ТЕМА 9. ПЛАН ТЕСТУВАННЯ. ТЕСТ-КЕЙС. ЧЕК-ЛИСТ. БАГ РЕПОРТ

Артефакти тестування

У відповідності до процесів та методологій розробки інформаційних систем, під час проведення тестування створюється і використовується певна кількість тестових артефактів (документи, моделі і т.д.). Найбільш поширеними тестовими артефактами є:

– План тестування (TestPlan) - це документ, який описує весь обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розкладу, критеріїв початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

– Набір тест кейсів і тестів (TestCase&Testsuite) - це послідовність дій, по якій можна перевірити чи відповідає тестована функція встановленим вимогам.

– Чек-лист (Checklist - контрольний список) - список, що містить ряд необхідних перевірок.

– Дефекти /Баг репорти (BugReports / Defects) - це документи, що описують ситуацію або послідовність дій, яка призвела до некоректної роботи об'єкта тестування, із зазначенням причин і очікуваного результату.

1. План тестування

Кожна методологія або процес намагаються нав'язати свої формати оформлення планів тестування. Хороший тест план повинен як мінімум описувати наступне:

Що треба тестувати?

– опис об'єкта тестування: системи, додатки, обладнання.

Що буде тестуватися?

– список функцій і опис тестової системи і її компонент окремо.

Як буде проводитися тестування?

– стратегія тестування, а саме: види тестування і їх застосування по відношенню до об'єкта тестування .

Коли буде проводитися тестування?

– послідовність проведення робіт: підготовка (TestPreparation), тестування (Testing), аналіз результатів (TestResultAnalisys) в розрізі запланованих фаз розробки.

Критерії початку тестування:

– готовність тестової платформи (тестового стенда);

– закінченість розробки необхідного функціоналу;

– наявність всієї необхідної документації;

– тощо.

Критерії закінчення тестування:

– результати тестування задовольняють критеріям якості продукту;

– вимоги до кількості відкритих багів виконані;

– витримка певного періоду без зміни вихідного коду програми CodeFreeze (CF);

– витримка певного періоду без відкриття нових багів ZeroBugBounce (ZBB)

– інші критерії.

Оточення тестованої системи (опис програмно-апаратних засобів).

Необхідні для тестування обладнання та програмні засоби (тестовий стенд і його конфігурація, програми для автоматизованого тестування і т.д.).

Ризики та шляхи їх вирішення.

Види тест планів.

Найчастіше на практиці доводиться стикатися з такими видами тест планів:

– Майстер Тест План (MasterPlanor MasterTestPlan);

– Тест План (TestPlan), детальний тест план;

– План приймальних випробувань (ProductAcceptancePlan) - документ, що описує набір дій, пов'язаних з приймальним тестуванням (стратегія, дата проведення, відповідальні працівники і т.д.).

Явна відмінність Майстер Тест Плану від просто Тест Плану в тому, що Майстер Тест План є більш статичним в силу того, що містить у собі високорівневу (HighLevel) інформацію, яка не схильна до частої зміни в процесі тестування і перегляду вимог. Сам же детальний тест план, який містить більш конкретну інформацію по стратегії, видам тестуванні, розкладом виконання робіт, є "живим" документом, який постійно зазнає змін, що відображають реальний стан справ на проекті. У повсякденному житті на проекті може бути один Майстер Тест План і кілька детальних тест планів, що описують окремі модулі однієї програми. Додатки 1 та 2 містять приклади шаблонів Тест Планів.

2. Тест-кейс

Тестовий випадок (TestCase) - це артефакт, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестової функції або її частини.

Безпосередньо сам тестовий випадок складається з 3 частин:

– PreConditions (Передумови) - або список кроків, які призводять перевіряти систему в стан, придатний для тестування, або список перевірок умов того, що система вже знаходитися в необхідному стані;

– TestCaseDescription (Опис тестового випадку) - список дій, за допомогою яких здійснюється основна перевірка функціоналу (після якої і звіряється фактичний результат з очікуваним);

– PostConditions (Після умови) - список дій, які повертають систему в початковий стан.

Під описом тест кейсу розуміється структура виду:

Action>ExpectedResult>TestResult

Результати Тест-кейсу:

– позитивний результат, якщо фактичний результат дорівнює очікуваному результату;

– негативний результат, якщо фактичний результат не дорівнює очікуваному результату. В цьому випадку, знайдена помилка;

– виконання тесту блоковано, якщо після одного з кроків продовження тесту неможливо. У цьому випадку так само, знайдена помилка.

Тест- кейсом перевіряється **одна конкретна річ**, і для цієї речі повинен бути тільки **один очікуваний результат**.

Деталізація Тест Кейсів (TestCaseSpecification) - це рівень деталізації опису тестових кроків і необхідного результату, при якому забезпечується розумне співвідношення часу проходження до тестового покриття.

Приклад тестового випадку:

Action	Expected Result	Test Result (passed/failed/blocked)
PreConditions		
do A1	verify B1	passed
do A2	verify B2	**failed**
Test Case Description:		
do A3	verify B3	**blocked**
PostConditions		

Спосіб опису тест кейсів і їх структура може в кожній компанії або команді бути різним: мати різні глибини опису необхідних дій і результатів, мати різні структурні складові. Але, хороша структурованість і висока зручність шаблонів тестових випадків, може досить скоротити час рутинних заповнень форм і підвищити ефективність команди в цілому.

Чого не повинно бути в Тест-кейсі:

– залежності від інших тест- кейсів;

– нечітких формулювань кроків або очікуваного результату;

– відсутності необхідної для проходження тест- кейса інформації;

– зайвої деталізації.

Тест-кейси повинні не повторювати вимоги, а перевіряти їх! Приклад невдалого тест-кейсу наведено на рисунку 1.

Значення в поле «Сума» має розраховуватися як сума значень з полів «А» і «В»

Ледачій тестувальник

- **Дії:**
  - Ввести значення в поля «А» і «В»
- **Очікуваний результат:**
  - Значення в поле «Сума» має розраховуватися як сума значень з полів «А» і «В»

або

- Див. в технічному завданні
- Відповідно до ТЗ, спостерігаємо очікувані значення
- і т.п.

Рисунок 1 – Невдалий тест-кейс

Приклад тест-кейсу, який відповідає необхідним вимогам наведено на рисунку 2.

Значення в поле «Сума» має розраховуватися як сума значень з полів «А» і «В»

Неледачий тестувальник

- **Дії:**
  1. В поле «А» ввести значення 2
  2. В поле «В» ввести значення 3
  3. Натиснути на кнопку «Розрахувати»
- **Очікуваний результат:**
  В поле «Сума» відобразилось значення 5

Рисунок 2 – Приклад коректного тест-кейсу

Тестовий набір (TestSuit) - набір тест кейсів, які об'єднані тим, що відносяться до одного модулю, функціональності, пріоритету або одного типу тестування (рис.3).

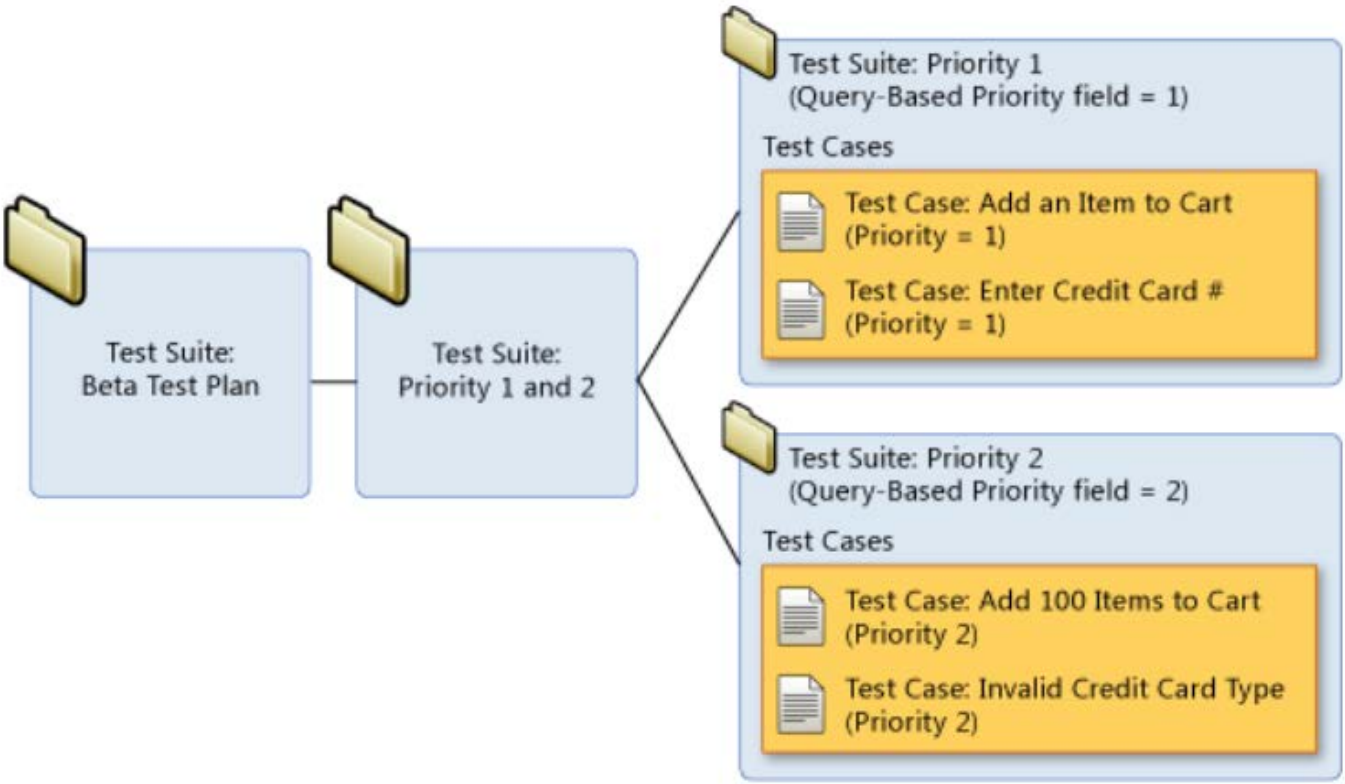

Рисунок 3 – Тестовий набір

3. Чек-лист

Чек-лист - це список, який містить ряд необхідних перевірок під час тестування програмного продукту. Відзначаючи пункти списку, команда або один тестувальник можуть дізнатися про поточний стан виконаної роботи і про якість продукту. Працюючи над проектом по чек-листу, виключена ймовірність повторної перевірки за тими ж кейсам, а також підвищується якість тестування, так як ймовірність залишити без уваги якийсь функціонал суттєво знижується. Тому дуже важливо знати з яких елементів складається чек-лист і вміти ним ефективно користуватися. Як правило, чек-листи роблять в Google-таблиці для забезпечення загального доступу всім QA- фахівцям.

Будь-який з чек-листів містить перелік блоків, секцій, сторінок, інших елементів, які слід протестувати. Пункти можуть містити як лінійну структуру, так і деревоподібну, з розділами / підрозділами або без них. Вони повинні бути максимально короткі і в той же час зрозумілі тестувальнику, який ще не знайомий з додатком. Пункти повинні бути однозначними, щоб їх не можна було зрозуміти якось інакше. Всі пункти повинні бути оформлені однією мовою: російською чи англійською. Як правило, кожен чек-лист має кілька стовпців. Кожен стовпець призначений для тестування на окремій платформі. Слід завжди вказувати назву пристрою, браузера і його версії.

Таким чином, чек-лист містить:

1. Список перевірок (з необхідним ступенем деталізації).
2. Оточення перевірки:
   - збірка, на якій проводилося тестування;
   - тестове оточення (якщо є);
   - інформація про тестувальників.
3. Результат перевірки.

Тестувати додаток може декілька людей одночасно. У цьому випадку кожен тестувальник бере по одній-дві платформи і тестує тільки на них. При цьому слід навпроти кожної платформи вказати тестувальника, який виконує зазначений обсяг робіт.

При проходженні чек-листів тестувальник зазначає статус навпроти кожного тестованого пункту. Можливі такі варіанти статусів:

4. «Passed» - перевірка пройдена успішно, багів, не знайдено;

5. «Failed» - знайдений один або більше багів;

6. «Blocked» - неможливо перевірити, тому що один з багів блокує поточну перевірку;

7. «InProgress» - поточний пункт, над яким працює тестувальник;

8. «Notrun» - ще не перевірено;

9. «Skipped» - перевірятися не буде з будь-якої причини. Наприклад, поточний функціонал ще не реалізований.

Для більшої наочності, як правило, кожен з статусів має свій колір (рисунок 4).

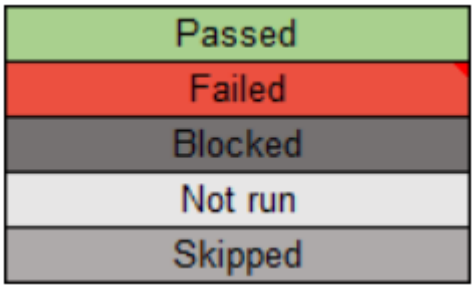

Рисунок 4 – Приклад маркування статусів чек-листа

Після завершення тестування не повинно бути осередків, позначених «Notrun».

Всі заведені по чек-листу баг-репорти повинні бути додані в примітки до кожної комірки зі статусом «Failed» (рисунок 5).

	Иванов А. А.	Петров Б. Б.	Фёдоров В.В.	Васечкин Г. Г.
Сайт "example.edu"	Google Chrome 64.0.3282.140	Mozilla Firefox 58.0.2	IE 11.0	Opera 51.0.2830.34
Регистрация и Личный профиль				
Регистрация на сайте	Passed			
Редактирование профиля	Failed			
Форма обратной связи				
Валидация полей	Failed	Failed	Passed	Passed
Отправка письма/уведомления	Blocked	Blocked	Passed	Passed
Доставка письма/уведомления	Blocked	Blocked	Skipped	Skipped

http://bt-w.qatestlab.com/view.php?id=...

Рисунок 5 – Приклад чек-листа

Цілком допустимо додавати примітки та до деяких комірок з іншими статусами, якщо це необхідно.

Для комірок зі статусом «Blocked» примітки з посиланнями на баг- репорти також обов'язкові. Однак, як правило, примітка в комірці зі статусом «Blocked» посилається на

раніше заведений баг- репорт, який в одному з попередніх пунктів вже відзначений як «Failed». Іншими словами: пункт, одного разу зазначений як «Failed» може бути блокером для декількох або всіх наступних пунктів чек-листа.

Відзначимо кілька основних моментів, які варто враховувати при роботі з чек-листами:

– по завершенні проходження чек-листа не повинно залишитися осередків зі статусом «Notrun»;

– всі комірки зі статусом «Failed» і «Blocked» обов'язково повинні мати примітки з посиланнями на баг- репорти;

– статус «Passed» встановлюється тільки для пунктів, які перевірені і не містять помилок.

Для складання ефективного чек-листа існує **кілька правил:**

1. Один пункт - одна операція.

Пункти чек-листа - це однозначні атомарні і повні операції. Наприклад, додавання товару в корзину сайту і оплата замовлення - дві різні завдання. У списку перевірок подібні операції оформлені окремими пунктами: доданий товар в корзину, оплата відправлена.

2. Пункти починаються з іменника.

Мета чек-листа - врахувати всі дії для найбільш повного покриття тестами ПО, тому складаючи пункти слід дотримуватися уніфікованої форми. Для зрозумілого і однозначного уявлення пункти краще починати з іменника - «Перевірка», «Додавання», «Відправлення» або дієслова невизначеної форми - «Перевірити», «Додати», «Відправити».

3. Складання чек-листа за рівнями деталізації.

Для зручності проходження чек-листа найкраще складати тести в тому вигляді, який буде послідовним виходячи з логіки використання функціоналу. Наприклад, в рамках розділу «Реєстрація та Особистий профіль»: реєстрація на сайті, редагування профілю; розділ «Форма зворотного зв'язку»: валідація полів, відправка листа, доставка листи.

Переваги використання чек-листів:

1. Використання чек-листів сприяє структуруванню інформації у співробітника.

2. При правильній записи необхідних дій у співробітника з'являється однозначне розуміння завдань. Це сприяє підвищенню швидкості навчання нових співробітників.

3. Чек-листи допомагають уникнути невизначеності та помилок пов'язаних з людським фактором. Збільшується покриття тестами програмного продукту.

4. Підвищується ступінь взаємозамінності співробітників.

5. Економія робочого часу. Написавши чек-лист одного разу його можна використовувати повторно, з огляду на актуальність інформації.

6. Підвищення бас фактора. В області розробки програмного забезпечення бас фактор («busfactor» - фактор автобуса) проекту - це міра зосередження інформації серед окремих членів проекту.

Хороший чек-лист - це документ, що описує, що має бути про тестовано. При цьому чек-лист може бути абсолютно різного рівня деталізації. На скільки детальним буде чек-лист, залежить від вимог до звітності, рівня знання продукту співробітниками і складності продукту.

4 Баг Репорт

9 вересня 1947 року вчені Гарвардського університету, які тестували обчислювальну машину Mark знайшли застряглого метелика між контактами електромеханічного реле, який привів до зупинки машини. Баг в перекладі з англійської - це метелик, жук, комаха. З тих пір будь-які неполадки в обчислювальній техніці називають **багом**.

Багом можна назвати:

- невідповідність вимогам або специфікації;
- збій програми, що приводить до її завершення;
- зависання програми (програмою неможливо користуватися);
- видача будь-яких програмних і системних помилок;
- помилки дизайну, програмного інтерфейсу, який заважає нормальній роботі;
- дрібні неточності і текстові помилки.

В якості синоніма до поняття баг використовують слово дефект. Дефект (баг) - це помилка в програмі, що викликає її неправильну і (або) непередбачувану роботу, також

дефектом називають відмінність між фактичним і очікуваним результатом. Через дефекти, допущені ще під час написання коду, програма може не виконувати закладених функцій, працювати не так, як зазначено в специфікації, або виконувати дії, які не передбачені. Такі випадки називаються збоями програми.

Баги є у всіх програмах і вони залишаються в них навіть після релізу. Це відбувається через те, що провести вичерпне тестування неможливо. Причиною тому є недолік часу, ресурсів, а також величезна кількість вхідних значень і сценаріїв перевірки. У реліз зазвичай випускається програмне забезпечення, яке містить незначні баги, а знайдені серйозні і критичні баги виправлені.

Баг або дефект репорт - це документ, що описує ситуацію або послідовність дій, щ призвела до некоректної роботи об'єкта тестування, із зазначенням причин і очікуваного результату.

Різні системи менеджменту дефектами, пропонують нам різні поля для заповнення і різні структури опису дефектів. В Додатку 4 наведено приклад шаблону для формування баг- репорту.

Баг- репорт - це технічний документ і в зв'язку з цим, мова опису проблеми повинна бути технічною. Повинна використовуватися правильна термінологія при використанні назви елементів, призначеного для користувача інтерфейсу (editbox, listbox, combobox, link, textarea, button, menu, popupmenu, titlebar, systemtray і т.д.), дій користувача (clicklink, pressthebutton , selectmenuitem і т.д.) і отриманих результатах (windowisopened, errormessageisdisplayed, systemcrashed і т.д.).

Обов'язковими полями баг репорту є:

- короткий опис (BugSummary),
- серйозність (Severity),
- кроки до відтворення (Stepstoreproduce),
- результат (ActualResult),
- очікуваний результат (ExpectedResult).

Короткий опис (BugSummary) містить одне речення, в якому треба вмістити сенс всього баг- репорту, а саме: коротко і ясно, використовуючи правильну термінологію сказати що і де не працює.

Короткий опис рекомендується складати за принципом «Де? Що? Коли?», який полягає у складенні речення, в якому факти дефекту викладені в наступній послідовності:

Де?: У якому місці інтерфейсу користувача або архітектури програмного продукту знаходиться проблема. Речення бажано починати з іменника.

Що?: Що відбувається або не відбувається згідно специфікації або вашому уявленню про нормальну роботу програмного продукту. При цьому вказуйте на наявність або відсутність об'єкта проблеми, а не на його зміст (його вказують в описі). Якщо зміст проблеми варіюється, всі відомі варіанти вказуються в описі. «Що?» - це НЕ іменник, це те, що сталося, свого роду ДІЄСЛОВО-уточнення («Що виконується?» Або «Що НЕ виконується?»).

Коли?: У який момент роботи програмного продукту, по настанню якої події або за яких умов проявляється проблема.

Чому послідовність повинна бути саме такою? У такому вигляді незнайомі дефекти зручніше сортувати по summary як показує практика (адже, швидше за все, саме серед дефектів інших інженерів буде проводитись пошук дублікатів).

Можна використовувати послідовність «Що? Де? Коли?». Головне, щоб всі описи багів відповідали єдиному шаблону.

Бувають випадки, коли немає можливості помістити всю необхідну для опису бага інформацію в поле «Summary». Причиною може стати встановлене обмеження на кількість символів в поле, або щоб уникнути накопичення великої кількості тексту в полі. У таких випадках рекомендується сформулювати опис бага таким чином, щоб в мінімальній кількості тексту максимально зрозуміло була описана суть заведеного багу, а вже в поле «Description» помістити більш детальний варіант опису багу.

Основні помилки при написанні багів репортів:

1. Недостатність наданих даних.

Не завжди одна і таж проблема проявляється при всіх впроваджуються значеннях і під будь-яким увійшов в систему користувачем, тому настійно рекомендується вносити всі необхідні дані в баг репорт

2. Визначення серйозності.

Дуже часто відбувається або завищення, або заниження серйозності дефекту, що може привести до неправильної черговості при вирішенні проблеми.

3. Мова опису.

Часто при описі проблеми використовуються неправильна термінологія або складні мовні звороти, які можуть ввести в оману людину, відповідальну за вирішення проблеми.

4. Відсутність очікуваного результату.

У випадках, якщо не вказано, що має бути необхідним поведінкою системи, витрачається час розробника на пошук цієї інформації, тим самим уповільнюється виправлення дефекту. Необхідно вказати пункт у вимогах, написаний тест кейс або ж особисту думку тестувальника, якщо ця ситуація не була документована.