УДК 004.4

А.В. Лучкова

Аналіз методів верифікації програмного забезпечення

Вінницький національний технічний університет, Україна

Анотація

У даній роботі представлений огляд методів верифікації програмного забезпечення (ПЗ). Пропонується класифікація відомих методів верифікації. В рамках запропонованої системи розглядаються як зрілі і широко використовувані при розробці методи верифікації ПЗ, так і недавно створені і використовувані поки тільки в рамках дослідницьких проектів.

Ключові слова: верифікація програмного забезпечення, тестування програмного забезпечення.

Abstract

This paper provides an overview of methods for verification of software. Proposed classification known methods of verification. As part of the proposed system are regarded as mature and widely used in the development of methods for software verification and recently created and used so far only in the research projects.

Keywords: verification software, software testing.

Будь-яка інформаційна система складається з апаратного та програмного забезпечення (ПЗ). На початку розвитку комп'ютерної техніки апаратна частина була більш складною і значно дорожчою. Однак гнучкість програмного забезпечення та (як виявилося згодом, оманлива) простота внесення в нього змін спонукали використовувати його для вирішення найрізноманітніших завдань на одному і тому ж або стандартизованому апаратному забезпеченні. Тому поступово ПЗ ускладнювалося, набувало все більшої цінності, і в останні десятиліття його вартість сягає від 30% до 90% вартості систем, в залежності від їх типу [1].

При побудові систем певного рівня складності люди в принципі не можуть уникнути помилок, а зростаюча складність надає все більше можливостей для помилок, при цьому ускладнюючи їх швидке виявлення. Для забезпечення коректності та надійності роботи таких систем велике значення мають різні методи верифікації та валідації, що дозволяють виявляти помилки на різних етапах розробки і супроводу програмного забезпечення (ПЗ), щоб послідовно усувати їх.

Під життєвим циклом програмного забезпечення зазвичай розуміють весь інтервал часу від моменту зародження ідеї про те, щоб створити або придбати програмну систему для вирішення певних завдань, до моменту повного припинення використання останньої її версії.

Артефактами життєвого циклу ПЗ називаються різні інформаційні сутності, документи і моделі, створювані або використовуються в ході розробки і супроводу ПЗ. Так, артефактами є технічне завдання, опис архітектури, модель предметної області на якомусь графічній мові, вихідний код, призначена для користувача документація і т.д.

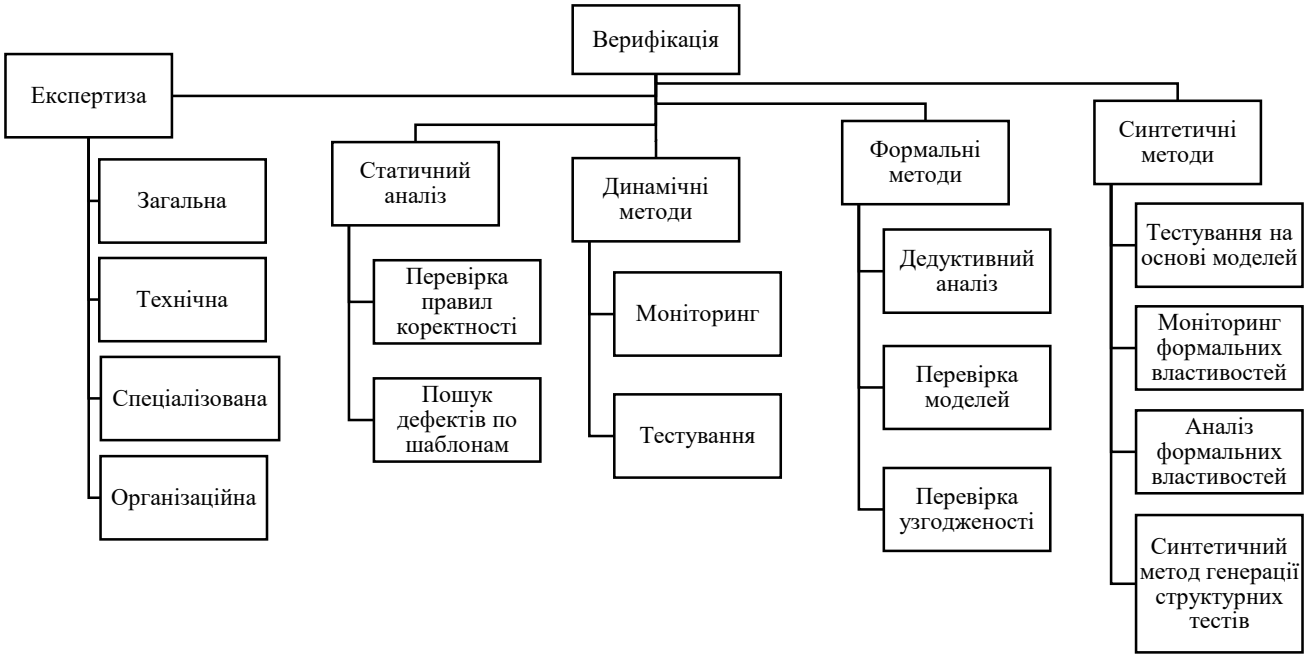
Верифікація програмного забезпечення – більш загальне поняття, ніж тестування. Метою верифікації є досягнення гарантії того, що верифікований об'єкт (вимоги або програмний код) відповідає вимогам, реалізований без непередбачених функцій і задовольняє проектним специфікаціям і стандартам. Процес верифікації включає в себе інспекції, тестування коду, аналіз результатів тестування, формування та аналіз звітів про проблеми. Таким чином, прийнято вважати, що процес тестування є складовою частиною процесу верифікації.

Валідація програмної системи – процес, метою якого є доказ того, що в результаті розробки системи ми досягли тих цілей, які планували досягти завдяки її використанню. Іншими словами, валідація – це перевірка відповідності системи очікуванням замовника.

Основним стандартом, що регулює планування і проведення верифікації ПЗ, є стандарт IEEE 1012 на процеси верифікації та валідації [2]. Цей стандарт містить опис наборів окремих завдань верифікації, що відповідають різним видам діяльності, рекомендований шаблон плану проведення перевірки та затвердження, визначення 4-х рівнів критичності ПЗ (від високої до мінімальної).

В цілому сучасна система міжнародних стандартів не містить цілісного підходу до верифікації ПЗ.

Загальноприйнятий поділ методів верифікації можна представити в вигляді діаграми



Експертизою ПЗ називають всі методи верифікації, в яких оцінка артефактів життєвого циклу ПЗ виконується людьми, що безпосередньо аналізують ці артефакти. Перевагою даного методу є те, що при його використанні виявляються в середньому 50-90% помилок [3]. Цей метод має і недоліки. Пошук помилок, оцінка і аналіз властивостей ПЗ людиною (зазвичай групою 2-5 осіб). Потрібні справжні експерти, програмісти з досвідом роботи не менше 10 років.

Статичний аналіз – аналіз без виконання програми. Методи статистичного аналізу можна розділити на два види: контроль того, що всі формалізовані правила коректності побудови цих артефактів виконані, та пошук типових помилок і дефектів в них на основі деяких шаблонів [4]. Часто інструменти статичного аналізу використовують обидва типи перевірок. Статичний аналіз можна вважати найбільш широко застосовуваним методом верифікації. Перевірені на практиці правила коректності коду або шаблони типових помилок переносяться в середовища розробки.

Переваги статичного аналізу:

– Автоматичний аналіз багатьох шляхів виконання одночасно.

– Виявлення помилок, що проявляються лише на одиничних шляхах виконання або на незвичайних вхідних даних.

– Можливість аналізу на неповному наборі вхідних файлів.

– Відсутність накладних витрат під час виконання програми.

Недоліки даного методу:

– Велика кількість помилкових спрацювань.

– Необхідна ручна перевірка результатів роботи, що вимагає значних часових, людських та матеріальних ресурсів

Динамічні методи верифікації використовують результати реальної роботи програмної системи або її прототипів, щоб перевіряти відповідність цих результатів вимогам і проектним рішенням.

Існує два основних види динамічних методів верифікації: моніторинг, в рамках якого йде тільки спостереження, запис і оцінка результатів роботи ПЗ при його звичайному використанні, і тестування, при якому ПЗ виконується в рамках заздалегідь підготовлених сценаріїв. Перевагою даного методу є висока точність виявлення помилок. А недоліками – необхідно мати набір вхідних даних та середовище виконання, а також високі вимоги до ресурсів.

Формальні методи верифікації. Їх відмітною особливістю є можливість проведення пошуку помилок на математичній моделі, без звернення до фізичної реалізації, що в деяких випадках досить зручно і економічно. Для проведення аналізу формальних моделей застосовуються специфічні техніки,

такі як дедуктивний аналіз, перевірка моделей, перевірка узгодженості. На жаль, для побудови таких моделей завжди необхідно виходити так само з коректності та адекватності моделі ПЗ [5]. Лише після правильної побудови цієї моделі можна автоматично проаналізувати деякі з її властивостей. Проте, в більшості випадків для ефективного аналізу від фахівців будуть потрібні глибокі знання математичної логіки і алгебри і деякого набору навичок роботи з цим апаратом.

В останні роки активно розробляються інструменти автоматичної генерації тестів на основі коду, які використовують додаткові джерела інформації. В якості таких джерел виступають статичний аналіз коду, формальний аналіз, моніторинг виконання раніше побудованих тестів і т.п. Оскільки в інструментах цього типу використовується зазвичай 3-4 техніки різних типів, методи, що лежать в їх основі, винесені в окремий різновид синтетичних методів верифікації [6]. Синтетичні методи верифікації поєднують підходи декількох типів – статичний аналіз, формальний аналіз властивостей ПЗ, тестування. Деякі з таких методів породили в останні 10-15 років самостійні галузі досліджень, в першу чергу, тестування на основі моделей і моніторинг формальних властивостей. Переваги та недоліки синтетичних методів визначаються комбінацією методів верифікації, які входять до її складу.

Після аналізу представлених методів верифікації програмного забезпечення можна зробити висновок, що кожен метод верифікації використовується в конкретному класі випадків в залежності від поставленої мети. Найактуальнішими, найбільш корисними та продуктивними можна вважати синтетичні методи верифікації ПЗ, оскільки вони так чи інакше намагаються поєднати переваги різних підходів до верифікації, зменшуючи їх недоліки. В даний час досягнуті значні успіхи в розробці таких методів і впровадженні їх у практику промислової розробки ПЗ.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Б.У. Боэм. Инженерное проектирование программного обеспечения. М.: Радо и связь, 1985. – 368 с.

2. IEEE 1012-2004 Standard for Software Verification and Validation. IEEE, 2005. – p. 153.

3. Y. K. Wong. Modern Software Review: Techniques and Technologies. IRM Press, 2006. – p. 368

4. L. Yu A light-weight static approach to analyzing UML behavioral properties / L. Yu, R. B. France, I. Ray, K. Lano.. Proc. of 12-th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007), pp. 56-63, 2007. – p. 79.

5. T. Ball Thorough Static Analysis of Device Drivers. In Proc. of EuroSys 2006/ T. Ball, E. Bounimova, B. Cook, V. Levin, J. Lichtenberg, C. McGarvey, B. Ondrusek, S. K. Rajamani, A. Ustuner., ACM SIGOPS Operating Systems Review, 2006. – p. 74.

6. M. Broy Model Based Testing of Reactive Systems / M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, A. Pretschner (eds.). LNCS 3472, Springer, 2005. – p. 273.

Лучкова Ангеліна Володимирівна, студентка групи 1КІ-15мс, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця.

Науковий керівник – ***Семеренко Василь Петрович,*** к.т.н., доцент, Вінницький національний технічний університет, м. Вінниця.

Luchkova Angelina Volodymyrivna, student of the group 1KI-15ms, Faculty of Information Technology and Computer Science, Vinnytsia National Technical University. Vinnitsa.

Supervisor – ***Semerenko Vasily Petrovich***, Ph.D., associate professor, Vinnytsia National Technical University. Vinnitsa.