УДК 004.415.53

О.В. Щербаков, Є.С. Луценко

*Харківський національний економічний університет, Харків*

# ОЦІНКА ЕФЕКТИВНОСТІ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АНАЛІЗУ КІЛЬКОСТІ ТА КРИТИЧНОСТІ ЗНАЙДЕНИХ ДЕФЕКТІВ

*На підставі аналізу структури традиційного звіту про дефекти програмних продуктів, виявлених при тестуванні, створена модифікована структура звіту.*

*Запропонована структура звіту дозволяє не тільки поліпшити взаємодію розробників та тестувальників програмного забезпечення, а і використовується як засіб контролю за їх діяльністю. При цьому кожне поле звіту розглядається з точки зору його врахування для оцінки ефективності тестування.*

*Розроблена методика кількісної оцінки ефективності тестування програмних продуктів на підставі баг-репорту. Ефективність роботи тестувальника оцінюється за формулою, що одержана евристичним шляхом і враховує кількість та критичність помилок. Встановлено мінімальний достатній рівень ефективності тестування при використанні зазначеної методики.*

***Ключові слова:*** *тестування, методика оцінки ефективності тестувальника, звіт про помилку, модифікована структура звіту про помилки, засіб контролю діяльності тестувальників.*

## 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. Відповідно, актуальним є здійснення аналізу та контролю діяльності тестувальників шляхом оцінки ефективності тестування. Для початку, визначимось з термінологією, що стосується процесу тестування.

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 [5], який розглядає тестування, як пошук дефектів, наводить таке означення:

Тестування – це процес виконання програми з метою виявлення помилок.

**Метою статті** є удосконалення структури звіту про виявлені дефекти програмних продуктів та розробка методики кількісної оцінки роботи тестувальника.

## Традиційна структура звіту про знайдені дефекти

Відповідно до наведених означень результатом діяльності тестувальника є встановлення дефектів (усі види тестування передбачають виявлення дефектів). Тому, аналіз ефективності тестування ґрунтується на аналізі знайдених дефектів. З цією метою необхідно розглянути загальновідому структуру звіту про знайдені дефекти (табл. 1) [6, 7].

У системах, що автоматизують процес формування звіту про помилки, використовуються різні набори полів формування (окрім обов'язкових).

## Модифікація структури звіту про знайдені дефекти

Пропонується наступна структура звіту, що є настільки деталізованою, що дозволяє гнучко підлаштуватися під різні проекти (різні методології і, відповідно, різні ролі, вимоги до тестування):

– короткий опис дефекту;

Таблиця 1

Загальновідома структура звіту про знайдені дефекти

| Назва поля | Характеристика поля |
|---|---|
| Короткий опис (обов'язкове поле) | Короткий опис проблеми, що вказує на причину та тип ситуації, що призвела до помилки |
| Компонент додатку | Назва частини або функції продукту, що тестуються |
| Проект | Назва проекту, що тестується |
| Номер версії | Версія, в якій була знайдена помилка |
| Критичність (обов'язкове поле) | Атрибут, що характеризує вплив дефекту на працездатність додатку.<br>Найбільш розповсюджена п'ятирівнева система градації критичності дефектів:<br>S1 Блокуючий (Blocker)<br>Блокуюча помилка, що призводить додаток у непрацездатний стан, у зв'язку з чим подальша робота з системою, що тестується або її ключовими функціями становиться неможливою. Рішення проблеми необхідне для подальшого функціонування системи.<br>S2 Критичний (Critical)<br>Критична помилка, некоректна бізнес-логіка, прогалини у системі безпеки, проблема, що призвела до тимчасового падіння серверу або призводить у неробочий стан деяку частину системи без можливості розв'язання проблеми з використанням інших вхідних точок. Рішення проблеми необхідно для подальшої роботи з ключовими функціями системи, що тестується.<br>S3 Значний (Major)<br>Значна помилка, внаслідок якої частина основної бізнес-логіки працює некоректно. Помилка не критична або є можливість для роботи з функцією, що тестується, з використанням інших вхідних точок.<br>S4 Незначний (Minor)<br>Незначна помилка, що не порушує бізнес-логіки тієї частини додатку, що тестується, очевидна проблема інтерфейсу користувача.<br>S5 Тривіальний (Trivial)<br>Тривіальна помилка, що не стосується бізнес-логіки додатку, невдале відтворення проблеми, малопомітне по засобам інтерфейсу користувача, проблема сторонніх бібліотек або сервісів, проблема, що не впливає на загальну якість програмного продукт |
| Пріоритет | Атрибут, що вказує на черговість виконання задачі та усунення дефекту. Це інструмент менеджера з планування робіт. Чим вищий пріоритет, тим швидше необхідно усунути дефект.<br>Має три рівня градації:<br>P1 Високий (High)<br>Помилка повинна бути виправлена якомога швидше, тобто її наявність є критичною для проекту.<br>P2 Середній (Medium)<br>Помилка повинна бути виправлена. Її наявність не є критичною, але проблема потребує обов'язкового розв'язання.<br>P3 Низький (Low)<br>Помилка повинна бути виправлена, її наявність не є критичною та не потребує термінового рішення. |
| Статус | Визначається відповідно до процедури, що використовується та життєвого циклу дефекту (bug workflow and life cycle) |
| Автор | Особа, яка створює звіт про помилку |
| Призначений на | Особа, яка призначена для усунення дефекту |
| Оточення: операційна система, сервіс-пак, версія браузеру… | Інформація про оточення при якому був виявлений дефект: операційна система, сервіс пак, назва та версія браузера (при тестуванні веб-додатків) тощо |
| Опис (обов'язкове поле) | Кроки, згідно з якими легко відтворюється ситуація, що призвела до помилки; фактичний результат, отриманий після проходження кроків для відтворення; очікуваний правильний результат |
| Додатки (вкладені файли) | Лог-файл, знімок екрану або будь-який інший документ, що сприяє виявленню причини помилки та вказує на спосіб розв'язання проблеми |

– назва проекту;

– функціональна область, в якій був виявлений дефект: це поле присутнє у більшості систем автоматизованого формування звітів про дефекти;

– версія продукту;

– критичність дефекту;

– пріоритет дефекту;

– складність дефекту: показник ступеня складності виявлення відповідного дефекту, короткий опис дій, виконаних для його виявлення;

– статус дефекту;

– дані про осіб, які знайшли/занесли дефект;

– дані про осіб, які призначені для усунення дефекту;

– дані про керівників (менеджер проекту, начальник відділу тестування);

– операційне оточення;

– опис дефекту: інформація для покрокового відтворення дефекту, фактичний результат, очікуваний результат, джерела, що підтверджують статус дефекту (специфікація; усна домовленість; «здоровий глузд»), додаткова інформація та роз'яснення;

– джерело сценарію тестування: підготовлені скрипти, складені для автоматизованого виконання або для відтворення вручну, відповідно до плану тестування; інтуїтивна методика тестування;

– відділ тестування: відділ, працівники якого знайшли дефект.

– вкладення (будь-які файли, що можуть допомогти відтворити/довести/виправити дефект);

– час/дата занесення дефекту.

Дані звіту використовують для того, щоб якомога ефективніше протестувати продукт, знайти помилки та оперативно їх виправити.

На їх підставі проводиться відповідний аналіз і визначається роль кожного поля для оцінки ефективності роботи тестувальника.

1. Короткий опис дефекту. Перевірка того, наскільки тестувальники дотримуються принципу «Що? Де? Коли?» [8].

2. Проект. Під час проведення оцінки роботи тестувальника необхідно враховувати кількість та складність проектів, до реалізації яких його залучено. Складність залежить від якості програмування - чим гірша якість, тим складніше працювати з ПЗ тестувальнику відповідної області тестування. При цьому, якщо постійні збої, низька швидкодія, це звичні обставини для роботи команди тестування з навантаженням, то іншим, наприклад, тестувальникам інтерфейсів, ці фактори можуть ускладнювати роботу.

3. Функціональна область ПЗ. Виявляється кількість та пріоритет дефектів в окремих областях/функціях програмного продукту, перевіряється функціональна область, яка закріплена за окремим тестувальником, тобто склад дефектів, які знайшов тестувальник порівнюється зі складом дефектів виявлених кінцевим користувачем у цій же області (за умови досягнення рівня бета-стадії розробки програмного продукту).

4. Версія продукту. Вимоги до тестування залежать від стадії розробки, бо вимоги, яких дотримуються в активній стадії розробки, не співпадають з вимогами у фазі підтримки [9].

5. Критичність дефекту. Аналіз кількості дефектів без врахування їх критичності не є ефективним бо менша кількість виявлених дефектів з високою критичністю є значно кращим результатом ніж велика кількість виявлених некритичних дефектів.

6. Пріоритет дефекту. Перевірка взаємодії менеджера тестування та тестувальників, а саме, узгодженість виставленого менеджером пріоритету з критичністю, виставленою тестувальниками.

7. Складність дефекту. Обов'язково враховується для оцінки ефективності роботи тестувальника, бо кількість часу, який він витрачає на відстеження, аналіз, занесення кожного дефекту, безпосередньо залежить від його складності.

8. Статус дефекту. Дозволяє оцінити швидкість перевірки дефектів тестувальниками, наприклад, шляхом перевірки швидкості зміни статусу з «дефект виправлено» на будь-який інший: «перевірено», «відкрити заново» (при цьому необхідно обов'язково враховувати складність дефекту та його пріоритет). Також важливо визначати як швидко менеджер адміністрував дефект після його занесення згідно з пріоритетом та в якій кількості, протягом якого часу, і з яким пріоритетом існують відкриті дефекти в системі.

9. Дані про осіб, що знайшли/занесли дефект. Наприклад особи, діяльність (ефективність роботи) яких перевіряється.

10. Дані про осіб, які призначені для усунення дефекту. Репутація розробників (якість виправлення ними дефектів) для коректного аналізу діяльності тестувальників.

11. Операційне оточення. Враховується складність та особливості операційного оточення, наприклад, операційної системи, що в свою чергу має власні дефекти, які заважають роботі тестувальників.

12. Опис дефекту. Перевірка коректності опису дефекту, наявність правильних кроків для відтворення, відповідність структури опису загальним стандартам або стандартам конкретної компанії. Зокрема, перевірка якості та структури надання інформації покрокового відтворення дефекту. Крім того, врахування кількості дефектів, що не змогли бути відтворені розробниками через надання недостатньої інформації/незадовільне викладення.

13. Джерело сценарію тестування. Якщо використовувати дані цього поля, існує можливість визначення найбільш ефективних засобів тестування, які є джерелами сценарію. Наприклад, для одного проекту найбільш ефективним є проведення тестування за суворим сценарієм, а для іншого - «інтуїтивними методами»; в одному випадку ефективнішим є «ручне» тестування, а в іншому - «автоматизоване» тестування. Крім того, наявність такої статистики дозволяє виявити тестувальників, які більш успішно здійснюють «інтуїтивне тестування», тестувальників, які з більшим успіхом виконують тестування за сценарієм та, відповідно, осіб, які складали ті сценарії, що дозволили виявити найбільшу кількість критичних дефектів.

14. Відділ тестування. Дає можливість визначити якість керівництва відділом на основі аналізу тенденцій до покращення/погіршення якості тестування у розрізі цього відділу.

15. Вкладення. Контролюється якість описання дефекту, наявність відповідних вкладень (наприклад, знімків екрану) там, де це необхідно.

16. Кількість дефектів. Роботу тестувальника не можна у цілому оцінювати тільки за кількістю знайдених ним дефектів [10], але і не враховувати цей фактор також не можливо, особливо, коли оцінюється співвідношення кількості дефектів, знайдених тестувальником та кількості дефектів, знайдених користувачем (за умови досягнення бета-стадії розробки програмного продукту).

17. Час/дата занесення дефекту. Можна визначати кількість та пріоритетність знайдених тестувальником та кінцевим користувачем дефектів у розрізі часу.

Таким чином, система звіту про помилки може використовуватися не тільки як інструмент для взаємодії тестувальників та розробників, а і як інструмент для контролю за їх діяльністю.

## Обов'язки осіб, які забезпечують тестування, відповідно до кожного поля дефекту

Обов'язки осіб, які відповідають за тестування програмного забезпечення, щодо заповнення полів розподіляються таким чином [11] (табл. 2).

Таблиця 2

Ролі осіб, що відповідають за тестування відповідно до кожного поля дефекту

| Поле | Відповідальний за заповнення поля |
|---|---|
| Короткий опис | Автор звіту про дефект |
| Проект | Автор звіту про дефект |
| Компонент додатку | Автор звіту про дефект |
| Номер версії | Автор звіту про дефект |
| Критичність | Автор звіту про дефект, але цей атрибут може бути змінений менеджером. |
| Пріоритет | Менеджер проекту або менеджер, що відповідальний за розробку компоненту, про який складено баг-репорт |
| Статус | Автор звіту про дефект. Якщо звіт вноситься до автоматизованої системи багтрекінгу, то встановлюється заздалегідь |
| Призначений на | Менеджер проекту або менеджер, що відповідальний за розробку компоненту, про який складено баг-репорт |
| Оточення | Автор звіту про дефект |
| Опис дефекту | Автор звіту про дефект |
| Додатки | Автор звіту про дефект або будь-який член командної групи , який вважає,що додані файли сприятимуть виправленню дефекту |

## Розробка методики кількісної оцінки роботи тестувальників на підставі баг-репорту

Пропонується кількісна оцінка роботи тестувальників на підставі баг-репорту, що враховує кількість та критичність виявлених помилок (табл. 3).

Таблиця 3

Градація значимості дефектів

| Градація значимості дефектів | Кількість дефектів, знайдених тестувальником | Кількість дефектів, знайдених замовником |
|---|---|---|
| Блокуючий | Blt | Bez |
| Критичний | Crt | Crz |
| Значний | Mat | Maz |
| Незначний | Mit | Miz |
| Тривіальний | Trt | Trz |
| Разом | Nt | Nz |

Коефіцієнти, що враховують значимість виявлених помилок, пропонується 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 параметри, що якісно характеризують кількість дефектів, виявлених тестувальником та замовником, обчислюються за формулами, відповідно:

$$Qt = \frac{1}{12}(Trt + Mit) + \frac{1}{6}Mat + \frac{1}{4}Crt + \frac{5}{12}Blt;$$

$$Q_z = \frac{1}{12}(Trz + Miz) + \frac{1}{6}Maz + \frac{1}{4}Crz + \frac{5}{12}Bez.$$

Ефективність роботи тестувальника можна оцінювати за такою формулою, одержаною нами евристичним шляхом:

$$E = k \cdot \ln\frac{Q_t}{Q_z},$$

де $k = \frac{N_t + N_z}{N_t{}^2}$ – коефіцієнт, що враховує складність логіки системи.

Шляхом аналізу даних відкритих баг-репортів було встановлено, що мінімальним достатнім рівнем ефективності є значення 0,05 (рис. 1). Відповідно

значення, $E < 0,05$ вказують на недостатню ефективність тестування ПЗ, а значення, $E \geq 0,05$ – на достатню або високу ефективність тестування.

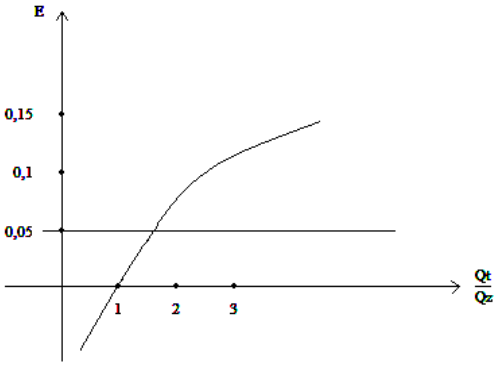


Рис. 1. Ефективність роботи тестувальника

## Висновки

Тестування програмного забезпечення є досить складним і важливим етапом розробки програмного забезпечення.

Тому оцінка ефективності роботи тестувальників ПЗ є важливим елементом комплексу заходів щодо забезпечення якості програмних продуктів. Запропонована деталізована структура баг-репорту та методика кількісної оцінки роботи тестувальників на підставі баг-репорту.

## Список літератури

*1. Про тестинг. Тестирование программного обеспечения - основные понятия и определения [Електронний ресурс]. – Режим доступу до ресурсу: http://www.protesting.ru/testing/.*

*2. Википедия. Качество программного обеспечения [Електронний ресурс]. – Режим доступу до ресурсу: http://ru.wikipedia.org/wiki/Качество_программного_обеспечения.*

*3. Словари и энциклопедии на Академике. Дефект [Електронний ресурс] – Режим доступу до ресурсу: http://dic.academic.ru/dic.nsf/business/3512.*

*4. Канер С. Тестирование программного обеспечения / С. Канер, Д. Фолк, Е. Нгуен Е. – К.: ДиаСофт, 2001. – 544 с.*

*5. Майерс Г. Искусство тестирования программ / Г. Майерс. – М.: Финансы и статистика, 1982.*

*6. Про тестинг. Основные поля баг / дефект репорта [Електронний ресурс]. – Режим доступу до ресурсу: http://www.protesting.ru/testing/bugstructure.html.*

*7. Про тестинг. Серьезность и Приоритет Дефекта [Електронний ресурс]. – Режим доступу до ресурсу: http://www.protesting.ru/testing/bugpriority.html.*

*8. Принцип "Где? Что? Когда?" [Електронний ресурс]. – Режим доступу до ресурсу: http://qanest.blogspot.com/2008/09/blog-post.html.*

*9. Оценка эффективности тестировщика. Количество найденных дефектов [Електронний ресурс]. – Режим доступу до ресурсу: http://inrecolan.ru/blog/viewpost/348.*

*10. Савин Р. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах / Роман Савин. – М.: Дело, 2007. – 312 с.*

*11. Про тестинг. Написание баг репорта [Електронний ресурс]. – Режим доступу до ресурсу: http://www.protesting.ru/testing/bugwriting.html.*

*Надійшла до редколегії 8.04.2011*

**Рецензент:** д-р фіз.-мат.. наук, проф. С.В. Смеляков, Харківський університет Повітряних Сил ім. І. Кожедуба, Харків.

### ОЦЕНКА ЭФФЕКТИВНОСТИ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВАНИИ АНАЛИЗА КОЛИЧЕСТВА И КРИТИЧНОСТИ ДЕФЕКТОВ

А.В. Щербаков, Е.С. Луценко

*На основании анализа структуры традиционного отчета о дефектах программных продуктов, которые были выявлены при тестировании, создана модифицированная структура отчета об ошибках. Предложенная структура отчета позволяет не только улучшить взаимодействие разработчиков и тестировщиков программного обеспечения, а и используется как средство контроля за их деятельностью. При этом каждое поле отчета рассматривается с точки зрения его учета для оценки эффективности тестирования. Разработана методика количественной оценки эффективности тестирования программных продуктов на основе баг-репорта. Эффективность роботы тестировщика оценивается за формулой, которая получена эвристическим путем и учитывает количество и критичность ошибок. Установлено минимальный достаточный уровень эффективности тестирования при использовании указанной методики.*

***Ключевые слова:*** *тестирование по, .методика оценки эффективности тестировщика, отчет об ошибке, модифицированная структура отчета об ошибки, средство контроля деятельности тестировщиков.*

### ESTIMATING OF SOFTWARE TESTING EFFECTIVENESS BY ANALYSIS OF QUANTITY AND CRITICALITY OF DEFECTS

A.V. Scherbakov, I.S. Lutsenko

*On the basis of the analysis of the structure of traditional reporting software defects that founded during testing, created a modified structure of the report. The proposed structure of the report can not only improve the interaction of developers and testers of the software, but also as a means of control over their activities. In this report, each field is considered in terms of its accounting for estimate the effectiveness of testing. Within the scope of the article the method of quantitative assessment of the effectiveness of software testing based on bug-report was developed. The effectiveness of the testers evaluated based on a formula that is derived heuristically and takes into account the number and criticality of errors. A minimum adequate level of testing using specified techniques was established.*

***Keywords:*** *software testing, estimate of testing effectiveness, bug report, modified structure of bug report, tool for control activities of testers.*