УДК 004.4

Кочергін І.В.
Хошаба О.М.

АНАЛІЗ МЕТОДІВ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ЮНІТ-ТЕСТІВ

Вінницький національний технічний університет

Анотація
У статті розглянуто аналіз методів автоматичної генерації юніт-тестів. Проаналізовано питання актуальності розробки системи автоматичної генерації юніт-тестів.
Ключові слова: система, автоматична генерація, юніт-тести, тестування.

Abstract
The article deals with the analysis of the methods of automatic generation of the unit tests. Analysed the question of topicality of development of the system for automating generation of unit-tests
Keywords: system, automatic generation, unit tests, testing.

Вступ

Юніт-тести це потужний інструмент для підтримки якості програмного забезпечення. Вони проводять первісну перевірку того, що програмний продукт задовольняє вимоги та поводиться, як задумано.

При правильній реалізації юніт-тести знижують кількість недоліків та знаходять їх на ранніх стадіях розробки, покращують читабельність коду, дозволяють перевикористовувати ділянки коду та підвищують ефективність роботи програмістів.

Метою дослідження є розробка програмної системи використання засобів автоматичної генерації юніт-тестів.

Об’єктом дослідження є процеси функціонування програмних систем автоматичної генерації юніт-тестів.

Предмет дослідження – методи та програмні додатки, до дозволяють використовувати системи автоматичної генерації юніт-тестів.

Головна задача – створення системи автоматичної генерації юніт-тестів для комфортного та швидкого тестування програм.

Аналіз методів

Для аналізу розглянемо наступні методи автоматичної генерації тестів [1]: випадкова генерація; генерація на основі алгоритмів пошуку; генетичні алгоритми; генерація тестових послідовностей.

Випадкова генерація – відносяться до якихось множин або діапазонів, кожен тест – комбінація значень, кожне з яких вибирається випадково з відповідної множини/діапазону.

Плюси:

– дуже просто реалізувати, є інструменти, які дозволяють автоматично генерувати синтаксично коректні юніт-тести;

– можна дуже швидко згенерувати багато тестів;

– знаходження помилок так само випадково, як і самі вхідні дані.

Генерація на основі алгоритмів пошуку полягає у тому, що треба розглянути генерацію даних як класичне завдання оптимізації, тобто задати цільову функцію та мінімально необхідний поріг досягнення результату, обмеження, класи еквівалентності для вхідних.

Властивості:

– можна «застрягти» в локальному оптимумі, якщо мінімально необхідний поріг занадто низький;

– дуже часто можливе кілька різних рішень і немає впевненості, що те, що було згенероване, найкраще;

– такі тести не беруть до уваги бізнес-логіку, техніки тест-дизайну та інше, тому якість тестування і, отже, знайдені баги також є випадковими.

Ідея методу генетичних алгоритмів полягає у тому, що треба згенерувати дуже багато наборів даних, змінювати компоненти між наборами, для кожного набору даних вимірювати значення цільової функції і порівнювати з мінімально необхідним порогом. В цілому схоже на погляд з боку теорії оптимізації, однак є можливість вийти з локального оптимуму в результаті кросовера, тобто, ймовірно, результати будуть кращими. Але і працювати такі алгоритми будуть повільнішими, ніж алгоритми пошуку, розглянуті вище.

Властивості:

– можна отримати кілька рішень, які відповідають цільовій функції.

– можна мати кілька цільових функцій, оскільки спочатку створюються набори і потім вимірюється цільовий показник. Цей підхід дозволяє взяти стільки цільових функцій, скільки потрібно, і розглядати їх одночасно.

Генерація тестових послідовностей полягає у класичній комбінаторній техніці тест-дизайну – взяти параметри, вибрати значення, задати обмеження та отримати всі можливі комбінації, що задовольняють обмеження. Pairwise є окремим випадком цього підходу, як і причина/наслідок.

Властивості:

– на вхід таким алгоритмам подається формальна модель, тобто параметри та значення, які вони набувають, умови, що накладаються на комбінації таких значень, параметри комбінації. Отже, результат проектування цілком передбачуваний і має строго той набір властивостей, які закладені в моделі;

– крім правил генерації та заданих залежностей між значеннями параметрів можна встановлювати ваги значень. Таким чином, можна регулювати частоту значень у тестах: там, де в комбінаціях все одно, яке значення вибрати, вага задає ймовірність такого вибору;

– окрім терезів значень, можна ставити і пріоритизацію, тобто порядок, в якому тести з'являться в наборі. Погана новина у тому, що така можливість є не у всіх інструментах.

Тобто, можна сказати, що кожен метод має право на існування і усе залежить від використання та знань у сфері тестування.

Висновки

Отже, було проведено аналіз методів автоматичної генерації юніт-тестів та зроблено висновок, що усі розглянуті методи є по-своєму уживаними та актуальними у різних ситуаціях. Метод випадкової генерації підійде для тих, хто бажає швидко згенерувати тести, генерація на основі алгоритмів пошуку – для тих, хто не хоче брати до уваги бізнес-логіку, генетичні алгоритми – для тих, хто хоче отримати кілька рішень, тестові послідовності – для створення пріоритизації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Автоматическая генерация тестов: подходы и инструменты. [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/automatic-test-generation/

Кочергін Іван Васильович – студент групи 2ПІ-18б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: darkvanyaster@gmail.com

Хошаба Олександр Мирославович – доцент кафедри програмного забезпечення, Вінницький національний технічний університет, Вінниця

Kocherhin Ivan V. – student of the group 2PI-18b, Department of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, email: darkvanyaster@gmail.com

Khoshaba Oleksandr M. – Associate Professor of Software, Vinnytsia National Technical University, Vinnytsia