**Лабораторна робота №02. Забезпечення якості та надійності програмного засобу**

**Мета:** засвоєння навичок забезпечення якості та надійності шляхом використання тестування та від лагодження, а також інтеграції Software Quality Control в Software Development Process.

**Методичні рекомендації до виконання лабораторної роботи.**

1. Опрацювати теоретичні відомості: повторити лекції 1-2.
2. Визначити характеристики якості та надійності для програмного засобу, розроблюваного за вашим проектом.
3. Обрати та обгрунтувати методику тестування розробленого ПЗ для перевірки цих характеристик.
4. Розробити пакети тестів для тестування розробленого ПЗ та визначити очікувані результати тестування. Інформацію надати у вигляді таблиці.
5. Визначити параметри, які дозволять зробити висновки щодо якості розробленого ПЗ за результатами тестування.
6. Робота повинна бути виконана згідно критеріїв оформлення документації та повинна містити

* Назва лабораторної роботи.
* Прізвище, група
* Назва проекту.
* Результати роботи оформлюються у вигляді таблиці.

1. По закінченню лабораторну роботу потрібно здати на перевірку викладачеві

По закінченню практичну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою на адресу [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com) . Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Файл з роботою повинен мати назву в такому форматі:

**PI<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвище англійською>**.. Наприклад, **РІ4101Р**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

Тему в заголовку листа записати

**ОПІ <Номер групи>-ЛР<Номер лабораторної>-<Прізвищеанглійською>**

**Строк виконання цієї роботи ІПЗ-41 – 09.10.2023**

**Теоретичні відомості**

***Інтеграція Software Quality Control в Software Development Process***

Software Development Process – структура накладена на розробку ПЗ. Процеси розробки ПЗ складаються з великої кількості етапів. Нижче наведені основні активності, які розташовані в послідовності «водоспадного» процесу (Waterfall process). Послідовність етапів в інших процесах може бути змінена.

Аналіз вимог (Requirements Analysis)

Складання вимог, яким повинен відповідати розроблювальний продукт, є першим кроком до створення необхідного ПЗ. Доки замовник, можливо, вважає, що він знає, що повинне робити необхідне ПЗ, може знадобитися знання й досвід у розробці ПЗ, щоб *розпізнати недостатність, двозначність і протиріччя у вимогах*.

Специфікації (Specification)

Ця активність точно описує ПЗ, яке потрібно розробити в строгій математичній формі. У реальності самі вдалі специфікації пишуться на зрозумілі й точно врегульовані додатки, які були успішно розроблені в минулому. Специфікації найбільш важливі для замовників які хочуть переконатися в стабільності майбутнього продукту.

Програмна архітектура (Software Architecture)

Архітектура ПЗ є абстрактним представленням розроблювального додатка. Основна мета даного етапу - переконатися що система буде відповідати вимогам, а також майбутні вимоги зможуть бути реалізовані. Ще одною важливою метою на даному етапі є розгляд всіх інтерфейсів між розроблювальним продуктом та іншим ПЗ, а також визначення необхідних апаратних засобів і ОС для успішної розробки.

Програмування (Coding)

На даному етапі створюють дизайни / проектні описи (design у цьому випадку - це фаза в розробці ПЗ, на якій функціональні специфікації транслюються в текст програми) і реалізують їх за допомогою певних технологій і мови програмування. Спрощення програмування за допомогою дизайнів є очевидною частиною розробки ПЗ, проте складання дизайнів не повинне бути більшою частиною даного етапу.

Тестування (Testing)

Тестування частин системи, особливо в тих місцях де код двох та більше розроблювачів повинен працювати разом.

Документування (Documentation)

Важлива частина (якій найчастіше не приділяють значення) - документування внутрішніх дизайнів ПЗ для подальшої підтримки і розширення. Документація - найважливіший етап для замовника.

Підтримка (Maintenance)

Підтримка й розширення ПЗ разом зі знайденими проблемами або новими вимогами можуть вимагати набагато більше часу, ніж розробка самого ПЗ. Не тільки додавання коду, що не був написаний відповідно до дизайну, а й визначення, як ПЗ працює з різних сторін після закінчення розробки, може потребувати значних зусиль зі сторони інженерів. Близько 2/3 роботи всіх інженерів - підтримка, проте така статистика може ввести в оману. Лише мала частина підтримки це виправлення помилок. Більша частина - це додавання нових можливостей системи.

***Статичне тестування***

Аналіз вимог

Процес статичного тестування починається з визначення й аналізу вимог до системи. Вивчаються й уточнюються:

– існуючі матеріали й методичні питання (технічне завдання на розробку, робітник і технічний проект та ін.);

– основні використовувані поняття;

– терміни й визначення;

– очікувані функціональні вимоги до системи, такі як вимоги до інтерфейсу (описують входи, одержувані із зовнішніх систем, і виходи, що направляються в зовнішні системи, чи накладаються на ці інтерфейси якісь обмеження), даним (описують вхідні й вихідні дані системи, формат даних, їх зберігання), продуктивності (описують проблеми масштабування й синхронізації, наприклад, скільки користувачів одночасно повинна обслуговувати система);

– вимоги до користувачів і людського фактору (хто буде працювати із програмним забезпеченням, облік необхідного рівня зручності й простоти використання);

– вимоги до фізичних засобів тестування (операційна система під керуванням якої виконується програмний продукт, і обчислювальна платформа, на якій експлуатується система);

– вимоги до безпеки (доступ до програмного продукту й керування даними), документації (визначається в якому виді вона повинна бути й чи повинна бути взагалі);

– вимоги до усунення несправностей (реакція системи на несправності);

– вимоги до супроводження (визначається, як виробляється усунення проблем, виявлених у системі).

На даній стадії документація по тестуванню не розробляється.

***Якість ПЗ***

Якість ПЗ визначається при відсутності в ньому дефектів. Дефектами можна вважати властивості і особливості ПЗ, які роблять майже неможливим отримання від нього заявленої функціональності. Це можуть бути неточності у визначенні вимог або помилки проектування.

Тестування програмного забезпечення – процес дослідження програмного забезпечення з метою отримання інформації про якість продукту. Всі існуючі на нинішній день методики тестування ПЗ не дозволяють цілком і повністю виявити усі дефекти, а також встановити коректність функціонування компонентів ПЗ. Через це усі існуючі методики тестування ПЗ являються засобом, який дозволяє формально встановити показник рівня якості розробленого ПЗ.

З точки зору стандарту ІSО 9126 якості програмних засобів можна визначити як сукупну характеристику досліджуваного ПЗ з урахуванням таких складових:

* надійність
* супроводжуваність
* практичність
* ефективність
* мобільність
* функціональність

Тестувальники використовують тестові скрипти на різних рівнях, їх пишуть для перевірки компонентів, в яких можливі помилки.

На тестування рекомендується виділяти від 8% до 25% часу, який йде на розробку системи. Часто тестоване програмне забезпечення збирається зі спеціальними настройками або бібліотеками і запускається в спеціальному оточенні - процес «покриття коду», практично це тестування білого ящику . Останнім часом існують різні методики тестування, які включають в себе кілька методологій, найчастіше такими методами є ТDD (*Test-driven development*) - керована тестами розробка, розробка через тестування. ТDD — технологія розробки програмного забезпечення, яка використовує короткі ітерації розробки, що починаються з попереднього написання тестів, які визначають необхідні покращення або нові функції. Кожна ітерація має на меті розробити код, який пройде ці тести. Нарешті, програміст або група вдосконалюють код для погодження змін. Один із ключових моментів TDD полягає у тому, що підготовка тестів перед написанням самого коду пришвидшує процес внесення змін.

Тестування базується на двох методологіях: UNIT-тестуванні (наприклад, для Java це на JUNIT-тестуванні) і рефакторингу. Це ітеративний процес, кроки повторюються до тих пір поки не досягнуто потрібної якості коду. ТDD дає переваги: спрощує розробку, дає можливість постійно використовувати регресійне тестування покращує комунікацію, пояснює вимоги до програм і покращує дизайн, зменшує складність за рахунок невнесення зайвого коду. JUNIT-тестування є тестовим для JAVA і використовує відповідні анотації. Попереднє написання тестів (до коду) дозволяє звести мінімум до часу між виявленням і виправленням дефектів, це одна з методик ефективної розробки ПЗ.

Існує декілька ознак по яким прийнято проводити класифікацію видів тестування, звичайно виділяють такі:

***По об'єкту тестування:***

* функціональні тестування
* тестування продуктивності
* стрес-тестування
* тестування стабільності
* тестування зручності використання
* тестування інтерфейсу користувача
* тестування безпеки
* тестування локалізації
* тестування сумісності.

***По знанню системи:***

* Тестування чорного ящику
* Тестування білого ящику
* Тестування сірого ящику

***За ступенем автоматизації:***

* ручне тестування
* автоматизоване
* напівавтоматизоване.

***За ступенем ізольованості компонент:***

* модульне тестування
* інтеграційне тестування
* системне тестування

***За часом проведення тестування***

***За ознакою позитивності сценаріїв***

***За ступенем підготовки до тестування***

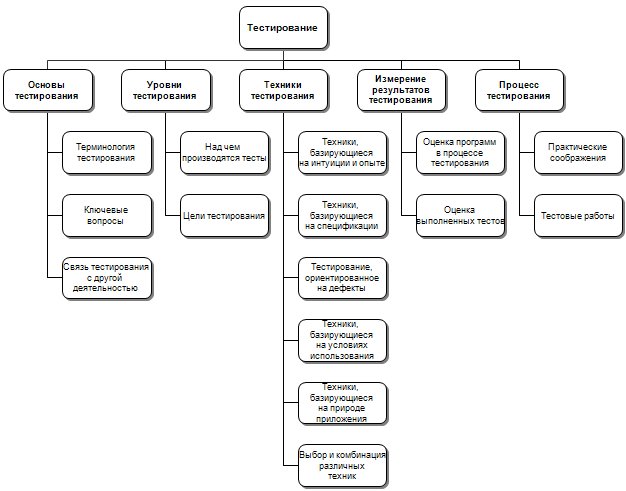
Тестування — це одна з технік контролю якості, що включає в себе:

* Планування робіт (Test Management)
* Проектування тестів (Test Design)
* Виконання тестування (Test Execution)
* Аналіз отриманих результатів (Test Analysis).

Існують такі рівні тестування:

* Приймальне тестування
* Установче тестування
* Модульне тестування
* Інтеграційне тестування
* Системне тестування
* Альфа-тестування
* Бета-тестування
* Тестування продуктивності
* Тестування «білого ящика» і «чорного ящика»
* Статичне і динамічне тестування
* Регресійне тестування

На рисунку 1. Зображено схему тестування програмного забезпечення.

Рисунок 3 – Тестування програмного забезпечення