Тема 2.2 Понятие тестового случая

Тестовый случай (**Test Case**) – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или ее части.

Под тест кейсом понимается структура вида:

Action –> Expected Result –> Test Result

Действие –> Ожидаемый результат –> Результат теста

Пример:

Action	Expected Result	Test Result (passed/failed/blocked)
Open page "login"	Login page is opened	Passed

Цель написания тест-кейсов:

Тестирование можно проводить и без тест-кейсов (не нужно, но можно; да, эффективность такого подхода варьируется в очень широком диапазоне, в зависимости от множества факторов). Наличие же тест-кейсов позволяет:

1) Структурировать и систематизировать подход к тестированию (без чего крупный проект почти гарантированно обречён на провал).

2) Вычислять метрики тестового покрытия (test coverage metrics) и принимать меры по его увеличению (тест-кейсы здесь являются главным источником информации, без которого существование подобных метрик теряет смысл).

3) Отслеживать соответствие текущей ситуации плану (сколько примерно понадобится тест-кейсов, сколько уже есть, сколько выполнено из запланированного на данном этапе количества и т.д.).

4) Уточнить взаимопонимание между заказчиком, разработчиками и тестировщиками (тест-кейсы зачастую намного более наглядно показывают поведение приложения, чем это отражено в требованиях).

5) Хранить информацию для длительного использования и обмена опытом между сотрудниками и командами (или, как минимум, не пытаться удержать в голове сотни страниц текста).

6) Проводить регрессионное тестирование и повторное тестирование (которые без тест-кейсов было бы вообще невозможно выполнить).

7) Повышать качество требований (написание чек-листов и тест-кейсов – хорошая техника тестирования требований).

8) Быстро вводить в курс дела нового сотрудника, недавно подключившегося к проекту.

Жизненный цикл тест-кейса

В структуре жизненного цикла для тест-кейса выделим набор состояний, в которых тест-кейс может находиться (см. рисунок ниже. Жирным шрифтом отмечены наиболее важные состояния).

Создан (new) – типичное начальное состояние практически любого артефакта. Тест-кейс автоматически переходит в это состояние после создания.

Запланирован (planned, ready for testing) – в этом состоянии тест-кейс находится, когда он или явно включён в план ближайшей итерации тестирования, или, как минимум, готов для выполнения.

Не выполнен (not tested) – в некоторых системах управления тест-кейсами это состояние заменяет собой предыдущее («запланирован»). Нахождение тест-кейса в данном состоянии означает, что он готов к выполнению, но ещё не был выполнен.

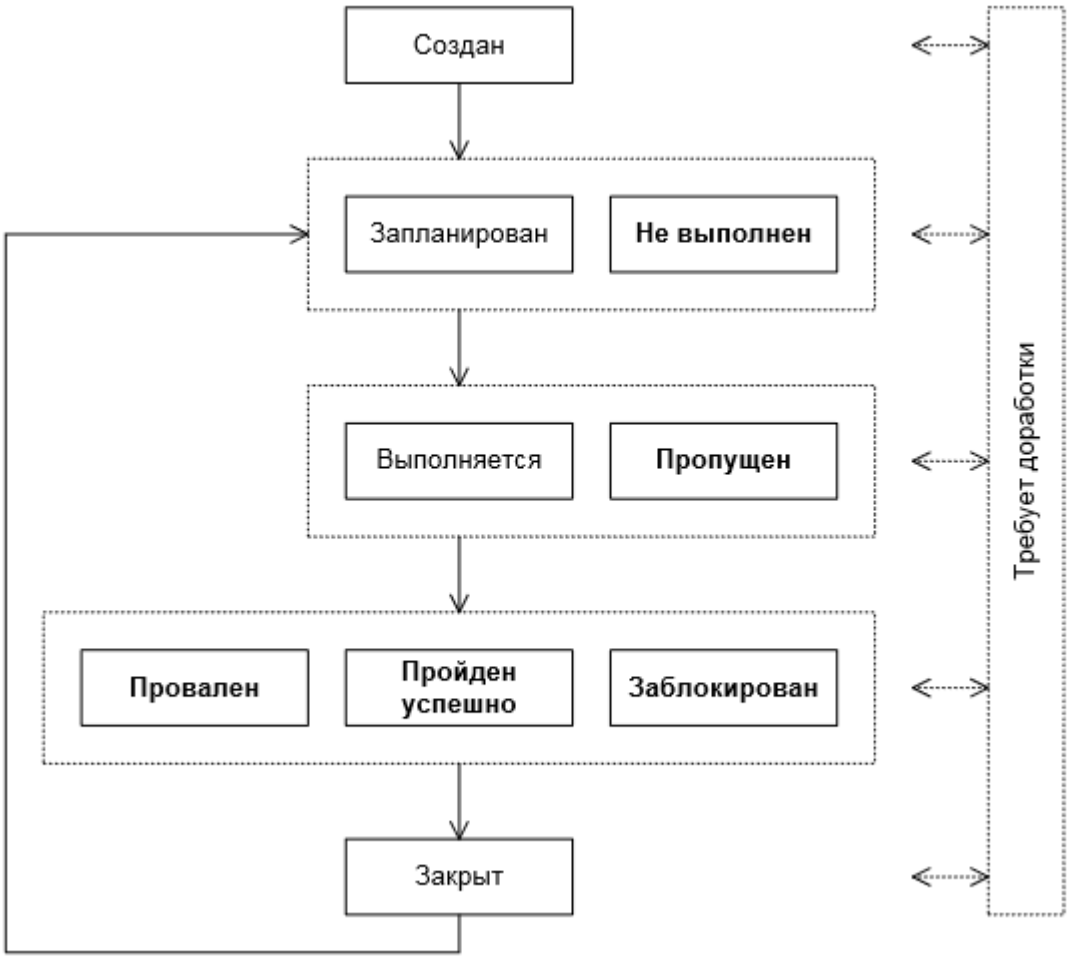


Рисунок – Жизненный цикл тест-кейса

Выполняется (work in progress) – если тест-кейс требует длительное время для выполнения, то он может быть переведён в это состояние для подчёркивания того факта, что работа идёт, и скоро можно ожидать её результатов. Если выполнение тест-кейса занимает мало времени, это состояние, как правило, пропускается, а тест-кейс сразу переводится в одно из трёх следующих состояний - «провален», «пройден успешно» или «заблокирован».

Пропущен (skipped) – бывают ситуации, когда выполнение тест-кейса отменяется по соображениям нехватки времени или изменения логики тестирования.

Провален (failed) – данное состояние означает, что в процессе выполнения тест-кейса был обнаружен дефект, заключающийся в том, что ожидаемый результат по как минимум одному шагу тест-кейса не совпадает с фактическим результатом. Если в процессе выполнения тест-кейса был «случайно» обнаружен дефект, никак не связанный с шагами тест-кейса и их ожидаемыми результатами, тест-кейс считается пройденным успешно (при этом, естественно, по обнаруженному дефекту создаётся отчёт о дефекте).

Пройден успешно (passed) – данное состояние означает, что в процессе выполнения тест-кейса не было обнаружено дефектов, связанных с расхождением ожидаемых и фактических результатов его шагов.

Заблокирован (blocked) – данное состояние означает, что по какой-то причине выполнение тест-кейса невозможно (как правило, такой причиной является наличие дефекта, не позволяющего реализовать некий пользовательский сценарий).

Закрыт (closed) – очень редкий случай, т.к. тест-кейс, как правило, оставляют в состояниях «провален / пройден успешно / заблокирован / пропущен». В некоторых системах управления тест-кейс переводят в данное состояние, чтобы подчеркнуть тот факт, что на данной итерации тестирования все действия с ним завершены.

Требует доработки (not ready) – как видно из схемы, в это состояние (или из него) тест-кейс может быть переведён в любой момент времени, если в нём будет обнаружена ошибка, если изменятся требования, по которым он был написан, или наступит иная ситуация, не позволяющая считать тест-кейс пригодным для выполнения и перевода в иные состояния.

Структура Тестовых Случаев (Test Case Structure)

Каждый тест кейс должен иметь 3 части:

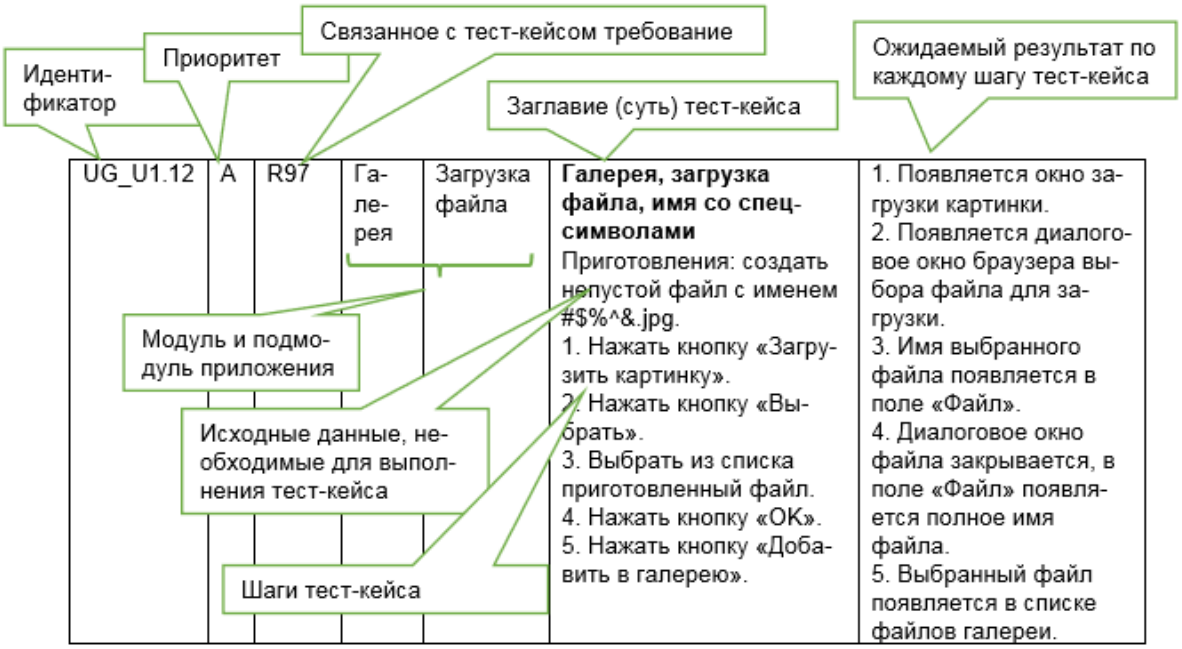
PreConditions	Список действий, которые приводят систему к состоянию пригодному для проведения основной проверки . Об этом говорит сайт https://intellect.icu . Либо список условий, выполнение которых говорит о том, что система находится в пригодном для проведения основного теста состояния.
Test Case Description	Список действий, переводящих систему из одного состояния в другое, для получения результата, на основании которого можно сделать вывод о удовлетворении реализации, поставленным требованиям
PostConditions	Список действий, переводящих систему в первоначальное состояние (состояние до проведения теста - initial state)

Примечание: **Post Conditions** не является обязательной частью. Это скорее всего - правило хорошего тона: "намусорил - убери за собой". Это особенно актуально при автоматизированном тестировании, когда за один прогон можно наполнить базу данных сотней или даже тысячей некорректных документов.

Пример содержания тест кейса:

Проверка отображения страницы		
Действие	Ожидаемый результат	Результат теста
Открыть страницу "Вход в систему"	– Окно «Вход в систему» открыто – Название окна - Вход в систему – Логотип компании отображается в правом верхнем углу – На форме 2 поля - Имя и Пароль – Кнопка Вход доступна – Ссылка «забыл пароль» – доступна	...

Основные поля и элементы тест кейса



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 (например, UR216_S12_DB_Neg).

Приоритет (priority) показывает важность тест-кейса. Он может быть выражен буквами (A, B, C, D, E), 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 трёх до пяти.

Приоритет тест-кейса может коррелировать с:

1) 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 некоей форс-мажорной ситуации, не позволяющей выполнить все запланированные тест-кейсы.

Связанное с тест-кейсом требование (requirement) показывает то основное требование, проверке выполнения которого посвящён тест-кейс (основное, поскольку один тест-кейс может затрагивать несколько требований). Наличие этого поля улучшает такое свойство тест-кейса, как прослеживаемость. Заполнение этого поля является не обязательным.

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). Как правило, иерархия модулей и подмодулей создаётся 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. В реальности проще всего отталкиваться от архитектуры и дизайна приложения.

Заглавие (суть) тест-кейса (title) призвано упростить и ускорить понимание основной идеи (цели) тест-кейса без обращения к его остальным атрибутам.

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

– состояние базы данных;

– состояние файловой системы и её объектов;

– состояние серверов и сетевой инфраструктуры.

Шаги тест-кейса (steps) описывают последовательность действий, которые необходимо реализовать в процессе выполнения тест-кейса.

Общие рекомендации по написанию шагов тест-кейса:

1. Начинайте с понятного и очевидного места, не пишите лишних начальных шагов (запуск приложения, очевидные операции с интерфейсом и т.п.).

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

3. Если вы пишете на русском языке, то используйте безличную форму (например, «открыть», «ввести», «добавить» вместо «откройте», «введите», «добавьте»), в английском языке не надо использовать частицу «to» (т.е. «запустить приложение» будет «start application», не «to start application»).

4. Соотносите 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 чётко прописанных значений и указаний.

5. Ссылайтесь на предыдущие шаги и их диапазоны для сокращения объёма текста (например, «повторить шаги 3–5 со значением…»).

6. Пишите шаги последовательно, без условных конструкций вида «если… то…».

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:

– Описывайте поведение системы так, чтобы исключить субъективное толкование (например, «приложение работает верно» — плохо, «появляется окно с надписью…» — хорошо).

– Пишите ожидаемый результат по всем шагам без исключения, если у вас есть хоть малейшие сомнения в том, что результат некоего шага будет совершенно тривиальным и очевидным (если вы всё же пропускаете ожидаемый результат для какого-то тривиального действия, лучше оставить в списке ожидаемых результатов пустую строку — это облегчает восприятие).

– Пишите кратко, но не в ущерб информативности.

– Избегайте условных конструкций вида «если… то…».

Набор тест-кейсов (test case suite, test suite, test set) – совокупность тест-кейсов, выбранных с некоторой общей целью или по некоторому общему признаку.

Наборы тест-кейсов можно разделить на **свободные** (порядок выполнения тест-кейсов не важен) и **последовательные** (порядок выполнения тест-кейсов важен).

Преимущества свободных наборов:

– Тест-кейсы можно выполнять в любом удобном порядке, а также создавать «наборы внутри наборов».

– Если какой-то тест-кейс завершился ошибкой, это не повлияет на возможность выполнения других тест-кейсов.

Преимущества последовательных наборов:

– Каждый следующий в наборе тест-кейс, в качестве входного состояния приложения, получает результат работы предыдущего тест-кейса, что позволяет сильно сократить количество шагов в отдельных тест-кейсах.

– Длинные последовательности действий куда лучше имитируют работу реальных пользователей, чем отдельные «точечные» воздействия на приложение.

Подходы к составлению наборов тест-кейсов:

– На основе чек-листов.

– На основе разбиения приложения на модули и подмодули. Для каждого модуля (или его отдельных подмодулей) можно составить свой набор тест кейсов.

– По принципу проверки самых важных, менее важных и всех остальных функций приложения.

– По принципу группировки тест-кейсов для проверки некоего уровня требований или типа требований, группы требований или отдельного требования.

– По принципу частоты обнаружения тест-кейсами дефектов в приложении (например, мы видим, что некоторые тест-кейсы раз за разом завершаются неудачей, значит, мы можем объединить их в набор, условно названный «проблемные места в приложении»).

– По архитектурному принципу: наборы для проверки пользовательского интерфейса и всего уровня представления, для проверки уровня бизнес-логики, для проверки уровня данных.

– По области внутренней работы приложения, например, «тест-кейсы, затрагивающие работу с базой данных», «тест-кейсы, затрагивающие работу с файловой системой», «тест-кейсы, затрагивающие работу с сетью».

– По видам тестирования.

Внешний вид типового тест кейса