2.4. Чек-листы, тест-кейсы, наборы тест-кейсов

2.4.1. Чек-листы

Как легко можно понять из предыдущих глав, тестировщику приходится работать с огромным количеством информации, выбирать из множества вариантов решения задач и изобретать новые. В процессе этой деятельности объективно невозможно удержать в голове все мысли, а потому продумывание и разработку тест-кейсов рекомендуется выполнять с использованием так называемых «чек-листов».

Чек-лист (checklist[276]) — набор идей [тест-кейсов]. Последнее слово не зря взято в скобки, т.к. в общем случае чек-лист — это просто набор идей: идей по тестированию, идей по разработке, идей по планированию и управлению — **любых** идей.

Чек-лист чаще всего представляет собой обычный и привычный нам список, который может быть:

- Списком, в котором последовательность пунктов не имеет значения (например, список значений некоего поля).
- Списком, в котором последовательность пунктов важна (например, шаги в краткой инструкции).
- Структурированным (многоуровневым) списком (вне зависимости от учёта последовательности пунктов), что позволяет отразить иерархию идей.

Важно понять, что нет и не может быть никаких запретов и ограничений при разработке чек-листов — главное, чтобы они помогали в работе. Иногда чек-листы могут даже выражаться графически (например, с использованием ментальных карт[277] или концепт-карт[278]), хотя традиционно их составляют в виде многоуровневых списков.

Поскольку в разных проектах встречаются однотипные задачи, хорошо продуманные и аккуратно оформленные чек-листы могут использоваться повторно, чем достигается экономия сил и времени.

Для того чтобы чек-лист был действительно полезным инструментом, он должен обладать рядом важных свойств.

Логичность. Чек-лист пишется не «просто так», а на основе целей и для того, чтобы помочь в достижении этих целей. К сожалению, одной из самых частых и опасных ошибок при составлении чек-листа является превращение его в свалку мыслей, которые никак не связаны друг с другом.

Последовательность и структурированность. Со структурированностью всё достаточно просто — она достигается за счёт оформления чек-листа в виде многоуровневого списка. Что до последовательности, то даже в том случае, когда пункты чек-листа не описывают цепочку действий, человеку всё равно удобнее воспринимать информацию в виде неких небольших групп идей, переход между которыми является понятным и очевидным (например, сначала можно прописать идеи простых позитивных тест-кейсов{76}, потом идеи простых негативных тест-кейсов, потом постепенно повышать сложность тест-кейсов, но не стоит писать эти идеи вперемешку).

---

[276] Понятие «чек-листа» не завязано на тестирование как таковое — это совершенно универсальная техника, которая может применяться в любой без исключения области жизни. В русском языке вне контекста информационных технологий чаще используется понятное и привычное слово «список» (например, «список покупок», «список дел» и т.д.), но в тестировании прижилась калькированная с английского версия — «чек-лист».

[277] «Mind map», Wikipedia. [http://en.wikipedia.org/wiki/Mind_map]

[278] «Concept map», Wikipedia. [http://en.wikipedia.org/wiki/Concept_map]

Полнота и неизбыточность. Чек-лист должен представлять собой аккуратную «сухую выжимку» идей, в которых нет дублирования (часто появляется из-за разных формулировок одной и той же идеи), и в то же время ничто важное не упущено.

Правильно создавать и оформлять чек-листы также помогает восприятие их не только как хранилища наборов идей, но как «требования для составления тест-кейсов». Эта мысль приводит к пересмотру и переосмыслению свойств качественных требований (см. главу «Свойства качественных требований»{40}) в применении к чек-листам.

Задание 2.4.a: перечитайте главу «Свойства качественных требований»{40} и подумайте, какие свойства качественных требований можно также считать и свойствами качественных чек-листов.

Итак, рассмотрим процесс создания чек-листа. В главе «Пример анализа и тестирования требований»{48} приведён пример итоговой версии требований{54}, который мы и будем использовать.

Поскольку мы не можем сразу «протестировать всё приложение» (это слишком большая задача, чтобы решить её одним махом), нам уже сейчас нужно выбрать некую логику построения чек-листов — да, их будет несколько (в итоге их можно будет структурированно объединить в один, но это не обязательно).

Типичными вариантами такой логики является создание отдельных чек-листов для:

- различных уровней функционального тестирования{73};
- отдельных частей (модулей и подмодулей{118}) приложения;
- отдельных требований, групп требований, уровней и типов требований{35};
- типичных пользовательских сценариев{137};
- частей или функций приложения, наиболее подверженных рискам.

Этот список можно расширять и дополнять, можно комбинировать его пункты, получая, например, чек-листы для проверки наиболее типичных сценариев, затрагивающих некую часть приложения.

Чтобы проиллюстрировать принципы построения чек-листов, мы воспользуемся логикой разбиения функций приложения по степени их важности (классификацию по убыванию степени важности функций приложения{73}) на три категории:

- Базовые функции, без которых существование приложения теряет смысл (т.е. самые важные — то, ради чего приложение вообще создавалось), или нарушение работы которых создаёт объективные серьёзные проблемы для среды исполнения. (См. «Дымовое тестирование»{73}).
- Функции, востребованные большинством пользователей в их повседневной работе. (См. «Тестирование критического пути»{74}).
- Остальные функции (разнообразные «мелочи», проблемы с которыми не сильно повлияют на ценность приложения для конечного пользователя). (См. «Расширенное тестирование»{75}).

Функции, без которых существование приложения теряет смысл

Сначала приведём целиком весь чек-лист для дымового тестирования, а потом разберём его подробнее.

- Конфигурирование и запуск.
- Обработка файлов:

		Форматы входных файлов		
		TXT	HTML	MD
Кодировки входных файлов	WIN1251	+	+	+
	CP866	+	+	+
	KOI8R	+	+	+

- Остановка.

Да, и всё. Здесь перечислены все ключевые функции приложения.

Конфигурирование и запуск. Если приложение невозможно настроить для работы в пользовательской среде, оно бесполезно. Если приложение не запускается, оно бесполезно. Если на стадии запуска возникают проблемы, они могут негативно отразиться на функционировании приложения и потому также заслуживают пристального внимания.

Примечание: в нашем примере мы столкнулись со скорее нетипичным случаем — приложение конфигурируется параметрами командной строки, а потому разделить операции «конфигурирования» и «запуска» не представляется возможным; в реальной жизни для подавляющего большинства приложений эти операции выполняются раздельно.

Обработка файлов. Ради этого приложение и разрабатывалось, потому здесь даже на стадии создания чек-листа мы не поленились создать матрицу, отражающую все возможные комбинации допустимых форматов и допустимых кодировок входных файлов, чтобы ничего не забыть и подчеркнуть важность соответствующих проверок.

Остановка. С точки зрения пользователя эта функция может не казаться столь уж важной, но остановка (и запуск) любого приложения связаны с большим количеством системных операций, проблемы с которыми могут привести к множеству серьёзных последствий (вплоть до невозможности повторного запуска приложения или нарушения работы операционной системы).

Функции, востребованные большинством пользователей

Следующим шагом мы будем выполнять проверку того, как приложение ведёт себя в обычной повседневной жизни, пока не затрагивая экзотические ситуации. Очень частым вопросом является допустимость дублирования проверок на разных уровнях функционального тестирования{73} — можно ли так делать. Одновременно и «нет», и «да». «Нет» в том смысле, что не допускается (не имеет смысла) повторение тех же проверок, которые только что были выполнены. «Да» в том смысле, что любую проверку можно детализировать и снабдить дополнительными деталями.

- Конфигурирование и запуск:
  - С верными параметрами
    - Значения SOURCE_DIR, DESTINATION_DIR, LOG_FILE_NAME указаны и содержат пробелы и кириллические символы (повторить для форматов путей в Windows и *nix файловых системах, обратить внимание на имена логических дисков и разделители

имён каталогов (“/” и “\”)).
        - Значение LOG_FILE_NAME не указано.
    - Без параметров.
    - С недостаточным количеством параметров.
    - С неверными параметрами:
        - Недопустимый путь SOURCE_DIR.
        - Недопустимый путь DESTINATION_DIR.
        - Недопустимое имя LOG_FILE_NAME.
        - DESTINATION_DIR находится внутри SOURCE_DIR.
        - Значения DESTINATION_DIR и SOURCE_DIR совпадают.
- Обработка файлов:
    - Разные форматы, кодировки и размеры:

		Форматы входных файлов		
		TXT	HTML	MD
Кодировки входных файлов	WIN1251	100 КБ	50 МБ	10 МБ
	CP866	10 МБ	100 КБ	50 МБ
	KOI8R	50 МБ	10 МБ	100 КБ
	Любая	0 байт		
	Любая	50 МБ + 1 Б	50 МБ + 1 Б	50 МБ + 1 Б
	-	Любой недопустимый формат		
	Любая	Повреждения в допустимом формате		

    - Недоступные входные файлы:
        - Нет прав доступа.
        - Файл открыт и заблокирован.
        - Файл с атрибутом «только для чтения».
- Остановка:
    - Закрытием окна консоли.
- Журнал работы приложения:
    - Автоматическое создание (при отсутствии журнала).
    - Продолжение (дополнение журнала) при повторных запусках.
- Производительность:
    - Элементарный тест с грубой оценкой.

Обратите внимание, что чек-лист может содержать не только «предельно сжатые тезисы», но и вполне развёрнутые комментарии, если это необходимо — лучше пояснить суть идеи подробнее, чем потом гадать, что же имелось в виду.

Также обратите внимание, что многие пункты чек-листа носят весьма высокоуровневый характер, и это нормально. Например, «повреждения в допустимом формате» (см. матрицу с кодировками, форматами и размерами) звучит расплывчато, но этот недочёт будет устранён уже на уровне полноценных тест-кейсов.

Остальные функции и особые сценарии

Пришло время обратить внимание на разнообразные мелочи и хитрые нюансы, проблемы с которыми едва ли сильно озаботят пользователя, но формально всё же будут считать ошибками.

- Конфигурирование и запуск:
    - Значения SOURCE_DIR, DESTINATION_DIR, LOG_FILE_NAME:
        - В разных стилях (Windows-пути + *nix-пути) — одно в одном стиле, другое — в другом.
        - С использованием UNC-имён.
        - LOG_FILE_NAME внутри SOURCE_DIR.

        - LOG_FILE_NAME внутри DESTINATION_DIR.
    - Размер LOG_FILE_NAME на момент запуска:
        - 2–4 ГБ.
        - 4+ ГБ.
    - Запуск двух и более копий приложения с:
        - Одинаковыми параметрами SOURCE_DIR, DESTINATION_DIR, LOG_FILE_NAME.
        - Одинаковыми SOURCE_DIR и LOG_FILE_NAME, но разными DESTINATION_DIR.
        - Одинаковыми DESTINATION_DIR и LOG_FILE_NAME, но разными SOURCE_DIR.
- Обработка файлов:
    - Файл верного формата, в котором текст представлен в двух и более поддерживаемых кодировках одновременно.
    - Размер входного файла:
        - 2–4 ГБ.
        - 4+ ГБ.

Задание 2.4.b: возможно, вам захотелось что-то изменить в приведённых выше чек-листах — это совершенно нормально и справедливо: нет и не может быть некоего «единственно верного идеального чек-листа», и ваши идеи вполне имеют право на существование, потому составьте свой собственный чек-лист или отметьте недостатки, которые вы заметили в приведённом выше чек-листе.

Как мы увидим в следующей главе, создание качественного тест-кейса может потребовать длительной кропотливой и достаточно монотонной работы, которая при этом не требует от опытного тестировщика сильных интеллектуальных усилий, а потому переключение между работой над чек-листами (творческая составляющая) и расписыванием их в тест-кейсы (механическая составляющая) позволяет разнообразить рабочий процесс и снизить утомляемость. Хотя, конечно, написание сложных и притом качественных тест-кейсов может оказаться ничуть не менее творческой работой, чем продумывание чек-листов.