ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ТЕЛЕКОМУНІКАЦІЙ

Кафедра системного аналізу

Пояснювальна записка

до магістерської роботи

на ступінь вищої освіти магістр

на тему: **«УДОСКОНАЛЕННЯ ТА АДАПТАЦІЯ СТВОРЕННЯ СУПУТНЬОЇ ДОКУМЕНТАЦІЇ ПРОЦЕСУ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»**

Виконала: студентка 6 курсу, групи САДМ-61

спеціальності 124 Системний аналіз
(шифр і назва спеціальності)

Лоранець Р.В.
(прізвище та ініціали)

Керівник Гордієнко Т.Б.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Нормоконтроль
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

ТЕЛЕКОМУНІКАЦІЙ

Кафедра Системного аналізу

Ступінь вищої освіти Магістр

Спеціальність 124 Системний аналіз

ЗАТВЕРДЖУЮ
Завідувач кафедри
системного аналізу
_____ Гордієнко Т.Б.
«__» _____ 2022 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Лоранець Риммі Володимирівні
(прізвище, ім'я, по батькові)

1. Тема роботи: Удосконалення та адаптація створення супутньої документації процесу тестування програмного забезпечення

Керівник роботи Гордієнко Тетяна Богданівна, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «__» 2022 року № _____

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи

3.1 Науково-технічна література та електронні джерела

3.2 Теоретичні дані про роль документації у тестуванні ПЗ

3.3 Аналіз міжнародного стандарту створення документації. Види документів та їх особливості.

4. Вивчення сфери тестування. Виявлення проблем і умов в робочому процесі, що спонукають до удосконалення та адаптації документів.

4.1 Побудова процесу адаптації, створення та ведення тестової документації в залежності від поточних робочих умов.

5. Перелік графічного матеріалу

5.1

5.2

5.3

5.4

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератути		Виконала
2	Аналіз та розбір міжнародного стандарту. Вивчення кожного його елементу.		Виконала
3	Пошук додаткової інформації до кожного типу документу.		Виконала
4	Пошук актуальних проблем в тестуванні в сучасних компаніях по розробці ПЗ		Виконала
5	Пошук рішення проблем. Вирішення проблеми неповноти документування процесу		Виконала
6	Розробка процесу удосконалення тестової документації в відповідності до поточного процесу тестування		Виконала
7	Розробка обов’язкових демонстраційних матеріалів		Виконала
8	Попередній захист роботи		Виконала
9	Пред’явлення роботи в деканат		Виконала

Студентка _____ Лоранець Р.В.
(підпис)

Керівник роботи _____ Гордієнко Т.Б.
(підпис)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

ПОДАННЯ
ГОЛОВІ ДЕРЖАВНОЇ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Направляється студент до захисту магістерської роботи Лоранець Р.В
(прізвище та ініціали)
за спеціальністю 124 Системний аналіз
(шифр і назва спеціальності)
на тему: Удосконалення та адаптація створення супутньої документації процесу тестування програмного забезпечення

Магістерська робота і рецензія додаються.

Директор ННІТ _____ В.І. Кравченко
(підпис) (прізвище та ініціали)

Довідка про успішність

Лоранець Р.В. за період навчання в Навчально-науковому інституті телекомунікацій
(прізвище та ініціали)
з 20__ року до 20__ року повністю виконав навчальний план за напрямом підготовки, магістр, 124, системний аналіз з таким розподілом оцінок за:
національною шкалою: відмінно ____%, добре ____%, задовільно ____%;
шкалою ECTS: A ____%; B ____%; C ____%; D ____%; E ____%.

Провідний фахівець інституту _____ _____
(підпис) (прізвище та ініціали)

Висновок керівника магістерської роботи

Студент Лоранець Римма Володимирівна показав свою інженерну підготовку разом із володінням теоретичного матеріалу із поглибленим розумінням методики розробки і впровадження програмних засобів, адже якісно вміє володіти новими комп'ютерними технологіями та вміє детально користуватися навчальною, довідковою і науково-технічною літературами. Всі поставлені задачі виконувались сумлінно та якісно згідно виставленого технічного завдання.

Магістерська робота виконана на високому рівні і заслуговує оцінку «відмінно», а студенту Лоранець Риммі Володимирівні – присвоєння кваліфікації Магістр з системного аналізу.

Керівник роботи _____ Гордієнко Т.Б.
(підпис) (прізвище та ініціали)
« » _____ 20__ року

Висновок кафедри про магістерську роботу

Магістерську роботу розглянуто. Студент Лоранець Р.В.
(прізвище та ініціали)
допускається до захисту даної роботи в Державній екзаменаційній комісії.

Завідувач кафедри
Системного аналізу (підпис) Т.Б. Гордієнко (прізвище та ініціали)

« » 20__ року

ВІДГУК РЕЦЕНЗЕНТА

по магістерській кваліфікаційній роботі

Студентці **Лоранець Риммі Володимирівні**

на тему: «УДОСКОНАЛЕННЯ ТА АДАПТАЦІЯ СТВОРЕННЯ СУПУТНЬОЇ ДОКУМЕНТАЦІЇ ПРОЦЕСУ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Актуальність:

Необхідність грамотного ведення та створення документації для процесу тестування програмного забезпечення. У багатьох випадках проект можуть відхилити на етапі пропозиції чи прийняття через відсутність або неповноти тестової документації. Тому слід доповнити, що документація є містком між компанією та клієнтом.

В основу створення документації був покладений міжнародний стандарт, але в свою чергу, він не вимагає конкретних методологій тестування, підходів, засобів або інструментів і не визначає документацію щодо їх використання.

Виходячи з вищесказаного актуальним питання доповнення, удосконалення тестової документації. Вимоги до тестування та проекту можуть змінюватись, тому необхідно фіксувати зміни в процесі.

Позитивні сторони:

Дослідження розкриває мету та завдання кваліфікаційної роботи. Проведено оцінювання стандартних документів та опис запропонованих удосконалень, котрі в повній мірі можуть покращити робочий процес тестування та якість програмного забезпечення.

Недоліки:

Частина роботи з прикладами удосконалень має містити більшу кількість графічних матеріалів.

Висновки:

Незважаючи на недоліки, магістерська кваліфікаційна робота заслуговує оцінку **відмінно**, а студентка Лоранець Римма Володимирівна – присвоєння їй кваліфікації Магістр з системного аналізу.

Якість проекту (роботи)	
Виконано на замовлення підприємства	
Виконано за тематикою НДР	
Виконано з макетом	
Виконано з застосуванням ЕОЬ та МПТ	√
Має практичну цінність	√
Проект-частина комплексної теми	

Підпис рецензента _____

(П.І.Б.)

РЕФЕРАТ

Текстова частина магістерської кваліфікаційної роботи 71 с., 2 табл., 2 рис., 15 джерел.

ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ТЕСТ ПЛАН, СПЕЦИФІКАЦІЯ ТЕСТ ДИЗАЙНУ, СПЕЦИФІКАЦІЯ ТЕСТОВОГО ПРИКЛАДУ, ЗВІТ ПРО ПЕРЕДАЧУ ТЕСТОВОГО ЕЛЕМЕНТА, ЖУРНАЛ ТЕСТІВ, ЗВІТ ПРО ТЕСТОВИЙ ІНЦИДЕНТ, ПІДСУМКОВИЙ ЗВІТ ТЕСТУВАННЯ, ЧИТ-ЛИСТ, ЧЕК-ЛИСТ.

Об'єкт дослідження – супутня документація для тестування програмного забезпечення.

Предмет дослідження – удосконалення та адаптація тестової документації.

Мета роботи – за допомогою адаптованої та обновленої документації, покращити робочий процес тестування, що в результаті призведе до підвищення якості програмного забезпечення.

Методи дослідження – аналіз та вивчення існуючих міжнародних стандартів, детальний розбір кожного тестового документа. Створення додаткових практичних документів.

Отримані результати та їх новизна – проаналізовано структуру міжнародного стандарту, а саме ISO/IEC/IEEE 29119-3:2021. Software and systems engineering. Software testing. Part 3: Test documentation. Здійснено детальний розгляд кожного елемента документу та виявлення його ролі у процесі тестування. Встановлено умови робочого процесу в сучасних IT компаніях, котрі вимагають адаптувати вже існуючу документацію, або підсилювати процес тестування із застосуванням додаткових документів. Запропоновано приклади складання таких документів як чек-лист та чит-лист.

Практичне значення одержаних результатів – матеріали проведеного дослідження стануть у нагоді для подальшого їх використання компанією у процесі тестування ПЗ, підвищення якості продукту, та розвитку даного напрямку вцілому.

ЗМІСТ

ВСТУП

У сучасному світі практично неможливо знайти сферу де б не використовуватись інформаційні технології. Завдяки цьому людині відкривається доступ до світових інформаційних ресурсів, що дають нові можливості для роботи і відпочинку. Вони багато в чому полегшують роботу та життя сучасної людини.

Інформаційні технології дуже швидко перетворилися на життєво важливий стимул розвитку не тільки світової економіки, а й інших сфер людської діяльності. Виробництво і транспорт, банки і біржі, засоби масової інформації і видавництва, оборонні системи, соціальні та правоохоронні бази даних, сервіс і охорона здоров'я, навчальні процеси, офіси для переробки наукової та ділової інформації, нарешті, Інтернет – усюди інформаційні технології.

Так як сфера інформаційних технологій є динамічною і її удосконалення відбувається кожен день, тому змінюється технічний прогрес, з'являється нове програмне забезпечення. Тому програмісти щодня займаються розробкою нових функцій, програм та технічних засобів для зручності користувачів.

Слід зазначити той факт, що програми, котрі створюється і використовується людьми можуть мати помилки. Якщо користувач допустить помилку, це може призвести до проблеми в роботі програми – вона виконується неправильно, а це означає, може повести себе зовсім не так, як очікувалось. Деякі помилки можуть бути незначними, в той час як інші – мати буквально руйнівні наслідки. Тому тестування програмного забезпечення набуває великого значення, бо кожен продукт повинен проходити перевірку, перш, ніж його можна буде ефективно та безпечно використовувати.

Одним з найголовніших, дещо бюрократичним, етапів тестування програмного забезпечення – є створення тестової документації.

В даній роботі буде піднята тема міжнародних стандартів ведення та створення тестової документації, а саме ISO/IEC/IEEE 29119 [1]. Це набір основних тестових документів, які пов'язані з динамічними аспектами тестування

програмного забезпечення (тобто виконання процедур і коду). Визначення призначення, структуру та зміст кожного основного документа. Незважаючи на те, що документи, описані в стандарті, зосереджені на динамічному тестуванні, деякі з них можуть бути застосовані до інших видів діяльності з тестування. Охоплено документацію на електронних носіях, а також паперову. Стандарт не вимагає конкретних методологій тестування, підходів, методів, засобів або інструментів і не визначає документацію щодо їх використання. Він також передбачає і не нав’язує конкретні методології для контролю документації, керування конфігурацією або забезпечення якості.

Наукова новизна роботи. Запропоновано універсальний підхід до складання таких документів як чек-лист та чит-лист з урахуванням умов робочого процесу в сучасних IT компаніях для покращення процесу тестування програмного забезпечення.

Практична цінність. Розглянуті підходи сприятимуть удосконаленню та синхронізуванню процесів забезпечення якості із стандартами на документи з тестування програмного забезпечення, що в довгостроковій перспективі дозволить заощадити кошти компанії на навчання та вирішення проблем, спричинених відсутністю документів для розробки тестування.

1 АНАЛІЗ ДОКУМЕНТІВ ТА СТАНДАРТІВ, ЯКІ РЕГЛАМЕНТУЮТЬ ПОРЯДОК СТВОРЕННЯ ТА ВЕДЕННЯ ТЕСТОВОЇ ДОКУМЕНТАЦІЇ

1.1 Роль документації у тестуванні

У процесі тестування можуть бути застосовані відмінні техніки, методики, практики. Оскільки у процесі тестування кожен має право вибирати те, що йому потрібно на даний момент, для вирішення конкретної проблеми, задачі в роботі.

Насправді, документ – це не проста формальність. Це дуже важлива частина розробки проектів та правильного функціонування поважаючої себе IT-компанії . Від грамотного ведення документації тестувальників залежать: терміни розробки, відстеження готовності продукту до запуску, аналіз помилок та того, що необхідно зробити, пріоритетність функціоналу, відповідальність членів команди за свої блоки завдань, звітність перед продакт-овнером, покращення роботи тощо. Тому єдиний підхід до оформлення, поєднаний з турботою про потенційного читача, спрощує роботу всієї команди.

Одна із важливих причин створення тестової документації полягає в тому, щоб зменшити або усунути будь-які невизначеності щодо діяльності тестування. Допомагає усунути неоднозначність, яка часто виникає під час розподілу завдань. Документація не лише пропонує систематичний підхід до тестування програмного забезпечення, але і виступає як навчальний матеріал для новачків у процесі тестування програмного забезпечення. Це також хороша стратегія маркетингу та продажів, щоб продемонструвати тестову документацію та зрілий процес тестування. Тестова документація допомагає запропонувати клієнту якісний продукт у визначені терміни. У розробці програмного забезпечення тестова документація також допомагає створити або налаштувати програму за допомогою документа конфігурації та посібників оператора. Ведення документації також допомагає підвищити прозорість із клієнтом та отримати його довіру.

Хороша документація та планування завжди призводять до кращої якості тестування програмного забезпечення та продуктів. Тому більшість успішних компаній приділяють велику увагу документам тестування програмного забезпечення, оскільки це є ключем успішного проекту.

У багатьох випадках проект навіть відхиляють на етапі пропозиції чи прийняття через відсутність документації. Це також може призвести до поганого імені організації на ринку. Не буде помилкою доповнити, що документація є містком між компанією та клієнтом.

Як правило, спроби вказати важливі моменти через повідомлення або дзвінки не є ефективними. Вся справа у людському факторі. У деяких занадто багато інформації для обробки та запам’ятовування. Інші можуть неправильно зрозуміти угоди. У підсумку кожен має власну інтерпретацію вимог і цілей.

Відсутність документації може серйозно вплинути на роботу інженерів із забезпечення якості, особливо при роботі зі складними продуктами або за умов, що часто змінюються. Нездатність зрозуміти, як має працювати функція та чому, призводить до появи нових помилок. Визначення пріоритету неправильної частини функціональних можливостей призводить до пропуску дефектів і надання неповних звітів. Приклади можна продовжувати нескінченно.

Усі приклади тестової документації, котрі викладені у стандартах, спеціалісти у різних компаніях сприймають по-різному. Тому корисно регулярно збиратися командами та випрацьовувати спільне бачення, власні стандарти та підходи. У довгостроковій перспективі це заощадить купу часу та підвищить якість роботи не тільки тестувальника але й команди в цілому.

Тестова документація є динамічною. Вона ефективна тише якщо команда контролю якості регулярно оновлює її. Якщо ми говоримо про документацію просто так, як вона існує, це немає ніякого сенсу. Вимоги можуть змінюватися. Пріоритети можуть змінюватися. Усе це впливає на охоплення тестуванням, необхідні ресурси тощо. Якщо команда не фіксує зміни, то вони отримують неефективні документи та невідповідності в процесі.

Створювати тестову документацію чи ні - кожна компанія має вирішити сама. Фахівці із забезпечення якості можуть рекомендувати клієнтам це робити, але клієнт вже вирішує погоджуватись чи ні. Що стосується переваг робочої тестової документації, що підтверджує процес, то вони цілком очевидні. Тестові артефакти допомагають тримати інформацію в порядку та дозволяють навіть новачкам легко зрозуміти, що відбувалося на проекті з самого початку. І хоча створення документації потребує додаткового часу, спроби з'ясувати заготовки для неї завжди набагато складніші та трудомісткіші.

Тестову документацію треба постійно актуалізувати, інакше вона не тільки стає марною, а й може заплутати тестувальників. Є перевантаження артефактів тестування, які створюються, переглядаються, затверджуються, використовуються, підтримуються та розповсюджуються. Тому є чіткі процедури, як створити документ, як ним користуватися, кому він має бути призначений.

Далі будуть розглянуті у роботі також всі види тестової документації їх роль та специфіку використання. Також опишу важливі етапи її удосконалення. Розберемо ситуації у компанії, котрі вимагають негайного удосконалення тестової документації.

1.2 Види документів та їх особливості.

Документи стандартів IEEE розробляються товариствами IEEE та Координаційними комітетами зі стандартів, Ради стандартів, Асоціації стандартів IEEE (IEEE-SA). Члени в комітетах служать добровільно та без винагороди. Вони не обов'язково є членами Інституту. Стандарти, розроблені в рамках IEEE, представляють консенсус широкого досвіду в області суб'єкта в рамках Інституту, а також ті види діяльності за межами IEEE, які висловили зацікавленість в участі в розробці стандарту.

Використання стандарту IEEE є цілком добровільним. Існування стандарту IEEE не передбачає що немає інших способів виробляти, тестувати, вимірювати, купувати, продавати або надавати інші товари та послуги, пов'язані зі сферою

застосування стандарту IEEE. Крім того, точка зору, висловлена в час затвердження та видання стандарту може бути зміненою через технологічний розвиток, стан продукту та коментарів, отриманих від користувачів стандарту. Кожен стандарт IEEE підлягає перегляду принаймні кожні п’ять років для перевірки та повторного підтвердження. Якщо документу більше п’яти років і не було повторно підтверджено, розумно зробити висновок, що його вміст, котрий все ще має певну цінність, не повністю відображає сучасний рівень техніки. Користувачів попереджають перевіряти, чи вони мають останнє видання будь-якого стандарту.

Коментарі щодо перегляду стандартів IEEE вітаються від будь-якої зацікавленої сторони, незалежно від членства в IEEE. Пропозиції щодо внесення змін до документів мають бути оформлені у спеціальній формі, а запропонована зміна тексту повинна бути разом з відповідними допоміжними коментарями.

Оскільки стандарти IEEE представляють консенсус усіх зацікавлених сторін, важливо переконатися, що будь-яке тлумачення також має збіг та баланс інтересів. З цієї причини IEEE та його члени товариства та координаційні комітети зі стандартів не можуть надати миттєву відповідь на інтерпретацію запитів, за винятком тих випадків, коли питання раніше було офіційно оформлено на розгляд.

Метою стандарту є опис набору основних тестових документів програмного забезпечення. Стандартизований тестовий документ може полегшити спілкування, забезпечуючи загальну систему відліку (наприклад, клієнті постачальник має те саме визначення для плану тестування). Визначення змісту стандартизованого тестового документа може служити контрольним списком повноти пов’язаного процесу тестування. Стандартизований набір також може забезпечити базовий рівень для оцінки поточної практики тестової документації. У багатьох організаціях використання цих документів значно підвищує керованість процесу тестування програмного забезпечення. Підвищена керованість є результатом значно покращеної видимості кожної фази процесу тестування.

Стандарт тестової документації визначає форму та зміст окремих тестових документів. У ньому не вказано необхідний набір тестових документів. Передбачається, що необхідний набір тестових документів буде визначено, коли буде застосовано прийнятий стандарт.

Документи, викладені у стандарті, охоплюють планування тестування програмного забезпечення, специфікації і звіти про тестування. План тестування визначає обсяг, підхід, ресурси та графік тестування. Він ідентифікує предмети і функції, котрі потрібно перевірити, завдання тестування, які необхідно виконати, відповідальний персонал за кожне завдання, та ризики, пов’язані з планом.

Специфікація тесту охоплюються трьома видами документів:

- Специфікація дизайну тесту вдосконалює підхід до тестування та визначає особливості, які повинні бути охоплені дизайном і пов'язаними з ним випробуваннями. Він також визначає тестові випадки та тестові процедури, якщо такі є, необхідні для цього завершити тестування та вказати критерії проходження функції.
- Специфікація тестового випадку документує фактичні значення, що використовуються для введення разом із очікуваними результатами. Тестовий приклад також визначає обмеження на тестові процедури, що є результатом використання цього конкретного тестового випадку. Тестові випадки відокремлені від тестових дизайнів, щоб дозволити використовувати їх у кількох дизайнах можливість повторного використання в інших ситуаціях.
- Специфікація процедури випробування визначає всі кроки, необхідні для експлуатації системи та здійснення визначення тестового випадку, щоб реалізувати відповідний тестовий дизайн. Процедури тестування розділені від специфікацій дизайну тесту, оскільки вони призначені для виконання крок за кроком і не повинні мати сторонніх деталей.

Звіт про тестування складається з чотирьох типів документів:

- Звіт про передачу тестових елементів визначає тестові елементи, які передаються для тестування у випадку, якщо залучаються окремі групи

розробки та тестування або у випадку, якщо бажано формальний початок виконання тесту.

- Тестовий журнал використовується командою тестування для запису того, що сталося під час виконання тесту.
- Звіт про інцидент тесту описує будь-яку подію, що відбувається під час виконання тесту, яка потребує подальшого розслідування.
- Підсумковий звіт про тестування підсумовує дії з тестування, пов’язані з однією або кількома специфікаціями проекту тестування.

Стандарт тестової документації описує набір основних тестових документів, які пов’язані з динамічними аспектами тестування програмного забезпечення (тобто виконання процедур і коду). Стандарт визначає мету, план і зміст кожного основного документа. Хоча документи, описані в стандарті, зосереджені на динамічному тестуванні, деякі з них можуть бути застосовані до інших видів тестування (наприклад, план тестування та звіт про інцидент тестування можуть використовувати для перегляду дизайну та коду).

Цей стандарт можна застосовувати до комерційного, наукового чи військового програмного забезпечення, яке працює на будь-якому цифровому пристрої. Застосовність не обмежується розміром, складністю чи критичністю програмного забезпечення. Однак, стандарт не визначає жодного класу програмного забезпечення, до якого його слід застосовувати. Стандарт стосується документація як початкового тестування розробки, так і тестування наступних версій програмного забезпечення. Для кожного виду програмного забезпечення, його можна застосовувати на всіх етапах тестування від тестування модуля до прийняття користувачем. Однак, оскільки всі базові тестові документи можуть бути не корисними на кожному етапі тестування, конкретні документи, які використовуватимуться на етапі, не визначено. Кожна організація, яка використовує стандарт, повинна буде вказати класи програмного забезпечення, до яких він застосовується, і конкретні документи, необхідні для конкретного етапу тестування.

Стандарт не вимагає спеціальних методологій тестування, підходів, методів, засобів або інструментів, а також не вказано документацію щодо їх використання. Може знадобитися додаткова тестова документація (наприклад, код, перевірочні контрольні листи та звіти). Стандарт також не передбачає і не нав’язує конкретні методології для контролю документації, управління конфігурацією або забезпечення якості. Додаткова документація (наприклад, план забезпечення якості) може знадобитися залежно від конкретних методологій, що використовуються.

У кожному стандартному документі зміст кожного розділу (тобто текст, який охоплює визначені теми) можуть бути адаптовані до конкретної програми та конкретної фази тестування. Окрім контенту для пошиття, до основного набору можуть бути додані додаткові документи, до будь-якого документа додаткові розділи та до будь-якого розділу можна додати додатковий вміст. Може бути корисно організувати деякі розділи підрозділи. Частина або весь вміст розділу може міститися в іншому документі, на який потім є посилання. Кожна організація, яка використовує стандарт, повинна вказати додаткові вимоги до змісту та конвенцій, щоб відобразити власні методики, підходи, засоби та інструменти для тестування, контролю документації, управління конфігурацією та забезпечення якості.

Стандарт може бути у вигляді документації на електронних носіях, а також на папері. Для документів, які потребують підписів, затвердження, слід використовувати папір, якщо система електронної документації не має безпечного механізму анотації затвердження, і цей механізм використовується.

У стандарті IEEE 829 існує вісім типів документів, які можна використовувати на трьох окремих етапах тестування програмного забезпечення:

- Тест план - Test Plan
- Специфікація тест дизайну - Test Design Specification
- Специфікація тестового прикладу - Test Case Specification
- Процедура тестування - Test Procedure
- Звіт про передачу тестового элемента - Test Item Transmittal Report

- Журнал тестів - Test Log
- Звіт про тестовий інцидент - Test Incident Report
- Підсумковий звіт тестування - Test Summary Report

Далі у таблиці показано зв’язки цих документів один з одним під час їх розробки та процесу тестування, який вони документують (Рис.1) Розглянемо кожен документ окремо і більш детально.

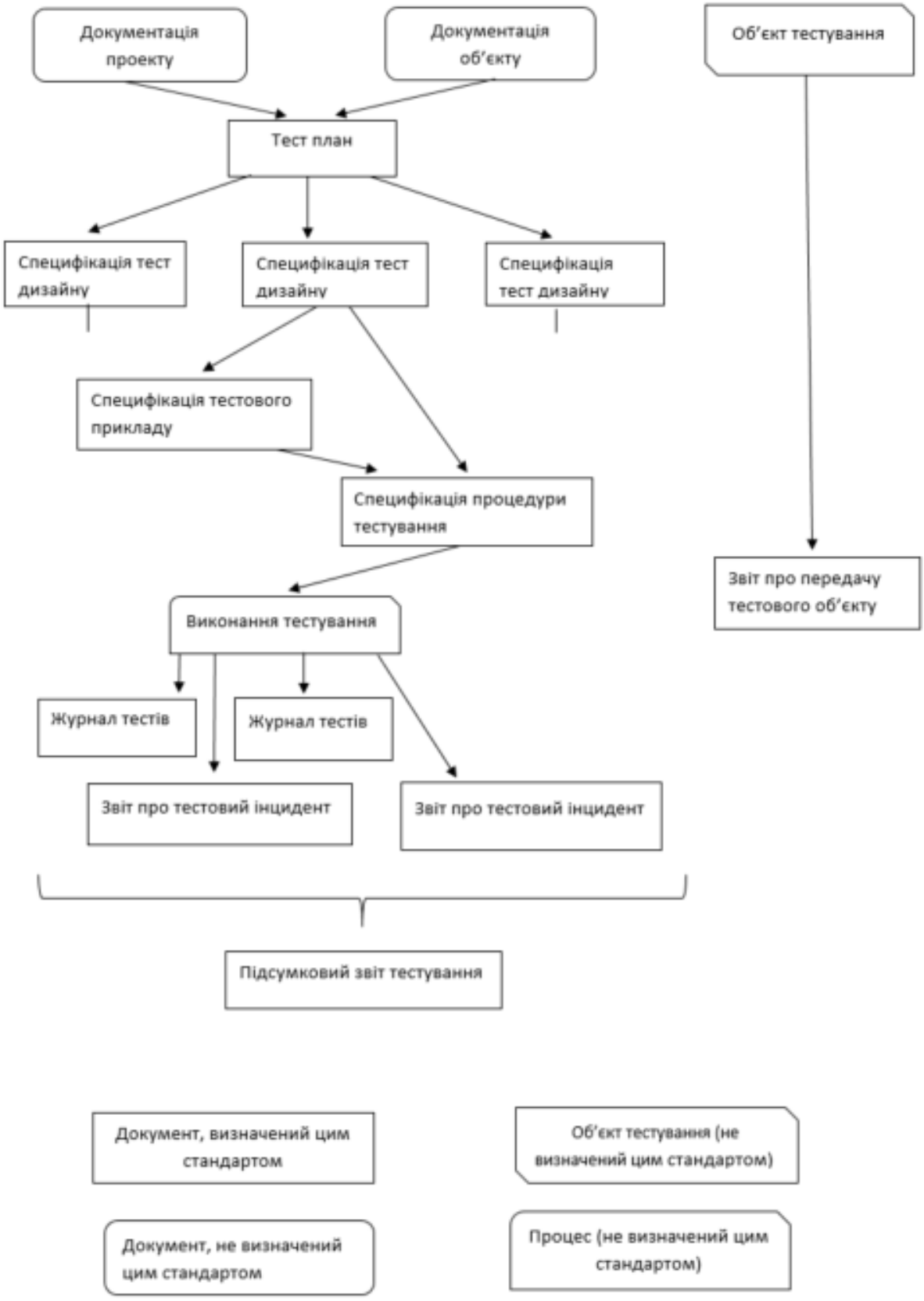

Рисунок 1 – Зв'язок документів під час розробки та процесу тестування

2. ДОСЛІДЖЕННЯ РЕГЛАМЕНТОВАНИХ МІЖНАРОДНИМ СТАНДАРТОМ ЕТАПІВ ДЛЯ СТВОРЕННЯ ТА ВЕДЕННЯ ТЕСТОВОЇ ДОКУМЕНТАЦІЇ

2.1 План тестування (Test Plan)

План тестування (Test Plan) - це детальний документ, у якому каталогізовано стратегію тестування, цілі, графік, оцінки, терміни виконання та ресурси, необхідні для завершення конкретного проекту, план для виконання тестів, необхідних для забезпечення належної роботи програмного забезпечення – під контролем менеджерів тестування.

Добре складений план тестування — це динамічний документ, який змінюється відповідно до прогресу в проекті та завжди залишається актуальним. Це точка відліку, на основі якої виконується та координується тестування між командою спеціалістів із забезпечення якості (QA).

План тестування також надається бізнес-аналітикам, керівникам проектів, командам розробників та будь-кому іншому, хто пов’язаний з проектом. Це головним чином забезпечує прозорість діяльності з контролю якості, щоб усі зацікавлені сторони знали, як буде тестуватися програмне забезпечення.

План розробляють менеджери з контролю якості або потенційні клієнти на основі вхідних даних членів команди з контролю якості (а іноді й інших). Його створення не повинно займати більше 1/3 часу, виділеного на весь проект.

Тест план є важливою складовою будь-якого грамотно організованого процесу тестування, оскільки містить в собі всю необхідну інформацію, що описує даний процес. Але в більшості випадків, з якими нам доведеться зіткнутися, тест-план буде грати більш формальну роль, але, все ж, його присутність має багато переваг. Наприклад:

- Можливість виставлення пріоритетів для задач з тестування.
- Побудова стратегії тестування, погодженої з усією командою.

- Можливість вести облік всіх необхідних ресурсів, як технічних, так і людських.
- Планування використання ресурсів на тестування.
- Прорахунок ризиків, можливих при проведенні тестування.
- Тест план допомагаэ особам, які не входять до команд контролю якості (розробникам, бізнес-менеджерам, командам, які займаються клієнтами), зрозуміти, як саме буде перевірятися веб-сайт або додаток.
- Пропонує чітке керівництво для інженерів із забезпечення якості для проведення тестування.
- Детально описані такі аспекти, як обсяг тесту, оцінка тесту, стратегія тощо. Об’єднання всієї цієї інформації в єдиний документ полегшує перегляд управлінському персоналу або повторне використання для інших проектів.

Залежно від конкретизації завдань, що описуються, тест-план може мати два рівня деталізації: майстер тест-план та детальний тест-план.

Детальний тест-план містить у собі завдання тестування для кожної команди, для кожного релізу або ітерації проекту. Створюється детальний тест-план або для декомпозованої частини проекту, або для невеликих проектів. Він може складатися з:

- Перелік областей тестування з пріоритетами.
- Стратегії тестування.
- Переліку можливих ризиків.
- Перелік необхідних ресурсів.
- Плану виконання проекту.

Майстер тест-план у свою чергу створюється або для організації процесу тестування між декількома командами, які тестують один проект, але мають різні завдання, або для проекту, який складається з безлічі ітерацій, які пов’язує якась загальна інформація, повторення якої в кожному релізі займає надто багато часу. У майстер тест-план входить:

- Загальна інформація про проект (посилання на документацію, баг-трекери, і т.д.)

- Положення, що описують процес тестування, заклади дефектів і т.д.
- Критерії готовності продукту до випуску.

План тестування повинен мати таку структуру:

- Ідентифікатор плану тестування: в даній частині вказується ідентифікатор, призначений цьому плану тестування.
- Вступ: в даній частині узагальнюють елементи програмного забезпечення та функції програмного забезпечення, які потрібно перевірити. Потреба в кожному предметі та його історія може бути включені. Посилання на такі документи, якщо вони існують, необхідні в плані тестування найвищого рівня:

1) Авторизація проекту;
2) План проекту;
3) План забезпечення якості;
4) План управління конфігурацією;
5) Відповідна політика;
6) Відповідні стандарти;

- Тестові елементи:в даній частині визначені тестові елементи, включаючи їхню версію/рівень перегляду. Також вказані характеристики їх передачі медіа, які впливають на вимоги до обладнання або вказують на необхідність логічних або фізичних перетворень завдяки котрим можна починати тестування (наприклад, програми повинні бути перенесені з стрічки на диск). Надається посилання на наступну документацію тестового елемента, якщо вона існує:

1) Специфікація вимог;
2) Специфікація проекту;
3) Посібник користувача;
4) Керівництво з експлуатації;
5) Інструкція зі встановлення.

Посилання на будь-які звіти про інциденти, пов’язані з тестовими елементами. Можуть бути визначені предмети, які слід спеціально виключити з тестування.

На цьому етапі визначаються цілі та очікувані результати виконання тесту. Оскільки всі тестування спрямовані на виявлення якомога більшої кількості дефектів, об’єкти повинні включати:

- Список усіх функцій програмного забезпечення – функціональність, графічний інтерфейс, стандарти продуктивності – які необхідно перевірити.
- Ідеальний результат або тест для кожного аспекту програмного забезпечення, яке потребує тестування. Це еталон, з яким будуть порівнюватися всі фактичні результати.
- Характеристики, що підлягають перевірці: в даній частині визначені всі функції програмного забезпечення та комбінації функцій програмного забезпечення для тестування. Визначена специфікація створення тестової документації, пов'язана з кожною функцією та кожною комбінацією функцій.
- Функції, які не підлягають тестуванню: в даній частині визначені всі функції та значущі комбінації функцій, які не будуть перевірені, а також причини.
- Підхід: в даній частині описаний загальний підхід до тестування. Для кожної основної групи функцій або комбінацій функцій вказано підхід, який забезпечить належне тестування цих груп ознак. Також вказуються основні види діяльності, методи та інструменти, які використовуються для тестування визначених груп ознак. Підхід описано досить детально, щоб дозволити визначити основні завдання тестування та оцінку часу, необхідного для виконання кожного з них. Вказано мінімальний бажаний ступінь повноти. Визначено прийоми, які будуть використовуватися для оцінки комплексності зусиль тестування (наприклад, визначення того, які оператори були виконані в принаймні один раз). Указано будь-які додаткові критерії завершення (наприклад, частоту помилок). Визначені значні обмеження для тестування, такі як доступність тестового елемента, доступність тестових ресурсів і терміни.

- Критерії проходження/непроходження пункту: в даній частині вказані критерії, які будуть використовуватися для визначення того, чи пройшов кожен тестовий елемент чи не пройшов тестування. Критерії виходу: визначає контрольні показники, які означають успішне завершення етапу тестування або проекту. Критерії виходу — це очікувані результати випробувань, які мають бути виконані перед переходом до наступного етапу розробки. Наприклад, 80% усіх тестових прикладів мають бути позначені як успішні, перш ніж певну функцію чи частину програмного забезпечення можна вважати придатною для загального використання.
- Критерії призупинення та вимоги до відновлення: в даній частині вказано критерії, які використовуються для призупинення всього або частини тестування для елементів тесту, вказано в тест плані. Вказано дії тестування, які необхідно повторити, коли тестування буде відновлено. Критерії призупинення: визначає контрольні показники для призупинення всіх тестів. Наприклад, якщо члени команди з контролю якості виявляють, що 50% усіх тестів виявилися невдалими, усе тестування призупиняється, доки розробники не усунуть усі виявлені на даний момент помилки.
- Результати тестування: в даній частині визначено документи з результату. Повинні бути включені такі документи:
  1) План тестування;
  2) Специфікації дизайну тесту;
  3) Тестові специфікації;
  4) Специфікації процедури тестування;
  5) Звіти про передачу тестових зразків;
  6) Журнали випробувань;
  7) Звіти про інциденти випробувань;
  8) Підсумкові звіти про випробування.

Вхідні та вихідні дані тесту слід ідентифікувати як результати. Також можуть бути включені засоби тестування (наприклад, драйвери модулів і заглушки).

Результати тестування посилаються на список документів, інструментів та іншого обладнання, яке необхідно створити, надати та підтримувати для підтримки діяльності з тестування в проекті. До, під час і після тестування потрібен інший набір результатів.

Тестові завдання: в даній частині визначено комплекс завдань, необхідних для підготовки та проведення тестування. Визначено всі міжзадачні залежності та необхідні будь-які спеціальні навички. На цьому етапі створюється детальна розбивка всіх ресурсів, необхідних для завершення проекту. Ресурси включають людські зусилля, обладнання та всю інфраструктуру, необхідну для точного та комплексного тестування. Ця частина плану тестування визначає міру ресурсів (кількість тестувальників і обладнання), необхідних для проекту. Це також допомагає керівникам тестування сформулювати правильно розрахований графік і оцінку для проекту.

Потреби середовища: в даній частині вказано як необхідні, так і бажані властивості тестового середовища. Ця специфікація повинна містити фізичні характеристики об’єктів, включаючи апаратне забезпечення, засоби зв’язку та системне програмне забезпечення, режим використання (наприклад, автономний) та будь-яке інше програмне забезпечення чи приладдя, необхідні для підтримки тесту. Також вказано рівень безпеки, який необхідно забезпечити для засобів тестування, системного програмного забезпечення та власних компонентів, таких як програмне забезпечення, дані та апаратне забезпечення. Визначено необхідні спеціальні тестові інструменти та будь-які інші потреби в тестуванні (наприклад, публікації чи офісне приміщення). Визначено джерело для всіх потреб, які зараз недоступні для тестової групи. Тестове середовище стосується налаштувань програмного та апаратного забезпечення, на яких спеціалісти з забезпечення якості (QA) запускають свої тести. В ідеалі тестове середовище має бути реальними пристроями, щоб тестувальники могли контролювати поведінку програмного забезпечення в умовах реального користувача. Незалежно від того, чи це ручне тестування, чи автоматизоване тестування, ніщо не може зрівнятися з реальними пристроями, інстальованими зі справжніми браузерами та

операційними системами, оскільки тестове середовище не підлягає обговоренню. Не шкодить результатам тестування за допомогою емуляторів або симуляторів.

Обов'язки: в даній частині визначені групи, відповідальні за управління, проектування, підготовку, виконання, спостереження, перевірку та вирішення. Крім того, визначені групи, відповідальні за надання тестових завдань та потреби середовища. Ці групи можуть включати розробників, тестувальників, оперативний персонал, представників користувачів, технічну підтримку, персонал адміністрування даних і персонал підтримки якості.

Потреби в кадрах і навчанні: в даній частині вказані потреби в тестовому персоналі за рівнем кваліфікації. Визначені варіанти навчання для отримання необхідних навичок.

Розклад: в даний пункт включені етапи тестування, визначені в розкладі проекту програмного забезпечення, а також усі події передачі елементів. Визначте будь-які додаткові етапи тестування. Оцінено час, необхідний для виконання кожного тестового завдання. Вказано розклад для кожного тестового завдання та етапу тестування. Для кожного ресурсу тестування (тобто приміщення, інструменти та персонал), вказано строки його використання. Для тестової оцінки треба розбити проект на менші завдання та виділіть час і зусилля, необхідні для кожного. Потім створити графік виконання цих завдань у визначений час із певною кількістю зусиль. Однак створення розкладу вимагає введення з кількох точок зору:

- Наявність співробітників, кількість робочих днів, терміни реалізації проекту, щоденна ресурсна забезпеченість.
- Ризики, пов’язані з проектом, який було оцінено на попередній стадії.
- Складання розкладу є поширеним терміном в управлінні проектами. Створивши надійний розклад у плануванні тестування, менеджер тестування може використовувати його як інструмент для моніторингу прогресу проекту, контролю за перевитратами. Щоб створити розклад проекту, менеджеру тестування потрібно кілька типів вхідних даних, наведених нижче:

- Співробітник і термін виконання проекту: робочі дні, термін виконання проекту, наявність ресурсів є факторами, які впливають на графік
- Оцінка проекту: на основі оцінки менеджер тестування знає, скільки часу потрібно для завершення проекту. Тому він може скласти відповідний графік проекту
- Ризик проекту: розуміння ризику допомагає менеджеру тестування додати достатньо додаткового часу до розкладу проекту, щоб впоратися з ризиками.
- Ризики та непередбачені обставини: в даній частині визначені припущення високого ризику плану тестування. Вказано плани на випадок непередбачених обставин для кожного (наприклад, затримка доставки тестових елементів може вимагати збільшеного розкладу нічних змін, щоб дотримуватися дати доставки).
- Схвалення: в даній частині вказано імена та посади всіх осіб, які повинні затвердити цей план. Передбачено місце для підписів і дати.

Секції розташовуються у вказаній послідовності. Додаткові розділи можуть бути включені відразу до схвалення. Якщо частина або весь вміст розділу знаходиться в іншому документі, то посилання на нього замість відповідного вмісту може бути вказаний матеріал. Довідковий матеріал повинен бути доданий до тестовий план або доступний користувачам плану.

Перед написанням тест плану треба якнайбільше дізнатися про продукт, який тестується, клієнта та кінцевих користувачів подібних продуктів. В ідеалі на цьому етапі треба бути зосередженому на таких питаннях як:

- Хто буде використовувати продукт?
- Яке основне призначення цього продукту?
- Як працює продукт?
- Які специфікації програмного та апаратного забезпечення?

Треба опитати клієнтів, дизайнерів і розробників, ознайомитися з товарною та проектною документацією, виконати покроковий огляд продукту.

План тестування в тестуванні програмного забезпечення є основою, на якій будується весь проект. Без достатньо розгорнутого та добре продуманого плану QA неминуче можна заплутатись з нечіткими, не визначеними цілями та термінами. Це дуже перешкоджає швидкому й точному тестуванню, уповільнює результати та затримує цикли випуску.

Основні етапи написання Тест плану можна коротко окреслити у схемі (Рис. 2)

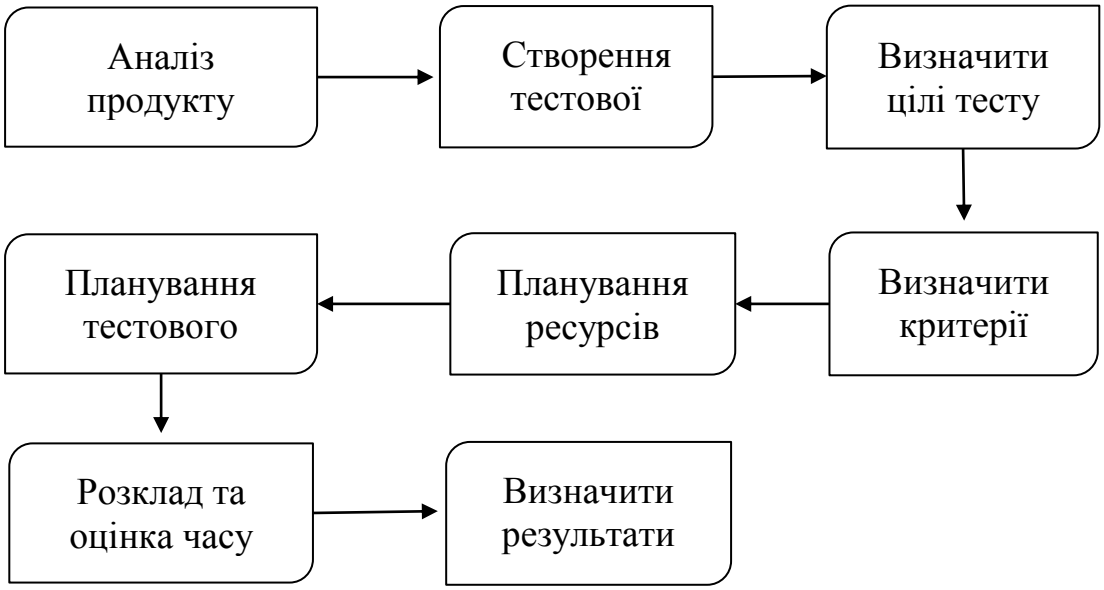

Рисунок 2 - Етапи написання Тест плану

2.2 Специфікація дизайну тесту - Test Design Specification

Однією з найважливіших цілей тестування є чітка порада щодо якості та ризику таким чином, щоб усі залучені сторони здобули довіру до продукту. Щоб це зробити, тестувальник повинен зібрати інформацію про поведінку системи. Одним із основних інструментів збору інформації є виконання тестів. Результати цих випадків дають інформацію про поведінку системи. Основні запитання: які тестові випадки? Скільки? І як ми отримуємо ці справи? Щоб відповісти на ці питання, дизайн тесту є незамінним.

Розробка правильного набору тестових випадків є важливою ланкою між стратегією тестування та реалізацією стратегії тестування – тестами, які виконуються.

Тест дизайн (Test Design) – етап процесу тестування програмного забезпечення, на якому проектуються та створюються тестові випадки, які відповідають заданим раніше цілям та критеріям тестування.

Тест дизайн - один із початкових етапів тестування програмного забезпечення, етап планування та проектування тестів. Тест дизайн є обдумуванням і написанням тестових випадків (test case), відповідно до вимог проекту, критеріями якості майбутнього продукту та фінальними цілями тестування.

Життєво важлива частина документа про тестування системи або програмного забезпечення, по суті, визначає тестове середовище для тестового продукту, комплексний підхід до тестування та розпізнає відповідний високий рівень тестових випадків. Основна мета цього звіту — вказати, які набори тестів і тестові приклади запускати, а які пропускати. Цей запис дає змогу колегам розпізнавати комбінацію функцій або набір функцій, які слід перевірити, і виділяти групу тестових прикладів, які дозволять їм легко перевірити ці функції продукту. Дані, записані в цьому звіті, в основному поділяються на три основні категорії, а саме:

- Умови тестування: тут наведено посилання на різні обмеження для тестування. Вони можуть покладатися на функціональність програмного забезпечення, продуктивність і структуру або можуть залежати від потреб клієнта.
- Детальний підхід: Методології, які використовуються для виконання тесту, описані в цьому записі детально. Причиною цієї частини документа є усунення вразливостей і складності в спеціалістах із забезпечення якості, як і в інших членах проекту.

Випадки тестування високого рівня: ціль цього сегменту специфікації дизайну тестування — запис даних, які характеризують функціональність продукту, без глибокого занурення в його функціональність. Це надає тестувальникам можливість відійти від охарактеризованих тестових випадків і досліджувати інші критичні та важливі тестові випадки.

Специфікація дизайну тесту є одним із записів, які можуть бути використані під час тестування програмного забезпечення. Він містить дані та параметри, які повинні відповідати методології перевірки.

Деякі недосвідчені тестувальники можуть пропустити формування тестової документації. Вони впевнені, що складання плану тестування, специфікації дизайну тесту або специфікації процедури тестування просто витрачає час. Це припущення недійсне.

Документація з тестування гарантує правильну перевірку фреймворку або програми. За допомогою специфікацій і планів тестування все чітко та добре організовано. Практика показує, що якщо виконання юзабіліті або функціонального тестування залежить від тестової документації, на цьому етапі воно буде все більш результативним.

Документи тестового дизайну служать для тестувальників програмного забезпечення, в яких пропонуються деталі процедури тестування та дії. Вони корисні для усунення сумнівів щодо методів і технік тестування. Тестові випадки документуються відповідно до стандарту тестової документації ISO/IEC/IEEE 29119-3.

Специфікація дизайну тесту складається з кількох компонентів. А саме:

- Ідентифікатор специфікації тестового дизайну визначає конкретний тип документа та допомагає відрізнити його від інших файлів.
- Функції, які потрібно перевірити – елементи або аспекти, які потребують перевірки. Компонент розглядається як об’єкт специфікації тестового дизайну.
- Approach Refinement – техніка або метод, який можна застосувати під час тестування мобільних або веб-додатків.
- Ідентифікація тесту. Це список тестових випадків, створених у дизайні тесту. Він містить ідентифікатор і короткий опис кожного тесту.
- Критерії склав/не склав. Передумови, які визначають, проходять чи не проходять компоненти програмного забезпечення процес тестування.

Існує три категорії специфікації дизайну тестування, кожна з яких виконує певну роль:

- Умови тестування – залежно від особливостей програмного забезпечення, вимог клієнта та обмежень тестування.
- Ретельний підхід. Він пояснює процедури, які допомагають позбутися невизначеності під час проведення випробувань продукту.
- Тестові випадки високого рівня. Він реєструє дані, що визначають функціональність програмного забезпечення.

Умови завдання визначають час і зусилля, доступні для тестування. Аналіз ризиків продукту (PRA) визначає, які частини системи є найважливішими для тестування та повинні бути перевірені більш ретельно. У стратегії тестування робиться огляд усього тесту та того, як зусилля тестування розподіляються між різними варіантами тесту, щоб найбільш ефективно охопити всі ризики. Частини тестового об’єкта, які тестуються, і ретельність, яка використовується під час тестування цих частин, визначають покриття тесту.

Щоб реалізувати стратегію тестування, стратегію тестування потрібно перекласти на тестові випадки. У багатьох випадках необхідно продемонструвати реалізацію стратегії тестування.

Як ми можемо спроектувати тестовий приклад? Це залежить від кількох факторів:

- Необхідне покриття
- Основа тестування - інформація про поведінку системи, на якій базуються тестові випадки.
- Спосіб організації процесу розробки програмного забезпечення (наприклад, водоспад проти Agile)
- Знання та досвід залучених людей
- Час і бюджет, доступний для виконання тестів

На основі цих факторів робиться вибір підходу до тестування, типу покриття та техніки розробки тесту.

Це призведе до набору тестових прикладів, які виконують стратегію тестування у спосіб, необхідний для виконання завдання.

Розробка ваших тестів за допомогою TMap Suite має деякі інші переваги:

- Використовуючи типи покриття під час розробки тестів, ви розробляєте тестові випадки з певним покриттям, спрямованим на пошук певних типів дефектів. Це дозволяє тестувати ефективніше. Таким чином ми знаходимо якомога більше дефектів із якомога меншою кількістю тестів.
- Ви можете зробити тести більш відтворюваними, оскільки вони розроблені за стандартним методом
- Стандартизований спосіб роботи робить тести більш незалежними від особи, яка розробляє та виконує ці тести.
- При використанні типів покриття великі частини дизайну та виконання тестів можна автоматизувати, наприклад, за допомогою тестування на основі моделі.

Хто насправді розробляє тести? Це залежить від типу організації. У більш традиційних компаніях із більш різким розмежуванням між ролями тестувальники та персонал із контролю якості відповідатимуть за розробку тестів — і, ймовірно, за всю іншу діяльність щодо тестів.

Однак «часи, вони змінюються». Через зростання автоматизації, гнучкості та DevOps межі між ролями в технологічних організаціях стають все тоншими. Сьогодні можна сказати, що тестування проводять усі. У розширенні це означає, що всі роблять тестовий дизайн.

Ось як це може виглядати на практиці:

- Інженери пишуть модульні тести, які передбачають розробку необхідних тверджень, налаштування та демонтаж класів, а також отримання тестових даних.
- Тестувальники, які не мають навичок кодування, можуть використовувати інструменти запису та відтворення для запису інтерфейсу користувача або наскрізних тестів, розроблених ними або іншими спеціалістами з контролю якості.

- Якщо організація приймає BDD (розробка, орієнтована на поведінку), вона може залучити клієнта або когось, хто виступає в якості його довіреної особи, до співпраці у створенні автоматизованих специфікацій.

Наведений вище список не є вичерпним. Але ви можете побачити, як розробка тестів може бути результатом роботи багатьох різних учасників.

Кожен може створити тестовий приклад - вам не обов'язково бути професійним тестером. Навіть машина може створити тестовий приклад.

Коли відбувається тест дизайн? Відповідь на це залежить від того, як справи йдуть в організації.

В організаціях, які все ще дотримуються більш традиційної, поетапної методології розробки, ви, ймовірно, також знайдете більш формальний життєвий цикл тестування програмного забезпечення, у якому розробка тестів є фазою з чітко визначеним місцем.

Однак сьогодні більшість організацій дотримуються гнучкого підходу до розробки програмного забезпечення. Такі підходи сприяють розробці за короткими ітеративними циклами. У таких сценаріях тестування – це вже не фаза, а дія, яку ви виконуєте якомога раніше в процесі. Якщо це так, дизайн тесту виконується кілька разів у контексті однієї ітерації.

Розглянувши «що», «чому», «хто» і «коли» дизайну тестів, єдине головне питання, яке залишилося, це «як». Розглянемо три методи розробки тестів.

Тестування класу еквівалентності

Одна з найскладніших речей під час розробки тестів — це знати, коли цього достатньо. Скільки прикладів вам потрібно? Наскільки повними мають бути ваші тестові дані?

Тестування класу еквівалентності, або розбиття класу еквівалентності, є вирішенням цієї дилеми. Цей метод говорить нам розділити наші тестові дані на великі «відра», які ми можемо вважати рівними. Тоді лише кілька значень у кожній групі — і на їхніх кордонах — достатньо, щоб переконатися, що наш тест є вичерпним.

Перехід до стану

Більшість функцій комерційного чи корпоративного програмного забезпечення можна описати як кінцеві автомати. Я знаю, що люди з інформатики люблять страшні назви, але ця концепція досить проста. Це стосується того, як дана система переходить з одного стану в інший після виконання певної дії.

Розробляючи тестовий приклад, ви можете використовувати діаграму або таблицю для посилання на очікувані зміни стану вашої програми. FitNesse, популярний інструмент для автоматизованого приймального тестування, робить саме це: він дозволяє розробникам, QA та іншим зацікавленим сторонам виражати поведінку своїх систем за допомогою таблиць, які зображують переходи між станами.

Дослідницьке тестування

Дослідницьке тестування — це, по суті, тестування, яке не дотримується жодного заздалегідь визначеного сценарію. Виконуючи таке тестування, тестувальники використовуватимуть свою креативність, знання предметної області та попередній досвід, щоб придумати нові та несподівані способи взаємодії з тестовою системою. Роблячи це, вони часто виявляють крайні випадки та несподівані проблеми.

Але якщо дослідницьке тестування за своєю природою не є сценарним, то як воно може бути технікою для розробки тестів?

Легко. За допомогою інструменту автоматизації тестування, який не вимагає навичок програмування, тестувальники можуть перетворити свої сеанси дослідницького тестування вручну в автоматизовані тестові випадки, які вони можуть виконати пізніше. Таким чином вони зберігають ідеї, отримані під час сеансів творчого тестування, перетворюючи їх у більш тривалу форму.

Специфікація дизайну тесту вдосконалює підхід до тестування та визначає функції, які повинні бути охоплені дизайном і пов’язаними з ним тестами. Він також визначає вимоги, тестові приклади та процедури тестування, необхідні для виконання тестування, і визначає критерії проходження/непроходження функції.

Специфікація проекту може включати необхідні розміри, фактори навколишнього середовища, ергономічні фактори, естетичні фактори, необхідне технічне

обслуговування тощо. Вона також може надавати конкретні приклади того, як має виконуватися проект, допомагаючи іншим працювати належним чином (керівництво для що людина має робити).

2.3 Специфікація тестового прикладу (Test Case Specification)

Один із документів де описані результати, що пропонується клієнту, специфікація тестового прикладу – це документ, який надає детальний підсумок того, які сценарії будуть тестуватися в програмному забезпеченні протягом життєвого циклу тестування програмного забезпечення (STLC). Цей документ визначає основну мету конкретного тесту та визначає необхідні вхідні дані, а також очікувані результати/виходи. Крім того, він виступає в якості посібника для виконання процедури тестування та окреслює критерії проходження та непроходження для визначення прийняття. Специфікація тестового випадку входить до числа тих документів, формат яких встановлено стандартом IEEE для тестового документа програмного забезпечення та системи (829-1998). За допомогою документа специфікації тестового випадку можна перевірити якість численних тестових випадків, створених на етапі тестування програмного забезпечення.

У документі зі специфікацією тестового прикладу описано докладний підсумок того, які сценарії будуть тестуватися, як вони тестуватимуться, як часто вони тестуватимуться тощо, і так далі, для певної функції. Він визначає мету конкретного тесту, визначає необхідні вхідні дані та очікувані результати, надає покрокові процедури для виконання тесту та описує критерії проходження/непроходження для визначення прийняття.

Специфікація тестового прикладу повинна бути виконана окремо для кожного блоку. На основі підходу, зазначеного в плані тестування, необхідно визначити функцію, яку потрібно перевірити для кожного блоку. Загальний підхід, викладений у плані, уточнюється в конкретних методах тестування, яких слід

дотримуватися, і в критеріях, які використовуватимуться для оцінювання. На основі цього тестові випадки вказуються для тестового блоку.

Однак план тестування — це набір усіх специфікацій тестування для певної області. План тестування містить огляд високого рівня того, що тестується для певної області функцій.

Є дві основні причини, чому тестові випадки вказуються перед їх використанням для тестування:

- Тестування має серйозні обмеження, і ефективність тестування значною мірою залежить від точної природи тесту. Навіть для певного критерію точний характер тестових випадків впливає на ефективність тестування.
- Створення хорошого тестового прикладу, який виявить помилки в програмах, є дуже творчим заняттям і залежить від тестувальника. Важливо переконатися, що набір використаних тестів має високу якість. Це головна причина для того, щоб специфікація тестового випадку була у формі документа.

Специфікація тестового прикладу розробляється на етапі розробки організацією, відповідальною за формальне тестування програми.

Документ із специфікацією тестового прикладу, розроблений організацією, яка відповідає за формальне тестування програмного забезпечення, має бути підготовлений окремо для кожного підрозділу, щоб забезпечити його ефективність і допомогти створити правильний та ефективний план тестування. Таким чином, для створення цього документа використовується такий формат:

- Цілі тесту: Мета тестування програмного забезпечення визначена тут детально. Релевантна та важлива інформація, яка може зробити процес зрозумілим для читача, в основному міститься в цьому розділі формату.
- Тестові елементи: Елементи (наприклад, специфікації вимог, специфікації дизайну, код тощо), необхідні для виконання конкретного тестового випадку. Це має бути надано у функції «Примітки» або «Додаток». Тут описано функції та умови, необхідні для тестування. Предмети та документи, які були потрібні перед виконанням конкретного тесту,

згадуються з належними доказами та записами. Крім того, він описує особливості та умови, необхідні для тестування. Інші важливі деталі, включені тут:

1) Специфікація вимог.
2) Технічне завдання на детальний проект.
3) Посібники користувача.
4) Інструкція з експлуатації.
5) Технічні характеристики системи.

- Специфікації вхідних даних: опис того, що потрібно (крок за кроком) для виконання тесту (наприклад, вхідні файли, значення, які потрібно ввести в поле тощо). Це потрібно вказати в полі «Дія». Після визначення попередніх умов команда працює разом, щоб визначити всі вхідні дані, необхідні для виконання тестів. Вони можуть відрізнятися залежно від рівня, для якого написаний тест.
- Вихідні характеристики: Опис того, як має виглядати система після виконання тесту. Це потрібно вказати в полі «Очікувані результати». Вони включають усі вихідні дані, необхідні для перевірки тестового випадку, такі як дані, таблиці, дії людини, умови, файли, таймінг тощо.
- Потреби середовища:тут команда визначає різні вимоги до середовища, заявлені командою після процесу тестування. Сюди входять архітектури системи, інструменти обладнання та програмного забезпечення, записи або файли, інтерфейси тощо.

Крім того, команда визначає будь-які спеціальні вимоги та обмеження щодо тестових випадків. Він складається з таких деталей, як:

- Обладнання: налаштування та обмеження.
- Програмне забезпечення: система, операційна система, інструменти тощо.
- Процедурні вимоги: спеціальне налаштування, розташування та ідентифікація виходу, операційні втручання тощо.
- Інше: поєднання програм, засобів, тренінгів тощо.

Внутрішні залежності: нарешті команда визначає тестові випадки будь-яких вимог або передумов. Тут тестові випадки задокументовані з посиланнями на інші вимоги та специфікації. Щоб забезпечити якість цих тестів, команда визначає наступні тести.

2.4 Специфікація процедури тестування (Test Procedure Specifications)

Специфікація процедури тестування — це документ, який визначає послідовність дій для виконання тесту. Процедури тестування перевіряють виконання вимоги. Розробку специфікації процедури тестування можна розпочати після завершення та схвалення тестових випадків і дизайну. Також даний документ створюється щоб вказати кроки для виконання набору тестових випадків або, загалом, кроки, які використовуються для аналізу програмного забезпечення для того, щоб оцінити набір ознак.

Специфікація процедури тестування включає такі елементи:

- Ідентифікатор специфікації процедури тестування. Потрібно вказати унікальний ідентифікатор, призначений цій специфікації процедури тестування. Надати посилання на відповідну специфікацію дизайну тесту.
- Призначення. Описана мета цієї процедури. Якщо ця процедура виконує будь-які тестові випадки, треба надати посилання для кожного з них. Крім того, надати посилання на відповідні розділи документації тестового елемента (наприклад, посилання на використання процедури).
- Особливі вимоги. В даному пункті визначено будь-які особливі вимоги, необхідні для виконання цієї процедури. Вони можуть включати необхідні процедури, вимоги до спеціальних навичок і особливі вимоги до середовища.
- Етапи процедури. Містить список кроків. ISO/IEC/IEEE 29119-3 описує наступні ключові слова, які необхідно використовувати в описі етапів процедури:

- Журнал. Описуються будь-які спеціальні методи або формати для реєстрації результатів виконання тесту, помічених інцидентів та будь-які інші події, пов’язані з тестуванням.
- Налаштування. Описано послідовність дій, необхідних для підготовки до виконання процедури.
- Старт. Описано дії, необхідні для початку виконання процедури.
- Продовження. Описано будь-які дії, необхідні під час виконання процедури.
- Вимірювання. Описано, як будуть проводитись тестові вимірювання (наприклад, опишіть, яким має бути час відповіді віддаленого терміналу).
- Закриття. В цьому пункті описані дії, котрі необхідні для призупинення тестування, коли цього вимагають позапланові події.
- Перезапуск. Визначено точки перезапуску процедури та описані дії, необхідні для перезапуску процедури в кожній із цих точок.
- Стоп. В даному пункті описано дії, необхідні для впорядкованого припинення виконання тестування.
- Обернути. В цьому пункті описано дії, необхідні для відновлення тестового середовища.
- Непередбачені витрати. Описано дії, необхідні для роботи з аномальними подіями, які можуть виникнути під час виконання.

Секції розташовуються у вказаній послідовності. У разі потреби можуть бути включені додаткові розділи кінець. Якщо частина або весь вміст розділу знаходиться в іншому документі, то посилання на цей матеріал можуть бути вказані замість відповідного вмісту. До тесту обов’язково додається довідковий матеріал специфікація процедури або доступна користувачам специфікація процедури.

2.5 Звіт про передачу тестового елемента (Test Item Transmittal Report)

Звіт про передачу тестових елементів – це документ, створений для ідентифікації тестових елементів, які передаються на тестування. Він включає особу, відповідальну за кожен пункт, його фізичне місцезнаходження та його

статус. Будь-які відхилення від поточних вимог і конструкцій товару зазначаються в цей звіт. Цей документ ідентифікує елемент програмного забезпечення, який надається групі програмного забезпечення для тестування. Розробники програмного забезпечення зазвичай створюють це, і воно зазвичай визначає конкретну версію, яка «готова до тестування».

Якщо вам потрібно запустити тестування програми в певну дату, зазначену в плані тестування, вам потрібно мати можливість знайти елемент і знати його поточний статус. Все це робиться за допомогою звіту про передачу тестового елемента.

Звіт про передачу тестового елемента повинен мати таку структуру:

- Ідентифікатор передавання звіту. Вказано унікальний ідентифікатор, призначений цьому звіту про передачу тестового елемента.
- Передані предмети. В даному пункті вказані тестові елементи, що передаються, включно з їх версією/версією рівня. Надано посилання на документацію та план тестування, що стосуються переданих елементів. Вказані особи, відповідальні за передані елементи.
- Місцезнаходження. Визначено місцезнаходження переданих предметів. Визначено носій, який містить елементи, що передаються. За необхідності вказано, як маркуються чи ідентифікуються конкретні носії.
- Статус. Описано статус тестових елементів, що передаються. Включено відхилення від документації на пункт, від попередні передачі цих елементів і з плану тестування. Перелічено повідомлення про інциденти, які очікуються вирішується переданими елементами. Вказано, чи є незавершені зміни в документації елемента, що можуть вплинути на елементи, перелічені в цьому звіті про передачу.
- Дозволи. В даній частині вказано імена та посади всіх осіб, які найбільше схвалюють цю передачу. Залишено місце для підписів і дат.

Секції розташовуються у вказаній послідовності. Додаткові розділи можуть бути включені безпосередньо перед затвердженням. Якщо частина або весь вміст розділу міститься в іншому документі, то посилання на цей матеріал може бути

вказано замість відповідного вмісту. Посилальний матеріал має бути доданий до звіту про передачу тестового елемента або доступний для користувачів звіту про передачу.

Процес виконання тексту здійснюється на етапах Інтеграції та Інсталяції.

Процес виконання тесту передбачає, що будь-яке тестування програмного забезпечення проводиться незалежною групою тестувальників. Це підходить для великих проектів.

Але навіть для невеликих проектів процес тестування має бути максимально формальним і включати відповідні засоби контролю та документацію, необхідні для проведення верифікації.

Процес приймає, що на цьому етапі елементи, які потрібно перевірити, контролюються керуванням конфігурацією.

Звіт про передачу тестового елемента надсилає елементи для тестування до управління конфігурації та компанії, що проводить тестування.

Процес виконання тесту генерує інші 3 звіти, показані на малюнку вище:

- Журнал випробувань
- Підсумковий звіт про тестування
- Тестовий звіт про інцидент

Разом вони показують, що відбувається під час тестування, документують будь-які відмінності між тим, що сталося, і тим, що очікувалося, і підсумовують загальний статус тестування програмного забезпечення.

Звіт про передачу об’єктів тестування – це формальні відносини між виробником тестів і компанією, що проводить тестування, згідно з якою об’єкти готуються до тестування. Він визначає всі частини, які необхідно перевірити.

Звіт про передачу тестового елемента містить:

- елементи, які підлягають перевірці
- їх розташування
- Специфікація процедури тестування
- План тестування

Особливо важливим є статус тестового елемента, який описує відхилення від документації тестового елемента чи його плану тестування, або будь-які виправлення, які відбулися після останньої передачі. Зрештою, він документує, хто дав схвалення для передачі у функцію тестування.

2.6 Журнал тестів (Test Log)

Журнал тестування містить історичний запис подій, які відбулися під час тестового запуску або виконання за розкладом, а також статус кожної точки перевірки.

Журнал тестування встановлює вердикт для кожного запуску таким чином:

- Pass-Пройдено- вказує на те, що всі точки перевірки відповідають або отримали очікувану відповідь. Наприклад, точка перевірки коду відповіді встановлюється на PASS, коли записаний код відповіді отримано під час відтворення. Якщо ваш тест не містить точок перевірки, PASS означає, що всі первинні запити в тесті були успішними.
- Fail-Невдача вказує на те, що принаймні одна точка перевірки не відповідає очікуваній відповіді або що очікувана відповідь не була отримана.
- Error-Помилка вказує на один із таких результатів: основний запит не було успішно надіслано на сервер, від сервера не було отримано відповіді на основний запит, або відповідь на основний запит була неповною або її не вдалося проаналізувати.
- Вердикт встановлюється як «Непереконливий», лише якщо ви надаєте спеціальний код, який визначає вердикт «Непереконливий».

Вердикт згортається з дочірніх елементів на тестовий рівень. Наприклад, якщо група користувачів містить 25 віртуальних користувачів і п’ять віртуальних користувачів мають невдалий вердикт, ця група користувачів має лише один невдалий вердикт, а не п’ять.

Перегляд журналів тестування:

Щоб переглянути записи про всі події, що відбулися під час тестового запуску або запуску за розкладом, а також статус кожної точки перевірки, відкрийте журнал тестування для цього запуску. Ви також можете порівняти подію з журналу тестування із запитом або відповіддю в тесті, щоб побачити різницю між записом і відтворенням тесту.

Перегляд помилок під час виконання тестів:

Щоб переглянути помилки та інші події під час виконання тесту, скористайтеся поданням Консолі подій виконання. Якщо під час тестового запуску виникнуть проблеми, ви можете перевірити подання Execution Event Console, щоб визначити, зупинити чи продовжити тест.

Експорт журналів тестування:

Щоб обробити дані тесту продуктивності в іншій програмі або використати засоби пошуку для пошуку тексту в журналі тестування, експортуйте журнал тестування в текстовий файл.

Експорт журналу подій:

Щоб переглянути всі події, що відбулися під час виконання тесту, з іншого файлу, ви можете експортувати ці дані з панелі журналу подій у XML, CSV або текстовий файл.

Експорт даних консолі подій:

Щоб переглянути помилки та інші події тестового запуску з іншого файлу, ви можете експортувати ці дані з подання Execution Event Console у XML, CSV або текстовий файл.

Перегляд коригувань часу відгуку сторінки:

Щоб переглянути коригування часу відповіді сторінки, виміряні під час тестового запуску або запуску за розкладом, відкрийте журнал тестування для цього запуску.

Журнал випробувань повинен мати таку структуру:

- Ідентифікатор тестового журналу: вказано унікальний ідентифікатор, призначений цьому тестовому журналу.

- Опис: включено інформацію, яка стосується всіх записів у журналі, за винятком випадків, спеціально зазначених у записі журналу.

Слід враховувати таку інформацію:

- Ідентифікуйте елементи, що тестуються, включаючи їх версії/версії. Для кожного з цих пунктів, запас посилання на звіт про передачу, якщо він існує.
- Визначте атрибути середовища, в якому проводиться тестування. Включіть ідентифікацію об’єкта, обладнання, що використовується (наприклад, обсяг пам’яті, що використовується, номер моделі процесора та номер і модель стрічкових накопичувачів та/або накопичувачів), використовуване системне програмне забезпечення та доступні ресурси (наприклад, обсяг доступної пам’яті).
- Записи діяльності та подій:для кожної події, включаючи початок і кінець діяльності, записана дата й час виникнення разом із особистістю автора. Записано ідентифікатор тестової процедури, що виконується, і надано посилання на її специфікацію. Записаний весь персонал, присутній під час виконання, включаючи випробувачів, операторів і спостерігачів. Також вказано функції кожної особи.

Для кожного виконання записано візуально спостережувальні результати (наприклад, згенеровані повідомлення про помилки, переривання та
запити на дії оператора). Також записані розташування будь-якого виводу. Записане успішне чи не успішне виконання тесту.

Записані будь-які умови програмного середовища, характерні для цього запису (наприклад, заміна обладнання).

Записано, що сталося до та після неочікуваної події. Записано обставини неможливості розпочати виконання тесту процедуру або нездатність завершити процедуру тестування (наприклад, збій живлення або проблема системного програмного забезпечення).

Записано ідентифікатор кожного звіту про інцидент тестування, коли він був створений.

2.7 Звіт про тестовий інцидент (Test Incident Report)

Звіт про інцидент тестування, підготовлений на етапі тестування програмного забезпечення, є документом, який підсумовує дефекти та проблеми, виявлені в програмному забезпеченні, і допомагає підтримувати прозорість серед команди розробників і тестувальників. Мета тут полягає в тому, щоб повідомити про всі ці інциденти, які потребують належної оцінки та розслідування, і відповідати очікуваним критеріям результату, встановленим на етапі планування.

Після реєстрації інцидентів, які відбуваються на місці, або після розгортання системи, нам також потрібен певний спосіб звітування, відстеження та керування ними. Найбільш поширеним є виявлення дефектів, повідомлених проти коду або самої системи. Однак є випадки, коли повідомлення про дефекти суперечать вимогам і специфікаціям конструкції, посібникам користувача та оператора, а також тестам.

Навіщо повідомляти про інциденти?

Нижче наведено багато переваг повідомлення про інциденти:

- У деяких проектах виявляється дуже велика кількість дефектів. Навіть у невеликих проектах, де виявлено 100 або менше дефектів, дуже важко відслідковувати їх усі, якщо у вас немає процесу звітування, класифікації, призначення та керування дефектами від виявлення до остаточного усунення.
- Звіт про інцидент містить опис поміченої неналежної поведінки та класифікацію цієї неналежної поведінки.
- Як і в будь-якому письмовому спілкуванні, під час написання допомагає мати чіткі цілі. Однією з спільних цілей таких звітів є надання програмістам, менеджерам та іншим детальної інформації про спостережувану поведінку та дефект.
- Іншим є підтримка аналізу тенденцій у сукупних даних про дефекти або для того, щоб краще зрозуміти певний набір проблем чи тестів, або для розуміння та звітування про загальний рівень якості системи. Нарешті, звіти

про дефекти, якщо їх проаналізувати за проектом і навіть за проектами, надають інформацію, яка може призвести до вдосконалення процесу розробки та тестування.

- Програмістам потрібна інформація у звіті, щоб знайти та виправити дефекти. Однак перш ніж це станеться, менеджери повинні переглянути та визначити пріоритетність дефектів, щоб обмежені ресурси тестування та розробників витрачалися на виправлення та підтверджувальне тестування найважливіших дефектів.

Хоча багато з цих інцидентів будуть помилками користувача або іншою поведінкою, не пов'язаною з дефектом, певний відсоток дефектів виходить із гарантії якості та тестування.

Відсоток виявлення дефектів, який порівнює польові дефекти з дефектами тесту, є важливим показником ефективності процесу тестування.

Звіт про інцидент тестування повинен мати таку структуру:

- Ідентифікатор звіту про інцидент тесту: вказано унікальний ідентифікатор, призначений цьому звіту про інцидент тесту.
- Резюме: підведено підсумок події. Визначено задіяні тестові елементи, вказано їхню версію/рівень перегляду. Посилання на відповідну специфікацію процедури тестування, специфікацію тестового випадку та журнал тестування.
- Опис інциденту: в даному пункті описана подія. Цей опис має включати такі елементи:

1) Вхідні дані;
2) Очікувані результати;
3) Фактичні результати;
4) Аномалії;
5) Дата і час;
6) Етап процедури;
7) навколишнє середовище;
8) Намагання повторити;

9) Тестувальники;

Повинні бути відповідні дії та спостереження, які можуть допомогти виділити та усунути причину інциденту (наприклад, треба описати будь-які тестові виконання, які можуть мати відношення до цього конкретного інциденту та будь-яких відхиленнях від опублікованої процедури тестування).

Вплив: якщо відомо, треба вказати, який вплив цей інцидент матиме на плани тестування, специфікації дизайну тестування, процедуру тестування специфікації або специфікації тестового випадку.

2.8 Підсумковий звіт тестування (Test Summary Report)

Високоякісні програми працюють краще та допомагають утримувати клієнтів. Подібним чином, прозорість щодо інформації про ризики та випуск підвищує довіру до IT-організації. Це серед численних переваг, які можуть надати підсумкові звіти про тестування.

Команди контролю якості, які створюють підсумкові звіти про тестування, знайдуть документи корисними для наступного:

- створювати дані, які можна аналізувати та застосовувати;
- покращити якість програмного забезпечення;
- підвищення ефективності процесу розробки;
- визначити вплив змін на процеси розвитку;
- відстежувати дефекти між збірками випусків;
- зменшити ризик; і
- забезпечити слід історичних даних.

Підсумкові звіти про тестування вимагають часу та зусиль, але вони можуть збільшити бізнес-цінність тестування програмного забезпечення та підвищити лояльність клієнтів.

Підсумковий звіт про тестування зазвичай містить інформацію про те, де, як і коли спеціалісти з контролю якості виконують тести для конкретної збірки випуску. Крім того, звіт містить успішні чи непройдені результати для нових і

існуючих тестів. У документі повідомляється про нові дефекти, а також про те, де або в якій функціональній області програми виникли помилки.

Перевага та комерційна цінність звіту містяться в цих даних. Відстежуючи результати підсумкового звіту про тестування з плином часу, команда може визначити, чи якість програмного забезпечення залишається незмінною в кожній функціональній області – чи існують області програми, які стають дедалі крихкішими.

Якість програми не покращиться сама по собі; його потрібно виміряти. Для успішного програмного забезпечення важливо вдосконалюватися, а не занепадати з часом. Лояльність клієнтів і довіра здобуваються завдяки покращенню якості, а не підтримці постійного рівня дефектів чи постійного зниження якості.

Створення зведених звітів про тестування, а потім забезпечення прозорості щодо зібраних даних, може підвищити цінність бренду та довіру клієнтів. Існуючі клієнти або, можливо, навіть нові клієнти можуть переглядати звіти за певний період часу, щоб отримати розуміння процесу розробки програми. Підсумковий звіт про тестування дає клієнтам загальне уявлення про те, які стандарти якості використовує команда та ефективність її тестування. Можна також додатково проаналізувати дані за допомогою діаграм та іншого візуального аналізу.

Коли IT-організація ділиться підсумковими звітами про тестування з клієнтами, ці користувачі краще розуміють, де можуть бути дефекти та які області програми, як правило, генерують дефекти. Можливо, незручно визнавати, що дефекти існують, але вони є і завжди будуть. Важливо те, як організація реагує на недоліки. Чи діє він швидко й достатньо, щоб їх виправити? Чи запобігають ці виправлення повторенню? В ідеалі підсумковий звіт про тестування демонструє клієнтам, що команда контролю якості/тестування чудово справляється з такими завданнями.

Крім того, важливі цінні дані про ефективність тестового покриття. Клієнти можуть виявити прогалини та вирішити заповнити ці прогалини за допомогою власного тестування прийнятності.

Прозорість створює довіру. Розвиток бізнес-бренду шляхом створення лояльної, надійної клієнтської бази означає обмін значущими та корисними даними з клієнтами. Підсумковий звіт про тестування надає достовірні, фактичні та корисні дані.

Дані в підсумкових звітах про тестування також корисні для управління ризиками на основі фактів. Фактори ризику визначають хід розробки програми від початку до кінця життя програми. Управління ризиками є невід'ємною частиною розробки прикладного програмного забезпечення. Розгляньте можливість використання даних кожного підсумкового звіту про тестування для створення ідей щодо управління ризиками.

Команда контролю якості може використовувати ті самі дані звіту для аналізу охоплення тестуванням, що стосується ідентифікації та управління ризиками. Тестування в певних областях може потребувати додаткової глибини, щоб забезпечити охоплення тестуванням і зменшити ризик дефектів. Наприклад, скажімо, організація проводить автоматизовані щоденні тести підключень API програми. Бізнесу комфортно, що з'єднання API є надійними. Однак під час виконання тесту ви виявляєте дефекти, через які API не відповідає. Без API для надання даних і передачі інформації програма не працюватиме.

Поширене запитання: що сталося між автоматизованим тестуванням і додатковим виконанням тесту, під час якого виявлено дефект? Ознайомлення зі підсумковими звітами про тестування може дати розуміння відповідних часових пояснень або проблеми з тестовим покриттям API.

Можливо, команді потрібно додати більше глибини автоматизованому тесту. Наприклад, розширте тест від простого підтвердження з'єднання до перевірки дійсності передачі даних. Або заплануйте виконання автоматичного тесту щогодини протягом певного періоду часу, щоб визначити, чи виникає дефект під час певних навантажень або в певні моменти часу. Ця проблема може полягати в тому, що певна функція програми закінчується під час завантаження або що програма не працює за певним сценарієм. Дані про дефекти, про які повідомляється в підсумковому звіті про тестування, можуть відстежити

проблему. Коли команда вирішує проблему, вона успішно знижує ризик збою API.

Підсумковий звіт тестування повинен мати таку структуру:

- ідентифікатор підсумкового звіту про випробування: вказано унікальний ідентифікатор, призначений цьому підсумковому звіту про тестування.
- Резюме: підведено підсумок оцінювання тестових завдань. Визначено перевірені елементи, вказана їхня версія/рівень перегляду. Вказано середовище, в якому проходили тестування. Для кожного тестового елемента надано посилання на такі документи, якщо вони існують: план тестування, специфікації дизайну тестування, специфікації процедури тестування, звіти про передачу тестових елементів, журнали тестування та звіти про інциденти тестування.
- Дисперсії: Повідомте про будь-які відхилення тестових елементів від їхніх специфікацій на дизайн. Вкажіть будь-які відхилення від тесту, проекти тестів або процедури тестування. Укажіть причину кожного відхилення.
- Комплексна оцінка: оцінена комплексність процесу тестування за критеріями комплексності, зазначеними в план тестування (4.2.6), якщо план існує. Визначте функції або комбінації функцій, яких було недостатньо перевірені та пояснити причини.
- Підсумок результатів: узагальнені результати тестування. Визені всі вирішені інциденти та підсумовані їх рішення. Визначені всі невирішені інциденти.
- Оцінювання: Надана загальна оцінка кожного елемента тесту, включаючи його обмеження. Ця оцінка базується на результати тестування та критерії проходження/непроходження рівня предмета. Може бути включена оцінка ризику відмови.
- Підсумок діяльності: узагальнені основні заходи та події тестування. Узагальнені дані про споживання ресурсів, наприклад, загальна кількість персоналу, їх рівень, загальний машинний час і загальний час, що минув для кожної з основних дій тестування.

- Схвалення: в даній частині вказані імена та посади всіх осіб, які повинні затвердити цей звіт. Виділене місце для підписів і дати.

Секції розташовуються у вказаній послідовності. Додаткові розділи можуть бути включені безпосередньо перед дозволи. Якщо частина або весь вміст розділу знаходиться в іншому документі, то посилання на цей матеріал
можуть бути вказані замість відповідного вмісту. До тесту обов'язково додається довідковий матеріал зведений звіт або доступний користувачам зведений звіт.

2.9 Умови поточного робочого процесу, що призводять до відступу від міжнародних стандартів.

IT-директори звикли до швидких змін. Нові технології з'являються на ринку, існуючі розвиваються, потреби бізнесу змінюються дрібницями, персонал приходить і йде. Проте все це разом ніщо в порівнянні з турбулентністю, яку сьогодні спостерігають IT-директори.

IT-лідери зараз борються з постійними збоями, пов'язаними з пандемією, геополітичною нестабільністю та економічною нестабільністю — на додаток до цих звичайних факторів.

Ця динаміка тепер змінює порядок денний у багатьох компаніях на 2022 рік, змушуючи багатьох реорганізувати свій список головних проблем.

Хоча цифрова трансформація надає організаціям унікальні можливості для впровадження інновацій і зростання, вона також змушує критично мислити та потенційно переосмислювати аспекти, які є ключовими для компанії.

З якими проблемами зараз стикається індустрія технологій? Останні пару років були, м'яко кажучи, незвичними. Компанії будь-якого розміру стикаються з безліччю проблем, від раптового попиту на роботу вдома та віддалену роботу до різких змін у системі безпеки. Цифрова трансформація прискорилася, лише до 2025 року вона досягне 1009,8 мільярдів доларів США, і компанії в усьому світі більше інвестують у можливості для інновацій.

Наближаючись до 2023 року, важко знати напевно, що може принести майбутнє. Найкращий спосіб для компаній переконатися, що вони готові до будь-

чого, — це розглянути потенційні виклики, які можуть зустрітися на їхньому шляху.

Багато в чому індустрія технологій відчуває фантастичний набір нових можливостей. Незважаючи на те, що за останні пару років пандемія була проблематичною, вона також спричинила прискорення цифрової трансформації приблизно на сім років. Технологічні рішення зараз розвиваються швидше, ніж будь-коли раніше.

Технології стали ключем не лише до виживання під час пандемії, але й до того, щоб компанії продовжували процвітати в наступні роки. Звісно, швидкість цифрових інновацій, яку відчули 97% керівників, не всім було легко конкурувати. Як помітили багато компаній протягом останнього року, нові інновації означають нові виклики, від відсутності доступу до відповідних талантів до проблем із конфіденційністю та відповідністю.

Технології є критично важливим компонентом для підвищення безпеки компаній. На жаль, у міру розвитку інструментів захисту та підтримки бізнесу розвиваються й методи, які злочинці використовують для злому цінних сховищ даних і систем.

Після пандемії близько 26% керівників стверджують, що вони спостерігали збільшення серйозності, обсягу та масштабу кібератак. Подібним чином у міру розвитку можливостей віддаленої роботи 61% роботодавців висловили занепокоєння щодо атак, спрямованих на незахищених віддалених співробітників. Переходячи в нову епоху роботи, компаніям потрібно буде переконатися, що вони мають правильні інструменти для захисту від усього: від фішингу та атак зловмисного програмного забезпечення до стратегій програм-вимагачів.

Технології вже давно є цінним інструментом, який допомагає компаніям ефективніше досягати цілей. Однак ці рішення можуть мати безпосередній вплив на бізнес-операції лише тоді, коли для їх використання доступні відповідні кваліфіковані спеціалісти.

Близько 93% роботодавців наразі стикаються зі значною нестачею кваліфікації своїх IT-фахівців. Усі IT-лідери, починаючи з IT-директора, мають допомогти у

відстеженні талантів, необхідних для використання найновіших технологічних рішень. У той же час технологічним компаніям також потрібно буде шукати співробітників, здатних надихнути нове покоління інновацій.

Незважаючи на те, що зростання числа віддалених і гібридних співробітників може надати командам доступ до ширшого вибору талантів, потрібні правильні управлінські рішення, щоб забезпечити процвітання цих співробітників.

Як і багато можливостей у технологічному секторі, автоматизація та робототехніка є чимось на кшталт палиці з двома кінцями. З одного боку, рішення автоматизації та роботизованої автоматизації процесів (RPA) стають все більш цінними в той час, коли компаніям потрібно працювати швидше та ефективніше. З іншого боку, впровадження робототехніки може легко спричинити проблеми з підтриманням людської сторони бізнесу. Компанії все ще повинні підтримувати справжні, справжні зв’язки зі своїми клієнтами, оскільки ми переходимо в нову епоху цифровізації, особливо під час кризи.

Керівники та бізнес-лідери, безумовно, можуть отримати вигоду від впровадження робототехніки та систем автоматизації для підтримки та розширення можливостей своїх співробітників. Однак потрібна правильна стратегія, щоб забезпечити відповідний баланс між людьми та робототехнікою.

Хоча більшість компаній прагнуть залишити останні пару років позаду, варто зазначити, що пандемія висвітлила деякі важливі проблеми в тому, як управляються компаніями. Ми виявили, що робочі процеси не настільки ефективні, як могли б бути, а також вони не захищені належним чином від ризику катастрофи.

Значна проблема, яка постане перед технологічними компаніями в 2023 році, полягатиме в тому, щоб зрозуміти, як належним чином захистити бізнес від технологічних проблем, з якими компанії зіткнулися в 2020 і 2021 роках. Ймовірно, це включатиме постійне впровадження різних практик, таких як гібридна та віддалена робота. Однак це також означатиме пошук нових типів стійкості в бізнес-ландшафті.

Найважливішим є те, що в найближчі роки ми побачимо збільшення кількості покупців технологій і компаній, які зосереджуються на гнучкості, хмарі та гнучкості. Епоха «фіксованих» рішень для комунікацій, аналітики та інших послуг закінчилася.

Хоча більшість компаній протягом останніх кількох років значною мірою покладалися на технології, щоб продовжувати рух (незважаючи на пандемію), довіра до технологій низька. Дезінформація та збільшення кібератак породили широку недовіру до технологій. Люди стурбовані тим, що може принести майбутнє, і ми ретельно перевіряємо кожну покупку техніки.

Наприклад, у майбутньому, ймовірно, що штучний інтелект продовжуватиме привертати увагу лише як інструмент для підвищення кваліфікації працівників, підвищення безпеки за допомогою біометрії та вдосконалення аналізу даних. Але перш ніж компанії зможуть повною мірою використовувати це, їм спершу потрібно подолати занепокоєння з приводу того, що рішення штучного інтелекту замінять роботу людей або діють неетично.

Щоб гарантувати, що технології можуть продовжувати змінювати світ на краще, компанії повинні бути готові довести свою цінність. Тут надзвичайно корисною буде аналітика в режимі реального часу та історична аналітика, що пропонує розуміння потужності кожної інновації. Чим більше доказів можуть запропонувати компанії, тим краще.

Згідно зі звітом The Standish Group, 31,1% проектів скасовується до завершення, а 52,7% проектів перевищують бюджет майже вдвічі більше, ніж було заплановано спочатку. Ці прості статистичні дані про проблеми розробки програмного забезпечення досить хороші, щоб відлякати людей від проекту розробки програмного забезпечення на замовлення їхньої мрії. Але розумні компанії тримаються, проводять свої дослідження, обходять проблеми та все одно отримують приємні переваги.

Добре відомо, що розробка програмного забезпечення на замовлення означає конкурентну перевагу, швидший час виходу на ринок, підвищення ефективності та зниження вартості, але лише тоді, коли її дуже ретельно сплановано та

виконано дуже досвідченою та професійною командою. Тому розглянемо найпоширеніші проблеми у розробці та тестуванні програмного забезпечення та виділимо обхідні шляхи, які допоможуть продовжувати проект.

Кожен будинок має фундамент, чи не так? Збір вимог є основою проекту програмного забезпечення. Нечіткі вимоги означають слабку основу і подальший слабший продукт.

Нечіткі вимоги також означають неправильну оцінку часу та вартості. Оцінка часу та вартості проекту розробки програмного забезпечення включає багато змінних, які сильно відрізняються від проекту до проекту на основі складності, кількості функцій і витонченості базових технологій. Отже, неадекватний збір вимог до програмного забезпечення дорівнює нереалістичним оцінкам часу та витрат.

Ще одна проблема – вимоги до програмного забезпечення, які постійно змінюються. Отже, у перший день ви попросили автомобіль із певним об’ємом двигуна, типом палива, потужністю, крутним моментом тощо. Наступного дня ви попросите збільшити крутний момент. Третій день, ви говорите, мені потрібні авторозсувні двері. Подивіться, що тут відбувається – якщо ви продовжуєте додавати вимоги, ваш обсяг змінюється, а також розвиток. Це призведе до значного збільшення вартості, а також часу розробки.

Рішення: додавання кожної окремої функції до продукту може здатися гарною ідеєю на перший погляд, але треба обдумати цю частину ще раз.

Потрібно документувати свої потреби. Навіть якщо здається, що це не матиме великого значення, обов’язково треба нотувати функції, функціональні можливості та повний процес користувача, який у спеціаліста в голові.

Під час обговорення цих вимог із компанією з розробки програмного забезпечення на замовлення треба переконатися, що вони ставлять запитання та перехресні запитання, щоб повністю зрозуміти потік програмного забезпечення для користувача.

Важливо переконатися, що постачальник програмного забезпечення розробив документ бізнес-вимог (BRD). Документ повинен містити план проекту з чіткими

цілями, який має виступати дорожньою картою як для розробників, так і для клієнта.

Зі сторони клієнта, щоб переконатися, що обсяг не змінюється вічно, почніть з дослідження ринку, щоб зрозуміти, які функції є в тренді та відповідатимуть меті програмного забезпечення.

Крім того, розбивка проекту на етапи допоможе контролювати оцінки та встановити чіткі очікування щодо того, коли компанія та що хоче отримати.

Погана комунікація може сповільнити проект до точки, коли він перестане бути актуальним або цінним. Проблеми зі зв'язком можуть бути мовними або не отримувати регулярні оновлення. Неправильне спілкування може призвести до пропуску термінів і запитів на функції, а також може спричинити додаткові витрати та час або неякісні продукти, які не відповідають очікуванням зацікавлених сторін.

Корпоративне програмне забезпечення є складним за своєю суттю. Нові технології можуть лякати. Це великий виклик для організацій, які переживають цифрову трансформацію – як з точки зору впровадження та інтеграції даних, так і з точки зору взаємодії з кінцевим користувачем. Керівники повинні враховувати це на ранніх етапах проекту трансформації та шукати найбільш інтуїтивно зрозумілі інтегровані системи.

Нові процеси та технології часто викликають проблеми у вигляді опору змінам з боку штатних співробітників, які вважають, що в тому, як вони зараз працюють, немає нічого поганого. Для впроваджень нового програмного забезпечення організації повинні забезпечувати всебічний навчальний курс, а також безперервну підтримку продуктивності співробітників, щоб допомогти співробітникам швидко стати продуктивними та досвідченими з інструментом, дозволяючи їм зрозуміти цінність цих нових процесів.

3 РОЗРОБКА ПІДХОДІВ ДО УДОСКОНАЛЕННЯ, СТВОРЕННЯ ТА ВЕДЕННЯ ТЕСТОВОЇ ДОКУМЕНТАЦІЇ В ПРОЦЕСІ ТЕСТУВАННЯ.

3.1 Основні причини збоїв запуску програмного забезпечення

Було проведено багато досліджень, щоб визначити основні причини збоїв запуску програмного забезпечення. Однією з головних причин таких збоїв виявилася низька якість процесу розробки програмного забезпечення. Головна мета виконання суворих випробувань на забезпечення якості — запобігти випуску продукції низької якості. Дрібні помилки, які проскочать, можуть призвести до великих фінансових втрат.

Шлях до створення високоякісного програмного забезпечення полягає у впровадженні ефективного менеджменту якості, який надає інструменти та методології для створення продуктів без помилок. Управління якістю програмного забезпечення — це загальний термін, що охоплює три основні аспекти: забезпечення якості, контроль якості та тестування.

Тестування — це основна діяльність, спрямована на виявлення та вирішення технічних проблем у вихідному коді програмного забезпечення та оцінку загальної зручності використання, продуктивності, безпеки та сумісності продукту. Це не лише основна частина забезпечення якості; це також невід'ємна частина процесу розробки програмного забезпечення.

Процеси тестування повинні бути добре сплановані, визначені та задокументовані. Хороша документація є інструментом, який створює ефективне спілкування в команді програмного забезпечення. Отже, ефективне планування передбачає створення планів якості та тестування проекту. Давайте розглянемо основні типи документації, яка підтримує процес контролю якості (Табл. 1).

Таблиця 1

Документи планування забезпечення якості

Ієрархія	Цілі	Політика перегляду	Ключові елементи	Користувачі
Політика тестування				
Компанія	Відобразити загальне ставлення та підхід компанії до тестування	Дуже рідко переглядається	✓ Значення тестування для компанії ✓ Тестові об’єкти ✓ Стандарти та критерії тестування ✓ Інструменти тестування ✓ Вимірювання успіху ✓ Шляхи вдосконалення	Стейкхолдери, QA спеціалісти, розробники
План управління якістю				
Проект	Визначити результати проекту, стандарти якості та ролі	Рідко переглядається	✓ Об’єкти якості ✓ Ключові результати ✓ Ролі та обов’язки ✓ Стандарти якості ✓ Інструменти	Стейкхолдери, QA спеціалісти, розробники
Стратегія тестування				
Продукт	Визначити підходи для тестування ПЗ та узгодити їх з QA та бізнес-цілями	Рідко переглядається	✓ Обсяг тестування ✓ Бюджетні обмеження ✓ Часові обмеження ✓ Галузеві стандарти ✓ Об’єкти тестування	Стейкхолдери, QA спеціалісти, розробники
Тест план				
Реліз/спринт	Охоплення обсягу тестування та заходів для версії продукту, набору функцій тощо	Регулярно переглядається	✓ Тестові завдання ✓ Підхід до тестування ✓ Критерії проходження та непроходження ✓ Функції для перевірки ✓ Результати ✓ Розклад ✓ Ризики ✓ Обов’язки	QA спеціалісти, розробники
Тестовий випадок				
Функція	Визначити умови тестування, щоб перевірити очікувану роботу функції	Постійно переглядається	✓ Ідентифікатор тестового випадку ✓ Опис ✓ Етапи тестування ✓ Тестові дані ✓ Очікуваний результат ✓ Фактичний результат ✓ Статус ✓ Дата створення ✓ Дата виконання	QA спеціалісти, розробники

Політика тестування (Test Policy)

Політика тестування - це документ найвищого рівня, який створюється на рівні організації. Він визначає принципи тестування, прийняті компанією, і основні цілі

тестування компанії. Тут також пояснюється, як проводитиметься тестування та як компанія вимірює ефективність і успішність тестування.

Немає стандартного підходу до створення тестової політики, але зазвичай він включає наступне:

- Визначення того, що означає тестування для компанії,
- Тестові цілі організації,
- Загальні стандарти та критерії тестування програмного забезпечення в проектах,
- Визначення термінів тестування для уточнення їх подальшого використання в інших документах,
- Перелік інструментів для підтримки процесу тестування,
- Методи та показники оцінки ефективності тестування, а також
- Шляхи вдосконалення процесів тестування.

План управління якістю

План управління якістю — це документ, який визначає прийнятний рівень якості продукції та описує, як проект досягне цього рівня. Це не обов’язковий документ, але він допоможе вам спланувати всі завдання, необхідні для того, щоб переконатися, що проект відповідає потребам і очікуванням вашого клієнта. Основною метою цього плану є підтримка керівників проектів і допомога в організації процесу шляхом визначення ролей, обов’язків і стандартів якості, яких необхідно досягти. Відповідно, він повинен містити вимоги до якості програмного забезпечення та описувати, як їх слід оцінювати.

Ключові компоненти плану управління якістю:

- Цілі якості
- Ключові результати та процеси проекту перевіряються на задовільний рівень якості
- Стандарти якості
- Діяльність з контролю та забезпечення якості
- Якісні ролі та обов'язки
- Якісні інструменти

- План звітування про проблеми з контролем і забезпеченням якості

Тест стратегії

Стратегія тестування – це більш конкретний документ на рівні продукту, який випливає з документу специфікації бізнес-вимог. Зазвичай керівник проекту або бізнес-аналітик створює стратегію тестування, щоб визначити підходи до тестування програмного забезпечення, які використовуються для досягнення цілей тестування. Стратегія тестування ґрунтується на бізнес-вимогах проекту, тому вона узгоджується з обов’язками керівника проекту.

Основними компонентами тестової стратегії є:

- Обсяг тестування
- Цілі тесту
- Бюджетні обмеження
- Зв'язок і звітність про стан
- Галузеві стандарти
- Тестування вимірювань і показників
- Звіт про дефекти та відстеження
- Управління конфігурацією
- Терміни
- Графік виконання тесту
- Ідентифікація ризиків

У невеликому проекті стратегія тестування є частиною плану тестування. Але для більшого проекту керівник проекту повинен створити стратегію тестування як окремий статичний документ, на основі якого кожен план тестування можна розробляти далі.

Хороший документ про стратегію тестування відповідає на такі запитання:

- Що таке продукт?
- Яку його частину (частини) слід перевірити?
- Як їх перевіряти?
- Коли має розпочатися тестування?
- Які критерії початку/завершення?

План тестування

План тестування — це документ, який описує, що тестувати, коли тестувати, як тестувати та хто виконуватиме тести. Він також описує обсяг і заходи тестування. План тестування містить цілі тестів, які необхідно провести, і допомагає контролювати ризики. Рекомендується мати план тестування, написаний досвідченою особою, як-от провідний спеціаліст з контролю якості або менеджер.

Хороший план тестування має містити розклад усіх необхідних заходів тестування, щоб контролювати час тестування вашої команди. Він також має визначити ролі кожного члена команди, щоб кожен чітко розумів, що потрібно. Не існує універсального способу створення плану тестування, оскільки він залежить від процесів, стандартів і інструментів керування тестуванням, які впроваджені в компанії. Відповідно до стандарту IEEE 829 документ плану тестування повинен містити таку інформацію:

- Ідентифікатор плану тестування
- Вступ
- Список літератури (список супутніх документів)
- Тестові елементи (продукт і його версії)
- Проблеми програмного забезпечення з ризиком
- Функції для перевірки
- Функції, які не підлягають тестуванню
- Підхід (стратегія)
- Критерії проходження або непроходження предмета
- Критерії призупинення
- Результати (документ плану тестування, тестові випадки, інструменти, журнали помилок, звіти про проблеми тощо)
- Тестове середовище (апаратне забезпечення, програмне забезпечення, інструменти)
- Розклад
- Потреби в кадрах і навчанні
- Обов'язки

- Ризики
- Дозволи

Ось кілька ключових рекомендацій для підвищення ефективності плану тестування:

- Зробіть свій план тестування коротким. Уникайте повторень або недоречності. Він повинен містити лише відповідну інформацію.
- Бути специфічним. Включіть усі деталі, напр. редакції та версії програм, щоб зробити документ доступним для пошуку.
- Оновіть план тестування. Це живий документ, який потрібно часто оновлювати на вимогу.
- Поділіться планом тестування зі своїми зацікавленими сторонами. Це дасть їм інформацію про ваші процеси тестування. Якість вашого плану тестування відображатиме якість тестування, яке виконуватиме ваша команда.

Тестові випадки

Підготовка ефективних тестів є невід'ємною частиною вдосконалення тестування програмного забезпечення. Згідно з визначенням, наданим ISTQB (Міжнародна кваліфікаційна рада з тестування програмного забезпечення, світовий лідер із сертифікації компетенцій у тестуванні програмного забезпечення), «Тестовий приклад — це набір вхідних значень, передумов виконання, очікуваних результатів і постумов виконання, розроблених для конкретної цілі чи умови тестування, наприклад, для виконання конкретного програмного шляху або для перевірки відповідності певній вимозі». Це один із ключових інструментів, який використовують тестувальники. Стандартний тест включає таку інформацію:

- Ідентифікатор тестового випадку
- Опис тестового випадку
- передумови
- Тестові кроки
- Тестові дані

- Очікуваний результат
- Реальний результат
- Статус
- Створений
- Дата створення
- Виконав
- Дата виконання

Щоб написати ефективні тестові приклади, використовуйте такі методи:

- Визначте перевірені вимоги. Визначте обсяг і мету тестування перед початком процесу тестування.
- Вимога замовника. Спеціаліст, який пише тестовий приклад, повинен добре розуміти особливості та вимоги користувача. Кожен тестовий приклад має бути написаний з урахуванням вимог клієнта.
- Пишіть вчасно. Найкращий час для написання тестових випадків – ранні етапи аналізу вимог і проектування. Таким чином спеціалісти з контролю якості можуть зрозуміти, чи всі вимоги можна перевірити чи ні.
- Просто і зрозуміло. Тестові приклади мають бути простими та зрозумілими. Кожен тестовий приклад має містити лише необхідні та відповідні кроки. Незалежно від того, скільки разів і ким він буде використовуватися, тестовий приклад повинен мати один очікуваний результат, а не кілька очікуваних результатів.
- Унікальні тестові випадки. Кожен тестовий приклад повинен мати унікальну назву. Це допоможе класифікувати, відстежувати та переглядати тестові випадки на наступних етапах.
- Тестові випадки повинні підтримуватися. Якщо вимоги змінюються, тестер повинен мати можливість налаштувати тестовий приклад.

Отже, перш ніж приступити до побудови процесу тестування, необхідно визначити функціональність системи, яка тестуватиметься.

3.2 Підготовка тестової документації для різних рівнів тестування

У більшості випадків для не надто масштабних проектів існує висока ймовірність тестування всього, що буде розроблено, але у більших проектах ми можемо зіткнутися з обмеженнями: певна функціональність може перевірятися іншою командою або сторонньою компанією, також у деяких випадках ми можемо використовувати тимчасові сервіси-заглушки, які не будуть використовуватися в проді, отже, не буде необхідності тестувати їх, для таких випадків ми, ймовірно, повинні будемо перевірити тільки інтеграцію, також деякі модулі можуть бути протестовані виключно на запит клієнта. На цьому етапі важливо домовитися про те, чи слід готувати тестову документацію для всіх рівнів тестування або лише для певних.

Перейдемо до тестової документації.

Два найбільш поширені варіанти – чек-листи та тест-кейси. Потрібно визначити який із них підходить для проекту в залежності від його складності, тривалості, стабільності та інших факторів. Якщо з якихось причин використання тест-кейсів чи чек-листів не є оптимальним, то можна розробити кастомні варіанти тестової документації. Також варто зазначити, що можна комбінувати кілька варіантів, у цьому випадку необхідно визначити умови, за яких буде використовуватися той чи інший тип документів.

Також спеціалістам з тестування програмного забезпечення буде корисним використовувати в роботі такий вид документа як Чит-лист (Cheat-sheet). Чит-листи складаються з їх подальшого багаторазового використання. У зв'язку з цим такі списки створюються щодо поширених складових програмного забезпечення, що часто зустрічаються, з якими належить працювати неодноразово не тільки на поточному проекті, але і на наступних.

Кожен окремий фахівець може розширювати перелік перевірок у своїх чит-листах з досвідом. На сьогоднішній день багато автоматизованих систем управління тестами також містять функціонал підтримки списків чит-листів, що

значною мірою спрощує їх створення, подальшу підтримку та практичне застосування.

Написання чит-листів доцільно з метою:

- Збереження часових ресурсів при складанні тестів для подібного функціоналу і наступних проектах.
- Стандартизація типів перевірок у компанії.
- Документування власних ідей покращення тестів.
- Майбутнього взаємообміну досвідом з іншими колегами.

Нижче наведено приклад чит-листу для тестування нового функціоналу (Табл. 2)

Таблиця 2 Чит-лист для тестування ПЗ

Етап	**Дії**
Приймальне тестування	Перевірка, чи працює функція чи ні
Функціональні аспекти	Треба взяти кожен елемент програми та порівняти його з письмовою функціональною специфікацією – це називається перевіркою
Поля	Необхідно перевірити на : - початковий стан кнопки - умови доступності поля на ВКЛ або ВИКЛ - мін або макс - прийняті символи - вимоги - бекенд-валідація проти фронтенд-валідації - чи зараховується пусте місце? - поведінка з неприйнятними символами - перевірити як і чи працює копіювання
Тестування кнопки	- Перевірити умови увімкнення або вимкнення доступності поля, негативного та позитивного тестування, а також обробки помилок - Заповнити недійсні дані в одному або кількох полях і натиснути кнопку, щоб побачити, як відреагує програма
Тестування клавіатури	- Подивитись на початковий стан клавіатури; перевірити наявність кнопки повернення або завершення, як очікувалося - Функція інтелектуального введення тексту має працювати належним чином, за винятком, наприклад, полів облікових даних - Ввести нестандартні символи, вибравши деякі з каталогу UTF-8. Також спробувати побачити, як програма поводиться

	з цікавими символами
Перевірка поведінки дублікатів даних, видалення поведінки даних, поведінки даних із вичерпаним терміном дії та орфографічних помилок	Повинні бути ретельно перевірені
Тестування навігації	Навігація вперед і назад. - Перевірити, чи дані, якими ви заповнили поля, залишаються збереженими під час перегляду, чи вам потрібно їх заповнювати повторно - Якщо в цій функції є посилання, перевірте, чи вони релевантні
Інтеграція пристрою	Як функція проводиться із зовнішнім фактором. - Перевірити, як ця функція діє, коли ви отримуєте дзвінок або текстове повідомлення, раптово відключається інтернет або користувач від'єднується через третю сторону. - Змінити орієнтацію екрана та перевірити функцію. - Видалити програму, перевстановити її та перевірити її поведінку. Крім того, спробувати встановити його разом із наявною старішою версією.
Кросплатформне тестування	Протестувати функцію на прийнятих версіях iOS і Android. - Перевірити найпоширеніші розміри екрана. Вони можуть змінити інтерфейс користувача помилковим чином. - Перевірити функцію на пристроях із низькою продуктивністю. - Придумати кулька стрес-тестів і подивитись, як поводиться програма, коли ви завантажуєте купу програм і зменшуєте доступну пам’ять пристрою та потужність ЦП.
Перевірити, чи функція працює та відображається належним чином усіма затвердженими мовами програми.	- Перевірити будь-які інші регіональні налаштування – перевірити, чи дотримується їх програма. - Варто також протестувати параметри доступності – подивіться, як поводиться програма.
Тестування безпеки.	Перевірити наявність вразливостей у програмі та можливий витік інформації в інші місця призначення.
Тестування синхронізації	Перевірити, коли локальні дані синхронізуються із сервером. - Використовувати інструмент регулювання мережі, щоб перевірити функціональність на різних смугах пропускання. Під час тестування функції синхронізації перевірте, як функція спілкується з хмарою. - Спробуйте виконати тестову синхронізацію під час відключення інтернету до, посередині та після обробки підключення до хмари. - Використовуючи той самий обліковий запис, спробувати

	внести зміни на кількох пристроях.
Тестування дизайну	Чи реалізований інтерфейс користувача такий самий, як той, який надіслали дизайнери?
Тестування інтеграції компонентів	Перевірити, як функція інтегрується з існуючими однорідними функціями; наприклад, як вхід або SLA впливають на функцію. - Перевірити за допомогою дійсних і недійсних даних.
Тестування продуктивності, тестування навантаження та рівні стрес-тестування	- Використовувати сценарії, щоб досягти ліміту специфікації, і уважно стежити за програмою, коли вона наближається до ліміту, на ліміті та перевищує існуючі ліміти.

Визначивши види та глибину тестування, а також тип тестової документації, переходимо до організації модифікації та зберігання. Потрібно підготувати місце для зберігання. Це може бути один із таких варіантів:

- Платна або безкоштовна система керування тестуванням (TMS)
- Віддалений репозиторій
- Розшарована мережева папка
- Розшарена колекція тестів у інструментах тестування (наприклад Postman)
- Загальнодоступний Google doc
- Інші інструменти.

Місце зберігання має бути обране на основі наявних ресурсів та потреб, важливо враховувати, що кожна зацікавлена сторона повинна мати доступ до тестової документації, але водночас ми маємо бути впевнені, що безпека забезпечена обмеженням доступу для третіх осіб.

Далі слід визначити необхідність та частоту бекапів документації, хто і коли має займатися цією діяльністю. Якщо не визначити відповідальну особу, терміни та тригери для резервного копіювання, то певний момент команда може залишитися без тестової документації, наприклад, у разі збою сервісу зберігання або проблем із втратою даних.

Підготовчі роботи виконано. Тепер настав час організувати процес розробки тестової документації. Для цього гарною практикою буде надання всім учасникам команди тестування шаблону документа. Під час підготовки шаблону пам'ятайте наступне:

- Зафіксуйте призначення документа (можна вказати назву, короткий опис, цільову аудиторію тощо)
- Зафіксуйте всі необхідні позначення у легенді документа (наприклад, статуси виконання тесту, назви модулів тощо)
- Зафіксуйте посилання специфікацію функціональності, на яку розробляється тестовий документ
- Визначити обов’язкові та необов’язкові атрибути
- Визначити причини заповнення необов’язкових атрибутів
- Визначити перелік статусів проходження тестів
- У разі потреби визначте список пріоритетів, а також список можливих значень інших атрибутів.

Шаблон повинен розроблятися з урахуванням специфіки кожного конкретного проекту, а також може бути розширений, скорочений або модифікований для певної функціональності, для тестування якої краще використовувати інші атрибути.

Тепер, коли команда тестування розпочинає розробку тестової документації, не треба забувати звертати увагу на наступні аспекти:

- Грамотність формулювань: назви елементів інтерфейсу користувача, дій системи та користувача повинні відповідати назвам у специфікації; необхідно стежити за стислою, однозначністю, точністю формулювань;
- Декомпозиція тестів: тести мають бути структурованими та послідовними;
- Простота сприйняття та можливість швидко знайти потрібний тест: слід використовувати інструменти для простої візуалізації, розбиття на компоненти та дії, чітку структуру, фільтрацію, пошук, угрупування, форматування тощо;
- Підготовка тестових даних;
- Уникнення копіювання: шляхом осмислення та переробки інформації із специфікації уникаємо копіювання вимог до тестових документів, а копіювання однакових тестів, що використовуються в різних модулях

системи, уникаємо шляхом винесення тестів, що повторюються, в окремі групи;

- Дотримання структури документа: всі необхідні атрибути повинні бути присутніми та заповнені;
- Підтримка тестової документації в актуальному стані.

Щоб команда тестування наслідувала найкращі практики, вона як мінімум повинна про них знати. Можна скласти документ, що описує вимоги до тестової документації, розмістити його у загальному доступі та по приходу новачка в команду, ознайомлювати його із загальноприйнятими у компанії практиками. Так з’явиться можливість створити однаковий стиль ведення тестової документації та зробити команду взаємозамінною.

На закінчення підіб’ємо підсумок, що необхідно зробити для ефективної організації роботи з тестовою документацією. Насамперед визначте функціональність, яку необхідно тестувати. Зафіксуйте види тестування, що використовуються і не використовуються, а також глибину тестування. Визначте тип документації та підготуйте шаблон. Організуйте зберігання тестів і не забувайте про резервне копіювання та підтримку актуальності документації. Побудуйте роботу так, щоб команда мала можливість дотримуватись найкращих практик. Дотримання цих нескладних принципів дозволить уникнути більшості труднощів з процесами.

Висновки

В результаті магістерської роботи, задачі, що були поставлені, були вирішені в повному обсязі із досягненням саме мети дослідження роботи: за допомогою адаптованої та обновленої документації, покращити робочий процес тестування, що в результаті призведе до підвищення якості програмного забезпечення.

У результаті проведених досліджень можна зробити висновок про те, що чим більший проект, тим детальніша документація потрібна. Використання лише контрольного списку, безсумнівно, призведе до помилкових уявлень про мету та обсяг проекту. Наявність якісної тестової документації має вирішальне значення, і вона повинна бути зрозумілою всім членам команди, включаючи тестувальників, розробників і керівників проектів. Зрештою, уся тестова документація має на увазі ту саму мету: задокументувати те, що було зроблено, що не було зроблено, і навіть чому щось не було зроблено. Якщо це не зафіксувати в документації, це призведе до невідповідностей і помилок у процесі. Інженери з контролю якості повинні тестувати кожну нову функціональність.

Для написання магістерської роботи була опрацьований та проаналізований міжнародний стандарт ведення та створення тестової документації, а саме ISO/IEC/IEEE 29119. Деально були розглянуті всі документи, описані у стандарті.

Проведено аналіз сучасного IT ринку, а саме проблеми, з якими найчастіше зустрічаються спеціалісти з тестування програмного забезпечення.

На підставі вивчених матеріалів, аналізу ринкових можливостей та процесу розробки програмного забезпечення було запропоновано введення додаткових документів, котрі більш детально описують етапи та елементи тестування, а саме чек-лист та чит-лист.

Запропоновані документи є більш гнучкими та можуть вміщувати досить вичерпний список дій, котрі відповідають потребам програми. Вони можуть коригуватися в залежності від програмного забезпечення, тому що документи повинні мати динамічний характер та регулярно змінюватись.

На підставі даної інформації було зроблено висновок, що даний підхід написання тестової документації може бути корисним для компанії з розробки програмного забезпечення, тому що використання його у процесі тестування зможе налагодити робочий процес та значно підвищити якість продукту.

ПЕРЕЛІК ПОСИЛАНЬ

1. ISO/IEC/IEEE 29119-3:2021. Software and systems engineering. Software testing. Part 3: Test documentation. ISO/IEC/IEEE, 2021, 84 p.
2. Software Testing Quality Standards. URL:

https://www.gcreddy.com/2021/07/software-testing-quality-standards.html (дата звернення 01.11.2022).

3. Why is Documentation Important in Software Testing? URL:

https://blog.e-zest.com/why-is-documentation-important-in-software-testing/ (дата звернення 01.11.2022).

4. Test Planning: A Detailed Guide. URL:

https://www.browserstack.com/guide/test-planning (дата звернення 01.11.2022).

5. How to test software requirements specification (SRS)? URL:

https://blog.e-zest.com/how-to-test-software-requirements-specification-srs/ (дата звернення 01.11.2022).

6. Test Design Specification. URL:

https://www.professionalqa.com/test-design-specification (дата звернення 01.11.2022).

7. What are incident reports in software testing? URL:

https://tryqa.com/what-are-incident-reports-in-software-testing/ (дата звернення 07.11.2022).

8. Test log overview. URL:

https://www.ibm.com/docs/en/rpt/9.1.0?topic=SSMMM5_9.1.0/com.ibm.rational.test.lt.common.doc/topics/ttestlogoverview.htm (дата звернення 11.11.2022)

9. 9 Critical Digital Transformation Challenges to Overcome (2023). URL:

https://whatfix.com/blog/digital-transformation-challenges/ (дата звернення 11.11.2022).

10. 11 Ways to Improve Software Testing through Planning, Work Environment, Automated Testing, and Reporting. URL:

https://www.altexsoft.com/blog/engineering/software-testing-qa-best-practices/ (дата звернення 24.11.2022).

11. Must Haves in Test Documentation. URL:

https://blog.testproject.io/2022/01/18/must-haves-in-test-documentation/ (дата звернення 25.11.2022).

12. The 10 biggest issues IT faces today. URL:

https://www.cio.com/article/228199/the-12-biggest-issues-it-faces-today.html (дата звернення 29.11.2022).

13. Test Planning Cheat Sheet. URL: https://agiletester.ca/test-planning-cheat-sheet/

(дата звернення 04.12.2022).

14. Testing Cheat Sheet by Maxxtro. URL:

https://cheatography.com/maxxtro/cheat-sheets/testing/ (дата звернення 13.12.2022).

15. III Науково-практична конференція «Проблеми комп'ютерної інженерії». Тестування у програмному забезпеченні. Лоранець Р.В. – ДУТ, 2022 – укр -