**Набор тест-кейсов** - это тест сьют (**это не чек лист**)

(test suite, test set) — совокупность тест-кейсов, выбранных с некоторой общей целью или по некоторому общему признаку. Иногда в такой совокупности результаты завершения одного тест-кейса становятся входным состоянием приложения для следующего тест-кейса.

test suite is a collection or container of many test cases related to each other. Grouping your test cases into adequately named suites makes it easier to manage them.

Множества отдельных тест-кейсов, крайне неудобно (более того, это ошибка!) каждый раз писать в каждом тест-кейсе одни и те же приготовления и повторять одни и те же начальные шаги. Намного удобнее объединить несколько тест-кейсов в набор или последовательность. И здесь мы приходим к классификации наборов тест-кейсов. В общем случае наборы тест-кейсов можно разделить на:

1) свободные (порядок выполнения тест-кейсов не важен)

Преимущества:

• 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

2) последовательные (порядок выполнения тест-кейсов важен).

Преимущества:

• 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.

**Пользовательские сценарии** (сценарии использования) К отдельному подвиду последовательных наборов тест-кейсов (или даже неоформленных идей тест-кейсов, таких, как пункты чек-листа) можно отнести пользовательские сценарии (или сценарии использования), представляющие собой цепочки действий, выполняемых пользователем в определённой ситуации для достижения определённой цели.

ПРИМЕР:

Допустим, пользователь хочет распечатать табличку на дверь кабинета с текстом «Идёт работа, не стучать!» Для этого ему нужно:

1) Запустить текстовый редактор.

2) Создать новый документ (если редактор не делает это самостоятельно).

3) Набрать в документе текст.

4) Отформатировать текст должным образом.

5) Отправить документ на печать.

6) Сохранить документ (спорно, но допустим).

7) Закрыть текстовый редактор.

Вот мы и получили пользовательский сценарий, пункты которого могут стать основой для шагов тест-кейса или целого набора отдельных тест-кейсов. Сценарии могут быть достаточно длинными и сложными, могут содержать внутри себя циклы и условные ветвления, но при всём этом они обладают рядом весьма интересных преимуществ:

• Сценарии показывают реальные и понятные примеры использования продукта (в отличие от обширных чек-листов, где смысл отдельных пунктов может теряться).

• Сценарии понятны конечным пользователям и хорошо подходят для обсуждения и совместного улучшения.

• Сценарии и их части легче оценивать с точки зрения важности, чем отдельные пункты (особенно низкоуровневых) требований.

• Сценарии отлично показывают недоработки в требованиях (если становится непонятно, что делать в том или ином пункте сценария, — с требованиями явно что-то не то).

• В предельном случае (нехватка времени и прочие форс-мажоры) сценарии можно даже не прописывать подробно, а просто именовать — и само наименование уже подскажет опытному специалисту, что делать.

**Принципы построения наборов тест-кейсов** Теперь — о самом главном: как формировать наборы тест-кейсов. Правильный ответ звучит очень кратко: логично. И это не шутка. Единственная задача наборов — повысить эффективность тестирования за счёт ускорения и упрощения выполнения тест-кейсов, увеличения глубины исследования некоей области приложения или функциональности, следования типичным пользовательским сценариям или удобной последовательности выполнения тест-кейсов и т.д. Набор тест-кейсов всегда создаётся с какой-то целью, на основе какой-то логики, и по этим же принципам в набор включаются тесты, обладающие подходящими свойствами. Если же говорить о наиболее типичных подходах к составлению наборов тест-кейсов, то можно обозначить следующее:

• На основе чек-листов, каждый пункт чеклиста может превратиться в несколько тест-кейсов — и вот мы получаем готовый набор

• На основе разбиения приложения на модули и подмодули{125} . Для каждого модуля (или его отдельных подмодулей) можно составить свой набор тесткейсов.

• По принципу проверки самых важных, менее важных и всех остальных функций приложения (именно по этому принципу мы составляли примеры чек-листов{116} ).

• По принципу группировки тест-кейсов для проверки некоего уровня требований или типа требований{39} , 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 (см. «многоуровневая архитектура»150 самостоятельно): 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», и т.д. • По видам тестирования

Не нужно заучивать этот список. Это всего лишь примеры — грубо говоря, «первое, что приходит в голову». Важен принцип: если вы видите, что выполнение некоторых тест-кейсов в виде набора принесёт вам пользу, создавайте такой набор.

Примечание: без хороших инструментальных средств управления тест-кейсами работать с наборами тест-кейсов крайне тяжело, т.к. приходится самостоятельно следить за приготовлениями, «недостающими шагами», изолированностью или обобщённостью, свободностью или последовательностью и т.д.