**Практическая работа**

**ТЕМА: ТЕСТОВЫЕ ДАННЫЕ И СЦЕНАРИИ**

**1.Основные понятия**

**Тестовый сценарий (test case) или тестовый случай** - набор входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанный для определенной цели или тестового условия. Например, выполнение определенного сценария работы программы или же для проверки соответствия определенному требованию.

Тестовые сценарии проверяют выполнение требований к приложению. Для этого используются тестовые оракулы.

**Требование (requirement) -** условия или возможности, необходимые пользователю для решения определенных задач или достижения определенных целей, которые должны быть достигнуты для выполнения контракта, стандартов, спецификации или других формальных документов.

**Тестовый оракул (test oracle)** - источник, при помощи которого можно определить ожидаемые результаты для сравнения с реальными результатами, выдаваемыми тестируемой системой. В роли тестового оракула могут выступать уже имеющаяся система (для эталонного тестирования), руководство пользователя, профессиональные знания специалиста, однако им не может быть программный код.

**Тестовое условие (test condition)** - объект или событие в компоненте или системе, которые должны быть проверены одним или несколькими тестовыми наборами. Например, функция, транзакция, свойство, атрибут качества или структурный элемент.

Тестовые сценарии, как минимум, состоят из **действий (шагов)** **и ожидаемых результатов**. В некоторых случаях описываются предусловия и постусловия. Также рекомендуется создавать краткое содержание тестового случая. В содержании обычно описывается цель теста и выбор методики тестирования. Некоторые тесты сопровождаются визуальной информацией - карты, схемы и т.д.

1. **Предусловие (Pre Conditions)** - список действий или критерии, которые приводят систему к состоянию, пригодному для проведения основной проверки.

2. **Шаги/Ожидаемые результаты (Actions/Res.ults)** - список действий, переводящих систему из одного состояния в другое, для по лучения результата, на основании которого можно сделать вывод об удовлетворении реализации поставленным требованиям.

3. **Постусловие (Post Conditions)** - список действий, переводящих систему в первоначальное состояние.

В качестве предусловия часто выступает задача - создать тестовые данные.

**Тестовые данные (test data)** - данные, которые существуют (например, в базе данных) на начало выполнения теста и влияют на работу или же испытывают влияние со стороны тестируемой системы или компонента.

Группа тестовых сценариев называется **тестовым набором (test suite)**. Группировать тесты можно по тестовым областям, объекту тестирования, типу тестов, цели тестов и т.д.

Мероприятия по проектированию тестов называются **тест-анализом, тест-дизайном**. Во время этих мероприятий вырабатывается подход для проверки функциональности на основе методик тестирования (методика черного ящика, методика, основанная на опыте; методика, основанная на спецификации; методика, основанная на структуре, и т.д.).

**2.Тестовые сценарии**

**Тестовая документация** состоит обычно из отдельных сценариев, которые называются тест-кейсами (англ. - test case) и могут быть для удобства объединены в группы или тест-сьюты (англ. - test suites - тестовые наборы).

Написание тестовой документации имеет много общего с написанием ПО: следует разбивать код на отдельные модули и избегать дублирования кода.

В общем случае тестовая документация может содержать заголовок, пошаговое описание, ожидаемый результат, критерий соответствия ожидаемого результата фактическому результату.

Заглавие (суть) тест-кейса (title) призвано упростить быстрое понимание основной идеи тест-кейса без обращения к его остальным атрибутам. Именно это поле является наиболее информативным при просмотре списка тест-кейсов.

Пример заголовков тест-кейсов приведен в табл. 1.

Заглавие тест-кейса может быть полноценным предложением, фразой, набором словосочетаний - главное, чтобы выполнялись следующие условия:

Таблица 1

**Пример плохих и хороших заголовков тест-кейсов**

|  |  |
| --- | --- |
| Плохо | Хорошо |
| Тест 1 | Запуск, одна копия, верные параметры |
| Тест 2 | Запуск одной копии с неверными путями |
| Тест 78 (улучшенный) | Запуск, много копий, без конфликтов |
| Остановка | Остановка по Ctrl+C |
| Закрытие | Остановка закрытием консоли |

1) информативность;

2) относительная уникальность (чтобы не путать разные тест кейсы).

**Тест-кейс** должен обязательно содержать хотя бы ожидаемый результат (даже, может быть, без описания действий, которые к нему ведут). Например, «Программа должна уметь показывать файлы формата ВМР». Такой тест-кейс представляет собой просто перепечатку из документа с требованиями.

Ожидаемые результаты (expected results) по каждому шагу тест-кейса описывают реакцию приложения на действия, описанные в поле «шаги тест-кейса». Номер шага соответствует номеру результата.

По написанию ожидаемых результатов можно порекомендовать следующее:

1) описывать поведение системы так, чтобы исключить субъективное толкование (например, «приложение работает верно» - плохо,

«появляется окно с надписью...» - хорошо);

2) писать ожидаемый результат по всем шагам без исключения, если есть хоть малейшие сомнения в том, что результат некоего шага будет совершенно тривиальным и очевидным;

3) писать кратко, но не в ущерб информативности;

4) избегать условных конструкций вида «если... то...».

Однако из такого тест-кейса непонятно, как осуществить проверку. Человек, незнакомый с программой, может не найти в ее интерфейсе, как показывать графические файлы, и напишет об отсутствии такой возможности.

Поэтому кроме ожидаемого результата необходимы еще шаги, которые позволят прийти к результату фактическому и сравнить его с ожидаемым результатом.

Шаги тест-кейса (steps) описывают последовательность действий, которые необходимо реализовать в процессе выполнения тест-кейса. Общие рекомендации по написанию шагов таковы:

1) необходимо начинать с понятного и очевидного места, не писать лишних начальных шагов (запуск приложения, очевидные операции с интерфейсом и т.п.);

2) даже если в тест-кейсе всего один шаг, нумеровать его (иначе возрастает вероятность в будущем случайно «приклеить» описание этого шага к новому тексту);

3) использовать безличную форму (например, «открыть», «ввести», «добавить», вместо «откройте», «введите», «добавьте»);

4) соотносить степень детализации шагов и их параметров с целью тест-кейса, его сложностью, уровнем и т.д. - в зависимости от этих и многих других факторов степень детализации может варьироваться от общих идей до предельно четко прописанных значений и указаний;

5) ссылаться на предыдущие шаги и их диапазоны для сокращения объема текста (например, «повторить шаги 3-5 со значением...»);

6) писать шаги последовательно, без условных конструкций вида

«если... то...».

Краткое описание тест-кейса имеет смысл вынести в заголовок. Термин «тест-кейс» может относиться к формальной записи тест-кейса в виде технического документа. Эта запись имеет общепринятую структуру, компоненты которой называются атрибутами (полями) тест-кейса.

В зависимости от инструмента управления тест-кейсами внешний вид их записи может немного отличаться, могут быть добавлены или убраны отдельные поля, но концепция остается неизменной.

Секцию «Предварительные шаги/предварительные условия», как и «Ожидаемый результат», тоже можно сделать гиперссылкой на соответствующую инструкцию. Таким образом, мы избавимся от дублирования информации в разных тестах и улучшим читабельность тестов и их поддержку, если в системе что-то поменяется. Вообще, процесс написания и поддержки тестовой документации имеет много общего с написанием и поддержкой программного обеспечения. Здесь так же важно избавляться от дублирования кода и выделять его в отдельные процедуры.

Часто один и тот же тест-кейс следует выполнять с разными тестовыми данными. Например, если программа должна уметь показывать файлы в формате ВМР, JPG и GIF, логично написать один тест-кейс и указать в специальной секции, что выполняться он должен с использованием трех файлов разного формата. По аналогии с программированием название формата выносится в параметр процедуры. Такой тест-кейс называется data-driven - управляемый данными. Тестовую документацию следует писать так, чтобы ее было легко поддерживать при изменениях в продукте.

Хорошим стилем в написании тестовой документации является низкоуровневые описание действий, имеющее ссылки на пошаговое описание. Такая документация пригодна для использования как опытными (знакомыми с продуктом), так и начинающими тестировщиками. Опытным тестировщикам не надо тратить время на чтение пошаговых инструкций (как правило, они их и так знают), а неопытные всегда смогут их прочитать.

Если же стоит задача написать тестовую документацию в кратчайшие сроки, приходится выбирать между пошаговым и высокоуровневым стилем (без ссылок). Как правило, следует выбирать высокоуровневый стиль, потому что, во-первых, так быстрее писать, а во-вторых, снабдить такой документ ссылками на пошаговое описание в дальнейшем будет проще, чем перерабатывать пошаговые инструкции.

Итак, тестовые сценарии необходимы по следующим причинам:

1) для проверки конкретной области, характеристики и т.д. функциональности с требуемой степенью детализации;

2) с целью получения полезной документации и использования как оракула;

3) для обучения новичков.

**3. Идентификатор тест-кейса, приоритет, время прохождения**

Тест-кейсы полезно снабжать уникальными идентификаторами, чтобы можно было легко на них ссылаться. Идентификатор (identifier) представляет собой уникальное значение, позволяющее однозначно отличить один тест-кейс от другого и используемое во всевозможных ссылках. В общем случае идентификатор тест-кейса может представлять собой просто уникальный номер, но (если позволяет инструментальное средство управления тест-кейсами) может быть и куда сложнее: включать префиксы, суффиксы и иные осмысленные компоненты, позволяющие быстро определить цель тест-кейса и часть приложения (или требований), к которой он относится.

Приоритет (priority) показывает важность тест-кейса. Логично, что наиболее важные, критичные тест-кейсы следует проходить в первую очередь, менее важные - во вторую и т.д., чтобы при нехватке времени пропускались менее необходимые вещи. Существует несколько методик расстановки приоритетов. Он может быть выражен буквами (А, В, С, D, Е), цифрами (1, 2, 3, 4, 5), словами («крайне высокий», «высокий», «средний», «низкий», «крайне низкий») или иным удобным способом. Количество градаций также не фиксировано, но чаще всего лежит в диапазоне от трех до пяти.

Приоритет тест-кейса может коррелировать:

1) с важностью требования, пользовательского сценария или функции, с которыми связан тест-кейс;

2) потенциальной важностью дефекта, на поиск которого направлен тест-кейс;

3) степенью риска, связанного с проверяемым тест-кейсом требованием, сценарием или функцией.

Основная задача этого атрибута - упрощение распределения внимания и усилий команды (более высокоприоритетные тест-кейсы получают их больше), а также упрощение планирования и принятия решения о том, чем можно пожертвовать в некой форс-мажорной ситуации, не позволяющей выполнить все запланированные тест-кейсы.

вязанное с тест-кейсом требование (requirement) показывает то основное требование, проверке выполнения которого посвящен тест-кейс (основное потому, что один тест-кейс может затрагивать несколько требований). Наличие этого поля улучшает такое свойство тест-кейса, как прослеживаемость.

Частые вопросы, связанные с заполнением этого поля, таковы.

1. Можно ли его оставить пустым? Да. Тест-кейс вполне мог разрабатываться вне прямой привязки к требованиям, и значение этого поля определить сложно. Хоть такой вариант и не считается хорошим, он достаточно распространен.

2. Можно ли в этом поле указывать несколько требований? Да, но чаще всего стараются выбрать одно, самое главное или «более высокоуровневое» (например, вместо того чтобы перечислять R56.1, R56.2, R56.3 и т.д., можно просто написать R56). Чаще всего в инструментах управления тестами это поле представляет собой выпадающий список, где можно выбрать только одно значение, и этот вопрос становится неактуальным. К тому же многие тест-кейсы все же направлены на проверку строго одного требования и для них этот вопрос также неактуален.

Модуль и подмодуль приложения (module and submodule) указывают на части приложения, к которым относится тест-кейс, и позволяют лучше понять его цель.

Идея деления приложения на модули и подмодули проистекает из того, что в сложных системах практически невозможно охватить взглядом весь проект целиком и вопрос «как протестировать это приложение» становится недопустимо сложным. Тогда приложение логически разделяется на компоненты (модули), а те, в свою очередь, - на более мелкие компоненты (подмодули). И вот уже для таких небольших частей приложения придумать чек-листы и создать хорошие тест-кейсы становится намного проще.

Как правило, иерархия модулей и подмодулей создается как единый набор для всей проектной команды, чтобы исключить путаницу из-за того, что разные люди будут использовать разные подходы к такому разделению или даже просто разные названия одних и тех же частей приложения.

Целесообразно также указывать планируемое время прохождения тест-кейса (при его создании) - для оценки трудозатрат на тестирование. Это время может быть скорректировано с учетом реальной истории прохождения.

**4.История изменений и история прохождений**

По аналогии с написанием программного обеспечения имеет смысл документировать все изменения в тестовой документации, чтобы можно было понять, почему, кто и когда их сделал. В простейшем случае тестовая документация может храниться в централизованной системе управления версиями, например CVS или SVN, в виде отдельного документа на каждый тест-кейс или на группу кейсов, связанных по смыслу.

Однако этот способ не так удобен, как применение специализированных систем поддержки тестовой документации, например TestRail или Testlink. Такие системы имеют ряд дополнительных функций, связанных с прохождением тест-кейсов:

1) возможность назначать тест-кейсы сотрудникам для прохождения;

2) возможность собирать статистику прохождений (фамилия сотрудника, потраченное время и найденные ошибки и дефекты).

Итак, к стандартным атрибутам тест-кейсов можно отнести следующие.

1. **Номер** - уникальный идентификатор тест-кейса. Его удобно использовать для однозначного понимания, о какой проверке идет речь.

2. **Название** - краткое описание сути проверки. Должно быть емким, кратким и понятным.

3. **Предварительные шаги** - описание действий, которые необходимо выполнить, но прямого отношения к проверке они не имеют (например, зарегистрироваться в системе для проверки создания элемента). Если предварительных шагов нет, то секция не заполняется.

4. **Шаги** - описание действий, необходимых для проверки (например, создание элемента).

5. **Ожидаемый результат (ОР)** - сама проверка: что мы ожидаем получить после выполнения шагов («Элемент создан»).

Также могут быть записаны **результаты для нескольких шагов из кейса:** один из вариантов записи тест-кейса с несколькими ожидаемыми результатами: результаты пишутся на разные пункты шагов выполнения проверки, т.е. на разные этапы сценария.

**5.Некоторые метрики по тест-кейсам**

1. Test Coverage - процент покрытия тестовыми случаями требований.

2. Passed/Failed Test Cases - результаты прохождения тестов, а именно отношение количества удачно пройденных к завершившимся с ошибками.

3. Not Run Test Cases - количество тестов, которые не запускались.

**6.Примеры тестовых сценариев**

**Пример 1.**

Заголовок: «Проверка того, что программа умеет показывать файлы формата ВМР».

Шаги:

1) нажать кнопку «Выбрать файл»;

2) выбрать файл с расширением ВМР;

3) нажать кнопку «Открыть».

Ожидаемый результат: содержимое файла показано в графическом виде в полноэкранном режиме.

Здесь сравнение ожидаемого и фактического результатов осуществить довольно просто и критерий соответствия не нужен.

**Пример 2.**

Заголовок: «Проверка изменения домашнего телефона пользователя в ActiveDirectory».

Шаги:

1) нажать кнопку «Создать пользователя»;

2) ввести имя пользователя, логин и пароль;

3) нажать кнопку «ОК»;

4) выбрать только что созданного пользователя в списке, кликнув на его логин;

5) нажать кнопку «Редактировать»;

6) ввести номер телефона в поле «Домашний телефон»;

7) нажать кнопку «ОК».

Ожидаемый результат: домашний телефон сохранился в Active Directory.

Здесь ожидаемый результат было бы неплохо также расписать в виде последовательности шагов: как посмотреть в ActiveDirectory, что домашний телефон сохранился. Это и будет описанием критерия соответствия. Можно записать эти шаги здесь же, начиная с номера 8, и уточнить ожидаемый результат:

8) авторизоваться на сервере AciveDirectory;

9) открыть оснастку dsa.msc;

10) найти пользователя по логину;

11) посмотреть значение поля Ноте phone.

Ожидаемый результат: это значение соответствует номеру телефона в поле «Домашний телефон».

Однако, строго говоря, эти шаги не являются частью тестируемого сценария. При изменении или удалении сценария эта инструкция может быть потеряна. Поэтому имеет смысл записать ее на специальном сайте - базе знаний (Кnowledge Base, КВ), а в тест-кейсе дать ссылку на эту инструкцию.

Кроме того, если эта инструкция будет использована в других тест-кейсах, нам не нужно будет ее каждый раз копировать, достаточно будет давать ссылку.

Таким образом, возвращаемся к предыдущему варианту и вставляем ссылку в поле «Ожидаемый результат».

Ожидаемый результат: домашний телефон сохранился в ActiveDirectory (смотри ссылку).

Рассмотренный выше тест-кейс должен проверять редактирование телефона пользователя. Однако же первые три шага не относятся к редактированию телефона. Это вспомогательные шаги по созданию пользователя: создается тестовый объект - объект, который мы будем использовать в тестах.

Исходные данные, необходимые для выполнения тест-кейса (precondition, preparation, initial data, setup), позволяют описать все то, что должно быть подготовлено до начала выполнения тест-кейса, например:

1) состояние базы данных;

2) состояние файловой системы и ее объектов;

3) состояние серверов и сетевой инфраструктуры. Целесообразно вынести шаги из примера в отдельную секцию

«Исходные данные» и описать более кратко.

Исходные данные.

Создать пользователя.

1. Открыть список пользователей.

2. Выбрать пользователя в списке, кликнув на его логин.

3. Нажать кнопку «Редактировать».

4. Ввести номер телефона в поле «Домашний телефон».

5. Нажать кнопку «ОК».

Ожидаемый результат: домашний телефон сохранился в Active Directoгy.

**Пример 3.**

Заголовок: проверка возможности входа в систему с валидными данными.

Шаги:

1) открыть браузер;

2) зайти на главную страницу веб-приложения;

3) в форме логина ввести:

3.1) username: Кirill;

3.2) password: 123456;

4) нажать «Enter».

Ожидаемый результат: пользователю открывается страница личного кабинета.

Пример 4.

Есть внутренний сайт компании, которая проводит Интернет «Самый\_лучший\_в\_своем\_роде» - www.test.ru. Примечание: www. test.ru - абстрактное обозначение сайта. Приводится здесь, чтобы показать ошибки в написании тест-кейсов.

На сайте можно заводить карточки обслуживаемых зданий и карточки их жильцов. Карточки создает администратор, на тестовой машине всегда есть пользователь с правами админа, логин/пароль ..:\_ admin/1. При входе на тестовый сервер есть дополнительная авторизация, чтобы туда не могли попасть люди «Извне», с логином и паролем test/test.

**Тест-кейс № 1**.

Заголовок: создание жильца без ФИО.

Шаги:

1) зайти на сайт www.test.ru (логин - test, пароль - test);

2) войти под учетной записью администратора (логин - admin, пароль - 1);

3) перейти на вкладку «Жильцы»;

4) нажать на кнопку «Создать карточку жильца»;

5) нажать на кнопку «Сохранить», не заполняя никакие данные.

Ожидаемый результат: появляется сообщение об ошибке «Заполните обязательные поля, отмеченные», карточка не сохраняется.

**Тест-кейс № 2.**

Заголовок: создание жильца, проверка поля «ФИО».

Шаги:

1) зайти на сайт www.test.ru (логин - test, пароль - test);

2) войти под учетной записью администратора (логин - admin, пароль - 1);

3) перейти на вкладку «Жильцы»;

4) нажать на кнопку «Создать карточку жильца»;

5) заполнить поле ФИО (см. табл. 2 «Ожидаемый результат»);

6) нажать на кнопку «Сохранить».

Ожидаемый результат приведен в табл. 2.

Таблица 2

**Ожидаемый результат**

|  |  |
| --- | --- |
| Вводимое значение | Ожидаемый результат |
| Киселева Ольга Евгеньевна | Ок, карточка сохраняется |
| «Оставить поле пустым» | Ошибка - «Заполните обязательные поля, отмеченные\*», карточка не сохраняется |
| 2\*4\*6\*8\*11\*14\*17\*20\*23\*26\*2  9\*32\*35\*38\*41\* | Ошибка - «Максимальная длина поля - 40 символов, введено - 41», карточка не сохраняется. (Такую строчку легко сформировать с помощью инструмента PerlClip.) PerlClip - это инструмент для подготовки стрессовых текстовых данных для проверки текстовых полей |
| &\*%#(л$@\*& | Ошибка - «Поле "ФИО" может содержать только буквы русского алфавита», карточка не сохраняется |
| Кiseleva Olga Evgenievna | Ошибка - «Поле "ФИО" может содержать только буквы русского алфавита», карточка не сохраняется |
| ... | ... |

**Ответить на контрольные вопросы**

1. Что такое тестовые сценарии?

2. Объясните, что представляют собой тестовые данные.

3. Что такое тестовый оракул?

4. Кому полезнее тестовые сценарии - новичкам или профессионалам?

5. Опишите стандартные атрибуты тест-кейса.

6. Назовите все возможные атрибуты тест кейса. Что они собой представляют?

**Практическое задание**

1. В соответствии с вариантом задания найти существующее веб-приложение, подходящее под требуемую предметную область.

Предметные области для выполнения задания:

1.1. Книжный интернет-магазин.

1.2. Портал поиска дешевых авиабилетов.

1.3. Приложение для смартфона «Заказ такси».

1.4. Приложение для поиска другого смартфона.

1.5. Мессенджер.

1.6. Онлайн-банк.

1.7. Онлайн-запись на прием к стоматологу.

1.8. Приложение для отслеживания заказа в пути.

1.9. Онлайн-переводчик.

1.10. Навигатор.

1.11. Приложение «Контроль расходов».

1.12. Программный продукт, ведущий учет студентов университета. 1 .13. Онлайн-кинотеатр.

1.14. Сайт покупки авиабилетов.

1.15. Сайт для практики иностранного языка.

1.16. Приложение, отслеживающее состояние здоровья человека.

1.17. Электронная зачетная книжка.

1.18. Сайт дистанционного обучения.

1.19. Сайт программы лояльности крупной сети магазинов.

1.20. Приложение распознавания музыки.

2. Разработать тестовые сценарии к веб-приложению на основе примеров, описанных выше.

3. После тестирования рассчитать известные вам метрики (Test Coverage, Passed/Failed Test Cases, Not Run Test Cases). **(Необязательно для практического выполнения)**

4. Оформить отчет по лабораторной работе.