УДК 004.414

С.О. ВОЛКОВА, О.М. ТРУНОВ

Миколаївський державний гуманітарний університет ім. Петра Могили, Україна

ТЕСТУВАННЯ ЯК ЗАСІБ ПІДВИЩЕННЯ ЯКОСТІ ТА НАДІЙНОСТІ СИСТЕМИ ДІАГНОСТИКИ

У даній статті викладаються методології підвищення та забезпечення необхідного рівня якості та надійності системи діагностики. Визначено критерії оцінки якості та надійності та їх залежність від проходження процесу тестування програмного продукту. Представлено особливості застосування методології ортогональної класифікації дефектів та здійснено класифікацію основних типів дефектів програмного продукту. Використано метод цілеспрямованої перевірки дефектів, як спосіб забезпечення якості та надійності програмного забезпечення.

програмне забезпечення, якість, надійність, тест кейс (тестовий випадок), ортогональна класифікація дефектів, метод цілеспрямованої перевірки.

Вступ

Проблема якості та надійності ПЗ (програмного забезпечення) систем діагностики стає сьогодні все більш гострою, особливо зі збільшенням використання інформаційних технологій і підвищенням складності ПЗ.

Висока якість програмних продуктів дає розробляючам не тільки конкурентні переваги та збільшення довіри клієнтів, але й полегшує супровід і розвиток ПЗ.

1. Постановка проблеми

Орієнтиром з якості ПЗ стали загальновизнані стандарти якості ISO 9000 [1]. Відповідно до формулювання ISO 8402, **під якістю розуміється** сукупність характеристик програмного продукту, що відносяться до його здатності задовольнити встановлені й очікувані потреби клієнтів.

Основними параметрами якості вважаються: функціональна повнота, відповідність вимогам законодавства, безпека інформації, простота експлуатації, що не потребує спеціальних знань в області інформаційних технологій, ергономічність інтерфейсу користувача, мінімізація витрат на експлуатацію, розвиток і модернізацію.

Під надійністю розуміється здатність системи виконувати задані функції, зберігаючи основні характеристики за певних умов експлуатації. Стосовно програмного забезпечення це насамперед безвідмовна робота, відсутність помилок, що перешкоджають нормальному функціонуванню підприємства (рис. 1).

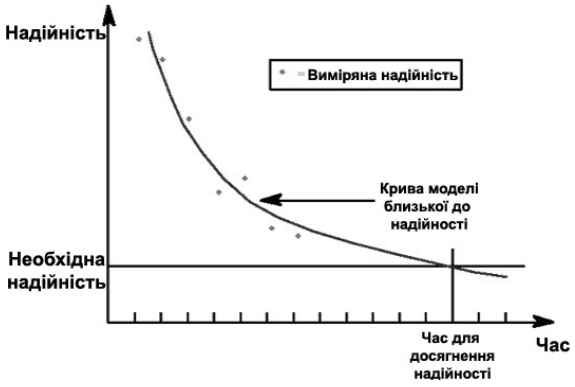

Рис.1. Залежність надійності програмного продукту від часу розробки ПЗ

Якість і надійність у комплексі забезпечують високі споживчі властивості ПЗ. У процесі створення програмного продукту вони одночасно й безупинно контролюються й удосконалюються. Однак наскільки реально забезпечити якість і надійність складної багатофункціональної системи при обмежених строках розробки? Для ілюстрації можна привести ре-

зультати опитування більше тисячі великих компаній, проведеного міністерством торгівлі й промисловості Великобританії. Виявилося, що середня частота відмов інформаційних систем склала: 1 відмову в рік – 40% компаній, 1 відмова на місяць – 29%, 1 відмова в тиждень – 15% компаній, 1 відмова в день – 7% й 5% компаній спостерігали в себе більше однієї відмови в день.

При цьому частка відмов і збоїв програмного забезпечення в загальному списку причин непрацездатності (простоїв) інформаційних систем становила 24% [2].

2. Розв'язок проблеми

Очевидно, що домогтися необхідної якості й надійності можна, тільки позначивши їх за пріоритетну мету й постійно просуватися до неї по наступних напрямках:

– організація промислового виробництва програмного забезпечення з чітко вираженою спеціалізацією, оптимальним розподілом функцій, повноважень і відповідальності персоналу;

– впровадження комплексу найбільш сучасних й ефективних технологій, включаючи як технології розробки й супроводу програмних продуктів, так і технологію керування розробками;

– розвиток системи якості [1] на основі рекомендацій ISO 9000-3.

В табл. 1 наведено критерії забезпечення якості ПЗ.

Таблиця 1

Критерії якості ПЗ

№	Критерії	К	З
1	Функціональна повнота	+	-
2	Ціна розробки	-	+
3	Відсутність дефектів	+	-
4	Зручність використання	+	-
5	Можливість внесення подальших змін	-	+
6	Легкість виправлення дефектів	-	-
7	Документація на реалізацію	-	-
8	Своєчасність виконання проекту	-	+

*де К – користувач, З – замовник.

Вид матриці критеріїв якості та систем забезпечення якості відображено в табл. 2.

Дані таблиці показують, що з восьми елементів загальної якості продукту тестування здатне оцінити й контролювати тільки три (1, 3, 4), причому найбільш ефективне тестування контролює відсутність дефектів (3).

Таблиця 2

Матриця критеріїв якості ПЗ

№	Т	ОК	АД	АПР	АР
1	+	–	–	–	+
2	–	–	–	+	–
3	+	+	–	–	–
4	+	–	–	–	+
5	–	+	+	–	–
6	–	+	+	–	–
7	–	+	–	+	–
8	–	–	–	+	–

*де № – порядковий номер критерію якості ПЗ (табл. 1), Т – тестування, ОК – огляди коду, АД – аналіз дизайну, АПР – аудит процесу розробки, АР – аналіз ринку.

На рис. 2 представлено залежність підвищення якості програмного продукту за рахунок тестування за концепцією B.Boehm [3].

Рис. 2. Підвищення якості ПЗ за рахунок тестування

На рис. 3 приведено спрощений алгоритм процесу тестування програмного продукту та на рис. 4 процес мінімізації помилок на різних стадіях розробки ПЗ, однак, тестування ПЗ не позиціюється як єдиний спосіб забезпечення якості [4]. Воно є частиною загальної системи забезпечення якості про-

дукту, елементи якого вибираються за критерієм найбільшої ефективності застосування в конкретному проекті.

Та як відомо [4, 5], повністю протестувати програму неможливо по наступних причинах:

1. Кількість всіх можливих комбінацій вхідних даних та послідовностей виконання коду програми занадто велика, щоб її можна було перевірити повністю.

2. Інтерфейс користувача програми (що включає всі можливі комбінації дій користувача і його переміщень по програмі) звичайно занадто складний для повного тестування.

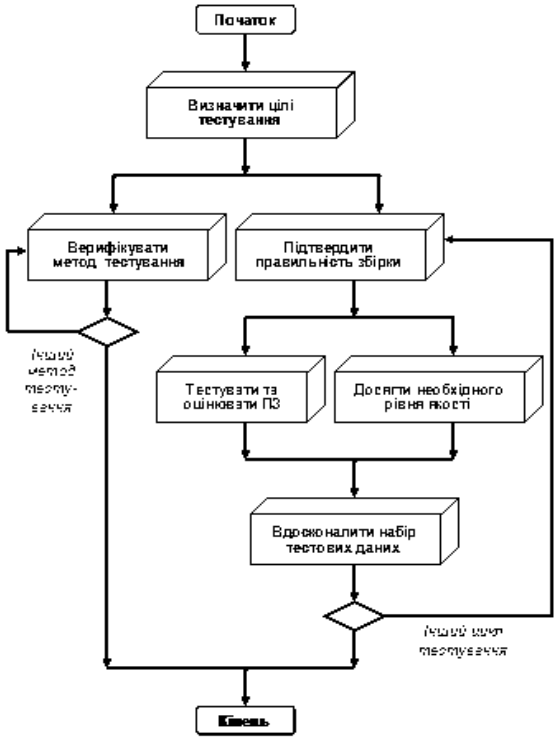

Рис. 3. Алгоритм процесу тестування ПЗ

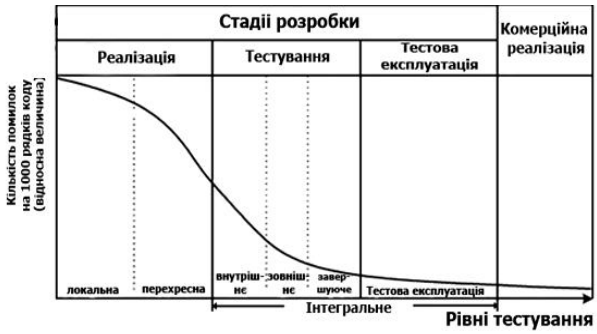

Рис. 4. Мінімізація помилок на різних стадіях розробки ПЗ

Варто відзначити характеристики вдалого тесту: існує обґрунтована ймовірність виявлення тестом помилки; набір тестів не повинен бути надлишковим; тест повинен бути найкращим у своїй категорії; тест не повинен бути занадто простим або занадто складним.

Однак згенерувати тест кейси (тестові випадки) для "повного" тестування продукту просто неможливо. Можна розробити мільйони тестів, але чи буде достатньо часу на їх виконання? Найімовірніше - ні. Тому доводиться ретельно вибирати тест кейси, які будуть виконуватися в процесі тестування.

На даний час існує кілька методологій розробки тест кейсів. Вони відрізняються як теоретичним підходом, так і практичною реалізацією. Найбільш часто вживана методологія розробки тестових випадків – методологія, в якій джерелами тестових випадків виступають випадки використання [6, 7].

Інший підхід до розробки тест кейсів дозволяє звернути увагу на те, які типи дефектів може містити в собі програмний продукт, і написати тестові випадки, що здатні їх виявити. Тобто він вимагає визначитися з тим, як система буде використана, і побудувати тестові випадки, які проведуть випробування системи на всіх етапах використання. Цей підхід називається метод цілеспрямованої перевірки побудований на основі класифікації ODC.

Класифікація ODC (Orthogonal Defect Classification – ортогональна класифікація дефектів) – це метод, розроблений корпорацією IBM з метою збору інформації про типи несправностей, які мають місце в розроблювальних програмних системах. Цей метод корисний при зборі й аналізі тестової інформації для того, щоб направити зусилля на вдосконалення процесу розробки в потрібному напрямку.

Ідея ODC полягає в поділі всіх дефектів на типи. Коли програміст виправляє дефект, те звичайно він виправляє якийсь певний тип дефекту. Тип визначається видом кінцевого виправлення. Призначення типів є очевидним для програмістів. У кожному випадку розходження проявляється в тому, чи є дже-

релом помилки пропущена частина коду або неправильна частина коду. По методу ODC виділяються наступні типи дефектів (помилок): функціональні, асигнування, інтерфейсу, перевірки, синхронізації, пакету, документації та алгоритмічні.

Функціональною помилкою є помилка, що значно впливає на можливості продукту, інтерфейс для кінцевого користувача, інтерфейс усередині продукту, інтерфейс із конфігурацією апаратної частини або помилка, що викликає глобальний крах системи й вимагає обов'язкової зміни дизайну. **Помилка асигнування** вказує на кілька невірних рядків коду, наприклад на ініціалізацію контрольних блоків або структуру даних. **Помилка інтерфейсу** відноситься до тих видів помилок, які пов'язані з обміном даних з іншими компонентами, модулями або драйверами пристроїв, викликами процедур, контрольними блоками або списками параметрів. **Помилки перевірки** пов'язані помилками програмної логіки в тих випадках, коли програма неправильно підтвердила дані й значення перед тим, як їх використати. **Помилки синхронізації (серіалізації)** – це ті помилки, які усуваються шляхом поліпшеного керування розподіленими ресурсами й ресурсами в реальному часі. **Помилки білду (пакету)** описують помилки, які трапляються через помилки в бібліотеках, керуванні змінами або контролем версій. **Помилки документації** можуть бути пов'язані як з помилками публікацій, так і записами керування. **Алгоритмічні помилки** включають помилки, пов'язані з недостатньою ефективністю або правильністю проблематики, що впливає на завдання, і може бути виправлена шляхом впровадження поліпшеного алгоритму локальної структури даних без зміни дизайну.

Обрані типи дефектів досить узагальнені для того, щоб бути застосованими до розробки будь-якого програмного продукту. Ступінь деталізації їх така, що їх можна співвіднести з будь-якою фазою розробки. Тест кейси розробляються відповідно до того, які типи дефектів можуть бути знайдені в даному програмному продукті й співвідносяться з видом діяльності розробки. Для того, щоб визначити необхідні групи тест кейсів будується матриця: по осях пишуться види дефектів і стадії процесу верифікації, як показано в табл. 3. Необхідно відмітити, що зараз розроблено математичні алгоритми для автоматичної генерації даних таблиць зв'язків [6].

Таблиця 3

Співвідношення дефектів та фаз розробки ПЗ

Стадія розробки ПЗ	**Типи дефектів**			
	Ф	П	С	А
Дизайн	●			
Дизайн низького рівня (ДНР)			●	
Кодування		●		●
Інспектування (ДВР)	●			
Інспектування (ДНР)			●	
Інспектування коду		●		●
Тестування модулів		●		●
Функціональне тестування	●			●
Тестування системи в цілому			●	

*Ф – функціональні, П – перевірки, С – синхронізації, А – алгоритмічні.

Іншим основним поняттям класифікації ODC є поняття "тригерів відмов", що являє собою умову, при якій помилка проявляється. Наприклад, коли продукт зібраний, передбачається, що всі його функції протестовано. Однак, виявляється, що нові помилки виникають при використанні продукту в іншому тестовому середовищі, при використанні інших апаратних засобів. Таким чином, хоч тип дефекту й той самий, необхідно задіяти різні типи тригерів відмов для того, щоб він виявився. При прояві помилки, тригер може бути визначений інженером, що спеціалізується на проблемному діагнозі.

Варто навести список шести категорій відмов, які розпізнаються методом ODC [7]:

1. Робочий обсяг/Підвищене навантаження.
2. Нормальний режим.
3. Відновлення/Виключення.
4. Пуск/Повторний пуск.
5. Конфігурація апаратних засобів.
6. Конфігурація програмних засобів.

Метод цілеспрямованої перевірки дефектів використовує декілька таких тригерів як орієнтири для вибору тестових випадків. Для забезпечення повноти тестування необхідно побудувати такі тестові випадки, які дозволяють задіяти кожний із цих тригерів дефектів [8].

3. Обговорення результатів

Кількісна оцінка знайдених дефектів по класифікації ODC дає можливість для ґрунтовного аналізу системи. Наприклад, якщо на рівні дослідження дизайну було знайдено багато функціональних помилок, то це значить, що можливо, дизайн не був добре продуманий і його необхідно переписати або переписати якісь певні частини коду, у яких було знайдено найбільше число помилок.

Складність впровадження ODC полягає в тому, що досить трудомістко застосувати даний підхід до розробки більших неоднорідних систем. Збільшується кількість груп тест кейсів і збільшується ймовірність виникнення помилки в непередбачуваних місцях. Також, для введення класифікації ODC критичними з наявність наступних факторів: добре спроектовані утиліти для відстеження дефектів й освіти людей, що займаються тестування (проводиться значна кількість різних тестів).

Висновки

Тестування програмного забезпечення є наразі найбільш очевидним способом забезпечення його надійності та якості.

Застосування методології ортогональної класифікації дефектів до кожного конкретного проекту дозволяє обирати елементи системи таким чином, щоб забезпечити прийнятну якість, виходячи із пріоритетів і наявних ресурсів. Обираючи елементи для системи забезпечення якості конкретного продукту, можна застосувати комбіноване тестування, огляди коду. При подібному виборі деякі якості, наприклад легкість модифікації й виправлення дефектів, не будуть оцінені й, можливо, виконані. Завданням тестування в розглянутому випадку є виявлення дефектів й оцінка зручності використання продукту, включаючи повноту функціональності.

Література

1. Загальне керівництво якістю й стандарти по забезпеченню якості (ISO 9000-1). Вказівки при розробці, поставці та обслуговуванні програмного забезпечення (ISO 9000-3).

2. Cockburn, A., Crystal/Clear: A Human-Powered Methodology for Small Teams, Addison-Wesley, 2000 [Електрон. ресурс]. – Режим доступу: http://members.aol.com/humansandt/crystal/clear.

3. Barry W. Boehm. Spiral Development: Experience, Principles, and Refinements. Spiral Experience Workshop, February 9, 2000 / Special Report CMU/SEI-2000-SR-008, July, 2000.

4. Макгрейгор Джон, Сайкс Девид. Тестування об’єктно-орієнтованого програмного забезпечення. Практ. пос. – К.: ТОВ "ТИД "ДС"", 2002. – 432 с.

5. Канер С., Фолк Д., Нгуен Е.К. Тестування програмного забезпечення. – К.: ДиаСофт, 2001. – 544 с.

6. Davis A.. Fifteen Principles of Software Engineering // IEEE Software. – 1994. – Vol. 11, № 6. – P. 94-101.

7. Шниман В. Отказоустойчивые компьютеры компании Stratus // Відкриті системи. – 1998. – № 1. – С. 13-22.

8. Принципи тестування ПЗ / Д. Коул, Т. Горэм, М. Макдональд, Р. Спарджеон // Відкриті системи, 1998. – № 2. – С. 60-63.

Корисні посилання

Загальні поняття тестування ПЗ:
http://en.wikipedia.org/wiki/Software_testing
Software quality assurance FAQ:
http://www.softwareqatest.com/qatfaq1.html

Надійшла до редакції 11.05.2007

Рецензент: д-р техн. наук М.Т. Фісун, Миколаївський державний гуманітарний університет ім. Петра Могили, Миколаїв.