Лабораторная работа №6

Разработка плана и проведение интеграционного тестирования

Цель работы: Освоить разработку плана и научится проводить интеграционное тестирование, научится составлять чек-лист.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ:

У клиента есть проблема: поступающие в огромном количестве его сотрудникам текстовые файлы приходят в разных кодировках, и сотрудники тратят много времени на перекодирование («ручной подбор кодировки»). Соответственно, он хотел бы иметь инструмент, позволяющий автоматически приводить кодировки всех текстовых файлов к некоей одной. Необходимо разработать проект с кодовым названием «Конвертер файлов».

После составления списка свойств качественных требований, характерных для чек-листов, необходимо определить свойства чек-листа, чтобы он был универсальным (т. е. чтобы его можно было использовать повторно на другом проекте).

Чек-лист — набор идей [тест-кейсов].

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 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.

Чек-листы могут выражаться графически (например, с использованием ментальных карт или концепт-карт), традиционно их составляют в виде многоуровневых списков. 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 на основе целей и для того, чтобы помочь в достижении этих целей. Одной из самых частых и опасных ошибок при составлении чек-листа является превращение его в никак не связанные друг с другом фразы.

Последовательность и структурированность. Структурированность достигается за счёт оформления чек-листа в виде многоуровневого списка. Последовательность, в случае, когда пункты чек-листа не описывают цепочку действий, удобнее воспринимать информацию в виде небольших групп идей, переход между которыми является понятным и очевидным.

Полнота и неизбыточность. Чек-лист должен представлять собой аккуратную «сухую выжимку» идей, в которых нет дублирования, и в то же время ничто важное не упущено. Правильно создавать и оформлять чек-листы помогает восприятие их не только как хранилища наборов идей, но и как «требования для составления тест-кейсов».

ПРОЦЕСС СОЗДАНИЯ ЧЕК-ЛИСТА

Большое приложение невозможно протестировать «все и сразу». Решением такой задачи является создание отдельных чек-листов для:

• различных уровней функционального тестирования;

• отдельных частей (модулей и подмодулей) приложения;

• отдельных требований, групп требований, уровней и типов требований;

• типичных пользовательских сценариев;

• 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 часть приложения.

Для иллюстрации принципов построения чек-листов, воспользуемся логикой разбиения функций приложения по степени их важности 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.

• Функции, востребованные большинством пользователей в их повседневной работе.

• Остальные функции (разнообразные «мелочи», проблемы с которыми не сильно повлияют на ценность приложения для конечного пользователя).

ФУНКЦИИ, БЕЗ КОТОРЫХ СУЩЕСТВОВАНИЕ ПРИЛОЖЕНИЯ ТЕРЯЕТ СМЫСЛ

Чек-лист для минимального набора тестов на явные ошибки

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

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).

Здесь перечислены все ключевые функции приложения.

ФУНКЦИИ, ВОСТРЕБОВАННЫЕ БОЛЬШИНСТВОМ ПОЛЬЗОВАТЕЛЕЙ

Следующий шаг выполнять проверку того, как приложение ведёт себя в повседневной жизни, пока не затрагивая экзотические ситуации. Очень частым вопросом является допустимость дублирования проверок на разных уровнях функционального тестирования. 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:
  - º 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.
  - º Без параметров.
  - º С недостаточным количеством параметров.
  - º 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:
  - º 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		

º Недоступные входные файлы:
- ■ Нет прав доступа.
- ■ Файл открыт и заблокирован.
- ■ Файл с атрибутом «только для чтения».

• Остановка: º Закрытием окна консоли.
• Журнал работы приложения:
º Автоматическое создание (при отсутствии журнала).
º Продолжение (дополнение журнала) при повторных запусках.
• Производительность:
º Элементарный тест с грубой оценкой.

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 многие пункты чек-листа носят весьма высокоуровневый характер, и это нормально. Например, «повреждения в допустимом формате» звучит расплывчато, но этот недочёт будет устранён уже на уровне полноценных тест-кейсов.

ОСТАЛЬНЫЕ ФУНКЦИИ И ОСОБЫЕ СЦЕНАРИИ

Нюансы, проблемы с которые не озаботят пользователя, но формально будут считать ошибками.

• Конфигурирование и запуск:
º 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.

º Размер LOG_FILE_NAME на момент запуска:
- ■ 2–4 ГБ.
- ■ 4+ ГБ.

º 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:
º Файл верного формата, в котором текст представлен в двух и более поддерживаемых кодировках одновременно.
º Размер входного файла:
- ■ 2–4 ГБ.
- ■ 4+ ГБ.

Задание. Если 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.

На чём базировались ваши правки существующего чек-листа? Можете ли вы аргументированно доказать преимущества предложенного вами варианта?

Пример чек-листа к приложению File Searcher по требованиям из лабораторной работы 1.

Проверка	Результат	Коментарии
1.Запуск прилажения FS	ок	
2. Отображение главной формы	ок	
2.1. Отображение меню	ок	
2.1.1. Main	ок	
2.1.2. Language	ок	
2.1.3. About	ок	
2.2. Отображение поле "Where to search?"	ок	
2.3. Отображение поле "What to search?"	ок	
2.4. Отображение окна результатов поиска	ок	
2.5. Отображение кнопки "Search"	ок	
3. Проверяем поле "Where to search?" на валидные значения	ок	
3.1. D:	ок	
3.2. D:\;C:\	ок	
4. Проверяем поле "Where to search?" на не валидные значения	ок	
4.1. пустую строку	ок	
4.2. не существующий диск	ок	
4.3. Текст	bug	
5. Проверяем поле "What to search"	ок	
5.1. Аудио файлы	ок	
5.2. Видео файлы	ок	
5.3. Файлы пакета MS office	ок	
6. Проверка кнопки поиска файлов	ок	
6.1. D:\ ; Аудио файлы	ок	
6.2. C:\ ; Видео файлы	ок	
6.3. C:\program\ ; Файлы пакета MS office	ок	
7. Проверка окна результатов поиска	ок	
7.1. Name	ок	
7.2. Path	ок	
7.3. Size	ок	
7.4. DataTime	ок	
8. Проверка панели "Сейчас проверяется"	ок	
9. Проверка закрытия приложения	ок	

Используя текстовый редактор Word, составить чек-лист по требованиям к приложению составленным в лабораторной работе 3.

Отчет по лабораторной работе должен содержать оформленный чек-лист по требованиям к приложению составленным в лабораторной работе 3 и ответы на контрольные вопросы.

Контрольные вопросы

1) Объективные причины усиления внимания к процессу тестирования
2) Основные критерии завершенности тестирования
3) Внешнее качество программного обеспечения
4) Внутреннее качество программного обеспечения
5) Назовите наиболее важные виды тестировании ПС
6) Сформулируйте цель интеграционного тестирования
7) Краткая характеристика интеграционного тестирования ПС
8) Плюсы и минусы интеграционного тестирования
9) Перечислите и опишите виды интеграционного тестирования