Сертифицированный специалист по тестированию программного обеспечения

Программа базового уровня

Версия 2005

International Software Testing Qualifications Board

История изменений

Версия	Дата	Примечания
RSTQB 2007	27 Ноября 2007	Перевод на русский язык
ISTQB 2005	01 Июля 2005	Программа сертификации Специалиста по тестированию ПО, Программа 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, 25 февраля 1999

Содержание

Благодарность

Документ создан рабочей группой базового уровня International Software Testing Qualifications Board: Thomas Muller (председатель), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhajarvi, Geoff Thompson and Erik van Veendendal. Основная команда благодарит команду редакторов, а также все национальные коллегии за предложения.

Отдельные слова благодарности (Австрия) Anastasios Kyriakopoulos, (Дания) Klaus Olsen, Christine Rosenbeck-Larsen, (Германия) Matthias Daigl, Uwe Hehn, Tilo Linz, Horst Pohlmann, Ina Schieferdecker, Sabine Uhde, Stephanie Ulrich, (Индия) Vipul Kocher, (Израиль) Shmuel Knishinsky, Ester Zabar, (Швеция) Anders Claesson, Mattias Nordin, Ingvar Nordstrom, Stefan Ohlsson, Kennet Osbjer, Ingela Skytte, Klaus Zeuge, (Швейцария) Armin Born, Sandra Harries, Silvio Moser, Reto Muller, Joerg Pietzsch, (Великобритания) Aran Ebbett, Isabel Evans, Julie Gardiner, Andrew Goslin, Brian Hambling, James Lyndsay, Helen Moore, Peter Morgan, Trevor Newton, Angelina Samaroo, Shane Saunders, Mike Smith, Richard Taylor, Neil Thompson, Pete Williams, (США) Dale Perry.

Предисловие к программе

Цель данного документа

Документ представляет собой основу для международной сертификации на квалификацию «Специалист по тестированию ПО». Выдается The International Software Testing Qualifications Board (здесь и далее ISTQB) национальным сертифицирующим органом для аккредитации тренинговых провайдеров на местном языке. Тренинговые провайдеры создают программы курсов и определяют соответствующие методы учебы для аккредитации, также программа призвана помочь кандидатам при подготовке и экзамену.

История и дополнительные данные по программе могут быть найдены в дополнении А.

Сертифицированный специалист по программному обеспечению, базовый уровень.

Квалификация «Базовый уровень» предназначена для всех кто занимается тестированием. Это включает в себя следующие роли: тестировщик, тестовый аналитик, тест инженер, тест консультант, тест менеджер и разработчик. Квалификация Базовый уровень» также подходит каждому, кто желает получить представление о тестировании программного обеспечения, например руководителям проектов, менеджерам по качеству, бизнес аналитикам, руководителям программных проектов, ИТ директорам и консультантам по управлению. Сдав экзамен на квалификацию базового уровня вы сможете получить следующий уровень квалификации в тестировании.

Уровни понимания приведены для каждой секции программы.

- К1: Запомнить, вспомнить или узнать;
- К2: Понять, уяснить, обосновать, сравнить, классифицировать, подытожить;
- К3: Применить.

Детали и примеры целей изучения приведены в дополнении Б

Все термины, приведенные как «Термины» сразу после секционных заголовков, должны быть усвоены, даже если явно не упоминаются в целях обучения.

Экзамен

Сертификационный экзамен будет основан на программе. Ответы на экзаменационные вопросы могут потребовать использования материалов, которые содержатся более чем в одной секции программы. Вопросы будут по всем секциям программы.

Экзамен проводится в виде тестов.

Экзамен может быть частью аккредитованного курса обучения или может сдаваться отдельно.

Аккредитация

Тренинг провайдеры, курсы которых следуют данной программе, могут быть аккредитованы национальной коллегией зарегистрированной в ISTQB. Правила аккредитации должны быть получены от коллегии или от объекта, который проводит аккредитацию. Аккредитованный курс должен соответствовать данной программе, экзамен ISTQB может быть частью курса.

Уровень детализации

Уровень детализации программы обеспечивает идентичность процесса обучения и сертификации в разных странах. Для достижения этой цели программа состоит из следующих частей:

- Общие руководящие цели, описывающие идею программы.
- Список информации для изучения, включая описание и ссылки на внешние источники, если необходимо.
- Цели изучения для каждой области знаний, описывающие результат познавательного изучения и образ мышления, который должен быть достигнут.
- Список терминов, который обучающиеся должны помнить и понимать.
- Описание ключевых понятий для изучения, включая источники, такие как доступная литература и стандарты.

Содержание программы не является описанием всех знаний в области тестирования ПО; оно отражает тот уровень детализации, который должен быть достигнут на курсах обучения базового уровня.

Как организована программа

Есть шесть основных разделов. Заголовки верхнего уровня показывают уровни целей изучения, описанных в разделе, и указывают время для изучения.

Каждый раздел состоит из нескольких частей. Каждая часть также имеет цели изучения и требуемое время. Подчасти, не имеющие указанного времени, включаются во время изучения части в целом.

1 Основы тестирования (К2)

Цели изучения базового тестирования

Цели определяют, что Вы будете способны делать после окончания каждого из модулей.

1.1 Почему тестирование необходимо? (К2)

- Описать на примерах, как дефект в программном обеспечении может нанести вред человеку, оборудованию или компании (К2)
- Подчеркнуть различие между основой дефекта и его эффектом (К2)
- Приводя примеры, объяснить, почему важно проводить тестирование (К2)
- Приводя примеры рассказать, почему тестирование является частью процесса обеспечения качества и как тестирование повышает качество (К2)
- Объяснить термины: ошибка, дефект, неисправность и соответствующие термины (К1)

1.2 Что такое тестирование? (К2)

- Назвать основные цели тестирования (К2)
- Описать цели тестирования в процессе разработки программного обеспечения, поддержке и операциях, в контексте поиска дефектов, обеспечение полноты и информации, и предотвращение дефектов (К2)

1.3 Общие принципы тестирования (К2)

- Объяснить фундаментальные принципы тестирования (К2)

1.4 Базовый процесс тестирования (К1)

- Рассказать о основной деятельности по тестированию, начиная от планирования и заканчивая закрытием, основные задачи на каждой из фаз тестирования (К1)

1.5 Психология тестирования (К2)

- Напомнить, что на успех в тестировании также имеют влияние психологические факторы (К1):
  - четкие цели;
  - баланс между самостоятельным и независимым тестированием;
  - важность общения и обратной связи на каждый дефект
- Сопоставить способ мышления специалиста по тестированию и разработчика

1.1 Почему тестирование необходимо (К2)

Термины

Ошибка, дефект, отказ, сбой, качество, риск, программное обеспечение, тестирование.

1.1.1 Контекст программных систем (К1)

Программные системы проникают в нашу жизнь все глубже и глубже, начиная от бизнес-приложений (например, банковский сектор) и заканчивая потребительскими товарами (например, машины). У многих людей был опыт работы с программным обеспечением, которое работало не так, как ожидалось. Программное обеспечение, которое работает неправильно может привести к многочисленным проблемам, включая потерю денег, времени или деловой репутации, не говоря уже о ранении или смерти.

1.1.2 Причина дефектов программного обеспечения (К2)

Человек может сделать ошибку, которая приведёт к дефекту (ошибке, сбою) в коде в программного обеспечения, системе или в документе. Если дефект выполнился в коде, система не сделает того, что она должна сделать (или сделает что-то, чего она не должна делать), что в свою очередь приведет к отказу. Дефекты в программном обеспечении, системах или документах могут быть причиной отказа системы, но не всегда.

Дефекты происходят, потому что людям свойственно ошибаться, есть ограничения во времени, код может быть сложным, инфраструктура комплексной, технологии могут меняться, а взаимодействие систем нетривиальным.

Отказы могут быть вызваны условиями окружающей среды: радиация, магнетизм, электрические поля и загрязнение может привести к перебоям в программно-аппаратных средствах или повлиять на выполнение и работу программного обеспечения в следствии изменения условий работы аппаратного обеспечения.

1.1.3 Роль тестирования в разработке программного обеспечения, поддержке и функционировании (К2)

Тщательное тестирование систем и документов может помочь уменьшить риски связанные с возникновением проблем в эксплуатации и внести лепту в качество программного обеспечения, если найденные дефекты исправлены до того, как система начнет эксплуатироваться.

Тестирование программного обеспечения может быть необходимо для выполнения условий контракта, юридических требований или некоторых промышленных стандартов.

1.1.4 Тестирование и качество (К2)

С помощью тестирования возможно оценить качество в терминах найденных дефектов, как для функциональных так и для нефункциональных требований к программному обеспечению и его характеристикам (надежность, удобство использования, эффективность и удобство эксплуатации). Детальную информацию о функциональном и нефункциональном тестировании смотрите в главе 2; детальную информацию о характеристиках программного обеспечения смотрите в «Проектирование программного обеспечения – качество продуктов программного обеспечения» (ISO 9126).

Тестирование может дать уверенность в качестве программного обеспечения, если найдено мало дефектов или не найдено совсем. Правильно спроектированный тест, который выполняется успешно уменьшает общий уровень риска в системе. Когда же во время тестирования найдены дефекты, качество системы возрастает, после того как эти дефекты исправлены.

Однажды сделанные ошибки не должны повторятся. Посредством понимания главных причин дефектов найденных в предыдущих проектах, процесс может быть улучшен, что в свою очередь должно предотвратить появление этих дефектов вновь, как следствие, улучшить качество будущей системы.

Тестирование должно быть частью деятельности по обеспечению качества (т.е. вместе со стандартами разработки, тренингами и анализом дефектов).

1.1.5 Когда тестировать достаточно? (K2)

Принимая решение о достаточности тестирования необходимо принимать во внимание технические и деловые риски, ограничения проекта, а также время и финансовые возможности. (Риски обсуждаются дальше, в Главе 5).

Тестирование должно обеспечивать достаточной информацией заинтересованные стороны, для того чтобы принимать решение о выпуске программного обеспечения или системы, которая была в тестировании, для следующего витка разработки или передаче заказчику.

1.2 Что такое тестирование (K2)

Термины

Код, отладка, разработка (программного обеспечения), требования, рассмотрение, тестовый базис, тестовый сценарий, тестирование, цели тестирования.

Предварительные знания

Согласно общему представлению тестирование это только выполнение тестов, т.е. выполнение программы. В действительности это только часть.

Деятельность по тестированию существует как до, так и после выполнения самого теста, таких работ как планирование и контроль, выбор тестовых условий, проектирование тестовых сценариев и проверка результатов, оценивание критериев завершенности, отчетность о процессе и системе, которая тестируется, завершение или закрытие (после как фаза тестирования была выполнена). Помимо этого тестирование включает в себя рассмотрение документов (включая исходные коды) и статический анализ.

Методы динамического и статического тестирование используются для достижения схожих целей, как для улучшения тестируемой системы, так и разработки и процесса тестирования.

Можно выделить несколько целей тестирования:

- поиск дефектов;
- убедиться в надлежащем уровне качества и получение необходимой информации;
- предотвращение ошибок.

Процесс проектирования тестов на ранних стадиях жизненного цикла (проверка тестового базиса через проектирование теста) может помочь предотвратить появление дефектов в коде. Анализ документов (например, требований) также помогает предотвратить появление дефектов в коде.

Различные точки зрения в тестировании принимают во внимание различные цели. Например, в тестировании в процессе разработки (например, компонентное, интеграционное и системное тестирование), главной целью может быть поиск наибольшего количества отказов для того, чтобы дефекты в программном обеспечении были идентифицированы и исправлены. В приемочном тестировании главной целью является подтверждение того, что система работает как ожидалось, и соответствует требованиям. В некоторых случаях главной целью тестирования является оценка качества программного обеспечения (без намерения исправления ошибок), предоставление информации о рисках в данное время заинтересованным сторонам. Тестирование поддержки часто включает проверку на отсутствие новых ошибок, внесенных во время изменений. Во время функционального тестирования главной целью является определение характеристик системы, таких как надежность и работоспособность.

Отладка не является тестированием. Тестирование может показывать неисправности, причинами которых являются дефекты. Отладка это часть процесса разработки, которая включает обнаружение происхождения дефекта, испарвление кода и проверка того, что дефект был исправлен правильно. Последующее подтверждающее тестирование гарантирует, что исправление на самом деле избавило нас от сбоя. Разработчик несет ответственность за отладку, а тестировщик за тестирование.

Процесс тестирования и деятельности объясняется в Секции 1.4

1.3 Общие принципы тестирования (К2)

Термины

Исчерпывающее тестирование.

Принципы

Принципы тестирования были разработаны более 40 лет назад и дают общие руководства для всех видом тестирования.

Принцип 1 - Тестирование показывает наличие дефектов

Тестирование может показать, что дефекты существуют, но не может подтвердить, что дефектов нет. Тестирование уменьшает вероятность присутствия ненайденных дефектов в программном обеспечении. Даже если не было найдено ни одного дефекта, это не гарантирует что программа работает правильно.

Принцип 2 - Исчерпывающее тестирование невозможно

Протестировать всё (все комбинации входных данных и предусловий) невозможно, за исключением тривиальных случаев. Вместо исчерпывающего тестирования мы используем риск и приоритеты чтобы сосредоточить усилия.

Принцип 3 – Раннее тестирование

Тестирование должно начаться как можно раньше в цикле разработки программного обеспечения, и должно сосредотачиваться на определенных целях.

Принцип 4 – Кластеризация дефектов

Наибольшее количество дефектов содержится в нескольких модулях, в которых наблюдается наибольшее количество сбоев.

Принцип 5 – Парадокс «пестицида»

Повторяя один и тоже тест снова и снова, мы в какой-то момент перестанем находить новые дефекты. Для того чтобы избежать этого тестовые сценарии должны время от времени пересматриваться и перерабатываться, новые тесты должны разрабатываться для проверки других частей системы, чтобы потенциально найти как можно больше дефектов.

Принцип 6 – Тестирование контекстно-зависимо

Разные контексты требуют различных подходов. Например, подходы к тестированию приложений, для которых важны безопасность, отличаются от подходов тестирования приложений электронной коммерции.

Принцип 7 – Заблуждение об отсутствии ошибок

Поиск дефектов не имеет смысла, если система не стабильна или не удовлетворят требованиям и ожиданиям пользователей.

1.4 Базовый процесс тестирования (К1)

Термины

Подтверждающее тестирование, критерии выхода, прецедент, регрессионное тестирование, тестовый базис, тестовое условие, покрытие теста, тестовые данные, выполнение теста, тестовый протокол, тест план, стратегия тестирования, итоговый тестовый отчет, тестовое обеспечение.

Предварительные знания

Самое очевидное в тестировании это выполнение тестов. Но для того, чтобы быть эффективным, планы тестирования должны включать время на планирование, разработку, подготовку для выполнения тестов, а также оценку результатов.

Базовый процесс тестирования состоит из следующих главных активностей:

- планирование и контроль;
- анализ и проектирование;
- внедрение и выполнение;
- оценивание критериев выхода и отчетность;
- завершение тестирования.

Несмотря на то, что логически они последовательны, в процессе деятельности они могут происходить одновременно или накладываться друг на друга.

1.4.1 Планирование и контроль (К1)

В процесс планирования включаются определение целей и заданий тестирования, а также методов их достижения и выполнения.

Контроль тестирования включает деятельность по сравнению действительного состояния с планом, сообщение о статусе, включая отклонение от плана. А также включает корректирующие действия для достижения целей и выполнения заданий. Для контроля процесса тестирования мониторинг деятельности должен проходить в течении всего проекта. При планировании принимается во внимание информация полученная в процессе мониторинга и контроля.

Задачи процесса планирования тестирования:

- Определение возможных рисков и целей тестирования;
- Определение подхода к тестированию (методы, тестовые элементы, покрытие, идентификация и взаимодействие команд, оборудование);
- Определение необходимых ресурсов (люди, среда тестирования, компьютеры);
- Реализация политики и/или стратегии тестирования;
- Определение сроков для анализа и проектирования тестов;
- Определение сроков для реализации, выполнения и оценки тестов;
- Определение критериев выхода.

Задачи процесса контроля тестирования:

- Измерение и анализ результатов;
- Контроль и документирование прогресса, тестовое покрытие и критерии выхода;
- Корректирующие действия;
- Принятие решений.

1.4.2 Анализ тестирования и проектирование (К1)

Анализ и проектирование это действия, в процессе которых общие цели тестирования трансформируются в явные условия тестирования и модели.

Задачи процесса анализа тестирования и проектирования:

- Обзор базиса тестирования (требования, архитектура, дизайн, интерфейс);
- Определение условия тестирования, требований и тестовых данных, основываясь на анализе элементов тестирования, спецификациях, поведение и структуре;
- Проектирование тестов;
- Оценка тестируемости требований и системы;
- Проектирование среды тестирования, определение инфраструктуры и инструментария.

1.4.3 Разработка и выполнение тестов (К1)

Разработка и выполнение тестов это процесс, в котором тестовые условия трансформируются в тестовые сценарии, наборы тестов и сред выполнения.

Важнейшие задачи разработки и выполнения тестов:

- Разработка и установки приоритетов для тестовых сценариев, подготовка тестовых данных, тестовых процедур и, по необходимости, подготовка инструментария и сценариев автоматизированного тестирования;
- Создание тестовых наборов из тестовых сценариев для последующего эффективного выполнения;
- Проверка тестового окружения на предмет правильности настройки;
- Регистрация результатов выполнения тестов, версий программного обеспечения и особенностей тестового инструментария;
- Сравнение полученных результатов с ожидаемыми.
- Составление отчетов о непредвиденных отклонениях и их анализ для установления причин их происхождения (т.е. дефект в коде, в тестовых данных, в документации, или ошибка в способе выполнения теста)
- Повторение тестов, как результате действий, которые были предприняты в результате непредвиденных отклонений. Например, повторное выполнение тестов, которые не выполнились в предыдущий раз, для того, что бы подтвердить, что дефект был исправлен (подтверждающее тестирование), выполнение исправленных тестов и/или выполнение тестов для того, что бы гарантировать, что дефект не появился в тех областях программного обеспечения, которые не изменяли, или исправление дефекта не скрыло другие дефекты (регрессионное тестирование).

1.4.4 Оценка критериев выхода и отчетность (К1)

Оценка критериев выхода это процесс сопоставления результатов тестов и поставленных целей тестирования. Рекомендуется делать для каждого уровня тестирования.

Оценивание критериев выхода включает в себя следующие задачи:
- Проверка протоколов тестирования на предмет соответствия установленным целям;
- Оценка необходимости проведения дополнительного тестирования или изменения критериев выхода;
- Составление отчетов для заинтересованных сторон.

1.4.5 Завершение тестовой деятельности (К1)

В финальной фазе процесса тестирования происходит сбор данных о проведенных тестах, фактах и цифрах. Например, когда программная система выпущена в работу, проект тестирования выполнен (или отменён), контрольные точки проекта были достигнуты, или поддержка выпущенной версии закончена.

Важнейшие задачи процесса завершения тестовой деятельности:

- Проверка того, что из запланированного было действительно поставлено, закрытие отчетности или обсуждение открытых вопросов, документация о принятии системы.
- Архивирование тестового окружения, инфраструктуры тестирования для дальнейшего использования.
- Передача тестового обеспечения поддерживающей организации.
- Анализ результатов, которые могут быть использованы для следующих проектов, а также для повышения уровня зрелости

1.5 Психология тестирования (К2)

Термины
Независимое тестирование

Предварительные знания
Способ мышления, необходимый в процессе тестирования или рассмотрения, отличается от способа мышления, необходимого для программирования и анализа. С правильным мышлением разработчик может тестировать собственный код, но разделение этой ответственности обычно делается для того, чтобы помочь сфокусировать усилия и обеспечить дополнительные преимущества, такие как независимый взгляд тренированного профессионала в тестировании. Независимое тестирование может иметь место на любом из уровне тестирования.

Определенный уровень независимости (избегающий предвзятости автора) часто более эффективен в поиске дефектов и сбоев. Независимость, тем не менее, не является заменой знания системы, поэтому разработчики тоже могут эффективно находить дефекты в собственном коде.

Можно определить несколько уровней независимости:

- Проектирование тестов человеком, который написал программу (низкий уровень независимости).
- Проектирование тестов другим человеком (например, из команды разработчиков).
- Проектирование тестов человеком из другой организационной группы (например, независимая команда тестировщиков).
- Проектирование тестов человеком из другой организации или компании (например, аутсорсинг или сертификация другой компанией).

Люди как и проекты управляются целями. Люди склонны подстраивать свои планы согласно целям установленным руководством или заказчиками, например, поиск дефектов или подтверждение, того что программное обеспечение работает. Поэтому четкая установка целей очень важна для тестирования.

Нахождение сбоев программы во время тестирования может быть воспринята как критика продукта или автора. Поэтому, тестирование часто рассматривается как разрушительная деятельность, даже если она конструктивна с точки зрения управления рисками. Поиск неисправностей в системе требует любопытства, профессионального пессимизма, критического взгляда, внимания к деталям, коммуникационных навыков общения с разработчиками, и опыта на котором базируется интуитивный поиск ошибок.

Чувства неприязни между тестировщиками и аналитиками, дизайнерами и разработчиками можно избежать, если ошибки будут подаваться конструктивно. Это применимо как к рассмотрению, так и к тестированию.

Тестировщику, как и руководителю тестирования, необходимы навыки межличностного общения для обмена информацией о дефектах, прогрессе работы и рисках конструктивным способом. Для автора программного обеспечения или документа, информация о дефекте сможет повысить его мастерство. Дефекты, найденные и исправленные во время тестирования, помогут сохранить время, а потом и деньги, а также уменьшить риски.

Проблемы могут возникать, если тестировщики рассматриваются как люди, приносящие плохие новости. Тем не менее, существует несколько способов улучшить взаимоотношения между тестировщиками и разработчиками:

- Плохой мир лучше хорошей войны – напомнить каждому об общих целях улучшения качества системы;
- Общаться нейтральным способом, сфокусироваться на фактах, не критиковать, например, описать цели, факты и анализ того, что вы нашли;
- Попытаться понять, что чувствуют другие люди и почему они так реагируют;
- Убедиться, что другие люди поняли что вы сказали и наоборот.

Ссылки

1.1.5 Black, 2001, Kaner, 2002
1.2 Beizer, 1990, Black, 2001, Myers, 1979
1.3 Beizer, 1990, Hetzel, 1998, Myers, 1979
1.4 Hetzel, 1998
1.4.5 Black, 2001, Craig, 2002
1.5 Black, 2001, Hetzel, 1998

2 Тестирование в жизненном цикле программного обеспечения (К2)

Цели изучения тестирования в жизненном цикле программного обеспечения

Цели определяют, что Вы сможете делать по окончании каждого модуля.

2.1 Модели разработки программного обеспечения (К2)

- Понять связь между разработкой, тестированием и продуктами работы в жизненном цикле процесса разработки, привести примеры, основываясь на характеристиках проектов, продуктов и ситуаций (К2)
- Осознать тот факт, что модели разработки программного обеспечения должны быть адаптированы к характеристикам проекта и продукта (К1)
- Назвать причины для различных уровней тестирования и признаки хорошего тестирования в любой модели жизненного цикла (К1)

2.2 Уровни тестирования (К2)

- Сравнить различные уровни тестирования: объекты тестирования, цели тестирования (например, функционального или структурного) и работы связанные с ними, люди, которые тестируют, типы дефектов и сбоев, которые могут быть определены. (К2)

2.3 Типы тестирования (К2)

- Сравнить четыре типа тестирования (функциональное, нефункциональное, структурное и тестирование изменений на примерах) (К2)
- Подчеркнуть, что функциональное и структурное тестирование может иметь место на любом из уровней тестирования (К1)
- Определить и описать нефункциональные типы тестов, основанных на нефункциональных требованиях (К2)
- Определить и описать типы тестов, основанных на анализе структуры или архитектуры программной системы (К2)
- Описать цели подтверждающего и регрессионного тестирования (К2)

2.4 Тестирование сопровождения (К2)

- Сравнить тестирование сопровождения (тестирование существующей системы) и тестирование нового приложения в аспектах типов тестов, условий и объемов тестирования (К2)
- Определить причины тестирования сопровождения (изменения, перемещения и утилизация) (К1)
- Описать роль регрессионного тестирования и анализа последствий в тестировании сопровождения (К2)

2.1 Модели разработки программного обеспечения (К2)

Термины

Коммерческое готовое программного обеспечение (COTS), возрастающая модель разработки, уровень тестирования, валидация, верификация, V-модель

Предварительные знания

Тестирование не существует изолированно, работы по тестированию связаны с работами по разработке программного обеспечения. Различные жизненные циклы разработки программного обеспечения требуют различные подходов к тестированию.

2.1.1 V-модель (К2)

Несмотря на то, что существует несколько вариантов V-модели, общепринятый вариант включает 4 уровня тестирования, которые соответствуют 4 уровням разработки.

Четыре уровня, которые используются в данной программе, следующие:

- компонентное (модульное) тестирование;
- интеграционное тестирование;
- системное тестирование;
- приёмочное тестирование.

На практике V-модель может содержать больше, меньше или совсем другие уровни разработки и тестирования, в зависимости от проекта и программного продукта. Например, может быть интеграционное компонентное тестирование после компонентного тестирования и системное интеграционное тестирование после системного.

Продукты программной разработки (такие как бизнес-сценарии, варианты использования, спецификации требований, документация проектирования и код) создающиеся во время разработки, являются базисом для тестов для одного или нескольких уровней. Ссылки на основные продукты работы включают в себя «Усовершенствованную модель зрелости процессов производства ПО» (CMMI) и «Жизненный цикл программного обеспечения» (IEEE/IEC 12207). Верификация и валидация (а также ранняя разработка тестов) могут проводиться во время разработки программных продуктов.

2.1.2 Итерационные модели разработки (К2)

Итерационная разработка это процесс при котором создание требований, разработка, конструирование и тестирование являются серией небольших разработок. Например: прототипирование, быстрая разработка приложения (RAD), Rational Unified Process (RUP) и модели гибкой разработки. Прирост, созданный на каждой итерации, может быть протестирован на различных уровнях в процессе разработки. Значимость регрессионного тестирования возрастает начиная со второй итерации. Верификация и валидация осуществляется на каждой итерации.

2.1.3 Тестирование в модели жизненного цикла (К2)

В любой модели жизненного цикла ПО, есть несколько признаков хорошего тестирования:

- Для каждого процесса разработки существует соответствующий процесс тестирования
- Каждый уровень тестирования имеет свои цели, относящиеся именно к этому уровню
- Анализ и проектирование тестов для данного уровня должны начинаться во время соответствующего процесса разработки
- Тестировщики должны быть вовлечены в процесс анализа документов как только первый черновой вариант доступен в жизненном цикле разработки

Уровни тестирования могут быть скомбинированы или реорганизованы, в зависимости от свойств проекта или архитектуры системы. Например, для интеграции коммерческого готового программного

продукта (COTS) в систему, покупатель может выполнять интеграционное тестирование на системном уровне (например, интеграция в инфраструктуру и другие системы или развертывание системы) и приемочное тестирование (функциональное и нефункциональное, пользовательское и\или операционное тестирование).

2.2 Уровни тестирования (К2)

Терминология

Альфа-тестирование, бета-тестирование, компонентное тестирование (также известное как модульное или программное тестирование), контрактное приемочное тестирование, драйвера, испытание в условиях эксплуатации, функциональные требования, интеграция, интеграционное тестирование, нефункциональные требования, эксплуатационное (приемочное) тестирование, правовое приемочное тестирование, тестирование надежности, заглушки, системное тестирование, разработка управляемая тестированием, тестовая среда, приемочное тестирование пользователем

Предварительные знания

Для каждого уровня тестирования можно определить следующее: общие цели, тестовую основу, объект тестирования, типичные дефекты и сбои, которые могут быть найдены, требования к вспомогательному программному обеспечению для тестирования, подходы и ответственность.

2.2.1 Компонентное тестирование (К2)

Целью компонентного тестирования является поиск дефектов и проверка функциональности компонентов (модулей, программ, объектов, классов и т.д.), которые можно протестировать отдельно. Это можно сделать отдельно от системы, в зависимости от контекста жизненного цикла программного обеспечения и системы. Для этого могут быть использованы заглушки, драйвера, симуляторы.

Компонентное тестирование может включать функциональные и нефункциональные характеристики, такие как поведение ресурсов (например, утечки памяти) или тестирование надежности, а также структурное тестирование (например, покрытие ветвей). Тестовые сценарии можно получить из спецификации компонента, архитектуры или модели данных.

При компонентном тестировании обычно имеется доступ к коду, а также среде разработки, такой как, например, набор структур для компонентного тестирования или средства отладки, также на практике часто привлекают разработчика кода. Дефекты как правило исправляются сразу после нахождения без составления отчета об ошибке.

Один из подходов к компонентному тестированию это подготовка и автоматизация тестовых сценариев перед кодированием. Этот подход называется процессом разработки управляемым тестированием. Этот подход итеративный и основан на циклах разработки тестовых сценариев, потом сборки и интеграции небольших частей кода и выполнении компонентных тестов до достижения правильного результата.

2.2.2 Интеграционное тестирование (К2)

В процессе интеграционного тестирования проверяются интерфейсы между компонентами, взаимодействие различные частей системы, таких как операционная система, файловая система, аппаратное обеспечение и взаимодействие между ними.

Интеграционное тестирование может иметь несколько уровней, а также может проводиться с различными целями. Например:

- Компонентное интеграционное тестирование тестирует взаимодействие между компонентами системы и производится после компонентного тестирования.
- Системное интеграционное тестирование тестирует взаимодействие между различными системами и может иметь место после системного тестирования. В этом случае организация, которая ведет разработку, может контролировать только одну сторону интерфейса, поэтому изменения могут дестабилизировать. Бизнес-процессы, реализованные как технологический процесс, могут включать несколько систем. Проблемы переносимости могут иметь существенное значение.

Чем больше степень интеграции, тем сложнее определить принадлежность дефекта к тому или иному компоненту, что может привести к повышению степени риска.

Стратегии систематического интеграционного тестирования могут основываться на архитектуре системы, функциональных задачах, последовательностях обработки транзакций или каких либо других особенностях системы или компонента. Для того, чтобы свести к минимуму риск позднего нахождения дефектов, интеграция, как правило, должна быть итеративной и системной, а не случайной.

В интеграционное тестирование может быть включено нефункциональное тестирование (например, тестирование производительности).

На каждой стадии интеграции, тестировщики концентрируются исключительно на интеграции как таковой. Например, если интегрируется модуль А в модуль В – наиболее интересным с точки зрения тестирования является взаимодействие между модулями, а не функциональность каждого из модулей. Для тестирования могут быть использованы как функциональный, так и структурный методы.

В идеале, тестировщики должны понимать архитектуру и ее влияние на интеграционное планирование. Если интеграционные тесты запланированы до того, как компоненты или система разработаны, то они (компоненты или система) могут разрабатываться в порядке, который необходим для наиболее эффективного тестирования.

2.2.3 Системное тестирование (К2)

Системное тестирование делает упор на поведении системы/ПО в целом, как определено в рамках проекта или программы.

В системном тестировании тестовая среда должна максимально соответствовать рабочей или производственной среде выполнения, для того чтобы минимизировать риски нахождения дефектов, которые были пропущены во время тестирования.

Системное тестирование может включать тесты, основанные на рисках и\или спецификациях требований, бизнес-процессах, вариантах использования или других высокоуровневых описаниях поведения системы, взаимодействия с операционной системой или ее ресурсами.

В процессе системного тестирования должны исследоваться функциональные и нефункциональные требования к системе. Требования могут быть представлены как текст или модель. Тестировщикам необходимо иметь дело с неполными или недокументированными требованиями. Системное тестирование различных аспектов системы начинается с использования подходящих методик черного ящика. Например, на основе комбинаций бизнес-правил можно построить таблицу решений. Структурные подходы (методика белого ящика) можно использовать для оценки покрытия тестами всех структурных элементов, например, таких как структура меню или навигация на веб-странице. (см. Главу 4.)

Часто системное тестирование выполнят независимая команда тестирования.

2.2.4 Приемочное тестирование (К2)

Приемочное тестирование часто ложится на плечи заказчиков или пользователей системы; другие субъекты тоже могут принимать участие.

Целью приемочного тестирования является проверка работоспособности системы, части системы или специфичных нефункциональных характеристик системы. Поиск дефектов не является главной целью приемочного тестирования. Приемочное тестирование может оценивать готовность системны для промышленной эксплуатации и использования, хотя необязательно является последним и окончательным уровнем тестирования. Например, интеграционные тесты большой системы могут проходить после приемочных.

Приемочное тестирование может выполняться не только как один из уровней тестирования, но и как подуровень на нескольких других уровнях тестирования, например:

- Коммерческое готовое ПО может быть подвергнуто приемочному тестированию, когда ПО установлено или интегрировано.
- Приемочное тестирование удобства использования может происходить во время компонентного тестирования.
- Приемочное тестирование новой функциональности может происходить после системного тестирования.

Типичные виды приемочного тестирования включают:

Пользовательское приемочное тестирование
Обычно проверяет то, насколько система подходит пользователям.

Эксплуатационное (приемочное) тестирование
Обычно проверяет то, насколько система подходит системным администраторам:

- тестирование резервирования/восстановления;
- тестирование восстановления после сбоев;
- управление пользователями;
- поддержка заданий;
- периодическая проверка уязвимостей безопасности.

Контрактное и правовое приемочное тестирование
Контрактное приемочное тестирование выполняется для сопоставление требований предъявляемых контрактом к программному обеспечению. Критерии приема должны быть установлены в процессе согласования контракта. Приемочное тестирование на соответствие стандартам выполняется для проверки соответствия стандартам, таким как государственные, юридическим или стандартам безопасности.

Альфа- и бета-тестирование
Разработчики рыночного или готового программного обеспечения часто хотят услышать мнение существующих или потенциальных пользователей системы перед тем, как продукт начнет продаваться. Альфа-тестирование выполняет в организации-разработчике. Бета-тестирование выполняется заказчиком на собственных мощностях. Оба вида тестирования выполняются потенциальными пользователями, а не разработчиками.

Могут также использоваться и другие термины, такие как производственное приемочное тестирование, приемочное тестирование на площадке заказчика для систем, которые тестируют до или после установки у заказчика.

2.3 Типы тестов: цели тестирования (К2)

Термины

Автоматизация, тестирование методом черного ящика, покрытие кода, подтверждающее тестирование, функциональное тестирование, тестирование возможности взаимодействия, нагрузочное тестирование, тестирование удобства эксплуатации, тестирование производительности, тестирование переносимости, регрессионное тестирование, тестирование надежности, тестирование безопасности, тестирование на основе спецификации, стрессовое тестирование, структурное тестирование, тестовый набор, тестирование удобства использования, тестирование методом белого ящика

Предварительные знания

Группа работ по тестированию может быть направлена на проверку системы (или части системы), основываясь на определенных целях или причинах тестирования.

Тип теста определяется конкретной целью тестирования, которая может быть тестированием функции ПО; нефункциональной характеристикой, такой как надежность и удобство использования, структура или архитектура ПО или системы; или относиться к изменениям, т.е. подтверждение того, что дефекты были исправлены (подтверждающее тестирование) и поиск непреднамеренных изменений (регрессионное тестирование).

Модель ПО может быть разработана и\или использована в структурном и функциональном тестировании. Например, в функциональном тестировании - модель потока процессов, модель переходов состояний или обычная языковая спецификация; для структурного тестирования - модель контроля процессов или модель структуры меню.

2.3.1 Тестирование функций (функциональное тестирование) (К2)

Функции, которые должны выполняться системой, подсистемой или компонентом, могут быть описаны как спецификация требований, варианты использования, или функциональные спецификации, или же быть недокументированы. Функцией является то, что система делает.

Функциональные тесты основываются на функциях и характеристиках (которые могут быть описаны в документации или поняты тестировщикам), могут быть выполнены на всех уровнях тестирования (тесты компонентов могут основываться на спецификации компонентов).

Методика, основанная на использовании спецификаций, может быть использована для извлечения тестовых условий и тестовых сценариев из функциональности ПО или системы. (см. Главу 4). Функциональное тестирование рассматривает внешнее поведение системы (тестирование методом черного ящика).

Один из вариантов функционального тестирования - тестирования безопасности, исследует функции (например, брандмауэра) в контексте угроз, таких как вирусы и другие вредоносные программы.

2.3.2 Тестирование характеристик ПО (нефункциональное тестирование) (К2)

Нефункциональное тестирование включает в себя, но не ограничивается следующими видами тестирования: тестирование производительности, нагрузочное тестирование, стрессовое тестирование, тестирование удобства использования, тестирование возможности взаимодействия, эксплуатационное тестирование, тестирование надежности и тестирование переносимости. Это тестирование того, как система работает.

Нефункциональное тестирование может выполняться на всех уровнях тестирования. Термин «нефункциональное тестирование» описывает, какие тесты необходимы для измерения характеристик системы и ПО в соответствии с различными шкалами, такими как например время ответа ПО. Эти тесты могут ссылаться на модели качества, например определенные в ‘Разработка программного обеспечения – Качество программного продукта’ (ISO 9126).

2.3.3 Тестирование структуры и архитектуры ПО (К2)

Структурное тестирование (методом белого ящика) может выполняться на всех уровнях тестирования. Структурную методику лучше всего применять после методики основанной на использовании спецификаций, для измерения охвата, посредством измерения покрытия структуры.

Покрытие - это часть структуры, которая была выполнена в процессе тестирования, выраженная в процентах покрытых элементов. Если покрытие не равняется 100%, тогда нужно разработать дополнительные тесты для пропущенных элементов, и как следствие увеличить покрытие. Методики покрытия описаны в главе 4.

На всех уровнях тестирования, а в особенности в компонентном и интеграционном компонентном тестировании, можно пользоваться специальным инструментарием для измерения покрытия элементов кода, таких как команды и решения. Структурное тестирование может основываться на архитектуре системы, например иерархии вызовов.

Методы структурного тестирования могут быть использованы на системном, системном интеграционном и приемочном уровнях (например, бизнес-модели и структуры меню).

2.3.4 Тестирование, относящееся к изменениям (подтверждающее и регрессионное тестирование) (К2).

Когда дефект обнаружен и исправлен, тогда ПО должно быть протестировано вновь для того, чтобы подтвердить, что основной дефект был успешно удален. Это называется подтверждающим тестированием. Отладка (исправление дефектов) это процесс разработки, а не процесс тестирования.

Регрессионное тестирование – повторяющееся тестирование уже протестированных программ после изменений, для обнаружения любых дефектов, внесенных или не обнаруженных в результате изменений. Эти дефекты могут быть или в ПО , которое тестируется, или в других, связанных или не связанных компонентах ПО. Выполняется после изменений ПО или среды. Глубина регрессионного тестирования основывается на риске пропуска дефектов в ПО, которое уже работало.

Тесты должны быть повторимыми, если они предназначены для подтверждающего тестирования или являются вспомогательными в регрессионном тестировании.

Регрессионное тестирование может иметь место на всех уровнях тестирования и применяться к функциональному, нефункциональному и структурному тестированию. Наборы регрессионных тестов могут выполняться много раз и, как правило, развиваются медленно, таким образом, регрессионное тестирование является твердым кандидатом на автоматизацию.

2.4 Тестирование сопровождения (K2)

Термины

Анализ влияния, тестирование сопровождения, миграция, модификации, изъятие из эксплуатации.

Предварительные знания

Однажды развернутая программная система служит года или десятилетия. С течением времени система и рабочая среда часто корректируются, изменяется или расширяется. Тестирование сопровождения происходит на существующей операционной системе и управляется модификациями, миграциями или выведением из эксплуатации ПО или системы.

Модификации включают запланированные улучшения, исправления и срочные изменения, изменения рабочей среды, такие как обновления версий операционной системы, базы данных или исправления в системе безопасности.

Миграционное тестирование сопровождения (миграция с одной платформы к другую) должно включать эксплуатационные испытания новой рабочей среды, также как измененное ПО.

Тестирование сопровождения изъятия системы из эксплуатации может включать тестирование переноса данных или архивирование, если необходим длительный период хранения данных.

В дополнение к тестированию того что было изменено, тестирование сопровождения включает интенсивное регрессионное тестирование тех частей системы, которые не подвергались изменениям. Область, которая будет включена в регрессионное тестирование, напрямую зависит от риска изменений в этой области, размера системы и масштаба изменений. В зависимости от изменений регрессионное тестирование может иметь место на любых уровнях тестирования и любых типах тестов.

Определение того, как существующая система может реагировать на изменения, называется анализом влияний и используется для того, чтобы решить, какое регрессионное тестирование необходимо выполнить.

Тестирование сопровождения может бытьсложным, если спецификации устарели или утрачены.

Ссылки

2.1.3 CMMI, Craig, 2002, Hetzel, 1998, IEEE 12207
2.2 Hetzel, 1998
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, 1998
2.3.4 Hetzel, 1998, IEEE 829
2.4 Black, 2001, Craig, 2002, Hetzel, 1998, IEEE 829

3 Статические методики (К 2).

Цели изучения статических методик

Цели определяют, что Вы сможете делать по завершению каждого модуля.

3.1. Анализ и процесс тестирования (К2)

- Узнать, какие продукты разработки ПО можно проверять с помощью различных статических методик. (К1)
- Описать важности и достоинства статических методик и их применение для оценки качества программных систем. (К2).
- Объяснить разницу между статическими и динамическими методами (К2).

3.2. Процесс анализа (К2)

- Назвать фазы, роли и ответственности типичного процесса формальной экспертизы(К1).
- Объяснить разницу между различными типами экспертизы: неформальная экспертиза, техническая экспертиза, сквозной контроль и инспекция (К2)
- Объяснить факторы успешного выполнения экспертизы (К2)

3.3 Статический анализ с помощью инструментальных средств (К2)

- Описать цели статического анализа и сравнить его с динамическим тестированием (К2)
- Назвать типичные дефекты и ошибки, которые могут быть найдены с помощью статического анализа, сравнить их с экспертизами и динамическим тестированием (К1)
- Перечислить типичные преимущества статического анализа (К1)
- Перечислить типичные ошибки кода и проектирования, которые могут быть определены с помощью инструментальных средств статического анализа (К1)

3.1 Анализ и процесс тестирования (K2)

Термины
Динамическое тестирование, экспертизы, статический анализ.

Предварительные знания
В процессе статического тестирования ПО не запускается; ПО анализируется вручную (экспертизы) или автоматически (статический анализ).

Экспертизы – это способ тестирования ПО (включая код) и может выполняться перед динамическими тестами. Дефекты, обнаруженные в процессе экспертизы в начале жизненного цикла, довольно часто дешевле исправить, чем те дефекты, которые были обнаружены во время прохождения тестов (например, дефекты, найденные в требованиях).

Экспертиза может быть полностью ручной, но также существуют инструментальные средства. Главная часть ручного процесса – это проверка и комментирование ПО. Экспертизе можно подвергнуть практически все: спецификации требований, спецификации проектирования и кода, планы тестирования, спецификации тестирования, тестовые сценарии, тестовые скрипты, руководства пользователя и веб-страницы.

Преимущества экспертизы заключаются в раннем обнаружении дефектов и связей, повышении продуктивности разработки, уменьшении времени разработки, уменьшении стоимости и времени тестирования и улучшении общения. Экспертиза может найти упущения, например, в требованиях, которые, как правило, не находятся в процессе динамического тестирования.

Экспертиза, статический анализ и динамическое тестирование имеют одну цель – нахождение дефектов. Они дополняют друг друга: различные методики находят различные типы дефектов эффективно и более рационально. В отличии от динамического тестирования, экспертиза находит, как правило, дефекты, в то время как динамическое тестирование - сбои.

Типичные дефекты, которые легче найти в процессе экспертизы, нежели в процессе динамического тестирования: отклонение от стандартов, дефекты требований, дефекты проектирования, неэффективная поддержка и некорректные спецификации интерфейсов.

3.2 Процесс экпертизы (K2)

Термины

Критерии входа, критерии выхода, формальная экспертиза, неформальная экспертиза, инспектирование, открытие, метрики, координатор/руководитель инспектирования, экспертная оценка, эксперт, экспертное совещание, процесс экспертизы, секретарь, техническая экспертиза, сквозной контроль.

Предварительные знания

Форма экспертизы может варьироваться от неформальной до строго формальной (т.е. хорошо структурированной и регулируемой). Формальность процесса экспертизы связана с такими факторам, как зрелость процесса разработки вообще, разные юридические или правовые требования или требования аудита.

Способ экспертизы зависит от установленных целей экспертизы (например, поиск дефектов, достижение понимания, дискуссия или решение посредством консенсуса)

3.2.1 Фазы формальной экспертизы (K1)

Процесс формальной экспертизы, как правило, включает следующие фазы:

- Планирование: выбор участвующих, определение ролей, определение критериев входа и выхода для более формального типа экспертизы (т.е. инспекции); выбор той части документа, которая будет рассматриваться.
- Открытие: распространение документов; объяснение целей, процесса и документов участникам; проверка критериев входа (для более формального типа экспертизы).
- Индивидуальная подготовка: работа, которая должна быть проделана каждым участником в процессе подготовки к экспертному совещанию, определение потенциальных дефектов, вопросы, комментарии.
- Экспертное собрание: обсуждение или запись с документированием результатов (для более формального типа экспертизы). Участники собрания могут просто отмечать дефекты, давать рекомендации для их исправления или принимать решения.
- Переработка: исправление найденных дефектов обычно выполняется автором.
- Завершение: проверка того, что дефекты были направлены, сбор метрик и проверка критериев выхода (для более формального типа экспертизы).

3.2.2 Роли и ответственные (K1)

Процесс типичной формальной экспертизы включает в себя следующие роли:

- Менеджер: принимает решения о прохождении экспертизы, выделяет время в проектном графике и определяет, были ли достигнуты цели.
- Координатор: человек, который руководит процессом экспертизы документа или набора документов, включая планирование, прохождение совещания и дальнейшими шагами после совещания. Если необходимо, координатор может выступать посредником между различными точками зрения и часто удачный результат зависит именно от координатора.
- Автор: автор или главный ответственный за документ, который будет обсуждаться.
- Эксперт: человек со специальным техническим образованием или знанием предметной области (также называется проверяющий или инспектор), который после необходимой подготовки определяет и описывает найденные в процессе экспертизы дефекты в ПО. Эксперт должен быть выбран таким образом, чтобы представлять отличную от автора точку зрения и роль в процессе; присутствие эксперта является обязательным.
- Секретарь: документирует все проблемы и вопросы, которые были определены во время совещания.

Взгляд на документы с различных ракурсов и использование контрольных листов может сделать экспертизу более эффективной и продуктивной. Например, контрольные листы основанные на точках зрения пользователя, специалиста сопровождения, тестировщика или функциях, или контрольные листы типичных проблем спецификаций требований.

3.2.3 Типы экспертизы (K2)

Один документ может быть предметом для обсуждения для нескольких совещаний. Порядок может меняться в зависимости от типа экспертизы. Например, неформальная экспертиза может осуществляется перед технической экспертизой, или инспекция может иметь место перед спецификацией требований и перед сквозным контролем с заказчиками. Основными характеристиками и целями общих типов экспертизы являются:

Неформальная экспертиза
Ключевые характеристики:
- Отсутствие формального процесса;
- Может быть в форме парного программирования или анализа проектирования или кода техническим руководителем;
- Частично может документироваться;
- Эффективность сильно зависит от того, кто проводит экспертизу;
- Главная цель: результат при минимуме затрат.

Сквозной контроль
Ключевые характеристики:
- Совещанием руководит автор;
- Сценарии, пробные прогоны, группы равноправных участников;
- Нет ограничений по времени;
- По выбору подготовка к совещанию, отчет о проведенной работе, список того, что было найдено, присутствие секретаря (не автора);
- На практике может быть как формальным, так и полностью неформальным;
- Главные цели: изучение, достижение понимания, поиск дефектов.

Техническая экспертиза
Ключевые характеристики:
- Документируется, определен процесс поиска дефектов, в который вовлечены как члены команды, так и технические эксперты;
- Может проходить без участия руководства;
- В идеале, ведется подготовленным модератором;
- Подготовка к совещанию;
- По выбору используются контрольные листы, отчетность о совещании, список того, что было найдено, участие руководства;
- На практике может быть как формальной, так и полностью неформальной;
- Главные цели: обсудить, принять решения, оценить альтернативы, найти дефекты, решить технические проблемы и проверить на соответствие спецификациям и стандартам.

Инспектирование
Ключевые характеристики:
- Ведется подготовленным координатором (не автором);
- Как правило, бывает в виде экзамена;
- Определены роли;
- Используются метрики;
- Формальный процесс основывается на правилах и контрольных листах с критериями входа и выхода;
- Подготовка к совещанию;
- Отчет о проведенной работе, список найденного;
- Формальный процесс последующих шагов;
- По выбору – улучшение процесса и рецензент;

- Главная цель: поиск дефектов.

3.2.4 Факторы для успешной экспертизы (K2)

Факторы успешной экспертизы включают:

- Четкие определенные цели;
- В целях экспертизы вовлечены соответствующие люди;
- Найденные дефекты объективно обсуждаются;
- Учитываются психологические аспекты (например, извлечение положительного опыта для автора);
- Используются те методики экспертизы, которые наиболее подходят к типу и уровню, а также продукту ПО;
- Если нужно, для увеличения эффективности идентификации дефектов используются контрольные листы и роли;
- Проводятся тренинги по методикам экспертизы, в особенности в более формальных методиках, таких как инспекция;
- Руководство поддерживает хороший процесс экспертизы, например, выделяет время в графике разработки проекта;
- Понимание важности изучения и улучшения процесса.

3.3 Статический анализ с помощью инструментальных средств (K2)

Термины
Компилятор, сложность, поток управления, поток данных, статический анализ.

Предварительные знания
Целью статического анализа является поиск дефектов в исходных кодах и моделях ПО. Статический анализ выполняется без запуска ПО, ПО проверяется с помощью инструментальных средств; динамическое тестирование выполняет программный код. Статический анализ определяет дефекты, которые сложно найти в процессе тестирования. С помощью статического анализа (как и экспертизы), как правило, находятся ошибки, а не сбои. Инструментальные средства статического анализа анализируют программный код (например, поток управления или данных), также как сгенерированный код, например html xml.

Преимуществами статического анализа являются:

- Раннее определение дефектов, предшествующее выполнению тестов;
- Раннее предупреждение о подозрительных аспектах кода или проектирования, путем вычисления метрик, таких как повышенная сложность;
- Нахождение дефектов, которые сложно найти в процессе динамического тестирования;
- Определение зависимостей и нарушений целостности в моделях в ПО, например, ссылок;
- Улучшенное сопровождение кода и моделей;
- Предотвращение дефектов, путем усвоения уроков полученных во время разработки.

Типичные дефекты, которые обнаруживаются инструментальными средствами статического анализа:

- Ссылка на переменную с неопределенным значением;
- Несоответствие интерфейса между моделями и компонентами;
- Переменные, которые никогда не были использованы;
- Никогда не выполняемый (мертвый) код;
- Нарушение стандартов программирования;
- Нарушение безопасности;
- Нарушение синтаксиса кода и моделей ПО.

Инструментальные средства статического анализа обычно используются разработчиками (проверка на соответствие определенным правилам или стандартам программирования) перед, а также во время тестирования компонентов и интеграционного тестирования, и проектировщиками во время моделирования ПО. Инструментальные средства статического анализа могут генерировать большое количество предупреждений, которые должны быть хорошо управляемы для более эффективного использования инструментальных средств.

Компиляторы иногда имеют встроенную поддержку для статического анализа кода, включая также и вычисление 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 Методика разработки тестов (К3)

Цели изучения методик разработки тестов

Цели определяют, что Вы будете уметь по окончании изучения каждого модуля.

4.1 Определение тестовых условий и создание тестовых примеров (К3).

- Провести различие между спецификацией разработки тестов, спецификацией тестовых сценариев и спецификацией тестовых процедур (К1).
- Сравнить термины: тестовые условия, тестовые сценарии и тестовые процедуры (К2).
- Написать тестовые сценарии: (К3)
  - Показывающие четкое следование требованиям.
  - Содержащие ожидаемые результаты.
- Перевести тестовые сценарии в хорошо структурированную спецификацию тестовой процедуры в соответствии с уровнем знаний тестеров. (К3)
- Написать график выполнения для данных тестовых сценариев, учитывая приоритеты, и технические и логические зависимости. (К3)

4.2 Категории методов разработки тестов (К2)

- Вспомнить что, каждый из подходов к написанию тестовых сценариев, на основе спецификации (черный ящик) и на основе структурны (белый ящик), применимы, и перечислить общие методики для каждого. (К1)
- Пояснить характеристики и различия между тестированием на основе спецификаций, структурным тестированием и тестированием основанным на опыте. (К2)

4.3 Методики тестирования, основанного на спецификации, или метод «черного ящика» (К3)

- Написать тестовые сценарии по данной программной модели используя следующие методики: (К3)
  - Эквивалентное разбиение.
  - Анализ граничных значений.
  - Таблицы решений.
  - Диаграммы перехода состояний.
- Понять основную цель каждой из методик, какой уровень или тип тестирования может использовать эту методику, и как может быть измерено покрытие. (К2)
- Понять концепцию тестирования по тестовым сценариям и ее преимущества. (К2)

4.4 Методики структурного тестирования или метод «Белого ящика» (К3)

- Описать концепцию и важность покрытия кода. (К2)
- Разъяснить концепцию покрытия команд и решений, понять, что эти концепции также могут быть использованы на других уровнях тестирования, кроме компонентного тестирования (например, на бизнес-процедурах на системном уровне). (К2)
- Написать тестовые сценарии по данной управляющей логике, используя следующие методы: (К3)
  - Тестирование команд.
  - Тестирование решений.
- Оценить полноту покрытия команд и решений. (К3)

4.5 Методики тестирования, основанного на опыте (К2)

- Вспомнить причины для написания тестовых сценариев, основанных на интуиции, опыте и знаниях про распространенные ошибки. (К1)
- Сравнить метод тестирования основанный на опыте и метод тестирования основанный на спецификации. (К2)

4.6 Выбор тестовой методики (К2)

- Перечислить факторы, которые влияют на выбор той или иной методики проектирования тестов для конкретного типа проблем, такие как тип системы, риск, требования заказчика, модели для имитации вариантов использования, модели требований или знания тестировщиков. (K2)

4.1 Определение тестовых условий и создание тестовых сценариев (К3).

Термины

Тестовые сценарии, спецификация тестовых сценариев, тестовое условие, тестовые данные, спецификация тестовой процедуры, тестовый скрипт, трассируемость.

Предварительные знания

Процесс идентификации тестовых условий и создания тестов состоит из нескольких шагов:

- Разработка тестов посредством определения тестовых условий.
- Определение тестовых сценариев.
- Определение тестовых процедур.

Процесс может быть пройден несколькими путями, начиная с очень свободного - с отсутствием или малым количеством документации, и заканчивая очень формализованным (как описано в этой секции). Уровень формальности зависит от контекста тестирования, включая организацию, зрелость процесса тестирования и разработки, временных ограничений и вовлеченных людей.

В процессе разработки теста анализируется базовая тестовая документация, чтобы понять, что нужно тестировать, т.е. определить тестовые условия. Тестовые условия определяются как элемент или событие, которое может быть проверенно одним или несколькими тестовыми сценариями (например функция, транзакция, качественная характеристика или структурный элемент).

Возможность проследить путь от тестовых условий к спецификации и требованиям позволяет определить как влияние на анализ, когда меняются требования, так и охват требований, который определяется набором тестов. В процессе разработки тестов детальный подход к тестированию разрабатываться, основываясь, среди прочего, на найденных рисках. (См. Главу 5 для более полной информации об анализе рисков).

В процессе разработки спецификации для тестовых сценариев разрабатываются и детально описываются, основываясь на методиках разработки тестов, тестовые сценарии и данные. Тестовый сценарий состоит из набора входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанных для покрытия определенных условий. «Стандарт документации тестирования ПО» (IEEE 829) описывает содержание спецификаций разработки тестов и тестовых сценариев.

Ожидаемые результаты должны быть разработаны как часть спецификации тестового сценария и должны включать выходные данные, изменения в данных и состояниях, и любые другие результаты теста. Если ожидаемый результат не был определен, тогда правдоподобный, но ошибочный результат может быть интерпретирован как правильный. Идеальный вариант – определить ожидаемый результат до выполнения теста.

Тестовые сценарии располагаются в порядке их выполнения; это спецификация процедуры тестирования. Тестовая процедура (или ручной тестовый скрипт) определяет последовательность действий для выполнения теста. Если тест запускается с помощью инструментария для выполнения тестов, то последовательность действий определяется в тестовом скрипте (автоматизированнойы тестовой процедуре).

Различные тестовые процедуры и автоматизированные тестовые скрипты последовательно объединяются в план выполнения тестов и определяют в каком порядке тестовые процедуры, и возможно тестовые скрипты должны выполняться, когда они должны быть выполнены и кем. План выполнения тестов учитывает такие факторы как регрессионные тесты, приоритеты, технические и логические зависимости.

4.2 Категории методик разработки тестов (K2)

Термины

Методика черного ящика, методика основанная на опыте, методика основанная на спецификации, методика основанная на структуре, методика белого ящика.

Предварительные знания

Цель методики разработки тестов это определить тестовые условия и тестовые сценарии.

Классическим разделением методик тестирования является разделение на «Метод черного ящика» и «Метод белого ящика». Метод «черного ящика» (его также называют метод тестирования по спецификации) это способ получить и выбрать тестовые условия или тестовые сценарии, основываясь на анализе документации, неважно функциональной или не функциональной, для компонента или системы безотносительно к внутренней структуре. Метод «белого ящика» (его также называют структурным или основанным на структуре) основывается на анализе внутренней структуры компонента или системы.

Некоторые методики можно сразу отнести к одной категории; другие имеют признаки нескольких категорий. В этом курсе к методу «черного ящика» относится методика основанная на спецификации и опыте, а к методу «белого ящика» относится методика основанная на структуре.

Характерные особенности методики основанной на спецификации:

- Модели, как формальные, так и нет, используются для определения решаемой проблемы для программного обеспечения или его компонента.
- Из этих моделей систематично получаются тестовые сценарии.

Характерные особенности методики основанной на структуре:

- Информация о том, как сконструировано программное обеспечение, используется для получения тестовых сценариев, например, код или модель.
- Степень покрытия может быть измерена для существующих тестовых сценариев, в дальнейшем тестовые сценарии можно получать систематично, увеличивая покрытие.

Характерные особенности методики основанной на опыте:

- Знания и опыт людей используются для получения тестовых сценариев:
  - Используются знания тестировщиков, разработчиков, пользователей и других заинтересованных сторон о программном обеспечении, его использовании и условиях эксплуатации.
  - Знания про возможные ошибки и их распространение.

4.3 Методика основанная на спецификации или метод «черного ящика» (К3)

Термины
Анализ граничных значений, тестирование таблицы решений, эквивалентное разбиение, тестирование переходов состояний, тестирование сценариев использования.

4.3.1 Эквивалентное разбиение (К3)

Входные данные программы или системы разбиваются на группы, от которых ожидается одинаковое поведение, т.е. их можно обрабатывать по единому сценарию. Эквивалентные сегменты (классы) могут быть найдены как для верных, так и для неверных данных, то есть значений, которые могут быть отброшены. Сегменты также могут быть определены для выходных значений, внутренних значений, значений изменяющихся со временем (например, до или после события) или для параметров интерфейсов (например, в процессе интеграционного тестирования). Тесты могут быть разработаны, чтобы покрывать сегменты. Эквивалентное разбиение (ЭР) применяется на всех уровнях тестирования.

Эквивалентное разбиение это методика, которая позволяет добиться покрытия входных и выходных значений. Она может быть применена для значений, которые вводятся человеком, значений вводимых посредством интерфейса системы, или параметров интерфейса в интеграционном тестировании.

4.3.2 Анализ граничных значений (К3)

Поведение на границах каждого эквивалентного сегмента зачастую некорректно, границы это места, в которых тестирование с большей вероятностью соберет урожай ошибок. Максимальное и минимальное значение сегмента это граничные значения. Граничные значения для правильного сегмента – это верные граничные значения; граничные значения для неправильного сегмента – неверные граничные значения. Тесты могут быть разработаны так, чтобы покрывать правильные и неправильные сегменты. Когда разрабатываются тестовые сценарии, выбираются значения для каждой границы.

Анализ граничных значений может быть применен на всех уровнях тестирования. Его достаточно просто применять, а способность находить дефекты велика; детальные спецификации здесь сыграют незаменимую роль.

Эта методика часто рассматривается, как расширение метода эквивалентного разбиения и также может быть использована для значений, вводимых человеком, например для временных или табличных граничных значений. Граничные значения также можно использовать для выбора тестовых данных.

4.3.3 Тестирование с использованием таблиц решений (К3)

Таблицы решений это правильный путь для сбора требований к системе, которая содержит логические условия, и для документирования внутренней структуры системы. Они могут быть использованы для записи сложных бизнес правил, которые должна реализовывать система. Спецификация анализируется и определяются условия и действия системы. Входные условия и действия, как правило устанавливаются как верные или ложные (Булево выражение). Таблицы решений содержат инициирующее условие, часто как комбинацию верности или ложности для всех входных условий, и результирующие действия для каждой комбинации условий. Каждая колонка таблицы соответствую бизнес-правилу и определяет уникальную комбинацию условий, которая приводит к действию, связанному с этим правилом. Стандарт покрытия часто используется с таблицами решений для того, чтобы иметь как минимум один тест для колонки, который, как правило, приводит к покрытию всех комбинаций инициирующих условий.

Преимущество тестирования с использованием таблиц решений в том, что оно создает комбинации условий, которые не могут быть проверены тестированием иным способом. Оно может применяться во всех случаях, когда поведение системы зависит от нескольких логических решений.

4.3.4 Тестирование переходов состояний (К3)

Система может по-разному реагировать на действия в зависимости от текущего состояния и истории (истории состояний). В этом случае этот аспект системы может быть показан как диаграмма перехода состояний. Это позволяет тестировщику посмотреть на систему с точки зрения состояний, перехода между состояниями, входных данных или событий, которые влекут за собой переход из одного состояний в другое, и действий, которые могут происходить в этих состояниях. Состояния системы или объекта, которые тестируются, различны, идентифицируемы и конечны. Таблица состояний показывает связь между состояниями и входными данными, а также возможные переходы, которые неверны. Тесты могут быть разработаны так, чтобы покрыть типичные переходы состояний, каждое состояние, выполнение каждого перехода, особой последовательности переходов состояний или недействительных переходов.

Тестирование переходов состояний широко используется в индустрии встраиваемого программного обеспечения и технической автоматизации вообще. Однако, этот подход можно использовать и для моделирования бизнес-объектов которые имеют особые состояния, или тестирования переходов экранных диалогов (например, интернет-приложения или бизнес-сценарии).

4.3.5 Тестирование сценариев использования (К2)

Тесты могут создаваться на основе сценариев использования или бизнес-сценариев. Сценарий использования показывают взаимодействие между действующими лицами, включая пользователей и систему, которая (система) создает выходные данные для пользователя. Каждый сценарий использования имеет предварительные условия, которые должны соблюдаться для того он работал правильно. Каждый сценарий использования заканчивается выходными условиями, которые являются видимыми результатами и конечным состоянием системы, после того как сценарий использования завершен. Как правило, сценарий использования имеет главный сценарий (наиболее вероятный) и иногда альтернативные ветки.

Сценарий использования описывает «течение процесса» в системе, основываясь на ее типичном использовании, таким образом, тестовые сценарии, полученные из сценариев использования, полезны для нахождения дефектов в последовательности операций процесса при реальном использовании системы. Сценарии использования очень полезны для разработки приемочных тестов при участии заказчика или пользователей. Они также могут быть полезны при нахождении интеграционных дефектов вызванных взаимодействием интерфейсов различных систем, которые не могут быть выявлены при раздельном тестировании этих систем.

4.4 Структурный подход или методика «белого ящика» (К3)

Термины

Покрытие кода, покрытие решений, покрытие команд, структурное тестирование, тестирование основанное на структуре, тестирование методом белого ящика.

Предварительные знания

Структурное тестирование или тестирование «белого ящика» основывается на уже определенной структуре программы или системы, как показано на следующих примерах:

- Компонентный уровень: структура самого кода, то есть команд, решений или веток.
- Интеграционный уровень: структурой может быть дерево вызовов (диаграмма, в которой одни модули вызывают другие).
- Системный уровень: структурой может быть структура меню, бизнес-процессы или структура веб-страницы.

В этой секции обсуждаются два относящиеся к покрытию кода метода – основанный на командах и оснований на решениях. Для тестирования решений может быть использована диаграмма потоков управления для визуализации альтернатив для каждого решения.

4.4.1 Тестирование и покрытие команд (К3)

В компонентном тестировании покрытие команд оценивается в процентах выполняемых команд, которые используются набором тестовых сценариев. Тестирование команд использует тестовые сценарии для выполнения отдельных команд, таким образом увеличивая степень покрытия.

4.4.2 Тестирование решений и покрытие (К3)

Покрытие решений, относится к тестированию ветвлений, и оценивается как процент результатов решений (например Истина или Ложь для условий Если), которые были обработаны в процессе выполнения теста. Тестирование решений использует тестовые сценарии для обработки отдельных результатов решений, таким образом увеличивая степень покрытия.

Тестирование решений это метод тестирования управляющей логики, так как он обеспечивает прохождение процесса управления через точки решения. Подход покрытия решений более мощный, чем подход покрытия команд: 100% покрытие решений гарантирует 100% покрытие команд, но не наоборот.

4.4.3 Другие структурные подходы (К1)

Существуют более мощные методы структурного покрытия, чем покрытие решений, например, покрытие условий или многократное покрытие условий.

Концепция покрытия также может быть использована и на других уровнях тестирования (например, интеграционный уровень), где процент модулей, компонентов или классов которые обрабатываются набором тестовых сценариев, может быть выражен в покрытии модулей, компонентов или классов.

При структурном тестировании кода полезно пользоваться соответствующим инструментарием.

4.5 Методика основанная на опыте (К2)

Термины
Предположение об ошибке, исследовательское тестирование.

Предварительные знания
Возможно, наиболее широко распространенной методикой является предположение о наличии ошибки. Тесты создаются на основе интуиции и опыта работы тестировщика с похожими приложениями и системами. При использовании для усиления систематического тестирования, интуитивное тестирование может быть использовано для нахождения особых тестов, которые невозможно получить с помощью формальных методов, в особенности, когда это применяется после использования формальных методов. Однако, эффективность метода очень сильно зависит от опыта тестировщиков. Структурный подход при методики предположения об ошибке - это составление списка возможных ошибок и создание тестов, которые нацелены на их воспроизведение. Эти списки дефектов или сбоев могут быть построены на опыте, доступных данных о дефектах или сбоях, а также из общих знаний о том, почему программа дает сбой.

Исследовательское тестирование – это параллельное проектирование тестов, выполнение тестов, изучение и регистрация тестов, основанное на испытательной таблице, содержащей цели тестирования и выполняемое в заданные временные рамки. Этот подход наиболее эффективен, когда спецификаций мало, либо они неполные и наблюдается нехватка времени, или для того, чтобы усилить или дополнить другие, более формальные методики. Этот тип тестирования может служить для проверки тестового процесса, чтобы удостовериться, что наиболее важные дефекты найдены.

4.6 Выбор методики тестирования (K2)

Термины
Нет специфичных терминов.

Предварительные знания
Выбор методики для тестирования зависит от множества факторов, включая тип системы, правовые стандарты, требования клиентов или контракта, степень риска, тип риска, цель тестирования, доступная документация, уровень знаний тестировщиков, время и бюджет, жизненный цикл разработки, модели сценариев использования и предыдущий опыт по типам найденных дефектов.

Некоторые методики более всего подходят для определенных ситуаций и уровней тестирования; другие подходят для всех уровней тестирования.

Ссылки

4.1 Craig, 2002, Hetzel, 1998, 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 (К3)

Цели изучения управления тестированием

Цели определяют, что Dы будете уметь по окончании изучения каждого модуля.

.

5.1 Организация тестирования (К2)

- Пойнять важность независимого тестирования (К1)
- Перечислить преимущества и недостатки независимого тестирования внутри организации (К2)
- Назвать различных членов команды, которые могут быть задействованы для создания команды тестирования (К1)
- Перечислить типичные задачи руководителя тестирования и тестировщика (К1)

5.2 Планирование тестов и оценка трудозатрат (К2)

- Определить различные уровни и цели планирования тестов (К1)
- Подвести итог по целям и содержанию тест плана, спецификации проектирования тестов и документации по тестовым процедурам в соответствии со «Стандартом по Тестовой Документации для Программного Обеспечения» (IEEE829) (К2)
- Вспомнить характерные факторы, которые влияют на объем работ, относящийся к тестированию (К1)
- Разделить две концептуально различные оценки трудозатрат: методика, основанная на метриках, и методика, основанная на экспертной оценке (К2)
- Разделить цели планирования тестирования в проекте: для отдельных уровней тестирования (например, системное тестирование) или отдельных целей тестирования (например, тестирование удобства использования), и для выполнения тестов (К2)
- Перечислить задачи подготовки и выполнения тестов, которые нуждаются в планировании (К1)
- Определить/обосновать соответствующие критерии выхода для определенных уровней тестирования и групп тестовых сценариев (например, для интеграционного тестирования, приемочного тестирования или тестовых сценариев для тестирования удобства использования) (К2)

5.3 Отслеживание и контроль прогресса тестирования (К2)

- Вспомнить общие метрики, которые используются для отслеживания подготовки и выполнения тестов (К1)
- Вспомнить и объяснить метрики тестирования для отчетов о тестировании и контроля тестирования (например, найденные и исправленные дефекты и выполненные успешно и неуспешно тесты) (К2)
- Подвести итог по целям и содержанию общего отчета о тестировании в соответствии со «Стандартом по Тестовой Документации для Программного Обеспечения» (IEEE829) (К2)

5.4. Управление конфигурациями (К2)

- Подвести итог, как управление конфигурациями помогает тестированию (К2)

5.5. Риски и тестирование (К2)

- Описать риск, как возможную проблему, которая подвергает угрозе достижение одной или более целей участников проекта (К2)
- Вспомнить, что риски характеризуются вероятностью (возникновения) и воздействием (урон, возникающий в случае возникновения) (К1)
- Определить разницу между рисками проекта и продукта (К2)
- Определить типичные риски продукта и проекта (К1)
- Описать, используя примеры, как анализ рисков и управление рисками может быть использовано для планирования тестирования (К2)

5.6. Управление отказами(К3)

- Описать содержание отчета прецедентов согласно «Стандарту по Тестовой Документации для Программного Обеспечения» (IEEE829)
- Написать отчет по прецеденту, который описывает наблюдение неисправности во время тестирования (К3)

5.1 Организация тестирования (К2)

Термины
Тестировщик, руководитель тестирования, тест менеджер.

5.1.1 Организация тестирования и независимость (К2)

Эффективность поиска дефектов во время тестирования и анализа может быть улучшена с помощью независимых тестировщиков. Альтернативы для независимости следующие:

- Независимые тестировщики внутри команды разработчиков;
- Независимая команда тестирования или группа внутри организации, отчитывающаяся руководству проекта или исполнительному руководству;
- Независимые тестировщики из бизнес-организации, сообщества пользователей или ИТ;
- Независимые специалисты тестирования для отдельных целей тестирования, такие как тестировщики удобства использования, тестировщики безопасности или тестировщики сертификации (которые сертифицируют ПО на соответствие стандартам и правилам);
- Независимые тестировщики, привлеченные на аутсосорсинг, или сторонние по отношению к организации.

Для больших, сложных или критичных с точки зрения безопасности проектов обычно лучше иметь различные уровни тестирования, где некоторые или все уровни тестирования выполняются независимыми тестировщиками. Разработчики могут участвовать в тестировании, особенно на нижних уровнях, но их недостаток объективности часто ограничивает их эффективность. Независимые тестировщики могут иметь право требовать и определять процессы тестирования и правила, но тестировщики должны принимать такие роли только в случае недвусмысленного разрешения поступать таким образом. Выгода независимости включает:

- Независимые тестировщики видят другие, отличающиеся дефекты, и они беспристрастны;
- Независимые тестировщики могут проверять предположения людей, сделанные во время спецификации и внедрения системы.

Недостатки включают:
- Изолированность от команды разработчиков (если тестировщики считаются абсолютно независимыми);
- Независимые тестировщики могут быть узким местом, как последняя контрольная точка;
- Разработчики теряют чувство ответственности за качество.

Задания тестирования могут быть сделаны людьми, которые имеют конкретные роли по тестированию, или кем-то в других ролях, например, руководителем проекта, менеджером по качеству, разработчиком, экспертом в области применения ПО, инфраструктуре или ИТ.

5.1.2 Задачи руководителя тестирования и тестировщика (К1)

В этом курсе описаны две позиции в тестировании: руководитель тестирования и тестировщик. Действия и задачи, выполняемые людьми этих двух ролей, зависят от проекта, содержания продукта, людьми в ролях и организацией.

Иногда руководителя тестирования называют тест менеджером или координатором тестирования. Роль руководителя тестирования может быть выполнена руководителем проекта, руководителем разработки, менеджером по качеству или руководителем группы тестирования. В больших проектах могут существовать две позиции: руководитель тестирования и тест-менеджер. Обычно руководитель

тестирования планирует, отслеживает и контролирует деятельность и задачи по тестированию, как определено в главе 1.4.

Типичные задания руководителя тестирования могут включать:

- Координирование стратегий тестирования и планов с руководителями проекта и другими людьми;
- Написание или анализ стратегии тестирования для проекта и тестовой политики для организации;
- Согласование перспектив тестирования с другими процессами в проекте, такими как интеграционное планирование;
- Планирование тестирования - обдумывание содержания и понимание рисков – включая выбор методов тестирования, оценка временных трудозатрат, стоимости тестирования, наличия ресурсов, определение уровней тестирования, циклов, методов и целей и планирование управления отказами;
- Инициирование написания спецификаций, подготовки, внедрения и выполнения тестов, отслеживание и контроль выполнения;
- Планирование адаптации, основываясь на результатах тестирования и прогресса (иногда задокументированного в отчетах) и принятие каких-либо решений и действий, необходимых для решения проблем;
- Установка верного управления конфигурациями обеспечения тестирования для трассируемости;
- Введение подходящих метрик для измерения прогресса тестирования и для измерения качества тестирования ПО
- Решение, что должно быть автоматизировано, до какого уровня и как;
- Выбор средств автоматизации тестирования и организация подготовки тестировщиков для использования этих средств;
- Решение о внедрении среды тестирования;
- Планирование тестов;
- Написание отчетов тестирования, основываясь на информации, собранной во время тестирования.

Типичные задачи тестировщиков могут включать:

- Экспертиза и содействие в планах тестирования;
- Анализ, экспертиза и оценка требований пользователя, спецификаций и моделей для тестирования;
- Создание спецификаций тестов;
- Установка среды тестирования (часто совместно с системными администраторами и руководством по сетям);
- Подготовка и получение тестовых данных;
- Внедрение тестов на всех уровнях тестирования, выполнение и запись тестов, оценка результатов и документирование отклонения от ожидаемых результатов;
- Использование средств администрирования или управления тестированием и средств для отслеживания тестирования, если есть необходимость;
- Автоматизация тестов (может быть выполнена при поддержке разработчика или эксперта по автоматизации тестов);
- Измерение производительности компонентов или систем;
- Экспертиза тестов, разработанных другими тестировщиками.

Люди, которые работают над анализом тестов, проектированием тестов, определенных типов тестов или автоматизации тестов, могут быть специалистами в этих ролях. В зависимости от уровня тестирования и рисков, относящихся к продукту и проекту, различные люди могут принимать на себя роль тестировщика, но при этом оставляя за собой некий уровень независимости. Обычно тестировщиками на уровне компонентов и интеграции могут быть разработчики, тестировщиками на приемочном уровне тестирования могут быть бизнес-эксперты и пользователи, а тестировщиками на эксплуатационном приемочном тестировании будут операторы.

5.2 Планирование тестирования и оценка трудозатрат (К2)

Термины

Критерии входа, критерии выхода, исследовательское тестирование, подход к тестированию, уровень тестирования, тест план, тестовая процедура, стратегия тестирования

5.2.1 Планирование тестирования (К2)

Этот раздел описывает цель планирования тестирования внутри проектов разработки и внедрения и для процесса поддержки. Планирование может быть задокументировано в проектном или эталонном тест плане и в отдельных планах тестирования для уровней тестирования, таких как системное тестирование и приемочное тестирование. Описание документации планирования тестов описывается в «Стандарте по Тестовой Документации для Программного Обеспечения» (IEEE829).

На планирование влияет тестовая политика организации, цели тестирования, риски, ограничения, критичность, тестируемость и наличие ресурсов. Чем дальше проект и планирование тестирования развивается, тем больше информации в наличии и тем больше деталей может быть включено в план.

Планирование тестирования – непрерывный процесс и выполняется во всех процессах и действиях жизненного цикла. Обратная связь от результатов тестовой деятельности используется для определения изменения рисков, таким образом, чтобы планирование можно было корректировать.

5.2.2 Действия по планированию тестирования (К2)

Действия по планированию тестирования могут включать:

- Определение общего подхода к тестированию (стратегии тестирования), включая определение уровня тестирования, критериев входа и выхода;
- Интегрирование и координирование действий тестирования с жизненным циклом ПО: приобретение, поставка, разработка, работа и поддержка;
- Принятие решений о том, что тестировать, какие роли будут выполнять тестирование, когда и как тестирование должно быть сделано, как результаты тестирования будет оценены, и когда останавливать тестирование (критерии выхода);
- Назначение ресурсов для различных имеющихся задач;
- Определение количества, уровней детализации, структуры и шаблонов для тестовой документации;
- Выбор метрик для отслеживания и контроля подготовки и выполнения тестирования, исправления дефектов и проблем рисков;
- Установка уровней детализации для тестовых процедур для предоставления достаточной информации, чтобы поддерживать повторяемость подготовки и выполнения тестирования.

5.2.3 Критерии выхода (К2)

Целью критериев выхода является определение момента, когда необходимо остановить тестирование, как, например, по окончанию уровня тестирования или когда набор тестов имеет определенную цель.

Обычно критерии выхода могут состоять из:

- Степень законченности, например, покрытия кода, функциональности или риска;
- Оценка плотности дефектов или измерения надежности;
- Стоимость;
- Остаточные риски, такие как неисправленные дефекты или недостаток тестового покрытия какой-либо области;
- График, который основывается на времени выхода ПО на рынок.

5.2.4 Оценка трудозатрат в тестировании (K2)

В этом курсе раскрываются два метода оценки трудозатрат в тестировании:

- Оценка трудозатрат в тестировании, основываясь на метриках или идентичных проектах, или основываясь на обычных значениях;
- Оценка заданий владельцами этих заданий или экспертами.

Как только оценка трудозатрат тестирования выполнена, могут быть определены существующие ресурсы и создано расписание.

Работы по тестированию могут зависеть от ряда факторов, включая:

- Характеристику продукта: качество спецификаций и другой информации, используемой для моделей тестирования (т.е. основы тестирования), размер продукта, сложность предметной области, требований надежности и безопасности и требований к документации.
- Характеристику процесса разработки: стабильность организации, использование средств разработки, процесс тестирования, квалификация вовлеченных людей и временные ограничения.
- Результат тестирования: количество дефектов и объем работы, которую необходимо переделать.

5.2.5 Методы тестирования (стратегии тестирования) (K2)

Одним из способов классификации методов тестирования или стратегии основывается на моменте времени, в котором основная работа по проектированию тестов началась:

- Предупредительные подходы, когда тесты проектируются как можно раньше.
- Реактивные подходы, когда проектирование тестов начинается после того, когда ПО или система были созданы.

Типичные подходы или стратегии включают:

- Аналитические методы, такие как тестирование, основанное на рисках, когда тестирование направлено на области наибольшего риска.
- Методы моделей, такие как стохастическое тестирование с использованием статистической информации об уроне неисправности (такие как модели возрастания надежности) или использования (такие как функциональный разрез).
- Методические подходы, основанные на неисправности (включая угадывание ошибок и атаку на неисправности), контрольных листах и характеристиках качества.
- Подходы, основанные на соответствии процессам или стандартам, такие, например, которые определены согласно индустриальным стандартам или различным гибким методологиям.
- Динамические и эвристические подходы, такие как исследовательское тестирование, когда тестирование динамично реагирует на запланированные события, и когда выполнение и оценка – два параллельных задания.
- Консультативные подходы, как например те, где тестовое покрытие управляется в основном советами и указаниями экспертов в технологиях и/или предметных областях, которые не входят в команду тестирования.
- Регрессионные подходы, которые включают повторное использование существующих материалов тестирования, всестороннюю автоматизацию функциональных регрессионных тестов и стандартных наборов тестов.

Различные подходы могут быть скомбинированы, например динамический подход, основанный на рисках.

Выбор методов тестирования должен рассматривать обстоятельства, включая:

- Риск провала проекта, риск сбоя продукта, риск неисправности продукта по вине человека, рабочее окружение и компанию.
- Опыт людей в выбранной методике, инструментарии и методах.
- Цель предпринимаемого тестирования и задачу команды тестирования.
- Регулирующие аспекты, такие как внешние и внутренние правила процесса разработки.
- Тип продукта и бизнеса.

5.3 Мониторинг и контроль прогресса тестирования (К2)

Термины

Плотность дефектов, уровень отказов, контроль тестирования, тестовое покрытие, мониторинг тестирования, отчет о тестировании.

5.3.1 Мониторинг прогресса тестирования (К1)

Целью мониторинга тестирования является предоставление результата и обзора тестового процесса. Информация, которая отслеживается, может быть собрана вручную или автоматически и может быть использована для измерения критериев выхода, таких как покрытие. Метрики могут быть использованы для оценки прогресса по сравнению с запланированным расписанием и бюджетом. Обычные тестовые метрики включают в себя:

- Процентное соотношение проделанной работы по подготовке тестовых сценариев (или процентное соотношение подготовленных и запланированных тестовых 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);
- Покрытие тестированием 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 Отчеты о тестировании (К2)

Отчеты о тестировании имеют отношение к предоставлению итоговой информации о работах по тестированию, включая:

- Что произошло во время тестирования, например, даты, когда критерии выхода были достигнуты.
- Проанализированная информация и метрики для поддержки рекомендаций и решений о последующих действиях, таких как оценка оставшихся дефектов, экономическое обоснование продолжения тестирования, оставшиеся риски и уровни уверенности в протестированном ПО.

План результирующего отчета о тестировании предоставлен в «Стандарте по Тестовой Документации для Программного Обеспечения» (IEEE829).

Метрики должны быть собраны во время и в конце уровня тестирования для оценки:

- Верности целей тестирования для этого уровня тестирования;
- Правильности выбора метода тестирования;
- Эффективности тестирования в отношении установленных целей.

5.3.3 Контроль тестирования (К2)

Контроль тестирования описывает любые действия, предпринятые для руководства и корректировки, произошедшие как результат собранной и предоставленной информации и метрик. Действия могут

покрывать любые тестовые действия и могут иметь воздействия на любые другие действия и задачи жизненного цикла ПО.

Примеры действий по контролю тестирования:

- Повторная расстановка приоритетов, когда возникает установленный риск (например, задержка выпуска ПО);
- Изменение графика тестирования согласно доступности тестовой конфигурации;
- Установка входного критерия требующего повторного тестирования исправления разработчиком перед принятием его в сборку .

5.4 Управление конфигурациями (K2)

Термины

Управление конфигурациями, контроль версий.

Предварительные знания

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

Для тестирования управления конфигурациями может гарантировать, что:

- Все пункты того, что необходимо протестировать определены, контролируются по версиям, все изменения записываются, связаны друг с другом и к разрабатываемыми элементами (объектами тестирования) таким образом, что трассируемость может быть обеспечена на протяжении всего процесса тестирования.
- Вся установленная документация и элементы ПО однозначно определена в документации по тестированию.

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

Во время планирования тестирования процедура конфигурационного управления и инфраструктура (средства) должны быть выбраны, задокументированы и внедрены.

5.5 Риски и тестирование (K2)

Термины
Риск продукта, риск проекта, риск, тестирование ориентированное на риски.

Предварительные знания
Риск может быть определен как вероятность возникновения события, опасности, угрозы или ситуации, и ее нежелательных последствий, как потенциальной проблемы. Уровень риска может быть определен как вероятность неблагоприятного события и его влияние (ущерб, проявляющийся как результат этого события).

5.5.1 Риски проекта (K1, K2)
Риски проектов – это риски, которые подвергают опасности способность проекта достигнуть его целей, и включают такие как:

- Проблемы поставщика:
  - сбой третьей стороны;
  - проблемы контракта.
- Организационные факторы:
  - недостатки навыков и штата;
  - личные проблемы людей и проблемы обучения;
  - политические проблемы, такие как:
    - проблемы с тем, что тестировщики в недостаточной степени сообщают о своих проблемах и результатов тестов;
    - неспособность следовать информации, найденной в процессе тестирования и экспертизы (например, отсутствие улучшений в практике разработки и тестирования).
  - несоответствующее отношение к тестированию или к ожидаемому результату тестирования (например, недооценка значения находимых дефектов в процессе тестирования).
- Технические проблемы:
  - проблемы в определении правильных требований;
  - распространенность мнения, что требования могут быть выполнены с условием существующих ограничивающих условий;
  - качество проектирования, кода и тестов.

При анализе, управлении и уменьшении этих рисков тест менеджер должен следовать твердо установившимся принципам управления проектами. План документа ‘Стандарт на Документацию для Тестирования Программного Обеспечения’ (IEEE 829) для тестовых планов требует, чтобы были прописаны риски и непредвиденные обстоятельства.

5.5.2 Риски Продукта (K2)
Потенциальные области сбоя (неблагоприятные будущие события или опасности) в программном обеспечении или системе известны как риски продукта, так как они подвергают риску качество продукта, например:

- Поставка программного обеспечения подверженного ошибкам.
- Возможность того, что программное/аппаратное обеспечение может причинить вред работнику или компании.
- Плохие характеристики программного обеспечения (например, функциональность, безопасность, надежность, удобство использования и производительность).
- Программное обеспечение, которое не выполняет предполагаемые функции.

Риски используются для определения того, где начинать тестирование и каким аспектам уделить особое внимание; тестирование используется для того, чтобы уменьшить риск возникновения неблагоприятных результатов или уменьшить их воздействие.

Риски продукта – это особенный тип рисков, который влияет на успех проекта. Тестирование это процесс, направленный на предоставление данных по оставшимся рискам путем измерения эффективности устранения критических дефектов и планов на случай возникновения непредвиденных дополнительных обстоятельств.

Подход к тестированию, основанный на рисках предоставляет превентивные возможности для уменьшения уровня рисков продукта, начиная с первоначальных этапов проекта. Он включает в себя идентификацию рисков продукта и их использование в управлении планированием тестов, спецификациями, приготовлением и выполнением тестов. В подходе, основанном на рисках – установленные риски могут быть использованы для:

- Определения методики тестирования для использования.
- Определения объема тестирования, которое должно быть выполнено.
- Установления приоритетов тестирования для того, чтобы обнаружить критические дефекты как можно раньше.
- Определения необходимости деятельности не связанной с тестированием, которая может быть предпринята для уменьшения рисков (например, предоставление тренинга для неопытных проектировщиков).

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

Для того чтобы быть уверенным в том, 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.
- Выполнению действий по работе с этими рисками.

В дополнение, тестирование может поддерживать определение новых рисков, может помогать определять, какие риски должны быть понижены и могут понижать неопределенность в отношении рисков.

5.6 Управление отказами (К3)

Термины
Записывание инцидентов.

Предварительные знания
Так как одной из целей тестирования является поиск дефектов, то разница между действительным и ожидаемым результатом должна быть записана как отказ. Отказы должны отслеживаться от открытия и классификации до исправления и подтверждения исправления. Для того чтобы управлять всеми отказами на должном уровне, организации необходимо установить процесс и правила для классификации.

Отказы могут возникать во время разработки, анализа, тестирования или использования ПО. Они могут быть найдены в коде или работающей системы, или в любом типе документации, включая документацию по разработке, документацию по тестированию, информацию для пользователей типа «Справка» или инструкцию к установке.

Отчет об отказах имеет следующие цели:

- Обеспечивает разработчика и других людей информацией о проблеме для того, чтобы идентифицировать, изолировать и исправить, если необходимо.
- Обеспечивает руководителя тестирования средством отслеживания качества системы, которая тестируется, и прогресса тестирования.
- Обеспечивает идеями для улучшения процесса тестирования.

Тестировщик или эксперт обычно записывает об отказе следующую информацию, если она известна:

- Дату ошибки, организацию, автора, подтверждения, статус;
- Область действия, серьезность и приоритет отказа;
- Ссылки, включая идентификатор спецификаций тестового сценария, который обнаружил проблему.

Детальный отчет об отказе может включать:

- Ожидаемый и действительный результаты;
- Дату нахождения отказа;
- Идентификатор или конфигурационный элемент ПО или системы;
- Процесс ПО или жизненного цикла системы, в котором ошибка была найдена;
- Описание дефекта для возможности исправления;
- Уровень влияния на заинтересованные стороны;
- Серьезность влияния на систему;
- Срочность/приоритет для исправления;
- Статус отказа (например, открыта, отложена, продублирована, в ожидании исправления, исправлена, в ожидании подтверждающего тестирования или закрыта);
- Заключение и рекомендации;
- Общие проблемы, такие как другие области, которые могут быть затронуты в результате исправления отказа;
- История изменений, например, последовательность действий, предпринятых членами команды проекта для выделения, исправления и подтверждения исправления отказа.

Структура отчета об отказе описывается в ‘Стандарте на Документацию для Тестирования Программного Обеспечения’ (IEEE 829) и называется отчетом отклонения от нормы.

Ссылки:
5.1.1 Black, 2001, Hetzel, 1998
5.1.2 Black, 2001, Hetzel, 1998
5.2.5 Black, 2001, Craig, 2002, IEEE 829, Kaner 2002

5.3.3 Black, 2001, Craig, 2002, Hetzel, 1998, IEEE 829
5.4 Craig, 2002
5.5.2 Black, 2001 , IEEE 829
5.6 Black, 2001, IEEE 829

6 Инструментальные средства поддержки тестирования (К2)

Цели изучения инструментальных средств поддержки тестирования

Цели определяют, что Вы будете уметь по окончании изучения каждого модуля.

6.1. Типы средств тестирования (К2)

- Классифицировать различные типы средств поддержки тестирования согласно работам процесса тестирования (К2)
- Определить инструментарий, который может помочь разработчикам в тестировании (К1)

6.2. Эффективное использование инструментальных средств: потенциальная выгода и риски (К2)

- Подвести итог по потенциальной выгоде и рискам при использовании средств автоматизации и инструментальных средств поддержки тестирования (К2)
- Осознать, что средства выполнения тестов могут иметь различные методики использования скриптов, включая методики управления данными и управления ключевыми словами (К1)

6.3. Внедрение инструментального средства в организации (К1)

- Описать основные принципы представления инструментария в организации (К1)
- Описать цель опытно-экспериментальной/пилотной фазы для оценки инструментария (К1)
- Осознать, что для хорошей инструментальной поддержки нужны иные условия, чем просто покупка этого средства (К1)

6.1 Типы инструментальных средств тестирования (К2)

Термины

Средство управления конфигурацией, средство измерения покрытия, средство отладки, драйвер, средство динамического анализа, средство управления инцидентами, средство нагрузочного тестирования, средство моделирования, средство мониторинга, средство тестирования производительности, эффект зонда, средство управления требованиями, средство поддержки процесса экспертизы, средство безопасности, средство статического анализа, средство стрессового тестирования, заглушка, тестовый сравниватель, средство подготовки тестовых данных, средство проектирования тестов, тестовый узел, средство выполнения тестов, средство управления тестами, средство оболочки модульных тестов.

6.1.1 Классификация средств тестирования(К2)

Есть множество средств тестирования, которые поддерживают различные аспекты тестирования. Средства тестирования классифицированы в этом курсе в соответствии с функциями тестирования, которые они поддерживают.

Некоторые средства поддерживают только одну функцию; другие могут поддерживать больше, чем одну функцию, но классифицируются по той функции, с которой они наиболее тесно связаны. Некоторые коммерческие средства тестирования предлагают поддержку только одного типа функционала; другие коммерческие средства тестирования предлагают комплекты или линейки средств, которые обеспечивают поддержку для многих или всех функций.

Средства тестирования могут улучшить эффективность тестирования путем автоматизации повторяющихся заданий. Средства тестирования также могут улучшить надежность тестирования путем, например, автоматизации сравнения больших объемов данных или эмуляции поведения.

Некоторые типы средств тестирования могут быть «агрессивными» в том смысле, что сам инструментарий влияет на результаты тестирования. Например, измерение времени может быть различным в зависимости от того, как Вы измеряете его, используя различные средства тестирования производительности, или Вы можете получить различную степень покрытия кода в зависимости от средства, которое Вы используете.

Некоторые средства обеспечивают поддержку более подходящую для разработчиков (например, во время тестирования компонентов или интеграции компонентов). В данной ниже классификации такие средства отмечены как «D».

6.1.2 Средства поддержки управления тестированием и тестирования (К1)

Средства управления применяются во всех процессах тестирования на протяжении всего жизненного цикла ПО.

Средства управления тестированием
Параметры средств управления тестирования включают:

1. Поддержку управления выполняемыми тестами и процессами тестирования.
2. Интерфейс для средств выполнения тестов, средств отслеживания дефектов и средств управления требованиями.
3. Независимое средство управления версиями или интерфейс для внешнего средства управления конфигурациями.
4. Средства для трассируемости тестов, результатов тестирования и отчетов об инцидентах с исходной документацией, такой как спецификация требований.
5. Запись результатов тестирования и генерация отчетов.
6. Количественный анализ (метрики), относящиеся к тестам (например, выполненные тесты и успешные тесты) и тестовым объектам (например, возникшие отказы), для того, чтобы дать

информацию о тестовых объектах и для того, чтобы контролировать и улучшать процесс тестирования.

Средства управления требованиями

Средства управления требованиями сохраняют требования, проверяют на согласованность и неопределенные (пропущенные) требования, позволяют расставлять приоритеты по требованиям и позволяют отдельным тестам трассироваться согласно с требованиями, функциями и/или опциями. Результаты трассировки могут быть включены в отчеты о тестировании. Покрытие требований, функций и/или опций набором тестов также может быть отражено в отчетах.

Средства управления инцидентами

Средства управления инцидентами хранят и управляют отчетами об отказах, то есть дефектах, сбоях или осознанных проблемах и отклонениях, и обеспечивает управление отчетами об инцидентах таким образом, что:

- Помогают выставлять приоритеты.
- Распределяют необходимые действия между людьми (например, тестирование исправлений или подтверждающее тестирование).
- Определение статусов (например, отменен, готов быть протестирован или отложен до следующего выпуска).

Эти средства позволяют постоянно отслеживать отчеты об инцидентах, часто содействуют статистическому анализу и предоставляют отчеты об инцидентах. Они также известны как средства отслеживания дефектов.

Средства управления конфигурациями

Средства управления конфигурациями не являются, строго говоря, средствами тестирования, но обычно они необходимы для того, чтобы отслеживать различные версии и сборки ПО и тесты.

Средства управления конфигурациями:

- Хранят информацию о версиях и сборках ПО и проведенным тестированием.
- Обеспечивают сопоставление между проведенным тестированием и результатом работы ПО и вариантами продукта.
- Особенно полезны, когда разработка ведется более чем на одной конфигурации аппаратного и программного обеспечения (например, для различных версий операционных систем, различных библиотек или компиляторов, различных браузеров или различных компьютеров).

6.1.3 Средства поддержки статического тестирования (K1)

Средства поддержки процесса экспертизы

Средства поддержки процесса экспертизы могут хранить информацию о процессах экспертизы, хранить и обмениваться экспертными комментариями, предоставлять информацию о дефектах и объемах работ, управлять связями с экспертными правилами и/или списками проверки, и обеспечивать трассируемость между документацией и исходным кодом. Также они могут предоставлять возможность экспертизы в реальном времени, которая полезна при географической распределенной команде.

Средства статического анализа (D)

Средства статического анализа помогают разработчикам, тестировщикам и сотрудникам обеспечения качества в поиске дефектов до начала динамического тестирования. Основные цели включают:

- Соблюдение стандартов программирования.
- Анализ структуры и зависимостей (например, ссылки на веб-страницах).
- Помощь в понимании кода.

Средства статического анализа могут извлекать метрики из исходного кода (например, сложность), которые могут дать ценную информацию, например, для анализа планирования или рисков.

Средства моделирования (D)

Средства моделирования способны проверять достоверность моделей ПО. Например, средство контроля моделей баз данных может найти дефекты и несоответствия в модели данных; иные средства моделирования могут найти дефекты в моделях состояний или объектных моделях. Эти средства зачастую могут помочь в составлении некоторых тестовых сценариях, основанных на моделях (см. также Средства проектирования тестов ниже).

Основное преимущество средств статического анализа и средств моделирования – выгода в стоимости нахождения большего числа дефектов на более ранних стадиях цикла разработки. Как результат, процесс разработки может ускориться и улучшиться за счет меньшего числа переработок.

6.1.4 Средства поддержки спецификации тестов (К1)

Средства проектирования тестов

Средства проектирования тестов генерируют входные данные или непосредственно тесты из требований, графического интерфейса пользователя, моделей проектирования (состояний, данных или объектов) или кода. Эти средства могут также генерировать и ожидаемые результаты (т.е. использовать тестовый оракул). Тесты, сгенерированные из модели состояний или объектов, полезны для проверки реализации модели в ПО, но редко достаточны для проверки всех аспектов ПО или системы. Они могут сэкономить драгоценное время и обеспечить повышенную глубину тестирования за счет полноты тестов, генерируемых этими средствами.

Другие средства этой категории могут помочь в поддержке генерации тестов, обеспечивая структурные шаблоны, иногда называемые тестовыми фреймами, которые генерируют тесты или тестовые заглушки, таким образом увеличивая скорость проектирования тестов.

Средства подготовки тестовых данных (D)

Средства подготовки тестовых данных оперируют с базами данных, файлами или передачей данных для определения тестовых данных, которые будут использоваться во время выполнения тестов. Преимущество таких средств в том, что реальные данные передаются в тестовую среду анонимно для защиты данных.

6.1.5 Средства поддержки выполнения и протоколирования тестов (К1)

Средства выполнения тестов

Средства выполнения тестов позволяют выполнять тесты автоматически или полуавтоматически, используя сохраненные входные данные и ожидаемые результаты, с использованием скриптового языка. Скриптовый язык позволяет управлять тестами с ограниченными затратами, например, повторять тест с различными данными или проверять различные части системы одинаковыми шагами. Обычно такие средства включают функции динамического сравнивания и предоставляют возможность протоколирования каждого запуска теста.

Средства выполнения тестов также могут использоваться для записи тестов, когда используются как средства записи/воспроизведения тестов. Запись тестовых входных данных во время исследовательского тестирования или бесскриптовое тестирование может быть полезно для воспроизведения и/или документирования теста, например, если произошел отказ.

Тестовый узел/средства оболочки модульных тестов (D)

Тестовые узлы могут облегчать тестирование компонентов или части системы, имитируя окружение, на котором объект тестирования будет работать. Это может быть сделано по причине того, что другие компоненты этого окружения еще не готовы и заменены заглушками и/или драйверами, или просто для того, чтобы обеспечить предсказуемое и контролируемое окружение, в котором любые отказы могут быть локализованы в объекте тестирования.

Оболочка может быть создана там, где часть кода, объект, метод или функция, модуль или компонент могут быть выполнены, вызывая объект тестирования и/или обеспечивая обратную связь с этим объектом. Это может быть сделано с помощью искусственных средств поддержки входных данных объекта тестирования и/или заглушек для получения выходных данных вместо реальных выходных данных.

Средства тестовых узлов также могут быть использованы для обеспечения оболочки выполнения промежуточного ПО (middleware), где языки, операционные системы и аппаратное обеспечение должны быть протестированы вместе.

Они могут называться средствами оболочки модульных тестов, когда они конкретно сфокусированы на компонентном уровне тестирования. Этот тип средств помогает в выполнении компонентных тестов параллельно с разработкой кода.

Тестовые сравниватели
Тестовые сравниватели устанавливают различия между файлами, базами данных или тестовыми результатами. Средства выполнения тестов обычно обычно включают динамические сравниватели, но сравнение после выполнения может быть выполнено отдельными средствами. Тестовый сравниватель может использовать тестового оракула, особенно если он автоматизирован.

Средства измерения покрытия (D)
Средства измерения покрытия могут быть интрузивными или неинтрузивными в зависимости от используемой методики измерения, того, что измеряется и языка программирования. Средства покрытия кода измеряют процент конкретных типов структур кода, которые были проверены (например, команды, ветви или решения, и вызовы модулей или функций). Эти средства показывают, насколько глубоко измеряемый тип структур был проверен набором тестов.

Средства безопасности
Средства безопасности проверяют на компьютерные вирусы и DoS-атаки. Брандмауэр, например, не совсем средство тестирования, но может быть использован в тестировании безопасности. Другие средства безопасности нагружают систему, ища определенные уязвимости в системе.

6.1.6 Средства поддержки тестирования производительности и мониторинга (K1)

Средства динамического анализа (D)
Средства динамического анализа находят дефекты, которые становятся очевидными только, когда ПО выполняется, такие как временные задержки и утечки памяти. Обычно они используются при тестировании компонентов и интеграции компонентов, а также при тестировании промежуточного ПО (middleware).

Средства тестирования производительности/нагрузочного тестирования/стрессового тестирования
Средства тестирования производительности отслеживают и сообщают о том, как ведет себя система при различных условиях использования. Они имитируют нагрузку на приложение, базу данных или окружение системы, такое как сеть или сервер. Часто эти средства называются в соответствии с тем, что они измеряют, например, для проверки нагрузки или стрессовых условий применяются средства нагрузочного и стрессового тестирования. Обычно они основаны на автоматически повторяемых тестах, контролируемых определенными параметрами.

Средства мониторинга
Средства мониторинга не совсем являются средствами тестирования, но они предоставляют информацию, которая может быть использована в целях тестирования и недоступна иными способами.

Средства мониторинга непрерывно анализируют, проверяют и сообщают об использовании конкретных системных ресурсов и выдают предупреждения, если возможны проблемы. Они хранят информацию о версии и сборке ПО и тестовой среды и обеспечивают трассируемость.

6.1.7 Средства поддержки тестирования приложений специфических областей (K1)

Отдельные примеры типов средств, приведенных выше, могут быть приспособлены для использования в определенных типах приложений. Например, существуют средства тестирования производительности специально для веб-приложений, средства статического анализа для определенных платформ разработки, и средства динамического анализа специально для тестирования аспектов безопасности.

Коммерческие пакеты средств могут специализироваться на определенных областях приложений (например, встраиваемые системы).

6.1.8 Средства поддержки использующие другие средства (K1)

Перечисленные здесь средства поддержки – не все типы средств, используемых тестировщиками. Это также могут быть электронные таблицы, SQL, средства отладки (D), например.

6.2 Эффективное использование инструментальных средств: потенциальные выгоды и риски (К2)

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

6.2.1 Потенциальные выгоды и риски использования инструментальных средств поддержки тестирования (для всех средств) (К2)

Просто покупка или аренда средства не гарантирует успех. Каждый тип средств может потребовать дополнительные затраты для достижения реальных и устойчивых выгод. Существуют потенциальные возможные преимущества использования средств, но также существуют и риски.

Потенциальные выгоды от использования инструментальных средств:

- Уменьшается повторяющаяся работа (например, запуск регрессионных тестов, ввод одинаковых тестовых данных, проверка на соответствие стандартам программирования).
- Большая целостность и повторяемость (например, тесты запускаются средством и получаются из требований).
- Объективная оценка (например, статических измерений, покрытия и поведения системы).
- Простота доступа к информации о тестах и тестировании (например, статистика и графики о прогрессе тестирования, уровне ошибок и производительности).

Риски использования средств тестирования включают:

- Нереалистичные ожидания от средства (включая функциональность и простоту использования).
- Недооценка времени, стоимости и объемов работ на начальное внедрение средства (включая обучение и внешние экспертизы).
- Недооценка времени и объемов работ, необходимых для получения значимого и устойчивого результата от использования средства (включая необходимость изменений в процессе тестирования и постоянного улучшения способа использования средства).
- Недооценка затрат на поддержку тестовых артефактов, генерируемых средством.
- Слишком большая надежда на средство (замена проектирования тестов или использование там, где ручное тестирование уместней).

6.2.2 Отдельные замечания по определенным типам средств (К1)

Средства выполнения тестов

Средства выполнения тестов выполняют скрипты, разработанные для реализации тестов, хранимых в электронном виде. Этот тип средств обычно требует значительных затрат для получения значительных результатов.

Создание тестов, путем записи действий тестировщика кажется привлекательным, но этот подход не применим для большого числа автоматизированных тестов. Записанный скрипт – это линейное представление с конкретными данными и действиями как часть каждого скрипта. Этот тип скриптов может быть нестабильным, если возникают непредвиденные события.

Подход, основанный на данных, разделяет входные тестовые данные, обычно в виде электронных таблиц, и использует более общий скрипт, который может считывать данные и выполнять одинаковый тест в различными данными. Тестировщики, не знакомые со скриптовым языком, могут вводить данные для предопределенных скриптов. В подходе, основанном на ключевых словах, электронная таблица содержит ключевые слова, описывающие действия, которые необходимо выполнить (также называемые словами действий) и тестовые данные. Тестировщики (даже если они не знакомы со скриптовым языком), могут создать тесты, используя ключевые слова, относящиеся к тестируемому приложению.

Для всех подходов необходима техническая экспертиза скриптового языка (тестировщиками или специалистами по автоматизации тестов).

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

Средства тестирования производительности

Средства тестирования производительности нуждаются в ком-то, кто обладает практическим опытом в тестировании производительности, для проектирования тестов и интерпретации результатов.

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 большое число сообщений. Предупреждения не предотвращают перевод исходного кода в исполнимый, но 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 для получения информации в наилучшем формате для текущих нужд организации. Отчеты должны быть спроектированы и отслеживаться таким образом, чтобы приносить пользу.

6.3 Внедрение инструментального средства в организации (К1)

Термины
Нет специфичных терминов.

Предварительные знания
Основные принципы внедрения средств в организации включают:

- Оценку зрелости, сильных и слабых сторон организации и поиск возможностей улучшенного процесса тестирования, поддерживаемого средствами.
- Оценку на основании доступных средств и объективных критериях.
- Опытную эксплуатацию для проверки требуемой функциональности и определения того, отвечает ли продукт целям.
- Оценка поставщика (включая обучение, поддержку и коммерческий фактор).
- Определение внутренних требований к руководству использования средств.

Опытная эксплуатация может быть проведена на малом пилотном проекте, позволяющем снизить влияние при обнаружении серьезных препятствий и неудачном пилотном проекте.

Пилотный проект имеет следующие цели:

- Узнать больше деталей о средстве.
- Посмотреть, как средство сочетается с существующими процессами и инструкциями, и как они должны будут измениться.
- Принять решение по стандартному использованию, хранению и поддержке средства и преимуществам тестирования (например, соглашения по именованию файлов и тестов, созданию библиотек и определнию модульности тестовых пакетов).
- Оценка, будут ли преимущества достигнуты приемлемыми затратами.

Факторы успеха развертывания средства в организации включают:

- Распространение средства на всю организацию постепенно.
- Адаптация и улучшение процессов для совместимости с использованием средства.
- Обеспечение обучения и руководства новых пользователей.
- Определение норм использования.
- Реализация способов получения опыта из использования средства.
- Мониторинг использования средства и преимуществ.

Ссылки:
6.2.2 Buwalda, 2001, Fewster, 1999
6.3 Fewster, 1999

7 Ссылки

Стандарты

ISTQB Glossary of terms used in Software Testing Version 1.0

[CMMI] Chrissis, M.B., Konrad, M. and Shrum, S. (2004) CMMI, Guidelines for Process Integration and Product Improvement, Addison Wesley: Reading, MA
Раздел 2.1

[IEEE 829] IEEE Std 829™ (1998/2005) IEEE Standard for Software Test Documentation (currently under revision)
Разделы 2.3, 2.4, 4.1, 5.2, 5.3, 5.5, 5.6

[IEEE 1028] IEEE Std 1028™ (1997) IEEE Standard for Software Reviews
Раздел 3.2

[IEEE 12207] IEEE 12207/ISO/IEC 12207-1996, Software life cycle processes
Раздел 2.1

[ISO 9126] ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality
Раздел 2.3

Книги

[Beizer, 1990] Beizer, B. (1990) Software Testing Techniques (2nd edition), Van Nostrand Reinhold: Boston
Разделы 1.2, 1.3, 2.3, 4.2, 4.3, 4.4, 4.6

[Black, 2001] Black, R. (2001) Managing the Testing Process (2nd edition), John Wiley & Sons: New York
Разделы 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
Раздел 6.2

[Copeland, 2004] Copeland, L. (2004) A Practitioner's Guide to Software Test Design, Artech House: Norwood, MA
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
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
Разделы 6.2, 6.3

[Gilb, 1993]: Gilb, Tom and Graham, Dorothy (1993) Software Inspection, Addison Wesley: Reading, MA
Разделы 3.2.2, 3.2.4

[Hetzel, 1988] Hetzel, W. (1988) Complete Guide to Software Testing, QED: Wellesley, MA
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 Pettticord, B. (2002) Lessons Learned in Software Testing, John Wiley & Sons:
Разделы 1.1, 4.5, 5.2

[Myers 1979] Myers, Glenford J. (1979) The Art of Software Testing, John Wiley & Sons:
Разделы 1.2, 1.3, 2.2, 4.3

[van Veenendaal, 2004] van Veenendaal, E. (ed.) (2004) The Testing Practitioner (Chapters 6, 8, 10), UTN Publishers: The Netherlands
Разделы 3.2, 3.3

Дополнение А – История курса

История документа.

Этот документ был подготовлен в 2004-2005 годах членами International Software Testing Qualifications Board (ISTQB). Первоначально был отредактирован специалистами, а позже передан на рассмотрение представителям международного сообщества специалистов по тестированию ПО. Правила использованные при составлении этого документа представлены в дополнении С.

Этот документ является программой для International Foundation Certificate in Software Testing, международной квалификации первого уровня, принятой ISTQB (www.istqb.org). Во время составления (2005) членами ISTQB являлись Австрия, Дания, Финляндия, Франция, Германия, Индия, Израиль, Япония, Корея, Норвегия, Польша, Португалия, Испания, Швеция, Швейцария, Нидерланды, Великобритания и США.

Цели Базового уровня сертификации

- Получить понятие тестирования как необходимой и профессиональной специализации в программной инженерии.
- Показать стандартный подход к развитию карьеры специалиста по тестированию
- Получить квалифицированным специалистам по тестированию признание работодателей, клиентов и коллег, и повысить интерес к профессии.
- Содействовать внедрению целостных и правильных практик тестирования во всех дисциплинах программной инженерии.
- Идентифицировать темы тестирования которые являются актуальными и важными в индустрии.
- Позволить производителям ПО нанимать сертифицированных специалистов таким обзором повышая свою коммерческую привлекательность над конкурентами рекламируя кадровую политику в отношении специалистов по тестированию.
- Показать возможность для специалистов по тестированию и лиц заинтересованных в тестировании получить международную сертификацию признанную в мире.

Цели международной квалификации (фрагмент конференции в Sollentuna, November 2001)

- Возможность сравнивать навыки тестирования в различных странах
- Возможность специалистам по тестированию проще пересекать границы стран
- Обеспечить интернациональным и международным проектам общее понимание проблем тестирования.
- Повысить количество сертифицированных специалистов по тестированию во всем мире.
- Оказать большее влияние \ ценность в качестве международного подхода, нежели подходов принятых в разных странах.
- Разработать общий базис для понятий и знаний по тестированию посредством программы и терминологии, и повысить уровень знаний по тестированию для всех.
- Популяризировать профессию тестирования по всему миру.
- Предоставить специалистам по тестированию 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 International Foundation Certificate in Software Testing является интерес кандидатов в области тестирования ПО. Однако кандидатам также рекомендуется:

- Иметь минимальное представление либо о разработке ПО, либо о тестировании ПО, и 6 месячный опыт работы в качестве специалиста по тестированию или разработчика ПО.
- Пройти курс аккредитированный по стандартам ISTQB (в одной из Коллегий признанной ISTQB).

Предпосылки и история Сертификации в области тестирования на базовый уровень

Международная сертификация специалистов по тестированию берет свое начало в Великобритании British Computer Society's Information Systems Examination Board (ISEB), когда в 1998 году была создана

Коллегия специалистов по тестированию ПО (www.bcs.org.uk/iseb). В 2002 году ASQF в Германии начала поддерживать немецкую квалификационную схему для специалистов по тестированию (www.asqf.de). Эта программа основана на программах ISEB и ASQF; включает в себя реорганизованную и обновленную программу с некоторыми дополнениями, акцент сделан на темы которые представляют наибольшую практическую ценность.

Существующая сертификация в области тестирования на базовый уровень (например от ISEB, ASQF или от любой коллегии признанной ISTQB), выданная до выпуска жтого Международного Сертификата, будет считаться эквивалентом Международного сертификата. Foundation Certificate остается действительным и нет надобности его возобновлять. Дата сдачи указана на сертификате.

В каждой конкретной стране местные особенности учитываются национальной Коллегией сертифицированной ISTQB. Обязанности национальных коллегий определены ISTQB, но реализуются только внутри конкретный стран. В обязанности коллегий различных стран включается аккредитация объектов обучения и прием экзаменов.

Дополнение Б - Цели обучения \ уровни знания.

Следующие цели обучения определены для этой программы. Экзамен по каждой теме будет проведем в соответствии с целями обучения по этой теме.

Уровень 1: запомнить (К1)
Кандидат осознает, запомнит и озвучит термин или понятие.

Пример
Выбрать определение «сбой» из:
- Не предоставление услуги конечному пользователю или заказчику
- Фактическое отклонение компонента или системы от ожидаемого поведения, действия или результата.

Уровень 2: понять (К2)
Кандидат может выбрать причины или объяснения положений которые относятся к теме, и может обобщать, сравнивать, классифицировать и приводить примеры для понятий тестирования.

Пример
Объяснить причину почему тесты должны быть разработаны как можно раньше:
- Чтобы найти дефекты когда стоимость их исправления невелика
- 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, а системное – на всей системе от начала и до конца.

Уровень 3: применять (К3)
Кандидат может выбирать правильное применение понятия или техники и использовать их в данном контексте.

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

Ссылки:

(For the cognitive levels of learning objectives)
Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon:

Дополнение B - Правила применимые к программе ISTQB базовый уровень

Приведенные ниже правила используются для разработки и редактирования программы

Общие правила

SG1. Программа должна быть понятной для людей с опытом работы в тестировании от 0 до 6 месяцев. (6-MONTH)

SG2. Программа должна быть скорее практической чем теоретической. (PRACTICAL)

SG3. Для целевой аудитории программа должна быть точно и недвусмысленный. (CLEAR)

SG4. Программа должна быть понятной для людей из разных стран и с легкостью переводимой на другие языки. (TRANSLATABLE)

SG5. Языком программы должн быть американский английский. (AMERICAN-ENGLISH)

Текущее содержание

SC1. Программа должна включать в себя новые понятия по тестированию и отображать лучшие практики на текущее время в тестировании ПО. Программа должна пересматриваться каждые 3-5 лет. (RECENT)

SC2. Жизненный цикл программа должен быть не менее 3-5 лет. (SHELF-LIFE)

Цели обучения

LO1. Цели обучения должны быть распределены между понятиями которые должны быть осознанны и запомнены (Уровень понимания К1) и понятиями которые кандидат должен усвоить концептуально (К2) и теми понятиями которые кандидат должен будет использовать на практике (К3). (KNOWLEDGE-LEVEL)

LO2. Содержание должно соответствовать целям обучения. (LO-CONSISTENT)

LO3. Для иллюстрации целей обучения программа должна включать в себя примеры экзаменационных вопросов для каждой из секций. (LO-EXAM)

Общая структура

ST1. Структура программы должна быть доступной и содержать перекрестные ссылки (CROSS-REF)

ST2. Количество повторов понятий должно быть минимизировано. (OVERLAP)

ST3. Все секции программы должны иметь единую структуру. (STRUCTURE-CONSISTENT)

ST4. На каждой странице программа должны быть: номер версии, дата издания и номера страниц. (VERSION)

ST5. Программа должна содержать временные оценки для раскрытия каждой темы (чтобы отобразить относительную важность каждой темы) (TIME-SPENT).

Ссылки

SR1. Источники и ссылки должны быть указаны ко всем понятиям программы, чтобы помочь тренинг провайдерам найти больше информации по теме. (REFS)

SR2. В местах где сложно точно определить источник должно быть приведено развернутое объяснение. Например определения приведены в глоссарии а термины в программе. (NON-REF DETAIL)

Источники информации

Термины которые используются в программе определены в «Глоссарии терминов ISTQB по тестированию ПО». Версия глоссария предоставляется ISTQB.

Список рекомендуемой литературы по тестированию ПО также включается в программу. Главный список литературы является частью секции ссылок.

Дополнение Г - Замечания для обучающих

Каждая тема программы имеет временные рамки. Цель этого показать относительные пропорции времени отведенного на каждую секция программы аккредитированного курса, так и указать минимальное время необходимое для каждой секции. Тренинг провайдеры могут использовать больше времени чем указано, как и кандидаты могут тратить больше времени на чтение и практику. Последовательность преподавания глав может не совпадать с порядком в котором они приведены в программе.

Программа содержит ссылки на действующие стандарты которые могут быть использованы при подготовке практического материала. В программе должны приводиться версии используемых стандартов. Другие публикации, шаблоны или стандарты не указанные в программе также могут быть использованы или упомянуты, но не являются предметом экзамена.

По следующим темам необходимы практические занятия:

4.3 Метод основанный на спецификации или метод «черного ящика».
Практические задания (короткие упражнения) должны включать в себя 4 метода:
эквивалентное разбиение, анализ граничных значений, таблицы принятия решений и диаграммы перехода состояний. Лекции и упражнения должны быть основаны на ссылках на каждый из этих методов.

4.4 Структурный подход или метод «белого ящика»
Практические задания (короткие упражнения) должны включать в себя оценку теста на 100% покрытие команд и 100% покрытие решений, на равне с создание тест кейсов.

5.6 Управление ошибками
Практические задания (короткие упражнения) должны включать создание отчетов об ошибке и их оценку.