Russian Software Testing
Qualifications Board

Сертифицированный тестировщик

Программа обучения Базового уровня

Версия 2018

International Software Testing Qualifications Board

История изменений

Версия	Дата	Содержание
RSTQB 2018	24 февраля 2019	Программа обучения Сертифицированный тестировщик Базового уровня, обновление 2018 Перевод на русский язык.
ISTQB 2018	27 апреля 2018	Основная версия.
ISTQB 2018	12 февраля 2018	Бета версия
ISTQB 2018	19 января 2018	Перекрестная проверка внутренней версии 3.0
ISTQB 2018	15 января 2018	Предварительная перекрестная проверка внутренней версии 2.9, включающая правки основной команды.
ISTQB 2018	9 декабря 2017	Альфа версия 2.5 – Техническое редактирование версии 2.0, новый контент не добавлен
ISTQB 2018	22 ноября 2017	Альфа версия 2.0 – Программа обучения Сертифицированный тестировщик Базового уровня обновление 2018 – см. Приложение C - Замечания к выпуску
ISTQB 2018	12 июня 2017	Альфа версия – Программа обучения Сертифицированный тестировщик Базового уровня обновление 2018 см. Приложение C - Замечания к выпуску
RSTQB 2011	13 апреля 2011	Программа обучения Сертифицированный тестировщик Базового уровня Перевод на русский язык
ISTQB 2011	1 апреля 2011	Программа обучения Сертифицированный тестировщик Базового уровня Выпуск сопровождения – см. Приложение E – Замечания к выпуску 2011
ISTQB 2010	30 марта 2010	Программа обучения Сертифицированный тестировщик Базового уровня Выпуск сопровождения – см. Приложение E – Замечания к выпуску 2010
RSTQB 2007	27 ноября 2007	Перевод на русский язык
ISTQB 2007	01 мая 2007	Программа обучения Сертифицированный тестировщик Базового уровня
ISTQB 2005	01 июля 2005	Программа обучения Сертифицированный тестировщик Базового уровня
ASQF V2.2	Июль 2003	Программа сертификации ASQF на специалиста по тестированию, базовый уровень, версия 2.2 “Lehrplan Grundlagen des Softwaretestens”
ISEB V2.0	25 февраля 1999	Программа сертификации ISEB на специалиста по тестированию, базовый курс, версия 2.0

Содержание

Благодарности

Перевод версии документа 2018 года выполнен Рабочей группой Базового Уровня АНО «Коллегия экспертов по качеству программного обеспечения» (Russian Software Testing Qualifications Board): Маргарита Трофимова (руководитель группы), Александр Александров (редактор), Екатерина Акулова, Екатерина Белая, Елена Костина, Александр Куцан, Антон Романов.

Этот документ официально опубликован Генеральной Ассамблеей ISTQB® 4 июня 2018.

Документ был подготовлен командой из International Software Testing Qualifications Board: Klaus Olsen (председатель), Tauhida Parveen (заместитель председателя), Rex Black (менеджер проекта), Debra Friedenberg, Judy McKay, Meile Posthuma, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh и Eshraka Zakaria.

Команда благодарит Rex Black и Dorothy Graham за техническое редактирование, команду редакторов, команду перекрестного рецензирования и членов коллегий за их предложения и вклад.

В редактировании, комментировании и голосовании по этой программе участвовали:

Tom Adams, Tobias Ahlgren, Xu Aiguo, Chris Van Bael, Katalin Balla, Graham Bath, Gualtiero Bazzana, Arne Becher, Veronica Belcher, Lars Hilmar Bjørstrup, Ralf Bongard, Armin Born, Robert Bornelind, Mette Bruhn-Pedersen, Geza Bujdoso, Earl Burba, Filipe Carlos, Young Jae Choi, Greg Collina, Alessandro Collino, Cui Zhe, Taz Daughtrey, Matthias Daigl, Wim Decoutere, Frans Dijkman, Klaudia Dussa-Zieger, Yonit Elbaz, Ofer Feldman, Mark Fewster, Florian Fieber, David Frei, Debra Friedenberg, Conrad Fujimoto, Pooja Gautam, Thorsten Geiselhart, Chen Geng, Christian Alexander Graf, Dorothy Graham, Michel Grandjean, Richard Green, Attila Gyuri, Jon Hagar, Kobi Halperin, Matthias Hamburg, Zsolt Hargitai, Satoshi Hasegawa, Berit Hatten, Wang Hongwei, Tamás Horváth, Leanne Howard, Chinthaka Indikadahena, J. Jayapradeep, Kari Kakkonen, Gábor Kapros, Beata Karpinska, Karl Kemminger, Kwanho Kim, Seonjoon Kim, Cecilia Kjellman, Johan Klintin, Corne Kruger, Gerard Kruijff, Peter Kunit, Hyeyong Kwon, Bruno Legeard, Thomas Letzkus, Alon Linetzki, Balder Lingegård, Tilo Linz, Hongbiao Liu, Claire Lohr, Ine Lutterman, Marek Majernik, Rik Marselis, Romanos Matthaios, Judy McKay, Fergus McLachlan, Dénes Medzihradszky, Stefan Merkel, Armin Metzger, Don Mills, Gary Mogyorodi, Ninna Morin, Ingvar Nordström, Adam Novak, Avi Ofer, Magnus C Ohlsson, Joel Oliviera, Monika Stocklein Olsen, Kenji Onishi, Francisca Cano Ortiz, Gitte Ottosen, Tuula Pääkkönen, Ana Paiva, Tal Pe'er, Helmut Pichler, Michaël Pilaeten, Horst Pohlmann, Andrew Pollner, Meile Posthuma, Vitalijs Puiso, Salvatore Reale, Stuart Reid, Ralf Reissing, Shark Ren, Miroslav Renda, Randy Rice, Adam Roman, Jan Sabak, Hans Schaefer, Ina Schieferdecker, Franz Schiller, Jianxiong Shen, Klaus Skafte, Mike Smith, Cristina Sobrero, Marco Sogliani, Murian Song, Emilio Soresi, Helder Sousa, Michael Sowers, Michael Stahl, Lucjan Stapp, Li Suyuan, Toby Thompson, Steve Toms, Sagi Traybel, Sabine Uhde, Stephanie Ulrich, Philippos Vakalakis, Erik van Veenendaal, Marianne Vesterdal, Ernst von Düring, Salinda Wickramasinghe, Marie Walsh, Søren Wassard, Hans Weiberg, Paul Weymouth, Hyungjin Yoon, John Young, Surong Yuan, Ester Zabar, и Karolina Zmitrowicz.

Рабочая группа Базового уровня (редакция 2018) International Software Testing Qualifications Board: Klaus Olsen (председатель), Tauhida Parveen (заместитель председателя), Rex Black (менеджер проекта), Dani Almog, Debra Friedenberg, Rashed Karim, Johan Klintin, Vipul Kocher, Corne Kruger, Sunny Kwon, Judy McKay, Thomas Müller, Igal Levi, Ebbe Munk, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms,

Stephanie Ulrich, Marie Walsh, Eshraka Zakaria, and Stevan Zivanovic. Основная команда благодарит команду редакторов и всех членов коллегий за их предложения.

Версия документа 2011 года создана Рабочей группой Базового уровня International Software Testing Qualifications Board: Thomas Muller (председатель), Debra Friedenberg.

Благодарим команду редакторов (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkönen, Eric Riou du Cosquier, Hans Shaefer, Stephanie Ulrich, Erik van Veendendaal), а также все Национальные коллегии за предложения к текущей версии программы обучения.

Версия документа 2010 года создана Рабочей группой Базового уровня International Software Testing Qualifications Board: Thomas Muller (председатель), Rahul Verma, Martin Klonk и Armin Beer, а также командой редакторов (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Tuula Pääkkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veendendaal) и всеми Национальными коллегиями с учетом их предложений.

Версия документа 2007 года создана Рабочей группой Базового уровня International Software Testing Qualifications Board: Thomas Muller (председатель), Dorothy Graham, Debra Friedenberg и Erik van Veendendaal, а также командой редакторов (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson и Wonil Kwon) и всеми Национальными коллегиями с учетом их предложений.

Версия документа 2005 года создана Рабочей группой Базового уровня International Software Testing Qualifications Board: Thomas Muller (председатель), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson и Erik van Veendendaal, а также командой редакторов и всеми Национальными коллегиями с учетом их предложений.

0 Предисловие к программе обучения

1.1 Цель этого документа

Программа обучения представляет собой основу международной сертификации на квалификацию Базового уровня в области тестирования программного обеспечения. ISTQB® распространяет эту программу:

1. Коллегиям для перевода ее на национальный язык и аккредитации организаторов обучения. Национальные коллегии могут приспособить программу обучения к особенностям конкретных языков и определить ссылки для адаптации к местным публикациям.
2. Сертификационным комиссиям для формирования экзаменационных вопросов на национальном языке адаптированные к целям обучения для каждой программы.
3. Организаторам обучения для разработки учебной программы и определения подходящих методов обучения.
4. Кандидатам на получение сертификатов для подготовки к экзамену (либо в рамках учебного курса, либо самостоятельно).
5. Международному сообществу разработки ПО и систем для повышения уровня профессионализма при тестировании ПО и систем, и использования как основы для книг и статей.

ISTQB® может разрешить другим лицам использовать эту программу обучения в своих целях при условии, что они обратятся и получат письменное разрешением от ISTQB.

1.2 Сертифицированный тестировщик ПО Базового уровня

Квалификация Базового уровня предназначена для всех, связанных с тестированием ПО, включая такие роли как тестировщики, тест аналитики, инженеры тестировщики, консультанты тестирования, руководители тестирования, тестировщики пользовательского приемочного тестирования и разработчики ПО. Эта квалификация также подходит для тех, кто желает получить понимание основ тестирования ПО, например, руководителей проектов, руководителей по качеству, руководителей разработки ПО, бизнес-аналитиков, ИТ-директоров и менеджеров-консультантов. Обладатели сертификата Базового уровня могут готовиться к более высокой квалификации в тестировании ПО.

ISTQB «Обзор базового уровня 2018» – это отдельный документ, который содержит следующую информацию:

- Бизнес-результаты для программы обучения
- Матрица, показывающая связь между бизнес-результатами и целями обучения
- Обзор программы обучения

1.3 Проверяемые цели обучения и необходимый уровень знаний

Цели обучения согласуются с бизнес-результатами и используются при сдаче экзамена для сертификации тестировщиков Базового уровня.

В основном все содержание этой программы обучения доступно проверке на уровне K1, за исключением введения и приложений. Это значит, что кандидату может быть предложено узнать,

запомнить и вспомнить термин или понятие, упомянутые в любой из шести глав. Уровни знаний конкретных целей обучения показаны в начале каждой главы и имеют классификацию:

- K1: запомнить
- K2: понять
- K3: применить

Детали и примеры для целей обучения приведены в Приложении B.

Определения всех терминов, перечисленные как ключевые слова сразу после заголовка главы, должны быть запомнены (K1), даже если явно не упоминаются в целях изучения.

1.4 Экзамен

Сертификационные экзамены Базового уровня основаны 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, за исключением введения и приложения. Стандарты, книги и другие учебные программы ISTQB включены в качестве ссылок, но их содержание, не рассматривается за рамками того, что обобщено в этой учебной программе из таких стандартов, книг и других учебных программ ISTQB.

Формат экзамена – тест с несколькими вариантами ответов. В нем 40 вопросов. Чтоб сдать экзамен, необходимо дать правильные ответы не менее, чем на 65% от всех вопросов (т.е. на 26 вопросов).

Экзамены можно сдавать как часть аккредитованного курса обучения или независимо (например, в экзаменационном центре или на открытом экзамене). Прохождение аккредитованного курса обучения не является необходимым условием для сдачи экзамена.

1.5 Аккредитация

Коллегии ISTQB, могут аккредитовать организаторам обучения материалы курсов, которые соответствуют этой Программе обучения. Организаторы обучения должны получить методические рекомендации по аккредитации от коллегии или организации, осуществляющей аккредитацию. Аккредитованный курс признается соответствующим этой Программе обучения и может включать в себя, как часть, сертификацию ISTQB.

1.6 Уровень детализации

Уровень детализации в этой Программе обучения позволяет проводить полноценное обучение и экзамены по всему миру. Для достижения этой цели программа обучения состоит из:

- Общих инструкций и целей, описывающих идею Базового уровня;
- Списка терминов, которые обучающиеся должны быть способны вспомнить;
- Целей изучения для каждой области знаний, описывающих результат изучения и образ мышления, который должен быть достигнут;
- Описания ключевых идей для изучения, включая такие источники, как одобренная литература или стандарты.

Содержание Программы обучения не является описанием всей области знаний в тестировании ПО. Оно отражает уровень детализации, который должен быть достигнут на курсах обучения

Базового уровня. Она сосредотачивается на концепциях и методах тестирования, которые могут применяться ко всем проектам программного обеспечения, включая проекты с гибкими методологиями. Эта учебная программа не содержит каких-либо определенных целей обучения, связанных с каким-либо конкретным жизненным циклом или методом разработки программного обучения, но в ней обсуждается, как эти концепции применяются в проектах по гибкой методологии, других типах итеративных и инкрементных жизненных циклов и в последовательных жизненных циклах.

1.7 Как эта программа организована

В программе шесть глав с содержанием для проверки. Заголовок верхнего уровня каждой главы определяет время, для более низких уровней время не указано. Для аккредитованных учебных курсов программа требует, как минимум 16,75 часов обучения, распределенных по шести главам следующим образом:

- Глава 1: Основы тестирования (175 мин)
- Глава 2: Тестирование в течение жизненного цикла ПО (100 мин)
- Глава 3: Статическое тестирование (135 мин)
- Глава 4: Методы тестирования (330 мин)
- Глава 5: Управление тестированием (225 мин)
- Глава 6: Инструменты тестирования (40 мин)

2. Основы тестирования – 175 мин

Ключевые слова

покрытие, отладка, дефект, ошибка, отказ, качество, обеспечение качества, первопричина, анализ тестирования, базис тестирования, тестовый сценарий, завершение тестирования, тестовое условие, контроль тестирования, тестовые данные, проектирование теста, выполнение теста, расписание выполнения тестов, реализация теста, мониторинг тестирования, объект тестирования, причина тестирования, тестовый предсказатель, планирование тестирования, процедура тестирования, набор тестов, тестирование, тестовое обеспечение, трассируемость, валидация, верификация

Цели обучения для главы «Основы тестирования»

1.1 Что такое тестирование?

FL-1.1.1 (K1) Определить основные цели тестирования

FL-1.1.2 (K2) Отличать тестирование от отладки.

1.2 Почему тестирование необходимо?

FL-1.2.1 (K2) Привести примеры почему тестирование необходимо.

FL-1.2.2 (K2) Описать, взаимосвязь между тестированием и обеспечением качества и привести примеры, как тестирование способствует повышению качества.

FL-1.2.3 (K2) Определить различие между ошибкой, дефектом и отказом.

FL-1.2.4 (K2) Определить различие между первопричиной дефекта и его эффектом.

1.3 Семь принципов тестирования

FL-1.3.1 (K2) Объяснить семь принципов тестирования.

1.4 Процесс тестирования

FL-1.4.1 (K2) Объяснить влияние контекста на процесс тестирования.

FL-1.4.2 (K2) Описать работы в тестировании и соответствующие задачи в процессе тестирования.

FL-1.4.3 (K2) Различать рабочие продукты, поддерживающие процесс тестирования.

FL-1.4.4 (K2) Объяснить значение поддержки трассируемости между требованиями и артефактами тестирования.

1.5 Психология тестирования

FL-1.5.1 (K1) Определить психологические факторы, которые влияют на успех тестирования.

FL-1.5.2 (K2) Объяснить разницу в мышлении, необходимую при работе тестировщика и разработчика.

2.1 Что такое тестирование?

Системы с программным обеспечением являются неотъемлемой частью нашей жизни, от бизнес-приложений (таких как банковское программное обеспечение) до потребительских товаров (таких как автомобили). Многие люди имели опыт использования программного обеспечения, которое не работало так, как ожидалось. Программное обеспечение, которое не работает корректно, может привести ко многим проблемам, включая потерю денег, времени или деловой репутации, и стать причиной травмы или смерти. Тестирование программного обеспечения – это способ оценить качество программного обеспечения и снизить риск отказа программного обеспечения.

Распространено заблуждение о тестировании, что оно состоит только из прогона тестов, т.е. выполнения программного обеспечения и проверки результатов. Как указано в разделе 1.4, тестирование программного обеспечения – это процесс, который состоит из множества различных активностей; и выполнение тестов (включая проверку результатов) является только одной из таких активностей. Процесс тестирования содержит также такие активности, как планирование, анализ, проектирование и реализация тестов, создание отчетов о ходе и результатах тестирования, а также оценка качества объекта тестирования.

Тестирование, во время которого задействуется выполнение тестируемого компонента или системы, называется динамическим тестированием. Другой вид тестирования, который не предполагает выполнение тестируемого компонента или системы, называется статическим тестированием. Таким образом, тестирование также включает рецензирование рабочих продуктов, таких как требования, пользовательские истории и исходный код.

Другое распространенное заблуждение о тестировании состоит в том, что оно полностью направлено на проверку требований, пользовательских историй или других спецификаций. Наравне с проверкой соответствия системы установленным требованиям, тестирование содержит в себе проверку, будет ли система удовлетворять потребности пользователей и других заинтересованных лиц в своей рабочей среде (средах).

Тестовые активности организованы и реализованы по-разному в различных жизненных циклах (см. раздел 2.1).

2.1.1 Основные цели тестирования

Для любого проекта цели тестирования могут включать:

- Оценку рабочих продуктов, таких как требования, пользовательские истории, проектирование и код
- Проверку, все ли указанные требования выполнены
- Проверку, завершен ли объект тестирования и работает, как ожидают пользователи и заинтересованные лица
- Создание уверенности в уровне качества объекта тестирования
- Предотвращение дефектов
- Обнаружение отказов и дефектов
- Предоставление заинтересованным лицам достаточной информации, позволяющей им принять обоснованные решения, особенно в отношении уровня качества объекта тестирования
- Снижение уровня риска ненадлежащего качества программного обеспечения (например, пропущенные сбои в работе)

- Соблюдение договорных, правовых или нормативных требований, или стандартов и/или проверка соответствия объекта тестирования таким требованиям и стандартам

Цели тестирования могут варьироваться в зависимости от контекста тестируемого компонента или системы, уровня тестирования и модели жизненного цикла разработки программного обеспечения. Эти различия могут включать:

- При компонентном тестировании одна из целей может заключаться в том, чтобы найти как можно больше сбоев, чтобы выявить и устранить основные дефекты на ранних стадиях. Другая цель может быть увеличением покрытия кода тестами компонентов.
- При приемочном тестировании одна из целей может заключаться в том, чтобы подтвердить, что система работает, как ожидалось, и удовлетворяет требованиям. Другая цель этого тестирования может быть предоставлением информации заинтересованным лицам о риске выпуска в установленный срок.

2.1.2 Тестирование и отладка

Стоит различать отладку и тестирование. Выполнение тестов может показать сбои, вызванные дефектами в программном обеспечении. Отладка – это деятельность разработки для нахождения, анализа и исправления таких дефектов. Последующее подтверждающее тестирование проверяет, устранены ли исправленные дефекты. В некоторых случаях тестировщики отвечают только за исходный тест и финальное подтверждающее тестирование, в то время как разработчики выполняют отладку и соответствующее компонентное тестирование. Однако в разработке по гибкой методологии и в некоторых других жизненных циклах тестировщики могут участвовать в отладке и тестировании компонентов.

Стандарт ИСО (ISO/IEC/IEEE 29119-1) содержит дополнительную информацию о концепциях тестирования программного обеспечения.

2.2 Почему тестирование необходимо?

Тщательное тестирование компонентов и систем, а также связанной с ними документации может помочь снизить риск отказов во время работы. Когда дефекты обнаружены и исправлены, это способствует качеству компонентов или систем. Также может потребоваться тестирование программного обеспечения на соответствие договорным или правовым требованиям, а также отраслевым стандартам.

2.2.1 Вклад тестирования в успех

На протяжении всей истории компьютеризации довольно часто поставка программного обеспечения и систем в эксплуатацию из-за наличия дефектов приводила к сбоям или иным образом не отвечала потребностям заинтересованных лиц. Однако, использование соответствующих методов тестирования может снизить частоту таких проблемных поставок, если эти методы применяются с соответствующим уровнем опыта тестирования, на соответствующих уровнях тестирования и соответствующих этапах жизненного цикла разработки программного обеспечения. Например:

- Наличие тестировщиков, участвующих в рецензировании требований или уточнении пользовательских историй, может обнаружить дефекты в этих рабочих продуктах. Выявление и устранение дефектов требований снижает риск разработки неправильной или нетестируемой функциональности.
- Наличие тесного сотрудничества тестировщиков с проектировщиками системы во время проектирования системы может повысить понимание архитектуры системы и способов

тестирования каждой стороной. Такое более глубокое понимание снижает риск возникновения фундаментальных дефектов проектирования и даёт возможность определить тесты на ранней стадии.

- Наличие тесного взаимодействия тестировщиков с программистами во время разработки кода может улучшить понимание кода и способов его тестирования каждой стороной. Такое более глубокое понимание снижает риск возникновения дефектов в коде и тестах.
- Наличие тестировщиков для валидации и верификации программного обеспечения до релиза может обнаружить сбои, которые в противном случае могли быть пропущены, и поддержать процесс устранения дефектов, вызвавших отказы (т.е. отладку). Это повышает вероятность того, что программное обеспечение удовлетворяет потребностям заинтересованных лиц и соответствует требованиям.

В дополнение к этим примерам достижение определенных целей тестирования (см. раздел 1.1.1) способствует общей успешной разработке и сопровождению программного обеспечения.

2.2.2 Обеспечение качества и тестирование

Хотя люди часто используют фразу «обеспечение качества» (или просто QA) для обозначения тестирования, тем не менее, обеспечение качества и тестирование не одно и то же, но эти понятия связаны. Их объединяет более крупное понятие - управление качеством. Управление качеством включает все виды деятельности, которые направляют и контролируют организацию в отношении качества. Помимо других видов деятельности, управление качеством содержит как обеспечение качества, так и контроль качества. Обеспечение качества, как правило, сосредоточено на соблюдении соответствующих процессов для обеспечения уверенности, что будут достигнуты соответствующие уровни качества. Если процессы осуществляются должным образом, рабочие продукты, созданные этими процессами, как правило, более высокого качества, что способствует предотвращению дефектов. Кроме того, использование анализа первопричин для выявления и устранения причин дефектов вместе с надлежащим применением результатов ретроспективы для улучшения процессов имеет большое значение для эффективного обеспечения качества.

Контроль качества включает различную деятельность, в том числе работы по тестированию, которые поддерживают достижение соответствующего уровня качества. Работы по тестированию являются частью общего процесса разработки и сопровождения программного обеспечения. Поскольку обеспечение качества связано с надлежащим выполнением всего процесса, обеспечение качества поддерживает надлежащее тестирование. Как указано в разделах 1.1.1 и 1.2.1, тестирование способствует достижению качества.

2.2.3 Ошибки, дефекты и отказы

Человек может совершить ошибку (просчет), которая может привести к появлению дефекта (недочета, помехи) в коде программы или в каком-либо другом сопутствующем продукте. Ошибка, приводящая к появлению дефекта в одном рабочем продукте, может вызвать ошибку, приводящую к появлению дефекта в связанном рабочем продукте. Например, ошибка извлечения требований может привести к дефекту требований, который затем приводит к ошибке программирования, что приводит к дефекту в коде.

Если дефект в коде выполняется, это может (но не обязательно во всех ситуациях) привести к отказу. Например, для некоторых дефектов требуются очень специфические входные данные или предварительные условия, чтобы вызвать отказ, который может произойти редко или никогда.

Ошибки могут возникать по многим причинам. Например,

- Нехватка времени
- Человек может ошибаться
- Неопытные или недостаточно квалифицированные участники проекта
- Недопонимание между участниками проекта, включая недопонимание требований и проектирования
- Сложность кода, проектирования, архитектуры, основной проблемы, которую надо решить, и или используемых технологий
- Непонимание внутрисистемных и межсистемных интерфейсов, особенно когда таких внутрисистемных и межсистемных взаимодействий много
- Новые, незнакомые технологии

Кроме отказов, вызванных дефектами в коде, отказы также могут быть вызваны условиями окружающей среды. Например, радиация, электромагнитные поля и загрязнения могут вызвать отказ в программно-аппаратных средствах или повлиять на выполнение программного обеспечения, изменяя условия работы аппаратных средств.

Не все неожиданные результаты теста являются отказами. Ложные срабатывания могут возникать из-за ошибок в способе выполнения тестов, из-за дефектов в тестовых данных, тестовой среде или другом тестовом окружении, а также по другим причинам. Может возникнуть и обратная ситуация, когда подобные ошибки или дефекты приводят к ошибочным негативным тестам. Ошибочные негативные тесты – это тесты, которые должны были обнаружить, но не обнаружили дефекты. Ошибочные позитивные тесты – это тесты, которые обнаружили дефекты, не являющиеся реальными дефектами.

2.2.4 Дефект, первопричина и эффект

Первопричины дефектов – это самые ранние действия или условия, которые способствовали созданию дефектов. Дефекты можно проанализировать для поиска первопричины, чтобы уменьшить возникновение подобных дефектов в будущем. Обращая внимание на наиболее существенные первопричины, анализ первопричин может привести к улучшению процессов, которые предотвратят появление значительного числа будущих дефектов.

Например, предположим, что одна строка неправильного кода приводит к жалобам клиента, что процентные платежи неверны. Дефектный код был написан для пользовательской истории, которая была неоднозначна, из-за того, что владелец продукта не понимал, как рассчитать проценты. Если в расчетах процентов существует большое количество дефектов, и первопричина этих дефектов в подобных недоразумениях, то можно обучить владельца продукта расчетам процентов, чтобы уменьшить такие дефекты в будущем.

В данном примере жалобы клиентов – это эффекты. Неправильные выплаты процентов являются отказами. Неправильный расчет в коде является дефектом, и он является результатом исходного дефекта - неоднозначности пользовательской истории. Первопричиной первоначального дефекта было отсутствие знаний у владельца продукта, который в свою очередь допустил ошибку при написании пользовательской истории. Процесс анализа первопричин обсуждается в программе подготовки ISTQB Экспертного уровня Тест Менеджер и в программе подготовки ISTQB Экспертного уровня Улучшение тестового процесса.

2.3 Семь принципов тестирования

За последние пятьдесят лет был предложен ряд принципов тестирования, которые являются общим руководством для тестирования в целом.

1. Тестирование демонстрирует наличие дефектов, а не их отсутствие

Тестирование может показать, что дефекты присутствуют, но не может доказать, что их нет. Тестирование снижает вероятность наличия дефектов, находящихся в программном обеспечении, но, даже если дефекты не были обнаружены, тестирование не доказывает его корректности.

2. Исчерпывающее тестирование недостижимо

Полное тестирование с использованием всех комбинаций вводов и предусловий физически невыполнимо, за исключением тривиальных случаев. Вместо попытки исчерпывающего тестирования должны использоваться анализ рисков, методы тестирования и расстановка приоритетов, чтобы сосредоточить усилия по тестированию.

3. Раннее тестирование сохраняет время и деньги

Для нахождения дефектов на ранних стадиях, как статические, так и динамические активности по тестированию должны быть начаты как можно раньше в жизненном цикле разработки программного обеспечения. Раннее тестирование иногда называют «сдвигом влево». Тестирование на ранних этапах жизненного цикла разработки программного обеспечения помогает сократить или исключить дорогостоящие изменения (см. раздел 3.1).

4. Кластеризация дефектов

Обычно небольшое количество модулей содержит большинство дефектов, обнаруженных во время тестирования перед выпуском, или отвечает за большинство эксплуатационных отказов. Предсказанные кластеры дефектов и фактические наблюдаемые кластеры дефектов в ходе тестирования или эксплуатации являются важными входными данными для анализа риска, используемого для сосредоточения усилий по тестированию (как указано в принципе 2).

5. Парадокс пестицида

Если одни и те же тесты будут выполняться снова и снова, в конечном счете эти тесты больше не будут находить новых дефектов. Для обнаружения новых дефектов может потребоваться изменение существующих тестов и тестовых данных, а также написание новых тестов. (Тесты больше не эффективны при обнаружении дефектов, так же как пестициды через некоторое время больше не эффективны при борьбе с вредителями). В некоторых случаях, таких как автоматизированное регрессионное тестирование, парадокс пестицидов имеет положительный результат, который является относительно низким числом регрессионных дефектов.

6. Тестирование зависит от контекста

Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение управления производством, в котором критически важна безопасность, тестируется иначе, чем мобильное приложение электронной коммерции (см. раздел 2.1).

7. Заблуждение об отсутствии ошибок

Некоторые организации ожидают, что тестировщики смогут выполнить все возможные тесты и найти все возможные дефекты, но принципы 2 и 1, соответственно, говорят нам, что это невозможно. Кроме того, ошибочно ожидать, что простое нахождение и исправление большого числа дефектов обеспечит успех системе. Например, тщательное тестирование всех указанных требований и исправление всех обнаруженных дефектов может привести к созданию системы, которая будет трудной в использовании, не будет соответствовать потребностям и ожиданиям пользователей или будет хуже по сравнению с другими конкурирующими системами.

2.4 Процесс тестирования

Нет универсального процесса тестирования программного обеспечения, но есть общие наборы тестовых активностей, без которых тестирование вряд ли достигнет поставленных целей. Эти наборы тестовых активностей и есть процесс тестирования. Правильный соответствующий процесс тестирования программного обеспечения в любой конкретной ситуации зависит от многих факторов. Какие тестовые активности участвуют в этом процессе и когда они происходят, можно обсудить в корпоративной стратегии тестирования.

2.4.1 Процесс тестирования в контексте

Контекстные факторы, которые влияют на корпоративный процесс тестирования, включают, в частности:

- Используемые модель жизненного цикла разработки программного обеспечения и проектные методологии
- Рассматриваемые уровни и типы тестирования
- Продуктовые и проектные риски
- Предметную область
- Ограничения, включая, в частности:
  - Бюджеты и ресурсы
  - Сроки
  - Сложность
  - Договорные и нормативные требования
- Организационные политики и практики
- Необходимые внутренние и внешние стандарты

В следующих разделах описываются общие аспекты организации процессов тестирования с точки зрения:

- Активностей и задач тестирования
- Рабочих продуктов тестирования
- Трассируемости между базисом тестирования и рабочими продуктами тестирования

Очень полезно, если базис тестирования (для любого рассматриваемого уровня и типа тестирования) имеет конкретные измеримые критерии покрытия. Критерии покрытия могут эффективно использоваться в качестве ключевых показателей эффективности (КПЭ) для управления мероприятиями, демонстрирующими достижение целей тестирования программного обеспечения (см. раздел 1.1).

Например, для мобильного приложения базис тестирования может состоять из списка требований и списка поддерживаемых мобильных устройств. Каждое требование является элементом базиса тестирования. Каждое поддерживаемое устройство также является элементом базиса тестирования. Для критериев покрытия может потребоваться по крайней мере один тестовый сценарий для каждого элемента базиса тестирования. После выполнения этих тестов заинтересованные лица узнают, выполнены ли указанные требования и наблюдались ли отказы на поддерживаемых устройствах.

Стандарт ИСО (ISO/IEC/IEEE 29119-2) содержит дополнительную информацию о процессах тестирования.

2.4.2 Активности и задачи в тестировании

Процесс тестирования состоит из следующих основных групп активностей:

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

Каждая группа активностей состоит из отдельных активностей, описание которых приводится в подразделах ниже. Каждая активность в рамках каждой группы активностей в свою очередь может состоять из нескольких отдельных задач, которые будут варьироваться от одного проекта или релиза к другому.

Более того, хотя многие из этих групп активностей могут выглядеть логически последовательными, они часто реализуются итеративно. Например, в гибкой методологии разработки задействуют небольшие итерации проектирования программного обеспечения, сборки и тестирование происходят постоянно, поддерживаемые непрерывным планированием. Поэтому в рамках этой методологии тестовые активности также повторяются непрерывно. Даже при последовательной разработке логически ступенчатая последовательность активностей будет включать перекрытие, сочетание, параллельное выполнение или пропуск этих активностей, поэтому, как правило, требуется их адаптация в контексте системы и проекта.

Планирование тестирования

Планирование тестирования состоит из активностей, которые определяют цели тестирования и подход к достижению целей тестирования с ограничениями, налагаемыми контекстом (например, определение подходящих методов тестирования и задач, а также формирование графика тестирования для соблюдения крайнего срока). Планы тестирования могут быть пересмотрены на основе обратной связи от мониторинга и контроля. Планирование тестирования далее объяснено в разделе 5.2.

Мониторинг и контроль тестирования

Мониторинг тестирования предполагает непрерывное сравнение фактического хода работы с планом тестирования, используя любые метрики мониторинга тестирования, определённые в плане тестирования. Контроль тестирования подразумевает принятие мер, необходимых для достижения целей плана тестирования (который может быть обновлен с течением времени). Мониторинг и контроль тестирования поддерживаются оценкой критериев выхода, которые в некоторых жизненных циклах называются критериями готовности (см. программу ISTQB Базовый уровень. Тестировщик в сфере Гибких методологий). Например, оценка критериев выхода для выполнения тестов в рамках заданного уровня тестирования может включать:

- Проверку результатов и журналов тестирования на соответствие заданным критериям покрытия
- Оценку уровня качества компонентов или систем на основе результатов и журналов тестирования

- Определение потребности в дополнительных тестах (например, потребуется написание и выполнение дополнительных тестов, если тесты, первоначально предназначенные для достижения покрытия определенного уровня риска продукта, не смогли этого сделать)

Прогресс тестирования по сравнению с планом сообщается заинтересованным лицам в отчетах о ходе тестирования, включая отклонения от плана и информацию для подтверждения какого-либо решения о прекращении тестирования.

Мониторинг и контроль тестирования более подробно описаны в разделе 5.3.

Анализ тестирования

В процессе анализа тестирования анализируют базис тестирования для определения тестируемых функций и установление соответствующих тестовых условий. Другими словами, анализ тестирования решает “что тестировать” с точки зрения измеримых критериев покрытия.

Анализ тестирования состоит из следующих основных активностей:

- Анализ базиса тестирования, применимого к рассматриваемому уровню тестирования. Базисом тестирования могут быть, например:
  - Спецификации требований, такие как бизнес-требования, функциональные требования, системные требования, пользовательские истории, бизнес-потребности, сценарии использования системы, или аналогичные рабочие продукты, которые описывают функциональные и нефункциональные компоненты или поведение системы
  - Информация о проектировании и реализации, такая как диаграммы или документы архитектуры системы или программного обеспечения, спецификации проектирования, потоки вызовов, диаграммы моделирования (например, UML или диаграммы «сущность-связь»), спецификации интерфейсов или аналогичные рабочие продукты, которые определяют структуру компонента или системы
  - Реализация самого компонента или системы, включая код, метаданные и запросы базы данных, а также интерфейсы
  - Отчеты анализа рисков, в которых могут быть рассмотрены функциональные, нефункциональные и структурные аспекты компонента или системы
- Оценка базиса тестирования и элементов тестирования для выявления дефектов различных типов, таких как:
  - Неоднозначность
  - Пропуски
  - Несоответствие
  - Неточность
  - Противоречивость
  - Избыточные утверждения
- Определение свойств и совокупность свойств для тестирования
- Установление и приоритизация тестовых условий для каждого свойства на основе анализа базиса тестирования, с учетом функциональных, нефункциональных и структурных характеристик, других бизнес и технических факторов, а также уровней рисков

- Обеспечение двунаправленной трассируемости между каждым элементом базиса тестирования и соответствующими тестовыми условиями (см. разделы 1.4.3 и 1.4.4)

Применение методов тестирования на основе черного ящика, белого ящика и на основе опыта может быть полезно в процессе анализа тестирования (см. главу 4) для уменьшения вероятности пропуска важных и определения более ясных и точных тестовых условий.

Выявление дефектов в ходе анализа тестирования является важным потенциальным преимуществом, особенно если процесс рецензирования не используется и/или если процесс тестирования тесно связан с процессом рецензирования. Такие активности по анализу тестирования не только проверяют, являются ли требования согласованными, сформулированными должным образом и полными, но также проверяют, правильно ли требования отражают потребности клиентов, пользователей и других заинтересованных лиц. Например, такие методы как разработка, основанная на описании поведения (BDD) и разработка через приемочные тесты (ATDD), которые затрагивают формирование тестовых условий и тестовых сценариев из пользовательских историй и критериев приемки перед кодированием, проверяют, корректируют и выявляют дефекты в пользовательских историях и критериях приемки (см. программу ISTQB Базовый уровень. Тестировщик в сфере Гибких методологий).

Проектирование тестов

Во время проектирования тестов тестовые условия воплощаются в высокоуровневые тестовые сценарии, наборы высокоуровневых тестовых сценариев и другое тестовое обеспечение. Так, анализ тестирования отвечает на вопрос «что тестировать?», а проектирование тестов отвечает на вопрос «как тестировать?».

Проектирование тестов состоит из следующих основных активностей:

- Проектирования и приоритизации тестовых сценариев и наборов тестовых сценариев
- Определения необходимых тестовых данных для поддержки тестовых условий и тестовых сценариев
- Проектирования тестового окружения и определения необходимой инфраструктуры и инструментов
- Отражения двунаправленной трассируемости между базисом тестирования, тестовыми условиями, тестовыми сценариями и процедурами тестирования (см. раздел 1.4.4)

Воплощение тестовых условий в тестовые сценарии и в наборы тестовых сценариев во время проектирования тестов часто включает использование методов тестирования (см. главу 4).

Так же, как и при анализе тестирования, проектирование тестов может привести к выявлению аналогичных типов дефектов в базисе тестирования. Кроме того, как и при анализе тестирования, выявление дефектов при проектировании тестов является важным потенциальным преимуществом.

Реализация тестов

Во время реализации тестов создается и/или подготавливается необходимое тестовое обеспечение для выполнения тестов, включая упорядочивание тестовых сценариев в процедурах тестирования. Таким образом, проектирование тестов отвечает на вопрос «как проверить?», в то время как реализация тестов отвечает на вопрос: «у нас теперь есть все для запуска тестов?». Реализация тестов – это активность, во время которой процедуры или сценарии тестирования выстраиваются в определенном порядке, чтобы облегчить выполнение тестов.

Реализация тестов состоит из следующих основных активностей:

- Разработка и расстановка приоритетов процедур тестирования и, возможно, создание автоматизированных сценариев тестирования
- Создание наборов тестов из процедур тестирования и (при наличии) автоматизированных сценариев тестирования
- Организация наборов тестов с расписанием выполнения тестов таким образом, чтобы тесты выполнялись эффективно (см. раздел 5.2.4)
- Построение тестового окружения (в том числе, возможно, тестовые стенды, сервисы виртуализации, симуляторы и другие элементы инфраструктуры) и проверка правильности настройки всего необходимого
- Подготовка тестовых данных и правильная загрузка их в тестовое окружение
- Проверка и обновление двунаправленной трассируемости между базисом тестирования, тестовыми условиями, тестовыми сценариями, процедурами тестирования и наборами тестов (см. раздел 1.4.4)

Часто задачи по проектированию тестов и реализации тестов объединяют.

При исследовательском тестировании и других типах тестирования, основанных на опыте, разработка и реализация тестов могут выполняться и документироваться как часть выполнения тестов. Исследовательское тестирование может основываться на концепциях тестирования (составленных в рамках анализа тестирования), и исследовательские тесты выполняются немедленно по мере их разработки и реализации (см. раздел 4.4.2).

Выполнение тестов

Во время выполнения тестов, наборы тестов запускаются в соответствии с расписанием выполнения тестов.

Выполнение тестов состоит из следующих основных активностей:

- Запись идентификаторов и версий элемента (-ов) тестирования или объекта тестирования, инструмента (-ов) тестирования, и тестового обеспечения
- Выполнение тестов вручную или с помощью инструментов выполнения тестов
- Сравнение фактических и ожидаемых результатов
- Анализ отклонений для установления их вероятных причин (например, отказы могут произойти из-за дефектов в коде, но также могут возникнуть ложные срабатывания (см. раздел 1.2.3))
- Составление отчетов о дефектах на основе наблюдаемых отказов (см. раздел 5.6)
- Протоколирование результатов выполнения тестов (например, пройден, не пройден, блокировка)
- Повторение тестовых действий, результаты которых привели к каждому из отклонений, либо в рамках запланированного тестирования (например, выполнение исправленного теста, подтверждающее тестирование и/или регрессионное тестирование)
- Проверка и обновление двунаправленной трассируемости между базисом тестирования, тестовыми условиями, тестовыми сценариями, процедурами тестирования и результатами тестирования

Завершение тестирования

Активности по завершению тестирования собирают данные из выполненных активностей тестирования для обобщения опыта, тестового обеспечения и любой другой соответствующей

информации. Активности по завершению тестирования происходят на вехах проекта, например, по завершению выпуска релиза программного обеспечения системы, завершению (отмене) проекта тестирования, окончанию итерации проекта с гибкой методологией (например, как часть итогового митинга), завершению уровня тестирования, или завершению сопровождения релиза.

Завершение тестирования состоит из следующих основных активностей:

- Проверка закрытия всех отчетов о дефектах, входящих запросов на изменение или набора задач продукта для любых дефектов, которые остаются не реализованными в конце выполнения тестирования
- Создание сводного отчета по тестированию для передачи заинтересованным лицам
- Завершение и архивирование тестового окружения, тестовых данных, инфраструктуры тестирования и другого тестового обеспечения для последующего использования
- Передача тестового обеспечения команде сопровождения, другим проектным командам, и/или другим заинтересованным лицам, которые могут извлечь выгоду из его использования
- Анализ полученных уроков для определения изменений, необходимых для будущих итераций, релизов и проектов
- Использование собранной информации для повышения зрелости процесса тестирования

2.4.3 Рабочие продукты тестирования

Создание рабочих продуктов тестирования — это часть процесса тестирования. Поскольку существует значительное различие в том, как организации реализовывают процесс тестирования, есть существенные различия и в типах рабочих продуктах, созданных в ходе этого процесса - как эти рабочие продукты организованы и управляются, и в названиях, используемых для этих рабочих продуктов. Эта программа соответствует описанному выше процессу тестирования, а также рабочим продуктам, описанным в этой программе и в Стандартном глоссарии терминов ISTQB. Стандарт ИСО (ISO/IEC/IEEE 29119-3) может также служить руководством для создания рабочих продуктов тестирования.

Используя средства управления тестированием и управления дефектами, можно отслеживать и управлять многими рабочими продуктами тестирования, описанными в этом разделе (см. главу 6).

Рабочие продукты планирования тестирования

Рабочие продукты планирования тестирования – это один или несколько планов тестирования. План тестирования содержит информацию о базисе тестирования, с которым другие рабочие продукты тестирования будут связаны через информацию о трассируемости (см. ниже и раздел 1.4.4), а также критерии выхода (или определение готовности), которые будут использоваться во время мониторинга и контроля тестирования. Планы тестирований описаны в разделе 5.2.

Рабочие продукты мониторинга и контроля тестирования

Рабочие продукты мониторинга и контроля тестирования – это обычно различные типы отчетов о тестировании, включая отчеты о ходе тестирования (выпускаемые оперативно и/или на регулярной основе) и сводные отчеты о тестировании (выпускаемые на различных этапах завершения). Все отчеты о тестировании должны обеспечить соответствующей аудитории подробные сведения о ходе тестирования на дату отчета, включая обобщение результатов выполнения тестирования после их получения.

Рабочие продукты мониторинга и контроля тестирования также должны учитывать интересы управления проектами, такие как завершение задач, распределение и использование ресурсов и трудозатрат.

Мониторинг и контроль тестирования, а также рабочие продукты, созданные в ходе этих активностей, более подробно описаны в разделе 5.3 данной программы.

Рабочие продукты анализа тестирования

Рабочие продукты анализа тестирования состоят из определенных и приоритизированных тестовых условий, каждое из которых в идеале двунаправленно прослеживается до конкретного элемента (элементов) базиса тестирования, который они покрывают. Для исследовательского тестирования анализ тестирования может включать и создание концепций тестирования. Анализ тестирования может также привести к обнаружению и составлению отчетов о дефектах в базисе тестирования.

Рабочие продукты проектирования тестов

Результатом проектирования тестов являются тестовые сценарии и наборы тестовых сценариев для выполнения тестовых условий, определенных в анализе тестирования. Часто хорошей практикой считается проектирование высокоуровневых тестовых сценариев без конкретных значений для входных данных и ожидаемых результатов. Такие высокоуровневые тестовые сценарии многократно используются для нескольких циклов тестирования с различными конкретными данными, но при этом все еще достаточно документируют область применения тестового сценария. В идеале, каждый тестовый сценарий двунаправленно прослеживается к тестовым условиям, которые он покрывает.

Проектирование тестов также приводит к разработке и/или установлению необходимых тестовых данных, к проектированию тестового окружения и определению инфраструктуры и инструментов, хотя объем документирования этих результатов может существенно различаться.

Тестовые условия, определенные в анализе тестирования, могут быть дополнительно уточнены при проектировании тестов.

Рабочие продукты реализации тестов

Рабочие продукты реализации тестов включают:

- Процедуры тестирования и последовательности этих процедур тестирования
- Наборы тестов
- Расписания выполнения тестов

В идеале после завершения реализации тестов достижение критериев покрытия, установленных в плане тестирования, может быть продемонстрировано посредством двунаправленной трассируемости между процедурами тестирования и конкретными элементами базиса тестирования, с помощью тестовых сценариев и тестовых условий.

В некоторых случаях реализация тестов включает создание рабочих продуктов, использующих или используемых инструментами, такими как службы виртуализации и автоматизированные тестовые сценарии.

Реализация тестов также может привести к созданию и проверке тестовых данных и тестового окружения. Полнота документирования результатов проверки данных и/или окружения может существенно различаться.

Тестовые данные служат для присвоения конкретных значений входным данным и ожидаемым результатам тестовых сценариев. Такие конкретные значения вместе с точными указаниями об использовании конкретных значений превращают высокоуровневые тестовые сценарии в

исполняемые низкоуровневые тестовые сценарии. Один и тот же высокоуровневый тестовый сценарий может использовать разные тестовые данные при выполнении в разных релизах объекта тестирования. Конкретные ожидаемые результаты, связанные с конкретными тестовыми данными, определяются с помощью тестового предсказателя.

При исследовательском тестировании во время выполнения тестов могут создаваться некоторые рабочие продукты для разработки и реализации тестов, хотя степень документирования исследовательских тестов (и их трассируемость до конкретных элементов базиса тестирования) может существенно различаться.

Тестовые условия, определенные в анализе тестирования, могут быть дополнительно уточнены при реализации тестов.

Рабочие продукты выполнения тестов

Рабочие продукты выполнения тестов включают:

- Документацию о состоянии отдельных тестовых сценариев или процедур тестирования (например, готов к запуску, пройден, не пройден, блокирован, осознанный пропуск и т. д.)
- Отчеты о дефектах (см. раздел 5.6)
- Документацию о том, какие элемент(ы) теста, объект(ы) тестирования, инструменты тестирования, и тестовое обеспечение были задействованы в тестировании

В идеале, как только выполнение тестов завершено, состояние каждого элемента базиса тестирования можно определить и сообщить через двунаправленную трассируемость к соответствующим процедурам тестирования. Например, можно указать, для каких требований успешно выполнены все запланированные тесты, для каких требований не прошли тесты и/или имеют связанные с ними дефекты, для каких требований запланировали тесты, которые ожидают выполнения. Это позволяет проверить, что критерии покрытия были достигнуты, и позволяет в понятных формулировках сообщать о результатах тестирования заинтересованным лицам.

Рабочие продукты завершения тестирования

Рабочие продукты завершения тестирования состоят из сводных отчетов тестирования, мероприятий по улучшению последующих проектов или итераций (например, ретроспектива после проекта с гибкой методологией), запросов на изменение или набора задач продукта, и окончательного тестового обеспечения.

2.4.4 Трассируемость между базисом тестирования и рабочими продуктами тестирования

Как указано в разделе 1.4.3, рабочие продукты тестирования и названия этих рабочих продуктов существенно различаются. Независимо от этих различий для осуществления эффективного мониторинга и контроля тестирования важно установить и поддерживать трассируемость на протяжении всего процесса тестирования между каждым элементом базиса тестирования и различными рабочими продуктами тестирования, связанных с этим элементом. В дополнение к оценке покрытия тестирования, хорошая трассируемость поддерживает:

- Анализ влияния изменений
- Проведение тестирования, поддающееся аудиту
- Достижение критериев управления ИТ
- Улучшение понятности отчетов о ходе тестирования и сводных отчетов, включая статус элементов базиса тестирования (например, требования, для которых прошли тесты, требования, для которых тесты провалились, и требования, ожидающие тестирования)

- Согласование технических аспектов тестирования с заинтересованными лицами в терминах, которые они могут понять
- Предоставление информации для оценки качества продукции, потенциала процессов и развития проекта в соответствии с бизнес-целями

Некоторые инструменты управления тестированием предоставляют модели рабочих продуктов тестирования, которые соответствуют части или всем рабочим продуктам тестирования, изложенных в этом разделе. Некоторые организации создают собственные системы управления для организации рабочих продуктов и обеспечения требуемой трассируемости информации.

2.5 Психология тестирования

Люди принимают участие в разработке программного обеспечения, в том числе и тестировании программного обеспечения. Таким образом, психология человека оказывает важное влияние на тестирование программного обеспечения.

2.5.1 Психология человека и тестирование

Выявление дефектов во время статического тестирования, такого как рецензирование требований или сессий уточнения пользовательских историй, или выявление отказов во время выполнения динамического теста может быть воспринято как критика продукта и его автора. Элемент человеческой психологии, называемый предвзятостью подтверждения, может затруднить принятие информации, которая не совпадает с текущими убеждениями. Например, поскольку разработчики ожидают, что их код будет правильным, у них есть предвзятость подтверждения, которое затрудняет принятие того, что код неправильный. В дополнение к предвзятости подтверждения, другие стереотипы людей могут затруднить понимание или принятие информации, полученной в результате тестирования. Кроме того, общепринятой человеческой чертой является то, что виноват носитель плохих новостей, а информация, полученная в результате тестирования, часто содержит плохие новости.

В результате этих психологических факторов некоторые люди могут воспринимать тестирование как разрушительную деятельность, даже если она в значительной степени способствует развитию проекта и качеству продукта (см. разделы 1.1 и 1.2). Чтобы попытаться уменьшить эти восприятия, информацию о дефектах и отказах следует сообщать конструктивным образом. Таким образом, напряженность между тестировщиками и аналитиками, владельцами продуктов, дизайнерами и разработчиками можно уменьшить. Это относится как к статическому, так и к динамическому тестированию.

Тестировщики и руководители по тестированию должны иметь хорошие коммуникационные навыки, чтобы иметь возможность эффективно сообщать о дефектах, отказах, результатах тестирования, ходе тестирования, и рисках, а также строить позитивные отношения с коллегами. Следующие примеры хорошо передают способы общения:

- Начните с сотрудничества, а не сражения. Напомните каждому об общей цели улучшения качества системы.
- Подчеркните преимущества тестирования. Например, информация о дефектах может помочь авторам улучшить свои рабочие продукты и навыки. Для организации, дефекты, найденные и зафиксированные во время тестирования, сохранят время и деньги и уменьшат общий риск качества продукции.
- Сообщайте результаты тестирования и другие выводы нейтральным, сфокусированном на фактах, способом, без критики автора. Пишите объективные и фактические отчеты о дефектах и выводах рецензирования.

- Попытайтесь понять, что другие люди чувствуют, и причины их негативной реакции на информацию.
- Убедитесь, что другой человек понял, что вы сказали, и наоборот.

Характерные цели тестирования обсуждались ранее (см. раздел 1.1). Четкое определение правильного набора целей тестирования имеет важные психологические последствия. Большинство людей склонно корректировать свои планы согласно целям, установленным командой, руководством и другими заинтересованными лицами. Также важно, чтобы тестировщики придерживались этих целей с минимальными личными предубеждениями.

2.5.2 Мышление тестировщика и разработчика

Разработчики и тестировщики часто думают по-разному. Основной целью разработчика является проектирование и создание продукта. Как обсуждалось ранее, цели тестирования включают верификацию и валидацию продукта, поиск дефектов до релиза и т. д. Это разные наборы целей, которые требуют разного образа мышления. Объединение этих образов мышления вместе помогает достичь более высокого уровня качества продукции.

Образ мышления отражает предположения человека и предпочтительные методы как для принятия решений, так и для решения проблем. Образ мышления тестировщика должен включать любопытство, профессиональный пессимизм, критический взгляд, внимание к деталям и мотивацию хороших и позитивных коммуникаций и отношений. Образ мышления тестировщика имеет тенденцию к развитию по мере приобретения опыта тестировщиком.

Образ мышления разработчика может включать в себя некоторые элементы образа мышления тестировщика, но успешные разработчики чаще заинтересованы в разработке и создании решений, чем в рассмотрении того, что может быть неправильно в этих решениях. Кроме того, подтверждение предвзятости затрудняет поиск ошибок в собственной работе.

При правильном образе мышления разработчики могут протестировать свой собственный код. Различные модели жизненного цикла разработки программного обеспечения часто имеют разные способы организации тестировщиков и мероприятий по тестированию. Выполнение некоторых действий независимыми тестировщиками повышает эффективность обнаружения дефектов, что особенно важно для больших, сложных или критически важных для безопасности систем. Независимые тестировщики привносят перспективу, которая отличается от перспективы авторов продукта работы (т.е. бизнес-аналитиков, владельцев продукта, дизайнеров и программистов), поскольку их предубеждения отличаются от авторских.

3. Тестирование в течение жизненного цикла разработки ПО – 100 мин

Ключевые определения

приемочное тестирование, альфа-тестирование, бета-тестирование, готовое программное обеспечение, тестирование интеграции компонентов, компонентное тестирование, подтверждающее тестирование, контрактное приемочное тестирование, функциональное тестирование, анализ влияния, интеграционное тестирование, тестирование в период сопровождения, нефункциональное тестирование, эксплуатационное приемочное тестирование, регрессионное тестирование, нормативное приемочное тестирование, каскадная модель разработки ПО, системное интеграционное тестирование, системное тестирование, базис тестирования, тестовый сценарий, тестовое окружение, уровень тестирования, объект тестирования, причина тестирования, тип тестирования, пользовательское приёмочное тестирование, тестирование методом белого ящика

Цели обучения для главы «Тестирование в течение жизненного цикла разработки ПО»

2.1. Модели жизненного цикла разработки ПО

FL-2.1.1. (K2) Объяснить взаимосвязь между мероприятиями по разработке программного обеспечения и мероприятиями по тестированию в жизненном цикле разработки программного обеспечения.

FL-2.1.2. (K1) Определить причины, по которым модели жизненного цикла разработки программного обеспечения должны быть адаптированы к контексту характеристик проекта и продукта.

2.2. Уровни тестирования

FL-2.2.1. (K2) Сравнить различные уровни тестирования с точки зрения целей, базиса тестирования, объектов тестирования, типичных дефектов и отказов, а также подходов и обязанностей.

2.3. Типы тестирования

FL-2.3.1. (K2) Сравнить функциональное, нефункциональное и тестирование методом белого ящика.

FL-2.3.2. (K1) Осознать, что функциональные, нефункциональные, а также тесты, исполняемые методом белого ящика, выполняются на любом уровне тестирования.

FL-2.3.3. (K2) Сравнить цели подтверждающего и регрессионного тестирования.

2.4. Тестирование в период сопровождения

FL-2.4.1. (K2) Обобщить предпосылки для тестирования в период сопровождения.

FL-2.4.2. (K2) Объяснить значение анализа влияния для тестирования в период сопровождения.

3.1 Модели жизненного цикла разработки ПО

Модель жизненного цикла разработки программного обеспечения описывает виды активностей, выполняемых на каждом этапе процесса разработки программного обеспечения, а также то, как эти активности связаны друг с другом логически и хронологически. Существует несколько различных моделей жизненного цикла разработки программного обеспечения, каждый из которых требует различных подходов к тестированию.

3.1.1 Разработка и тестирование программного обеспечения

Важной частью работы тестировщика является знание распространенных моделей жизненного цикла разработки ПО, на основании которых происходит выбор соответствующих активностей по тестированию.

Для любой модели жизненного цикла разработки существуют несколько показателей качественного тестирования:

- Для каждой активности разработки существует соответствующая активность тестирования.
- У каждого уровня тестирования есть цели, характерные для данного уровня.
- Анализ и проектирование тестов для определенного уровня тестирования начинаются на стадии соответствующих активностей разработки.
- Тестировщики должны участвовать в обсуждениях для определения и уточнения требований и дизайна, а также вовлекаться в рецензирование рабочих продуктов (например, требований, дизайна, пользовательских историй и т.п.), как только будут доступны первые их черновые версии.

Независимо от того, какая модель жизненного цикла разработки выбрана, активности тестирования следует начинать на ранних стадиях жизненного цикла, придерживаясь принципа раннего тестирования.

Эта программа обучения классифицирует распространенные модели жизненного цикла разработки следующим образом:

- Последовательные модели разработки
- Итерационные и инкрементальные модели разработки

Последовательная модель разработки описывает процесс разработки программного обеспечения в качестве линейного, последовательного потока активностей. Это означает, что любую фазу процесса разработки следует начинать после того, как предыдущая фаза завершена. Теоретически фазы не пересекаются друг с другом, но на практике бывает полезным получать раннюю обратную связь от следующей фазы.

При работе по модели «Водопад», активности разработки (например, анализ требований, дизайн, написание кода, тестирование) выполняются каждая по окончании предыдущей. В данной модели активности тестирования осуществляются только после того, как все активности разработки выполнены.

В отличие от модели «Водопад», V-модель интегрирует тестовый процесс на протяжении процесса разработки, реализуя принцип раннего тестирования. Более того, V-модель включает уровни тестирования, относящиеся к соответствующей фазе разработки, что также поддерживает раннее тестирование (смотрите раздел 2.2 для рассмотрения уровней тестирования). В этой модели выполнение тестов, связанных с каждым уровнем тестирования, продолжается последовательно, но в некоторых случаях происходят пересечения.

Результатом работы с применением последовательных моделей разработки является ПО, которое содержит полный набор функциональности, но для его поставки заказчику и пользователям зачастую уходят месяцы и даже годы разработки.

Инкрементальная разработка подразумевает определение требований, дизайн, сборку и тестирование системы по частям. Это приводит к тому, что функциональность ПО растет постепенно. Размер таких функциональных приращений разнится в большую или меньшую сторону в зависимости от выбранных методов работы. Приращение функциональности может включать всего одно изменение в пользовательском интерфейсе или новую опцию запроса.

Итеративная разработка используется в случае, когда разрабатываемый функционал должен быть определен, спроектирован, построен и протестирован в течение нескольких этапов, зачастую с фиксированной продолжительностью каждого из них. Итерации могут включать в себя как набор изменений функционала, реализованного в предыдущих итерациях, так и изменения всего разрабатываемого продукта в целом. На выходе каждой итерации получается рабочий программный продукт, который, по сути, является растущим набором общего функционала, и продолжается это до тех пор, пока не будет выпущена финальная версия этого продукта или разработка не будет остановлена.

В качестве примера можно привести следующие методологии:

- RUP (Rational Unified Process): методология: каждая итерация может быть относительно долгой (например, от двух до трех месяцев), а приращения функциональности соответственно большими, к примеру, две или три группы связанных функциональностей.
- Скрам: каждая итерация может быть относительно короткой (например, часы, дни или несколько недель), а приращения функциональности соответственно небольшими, как, например, несколько улучшений и/или две или три новые функции.
- Канбан: осуществляется с использованием или без использования итераций фиксированной продолжительности, по завершению которых выпускается либо единственная доработка или функциональность, либо группа функциональностей, объединенных вместе.
- Спиральная модель разработки (прототипирование): подразумевает создание экспериментальных приращений, некоторые из которых могут быть значительно переработаны или даже отвергнуты в последующей работе по разработке.

Компоненты или системы, разрабатываемые с использованием этих моделей, часто подразумевают пересечение и повторение уровней тестирования на протяжении процесса разработки. В идеале, каждая функциональность тестируется на нескольких уровнях тестирования по мере приближения к выпуску продукта. В некоторых случаях, команды используют непрерывную поставку или непрерывное развертывание, которые подразумевают существенную автоматизацию многих уровней тестирования в процессе поставки. Предпринимаемые усилия по разработке ПО с использованием этих методов, также включают концепцию самоорганизующихся команд, что может изменить как организацию работ по тестированию, так и взаимоотношения между тестировщиками и разработчиками.

Применение этих методов в итоге формирует готовый продукт, который может быть выпущен конечным пользователям на основе постепенного приращения функциональности или более традиционным способом изготовления основного релиза. Независимо от того, поставлялись ли приращения конечным пользователям, важность регрессионного тестирования растет с ростом разрабатываемой системы.

В отличие от результатов работы с применением последовательных моделей разработки, результатом работы с применением итеративных и инкрементальных моделей является ПО,

готовое к использованию через недели или даже дни, но до поставки полного набора требований продукта могут уйти многие месяцы и даже годы.

Для получения дополнительной информации о тестировании в контексте гибких методологий разработки предлагаем изучить ISTQB Базовый уровень. Тестировщик в сфере Гибких методологий, [Black 2017], [Crispin 2008] и [Gregory 2015].

3.1.2 Выбор модели жизненного цикла разработки в зависимости от ситуации

Модели жизненного цикла разработки программного обеспечения должны выбираться и быть адаптированы к контексту характеристик проекта и продукта. Соответствующая модель жизненного цикла разработки должна быть выбрана и адаптирована на основе цели проекта, типа разрабатываемого продукта, бизнес-приоритетов (например, срок вывода продукта на рынок), и выявленных рисков продукта и проекта. Например, небольшую внутреннюю административную систему следует разрабатывать и тестировать иначе, нежели критически важную для безопасности систему, например, систему управления тормозами автомобиля. Как другой пример, в некоторых случаях организационные и культурные аспекты могут осложнять общение между членами команды, что может препятствовать итеративной разработке.

В зависимости от контекста проекта может потребоваться объединение или реорганизация уровней тестирования и/или тестовых активностей. Например, для интеграции готового коммерческого решения в более крупную систему покупатель этого решения может выполнить тестирование возможности взаимодействия на уровне системного интеграционного тестирования (например, интеграцию с инфраструктурой и другими системами) и на уровне приемочного тестирования (функциональное и/или нефункциональное, наряду с пользовательским приемочным и эксплуатационным приемочным тестированием). Смотрите раздел 2.2 для рассмотрения уровней тестирования и раздел 2.3 для рассмотрения типов тестирования.

Кроме этого, сами модели жизненного цикла разработки могут сочетаться одна с другой. Например, V-модель может быть использована для разработки и тестирования систем серверного уровня и их интеграции, тогда как методология гибкой разработки может быть использована для разработки и тестирования пользовательского интерфейса (ПИ) и функциональности. На ранних стадиях проекта может быть использовано прототипирование, с переходом на инкрементальную модель разработки после завершения экспериментальной фазы.

Системы с общим названием «Интернет вещей», которые состоят из множества различных объектов, таких как устройства, продукты и сервисы, часто разрабатываются с применением разных моделей жизненного цикла разработки для каждого объекта. Это ведет к определенной сложности для разработки версий таких систем. Кроме того, сильный акцент делается уже на поздние фазы жизненного цикла, после внедрения (например, фазы использования, обновления и вывода из эксплуатации).

3.2 Уровни тестирования

Уровни тестирования – это группы активностей тестирования, которые организуются и управляются как единое целое. Каждый уровень тестирования — это реализация процесса тестирования, состоящего из мероприятий, описанных в разделе 1.4, и исполняемого в отношении ПО, находящегося на конкретном уровне разработки, начиная с отдельных модулей и компонентов и заканчивая целыми системами или, где применимо, группами систем. Уровни тестирования связаны с другими активностями в рамках жизненного цикла разработки. Уровни, используемые в данной программе обучения, следующие:

- Компонентное тестирование

- Интеграционное тестирование
- Системное тестирование
- Приемочное тестирование

Уровни тестирования характеризуются следующими признаками:

- Конкретные цели
- Базис тестирования, на который ссылаются для получения тестовых сценариев
- Объект тестирования (то есть то, что будет протестировано)
- Типичные дефекты и отказы
- Специфические подходы и зоны ответственности

Для каждого уровня тестирования требуется подходящая среда тестирования. Например, при приемочном тестировании идеально использовать окружение, максимально приближенное к реальному. В тоже время при компонентном тестировании разработчики обычно используют собственную среду разработки.

3.2.1 Компонентное тестирование

Цели компонентного тестирования

Компонентное тестирование (также известное как модульное тестирование) фокусируется на компонентах, которые могут быть проверены отдельно. Цели компонентного тестирования включают:

- Снижение риска
- Проверку, соответствует ли функциональное и нефункциональное поведение компонентов установленным проектным требованиям
- Укрепление уверенности в качестве компонента
- Обнаружение дефектов в компоненте
- Предотвращение пропуска дефектов на более высокие уровни тестирования

В некоторых случаях, особенно в инкрементных и итеративных моделях разработки (например, гибкой методологии разработки), где изменения кода происходят непрерывно, автоматизированные регрессионные компонентные тесты играют ключевую роль в создании уверенности в том, что изменения не повредили существующий функционал.

Компонентное тестирование часто выполняется изолированно от остальной системы, в зависимости от модели жизненного цикла разработки программного обеспечения и системы, для которой могут потребоваться макеты разрабатываемых объектов, виртуализация служб, стенды, заглушки и драйверы. Компонентное тестирование может охватывать как функциональные (например, правильность вычислений), так и нефункциональные характеристики (например, поиск утечек памяти) и структурные свойства (например, тестирование решений).

Базис тестирования

Примеры рабочих продуктов, которые могут использоваться в качестве базиса тестирования для компонентного тестирования, включают:

- Детальный дизайн
- Код

- Модель данных
- Спецификации компонента

Объекты тестирования

Типичными объектами для компонентного тестирования являются:

- Компоненты, модули
- Код и структуры данных
- Классы
- Модули БД

Типичные дефекты и отказы

Примеры типичных дефектов и отказов при компонентном тестировании включают:

- Неправильная работа функциональности (например, не так, как описано в спецификации)
- Проблемы с потоками данных
- Неправильные код и логика

Дефекты обычно исправляются, как только они обнаруживаются, часто без оформления, в соответствующей системе управления дефектами. Однако, когда разработчики оформляют отчеты о дефектах, создается важная информация для анализа первопричин дефектов и улучшения процесса.

Специфические подходы и зоны ответственности

Компонентное тестирование обычно выполняется разработчиком, который написал код, но это как минимум требует доступа к тестируемому коду. Разработчики могут чередовать разработку компонентов с обнаружением и устранением дефектов. Разработчики часто пишут и выполняют тесты после написания кода для компонента. Тем не менее, написание автоматизированных тестовых сценариев для компонентов может предшествовать написанию кода приложения, особенно в методологии гибкой разработки.

Например, рассмотрим разработку на основе тестов. Разработка на основе тестов является высоко итеративной, и основана на циклах разработки автоматизированных тестов, затем построении и интеграции небольших фрагментов кода, а уже после – выполнении компонентного тестирования, исправлении проблем и рефакторинга кода. Этот процесс продолжается до тех пор, пока компонент не будет полностью собран и все компонентные тесты не пройдут успешно. Разработка на основе тестов является примером подхода «сначала тестирование». Хотя разработка на основе тестов берет начало в методологии экстремального программирования, она распространилась на другие формы гибких методологий разработки, а также на последовательные жизненные циклы (см. программу ISTQB Базовый уровень. Тестировщик в сфере Гибких методологий).

3.2.2 Интеграционное тестирование

Цели интеграционного тестирования

Интеграционное тестирование фокусируется на взаимодействии между компонентами или системами. Цели интеграционного тестирования включают:

- Снижение риска
- Проверка, соответствует ли функциональное и нефункциональное поведение интерфейсов установленным проектным требованиям

- Повышение уверенности в качестве интерфейсов
- Обнаружение дефектов (которые могут быть в самих интерфейсах или внутри компонентов или систем)
- Предотвращение пропуска дефектов на более высокие уровни тестирования

Как и при компонентном тестировании, в некоторых случаях автоматизированные регрессионные интеграционные тесты поддерживают уверенность в том, что изменения не повредили существующие интерфейсы, компоненты или системы.

В этой учебной программе описаны два разных уровня интеграционного тестирования, которые могут выполняться на тестовых объектах различного размера следующим образом:

- Компонентное интеграционное тестирование фокусируется на взаимодействиях и интерфейсах между интегрированными компонентами. Оно выполняется после компонентного и, как правило, автоматизируется. При итеративной и инкрементальной разработке компонентное интеграционное тестирование обычно является частью процесса непрерывной интеграции
- Системное интеграционное тестирование фокусируется на взаимодействиях и интерфейсах между системами, пакетами и микросервисами. Системное интеграционное тестирование также может охватывать взаимодействия и интерфейсы, предоставляемые сторонними организациями (например, веб-сервисы). В этом случае организация-разработчик не контролирует внешние интерфейсы, которые могут создавать различные сложности для тестирования (например, проверяя, что блокирующие тесты дефекты устранены в коде сторонней организации, подготавливая тестовые среды и т. д.). Системное интеграционное тестирование может быть выполнено после системного тестирования или параллельно с выполняемыми активностями по системному тестированию (как в последовательной разработке, так и в итеративной и инкрементальной разработке).

Базис тестирования

Примеры рабочих продуктов, которые могут использоваться в качестве базиса тестирования для интеграционного тестирования, включают:

- Дизайн продукта и системы
- Диаграммы последовательности
- Спецификации интерфейса и протокола связи
- Сценарии использования системы
- Архитектура на уровне компонентов или системы
- Рабочие процессы
- Спецификации, описывающие внешние интерфейсы

Объекты тестирования

Типичными объектами тестирования при интеграционном тестировании являются:

- Подсистемы
- Базы данных
- Инфраструктура
- Интерфейсы

- Программные интерфейсы приложения (API)
- Микросервисы

Типичные дефекты и отказы

Примеры типичных дефектов и отказов при компонентном интеграционном тестировании включают:

- Некорректные данные, отсутствующие данные или неправильная кодировка данных
- Неверная последовательность или временные характеристики обращения к интерфейсам
- Несовместимость интерфейсов
- Сбои связи между компонентами
- Необработанные или неправильно обработанные сбои связи между компонентами
- Неправильные предположения о назначении, единицах или границах данных, передаваемых между компонентами

Примерами типичных дефектов и отказов для системного интеграционного тестирования являются:

- Несогласованные структуры сообщений между системами
- Некорректные данные, отсутствующие данные, или неправильная кодировка данных
- Несовместимость интерфейсов
- Сбои связи между системами
- Необработанные или неправильно обработанные сбои связи между системами
- Неправильные предположения о значении, единицах или границах данных, передаваемых между системами
- Несоблюдение обязательных правил безопасности

Специфические подходы и зоны ответственности

Компонентные интеграционные и системные интеграционные тесты должны быть сосредоточены на интеграции как таковой. Например, если интегрировать модуль A с модулем B, тесты должны быть сосредоточены на связи между модулями, а не на функциональности отдельных модулей, поскольку они должны быть проверены во время компонентного тестирования. Если интегрировать систему X с системой Y, тесты должны быть сосредоточены на связи между системами, а не на функциях отдельных систем, поскольку они должны быть проверены во время системного тестирования. Для этого могут применяться функциональные, нефункциональные и структурные тесты.

Компонентное интеграционное тестирование часто является обязанностью разработчиков. А системное интеграционного тестирование, как правило, обязанность тестировщиков. В идеале тестировщики, проводящие системное интеграционного тестирование, должны понимать архитектуру системы и влиять на интеграционное планирование.

Если интеграционные тесты и стратегия интеграции планируются до создания компонентов или систем, эти компоненты или системы могут быть построены в порядке, необходимом для наиболее эффективного тестирования. Стратегии систематической интеграции могут основываться на архитектуре системы (например, сверху вниз и снизу вверх), функциональных задачах, последовательностях обработки транзакций или другом аспекте работы системы или

компонентов. Чтобы упростить локализацию дефектов и обнаруживать их на раннем этапе, интеграция должна обычно быть последовательной (т. е. небольшое количество дополнительных компонентов или систем за раз), а не методом «большого взрыва» (т. е. интеграция всех компонентов или систем сразу). Анализ рисков наиболее сложных интерфейсов может помочь определить фокус интеграционного тестирования.

Чем больше объем интеграции, тем труднее становится выявлять дефекты конкретного компонента или системы, что может привести к увеличению риска и увеличению времени поиска неисправностей. Это одна из причин того, что непрерывная интеграция, когда программное обеспечение интегрируется методом «компонент за компонентом» (т.е. функциональная интеграция), стала обычной практикой. Такое непрерывное интегрирование часто подразумевает автоматическое регрессионное тестирование, в идеале на нескольких уровнях тестирования.

3.2.3 Системное тестирование

Цели системного тестирования

Системное тестирование фокусируется на поведении и возможностях целой системы или продукта, часто учитывая сквозные задачи, которые может выполнять система, и нефункциональное поведение, которое она демонстрирует при выполнении этих задач.

Цели системного тестирования включают:

- Снижение риска
- Проверка, соответствует ли функциональное и нефункциональное поведение системы установленным проектным требованиям, дизайну и спецификациям
- Проверка, что система реализована полностью и будет работать, как ожидалось
- Повышение уверенности в качестве системы в целом
- Обнаружение дефектов
- Предотвращение попадания дефектов на более высокие уровни тестирования или в среду эксплуатации

Для определенных систем целью может быть проверка качества данных. Как и в случае компонентного и интеграционного тестирования, в некоторых случаях автоматическое регрессионное тестирование системы демонстрирует, что изменения не повредили существующий функционал или возможности конечных пользователей. Системное тестирование часто дает информацию, которая используется заинтересованными сторонами для принятия решения о релизе. Системное тестирование также может проверять выполнение законодательных или нормативных требований или стандартов.

Тестовая среда для системного тестирования должна идеально соответствовать конечной целевой или эксплуатационной среде.

Базис тестирования

Примеры рабочих продуктов, которые могут использоваться в качестве базиса тестирования для системного тестирования, включают:

- Системные требования и требования к продукту (функциональные и нефункциональные)
- Отчеты об анализе рисков
- Сценарии использования
- Бизнес-потребности и пользовательские истории

- Модели поведения системы
- Диаграммы состояний
- Системные и пользовательские руководства

Объекты тестирования

Типичные объекты системного тестирования включают:

- Приложения
- Аппаратные / программные системы
- Тестируемая система
- Операционные системы
- Конфигурация системы и конфигурация данных

Типичные дефекты и отказы

Примеры типичных дефектов и сбоев при системном тестировании включают:

- Некорректные вычисления
- Некорректное или неожиданное функциональное или нефункциональное поведение системы
- Некорректное управление и/или передача данных внутри системы
- Невозможность правильно и полностью выполнить функциональные задачи конечными пользователями
- Неспособность системы работать правильно в среде эксплуатации
- Неспособность системы работать так, как описано в системных и пользовательских руководствах

Специфические подходы и зоны ответственности

Системное тестирование должно быть сосредоточено на общем (как функциональном, так и нефункциональном) поведении системы с точки зрения конечных пользователей. При системном тестировании следует использовать наиболее подходящие методы (см. главу 4) для конкретного аспекта тестируемой системы. Например, может быть применена таблица альтернатив, чтобы проверить, соответствует ли функциональное поведение бизнес-правилам.

Системное тестирование часто проводит независимая группа тестировщиков. Ошибки в спецификациях (например, отсутствие пользовательских историй, неверно определенные бизнес-требования и т. д.) могут привести к отсутствию понимания или разногласиям насчет ожидаемого поведения системы. Такие ситуации могут вызывать ложно позитивные или ложно негативные результаты тестирования, которые отнимают время и снижают эффективность обнаружения дефектов. Раннее вовлечение тестировщиков в разработку пользовательских историй или в активности статического тестирования, таких как рецензирование, помогает снизить частоту возникновения таких ситуаций.

3.2.4 Приемочное тестирование

Цели приемочного тестирования

Приемочное тестирование, как и системное тестирование, обычно фокусируется на поведении и возможностях системы или продукта в целом. Цели приемочного тестирования включают:

- Продемонстрировать уверенность в качестве системы в целом
- Проверить, что система завершена и будет работать как ожидалось
- Проверить, соответствует ли функциональное и нефункциональное поведение системы установленным проектным требованиям

Приемочное тестирование может дать информацию для оценки готовности системы к развертыванию и использованию конечным пользователем. Во время приемочного тестирования могут быть обнаружены дефекты, но их поиск зачастую не является целью, и обнаружение значительного количества дефектов во время приемочных испытаний может в некоторых случаях рассматриваться как основной риск проекта. Приемочное тестирование также может служить демонстрацией удовлетворения системы законодательным и нормативным требованиям или стандартам.

Типичными формами приемочного тестирования являются:

- Пользовательское приемочное тестирование
- Эксплуатационное приемочное тестирование
- Контрактное и нормативное приемочное тестирование
- Альфа-тестирование и бета-тестирование

Каждая из форм описана в следующих четырех подразделах.

Пользовательское приемочное тестирование

Приемочное тестирование системы пользователями обычно сосредоточено на проверке пригодности использования системы предполагаемыми пользователями в реальной или моделируемой рабочей среде. Основная цель заключается в получении уверенности в том, что пользователи могут использовать систему для удовлетворения своих потребностей, а также в соответствии системы требованиям и ее способности выполнять поставленные бизнесом задачи с минимальными трудностями, затратами и рисками.

Эксплуатационное приемочное тестирование

Приемочное тестирование системы сотрудниками или администраторами систем обычно проводятся в (имитируемой) среде эксплуатации. В тестах основное внимание уделяется эксплуатационным аспектам, которые могут включать:

- Тестирование резервного копирования и восстановления
- Установка, удаление и обновление
- Восстановление после полного отказа (краха) системы
- Управление пользователями
- Задачи сопровождения (обслуживания)
- Задачи загрузки и миграции данных
- Проверки уязвимостей
- Тестирование производительности

Основная цель эксплуатационного приемочного тестирования – получение уверенности в том, что операторы или системные администраторы смогут поддерживать работоспособность системы для пользователей в среде эксплуатации, даже в исключительных или сложных условиях

Контрактное и нормативное приемочное тестирование

Контрактное приемочное тестирование проводится в соответствии с указанными в контракте критериями приемки специализированного программного обеспечения. Критерии приемки должны определяться, когда стороны заключают контракт. Контрактное приемочное тестирование часто выполняется пользователями или независимой группой тестировщиков.

Нормативное приемочное тестирование проводится в соответствии с любыми нормативами, которые должны соблюдаться, например, в отношении правительственных или юридических норм, а также норм безопасности. Нормативное приемочное тестирование часто выполняется пользователями или независимой группой тестировщиков, иногда с результатами, которые засвидетельствованы или проверяются регулирующими органами.

Основная цель проведения контрактного и нормативного приемочного тестирования заключается в укреплении уверенности в том, что достигнуто соответствие контрактным или нормативным требованиям.

Альфа-тестирование и бета-тестирование

Альфа-тестирование и бета-тестирование обычно используются разработчиками готовых коммерческих решений, которые хотят получить обратную связь от потенциальных или существующих пользователей, клиентов и/или операторов до того, как программный продукт будет выставлен в коммерческую продажу. Альфа-тестирование проводится на мощностях компании разработчика, но не командой самого разработчика, а потенциальными или существующими клиентами и/или операторами или независимой группой тестирования. Бета-тестирование проводится потенциальными или существующими клиентами и/или операторами на их собственных мощностях Бета-тестирование может проходить после альфа-тестирования или даже без предшествующего альфа-тестирования.

Одной из целей альфа- и бета-тестирования является получение уверенности потенциальных или существующих клиентов и/или операторов в том, что они смогут использовать систему в нормальных, повседневных условиях и в эксплуатационных средах для достижения своих целей с минимальными трудностями, затратами и рисками. Другой целью может быть обнаружение дефектов, связанных с условиями и средой эксплуатации, в которых система будет использоваться, особенно когда команде разработчиков трудно воспроизвести эти условия и среды.

Базис тестирования

Примеры рабочих продуктов, которые могут использоваться в качестве базиса тестирования для любой формы приемочного тестирования, включают:

- Бизнес-процессы
- Пользовательские и бизнес-требования
- Нормативы, юридические контракты и стандарты
- Сценарии использования системы
- Системные требования
- Системная или пользовательская документация
- Процедуры установки программного обеспечения
- Отчеты анализа риска

Кроме того, в качестве тестового базиса разработки тестовых сценариев для эксплуатационного приемочного тестирования могут использоваться один или несколько следующих рабочих продуктов:

- Процедуры резервного копирования и восстановления
- Процедуры восстановления после полного отказа системы
- Нефункциональные требования
- Эксплуатационная документация
- Инструкции по развертыванию и установке
- Целевые показатели производительности
- Пакеты баз данных
- Стандарты или нормативы безопасности

Типичные объекты тестирования

Типичными объектами тестирования для любой формы приемочного тестирования являются:

- Тестируемая система
- Конфигурация системы и конфигурационные данные
- Бизнес-процессы для полностью интегрированной системы
- Восстановление системы и «горячего узла» (для непрерывности бизнес-процессов и аварийного восстановления)
- Операционные и эксплуатационные процессы
- Формы
- Отчеты
- Существующие и преобразованные производственные данные

Типичные дефекты и отказы

Примеры типичных дефектов для любой формы приемочного тестирования включают:

- Системные рабочие процессы не отвечают бизнес-требованиям или требованиям пользователей
- Бизнес-требования некорректно реализованы
- Система не соответствует контрактным и/или нормативным требованиям
- Нефункциональные сбои, такие как уязвимости в системе безопасности, недостаточная производительность при высоких нагрузках или неправильная работа на поддерживаемой платформе

Специфические подходы и зоны ответственности

Приемочное тестирование часто является обязанностью клиентов, бизнес-пользователей, владельцев продуктов или операторов системы, а также других заинтересованных сторон.

В последовательном жизненном цикле разработки приемочное тестирование часто считается последним уровнем тестирования, но оно может проводиться и в другое время, например:

- Приемочное тестирование коммерческого готового программного обеспечения может проводиться, когда ПО установлено или интегрировано

- Приемочное тестирование нового функционального улучшения может производиться перед системным тестированием

При итеративной разработке проектные группы могут использовать различные формы приемочного тестирования как во время итерации, так и в ее конце, например, те, которые направлены на проверку новой функциональности по ее критериям приемки, и те, которые направлены на подтверждение того, что новая функциональность удовлетворяет потребностям пользователей. Кроме того, альфа-тесты и бета-тесты могут выполняться либо в конце каждой итерации, либо после завершения каждой итерации, либо после серии итераций. Пользовательское приемочное тестирование, эксплуатационное приемочное тестирование, а также контрактное и нормативное приемочное тестирование могут проводиться аналогично либо по завершении каждой итерации, либо после завершения каждой итерации, либо после серии итераций.

3.3 Типы тестирования

Тип тестирования – это совокупность активностей тестирования, направленных на тестирование заданных характеристик системы или ее части, основываясь на конкретных целях. К таким целям можно отнести следующие:

- Оценку функциональных характеристик качества системы, таких как полнота, корректность, целесообразность
- Оценку нефункциональных характеристик качества, таких как надежность, продуктивность работы, безопасность, совместимость и удобство использования
- Оценку правильности, полноты структуры или архитектуры компонента или системы, их соответствие спецификации
- Оценку влияния изменений, например, подтверждение того, что дефекты были исправлены (подтверждающее тестирование) и поиск непреднамеренных изменений в поведении, вызванных изменениями в программном обеспечении или окружении (регрессионное тестирование)

3.3.1 Функциональное тестирование

Функциональное тестирование системы включает тесты по оценке функций, которые должна выполнять система. Функциональные требования могут быть описаны в рабочих продуктах (требования, спецификация, бизнес-потребность, пользовательская история, сценарий использования) или в функциональной спецификации, а могут быть вообще не задокументированы. Функции системы дают ответ на вопрос «что делает система».

Функциональные тесты должны выполняться на всех уровнях тестирования (например, тесты для компонентов системы могут основываться на спецификации компонента), поэтому фокус тестирования различается для каждого уровня (см. раздел 2.2).

Функциональное тестирование рассматривает поведение системы, поэтому для получения тестовых условий и тестовых сценариев могут быть использованы техники тестирования методом черного ящика (см. раздел 4.2).

Оценить полноту функционального тестирования можно с использованием покрытия функциональности тестами. Покрытие функциональности – это мера, с которой конкретный тип элемента функциональности был охвачен тестами. Уровень покрытия вычисляется в процентах от общего количества элементов (или типов элементов). Например, используя трассируемость тестов и соответствующих требований, можно вычислить процент требований, которые покрыты тестами, и выявить пробелы в покрытии.

Проектирование и выполнение функциональных тестов может потребовать специальных навыков и знаний, относящихся к конкретной области бизнес задачи, решаемой программным обеспечением (например, знания в области программного обеспечения геологического моделирования в нефте- и газодобывающей отрасли) или специфической предметной области (например, программное обеспечение компьютерных игр, которые предоставляют интерактивный развлекательный контент).

3.3.2 Нефункциональное тестирование

Нефункциональное тестирование системы выполняется для оценки таких характеристик системы и программного обеспечения, как удобство использования, производительность или безопасность. За классификацией характеристик качества программного обеспечения следует обратиться к стандарту ИСО (ISO/IEC 25010). Нефункциональное тестирование – это проверка того, «насколько хорошо работает система».

Вопреки всеобщему заблуждению, нефункциональное тестирование может и, чаще всего, должно выполняться на всех уровнях тестирования, и как можно раньше. Несвоевременное обнаружение нефункциональных дефектов может быть угрозой успеха всего проекта.

Для получения тестовых условий и тестовых сценариев в нефункциональном тестировании могут использоваться техники тестирования методом черного ящика (см. раздел 4.2). Например, анализ граничных значений может использоваться для определения пиковых нагрузок при тестировании производительности.

Полноту нефункционального тестирования можно оценить через нефункциональное покрытие. Нефункциональное покрытие – это степень, с которой какая-либо нефункциональная характеристика покрыта тестами, которая выражается в процентном соотношении покрытых тестами характеристик к их общему числу. Например, используя трассируемость тестов к соответствующим им поддерживаемым устройствам для мобильного приложения, можно вычислить процент поддерживаемых устройств, используемых при тестировании совместимости, тем самым определив потенциальные пробелы в покрытии.

Проектирование и выполнение нефункциональных тестов может потребовать специальных навыков и знаний, например, знаний о слабых сторонах, свойственных той или иной структуре проекта или технологии (это могут быть уязвимости безопасности, связанные с конкретным языком программирования), знание специфической базы пользователей(например, представления о пользователях управленческих систем в медицинских учреждениях).

С более подробной информацией о тестировании нефункциональных характеристик качества можно ознакомиться в программах обучения ISTQB-ATA Продвинутый уровень. Тест-аналитик, ISTQB-ATTA Продвинутый уровень. Технический тест-аналитик, ISTQB-SEC Продвинутый уровень. Тестировщик безопасности, и других специализированных программах ISTQB.

3.3.3 Тестирование методом белого ящика

Тестирование методом белого ящика основывается на внутренней структуре системы или ее реализации. Под внутренней структурой подразумевается программный код, архитектура, принципы работы и/или потоки данных внутри системы (см. раздел 4.3).

Полноту тестирования методом белого ящика оценивают с помощью структурного покрытия. Структурное покрытие – это мера, с которой какой-либо тип структурного элемента был покрыт тестами. Структурное покрытие измеряется в процентах от общего количества элементов, охваченного тестами.

На уровне компонентного тестирования покрытие кода измеряется в процентах покрытия тестами кода компонента и может быть вычислено с точки зрения различных аспектов кода

(элементов покрытия), таких как процент исполняемых операторов (покрытие операторов), тестируемых в компоненте, или процент покрытия разветвлений (покрытие альтернатив). Эти типы покрытия в совокупности называются покрытием кода. На уровне интеграционного тестирования компонентов тестирование методом белого ящика может основываться на архитектуре системы, например, на взаимодействии между компонентами. В этом случае структурное покрытие измеряется в процентах от количества интерфейсов, задействованных в тестах.

Проектирование и выполнение тестирования методом белого ящика может потребовать специальных навыков и знаний, таких как знание о методах сборки приложения (например, при использовании инструментов покрытия кода), о том, как хранятся данные (для оценки оптимальности запросов к базам данных) и умений пользоваться и интерпретировать результаты работы инструментов покрытия кода.

3.3.4 Тестирование, связанное с изменениями

Когда в систему вносятся изменения, выполненные для исправления дефекта, либо из-за новой или изменяющейся функциональности, необходимо провести тестирование, чтобы подтвердить, что изменения исправили дефект или что функциональность правильно реализована и изменения не вызвали каких-либо непредвиденных неблагоприятных последствий.

- Подтверждающее тестирование: после того как дефект исправлен, программное обеспечение может быть протестировано с использованием всех тех же тестовых сценариев, которые завершились с ошибкой из-за найденного дефекта. Эти тестовые сценарии должны быть повторно выполнены на новой версии программного обеспечения. Как минимум, на новой версии программного обеспечения должны быть повторно выполнены шаги по воспроизведению сбоев, вызванных дефектом. Целью подтверждающего тестирования является удостоверение в том, что найденный дефект был исправлен.
- Регрессионное тестирование: может случиться так, что изменение, внесенное в одну часть кода, будь то исправление или что-либо другое, может случайно повлиять на поведение других частей кода, будь то внутри одного и того же компонента, в других компонентах одной и той же системы или даже в других системах. Доработки могут включать изменения в среде, такие как новая версия операционной системы или системы управления базами данных. Такие непреднамеренные побочные эффекты называются регрессиями. Регрессионное тестирование включает выполнение тестов для обнаружения таких непреднамеренных побочных эффектов.

Подтверждающее тестирование и регрессионное тестирование проводятся на всех уровнях тестирования.

Особенно в итеративных и инкрементальных жизненных циклах разработки (например, гибкая методология разработки), новые функции, изменения существующих функций и рефакторинг кода приводят к частым изменениям кода, что также требует тестирования, связанного с изменениями. Из-за меняющегося характера системы подтверждающее тестирование и регрессионное тестирование очень важны. Это особенно актуально для «Интернета вещей», где отдельные объекты (например, устройства) часто обновляются или заменяются. Регрессионные тесты выполняются много раз и обычно проходят медленно, поэтому регрессионное тестирование - это серьезный кандидат на автоматизацию.

3.3.5 Типы и уровни тестирования

Любой из описанных выше типов тестирования можно выполнять на любом уровне тестирования. Для иллюстрации ниже будут даны примеры функциональных, нефункциональных, структурных

тестов и тестов, связанных с изменениями для всех уровней тестирования банковского приложения, начиная с функциональных тестов:

- Компонентное тестирование: тесты разрабатываются на основе того, как компонент должен вычислять сложные проценты
- Тестирование интеграции компонентов: тесты разрабатываются на основе того, как информация учетной записи, созданная в пользовательском интерфейсе, передается модулю бизнес-логики
- Системное тестирование: тесты разрабатываются на основе того, как владельцы счетов могут подать заявку на кредитную линию на своих расчетных счетах
- Системное интеграционное тестирование: тесты разрабатываются на основе того, как система использует внешний микросервис для проверки кредитного рейтинга владельца счета
- Приемочное тестирование: тесты разрабатываются на основе того, как банкир обрабатывает одобрение или отклонение заявки на получение кредита

Примерами нефункциональных тестов являются:

- Компонентное тестирование: разрабатываются тесты производительности, предназначенные для оценки количества циклов ЦП, необходимых для выполнения расчета сложных процентов
- Тестирование интеграции компонентов: разрабатываются тесты на защищенность от уязвимостей при переполнении буфера данными, переданными из пользовательского интерфейса в модуль бизнес-логики
- Системное тестирование: разрабатываются тесты на переносимость, предназначенные для проверки того, работает ли слой представления во всех поддерживаемых браузерах и мобильных устройствах
- Системное интеграционное тестирование: разрабатываются тесты надежности, предназначенные для оценки работоспособности системы, если микросервис кредитной оценки не отвечает
- Приемочное тестирование: разрабатываются тесты практичности, предназначенные для оценки доступности интерфейса обработки банковских процессов для людей с ограниченными возможностями

Примерами тестов методом белого ящика являются:

- Компонентное тестирование: разрабатываются тесты, предназначенные для обеспечения полного покрытия операторов и альтернатив (см. раздел 4.3) для всех компонентов, которые выполняют финансовые расчеты
- Тестирование интеграции компонентов: разрабатываются тесты, предназначенные для того, чтобы проверить, как каждый экран в интерфейсе браузера передает данные на следующий экран и в модуль бизнес-логики
- Системное тестирование: разрабатываются тесты, предназначенные для покрытия последовательности веб-страниц, которые могут возникать при работе с кредитной линией
- Системное интеграционное тестирование: разрабатываются тесты, предназначенные для исполнения всех возможных типов запросов, отправляемых на микросервис кредитной оценки

- Приемочное тестирование: разрабатываются тесты, предназначенные для охвата всех поддерживаемых файловых структур финансовых данных и диапазонов значений для банковских переводов.

Наконец, примерами тестов, связанных с изменением являются

- Компонентное тестирование: для каждого компонента разрабатываются автоматические регрессионные тесты и включаются в среду непрерывной интеграции
- Тестирование интеграции компонентов: разрабатываются тесты, для подтверждения исправлений дефектов, связанных со связями, по мере того как исправления проверяются в репозитории кода
- Системное тестирование: повторно выполняются все тесты для конкретного рабочего процесса, если любой из аспектов этого процесса меняется
- Системное интеграционное тестирование: ежедневно повторяются тесты для приложения, взаимодействующего с микросервисом кредитной оценки, в рамках непрерывного развертывания этого микросервиса
- Приемочное тестирование: все ранее неудавшиеся тесты повторно выполняются после устранения дефекта, обнаруженного при приемочных испытаниях

Хотя в этом разделе приведены примеры каждого типа теста на каждом уровне, не обязательно иметь каждый тип теста, представленный на каждом уровне для всех разрабатываемых систем. Тем не менее, важно выполнять соответствующие типы тестов на каждом уровне, особенно на самом раннем этапе.

3.4 Тестирование в период сопровождения

После развертывания в эксплуатационных средах программное обеспечение и системы необходимо поддерживать. Практически невозможно избежать различных изменений в поставляемом программном обеспечении и системах, которые либо должны исправлять дефекты, обнаруженные при промышленной эксплуатации, либо добавлять новые функции, либо удалять или изменять уже существующий функционал. Сопровождение также необходимо для сохранения или улучшения нефункциональных характеристик качества компонента или системы в течение всего срока службы, особенно эффективности производительности, совместимости, надежности, безопасности и переносимости.

В процессе сопровождения системы после внесения в нее изменений должно выполняться и тестирование как с целью оценки успеха, ради которого были сделаны изменения, так и для проверки возможных побочных эффектов (например, регрессий) в частях системы, которые остаются неизменными (что обычно является большей частью системы). Тестирование в период сопровождения фокусируется на тестировании изменений в системе, а также на тестировании неизмененных частей, на которые могли повлиять изменения. Сопровождение может включать запланированные релизы и незапланированные релизы (срочные исправления).

Релиз на этапе сопровождения системы может потребовать проведения тестирования в период сопровождения на нескольких уровнях тестирования с использованием различных типов тестов в зависимости от его объема. Объем тестирования в период сопровождения зависит от:

- Степени риска изменения, например, степени, в которой измененная область программного обеспечения взаимодействует с другими компонентами или системами
- Размера существующей системы
- Величины внесенных изменений

3.4.1 Необходимые условия для тестирования в период сопровождения

Существует несколько причин, почему требуется сопровождение программного обеспечения и, соответственно, тестирование в период сопровождения как для запланированных, так и для незапланированных изменений.

Мы можем классифицировать необходимые условия для тестирования в период сопровождения следующим образом:

- Модификации, такие как запланированные улучшения (например, базирующиеся на графике выпуска обновлений), корректирующие и аварийные изменения, изменения среды эксплуатации (например, запланированные обновления операционной системы или базы данных), обновления коммерческого готового программного обеспечения и исправления дефектов и уязвимостей
- Миграция, например, с одной платформы на другую, которая может потребовать проведения эксплуатационных тестов новой среды, а также измененного программного обеспечения или тестов преобразования данных, когда данные будут перенесены в поддерживаемую систему из другого приложения
- Снятие с эксплуатации, например, когда заканчивается жизненный цикл приложения

Когда приложение или система снимаются с эксплуатации, то может потребоваться тестирование переноса или архивирования данных, если требуются длительные периоды хранения данных. Также может потребоваться тестирование процедур восстановления после архивирования для случаев длительных периодов хранения. Кроме того, может потребоваться регрессионное тестирование, чтобы гарантировать, что все функциональные возможности, которые остаются в эксплуатации, все еще работают.

Для «Интернета вещей» тестирование в период сопровождения может быть вызвано внедрением в общую систему совершенно новых или модифицированных элементов, таких как аппаратные устройства и программные службы. Тестирование в период сопровождения для таких систем делает особый акцент на интеграционном тестировании на разных уровнях (например, сетевом уровне, уровне приложений) и на аспектах безопасности, в частности тех, которые касаются персональных данных.

3.4.2 Анализ влияния для тестирования в период сопровождения

Анализ влияния оценивает изменения, которые были сделаны для обновленной эксплуатируемой версии для определения предполагаемых последствий, а также ожидаемых и возможных побочных эффектов изменения, и определяет области в системе, на которые может повлиять изменение. Анализ влияния также может помочь определить воздействие изменения на существующие тесты. Побочные эффекты и затронутые области в системе необходимо проверить на регрессии, возможно, после обновления любых существующих тестов, затронутых изменением.

Анализ влияния может быть выполнен до внесения изменений, чтобы решить, следует ли вносить изменения, исходя из возможных последствий в других областях системы.

Анализ влияния может быть затруднен если:

- Спецификации (например, бизнес-требования, истории пользователей, архитектура) устарели или отсутствуют
- Тестовые сценарии не задокументированы или устарели
- Двунаправленная прослеживаемость между тестами и базисом тестирования не поддерживалась

- Инструментарий поддержки устарел или вовсе отсутствует
- Участники процесса не имеют основополагающих знаний о системе
- Во время разработки уделялось недостаточно внимания сопровождаемости программного обеспечения

4. Статические методы тестирования – 135 мин

Ключевые слова

свободное рецензирование, рецензирование на основе чек-листов, динамическое тестирование, формальное рецензирование, неформальное рецензирование, инспекция, прочтение, основанное на точке зрения, рецензирование, ролевое рецензирование, рецензирование, основанное на сценарии, статический анализ, статическое тестирование, технический анализ, разбор

Цели обучения для главы «Статические методы тестирования»

3.1 Основы статического тестирования

FL-3.1.1 (K1) Распознавать типы программных продуктов, которые можно исследовать различными статическими методами

FL-3.1.2 (K2) Использовать примеры для описания значимости использования статических методов

FL-3.1.3 (K2) Объяснять разницу между статическими и динамическими методами, учитывая цели, типы выявляемых дефектов, и роль этих методов в жизненном цикле программного обеспечения

3.2 Процесс анализа

FL-3.2.1 (K2) Подведение итогов анализа рабочего продукта

FL-3.2.2 (K1) Распознавать разные роли и ответственность в формальном анализе

FL-3.2.3 (K2) Объяснять разницу между различными видами анализа: неформальный анализ, пошаговый разбор, технический анализ и инспекция

FL-3.2.4 (K3) Применение техники анализа для обнаружения дефектов в рабочем продукте

FL-3.2.5 (K2) Объяснять факторы, способствующие успешному анализу

4.1 Основы статического тестирования

В отличие от динамического тестирования, которое требует исполнения кода, статическое тестирование опирается на выполняемую человеком экспертизу рабочих продуктов (например, рецензирование) или инструментальную оценку кода или других рабочих продуктов (например, статический анализ). Оба типа статического тестирования оценивают работу тестируемого продукта без фактического исполнения кода или работы тестируемого продукта. Статический анализ, безусловно, необходим для критически важных систем (например, для программного обеспечения в авиации, медицине или атомной энергетике), но статический анализ также стал важен и распространен в других областях. Например, статический анализ является важной частью тестирования безопасности. Проведение статического анализа также часто включается в автоматизированный процесс сборки и поставки систем, например, в разработке по гибкой методологии присутствуют непрерывные поставка и развертывание.

4.1.1 Рабочие продукты, которые могут быть проверены с помощью статических методов

Практически любой продукт может быть исследован с использованием статического тестирования (рецензирования или статического анализа), например:

- Спецификации, включая бизнес-требования, функциональные требования и требования безопасности
- Эпики, пользовательские истории, критерии приемки
- Архитектура и проектные спецификации
- Код
- Тестовая документация, включая тест-планы, тест-кейсы, тестовые процедуры и автоматизированные тестовые сценарии
- Руководства пользователя
- Веб-страницы
- Контракты, проектные планы, графики и бюджеты
- Модели, такие как диаграммы действий, которые могут использоваться для тестирования на основе моделей (см. ISTQB-MBT Базовый уровень. Тестирование на основе моделей, [Kramer 2016]).

Рецензирование может быть применено к любому рабочему продукту, который участники анализа понимают и могут прочесть. Статический анализ может быть эффективно применен к любому рабочему продукту с формальной структурой (обычно код или модели), для которого существует соответствующий инструмент статического анализа. Статический анализ может использовать инструменты, оценивающие рабочие продукты, написанные на естественном языке, такие как требования (например, инструменты проверки орфографии, грамматики, удобочитаемости).

4.1.2 Преимущества статических методов

Статические методы обеспечивают множество преимуществ. При применении на ранней стадии разработки программного обеспечения статическое тестирование позволяет обнаруживать дефекты до проведения динамического тестирования (например, в требованиях или проектных спецификациях, актуализация списка требований к продукту и т.д.). Дефекты, обнаруженные на ранней стадии, гораздо дешевле исправить, чем дефекты, обнаруженные на более поздней

стадии жизненного цикла, особенно в сравнении с дефектами, найденными после поставки и активного использования программного обеспечения. Использование методов статического тестирования для обнаружения дефектов и затем их быстрое исправление почти всегда намного дешевле для компании, чем использование динамического тестирования для обнаружения дефектов в тестовом объекте, а затем их исправление, особенно при рассмотрении дополнительных расходов, связанных с обновлением других продуктов и регрессионным тестированием.

Дополнительные преимущества статического тестирования:

- Обнаружение и исправление дефектов более эффективно, до проведения динамического тестирования
- Идентификация дефектов, которые сложно обнаружить при динамическом тестировании
- Предотвращение дефектов дизайна или кодирования путем выявления несоответствий, неоднозначностей, противоречий, упущений, неточностей и избыточности требований
- Повышение производительности разработки, включая улучшение дизайна и сопровождаемости кода
- Сокращение затрат и времени на разработку
- Сокращение затрат и времени на тестирование
- Снижение общей стоимости качества в течение всего срока службы программного обеспечения из-за меньшего количества сбоев в жизненном цикле после поставки в эксплуатацию
- Улучшение коммуникации между членами команды в процессе участия в рецензировании

4.1.3 Различия между статическими и динамическими методами

Статическое и динамическое тестирование могут иметь одни и те же цели (см. раздел 1.1.1), например, предоставление оценки качества продукта и выявление дефектов как можно раньше. Статическое и динамическое тестирование дополняют друг друга, обнаруживая различные типы дефектов.

Одно из основных отличий заключается в том, что статическое тестирование обнаруживает дефекты в рабочих продуктах напрямую, а не идентифицирует сбои, вызванные дефектами при запуске программного обеспечения. Дефект может находиться в рабочем продукте очень долгое время, не вызывая сбоя. Путь, в котором находится дефект, может выполняться редко или быть труднодоступен, поэтому создать и выполнить динамический тест, который обнаружит этот дефект, может быть непросто. Статическое тестирование может найти дефект гораздо меньшими усилиями.

Другое отличие состоит в том, что статическое тестирование может быть использовано для улучшения согласованности и внутреннего качества, а динамическое тестирование обычно фокусируется на видимом поведении программного обеспечения.

Типичные дефекты, которые легче и дешевле найти и исправить с помощью статического тестирования:

- Дефекты требований (например, несоответствия, неоднозначности, противоречия, упущения, неточности, избыточность)
- Конструктивные дефекты (например, неэффективные алгоритмы или структуры базы данных, сильные связи, низкая связность)
- Отклонения от стандартов (например, несоблюдение стандартов кодирования)

- Неправильные спецификации интерфейса (например, различные единицы измерения)
- Уязвимость системы (например, уязвимость переполнения буфера)
- Пробелы или неточности в отслеживаемости и охвате тестовой базы (например, отсутствие тестов для критериев приемки)

Более того, большинство типов дефектов ремонтопригодности можно найти только при статическом тестировании (например, неправильная модуляризация, плохое повторное использование компонентов, код, который трудно анализировать и модифицировать, не получая новые дефекты).

4.2 Процесс рецензирования

Рецензирование может быть формальными и неформальными. Неформальное рецензирование характеризуется отсутствием необходимости соблюдения конкретного процесса и отсутствием формальной документации. Формальное рецензирование характеризуются участием команды, документированием результата рецензирования и документированием процесса рецензирования. Формальность процесса рецензирования связана с такими факторами, как модель жизненного цикла разработки программного обеспечения, зрелость процесса разработки, сложность рассматриваемого рабочего продукта, любые юридические или нормативные требования и/или необходимость проведения аудита.

Фокус рецензирования зависит от согласованных целей (например, поиск дефектов, получение понимания, обучение тестировщиков и новых членов команды, или обсуждение и принятие решения путем консенсуса).

Стандарт ИСО (ISO/IEC 20246) содержит более подробное описание процесса рецензирования работы продукта, включая роли и методы рецензирования.

4.2.1 Процесс рецензирования рабочего продукта

Процесс рецензирования включает следующие основные мероприятия

Планирование

- Определение объема работы, который включает цель рецензирования, какие документы или части документа подлежат рассмотрению и характеристики качества, подлежащие оценке
- Оценка длительности и трудозатрат
- Определение характеристик рецензирования, таких как тип рецензирования, роли, действия и чек-листы
- Выбор людей для участия в рецензировании и распределение ролей
- Определение критериев входа и выхода для более формальных типов рецензирования (например, инспекций)
- Проверка соответствия критериев входа (для более формальных типов рецензирования)

Инициирование рецензирования

- Распространение рабочего продукта (физически или электронным способом) и таких материалов, как бланки описания дефектов, чек-листы и связанные с ними рабочие продукты
- Объяснение охвата, целей, процессов, ролей и рабочего продукта участникам процесса

- Ответы на любые вопросы, возникшие у участников при рецензировании

Индивидуальное рецензирование (то есть индивидуальная подготовка)

- Рецензирование всего или части рабочего продукта
- Определение потенциальных недостатков, рекомендаций и вопросов

Коммуникация по вопросам и анализ

- Сообщение выявленных потенциальных дефектов (например, на совещании по рецензированию)
- Анализ потенциальных дефектов, назначение их и установление статуса
- Оценка и документирование характеристик качества
- Оценка результатов рецензирования по критериям выхода для принятия решения по результатам обзора (отклонен; необходимы изменения; принять, возможно с незначительными изменениями)

Внесение изменений и отчетность

- Создание отчетов о дефектах для тех результатов, которые требуют изменений
- Исправления обнаруженных при рецензировании дефектов (как правило, сделанные автором)
- Обсуждение дефектов с соответствующим лицом или командой (когда найдены в рабочем продукте, обсуждаемом на рецензировании)
- Регистрация обновленного состояния дефектов (в официальном рецензировании), включающих соглашение автора комментария
- Сбор метрик (для формальных типов рецензирования)
- Проверка выполнения критериев выхода (для формальных типов рецензирования)
- Принятие рабочего продукта при достижении критериев выхода

Результаты рецензирования рабочего продукта различаются в зависимости от типа и формальности, что описано в разделе 3.2.3.

4.2.2 Роли и ответственности в формальном рецензировании

Типичное формальное рецензирование включает в себя следующие роли:

Автор

- Создает рассматриваемый рабочий продукт
- Исправляет дефекты в рассматриваемом продукте

Менеджер

- Отвечает за планирование рецензирования
- Принимает решение о проведении рецензирования
- Определяет персонал, бюджет и время
- Отслеживает текущую экономическую эффективность
- Принимает управленческие решения в случае неадекватных результатов

Ведущий (часто называется модератором)

- Обеспечивает эффективное проведение рецензирования
- При необходимости обеспечивает посредничество между различными точками зрения
- Часто это именно тот человек, от которого зависит успех рецензирования

Руководитель рецензирования

- Берет на себя общую ответственность за рецензирование
- Решает, кто будет участвовать, организует время и место проведения

Рецензенты

- Могут быть предметными экспертами, лицами, работающими в проекте, заинтересованными сторонами, заинтересованными в рабочем продукте лицами и/или лицами с узкими техническими или бизнес-знаниями
- Определяют потенциальные дефекты в рассматриваемом продукте
- Могут представлять различные взгляды на продукт (например, тестировщик, программист, пользователь, оператор, бизнес-аналитик, эксперт по используемости и т.д.)

Секретарь (или регистратор)

- Собирает потенциальные дефекты, обнаруженные в ходе индивидуальной проверки
- Записывает новые потенциальные дефекты, открытые вопросы и решения на совещании по рецензированию (во время проведения)

В некоторых типах рецензирования один человек может играть более, чем одну роль, а действия, связанные с одной ролью, также могут варьироваться в зависимости от типа рецензирования. Кроме того, с появлением инструментов поддержки процессов рецензирования, особенно для регистрации дефектов, открытых вопросов и решений, часто нет необходимости в секретаре. Также возможны более подробно описанные роли, как в стандарте ИСО (ISO/IEC 20246).

4.2.3 Типы рецензирования

Хотя рецензирование может использоваться по-разному, одна из основных целей – выявление дефектов. Все типы рецензирования могут помочь в обнаружении дефектов, а выбранный тип рецензирования среди других критериев должен основываться на потребностях проекта, доступных ресурсах, типе продукта и рисках, области бизнеса и корпоративной культуре. Типы рецензирования могут быть классифицированы в соответствии с различными параметрами. Ниже перечислены четыре наиболее распространенных типа обзора и связанных с ними параметрами.

Неформальное рецензирование (например, партнерская проверка, парное рецензирование)

- Основная цель: выявление потенциальных дефектов
- Возможны дополнительные цели: генерация новых идей и решений, быстрое решение мелких проблем
- Не основан на формальном (документированном) процессе
- Может не включать обзорную встречу
- Может быть выполнен коллегой автора (партнерская проверка) или другими людьми

- Результаты могут быть задокументированы
- Полезность зависит от рецензентов
- Необязательное использование чек-листов
- Очень часто используется в гибких методологиях

Пошаговый разбор

- Основные цели: найти дефекты, улучшить программный продукт, рассмотреть альтернативные варианты реализации, оценить соответствие стандартам и спецификациям
- Возможные дополнительные цели: обмен идеями о техниках и вариантах стиля, обучение участников, достижение консенсуса
- Индивидуальная подготовка перед совещанием по рассмотрению необязательна
- Обзорное собрание обычно проводится автором программного продукта
- Секретарь обязателен
- Использование чек-листов необязательно
- Может принимать форму сценариев, пробных прогонов или имитаций
- Могут быть получены записи потенциальных дефектов и отчеты по рецензированию
- На практике варьируется от довольно неформального до очень формального

Техническое рецензирование

- Основные цели: достижение консенсуса, выявление потенциальных дефектов
- Возможные дальнейшие цели: оценка качества и укрепление доверия к рабочему продукту, генерация новых идей, мотивация авторов, предоставление возможности авторам улучшить будущий рабочий продукт, рассмотрение альтернативных вариантов реализации
- Рецензенты должны быть специалистами той же отрасли, что и автор, а технические эксперты – специалистами в той же или другой дисциплине
- Требуется индивидуальная подготовка перед собранием по рецензированию
- Совещание по рецензированию – необязательно, в идеале проводится подготовленным ведущим (обычно не автором)
- Секретарь обязателен, в идеале – не автор
- Использование чек-листов необязательно
- Обычно получаются записи потенциальных дефектов и отчеты по рецензированию

Инспекция

- Основные цели: выявление потенциальных дефектов, оценка качества и укрепление доверия к рабочему продукту, предотвращение аналогичных дефектов в будущем с помощью обучения автора и анализа основных причин появления дефектов
- Возможные дальнейшие цели: мотивировать авторов, дать авторам возможность улучшить рабочий продукт и процесс разработки в будущем, достижение консенсуса
- Выполняется конкретный процесс с формальными документами на основе правил и чек-листов

- Используются четко определенные роли, указанные в разделе 3.2.2. Роли являются обязательными и могут включать специального чтеца (который читает рабочий продукт во время обзорного собрания)
- Требуется индивидуальная подготовка перед собранием по обзору
- Рецензенты – специалисты той же отрасли, что и автор или эксперты в других дисциплинах, имеющих отношение к рабочему продукту
- Указаны критерии входа и выхода
- Секретарь обязателен
- Совещание по рецензированию проводится специальным ведущим (не автором)
- Автор не может выступать в качестве руководителя рецензирования, чтеца или секретаря
- Создаются записи о дефектах и отчеты о рецензировании
- Собираются критерии и используются для улучшения всего процесса разработки программного обеспечения, включая процесс проверки

Один рабочий продукт может быть предметом более одного типа рецензирования. Если используется более, чем один тип рецензирования, порядок может отличаться. Например, неформальное рецензирование может быть проведено до технического рецензирования для того, чтобы убедиться, что продукт готов к техническому обзору.

Типы рецензирования, описанные выше, могут быть выполнены специалистами одной области, то есть проведены коллегами по аналогии с приблизительным организационным уровнем.

Типы дефектов, обнаруженных при рецензировании, различаются, в частности зависят от рассматриваемого рабочего продукта. См. раздел 3.1.3, где рассматриваются примеры дефектов, которые можно найти при рецензировании разных рабочих продуктов и см. [Glib 1993] для получения информации о формальном рецензировании.

4.2.4 Применение методов рецензирования

Существует ряд методов рецензирования, которые применяются для индивидуальных проверок (то есть для индивидуальной подготовки) для выявления дефектов. Эти методы могут использоваться во всех описанных выше типах рецензирования. Эффективность метода может отличаться в зависимости от типа используемого рецензирования. Примеры различных методов индивидуального рецензирования перечислены ниже.

Свободное рецензирование

При свободном рецензировании предоставляется небольшое или вообще отсутствует руководство по выполнению. Рецензенты часто последовательно читают продукт, идентифицируя и документируя проблемы, с которыми они сталкиваются. Свободное рецензирование – это часто используемый метод, требующий небольшой подготовки. Эта техника в значительной степени зависит от навыков рецензента и может привести к повторяющимся вопросам у разных рецензентов.

Рецензирование, основанное на чек-листах

Рецензирование, основанное на чек-листах, представляет собой систематический метод, при котором рецензенты обнаруживают проблемы, основанные на чек-листах, распространяемых в начале рассмотрения (например, ведущим). Чек-лист состоит из набора вопросов, основанных на потенциальных дефектах, определенных исходя их опыта. Чек-листы должны соответствовать типу рассматриваемого рабочего продукта, их следует регулярно обновлять для

охвата проблем, пропущенных в предыдущих рецензированиях. Основным преимуществом методики, основанной на чек-листах, является систематизированное покрытие характерных типов дефектов. Нужно следить за тем, чтобы искать дефекты не только по чек-листу, но и за его пределами.

Рецензирование по сценарию и сухие прогоны

В рецензировании по сценариям рецензентам предоставляются структурированные рекомендации о том, как нужно рассматривать рабочий продукт. Сценарный подход поддерживает рецензентов при выполнении сухих прогонов рабочего продукта, основанных на ожидаемом использовании продукта (если продукт задокументирован в подходящем формате, например, в формате сценариев использования системы). Эти сценарии, в отличие от чек-листов, дают рецензентам более четкое представление о том, как идентифицировать специфические типы дефектов. Как и с рецензированием по чек-листам, чтобы не пропустить другие типы дефектов (например, отсутствующие функции), рецензенты не должны ограничиваться документированным сценарием.

Ролевое рецензирование

Ролевое рецензирование – это метод, в котором рецензенты оценивают рабочий продукт с точки зрения отдельных ролей заинтересованных сторон. Типичные роли – это конкретные типы конечных пользователей (опытные, неопытные, взрослый, ребенок и т.д.), а также конкретные роли в организации (администратор, системный администратор, тестировщик производительности и т.д.).

Рецензирование на основе точки зрения

При анализе на основе точки зрения, аналогичном ролевому анализу, рецензенты берут на себя различные точки зрения заинтересованных сторон при индивидуальном рассмотрении. Типичные точки зрения заинтересованных сторон включают в себя конечного пользователя, маркетолога, дизайнера, тестировщика или оператора. Использование различных точек зрения заинтересованных сторон приводит к большей глубине при индивидуальном рассмотрении с меньшим дублированием вопросов среди рецензентов.

Кроме того, чтение на основе точки зрения требует от рецензентов поиска пути применения рабочего продукта, что позволит разработать нужный продукт. Например, тестировщик пытается сгенерировать приемочные испытания на основе точки зрения по требованиям, чтобы узнать, достаточно ли информации по продукту было включено. Кроме того, при чтении, основанном на точке зрения, ожидается использование чек-листов.

Эмпирические исследования показали, что рецензирование на основе точки зрения является наиболее эффективным общим методом рассмотрения требований к техническим продуктам. Ключевым фактором успеха является основанное на рисках рассмотрение и взвешивание различных точек зрения заинтересованных сторон. См. [Shul 2000] для получения подробной информации о рецензировании, основанном на точке зрения и [Sauer 2000] для эффективности различных типов рецензирования.

4.2.5 Факторы успеха рецензирования

Чтобы получить успешное рецензирование, должны использоваться соответствующие типы и методы рецензирования. Кроме того, существует ряд других факторов, которые влияют на результаты рецензирования.

Организационные факторы успеха включают:

- Каждое рецензирование имеет четкие цели, определенные при планировании рецензирования, и используемые как измеримые критерии выхода

- Применяются типы рецензирования, которые подходят для типа и уровня программных продуктов и участников
- Любые используемые методы рецензирования, такие как анализ на основе чек-листов или на основе ролей, пригодны для эффективной идентификации дефектов в рабочем продукте, подлежащем рецензированию
- Любые чек-листы учитывают основные риски и обновляются
- Большие документы пишутся и пересматриваются небольшими частями, поэтому контроль качества осуществляется путем предоставления авторам отзывов о дефектах на более ранней стадии и более часто
- У участников есть достаточно времени для подготовки
- Рецензирование запланировано с соответствующим уведомлением
- Менеджмент поддерживает процессы обзоров (например, путем предоставления достаточного времени для анализа в рамках проекта)

Связанные с людьми факторы успеха обзоров:

- Для достижения целей обзора вовлечены нужные люди, например, с разными наборами навыков и точками зрения, использующие документ в качестве начала работы
- Тестировщики рассматриваются как ценные рецензенты, вносящие свой вклад в обзор и узнающие о работе продукта, что позволяет составлять более эффективные тесты на более ранних стадиях
- Участники уделяют достаточно времени и внимания деталям
- Рецензирование проводится на небольших частях продукта, рецензенты не теряют концентрацию во время индивидуального рецензирования и/или во время обзорного собрания (когда проводится)
- Найденные дефекты признаются, оцениваются и обрабатываются объективно
- Совещание эффективно управляется, поэтому участники считают его полезным использованием своего времени
- Рецензирование проводится в атмосфере доверия; результат не используется для оценки участников
- Участники избегают поведения, указывающего на скуку, раздражение или враждебность к другим участникам
- Обеспечивается адекватное обучение, особенно для более формальных типов рецензирования, таких как инспекция
- Повышается культура обучения и совершенствуются процессы

См. [Glib 1993], [Wiegers 2002] и [van Veenendaal 2004] для получения дополнительной информации об успешном рецензировании.

5. Методы проектирования тестов – 330 мин

Ключевые слова

разработка тестов методом черного ящика, анализ граничных значений, тестирование на основе чек-листов, покрытие, покрытие условий, тестирование с помощью таблицы альтернатив, предположение об ошибках, эквивалентное разбиение, метод проектирования тестов на основе опыта, исследовательское тестирование, тестирование таблицы переходов, покрытие операторов, метод проектирования тестов, тестирование по сценариям использования, разработка тестов методом белого ящика

Цели обучения для главы «Методы проектирования тестов»

4.1 Категории методов проектирования тестов

FL-4.1.1 (K2) Рассказать о характеристиках, сходствах и различиях между тестированием методом черного ящика, тестированием методом белого ящика и тестированием на основе опыта.

4.2 Тестирование методом черного ящика

FL-4.2.1 (K3) Применить метод эквивалентного разбиения для создания тестовых сценариев на основе заданных требований.

FL-4.2.2 (K3) Применить метод анализа граничных значений для создания тестовых сценариев на основе заданных требований.

FL-4.2.3 (K3) Применить метод таблицы альтернатив для создания тестовых сценариев на основе заданных требований.

FL-4.2.4 (K3) Применить метод тестирования таблицы переходов для создания тестовых сценариев на основе заданных требований.

FL-4.2.5 (K2) Объяснить, как создать тестовые сценарии из сценария использования.

4.3 Тестирование методом белого ящика

FL-4.3.1 (K2) Объяснить смысл покрытия операторов

FL-4.3.2 (K2) Объяснить смысл покрытия условий

FL-4.3.3 (K2) Объяснить пользу покрытия условий и операторов

4.4 Тестирование на основе опыта

FL-4.4.1 (K2) Объяснить смысл предположения об ошибках

FL-4.4.2 (K2) Объяснить смысл исследовательского тестирования

FL-4.4.3 (K2) Объяснить смысл тестирования на основе чек-листов

5.1 Категории методов проектирования тестов

Цель метода проектирования тестов, включая перечисленные ниже, заключается в определении тестовых условий, тестовых сценариев и тестовых данных.

5.1.1 Выбор метода проектирования тестов

Выбор конкретного метода проектирования тестов зависит от множества факторов, включая:

- Тип системы или компонента
- Сложность системы или компонента
- Нормативные документы
- Требования пользователей или контракта
- Уровни рисков
- Типы рисков
- Задачи тестирования
- Доступную документацию
- Знания и опыт тестировщиков
- Доступные инструменты
- Ресурсы времени и бюджет
- Жизненный цикл разработки
- Ожидаемое использование системы
- Прошлый опыт использования методов проектирования тестов для компонента или системы
- Ожидаемые типы дефектов компонента или системы

Некоторые методы применимы к конкретным ситуациям и уровням тестирования, другие применимы на всех уровнях. Обычно, тестировщики используют комбинации различных методов в процессе создания тестовых сценариев для достижения наилучших результатов.

Применение тех или иных методов на этапе анализа, дизайна и реализации тестов может варьироваться от неформального (минимум либо полное отсутствие документации) до строго формального. Уровень формальности зависит от контекста, включая зрелость процессов разработки и тестирования, временных ограничений, требований безопасности и требований регуляторов, знания и опыта вовлеченных людей, а также используемой модели жизненного цикла.

5.1.2 Категории методов проектирования тестов и их характеристики

Здесь и далее методы проектирования тестов делятся на методы черного ящика, методы белого ящика и методы, основанные на опыте.

Методы черного ящика (поведенческие, или методы, основанные на поведении) основываются на анализе соответствующего базиса тестирования (формальных требований, спецификаций, сценариев использования, пользовательских историй или бизнес-процессов). Эти методы применимы как для функционального, так и для нефункционального тестирования. Методы

черного ящика сосредотачиваются на связи входных данных и выходных результатов объекта тестирования, а не на его внутренней структуре.

Методы белого ящика (структурные, или методы, основанные на структуре) основаны на анализе архитектуры, детального проектирования, внутренней структуры или кода компонента либо системы. В отличие от методов черного ящика методы белого ящика сосредотачиваются на структуре и обработке внутри объекта тестирования.

Методы, основанные на опыте, используют опыт разработчиков, тестировщиков и пользователей для проектирования, реализации и выполнения тестов. Их часто совмещают с методами черного и белого ящиков.

Общие характеристики методов черного ящика:

- Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя требования, спецификации, сценарии использования и пользовательские истории
- Тестовые сценарии могут использоваться для определения несоответствий и отклонений между требованиями и реализацией
- Измерение покрытия основано на элементах базиса тестирования и методе проектирования, применяемом к базису тестирования

Общие характеристики методов белого ящика:

- Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя код, архитектуру, детальную архитектуру или любой другой источник информации о структуре программного обеспечения
- Измерение покрытия основано на элементах структуры (коде, интерфейсах и т.д.)
- Спецификации используются как источник дополнительной информации для определения ожидаемых результатов тестовых сценариев

Общие характеристики методов, основанных на опыте:

- Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя знания и опыт тестировщиков, разработчиков, пользователей и других заинтересованных лиц

Эти знания и опыт включают в себя возможное использование ПО, его окружение, возможные дефекты и их распределение.

Международный стандарт ИСО (ISO/IEC/IEEE 29119-4) содержит описание методов проектирования тестов и соответствующие им методы измерения покрытия (см. также [Craig 2002] и [Copeland 2004] для описания дополнительных методов).

5.2 Методы черного ящика

5.2.1 Эквивалентное разбиение

Эквивалентное разбиение делит данные на группы (классы эквивалентности), которые, как предполагается, обрабатываются схожим образом [Kaner 2013], [Jorgensen 2014]. Области эквивалентности могут быть как для правильных, или позитивных, так и неправильных, или негативных, значений.

- Позитивные значения – это значения, которые должны быть приняты. Класс, содержащий позитивные значения, называется «действительный класс эквивалентности».

- Негативные значения – значения, которые должны быть отвергнуты. Класс, содержащий негативные значения, называется «недействительный класс эквивалентности».
- Классы могут быть определены для любых данных, относящихся к объекту тестирования, включая: входные данные, выходные данные, внутренние данные, данные, связанные со временем (например, до или после события), интерфейсные параметры (например, в случае интеграционного тестирования компонентов).
- Любой класс может быть при необходимости разделен на подклассы
- Каждое значение должно принадлежать только одному классу эквивалентности
- Во избежание маскирования дефектов негативные классы эквивалентности в тестовых сценариях следует использовать по отдельности, то есть избегать комбинаций одних негативных классов с другими. Дефекты могут быть маскированы, если при наличии нескольких дефектов обнаруживается только один из них.

Для достижения 100% покрытия с помощью этого метода, тестовые сценарии должны покрывать все позитивные и негативные классы, проверяя хотя бы одно значение из каждого класса. Покрытие вычисляется как отношение количества тестируемых классов к общему числу классов. Метод эквивалентного разбиения применим на всех уровнях тестирования.

5.2.2 Анализ граничных значений

Метод анализа граничных значений является продолжением метода эквивалентного разбиения, но может быть применим, только если классы состоят из упорядоченных числовых значений. Максимальное и минимальное значение класса являются его границами [Beizer 1990].

Для примера, предположим, что некоторое поле ввода принимает положительное целое число от 0 до 9; ввод осуществляется через клавиатуру, поэтому нечисловые входные значения исключены. Допустимы значения от 1 до 5 включительно. Таким образом, можно выделить три области эквивалентности: негативная (слишком маленькие), позитивная, негативная (слишком большие). Для позитивной области эквивалентности значения 1 и 5 будут граничными. Для области с негативными большими значениями границами будут значения 6 и 9. Для области с негативными малыми значениями будет только одна граница – 0, поскольку эта область состоит из одного элемента.

В рассмотренном выше примере мы можем определить по два граничных значения на каждую из границ. Граница между негативными малыми и позитивными значениями дает нам тестовые значения 0 и 1. Граница между позитивными и негативными большими значениями дает нам тестовые значения 5 и 6. Вариация данного метода определяет три граничных значения на каждую из границ: значения перед, на и сразу после границы. Для рассматриваемого примера применение этого метода даст следующие тестовые значения: для нижней границы – 0,1,2; для верхней границы – 4,5,6 [Jorgensen 2014].

Некорректное поведение более вероятно на границах класса, чем внутри класса. Важно помнить, что специфицированные и реализованные границы могут быть смещены вверх или вниз относительно истинных значений, они могут быть пропущены или дополнены новыми границами. Анализ и тестирование граничных значений может обнаружить большинство подобных дефектов, вынуждая программное обеспечение демонстрировать поведение, относящееся к области эквивалентности, отличной от той, к которой должна относиться граничная точка.

Анализ граничных значений может использоваться на любом уровне тестирования. Данный метод применяется при тестировании требований, в которых присутствуют диапазоны значений (включая даты и время). Покрытие вычисляется как отношение числа тестируемых граничных значений к общему числу граничных значений и чаще всего выражается в процентах.

5.2.3 Тестирование с помощью таблицы альтернатив

Комбинаторные методы полезны при тестировании требований, содержащих условия, которые дают разные результаты в зависимости от комбинаций. Одним из таких методов является тестирование с помощью таблицы альтернатив.

Таблицы альтернатив – хороший способ записи сложных бизнес-правил, которые должны быть реализованы в системе. В процессе создания таблицы, тестировщик определяет условия (входы) и результирующие действия системы (выходы). Пары условий и действий образуют строки таблицы, при этом условия указываются сверху, а действия – снизу. Каждый столбец представляет собой бизнес-правило с уникальной комбинацией условий и действий, связанных с этим правилом.

Значения условий часто отображаются в виде логических (истина или ложь) или дискретных (красный, синий, зеленый) значений, но могут быть также в виде чисел или числовых диапазонов. В одной таблице могут сочетаться значения разных типов.

Обозначения, используемые для таблиц альтернатив:

Условия:

- Y означает, что условие истинно (используется также обозначение T или 1)
- N означает, что условие ложно (используется также обозначение F или 0)
- – означает, что значение условия может быть любым (используется также обозначение N/A)

Действия:

- X означает, что действие должно быть выполнено (используется также обозначение Y, T или 1)
- Пустое поле означает, что действие не должно выполняться (используется также обозначение N, F или 0)

Полная таблица альтернатив содержит по столбцу на каждую комбинацию условий. Таблицу можно сократить, убрав столбцы, которые содержат несуществующие комбинации или комбинации, не влияющие на результат. Более подробная информация размещена в программе подготовки продвинутого уровня (ISTQB-ATA Продвинутый уровень, Тест-аналитик).

Стандарт покрытия для таблиц альтернатив подразумевает наличие хотя бы одного теста для каждого столбца таблицы. Обычно это подразумевает покрытие всех комбинаций условий. Покрытие измеряется как отношение количества правил, проверенных хотя бы одним тестом, к общему числу правил, выраженное в процентах.

Преимущество метода заключается в том, что он выявляет комбинации условий, которые могли быть не проверены при тестировании. Метод помогает определить несоответствия в требованиях, может быть применен во всех ситуациях и на любом уровне, где поведение программного обеспечения зависит от комбинации условий.

5.2.4 Тестирование с помощью таблицы переходов

В зависимости от прошлых условий и состояния система может вести себя по-разному. Описать прошлое системы можно с помощью концепции состояний. Диаграмма состояний и переходов показывает начальное и конечное состояния системы, а также описывает переходы между состояниями. Каждый переход вызывается событием (например, вводом данных пользователем). Если одно и то же событие может привести к разным переходам, выбор

перехода может задаваться контрольным условием. Смена состояния может завершаться выполнением какого-либо действия (вывод результатов или сообщения об ошибке и т.д.).

Таблица переходов представляет собой все возможные комбинации начальных и конечных состояний, включая действительные и недействительные переходы, инициирующие события, защитные условия и результирующие действия. Диаграммы состояний и переходов обычно, показывают только действительные переходы и исключают недействительные переходы.

Тесты создаются для покрытия типичной последовательности состояний, покрытия каждого возможного состояния, покрытия каждого возможного перехода, проверки специфических последовательностей переходов, или для проверки недействительных переходов.

Тестирование с помощью таблицы переходов наиболее распространено в сфере встроенного ПО и при тестировании программных меню. Метод также подходит для моделирования бизнес-сценариев, имеющих конкретные состояния, или для тестирования переходов по экранным формам. Концепция состояний абстрактна, она может отражать несколько строк кода или целый бизнес-процесс.

Покрытие измеряется как отношение числа протестированных состояний или переходов к общему числу состояний или переходов в объекте тестирования, выраженное в процентах. Более подробная информация размещена в программе подготовки продвинутого уровня (ISTQB-ATA Продвинутый уровень, Тест-аналитик).

5.2.5 Тестирование с помощью сценариев использования

Тесты можно разработать на основе сценариев использования, которые представляют собой способ описания взаимодействий с программными объектами. В сценариях использования всегда присутствуют участники (пользователи, внешние устройства и компоненты) и субъекты (компоненты или системы, к которым применяется сценарий использования).

Каждый сценарий использования описывает поведение субъекта при взаимодействии с участником (UML 2.5.1 2017). Сценарий использования может быть описан взаимодействиями и активностями, предусловиями и постусловиями или естественным языком, если это возможно. Взаимодействия между участниками и субъектом могут приводить к изменению состояния субъекта; они могут изображаться графически, с помощью диаграмм или моделей бизнес-процессов.

Сценарий использования может отражать различные варианты поведения, включая исключительные ситуации и обработку ошибок (реакцию системы и восстановление из-за ошибок программирования, приложений и связи, например, в результате чего появляется сообщение об ошибке). Тесты разрабатываются с целью проверки различных вариантов поведения (базового, исключительного, альтернативного, обработки ошибок). Покрытие может выражаться как процент протестированных вариантов поведения к общему числу вариантов.

Более подробная информация о покрытии размещена в программе подготовки продвинутого уровня (ISTQB-ATA Продвинутый уровень, Тест-аналитик).

5.3 Методы белого ящика

Тестирование белого ящика основывается на внутренней структуре объекта тестирования. Методы могут применяться на всех этапах, однако, методы, рассмотренные ниже, чаще всего используются в модульном тестировании.

Существуют и другие методы, используемые в тестировании критичных систем и обеспечивающие более сильное покрытие, но здесь они не рассматриваются. Более подробная

информация об этих методах размещена в программе подготовки продвинутого уровня (ISTQB Продвинутый уровень, Технический тест-аналитик).

5.3.1 Тестирование и покрытие операторов

Тестирование операторов направлено на проверку исполняемых операторов в коде. Покрытие вычисляется как отношение количества операторов, выполненных тестом, к общему числу операторов в тестируемом коде.

5.3.2 Тестирование и покрытие условий

Тестирование условий направлено на проверку логических условий в коде, а также кода, выполняемого в зависимости от исхода условия. Для этого тесты следуют потокам управления, которые выходят из условия (путь для выхода «истина» и для выхода «ложь»; для оператора выбора (CASE) тесты потребуются для всех возможных результатов, включая выход по умолчанию).

Покрытие вычисляется как отношение числа исходов условий, проверенных тестом, к общему числу исходов тестируемых условий.

5.3.3 Ценность тестирования операторов и условий

Стопроцентное покрытие операторов означает проверку всех исполняемых инструкций в коде хотя бы один раз, но это не дает уверенности в проверке логики условий. Из двух методов, рассматриваемых здесь, тестирование операторов обеспечивает более слабое покрытие, чем тестирование условий.

Стопроцентное покрытие условий проверяет все ветки потока управления, что включает проверку выходов «истина» и «ложь» для условных операторов. Покрытие условий позволяет находить ситуации, в которых исполняемый код может быть пропущен в зависимости от результата условия.

Стопроцентное покрытие условий гарантирует стопроцентное покрытие операторов, но не наоборот.

5.4 Методы, основанные на опыте

При использовании методов тестирования, основанных на опыте, тесты создаются на основе умения, интуиции и опыта тестировщика. Данные методы могут пригодиться при создании тестов, которые невозможно получить, применяя другие, более системные методы. В зависимости от выбранного подхода и прошлого опыта можно получить очень разные степени покрытия и эффективности тестирования. Измерение покрытия для данных методов может быть затруднительным или вообще невозможным.

Наиболее популярные методы рассматриваются в последующих разделах.

5.4.1 Предположение об ошибках

Предположение об ошибках – это способ предотвращения ошибок, дефектов и отказов, основанный на знаниях тестировщика, включающих:

- Историю работы приложения в прошлом
- Наиболее вероятные типы дефектов, допускаемых при разработке

- Типы дефектов, которые были обнаружены в схожих приложениях

Структурированный подход к предположению об ошибках предполагает создание списка всех возможных ошибок, дефектов и отказов с последующей разработкой тестов, направленных на поиск дефектов из этого списка. Списки отказов и дефектов могут быть построены на основе опыта, исторических данных об отказах и ошибках, а также на общих знаниях о причинах отказа программ.

5.4.2 Исследовательское тестирование

Во время исследовательского тестирования неформальные (т.е. не созданные заранее) тестовые сценарии разрабатываются, выполняются, анализируются и оцениваются динамически во время выполнения тестов. Результаты тестирования используются для изучения компонента или системы и последующей разработки тестовых сценариев для непокрытых областей.

Исследовательское тестирование может проводиться сессиями, что позволяет структурировать активность. При использовании сессионного подхода, исследовательское тестирование выполняется в определенном промежутке времени, при этом тестировщик использует концепцию тестирования, содержащую цели, и отмечает выполненные действия и обнаруженные факты.

Исследовательское тестирование лучше всего подходит в ситуациях, когда документация недостаточная, либо вовсе отсутствует, в условиях очень сжатых сроков и как дополнение к другим, более формальным, методам тестирования.

Исследовательское тестирование относится к реактивным стратегиям тестирования и может включать использование различных методов черного и белого ящиков или методов, основанных на опыте.

5.4.3 Тестирование на основе чек-листов

При тестировании по чек-листам тестировщик проектирует, реализует и выполняет тесты, покрывающие тестовые условия, указанные в чек-листе. В качестве составной части анализа тестировщики могут создавать новые или расширять чек-листы, либо использовать готовые чек-листы, не меняя их. Такие списки могут быть построены на опыте, на исторических данных об ошибках, на информации о приоритетах для пользователей и понимании, как и почему происходят отказы в программе.

Чек-листы могут создаваться для поддержки разных видов тестирования, включая функциональное и нефункциональное тестирование. При отсутствии детальных тестовых сценариев, чек-листы помогают определить направления тестирования и увеличивают согласованность тестирования. Поскольку чек-листы содержат общее описание, это снижает повторяемость результатов.

6. Управление тестированием – 225 мин

Ключевые слова

управление конфигурацией, управление дефектами, критерии входа, критерии выхода, риск продукта, риск проекта, риск, уровень риска, тестирование, основанное на рисках, подход к тестированию, контроль тестирования, оценка затрат на тестирование, руководитель тестирования, мониторинг тестирования, план тестирования, планирование тестирования, отчет о ходе тестирования, стратегия тестирования, итоговый отчет о тестировании, тестировщик

Цели обучения для главы «Управление тестированием»

5.1 Организация тестирования

FL-5.1.1 (K2) Объяснить преимущества и недостатки независимого тестирования

FL-5.1.2 (K1) Определить задачи руководителя тестирования и тестировщика

5.2 Планирование и оценка тестирования

FL-5.2.1 (K2) Кратко описать цели и содержания плана тестирования

FL-5.2.2 (K2) Описать различия между различными стратегиями тестирования

FL-5.2.3 (K2) Привести примеры потенциальных критериев входа и выхода

FL-5.2.4 (K3) Применить знания о приоритезации, технических и логических зависимостях к графику выполнения теста для заданного набора тестовых сценариев

FL-5.2.5 (K1) Определить факторы, которые влияют на затраты, связанные с тестированием

FL-5.2.6 (K2) Объяснить разницу между двумя подходами: оценкой, основанной на метриках, и экспертной оценкой

5.3 Контроль и мониторинг

FL-5.3.1 (K1) Вспомнить метрики, используемые в тестировании

FL-5.3.2 (K2) Кратко описать цели, содержание и аудиторию отчета о тестировании

5.4 Управление конфигурацией

FL-5.4.1 (K2) Кратко описать, как управление конфигурацией поддерживает тестирование

5.5 Риски и тестирование

FL-5.5.1 (K1) Оценить уровень риска, используя вероятность и влияние

FL-5.5.2 (K2) Указать различие рисков проекта и продукта

FL-5.5.3 (K2) Описать, используя примеры, как анализ риска продукта может влиять на тщательность и объем тестирования

5.6 Управление дефектами

FL-5.6.1 (K3) Написать отчет, описывающий дефекты, найденные во время тестирования

6.1 Организация тестирования

6.1.1 Независимость тестирования

Задачи тестирования могут выполняться как людьми в конкретной роли, связанной с тестированием, так и людьми в иной роли (например, заказчиками). Конкретный уровень независимости часто делает тестировщика более эффективным в поиске дефектов из-за психологических различий разработчика и тестировщика (см. раздел 1.5). Независимость не является, однако, альтернативой осведомленности, и разработчики могут находить много дефектов в своем собственном коде.

Уровни независимости тестирования включают следующие (от низкого уровня до высокого уровня независимости):

- Нет независимых тестировщиков; разработчики тестируют собственный код, других форм тестирования нет
- Независимые разработчики или тестировщики в команде разработки или в команде проекта; это могут быть разработчики, тестирующие продукты своих коллег
- Независимая команда или группа тестирования внутри организации, отчитывающаяся перед руководством проекта или руководством организации
- Независимые тестировщики из организации заказчика или сообщества пользователей. Могут специализироваться на отдельных типах тестирования, таких как тестирование практичности, тестирование безопасности, тестирование производительности, тестирование на соответствие нормативным документам или тестирование переносимости
- Независимые тестировщики, внешние по отношению к организации, работающие либо на территории компании (инсорсинг), либо вне ее (аутсорсинг)

Для большинства проектов лучше иметь несколько уровней тестирования, часть из которых выполняется независимыми тестировщиками. Разработчики должны участвовать в тестировании, особенно на более низких уровнях, чтобы контролировать качество своей работы.

Способ организации независимого тестирования зависит от модели жизненного цикла. Например, в гибкой разработке тестировщики могут быть частью команды разработчиков. В некоторых организациях, использующих гибкие методологии разработки программного обеспечения, тестировщики могут также считаться частью большой независимой группы тестирования. Кроме того, в таких организациях владельцы продуктов могут выполнять приемочные испытания для проверки пользовательских историй в конце каждой итерации.

К потенциальным преимуществам независимого тестирования можно отнести:

- Эффективное распознавание различных видов отказов по сравнению с разработчиками из-за разницы подходов, технических перспектив и предубеждений
- Возможность независимого тестировщика проверять, оспаривать или опровергать допущения, сделанные заинтересованными сторонами во время проектирования и внедрения системы

К потенциальным недостаткам независимого тестирования относятся:

- Изоляция от команды разработчиков, что приводит к отсутствию сотрудничества, задержкам с предоставлением обратной связи команде разработчиков или соперничеству с командой разработчиков
- Потеря разработчиками чувства ответственности за качество

- Восприятие независимых тестировщиков как «узкого места» и обвинения в задержках релиза.
- Недостаточность у независимых тестировщиков какой-либо важной информации (например, об объекте тестирования).

Многие организации могут успешно пользуются преимуществами независимого тестирования, избегая при этом указанных недостатков.

6.1.2 Задачи руководителя тестирования и тестировщика

В этой программе рассматриваются две тестовые роли - руководителя тестирования и тестировщика. Действия и задачи, выполняемые людьми в этих двух ролях, зависят от контекста проекта и продукта, навыков людей в этих ролях и организации.

Руководитель тестирования отвечает за процесс тестирования и успешное руководство активностями тестирования. Роль руководителя тестирования может выполняться профессиональным руководителем тестирования, руководителем проекта, руководителем разработки, или руководителем по обеспечению качества. В крупных проектах или организациях несколько команд тестирования могут отчитываться перед одним руководителем тестирования, куратором или координатором тестирования, при этом каждая команда возглавляется лидером тестирования или ведущим тестировщиком.

Типичные задачи руководителя тестирования могут включать:

- Разработку или рецензирование тестовой политики и стратегии тестирования для организации
- Планирование тестирования, с учетом контекста и понимания целей и рисков тестирования. Это может включать выбор тестовых подходов, оценку времени тестирования, трудозатрат и стоимости, привлечение ресурсов, определение уровней тестирования и циклов тестирования и планирование управления дефектами
- Составление и обновление планов тестирования
- Согласование планов тестирования с руководителями проектов, владельцами продуктов и другими участниками
- Координацию активностей тестирования с другими проектами, такими как планирование интеграции
- Инициирование анализа, разработки, реализации и выполнения тестов, отслеживание прогресса и результатов тестирования, контроль выполнения критериев выхода (или критериев готовности)
- Подготовку и предоставление отчетов о статусе тестирования и сводных отчетов о тестировании на основе собранной информации
- Адаптацию планов в зависимости от прогресса и результатов тестирования (и выполнение действий, необходимых для контроля тестирования)
- Поддержку настройки системы управления дефектами и конфигурацией тестового обеспечения
- Выбор подходящих метрик для измерения результатов тестирования и оценки качества процесса тестирования и продукта
- Выбор и внедрение инструментов поддержки процесса тестирования, включая рекомендации по выбору инструментария (и, возможно, приобретение и / или поддержку),

выделение времени и трудозатрат для пилотных проектов, обеспечение постоянной поддержки в использовании инструмента/инструментов

- Решения о создании тестовой среды/сред
- Демонстрацию ценности тестировщиков, группы тестирования и профессии тестировщика в организации
- Развитие навыков и карьеры тестировщиков (например, посредством планов обучения, оценки эффективности, коучинга и т.д.)

Реализация роли руководителя тестирования зависит от используемой модели жизненного цикла разработки. Например, в гибкой разработке некоторые из упомянутых выше задач выполняются командой, особенно те, что связаны с повседневным тестированием. Они выполняются внутри команды, часто с помощью тестировщиков, работающих в команде. Задачи, охватывающие несколько команд или всю организацию, или связанные с управлением персоналом, могут выполняться руководителями тестирования (иногда их называют кураторами тестирования) за пределами команды разработки. См. [Black 2009] для получения дополнительной информации об управлении процессом тестирования.

К типичным задачам тестировщика могут относиться:

- Рецензирование и разработка планов тестирования
- Анализ, рецензирование и оценка требований, пользовательских историй и критериев приемки, спецификаций и моделей (базиса тестирования) на предмет тестируемости
- Определение и документирование тестовых условий, установление связей между тестовыми сценариями, тестовыми условиями и базисом тестирования
- Проектирование, настройка и проверка тестовой среды/сред, зачастую вместе с системным администрированием и управлением сетью
- Проектирование и разработка тестовых сценариев и процедур тестирования
- Подготовка и получение тестовых данных
- Создание подробного расписания выполнения тестов
- Выполнение тестирования, оценка результатов и документирование отклонений от ожидаемых результатов
- Использование соответствующих инструментов поддержки процесса тестирования
- Автоматизация процесса тестирования по мере необходимости (может поддерживаться разработчиком или экспертом по автоматизации тестирования)
- Оценка нефункциональных характеристик, таких как производительность, надежность, практичность, безопасность, совместимость и переносимость
- Рецензирование тестов, разработанных другими тестировщиками

Люди, занимающиеся тест-анализом, проектированием тестов, выполняющие специфические виды тестирования или занимающиеся автоматизацией, могут быть специалистами в этих ролях. В зависимости от рисков проекта и продукта, а также используемой модели жизненного цикла, роль тестировщика на различных уровнях тестирования могут выполнять разные люди. Например, на уровне компонента и интеграции компонентов роль тестировщика часто выполняют разработчики. На уровне приемки роль тестировщика могут выполнять бизнес-аналитики, эксперты и пользователи. На уровне системы и интеграции систем роль тестировщика могут выполнять специалисты независимой команды тестирования. На уровне эксплуатационной

приемки роль тестировщика часто выполняют специалисты по сопровождению и/или администрированию системы.

6.2 Планирование и оценка тестирования

6.2.1 Цель и содержание плана тестирования

В плане тестирования перечисляются работы по тестированию для проектов разработки и сопровождения. Планирование зависит от политики тестирования и стратегии тестирования организации, используемых методов и жизненных циклов разработки (см. раздел 2.1), объема тестирования, целей, рисков, ограничений, критичности, тестируемости и доступности ресурсов.

По ходу проекта становится доступно больше информации, и в план тестирования могут быть включены дополнительные сведения. Планирование тестирования - непрерывная деятельность, которая выполняется в течение всего жизненного цикла продукта. (Обратите внимание, что жизненный цикл продукта может выходить за рамки проекта и включать в себя фазу сопровождения.) Обратная связь в ходе работ по тестированию должна использоваться для идентификации изменяющихся рисков с последующей корректировкой планов. План может быть оформлен как в виде главного плана тестирования, так и в виде уровневых планов (план системного тестирования, план приемочного тестирования) или планов для отдельных видов тестирования (план тестирования практичности, план тестирования производительности). Мероприятия по планированию тестирования могут включать в себя следующие работы, часть из которых может быть отражена в плане тестирования:

- Определение объема, целей и рисков тестирования
- Определение общего подхода к тестированию
- Координацию работ по тестированию и их совмещение с другими работами в рамках жизненного цикла программного обеспечения
- Принятие решений о том, что тестировать, кто будет выполнять тестирование и как должны выполняться работы по тестированию
- Планирование анализа, проектирования, реализации и выполнения тестов, оценки результатов тестирования с указанием сроков (при последовательной разработке) либо итераций (при итеративной разработке)
- Выбор метрик для мониторинга и контроля тестирования
- Формирование бюджета тестирования
- Определение структуры и уровня детализации тестовой документации (например, путем предоставления шаблонов или примеров документов)

Содержание планов тестирования различается и может выходить за пределы указанных тем. Примеры планов тестирования можно найти в стандарте ИСО (ISO/IEC/IEEE 29119-3).

6.2.2 Стратегия тестирования и подходы к тестированию

Стратегия тестирования содержит верхнеуровневое описание процесса тестирования, как правило, на уровне продукта или организации. К распространенным типам стратегий тестирования относятся:

- **Аналитический подход –** базируется на анализе некоторого фактора (например, требования или риска). Тестирование, основанное на рисках, является примером

аналитического подхода, при котором тесты разрабатываются и ранжируются по приоритетам в зависимости от уровня риска.

- **Тестирование на основе моделей –** подход, при котором тесты разрабатываются на основании модели некоторого аспекта продукта, такого как функция, бизнес-процесс, внутренняя структура или нефункциональная характеристика (например, надежность). Примерами являются модели бизнес-процессов, модели состояния и модели роста надежности.
- **Методический подход –** основан на систематическом использовании некоторого предопределенного набора тестов или тестовых условий, таких как классификация общих или вероятных типов отказов, список важных характеристик качества или корпоративный стандарт дизайна мобильных приложений или веб-страниц.
- **Тестирование на основе процесса** (или стандарта) – подразумевает анализ, проектирование и выполнение тестов в соответствии с внешними правилами или стандартами, такими как: отраслевые стандарты, документация процесса, базис тестирования или любой другой нормативной базой, используемой в организации.
- **Направленный** (или консультативный) **подход** – определяется, прежде всего, советами, руководствами или инструкциями заинтересованных сторон, экспертов предметной области или экспертов по технологиям, которые могут находиться вне команды тестирования или организации.
- **Тестирование на основе минимизации регресса –** нацелено на проверку работоспособности существующих возможностей ПО. Эта стратегия тестирования подразумевает повторное использование существующего тестового обеспечения (особенно тестовых сценариев и тестовых данных), обширную автоматизацию регрессионных тестов и стандартные наборы тестов.
- **Реактивный подход –** тестирование в данном случае не является спланированным заранее, но зависит от тестируемого компонента, системы или событий, происходящих при выполнении тестов. Новые тесты проектируются, разрабатываются и выполняются, исходя из знаний, ранее полученных при тестировании. Исследовательское тестирование является распространенной методикой, применяемой в реактивных стратегиях.

Подходящая стратегия тестирования может быть создана путем объединения нескольких стратегий тестирования. Например, тестирование на основе рисков (аналитическая стратегия) может сочетаться с исследовательским тестированием (реактивная стратегия); они дополняют друг друга и могут обеспечить более эффективное тестирование при совместном использовании.

Стратегия тестирования дает общее описание процесса тестирования, в то время как подход к тестированию адаптирует стратегию к конкретному проекту или релизу. Подход к тестированию является отправной точкой при выборе методов тестирования, уровней и типов тестирования, а также при определении критериев входа и выхода (или критериев готовности и критериев завершения соответственно). Построение стратегии тестирования зависит от решений, принятых в отношении сложности и целей проекта, типа разрабатываемого продукта, анализа рисков продукта. Выбранный подход зависит от контекста и может учитывать такие факторы, как риски, безопасность, доступные ресурсы и навыки, технологии, характер системы (например, разработанная на заказ или готовый коммерческий продукт), цели тестирования и правила.

6.2.3 Критерии входа и выхода (критерии готовности и критерии завершения)

Для обеспечения эффективного управления тестированием и качеством программного обеспечения рекомендуется иметь критерии, которые определяют, когда начинается и

завершается каждая из работ по тестированию. Критерии входа (или критерии готовности в гибкой разработке) определяют условия, которые должны быть выполнены до начала работ. Если критерии входа не выполнены, вполне вероятно, что выполняемая задача окажется более сложной, более трудоемкой, более дорогостоящей и более рискованной. Критерии выхода (или критерии завершения в гибкой разработке) определяют, какие условия должны быть выполнены, чтобы завершить уровень тестирования или набор тестов. Критерии входа и выхода должны быть определены для каждого уровня и типа тестирования и могут отличаться в зависимости от целей тестирования.

Типичные критерии входа включают:

- Доступность тестируемых требований, пользовательских историй и/или моделей (например, при использовании стратегии тестирования на основе моделей)
- Наличие элементов тестирования, которые удовлетворяют критериям выхода для предыдущих уровней тестирования
- Доступность тестовой среды
- Наличие необходимых инструментов тестирования
- Наличие тестовых данных и других необходимых ресурсов

Типичные критерии выхода включают:

- Выполнение запланированных тестов
- Достижение определенного уровня покрытия (например, требований, пользовательских историй, критериев приемки, рисков, кода)
- Количество открытых дефектов ниже оговоренного порогового значения
- Низкая оценка количества еще не обнаруженных дефектов
- Соответствие требуемым значениям оценок надежности, производительности, практичности, безопасности и других характеристик качества

Даже если критерии завершения не выполняются, тестирование может быть сокращено из-за превышения бюджета, истечения запланированного времени и/или необходимости вывода продукта на рынок. Завершение тестирования может быть приемлемым, если заинтересованные лица со стороны проекта и бизнеса рассмотрели и приняли риск вывода продукта в промышленную эксплуатацию без дальнейшего тестирования.

6.2.4 Расписание выполнения тестов

После того, как тестовые сценарии и процедуры (в том числе автоматизированные) разработаны, их объединяют в наборы тестов. Эти наборы располагаются в соответствии с расписанием тестирования, которое задает последовательность их выполнения. Расписание должно учитывать такие факторы как: приоритет, зависимости между тестами и/или тестируемыми функциями, необходимость выполнения подтверждающих тестов и регрессионных тестов и наиболее эффективную последовательность выполнения тестов.

В идеальном случае тестовые сценарии упорядочиваются на основе их приоритетов, при этом сначала выполняются тестовые сценарии с наивысшим приоритетом. Однако эта практика может не работать, если тесты или тестируемые функции имеют зависимости. Если тестовый сценарий с более высоким приоритетом зависит от тестового сценария с более низким приоритетом, то сначала выполняется тестовый сценарий с более низким приоритетом. Аналогичным образом, если в тестовых сценариях есть зависимости, они должны быть упорядочены соответствующим образом, без учета относительных приоритетов.

Подтверждающие и регрессионные тесты тоже должны иметь приоритет, исходя из важности обратной связи об изменениях, но здесь также могут применяться зависимости.

Иногда могут использоваться разные последовательности тестов, имеющие разные уровни эффективности. В таких случаях должен быть достигнут компромисс между эффективностью выполнения тестов и соблюдением приоритетов.

6.2.5 Факторы, влияющие на затраты на тестирование

Оценка затрат на тестирование подразумевает прогнозирование объема связанной с тестированием работы, которая необходима для достижения целей тестирования проекта, релиза или итерации. Факторы, влияющие на затраты, могут включать характеристики продукта, характеристики процесса разработки, характеристики людей и результаты тестирования.

Характеристики продукта включают:

- Риски, связанные с продуктом
- Качество базиса тестирования
- Размер продукта
- Сложность предметной области продукта
- Требования к характеристикам качества (например, безопасности, надежности)
- Требуемый уровень детализации тестовой документации
- Требования к юридическому и нормативному соответствию

Характеристики процесса разработки включают:

- Стабильность и зрелость организации;
- Используемую модель разработки
- Подход к тестированию
- Используемые инструменты
- Процесс тестирования
- Сжатость сроков

Характеристики людей включают:

- Навыки и опыт, особенно в аналогичных проектах и продуктах (например, знание предметной области)
- Командная сплоченность и лидерство

Результаты тестирования включают:

- Количество и критичность выявленных дефектов
- Объем требуемых доработок

6.2.6 Методы оценки затрат на тестирование

Существует несколько методов, используемых для оценки затрат на адекватное тестирование. Наиболее популярные методы - это:

- Метод, основанный на метриках - оценка затрат, использующая метрики ранее выполненных проектов или типовые значения
- Метод экспертной оценки - оценка затрат на основе опыта владельцев задач тестирования или экспертов

Например, в гибкой разработке диаграммы сгорания являются примерами метода, основанного на метриках: трудозатраты фиксируются и отслеживаются, а затем используются для оценки скорости работы команды и определения объема работы, которую команда может выполнить в следующей итерации. Покер планирования является примером метода, основанного на экспертизе, поскольку члены команды оценивают трудозатраты на основе своего опыта (ISTQB-AT Базовый уровень. Тестировщик в сфере Гибких методологий).

В итерационных методологиях примером метода, основанного на метриках, является модель устранения дефектов: сбор информации о количестве дефектов и времени на их исправление дает базис для оценки схожих проектов в будущем. Метод «Дельфи», с другой стороны, является экспертным методом, в котором группа экспертов дает оценки, исходя из своего опыта. (ISTQB-ATM Продвинутый уровень. Тест-менеджер).

6.3 Контроль и мониторинг тестирования

Цель мониторинга тестирования заключается в сборе информации и обеспечении обратной связи о состоянии тестирования. Требуемая информация может собираться вручную или автоматически и использоваться для отслеживания прогресса тестирования, оценки выполнения критериев выхода или критериев готовности (в случае гибкой разработки), таких, как обеспечение требуемого покрытия рисков продукта, требований или критериев приемки.

Контроль тестирования представляет собой любые корректирующие действия, предпринятые на основании полученной информации или метрик. Действия могут охватывать любую активность тестирования и влиять на любую активность жизненного цикла программного обеспечения.

Примеры действий по контролю тестирования:

- Повторная приоритизация тестов при реализации риска (например, нарушения сроков поставки программного обеспечения)
- Изменение графика тестирования из-за доступности или недоступности тестовой среды или других ресурсов
- Повторная проверка выполнения критериев входа или выхода для элемента тестирования, который дорабатывался

6.3.1 Контроль и мониторинг тестирования

Метрики могут собираться во время и по завершении тестирования, чтобы оценить:

- Прогресс относительно запланированного графика и бюджета
- Текущее качество объекта тестирования
- Адекватность подхода к тестированию
- Эффективность активностей тестирования по достижению целей тестирования

Типичные метрики тестирования включают:

- Процент выполненных работ по подготовке тестовых сценариев (или процент разработанных тестовых сценариев)

- Процент выполненных работ по подготовке тестовой среды
- Метрики выполнения тестов: количество выполненных/невыполненных тестовых сценариев, количество тестовых условий или сценариев, выполненных успешно/неуспешно
- Информацию о дефектах: плотность дефектов, количество обнаруженных и исправленных дефектов, частоту отказов и результаты подтверждающих тестов
- Покрытие тестами требований, пользовательских историй, критериев приемки, рисков или кода
- Информацию о выполнении задач, распределении и использовании ресурсов, трудозатратах
- Стоимость тестирования, включая сравнение стоимости с выгодой от нахождения следующего дефекта или от выполнения следующего теста

6.3.2 Цели, содержание и аудитория отчетов о тестировании

Цель отчетности состоит в обобщении и предоставлении информации по итогам и во время тестирования. Отчет, подготовленный во время тестирования, называется отчетом о ходе тестирования; отчет, подготовленный по итогам, называется итоговым отчетом о тестировании.

Во время мониторинга и контроля тестирования руководитель тестирования регулярно публикует для заинтересованных сторон отчеты о ходе тестирования. Помимо разделов, общих для отчета о ходе тестирования и итогового отчета о тестировании, отчет о ходе тестирования может включать:

- Статус активностей тестирования и прогресс по сравнению с планом тестирования
- Факторы, препятствующие прогрессу
- Тестирование, запланированное на следующий отчетный период
- Качество объекта тестирования

Когда критерии выхода выполнены, руководитель тестирования готовит итоговый отчет о тестировании. В этом отчете приводится краткое описание выполненного тестирования (на основе последнего отчета о ходе тестирования и другой информации).

Обычно итоговые отчеты и отчеты о ходе тестирования включают:

- Резюме проведенного тестирования
- Информацию о том, что произошло во время тестирования
- Информацию об отклонениях от плана, включая отклонения в расписании, длительности выполнения или затратах
- Информацию о качестве тестирования и качестве продукта с точки зрения критериев выхода или критериев завершения
- Информацию о факторах, которые блокировали или продолжают блокировать тестирование
- Метрики дефектов, тестовых сценариев, покрытия, прогресса тестирования и использования ресурсов (например, как описано в 5.3.1)
- Информацию об остаточных рисках (см. раздел 5.5)
- Перечень тестовых артефактов, которые можно повторно использовать

Содержимое отчета о тестировании зависит от проекта, корпоративных требований и жизненного цикла разработки программного обеспечения. Например, для сложных, формальных проектов с большим числом заинтересованных лиц может потребоваться более подробная и строгая отчетность. В качестве другого примера: в гибкой разработке отчеты о ходе тестирования могут быть частью панели задач, сводки по дефектам и диаграммы сгорания, которые обсуждаются на ежедневных встречах (см. ISTQB AT Базовый уровень. Тестировщик в сфере Гибких методологий).

Помимо контекста проекта, при подготовке отчетности нужно учитывать особенности аудитории. Тип и объем информации, включаемые в отчет для технических специалистов или группы тестирования, будут отличаться от информации в отчете для менеджмента. В первом случае может быть важна подробная информация о типах дефектов и тенденциях. В последнем случае может быть уместным отчет более высокого уровня (например, статус дефектов, сгруппированных по приоритету, информация о бюджете и расписании, успешные/неуспешные/непроверенные тестовые условия).

Стандарт ИСО (ISO/IEC/IEEE 29119-3) описывает два типа отчетов о тестировании: отчет о ходе тестирования и отчет о завершении тестирования (называемый в этом документе итоговым отчетом) и содержит структуру и примеры оформления отчетов каждого типа.

6.4 Управление конфигурацией

Целью управления конфигурацией является обеспечение и поддержка целостности компонента или системы, тестового обеспечения и их взаимосвязей между собой на протяжении жизненного цикла проекта и продукта.

Для поддержки тестирования управление конфигурацией может потребовать выполнения следующих условий:

- Все элементы тестирования однозначно идентифицированы, связаны между собой, находятся под версионным контролем, все изменения в них отслеживаются
- Все элементы тестового обеспечения однозначно идентифицированы, связаны между собой и с версиями элементов тестирования, находятся под версионным контролем, все изменения отслеживаются, обеспечивая трассируемость на протяжении всего процесса тестирования
- Все документы и программные элементы идентифицированы и однозначно указаны в тестовой документации

Инфраструктура и процедуры управления конфигурацией должны быть подготовлены и выполнены на этапе планирования тестирования.

6.5 Риски и тестирование

6.5.1 Определение риска

Риск подразумевает наступление некоторого негативного события в будущем. Уровень риска можно определить через вероятность события и серьезность последствий.

6.5.2 Риски проекта и продукта

Риск продукта – это возможное несоответствие некоторого артефакта (спецификации, компонента, системы или теста и т.д.) потребностям пользователей и заинтересованных сторон.

Когда риски продукта связаны с конкретными характеристиками качества (функциональной пригодностью, надежностью, производительностью, практичностью, безопасностью, совместимостью, сопровождаемостью и переносимостью), их также называют рисками качества.

Примеры рисков продукта:

- Программное обеспечение не выполняет функции, указанные в спецификации
- Программное обеспечение не выполняет функции, ожидаемые пользователями, клиентами и/или заинтересованными сторонами
- Системная архитектура не поддерживает нефункциональные требования
- Вычисления выполняются неверно в каких-то ситуациях
- Неверная реализация в коде структуры управления циклом
- Неадекватное время отклика высоконагруженной системы
- Отзывы пользователей о продукте показывают, что их ожидания не оправдались

Риски проекта связаны с событиями, препятствующими достижению целей проекта. Примеры рисков проекта:

- Проектные проблемы:
  - Задержки поставки, выполнения задач, выполнения критериев выхода или критериев готовности
  - Неточные оценки, перераспределение средств на проекты с более высоким приоритетом или общие сокращения затрат по всей организации могут привести к неадекватному финансированию
  - Поздние изменения могут привести к существенным доработкам
- Организационные проблемы:
  - Недостаток навыков, обучения или численности персонала
  - Проблемы с персоналом могут вызвать конфликты и серьезные трудности
  - Пользователи, бизнес-пользователи или эксперты предметной области могут быть заняты другими работами
- Политические проблемы:
  - Тестировщики не могут адекватно сообщать о своих потребностях и/или результатах тестирования
  - Разработчики и/или тестировщики не могут отслеживать информацию, полученную при тестировании и рецензировании (не стремясь улучшить методы разработки и тестирования)
  - Может быть неправильное отношение или ожидания от тестирования (недооценивается важность обнаружения дефектов во время тестирования и т.д.)
- Технические проблемы:
  - Требования могут быть определены недостаточно хорошо

- Требования могут быть невыполнимыми в текущих условиях
- Тестовая среда может быть не готова вовремя
- Преобразование данных, планирование миграции и их инструментальная поддержка могут быть не готовы вовремя
- Слабые стороны процесса разработки могут влиять на согласованность или качество артефактов проекта, таких как дизайн, код, конфигурация, тестовые данные и тестовые сценарии
- Проблемы управления дефектами могут привести к накоплению дефектов и росту технического долга

- Проблемы с поставщиками:
  - Третья сторона может не предоставить необходимый продукт или услугу, или обанкротиться
  - Контрактные проблемы могут вызвать серьезные трудности для проекта

Проектные риски могут влиять как на разработку, так и на тестирование. В некоторых случаях руководители проектов несут ответственность за управление всеми проектными рисками; руководители тестирования обычно отвечают за управление рисками, связанными с тестированием проекта.

6.5.3 Тестирование, основанное на рисках, и качество продукта

Риски используются для распределения усилий во время тестирования. Они используются для принятия решения о том, где и когда начинать тестирование, и для выявления областей, требующих большего внимания. Тестирование используется с целью снижения вероятности возникновения неблагоприятного события и снижения последствий при наступлении этого события. Тестирование используется как деятельность по снижению риска и для обеспечения обратной связи, как по выявленным рискам, так и по остаточным (нерешенным) рискам.

Тестирование, основанное на рисках, обеспечивает возможность заранее снизить уровень риска продукта. Оно включает анализ рисков продукта (идентификацию рисков, оценку вероятности и последствий от их наступления). Полученная информация используется для планирования тестирования, разработки, подготовки и выполнения тестовых сценариев, а также для мониторинга и контроля тестирования. Ранний анализ рисков продукта способствует успеху проекта.

В рамках подхода, основанного на рисках, результаты анализа рисков продукта могут быть использованы для:

- Выбора методов тестирования
- Выбора уровней и типов тестирования, которые необходимо выполнить (например, тестирования безопасности, тестирования доступности)
- Определения объема тестирования
- Приоритезации тестирования с целью найти критические дефекты как можно раньше
- Выявления каких-либо дополнительных мер, помимо тестирования, которые могут снизить риски (например, обучения менее опытных проектировщиков)

Тестирование, основанное на рисках, опирается на коллективные знания заинтересованных сторон проекта и выполнение ими анализа рисков продукта. Чтобы минимизировать вероятность неуспеха продукта, активности по управлению рисками обеспечивают дисциплинированный подход к:

- Анализу (и повторной оценке на регулярной основе) того, что может пойти не так (рисков)
- Определению, какие риски важны для дальнейшей работы
- Выполнению действий по снижению этих рисков
- Разработке планов действий в чрезвычайных ситуациях, если риски станут реальными событиями

Кроме того, тестирование может выявить новые риски, помочь определить, какие риски следует смягчить, и снизить неопределенность в отношении рисков.

6.6 Управление дефектами

Поскольку одной из целей тестирования является обнаружение дефектов, необходимо регистрировать дефекты, обнаруженные во время тестирования. Способ регистрации дефектов может варьироваться в зависимости от контекста, тестируемого компонента или системы, уровня тестирования и модели жизненного цикла. Любые выявленные дефекты должны быть изучены и отслеживаться от момента обнаружения и классификации до принятия решения по ним (например, исправления дефектов и успешного подтверждающего тестирования решения, отсрочки до следующего релиза, трактовки как постоянного ограничения продукта и т. д.). Чтобы управлять дефектами до принятия решения по ним, в организации должен существовать процесс управления дефектами, который включает в себя жизненный цикл и правила классификации дефектов. Этот процесс должен быть согласован со всеми, кто участвует в управлении дефектами, включая проектировщиков, разработчиков, тестировщиков и владельцев продуктов. В некоторых организациях регистрация и отслеживание дефектов могут быть очень слабо формализованными.

Во время процесса управления дефектами некоторые из отчетов могут содержать описания ложных срабатываний, а не фактических сбоев из-за дефектов. Например, тест мог пройти неуспешно при сбое сетевого соединения или таймауте. Такое поведение не является следствием дефекта объекта тестирования, но аномалией, которую необходимо исследовать. Тестировщики должны попытаться свести к минимуму количество ложных срабатываний, принимаемых за дефекты.

Дефекты могут быть обнаружены во время кодирования, статического анализа, рецензирования, динамического тестирования или эксплуатации программного продукта. Дефекты могут сообщать о проблемах в коде или эксплуатируемых системах, в документации любого типа, включая требования, пользовательские истории и критерии приемки, документацию разработки, документацию тестирования, руководства пользователя или руководства по установке. Чтобы иметь продуктивный и эффективный процесс управления дефектами, организации могут определять стандартный набор атрибутов, правила классификации и жизненный цикл дефектов.

Типичные отчеты о дефектах имеют следующие цели:

- Предоставлять разработчикам и другим сторонам информацию о произошедших негативных событиях, чтобы они могли определить побочные эффекты, изолировать проблему с минимальными затратами на воспроизведение и исправить потенциальные дефекты по мере необходимости, или решать проблемы другими способами
- Обеспечить руководителей тестирования инструментами отслеживания качества продукта и влияния на тестирование (например, если сообщается о большом количестве

дефектов, то тестировщики будут вынуждены тратить много времени на отчетность по найденным дефектам вместо того, чтобы запускать тесты; следовательно, нужно больше подтверждающего тестирования)

- Предоставить идеи для совершенствования процессов тестирования и разработки

Сообщение о дефекте, созданное во время динамического тестирования, обычно включает:

- Идентификатор дефекта
- Заголовок и краткое описание найденного дефекта
- Дату сообщения о дефекте, информацию об авторе сообщения
- Идентификацию элемента тестирования (проверяемого элемента конфигурации) и среды
- Фазу жизненного цикла разработки, в которой обнаружен дефект
- Описание дефекта, достаточное для его воспроизведения и принятия решения, включая системные журналы, скриншоты, дампы базы данных или записи (если они созданы во время выполнения теста)
- Ожидаемые и фактические результаты
- Область или степень влияния дефекта на интересы заинтересованной стороны (критичность)
- Срочность/приоритет для исправления
- Статус дефекта (например, открыт, отложен, дубликат, ожидает исправления, ожидает проверки, повторно открыт, закрыт)
- Выводы, рекомендации и согласования
- Глобальные проблемы, например, области, которые будут затронуты исправлением дефекта
- История изменений, например, последовательность действий членов команды проекта, чтобы изолировать дефект, исправить и подтвердить исправление дефекта
- Ссылки, включая ссылку на тестовый сценарий, который обнаружил дефект

Некоторые из этих деталей могут автоматически включаться и/или настраиваться при использовании инструментов управления дефектами. Например, автоматическое присваивание идентификатора дефекта, назначение и обновление статуса дефекта на протяжении жизненного цикла и т.д. Дефекты, обнаруженные при статическом тестировании, в частности, при рецензировании, как правило, документируются по-другому, например, в примечаниях к протоколу совещания.

Пример содержимого сообщения о дефекте можно найти в стандарте ИСО (ISO/IEC/IEEE 29119-3) (где сообщения о дефектах называются сообщениями об инцидентах).

7. Инструменты тестирования – 40 мин

Ключевые слова

тестирование на основе данных, тестирование на основе ключевых слов, инструменты нагрузочного тестирования, автоматизация тестирования, инструменты управления тестами, инструменты выполнения тестов

Цели обучения для главы «Инструменты тестирования»

6.1 Классификация инструментов тестирования

FL-6.1.1 (K2) Классифицировать инструменты тестирования в соответствии с целями и выполняемыми ими действиями в процессе тестирования

FL-6.1.2 (K1) Определить преимущества и риски автоматизации тестирования

FL-6.1.3 (K1) Вспомнить об особенностях инструментов выполнения тестов и управления тестированием

6.2 Эффективное использование инструментов

FL-6.2.1 (K1) Определить ключевые принципы выбора инструмента

FL-6.2.2 (K1) Перечислить цели использования пилотного проекта для внедрения инструментов

FL-6.2.3 (K1) Определить факторы успеха для оценки, внедрения, развертывания и непрерывной поддержки инструмента тестирования в масштабах организации

7.1 Инструменты тестирования

Инструменты тестирования могут использоваться для одной и более активностей тестирования. Это могут быть:

- Инструменты, которые напрямую используются в тестировании, например, инструменты выполняющие тесты, подготавливающие тестовые данные
- Инструменты, помогающие управлять требованиями, тестовыми сценариями, тестовыми процедурами, автоматизированными тестовыми скриптами, результатами тестирования, тестовыми данными, дефектами, а также инструменты создания отчетов и мониторинга выполнения тестов
- Инструменты, используемые для исследования и оценки
- Любые инструменты, которые используется при тестировании (электронная таблица - это тоже инструмент тестирования)

7.1.1 Классификация инструментов тестирования

Инструменты, поддерживающие тестирование, могут иметь одну или несколько целей, в зависимости от контекста:

- Повысить эффективность тестирования за счет автоматизации повторяющихся действий или задач, которые требуют значительных трудозатрат при выполнении вручную (например, выполнение тестов, регрессионное тестирование)
- Повысить эффективность за счет поддержки ручного тестирования на протяжении всего процесса тестирования (см. раздел 1.4)
- Повысить качество тестирования, предоставляя более содержательное тестирование и более высокий уровень воспроизводимости дефектов
- Автоматизировать действия, которые невозможно выполнить вручную (например, масштабное тестирование производительности)
- Увеличить надежность тестирования (например, путем автоматизации сравнения большого объема данных или моделирования поведения)

Классификация инструментов может быть разной в зависимости от нескольких критериев: цели, цены, предоставляемой лицензии (например, коммерческий продукт или с открытым исходным кодом), используемой технологии. В нашей программе обучения инструменты классифицированы в соответствии с видом деятельности, которому они сопутствуют.

Одни инструменты способны поддерживать один (всегда или как правило) вид деятельности, другие могут использоваться для выполнения сразу нескольких активностей, но классифицируются они в соответствии с той активностью, с которой наиболее тесно связаны. Инструменты от одного производителя могут быть объединены в единый интегрированный пакет, особенно если были разработаны для совместного использования.

Некоторые типы инструментов могут стать помехой, влияя на результат теста. Например, фактическое время ответа приложения может изменяться из-за дополнительных операций, выполняемых инструментом нагрузочного тестирования. Последствия использования подобных инструментов называют побочным эффектом.

Другие инструменты предлагают поддержку наиболее подходящую для разработчиков (например, инструменты, используемые при выполнении компонентного и интеграционного тестирования). Такие инструменты отмечены буквой “D” в нижеследующих разделах.

Инструменты управления тестированием и тестовым окружением

Инструменты управления применяются к любым активностям тестирования на протяжении всего жизненного цикла программного продукта. Примерами инструментов, поддерживающих управление тестированием и тестовым окружением, являются:

- Инструменты управления тестированием и инструменты управления жизненным циклом приложения (ALM - application lifecycle management)
- Инструменты управления требованиями (например, трассируемость объектов тестирования)
- Инструменты управления дефектами
- Инструменты управления конфигурацией
- Инструменты непрерывной интеграции (D)

Инструменты статического тестирования

Активности и выгоды, связанные с инструментами статического тестирования, описаны в главе 3. Примерами таких инструментов являются:

- Инструменты рецензирования
- Инструменты статического анализа (D)

Инструменты проектирования и реализации тестов

На этапе разработки и реализации тестов инструменты проектирования тестов способствуют созданию поддерживаемых рабочих продуктов. Этот этап включает в себя проектирование и реализацию тестовых сценариев, процедур тестирования и тестовых данных. Примерами таких инструментов являются:

- Инструменты проектирования тестов
- Инструменты тестирования на основе модели
- Инструменты подготовки тестовых данных
- Инструменты разработки тестов через приемочное тестирование (ATDD – acceptance test driven development) и на основе поведения (BDD - behavior driven development)
- Инструменты разработки через тестирование (D)

Некоторые инструменты проектирования и реализации тестов могут поддерживать исполнение и протоколирование тестов или напрямую предоставлять необходимые данные инструментам, поддерживающим эти действия.

Инструменты выполнения тестов и протоколирования

Существует множество инструментов для поддержки и улучшения активностей по выполнению тестов и протоколированию действий. Примерами таких инструментов являются:

- Инструменты выполнения тестов (например, запуск регрессионных тестов)
- Инструменты покрытия (например, покрытие требований, покрытие кода (D))
- Тестовые обвязки (D)
- Интегрированные среды компонентного тестирования (D)

Инструменты динамического анализа и тестирования производительности

Без инструментов динамического анализа и инструментов, предназначенных для измерения производительности, вручную невозможно эффективно выполнить тестирование производительности и нагрузки. Примерами таких инструментов являются:

- Инструменты тестирования производительности
- Инструменты мониторинга
- Инструменты динамического анализа

Инструменты для специализированных нужд тестирования

В дополнение к инструментам, которые сопутствуют основным процессам тестирования, существует множество инструментов для конкретных нужд тестирования. Примерами могут быть инструменты, ориентированные на:

- Оценку качества данных
- Преобразование и перенос данных
- Тестирование практичности
- Тестирование доступности
- Тестирование локализации
- Тестирование безопасности (D)
- Тестирование переносимости (например, тестирование программного продукта на нескольких поддерживаемых платформах)

7.1.2 Преимущества и риски автоматизации тестирования

Покупка какого-либо инструмента не дает гарантии его успешного использования. Ввод в эксплуатацию нового инструмента всегда требует усилий, чтобы достичь реальной и долгосрочной выгоды. Говоря про потенциальную выгоду и возможности использования инструментов тестирования, необходимо помнить и про риски. Особенно, это касается инструментов выполнения тестирования (часто называемые инструментами автоматизации).

Преимущества использования инструментов, сопровождающих выполнение тестирования:

- Уменьшение повторяющихся действий, выполняемых вручную (например, выполнение регрессионных тестов, задачи по развертыванию/свертыванию окружения, повторный ввод одних и тех же тестовых данных, проверка на соответствие стандартам оформления кода) и за счет этого экономия времени
- Увеличение целостности и стабильности (например, последовательно связанные тестовые данные, тесты, выполняемые инструментом в одном и том же порядке и с одинаковой частотой, тесты, полученные из требований)
- Более объективная оценка (например, статические измерения, покрытие)
- Улучшение доступности информации о тестировании (например, статистика и графики процесса тестирования, доля дефектов и информация о тестировании производительности)

Риски использования инструментов тестирования:

- Нереалистичные ожидания от использования инструмента (включая функциональность и простоту использования)
- Недооценка времени, стоимости и трудозатрат, необходимых для внедрения инструмента (например, обучение, внешняя экспертиза)
- Недооценка времени и трудозатрат, необходимых для достижения значительной и постоянной выгоды от использования инструмента (включая необходимость изменений в процессе тестирования и непрерывного улучшения способов использования инструмента)
- Недооценка трудозатрат, необходимых для поддержки тестовых данных, сгенерированных инструментом
- Чрезмерная зависимость от инструмента (замена проектирования тестов или использование автоматизированного тестирования там, где уместнее использовать ручное тестирование)
- Пренебрежение контролем версий тестовых ресурсов в инструменте
- Пренебрежение взаимодействием между ключевыми инструментами тестирования - инструментами управления требованиями, управления конфигурацией, дефектами, а также пренебрежение взаимодействием между инструментами от разных производителей. Риски, связанные с уходом производителя инструмента из бизнеса, выходом инструмента из обращения, продажей инструмента другому поставщику
- Медленная реакция производителя по вопросам поддержки, обновления и исправления дефектов
- Приостановка проекта с открытым исходным кодом
- Отсутствие возможности поддерживать новую платформу или технологию
- Инструмент может не находиться в собственности поставщика (например, только курирование, обновление)

7.1.3 Особенности использования инструментов выполнения и управления тестами

Для успешной реализации тестов существует ряд факторов, которые необходимо учитывать при выборе и последующей интеграции инструментов выполнения и управления тестами

Инструменты выполнения тестов

Инструменты выполнения тестов исполняют тестовые скрипты. И зачастую, используя такие инструменты, необходимо приложить значительные усилия, чтобы достигнуть существенного результата.

Запись действий, выполняемых вручную, в тестовые скрипты выглядит заманчиво, но подходит не для всех тестов. Созданный таким образом скрипт – это линейная последовательность конкретных действий с конкретными данными. И в случае возникновения непредусмотренных событий данный скрипт окажется ненадежным. Последнее поколение инструментов, использующих технологию захвата действий пользователя, повысило применимость инструментов подобного класса. Тем не менее, сгенерированные скрипты по-прежнему требуют постоянной поддержки, поскольку интерфейс пользователя системы со временем меняется.

При тестировании на основе данных (data-driven testing approach) выбираются входные данные и ожидаемый результат, обычно сформированные в виде таблицы, и используется единый скрипт, который, считывая входные данные из таблицы, выполняется с разным набором данных.

Даже те тестировщики, которые не знакомы со скриптовыми языками, впоследствии могут создавать новые наборы тестовых данных для таких ранее определенных скриптов.

При тестировании на основе ключевых слов (keyword driven testing), единый скрипт обрабатывает ключевые слова, описывающие действия, которые нужно предпринять (так называемые слова-действия), затем по ключевым словам вызываются скрипты для обработки соответствующих тестовых данных. Тестировщики, даже будучи не знакомыми со скриптовыми языками, по ключевым словам и соответствующим данным могут подбирать тесты под тестируемое приложение. Дополнительные подробности и примеры подходов на основе данных и ключевых слов приведены в программе обучения ISTQB-TAE для инженеров по автоматизации тестирования продвинутого уровня [Fewster 1999], [Buwalda 2001]

Приведенные выше подходы нуждаются в экспертизе скриптов тестировщиком, разработчиком или специалистом по автоматизации тестов.

Независимо от используемого метода написания скрипта, в каждом случае необходимо сопоставить ожидаемый и фактический результаты либо динамически (во время выполнения теста), либо, сохранив результаты теста, выполнить сравнение после прохождения тестов.

Инструменты тестирования на основе модели позволяют интерпретировать функциональные требования в виде модели, например, в виде диаграммы действий. Эта задача обычно выполняется разработчиком системы. Такой инструмент интерпретирует модель для создания спецификаций тестовых сценариев, которые впоследствии могут быть сохранены в инструменте управления тестированием и/или могут использоваться инструментом выполнения тестов (см. также программу ISTQB-MBT Базовый уровень. Тестирование на основе моделей).

Инструменты управления тестированием

Инструменты управления тестированием зачастую должны быть интегрируемы с другими инструментами или электронными таблицами по ряду различных причин:

- для получения необходимой информации в удобном для организации формате
- для обеспечения постоянной трассируемости требований в инструменте управления требованиями
- для обеспечения связи между версиями объекта тестирования в инструменте управления конфигурацией

Особенно важно это учитывать при использовании интегрированного инструмента (например, инструмента управления жизненным циклом приложения), который включает в себя модуль управления тестированием (и, возможно, систему управления дефектами), а также другие модули (например, планирования проекта и информации о бюджете), которые используются различными группами внутри организации.

7.2 Эффективное использование инструментов

7.2.1 Ключевые принципы выбора инструментов

Ключевыми принципами выбора инструментов для использования в организации являются:

- Оценка зрелости организации, ее сильных и слабых сторон
- Определение возможностей улучшения процесса тестирования, поддерживаемых инструментами
- Понимание технологий, используемых объектом (объектами) тестирования для выбора совместимого с технологией инструмента

- Инструменты сборки и непрерывной интеграции, используемые в организации, для обеспечения совместимости и интеграции инструментов
- Оценка инструментов на соответствие предъявляемым требованиям и объективности критериев
- Наличие бесплатного пробного периода использования (если есть, то на какой срок)
- Оценка надежности производителя (включая обучение, поддержку и коммерческие аспекты) или поддержка некоммерческих инструментов (например, с открытым исходным кодом)
- Определение внутренних требований передачи знаний и опыта по использованию инструментов
- Оценка необходимости обучения с учетом имеющихся навыков тестирования (и автоматизации тестирования) тех, кто напрямую будет работать с инструментами
- Определение достоинств и недостатков различных вариантов лицензирования продуктов (например, коммерческих, с открытым исходным кодом)
- Оценка соотношения затрат и выгоды применительно к конкретному бизнес-сценарию (если необходимо)

Наконец, оценка правильности концепции, чтобы оценить, будет ли инструмент эффективен применительно к тестируемому программному продукту и текущей инфраструктуре, или определить объем изменений инфраструктуры для эффективного использования инструмента.

7.2.2 Пилотный проект для оценки инструмента

После выбора инструмента и успешного доказательства правильности концепции внедрение инструмента в организации обычно начинается с выбора пилотного проекта, целями которого являются:

- Получить глубокие знания об инструменте, поняв его слабые и сильные стороны
- Оценить, насколько инструмент соответствует существующему процессу и практикам применения, и определить, что необходимо изменить
- Выбрать пути использования, управления, хранения и поддержки инструмента и тестов (например, определить правила именования файлов и тестов, выбрать стандарт программирования, создать библиотеки, определить модульность тестовых наборов)
- Оценить, будет ли выгода оправдывать цену.
- Определить метрики, которые необходимо собирать и включать в отчет, и настроить инструмент соответствующим образом

7.2.3 Факторы успеха

Факторами успеха для оценки, внедрения, развертывания и постоянной поддержки инструмента в организации являются:

- Постепенное внедрение инструмента в масштабах организации
- Адаптация и улучшение процессов в соответствии с использованием инструмента
- Проведение обучения для тех, кто пользуется инструментом
- Определение правил использования инструмента (например, внутренних стандартов автоматизации)

- Реализация способа сбора информации о фактическом использовании инструмента
- Мониторинг использования инструмента и выгоды
- Обеспечение поддержки инструмента для команды тестирования
- Накопление извлеченного опыта от всех пользователей

Также важно обеспечить техническую и организационную интеграцию инструмента в жизненный цикл разработки программного обеспечения, который может включать отдельные организации, отвечающие за отдельные процессы, и/или сторонних поставщиков.

Об опыте и советах по использованию инструментов выполнения тестов см. [Graham 2012].

8. Ссылки

Стандарты

ISO/IEC/IEEE 29119-1 (2013) Software and systems engineering - Software testing - Part 1: Concepts and definitions

ISO/IEC/IEEE 29119-2 (2013) Software and systems engineering - Software testing - Part 2: Test processes

ISO/IEC/IEEE 29119-3 (2013) Software and systems engineering - Software testing - Part 3: Test documentation

ISO/IEC/IEEE 29119-4 (2015) Software and systems engineering - Software testing - Part 4: Test techniques

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

ISO/IEC 20246: (2017) Software and systems engineering — Work product reviews

UML 2.5, Unified Modeling Language Reference Manual, http://www.omg.org/spec/UML/2.5.1/, 2017

Документы ISTQB

ISTQB Glossary

ISTQB Foundation Level Overview 2018

ISTQB-MBT Foundation Level Model-Based Tester Extension Syllabus

ISTQB-AT Foundation Level Agile Tester Extension Syllabus

ISTQB-ATA Advanced Level Test Analyst Syllabus

ISTQB-ATM Advanced Level Test Manager Syllabus

ISTQB-SEC Advanced Level Security Tester Syllabus

ISTQB-TAE Advanced Level Test Automation Engineer Syllabus

ISTQB-ETM Expert Level Test Management Syllabus

ISTQB-EITP Expert Level Improving the Test Process Syllabus

Книги и статьи

Beizer, B. (1990*) Software Testing Techniques (2e)*, Van Nostrand Reinhold: Boston MA

Black, R. (2017) *Agile Testing Foundations*, BCS Learning & Development Ltd: Swindon UK

Black, R. (2009) *Managing the Testing Process (3e)*, John Wiley & Sons: New York NY

Buwalda, H. et al. (2001) *Integrated Test Design and Automation*, Addison Wesley: Reading MA

Copeland, L. (2004) *A Practitioner's Guide to Software Test Design*, Artech House: Norwood MA

Craig, R. and Jaskiel, S. (2002) *Systematic Software Testing*, Artech House: Norwood MA

Crispin, L. and Gregory, J. (2008) *Agile Testing*, Pearson Education: Boston MA

Fewster, M. and Graham, D. (1999) *Software Test Automation*, Addison Wesley: Harlow UK

Gilb, T. and Graham, D. (1993) *Software Inspection*, Addison Wesley: Reading MA

Graham, D. and Fewster, M. (2012) *Experiences of Test Automation*, Pearson Education: Boston MA

Gregory, J. and Crispin, L. (2015) *More Agile Testing*, Pearson Education: Boston MA

Jorgensen, P. (2014) *Software Testing, A Craftsman's Approach (4e)*, CRC Press: Boca Raton FL

Kaner, C., Bach, J. and Pettichord, B. (2002) *Lessons Learned in Software Testing*, John Wiley & Sons: New York NY

Kaner, C., Padmanabhan, S. and Hoffman, D. (2013) *The Domain Testing Workbook*, Context-Driven Press: New York NY

Kramer, A., Legeard, B. (2016) *Model-Based Testing Essentials: Guide to the ISTQB Certified ModelBased Tester: Foundation Level*, John Wiley & Sons: New York NY

Myers, G. (2011) *The Art of Software Testing*, (3e), John Wiley & Sons: New York NY

Sauer, C. (2000) "The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research," *IEEE Transactions on Software Engineering*, *Volume 26, Issue 1, pp 1-*

Shull, F., Rus, I., Basili, V. July 2000. "How Perspective-Based Reading can Improve Requirement Inspections." *IEEE Computer, Volume 33, Issue 7, pp 73-79*

van Veenendaal, E. (ed.) (2004) *The Testing Practitioner* (Chapters 8 - 10), UTN Publishers: The Netherlands

Wiegers, K. (2002) *Peer Reviews in Software*, Pearson Education: Boston MA

Weinberg, G. (2008) *Perfect Software and Other Illusions about Testing*, Dorset House: New York NY

Другие ресурсы (явно неупомянутые в этой программе обучения)

Black, R., van Veenendaal, E. and Graham, D. (2012) *Foundations of Software Testing: ISTQB Certification (3e)*, Cengage Learning: London UK

Hetzel, W. (1993) *Complete Guide to Software Testing (2e)*, QED Information Sciences: Wellesley MA

Spillner, A., Linz, T., and Schaefer, H. (2014) *Software Testing Foundations (4e)*, Rocky Nook: San Rafael CA

9. Приложение A – О происхождении программы обучения

История этого документа

Этот документ является программой обучения для получения международной квалификации первого уровня, утвержденной ISTQB (www.istqb.org).

Данный документ разрабатывался в период с 2004 по 2018 гг. членами рабочей группы, назначенными Международной Коллегией по Квалификации Тестировщиков Программного Обеспечения (ISTQB). Версия документа от 2018 года первоначально рецензировалась всеми представителями коллегии ISTQB, а затем представителями из международного сообщества тестирования программного обеспечения.

Задачи базового уровня сертификации

- Получить официальное признание тестирования как основной профессиональной специализации в разработке программного обеспечения
- Обеспечить фундамент развития карьеры тестировщика
- Дать возможность квалифицированным специалистам по тестированию получить признание работодателей, клиентов и коллег, а также повысить значимость профессии
- Содействовать внедрению целостных и правильных практик тестирования во всех дисциплинах разработки программного обеспечения
- Определить темы тестирования, которые являются актуальными и значимыми для отрасли
- Позволить производителям программного обеспечения нанимать сертифицированных специалистов, и тем самым повышать свою коммерческую привлекательность по сравнению с конкурентами, рекламируя кадровую политику в отношении специалистов по тестированию
- Обеспечить возможность специалистам по тестированию и заинтересованным в тестировании людям, получить квалификацию, признанную в мире

Цели международной квалификации

- Дать возможность сравнивать знания и навыки в области тестирования в различных странах
- Дать возможность специалистам по тестированию проще пересекать границы стран
- Обеспечить межнациональным / международным проектам одинаковое понимание вопросов тестирования
- Увеличить количество сертифицированных специалистов по тестированию во всем мире
- Усилить влияние и ценность в качестве международной инициативы

- Разработать единую международную основу понятий и знаний о тестировании посредством программы обучения и терминологии, а также повысить уровень знаний всех участников
- Продвигать тестирование как профессию в большем количестве стран
- Предоставить специалистам по тестированию возможность получить квалификацию на родном языке
- Наладить обмен знаниями и ресурсами между странами.
- Обеспечить международное признание тестировщиков и этой квалификации за счет участия множества стран

Требования для получения квалификации

Базовым критерием для сдачи экзамена Международной Коллегии по Квалификации Тестировщиков Программного Обеспечения является интерес кандидатов к тестированию программного обеспечения. Однако кандидатам также рекомендуется:

- Иметь минимальное представление о разработке программного обеспечения или его тестировании, например, опыт приемочного тестирования или разработки программного обеспечения в течение полугода.
- Пройти курс подготовки, аккредитованный одной из Коллегий, признанной ISTQB, в соответствии со стандартами ISTQB.

Происхождение и история базовой сертификации в области тестирования программного обеспечения

Независимая сертификация специалистов по тестированию началась в Великобритании Советом по Исследованию Информационных Систем (ISEB), образованным Британским Компьютерным Сообществом, когда в 1998 была создана Коллегия по тестированию ПО (www.bcs.org.uk/iseb). В 2002 году ASQF (Ассоциация специалистов в области качества ПО и профессионального образования) в Германии начала поддержку немецкой программы подготовки тестировщиков (www.asqf.de). Этот курс основывается на программах ISEB и ASQF; содержимое программы реорганизовано, обновлено и дополнено, а акцент делается на темы, которые будут иметь практическое значение для тестировщиков.

Существующая сертификация базового уровня в области тестирования ПО (например, от ISEB, ASQF или от любой коллегии, признанной ISTQB), выданная до выпуска Международного Сертификата, будет считаться эквивалентом Международного Сертификата. Сертификат остается действительным и не требует подтверждения. Дата вручения указывается на выдаваемом сертификате.

В каждой стране-участнице местные особенности контролируются признанной ISTQB национальной или региональной коллегией тестирования ПО. Обязанности национальных коллегий
определяются ISTQB и выполняются в каждой стране. Планируется включить в обязанности коллегий аккредитацию учебных заведений и проведение экзаменов.

10. Приложение Б – Цели обучения / Уровень знаний

Следующие цели обучения определены применительно к текущей программе обучения. Каждая тема в программе будет проверяться в соответствии с целями обучения, определенными для нее.

Уровень 1: Запомнить (К1)

Необходимо различать, запоминать и использовать термин или понятие.

Ключевые слова: идентифицировать, запомнить, найти, использовать, различать, узнать

Примеры:

Можно определить понятие “отказ” как:

- “Отказ в предоставлении услуги конечному пользователю или заказчику» или
- «Отклонение поведения компонента или системы от ожидаемого поведения, действия или результата»

Уровень 2: Понять (К2)

Кандидат способен указать причины или пояснить понятия, относящиеся к теме, а также резюмировать, сравнивать, классифицировать, разделять по категориям и приводить примеры.

Ключевые слова: резюмировать, обобщать, абстрагировать, классифицировать, сравнивать, изображать, сопоставлять, приводить примеры, интерпретировать, переводить, формулировать, делать выводы и заключения, разделять по категориям, строить модели

Примеры:

Пояснить причину, почему анализ и проектирование тестов должны выполняться как можно раньше:

- Исправление найденных дефектов будет стоить дешевле
- Наиболее серьезные дефекты будут найдены раньше

Найти сходства и различия между интеграционным и системным тестированием:

- Сходства: объекты тестирования для интеграционного и системного тестирования включают более чем один компонент и в обоих случаях включают нефункциональные типы тестирования
- Различия: в интеграционном тестировании упор делается на интегрируемость и взаимодействие компонентов, в системном - на всю систему в целом от начала до конца.

Уровень 3: Применить (К3)

Кандидат умеет выбирать правильное применение понятия или техники и использовать их в данном контексте.

Ключевые слова: реализовать, выполнить, использовать, следовать процедуре, применить процедуру

Примеры:

- Правильно идентифицировать граничные значения допустимых и недопустимых классов.

- Правильно выбирать тестовые сценарии из заданной диаграммы переходов, чтобы покрыть все переходы

Ссылка (К уровням целей обучения)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon: Boston MA

11. Приложение B – Описание изменений

Программа обучения сертификации базового уровня 2018 года является новой версией программы 2011 года. По этой причине подробности об изменениях глав и разделов исключены. Однако здесь приводится краткое содержание основных изменений. В дополнение, в отдельном документе содержатся все изменения, касающиеся целей обучения программы базового уровня в период с 2011 по 2018 годы

С начала 2017 года более 550 000 человек из более чем 100 стран сдавали экзамен базового уровня, и более 500 000 прошли сертификацию. Если считать, что все они прочитали программу обучения базового уровня для подготовки к экзамену, то можно считать ее наиболее читаемым документом в мире.

В этой версии все цели обучения были отредактированы так, чтобы сделать их атомарными и создать четкое соотнесение целей к содержимому раздела (и экзаменационными вопросами) и обратно, от содержимого раздела к изучаемым целям. Кроме того, используя проверенные эвристические правила и формулы, которые основаны на анализе целей обучения, рассматриваемых в каждой главе других программ обучения ISTQB, более реалистично определено время на изучение каждой главы в сравнении с версией от 2011 года.

Несмотря на то, что данная программа содержит лучшие практики и методы, выдержавшие испытание временем, мы внесли изменения в представление материала, особенно с точки зрения методов (например, Scrum и непрерывное развертывание) и технологий (например, Интернет вещей) разработки ПО. Мы обновили приведенные стандарты на более современные:

1. ISO/IEC/IEEE 29119 заменяет IEEE Standard 829.
2. ISO/IEC 25010 заменяет ISO 9126.
3. ISO/IEC 20246 заменяет IEEE 1028.

Кроме того, поскольку сфера деятельности ISTQB значительно выросла за последнее десятилетие, мы добавили дополнительные перекрестные ссылки на соответствующие материалы из других программ ISTQB, а также тщательно проверили согласование программы обучения с глоссарием ISTQB. Целью было сделать версию более легкой для чтения, понимания, изучения и перевода, а также повысить практическую пользу и соблюсти баланс между знаниями и навыками.

При необходимости детального анализа изменений, сделанных в этой версии, см. документ ISTQB Certified Tester Foundation Level Overview 2018.