УДК 004.4

М.В. Дідковська, канд. техн. наук, Ю.В. Бухтіяров, Д.О. Горобченко

Система автоматичного створення тестів на базі UML-діаграм варіантів використання

Предложена система автоматизации процесса создания тестов, которая дает возможность уменьшить временные и финансовые затраты, необходимые для процесса тестирования. Результатами работы системы являются шаблоны тестов, содержащие входные данные для теста, условия выполнения и ожидаемые результаты. Создание тестов в автоматическом режиме на базе анализа UML-диаграмм вариантов использования позволяет избежать пропуска важных тестов и неполноты покрытия спецификации.

System for automated tests' creation, that allows to reduce the time and costs required for the testing process, is proposed. The results of system functionality are tests' templates that contain input data, conditions of execution and expected results. Tests' creation in automatic mode based on the analysis of the UML use case diagrams allows to avoid omission of important tests and the incomplete cover sheet.

Ключові слова: *тестування, автоматизація тестування, тестування за сценаріями, діаграма варіантів використання, UML.*

Вступ

Згідно із сучасними методиками проектування, процес створення й супроводу інформаційних систем описується у вигляді життєвого циклу, який представляють як ітеративну послідовність певних стадій і виконуваних на них процесів [1]. Для кожної стадії визначають склад і послідовність виконуваних робіт, очікувані результати, необхідні методи й засоби, ролі та відповідальність учасників тощо. Такий формальний опис життєвого циклу інформаційних систем дає можливість спланувати й організувати процес колективної розробки проекту та забезпечити керування цим процесом.

Найбільш важливим і складним етапом життєвого циклу програмного забезпечення (ПЗ) інформаційних систем є фаза аналізу вимог та створення специфікацій. Згідно з даними корпорації IBM, на цьому етапі в проект вноситься до 56 % помилок, які потім можуть бути виявлені [2]. Забезпечення якості майбутнього програмного забезпечення слід розпочинати саме на цій фазі.

Методом контролю якості є тестування. В роботах [3, 4] було проаналізовано підхід до створення специфікацій із використанням діаграм UML (Unified Modeling Language — уніфікована мова моделювання) та запропоновано метод здійснення тестування якості вимог до ПЗ на основі вище зазначених діаграм. Проте на сьогодні всі тести на початкових етапах життєвого циклу ПЗ створюються вручну, що не виключає можливості неповного покриття специфікації та пропускання важливих тестів через людський фактор.

Отже, метою роботи є розробка системи автоматичного створення тестів для перевірки специфікацій, яка дозволить зкоротити витрати та підвищити якість проекту в цілому.

1. Діаграми варіантів використання як специфікація проекту

Специфікації програмного продукту містять як функціональні, так і системні вимоги до проекту. Апаратом для моделювання функціональних вимог є UML діаграми варіантів використання(ДВВ) [5].

Діаграма варіантів використання є концептуальним представленням системи в процесі її проектування й розробки.

Розробка діаграми варіантів використання має на меті:

- визначення загальних меж і контексту модельованої системи на початкових етапах проектування;
- формулювання загальних вимог до функціональної поведінки проектуємої системи;
- розробку концептуальної моделі системи для її наступної деталізації у формі логічних і фізичних моделей;
- підготовку вихідної документації для взаємодії розроблювачів системи з її замовниками та користувачами.

Суть ДВВ полягає ось у чому: проектовану систему представляють у вигляді множини акторів, що взаємодіють із системою за допомогою так званих варіантів використання. При цьому актором (actor), або дійовою особою, називається будь-яка сутність, що взаємодіє із системою ззовні [5]. Це може бути людина, технічний пристрій, програма або інша система, що слугуватиме джерелом впливу на моделюєму систему так, як визначить розробник.

У свою чергу, варіант використання (use case) слугує для опису сервісів, які система надає акторові. Інакше кажучи, кожний варіант використання визначає деякий набір дій, виконуваний системою під час діалогу з актором. При цьому аспекти внутрішньої реалізації, на даному етапі проектування, не розглядаються.

Зауважимо, що в найзагальнішому випадку діаграма варіантів використання є графом спеціального виду, вершинами якого є варіанти використання, актори та обмеження, а дугами – взаємозв'язки між цими елементами.

Вочевидь, від якості проектування функціональних вимог до системи залежатиме і якість майбутньої системи. Для тестування специфікацій та вимог до програмного забезпечення використовують тестові сценарії (test case). Тестовий сценарій – це набір вхідних даних, очікуваних результатів та умов виконання, призначений для визначення відповідності між специфікацією і реалізацією поставленого завдання. При проектуванні складних систем виникають труднощі при створенні тестових сценаріїв тестером, оскільки людський мозок не може осягнути весь спектр вимог та обмежень, які потрібно протестувати. Оскільки використання UML-діаграм є стандартом де-факто при розробці програмного забезпечення, то є можливість створення спеціального програмного забезпечення для автоматичного генерування тестових сценаріїв.

Отже, беручи до уваги зростаючу складність програмного забезпечення і ціну помилки, обумовлену неякісним тестуванням, запропонуємо підхід до автоматизації створення тестів на базі UML-діаграм.

2. Процедура автоматизації створення тестів на основі діаграм варіантів використання

З метою автоматизації створення тестів проаналізуємо структурні особливості ДВВ. Основними елементами ДВВ є актори, варіанти використання, зв'язки та обмеження.

За означенням тест (або контрольний приклад) [6] — це звіт, що містить:

1) *вхідні дані тесту* – інформацію, яку програма отримує із зовнішнього джерела, як-то пристрій, інша програма або людина;
2) *умови виконання* – вимоги для проведення тесту, наприклад, певний стан бази даних або конфігурація пристрою;
3) *очікувані вихідні дані* – передбачуваний результат роботи коду.

Тести мають бути розроблені для кожної ДВВ. При цьому варіант використання формує безпосередньо вхідні дані, актор та обмеження надають інформацію для створення умов виконання, а очікувані результати є знову ж варіантом використання або його запереченням, залежно від обмежень. Обмеження мають бути проаналізовані за допомогою класів еквівалентності та граничних елементів [6]. Якщо тестові дані належать до правильного класу еквівалентності, то очікуваним результатом є коректне виконання варіанта використання, якщо ж до неправильного — то невиконання варіанта використання й повідомлення про помилку.

Проілюструємо це на прикладі.

На рис.1 зображено найпростішу ДВВ надсилання повідомлення — зареєстрований користувач має можливість надіслати повідомлення, розмір якого має не перевищувати 10000 символів.

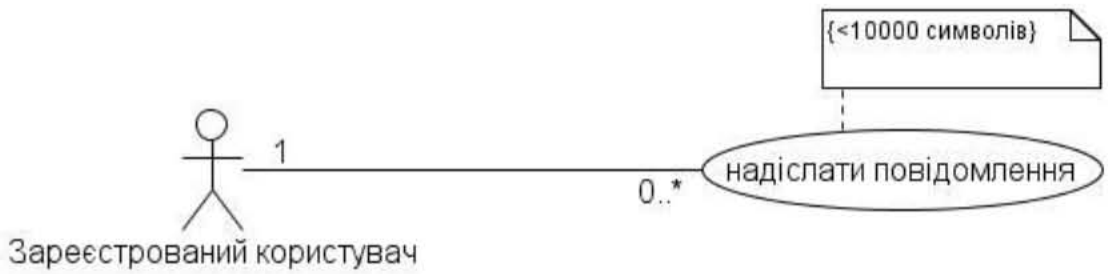

Рис 1. ДВВ надсилання повідомлення

З ДВВ мають бути згенеровані такі тести:

- Тест 1.

  Вхідні дані тесту – *надіслати повідомлення*.

  Умови виконання – *зареєстрований користувач; розмір повідомлення <10000 символів.*

  Очікувані вихідні дані – коректне виконання «*надіслати повідомлення*».

- Тест 2.

  Вхідні дані тесту – *надіслати повідомлення*.

  Умови виконання – *зареєстрований користувач; розмір повідомлення =10000 символів.*

  Очікувані вихідні дані – невиконання «*надіслати повідомлення*» та повідомлення про помилку.

- Тест 3.

  Вхідні дані тесту – *надіслати повідомлення*.

  Умови виконання – *зареєстрований користувач; розмір повідомлення >10000 символів.*

  Очікувані вихідні дані – невиконання «*надіслати повідомлення*» та повідомлення про помилку.

- Тест 4.

  Вхідні дані тесту – *надіслати повідомлення*.

  Умови виконання – *незареєстрований користувач; розмір повідомлення < 10000 символів.*

  Очікувані вихідні дані – невиконання *«надіслати повідомлення»* та повідомлення про помилку.

Для отримання тестів використано метод граничних умов та класів еквівалентності.

За методом граничних умов необхідно протестувати реакцію системи на значення, які знаходяться безпосередньо на межі, менші за неї та більші, тобто на кожне обмеження, що є в системі, має бути спроектовано три тести. На ДВВ надсилання повідомлення (рис.1) представлене обмеження на розмір повідомлення — 10000 символів, відповідно до методу граничних умов, створюються тести для повідомлень із розміром рівно 10000 символів, менше та більше 10000 символів (тести 1—3).

Метод класів еквівалентності полягає в тому, що мають бути проведені тести для представників як правильних класів еквівалентності так і неправильних. Наприклад, правильним класом еквівалентності є належність до ролі *зареєстрований користувач*, а неправильним — *не є зареєстрованим користувачем* (тести 1, 4).

На практиці ДВВ є не лише набором бінарних відносин актор – варіант використання, а й послідовністю взаємодій. Зважаючи на те, що перевірка функціонування окремих складових системи ще не гарантує перевірку системи загалом, необхідно згенерувати тести, спрямовані на перевірку всіх послідовностей взаємодії, тобто шляхів між акторами й варіантами використання. В цьому випадку вхідними даними буде послідовність варіантів використання, що міститься на ДВВ. Умови та очікувані результати створюються аналогічно з вище наведеним.

Отже, для аналізу кожної ДВВ пропонується згенерувати тести як для всіх варіантів використання, так і для їх послідовностей з урахуванням акторів та обмежень.

Розглянемо аспекти побудови системи автоматичної генерації тестів з ДВВ.

3. Концептуальні основи побудови програмного забезпечення для автоматичного створення тестів

Вхідними даними для системи автоматичної генерації тестів є ДВВ. Проте безпосередньо працювати з множиною графічних примітивів не є зручним. Нині всі сучасні програми для створення UML-діаграм підтримують експорт у файл формату XMI. XMI (Extensible Markup Language Metadata Interchange) — стандарт консорціуму OMG, для обміну мéта-інформацією, який може застосовуватися для будь-якої метамоделі, котра відповідає специфікації MOF (Meta-Object Facility). Основною метою OMG є розробка стандартів для розподілених об'єктно-орієнтованих систем і модельно-орієнтованих стандартів. Тому пропонується вибрати цей формат файлів як опис вхідних даних для створюваної програми.

Як було зазначено вище, UML-діаграму варіантів використання можна подати у вигляді змішаного графа.

Змішаний граф G — це граф, у якому деякі ребра можуть бути орієнтованими, а деякі — неорієнтованими. Він записується

впорядкованою трійкою $G = (V, E, A)$, для якої виконуються нижченаведені умови:

V — множина вершин;

E — множина невпорядкованих пар окремих вершин, які називаються ребрами;

A — це множина впорядкованих пар різних вершин, що називаються дугами.

Дуга — це впорядкована пара вершин (v,w), де вершину v називають початком, а w кінцем дуги.

Розділимо основні елементи ДВВ на вершини, ребра та дуги.

Вершинами є:

- варіанти використання;
- актори;
- обмеження;
- примітки.

До ребер належать:

- асоціації.

Дугами вважатимемо:

- розширення;
- включення;
- узагальнення.

Пропонується перетворити вхідні дані (ДВВ у форматі XMI) на граф вище означеного виду. Для організації зберігання графа в пам'яті має сенс використовувати матрицю суміжності. Матриця суміжності – це таблиця, де як рядки, так і стовпці відповідають вершинам графа. В кожному елементі цієї матриці записується число, яке визначає тип зв'язку від вершини-рядка до вершини-стовпця. Недоліком цього методу збереження графа є вимоги до оперативної пам'яті — квадратична залежність від кількості вершин графа. Однак за рекомендацією консорціуму OMG кількість варіантів використання на одній ДВВ не повинна перевищувати 25 [5], тому в даному випадку цей недолік не має істотного впливу, проте наявність швидких і простих алгоритмів для обробки матриці суміжності підтверджує правильність вибору цього методу.

Зважаючи на те, що елементи матриці відповідають вершинам, ребрам та дугам графа, створеного з відповідної ДВВ, з нього отримують всю необхідну інформацію для створення шаблону тесту, а саме – вхідні дані, умови виконання та очікувані результати, за процедурою показаною в попередньому параграфі. Обхід матриці суміжності дозволяє виявити всі існуючі маршрути між вершинами графа, тобто послідовності на ДВВ.

Робота системи автоматичного створення тестів полягає в тому, що на вхід (у блок розбирання XMI) подається ДВВ у форматі XMI, після розбирання XMI у блоці побудови графа UML моделі формується відповідний граф та зберігається у вигляді матриці суміжності. У блоці генерації тестових сценаріїв здійснюється обхід матриці та формування шаблону тестів, який містить вхідні дані, очікуваня результати та умови виконання.

Структурна схема програми представлена на рис. 2.

Рис 2. Структурна схема програми автоматичного створення тестів.

Висновки

Запропонована система автоматизує процес створення тестів, а отже, дає можливість зменшити кількість часу і коштів, необхідних для процесу тестування. Результатами роботи системи є шаблони тестів, що містять вхідні дані для тесту, умови виконання та очікувані результати. Генерація тестів в автоматичному режимі на базі аналізу ДВВ дозволяє уникнути пропускання важливих тестів та неповноти покриття специфікації.

Представлена структура системи автоматичного створення тестів може бути рекомендована до використання в подальших розробках, а саме для генерування тестів не тільки з ДВВ, а й інших UML-діаграм, таких як діаграма станів та діаграма послідовностей.

Література

1. *Kruchten Ph*. Rational Unified Process: An Introduction. – Addison-Wesley, 2003. – 277 p.
2. *Patton R*. Software Testing. – 2nd edn. Sams, 2005. – 408 p.
3. *Дідковська М.В.* Дослідження та аналіз графічних моделей функціональних вимог до Web-проектів // Наукові вісті. – 2007, № 6. – С. 49–54.
4. *Дидковская М.В.* Создание тестов и оценивание времени тестирования с помощью UML диаграмм вариантов использования // Электроника и связь. – 2007, № 2(37). – С. 79–81
5. *Object Management Group*. UML 2.0 Superstructure Specification. – Framingham, Massachusetts, 2004. – http://www.omg.org/cgi-bin/doc?formal/05-04-01.pdf
6. *Майерс. Г.* Искусство тестирования программ: Пер с англ.: под ред. Б.А. Позина. – М.: Финансы и статистика, 1982. – 172 с.