг-репорту відповідно до встановленої градації помилок.

3. Тривалість періоду, протягом якого тестувальник виявив дефекти програмного продукту.

Для побудування звичайної диференціальної моделі використовуємо такі означення:

$Q(t)$ – приведена кількість дефектів, виявлених за певний проміжок часу;

t – час, облік якого ведеться у робочих днях;

$\frac{dQ}{dt}$ – «швидкість» виявлення дефектів програмного продукту, тобто кількість помилок, знайдених в одиницю часу.

Вибір виду диференціального рівняння відносно функції $Q = Q(t)$ ґрунтується на таких міркуваннях, що є наслідком аналізу реальних результатів роботи тестувальника.

1. Від початку тестування функція $Q = Q(t)$ монотонно зростає.

2. «Швидкість» зростання функції $Q = Q(t)$ уповільнюється ближче до завершення процесу тестування, тобто обернено пропорційна Q.

3. На початковому етапі тестування $\frac{dQ}{dt}$ змінюється за квазілінійним законом.

Зазначені припущення щодо зміни $Q = Q(t)$ добре узгоджуються з особливостями поведінки функції $y = \ln(x)$, тому диференціальне рівняння записуємо у вигляді:

$$\frac{dQ}{dt} = \ln Q \qquad (1)$$

з початковою умовою:

$$Q(t_0) = Q_0 \qquad (2)$$

Розв'язок цієї задачі Коші здійснюється таким чином:

$$\int \frac{dQ}{\ln Q} = \int dt + c \qquad (3)$$

або

$$t = \int \frac{dQ}{\ln Q} - c \qquad (4)$$

де c – константа інтегрування ; а $I = \int \frac{dQ}{\ln Q}$ - «інтегральний логарифм» який елементарно не інтегрується.

Розв'язок інтегрального логарифму можна репрезентувати у вигляді нескінченого ряду:

$$I = \int \frac{dQ}{\ln Q} = \ln\ln Q + \ln Q + \frac{(\ln Q)^2}{2 \cdot 2!} + \frac{(\ln Q)^2}{3 \cdot 3!} + ... \qquad (5)$$

Якщо обмежимось першими числами ряду (залишаємо чотири члени), маємо наближену форму розв'язку:

$$I \approx \ln\ln Q + \ln Q + \left(\frac{1}{4} + \frac{1}{81}\right)(\ln Q)^2$$

Після очевидних перетворень одержуємо:

$$I = \ln\ln Q + \ln Q + 0{,}137(\ln Q)^2 \text{ , або}$$

$$I = \ln(Q\ln Q) + 0{,}137(\ln Q)^2 \qquad (6)$$

Константу інтегрування C обираємо у такому вигляді $C = -\ln D$, тоді на підставі (4) та (6) маємо:

$$t = \ln(DQ\ln Q) + 0{,}137(\ln Q)^2 \qquad (7)$$

Чисельне значення константи D знаходимо відповідно до початкової умови, що узгоджується з доступними даними результатів роботи тестувальників, а саме, $Q(3) = 18$:

$$3 = \ln(D \cdot 18 \cdot \ln 18) + 0{,}137(\ln 18)^2.$$

Звідки маємо $D = 0{,}126$.

Остаточно одержуємо формулу:

$$t = \ln(0{,}126Q \cdot \ln Q) + 0{,}137(\ln Q)^2 \qquad (8)$$

Для оцінки якості роботи тестувальника на підставі формули (8) необхідно обчислити приведену кількість виявлених дефектів з врахуванням градації їх значимості, що наведена у табл. 1.

Таблиця 1

Градація значимості дефектів

Градація значимості дефектів	Кількість дефектів, знайдених тестувальником
Блокуючий	Blt
Критичний	Crt
Значний	Mat
Незначний	Mit
Тривіальний	Trt
Разом	Nt

Коефіцієнти, що враховують значимість виявлених помилок, можна обирати відповідно до перших п'яти чисел ряду Фібоначчі (1, 1, 2, 3, 5), сума яких становить 12, а саме:

$$k_1 = k_2 = \frac{1}{12};\ k_3 = \frac{1}{6};\ k_4 = \frac{1}{4};\ k_5 = \frac{5}{12}.$$

Тоді кількість дефектів, виявлених тестувальником обчислюється за формулою:

$$Q_m = \frac{1}{12}(Trt + Mit) + \frac{1}{6}Mat + \frac{1}{4}Crt + \frac{5}{12}Blt. \quad (9)$$

Далі за формулою (8) визначається:

$$t^* = \ln(0{,}126 Q_m \ln Q_m) + 0{,}137(\ln Q_m)^2.$$

Якщо тривалість часу t_m, фактично витраченого тестувальником на виявлення дефектів у кількості Q_m, не перевищує значення параметру t^* більше ніж на 5%, роботу тестувальника можна визнати ефективною. При цьому, чим більше зменшується t_m у порівнянні з t^*, тим ефективніше робота тестувальника.

Висновки

Тестування програмного забезпечення є досить складним і важливим етапом розробки програмного забезпечення. При цьому, оцінка ефективності роботи тестувальників ПЗ є необхідним елементом комплексу заходів щодо забезпечення якості програмних продуктів. Відсутність ефективних методів такої оцінки може негативно впливати на якість ПЗ. Запропонована методика дозволяє отримати кількісні оцінки ефективності тестування програмних продуктів на підставі математичної моделі. Отримані за допомогою даної методики оцінки дозволять підвищити ефективність як процесу тестування, так і процесу розробки ПЗ в цілому. Таким чином розроблена методика показує результати, що враховують більше факторів та відповідно мають більш високе практичне значення при застосуванні у реальному процесу тестування.

Список літератури

1. *Соммервилл, И. Инженерия программного обеспечения / пер. с англ. А.А. Минько, А.А. Момотюк, Г.И. Сингаевская. – М.: ВИЛЬЯМС, 2002. – 624 с.*
2. *Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. — Киев: Диа-Софт, 2001. – 544 с.*
3. *Калбертсон Роберт, Браун Крис, Кобб Гэри. Быстрое тестирование. — М.: «Вильямс», 2002. – 374 с.*
4. *Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. — М.: БИНОМ, 2008. — 368 с.*
5. *Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. — СПб.: Питер, 2004. — 320 с.*
6. *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. Москва, "Книга и бизнес", 2001.*
7. *Bourdonov I., Kossatchev A., Petrenko A., and Galter D.. KVEST: Automated Generation of Test Suites from Formal Specifications //FM'99: Formal Methods. LNCS 1708, Springer-Verlag, 1999, pp. 608—621.*
8. *M. Barnett, M. Fahndrich, P. de Halleux, F. Logozzo, N. Tillmann. Exploiting the Synergy between Automated-Test-Generation and Programming-by-Contract. Proc. of ICSE 2009, Vancouver, Canada, May 2009.*
9. *Д.Коул, Т. Горэм, М. МакДональд, Р. Спарджеон. Принципы тестирования ПО // Открытые системы. – №2. – 1998. – С. 60-63.*

Надійшла до редколегії 20.03.2012

Рецензент: канд. техн. наук, доц. І.О. Золотарьова, Харківський національний економічний університет, Харків.

МЕТОДИКА КОЛИЧЕСТВЕННОЙ ОЦЕНКИ ЭФФЕКТИВНОСТИ РАБОТЫ ИНЖЕНЕРА ПО ОБЕСПЕЧЕНИЮ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

А.В. Щербаков, Е.С. Луценко, Ю.И. Скорин

В статье доказывается значимость оценки работы тестировщиков для управления процессом разработки программного обеспечения. Рассмотрены существующие подходы и технологии тестирования. Проведен анализ методов работы инженера по обеспечению качества программного обеспечения. Предложена методика количественной оценки эффективности тестирования программных продуктов на основе математической модели. Сделаны выводы относительно внедрения предложенной методики в процессе тестирования.

Ключевые слова: *программное обеспечение, процесс разработки ПО, тестирование, инженер по обеспечению качества программного обеспечения, оценка эффективности.*

METHOD OF QUANTITATIVE EVALUATION OF THE EFFECTIVENESS OF THE QUALITY ASSURANCE SOFTWARE ENGINEER

O.V. Shcherbakov, E.S. Lutcenko, Y.I.Skorin

The article proves the importance of evaluation of the testers for the management of software development. The existing approaches and testing technology. The analysis methods of engineering to ensure software quality. The method of quantifying the effectiveness of software testing, based on a mathematical model. The conclusions concerning the introduction of the proposed method in the testing process.

Keywords: *software, development process, testing, quality assurance software engineer, performance evaluation.*