Test case

Explanation # 1-

you could think of a test case as the most fundamental part of a software testing approach’s hierarchy. It is what most people would consider “a test.”

What does a test case include? Before all, a test case has a sequence of steps (пошаговый сценарий) that need to be carried out during the test execution. However, executing a bunch of steps isn’t enough. Therefore, a test case also needs to contain some **verification** in which the expected outcome is compared to the actual result. Consequently, the result of that comparison is what will tell you whether the test has passed or not.

Explanation # 2-

последовательность шагов (step by step action ) которые нужно выполнить для того чтобы проверить работоспособность нашего продукта (приложения, сайта, программы). Т.е. это в какой то мере пошаговая инструкция что нужно зделать и какой результат мы ожидаем. Тест кейс состоит из:

1) Названия - должно отражать суть проверки, что мы хотим проверить.

2) Шаги - которые нужно выполнить чтобы зделать проверку.

3)Ожидаемый результат. У каждого шага может быть ожидаемый результат или в самом конце проверки.

Explanation # 3-

— набор входных данных, условий выполнения(шагов) и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства. Под тест-кейсом также может пониматься соответствующий документ, представляющий формальную запись тест-кейса

если у тест-кейса не указаны входные данные, условия выполнения и ожидаемые результаты, и/или не ясна цель тест-кейса — это плохой тест-кейс

**Высокоуровневый тест-кейс** (high level test case288) — тест-кейс без конкретных входных данных и ожидаемых результатов. Как правило, ограничивается общими идеями и операциями, схож по своей сути с подробно описанным пунктом чек-листа. Достаточно часто встречается в интеграционном тестировании и системном тестировании, а также на уровне дымового тестирования. Может служить отправной точкой для проведения исследовательского тестирования или для создания низкоуровневых тест-кейсов.

**Низкоуровневый тест-кейс** (low level test case289) — тест-кейс с конкретными входными данными и ожидаемыми результатами. Представляет собой «полностью готовый к выполнению» тест-кейс и вообще является наиболее классическим видом тест-кейсов. Начинающих тестировщиков чаще всего учат писать именно такие тесты, т.к. прописать все данные подробно — намного проще, чем понять, какой информацией можно пренебречь, при этом не снизив ценность тест-кейса.

**Спецификация тест-кейса** (test case specification) — документ, описывающий набор тест-кейсов (включая их цели, входные данные, условия и шаги выполнения, ожидаемые результаты) для тестируемого элемента.

**Тест-сценарий** (test scenario, test procedure specification, test script) — документ, описывающий последовательность действий по выполнению теста (также известен как «тест-скрипт»).

**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:

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

• 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).

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

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

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

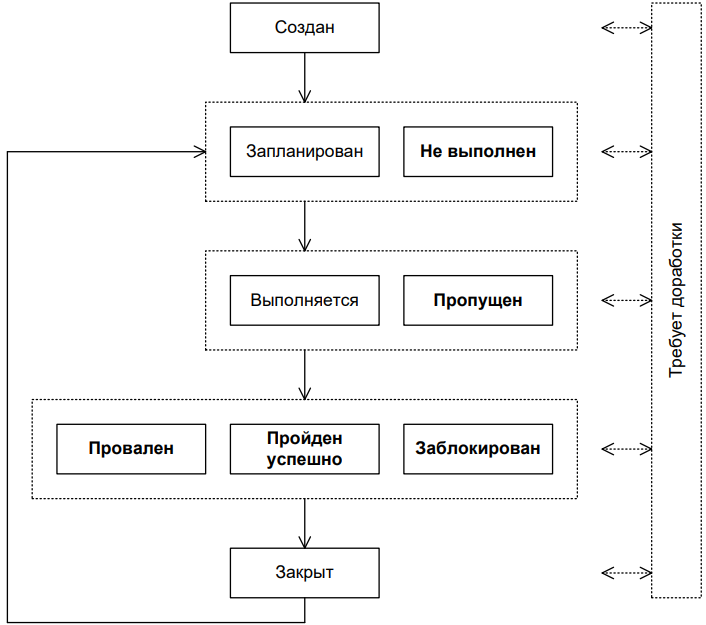
• Проводить регрессионное тестирование и повторное тестирование (которые без тест-кейсов было бы вообще невозможно выполнить).

• Повышать качество требований (мы это уже рассматривали: написание чеклистов и тест-кейсов — хорошая техника тестирования требований).

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

**Жизненный цикл тест-кейса**

В отличие от отчёта о дефекте, у которого есть полноценный развитый жизненный цикл, для тест-кейса речь скорее идёт о наборе состояний (см. рисунок 2.4.a), в которых он может находиться (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.

**• Выполняется** (work in progress) — если тест-кейс требует длительного времени на выполнение, 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).

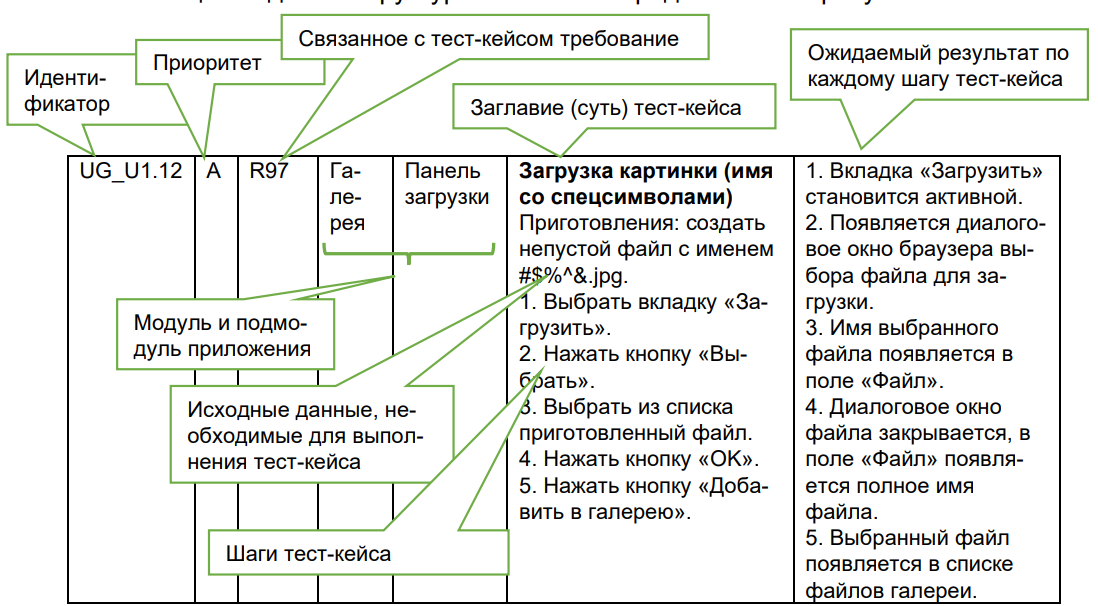
**• Закрыт** (closed) — очень редкий случай, т.к. тест-кейс, как правило, оставляют в состояниях «провален / пройден успешно / заблокирован / пропущен». В данное состояние в некоторых системах управления тест-кейс переводят, чтобы подчеркнуть тот факт, что на данной итерации тестирования все действия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

**• Требует доработки** (not ready) — как видно из схемы, в это состояние (и 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.

для тест-кейса описанное выше носит общий рекомендательный характер, рассматривается скорее как разрозненный набор состояний (а не строгий жизненный цикл) и может сильно отличаться в разных компаниях

**Атрибуты** (поля) **тест-кейса**

Как уже было сказано выше, термин «тест-кейс» может относиться к формальной записи тест-кейса в виде технического документа. Эта запись имеет общепринятую структуру, компоненты которой называются атрибутами (полями) тесткейса. В зависимости от инструмента управления тест-кейсами внешний вид их записи может немного отличаться, могут быть добавлены или убраны отдельные поля, но концепция остаётся неизменной.



**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:

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

• потенциальной важностью дефекта{179} , на поиск которого направлен тесткейс;

• 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 — потому, что один тест-кейс может затрагивать несколько требований). Наличие этого поля улучшает такое свойство тест-кейса, как прослеживаемость{143} .

Частые вопросы, связанные с заполнением этого поля, таковы:

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

• 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 всё же направлены на проверку строго одного требования, и для них этот вопрос также неактуален.

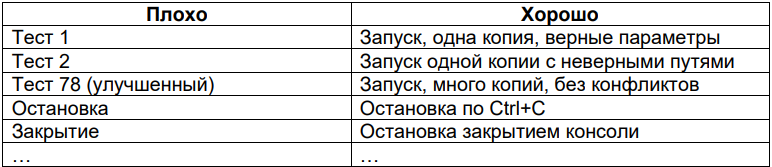
**Модуль и подмодуль приложения** (module and submodule) указывают на части приложения, к которым относится тест-кейс, и позволяют лучше понять его цель. Идея деления приложения на модули и подмодули проистекает из того, что в сложных системах практически невозможно охватить взглядом весь проект целиком, 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 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.

Модули и подмодули можно выделять на основе графического интерфейса пользователя (крупные области и элементы внутри них), на основе решаемых приложением задач и подзадач и т.д. Главное, чтобы эта логика была одинаковым образом применена ко всему приложению.

Частая ошибка! Модуль и подмодуль приложения — это НЕ действия, это именно структурные части, «куски» приложения. В заблуждение вас могут ввести такие названия, как, например, «печать, настройка принтера» (но здесь имеются в виду именно часть приложения, отвечающее за печать, часть приложения отвечающее за настройку принтера (и названы они отглагольными существительными), а не процесс печати или настройки принтера).

Наличие полей «Модуль» и «Подмодуль» улучшает такое свойство тесткейса, как прослеживаемость

**Заглавие** (суть) тест-кейса (title) призвано упростить и ускорить понимание основной идеи (цели) тест-кейса без обращения к его остальным атрибутам. Именно это поле является наиболее информативным при просмотре списка тесткейсов. Сравните.



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

• Информативность.

• Хотя бы относительная уникальность (чтобы не путать разные тест-кейсы).

Внимание! Частая ошибка! Если инструмент управления тест-кейсами не требует писать заглавие, его всё равно надо писать. Тест-кейсы без заглавий превращаются в мешанину информации, использование которой сопряжено с колоссальными и совершенно бессмысленными затратами. И ещё одна небольшая мысль, которая может помочь лучше формулировать заглавия. В дословном переводе с английского «test case» обозначает «тестовый случай (ситуация)». Так вот, заглавие как раз и описывает этот случай (ситуацию), т.е. что происходит в тест-кейсе, какую ситуацию он проверяет.

**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. То, что описывается в этом поле, готовится БЕЗ использования тестируемого приложения, и таким образом, если здесь возникают проблемы, нельзя писать отчёт о дефекте в приложении. Эта мысль очень и очень важна, потому поясним её простым жизненным примером. Представьте, что вы дегустируете конфеты. В поле «исходные данные» можно прописать «купить конфеты таких-то сортов в таком-то количестве». Если таких конфет нет в продаже, если закрыт магазин, если не хватило денег и т.д. — всё это НЕ проблемы вкуса конфет, и нельзя писать отчёт о дефекте конфет вида «конфеты невкусные потому, что закрыт магазин».

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

Общие рекомендации по написанию шагов таковы:

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

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

• если вы пишете на русском языке, 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»);

• 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;

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

• пишите шаги последовательно, без условных конструкций вида «если… то…».

Внимание! Частая ошибка! Категорически запрещено ссылаться на шаги из других тест-кейсов и другие тест-кейсы целиком: если те, другие тесткейсы будут изменены или удалены, ваш тест-кейс начнёт ссылаться на неверные данные или в пустоту, а если в процессе выполнения те, другие тест-кейсы или шаги приведут к возникновению ошибки, вы не сможете закончить выполнение вашего тест-кейса.

**Ожидаемые результаты** (expected results) по каждому шагу тест-кейса описывают ожидаемую 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…».

Внимание! Частая ошибка! В ожидаемых результатах ВСЕГДА описывается КОРРЕКТНАЯ работа приложения. Нет и не может быть ожидаемого результата в виде «приложение вызывает ошибку в операционной системе и аварийно завершается с потерей всех пользовательских данных». При этом корректная работа приложения вполне может предполагать отображение сообщений о неверных действиях пользователя или неких критических ситуациях. Так, сообщение «Невозможно сохранить файл по указанному пути: на целевом носителе недостаточно свободного места» — это не ошибка приложения, это его совершенно нормальная и правильная работа. Ошибкой приложения (в этой же ситуации) было бы отсутствие такого сообщения, и/или повреждение, или потеря записываемых данных

**Свойства качественных тест-кейсов**

**Правильный технический язык, точность и единообразие формулировок**. Это свойство в равной мере относится и к требованиям, и к тест-кейсам, и к отчётам о дефектах — к любой документации. Основные идеи уже были описаны (см. главу «Атрибуты (поля) тест-кейсов»{124} ), а из самого общего и важного напомним и добавим:

• пишите лаконично, но понятно;

• используйте безличную форму глаголов (например, «открыть» вместо «откройте»);

• обязательно указывайте точные имена и технически верные названия элементов приложения;

• не объясняйте базовые принципы работы с компьютером (предполагается, что ваши коллеги знают, что такое, например, «пункт меню» и как с ним работать);

• везде называйте одни и те же вещи одинаково (например, нельзя в одном тест-кейсе некий режим работы приложения назвать «графическое представление», а в другом тот же режим — «визуальное отображение», т.к. многие люди могут подумать, что речь идёт о разных вещах);

• следуйте принятому на проекте стандарту оформления и написания тесткейсов (иногда такие стандарты могут быть весьма жёсткими: вплоть до регламентации того, названия каких элементов должны быть приведены в двойных кавычках, а каких — в одинарных).

**Баланс между специфичностью и общностью**. Тест-кейс считается тем более специфичным, чем более детально в нём расписаны конкретные действия, конкретные значения и т.д., т.е. чем в нём больше конкретики. Соответственно, тест-кейс считается тем более общим, чем в нём меньше конкретики.

сами по себе специфичность или общность тесткейса не являются чем-то плохим, но резкий перекос в ту или иную сторону снижает качество тест-кейса. Т.е. не надо впадать в крайности и старатся расписывать всё прям до мелких мелочей, и в тоже время не стоит слишком обобщать, одним словом.

**Баланс между простотой и сложностью**. Здесь не существует академических определений, но принято считать, что простой тест-кейс оперирует одним объектом (или в нём явно виден главный объект), а также содержит небольшое количество тривиальных действий; сложный тест-кейс оперирует несколькими равноправными объектами и содержит много нетривиальных действий.

Преимущества простых тест-кейсов:

• их можно быстро прочесть, легко понять и выполнить;

• они понятны начинающим тестировщикам и новым людям на проекте;

• они делают наличие ошибки очевидным (как правило, в них предполагается выполнение повседневных тривиальных действий, проблемы с которыми видны невооружённым взглядом и не вызывают дискуссий);

• они упрощают начальную диагностику ошибки, т.к. сужают круг поиска.

Преимущества сложных тест-кейсов:

• при взаимодействии многих объектов повышается вероятность возникновения ошибки;

• пользователи, как правило, используют сложные сценарии, а потому сложные тесты более полноценно эмулируют работу пользователей;

• программисты редко проверяют такие сложные случаи (и они совершенно не обязаны это делать).

Как и в случае специфичности и общности, сами по себе простота или сложность тест-кейсов не являются чем-то плохим (более того — рекомендуется начинать разработку и выполнение тест-кейсов с простых, а затем переходить ко всё более и более сложным), однако излишняя простота и излишняя сложность также снижают качество тест-кейса.

**«Показательность»** (высокая вероятность обнаружения ошибки). Начиная с уровня тестирования критического пути, можно утверждать, что тест-кейс является тем более хорошим, чем он более показателен (с большей вероятностью обнаруживает ошибку). Именно поэтому мы считаем непригодными слишком простые тест-кейсы — они непоказательны.

**Последовательность в достижении цели**. Суть этого свойства выражается в том, что все действия в тест-кейсе направлены на следование единой логике и достижение единой цели и не содержат никаких отклонений.

**Отсутствие лишних действий**. Чаще всего это свойство подразумевает, что не нужно в шагах тест-кейса долго и по пунктам расписывать то, что можно заменить одной фразой:

**Неизбыточность по отношению к другим тест-кейсам.** В процессе создания множества тест-кейсов очень легко оказаться в ситуации, когда два и более тест-кейса фактически выполняют одни и те же проверки, преследуют одни и те же цели, направлены на поиск одних и тех же проблем. Способ минимизации количества таких тест-кейсов подробно описан в главе «Виды и направления тестирования{67}» (см. такие техники тестирования, как использование классов эквивалентности{94} и граничных условий{95} ). Если вы обнаруживаете несколько тест-кейсов, дублирующих задачи друг друга, лучше всего или удалить все, кроме одного, самого показательного, или перед удалением остальных на их основе доработать этот выбранный самый показательный тест-кейс.

**Демонстративность** (способность демонстрировать обнаруженную ошибку очевидным образом). Ожидаемые результаты должны быть подобраны и сформулированы таким образом, чтобы любое отклонение от них сразу же бросалось в глаза и становилось очевидным, что произошла ошибка.

**Прослеживаемость**. Из содержащейся в качественном тест-кейсе информации должно быть понятно, какую часть приложения, какие функции и какие требования он проверяет. Частично это свойство достигается через заполнение соответствующих полей тест-кейса{124} («Ссылка на требование», «Модуль», «Подмодуль»), но и сама логика тест-кейса играет не последнюю роль, т.к. в случае серьёзных нарушений этого свойства можно долго с удивлением смотреть, например, на какое требование ссылается тест-кейс, и пытаться понять, как же они друг с другом связаны.

**Возможность повторного использования.** Это свойство редко выполняется для низкоуровневых тест-кейсов{121} , но при создании высокоуровневых тесткейсов{120} можно добиться таких формулировок, при которых:

• тест-кейс будет пригодным к использованию с различными настройками тестируемого приложения и в различных тестовых окружениях;

• тест-кейс практически без изменений можно будет использовать для тестирования аналогичной функциональности в других проектах или других областях приложения.

Примером тест-кейса, который тяжело использовать повторно, может являться практически любой тест-кейс с высокой специфичностью.

**Повторяемость**. Тест-кейс должен быть сформулирован таким образом, чтобы при многократном повторении он показывал одинаковые результаты. Это свойство можно разделить на два подпункта:

• во-первых, даже общие формулировки, допускающие разные варианты выполнения тест-кейса, должны очерчивать соответствующие явные границы (например: «ввести какое-нибудь число» — плохо, «ввести целое число в диапазоне от -273 до +500 включительно» — хорошо);

• действия (шаги) тест-кейса по возможности не должны приводить к необратимым (или сложно обратимым) последствиям (например: удалению данных, нарушению конфигурации окружения и т.д.) — не стоит включать в тест-кейс такие «разрушительные действия», если они не продиктованы явным образом целью тест-кейса; если же цель тест-кейса обязывает нас к выполнению таких действий, в самом тест-кейсе должно быть описание действий по восстановлению исходного состояния приложения (данных, окружения).

**Соответствие принятым шаблонам оформления и традициям**. С шаблонами оформления, как правило, проблем не возникает: они строго определены имеющимся образцом или вообще экранной формой инструментального средства управления тест-кейсами. Что же касается традиций, то они отличаются даже в разных командах в рамках одной компании, и тут невозможно дать иного совета, кроме как «почитайте уже готовые тест-кейсы перед тем как писать свои»

В тест кейсах не указывается фактический результат.