



# Сертифицированный тестировщик

## Программа обучения Продвинутого уровня для тест-аналитика (СТАЛ-ТА)

v3.1.2 RU-1

International Software Testing Qualifications Board



Уведомление об авторских правах © International Software Testing Qualifications Board (далее просто ISTQB®)

ISTQB® является зарегистрированной торговой маркой International Software Testing Qualifications Board.

Авторские права © 2025 авторы перевода v3.1.2: Михаил Костецкий (руководитель группы), Александр Александров (редактор), Ксения Кондакова, Андрей Конушин, Елена Костина, Юрий Левченко, Дмитрий Фомин, Павел Шариков.

Авторские права © 2021-2022 авторы обновления v3.1.0, 3.1.1 и 3.1.2: Wim Decoutere, István Forgács, Matthias Hamburg, Adam Roman, Jan Sabak, Marc-Florian Wendland.

Авторские права © 2019 авторы обновления 2019: Graham Bath, Judy McKay, Jan Sabak, Erik van Veenendaal

Авторские права © 2012 авторы обновления: Judy McKay, Mike Smith, Erik van Veenendaal

Все права защищены. Авторы передают свои права International Software Testing Qualifications Board (далее ISTQB®). Авторы (владельцы авторских прав в данный момент) и ISTQB® (как будущий владелец авторских прав) договорились о следующих условиях использования:

- Выдержки из этого документа для некоммерческого использования могут быть скопированы, если указан источник. Любая аккредитованная обучающая компания может использовать эту программу обучения в качестве основы для учебного курса, если авторы и ISTQB® указаны как источник и владельцы авторских прав программы обучения и при условии, что в любой рекламе таких курсов данная программа обучения может быть упомянута только после письменного уведомления об аккредитации материалов тренингов коллегами, признанными ISTQB®.
- Любое частное лицо или группа частных лиц может использовать программу как основу для статей, книг или других производных письменных материалов, если авторы и ISTQB® упомянуты как источник и владельцы авторских прав программы.
- Любое другое использование этой программы запрещено без предварительного письменного одобрения ISTQB®.
- Любая коллегия, признанная ISTQB®, может переводить эту программу обучения при условии, что она воспроизводит вышеупомянутое Уведомление об авторских правах в переведенной версии программы.

## История изменений

Версия	Дата	Содержание
2012 v2.0	2012-10-19	Первая версия отдельной программы обучения AL-TA
2019 v3.0	2019-10-19	Значительное обновление с общей переработкой и сокращением области применения
v3.1.0	2021-03-03	Незначительное обновление с переработанным разделом 3.2.3 и различными улучшениями формулировок
v3.1.1	2021-05-15	Ошибки: Уведомление об авторских правах было адаптировано к текущим стандартам ISTQB®.
v3.1.2	2022-01-31	Ошибки: Исправлены незначительные дефекты форматирования, грамматики и формулировок
v3.1.2 RU-1	2025-03-03	Перевод на русский язык

## Содержание

История изменений .....	3
Содержание .....	4
Благодарности .....	7
0. Предисловие к программе обучения .....	9
0.1 Цель этого документа .....	9
0.2 Сертифицированный тестировщик программного обеспечения продвинутого уровня .....	9
0.3 Цели обучения, подлежащие проверке, и когнитивные уровни знаний .....	9
0.4 Экзамен продвинутого уровня для тест-аналитика .....	10
0.5 Требования для допуска к экзамену .....	10
0.6 Ожидаемый опыт .....	10
0.7 Аккредитация курсов .....	10
0.8 Уровень детализации программы обучения .....	10
0.9 Как организована эта программа обучения .....	11
1. Задачи тест-аналитика в процессе тестирования — 150 минут .....	12
1.1 Введение .....	13
1.2 Тестирование в жизненном цикле программного обеспечения .....	13
1.3 Анализ тестирования .....	15
1.4 Проектирование тестов .....	16
1.4.1 Низкоуровневые и высокоуровневые тестовые сценарии .....	17
1.4.2 Проектирование тестовых сценариев .....	18
1.5 Реализация тестов .....	20
1.6 Выполнение тестов .....	22
2. Задачи тест-аналитика при тестировании, основанном на рисках — 60 минут .....	23
2.1 Введение .....	24
2.2 Определение рисков .....	24
2.3 Оценка рисков .....	25
2.4 Смягчение рисков .....	26
2.4.1 Приоритизация тестов .....	26
2.4.2 Корректировка тестирования для будущих циклов тестирования .....	27
3. Методы тестирования - 630 минут .....	28
3.1 Введение .....	29
3.2 Методы тестирования черного ящика .....	29

3.2.1 Эквивалентное разбиение .....	30
3.2.2 Анализ граничных значений.....	31
3.2.3 Тестирование таблицы решений.....	32
3.2.4 Тестирование таблицы переходов .....	34
3.2.5 Метод деревьев классификации .....	36
3.2.6 Попарное тестирование .....	37
3.2.7 Тестирование по сценариям использования.....	39
3.2.8 Комбинирование методов .....	40
3.3 Методы тестирования на основе опыта. ....	41
3.3.1 Предположение об ошибках .....	42
3.3.2 Тестирование на основе чек-листов .....	43
3.3.3 Исследовательское тестирование .....	44
3.3.4 Метод создания тестовых сценариев на основе дефектов .....	45
3.4 Применение наиболее подходящего метода.....	46
4. Тестирование характеристик качества программного обеспечения – 180 минут.....	47
4.1 Введение .....	48
4.2 Характеристики качества для тестирования бизнес-домена .....	49
4.2.1 Тестирование функциональной корректности .....	50
4.2.2 Тестирование функциональной пригодности.....	50
4.2.3 Тестирование функциональной полноты .....	50
4.2.4 Тестирование возможности взаимодействия.....	50
4.2.5 Оценка практичности.....	51
4.2.6 Тестирование переносимости .....	54
5. Рецензирование – 120 минут .....	56
5.1 Введение .....	57
5.2 Использование чек-листов в рецензировании.....	57
5.2.1 Рецензирование требований .....	57
5.2.2 Рецензирование пользовательских историй.....	58
5.2.3 Составление чек-листов.....	59
6. Инструменты тестирования и автоматизация – 90 минут .....	60
6.1 Введение .....	61
6.2 Тестирование на основе ключевых слов.....	61
6.3 Типы инструментов тестирования .....	62
6.3.1 Инструменты для проектирования тестов .....	62
6.3.2 Инструменты подготовки тестовых данных.....	62
6.3.3 Автоматизированные инструменты выполнения тестов.....	63

7.	Ссылки.....	64
7.1	Стандарты .....	64
7.2	Документы ISTQB® и IREB .....	64
7.3	Книги и статьи .....	64
7.4	Другие ссылки .....	65
8.	Приложение А.....	67

## Благодарности

Перевод версии документа 3.1.2 выполнен рабочей группой АНО «Коллегия экспертов по качеству программного обеспечения» (Russian Software Testing Qualifications Board, RSTQB): Михаил Костецкий (руководитель группы), Александр Александров (редактор), Ксения Кондакова, Андрей Конушин, Елена Костина, Юрий Левченко, Дмитрий Фомин, Павел Шариков.

Этот документ был подготовлен рабочей группой Программы обучения Тест-аналитик Продвинутого уровня ISTQB: Mette Bruhn-Pedersen (руководитель рабочей группы); Matthias Hamburg (владелец продукта); Wim Decoutere, István Forgács, Adam Roman, Jan Sabak, Marc-Florian Wendland (авторы).

Рабочая группа благодарит Paul Weymouth и Richard Green за их техническую редактуру, Gary Mogyoródi за проверки соответствия Глоссарию и национальные коллегии за их замечания, относящиеся к версии 2019 данной Программы обучения и предложенные изменения.

В редактировании и предоставлении замечаний принимали участие:

Gery Ágncsz, Armin Born, Chenyifan, Klaudia Dussa-Zieger, Chen Geng (Kevin), Istvan Gercsák, Richard Green, Ole Chr. Hansen, Zsolt Hargitai, Andreas Hetz, Tobias Horn, Joan Killeen, Attila Kovacs, Rik Marselis, Marton Matyas, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Pe'er, Palma Polyak, Nishan Portoyan, Meile Posthuma, Stuart Reid, Murian Song, Péter Sótér, Lucjan Stapp, Benjamin Timmermans, Chris van Bael, Stephanie van Dijk, Paul Weymouth.

Документ был официально выпущен ISTQB® 23 февраля 2021 года.

Последующие формальные, грамматические и фразеологические улучшения были предложены Tal Pe'er, Stuart Reid, Marc-Florian Wendland и Matthias Hamburg, реализованы и опубликованы в обновлениях 3.1.1 и 3.1.2.

Версия 2019 данного документа была разработана основной командой рабочей группы Продвинутого уровня ISTQB: Graham Bath, Judy McKay, Mike Smith

Следующие специалисты принимали участие в рецензировании, комментариях и обсуждении версии 2019 этой программы обучения:

Laura Albert, Markus Beck, Henriett Braunné Bokor, Francisca Cano Ortiz, Guo Chaonian, Wim Decoutere, Milena Donato, Klaudia Dussa-Zieger, Melinda Eckrich-Brajer, Péter Földházi Jr, David Frei, Chen Geng, Matthias Hamburg, Zsolt Hargitai, Zhai Hongbao, Tobias Horn, Ágota Horváth, Beata Karpinska, Attila Kovács, József Kreisz, Dietrich Leimsner, Ren Liang, Claire Lohr, Ramit Manohar Kaul, Rik Marselis, Marton Matyas, Don Mills, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Peer, Pálma Polyák, Meile Posthuma, Lloyd Roden, Adam Roman, Abhishek Sharma, Péter Sótér, Lucjan Stapp, Andrea Szabó, Jan te Kock, Benjamin Timmermans, Chris Van Bael, Erik van Veenendaal, Jan Versmissen, Carsten Weise, Robert Werkhoven, Paul Weymouth.

Версия 2012 данного документа была разработана основной командой рабочей подгруппы Тест-аналитика Продвинутого уровня ISTQB: Judy McKay (руководитель), Mike Smith, Erik van Veenendaal.

На момент завершения работы над Программой обучения Продвинутого уровня рабочая группа имела следующий состав (в алфавитном порядке)

Graham Bath, Rex Black, Maria Clara Choucair, Debra Friedenberg, Bernard Homès (заместитель руководителя), Paul Jorgensen, Judy McKay, Jamie Mitchell, Thomas Mueller, Klaus Olsen, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Mike Smith (руководитель), Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto.

## Сертифицированный тестировщик Продвинутый уровень для тест-аналитика

---



Следующие специалисты принимали участие рецензировании, комментариях и обсуждении версии 2012 этой программы обучения:

Graham Bath, Arne Becher, Rex Black, Piet de Roo, Frans Dijkman, Mats Grindal, Kobi Halperin, Bernard Homès, Maria Jönsson, Junfei Ma, Eli Margolin, Rik Marselis, Don Mills, Gary Mogyorodi, Stefan Mohacsi, Reto Mueller, Thomas Mueller, Ingvar Nordstrom, Tal Pe'er, Raluca Madalina Popescu, Stuart Reid, Jan Sabak, Hans Schaefer, Marco Sogliani, Yaron Tsubery, Hans Weiberg, Paul Weymouth, Chris van Bael, Jurian van der Laar, Stephanie van Dijk, Erik van Veenendaal, Wenqiang Zheng, Debi Zylbermann.

## 0. Предисловие к программе обучения

### 0.1 Цель этого документа

Эта программа представляет основу международной сертификации на квалификацию продвинутого уровня для тест-аналитика. ISTQB® предоставляет эту программу:

1. Национальным коллегиям для перевода на национальный язык и аккредитации организаторов обучения. Национальные коллегии могут адаптировать программу обучения к особенностям местных языков и определить ссылки для адаптации к местным публикациям.
2. Экзаменационным комиссиям для формирования экзаменационных вопросов на местном языке, адаптированные к целям обучения каждого курса.
3. Организаторам обучения для разработки материалов обучения и определения соответствующих методов обучения.
4. Кандидатам на получение сертификатов для подготовки к экзамену (в рамках учебного курса или самостоятельно).
5. Международному сообществу разработчиков ПО и систем для продвижения профессии тестировщика ПО и систем, и использования как основы для книг и статей.

ISTQB® может разрешить другим организациям использовать этот учебный план для других целей при условии, что они получают предварительное письменное разрешение.

### 0.2 Сертифицированный тестировщик программного обеспечения продвинутого уровня

Базовая квалификация продвинутого уровня состоит из трех отдельных учебных программ, относящихся к следующим ролям:

- Руководитель тестирования
- Тест-аналитик
- Технический тест-аналитик

Обзор продвинутого уровня ISTQB® 2019 представлен в отдельном документе [ISTQB\_AL\_OVIEW], который включает следующую информацию:

- Бизнес-результаты для каждой программы обучения
- Матрица, показывающая трассируемость между бизнес-результатами и целями обучения
- Краткое изложение каждой программы обучения
- Взаимосвязи между программами обучения

### 0.3 Цели обучения, подлежащие проверке, и когнитивные уровни знаний

Цели обучения поддерживают ожидаемые бизнес-результаты и используются для создания экзамена, необходимого для получения сертификации продвинутого уровня для тест-аналитика.

Уровни знаний для конкретных целей обучения на уровнях K2, K3 и K4 представлены в начале каждой главы и классифицируются следующим образом:

- K2: Понять
- K3: Применить
- K4: Проанализировать

Следует запомнить определения всех терминов, перечисленных в качестве ключевых слов непосредственно под заголовками глав (K1), даже если они явно не указаны в целях обучения.

## 0.4 Экзамен продвинутого уровня для тест-аналитика

Сертификационный экзамен Продвинутого уровня для тест-аналитика основан на данной программе обучения. Ответы на экзаменационные вопросы могут потребовать использования материала, основанного более чем на одной главе этой программы обучения. Все разделы программы обучения поддаются экзаменационной проверке, кроме Предисловия и Приложений. Стандарты, книги и другие программы обучения ISTQB® включены в качестве ссылок, но их содержание не проверяется на экзамене, за исключением того, что обобщенно в этой программе обучения из этих стандартов, книг и других программ обучения ISTQB®.

Формат экзамена — тест с несколькими вариантами ответов. Экзамен состоит из 40 вопросов. Для успешной сдачи экзамена необходимо набрать не менее 65% от общего количества баллов.

Экзамены могут быть сданы в рамках аккредитованного учебного курса или самостоятельно (например, в экзаменационном центре или на открытом экзамене). Завершение аккредитованного учебного курса не является обязательным требованием для сдачи экзамена.

## 0.5 Требования для допуска к экзамену

Критерием допуска к сдаче экзамена на получение сертификата Продвинутого уровня для тест-аналитика является наличие у кандидата сертификата Базового уровня ISTQB®.

## 0.6 Ожидаемый опыт

Ни одна из целей обучения продвинутого уровня для тест-аналитика не предполагает наличия определенного опыта.

## 0.7 Аккредитация курсов

Коллегия-член ISTQB® может аккредитовать обучающие организации, учебные материалы которых соответствуют этой программе обучения. Обучающие организации должны получить руководство по аккредитации от коллегии-члена или органа, который проводит аккредитацию. Аккредитованный курс признается соответствующим этой программе, и в рамках курса разрешается сдавать экзамен ISTQB®.

## 0.8 Уровень детализации программы обучения

Уровень детализации этой программы обучения позволяет проводить согласованные на международном уровне курсы и экзамены. Для достижения этой цели программа обучения состоит из:

- Общих целей обучения, описывающих намерения продвинутого уровня для тест-аналитика.
- Списка терминов, которые кандидаты должны запомнить.

- Целей обучения для каждой области знаний с описанием когнитивных результатов обучения, которые должны быть достигнуты.
- Описание ключевых понятий, включая ссылки на такие источники, как литература или стандарты.

Содержание программы не является описанием всей области знаний по тестированию ПО; оно отражает уровень детализации, который необходимо охватить в учебных курсах Продвинутого уровня. Учебный план фокусируется на материале, который может быть применен ко всем проектам по разработке программного обеспечения, включая гибкую методологию разработки программного обеспечения. Программа обучения не содержит специфических целей обучения, относящихся к какому-либо конкретному жизненному циклу разработки программного обеспечения, но рассматривает, как эти концепции применяются в гибкой методологии разработки программного обеспечения, других типах итеративных и инкрементных жизненных циклов, а также в последовательных жизненных циклах.

## 0.9 Как организована эта программа обучения

Программа обучения содержит 6 глав с подлежащим изучению содержанием. В заголовке верхнего уровня каждой главы указано время обучения для этой главы. Ниже уровня главы время не указывается. Для аккредитованных учебных курсов программа требует минимум 20 часов 30 минут обучения, распределенных по шести главам следующим образом:

- Глава 1: Задачи тест-аналитика в процессе тестирования (150 минут)
- Глава 2: Задачи тест-аналитика при тестировании, основанном на рисках (60 минут)
- Глава 3: Методы тестирования (630 минут)
- Глава 4: Тестирование характеристик качества программного обеспечения (180 минут)
- Глава 5: Рецензирование (120 минут)
- Глава 6: Инструменты тестирования и автоматизация (90 минут)

# 1. Задачи тест-аналитика в процессе тестирования — 150 минут

## Ключевые слова

Анализ тестирования, базис тестирования, выполнение теста, критерии выхода, набор тестов, проектирование теста, процедура тестирования, реализация теста, расписание выполнения тестов, тест, тестовые данные, тестовое условие, тестовый сценарий высокого уровня, тестовый сценарий низкого уровня.

## Цели обучения главы «Задачи тест-аналитика в процессе тестирования»

### 1.1 Введение

Цели обучения отсутствуют

### 1.2 Тестирование в жизненном цикле программного обеспечения

ТА-1.2.1 (K2) Объяснить, как и почему время и уровень вовлеченности тест-аналитика варьируются при работе с различными моделями жизненного цикла разработки программного обеспечения

### 1.3 Анализ тестирования

ТА-1.3.1 (K2) Резюмировать соответствующие задачи тест-аналитика при проведении аналитических активностей

### 1.4 Проектирование тестов

ТА-1.4.1 (K2) Объяснить, почему тестовые условия должны быть понятны заинтересованным сторонам

ТА-1.4.2 (K4) Для данного сценария проекта выбрать соответствующий уровень проектирования тестовых сценариев (высокого уровня или низкого уровня).

ТА-1.4.3 (K2) Объяснить проблемы, которые следует учитывать при проектировании тестовых сценариев

### 1.5 Реализация тестов

ТА-1.5.1 (K2) Резюмировать соответствующие задачи тест-аналитика при проведении активностей по реализации тестов

### 1.6 Выполнение тестов

ТА-1.6.1 (K2) Резюмировать соответствующие задачи тест-аналитика при проведении активностей по выполнению тестов

## 1.1 Введение

В программе обучения ISTQB® базового уровня процесс тестирования описан как состоящий из следующих активностей:

- Планирование тестирования
- Мониторинг и контроль тестирования
- Анализ тестирования
- Проектирование тестов
- Реализация тестов
- Выполнение тестов
- Завершение тестирования

В текущей программе обучения продвинутого уровня Тест-аналитик более глубоко рассматриваются активности, которые имеют непосредственное отношение к тест-аналитику. Это позволяет дополнительно улучшить процесс тестирования для соответствия различным моделям жизненного цикла разработки программного обеспечения (SDLC).

Определение подходящих тестов, их разработка и реализация и их последующее выполнение являются основными зонами ответственности тест-аналитика. Безусловно, важно понимать и остальные этапы процесса тестирования, но большая часть работы тест-аналитика обычно сосредоточена на следующих активностях:

- Анализ тестирования
- Проектирование тестов
- Реализация тестов
- Выполнение тестов

Другие активности процесса тестирования подробно описаны в программе Базового уровня и не нуждаются в дальнейшем рассмотрении на этом уровне.

## 1.2 Тестирование в жизненном цикле программного обеспечения

При определении стратегии тестирования следует учитывать весь жизненный цикл разработки программного обеспечения. Степень вовлечения тест-аналитика в разнообразных жизненных циклах разработки программного обеспечения различна; степень участия, продолжительность участия, доступная информация и ожидания также могут сильно варьироваться. Тест-аналитик должен знать, какие типы информации следует предоставлять другим, связанным с ним, организационным ролям, таким как:

- Руководству по проектированию и управлению требованиями – обратную связь о рецензировании требований
- Проектному руководству – план работ
- Управлению конфигурацией и изменениями – результаты тестирования верификации сборки, информация о версии
- Разработке программного обеспечения – информирование о найденных дефектах
- Сопровождению программного обеспечения – отчеты о дефектах, эффективность устранения дефектов, подтверждающее тестирование
- Технической поддержке – подробная документация по обходным путям и известным проблемам
- Подготовка технической документации (например, спецификации проектирования базы данных, документация тестового окружения) – внесение актуальных изменений в эти документы, а также техническое рецензирование документов

Этапы тестирования должны быть согласованы с выбранным жизненным циклом разработки программного обеспечения, характер которого может быть последовательным, итеративным, инкрементным или гибридным. Например, в последовательной V-модели процесс тестирования, применяемый к уровню системного тестирования, может быть выстроен следующим образом:

- Планирование тестирования системы происходит одновременно с планированием проекта, а мониторинг и контроль тестирования продолжается до его завершения. Это повлияет на план работ, предоставляемый тест-аналитиком для целей управления проектом
- Анализ и проектирование системных тестов зависит от таких документов, как спецификация системных требований, системная и архитектурная (высокоуровневая) проектная спецификация, а также компонентная (низкоуровневая) проектная спецификация
- Реализация тестового окружения системы может начаться во время проектирования системы, хотя основная ее часть обычно происходит одновременно с кодированием и компонентным тестированием, причем работа по реализации тестирования системы часто растягивается на несколько дней, вплоть до начала выполнения тестирования системы
- Выполнение системного теста начинается, когда критерии входа выполнены или, если необходимо, отменены, что обычно означает, что по крайней мере компонентное тестирование, а часто и тестирование интеграции компонентов выполнили свои критерии выхода. Выполнение системного тестирования продолжается до тех пор, пока не будут выполнены критерии выхода системного тестирования
- Действия по завершению системного тестирования происходят после выполнения критериев выхода системного тестирования.

Итерационные и инкрементальные модели могут не следовать одному и тому же порядку действий и могут исключать некоторые из действий. Например, итерационная модель может использовать сокращенный набор активностей по тестированию для каждой итерации. Анализ, проектирование, реализация и выполнение тестов могут проводиться для каждой итерации, тогда как планирование высокого уровня осуществляется в начале проекта, а задачи завершения - в конце.

В гибкой методологии разработки программного обеспечения обычно используется менее формализованный процесс и более тесные рабочие взаимоотношения с заинтересованными сторонами проекта, что позволяет легко вносить изменения в проект. В данном случае не может быть четкого определения роли тест-аналитика. Ведется менее полная тестовая документация, а общение происходит короче и чаще.

Гибкая разработка программного обеспечения подразумевает тестирование с самого начала. Тестирование начинается с самого начала разработки продукта, как только разработчики выполняют свою первоначальную работу по архитектуре и проектированию. Рецензирование может быть неформальным и проводится постоянно по мере развития программного обеспечения. Вовлеченность ожидается на протяжении всего проекта, а задачи тест-аналитика будут выполняться всей командой.

Итеративные и инкрементальные модели варьируются от гибкой разработки программного обеспечения, где существует ожидание изменений по мере развития требований заказчика, до гибридных моделей, например, итеративная/инкрементальная разработка в сочетании с последовательной моделью разработки (водопадная модель, V-модель). В таких гибридных моделях тест-аналитики должны быть задействованы в определенных элементах последовательных мероприятий, как планирование и проектирование, а затем переходить к более интерактивной роли в течение итеративно-инкрементальных мероприятий.

Независимо от используемого жизненного цикла разработки программного обеспечения, тест-аналитики должны понимать, чего от них ожидают и когда они должны быть вовлечены в процесс разработки. Тест-аналитики вносят эффективный вклад в качество программного обеспечения, адаптируя свою деятельность и момент привлечения к конкретному жизненному циклу разработки программного обеспечения, вместо того, чтобы придерживаться заранее определенной ролевой модели.

### 1.3 Анализ тестирования

Во время планирования тестирования определяется объем тестирования на проекте. Во время анализа тестирования тест-аналитики используют эти данные для:

- Анализа базиса тестирования
- Выявления дефектов различных типов в базисе тестирования
- Определения и приоритизации тестовых условий и функций, подлежащих тестированию
- Обеспечения двунаправленной трассируемости между каждым элементом базиса тестирования и соответствующим тестовыми условиями
- Выполнения задач, связанных с тестированием, основанным на рисках (см. главу 2)

Для того чтобы тест-аналитики могли успешно приступить к анализу тестов, необходимо соблюдение следующих критериев входа:

- Существует комплекс знаний (например, требования, пользовательские истории), описывающий объект тестирования и служащий базисом тестирования (см. [ISTQB\_FL\_SYL] разделы 1.4.2 и 2.2 или перечень других возможных источников базиса тестирования).
- Базис тестирования прошел рецензирование с приемлемыми результатами и был обновлен по мере необходимости после рецензирования. Обратите внимание, что, если должны быть описаны высокоуровневые тестовые сценарии (см. раздел 1.4.1), то базис тестирования еще может не быть полностью определен. В гибкой разработке программного обеспечения этот цикл рецензирования будет итерационным, поскольку пользовательские истории уточняются в начале каждой итерации.
- Есть утвержденный бюджет и график для выполнения оставшихся задач по тестированию данного объекта

Тестовые условия обычно определяются путем анализа базиса тестирования в сочетании с целями тестирования (определенными при планировании тестирования). В некоторых ситуациях, когда документация может быть старой или отсутствовать, тестовые условия могут быть определены путем обсуждения с соответствующими заинтересованными сторонами (например, на встречах или во время планирования итераций). В гибкой разработке программного обеспечения критерии приемки, определяемые как часть пользовательских историй, часто используются в качестве основы для разработки теста.

Хотя тестовые условия обычно специфичны для конкретного элемента тестирования, существуют некоторые рекомендации для тест-аналитика.

- Обычно рекомендуется определять условия тестирования на разных уровнях детализации. Первоначально определяются условия высокого уровня для определения общих целей тестирования, например, «функциональность экрана X». Затем определяются более детальные условия, как основа для конкретных тестовых случаев, например, «экран X отклоняет номер счета, который на одну цифру меньше правильной длины». Использование такого иерархического подхода к определению

тестовых условий может помочь обеспечить достаточное покрытие для высокоуровневых элементов. Этот подход также позволяет тест-аналитику начать работу над определением высокоуровневых тестовых условий для пользовательских историй, которые еще не были доработаны.

- Если продуктовые риски были определены, то тестовые условия, которые будут необходимы для устранения каждого продуктового риска, должны быть определены и прослежены до этого элемента риска.

Применение методов тестирования (как определено в стратегии тестирования и/или плане тестирования) может быть полезным в задачах, связанных с тест-анализом, и может быть использовано для достижения следующих целей:

- Определение тестовых условий
- Снижение вероятности пропуска важных тестовых условий
- Определение более точных и достоверных тестовых условий

После определения и уточнения тестовых условий можно провести рецензирование этих условий с заинтересованными сторонами, чтобы убедиться, что требования поняты однозначно и что тестирование соответствует целям проекта.

После проведения анализа тестирования для заданной области (например, конкретной функции), тест-аналитику следует знать, какие конкретные тесты должны быть разработаны для этой области.

## 1.4 Проектирование тестов

Придерживаясь заранее определенного объема тестирования, тест-аналитик проектирует тесты для их последующей реализации и выполнения. Проектирование тестов включает следующие действия:

- Определение того, в какой области тестирования целесообразны низкоуровневые или высокоуровневые тестовые сценарии
- Определение метода (методов) тестирования, который позволит достичь необходимого покрытия. Методы, которые могут быть использованы, определяются во время планирования тестирования.
- Использование методов тестирования для разработки тестовых сценариев и тестовых наборов, которые охватывают выявленные тестовые условия
- Определение необходимых тестовых данных для обеспечения реализации тестовых условий и тестовых сценариев
- Проектирование тестового окружения и определение любой необходимой инфраструктуры, включая инструменты
- Фиксация двунаправленной трассировки (например, между базисом тестирования, тестовыми условиями и тестовыми сценариями)

Критерии приоритизации, определенные в ходе анализа рисков и планирования тестирования, должны применяться на протяжении всего процесса, от анализа и проектирования до внедрения и выполнения.

В зависимости от типов проектируемых тестов, одним из критериев входа проектирования тестов может быть наличие инструментов, которые будут использоваться во время проектирования.

При проектировании тестов тест-аналитик должен учитывать, по крайней мере, следующее:

- Некоторые элементы тестирования лучше рассматривать, определяя только тестовые условия, а не углубляясь в определение сценариев тестирования, которые дают последовательность инструкций, необходимых для выполнения теста. В этом случае следует определить тестовые условия, чтобы использовать их в качестве руководства для тестирования без сценариев тестирования.
- Критерии прохождения / непрохождения должны быть четко определены.
- Тесты должны быть разработаны так, чтобы быть понятными другим тестировщикам, а не только автору. Если автор не является исполнителем теста, другим тестировщикам потребуется прочитать и понять ранее указанные тесты, чтобы понять цели тестирования и относительную значимость теста.
- Тесты также должны быть понятными для других заинтересованных сторон, таких как разработчики (которые могут рецензировать тесты) и аудиторы (которым, возможно, придется утверждать тесты).
- Тесты должны охватывать все виды взаимодействия с объектом тестирования и не должны ограничиваться взаимодействием с пользовательским интерфейсом. Они также могут включать, например, взаимодействие с другими системами, технические или физические события. (Более подробную информацию см. в [IREB\_CPFE]).
- Тесты должны быть разработаны для проверки интерфейсов между различными элементами объекта тестирования, а также поведения самих элементов.
- Усилия по разработке тестов должны быть приоритезированы и сбалансированы, чтобы соответствовать уровню риска и ценности для бизнеса.

#### 1.4.1 Низкоуровневые и высокоуровневые тестовые сценарии

Одной из задач тест-аналитика является определение необходимого уровня проработки тестовых сценариев для конкретного случая. Низкоуровневые и высокоуровневые тестовые сценарии рассматриваются в [ISTQB\_FL\_SYL]. Ниже приведены некоторые преимущества и недостатки их использования:

Низкоуровневые тестовые сценарии обеспечивают следующие преимущества:

- Неопытные тестировщики могут полагаться на подробную информацию, предоставленную в рамках проекта. Тестовые сценарии низкого уровня предоставляют всю конкретную информацию и процедуры, необходимые тестировщику для выполнения тестового сценария (включая любые требования к данным) и проверки фактических результатов.
- Тесты могут быть повторно проведены разными тестировщиками, при этом должны быть получены одинаковые результаты.
- Могут быть выявлены неочевидные дефекты в базисе тестирования.
- При необходимости уровень детализации допускает независимую верификацию тестов, например, аудит.
- Время, затрачиваемое на автоматизированную реализацию тестовых сценариев, может быть сокращено.

Низкоуровневые тестовые сценарии имеют следующие недостатки:

- Они могут потребовать значительных усилий для создания и сопровождения.
- Они, как правило, ограничивают изобретательность тестировщика во время выполнения.
- Они требуют, чтобы базис тестирования был четко определен.

- Их трассировка с тестовыми условиями может потребовать больше усилий, чем для высокоуровневых тестовых сценариев.

Высокоуровневые тестовые сценарии обеспечивают следующие преимущества:

- Они дают методические рекомендации по тому, что должно быть протестировано, и позволяют тест-аналитику варьировать фактические данные или даже процедуру выполнения теста.
- Они могут обеспечить лучшее покрытие рисков, чем тестовые сценарии низкого уровня, поскольку при каждом выполнении они будут несколько отличаться.
- Они могут быть определены на ранней стадии процесса разработки требований.
- Они опираются на опыт тест-аналитика в тестировании и объекте тестирования при выполнении теста.
- Они могут быть определены, когда не требуется подробная и формальная документация.
- Они лучше подходят для повторного использования в различных циклах тестирования, когда можно использовать различные тестовые данные.

Высокоуровневые тестовые сценарии имеют следующие недостатки:

- Они менее воспроизводимы, что затрудняет верификацию. Это связано с отсутствием в них подробного описания, которое можно найти в тестовых сценариях низкого уровня.
- Для их выполнения могут потребоваться более опытные тестировщики
- При автоматизации на основе высокоуровневых тестовых сценариев отсутствие деталей может привести к валидации некорректных фактических результатов или пропуску элементов, которые должны быть проверены.

Высокоуровневые тестовые сценарии могут быть использованы для разработки низкоуровневых тестовых сценариев, когда требования становятся более четкими и стабильными. В этом случае создание тестовых сценариев происходит последовательно, от высокоуровневых к низкоуровневым, и только низкоуровневые тестовые сценарии используются для выполнения.

#### 1.4.2 Проектирование тестовых сценариев

Тестовые сценарии разрабатываются путем поэтапной разработки и доработки конкретных тестовых условий с использованием методов тестирования (см. главу 3). Тестовые сценарии должны быть воспроизводимыми, проверяемыми и отслеживаемыми вплоть до базиса тестирования (например, требований).

Проектирование тестов заключается в определении следующего:

- Цель (т.е. наблюдаемая, измеримая цель выполнения теста)
- Предусловия, такие как требования к проекту или требования к тестовому окружению и планы по их поставке, состояние системы перед выполнением теста и т.д.
- Требования к тестовым данным (как входные данные для тестового сценария, так и данные, которые должны существовать в системе для выполнения тестового сценария)
- Ожидаемые результаты с четкими критериями прохождения / непрохождения теста

- Постусловия, такие как затронутые данные, состояние системы после выполнения теста, триггеры для последующей обработки и т.д.

Особую сложность может представлять определение ожидаемого результата теста. Вычислять его вручную часто утомительно и чревато ошибками; если возможно, предпочтительнее найти или создать автоматизированный тестовый оракул. При определении ожидаемого результата тестировщики имеют дело не только с выводом на экран, но и с данными и внешними условиями. Если базис тестирования четко определен, то определение правильного результата теоретически должно быть простым. Однако документация по базису тестирования может быть нечеткой, противоречивой, не охватывать ключевые области или полностью отсутствовать. В таких случаях тест-аналитик должен обладать предметной экспертизой или иметь доступ к знаниям в данной области. Кроме того, даже когда базис тестирования хорошо описан, сложные взаимодействия множественных запросов и ответов могут затруднить определение ожидаемых результатов; поэтому необходим тестовый оракул. В гибкой разработке программного обеспечения тестовым оракулом может быть владелец продукта. Выполнение тестовых сценариев без возможности определить правильность фактических результатов может иметь очень низкую добавленную ценность или пользу, часто приводя к недопустимым отчетам о тестировании или ложной уверенности в системе.

Описанные выше действия могут быть применены ко всем уровням тестирования, при различных базисах тестирования. При анализе и проектировании тестов важно помнить о целевом уровне для теста, а также о назначении теста. Это поможет определить необходимый уровень детализации, а также любые инструменты, которые могут понадобиться (например, драйверы и заглушки на уровне компонентного тестирования).

Во время разработки тестовых условий и тестовых сценариев обычно создается определенный объем документации, в результате чего получают рабочие продукты тестирования. На практике степень документирования рабочих продуктов тестирования значительно варьируется. На это может повлиять любой из следующих факторов:

- Риски проекта (что должно/не должно быть задокументировано)
- Дополнительная ценность, которую документация привносит в проект
- Стандарты, которым необходимо следовать, и/или нормативные требования, которые должны быть соблюдены
- Жизненный цикл разработки программного обеспечения или используемый подход (например, гибкая методология нацелена на «достаточность» документации)
- Требование трассируемости от базиса тестирования до анализа и проектирования тестов

В зависимости от объема тестирования, тестовый анализ и проектирование учитывают характеристики качества тестируемого объекта (объектов). Стандарт ISO 25010 [ISO25010] является полезным руководством. При тестировании аппаратных/программных систем могут применяться дополнительные характеристики.

Мероприятия по анализу и проектированию тестов могут быть улучшены за счет их связи с рецензированием и статическим анализом. Фактически, проведение тест-анализа и проектирование тестов часто являются формой статического тестирования, поскольку в ходе этой деятельности могут быть обнаружены проблемы в документах базиса тестирования. Тест-анализ и проектирование тестов на основе спецификации требований - отличный способ подготовиться к встрече по рецензированию требований. Чтение требований, чтобы использовать их для создания тестов, требует понимания требования и умения определить

способ оценки выполнения требования. Эта деятельность часто выявляет отсутствующие требования, неоднозначные требования, а также не поддающиеся тестированию, или не имеющие установленных критериев приемки. Аналогичным образом, рабочие продукты тестирования, такие как тестовые сценарии, анализ рисков и планы тестирования, могут быть подвергнуты рецензированию.

Если для тестирования требуется инфраструктура, которая труднореализуема, тест-аналитик должен определить подробные требования к тестовой инфраструктуре во время проектирования тестов. Если эти требования не будут выполнены вовремя, реализация тестов может быть не выполнена в срок из-за непредвиденных затрат времени и ресурса. Следует помнить, что инфраструктура тестирования включает не только объекты тестирования и тестовое программное обеспечение. Например, требования к инфраструктуре могут включать помещения, оборудование, персонал, программное обеспечение, инструменты, периферийные устройства, коммуникационное оборудование, авторизацию пользователей и другие элементы, необходимые для проведения испытаний.

Критерии выхода для анализа тестирования и проектирования тестов будут варьироваться в зависимости от параметров проекта, но для всех элементов, обсуждаемых в этих двух подразделах, следует рассмотреть возможность включения в установленные критерии выхода. Важно, чтобы критерии выхода были измеримыми, и чтобы вся информация, необходимая для последующих этапов, была предоставлена, и была проведена вся необходимая подготовка.

## 1.5 Реализация тестов

Реализация тестов требует готовности тестового программного обеспечения, необходимого для выполнения теста, на основе анализа и проектирования теста. Она включает следующие действия:

- Разработка процедур тестирования и, возможно, создание автоматизированных сценариев тестирования
- Организация тестовых процедур и автоматизированных тестовых сценариев (если таковые имеются) в тестовые наборы для выполнения в рамках конкретного тестового прогона
- Консультация с руководителем тестирования по поводу приоритизации тестовых сценариев и наборов тестов для выполнения
- Создание расписания выполнения тестов, включая распределение ресурсов, позволяющего начать выполнение тестов (см. [ISTQB\_FL\_SYL] Раздел 5.2.4)
- Завершение подготовки тестовых данных и тестовых окружений
- Обновление трассируемости между базисом тестирования и тестовым обеспечением, таким как тестовые условия, тестовые сценарии, тестовые процедуры, автоматизированные тестовые сценарии и тестовые наборы.

В процессе реализации тестов тест-аналитики определяют эффективный порядок выполнения тестовых сценариев и создают тестовые процедуры. Определение тестовых процедур требует тщательного выявления ограничений и зависимостей, которые могут повлиять на последовательность выполнения тестов. Тестовые процедуры документируют любые предусловия (например, загрузка тестовых данных из хранилища данных) и любые действия после выполнения (например, сброс состояния системы).

Тест-аналитики определяют процедуры тестирования и автоматизированные тестовые сценарии, которые можно сгруппировать (например, все они относятся к тестированию конкретного бизнес-процесса высокого уровня), и объединяют их в наборы тестов. Это позволяет выполнять связанные тестовые сценарии совместно.

Тест-аналитики выстраивают тестовые наборы в расписании выполнения тестов таким образом, чтобы обеспечить эффективное выполнение тестов. Если используется стратегия тестирования на основе рисков, то уровень риска будет основным фактором при определении порядка выполнения тестовых сценариев. Могут существовать и другие факторы, определяющие порядок выполнения тестовых сценариев, такие как наличие необходимого персонала, оборудования, данных и функциональности, подлежащей тестированию.

Нередко код выпускается по частям, и усилия по тестированию должны быть скоординированы с последовательностью, в которой программное обеспечение становится доступным для тестирования. Особенно в итеративных и инкрементальных моделях разработки важно, чтобы тест-аналитик тесно сотрудничал с командой разработчиков, чтобы гарантировать, что программное обеспечение будет выпущено для тестирования в порядке, обеспечивающем тестируемость.

На уровень детализации и связанную с этим сложность работы, выполняемой в ходе реализации теста, может влиять детализация тестовых условий и тестовых сценариев. В некоторых случаях применяются нормативные правила, и рабочие продукты тестирования должны предоставлять доказательства соответствия применяемым стандартам, таким как стандарт США DO-178C (в Европе ED-12C). [RTCA DO-178C/ED-12C].

Как указано выше, тестовые данные необходимы для большинства видов тестирования, и в некоторых случаях эти наборы данных могут быть довольно большими. В процессе реализации тест-аналитики создают входные данные и данные окружения для загрузки в базы данных и другие подобные хранилища. Эти данные должны соответствовать цели, чтобы обеспечить возможность обнаружения дефектов. Тест-аналитики могут также создавать данные для использования в тестировании на основе данных и в тестировании на основе ключевых слов (см. раздел 6.2), так же, как и для ручного тестирования.

Реализация тестов также связана с тестовым окружением (-ями). В ходе этой деятельности окружение(-ия) должно быть полностью настроено и проверено перед выполнением теста. Очень важно, чтобы тестовое окружение соответствовало цели, т.е. чтобы оно позволяло выявлять дефекты, присутствующие во время запланированного тестирования, работало нормально, когда сбои не происходят, и соответствовало, если требуется, производственному окружению или окружению конечного пользователя для более высоких уровней тестирования. Изменение тестового окружения может потребоваться во время выполнения теста в зависимости от непредвиденных изменений, результатов тестирования или других соображений. Если изменения окружения происходят во время выполнения, важно оценить влияние изменений на уже проведенные тесты.

Во время реализации теста тест-аналитики должны убедиться, что лица, ответственные за создание и поддержание тестового окружения, известны и доступны, и что все тестовое обеспечение и инструменты поддержки тестирования и связанные с ними процессы готовы к использованию. Это включает управление конфигурацией, управление дефектами, протоколирование и управление тестированием. Кроме того, тест-аналитики должны проверить процедуры сбора данных для оценки текущего состояния по критериям выхода и отчетности о результатах тестирования.

Целесообразно использовать сбалансированный подход к реализации тестирования, определенный в ходе планирования. Например, аналитические стратегии тестирования, основанные на рисках, часто сочетаются с реактивными стратегиями тестирования. В этом случае, некоторый процент усилий по реализации тестов выделяется на тестирование, которое не следует по заранее определенным сценариям (без сценариев).

Тестирование без сценариев не должно быть хаотичным или бесцельным, поскольку оно может быть непредсказуемым по продолжительности и покрытию и давать низкую результативность по дефектам.

Скорее, его следует проводить в рамках ограниченных по времени сессий, каждой из которых дается первоначальное направление в виде концепции тестирования, но с возможностью отклониться от предписаний концепции, если в ходе сессии обнаруживаются потенциально более продуктивные возможности тестирования. За многие годы тестировщики разработали множество методов тестирования на основе опыта, таких как атаки [Whittaker03], предположение об ошибках [Myers11] и исследовательское тестирование [Whittaker09]. Анализ тестов, проектирование тестов и реализация тестов все еще имеют место, но они происходят в основном во время выполнения тестов.

При следовании таким стратегиям реактивного тестирования результаты каждого теста влияют на анализ, проектирование и реализацию последующих тестов. Хотя эти стратегии легки и часто эффективны при поиске дефектов, они имеют некоторые недостатки, в том числе следующие:

- Требуется экспертиза тест-аналитика
- Продолжительность тестирования может быть трудно предсказуема
- Сложность в оценке покрытия
- Воспроизводимость тестов может не быть обеспечена без качественной документации или поддержки инструментов

## 1.6 Выполнение тестов

Выполнение теста проводится в соответствии с расписанием выполнения тестов и включает следующие задачи: (см. [ISTQB\_FL\_SYL])

- Выполнение ручных тестов, включая исследовательское тестирование
- Выполнение автоматизированных тестов
- Сравнение фактических результатов с ожидаемыми результатами
- Анализ аномалий для установления их вероятных причин
- Составление отчетов о дефектах на основе найденных отказов
- Протоколирование фактических результатов выполнения тестов
- Обновление трассируемости между базисом тестирования и тестовым обеспечением для учета результатов тестирования
- Выполнение регрессионных тестов

Перечисленные выше задачи по выполнению теста могут выполняться как тестировщиком, так и тест-аналитиком.

Ниже перечислены типичные дополнительные задачи, которые может выполнять тест-аналитик:

- Обнаружение скопления дефектов, которые могут указывать на необходимость более тщательного тестирования конкретной части объекта тестирования
- Внесение предложений по проведению будущих сеансов исследовательского тестирования на основе находок исследовательского тестирования
- Выявление новых рисков на основе информации, полученной при выполнении заданий по проведению тестирования
- Внесение предложений по улучшению любого из рабочих продуктов, полученных в результате деятельности по внедрению теста (например, усовершенствование процедур тестирования)

## 2. Задачи тест-аналитика при тестировании, основанном на рисках — 60 минут

### Ключевые слова

Определение рисков, риск продукта, смягчение рисков, тестирование на основе рисков.

### Цели обучения главы «Задачи тест-аналитика при тестировании, основанном на рисках»

#### Задачи тест-аналитика при тестировании, основанном на рисках

ТА-2.1.1 (К3) Для заданной ситуации принимать участие в определении рисков, проводить оценку рисков и предлагать соответствующие меры по их смягчению

## 2.1 Введение

Руководители тестирования часто несут общую ответственность за создание и управление стратегией тестирования на основе рисков. Обычно они просят привлечь тест-аналитика, чтобы обеспечить правильную реализацию подхода, основанного на рисках.

Тест-аналитики должны активно вовлекаться в задачи тестирования, основанного на рисках:

- Определение рисков
- Оценка рисков
- Смягчение рисков

Эти задачи выполняются итеративно на протяжении всего жизненного цикла разработки программного обеспечения, для противодействия возникающим рискам, изменения приоритетов, а также для регулярной оценки статуса риска и информировании о нем (более подробно см. в [vanVeenendaal12] и [Black02]). В гибкой методологии разработки программного обеспечения эти три задачи часто объединяются в так называемую сессию рисков с фокусом либо на итерации, либо на релизе.

Тест-аналитики должны работать в рамках системы тестирования на основе рисков, установленной для проекта руководителем тестирования. Они должны использовать свои знания о рисках бизнес-области, присущих проекту, таких как риски, связанные с безопасностью, деловыми и экономическими проблемами, политическими факторами и др.

## 2.2 Определение рисков

Привлекая широкую выборку заинтересованных сторон, процесс идентификации рисков с наибольшей вероятностью позволит выявить как можно большее количество существенных рисков.

Тест-аналитики часто обладают уникальными знаниями, касающимися конкретной бизнес-области тестируемой системы. Это означает, что они особенно хорошо подходят для следующих задач:

- Проведение экспертных интервью с экспертами и пользователями в данной сфере
- Проведение независимых оценок
- Использование шаблонов рисков
- Участие в рабочих группах по управлению рисками
- Участие в мозговых штурмах с потенциальными и текущими пользователями
- Определение чек-листов тестирования
- Обращение к опыту работы с аналогичными системами или проектами.

В частности, тест-аналитики должны тесно сотрудничать с пользователями и другими экспертами (например, системными аналитиками, бизнес-аналитиками), чтобы определить области бизнес-рисков, на которые следует обратить внимание во время тестирования. В гибкой разработке программного обеспечения такое тесное взаимодействие с заинтересованными

сторонами позволяет проводить определение рисков на регулярной основе, например, во время встреч по планированию итераций.

Примеры рисков, которые могут быть выявлены в проекте, включают:

- Проблемы с функциональной корректностью, например, неправильные расчеты
- Проблемы с удобством использования, например, недостаточное количество горячих клавиш
- Проблемы переносимости, например, невозможность установки приложения на некоторые платформы

## 2.3 Оценка рисков

В то время как определение рисков заключается в выявлении как можно большего количества рисков, оценка риска — это изучение этих выявленных рисков. В частности, классификация каждого риска и определение уровня риска.

Определение уровня риска обычно включает оценку вероятности наступления риска и степень его воздействия для каждого элемента риска. Вероятность риска обычно интерпретируется как вероятность того, что потенциальная проблема может существовать в тестируемой системе и возникнуть, когда система находится в эксплуатации. Технические тест-аналитики должны внести свой вклад в поиск и понимание вероятности возникновения для каждого элемента риска, в то время как тест-аналитики вносят свой вклад в понимание влияния проблемы на бизнес в случае ее возникновения (в гибкой разработке программного обеспечения это ролевое различие может проследиваться менее четко).

Влияние риска часто интерпретируется как серьезность последствий для пользователей, клиентов или других заинтересованных сторон. Другими словами, оно возникает в результате бизнес-риска. Тест-аналитики должны внести свой вклад в определение и оценку потенциального воздействия на бизнес-область или пользователя для каждого элемента риска. К факторам, влияющим на бизнес-риск, относятся следующие:

- Частота использования затронутой функции
- Потери бизнеса
- Финансовый ущерб
- Экологические или социальные потери, или ответственность
- Гражданские или уголовные правовые санкции
- Вопросы безопасности
- Штрафы, потеря лицензии
- Отсутствие разумных обходных путей, когда пользователи не могут больше работать
- Доступность функционала
- Очевидность отказа, ведущая к огласке и потенциальному ущербу имиджу
- Потеря клиентов

Учитывая имеющуюся информацию о рисках, тест-аналитики должны определить уровни бизнес-риска в соответствии с рекомендациями, предоставленными руководителем тестирования. Они могут быть классифицированы с использованием шкалы (числовые значения или низкий/средний/высокий), или цветов светофора. После присвоения вероятности риска и его воздействия руководители тестирования используют эти значения для определения уровня риска для каждого элемента риска. Этот уровень риска затем используется для определения приоритетности мероприятий по снижению риска. [vanVeenendaal12].

## 2.4 Смягчение рисков

В ходе проекта тест-аналитики должны стремиться сделать следующее:

- Снизить риск продукта путем разработки эффективных тестовых сценариев, которые однозначно демонстрируют, пройдены тесты или нет, а также путем участия в рецензировании рабочих продуктов программного обеспечения, таких как требования, дизайн и пользовательская документация.
- Осуществить соответствующие действия по смягчению риска, определенные в стратегии и плане тестирования (например, протестировать бизнес-процесс с особо высоким риском, используя конкретные методы тестирования).
- Осуществить переоценку известных рисков на основе дополнительной информации, собранной по мере реализации проекта, корректируя вероятность рисков, влияние рисков или и то, и другое, в зависимости от ситуации
- Выявление новых рисков на основе информации, полученной в ходе тестирования

Если речь идет о рисках продукта, то тестирование вносит существенный вклад в снижение таких рисков. Обнаруживая дефекты, тестировщики снижают риск, обеспечивая осведомленность о дефектах и возможность их устранения до выпуска продукта. Если тестировщики не находят дефектов, то тестирование снижает риск, предоставляя доказательства того, что при определенных условиях (т.е. проверенных условиях) система работает правильно. Тест-аналитики помогают определить варианты смягчения риска, изучая возможности сбора точных тестовых данных, создавая и тестируя реалистичные пользовательские сценарии, проводя или контролируя исследования удобства использования и т.д.

### 2.4.1 Приоритизация тестов

Уровень риска также используется для определения приоритетности тестов. Тест-аналитик может определить, что существует высокий риск в области точности транзакций в системе бухгалтерского учета. В результате, чтобы снизить риск, тестировщик может работать с другими экспертами в области бизнеса, чтобы собрать надежный набор тестовых данных, который может быть обработан и проверен на достоверность. Аналогично, тест-аналитик может определить, что вопросы удобства использования представляют собой значительный риск для нового объекта тестирования. Чтобы не ждать пользовательское приемочное тестирование для обнаружения каких-либо проблем, тест-аналитик может отдать предпочтение раннему тестированию удобства использования на основе прототипа, чтобы помочь раньше выявить и решить проблемы дизайна в части удобства использования, до пользовательского приемочного тестирования. Такое определение приоритетов должно быть рассмотрено как можно раньше на стадии планирования, чтобы график мог соотнести необходимое тестирование в нужный момент.

В некоторых случаях все тесты с наивысшим риском проводятся перед любыми тестами с более низким риском, и тесты проводятся в строгой последовательности рисков (так называемый «поиск в глубину»); в других случаях используется выборочный подход для отбора выборки тестов по всем идентифицированным областям риска с использованием уровня риска для взвешивания выборки, в то же время обеспечивая охват каждого риска по крайней мере один раз (так называемый «обход в ширину»).

Независимо от того, проводится ли тестирование на основе оценки рисков по обходу в глубину или по обходу в ширину, существует вероятность того, что время, отведенное на тестирование, может быть израсходовано без выполнения всех тестов. Тестирование на основе оценки риска позволяет тестировщикам отчитываться перед руководством об оставшемся уровне риска на данный момент и позволяет руководству решить, следует ли продлить тестирование или

переложить оставшийся риск на пользователей, заказчиков, службу поддержки/техническую поддержку и/или операционные службы.

#### 2.4.2 Корректировка тестирования для будущих циклов тестирования

Оценка рисков не является действием, которое выполняется один раз перед началом реализации тестов; это непрерывный процесс. Каждый будущий запланированный цикл тестирования должен быть подвергнут новому анализу рисков с учетом таких факторов, как:

- Любые новые или существенно измененные продуктовые риски
- Нестабильные или подверженные сбоям участки, обнаруженные во время тестирования
- Риски от исправленных дефектов
- Типичные дефекты, обнаруженные во время тестирования
- Недостаточно проверенные области (недостаточное покрытие требований)

## 3. Методы тестирования - 630 минут

### Ключевые слова

Анализ граничных значений, исследовательское тестирование, классификация дефектов, концепция тестирования, метод деревьев классификации, метод создания тестовых сценариев на основе дефектов, метод тестирования на основе опыта, попарное тестирование, предположение об ошибках, тестирование методом черного ящика, тестирование на основе чек-листов, тестирование на основе опыта, тестирование по сценариям использования, тестирование таблицы переходов, тестирование таблицы решений, эквивалентное разбиение.

### Цели обучения главы «Методы тестирования»

#### 3.1 Введение

Цели обучения отсутствуют

#### 3.2 Методы тестирования черного ящика

- TA-3.2.1 (K4) Проанализировать предоставленный(е) элемент(ы) спецификации и разработать тестовые сценарии, применяя эквивалентное разбиение
- TA-3.2.2 (K4) Проанализировать предоставленный(е) элемент(ы) спецификации и разработать тестовые сценарии, применяя анализ граничных значений
- TA-3.2.3 (K4) Проанализировать предоставленный(е) элемент(ы) спецификации и разработать тестовые сценарии, применяя тестирование таблицы решений
- TA-3.2.4 (K4) Проанализировать предоставленный(е) элемент(ы) спецификации и разработать тестовые сценарии, применяя тестирование таблицы переходов
- TA-3.2.5 (K2) Объяснить, как диаграммы деревьев классификации поддерживают методы тестирования
- TA-3.2.6 (K4) Проанализировать предоставленный(е) элемент(ы) спецификации и разработать тестовые случаи, применяя попарное тестирование.
- TA-3.2.7 (K4) Проанализировать предоставленный элемент(ы) спецификации и разработать тестовые сценарии, применяя тестирование по сценариям использования
- TA-3.2.8 (K4) Проанализировать систему или ее спецификацию требований, чтобы определить вероятные типы дефектов и выбрать соответствующие методы тестирования методом черного ящика

#### 3.3 Методы тестирования на основе опыта

- TA-3.3.1 (K2) Объяснить принципы методов тестирования на основе опыта, а также их преимущества и недостатки по сравнению с методами тестирования методом черного ящика и методами создания тестовых сценариев на основе дефектов
- TA-3.3.2 (K3) Определить исследовательские тесты на основе данного сценария
- TA-3.3.3 (K2) Описать применение методов создания тестовых сценариев на основе дефектов и отличить их использование от методов тестирования методом черного ящика

#### 3.4 Применение наиболее подходящего метода

- TA-3.4.1 (K4) Для данной проектной ситуации определить, какие методы тестирования методом черного ящика или методы тестирования на основе опыта следует применить для достижения конкретных целей

## 3.1 Введение

Рассмотренные в этой главе методы тестирования делятся на следующие категории:

- Черного ящика
- На основе опыта

Эти методы являются взаимодополняющими и могут использоваться по мере необходимости для любой конкретной активности тестирования, независимо от того, на каком уровне выполняется тестирование.

Обратите внимание, что обе категории методов могут быть использованы для тестирования функциональных и нефункциональных характеристик качества. Тестирование характеристик программного обеспечения рассматривается в следующей главе.

Методы тестирования, обсуждаемые в этих разделах, могут быть сосредоточены в первую очередь на определении оптимальных тестовых данных (например, на основе эквивалентного разбиения) или разработке процедур тестирования (например, на основе моделей переходов). Часто применяется комбинирование методов для создания полноценных тестовых сценариев.

## 3.2 Методы тестирования черного ящика

Методы тестирования черного ящика представлены в программе обучения базового уровня [ISTQB\_FL\_SYL].

Основные характеристики методов тестирования черного ящика включают:

- Создание моделей, таких как диаграммы переходов и таблицы решений, во время разработки теста в соответствии с методом тестирования
- Систематическое получение тестовых условий из этих моделей

Методы тестирования обычно предоставляют критерии покрытия, которые можно использовать для оценки деятельности по проектированию и выполнению тестов. Выполнение всех критериев покрытия не означает, что весь набор тестов завершен, а скорее то, что модель больше не предлагает никаких дополнительных тестов для увеличения покрытия на основе данного метода.

Тестирование черного ящика обычно основано на конкретной форме спецификаций, такой как спецификация требований к системе или пользовательские истории.

Поскольку спецификация должна описывать поведение системы, особенно в области ее функциональности, создание тестов на основе требований часто является частью тестирования поведения системы. В некоторых случаях может не быть спецификаций, но есть подразумеваемые требования, например, замена функциональности устаревшей системы.

Существует ряд методов тестирования черного ящика. Эти методы нацелены на различные сценарии и типы программного обеспечения. В следующих разделах показана применимость каждого метода, некоторые ограничения и трудности, с которыми может столкнуться тест-аналитик, метод, с помощью которого измеряется покрытие, и типы дефектов, на которые ориентированы методы.

Пожалуйста, обратитесь к [ISO29119-4], [Bath14], [Beizer95], [Black07], [Black09], [Copeland04], [Craig02], [Forgács19], [Kooomen06] и [Myers11] для получения дополнительной информации.

### 3.2.1 Эквивалентное разбиение

Эквивалентное разбиение — это метод, используемый для уменьшения количества тестовых сценариев, необходимых для эффективного тестирования обработки входных, выходных, внутренних и связанных со временем значений. Метод используется для создания эквивалентных разбиений (часто называемых классами эквивалентности) - наборов значений, которые должны одинаково обрабатываться. При выборе одного репрезентативного значения из класса предполагается проверка всех элементов в том же классе.

Обычно несколько параметров определяют поведение объекта тестирования. Для объединения эквивалентных разбиений различных параметров в тестовые сценарии, могут применяться различные методы.

#### Применимость

Этот метод применим на любом уровне тестирования и подходит в тех случаях, когда ожидается, что все элементы набора тестируемых значений будут обрабатываться одинаково и когда наборы значений, используемые приложением, не пересекаются. Областью эквивалентности может быть любое непустое множество значений, например: упорядоченное, неупорядоченное, дискретное, непрерывное, бесконечное, конечное или даже единичное. Выбор наборов значений применим к допустимым и недопустимым эквивалентным областям (т.е. областям, содержащим значения, которые следует считать недопустимыми для тестируемого программного обеспечения).

Эквивалентное разбиение наиболее эффективно при использовании в сочетании с анализом граничных значений, который расширяет тестовые значения, включая значения на границах разделов. Эквивалентное разбиение, с использованием допустимых областей эквивалентности, является широко используемым методом тестирования минимальной работоспособности (smoke) новой сборки или нового релиза, поскольку она быстро определяет, работает ли базовая функциональность.

#### Ограничения/Трудности

Если предположение неверно и значения в областях эквивалентности обрабатываются не совсем одинаково, этот метод может пропустить дефекты. Также важно тщательно выбирать области эквивалентности. Например, поле ввода, принимающее положительные и отрицательные числа, лучше тестировать как две допустимых эквивалентных области, одна для положительных и одна для отрицательных чисел, поскольку существует вероятность различной обработки. Допустимость ввода нуля может являться поводом для выделения еще одной области. Тест-аналитику важно понять, что лежит в основе обработки, чтобы определить наилучшее эквивалентное разбиение значений. Для этого может потребоваться поддержка в понимании проектирования кода.

Тест-аналитик должен также учитывать возможные зависимости между областями эквивалентности различных параметров. Например, в системе бронирования авиабилетов параметр «сопровождающий взрослый» может использоваться только в сочетании с возрастной категорией «ребенок».

#### Покрытие

Покрытие определяется путем определения числа эквивалентных областей, в которых было проверено значение, и деления этого числа на количество эквивалентных областей, которые были идентифицированы. Покрытие эквивалентного разбиения указывается в процентах. Использование нескольких значений для одной области эквивалентности не увеличивает процент покрытия.

Если поведение объекта тестирования зависит от одного параметра, то каждая область эквивалентности, будь она допустимая или недопустимая, должна покрываться хотя бы раз.

В случае более чем одного параметра тест-аналитик должен выбрать простой или комбинаторный тип покрытия в зависимости от риска [Offutt16]. Поэтому важно различать комбинации, содержащие только допустимые эквивалентные области, и комбинации, содержащие одну или несколько недопустимых эквивалентных областей. Для комбинаций, содержащих только допустимые эквивалентные области, минимальным требованием является покрытие всех допустимых областей по всем параметрам. Минимальное количество тестовых сценариев, необходимых в таком тестовом наборе, равно наибольшему количеству допустимых областей параметра, при условии, что параметры независимы друг от друга. Более детальные типы покрытия, связанные с комбинаторными методами, включают попарное покрытие (см. раздел 3.2.6 ниже) или полное покрытие любой комбинации допустимых областей эквивалентности. Недопустимые эквивалентные области должны тестироваться как минимум индивидуально, то есть в комбинации с допустимыми областями для других параметров, чтобы избежать маскировки дефектов. Таким образом, каждая недопустимая эквивалентная область вносит один тестовый сценарий в набор тестов для простого покрытия. В случае высокого риска в набор тестов могут быть добавлены дополнительные комбинации, например, состоящие только из недопустимых разделов или из пар недопустимых разделов.

### Типы дефектов

Тест-аналитик использует этот метод для обнаружения дефектов в обработке различных значений данных.

### 3.2.2 Анализ граничных значений

Анализ граничных значений используется для проверки правильной обработки значений, находящихся на границах упорядоченных областей эквивалентности. Обычно используются два метода для определения граничных значений: методом двух точек или методом трех точек. При тестировании граничных значений методом двух точек используется граничное значение (на границе) и значение, находящееся за границей (с наименьшим возможным приращением, исходя из требуемой точности). Например, для сумм в валюте с двумя знаками после запятой, если раздел включает значения от 1 до 10, тестовые значения двух значений для верхней границы будут 10 и 10.01. Тестовые значения для нижней границы были бы 1 и 0,99. Границы определяются максимальным и минимальным значениями в определенной области эквивалентности.

При тестировании граничных значений методом трех точек используются значения до, на и за границей области. В предыдущем примере тесты верхней границы будут включать 9,99, 10 и 10,01. Тесты нижней границы будут включать 0,99, 1 и 1,01. Решение об использовании метода двух точек или метода трех точек должно основываться на риске, связанном с тестируемым элементом, при этом трехточечный граничный подход должен использоваться для элементов с более высоким риском.

### Применимость

Этот метод применим на любом уровне тестирования и подходит, когда существуют упорядоченные области эквивалентности. По этой причине метод анализа граничных значений часто используется вместе с методом эквивалентного разбиения. Упорядоченные области эквивалентности необходимы из-за концепции нахождения на границе и за границей. Например, диапазон чисел является упорядоченной областью эквивалентности. Область, состоящая из некоторых текстовых строк, тоже может быть упорядочена, например, лексикографически, но если упорядочение не имеет значения с точки зрения бизнеса или технической точки зрения, то значения границ не должны быть в центре внимания.

В дополнение к диапазонам чисел, есть области эквивалентности, для которых может быть применен анализ граничных значений:

- Числовые атрибуты нечисловых переменных (например, длина)
- Количество итераций циклов, включая циклы в диаграммах переходов
- Количество элементов итерации в хранимых структурах данных, таких как массивы
- Размер физических объектов, например, памяти
- Длительность действий

### Ограничения/Трудности

Поскольку точность этого метода зависит от точного определения эквивалентных областей, на основе которой следует правильно определить границы, она подвержена тем же ограничениям и трудностям, что и метод эквивалентного разбиения. Тест-аналитик также должен учитывать точность допустимых и недопустимых значений, чтобы точно определить значения, которые должны быть протестированы. Только упорядоченные области эквивалентности могут быть использованы для анализа граничных значений, но они не ограничивают диапазон допустимых входных данных. Например, при тестировании количества ячеек, поддерживаемых электронной таблицей, существует область эквивалентности, содержащая количество ячеек вплоть до максимально допустимого (граница) и другая область эквивалентности, начинающаяся с одной ячейки сверх максимума (за границей).

### Покрытие

Покрытие определяется путем определения количества граничных условий, которые были протестированы, и деления этого количества на количество идентифицированных граничных условий (либо с использованием метода двух точек, либо метода трех точек). Покрытие указывается в процентах.

Как и для классов эквивалентности, при наличии нескольких параметров тест-аналитик должен выбрать простой или комбинаторный тип покрытия, в зависимости от риска.

### Типы дефектов

Анализ граничных значений надежно обнаруживает смещение или пропуск границ, а также может выявить случаи необходимости дополнительных границ. Этот метод позволяет найти дефекты, связанные с обработкой граничных значений, особенно ошибки с логикой «меньше, чем» и «больше, чем» (т.е. смещение). Он также может быть использован для поиска нефункциональных дефектов, например, система поддерживает 10 000 одновременных пользователей, но не 10 001.

## 3.2.3 Тестирование таблицы решений

Таблица решений — это табличное представление набора условий и связанных с ними действий, выраженное в виде правил, указывающих, какое действие должно произойти при определенном наборе значений условий [OMG-DMN]. Тест-аналитики могут использовать таблицы решений для анализа правил, применимых к тестируемому программному обеспечению, и проектирования тестов, покрывающих эти правила.

Условия и результирующие действия объекта тестирования образуют строки таблицы решений, обычно условия располагаются сверху, а действия - внизу. Первый столбец таблицы содержит описания условий и действий. Следующие столбцы, называемые правилами, содержат значения условий и соответствующие им значения действий.

Таблицы решений, в которых условия являются булевыми с простыми значениями «Т» («true», истина) и «F» («false», ложь), называются таблицами решений с ограниченным набором правил. Примером такого условия является «Доход пользователя <1000». Таблицы решений с расширенным входом позволяют использовать условия, имеющие несколько значений, которые могут представлять собой дискретные элементы или наборы элементов. Например, условие «Доход пользователя» может принимать одно из трех возможных значений: «меньше 1000», «от 1000 до 2000» и «больше 2000».

Простые действия принимают булевы значения «Т» и «F» (например, действие «Допустимая скидка = 20%» принимает значения «Т», обозначаемое «Х», если действие должно произойти, и «F», обозначаемое «-», если нет). Как и в случае с условиями, действия могут принимать значения из других областей. Например, действие «Допустимая скидка» может принимать одно из пяти возможных значений: 0%, 10%, 20%, 35% и 50%.

Тестирование таблиц решений начинается с разработки таблиц решений на основе спецификации. Правила, содержащие невыполнимые комбинации значений условий, исключаются или помечаются как «невыполнимые». Далее тест-аналитик должен провести рецензирование таблицы решений с другими заинтересованными сторонами. Тест-аналитик должен обеспечить однозначность правил в таблице (т.е. правила не пересекаются), полноту (т.е. содержат правило для каждой выполнимой комбинации значений условий) и корректность (т.е. моделируют предполагаемое поведение).

Основной принцип тестирования таблиц решений заключается в том, что правила формируют тестовые условия.

При проектировании тестового сценария для покрытия конкретного правила, тест-аналитик должен понимать, что входные параметры тестового сценария могут отличаться от параметров, указанных в условиях таблицы решений. Например, значение «Т» для условия «возраст ≥ 18?» может потребовать от тестировщика вычисления возраста на основе входных параметров - даты рождения и текущей даты. Аналогично, ожидаемые результаты тестового сценария могут быть косвенными последствиями действий таблицы решений.

После того как таблица решений готова, правила необходимо реализовать в виде тестовых сценариев, выбрав входные значения (и ожидаемые результаты), которые удовлетворяют условиям и действиям.

### Свернутые таблицы решений

При попытке проверить каждую возможную комбинацию входных параметров в соответствии с условиями, таблицы решений могут стать очень большими. Полная таблица решений с ограниченным набором правил с  $n$  условиями имеет  $2^n$  правил. Метод систематического сокращения количества комбинаций называется тестированием свернутой таблицы решений [Mosley93]. При использовании этого метода группа правил с одинаковым набором действий может быть сокращена (свернута) до одного правила, если в этой группе некоторые условия не имеют значения для действия, а все остальные условия остаются неизменными. В этом полученном правиле значения нерелевантных условий обозначаются как «не имеет значения», и обычно отмечаются прочерком «-». Для условий со значениями «не имеет значения» тест-аналитик может указать произвольные допустимые значения для реализации теста.

Еще один случай для сворачивания правил возникает, когда значение условия неприменимо в сочетании с некоторыми другими значениями условий или когда два или более условий имеют противоречивые значения. Например, в таблице решений для платежей по карте, если условие «карта действительна» ложно, то условие «допустимый ПИН-код» неприменимо.

Свернутые таблицы решений могут содержать гораздо меньше правил, чем полные таблицы решений, что приводит к меньшему количеству тестовых сценариев и меньшим усилиям. Если

заданное правило имеет записи «не имеет значения», и только один тестовый сценарий покрывает это правило, то для этого правила будет проверено только одно из нескольких возможных значений условия, и дефекты, связанные с другими значениями, могут остаться необнаруженными. Следовательно, для высоких уровней риска, в согласовании с руководителем тестирования, тест-аналитик должен определить отдельные правила для каждой возможной комбинации значений одиночных условий, а не сворачивать таблицу решений.

### Применимость

Тестирование таблиц решений обычно применяется на интеграционном, системном и приемочном уровнях тестирования. Оно также может быть применимо к компонентному тестированию, когда компонент отвечает за набор логики принятия решений. Этот метод особенно полезен, когда тестируемый объект определен в виде блок-схем или таблиц бизнес-правил.

Таблицы решений также являются методом определения требований, и иногда спецификации требований могут быть уже определены в этом формате. Тест-аналитик все равно должен участвовать в рецензировании таблиц решений и анализировать их перед началом проектирования тестов.

### Ограничения/Трудности

При рассмотрении комбинаций условий найти все взаимодействующие условия может быть непросто, особенно если требования определены не до конца или вообще не документированы. Необходимо тщательно выбирать условия, рассматриваемые в таблице решений, чтобы количество комбинаций этих условий оставалось управляемым. Иначе количество правил будет расти экспоненциально.

### Покрытие

Общим стандартом покрытия для этого метода является покрытие каждого правила таблицы решений одним тестовым сценарием. Покрытие измеряется как количество правил, покрытых набором тестов, деленное на общее количество возможных правил, выраженное в процентах.

Анализ граничных значений и эквивалентное разбиение могут быть объединены с методом таблиц решений, особенно в случае развернутых таблиц решений. Если условия содержат упорядоченные эквивалентные области, граничные значения могут быть использованы в качестве дополнительных записей, приводящих к дополнительным правилам и тестовым сценариям.

### Типы дефектов

Типичные дефекты включают неправильную логическую обработку на основе определенных комбинаций условий, приводящую к неожиданным результатам. В процессе создания таблиц решений дефекты могут быть обнаружены в спецификации. Нередко можно подготовить набор условий и определить, что ожидаемый результат не определен для одного или нескольких правил. Наиболее распространенными видами дефектов являются пропуски действий (т.е. отсутствует информация о том, что должно произойти на самом деле в определенной ситуации) и противоречия.

## 3.2.4 Тестирование таблицы переходов

Тестирование таблицы переходов используется для проверки способности тестового объекта входить в определенные состояния и выходить из них посредством допустимых переходов, а также для попытки войти в недопустимые состояния или покрыть недопустимые переходы. События заставляют объект тестирования переходить из состояния в состояние и выполнять действия. События могут определяться условиями (иногда называемыми защитными условиями)

или защитными переходами), которые влияют на выбор пути перехода. Например, событие входа в систему с допустимой комбинацией имени пользователя и пароля вызовет переход отличный от события входа в систему с недопустимым паролем. Эта информация представлена в диаграмме переходов или в таблице переходов (которая также может включать потенциальные недопустимые переходы между состояниями).

### Применимость

Тестирование таблицы переходов применимо для любого программного обеспечения, которое имеет определенные состояния и события, вызывающие переходы между этими состояниями (например, смена экранов). Тестирование таблицы переходов может использоваться на любом уровне тестирования. Встроенное программное обеспечение, web-приложения и любой тип транзакционного программного обеспечения являются хорошими кандидатами для этого типа тестирования. Системы управления, например, светофорные контроллеры, также являются хорошими кандидатами для такого типа тестирования.

### Ограничения/Трудности

Определение переходов часто является самой сложной частью составления диаграммы переходов или таблицы переходов. Если объект тестирования имеет пользовательский интерфейс, различные экраны, которые отображаются для пользователя, часто представлены состояниями. Для встроенного программного обеспечения переходы могут зависеть от состояний аппаратного обеспечения.

Помимо самих переходов, основной единицей тестирования таблицы переходов является одиночный переход. Простое тестирование всех одиночных переходов позволит обнаружить некоторые виды дефектов перехода между состояниями, но еще больше дефектов можно обнаружить при тестировании последовательностей переходов. Один отдельный переход называется 0-переключением; последовательность из двух последовательных переходов называется 1-переключением; последовательность из трех последовательных переходов называется 2-переключением, и так далее. В общем случае N-переключений представляет собой N+1 последовательных переходов [Chow1978]. С увеличением N число N-переключений растет очень быстро, что затрудняет достижение покрытия N-переключений разумным и небольшим числом тестов.

### Покрытие

Как и в других видах методов тестирования, существует иерархия уровней покрытия. Минимально допустимый уровень покрытия заключается в посещении каждого состояния и прохождении каждого перехода хотя бы один раз. Покрытие 100% переходов (также известное как 100% покрытие 0-переключений) гарантирует, что каждое состояние будет посещено и каждый переход будет пройден, если только проект системы или модель переходов (диаграмма или таблица) не содержат дефектов. В зависимости от отношений между состояниями и переходами, может потребоваться пройти некоторые переходы более одного раза, чтобы выполнить другие переходы один раз.

Термин «покрытие N-переходов» относится к количеству покрытых переключений длины N+1 в процентах от общего количества переключений этой длины. Например, для достижения 100% охвата 1-переключения требуется, чтобы каждая допустимая последовательность из двух последовательных переходов была протестирована хотя бы один раз. Такое тестирование может обнаружить некоторые типы отказов, которые при 100% охвате 0-переключений будут пропущены.

«Круговое покрытие» относится к ситуациям, в которых последовательности переходов образуют циклы. «Круговое покрытие» достигает 100%, когда все циклы из любого состояния обратно в то же состояние были протестированы для всех переходов, в которых циклы

начинаются и заканчиваются. Этот цикл не может содержать более одного вхождения какого-либо конкретного состояния (кроме начального/конечного) [Offutt16].

При любом из этих подходов повышение уровня покрытия будет заключаться в попытке включить все недопустимые переходы, определенные в таблице переходов. Требования к покрытию при тестировании таблицы переходов должны указывать, включены ли недопустимые переходы.

Проектирование тестовых примеров для достижения желаемого покрытия поддерживается диаграммой переходов или таблицей переходов для конкретного тестового объекта. Эта информация также может быть представлена в таблице, которая показывает переходы N-переключений для определенного значения N [Black09].

Можно использовать ручную процедуру для определения элементов, которые должны быть покрыты (например, переходы, состояния или N-переходы). Один из предлагаемых методов - распечатать диаграмму переходов и таблицу переходов и использовать ручку или карандаш, чтобы отметить покрытые элементы, пока не будет достигнут необходимый уровень покрытия [Black09]. Для более сложных диаграмм переходов и таблиц переходов такой подход будет слишком трудоемким. Поэтому для поддержки тестирования таблицы переходов следует использовать соответствующее программное обеспечение.

### Типы дефектов

К типичным дефектам относятся: (см. также [Beizer95]):

- Неверные типы или значения событий
- Неверные типы действий или значения
- Неверное начальное состояние
- Невозможность достичь определенного конечного состояния(ей)
- Невозможность войти в требуемые состояния
- Лишние (ненужные) состояния
- Невозможность правильно выполнить определенный допустимый переход(ы)
- Возможность выполнения недопустимых переходов
- Неправильные защитные условия

В процессе создания модели перехода в документации спецификации могут быть обнаружены дефекты. Наиболее распространенными видами дефектов являются пропуски (т.е. отсутствие информации о том, что должно произойти в определенной ситуации) и противоречия.

### 3.2.5 Метод деревьев классификации

Деревья классификации поддерживают некоторые методы тестирования «черного ящика», позволяя создать графическое представление пространства данных, применимое к объекту тестирования.

Данные организованы в классификации и классы следующим образом:

- Классификации: представляют параметры в пространстве данных для объекта тестирования, такие как входные параметры (которые могут также содержать состояния окружения и предварительные условия) и выходные параметры. Например, если приложение может быть настроено различными способами, классификации могут включать клиент, браузер, язык и операционную систему.
- Классы: каждая классификация может иметь любое количество классов и подклассов, описывающих входные значения параметра. Каждый класс, или эквивалентная область, представляет собой конкретное значение в рамках классификации. В приведенном выше примере классификация языка может

включать эквивалентные области для английского, французского и испанского языков.

Деревья классификации позволяют тест-аналитикам вводить комбинации по своему усмотрению. Это включает, например, попарные комбинации (см. раздел 3.2.6), тройные комбинации и одиночные комбинации.

Дополнительная информация об использовании метода деревьев классификации представлена в [Bath14] и [Black09].

### **Применимость**

Создание дерева классификации помогает тест-аналитику определить параметры (классификации) и их эквивалентные области (классы), которые представляют интерес.

Дальнейший анализ диаграммы дерева классификации позволяет определить возможные граничные значения и выявить определенные комбинации входных параметров, которые либо представляют особый интерес, либо могут быть исключены (например, из-за их несовместимости). Полученное дерево классификации может быть использовано для поддержки эквивалентного разбиения, анализа граничных значений или попарного тестирования (см. раздел 3.2.6).

### **Ограничения/Трудности**

По мере роста количества классификаций и/или классов диаграмма увеличивается, а удобство ее использования снижается. Кроме того, метод деревьев классификации не создает полных тестовых сценариев, а только комбинации тестовых данных. Тест-аналитики должны предоставить результаты для каждой тестовой комбинации, чтобы создать полные тестовые сценарии.

### **Покрытие**

Тестовые сценарии могут быть спроектированы для достижения, например, минимального покрытия классов (т.е. все значения в классификации проверяются хотя бы один раз). Тест-аналитик может также решить охватить попарные комбинации или использовать другие виды комбинаторного тестирования, например, трехстороннее.

### **Типы дефектов**

Типы обнаруженных дефектов зависят от метода (методов), который поддерживает деревья классификации (например, эквивалентное разбиение, анализ граничных значений или попарное тестирование).

## **3.2.6 Попарное тестирование**

Попарное тестирование используется при тестировании программного обеспечения, в котором несколько входных параметров, каждый из которых имеет несколько возможных значений, должны быть проверены в комбинации, что приводит к большому количеству комбинаций, чем возможно проверить за отведенное время. Входные параметры могут быть независимыми в том смысле, что любой параметр (т.е. любое выбранное значение для любого одного входного параметра) может быть объединен с любым другим фактором, однако это не всегда так (см. примечание о функциональных моделях ниже). Комбинация конкретного параметра (переменной или фактора) с конкретным значением этого параметра называется парой параметр-значение (например, если «цвет» - это параметр с семью допустимыми значениями, включая «красный», то «цвет = красный» может быть парой параметр-значение).

Попарное тестирование использует комбинаторные методы, чтобы гарантировать, что каждая пара параметр-значение тестируется один раз в сочетании с каждой парой параметр-значение каждого другого параметра (т.е. проверяются «все пары» пар параметр-значение для любых двух различных параметров), избегая при этом тестирования всех комбинаций пар параметр-значение. Если тест-аналитик использует ручной подход, то создается таблица с тестовыми сценариями, представленными строками и одним столбцом для каждого параметра. Затем тест-аналитик заполняет таблицу значениями так, чтобы все пары значений можно было идентифицировать в таблице (см. [Black09]). Любые записи в таблице, которые остались пустыми, тест-аналитик может заполнить значениями, опираясь на свои знания в предметной области.

Существует ряд инструментов, которые могут помочь тест-аналитику в решении этой задачи (примеры см. на сайте [www.pairwise.org](http://www.pairwise.org)). Они требуют в качестве входных данных список параметров и их значений и генерируют подходящий набор комбинаций значений каждого параметра, который покрывает все пары параметров-значений. Выходные данные инструмента могут быть использованы в качестве входных данных для тестовых сценариев.

Обратите внимание, что тест-аналитик должен предоставить ожидаемые результаты для каждой создаваемой инструментом комбинации.

Деревья классификации (см. раздел 3.2.5) часто используются в сочетании с попарным тестированием [Bath14]. Проектирование деревьев классификации поддерживается инструментами и позволяет визуализировать комбинации параметров и их значения (некоторые инструменты предлагают попарное улучшение). Это помогает выявить следующую информацию:

- Входные данные, которые будут использоваться методом попарного тестирования.
- Особые комбинации, представляющие интерес (например, часто используемые или являющиеся частым источником дефектов)
- Особые комбинации, которые несовместимы. Это не означает, что комбинированные факторы не будут влиять друг на друга; они вполне могут, но должны влиять друг на друга в приемлемых пределах.
- Логические отношения между переменными. Например, «если переменная\_1 = x, то переменная\_2 не может быть y». Деревья классификации, которые отражают эти отношения, называются «функциональными моделями».

## Применимость

Проблема наличия слишком большого количества комбинаций значений параметров проявляется, по меньшей мере, в двух различных ситуациях, связанных с тестированием. Некоторые элементы тестирования включают несколько параметров, каждый из которых имеет несколько возможных значений, например, экран с несколькими полями ввода. В этом случае комбинации значений параметров составляют входные данные для тестовых сценариев. Кроме того, некоторые системы могут быть конфигурируемыми по нескольким параметрам, что приводит к потенциально большему пространству конфигурации. В обоих случаях попарное тестирование может быть использовано для определения подмножества комбинаций, которое является управляемым и выполнимым.

Для параметров с большим количеством значений сначала может быть применено эквивалентное разбиение или какой-либо другой механизм выбора для каждого параметра в отдельности, чтобы сократить количество значений для каждого параметра, а затем применяется попарное тестирование для сокращения набора результирующих комбинаций. Фиксирование параметров и их значений в дереве классификации помогает это осуществить.

Эти методы обычно применяются на уровнях тестирования интеграции компонентов, системных тестов и системного интеграционного тестирования.

### **Ограничения/Трудности**

Основным ограничением этих методов является предположение, что результаты нескольких тестов являются репрезентативными для всех тестов и что эти несколько тестов представляют ожидаемое использование. Если существует непредвиденное взаимодействие между определенными переменными, оно может остаться незамеченным при использовании данного метода тестирования, если именно эта конкретная комбинация не тестировалась. Эти методы может быть трудно объяснить нетехнической аудитории, так как они могут не понять логическое сокращение тестов. Любое такое объяснение должно быть сбалансировано упоминанием результатов эмпирических исследований [Kuhn16], которые показали, что в исследуемой области медицинских устройств 66% отказов были вызваны одной переменной и 97% - либо одной переменной, либо двумя взаимодействующими переменными. Существует остаточный риск того, что попарное тестирование может не выявить отказы в системах, при взаимодействии трех и более переменных.

Определение параметров и их соответствующих значений иногда является трудновыполнимой задачей. Поэтому, по возможности, эту задачу следует выполнять с помощью деревьев классификации (см. раздел 3.2.5). Поиск минимального набора комбинаций, удовлетворяющих определенному уровню покрытия, трудно выполнить вручную. Для поиска минимально возможного набора комбинаций можно использовать инструменты. Некоторые инструменты поддерживают возможность принудительного включения или исключения некоторых комбинаций из окончательного набора комбинаций. Тест-аналитик может использовать эту возможность, чтобы подчеркнуть или ослабить значение факторов, основанных на знании предметной области или информации об использовании продукта.

### **Покрытие**

Покрытие 100% попарных комбинаций требует включения каждой пары значений любой пары параметров в по крайней мере одну комбинацию.

### **Типы дефектов**

Наиболее распространенным типом дефектов, обнаруженных с помощью этого метода тестирования, являются дефекты, связанные с комбинированными значениями двух параметров.

## **3.2.7 Тестирование по сценариям использования**

Тестирование по сценариям использования обеспечивает транзакционные, основанные на поведении тесты, которые должны эмулировать предполагаемое использование компонента или системы, указанного в сценарии использования. Сценарии использования определяются, исходя из взаимодействий между действующими лицами и компонентом или системой, которые достигают некоторой цели. Действующими лицами могут быть люди-пользователи, внешнее оборудование, другие компоненты или системы.

Общий стандарт для сценариев использования представлен в [OMG-UML].

### **Применимость**

Тестирование по сценариям использования обычно применяется в системном и приемочном тестировании. Оно также может использоваться в интеграционном тестировании, если поведение компонентов или систем определено сценариями использования. Сценарии использования часто являются базисом тестирования производительности, поскольку они отражают реальное использование системы. Действия, описанные в сценариях использования,

могут быть назначены виртуальным пользователям для создания реалистичной нагрузки на систему (при условии, что в них или для них указаны требования к нагрузке и производительности).

### **Ограничения/Трудности**

Для того чтобы быть допустимыми, сценарии использования должны передавать реалистичные операции пользователя. Спецификации сценариев использования - это форма проектирования системы. Требования к тому, что пользователи должны выполнять, должны исходить от пользователей или представителей пользователей, и должны быть проверены на соответствие бизнес-требованиям перед проектированием соответствующих сценариев использования. Ценность варианта использования снижается, если он не отражает реальные требования пользователей и организации, а скорее мешает, чем помогает выполнению задач пользователя.

Точное определение исключений, альтернатив и поведения при обработке ошибок является важным условием детального покрытия. Сценарии использования следует рассматривать как руководство, но не как полное определение того, что должно быть протестировано, поскольку они могут не давать четкого определения всего набора требований. Также может быть полезно создать другие модели, такие как блок-схемы и/или таблицы решений, на основе описания сценариев использования для повышения точности тестирования и проверки самого сценария использования. Как и в случае с другими формами спецификации, это, вероятно, выявит логические аномалии в спецификации сценария использования, если они существуют.

### **Покрытие**

Минимально допустимым уровнем покрытия сценария использования является наличие одного тестового сценария для базового поведения и достаточного количества дополнительных тестовых сценариев для покрытия каждого альтернативного поведения и поведения при обработке ошибок. Если требуется минимальный набор тестов, в один тестовый сценарий можно включить несколько альтернативных вариантов поведения при условии их совместимости. Если требуются более эффективная диагностика (например, для помощи в изоляции дефектов), можно разработать один дополнительный тестовый сценарий на каждое альтернативное поведение, хотя вложенные альтернативные поведения все равно потребуют объединения некоторых из них в один тестовый сценарий (например, различие между завершением работы и альтернативным поведением без завершения в рамках обработки исключений с "повторной попыткой").

### **Типы дефектов**

Дефекты включают неправильную обработку определенных поведений, пропущенные альтернативные поведения, неправильную обработку представленных условий и плохо реализованные или неправильные сообщения об ошибках.

## **3.2.8 Комбинирование методов**

Иногда методы комбинируются для создания тестовых сценариев. Например, условия, определенные в таблице решений, могут быть подвергнуты эквивалентному разбиению, чтобы обнаружить несколько способов, которыми условие может быть выполнено. Тогда тестовые сценарии будут охватывать не только каждую комбинацию условий, но и те условия, которые разделены на части, должны быть созданы дополнительные тестовые сценарии, чтобы охватить эквивалентные области. При выборе конкретного метода для применения тест-аналитик должен учитывать применимость метода, ограничения и трудности, а также цели тестирования с точки зрения покрытия и выявляемых дефектов. Эти аспекты описаны для отдельных методов, рассматриваемых в данной главе. Возможно, не существует единственного «лучшего» метода для той или иной ситуации. Сочетание подходящих методов часто является наиболее

эффективным способом достижения поставленных целей тестирования, при условии, что имеется достаточно времени и навыков для правильного применения методов.

### 3.3 Методы тестирования на основе опыта.

Тестирование на основе опыта использует навыки и интуицию тестировщиков, а также их опыт работы с аналогичными приложениями или технологиями для целенаправленного тестирования, ориентированного на повышение эффективности обнаружения дефектов. Эти методы тестирования варьируются от «быстрых тестов», в которых тестировщик не имеет формально запланированных действий, до запланированных сессий тестирования, основанных на концепции тестирования для сеансов сценарного тестирования. Они почти всегда полезны, но имеют особую ценность, когда могут быть достигнуты аспекты, включенные в приведенный ниже список преимуществ.

Тестирование на основе опыта имеет следующие преимущества:

- Может быть хорошей альтернативой более структурированным подходам в случаях, когда отсутствует документация по системе.
- Можно применять, когда время тестирования сильно ограничено.
- Позволяет применять имеющийся опыт в данной области и технологии при тестировании. Могут привлекаться те, кто не участвует в тестировании, например, бизнес-аналитики, заказчики или клиенты.
- Может обеспечить раннюю обратную связь с разработчиком.
- Помогает команде ознакомиться с программным обеспечением в процессе его создания.
- Эффективно, при анализе эксплуатационных отказов.
- Позволяет применять различные методы тестирования.

Тестирование на основе опыта имеет следующие недостатки:

- Может быть неуместно в системах, требующих подробной тестовой документации.
- Трудно добиться высоких уровней повторяемости.
- Ограниченная возможность точной оценки покрытия.
- Тесты в меньшей степени подходят для последующей автоматизации.

При использовании реактивных и эвристических подходов тестировщики обычно применяют тестирование на основе опыта, которое в большей степени реагирует на события, чем заранее спланированные подходы к тестированию. Кроме того, выполнение и оценка являются параллельными задачами. Некоторые структурированные подходы к тестированию на основе опыта не являются полностью динамичными, т.е. тесты не создаются полностью одновременно с выполнением теста тестировщиком. Например, это может иметь место, когда предположение об ошибках используется для определения конкретных аспектов тестового объекта перед выполнением теста.

Стоит отметить, несмотря на то, что для обсуждаемых здесь методов приведены некоторые концепции покрытия, методы тестирования, основанные на опыте, не имеют формальных критериев покрытия.

### 3.3.1 Предположение об ошибках

При использовании метода предположения об ошибках тест-аналитик использует свой опыт, чтобы предположить наличие потенциальных ошибок, которые могли быть допущены при проектировании и разработке кода. Когда предполагаемые ошибки зафиксированы, тест-аналитик определяет наилучшие методы, которые стоит использовать для выявления возникающих дефектов. Например, если тест-аналитик ожидает, что при вводе недопустимого пароля в программном обеспечении возникнут отказы, то будут проведены тесты на ввод различных значений в поле пароля, чтобы проверить, действительно ли была допущена ошибка и привела ли она к дефекту, который можно рассматривать как отказ при выполнении тестов.

Помимо использования в качестве метода тестирования, предположение об ошибках также полезно при анализе рисков для определения потенциальных типов отказов. [Myers11].

#### Применимость

Предположение об ошибках применяется в основном на этапах интеграционного и системного тестирования, но может использоваться на любом уровне тестирования. Этот метод часто используется вместе с другими методами и помогает расширить область применения существующих тестовых сценариев. Предположение об ошибках также может быть эффективно использовано при тестировании новой версии программного обеспечения для проверки на наличие распространенных дефектов перед началом более строгого сценарного тестирования.

#### Ограничения/Трудности

Следующие ограничения и трудности относятся к предположению об ошибках:

- Покрытие трудно определить, и оно сильно варьируется в зависимости от возможностей и опыта тест-аналитика.
- Лучше всего использовать этот метод опытному тестировщику, который знаком с типами дефектов, обычно возникающих в тестируемом коде.
- Часто применяется, но не всегда документируется и поэтому может быть менее воспроизводимым, чем другие формы тестирования.
- Тестовые сценарии могут быть задокументированы, но таким образом, что только автор их понимает и может воспроизвести.

#### Покрытие

При использовании классификации дефектов покрытие определяется путем деления количества протестированных элементов списка классификации на общее количество элементов списка классификации и представления покрытия в процентах. Без классификации дефектов покрытие ограничено опытом и знаниями тестировщика, а также временем, которое ему доступно. Количество дефектов, обнаруженных с помощью этого метода, будет зависеть от того, насколько хорошо тестировщик может выделить проблемные области.

#### Типы дефектов

Типичные дефекты — это обычно те, которые определены в конкретной классификации дефектов или «предположены» тест-аналитиком, которые, возможно, могли быть пропущены при тестировании «черного ящика».

### 3.3.2 Тестирование на основе чек-листов

Применяя метод тестирования на основе чек-листов, опытный тест-аналитик использует высокоуровневый, обобщенный список элементов, которые следует отметить, проверить или запомнить, или набор правил или критериев, на соответствие которым должен быть верифицирован объект тестирования. Эти чек-листы состояются на основе набора стандартов, опыта и других аспектов. Например, стандартный чек-лист пользовательского интерфейса может быть использован в качестве основы для тестирования приложения. В гибкой методологии разработки программного обеспечения чек-листы могут быть построены на основе критериев приемки пользовательской истории.

#### Применимость

Тестирование на основе чек-листов наиболее эффективно в проектах с опытной командой тестировщиков, которая знакома с тестируемым программным обеспечением или знакома с областью, охватываемой чек-листом (например, для успешного применения чек-листа пользовательского интерфейса тест-аналитик может быть знаком с тестированием пользовательского интерфейса, но не с особенностями конкретной тестируемой системы). Поскольку чек-листы являются высокоуровневыми и, как правило, не содержат детализированных шагов, обычно встречающихся в тестовых сценариях и процедурах тестирования, знания тестировщика используются для восполнения пробелов. Благодаря отсутствию детальных шагов, чек-листы требуют минимальных затрат на поддержку и могут применяться к аналогичным релизам многократно.

Чек-листы хорошо подходят для проектов, в которых программное обеспечение быстро выпускается и изменяется. Это помогает сократить время на подготовку и сопровождение тестовой документации. Они могут использоваться на любом уровне тестирования, а также применяются для регрессионного тестирования и тестирования минимальной работоспособности (smoke).

#### Ограничения/Трудности

Высокоуровневый характер чек-листов может повлиять на воспроизводимость результатов тестирования. Возможно, что несколько тестировщиков будут по-разному интерпретировать чек-листы и применять различные подходы к выполнению элементов чек-листа. Это может привести к различным результатам тестирования, даже если используется один и тот же чек-лист. Это может привести к более широкому покрытию, иногда за счет снижения воспроизводимости. Чек-листы также могут привести к чрезмерной уверенности в достигнутом уровне покрытия, поскольку текущее тестирование зависит от решений тестировщика. Чек-листы могут быть получены из более подробных тестовых сценариев или списков и имеют тенденцию расти со временем. Для того чтобы убедиться, что чек-листы покрывают важные аспекты тестируемого программного обеспечения, необходимо их сопровождение.

#### Покрытие

Покрытие можно определить, как количество протестированных элементов чек-листа, разделенное на общее количество элементов чек-листа, с указанием результата в процентах. Покрытие настолько хорошо, насколько хорош чек-лист, но из-за высокоуровневого характера чек-листа результаты будут отличаться в зависимости от тест-аналитика, который выполняет проверки по чек-листу.

#### Типы дефектов

Типичные дефекты, обнаруженные с помощью этого метода, вызывают сбои, возникающие в результате изменения данных, последовательности шагов или общего рабочего процесса во время тестирования.

### 3.3.3 Исследовательское тестирование

Исследовательское тестирование характеризуется тем, что тестировщик одновременно узнает об объекте тестирования и его дефектах, планирует предстоящую работу по тестированию, проектирует и выполняет тесты, а также сообщает о результатах. Тестировщик динамически корректирует цели тестирования во время выполнения и готовит только упрощенную документацию. [Whittaker09].

#### Применимость

Хорошее исследовательское тестирование является спланированным, интерактивным и творческим. Оно не требует большого количества документации о тестируемой системе и часто используется в ситуациях, когда документация недоступна или недостаточна для других методов тестирования. Исследовательское тестирование часто используется для дополнения других методов тестирования и служит основой для разработки дополнительных тестовых сценариев. Исследовательское тестирование часто используется в гибкой методологии разработки программного обеспечения для гибкого и быстрого тестирования пользовательских историй с минимальной документацией. Однако этот метод может применяться и в проектах, использующих последовательную модель разработки.

#### Ограничения/Трудности

Исследовательское тестирование не обеспечивает какого-либо уровня покрытия. Более того, воспроизведение выполненных тестов может быть затруднено. Использование концепций тестирования для определения областей, которые должны быть покрыты в ходе тестовой сессии, и временных рамок для определения времени, отведенного на тестирование, - это методы, используемые для управления исследовательским тестированием. В конце тестовой сессии или набора тестовых сессий руководитель тестирования может провести встречу по подведению итогов, чтобы собрать результаты тестирования и определить концепции тестирования для следующих тестовых сессий.

Еще одна сложность с сеансами исследовательского тестирования заключается в точном отслеживании их в системе управления тестированием. Иногда для этого создаются тестовые сценарии, которые фактически являются сеансами исследовательского тестирования. Это позволяет отслеживать время, отведенное на исследовательское тестирование, и запланированное покрытие вместе с другими тестовыми мероприятиями.

Поскольку воспроизводимость может быть труднодостижимой при исследовательском тестировании, это может вызвать проблемы при необходимости вспомнить шаги для воспроизведения отказа. Некоторые организации используют функцию записи/воспроизведения инструмента автоматизации тестирования для записи шагов, предпринятых тестировщиком. Это обеспечивает полную запись всех действий во время сеанса исследовательского тестирования (или любого сеанса тестирования на основе опыта). Анализ деталей для выявления фактической причины отказа может быть утомительным, но, по крайней мере, есть запись всех шагов, которые были предприняты.

Для записи сеансов исследовательского тестирования могут использоваться другие инструменты, но они не фиксируют ожидаемые результаты, поскольку не перехватывают взаимодействие с пользовательским графическим интерфейсом. В этом случае ожидаемые результаты должны быть записаны, чтобы при необходимости можно было провести надлежащий анализ дефектов. В целом, рекомендуется вести записи во время проведения исследовательского тестирования, для поддержания воспроизводимости там, где это необходимо.

#### Покрытие

Критерии тестирования могут быть разработаны для конкретных задач, целей и результатов.

Затем планируются сеансы исследовательского тестирования для достижения этих критериев. В концепции также может быть определено, где сосредоточить усилия тестирования, что входит и выходит за рамки сеанса тестирования, и какие ресурсы должны быть выделены для завершения запланированных тестов. Сессия тестирования может быть использована для сосредоточения на определенных типах дефектов и других потенциально проблемных областях, которые могут быть устранены которые можно устранить без формального тестирования по сценарию.

### Типы дефектов

Типичными дефектами, обнаруживаемыми при исследовательском тестировании, являются проблемы на основе сценариев, которые пропускаются во время тестирования функциональной пригодности по сценарию, проблемы, выходящие за пределы функциональных границ, а также проблемы, связанные с рабочим процессом. Проблемы производительности и безопасности также иногда обнаруживаются в ходе исследовательского тестирования.

### 3.3.4 Метод создания тестовых сценариев на основе дефектов

Метод создания тестовых сценариев на основе дефектов - это метод, в котором тип искомого дефекта используется в качестве базиса проектирования тестов, а тесты планомерно получают из того, что известно о типе дефекта. В отличие от тестирования «черного ящика», когда тесты создаются на основе базиса тестирования, при тестировании на основе дефектов тесты извлекаются из списков, в которых сосредоточены дефекты. В целом, списки могут быть организованы по типам дефектов, основным причинам, признакам отказа и другим данным, связанным с дефектами. Стандартные списки применимы к разным типам программного обеспечения и не зависят от конкретного продукта. Использование этих списков помогает задействовать знания отраслевых стандартов для получения конкретных тестов. Придерживаясь отраслевых списков, можно отслеживать показатели появления дефектов по всем проектам и даже в разных организациях. Наиболее распространенные списки дефектов - это списки, составленные с учетом специфики организации или проекта и использующие специальные знания, и опыт.

Тестирование на основе дефектов может также использовать списки выявленных рисков и сценариев рисков в качестве базиса для целенаправленного тестирования. Этот метод тестирования позволяет тест-аналитику сосредоточиться на определенном типе дефекта или систематически работать со списком известных и распространенных дефектов определенного типа. На основе этой информации тест-аналитик создает тестовые условия и тестовые сценарии, которые приведут к обнаружению дефекта (если он существует).

### Применимость

Тестирование на основе дефектов может применяться на любом уровне тестирования, но чаще всего оно применяется при системном тестировании.

### Ограничения/Трудности

Существует множество классификаций дефектов, которые могут быть ориентированы на определенные виды тестирования, например, на тестирование практичности. Важно выбрать классификацию, которая применима к тестируемому программному обеспечению (если таковая имеется). Например, может не быть классификаций, доступных для инновационного программного обеспечения. Некоторые организации составили свои собственные классификации вероятных или часто встречающихся дефектов. Какая бы классификация дефектов ни использовалась, важно определить ожидаемое покрытие до начала тестирования.

## Покрытие

Метод обеспечивает критерии покрытия, которые используются для определения того, когда все полезные тестовые сценарии были определены. Элементами покрытия могут быть структурные элементы, элементы спецификации, сценарии использования или любая их комбинация, в зависимости от списка дефектов. На практике критерии покрытия для метода тестирования на основе дефектов, как правило, менее систематизированы, чем для метода тестирования «черного ящика», поскольку даются только общие правила покрытия, а конкретное решение о том, что является границей полезного покрытия, принимается по собственному усмотрению. Как и в случае с другими методами, критерии покрытия не означают, что весь набор тестов является полным, а скорее означают, что рассматриваемые дефекты больше не предполагают никаких полезных тестов, основанных на данном методе.

## Типы дефектов

Типы обнаруженных дефектов обычно зависят от используемой классификации дефектов. Например, если используется список дефектов пользовательского интерфейса, большинство обнаруженных дефектов, скорее всего, будут связаны с пользовательским интерфейсом, но другие дефекты могут быть обнаружены как побочный продукт конкретного тестирования.

## 3.4 Применение наиболее подходящего метода

Методы тестирования «черного ящика» и методы тестирования на основе опыта наиболее эффективны при совместном использовании. Методы тестирования на основе опыта заполняют пробелы в достижении целей тестирования, возникающие из-за систематических недостатков методов тестирования «черного ящика».

Не существует одного идеального метода для всех ситуаций. Тест-аналитику важно понимать преимущества и недостатки каждого метода и уметь выбрать лучший метод или набор методов для конкретной ситуации, учитывая тип проекта, график, доступ к информации, навыки тестировщика и другие факторы, которые могут повлиять на выбор.

При обсуждении каждого метода тестирования на основе «черного ящика» и метода тестирования на основе опыта (см. разделы 3.2 и 3.3 соответственно) информация, представленная в разделах «применимость», «ограничения/трудности» и «покрытие», помогает тест-аналитику выбрать наиболее подходящие методы тестирования для применения.

## 4. Тестирование характеристик качества программного обеспечения – 180 минут

### Ключевые слова

Возможность взаимодействия, защищенность от ошибок пользователя, изучаемость, лист оценки и анализа веб-сайтов (WAMMI), лист оценки практичности программного обеспечения (SUMI), общедоступность, пользовательский опыт, практичность, работоспособность, совместимость, функциональная корректность, функциональная полнота, функциональная пригодность, эстетика пользовательского интерфейса.

### Цели обучения главы «Тестирование характеристик качества программного обеспечения»

#### 4.1 Введение

Цели обучения отсутствуют

#### 4.2 Характеристики качества для тестирования бизнес-домена

- TA-4.2.1 (K2) Объяснить, какие методы тестирования подходят для проверки функциональной полноты, функциональной корректности и функциональной пригодности
- TA-4.2.2 (K2) Определить типичные дефекты, на которые следует ориентироваться при проверке характеристик функциональной полноты, функциональной корректности и функциональной пригодности
- TA-4.2.3 (K2) Определить, когда характеристики функциональной полноты, корректности и пригодности должны быть протестированы в жизненном цикле разработки программного обеспечения
- TA-4.2.4 (K2) Объяснить подходы, которые будут подходящими для верификации и валидации как реализации требований к практичности, так и соответствия ожиданиям пользователя
- TA-4.2.5 (K2) Объяснить роль тест-аналитика в тестировании возможности взаимодействия, включая определение дефектов, на которые следует ориентироваться
- TA-4.2.6 (K2) Объяснить роль тест-аналитика в тестировании переносимости, включая определение дефектов, на которые следует ориентироваться
- TA-4.2.7 (K4) Для заданного набора требований определить тестовые условия, необходимые для проверки функциональных и/или нефункциональных характеристик качества в рамках задач тест-аналитика

## 4.1 Введение

Если в предыдущей главе описывались конкретные методы, доступные тестировщику, то в этой главе рассматривается применение этих методов для оценки характеристик, используемых для описания качества программных приложений или систем.

В программе обучения рассматриваются характеристики качества, которые могут быть оценены тест-аналитиком. Атрибуты, которые должен оценивать технический тест-аналитик, рассматриваются в программе обучения продвинутого уровня Технический тест-аналитик [СТАЛ-ТТА].

Описание характеристик качества продукта, представленное в стандарте ISO 25010 [ISO25010], используется в качестве руководства для описания характеристик. Модель качества программного обеспечения ISO разделяет качество продукта на различные характеристики качества продукта, каждая из которых может иметь подхарактеристики. Эти характеристики представлены в таблице ниже, где также представлено указание на то, какие характеристики/подхарактеристики охватываются программами обучения Тест-аналитика и Технического тест-аналитика:

Характеристики	Подхарактеристики	Тест-аналитик	Технический тест-аналитик
Функциональная пригодность	Функциональная корректность, функциональная целесообразность, функциональная полнота	X	
Надежность	Зрелость, устойчивость к недочетам, восстанавливаемость, доступность		X
Практичность	Определимость пригодности, изучаемость, работоспособность, эстетика пользовательского интерфейса, защищенность от ошибок пользователя, общедоступность	X	
Эффективность производительности	Временные характеристики, использование ресурсов, потенциальные возможности		X
Сопровождаемость	Анализируемость, модифицируемость, тестируемость, модульность, возможность многократного использования		X
Переносимость	Адаптируемость, устанавливаемость, взаимозаменяемость	X	X
Безопасность	Конфиденциальность, целостность, неподдельность, отслеживаемость, подлинность		X
Совместимость	Существование		X
	Возможность взаимодействия	X	

Хотя такое распределение работы может отличаться в разных организациях, именно оно используется в соответствующих программах обучения ISTQB®.

Для всех характеристик и подхарактеристик качества, рассмотренных в этом разделе, необходимо определить типичные риски, чтобы сформировать и задокументировать соответствующую стратегию тестирования. Тестирование характеристик качества программного обеспечения требует особого внимания к срокам жизненного цикла разработки программного обеспечения, необходимым инструментам, наличию программного обеспечения и документации, а также технической экспертизе. Без стратегии тестирования каждой характеристики и ее особенностей тестировщик не может корректно спланировать работы. [Bath14]. Некоторые виды тестирования, например, тестирование практичности, могут потребовать выделения специальных человеческих ресурсов, тщательного планирования, специальных лабораторий, специфических инструментов, специализированных навыков тестирования и, в большинстве случаев, значительного количества времени. В некоторых случаях тестирование практичности может проводиться отдельной группой экспертов по практичности использования или экспертами по пользовательскому опыту.

Хотя тест-аналитик может не отвечать за характеристики качества, требующие более технического подхода, важно, чтобы он знал о других характеристиках и понимал, какие области тестирования пересекаются. Например, объект тестирования, который не прошел тестирование производительности, скорее всего, не пройдет тестирование удобства использования, если он слишком медленный, чтобы пользователь мог эффективно его использовать. Аналогично, объект тестирования с проблемами взаимодействия с некоторыми компонентами, вероятно, не готов к тестированию переносимости, поскольку это, как правило, скрывает более базовые проблемы при изменении окружения.

## 4.2 Характеристики качества для тестирования бизнес-домена

Тестирование функциональной пригодности является основным направлением работы тест-аналитика. Тестирование функциональной пригодности сосредоточено на том, «что» делает объект тестирования. Базисом тестирования функциональной пригодности обычно являются требования, спецификация, специфический опыт в конкретной области или предполагаемая потребность. Тестирование функциональной пригодности зависит от уровня тестирования, на котором оно проводится, а также может зависеть от жизненного цикла разработки программного обеспечения. Например, тестирование функциональной пригодности, проводимое во время интеграционного тестирования, проверяет функциональную пригодность взаимодействующих компонентов, которые реализуют одну определенную функцию. На уровне системного тестирования тестирование функциональной пригодности включает проверку функциональной пригодности системы в целом. Для систем, состоящих из систем, тестирование функциональной пригодности в основном сосредоточено на сквозном тестировании интегрированных систем. При тестировании функциональной пригодности используется широкий спектр методов тестирования (см. главу 3).

В гибкой методологии разработки программного обеспечения тестирование функциональной пригодности обычно включает следующее:

- Тестирование конкретной функциональности (например, пользовательских историй), запланированной для реализации в конкретной итерации
- Регрессионное тестирование для всех оставшихся без изменений функциональных возможностей

В дополнение к тестированию функциональной пригодности, описанному в данном разделе, существуют также определенные характеристики качества, входящие в зону ответственности тест-аналитика, которые считаются не функциональными (сфокусированными на том, «как» объект тестирования обеспечивает функциональность) областями тестирования.

#### 4.2.1 Тестирование функциональной корректности

Функциональная корректность включает проверку соответствия приложения указанным или подразумеваемым требованиям, а также может включать достоверность вычислений. Тестирование функциональной корректности использует многие из методов тестирования, описанных в Главе 3, и часто использует спецификацию или унаследованную систему в качестве тестового предсказателя. Тестирование функциональной корректности может проводиться на любом уровне тестирования и направлено на некорректное обращение с данными или ситуациями.

#### 4.2.2 Тестирование функциональной пригодности

Тестирование функциональной пригодности включает в себя оценку и валидацию пригодности набора функций для выполнения конкретных задач. Это тестирование может быть основано на функциональном дизайне (например, сценарии использования и/или пользовательской истории). Тестирование функциональной пригодности обычно проводится во время системного тестирования, но может проводиться и на поздних этапах интеграционного тестирования. Дефекты, обнаруженные в ходе этого тестирования, являются признаками того, что система не сможет удовлетворить потребности пользователя таким образом, который будет считаться приемлемым.

#### 4.2.3 Тестирование функциональной полноты

Тестирование функциональной полноты выполняется для определения покрытия конкретных задач и целей пользователя реализованной функциональностью. Трассируемость между элементами спецификации (например, требованиями, пользовательскими историями, сценариями использования) и реализованной функциональностью (например, функцией, компонентом, последовательностью действий) необходима для определения требуемой функциональной полноты. Измерение функциональной полноты может варьироваться в зависимости от конкретного уровня тестирования и/или используемого жизненного цикла разработки программного обеспечения. Например, функциональная полнота для гибкой разработки программного обеспечения может быть основана на реализованных пользовательских историях и наборах функциональности. Функциональная полнота для системного интеграционного тестирования может быть сосредоточена на охвате высокоуровневых бизнес-процессов.

Определение функциональной полноты обычно поддерживается инструментами управления тестированием, если тест-аналитик поддерживает трассируемость между тестовыми сценариями и элементами спецификации функциональности. Более низкие, чем ожидалось, уровни функциональной полноты являются признаками того, что система не была полностью реализована.

#### 4.2.4 Тестирование возможности взаимодействия

Тестирование возможности взаимодействия верифицирует обмен информацией между двумя или более системами или компонентами. Тесты нацелены на возможности обмена информацией и последующим использованием этой информации. Тестирование должно охватывать все предполагаемые целевые окружения (включая вариации оборудования, программного обеспечения, промежуточного ПО, операционной системы и т.д.), чтобы гарантировать, что обмен данными будет работать должным образом. В реальности это может быть осуществимо только для относительно небольшого количества окружений. В этом случае тестирование возможности взаимодействия может быть ограничено выбранной репрезентативной группой окружений. Тестирование возможности взаимодействия требует, чтобы комбинации

предполагаемых целевых окружений были определены, сконфигурированы и доступны команде тестирования. Затем эти окружения тестируются с использованием набора тестовых сценариев функциональной пригодности, которые проверяют различные точки обмена данными, присутствующие в окружении.

Возможность взаимодействия связана с тем, как различные компоненты и программные системы взаимодействуют друг с другом. Программное обеспечение с хорошими характеристиками возможности взаимодействия может быть интегрировано с рядом других систем без необходимости внесения серьезных изменений или существенного влияния на нефункциональное поведение. Количество изменений и усилия, необходимые для реализации и тестирования этих изменений, могут быть использованы в качестве измерения возможности взаимодействия.

Тестирование возможности взаимодействия программного обеспечения может, например, быть сосредоточено на следующих особенностях разработки функционала:

- Использование общеотраслевых стандартов коммуникации, таких как XML
- Способность автоматически определять коммуникационные потребности систем, с которыми происходит взаимодействие, и соответствующим образом настраивать их

Тестирование возможности взаимодействия может быть особенно важно для следующего:

- Готовые коммерческие программные продукты и инструменты
- Приложения на основе системы систем
- Системы, основанные на Интернете вещей
- Веб-сервисы с возможностью подключения к другим системам

Этот тип тестирования выполняется в ходе тестирования интеграции компонентов и системного интеграционного тестирования. На уровне системной интеграции этот тип тестирования проводится для определения того, насколько хорошо полностью разработанная система взаимодействует с другими системами. Поскольку системы могут взаимодействовать на нескольких уровнях, тест-аналитик должен разбираться в этих взаимодействиях и уметь создавать условия, в которых будут выполняться различные взаимодействия. Например, если две системы будут обмениваться данными, тест-аналитик должен уметь создавать необходимые данные и операции, необходимые для осуществления обмена данными. Важно помнить, что не все взаимодействия могут быть четко указаны в документах требований. Вместо этого, многие из этих взаимодействий будут определены только в документах по архитектуре и проектированию системы. Тест-аналитик должен быть способен и готов к изучению этой документации, чтобы определить точки обмена информацией между системами и между системой и ее окружением, чтобы убедиться, что все они протестированы. Такие методы, как эквивалентное разбиение, анализ граничных значений, таблицы решений, диаграммы переходов, тестирование по сценариям использования и попарное тестирование, применимы для тестирования возможности взаимодействия. Типичные обнаруженные дефекты включают неправильный обмен данными между взаимодействующими компонентами.

### 4.2.5 Оценка практичности

Тест-аналитики часто находятся в таком положении, когда необходимо координировать и поддерживать оценку практичности. Это может включать в себя определение тестов практичности или выполнение роли модератора, работающего с пользователями над проведением тестов. Для эффективного выполнения этой задачи тест-аналитик должен понимать основные аспекты, цели и подходы, связанные с этими видами тестирования. Пожалуйста, обратитесь к программе обучения продвинутого уровня ISTQB® Специалист по

тестированию практичности [ISTQB\_UT\_SYL] за подробностями, выходящими за рамки описания, представленного в этом разделе.

Важно понимать, почему пользователи могут испытывать трудности при использовании системы или не иметь положительного пользовательского опыта (UX) (например, при использовании программного обеспечения для сферы развлечения). Чтобы понять это, прежде всего, необходимо осознать то, что термин «пользователь» может применяться к широкому кругу различных типов людей, начиная от ИТ-специалистов и заканчивая детьми и людьми с ограниченными возможностями.

#### 4.2.5.1 Аспекты практичности

Ниже перечислены три аспекта, рассматриваемые в данном разделе:

- Практичность в соответствии со стандартом ISO 25010
- Пользовательский опыт (UX) как обобщение практичности
- Общедоступность как подхарактеристика практичности

#### Практичность

Тестирование практичности направлено на выявление дефектов программного обеспечения, которые влияют на способность пользователя выполнять задачи с помощью пользовательского интерфейса. Такие дефекты могут повлиять на способность пользователя добиваться результата эффективно, результативно, или с удовлетворением. Проблемы практичности могут привести к путанице, ошибкам, задержкам или полному невыполнению запросов со стороны пользователя.

Ниже перечислены подхарактеристики практичности [ISO 25010]; их определения см. в [ISTQB\_GLOSSARY]:

- Определимость пригодности (т.е. понятность)
- Изучаемость
- Работоспособность
- Эстетика пользовательского интерфейса (т.е. привлекательность)
- Защищенность от ошибок пользователя
- Общедоступность (см. ниже)

#### Пользовательский опыт (UX)

Оценка пользовательского опыта рассматривает весь опыт работы пользователя с объектом тестирования, а не только прямое взаимодействие. Это особенно важно для объектов тестирования, где такие факторы, как получение удовольствия и удовлетворенность пользователя, имеют решающее значение для успеха бизнеса.

Типичные факторы, влияющие на пользовательский опыт, включают в себя следующее:

- Имидж бренда (т.е. доверие пользователя к производителю)
- Интерактивное поведение
- Полезность объекта тестирования, включая системы помощи, поддержки и обучения

#### Общедоступность

Важно учитывать общедоступность программного обеспечения для тех, у кого есть особые потребности или ограничения в его использовании. К ним относятся люди с ограниченными возможностями. При тестировании общедоступности следует учитывать соответствующие

стандарты, такие как Руководство по обеспечению доступности веб-контента (WCAG), и законодательство, такое как Законы о дискриминации по инвалидности (Северная Ирландия, Австралия), Закон о равенстве 2010 года (Англия, Шотландия, Уэльс) и Раздел 508 (США). Общедоступность, как и практичность, должна учитываться на этапе проектирования. Тестирование часто происходит на уровне интеграции и продолжается на уровне системного тестирования и приемочного тестирования. Дефекты обычно обнаруживаются, когда программное обеспечение не соответствует установленным правилам или стандартам, определенным для программного обеспечения.

Типичные меры по улучшению общедоступности ориентированы на возможности, которые предоставляются пользователям с ограниченными возможностями для взаимодействия с приложением. Эти меры включают в себя следующее:

- Распознавание голоса для ввода данных
- Обеспечение того, чтобы нетекстовый контент, предоставляемый пользователю, имел эквивалентную текстовую альтернативу
- Возможность изменения размера текста без потери содержания или функциональности

Руководства по общедоступности помогают тест-аналитику, предоставляя источник информации и чек-листы, которые можно использовать для тестирования (примеры руководств по общедоступности приведены в [ISTQB\_UT\_SYL]). Кроме того, существуют инструменты и плагины для браузеров, помогающие тестировщикам выявлять проблемы общедоступности, такие как неправильный выбор цвета на веб-страницах, нарушающий положение по дальтонизму.

#### 4.2.5.2 Подходы к оценке удобства использования

Практичность, пользовательский опыт и общедоступность могут быть проверены с помощью одного или нескольких из следующих подходов:

- Тестирование практичности
- Рецензирование практичности
- Опросы и анкетирование пользователей

#### Тестирование практичности

Тестирование практичности оценивает легкость, с которой пользователи могут использовать или научиться использовать систему для достижения определенной цели в определенном контексте. Тестирование практичности направлено на измерение следующего:

- Результативность - способность объекта тестирования позволять пользователям достигать заданных целей с точностью и полнотой в определенном контексте использования
- Эффективность - способность объекта тестирования позволить пользователям затрачивать соответствующее количество ресурсов в зависимости от эффективности, достигнутой в определенном контексте использования.
- Удовлетворенность - способность объекта тестирования удовлетворять пользователей в определенном контексте использования

Важно отметить, что проектирование и конкретизация тестов практичности часто проводится тест-аналитиком в сотрудничестве с тестировщиками, обладающими специальными навыками тестирования практичности, и инженерами по проектированию практичности, понимающими процесс проектирования, ориентированного на человека (подробнее см. в [ISTQB\_UT\_SYL]).

## Рецензирования практичности

Инспекции и рецензирования — это вид тестирования, проводимый с точки зрения практичности, который помогает повысить уровень вовлеченности пользователя. Это может быть экономически эффективным за счет обнаружения проблем практичности в спецификациях требований и проектах на ранних этапах жизненного цикла разработки программного обеспечения. Эвристическая оценка (систематическая проверка дизайна пользовательского интерфейса на практичность) может быть использована для поиска проблем практичности в дизайне, чтобы их можно было решить в рамках итерационного процесса проектирования. Для этого необходимо, чтобы небольшая группа экспертов изучила интерфейс и оценила его соответствие признанным принципам практичности («эвристика»). Рецензирования более эффективны, когда пользовательский интерфейс нагляден. Например, снимки экрана обычно легче понять и интерпретировать, чем простое описание функциональности, предоставляемой конкретным экраном. Визуализация важна для адекватного анализа документации с точки зрения практичности.

## Опросы и анкетирование пользователей

Для сбора наблюдений и отзывов о поведении пользователя в системе могут применяться методы опроса и анкетирования. Стандартизированные и общедоступные опросы, такие как лист оценки удобства использования программного обеспечения (SUMI) и лист оценки и анализа веб-сайтов (WAMMI), позволяют провести сравнение с базой данных предыдущих измерений удобства использования. Кроме того, поскольку лист оценки удобства использования программного обеспечения обеспечивает осязаемые измерения удобства использования, это может обеспечить набор критериев завершения / приемки.

### 4.2.6 Тестирование переносимости

Тесты на переносимость относятся к уровню, в котором программный компонент или система могут быть перенесены в предназначенное для них окружение, либо как новая установка, либо из существующего окружения.

Классификация характеристик качества продукта по стандарту ISO 25010 включает следующие подхарактеристики переносимости:

- Устанавливаемость
- Приспособляемость
- Заменяемость

Задача по определению рисков и разработке тестов для характеристик переносимости разделяется между тест-аналитиком и техническим тест-аналитиком (см. [ISTQB\_ALTTA\_SYL] раздел 4.7).

#### 4.2.6.1 Тестирование устанавливаемости

Программное обеспечение проходит тестирование устанавливаемости, а для установки и удаления программного обеспечения в целевом окружении используются письменные процедуры.

Типичные цели тестирования, которые находятся в центре внимания тест-аналитика, включают:

- Валидация возможности успешной установки различных конфигураций программного обеспечения. В тех случаях, когда может быть настроено большое количество параметров, тест-аналитик может разработать тесты с использованием попарных методов тестирования, чтобы уменьшить количество

тестируемых комбинаций параметров и сосредоточиться на конкретных конфигурациях, представляющих интерес (например, часто используемых).

- Тестирование функциональной корректности процедур установки и удаления.
- Проведение тестов на функциональную пригодность после установки или деинсталляции для выявления любых дефектов, которые могли быть внесены (например, неправильные конфигурации, недоступные функции).
- Выявление проблем удобства использования в процедурах установки и удаления (например, для подтверждения того, что пользователям предоставляются понятные инструкции и обратная связь/сообщения об ошибках при выполнении процедуры).

#### 4.2.6.2 Тестирование приспособляемости

Тестирование приспособляемости проверяет, может ли данное приложение быть эффективно и результативно адаптировано для корректного функционирования во всех предполагаемых целевых окружениях (аппаратное, программное, межплатформенное, операционных систем, облачное и т.д.). Тест-аналитик поддерживает тестирование приспособляемости, определяя целевые окружения (например, версии различных поддерживаемых мобильных операционных систем, различные версии браузеров, которые могут быть использованы) и проектируя тесты, которые охватывают комбинации этих окружений. Затем целевые окружения тестируются с помощью набора тестовых сценариев функциональной пригодности, которые проверяют различные компоненты, присутствующие в окружении.

#### 4.2.6.3 Тестирование заменяемости

Тестирование заменяемости направлено на способность компонентов или версий программного обеспечения в системе заменяться на другие. Это может быть особенно актуально для системных архитектур, основанных на Интернете вещей, где замена различных аппаратных устройств и/или программных установок является обычным явлением. Например, аппаратное устройство, используемое на складе для регистрации и контроля количества запасов, может быть заменено на более совершенное аппаратное устройство (например, с лучшим сканером) или установленное программное обеспечение может быть обновлено до новой версии, позволяющей автоматически отправлять заказы на обновление запасов в систему поставщика.

Тестирование заменяемости может проводиться тест-аналитиком параллельно с тестированием функциональной интеграции, когда доступно более одного альтернативного компонента для интеграции в целостную систему.

## 5. Рецензирование – 120 минут

### Ключевые слова

Рецензирование на основе чек-листов.

### Цели обучения главы «Рецензирование»

#### 5.1 Введение

Цели обучения отсутствуют

#### 5.2 Использование чек-листов в рецензировании

- TA-5.2.1 (K3) Определить проблемы в спецификации требований в соответствии с информацией чек-листа, предоставленной в программе обучения
- TA-5.2.2 (K3) Определить проблемы в пользовательской истории в соответствии с информацией чек-листа, предоставленной в программе обучения

## 5.1 Введение

Тест-аналитики должны быть активными участниками процесса рецензирования, предоставляя свои уникальные точки зрения. При правильном подходе, рецензирования могут стать самым большим и экономически эффективным вкладом в общее качество работы.

## 5.2 Использование чек-листов в рецензировании

Рецензирование на основе чек-листов — это наиболее распространенный метод, используемый тест-аналитиком при рецензировании базиса тестирования. Чек-листы используются во время рецензирования, чтобы напомнить участникам о необходимости проверить конкретные пункты во время рецензирования. Они также могут помочь обеспечить независимость рецензирования (например, «Это тот же чек-лист, который мы используем для каждого рецензирования. Чек-лист используется не только для проверки именно вашего рабочего продукта»).

Рецензирование на основе чек-листа может проводиться в общем случае для всех рецензирования или может быть сосредоточено на конкретных характеристиках качества, областях или типах документов. Например, чек-лист может проверять общие свойства документа, такие как наличие уникального идентификатора, отсутствие указаний с пометкой «предстоит определить», правильность форматирования и подобные элементы. Конкретный чек-лист для документа требований может содержать проверки на правильное использование терминов «должен» и «следует», проверки на тестируемость каждого заявленного требования и т.д.

Формат требований также может указывать на тип используемого чек-листа. Документ с требованиями в формате повествовательного текста будет иметь другие критерии проверки, чем документ, основанный на диаграммах.

Чек-листы также могут быть ориентированы на определенный аспект, например:

- Набор навыков программиста/архитектора или набор навыков тестировщика - в случае тест-аналитика наиболее подходящим будет контрольный список набора навыков тестировщика
- Определенный уровень риска (например, в системах, критически важных для безопасности) - контрольные списки обычно включают конкретную информацию, необходимую для определения уровня риска
- Конкретная метод тестирования - контрольный список будет сосредоточен на информации, необходимой для конкретного метода (например, правила, которые должны быть представлены в таблице решений)
- Конкретные элементы спецификации, такие как требование, сценарий использования или пользовательская история - обсуждаются в следующих разделах и обычно имеют другую направленность, чем те, которые используются техническим тест-аналитиком для рецензирования кода или архитектуры

### 5.2.1 Рецензирование требований

Следующие элементы являются примером того, что может включать в себя чек-лист, ориентированный на требования:

- Источник требования (например, сотрудник, департамент)
- Тестируемость каждого требования
- Приоритет каждого требования
- Критерии приемки для каждого требования
- Наличие структуры сценариев использования, если применимо

- Уникальный идентификатор каждого требования, варианта использования и пользовательской истории
- Версионирование каждого требования/примера использования/пользовательской истории
- Трассируемость каждого требования к требованию бизнеса/маркетинга
- Трассируемость между требованиями и/или вариантами использования (если применимо)
- Использование согласованной терминологии (например, использование глоссария)

Важно помнить, что если требование не поддается тестированию, то есть оно определено таким образом, что тест-аналитик не может определить, как его протестировать, то в этом требовании есть дефект. Например, требование, которое гласит: «Программное обеспечение должно быть очень удобным для пользователя», не поддается тестированию. Как тест-аналитик может определить, является ли программное обеспечение удобным для пользователя или даже очень удобным? Если вместо этого требование гласит: «Программное обеспечение должно соответствовать стандартам практичности (удобства использования), изложенным в документе стандартов практичности, версия xxx», и если документ о стандартах практичности существует, то это тестируемое требование. Это также всеобъемлющее требование, потому что это одно требование относится к каждому элементу интерфейса. В данном случае одно требование может легко породить множество отдельных тестовых случаев в нетривиальном приложении. Трассируемость от этого требования или, возможно, от документа о стандартах практичности к тестовым сценариям также очень важна, потому что, если спецификация практичности изменится, все тестовые сценарии должны быть пересмотрены и обновлены по мере необходимости.

Требование также является нетестируемым, если невозможно определить, пройден тест или не пройден, или невозможно спроектировать тест, который может пройти или провалиться.

Простой чек-лист<sup>1</sup> для рецензирования примеров использования может включать следующие вопросы:

- Четко ли определено основное поведение (путь)?
- Выявлены ли все альтернативные сценарии (пути) с учетом обработки ошибок?
- Определены ли сообщения пользовательского интерфейса?
- Существует ли только одно основное поведение (должно быть, иначе существует множество вариантов использования)?
- Можно ли протестировать каждое поведение?

### 5.2.2 Рецензирование пользовательских историй

В гибкой методологии разработки программного обеспечения требования обычно принимают форму пользовательских историй. Эти истории представляют собой небольшие блоки наглядных функциональных возможностей. В то время как сценарий использования представляет собой пользовательскую операцию, которая проходит через несколько областей функциональности, пользовательская история — это более изолированная функциональность, и, как правило, ее масштаб оценивается по времени, необходимым для ее разработки. Чек-лист для пользовательской истории может включать следующее:

- Подходит ли история для цели итерации/спринта?

<sup>1</sup> Экзаменационный вопрос предоставит набор пунктов чек-листа сценария использования для ответа на него

- Написана ли история с точки зрения человека, который ее запрашивает?
- Определены ли критерии приемки и поддаются ли они тестированию?
- Является ли функциональность четко определенной и однозначной?
- Является ли эта история независимой от каких-либо других?
- Приоритизирована ли история?
- Соблюдается ли в истории общепринятый формат:

Как <тип пользователя>, я хочу <некоторая цель>, чтобы <некоторая причина> [Cohn04].

Если история описывает новый интерфейс, то целесообразно использовать общий чек-лист истории (такой, как приведенный выше) и подробный чек-лист пользовательского интерфейса.

### 5.2.3 Составление чек-листов

Чек-лист может быть составлен на основе следующего:

- Организация (например, учет политики компании, стандартов, конвенций, правовых ограничений)
- Усилия по проекту/разработке (например, направленность, технические стандарты, риски)
- Тип рецензируемого рабочего продукта (например, рецензии кода могут быть подобраны с учетом на конкретные языки программирования)
- Уровень риска рецензируемого рабочего продукта
- Используемые методы тестирования

Хорошие чек-листы позволяют найти проблемы, а также помогут начать обсуждение других элементов, которые могли быть не упомянуты в чек-листе. Использование комбинации чек-листов является эффективным способом чтобы гарантировать, что рецензирование позволяет достичь (или добиться) высококачественного рабочего продукта. Использование рецензирования на основе чек-листов с использованием стандартных чек-листов, упомянутых в программе обучения базового уровня, и разработка чек-листов, индивидуальных для конкретной организации, таких как показанные выше, поможет тест-аналитику быть эффективным при проведении рецензирования.

Более подробную информацию о рецензированиях и инспекциях см. в [Gilb93] и [Wieggers03].  
Дополнительные примеры контрольных списков можно получить из ссылок в Разделе 7.4.

## 6. Инструменты тестирования и автоматизация – 90 минут

### Ключевые слова

Автоматизированный сценарий тестирования, выполнение теста, подготовка тестовых данных, проектирование теста, тестирование на основе ключевых слов.

### Цели обучения главы «Инструменты тестирования и автоматизация»

#### 6.1 Введение

Цели обучения отсутствуют

#### 6.2 Тестирование на основе ключевых слов

ТА-6.2.1 (К3) Для заданного сценария определить соответствующие активности тест-аналитика в проекте по тестированию на основе ключевых слов

#### 6.3 Типы инструментов тестирования

ТА-6.3.1 (К2) Объяснить использование и типы инструментов тестирования, применяемых при проектировании теста, подготовке тестовых данных и выполнении теста

## 6.1 Введение

Инструменты тестирования могут значительно повысить эффективность и точность тестирования. В этой главе описаны инструменты тестирования и подходы к автоматизации, которые использует тест-аналитик. Следует отметить, что тест-аналитики работают вместе с разработчиками, инженерами по автоматизации тестирования и техническими тест-аналитиками для создания решений по автоматизации тестирования. В частности, тестирование на основе ключевых слов требует привлечения тест-аналитика и использования его опыта работы с бизнесом и функциональностью системы.

Дополнительная информация по теме автоматизации тестирования и роли инженера по автоматизации тестирования представлена в программе обучения ISTQB® Advanced Level Test Automation Engineer [ISTQB\_TAE\_SYL].

## 6.2 Тестирование на основе ключевых слов

Тестирование на основе ключевых слов является одним из основных подходов к автоматизации тестирования и предполагает предоставление тест-аналитиком основной информации: ключевых слов и данных.

Ключевые слова (иногда называемые словами действия) в основном, но не исключительно, используются для представления высокоуровневых бизнес-взаимодействий с системой (например, «отменить заказ»). Каждое ключевое слово обычно используется для представления ряда подробных взаимодействий между субъектом и тестируемой системой. Последовательности ключевых слов (включая соответствующие тестовые данные) используются для определения тестовых сценариев [Buwalda02].

В автоматизации тестирования ключевое слово реализуется в виде одного или нескольких выполняемых автоматизированных тестовых сценариев. Инструменты считывают тестовые сценарии, написанные в виде последовательности ключевых слов, которые вызывают соответствующие автоматизированные тестовые сценарии, реализующие функциональность ключевого слова. Автоматизированные сценарии тестирования реализуются по модульному принципу, что позволяет легко сопоставить их с конкретными ключевыми словами. Для реализации этих модульных автоматизированных сценариев тестирования необходимы навыки программирования.

Ниже перечислены основные преимущества тестирования на основе ключевых слов:

- Ключевые слова, относящиеся к конкретному приложению или бизнес-домену, могут быть определены экспертами в данной области. Это может сделать задачу спецификации тестовых сценариев более эффективной.
- Специалист, обладающий в основном знаниями в предметной области, может воспользоваться преимуществами выполнения автоматических тестовых сценариев (после того, как ключевые слова будут реализованы в виде автоматизированных сценариев тестирования) без необходимости понимать основной код автоматизации.
- Использование модульного метода написания позволяет инженеру по автоматизации тестирования эффективно поддерживать тестовые сценарии, когда происходят изменения в функциональности и интерфейсе тестируемого программного обеспечения [Bath14].
- Спецификации тестовых сценариев не зависят от их реализации.

Тест-аналитики обычно создают и поддерживают данные о ключевых словах/словах действиях. Они должны понимать, что для реализации ключевых слов необходимо разработать автоматизированный тестовый сценарий. После определения ключевых слов и используемых

данных автоматизатор тестирования (например, технический тест-аналитик или инженер по автоматизации тестирования) переводит ключевые слова бизнес-процесса и действия низкого уровня в автоматизированные тестовые сценарии.

Хотя тестирование на основе ключевых слов обычно проводится во время системного тестирования, разработка кода может начинаться раньше уже на этапе проектирования теста. В итеративной окружении, особенно при использовании непрерывной интеграции/непрерывного развертывания, разработка автоматизированных тестов является непрерывным процессом.

После создания ключевых слов и данных тест-аналитик берет на себя ответственность за выполнение автоматизированных сценариев тестирования, содержащих ключевые слова, и анализ любых возникающих отказов.

При обнаружении аномалии тест-аналитик должен помочь в изучении причины сбоя, чтобы определить, в чем заключается дефект: в ключевых словах, входных данных, самом автоматизированном сценарии тестирования или в тестируемом приложении. Обычно первым шагом в устранении неполадок является выполнение того же теста с теми же данными вручную, чтобы проверить, не кроется ли отказ в самом приложении. Если это не выявило отказ, тест-аналитик должен просмотреть последовательность тестов, приведших к отказу, чтобы определить, возникла ли проблема на предыдущем этапе (возможно, при введении неправильных входных данных), но дефект проявился только на более поздних этапах обработки. Если тест-аналитик не может определить причину отказа, информацию об устранении неполадок следует передать техническому тест-аналитику или разработчику для дальнейшего анализа.

## 6.3 Типы инструментов тестирования

Большая часть работы тест-аналитика требует эффективного использования инструментов. Эта эффективность повышается благодаря следующему:

- Понимание того, какие инструменты использовать
- Понимание, что инструменты могут повысить эффективность тестирования (например, помогая обеспечить лучшее покрытие за отведенное время).

### 6.3.1 Инструменты для проектирования тестов

Инструменты для проектирования тестов используются для помощи в создании тестовых сценариев и тестовых данных, применяемых для тестирования. Эти инструменты могут работать с определенными форматами документов требований, моделями (например, UML) или данными, предоставленными тест-аналитиком. Инструменты проектирования тестов часто разрабатываются и создаются для работы с определенными форматами и инструментами, например, с конкретными инструментами управления требованиями.

Инструменты проектирования тестов могут предоставлять тест-аналитику информацию для использования при определении типов тестов, необходимых для получения конкретного целевого уровня покрытия, уверенности в системе или действий по снижению рисков продукта. Например, инструменты дерева классификации генерируют (и отображают) набор комбинаций, необходимых для достижения полного покрытия на основе выбранного критерия покрытия. Эта информация затем может быть использована тест-аналитиком для определения тестовых сценариев, которые должны быть выполнены.

### 6.3.2 Инструменты подготовки тестовых данных

Инструменты подготовки тестовых данных могут обеспечить следующие преимущества:

- Проанализировать документ, например, документ с требованиями или даже исходный код, чтобы определить данные, необходимые во время тестирования для достижения определенного уровня покрытия.
- Взять набор данных из рабочей системы и «очистить» или обезличить его, чтобы удалить любую личную информацию, сохранив при этом внутреннюю целостность этих данных. Затем очищенные данные можно использовать для тестирования без риска утечки безопасности или неправомерного использования личной информации. Это особенно важно там, где требуются большие объемы реалистичных данных и где существуют риски безопасности и конфиденциальности данных.
- Генерировать синтетические тестовые данные из заданных наборов входных параметров (например, для использования в случайном тестировании). Некоторые из этих инструментов анализируют структуру базы данных, чтобы определить, какие входные данные потребуются от тест-аналитика.

### 6.3.3 Автоматизированные инструменты выполнения тестов

Инструменты выполнения тестов используются тест-аналитиками на всех уровнях тестирования для запуска автоматизированных тестов и проверки фактических результатов. Целью использования инструмента выполнения тестов обычно является одна или несколько из следующих задач:

- Снижения затрат (в плане усилий и/или времени)
- Запуска большего количества тестов
- Запуска одного и того же теста на многих окружениях
- Организация (обеспечение) повторяемости выполнения тестов
- Запуска тестов, которые невозможно выполнить вручную (например, тесты, проверяющие большие данные)

Эти цели часто пересекаются с основными целями увеличения покрытия при снижении затрат.

Возврат инвестиций в инструменты выполнения тестов обычно наиболее высока при автоматизации регрессионных тестов из-за низкого уровня ожидаемого обслуживания и многократного выполнения тестов. Автоматизация тестирования минимальной работоспособности (smoke tests) также может быть эффективным применением автоматизации из-за частого использования тестов и необходимости быстрого получения результатов тестирования.

Несмотря на то, что стоимость выше, возможность иметь автоматизированный способ оценки новой сборки в окружение непрерывной интеграции окупает эти затраты.

Инструменты выполнения тестов обычно используются при системном и интеграционном тестировании. Некоторые инструменты, в частности инструменты тестирования API, могут также использоваться при компонентном тестировании. Использование инструментов там, где они наиболее применимы, поможет повысить возврат инвестиций.

## 7. Ссылки

### 7.1 Стандарты

[ISO25010] ISO/IEC 25010 (2011) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models, Chapter 4

[ISO29119-4] ISO/IEC/IEEE 29119-4 Software and Systems Engineering – Software Testing – Part 4, Test Techniques, 2015

[OMG-DMN] Object Management Group: OMG® Decision Model and Notation™, Version 1.3, December 2019; url: [www.omg.org/spec/DMN/](http://www.omg.org/spec/DMN/), Chapter 8

[OMG-UML] Object Management Group: OMG® Unified Modeling Language®, Version 2.5.1, December 2017; url: [www.omg.org/spec/UML/](http://www.omg.org/spec/UML/)

[RTCA DO-178C/ED-12C] Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12C, 2013., Chapter 1

### 7.2 Документы ISTQB® и IREB

[IREB\_CPRES] IREB Certified Professional for Requirements Engineering Foundation Level Syllabus, Version 2.2.2, 2017

[ISTQB\_AL\_OVIEW] ISTQB® Advanced Level Overview, Version 2.0

[ISTQB\_ALTTA\_SYL] ISTQB® Advanced Level Technical Test Analyst Syllabus, Version 2019

[ISTQB\_FL\_SYL] ISTQB® Foundation Level Syllabus, Version 2018

[ISTQB\_GLOSSARY] Standard glossary of terms used in Software Testing, url: <https://glossary.istqb.org/>

[ISTQB\_TAE\_SYL] ISTQB® Advanced Level Test Automation Engineer Syllabus, Version 2017

[ISTQB\_UT\_SYL] ISTQB® Foundation Level Specialist Syllabus Usability Testing, Version 2018

### 7.3 Книги и статьи

[Bath14] Graham Bath, Judy McKay, «The Software Test Engineer's Handbook (2nd Edition)», Rocky Nook, 2014, ISBN 978-1-933952-24-6

[Beizer95] Boris Beizer, «Black-box Testing», John Wiley & Sons, 1995, ISBN 0-471-12094-4

[Black02] Rex Black, «Managing the Testing Process (2nd edition)», John Wiley & Sons: New York, 2002, ISBN 0-471-22398-0

[Black07] Rex Black, «Pragmatic software testing: Becoming an effective and efficient test professional», John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Black09] Rex Black, «Advanced Software Testing, Volume 1», Rocky Nook, 2009, ISBN 978-1-933-952-19-2

[Buwalda02] Hans Buwalda, «Integrated Test Design and Automation: Using the Test Frame Method», Addison-Wesley Longman, 2002, ISBN 0-201-73725-6

- [Chow1978] T.S. Chow, Testing Software Design Modeled by Finite-State Machines, IEEE Transactions on Software Engineering vol. SE-4, issue 3, May 1978, pp. 178-187
- [Cohn04] Mike Cohn, «User Stories Applied: For Agile Software Development», Addison-Wesley Professional, 2004, ISBN 0-321-20568-5
- [Copeland04] Lee Copeland, «A Practitioner's Guide to Software Test Design», Artech House, 2004, ISBN 1-58053-791-X
- [Craig02] Rick David Craig, Stefan P. Jaskiel, «Systematic Software Testing», Artech House, 2002, ISBN 1-580-53508-9
- [Forgács19] István Forgács, Attila Kovács, «Practical Test Design», BCS, 2019, ISBN 978-1-780-1747-23
- [Gilb93] Tom Gilb, Dorothy Graham, «Software Inspection», Addison-Wesley, 1993, ISBN 0-201-63181-4
- [Koomen06] Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon «TMap NEXT, for result driven testing», UTN Publishers, 2006, ISBN 90-72194-80-2
- [Kuhn16] Richard Kuhn et al, «Introduction to Combinatorial Testing», CRC Press, 2016, ISBN 978-0-429-18515-1
- [Mosley93] Daniel J. Mosley, The Handbook of MIS Application Software Testing, Yourdon Press, Prentice-Hall. 1993, ISBN 978-0-13-907007-5
- [Myers11] Glenford J. Myers, «The Art of Software Testing» 3rd Edition, John Wiley & Sons, 2011, ISBN: 978-1-118-03196-4
- [Offutt16] Jeff Offutt, Paul Ammann, Introduction to Software Testing» 2nd Edition, Cambridge University Press, 2016, ISBN 13: 978-1-107-17201-2,
- [vanVeenendaal12] Erik van Veenendaal, «Practical risk-based testing.» Product Risk Management: The PRISMA Method», UTN Publishers, 2012, ISBN 978-94-9098-607-0
- [Wiegers03] Karl Wiegers, «Software Requirements 2», Microsoft Press, 2003, ISBN 0-735-61879-8
- [Whittaker03] James Whittaker, «How to Break Software», Addison-Wesley, 2003, ISBN 0-201-79619-8
- [Whittaker09] James Whittaker, «Exploratory software testing: tips, tricks, tours, and techniques to guide test design», Addison-Wesley, 2009, ISBN 0-321-63641-4

## 7.4 Другие ссылки

Ссылки выше указывают на информацию, доступную в Интернете и других источниках. Несмотря на то, что эти ссылки были проверены на момент публикации данной программы обучения, ISTQB® не может нести ответственность, если ссылки больше не доступны.

- Глава 3
  - Czerwonka, Jacek: [www.pairwise.org](http://www.pairwise.org)
  - Defect taxonomy: [www.testingeducation.org/a/bsct2.pdf](http://www.testingeducation.org/a/bsct2.pdf)
  - Sample defect taxonomy based on Boris Beizer's work: [inet.uni2.dk/~vinter/bugtaxst.doc](http://inet.uni2.dk/~vinter/bugtaxst.doc)
  - Good overview of various taxonomies: [testingeducation.org/a/bugtax.pdf](http://testingeducation.org/a/bugtax.pdf)
  - Heuristic Risk-Based Testing By James Bach

- Exploring Exploratory Testing, Cem Kaner and Andy Tinkham,  
[www.kaner.com/pdfs/ExploringExploratoryTesting.pdf](http://www.kaner.com/pdfs/ExploringExploratoryTesting.pdf)
- Pettichord, Bret, «An Exploratory Testing Workshop Report»,  
[www.testingcraft.com/exploratorypettichord](http://www.testingcraft.com/exploratorypettichord)
- Глава 5
  - <http://www.tmap.net/checklists-and-templates>

## 8. Приложение А

Следующая таблица основана на полной таблице, представленной в стандарте ISO 25010. Она сосредоточена только на характеристиках качества, рассмотренных в программе обучения для тест-аналитика, и сравнивает термины, используемые в ISO 9126 (использовавшиеся в версии программы 2012 года), с терминами в новом стандарте ISO 25010 (используемыми в этой версии).

ISO/IEC 25010	ISO/IEC 9126-1	Примечания
<b>Функциональная пригодность</b>	<b>Функциональность</b>	
Функциональная полнота		
Функциональная корректность	Точность	
Функциональная пригодность	Пригодность	
	Возможность взаимодействия	Перемещено в термин Совместимость
<b>Практичность</b>		
Определимость пригодности	Понятность	Новое название
Изучаемость	Изучаемость	
Работоспособность	Работоспособность	
Защищенность от ошибок пользователя		Новая подхарактеристика
Эстетика пользовательского интерфейса	Привлекательность	Новое название
Общедоступность		Новая подхарактеристика
<b>Совместимость</b>		Новое определение
Возможность взаимодействия		
Сосуществование		Отражено в программе обучения технического тест-аналитика