Russian Software Testing
Qualifications Board

Сертифицированный тестировщик

Программа обучения Базового уровня

Версия 2011

International Software Testing Qualifications Board

История изменений

Версия	Дата	Примечания
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	1 мая 2007	Программа обучения Сертифицированный тестировщик Базового уровня Выпуск сопровождения – см. Приложение E – Замечания к выпуску 2007
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 от 25 февраля 1999 года

Оглавление

Благодарности

Перевод версии документа 2011 года выполнен Рабочей группой Базового Уровня Russian Software Testing Qualifications Board: Андрей Конушин (председатель), Александр Александров (редактор), Алексей Александров, Татьяна Смехнова, Елена Абрамова.

Благодарим команду переводчиков: Александр Бурдейный, Елена Костина, Сергей Косьяненко, Александр Смелов.

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.

Предисловие к Программе обучения

Цель данного документа

Программа обучения представляет собой основу для международной сертификации на квалификацию Базового уровня в области тестирования программного обеспечения. The International Software Testing Qualifications Board (ISTQB) распространяет ее среди Национальных коллегий для последующей аккредитации ими обучающих организаций и лиц, а также составления экзаменационных вопросов на местном языке. Обучающие организации и лица определяют подходящие методы обучения и создают программы обучения для аккредитации. Программа обучения призвана помочь кандидатам при подготовке к сертификации. Информация об истории и предпосылках к программе обучения может быть найдена в Приложении А.

Сертифицированный тестировщик ПО Базового уровня

Квалификация Базового уровня предназначена для всех, связанных с тестированием ПО, включая такие роли как тестировщики, тест-аналитики, проектировщики тестов, консультанты тестирования, руководители тестирования, тестировщики приемочных испытаний 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 готовы к более высокой квалификации в тестировании ПО.

Цели изучения/Необходимый уровень знаний

Цели изучения указаны для каждого раздела этой Программы обучения и классифицируются следующим образом:

- K1: запомнить
- K2: понять
- K3: применить
- K4: проанализировать

Детали и примеры целей изучения приведены в Приложении В.

Все термины, перечисленные в секции «Термины» сразу после заголовка главы, должны быть запомнены (K1), даже если явно не упоминаются в целях изучения.

Экзамены

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.

Формат экзамена – тест с несколькими вариантами ответов.

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.

Аккредитация

Национальная коллегия, входящая в ISTQB, может аккредитовывать обучающие организации, материалы курсов которых соответствуют этой Программе обучения. Обучающие организации 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.

Более подробные рекомендации для обучающих организаций приведены в Приложении D.

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;
- Перечня информации для изучения, включая описание и ссылки на дополнительные источники при необходимости;
- Целей изучения для каждой области знаний, описывающих результат изучения и образ мышления, который должен быть достигнут;
- Списка терминов, которые обучающиеся должны быть способны изучить и понять;
- 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.

Как организована данная Программа обучения

В Программе шесть основных глав. Заголовок верхнего уровня для каждой главы показывает верхний уровень целей изучения, которые покрыты данной главой, а также указывает время на изучение главы. Например:

2. Тестирование в жизненном цикле ПО (K2)	115 минут

Этот заголовок показывает, что глава 2 имеет цели изучения K1 (подразумевается, если указан более высокий уровень) и K2 (но не K3), и изучение материала главы должно занимать 115 минут. В каждой главе есть несколько разделов. Каждый раздел также имеет цели изучения и требуемое

количество времени. Подразделы, не имеющие указанного времени, включаются в общее время раздела.

1 Основы тестирования (K2) — 155 минут

Цели изучения основ тестирования

Цели определяют Ваши возможности после завершения работы с каждым модулем.

1.1 Почему тестирование необходимо? (K2)

Описать с примерами, как дефект в программном обеспечении может причинить вред человеку, окружающей среде или компании (K2)

LO-1.1.1 Используя примеры, объяснить, почему возникновение дефекта может нанести ущерб человеку, оборудованию или компании (K2)

LO-1.1.2 Определить различие между первопричиной дефекта и его эффектом (K2)

LO-1.1.3 Используя примеры, привести причины, почему тестирование необходимо (K2)

LO-1.1.4 Описать, почему тестирование - это часть обеспечения качества и привести примеры, как тестирование способствует повышению качества (K2)

LO-1.1.5 Используя примеры, объяснить и сравнить термины ошибка, дефект, недочет, отказ и соответствующие термины просчет и помеха (K2)

1.2 Что такое тестирование? (K2)

LO-1.2.1 Вспомнить общие цели тестирования (K1)

LO-1.2.2 Предоставить примеры, отражающие цели тестирования на различных стадиях жизненного цикла программного обеспечения (K2)

LO-1.2.3 Отличать тестирование от отладки (K2)

1.3 Семь принципов тестирования (K2)

LO-1.3.1 Объяснить семь принципов в тестировании (K2)

1.4 Основные процессы тестирование (K1)

LO-1.4.1 Вспомнить пять основных работ в тестировании и соответствующие задачи от планирования до завершения тестирования (K1)

1.5 Психология тестирования (K2)

LO-1.5.1 Вспомнить психологические факторы, которые влияют на успех тестирования (K1)

LO-1.5.2 Сравнить мышление тестировщика и разработчика (K2)

1.1 Почему тестирование необходимо (K2)	20 минут

Терминология

Помеха, дефект, ошибка, недочет, просчет, качество, риск

1.1.1 Системный контекст программного обеспечения (K1)

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 от него 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.

1.1.2 Причины дефектов в программном обеспечении (K2)

Человек может сделать ошибку (просчет), которая порождает дефект (недочет, помеху) в программном коде или документе. Если код с дефектом выполнен, то система может быть не в состоянии сделать то, что должна делать (или сделать то, что от нее не ожидают), порождая отказ. Дефекты в программном обеспечении, системах или документах могут в результате привести к отказам, но не все дефекты дают такой результат.

Дефекты встречаются, потому что люди склонны ошибаться, существует нехватка времени, сложность кода, сложность инфраструктуры, изменения технологий и /или много системных взаимодействий.

Отказы так же 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.

1.1.3 Роль тестирования в разработке программного обеспечения, сопровождении и функционировании программного обеспечения (K2)

Доскональное тестирование систем и документации может уменьшить риск возникновения проблем во время функционирования и способствует повышению качества системы программного обеспечения, если найденные дефекты исправлены прежде, чем система передана в эксплуатацию.

Тестирование программного обеспечения также может быть требованием для удовлетворения контракту или требованиям законодательства, или специализированным промышленным стандартам.

1.1.4 Тестирование и качество (K2)

Тестирование - это возможный способ оценки качества программного обеспечения в терминах найденных дефектов, как для функциональных требований, так и для нефункциональных требований и характеристик программного обеспечения (например, надежность, практичность, эффективность, сопровождаемость и переносимость). Для получения дополнительной информации о нефункциональном тестировании смотрите Главу 2; для получения дополнительной информации о характеристиках качества программного обеспечения смотрите стандарт ISO 9126 «Информационная технология. Оценка программного продукта»

Тестирование может породить уверенность в качестве программного обеспечения, если не найдены или найдено немного дефектов. Должным образом разработанный тест, который пройден успешно, уменьшает общий уровень риска в системе. Когда во время тестирования находятся ошибки, качество систем программного обеспечения повышается, если эти дефекты исправлены.

Следует извлекать уроки из предыдущих проектов. Понимая первопричины дефектов, найденных в других проектах, можно улучшить процессы, что в свою очередь должно предотвратить повторное проявление этих дефектов и, как следствие, повышение качества будущих систем. Это и есть подход к обеспечению качества.

Тестирование также является деятельностью по обеспечению качества (наравне с разработкой стандартов, обучением и анализом дефектов).

1.1.5 Когда заканчивать тестирование? (K2)

Для принятия решения о достаточном объеме тестирования, необходимо принимать во внимание уровень рисков, включая технические риски, риски безопасности и бизнес риски, а так же проектные ограничения, такие как время и бюджет. Риски обсуждаются далее в Главе 5

Тестирование должно предоставить достаточную информацию заинтересованным лицам, чтобы принять обоснованные решения о передаче программного обеспечения или системы, прошедшей тестирование, на следующий шаг разработки или передачи клиентам.

1.2 Что такое тестирование? (K2)	30 минут

Терминология

Отладка, требование, рецензирование, тестовый сценарий, тестирование, цель тестирования

Введение

Бытует мнение, что тестирование состоит только из прогона тестов, то есть выполнения самой программы. Но это только часть тестирования, и далеко не все, что в него входит.

Активности в тестировании существуют как до, так и после выполнения самих тестов. В эти активности входят планирование и управление, выбор тестовых условий, разработка и выполнение тестовых сценариев, проверка результатов, оценка критериев выхода, создание отчетов о процессе тестирования и об испытываемой системе и закрытие или завершающие действия после того, как фаза тестирования была выполнена. Тестирование также включает рецензирование документации (включая исходный код) и проведение статического анализа.

И динамическое и статическое тестирования используются для достижения аналогичных целей, предоставляя информацию, которая может способствовать улучшению, как испытываемой системы, так и процессов разработки и тестирования.

Цели тестирования:

- Обнаружение дефектов
- Повышение уверенности в уровне качества
- Предоставление информации для принятия решений
- Предотвращение дефектов

Цели процессов и действия, связанные с проектированием тестов на раннем этапе жизненного цикла программного обеспечения (например, при компонентном, интеграционном и системном тестировании), могут помочь предотвратить попадание дефектов в код. Рецензирование документов (например, требований), идентификация и разрешение проблем также помогают предотвратить появление дефектов в коде.

Разные точки зрения в тестировании преследуют разные цели. Например, в тестировании на этапе разработки (таком, как компонентное, интеграционное и системное тестирование), основная цель может заключаться в том, чтобы вызвать как можно больше отказов, чтобы дефекты в программном обеспечении были идентифицированы и могли быть исправлены. В приемочном тестировании основная цель может состоять в том, чтобы подтвердить, что система работает,

как ожидалось и повысить уверенность в том, что она удовлетворяет требованиям. В некоторых случаях основная цель тестирования может состоять в том, чтобы оценить качество программного обеспечения (без намерения исправлять дефекты) и дать информацию заинтересованным лицам о рисках выпуска системы в установленный срок. Тестирование в период сопровождения в основном заключается в проверке отсутствия новых дефектов, которые могли попасть во время разработки изменений. Во время эксплуатационного тестирования основная цель может заключаться в том, чтобы оценить системные характеристики, такие как надежность или доступность.

Стоит различать отладку и тестирование. Динамическое тестирование может выявить отказы, вызванные дефектами. Отладка – это действия разработчиков, которые находят, анализируют и устраняют причину отказа. Повторное тестирование гарантирует, что изменение действительно предотвращает отказ. Ответственность за тестирование обычно несут тестировщики, а за отладку - разработчики.

Подробнее о процессе тестирования и его активностях говорится в Разделе 1.4.

1.3 Семь принципов тестирования (K2)	35 минут

Терминология

Исчерпывающее тестирование

Принципы

Эти принципы тестирования были предложены в последние 40 лет и являются общим руководством для тестирования в целом.

Принцип 1 – Тестирование демонстрирует наличие дефектов

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, это не доказывает его корректности.

Принцип 2 – Исчерпывающее тестирование недостижимо

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

Принцип 3 – Раннее тестирование

Чтобы найти дефекты как можно раньше, активности по тестированию должны быть начаты как можно раньше в жизненном цикле разработки программного обеспечения или системы, и должны быть сфокусированы на определенных целях.

Принцип 4 – Скопление дефектов

Усилия тестирования должны быть сосредоточены пропорционально ожидаемой, а позже реальной плотности дефектов по модулям. Как правило, большая часть дефектов, обнаруженных при тестировании или повлекших за собой основное количество сбоев системы, содержится в небольшом количестве модулей.

Принцип 5 – Парадокс пестицида

Если одни и те же тесты будут прогоняться много раз, в конечном счете этот набор тестовых сценариев больше не будет находить новых дефектов. Чтобы преодолеть этот “парадокс пестицида”, тестовые сценарии должны регулярно рецензироваться и корректироваться, новые тесты должны быть разносторонними, чтобы охватить все компоненты программного обеспечения, или системы, и найти как можно больше дефектов.

Принцип 6 – Тестирование зависит от контекста

Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем сайт электронной коммерции.

Принцип 7 – Заблуждение об отсутствии ошибок.

Обнаружение и исправление дефектов не помогут, если созданная система не подходит пользователю и не удовлетворяет его ожиданиям и потребностям.

1.4 Основной процесс тестирования (K1)	35 минут

Терминология

Подтверждающее тестирование, повторное тестирование, критерии выхода, инцидент, регрессионное тестирование, базис тестирования, тестовое условие, тестовое покрытие, тестовые данные, тестовое выполнение, протокол тестирования, план тестирования, процедура тестирования, политика тестирования, набор тестов, итоговый отчет о тестировании, тестовое обеспечение

Введение

Активность тестирования, которую легче всего увидеть - это выполнение тестов. Но чтобы быть эффективным и рациональным, планы тестирования должны включать время, которое будет потрачено на планирование тестов, разработку тестовых сценариев, подготовку к выполнению и оценку результатов.

Основной процесс тестирования состоит из следующих направлений деятельности:

- Планирование и управление
- Анализ и проектирование
- Внедрение и реализация
- Оценка критериев выхода и создание отчетов
- Действия по завершению тестов

Несмотря на логическую последовательность, действия в процессе могут накладываться друг на друга или происходить одновременно. Для конкретных системы и проекта обычно требуется адаптация этих направлений деятельности.

1.4.1 Планирование и управление тестированием(K1)

Планирование тестирования – это действия, направленные на определение целей тестирования и описание задач тестирования для достижения этих целей и миссии.

Управление тестированием – это постоянное сопоставление текущего положения дел с планом и отчетность о состоянии дел, включая отклонения от плана. Это позволяет предпринять меры, необходимые для достижения миссии и целей проекта. Чтобы обеспечить управление тестированием, активности тестирования должны проверяться в течение всего проекта. Планирование тестирования учитывает данные, полученные при проверке и управлении.

Подробнее о задачах планирования и управления говорится в Главе 5 этой программы обучения.

1.4.2 Анализ и проектирование тестов (K1)

Анализ и проектирование тестов - это деятельность, во время которой общие цели тестирования материализуются в тестовые условия и тестовые сценарии.

Для анализа и проектирования тестов поставлены следующие основные задачи:

- Рецензирование базиса тестирования (такого, как требования, уровень целостности программного обеспечения [1] (уровень риска), отчеты об анализе рисков, архитектура, дизайн, технические требования к интерфейсу)
- Оценка тестируемости базиса тестирования и объектов тестирования
- Идентификация и расстановка приоритетов условий тестирования, основанных на анализе элементов тестирования, спецификации, поведении и структуры программного обеспечения
- Разработка и расстановка приоритетов тестовых сценариев высокого уровня
- Выявление необходимых данных для поддержки тестовых условий и тестовых сценариев
- Проектирование и установка тестового окружения и выявление необходимой инфраструктуры и инструментов
- Создание двунаправленной трассируемости между тестовым базисом и тестовым сценарием

1.4.3 Реализация и выполнение тестов (K1)

Реализация и выполнение тестов – это деятельность, где процедуры тестирования или автоматизированные сценарии задаются последовательностью тестовых сценариев, а также собирается любая информация, необходимая для выполнения тестов, разворачивается окружающая среда, и запускаются тесты.

Для реализации и выполнения тестов поставлены следующие основные задачи:

- Завершение, реализация и расстановка приоритетов тестовых сценариев (включая проектирование тестовых данных)
- Разработка и расстановка приоритетов процедур тестирования, создание тестовых данных и, если потребуется, подготовка тестовых обвязок и написание автоматизированных сценариев тестирования
- Создание тестовых наборов на основе процедур тестирования для эффективного выполнения тестов
- Проверка правильности настройки тестового окружения

[1] Показатель ряда системных характеристик, которому программное обеспечение соответствует или должно соответствовать (например, сложность программного обеспечения, оценка степени рисков, уровень безопасности, уровень надежности, желаемая производительность, стрессоустойчивость или стоимость), которые определены, чтобы продемонстрировать важность программного обеспечения заинтересованным лицам.

- Проверка и обновление двунаправленной трассируемости между тестовым базисом и тестовым сценарием
- Выполнение процедур тестирования либо вручную, либо используя инструменты выполнения тестов, согласно заданному плану
- Регистрация результатов выполнения тестов и запись наименований и версий объекта тестирования, тестовых инструментов и тестового обеспечения
- Сравнение фактических и ожидаемых результатов
- Отчет о несоответствиях как об инцидентах и их анализ для установки причины (например, дефект в коде, в конкретных тестовых данных, в тестовом документе, или ошибка выполнения теста)
- Повторение тестовых действий, результаты которых привели к каждому из несоответствий. Например, повторное выполнение теста, который ранее не прошел, чтобы подтвердить исправление (подтверждающее тестирование), выполнение исправленных тестов и/или повторное выполнения тестов с целью убедиться, что дефекты не появились в той области программного обеспечения, которая не изменялась, или что исправление дефекта не повлекло за собой других дефектов (регрессионное тестирование)

1.4.4 Оценка критериев выхода и отчетность (K1)

Оценка критериев выхода - это деятельность, где выполнение тестов оценивается согласно определенным целям. Она должна быть выполнена для каждого уровня тестирования (см. Раздел 2.2).

Для оценки критериев выхода поставлены следующие основные задачи:

- Сверка протокола тестирования в сравнении с критериями выхода, определенными в плане тестирования
- Анализ необходимости использования дополнительных тестов или изменения критериев выхода
- Написание итогового отчета о тестировании для заинтересованных лиц

1.4.5 Действия по завершению тестирования (K1)

Действия по завершению тестирования собирают данные о завершенных испытаниях для объединения опыта, тестового обеспечения, фактов и цифр. Действия по завершению тестирования происходят на тех этапах проекта, когда система программного обеспечения выпущена, тестирование завершено (или прервано), этап был завершен, или релиз по сопровождению был закончен.

Для действий по завершению тестирования поставлены следующие основные задачи:

- Проверка, что запланированные результаты достигнуты
- Закрытие отчетов об инцидентах или внесение изменений в записи по каждому из открытых инцидентов
- Документирование приемки системы

- Завершение и архивирование тестового обеспечения, тестового окружения и инфраструктуры тестирования для последующего использования
- Передача тестового обеспечения организации сопровождения
- Анализ полученных уроков для определения изменений, необходимых для будущих релизов и проектов
- Использование собранной информации для повышения зрелости процесса тестирования

1.5 Психология тестирования (K2)	25 минут

Терминология

Предположение об ошибках, независимость

Введение

Образ мышления, используемый при тестировании или рецензировании, отличается от того, который используется при разработке программного обеспечения. С соответствующим образом мышления разработчики в состоянии протестировать свой собственный код, но разделение этой ответственности с тестировщиками обычно делается для того, чтобы помочь сфокусировать усилия и обеспечить дополнительные выгоды, такие, как независимый взгляд обученными и профессиональными тестировщиками. Независимое тестирование можно проводить на любом уровне тестирования.

Тестировщик с определенной степенью независимости (лишенный предвзятости автора) часто более эффективен при обнаружении дефектов и отказов. Однако независимость не является заменой знаний, поэтому и разработчики могут эффективно находить дефекты в собственном коде. Ниже определены несколько уровней независимости от низкого до высокого:

- Тесты разработаны человеком, который написал тестируемую программу (низкий уровень независимости)
- Тесты разработаны другими людьми (например, из команды разработчиков)
- Тесты разработаны людьми из другой организационной группы (например, независимая группа тестирования) или специалистами-тестировщиками (например, специалистами по тестированию практичности или производительности)
- Тесты разработаны людьми из другой организации или компании (например, аутсорсинг или сертификация силами внешней организации)

Для людей и проектов характерна целевая направленность. Люди склонны корректировать свои планы согласно целям, установленным руководством и другими заинтересованными лицами, например, поиск дефектов или подтверждение того, что в программном обеспечении достигнуты цели. Поэтому важно ясно и точно определить цели тестирования.

Нахождение отказов во время тестирования может быть воспринято как критика продукта и его автора. Поэтому тестирование часто рассматривается как разрушительная деятельность, даже если она конструктивна с точки зрения управления рисками. Поиск отказов в системе требует любопытства, профессионального пессимизма, критического взгляда, внимания к деталям, хорошей коммуникации с разработчиками и опыта, на котором могут основываться предположения об ошибках.

Если сообщения об ошибках, дефектах и отказах конструктивны, то плохих отношений между тестировщиками и аналитиками, проектировщиками и разработчиками можно избежать. Это применимо и к дефектам, найденным во время рецензирования.

Тестировщикам и ведущим специалистам по тестированию необходимы хорошие коммуникационные навыки, чтобы фактическая информация о дефектах, ходе работы и рисках сообщалась конструктивно. Для авторов программного обеспечения или документации информация о дефекте может помочь усовершенствовать свои навыки. Дефекты, найденные и исправленные во время тестирования, позднее сэкономят время и деньги, уменьшат риски.

Коммуникационные проблемы могут возникнуть, в частности, если тестировщики рассматриваются только как вестники нежелательных сообщений о дефектах. Однако есть несколько способов улучшить коммуникации и отношения между тестировщиками и другими коллегами:

- Начните с сотрудничества, а не сражения – напомните каждому об общей цели улучшения качества системы
- Сообщайте результаты о продукте нейтральным способом, сфокусированном на фактах, без критики автора, например, опишите цели, отчеты о фактических инцидентах и результаты рецензирования.
- Попытайтесь понять, что другие люди чувствуют и почему они так реагируют
- Убедитесь, что другой человек понял, что вы сказали, и наоборот

1.6 Кодекс этики	10 минут

Участие в тестировании программного обеспечения позволяет людям узнать конфиденциальную и секретную информацию. Одной из причин, зачем необходим кодекс этики, является необходимость гарантировать, что не будет утечки информации. Признавая кодекс этики для инженеров Ассоциации по вычислительной технике (ACM) и Института инженеров по электротехнике и радиоэлектронике (IEEE), ISTQB заявляет о следующем кодексе этики:

ОБЩЕСТВО – Сертифицированные тестировщики программного обеспечения должны действовать согласно интересам общества

КЛИЕНТ И РАБОТОДАТЕЛЬ – Сертифицированные тестировщики программного обеспечения должны действовать согласно интересам клиента и работодателя, если они не противоречат интересам общества.

ПРОДУКТ – Сертифицированные тестировщики программного обеспечения должны быть уверены в том, что компоненты, которые они проверяют (в тестируемых продуктах или системах), соответствуют наивысшим возможным профессиональным стандартам.

ОЦЕНКИ – Сертифицированные тестировщики программного обеспечения должны поддерживать целостность и независимость своих профессиональных оценок.

УПРАВЛЕНИЕ – Сертифицированные руководители тестирования программного обеспечения и ведущие специалисты должны присоединяться и продвигать этические подходы к управлению тестированием программного обеспечения.

ПРОФЕССИЯ – Сертифицированные тестировщики программного обеспечения должны поднимать престиж и репутацию своей профессии в интересах общества.

КОЛЛЕГИ – Сертифицированные тестировщики программного обеспечения должны быть справедливыми, оказывать поддержку своим коллегам, и содействовать сотрудничеству с разработчиками программного обеспечения.

ЛИЧНАЯ ОТВЕТСТВЕННОСТЬ – Сертифицированные тестировщики программного обеспечения должны постоянно учиться навыкам своей профессии и способствовать продвижению этического подхода к своей деятельности.

Ссылки

1.1.5 Black, 2001, Kaner, 2002

1.2 Beizer, 1990, Black, 2001, Myers, 1979

1.3 Beizer, 1990, Hetzel, 1988, Myers, 1979

1.4 Hetzel, 1988

1.4.5 Black, 2001, Craig, 2002

1.5 Black, 2001, Hetzel, 1988

2 Место тестирования в жизненном цикле (ЖЦ) разработки ПО (K2) — 115 минут

Цели изучения места тестирования в жизненном цикле (ЖЦ) разработки ПО

Цели кратко описывают то, что вы сможете сделать по завершении изучения каждого раздела.

2.1 Модели разработки ПО (K2)

LO-2.1.1 Описать связи между разработкой, тестированием и результатами этих работ в ЖЦ разработки ПО, приводя примеры проектов и типов работ. (K2)

LO-2.1.2 Обосновать, что модели разработки ПО должны быть адаптированы в контексте проектов, в которых они используются, и характеристик разрабатываемых продуктов.(K1)

LO-2.1.3 Назвать характеристики качественного тестирования, которые применимы к любой модели ЖЦ (K1)

2.2 Уровни тестирования (K2)

LO-2.2.1 Сравнить различные уровни тестирования: типичные объекты тестирования, цели различных видов тестирования (например, функционального или структурного) и работы, связанные с ними, тестировщиков, типы дефектов, которые могут быть найдены (K2)

2.3 Типы тестирования(K2)

LO-2.3.1 Сравнить четыре типа тестирования (функциональное, нефункциональное. структурное и тестирование вносимых изменений). Привести примеры.(K2)

LO-2.3.2 Обосновать, что функциональное и структурное тестирование можно проводить на любом уровне тестирования. (K1)

LO-2.3.3 Идентифицировать и описать нефункциональные типы тестирования, основанные на нефункциональных требованиях (K2)

LO-2.3.4 Идентифицировать и описать типы тестирования, основанные на анализе структуры и архитектуры тестируемой системы. (K2)

LO-2.3.5 Описать цели подтверждающего и регрессионного тестирования (K2)

2.4 Тестирование в период сопровождения (K2)

LO-2.4.1 Сравнить тестирование в период сопровождения (тестирование существующей системы) с тестированием разрабатываемого приложения по типам тестирования, причинам тестирования и тестовому окружению. (K2)

LO-2.4.2 Определить индикаторы тестирования в период сопровождения (изменение, перемещение и прекращение эксплуатации) (K1)

LO-2.4.3. Описать роль регрессионного тестирования и анализа его воздействия в период сопровождении (K2)

2.1 Модели разработки ПО (K2)	20 минут

Терминология

Коробочный программный продукт(COTS), итеративно-инкрементальная модели разработки ПО, валидация, верификация, V-модель

Введение

Тестирование не является изолированным процессом, работы по тестированию связаны с другими работами по разработке ПО. Различные модели разработки ПО требуют различных подходов к тестированию.

2.1.1 V-модель (Последовательная модель разработки) (K2)

Хотя разновидностей V- модели существует много, все они используют четыре уровня тестирования, соответствующие четырем уровням разработки.

Четыре уровня, используемые в данном пособии, это:

- Компонентное (модульное) тестирование
- Интеграционное тестирование
- Системное тестирование
- Приемочное тестирование

На практике V-модель может иметь большее или меньшее количество уровней разработки и тестирования, все зависит от конкретного проекта и разрабатываемого продукта. Например, компонентное интеграционное тестирование может проводиться после компонентного, а системное интеграционное после системного.

Артефакты программной разработки (такие как бизнес-сценарии или сценарии использования, технические требования, документы по дизайну и код) создающиеся во время процесса разработки часто используются для одного и более уровней тестирования. Ссылки на оригинальные рабочие артефакты можно найти в «Интегрированной модели зрелости процессов производства ПО» (CMMI) и «Жизненном цикле программного обеспечения» (IEEE/IEC 12207). Верификация и валидация (а также ранняя разработка тестов) могут быть выполнены во время разработки ПО.

2.1.2 Итеративно-инкрементные модели разработки (K2)

Итеративно-инкрементный процесс разработки – это процесс, основывающийся на требованиях, дизайне, сборке и тестировании системы, созданной в результате серии коротких циклов разработки. Примеры: прототипирование, быстрая разработка приложений (RAD), Rational Unified Process (RUP) и гибкие (agile) методологии разработки. Система, получаемая в результате итерации, может быть протестирована на нескольких уровнях в процессе разработки каждой итерации. Доработка, добавляемая к разработанному ранее, провоцирует

расширение системы, которое также должно быть протестировано. Регрессионное тестирование становится все более и более важным от итерации к итерации. Верификация и валидация могут выполняться на каждом этапе доработки системы.

2.1.3 Тестирование в модели ЖЦ ПО (K2)

В любой модели ЖЦ ПО имеется несколько характеристик качественного тестирования:

- Каждому процессу разработки соответствует свой процесс тестирования
- Каждый уровень тестирования имеет свои цели тестирования
- Анализ и дизайн тестов для какого-либо уровня тестирования должны начинаться одновременно с соответствующей деятельностью разработчиков
- Тестировщики должны быть вовлечены в процесс просмотра и рецензирования документов, как только становятся доступными их предварительные версии.

Уровни тестирования могут быть объединены или реорганизованы в зависимости от природы проекта или архитектуры системы. Например, для интеграции коробочного продукта в какую-либо систему заказчик может выполнить интеграционное тестирование на уровне системного тестирования (например, интеграция с инфраструктурой и другими системами или развертывание системы) и приемочного тестирования (функциональное и\или нефункциональное, и пользовательское и/или эксплуатационное).

2.2 Уровни тестирования (K2)	40 минут

Терминология

Альфа тестирование, бета тестирование, компонентное тестирование, драйвер, тестирование в условиях эксплуатации, функциональные требования, интеграция, интеграционное тестирование, нефункциональные требования, тестирование надежности, заглушка, системное тестирование, тестовое окружение, уровень тестирования, разработка, управляемая тестированием, пользовательское приемочное тестирование.

Введение

Для каждого уровня тестирования может быть определено: цели, артефакты процесса разработки, на основании которых будут разработаны тестовые сценарии, объекты тестирования, типичные дефекты и отказы, которые могут быть найдены во время тестирования, требования к тестовой обвязке, инструментарий и ответственность участников.

Конфигурационные данные для тестирования системы нужно рассматривать во время планирования тестирования, если такие данные необходимы.

2.2.1 Компонентное тестирование (K2)

Базис тестирования:

- Требования к компонентам
- Детальный дизайн
- Код

Типичные объекты тестирования:

- Компоненты
- Программы
- Программы конвертации и миграции данных
- Модули БД

Компонентное тестирование (также известное как модульное) занимается поиском дефектов и верификацией функционирования программных модулей, программ, объектов, классов и т.п., которые можно протестировать изолированно. Это может быть сделано изолированно от остальной части системы, в зависимости от контекста ЖЦ разработки и системы. В процессе могут быть использованы заглушки, драйвера и эмуляторы.

Компонентное тестирование может включать как тестирование функциональности и специфичных нефункциональных характеристик, таких как поведение ресурсов (например, поиск утечки памяти) или тестирование надежности, так и структурное тестирование (например, покрытие кода). Тестовые

сценарии разрабатываются на основе артефактов процесса разработки, таких как спецификация компонентов, дизайн или модель БД.

Обычно, компонентное тестирование производится с доступом к тестируемому коду и с поддержкой рабочего окружения, такого как фреймворк модульного тестирования или утилиты отладки. На практике компонентное тестирование обычно производится разработчиками, которые пишут код. Дефекты обычно исправляются сразу после того, как становятся известны, без занесения их в базу дефектов.

Один из подходов к компонентному тестированию – составить автоматизированные тестовые сценарии до кодирования. Это называется разработкой, управляемой тестированием. Этот подход состоит из множества итераций и основывается на циклах разработки тестовых сценариев, написании и интеграции небольшого участка кода и выполнении компонентного тестирования, корректируя любые проблемы и выполняя тесты, пока не будет получен положительный результат.

2.2.2 Интеграционное тестирование (K2)

Базис тестирования:

- Проект системы
- Архитектура
- Бизнес-процессы
- Сценарии использования

Типичные объекты тестирования:

- БД подсистем
- Инфраструктура
- Интерфейсы

Конфигурация системы

- Конфигурационные данные

Интеграционное тестирование проверяет интерфейсы между компонентами, взаимодействие различных частей системы, таких как операционная системы, файловая система, аппаратное обеспечение, и интерфейсы между системами.

Интеграционное тестирование может состоять из одного или более уровней и может быть выполнено на тестовых объектах разного размера следующим образом:

1. Компонентное интеграционное тестирование проверяет взаимодействие между программными компонентами и производится после компонентного тестирования
2. Системное интеграционное тестирование проверяет взаимодействие между системами или между аппаратным обеспечением и может быть выполнено после системного тестирования. В этом случае, разработчики могут управлять только одной стороной интерфейса. Однако, это может

рассматриваться как риск. Бизнес-процессы могут включать последовательность систем; могут быть важны кроссплатформенные различия.

Чем больше объем интеграции, тем труднее становится изолировать дефекты отдельного компонента или системы, которые могут привести к увеличению рисков и требующих дополнительного времени для решения проблем.

Стратегии системного интеграционного тестирования могут основываться на архитектуре системы (такой как нисходящая или восходящая), функциональных задачах, последовательности обработки транзакций или других аспектах системы и ее компонентов. Для того, чтобы упростить процесс изоляции отказа и как можно раньше обнаруживать дефекты, интеграция должна проводиться по возрастающей, а не происходить по сценарию «большого взрыва».

Тестирование специфичных нефункциональных характеристик (например, производительности), может быть включено в интеграционное тестирование, наравне с функциональным.

На каждой стадии интеграции тестировщики концентрируют все внимание именно на интеграции как таковой. Например, если интегрируется модуль А с модулем В, они проверяют взаимодействие модулей, а не функциональность каждого из них, т.к. она должна быть проверена во время компонентного тестирования. Для тестирования могут использоваться как функциональный, так и структурный подходы.

В идеале, тестировщики должны понимать архитектуру и ее влияние на интеграционное планирование. Если интеграционное тестирование планируется до разработки компонентов или системы, эти компоненты могут разрабатываться в порядке, обеспечивающем наиболее эффективное тестирование.

2.2.3 Системное тестирование (K2)

Базис тестирования:

- Система и спецификация требований к программному обеспечению
- Сценарии использования
- Функциональная спецификация
- Отчеты об анализе степени риска

Типичные объекты тестирования:

- Руководство по эксплуатации системы
- Конфигурация системы

Конфигурационные данные

Системное тестирование сконцентрировано на поведении тестового объекта как целостной системы или продукта. Область тестирования должна быть четко определена в главном плане тестирования либо плане тестирования для конкретного уровня тестирования.

Во время системного тестирования тестовое окружение должно быть как можно ближе к предполагаемому эксплуатационному окружению системы для минимизации риска пропуска отказов, связанных с эксплуатационным окружением системы.

Системное тестирование может включать тесты, основанные на рисках или спецификациях требований, бизнес-процессах, сценариях использование системы, или других высокоуровневых текстовых описаниях или моделях поведения системы, взаимодействия с ОС и системными ресурсами.

Системное тестирование должно заниматься исследованием функциональных и нефункциональных требований к системе и качеством обрабатываемых данных. Тестировщики также должны уметь выполнять свои обязанности в случае неполных или недокументированных требований. Системное тестирование функциональных требований начинается с тестирования на основе спецификаций (тестирования методом черного ящика), для различных аспектов системы. Например, таблица решений может быть создана на основе бизнес-правил работы системы. Тестирование на основе структуры (тестирование методом белого ящика) может использоваться для оценки тщательности тестирования того или иного структурного элемента, такого как структура меню или навигация веб-страницы (см. Главу 4).

Системное тестирование чаще всего выполняет независимая тестовая команда.

2.2.4 Приемочное тестирование (K2)

Базис тестирования:

- Пользовательские требования
- Системные требования
- Сценарии использования
- Бизнес процессы
- Отчеты об анализе степени риска

Типичные объекты тестирования:

- Бизнес-процессы на полностью интегрированной системе
- Процессы эксплуатации и обслуживания
- Процедуры использования
- Форы
- Отчеты

Конфигурационные данные.

Приемочным тестированием системы чаще всего занимаются заказчики или пользователи системы, а также другие заинтересованные лица.

Основная цель приемочного тестирования – проверка работоспособности системы, частей системы или отдельных нефункциональных характеристик системы. Поиск дефектов не является главной целью приемочного тестирования.

Приемочное тестирование оценивает готовность системы к развертыванию и использованию, хотя это не обязательно самый последний уровень тестирования. Например, крупномасштабные тесты по системной интеграции можно провести именно во время приемочного тестирования системы.

Приемочное тестирование может проводиться в различные моменты ЖЦ разработки, например:

- Для коробочного продукта приемочное тестирование можно провести при установке или интеграции
- Приемочное тестирование удобства использования компонента можно провести во время компонентного тестирования
- Приемочное тестирование новой функциональности можно проводить до системного тестирования

Типичные виды приемочного тестирования:

Пользовательское приемочное тестирование

Обычно проверяет готовность системы для использования в бизнесе.

Эксплуатационное (приемочное) тестирование

Приемочное тестирование, проводимое системными администраторами, включает:

- Тестирование резервного копирования \ восстановления
- Восстановление после сбоев
- Управление пользователями
- Задачи сопровождения
- Задачи загрузки и миграции данных
- Периодическая проверка уязвимостей системы

Контрактное и правовое приемочное тестирование

Контрактное приемочное тестирование выполняется для проверки требований, предъявляемых контрактом в к разрабатываемому ПО. Критерий приема должен быть определен непосредственно в контракте. Приемочное тестирование на соответствие стандартам выполняется для проверки соответствия стандартам государственным, юридическим или стандартам безопасности.

Альфа и бета тестирование (или тестирование в условиях эксплуатации)

Разработчики рыночного, или коробочного, ПО часто хотят получить отзывы от потенциальных или существующих заказчиков до того, как начнется продажа продукта. Альфа тестирование выполняется организацией, разрабатывающей продукт, но не группой разработчиков. Бета тестирование, или тестирование в условиях эксплуатации, выполняется покупателями или потенциальными заказчиками на их собственных мощностях.

В организации могут использоваться и другие термины приемочного тестирования, такие как производственное приемочное тестирование и стороннее приемочное тестирование для систем, которые проверяются до и после установки

на стороне заказчика.

2.3 Типы тестирования (K2)	40 минут

Терминология

Тестирование методом черного ящика, покрытие кода, функциональное тестирование, тестирование взаимодействия, нагрузочное тестирование, тестирование восстановления, тестирование производительности, тестирование переносимости, тестирование надежности, тестирование безопасности, стресс тестирование, структурное тестирование, тестирование удобства использования, тестирование методом белого ящика.

Введение

Группы активностей тестирования могут быть направлены на проверку работоспособности системы (или части системы), принимая за основу различные цели и причины для тестирования.

Типы тестирования определяются целями тестирования, которые могут быть следующими:

- Функция, выполняемая программой
- Нефункциональная характеристика качества, такая как надежность или удобство использования
- Структура или архитектура программы или системы

Подтверждение изменений, т.е. подтверждение, что дефект был исправлен (подтверждающее тестирование) и поиск непреднамеренных изменений (регрессионное тестирование)

Модель программного обеспечения может быть разработана и\или использована во время структурного (например, модель потока управления или модель структуры меню), нефункционального тестирования например, модель нагрузки, модель удобства использования, модель угроз безопасности) и функционального тестирования (например, модель бизнес-процессов, модель переходов состояний или спецификация не естественном языке).

2.3.1 Тестирование функций (Функциональное тестирование) (K2)

Функции, которые выполняет система, подсистема или компонент, могут быть описаны в таких артефактах процесса разработки как спецификация требований, сценарии использования системы или функциональная спецификация, либо могут быть недокументированны. Эти функции описывают, «что» эта система делает.

Функциональные тесты разрабатываются на основе функций и возможностей системы (описанных в документах или понятных тестировщикам) и их взаимодействия со специфичными системами и могут быть выполнены на всех уровнях тестирования (например, тесты для компонентов могут основываться на спецификациях компонентов).

Методы разработки тестов на основе спецификаций используются для извлечения информации о тестовых условиях и тестовых сценариях из функциональности программы или системы (см. Главу 4). Функциональное тестирование рассматривает внешнее поведение программного обеспечения (тестирование методом черного ящика).

Один из типов функционального тестирования, тестирование безопасности, исследует функции (например, брандмауэр) касающиеся обнаружения угроз, таких как вирусы, поступающих извне. Другой тип функционального тестирования, тестирование возможности взаимодействия, оценивает способность программного продукта взаимодействовать с одним или более указанными компонентами или системами.

2.3.2 Тестирование нефункциональных характеристик (Нефункциональное тестирование) (K2)

Нефункциональное тестирование включает, но не ограничивается, нагрузочное тестирование, тестирование производительности, стресс-тестирование, тестирование удобства использования, тестирование восстановления, тестирование надежности и тестирование переносимости. Это тестирование того, «как» система работает.

Нефункциональное тестирование может выполняться на всех уровнях тестирования. Термин нефункциональное тестирование описывает тесты, необходимые для оценки характеристик систем и программ, которые могут быть количественно измерены, такие как время отклика при тестировании производительности. Эти тесты могут ссылаться на модели качества, такие как «Разработка программного обеспечения – Качество программного продукта» (ISO 9126). Нефункциональное тестирование рассматривает внешнее поведение программного обеспечения и в большинстве случаев использует разработку тестов методом черного ящика.

2.3.3 Тестирование структуры/архитектуры программного обеспечения (Структурное тестирование) (K2)

Структурное тестирование (тестирование методом белого ящика) может выполняться на всех уровнях тестирования. Структурные методы тестирования лучше всего использовать после методов разработки тестов на основе спецификации, чтобы измерить тщательность тестирования, используя измерения покрытия структуры программы.

Покрытие – это часть структуры программы, которая была охвачена тестированием, выраженная в процентах. Если покрытие не равно 100%, то необходимо разрабатывать дополнительные тесты для покрытия пропущенных участков программы. Методики покрытия описаны в главе 4.

На всех уровнях тестирования, особенно в компонентном и компонентном интеграционном тестировании, могут использоваться инструментальные средства для измерения покрытия кода. Структурное тестирование может основываться на архитектуре системы, такой как иерархия вызовов.

Методы структурного тестирования могут быть применены на системном, системном интеграционном и приемочном уровнях (например, применены к бизнес-моделям или структурам меню).

2.3.4 Тестирование изменений: подтверждающее и регрессионное тестирование (K2)

После того, как дефект обнаружен и исправлен, программу необходимо перепроверить, чтобы убедиться, что исходный дефект успешно устранен. Это называется подтверждением. Отладка (локализация и исправление дефекта) относится к процессу разработки, а не тестирования.

Регрессионное тестирование – это повторное тестирование уже протестированных программ после внесения в них изменений, чтобы обнаружить дефекты, внесенные или пропущенные в результате этих действий. Эти дефекты могут быть как в проверяемом компоненте, так и в связанном или несвязанным с ним. Регрессионное тестирование выполняется, когда в программное обеспечение или его окружение вносятся изменения. Глубина регрессионного тестирования оценивается риском пропуска дефектов в программном обеспечении, которое работало ранее.

Тесты должны быть повторяемыми, если они должны использоваться для подтверждающего или регрессионного тестирования.

Регрессионное тестирование может выполняться на всех уровнях тестирования и включает функциональное, нефункциональное и структурное тестирование. Регрессионные наборы тестов запускаются множество раз и меняются медленно, поэтому регрессионное тестирование является хорошим кандидатом на автоматизацию.

2.4 Тестирование в период сопровождения (K2)	14 минут

Терминология

Анализ влияния, тестирование в период сопровождения.

Введение

После установки система программного обеспечения обычно находится в эксплуатации в течение многих лет. В это время сама система, ее конфигурация или среда исполнения часто изменяются или расширяются. Ранее планирование релизов крайне важно для успешного тестирования в период сопровождения. При этом необходимо отличать запланированные выпуски и срочные исправления. Тестирование в период сопровождения выполняется на текущей ОС и может быть вызвано модификацией, переносом или прекращение эксплуатации данной системы.

Модификация включает в себя запланированные изменения (например, в рамках релиза), корректирующие и срочные изменения, изменения в окружении, такие как запланированные модификации ОС или СУБД, плановые модификации коробочных продуктов, или патчи для исправления недавно выявленных слабых мест в ОС.

Тестирование в период сопровождения при переносе (например, между платформами) должно включать как эксплуатационные тесты нового окружения, так и изменения в самой программе. Тестирование переноса также необходимо, когда данные из какой-либо системы переносятся в сопровождаемую систему.

Тестирование в период сопровождения для прекращения эксплуатации системы может включать в себя тестирование переноса данных или архивации, если присутствуют данные за большие периоды времени.

В дополнение к тестированию изменений, тестирование периода сопровождения включает регрессионное тестирование тех частей программы, которые не подвергались изменениям. Граница области, которая будет включена в тестирование, определяются риском изменений и отношением размера существующей системы к размеру внесенных изменений. В зависимости от вносимых изменений, тестирование в период сопровождения может проводиться на любом из уровней или видов тестирования. Определение, каким образом внесенные изменения могут повлиять на систему, называется анализом влияния и используется при определении объема регрессионных тестов.

Тестирование в период сопровождения затруднено, если технические требования являются устаревшими или отсутствуют вообще, либо тестировщики не обладают достаточными знаниями предметной области.

Ссылки

2.1.3 CMMI, Craig, 2002, Hetzel, 1988, IEEE 12207

2.2 Hetzel, 1988

2.2.4 Copeland, 2004, Myers, 1979

2.3.1 Beizer, 1990, Black, 2001, Copeland, 2004

2.3.2 Black, 2001, ISO 9126

2.3.3 Beizer, 1990, Copeland, 2004, Hetzel, 1988

2.3.4 Hetzel, 1988, IEEE STD 829-1998

2.4 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE STD 829-1998

3 Статические методы (K2) — 60 минут

Цели изучения Статических методов тестирования

Цели определяют, что нужно знать для завершения каждого модуля.

3.1 Статические методы и Процесс тестирования (K2)

LO-3.1.1 Объяснять какие продукты ПО могут быть проверены с помощью различных статических методов (K1)

LO-3.1.2 Охарактеризовать важность и значение статических методов для оценки качества программных продуктов (K2)

LO-3.1.3 Объяснить разницу между статическими и динамическими методами, рассматривая цели, типы дефектов, которые должны быть обнаружены, а так же роль этих методов в жизненном цикле ПО (K2)

3.2 Процесс анализа (K2)

LO-3.2.1 Вспомнить действия, роли и обязанности при типичном формальном рецензировании (K1)

LO-3.2.2 Объяснить отличия между различными типами анализа: неформальным рецензированием, техническим анализом, сквозным контролем и инспекцией (K2)

LO-3.2.3 Объяснить факторы успешного выполнения рецензирования (K2)

3.3 Статический анализ с помощью инструментальных средств (K2)

LO-3.3.1 Вспомнить типичные дефекты и ошибки, которые могут быть обнаружены при статическом анализе, и сравнить с рецензированием и динамическим тестированием (K1)

LO-3.3.2 Охарактеризовать на примерах типичные преимущества статического анализа (K2)

LO-3.3.3 Перечислить типичные дефекты в коде и дизайне, которые могут быть определены с помощью средств статического анализа (K1)

3.1 Статические методы и процесс тестирования (К2) — 15 минут

Терминология

Динамическое тестирование, статическое тестирование.

Введение

В отличие от динамического тестирования, которое требует запуска ПО, при статическом тестировании код или проектная документация исследуется вручную (рецензирование) или с помощью автоматизированных средств анализа (статический анализ) без исполнения кода.

Рецензирование - вид тестирования ПО (включая код), который может проводиться перед динамическим тестированием. Исправление дефектов, обнаруженных во время рецензирования на ранних этапах жизненного цикла ПО (например, дефектов, найденных в требованиях), часто обходится значительно дешевле по сравнению с дефектами, найденными во время выполнения тестов и исполнения кода.

Рецензирование может проводиться вручную или с помощью специальных программных средств. Главной составляющей ручного процесса является исследование и комментирование программного продукта. Рецензирование может проводиться для любого продукта, связанного с разработкой ПО, включая спецификации требований и дизайна, код, планы тестирования, спецификации тестирования, тестовые сценарии, руководства пользователя, а также веб-страницы.

Преимущества рецензирования: раннее обнаружение и исправление дефектов, улучшение продуктивности разработки, уменьшение времени разработки, уменьшение времени и стоимости тестирования, сокращение стоимости жизненного цикла, уменьшение числа дефектов и улучшение коммуникаций. Во время рецензирования могут быть найдены упущения, например, в требованиях, которые маловероятно найти во время динамического тестирования.

У рецензирования, статического анализа и динамического тестирования общая цель – обнаружение дефектов. Методы дополняют друг друга, так как с разной эффективностью находят различные типы дефектов. В отличие от динамического тестирования статические методы находят причины сбоя (дефекты), а не сами отказы.

Типичные дефекты, которые легче найти при рецензировании, чем при динамическом тестировании: отклонения от стандартов, дефекты в требованиях или дизайне, недостаточная пригодность к сопровождению и некорректные спецификации интерфейса.

3.2 Процесс рецензирования (K2)	25 минут

Терминология

Критерий входа, формальнаое рецензирование, неформальное рецензирование, инспекция, метрика, модератор, равноправный анализ, эксперт, секретарь, технический анализ, сквозной контроль.

Введение

Типы рецензирования варьируются от неформальных, когда нет письменных инструкций для экспертов, до формальных, которые предполагают участие команды, документирование результатов рецензирования и процесса ее проведения. Формальность процесса рецензирования связана с такими факторами, как зрелость процесса разработки, любые юридические или процессные требования или необходимость аудита.

Стиль проведения рецензирования зависит от согласованных целей (например, нахождение дефектов, достижение понимания, обучение тестировщиков или новых членов команды, или обсуждение и принятие согласованного решения).

3.2.1 Действия (шаги) формального рецензирования (K1)

Типичное формальное рецензирование включает следующие действия:

1. Планирование:
   - определение критерия рецензирования,
   - выбор участников,
   - распределение ролей
   - Определение критериев входа и выхода для более формальных типов рецензирования (например, инспекций)
   - Отбор частей документации для рецензирования
   - Проверка критерия входа (для более формальных типов рецензирования)
2. Старт:
   - Распределение документов
   - Объяснение целей, процесса участникам
3. Индивидуальная подготовка:
   - Подготовка к экспертному собранию - анализ документов
   - Конспектирование потенциальных дефектов, вопросов и комментариев
4. Изучение/оценка/запись результатов (встреча экспертов)

- Обсуждение или формирование списка, с документированием результатов или времени (для более формальных типов рецензирования)
- Конспектирование дефектов, создание рекомендаций по управлению дефектами, принятие решений о дефектах.
- Изучение/оценка и запись во время любых реальных встреч или отслеживание виртуального группового общения

5. Повторная обработка:
   - Исправление найденных дефектов (обычно исполняется автором)
   - Запись обновленного статуса дефектов (в формальных рецензированиях)
6. Отслеживание:
   - Проверка того, что дефекты были назначены
   - Сбор метрик
   - Проверка критерия выхода (для более формальных рецензирований)

3.2.2 Роли и Обязанности (K1)

Обычно формальное рецензирование включает следующие роли:

- Менеджер: принимает решение о проведении рецензирования, выделяет время в графике проекта и определяет, были ли достигнуты цели рецензирования
- Модератор: руководит проведением рецензирования документа или набора документов, включая планирование рецензирования, проведение встречи и отслеживание. При необходимости, модератор становится посредником между различными точками зрения и часто от него зависит успешное завершение рецензирования.
- Автор: автор или главный ответственный за документ(ы) для рецензирования
- Эксперты: люди со специальным техническим или бизнес опытом и знаниями (часто называются проверяющими или инспекторами), которые после необходимой подготовки, определяют и описывают проблемы и вопросы, найденные в проверяемом ПО. Экспертов выбирают таким образом, чтобы они представляли различные точки зрения и роли в процессе рецензирования. Эксперты должны принимать участие во всех встречах экспертов.
- Секретарь: документирует все проблемы и открытые вопросы, которые были определены во время экспертной встречи

Рецензирования могут быть намного эффективнее при наличии различных точек зрения и использовании контрольных списков. Например, контрольный список, основанный на различных точках зрения, а именно пользователя, специалиста службы поддержки, тестировщика или оператора, или контрольный список,

состоящий из типичных проблем, могут помочь выявить проблемы, которые не были найдены ранее.

3.2.3 Типы рецензирований (K2)

Отдельный программный продукт или связанная с ним работа могут объектом более чем для одного рецензирования. Если используются различные типы рецензирования, то порядок проведения может варьироваться. Например, неформальное рецензирование может быть проведено перед техническим анализом или инспекция спецификаций требований может проводиться перед сквозным контролем с заказчиками.

Основное характеристики, опции и цели основных типов рецензирования:

Неформальное рецензирование

- Отсутствие формального процесса
- Может принимать форму парного программирования или рецензирования дизайна или кода техническим руководителем
- Результаты могут быть документированы
- Эффективность зависит от экспертов
- Главные цели: результат при минимуме затрат

Сквозной контроль

- Встреча проводится автором
- Может быть в форме сценариев, сухих прогонов, участия членов команды
- Не ограниченные по времени сессии
    - Необязательно - подготовка экспертов перед встречей,
    - Необязательно - подготовка отчета по рецензированию с записью найденных проблем вопросов
- Наличие секретаря (не автора) необязательно
- На практике могут варьироваться от неформальных до строго формальных
- Главные цели: обучение, достижение понимания, поиск дефектов

Технический анализ

- Документированный процесс определения и нахождения дефектов, который включает участников команды, технических экспертов и, возможно, руководство
- Может проводиться как равноправный анализ без участия руководства
- В идеале проводится обученным модератором (не автором)
- Предварительная подготовка экспертов
- Использование контрольных списков необязательно

- Подготовка отчета о рецензировании, который включает список найденных проблем и вопросов, заключение, соответствует ли программный продукт требованиям, а также необходимые рекомендации
- На практике могут варьироваться от сильно до строго формальных
- Главные цели: обсуждение, принятие решений, оценка альтернатив, нахождение дефектов, решение технических проблем и проверка соответствия требованиям, планам, нормам и стандартам

Инспекция

- Проводится обученным модератором
- Обычно сопровождается равноправным исследованием
- Роли определены
- Включает сбор метрик
- Формальный процесс основан на правилах и контрольных списках
- Определены критерии входы и выхода для приемки программного продукта
- Предварительная подготовка перед встречей
- Инспекционный отчет включает список найденных проблем и вопросов
- Формальный процесс отслеживания
    - Необязательно - процесс улучшения компонентов
- Наличие рецензента необязательно
- Главная цель: поиск дефектов

Сквозной контроль, технический анализ или инспекция могут проводиться внутри группы профессионалов, например, коллег одного организационного уровня. Такой тип рецензирования называется равноправным анализом.

3.2.4 Факторы успешного проведения (K2)

Факторы успешного проведения рецензирования:

- Каждое рецензирование должно иметь четкие, определенные заранее цели
- Вовлечение правильных людей в соответствии с целями рецензирования
- Тестировщики - ценные эксперты, которые вносят свой вклад в рецензирование, а также изучают продукт, что позволяет создавать тесты раньше
- Найденные дефекты ожидаемы и выражены объективно
- Необходимо учитывать человеческий фактор и психологические аспекты (например, создание положительного опыта для автора)
- Рецензирование должно проводиться в атмосфере доверия, результаты не должны использоваться для оценки участников

- Должны применяться те методы рецензирования, которые подходят для достижения поставленных целей, соответствуют типу и уровню программного продукта и экспертов
- Использование контрольных списков или ролей, если необходимо увеличение эффективности обнаружения дефектов
- Обучение методам рецензирования, особенно для более формальных типов рецензирования, таких как инспекция
- Руководство поддерживает хороший процесс рецензирования (например, выделяет соответствующее время в графиках разработки проекта)
- Акцент на изучении и улучшении процесса

3.3 Статический анализ с помощью инструментальных средств (K2) — 20 минут

Терминология

Компилятор, коэффициент сложности, поток управления, поток данных, статический анализ

Введение

Цель статического тестирования - нахождение дефектов в коде или моделях ПО. Фактически статический анализ – это исследование ПО с помощью специального инструмента без его запуска, при динамическом тестировании ПО требуется запуск кода. Статический анализ выявляет дефекты, которые сложно найти при динамическом тестировании. Так же, как и рецензирования, статический анализ больше находит дефекты, чем сбои. Инструментальные средства статического анализа анализируют код программы (например, потоки управления и поток данных), а так же сгенерированный код, например, HTML или XML.

Преимущества статического анализа:

- Раннее обнаружение дефектов до исполнения тестов
- Раннее предупреждение о подозрительных аспектах в коде или дизайне с помощью вычисления метрик, таких как коэффициент сложности
- Определение дефектов, которые сложно обнаружить с помощью динамического тестирования
- Определение зависимостей и нарушений целостности в моделях ПО, например ссылок
- Улучшение пригодности к сопровождению кода и дизайна
- Предотвращение дефектов путем усвоения уроков, полученных во время разработки

Типичные дефекты, которые могут быть найдены при статическом анализе:

- Обращение к переменной, которой не присвоено значение
- Несоответствие интерфейсов между модулями и компонентами
- Переменные, которые не используются или некорректно объявлены
- Невыполняемые ветки кода
- Пропущенная или неверная логика (например, бесконечные циклы)
- Излишне сложные конструкции
- Отклонение от стандартов программирования
- Уязвимость в безопасности
- Нарушение синтаксиса в коде или моделях ПО

Инструментальные средства статического анализа используются разработчиками (проверка предопределенных правил или стандартов программирования) до или во время компонентного или интеграционного тестирования или при добавлении кода в инструменте управления конфигурацией, а также дизайнерами во время моделирования ПО. Инструменты статического анализа могут выдавать большое количество предупреждений, которые необходимы для более эффективного использования инструмента.

Компиляторы могут иметь встроенную поддержку для статического анализа, включая возможность вычисления метрик.

Ссылки:

3.2 IEEE 1028

3.2.2 Gilb, 1993, van Veenendaal, 2004

3.2.4 Gilb, 1993, IEEE 1028

3.3 Van Veenendaal, 2004

4 Методы проектирования тестов (K3) — 285 минут

Цели изучения методов проектирования тестов:

Цели определяют Ваши возможности после завершения работы с каждым модулем.

4.1 Процесс разработки тестов (K2)

LO-4.1.1 Различия между спецификациями проектирования тестов, спецификациями тестовых сценариев и спецификациями процедур тестирования. (K2)

LO-4.1.2 Сравнение терминов тестовое условие, тестовый сценарий и процедура тестирования. (K2)

LO-4.1.3 Оценка качества тестовых сценариев. Могут ли они:

- Показать четкую трассируемость к требованиям
- Содержать ожидаемый результат. (K2)

LO-4.1.4 Перевести тестовые сценарии в хорошо структурированную спецификацию процедуры тестирования на уровне детализации, соответствующей знаниям тестировщиков

4.2 Категории методов проектирования тестов (K2)

LO-4.2.1 Привести причины, по которым полезными являются как методы разработки тестов на основе спецификаций (методы черного ящика), так и методы, основанные на структуре (методы белого ящика), а также перечислить типовые приемы для каждого из этих методов. (K1)

LO-4.2.2 Объяснить характеристики и различия между методом разработки тестов на основе спецификаций, методом, основанным на структуре и методом на основе опыта. (K2)

4.3 Метод разработки тестов на основе спецификаций, или метод черного ящика (K3)

LO-4.3.1 Создать тестовые сценарии для приведенных моделей программного обеспечения при помощи следующих методов проектирования тестов: (K3)

- эквивалентное разбиение;
- анализ граничных значений;
- тестирование таблицы решений;
- тестирование таблицы переходов.

LO-4.3.2 Объяснить назначение каждого из четырех методов, какие уровни и типы тестирования могут использовать данные методы, и каким образом может быть измерено покрытие при использовании данных методов. (K2)

LO-4.3.3 Понять принцип тестирования по сценариям использования и его выгоды. (K2)

4.4 Методы, основанные на структуре, или методы белого ящика (K3)

LO-4.4.1 Описать принцип и важность покрытия кода. (K2)

LO-4.4.2 Объяснить принципы покрытия операторов и альтернатив и понять, что данные принципы также могут быть использованы в уровнях тестирования, отличных от компонентного тестирования (например, в тестировании бизнес-процессов на системном уровне). (K2)

LO-4.4.3 Написать тестовые сценарии для приведенных потоков управления с использованием следующих методов проектирования тестов:

- тестирование операторов;
- тестирование альтернатив. (K3)

LO-4.4.4 Оценить покрытие операторов и альтернатив в отношении критериев выхода (K3)

4.5 Методы создания тестов на основе опыта (K2)

LO-4.5.1 Вспомнить мотивы для создания тестовых сценариев, основанных на интуиции, опыте и знаниях об обычных дефектах. (K1)

LO-4.5.2 Сравнить методы создания тестов на основе опыта с методами разработки тестов на основе спецификаций. (K2)

4.6 Выбор методов тестирования. (K2)

LO-4.6.1 Классифицировать методы проектирования тестов по их соответствию данному контексту, исходя из базиса тестирования, соответствующих моделей и характеристик программного обеспечения. (K2)

4.1 Процесс разработки тестов (K2)	15 минут

Терминология

спецификация тестовых сценариев, проектирование теста, расписание выполнения тестов, спецификация процедуры тестирования, автоматизированный сценарий тестирования, трассируемость.

Вводная информация.

Процесс, описанный в этом разделе, может быть выполнен различными способами - от крайне неформальных с минимальным документированием или полным отсутствием документации, до крайне формальных, как описано ниже. Уровень формализации зависит от контекста тестирования, включающего в себя организацию, зрелость процессов тестирования и разработки, ограничений по времени и квалификации вовлеченных сотрудников.

Во время анализа тестирования базис тестирования анализируется с целью определить, что будет тестироваться, то есть определяются тестовые условия. Тестовые условия определяются как сущности или события, которые будут проверяться одним или несколькими тестовыми сценариями (например, функция, транзакция, характеристика качества или структурный элемент).

Установление трассируемости от тестовых условий назад к спецификациям и требованиям позволяет как определить последствия при изменяющихся требованиях, так и установить покрытие требований набором тестов. При анализе тестирования реализуется детализированный подход к тестированию с целью выбора методов проектирования тестов, основываясь, в том числе, на выявленных рисках (см. Главу 5).

Во время проектирования тестов определяются и специфицируются тестовые сценарии и тестовые данные.

Тестовый сценарий состоит из набора входных значений, предусловий выполнения, ожидаемых результатов и постусловий, определяемых для покрытия определенных тестовых условий (или тестового условия) или целей (цели) тестирования. Содержание спецификаций проектирования тестов (включая тестовые условия) и спецификаций тестовых сценариев описывается в стандарте «Документация при тестировании программ» (IEEE STD 829-1998)

Ожидаемые результаты должны создаваться как часть спецификаций тестовых сценариев и включать в себя выходные данные, изменения в данных и состояниях, и любые иные последствия теста. Если ожидаемые результаты не были определены, правдоподобные, но ошибочные результаты могут быть приняты за корректные.

В идеальных условиях ожидаемые результаты должны быть определены до момента выполнения теста.

Во время реализации теста тестовые сценарии разрабатываются, реализуются, получают приоритеты и формируют спецификацию процедуры тестирования

(IEEE STD 829-1998). Процедура тестирования (или ручной сценарий тестирования) описывает последовательность действий для выполнения теста. Если тесты запускаются с использованием инструмента выполнения тестов, последовательность действий описывается автоматизированным тестовым сценарием.

Различные процедуры тестирования и автоматические тестовые сценарии собираются в расписание выполнения тестов, определяющее, в какой очередности, когда и кем эти тестовые процедуры и сценарии должны быть выполнены.

Расписание выполнения тестов должно учитывать такие факторы, как регрессионные тесты, приоритеты и технические и логические зависимости.

4.2 Категории методов проектирования тестов (К2)	15 минут

Терминология

Разработка тестов методом черного ящика, разработка тестов методом белого ящика, метод создания тестов на основе опыта, метод разработки тестов на основе спецификации, структурный метод разработки тестов.

Введение

Целью метода проектирования тестов является определение тестовых условий и тестовых сценариев. Классическим является разделение методов тестирования на методы черного и белого ящиков. Методы черного ящика (включающие в себя методы разработки тестов на основе спецификаций и на основе опыта) – это способ определить и выбрать тестовые условия или сценарии для компонента или системы (как функциональные, так и не функциональные), на основе анализа базиса тестирования и опыта разработчиков, тестировщиков и пользователей, без отсылки к внутренней структуре компонента или системы. Методы белого ящика (также называемые структурными, или основанными на структуре) основываются на анализе структуры компонента или системы. Оба этих метода могут быть объединены с методом создания тестов на основе опыта, чтобы использовать опыт разработчиков, тестировщиков и пользователей для определения того, что должно быть протестировано. Некоторые методы могут быть однозначно отнесены к определенной категории, другие же сочетают в себе несколько категорий.

Данная программа относит подходы, основанные на спецификациях или опыте к методам черного ящика, и подходы, основанные на структуре – к методам белого ящика. Также рассматриваются методы создания тестов на основе опыта.

Общие признаки подходов, основанных на спецификациях:

- Для описания задач, которые должны быть решены, программных продуктов или их компонентов, используются модели - формальные или неформальные.
- Из этих моделей систематически выводятся тестовые сценарии.

Общие признаки подходов, основанных на структуре:

- тестовые сценарии выводятся на основе информации о том, как спроектировано программное обеспечение (например, на основе программного кода и подробного описания проектного решения).
- для программного обеспечения может быть измерена величина покрытия для имеющихся тестовых сценариев, и последующие тестовые сценарии могут разрабатываться для систематического увеличения покрытия.

Общие признаки методов на основе опыта:

- для определения тестовых сценариев используются человеческие знания и опыт.

- знания тестировщиков, разработчиков, пользователей и заинтересованных лиц о программном продукте, его использовании и окружении, являются одним из источников информации.
- знания о вероятных дефектах и их распределении являются другим источником информации

4.3 Методы, основанные на спецификациях, или методы черного ящика (К3)	150 минут

Терминология

Анализ граничных значений, тестирование таблицы решений, эквивалентное разбиение, тестирование таблицы переходов, тестирование по сценариям использования.

4.3.1 Эквивалентное разбиение (К3)

Входные данные для программного обеспечения или системы разбиваются на группы, от которых ожидается сходное поведение, то есть они должны обрабатываться аналогичным образом. Эквивалентные области (или классы) могут быть определены как для валидных, так и для невалидных данных, то есть тех значений, которые должны отвергаться. Области также могут быть определены для выходных данных, внутренних значений, значений, зависящих от времени (например, до или после некоторого события) и для параметров интерфейса (например, во время интеграционного тестирования). Тесты могут разрабатываться для покрытия всех валидных и всех невалидных классов. Эквивалентное разбиение применимо на всех уровнях тестирования.

Эквивалентное разбиение может быть использовано с целью покрытия входных и выходных данных. Оно может применяться при ручном вводе данных, при передаче данных через интерфейсы в систему, или при проверке параметров интерфейсов в интеграционном тестировании.

4.3.2 Анализ граничных значений (К3)

Поведение на границах эквивалентных областей имеет наибольшие шансы быть некорректным, таким образом границы являются потенциальным источником дефектов. Минимальные и максимальные значения сегмента являются граничными значениями. Граничное значение для валидного сегмента является валидным граничным значением, для невалидного сегмента – невалидным. Тесты могут разрабатываться для покрытия как валидных, так и невалидных граничных значений. При разработке тестовых сценариев выбирабтся тесты для каждого граничного значения.

Анализ граничных значений может применяться на всех уровнях тестирования. Он относительно легок в применении и эффективен при поиске дефектов. Для выделения интересующих нас границ крайне полезны подробные спецификации.

Данный метод часто рассматривается как дополнение к методу эквивалентного разбиения. Он может использоваться для классов эквивалентности данных, вводимых на экране, так и, например, для классов эквивалентности временных диапазонов (например, таймауты или требования по быстродействию транзакций) или для размерности таблиц (например, размер таблицы 256*256).

4.3.3 Тестирование таблицы решений (К3)

Таблицы решений – хороший метод для сбора системных требований, содержащих логические условия и документирования внутреннего дизайна системы. Они могут использоваться для записи сложных бизнес-правил, которые должна реализовывать система. Анализируются спецификации и определяются условия и действия системы. Входные условия и действия чаще всего формулируются таким образом, чтобы они могли принимать логические значения «истина» или «ложь».

Таблица решений содержит триггерные условия, обычно комбинации значений «истина» и «ложь» для всех входных условий, и результирующие действия для каждой комбинации условий. Каждый столбец таблицы соотносится с бизнес-правилом, определяющим уникальную комбинацию условий и результат выполнения действий, связанных с этим правилом. Стандартом покрытия для тестирования таблицы решений обычно является наличие хотя бы одного теста для каждой колонки, что обычно включает в себя покрытие всех комбинаций триггерных условий.

Сильной стороной тестирования таблицы решений является то, что она создает комбинации условий, которые могли бы быть не проверены в ходе тестирования иным способом. Этот метод может быть применен ко всем ситуациям, в которых действие программного продукта зависит от нескольких логических альтернатив.

4.3.4 Тестирование таблицы переходов (К3)

Система может показывать различные отклики в зависимости от текущих условий или предшествовавшей истории состояний. Данный метод позволяет тестировщику рассматривать систему с точки зрения её состояний, переходов между состояниями, входов или событий, активизирующих изменения состояний (переходы) и действия, к которым приводят эти переходы. Состояния системы или тестируемого объекта разделяемы, определяемы и конечны.

Таблица состояний демонстрирует связи между состояниями и входами и может подсказать возможные некорректные переходы.

Тесты могут разрабатываться для покрытия типовой последовательности состояний, для покрытия каждого состояния, для выполнения каждого перехода, для выполнения определенных последовательностей переходов или же для тестирования некорректных переходов.

Тестирование таблицы переходов чаще всего используется в индустрии встроенного программного обеспечения и автоматизации в целом. Однако эта методика также подходит для моделирования бизнес-объектов, имеющих определенные состояния, или тестирования последовательностей диалоговых блоков (т.е. интернет-приложений или бизнес-сценариев).

4.3.5 Тестирование по сценариям использования (К2)

Тесты могут базироваться на сценариях использования. Сценарий использования описывает взаимодействия между участниками (включая пользователей и систему) приводящие к полезным результатам для заказчика или пользователя системы. Сценарии использования могут быть описаны на уровне абстракций

(бизнес сценарий использования, уровень бизнес-процессов, не связанный с технологией) или на системном уровне (сценарий использования системы на уровне системного функционала). Каждый сценарий использования имеет предусловия, необходимые для успешного выполнения сценария. Каждый сценарий использования завершается постусловиями, являющимися наблюдаемыми результатами и итоговым состоянием системы после выполнения сценария. Сценарий использования обычно имеет основной (наиболее вероятный) сценарий и альтернативные сценарии.

Сценарии использования описываются как «потоки процессов» в системе, основанные на типовом предполагаемом использовании. Таким образом, тестовые сценарии, построенные на основе сценариев использования, являются наиболее полезными для выявления дефектов в потоках процессов во время реального использования системы. Сценарии использования очень полезны для разработки приёмочных тестов с участием заказчика или пользователей. Также они могут выявить дефекты интеграции, вызванные взаимодействием различных компонентов, не выявляемые индивидуальным тестированием компонентов.

Проектирование тестовых сценариев на основе сценариев использования может быть объединено с другими методами, основанными на спецификации.

4.4 Тестирование на основе структуры, или методы белого ящика (К3)	60 минут

Терминология

Покрытие кода, покрытие альтернатив, покрытие операторов, тестирование на основе структуры.

Введение

Тестирование на основе структуры, или тестирование методом белого ящика, основывается на конкретной структуре программного продукта или системы, как рассмотрено в следующих примерах:

- Компонентный уровень: структура компонента программного обеспечения, т.е. операторы, альтернативы, ветви или определенные пути
- Интеграционный уровень: структура может быть представлена деревом вызовов (диаграмма, в которой модули вызывают другие модули)
- Системный уровень: структура может представлять собой структуру меню, бизнес-процессов или же схему веб-страницы.

В этом разделе рассматриваются три метода проектирования тестов на основе структуры для покрытия кода: основанные на операторах, ветвях и альтернативах. Для визуализации тестирования альтернатив может использоваться диаграмма потока управления.

4.4.1 Тестирование операторов и покрытие (К3)

В компонентном тестировании покрытие операторов - это доля операторов, проверенных во время выполнения набора сценария тестирования. При тестировании операторов тестовые сценарии создаются таким образом, чтобы выполнять определенные операторы и обычно увеличивать покрытие операторов.

Величина покрытия операторов определяется как отношение числа выполняемых операторов, покрытых тестовыми сценариями (разработанными или выполненными) к общему числу операторов в тестируемом коде.

4.4.2 Тестирование альтернатив и покрытие (К3)

Покрытие альтернатив, связанное с тестированием ветвей - это доля результатов альтернатив (например, вариантов «Истина» и «Ложь» для оператора «Если»), проверенных набором сценариев тестирования. В методе тестирования альтернатив тестовые сценарии создаются для выполнения определенных результатов альтернатив. Ветви исходят из точек альтернатив в программном коде и показывают передачу управления различным участкам кода.

Покрытие альтернатив определяется отношением числа всех результатов альтернатив, покрытых разработанными или выполненными тестовыми сценариями к числу всех возможных результатов альтернатив в тестируемом коде.

Тестирование альтернатив - это вид тестирования потока управления, так как оно описывает прохождение определенного потока через точки альтернативы. Покрытие альтернатив более строгое, чем покрытие операторов: 100% покрытие альтернатив обеспечивает 100% покрытие операторов, но не наоборот.

4.4.3 Другие методы, основанные на структуре (К1)

Имеются и более высокие уровни покрытия структуры после покрытия альтернатив. Например, покрытие условий и покрытие множественных условий.

Принцип покрытия также может быть применен на других уровнях тестирования. Например, на уровне интеграции процент модулей, компонентов или классов, проверенных набором тестовых сценарием, может быть выражен как покрытие модулей, компонентов или классов.

Для структурного тестирования крайне полезна инструментальная поддержка.

4.5 Методы, основанные на опыте (К2)	30 минут

Терминология

Исследовательское тестирование, атака (на недочеты)

Введение

Когда тесты создаются на основе мастерства тестировщика, его интуиции и опыте работы с подобными приложениями или технологиями, это называется тестированием, основанным на опыте. Данный вид тестирования полезен как дополнение более систематических методов для разработки специальных тестов, не всегда очевидных при использовании более формальных методик, особенно когда используется после таких методик. Однако полезность этого метода может сильно варьироваться в зависимости от опыта тестировщика.

Наиболее часто используемым методом, основанным на опыте, является предположение об ошибках. Зачастую тестировщики ожидают дефекты, исходя из своего опыта. Организованным подходом к предположению об ошибках является создание списка возможных дефектов и разработка тестов для атаки этих дефектов. Данный подход называется атакой, или атакой на недочеты. Списки дефектов и отказов могут быть созданы на основе опыта, доступной информации о дефектах и отказах и общего представления о том, почему программное обеспечение может отказать.

Исследовательское тестирование - это параллельная разработка тестов, их выполнение, протоколирование тестирования и изучение, основанные на концепции тестирования, включающей в себя цели тестирования, и проводимые в определенных временных рамках. Данный подход наиболее полезен при наличии неполных или неактуальных спецификаций и жестких временных ограничений, или при усилении или дополнении более формального тестирования. Он может служить проверкой процесса тестирования для уверенности в том, что наиболее важные дефекты обнаружены.

4.6 Выбор методов тестирования (K2)	15 минут

Терминология

Специфичные термины отсутствуют.

Введение

Выбор метода тестирования зависит от некоторого количества факторов, включающих тип системы, нормативные стандарты, требования заказчика или контракта, уровни рисков, типы рисков, цели тестирования, доступную документацию, знания тестировщиков, время и бюджет, жизненный цикл разработки, модели сценариев использования и предыдущий опыт о типах найденных дефектов.

Некоторые методы более применимы в определенных ситуациях, другие применимы ко всем уровням тестирования.

При проектировании тестовых сценариев тестировщики обычно используют комбинации методов тестирования, включающих в себя методы, основанные на процессах, правилах и данных, с целью обеспечить адекватное покрытие тестируемого объекта.

Ссылки

4.1 Craig, 2002, Hetzel, 1988, IEEE 829

4.2 Beizer, 1990, Copeland, 2004

4.3.1 Copeland, 2004, Myers, 1979

4.3.2 Copeland, 2004, Myers, 1979

4.3.3 Beizer, 1990, Copeland, 2004

4.3.4 Beizer, 1990, Copeland, 2004

4.3.5 Copeland, 2004

4.4.3 Beizer, 1990, Copeland, 2004

4.5 Kaner, 2002

4.6 Beizer, 1990, Copeland, 2004

5 Управление тестированием (K3) — 170 минут

Цели изучения управления тестированием

Цели определяют, что нужно знать для завершения каждого модуля.

5.1 Организация тестирования (K2)

LO-5.1.1 Понять важность независимости тестирования (K1)

LO-5.1.2 Объяснить преимущества и недостатки независимости тестирования внутри организации (K2)

LO-5.1.3 Обозначить роли, которые могут быть рассмотрены при создании команды тестировщиков (K1)

LO-5.1.4 Перечислить типичные задачи тестировщика и руководителя тестирования (K1)

5.2 Планирование и оценка тестирования (K3)

LO-5.2.1 Определить уровни и цели планирования тестирования (K1)

LO-5.2.2 Определить цели и содержание плана тестирования, спецификации проектирования тестов и спецификации процедуры тестирования согласно «Стандарту по Тестовой Документации для Программного Обеспечения» (IEEE Std-829-1998) (K2)

LO-5.2.3 Различать концептуально разные подходы к тестированию: аналитический, основанный на моделях, методический, проверка на соответствие процессу\стандартам, динамический\эвристический, консультативный и регрессионный (K2)

LO-5.2.4 Различать задачи планирования тестирования и построения графика проведения тестирования (K2)

LO-5.2.5 Написать график проведения тестирования для данного набора тестовых сценариев, согласно приоритетам, технической и логической зависимости (K3)

LO-5.2.6 Перечислить задачи по подготовке и проведению тестирования, которые должны быть обозначены во время планирования тестирования (K1)

LO-5.2.7 Вспомнить основные факторы, влияющие на трудозатраты, связанные с тестированием (K1)

LO-5.2.8 Различать 2 концептуально различных подхода к оценке тестирования: основанный на метриках и основанный на экспертной оценке (K2)

LO-5.2.9 Идентифицировать и объяснить адекватные критерии входа и выхода тестирования для различных уровней тестирования и наборов тестовых сценариев (например для интеграционного, приемочного тестирования или набора тестов для тестирования удобства использования) (K2)

5.3 Мониторинг прогресса и контроль тестирования (K2)

LO-5.3.1 Вспомнить распространенные метрики, используемые для мониторинга подготовки и проведения тестирования (K1)

LO-5.3.2 Объяснить и сравнить тестовые метрики для тестовой отчетности и контроля тестирования по цели и применению (например, количество найденных и исправленных дефектов, количество успешно пройденных и неудачных тестов) (K2)

LO-5.3.3 Перечислить цели и содержание итогового отчета по тестированию согласно «Стандарту по Тестовой Документации для Программного Обеспечения» (IEEE Std-829-1998) (K2)

5.4 Управление конфигурацией (K2)

LO-5.4.1 Объяснить, как управление конфигурацией обеспечивает поддержку тестированию (K2)

5.5 Риски и тестирование (K2)

LO-5.5.1 Определить риск как возможную проблему, способную помешать достижению одной или нескольких целей участников проекта (K2)

LO-5.5.2 Вспомнить, как уровень риска определяется вероятностью возникновения проблемы и ее влияния (причиненный ущерб, если риск произойдет) (K1)

LO-5.5.3 Понимать разницу между рисками проекта и продукта (K2)

LO-5.5.4 Определять основные риски проекта и продукта (K1)

LO-5.5.5 Описать на примерах как анализ и управление рисками могут быть использованы при планировании тестирования (K2)

5.6 Управление инцидентами (K3)

LO-5.6.1 Определить содержание отчета об инцидентах согласно «Стандарту по Тестовой Документации для Программного Обеспечения» (IEEE Std-829-1998) (K1)

LO-5.6.2 Написать отчет об инцидентах, описывающий наблюдения за отказами во время тестирования. (K3)

5.1 Организация тестирования (K2)	30 минут

Терминология

Тестировщик, руководитель тестирования, менеджер тестирования

5.1.1 Организация и независимость тестирования (K2)

Эффективность поиска дефектов и рецензирования может быть повышена с помощью независимых тестировщиков, Варианты независимости могут быть следующими:

- Отсутствие независимых тестировщиков: разработчики тестируют собственный код;
- Независимые тестировщики в команде разработчиков;
- Независимая команда или группа тестирования в организации, отчитывающаяся менеджеру проекта или исполнительному менеджеру;
- Независимые тестировщики из бизнес-организации или сообщества пользователей;
- Независимые специалисты тестирования для отдельных типов тестирования, например, тестировщики удобства использования, тестировщики безопасности или тестировщики сертификации (которые сертифицируют ПО на соответствие стандартам и правилам);
- Независимые тестировщики, привлеченные на аутсорсинг или сторонние по отношению к организации.

Для больших, сложных или критичных с точки зрения безопасности проектов обычно лучше иметь несколько уровней тестирования, при этом некоторые или все уровни выполняются независимыми тестировщиками. Разработчики также могут участвовать в тестировании, особенно на низких уровнях, но недостаток объективности зачастую ограничивает их эффективность. Независимые тестировщики могут иметь право определять правила и процессы тестирования, но принимать роли в процессах должны только после недвусмысленного разрешения на это.

Преимущества независимого тестирования:

- Независимые тестировщики беспристрастны, видят другие, отличные дефекты;
- Независимые тестировщики могут проверять предположения, сделанные во время создания спецификаций и разработки системы.

Недостатки независимого тестирования:

- Изолированность от команды разработчиков (в случае полной независимости тестировщиков);
- Разработчики теряют чувство ответственности за качество;

- Независимые тестировщики могут быть узким местом, их могут обвинить в задержке выпуска продукта.

Задачи тестирования могут выполняться людьми в специальной тестовой роли или кем-то в другой роли, например, менеджером проекта, менеджером по качеству, разработчиком, экспертом в бизнесе или предметной области, инфраструктуре или ИТ.

5.1.2 Задачи руководителя тестирования и тестировщика (K1)

В этом курсе рассматриваются две позиции в тестировании: руководитель тестирования и тестировщик. Действия и задачи, выполняемые этими людьми, зависят от контекста проекта и продукта, от людей, выполняющих роли, и от организации.

Иногда руководитель тестирования называется менеджером или координатором тестирования. Роль руководителя тестирования может исполнять менеджер проекта, менеджер разработки, менеджер по качеству или управляющий группой тестирования. В больших проектах могут существовать 2 позиции: руководитель тестирования и менеджер тестирования. Обычно руководитель тестирования производит планирование, мониторинг и контроль тестирования, а также выполняет задачи, определенные в разделе 1.4

Типичные задачи руководителя тестирования могут включать:

- Координирование стратегий тестирования и планов с менеджерами проектов и другими людьми;
- Составление и анализ стратегии тестирования для конкретного проекта и тестовой политики для организации;
- Согласование перспектив тестирования с другими проектными процессами, такими как интеграционное планирование;
- Планирование тестов согласно содержанию и пониманию рисков, включая выбор методов тестирования, оценка времени, трудозатрат, стоимости тестирования и наличия ресурсов, определение уровней и циклов тестирования, а также планирование управления инцидентами;
- Инициирование спецификаций тестирования, подготовка, создание и исполнение тестов, отслеживание и контроль результатов тестирования, проверка критерия выхода;
- Планирование адаптации тестирования согласно результатам и прогрессу тестирования (иногда задокументировано в отчетах), а так же принятие решений и выполнение действий по решению проблем;
- Установка адекватного управления конфигурацией для обеспечения трассируемости тестирования;
- Принятие решение о том, что, как и до какого уровня должно быть автоматизировано;
- Выбор средства поддержки тестирования и организация обучения тестировщиков по его использованию;

- Принятие решение о внедрении среды тестирования;
- Написание отчетов тестирования, основанных на информации, полученной при проведении тестирования.

Типичные задачи тестировщика могут включать:

- Рецензирование и дополнение планов тестирования;
- Анализ, рецензирование и оценка тестируемости пользовательских требований, спецификаций и моделей;
- Создание спецификаций тестирования;
- Настройка тестового окружения (часто совместно с системными администраторами и руководством по сетям);
- Подготовка и получение тестовых данных;
- Разработка тестов на всех уровнях тестирования, исполнение и запись тестов, оценка результатов и документирование отклонений от ожидаемых результатов;
- Использование при необходимости средств администрирования или управления тестированием, а также отслеживания тестирования;
- Автоматизация тестов (возможно при поддержке разработчика или эксперта по автоматизации);
- Измерение производительности компонентов или системы (если необходимо);
- Рецензирование тестов, разработанных другими тестировщиками.

Люди, работающие над анализом, проектированием, определенными типами тестов или автоматизацией могут быть специалистами в этих ролях. В зависимости от уровня тестирования и рисков, относящихся к продукту или проекту, различные люди могут принимать на себя роль тестировщика, соблюдая некий уровень независимости. Обычно тестировщиками на уровне компонентов или интеграции являются разработчики, на приемочном уровне это могут быть бизнес-эксперты и пользователи, на эксплуатационном приемочном – операторы.

5.2 Планирование и оценка тестирования (K3)	40 минут

Терминология

Подход к тестированию, стратегия тестирования

5.2.1 Планирование тестирования (K2)

Этот раздел описывает цели планирования тестирования в рамках проектов разработки и внедрения. Планирование может быть задокументировано в главном плане тестирования, а также в отдельных планах для уровней, таких как системное или приемочное тестирование. Описание документации по планированию тестирования содержится в «Стандарте по Тестовой Документации для Программного Обеспечения» (IEEE Std-829-1998)

На планирование влияют тестовая политика организации, объем тестирования, объекты тестирования, риски, ограничения, критичность, тестируемость и наличие ресурсов. Чем дальше развиваются проект и планирование тестирования, тем больше доступной информации и больше деталей может быть включено в план. Планирование тестирования – непрерывный процесс, выполняемый во время всего жизненного цикла. Обратная связь от результатов тестовой деятельности используется для определения изменения рисков, таким образом, что бы планирование можно было корректировать.

5.2.2 Действия по планированию тестирования (K3)

Действия по планированию тестирования всей системы или ее части могут включать:

- Определение объема, рисков и целей тестирования;
- Определение общего подхода к тестированию, включая уровни тестирования и критерия входа;
- Интегрирование и координация действий тестирования с жизненным циклом ПО (приобретение, поставка, разработка, функционирование и поддержка);
- Принятие решений о том, что тестировать, какие роли нужны для выполнения тестирования, когда и как проводить тестирование и как оценивать результаты;
- Составление расписания анализа и проектирования тестов;
- Составление расписания подготовки, исполнения и оценки тестов;
- Назначение ресурсов для различных, определенных ранее, действий;
- Определение размера, уровня детализации, структуры и шаблонов для тестовой документации;
- Выбор метрик для мониторинга и контроля подготовки и проведения тестирования, исправления дефектов, проблем и рисков;

- Установка уровня детализации для тестовых процедур для предоставления достаточной информации, чтобы поддерживать повторяемость подготовки и проведения тестирования.

5.2.3 Критерий входа (K2)

Критерий входа определяет, когда нужно начинать тестирование, например, для начала уровня тестирования или когда набор тестов готов для исполнения.

Обычно критерии входа могут покрывать:

- Готовность и доступность тестового окружения;
- Готовность средства тестирования в окружении;
- Доступность тестируемого кода;
- Доступность тестовых данных.

5.2.4 Критерий выхода (K2)

Критерий выхода определяет, когда нужно прекращать тестирование, например, по окончании уровня тестирования или когда набор тестов достиг определенной цели.

Обычно критерии выхода могут покрывать:

- Тщательность оценки, например покрытие кода, функциональности или рисков;
- Оценку плотности дефектов или измерение надежности;
- Стоимость.
- Остаточные риски, такие как неисправленные дефекты или недостаток тестового покрытия какой-либо области;
- План, основанный на времени выхода ПО на рынок.

5.2.5 Оценка тестирования (K2)

В данном курсе рассматриваются два подхода к оценке трудозатрат в тестировании:

- Основанный на метриках: оценка трудозатрат основана на метриках предыдущих или сходных проектов или основана на типичных значениях;
- Основанный на экспертной оценке: оценка задач производится владельцем этих задач или экспертом.

Как только оценка трудозатрат выполнена, можно определить ресурсы или составить расписание.

Работы по тестированию могут зависеть от ряда факторов, включая:

- Характеристики продукта: качество спецификаций или другой информации, используемых моделей тестирования (т.е. основы тестирования), размер продукта, сложность предметной области, требования к надежности и безопасности и требования к документации;

- Характеристики процесса разработки: стабильность организации, используемые средства, процессы тестирования, квалификация вовлеченных людей и временные ограничения;
- Результат тестирования: количество дефектов и объем работы, которую необходимо переделать.

5.2.6 Стратегия тестирования, подход к тестированию (K2)

Подход к тестированию является реализацией стратегии тестирования для конкретного проекта.

Подход к тестированию определяется и детализируется в плане тестирования и спецификации проектирования тестов и обычно включает в себя решения, основанные на (тестовых) целях проекта и оценке рисков. Это начальная точка для планирования тестовых процессов, выбора метода и типа тестов и определения критериев входа и выхода.

Выбранный подход к тестированию зависит от контекста и может учитывать риски, угрозы и безопасность, доступные ресурсы и квалификацию, технологию, природу системы (например, индивидуальная сборка или коммерческое готовое ПО), цели тестирования и регламент.

Типичные подходы к тестированию:

- Аналитические подходы, например, тестирование, основанное на рисках, где тестирование направлено на области с наивысшим риском;
- Подходы, основанные на моделях, например, стохастическое тестирование, в котором используется статистическая информация об уровне сбоев (например, модель возрастающей надежности) или использования (например, рабочие профили);
- Методические подходы, например, основанные на дефектах (включая предположение об ошибке или атаки), на опыте, чек-листах и характеристиках качества;
- Подходы, основанные на соответствии процессам или стандартам, например, которые определены согласно индустриальным стандартам или гибким методологиям;
- Динамические и эвристические подходы, такие как исследовательское тестирование, при котором тестирование больше реагирует на события, чем на запланированный сценарий, и при котором проведение и оценка тестирования являются параллельными задачами;
- Консультативные подходы, например те, для которых тестовое покрытие управляется в основном советами и указаниями экспертов в технологиях и\или предметной области и которые не входят в команду;
- Регрессионные подходы, которые включают повторное использование существующих материалов тестирования, всестороннюю автоматизацию функциональных регрессионных тестов и стандартных наборов тестов.

5.3 Мониторинг прогресса и контроль тестирования (K2)	20 минут

Терминология

Плотность дефектов, интенсивность отказов, контроль тестирования, мониторинг тестирования, отчет о результатах тестирования

5.3.1 Мониторинг прогресса тестирования (K1)

Целью мониторинга тестирования является предоставление результата и обзора процесса тестирования. Информация отслеживается вручную или автоматически и может быть использована для измерения критериев выхода, таких как покрытие. Метрики также могут быть использованы для оценки прогресса тестирования по сравнению с запланированным расписанием и бюджетом.

Обычные тестовые метрики включают в себя:

- Процент проделанной работы по подготовке тестовых сценариев (или процентное соотношение запланированных и подготовленных сценариев);
- Процент проделанной работы по подготовке тестового окружения;
- Выполнение тестовых сценариев (например, количество выполненных\невыполненных тестовых сценариев, успешно пройденных\неудачных тестовых сценариев);
- Информация о дефектах (например, плотность дефектов, количество найденных и исправленных дефектов, интенсивность отказов и результаты повторного тестирования);
- Тестовое покрытие требований, рисков или кода;
- Субъективная уверенность тестировщиков в продукте;
- Даты контрольных точек тестирования;
- Стоимость тестирования, включая стоимость по сравнению с выгодой нахождения следующего дефекта или запуска следующего теста.

5.3.2 Отчетность по тестированию (K2)

Отчеты о тестировании предоставляют итоговую информацию о тестировании, включая:

- Что произошло во время цикла тестирования, например, даты, когда были достигнуты критерии выхода;
- Проанализированную информацию и метрики для поддержки рекомендаций и решений о последующих действиях, таких как оценка оставшихся дефектов, экономическое обоснование продолжения тестирования, оставшиеся риски и уровень уверенности в тестируемом ПО.

Структура отчета о результатах тестирования приводится в «Стандарте по тестовой Документации для Программного Обеспечения» (IEEE Std 829-1998).

Метрики, которые необходимо собрать во время и в конце уровня тестирования:

- Соответствие целей тестирования уровню тестирования;
- Адекватность выбора подхода к тестированию;
- Эффективность тестирования в отношении установленных целей.

5.3.3 Контроль тестирования (K2)

Контроль тестирования описывает любые направляющие или корректирующие действия, принятые как результат по полученной и собранной информации и значениям метрик. Контроль тестирования может затрагивать любые действия по тестированию, а так же воздействовать на другие действия и задачи жизненного цикла ПО.

Примеры действий по контролю тестирования:

- Принятие решений на основании данных мониторинга тестирования;
- Повторная расстановка приоритетов при возникновении установленного риска (например, задержка выпуска ПО);
- Изменение графика тестирования согласно доступности тестового окружения;

Установка критерия входа, требующего повторного (подтверждающего) тестирования исправлений, сделанных разработчиком, перед принятием их в сборку.

5.4 Управление конфигурацией (K2)	10 минут

Терминология

Управление конфигурацией, управление версиями

Введение

Целью управления конфигурацией является установка и поддержка интегрируемости продуктов (компонентов, данных и документации) ПО или системы на протяжении жизненного цикла проекта или продукта.

Для тестирования управление конфигурацией может гарантировать:

- Все пункты того, что нужно протестировать, определены, контролируются по версиям, все изменения учитываются, связаны с разрабатываемыми элементами (объекты тестирования) и друг с другом так, что бы обеспечить трассировку на протяжении всего процесса тестирования;
- Вся установленная документация и элементы ПО однозначно определены в документации по тестированию.

Для тестировщика, управление конфигурацией помогает однозначно определить (и воспроизвести) элемент тестирования, документацию тестирования, тесты и средства тестирования.

Процедура управления конфигурацией и инфраструктура (средства) выбираются, документируются и внедряются на стадии планирования тестирования.

5.5 Риски и тестирование (K2)	30 минут

Терминология

Риски продукта, Риски проекта, риск, ориентированное на риски тестирование

Введение

Риск может быть определен как вероятность возникновения события, опасности, угрозы или ситуации, которая выражается в нежелательных последствиях или потенциальной проблеме. Уровень риска определяется вероятностью возникновения неблагоприятного события и его влияния (ущерб, проявляющийся в результате этого события).

5.5.1 Риски проекта (K2)

Риски проекта – это риски, которые влияют на способность проекта достигнуть его целей, и включают:

- Организационные факторы:
  - Недостаток квалификации, подготовки и сотрудников;
  - Личные проблемы сотрудников;
  - Политические проблемы, такие как:
    - Тестировщики в недостаточной степени сообщают о своих проблемах и результатах тестирования;
    - Неспособность следовать информации, полученной во время тестирования или рецензирования (например, не улучшать практики разработки или тестирования);
  - Неверное отношения к тестированию или ложные ожидания (например, не принимать во внимание значение найденных во время тестирования дефектов);
- Технические проблемы:
  - Проблемы в определении верных требований;
  - Объем, при котором требования не могут соответствовать заданным ограничениям;
  - Вовремя не готово тестовое окружение;
  - Позднее преобразование данных, планирование миграции и разработки тестовых данных и средств преобразования\миграции тестовых данных;
  - Низкое качество проектирования, кода, конфигурационных и тестовых данных и тестов;
- Проблема поставщика:

- Отказ третьей стороны;
- Проблемы контракта.

При анализе, управлении и уменьшении этих рисков менеджер тестирования должен следовать хорошо обоснованным принципам управления проектом. «Стандарт по тестовой Документации для Программного Обеспечения» (IEEE Std 829-1998) содержит шаблон плана тестирования, требующий определения рисков и непредвиденных обстоятельств.

5.5.2 Риски продукта (K2)

Риски продукта – это потенциальные области сбоя (неблагоприятные будущие события или опасность) в ПО или системе, т.к. они подвергают риску качество продукта, например:

- Поставка потенциально ненадежного ПО;
- Возможность того, что программное\аппаратное обеспечение может нанести вред человеку или компании;
- Плохие характеристики ПО (например, функциональность, надежность, удобство использования или производительность);
- Неполнота и низкое качество данных (например, проблемы миграции, преобразования и перемещения данных, отклонение от стандарта данных);
- ПО, которое не выполняет предполагаемых функций.

Риски используются для определения того, где начинать тестирование и каким аспектам уделить большее внимание, тестирование используется для уменьшения риска возникновения неблагоприятных эффектов или их последствий.

Риски продукта – это особенный тип рисков, который влияет на успех продукта. Тестирование как действие, контролирующее риски, позволяет определить остаточные риски путем измерения эффективности исправления критичных дефектов и плана на случай непредвиденных обстоятельств.

Подход к тестированию, основанный на рисках, предоставляет превентивные возможности уменьшения рисков продукта, начиная с ранних стадий проекта. Он включает в себя идентификацию рисков продукта и их использование в планировании и контроле тестирования, требованиях, подготовке и выполнении тестов. В подходе, основанном на рисках, их можно использовать для:

- Определения методики тестирования для использования;
- Определения объема тестирования, которое должно быть выполнено;
- Установления приоритетов для тестирования, что бы найти критичные дефекты как можно раньше;
- Определения, нужны ли дополнительные, не связанные с тестированием действия по уменьшению рисков (например, проведение тренинга для неопытных проектировщиков).

Тестирование, основанное на рисках, использует коллективное знание и понимание участников проекта для определения рисков и уровней тестирования, необходимых для работы с этими рисками.

Для того, что бы удостовериться, что сбой в продукте минимизирован, действия по управлению рисками обеспечивают строгий порядок подхода к:

- Оценке (и переоценке на регулярной основе) того, что может пойти неверно (риски);
- Определению, какие риски наиболее важны для решения;
- Выполнение действий по работе с этими рисками.

В дополнение, тестирование может поддерживать нахождение новых рисков, помогает определить какие риски должны быть уменьшены, а также может снижать неопределенность в отношении рисков.

5.6 Управление инцидентами (K3)	40 минут

Терминология

Регистрирование инцидентов, управление инцидентами, отчет по инцидентам

Введение

Так как одной из целей тестирования является поиск дефектов, то разница между действительным и ожидаемым результатом должна быть зарегистрирована как инцидент. После обязательного изучения инцидента, он может быть определен как дефект. Нужно определить приемлемые действия по распределению инцидентов и дефектов. Инциденты и дефекты должны отслеживаться от момента обнаружения и классификации до исправления и подтверждения, что проблема решена. Для управления всеми инцидентами до их завершения в организации необходимо установить процесс управления инцидентами и правила их классификации.

Инциденты могут быть обнаружены во время разработки, рецензирования, тестирования или использования ПО. Они могут быть найдены в коде, работающей системе, документации любого типа, включая требования, документы разработки, тестовые документы и документы для пользователей, например «Справка» или руководство по установке.

Цели отчета по инцидентам:

- Обеспечить разработчиков и другие заинтересованные стороны информацией о проблеме, что бы идентифицировать, изолировать и исправить, если это необходимо;
- Обеспечить руководителя тестирования средством отслеживания качества тестируемой системы и прогресса тестирования;
- Обеспечивать идеями по улучшению процесса тестирования.

Отчета об инцидентах может содержать:

- Дату нахождения, нашедшую организацию и автора инцидента;
- Ожидаемый и фактический результат;
- Идентификатор тестового (конфигурационного) элемента и окружение;
- Жизненный цикл ПО или системы, в котором был обнаружен инцидент;
- Описание, необходимое для воспроизведения и решения инцидента, включая логи, дампы базы или снимки экрана;
- Уровень влияния на заинтересованные стороны;
- Серьезность влияния на систему;
- Срочность\приоритет для исправления;

- Статус инцидента (например, открыт, отложен, копия, ожидает исправления, ожидает подтверждающего тестирования, закрыт);
- Заключение, рекомендации и подтверждение;
- Общие проблемы, например, другие области, которые могут быть затронуты при решении инцидента;
- История изменений, например, последовательность действий, принятых членами команды проекта для изолирования, исправления и подтверждения исправления инцидента;
- Ссылки, включая идентификатор на спецификацию тестового сценария, который выявил проблему.

Структура отчета об инцидентах также описана в «Стандарте по тестовой Документации для Программного Обеспечения» (IEEE Std 829-1998)

Ссылки:

5.1.1 Black, 2001, Hetzel, 1988

5.1.2 Black, 2001, Hetzel, 1988

5.2.5 Black, 2001, Craig, 2002, IEEE Std 829-1998, Kaner 2002

5.3.3 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE Std 829-1998

5.4 Craig, 2002

5.5.2 Black, 2001 , IEEE Std 829-1998

5.6 Black, 2001, IEEE Std 829-1998

6 Инструментальные средства поддержки тестирования (K2) — 80 минут

Цели изучения инструментальных средств поддержки тестирования

Цели определяют Ваши возможности после завершения работы с каждым модулем.

6.1 Типы инструментов тестирования (K2)

LO-6.1.1 Классифицировать различные типы инструментов тестирования в соответствии с их целями и деятельностью в фундаментальном процессе тестирования и жизненном цикле программного обеспечения (K2)

LO-6.1.3 Объяснить значение инструмента тестирования как термина, а также цели инструментальных средств поддержки (K2)

6.2 Эффективное использование инструментальных средств: выгоды и риски (K2)

LO-6.2.1 Обобщить выгоды и риски использования автоматизации тестирования и инструментальных средств поддержки тестирования. (K2)

LO-6.2.2 Вспомнить особенности средств выполнения тестов, статического анализа и инструментов управления тестами. (K1)

6.3 Внедрение инструментального средства в организацию (K1)

LO-6.3.1 Установить основные принципы внедрения инструментального средства в организацию(K1)

LO-6.3.2 Установить цели опытно-экспериментальной оценки инструментального средства и пилотной фазы его внедрения. (K1)

LO-6.3.3 Объяснить, , что, кроме приобретения инструмента, требуется для его качественной поддержки. (K1)

6.1 Типы инструментов тестирования (К2)	45 минут

Терминология

Средство управления конфигурацией, инструмент покрытия, инструмент отладки, инструмент динамического анализа, инструмент управления инцидентами, инструмент нагрузочного тестирования, инструмент моделирования, инструмент мониторинга, инструмент тестирования производительности, эффект зондирования, инструмент управления требованиями, инструмент рецензирования, средство защиты, инструмент статистического анализа, инструмент стрессового тестирования, тестовый компаратор, инструмент подготовки тестовых данных, инструмент проектирования тестов, тестовая обвязка, инструмент выполнения тестов, инструмент управления тестированием, инструмент интегрированной среды модульного тестирования.

6.1.1 Применение инструментов в тестировании (К2)

Инструменты тестирования могут быть использованы для одного или нескольких действий, помогающих тестированию. К инструментам тестирования относятся:

1. Инструменты, используемые непосредственно в тестировании, такие как инструмент выполнения тестов, инструмент подготовки тестовых данных и инструмент сравнения результатов.
2. Инструменты, помогающие в организации процесса тестирования, среди них инструменты, используемые для управления тестами, результатами тестов, данными, требованиями, инцидентами, дефектами и так далее, и инструменты для создания отчетов и мониторинга процесса выполнения тестов.
3. Инструменты, используемые для «зондирования» или, проще говоря, исследования, например, инструменты мониторинга файловой активности во время работы приложения.
4. Любой инструмент, оказывающий поддержку тестированию (электронная таблица также может считаться тестовым инструментом в этом смысле).

Поддержка тестирования инструментами автоматизации может преследовать одну или более из перечисленных ниже целей в зависимости от контекста:

- Улучшить эффективность тестовых активностей, автоматизируя повторяющиеся задачи, или поддерживая тестовые активности, такие как планирование, тест дизайн, тестовые отчеты и мониторинг;
- Автоматизировать действия, требующие значительных затрат ресурсов для выполнения вручную (например, статическое тестирование);
- Автоматизировать действия, которые не могут быть выполнены вручную (например, крупномасштабное тестирование производительности клиент-серверных приложений);

- Увеличить надежность тестирования (например, проведя автоматизацию сравнения больших объемов данных или имитации поведения).

Термин «тестовая среда» часто используется в области тестирования и разработки программного обеспечения как минимум в трех значениях:

- Многократно используемые и расширяемые библиотеки, которые могут быть использованы для построения инструментов тестирования (также известные как тестовые обвязки);
- Тип проектирования тестов, применяемого для автоматизации тестирования (например, управляемое данными или на основе ключевых слов);
- Общий процесс исполнения тестирования;

В этой программе термин «тестовая среда» используется в первых двух значениях, как указано в Главе 6.1.6.

6.1.2 Классификация инструментов тестирования (К2)

Существует много инструментов, поддерживающих различные аспекты тестирования. Инструменты могут быть классифицированы по-разному: по назначению, по виду поставки (коммерческий/бесплатный/с открытым исходным кодом/условно-бесплатный), по используемым технологиям и так далее.

Некоторые инструменты поддерживают только одну деятельность тестирования, другие могут поддерживать более одной, но классифицированы согласно той деятельности, с которой теснее всего связаны. Инструменты от одного производителя могут быть объединены в один программный пакет, особенно если были разработаны для совместного использования.

Некоторые типы инструментов тестирования могут обладать побочными эффектами, влияя на фактический результат теста. Например, фактическое время выполнения теста может отличаться из-за дополнительных операций, выполняемых инструментом, или вы можете получить неправильный результат измерения покрытия кода. Следствием работы таких инструментов является эффект зондирования.

Некоторые инструменты предлагают возможности, более подходящие для разработчиков (например, инструменты, используемые на фазе компонентного или интеграционного тестирования). Такие инструменты помечаются буквой «(D)» в списке, приведенном ниже.

6.1.3 Инструменты для управления тестированием и тестами (К1)

Инструменты управления применяются ко всем тестовым активностям, проходящим в течение жизненного цикла программного продукта.

Инструменты управления тестированием

Эти инструменты предоставляют возможности для выполнения тестов, отслеживания дефектов и управления требованиями, совместно с поддержкой количественного анализа и отчетов о целях (объектах) тестирования.

Инструменты управления требованиями

Эти инструменты позволяют хранить и управлять отчетами об инцидентах, такие как дефекты, отказы в работе, запросы об изменениях или осознанные проблемы и аномалии, и помогают в управлении жизненным циклом инцидентов, дополнительно предоставляя поддержку статистического анализа.

Средства управления конфигурацией

Хотя и не являются тестовыми инструментами как таковыми, в обязательном порядке используются для хранения и управления версиями тестового обеспечения и всего связанного с ним программного обеспечения, главным образом, когда приходится настраивать более одного программного/аппаратного тестового окружения (в случаях использования различных версий операционных систем, компиляторов, браузеров и так далее).

6.1.4 Инструменты статического тестирования (K1)

Инструменты статического тестирования предоставляют экономически эффективный способ нахождения дефектов на ранних стадиях процесса разработки.

Инструмент рецензирования

Эти инструменты помогают в процессе рецензирования, организации контрольных списков, обзоров (результатов) рецензирования и используются для хранения и передачи рецензионных комментариев, отчетов по дефектам и объемам работ.

Инструменты статического анализа (D)

Инструменты этого вида помогают разработчикам и тестировщикам находить дефекты до начала фазы динамического тестирования, предоставляя возможности обеспечения (и соблюдения) стандартов написания кода (включая написание безопасного кода), анализа структур и зависимостей. Они также могут помочь в планировании или анализе рисков с помощью метрик (например, сложности).

Инструменты моделирования

Такие инструменты используются для проверки программных моделей (таких как модель физических данных для реляционных баз данных) путем перечисления несоответствий и нахождения дефектов. Инструменты моделирования также часто помогают в создании тестовых сценариев, основанных на модели.

6.1.5 Инструменты для работы с тестовыми спецификациями (K1)

Инструменты проектирования тестов

Этот вид инструментов используется для создания тестовых данных или исполнимых тестов и/или тестовых оракулов на основе требований, графического пользовательского интерфейса, моделей дизайна (состояний, данных или объектов) или кода.

Инструменты подготовки тестовых данных

Инструменты подготовки тестовых данных работают с базами данных, файлами или пакетами данных, чтобы подготовить тестовые данные, которые будут

использоваться в процессе выполнения тестов, обеспечивая безопасность на основе анонимности данных.

6.1.6 Инструменты выполнения тестов и протоколирования (K1)

Инструменты выполнения тестов

Инструменты этого вида позволяют исполнять тесты автоматически или полуавтоматически с помощью скриптовых языков программирования, используя сохраненные входные данные и ожидаемые результаты, и обычно предоставляют протокол запуска каждого теста. Также они могут быть использованы для записи тестов, обычно конфигурируя параметры данных и другие настройки тестов с использованием скриптового языка программирования или графического пользовательского интерфейса.

Тестовая обвязка/Инструменты интегрированной среды модульного тестирования (D)

Обвязка или среда модульных тестов содействует в тестировании компонентов, или частей системы, имитируя среду, в которой будет использоваться тестовый объект, путем предоставления фиктивных объектов – заглушек или тестовых драйверов.

Тестовые компараторы

Тестовые компараторы находят различия между файлами, базами данных или результатами тестов. Инструменты выполнения тестов чаще всего включают в себя динамические компараторы, но сравнение результатов после запуска может быть проведено с использованием отдельного компаратора. Тестовый компаратор может использовать тестовый оракул, особенно если компаратор автоматизирован.

Инструменты измерения покрытия (D)

Такие инструменты измеряют процент структур кода указанного типа, которые были задействованы тестовым набором (такие как операторы, ветвления, модули и вызовы функций).

Инструменты тестирования безопасности

Инструменты этого типа используются для оценки характеристик безопасности системы (программного продукта). Это включает оценку способность программы защитить конфиденциальность данных, целостность, аутентификацию, авторизацию, доступность и безотказность. Инструменты безопасности чаще всего сфокусированы на одной технологии, платформе и цели.

6.1.7 Инструменты для производительности и мониторинга (K1)

Инструменты динамического анализа (D)

Инструменты динамического анализа позволяют находить дефекты, появляющиеся лишь в момент выполнения программы, такие как временные зависимости или утечки памяти. Они чаще всего используются в компонентном и интеграционном тестировании и в тестировании, проводимом на промежуточных этапах жизненного цикла продукта.

Инструменты тестирования производительности/тестирования нагрузки/стрессового тестирования

Инструменты тестирования производительности позволяют наблюдать и составлять отчеты о поведении системы при различных моделируемых условиях эксплуатации - различном количестве пользователей, использующих систему одновременно, скорости их работы, частоты и относительной доли транзакций. Имитация нагрузок достигается путем создания виртуальных пользователей, использующих выбранный набор операций, на нескольких тестовых машинах, обычно называемых генераторами нагрузки.

Инструменты мониторинга

Инструменты мониторинга проводят длительный анализ, контролируют и составляют отчет об использовании конкретных системных ресурсов и предупреждают о возможных проблемах.

6.1.8 Инструмент поддержки конкретных потребностей тестирования (К1)

Оценка качества данных

Данные являются основой проектов, связанных с преобразованием и миграцией данных, а также хранилищ данных. Их атрибуты могут изменяться в зависимости от критичности и объема данных. Для проверки правил миграции и преобразования данных должны быть использованы подходящие инструменты оценки качества, обеспечивая достоверность, полноту и соответствие обработанных данных стандарту, выбранному исходя из конкретных условий.

Для тестирования практичности (удобства использования) существуют другие тестовые инструменты.

6.2 Эффективное использование инструментальных средств: выгоды и риски(K2)	20 минут

Терминология

Тестирование, управляемое данными, тестирование на основе ключевых слов, скриптовый язык

6.2.1 Выгоды и риски использования инструментальных средств тестирования (для всех средств) (K2)

Просто приобретение или аренда средства не гарантирует успех в работе с ним. Каждый тип средств может потребовать дополнительных затрат для достижения реальных и устойчивых выгод. При использовании средств существуют как возможные преимущества, так и риски.

Выгоды от использования инструментальных средств:

- Уменьшается повторяющаяся работа (например, запуск регрессионных тестов, повторный ввод одинаковых тестовых данных, проверка на соответствие стандартам программирования).
- Большая целостность и повторяемость (например, тесты, запускаемые средством в одной и той же последовательности, с одной и той же частотой, а также тесты на основе требований).
- Объективная оценка (например, статические измерения, покрытие).
- Простота доступа к информации о тестах или тестировании (например, статистика и графики, отображающие прогресс тестирования, доли инцидентов и производительность).

Риски, возникающие при использовании средств тестирования:

- Нереалистичные ожидания от использования средства (включая функциональность и простоту использования).
- Недооценка времени, стоимости и объемов работ на первоначальное внедрение средства (включая обучение и получение экспертизы извне).
- Недооценка времени и объемов работ, необходимых для получения значимого и устойчивого результата от использования средства (включая необходимость изменений в процессе тестирования и постоянного улучшения методики работы со средством).
- Недооценка затрат на поддержку тестов, генерируемых средством.
- Возложение излишних надежд на средство (замена проектирования тестов или использование автоматизированных тестов там, где ручное тестирование было бы уместней).
- Пренебрежение контролем версий тестов внутри средства.

- Пренебрежение вопросами взаимодействия между критическими средствами, такими как средства управления требованиями, средства контроля версий, средства управления инцидентами, средства управления дефектами, а особенно если эти средства от разных производителей.
- Риск того, что поставщик средства перестанет существовать, откажется от средства или продаст его иному поставщику.
- Слабая поддержка со стороны поставщика, в том числе по вопросам обновлений и исправлений дефектов.
- Риск приостановки бесплатного продукта или продукта с открытым кодом.
- Непредвиденные риски, как например, невозможность поддержки новой платформы.

6.2.2 Отдельные замечания для инструментов определенных типов (K1)

Средства выполнения тестов

Средства выполнения тестов выполняют тесты, используя автоматизированные тестовые скрипты. Этот тип средств обычно требует значительных затрат для получения серьезных результатов.

Создание тестов путем записи действий тестировщика кажется привлекательным, но данный подход не масштабируется для большого числа автоматизированных тестовых скриптов. Созданный таким образом скрипт – это линейное представление с конкретными данными и действиями внутри каждого скрипта. В случае возникновения непредвиденных событий, данный скрипт может оказаться нестабильным.

Тестирование на основе данных обычно использует входные тестовые данные в виде электронных таблиц и более универсальный скрипт, который может считывать данные и выполнять один и тот же тест с различными данными. Даже не знакомые со скриптовым языком тестировщики могут создавать тестовые данные для этих предопределенных скриптов.

В тестировании на основе данных существуют также и другие технические подходы где, например, вместо прописанных в таблицы комбинаций данные генерируются алгоритмами с конфигурируемыми параметрами времени выполнения. Например, средство может использовать алгоритм генерации случайного ID пользователя, и для повторяемости в пределах набора и для управления случайностью вводится некоторая неоднородность.

При тестировании на основе ключевых слов, в таблицах, помимо тестовых данных, содержатся ключевые слова, поясняющие действия, которые нужно предпринять (называемые также словами действия). Тестировщики, даже будучи не знакомыми с скриптовым языком, могут определять тесты по ключевым словам, которые можно подобрать под тестируемое приложение.

Для всех подходов необходим технический анализ скриптового языка (тестировщиками или специалистами по автоматизации тестов).

В любом случае при использовании скриптового языка ожидаемые результаты должны быть сохранены для последующего сравнения.

Средства статического анализа

Средства статического анализа, применимые к исходному коду, могут помочь соблюсти стандарты программирования, но если они применяются к существующему коду, то могут повлечь выдачу большого числа сообщений. Предупреждения не останавливают компиляцию исходного кода, а в идеале должны помогать поддерживать код в будущем. Эффективным подходом является постепенная реализация с начальными фильтрами для исключения некоторых сообщений.

Средства поддержки управления тестированием

Средства поддержки управления тестированием должны взаимодействовать с другими средствами или электронными таблицами так, чтобы получаемая информация была полезной и имела наиболее удобный для текущих нужд организации формат.

6.3 Внедрение инструментального средства в организацию (K1) — 15 минут

Терминология

Специфичных терминов нет.

Введение

Основные принципы внедрения средства в организацию включают:

- Исследование зрелости, сильных и слабых сторон организации и поиск возможностей для улучшенного процесса тестирования, поддерживаемого инструментальными средствами.
- Оценку на основании предъявляемых требований и объективных критериев.
- Опытную эксплуатацию, используя инструмент в течение оценочного периода для определения его способности эффективно функционировать с тестируемым продуктом внутри текущей инфраструктуры и для определения изменений, которые нужно внести в инфраструктуру для эффективного использования средства
- Оценку поставщика (включая обучение, поддержку и коммерческие аспекты) или поставщиков услуг поддержки в случае в случае некоммерческих продуктов.
- Определение внутренних требований к передаче знаний и обучению использования средств.
- Оценку потребности в обучении в соответствии с текущими навыками команды автоматизации тестов.
- Оценку соотношения затрат и выгод для конкретной ситуации

Внедрение выбранного средства в организацию начинается с пилотного проекта, целями которого являются:

- Узнать больше подробностей о средстве.
- Посмотреть, как средство сочетается с существующими процессами и практиками, и что должно будет измениться.
- Принять решение по стандартному использованию, управлению, хранению и поддержке средства и тестов (например, соглашения по именованию файлов и тестов, созданию библиотек и определению модульности тестовых наборов).
- Оценить, будут ли получены преимущества при разумных затратах.

Факторы успеха развертывания средства в организации включают:

- Постепенное распространение средства во всей организации.
- Адаптация и усовершенствование процессов для совместимости с использованием средства.

- Обеспечение обучения и руководства новых пользователей.
- Определение порядка использования.
- Реализация способа сбора опыта реальной эксплуатации средства.
- Мониторинг использования средства и преимуществ.
- Обеспечение поддержки команде тестировщиков для выбранного средства.
- Сбор извлеченных уроков от всех команд.

Ссылки

6.2.2 Buwalda, 2001, Fewster, 1999

6.3 Fewster, 1999

7 Ссылки

Стандарты

ISTQB Глоссарий терминов, используемых в тестировании ПО Версия 2.1

[CMMI] Chrissis, M.B., Konrad, M. and Shrum, S. (2004) CMMI, Guidelines for Process Integration and Product Improvement, Addison Wesley: Reading, MA

См. раздел 2.1

[IEEE Std 829-1998] IEEE Std 829 (1998) IEEE Standard for Software Test Documentation,

См. разделы 2.3, 2.4, 4.1, 5.2, 5.3, 5.5, 5.6

[IEEE 1028] IEEE Std 1028 (2008) IEEE Standard for Software Reviews and Audits,

См. раздел 3.2

[IEEE 12207] IEEE 12207/ISO/IEC 12207-2008, Software life cycle processes,

См. раздел 2.1

[ISO 9126] ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality,

См. раздел 2.3

Книги

[Beizer, 1990] Beizer, B. (1990) Software Testing Techniques (2nd edition), Van Nostrand Reinhold: Boston

См. разделы 1.2, 1.3, 2.3, 4.2, 4.3, 4.4, 4.6

[Black, 2001] Black, R. (2001) Managing the Testing Process (3rd edition), John Wiley & Sons: New York

См. раздел 1.1, 1.2, 1.4, 1.5, 2.3, 2.4, 5.1, 5.2, 5.3, 5.5, 5.6

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

См. раздел 6.2

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

См. разделы 2.2, 2.3, 4.2, 4.3, 4.4, 4.6

[Craig, 2002] Craig, Rick D. and Jaskiel, Stefan P. (2002) Systematic Software Testing, Artech House: Norwood, MA

См. разделы 1.4.5, 2.1.3, 2.4, 4.1, 5.2.5, 5.3, 5.4

[Fewster, 1999] Fewster, M. and Graham, D. (1999) Software Test Automation, Addison Wesley: Reading, MA

См. разделы 6.2, 6.3

[Gilb, 1993]: Gilb, Tom and Graham, Dorothy (1993) Software Inspection, Addison Wesley: Reading, MA

См. разделы 3.2.2, 3.2.4

[Hetzel, 1988] Hetzel, W. (1988) Complete Guide to Software Testing, QED: Wellesley, MA

См. разделы 1.3, 1.4, 1.5, 2.1, 2.2, 2.3, 2.4, 4.1, 5.1, 5.3

[Kaner, 2002] Kaner, C., Bach, J. and Petticord, B. (2002) Lessons Learned in Software Testing,

John Wiley & Sons: New York

8 Приложение A – Происхождение курса

История данного документа

Данный документ разрабатывался с 2004 по 2007 год членами рабочей группы, назначенными Международной Коллегией по Квалификации Тестировщиков Программного Обеспечения (ISTQB). Первоначально он рецензировался избранной группой специалистов, а позже – представителями международного сообщества тестировщиков. Правила, использованные при разработке данного документа, приведены в приложении C.

Документ является программой Международного Фонда Сертификации в Тестировании Программного Обеспечения, квалификацией первого уровня, утвержденной ISTQB (www.istqb.org).

Цели Базового уровня сертификации

- Дать возможность тестированию получить признание как необходимой профессиональной специализации в программной инженерии.
- Предложить стандартный подход к развитию карьеры специалиста по тестированию.
- 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 программной инженерии.
- Выявить актуальные и важные в индустрии темы тестирования.
- 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 просто лицам, заинтересованным в тестировании, получить международную сертификацию, признанную в мире.

Цели международной квалификации (адаптировано на основе материалов конференции в Соллентуне, ноябрь 2001)

- Дать возможность сравнивать навыки тестирования в различных странах
- Дать возможность специалистам по тестированию проще пересекать границы стран.

- Обеспечить международным проектам общее понимание вопросов тестирования.
- Повысить количество сертифицированных специалистов по тестированию во всем мире.
- Оказать большее влияние и поднять ценность в качестве международного, а не локального подхода
- Разработать общий базис для понятий и знаний по тестированию посредством программы и терминологии, а также повысить уровень знаний по тестированию для всех участников.
- Популяризировать тестирование как профессию в большем количестве стран.
- Предоставить специалистам по тестированию возможность получить квалификацию на родном языке.
- Наладить международный обмен знаниями и ресурсами.
- Обеспечить международное признание специалистов по тестированию и этой квалификации посредством вовлечения разных стран.

Базовые требования для сертификации

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

- Иметь минимальное представление либо о разработке программного обеспечения, либо о его тестировании, как, например, шестимесячный опыт работы в качестве специалиста по тестированию или разработчика программного обеспечения.
- Пройти аккредитованный ISTQB курс подготовки (или же курс подготовки одной из Коллегий, признанной ISTQB).

Предпосылки и история сертификации в области тестирования на базовый уровень

Независимая сертификация специалистов по тестированию началась в Великобритании в образованном Британским Компьютерным Сообществом Совете по Исследованию Информационных Систем (ISEB), когда в 1998 была создана Коллегия специалистов по тестированию ПО (www.bcs.org.uk/iseb). В 2002 году ASQF в Германии начала поддержку немецкой квалификационной схемы для специалистов по тестированию (www.asqf.de). Данный курс основан на программах ISEB и ASQF; он включает в себя реорганизованную, обновленную и дополненную программу, где упор сделан на аспекты, представляющие наибольшую практическую ценность для тестировщиков.

Существующая сертификация базового уровня в области тестирования (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 указана на самом сертификате.

Местные особенности учитываются признанной ISTQB национальной коллегией тестировщиков в каждой из стран-участниц. Обязанности национальных коллегий определены ISTQB и реализуются в каждой стране. В обязанности коллегий различных стран включается аккредитация образовательных учреждений и организация экзаменов.

9 Приложение B – Цели обучения / уровень знаний

Для данной программы определены представленные ниже цели обучения. В соответствии с целями обучения будет проведен экзамен по каждой теме.

Уровень 1: Запомнить (K1)

Кандидат осознает, запомнит и повторит термин или определение.

Ключевые слова: запомнить, найти, повторить, осознать, узнать

Пример

Кандидат может выбрать определение термина «сбой» из вариантов:

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

Уровень 2: Понять (K2)

Кандидат может указать причины или объяснения относящимся к теме положениям, и может обобщать, сравнивать, классифицировать, категоризировать и приводить примеры для понятий тестирования.

Ключевые слова: обобщать, абстрагировать, классифицировать, сравнивать, наносить, приводить антонимы, приводить примеры, интерпретировать, переводить, представлять, делать выводы и заключения, категоризировать, конструировать модели

Примеры

Объяснить причину, почему тесты должны быть разработаны как можно раньше:

- Чтобы найти дефекты, когда стоимость их устранения меньше
- Чтобы найти наиболее важные дефекты как можно раньше

Объяснить сходства и различия между интеграционным и системным тестированием:

- Сходства: тестирование более чем одного компонента и не функциональное тестирование,
- Различия: интеграционное тестирование сконцентрировано на интерфейсах и взаимодействиях, в то время как системное – на всей системе от начала и до конца.

Уровень 3: Применить (K3)

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

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

Пример

- Может идентифицировать граничные значения допустимых и недопустимых множеств.
- Может выбирать тестовые сценарии из диаграмм переходов, для того чтобы покрыть все переходы

Уровень 4: Анализировать (K4)

Кандидат может разделить информацию о процедуре или методе на составные части для лучшего понимания, а также различать факты и выводы. Типичный случай – анализ документа, программного продукта или ситуации в проекте, чтобы предложить подходящие решения.

Ключевые слова: анализировать, организовать, найти связи, интегрировать, схема, разобрать, структура, атрибут, дифференцировать, различие, различать, сфокусировать, выбрать

Пример

- Проанализировать риски продукта и предложить превентивные меры, а также меры по коррекции и смягчению возможных последствий
- Объяснить, какие части отчета об инцидентах основаны на фактах, а какие являются выводами из результатов

Ссылки

(Для когнитивных уровней целей обучения)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001)

Таксономия для обучения, преподавания и оценки: версия таксономии Блума для образовательных целей, Allyn & Bacon

10 Приложение C – Правила, применяемые к ISTQB

Основы Программы обучения

Правила, приведенные в этом разделе, были использованы во время создания и рецензирования данной Программы обучения.

10.1 Общие правила.

SG1. Программа обучения должна быть понятна людям с опытом работы в тестировании от 0 до 6 месяцев (или более). (6-Месяцев)

SG2. Программа обучения должна быть больше практической, чем теоретической. (Практичность)

SG3. Программа обучения должна быть понятной и однозначной. (Ясность)

SG4. Программа обучения должна быть понятной людям из разных стран и легко переводимой на различные языки. (Переводимость)

SG5. Программа обучения должна быть написана с использованием американского варианта английского языка. (Американский английский)

10.2 Содержание на настоящий момент

SC1. Программа обучения должна отражать современные подходы к тестированию и лучшие практики в тестировании программного обеспечения. Программа обучения должна подлежит пересмотру каждые 3 – 5 лет. (Актуальность)

SC2. Программа обучения должна минимизировать временные задержки, такие как текущие рыночные условия, оставаясь актуальной на протяжении 3-х – 5-ти лет. (Срок хранения)

10.3 Цели обучения

LO1. Изучаемый материал должен быть разделен на части, которые должны быть запомнены или быть узнаваемы (уровень К1), части, которые читатель должен понять (уровень К2), части, которые читатель сможет применить на практике (К3) и части, которые могут быть использованы читателем для анализа документа, программного обеспечения или состояния проекта (К4). (Уровень знаний)

LO2. Описание содержимого должно быть согласованным с целями обучения. (Согласованность с целями обучения)

LO3. Пример вопроса будет должен быть продемонстрирован в конце каждой важной части Программы обучения для иллюстрации целей обучения. (Цель обучения - пример)

10.4 Общая структура

ST1. Структура Программы обучения должна быть ясной и позволять ссылаться на материалы других частей, а также экзаменационные вопросы и другую сопутствующую документацию. (Перекрестные ссылки)

ST2. Пересечения между частями Программы обучения должны быть минимальны. (Пересечения)

ST3. Все главы Программы обучения должны иметь одинаковую структуру. (Постоянство структуры)

ST4. Программы обучения должна иметь версию, дату издания, и все страницы должны быть пронумерованы. (Версионность)

ST5. В Программе обучения должно быть указано количество времени, которое должно быть потрачено на каждую главу (чтобы отразить относительную важность темы). (Время на обучение)

10.5 Ссылки

SR1. Источники информации и ссылки будут предоставлены в Программе обучения для получения большего количества информации по темам. (Ссылки)

SR2. Больше информации должно быть предоставлено в Программе обучения в тех местах, где источники не ясны и не указаны явно. Например, все определения вынесены в глоссарий, так что в Программе обучения используются только термины. (Детали без ссылок)

Источники информации

Термины, используемые в Программе обучения, определены в Глоссарии терминов ISTQB, используемых в тестировании программного обеспечения.

Список рекомендованных книг по тестированию программного обеспечения предоставляется параллельно с Программой обучения. Основной список книг приведен в разделе Ссылки.

11 Приложение D – Замечания для обучающих организаций

Каждый заголовок большого раздела программы обучения содержит требуемое для его изучения время в минутах. Цель этого указания – дать рекомендации по относительному распределению времени, которое выделяется на каждый раздел аккредитованного курса, а также указать приблизительное минимальное время для изучения каждого раздела. Обучающие организации могут использовать времени больше, чем указано, и кандидаты также могут потратить больше времени на чтение и изучение материалов. Учебный план курса не обязательно должен следовать порядку, указанному в программе обучения.

Программа обучения содержит ссылки на установленные стандарты, которые должны быть использованы при подготовке материалов тренингов. Версия каждого используемого стандарта должна быть в точности той, которая указана в текущей версии программы обучения. Прочие материалы, шаблоны или стандарты, на которые нет ссылок в данной программе обучения, могут быть также использованы, но не будут входить в экзаменационный тест.

Практические упражнения необходимо включить в обучающие материалы для целей уровня К3 и К4.

12 Приложение E – Изменения

12.1 Версия 2010 года

1. Изменения в целях изучения(LO) включают следующие прояснения:
   a. Изменена формулировка целей изучения (содержание и уровень остались без изменений): LO-1.2.2, LO-1.3.1, LO-1.4.1, LO-1.5.1, LO-2.1.1, LO-2.1.3, LO-2.4.2, LO-4.1.3, LO-4.2.1, LO-4.2.2, LO-4.3.1, LO-4.3.2, LO-4.3.3, LO-4.4.1, LO-4.4.2, LO-4.4.3, LO-4.6.1, LO-5.1.2, LO-5.2.2, LO-5.3.2, LO-5.3.3, LO-5.5.2, LO-5.6.1, LO-6.1.1, LO-6.2.2, LO-6.3.2
   b. Пункт LO-1.1.5 претерпел изменение формулировки и усовершенствован до уровня К2. Вызвано ожидаемым сравнением между терминами, связанными с понятием дефекта.
   c. LO-1.2.3 (К2) Был добавлен. Содержание было охвачено в версии Программы обучения от 2007 года.
   d. LO-3.1.3 (К2) теперь объединяет содержание LO-3.1.3 и LO-3.1.4
   e. LO-3.1.4 убран из версии 2010 года, так как частично избыточен вместе с LO-3.1.3.
   f. LO-3.2.1 переформулирован для обеспечения соответствия содержанию Программы обучения версии 2010 года.
   g. LO-3.3.2 был изменен, а его уровень изменился с К1 на К2 для соответствия LO-3.1.2
   h. LO-4.4.4 изменен для обеспечения ясности, уровень был изменен с К3 на К4. Причина: LO-4.4.4 был написан в манере К4.
   i. LO-6.1.2 (К1) удален из версии 2010 года был заменен на LO-6.1.3, (К2). В версии 2010 года нет LO-6.1.2.
2. Согласованное применение термина «подход к тестированию» в соответствии с определением из глоссария. Термин «стратегия тестирования» не потребуется запоминать.
3. Глава 1.4 теперь содержит концепцию прослеживания связей между базисом тестирования и тестовыми сценариями.
4. Глава 2.x теперь содержит объекты тестирования и базис тестирования.
5. Повторное тестирование теперь является основным словарным термином, а не частью подтверждающего тестирования.
6. Аспект качества данных и тестирования был добавлен в некоторых главах Программы обучения: качество данных и риск в главах 2.2, 5.5, 6.1.8
7. Глава 5.2.3 Критерии входа вынесены в отдельную главу для совместного обсуждения с критериями выхода (критерии входа добавлены в LO-5.2.9)
8. Правильное применение терминов «стратегия тестирования» и «подход к тестированию» в соответствии с определением в глоссарии.

9. Глава 6.1 укорочена из-за слишком длинного для 45-минутного урока описания средств.
10. Выпущен стандарт IEEE Std 829:2008. Эта версия Программы обучения пока не учитывает эту новую версию. Глава 5.2 ссылается на главный план тестирования. Содержание главного плана тестирования охватывается содержимым плана тестирования на разных уровнях планирования: планы тестирования для уровней тестирования могут быть созданы так же, как и планы тестирования на проектном уровне покрывают различные уровни тестирования. Последнее называется главный план тестирования, как в самой программе обучения, так и в глоссарии ISTQB.
11. Кодекс этики был перемещен из CTAL в CTFL.

12.2 Версия 2011 года

Изменения, сделанные в промежуточном релизе 2011

1. Общее: термин «рабочая часть» заменен на «рабочая группа».
2. Термин пост-условие заменен на постусловие для обеспечения соответствия с глоссарием ISTQB версии 2.1.
3. Первое упоминание: ISTQB заменено на ISTQB®.
4. Предисловие «Программы обучения: описания необходимых уровней знаний» было удалено, так как являлось избыточным по отношению к Приложению B.
5. Глава 1.6: так как не преследовалось цели определять цель обучения для «Кодекса этики», уровень знаний был убран из этой секции.
6. Главы 2.2.1, 2.2.2, 2.2.3, 2.2.4, 3.2.3: исправлены проблемы с форматированием списков.
7. Глава 2.2.2: слово «отказ» не было верным для фразы «…изолировать отказы отдельного компонента…». Поэтому заменено на «дефект».
8. Глава 2.3: исправлено форматирование списка целей тестирования, относящегося к терминам в главе Типы Тестирования (K2).
9. Глава 2.3.4: изменено описание отладки для соответствия с глоссарием ISTQB версии 2.1.
10. Глава 2.4: удалено слово «обширное» из фразы «включает обширное регрессионное тестирование», потому что «обширное» зависит от изменения (размера, рисков, значения и прочего), как описано в следующем предложении.
11. Глава 3.2: слово «включая» было удалено для уточнения предложения.
12. Глава 3.2.1: Из-за неверного форматирования процесс рецензирования имел 12 главных действий вместо 6, как было положено. Число действий было изменено на 6, что делает главу соответствующей Программе обучения от 2007 года и Программе обучения Повышенного уровня от 2007 года.
13. Глава 4: слово «разрабатываемых» заменено словом «определяемых», так как тестовый сценарий может быть определен, а не разработан.
14. Глава 4.2: изменен текст для объяснения, как методы белого и черного ящиков могут быть использованы совместно с методом создания тестов на основе опыта.

15. Глава 4.3.5: формулировка «между участниками, включая пользователей и систему» заменена на «между участниками (включая пользователей и систему)».
16. Глава 4.3.5: формулировка «альтернативные пути» заменена на «альтернативные сценарии».
17. Глава 4.4.2: предложение, уточняющее цель тестирования ветвей, было изменено для разъяснения термина в тексте Главы 4.4.
18. Глава 6.1: заголовок «6.1.1 Понимание смысла и цели применения инструментов в тестировании (K2)» заменен на «6.1.1 Применение инструментов в тестировании (K2)».
19. Глава 7 / Книги: в списке указана третья редакция книги [Black, 2001], заменив вторую редакцию.
20. Приложение D: главы, требующие практических упражнений, были заменены на общее требование для всех целей обучения уровня K3 и выше. Это требование указано в Процессе Аккредитации ISTQB (версия 1.26).
21. Приложение E: Цели обучения, измененные между версиями 2007 и 2010 годов, сейчас отображены правильно.