https://doi.org/10.31891/2219-9365-2025-82-24
УДК 004.415.53

КИРИЧУК Юрій
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
https://orcid.org/0000-0001-8638-6060
kirichuky@gmail.com
СТАХОВА Анжеліка
Маріупольський державний університет
https://orcid.org/0000-0001-5171-6330
sap@nau.edu.ua
НАЗАРЕНКО Наталія
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
https://orcid.org/0000-0001-6100-4193
N_Nazarenko@kpi.ua
ЗАЄЦЬ Сергій
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
zssvp0204@gmail.com

ОГЛЯД МЕТОДІВ, МОДУЛІВ ТЕСТУВАННЯ ПІДСИСТЕМ ТА ПРОГРАМИ ЗАГАЛОМ

У цьому дослідженні вивчалися методи забезпечення якості програмного забезпечення, а також проблеми, з якими стикаються під час реалізації якості програмного забезпечення у прагненні покращити якість програмного забезпечення. Відомо, що розробка якісного програмного продукту - важлива потреба для індустрії програмного забезпечення. Визначено, що зосередження уваги на якості продукту дозволяє кінцевим користувачам програмного забезпечення більш легко та ефективно адаптувати продукт. Якість відіграє життєво важливу роль користувачів програмного забезпечення. Це підтвердження всіх вимог щодо задоволеності клієнтів. Отже, важливо визначити правильний процес розробки програмного забезпечення, що веде до якісного програмного продукту. В роботі, розглядаються основні поняття в області тестування програмного забезпечення, критерії вибору тестів, оцінка відтестованості проекту. Значна увага приділяється методам тестування програмного забезпечення, всі існуючі методи тестування діють у рамках формального процесу перевірки програмного забезпечення, що досліджується або розробляється. Такий процес формальної перевірки може довести, що дефекти відсутні з точки зору використовуваного методу. Тобто, немає ніякої можливості точно встановити або гарантувати відсутність дефектів у програмному продукті з урахуванням людського фактора, присутнього на всіх етапах життєвого циклу програмного забезпечення. Розглядаються питання автоматизації процесу тестування та питання зв'язку між процесом тестування і якістю програмного забезпечення. Таким чином, тестування є одним із способів розробки якісного програмного продукту і входить до набору ефективних засобів сучасної системи забезпечення якості програмного продукту.

Ключові слова: тестування, якість програмного забезпечення, методи оцінки якості, верифікація програм, засіб специфікації модулів.

KYRYCHUK Yurii
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”
STAKHOVA Anzhelika
Mariupol State University
NAZARENKO Natalia, ZAYETS Serhii
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

OVERVIEW OF METHODS, MODULES TESTING OF SUBSYSTEMS AND PROGRAMS IN GENERAL

This study examined the methods of software quality assurance, as well as the problems encountered in the implementation of software quality in an effort to improve software quality. It is known that developing a quality software product is an important need for the software industry. It was determined that focusing on product quality allows software end users to adapt the product more easily and efficiently. Quality is a vital role for software users. This is a confirmation of all customer satisfaction requirements. Therefore, it is important to determine the right software development process that leads to a quality software product. In the work, the basic concepts in the field of software testing, criteria for selection of tests, assessment of project testing are considered. Considerable attention was paid to software testing methods, all existing testing methods operate within the formal process of testing software that is being researched or developed. Such a formal verification process can prove that defects are absent from the point of view of the method used. That is, there is no way to accurately identify or guarantee the absence of defects in the software product, taking into account the human factor present at all stages of the software life cycle. The issues of automation of the testing process and the relationship between the testing process and software quality were considered. Thus, testing is one of the ways to develop a quality software product and is part of a set of effective tools for a modern software product quality assurance system.

Keywords: testing, software quality, methods of quality assessment, program verification, module specification tool.

Стаття надійшла до редакції / Received 01.04.2025
Прийнята до друку / Accepted 01.05.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Наприкінці XX ст. у сфері IT- технологій об'єми і розміри проектів із розроблення програмних систем значно зросли. Такі масштабні проекти унеможливили ручне тестування програмних засобів (ПЗ), що, в свою чергу, спричинило активний розвиток засобів автоматизації процесів тестування. До 2000-х років поняття якості ПЗ розширилося настільки, що з'явилася необхідність виділення окремого виду діяльності при створенні програмних засобів, тобто забезпечення якості (Quality assurance, QA) [1]. На сьогодні автоматизоване тестування суттєво поширене, а засоби автоматизованого тестування часто вбудовані у середовище програмування.

Сьогодні в науковій українській літературі бракує інформації про найактуальніше питання, що постало, тобто питання забезпечення якості програмного продукту, що розробляється, а саме процесу тестування, який є одним із найефективніших шляхів підвищення якості при розробці програмного продукту та є частиною системи забезпечення якості програмного продукту.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є розкриття зв'язку між процесом тестування програмного забезпечення і якістю програмного забезпечення.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Для виконання заходів з забезпечення якості ПЗ розробники створюють спеціальні групи контролю якості, вони мають назву QA. Група QA всередині компанії виконує роль вимогливого користувача. Деякі компанії забороняють неформальне спілкування між командою розробників і командою тестувальників. Успіх продукту також залежить від відповідальності групи QA. Якщо кінцевим користувачам з якихось причин не сподобається програмний продукт, це стане причиною для втрати репутації та споживачів назавжди.

Помилки, які не були знайдені на стадії розроблення згодом будуть дорого коштувати організації-розробнику. Традиційна стратегія розроблення програмного забезпечення підпорядковується фундаментальному правилу, яке визначає, що впродовж роботи над проектом вартість внесення змін у створюване програмне забезпечення збільшується за експонентою (рис. 1).

Фактично, його майбутні витрати залежать від того, наскільки якісно виконані початкові етапи розробки програмного забезпечення [2]. З метою підвищення якості розробки та уникнення ризиків, пов'язаних із «нечіткими», незрозумілими або постійно мінливими вимогами, запропоновано методологію XP (eXtreme Programming).

Слід зазначитити, що програмування та тестування — це різні види діяльності. Програмування — процес синтезу, а тестування – це процес аналізу.

Тестування (Software Testing) — діяльність конкретного постачальника, що виконується з метою оцінки та покращення якості ПЗ. Ця діяльність полягає у виявленні дефектів і проблем програмного забезпечення. Тестування ПЗ включає в себе діяльність із планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) та аналіз отриманих результатів (Test Analysis).

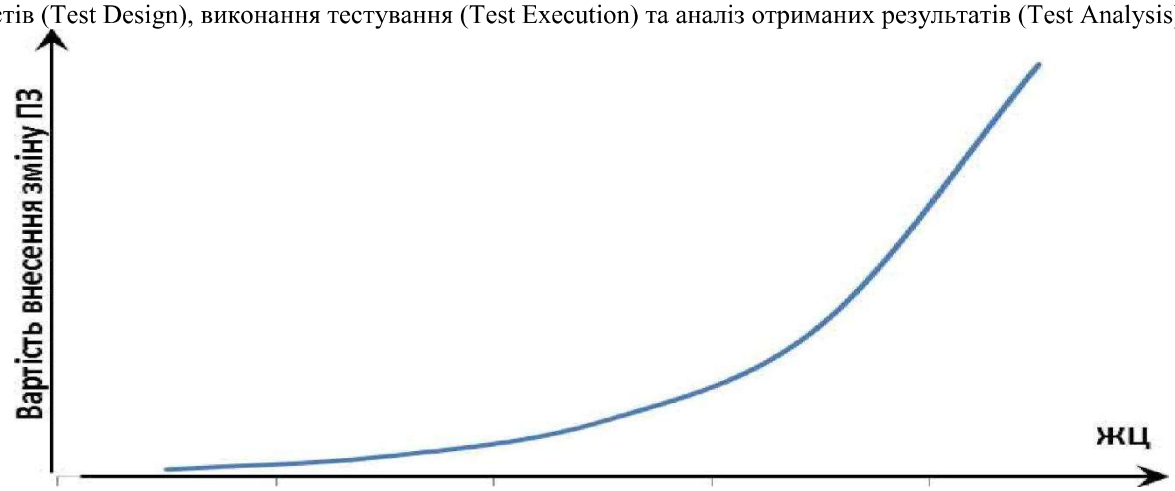

Рис. 1. Залежність вартості змін проектів від часу

Тестування вимагає уваги та старанності від виконавця. Тестувальник повинен шукати помилки в будь-якому місці системи.

Якщо сам програміст виконує початкове тестування для налагодження коду, то наступні етапи перевірки повинні бути підготовлені і виконані іншими програмістами, щоб перевірка була повноцінною і

завершеною, а не лише для виявлення очікуваних проблемних місць.

Щоб спланувати процес тестування, необхідно мати уявлення про кількість необхідних перевірок. Зважена статистика тестування наступна:

- при системному тестуванні у відділі контролю якості на ліквідацію однієї помилки потрібно від 4 до 16 годин;
- на 1000 рядків коду програміст у середньому робить 100 помилок, 70% цих помилок усуваються на стадії налагодження коду;
- помилка, що була виявлена у ході експлуатації, потребує від 35 до 90 годин для її усунення.

Усі ці дані вказують, наскільки важливо виявляти та виправляти проблеми на початку розробки програмного продукту. Слід відзначити, що парне програмування, що є особливим підходом до підвищення якості програмування. Це зменшує помилки на 15% порівняно з традиційним простим кодуванням.

Найстарішим методом тестування є контроль усіх входів і виходів, як запланованих, так і несанкціонованих (повне тестування). Програма повинна адекватно реагувати на помилкові ситуації, не втрачаючи стабільності в роботі[3]. Тестування також вимагає перевірки виконання всіх наданих системних функцій. Перелік таких тестів складається на ранніх етапах проектування паралельно з описом особливостей програмного продукту.

Види тестування можна класифікувати за різними показниками:

1. ***За рівнем знання системи:***
   - *сірий ящик (Gray-boxtesting)* - тестування із частковим вивченням коду програми;
   - *білий ящик {White-box testing)* - тестування із вивченням коду програми;
   - *чорний ящик (Black-box testing)* - без перегляду програмного коду;
2. *За об'єктом тестування:*
   - *юзабіліті-тестування(Usability testing)* - перевірка ергономічності ПЗ;
   - *тестування продуктивності (Performance testing)* - перевірка стабільності роботи програми або її частини при заданому навантаженні (Load testing), при перевантаженні (Stress testing) та тривалому середньому навантаженні (Stability testing);
   - *функціональне (Functional testlng)* - перевірка виконання заданих функцій;
   - *перевірка інтерфейсу користувача (UI testing);*
   - *перевірка безпеки (Securlty testing)* - тестування роботи системи з точки зору безпеки інформації;
   - *тестування сумісноті (Compatibillty testing)* - не функціональне тестування, метою є перевірка коректної роботи ПЗ у певному оточенні.
3. *За часом виконання тестування:*
   - *півгодинне тестування (Smoke testing)* - коротка перевірка функцій системи, зазвичай, один тест для кожної функції;
   - *альфа-тестування (Alpha testing)* - перевірка роботи ПЗ перед випуском силами організації - розробника;
   - *регресійне тестування (Regression testing)* - перевірка роботи протестованих модулів;
   - *тестування в період передачі ПЗ (Acceptance testing);*
   - *тестування нової функціональності {New feature testing);*
   - *бета-тестування (Beta testing)* - тестування ПЗ перед передачею в експлуатацію сторонніми силами.
4. *За ступенем ізольованості компонентів:*
   - *системне тестування (System / end-to-end testing)* - перевірка роботи створеного ПЗ з метою виявлення відповідності заданим функціям;
   - *інтеграційне тестування {Integration testing)* - перевірка роботи модулів, об'єднаних у групу;
   - *компонентне тестування (Component / Unit testing)* - перевірка коректності роботи окремого модуля або компонента системи.
5. *За ступенем автоматизації:*
   - *напівавтоматизоване тестування (Semiautomated testing);*
   - *автоматизоване тестування (Automated testing);*
   - *ручне тестування (Manual testing).*
6. *За ступенем підготовленості до тестування:*
   - *інтуїтивне тестування (Ad hoc testing)* - тестування ПЗ без попереднього плану дозволяє виявляти помилки в роботі ПЗ ще на ранніх стадіях;

- *формальне тестування (Formal testing)* - тестування згідно встановлених процедур та плану.

7. *За ознакою позитивності сценаріїв:*
   - *перевірка негативних сценаріїв (Negative testing);*
   - *перевірка позитивних сценаріїв (Positive testing).*

Як і будь-який процес, тестування має методи оцінки якості .

Один із способів навмисно вставити ряд помилок у програму під час розробки. Після тестування підраховується, скільки таких помилок було виявлено. Деякі знайдені помилки показують якість тестування та допомагають оцінити обсяг роботи, яку необхідно виконати, щоб повністю усунути проблеми. Плануючи тестування, необхідно визначити очікувані результати випробувань, які дозволять оцінити готовність створеного продукту[4]. Бажано узгодити їх із замовником, щоб були чітко визначені показники завершеності проекту та затверджені результати.

Якщо розробники тестів мають досконало розуміти систему, то виконавці тестів можуть не мати високу кваліфікацію. Цю роль часто грають новачки в компаніях без досвіду роботи.

Для підвищення рівня контролю якості кожного проекту слід створити базу даних помилок, до якої вноситься така інформація:

- за принципом: Що? Де? Коли?;
- опис помилки;
- версія продукту;
- у якому модулі була знайдена помилка;
- статус помилки:
  - open: знайдена;
  - can't reproduce: неможливо повторити;
  - fixed: виправлена;
  - won’t fix: не помилка;
  - by design: помилка проектування;
  - regression: виправлена помилка з'явилася знову;
  - postponed: зараз виправити важко, буде виправлена у наступній версії;
- важливість (severity) помилки:
  - major problem: програма частково не працює, часткова втрата даних;
  - crash: повна втрата даних;
  - trivial: зараз не потрібно виправляти;
  - minor problem: програма працює не так, як очікувалось, але дані не втрачаються;
- пріоритет помилки:
  - high: поставити продукт із такою помилкою не можна, але можна перейти до наступної версії;
  - highest: не можна поставити продукт, не можна перейти до наступної версії;
  - low: косметичні поліпшення - помилка залишається до наступної версії;
  - medium: помилка буде виправлена.

Така база даних дозволяє проаналізувати якість роботи окремих груп програмістів, можливості інструментальних засобів, що використовуються та оцінити якість проекту. Також керівники можуть відслідковувати ризики розроблення ПЗ завдяки помилками, що з’явились, найвищих рівнів важливості та пріоритету[5].

Дані з бази помилок дозволяють приймати рішення про можливість випуску програмного продукту (помилки з серйозністю "crash" або з пріоритетом "highest/high" не дозволяють поставляти програмне забезпечення). Якщо ПЗ обов'язково потрібно надати (встановити) замовнику і немає часу на виправлення помилок, функції, що виконуються з помилками, можуть бути виключені за домовленістю.

Види тестів

Одним з найбільш об'єктивних методів оцінки якості ПЗ є випробування програми.

Випробування програми можна провести, наприклад, для визначення ступеня відповідності готової програми вимогам, сформульованим у завданні.

Тестування також може забезпечити більш точні оцінки таких параметрів, як:

- максимальний обсяг необхідної оперативної пам'яті;
- середній час вирішення завдання;
- показники завантаження зовнішніх пристроїв, необхідні для оцінки вартості вирішення задачі.

Також важливою, метою проведення випробувань ПЗ є спрямований пошук помилок роботи ПЗ[6]. Для будь-яких цілей програма випробовується, одним з найважливіших питань є вибір вхідних даних

програми. Такі спеціально відібрані вихідні дані для конкретних цілей називаються тестами. Тестування програми за допомогою тестів називається тестуванням, яке починається під час процесу налагодження.

Програма пропонує перший тестовий приклад одразу після трансляції. Подальше тестування та налагодження тісно пов'язані між собою. Якщо наступна перевірка не вдається, програма генерує результат, який не відповідає тестовим даним, виникає проблема з пошуком і виправленням помилки, і це інший процес - налагодження.

З іншої сторони, для локалізації помилки необхідно вибрати такі тестові приклади або вихідні дані, які б сприяли найбільш явному прояву виявленої, але ще не локалізованої помилки.

Тестування є етапом процесу розробки програмного забезпечення, який включає підбір тестових випадків і тестування програми на них для виявлення помилок[7].

Налагодження є етапом у процесі створення програми, який включає пошук та виправлення помилок, виявлених під час трансляції та тестування.

Етап тестування включає три головних елементи:

1) врахування внеску кожного тестового прикладу в процес тестування;
2) аналіз вихідних і проміжних результатів ПЗ;
3) генерація тестових прикладів і контроль процесу тестування.

Перший елемент визначається критерієм повноти тестування, який тісно пов'язаний з конкретним методом тестування.

Для другого і третього елементів вихідними даними є документація ПЗ, на основі якої генеруються тестові приклади. Ці приклади надають спосіб виконання програми, визначають ступінь узгодження з результатом, отриманим у прикладі.

Тестування проводиться на чотирьох рівнях:

1) рівень з'єднань, на якому здійснюються пошук помилок міжкомпонентного інтерфейсу;
2) рівень окремого програмного компонента (модуля);
3) рівень зовнішніх функцій, на якому шукають розбіжності між функціями програмного забезпечення та зовнішніми програмними специфікаціями;
4) комплексне тестування, випробування програмного комплексу з точки зору відповідності вихідним цілям.

Цілком логічно, що тестування зв'язується з організацією розробки програм. Тому, два способи створюють «фундамент», на якому будуються різні методи тестування.

Висхідне тестування передбачає, що програма будується і тестується знизу вгору. Модулі (компоненти найнижчого рівня) тестуються автономно. Від надійності тестування цих модулів залежить успіх подальшого процесу. Після цього відбувається перехід до модулів, які посилаються на модулі, які вже були протестовані. На цьому етапі необхідно перевірити інтерфейс.

Щоб реалізувати тестування знизу вгору, потрібно написати невелику управляючу процедуру для кожного модуля, іншими словами - драйвер. Значення вхідних змінних і структура даних надаються драйвером. Драйвер викликає тестований модуль один за одним і пропонує йому новий тестовий приклад кожного разу, коли він викликається.

Основними недоліками тестування знизу вгору є:

- серйозні помилки в специфікаціях, алгоритмах і інтерфейсах можуть виникати лише при тестуванні комплексів на високому рівні, на завершальному етапі;
- необхідність розробки драйверів і тестів для кожного рівня тестування, що призводить до великої кількості додаткових програм, які стають непотрібними при завершенні роботи над комплексом.

Спадне тестування. Автономно тестується тільки головний модуль (головна програма). Після тестування основного модуля до нього поступово підключаються комплекси (модулі) та компоненти наступного рівня тощо, поки програма не буде повністю зібрана та протестована.

Як тестувати комплекси, якщо компоненти, що в них містяться, ще не перевірені і, можливо, ще не написані? Заглушки використовуються для моделювання функцій ще не створених модулів, які моделюють роботу відсутнього модуля.

Недоліки тестування «спадного» такі ж, як і для «висхідного». Підвищуються вимоги до складності та якості заглушок, які потім прибираються. Переваги тестування:

- метод дозволяє поєднувати модульне тестування, тестування зв'язків і тестування функцій введення;
- рівномірний розподіл тестової роботи на всьому періоді створення комплексу, і такий розподіл дозволяє виявити помилки в основному модулі на ранній стадії розробки.

На практиці рідко вдається використовувати один спосіб, необхідно використовувати ряд комбінованих способів.

Методи тестування компонентів

Перший метод розглядає тестування програм як «чорний ящик», без урахування внутрішньої структури програми (компонентів), тести будуються на основі функціональних властивостей програми, тобто спираються на її функціональні властивості. Такий підхід називається функціональним тестуванням.

При використанні структурного тестування враховується внутрішня структура програми. Інформація шляхом аналізу проходження даних від входу до виходу служить для раціональної організації тестування.

Для оцінки повноти тестування використовуються три критерії:

1) Якщо кожен оператор був виконаний хоча б один раз, то тестування вважається завершеним;
2) Якщо принаймні один перехід був здійснений для кожної дуги блок-схеми програми в процесі вирішення тестового прикладу, то тестування вважається завершеним;
3) Якщо кожен шлях проходить від входу до виходу принаймні один раз у процесі вирішення тестового прикладу, тестування завершується.

Налагодження – діяльність певного виконавця, яка зосереджується на процесі пошуку й налагодження програми на основі результатів самої програми, повідомлення компілятора та операційної системи. Важливою особливістю процесу налагодження є здатність виконавця експериментувати на ЕОМ для виявлення помилок. Але експерименти необхідно проводити в суворій відповідності з планом.

Процес налагодження полягає в багаторазовому повторенні наступних трьох етапів:

1) Виявлення помилки;
2) Локалізація помилки;
3) Виправлення помилки.

Виявлення помилки при налагодженні здійснюється шляхом прорахунку спеціально підібраних завдань на ЕОМ, результати яких відомі заздалегідь. Якщо тестовий приклад розв'язано правильно, то виникає більш складне завдання. Якщо програма не запускається, значить, вона має хоча б одну помилку.

Наступним кроком є визначення точного місця помилки. Локалізація помилок є процес вирішення неправильних завдань.

Етап *виправлення помилки* є найпростішим, але в той же час головною проблемою є не внесення нової помилки при виправленні.

Характерною особливістю програмного продукту є практична нездатність провести комплексне і повне тестування в нетривіальних випадках з метою виявлення помилок.

Відомий вислів Е. Дейкстри говорить, що експериментальне тестування програм може служити доказом наявності в них помилок, але ніколи не доводить їх відсутність[8]. Тому закономірно, що програмісти знайдуть можливість сформулювати і довести певні твердження про правильність створеної програми, подібно до того, як формулюються і враховуються в математиці теореми та розв'язки.

Верифікація — ідея математичного доказу коректності програм.

Верифікація зазвичай зводиться до підтвердження правильності програми щодо введених даних та вихідних специфікацій.

Найвідоміший метод називається методом індуктивних тверджень.

У цьому методі першим кроком є запис твердження про властивості вхідних і вихідних даних програми, а також результати в ряді проміжних точок, які називаються точками розрізу. Ці твердження сформульовані у певній формально-логічній системі. На основі цих тверджень і семантики операторної схеми програми формулюються умови перевірки за допомогою певних перетворень, які потім виконуються. Якщо умови виконані, то програма правильна щодо вхідних і вихідних даних.

Якщо довести істинність умов неможливо, то можна стверджувати, що є помилки в програмі або помилки в процедурі доведення (наприклад, помилкове твердження в якомусь пункті розділу).

Якщо програміст хоче написати правильну програму, він повинен дотримуватися наступних кроків:

- написати програму;
- визначити вхідні і вихідні умови для цієї програми;
- розрізати цикли і забезпечити кожну точку розрізу індуктивним твердженням;
- отримати верифікаційні умови;
- довести істинність верифікаційних умов;
- якщо не вдасться, то можна стверджувати, що якесь індуктивне твердження записано невірно або програма містить помилку.

Очевидно, що методи верифікації знайдуть широке застосування на практиці, якщо їх можна буде сформувати настільки явно, що можна буде перевіряти програми за допомогою комп'ютерних технологій.

Використання на практиці систем програмування для доведення коректності програм ефективно лише за допомогою ЕОМ.

В даний час інтенсивно розвиваються та вдосконалюються методи математичного доведення теорем. Їх застосування в програмуванні сприяє подальшому прогресу у цій галузі.

Ідеї перевірки програм впливають на загальну культуру програмування. Необхідність формального

опису вхідних і вихідних даних приводить програміста до правильного визначення інтерфейсу між програмними модулями. Механізм тверджень є чудовим інструментом для специфікації модулів. Крім того, намагаючись розробити індуктивні твердження, програміст змушений більш ретельно і глибше аналізувати свою програму і таким чином знаходити в ній помилки.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

В роботі визначено, що тестування, на сьогодні, є самим надійним способом забезпечення якості програмного забезпечення, що розробляється. Також тестування входить до набору результативних засобів сучасної системи забезпечення якості програмного продукту. З технічної точки зору тестування полягає в запуску програми на наборі вихідних даних і порівнянні отриманих результатів з раніше відомими, з метою визначення відповідності різних властивостей і характеристик програмного продукту замовленим функціям. Як одна з основних фаз процесу розробки програмного продукту, тестування характеризується відносно великим внеском у загальну трудомісткість розробку продукту, в результаті чого може бути досягнутий найбільший ефект у скороченні праці, особливо на етапі тестування.

Показано, що основний внесок в автоматизацію або генерацію коду слід робити особливо на цьому етапі. Хоча автоматизація тестування є відносно поширеною практикою в сучасному промисловому програмуванні або технологія перевірки вимог і специфікацій тільки починає робити свої перші кроки. Отже, завдання на найближче майбутнє полягає в тому, щоб розподілити робоче навантаження таким чином, щоб загальні витрати на виявлення більшості дефектів мали тенденцію бути мінімізованими через величезну кількість виявлення помилок на ранніх етапах розробки програмного продукту.

References

[1] O.V. Pomorova, T. O. Govorushchenko, "Analysis of methods and analysis of the quality of software systems," Radioelectronic and computer systems, no. 6 (40), p. 148–158, 2009.

[2] Kaner S., Folk D., Nguyen E.K. Software testing: Translated from English. – Kiev: DiaSoft. – 2000. – 544 p.

[3] Korotun T.M. Models and methods of engineering testing of software systems in conditions of limited resources. - Author's abstract. dis. ... Candidate of Physics and Mathematics sciences. – Kiev: Institute of Cybernetics. V.M. Glushkov National Academy of Sciences of Ukraine, 2005. – 21 p.

[4] Lavrischeva E.M., Korotun T.M. Construction of the process of testing software systems // Problems of programming. – 2002. –№1–2. – S. 272–281.

[5] Software Engineering. An Object-Oriented Perspective. Eric J. Braude. John Wiley and Sons, 2001, P 560.

[6] Dustin E., Rashka J., Paul J. Automated Software Testing Introduction, Management and Performance -Addison-Wesley.: Publishing house: Lori, 1999. – 589 p.

[7] Gundecha M. Selenium Testing Tools Cookbook – Second Edition 2nd Edition / Unmesh Gundecha – Packt Publishing: November 23, 2019. – 326 p.

[8] Dijkstra E. The discipline of programming / per. from English I. Kh. Zusman; ed. E.Z. Lyubimsky. – M.: Mir, 1978. – 275 p.