Курс «Тестирование ПО»
Тема 1
«Введение в тестирование»

Тестирование ПО. Введение.

Для начала – немного истории. Когда-то, «когда компьютеры были большими, а программы маленькими», о тестировании ПО практически не говорили. Тогда разработкой ПО занимались крупные фирмы и институты, количество разработанных продуктов измерялось единицами в год, процесс разработки ПО был крайне медленной кропотливой работой.

Но время шло своим ходом, появились достаточно дешёвые и общедоступные компьютеры, область их применения значительно расширилась и, соответственно, резко возросло количество разрабатываемых программ и фирм, занимающейся такой разработкой.

Основной всплеск интереса к теме тестирования пришёлся на 90-е годы и начался в США.

Бурное развитие систем автоматизированной разработки ПО (CASE-средств) и сетевых технологий привело к росту рынка производства ПО и к пересмотру вопросов обеспечения качества и надёжности разрабатываемых программ.

Тестирование ПО. Введение.

Резко усилившаяся конкуренция между производителями ПО потребовала особого внимания к качеству создаваемых продуктов, т.к. теперь у потребителя был выбор: многие фирмы предлагали свои продукты и услуги по достаточно приемлемым ценам, а потому можно было обратиться к тем, кто разработает программу не только быстро и дёшево, но и качественно.

Ситуация осложнилась тем фактом, что в настоящее время компьютеризации подвержены практически все области человеческой жизни. И вопрос о качестве ПО начинает приобретать особую важность: сегодня это уже не только комфорт от работы в той или иной программе, сегодня ПО управляет оборудованием в больницах, диспетчерскими системами в аэропортах, атомными реакторами, космическими кораблями и т.д.

Тестирование ПО. Введение.

Осознав тот факт, что обеспечение высокого качества разрабатываемого ПО – это реальный путь «обойти» конкурентов, многие компании во всём мире вкладывают всё больше средств в обеспечение качества своих продуктов, создавая собственные группы и отделы, занимающиеся тестированием, или передавая тестирование своих продуктов сторонним организациям.

Наиболее крупные компании, заботящиеся о своей репутации и желающие пройти сертификацию на высокий уровень CMMI (Capability Maturity Model Integration) создают свои собственные системы управления качеством (Quality Management System), направленные на постоянное совершенствование производственных процессов и постоянное повышение качества ПО.

Тестирование ПО. Понятие тестирования.

Сегодня тестирование стало обязательной частью процесса производства ПО. Оно направлено на обнаружение и устранение как можно большего числа ошибок. Следствием такой деятельности является повышение качества ПО по всем его характеристикам (будут рассмотрены ниже).

Тестирование программного обеспечения (software testing) – это процесс анализа или эксплуатации программного обеспечения с целью выявления дефектов. Тестирование определяется как процесс потому, что тестирование – это плановая и упорядоченная, протяжённая во времени деятельность. Этот момент очень важен, поскольку в условиях ограниченного времени, выделенного на разработку и тестирование, хорошо продуманный подход быстрее приводит к обнаружению программных ошибок, чем плохо спланированное тестирование.

Тестирование ПО. Характеристики качества.

Определения характеристик качества (ISO 9126-1):

Функциональные возможности - способность программного средства обеспечивать решение задач, удовлетворяющих сформулированные потребности заказчиков и пользователей при применении комплекса программ в заданных условиях.

Функциональная пригодность - набор атрибутов, определяющих назначение, номенклатуру, основные, необходимые и достаточные функции программного средства, соответствующие техническому заданию и спецификациям требований заказчика или потенциального пользователя.

Правильность (корректность) - способность программного средства обеспечивать правильные или приемлемые для пользователя результаты и внешние эффекты.

Способность к взаимодействию - свойство программных средств и их компонентов взаимодействовать с одной или большим числом компонентов внутренней и внешней среды.

Защищённость - способность компонентов программного средства защищать программы и информацию от любых негативных воздействий.

Надёжность - обеспечение комплексом программ достаточно низкой вероятности отказа в процессе функционирования программного средства в реальном времени.

Эффективность - свойства программного средства, обеспечивающие требуемую производительность решения функциональных задач, с учётом количества используемых вычислительных ресурсов в установленных условиях.

Практичность (применимость) - свойства программного средства, обусловливающие сложность его понимания, изучения и использования, а также привлекательность для квалифицированных пользователей при применении в указанных условиях.

Сопровождаемость - приспособленность программного средства к модификации и изменению конфигурации и функций.

Мобильность - подготовленность программного средства к переносу из одной аппаратно-операционной среды в другую.

Тестирование ПО. Ещё немного о качестве.

Качество программного продукта характеризуется набором свойств, определяющих, насколько продукт хорош с точки зрения всех заинтересованных сторон: заказчика продукта, спонсора, конечного пользователя, разработчиков и тестировщиков продукта, инженеров поддержки, сотрудников отделов маркетинга, обучения и продаж.

Каждый из участников может иметь различное представление о продукте и о том, насколько он хорош или плох, то есть о том, насколько высоко качество продукта.

Таким образом, постановка задачи обеспечения качества продукта переходит в задачу определения заинтересованных лиц, их критериев качества и нахождения оптимального решения, удовлетворяющего этим критериям. Тестирование является наиболее простым способом обеспечения качества.

Тестирование ПО. Что тестировать?

Тестировать можно (и нужно!):

Программы при их непосредственном запуске и исполнении (software).
Код программ без запуска и исполнения (code).
Прототип программного продукта (product prototype).
Проектную документацию (project documentation):

- Требования к программному продукту (product requirements).
- Функциональные спецификации к программному продукту (functional specifications).
- Документы, описывающие архитектуру (product architecture), дизайн (product design).
- План проекта (project plan) и тестовый план (test plan).
- Тестовые сценарии (test cases).

Сопроводительную документацию (и документацию для пользователей):

- Интерактивную помощь (on-line help).
- Руководства по установке (Installation guide) и использованию программного продукта (user manual).

Необходимость тестирования того или иного рабочего продукта (work product) определяется конкретным проектом и условиями его разработки.

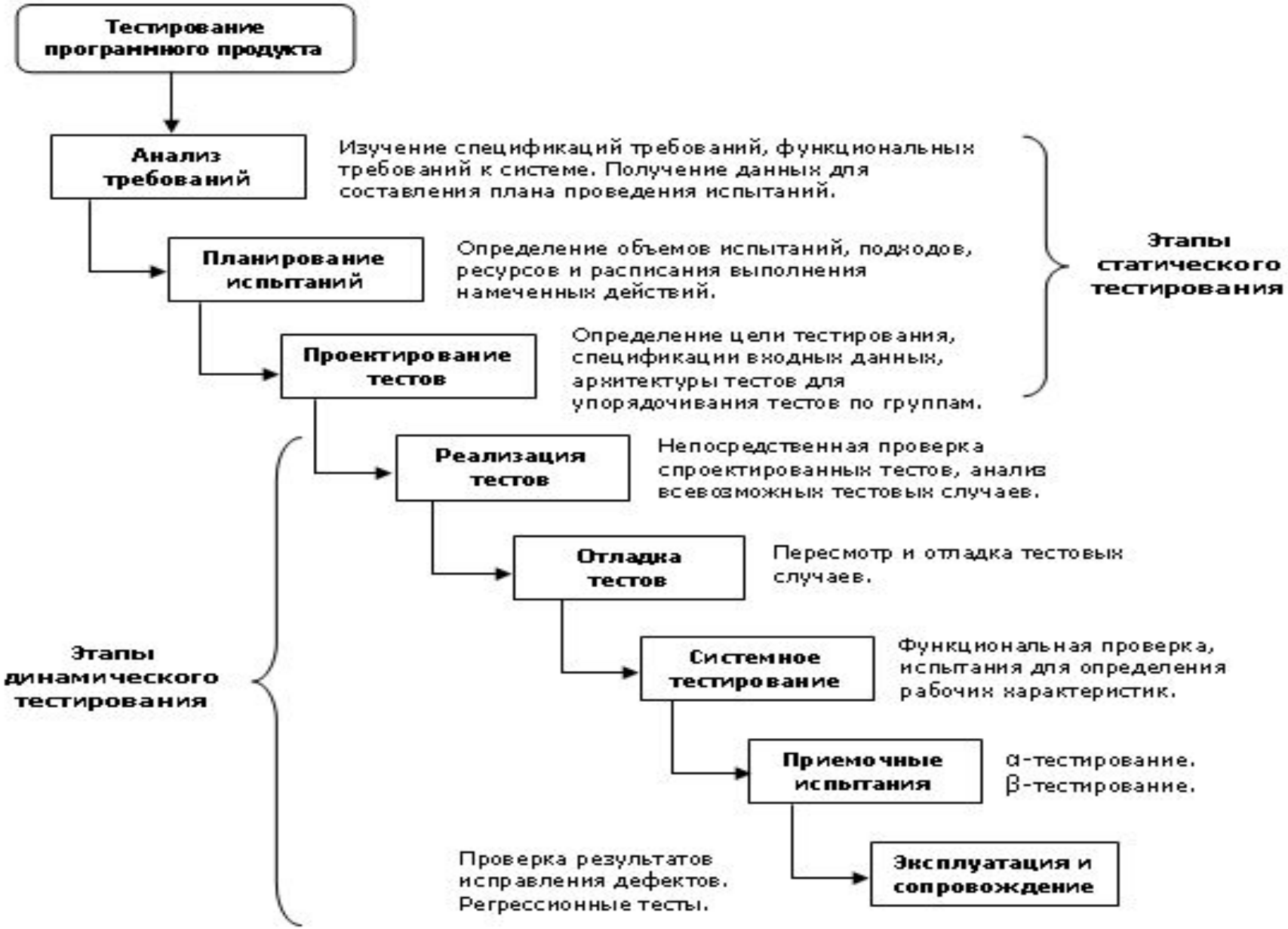
Статическое и динамическое тестирование

Существуют два основных направления тестирования.

Статическое тестирование (static testing) - это процесс анализа самой разработки программного обеспечения, иными словами – это тестирование без запуска программы. Статическое тестирование предусматривает проверку программного кода, требований к программному продукту, функциональной спецификации, архитектуры, дизайна и т.д. Статическое тестирование является одним из наиболее эффективных способов выявления дефектов на ранних стадиях работы над проектом, благодаря чему достигается существенная экономия времени и затрат на разработку.

Динамическое тестирование (dynamic testing) - это тестовая деятельность, предусматривающая эксплуатацию (запуск) программного продукта. Динамическое тестирование включает в себя запуск программы, выполнение всех её функциональных модулей и сравнение её фактического поведения с ожидаемым.

Статическое и динамическое тестирование



Этапы тестирования (1/8)

Анализ требований

Процесс статического тестирования начинается с определения и анализа требований к системе. Изучаются существующие материалы и методические вопросы (техническое задание на разработку, рабочий и технический проект, пр.), уясняются основные используемые понятия, термины и определения, ожидаемые функциональные требования к системе, такие как требования к интерфейсу (описывает входы, получаемые из внешних систем, и выходы, направляемые во внешние системы. Накладываются ли на эти интерфейсы какие-то ограничения?), данным (описывают входные и выходные данные системы, формат данных, их хранение), производительности (описывают проблемы масштабирования и синхронизации, например, сколько пользователей одновременно должна обслуживать система), требования к пользователям и человеческому фактору (кто будет работать с программным обеспечением, учёт необходимого уровня удобства и простоты использования), физическим средствам тестирования (операционная система под управлением которой выполняется программный продукт, и вычислительная платформа, на которой эксплуатируется система), безопасности (доступ к программному продукту и управление данными), документации (определяется в каком виде она должна быть и должна ли быть вообще), устранению неисправностей (реакция системы на неисправности), сопровождению (определяется, как производится устранение проблем обнаруженных в системе).

Этапы тестирования (2/8)

Планирование испытаний

Для того, чтобы тестирование было эффективным, необходимо потратить значительные средства и усилия на планирование. На этапе планирования определяется стратегия тестирования, производится оценка времени для проведения тестовых работ, определяется состав и структура испытательной системы (выявление аппаратных и программных средств тестирования), подготовка и утверждение плана проведения испытаний. План проведения испытаний даёт описание подхода, который предусматривается задействовать при проведении тестирования, а также объём трудозатрат на тестирование.

Этапы тестирования (3/8)

Проектирование тестов

На этапе проектирования и разработки тестов определяются цели тестов, спецификации для ввода каждого теста, тестовая конфигурация. Производится автоматизация часто используемых тестов, требующих больших затрат времени. Разрабатывается методика тестирования, обладающая требуемым уровнем детализации. В результате выполнения этих действий будет получен набор тестовых случаев (test cases), который может использоваться для проведения испытаний.

Этапы тестирования (4&5/8)

Реализация и отладка тестов

После этапа проектирования тест необходимо проверить на наличие дефектов с целью их немедленного устранения, и затем испытать на некоторой сборке программного продукта. После прогона тестов необходимо провести анализ неудачных исходов прогона тестов, выяснить причину источника неудачи теста - программный код или сам тест.

Этапы тестирования (6/8)

Системное тестирование

Входными данными для системного тестирования является набор отлаженных тестов. Системное тестирование проводится для удостоверения того, что программное обеспечение делает именно то, что от него ожидает пользователь. Используются два основных типа испытаний: функциональная проверка и испытания для определения рабочих характеристик.

Функциональная проверка использует набор тестов, которые определяют, выполняет ли система всё то, что она должна делать с точки зрения пользователя. Производится тестирование функций программного продукта (т.е. поиск различий между разработанной программой и её спецификацией). Проводится многократное тестирование каждой функции системы, множества их сочетаний, имитация и проверка на наличие сбоев, применительно к каждому функциональному требованию.

После того, как тестирование подтвердило адекватность базовой функциональности системы, задачей тестирования становятся испытания для определения рабочих характеристик. В рамках данной задачи выполняется тестирование в предельных режимах, нагрузочные испытания (так например, если программа рассчитана на подключение к ней большого числа пользователей, то её необходимо тестировать при максимально возможной нагрузке. Для программ, выполняющих операции «запись-чтение», проводится тестирование операций обработки больших массивов данных с оценкой изменения времени отклика для выявления коэффициента пропорциональности падения быстродействия с увеличением количества обрабатываемых данных. Проводятся испытания на надёжность и эксплуатационную готовность. Также проводится ряд дополнительных испытаний, таких как проверка безопасности, установочная проверка, проверка на совместимость, проверка удобства и простоты использования.

Этапы тестирования (7/8)

Приёмочные испытания

По завершении системного тестирования продукт может быть передан пользователю для проведения приёмочных испытаний (acceptance testing). Производится бета-тестирование, также возможен случай, когда заказчик выполняет заранее определенный набор тестовых случаев, имитирующих типовые условия, в которых система будет работать после ввода в эксплуатацию. Заключительным типом приёмочных испытаний является установочная проверка (installation testing), по условиям которой завершённая версия программного продукта устанавливается на площадках заказчика с целью получить от него подтверждение, что программный продукт соответствует всем требованиям.

Этапы тестирования (8/8)

Сопровождение

Сопровождение обозначает проверку результатов исправления дефектов, которые были найдены заказчиком в процессе эксплуатации программного продукта, тестирование расширенных функциональных возможностей и выполнение регрессионных тестов на новых версиях программного продукта. Цель всего этого заключается в получении подтверждения того, что ранее исправно работавшие функциональные средства не пострадали от внесения изменений в программный продукт, и новые добавленные функции также работают корректно.

Методы тестирования. Белый ящик.

Для тестирования программного кода без его непосредственного запуска применяется метод белого ящика (white-box testing, glass-box testing).

При тестировании с использованием метода белого ящика тестировщик имеет доступ к исходному коду ПС и может писать код, который связан с библиотеками тестируемого ПС. Это типично для компонентного тестирования (юнит-тестирования, unit-testing), при котором тестируются только отдельные части системы. Такие тесты основаны на знании кода приложения и его внутренних механизмов.

Метод белого ящика часто используется на стадии, когда приложение ещё не собрано воедино, но необходимо проверить каждый из его компонентов, модулей, процедур и подпрограмм.

Также отметим, что компонентным тестированием (unit testing), чаще всего занимается программист, хорошо понимающий код, или тестировщик, имеющий прекрасные знания в области программирования.

Методы тестирования. Чёрный ящик.

При использовании метода чёрного ящика (black-box testing) тестировщик имеет доступ к ПО только через те же интерфейсы, что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования.

Например, тестирующий модуль может виртуально нажимать клавиши или кнопки мыши в тестируемой программе с помощью механизма взаимодействия процессов. Эти события вызывают тот же отклик, что и реальные нажатия клавиш и кнопок мыши.

Как правило, тестирование чёрного ящика ведётся с использованием спецификаций или иных документов, описывающих требования к системе.

Тестировщик тестирует программу так, как с ней будет работать конечный пользователь, и он ничего не знает о внутренних механизмах и алгоритмах, по которым работает программа.

Другими словами, он запускает приложение на выполнение и тестирует его функциональность, используя пользовательский интерфейс для ввода входных и получения выходных данных.

Но как при этом обрабатываются входные данные, он не знает. Цель данного метода – проверить работу всех функций приложения на соответствие функциональным требованиям.

Методы тестирования. Ч и Б ящики.

Основная разница между тестированиями по методу чёрного и белого ящиков состоит в том, что метод чёрного ящика может скрыть проблемы, которые метод белого ящика обнаружит. Так, метод чёрного ящика может не сообщить о некорректном функционировании объекта, потому что проблемы в работе оказались незаметны.

Метод белого ящика может обнаружить этот некорректный объект или функцию, проведя их по специальному пути исполнения кода. Этот метод позволяет тестировщику заглянуть внутрь «чёрного ящика» и сосредоточиться на внутренней информации, которая и определяет поведение системы, позволяет управлять выбором нужных тестовых данных.

Как вы думаете, каковы основные преимущества метода белого ящика? А чёрного ящика?

Методы тестирования. Серый ящик.

Существует также метод серого ящика (gray box testing), который представляет собой нечто среднее между методами белого и чёрного ящиков. Этот метод, как правило, используется при тестировании веб-приложений, когда тестировщик знает принципы функционирования технологий, на которых построено приложение, но может не видеть кода самого приложения.

Т.о., говоря о тестировании методом чёрного ящика, мы говорим о функциональном тестировании (functional testing). Функциональное тестирование ещё называют поведенческим или тестированием на поведенческом уровне. Тестировщики, занимающиеся функциональным тестированием, используют преимущественно метод чёрного ящика, а программисты, перед тем как передать приложение в тестирование, проверяют свои модули методом белого ящика.

Функциональное тестирование

Говоря о функциональном тестировании, мы говорим об одном из процессов жизненного цикла программного продукта, который проводится с целью получения объективных доказательств функционирования программного продукта в соответствии с установленными либо подразумеваемыми заказчиком требованиями к программному продукту. В процессе функционального (а также любого другого) тестирования тестировщик ищет дефекты (defect, bug).

Что же такое “баг”? Это изъян в работе программного продукта, который вызывает несоответствие ожидаемых результатов его выполнения и фактически полученных результатов. То есть, это невыполнение требования, связанного с предполагаемым или установленным использованием. Дефект может возникнуть на стадии формулирования требований, на стадии проектирования (вот почему так важно применять статическое тестирование), или на стадии кодирования, или же его причина может крыться в некорректной конфигурации или данных. Дефектом может быть так же что-то другое, что не соответствует ожиданиям заказчика и что может быть, а может и не быть определено в спецификации к программному продукту. **Как вы думаете, какие аспекты поведения программного продукта никогда не отражаются в требованиях?**

Функциональное тестирование подразделяется на ручное (manual testing) и автоматическое или автоматизированное (automated testing). Ручное тестирование подразумевает выполнение тестов вручную. В свою очередь автоматизированное тестирование подразумевает привлечение специальных инструментов для автоматизации тестирования.

Цели функционального тестирования

Для чего тестируют программы? Для того чтобы найти в них ошибки, и чем больше и чем более серьёзные ошибки будут найдены, тем лучше. Времени на подготовку к тестированию и само тестирование, как правило, всегда недостаточно, поэтому процесс тестирования должен быть организован максимально эффективно. А для этого надо правильно спланировать, расставить приоритеты задачам, составить грамотные тестовые случаи сценарии (test cases and test scenarios). При этом надо помнить, что если тест позволил выявить проблему, значит он успешный. А тест, не выявивший проблем, был потерей времени.

Бытует мнение, что конечной целью проведения тестирования является не просто обнаружение ошибок, но и полное избавление тестируемого объекта от всех ошибок, доказательство того, что все ошибки найдены и больше их нет. Истинное положения вещей несколько иное. Протестировать программу полностью невозможно. Это убедительно доказывает Сем Канер с соавторами в книге «Тестирование программного обеспечения», приводя в качестве примера небольшую программу, описанную Гленфордом Майерсом (Glenford Myers) ещё в 1979 году и состоящую из цикла с несколькими операторами IF (примерно 20 строк кода), на полное тестирование которой ушел бы миллион лет (100 триллионов путей выполнения).

С базовыми методиками тестирования, позволяющими сократить время тестирования до реальных сроков разработки, мы познакомимся позже. А пока можно с уверенностью сказать, что **целью тестирования является обнаружение ошибок в тестируемом объекте**, а не доказательство их отсутствия.

Говоря более формальным языком, целями процесса функционального тестирования в области качества являются:

- Подтверждение качества программного продукта, соответствующего показателям, установленным заказчиком.
- Обнаружение и документирование дефектов программного продукта.
- Принятие объективного решения, зафиксированного в отчёте о результатах тестирования, о возможности поставки программного продукта заказчику.

Виды тестирования (1/6)

Инсталляционное тестирование (installation testing)

В процессе инсталляционного тестирования проверяется корректность установки и удаления программного продукта в среде, максимально приближенной к эксплутационной. Об этом аспекте корректной работы программного обеспечения очень часто просто забывают (и напрасно). Правильно выполненная установка программы -- необходимое условие её корректной дальнейшей работы.

Проверка правильности установки должна быть обязательным элементом проекта по тестированию любого продукта. Если программу невозможно корректно установить, и при этом что-то не будет работать или будет работать неправильно, работа по тестированию самого программного тестирования бессмысленна. Почему? Потому что заказчику не нужен продукт, который даже невозможно установить. Если пользователь уже на этапе установки сталкивается с проблемами в разработанном программном продукте, что он подумает о самом программном продукте? Будет ли он связываться с такой фирмой-разработчиком?

Виды тестирования (2/6)

Регрессионное тестирование (regression testing)

Повторное выполнение тестов для проверки того, что изменения, внесённые в программу в результате разработки новой или изменения существующей функциональности, устранения ошибок, не повлияли на функциональность, которая не изменялась (т.е. текущая версия ведёт себя идентично предыдущей, за исключением измененных областей).

Виды тестирования (3/6)

Тестирование новой функциональности (new feature testing)

В данном виде тестирования акцент делается на тестировании новой функциональности, появившейся в конкретном выпуске (build) программного продукта.

Виды тестирования (4/6)

Конфигурационное тестирование (configuration testing)

С помощью конфигурационных тестов проверяется совместимость продукта с различным программным (software) и аппаратным (hardware) обеспечением. Как правило, программный продукт делается с тем расчётом, чтобы он сразу работал в максимально разнообразной внешней среде. Если же речь идёт о «коробочном продукте», то фактор совместимости приобретает ещё более важное значение. Для того, чтобы выяснить реакцию продукта на окружение и соседство с другим программным обеспечением, и проводят данные тесты.

Виды тестирования (5/6)

Тестирование совместимости (compatibility testing)

Тестирование совместимости помогает убедиться в функциональных возможностях и надёжности работы продукта в поддерживаемых браузерах (если речь идет о Web-приложениях) и операционных системах. Также может проверяться работоспособность продукта при использовании различных аппаратных платформ.

Виды тестирования (6/6)

Тестирование удобства эксплуатации (usability testing)

Тестирование интерфейса человек/машина производится в отношении таких моментов как внешний вид пользовательского интерфейса, удобство навигации (преимущественно для Web-сайтов). Практичность и удобство использования – очень важные характеристики программного продукта. Например, программа может вполне соответствовать всем предъявляемым к ней требованиям с точки зрения функциональности. Но функции реализованы неудобно: некоторые шаги приходится повторять много раз, тогда как по логике достаточно выполнить однажды; расположение элементов интерфейса нелогично, программа быстро вызывает утомление и т.д. Для выявления такого рода недочётов и применяют тесты на удобство использования. Часто эта группа тестов относится к категории некритичных, но когда речь идёт, например, о рыночном готовом продукте, пренебрегать удобством эксплуатации весьма опасно.

Виды тестирования. Их много…

Перечисленные выше виды тестирования – базовый набор, но далеко не полный. В зависимости от назначения системы испытаниям подвергаются различные аспекты её функциональности, в соответствии с приоритетами задач, которые система должна решать.

Также выделяют следующие виды тестирования:

- Тестирование прототипа (prototype testing)
- Интеграционное тестирование (integration testing)
- Тестирование безопасности (security testing)
- Тестирование интернационализации (internationalisation testing)
- Локализационное тестирование (localisation testing)
- Компонентное тестирование (unit testing)
- Системное тестирование (system testing)
- Исследовательское тестирование (exploratory testing)
- Тестирование документации (documentation testing)
- Тестирование производительности (performance testing)
- Нагрузочное тестирование (load testing)
- Стрессовое тестирование (stress testing)

И т.д. и т.п. …

Виды тестирования (продолжение) (1/12)

Тестирование прототипа (prototype testing)

Это метод выявления структурных, логических ошибок и ошибок проектирования на ранней стадии развития продукта до начала фактической разработки. Основная цель тестирования прототипа – выявить потенциальные проблемы в приложении, проверить, насколько приложение соответствует потребностям и ожиданиям пользователя и обнаружить расхождения с требованиями к графическому интерфейсу пользователя.

Тестирование прототипа охватывает следующие аспекты:

Структура приложения;

Формы;

Прототип бизнес-логики;

Логические связи между модулями;

Навигация;

Графический интерфейс пользователя.

Под тестированием прототипа также может пониматься исследование готового (старого или аналогичного) продукта, новую (или альтернативную) версию которого планируется разработать. Такое тестирование позволяет быстро сформировать перечень основных требований к продукту.

Виды тестирования (продолжение) (2/12)

Интеграционное тестирование (integration testing)

Такой вид тестирования может представлять два направления деятельности:

1) Проверку того, как отдельные модули приложения взаимодействуют между собой.
2) Проверку того, как приложение взаимодействует с каким-то внешним приложением или компонентом системы.

Виды тестирования (продолжение) (3/12)

Тестирование безопасности (security testing)

Тестирование безопасности представляет собой ряд работ: от разработки политики безопасности до тестирования безопасности на уровне приложения, операционной системы и сетевой безопасности.

Тестирование безопасности может иметь различную степень покрытия:

Первичное тестирование безопасности.

Полное тестирование приложения.

Полное тестирование приложения и сервера.

Для тестирования безопасности на начальном уровне применяются специальные утилиты, способные проверить приложение или систему на наличие элементарных уязвимостей (список которых, однако, может включать тысячи и десятки тысяч пунктов).

На более глубоком уровне тестированием безопасности занимаются эксперты, т.к. данный вид тестирования требует глубочайших знаний особенностей технологий, с использованием которых разработано приложение.

Виды тестирования (продолжение) (4/12)

Тестирование интернационализации (internationalisation testing)

Этот вид тестирования проверят готовность приложения к работе с различными языковыми интерфейсами. В частности, проверяется способность корректно отображать шрифты, пункты меню, производить поиск, сортировку, способность приложения обрабатывать файлы, поименованные на различных языках.

Следующей стадией, как правило, является локализационное тестирование.

Виды тестирования (продолжение) (5/12)

Локализационное тестирование (localisation testing)

Проверяет, насколько корректно продукт адаптирован к работе на том или ином языке: всё ли переведено и переведено правильно, не нарушилась ли логика построения интерфейса и обработки данных и т.д.

Для локализационного тестирования рекомендуется обязательно приглашать в команду носителя того языка, перевод на который тестируется.

В противном случае мы рискуем увидеть нечто подобное:

Пункт меню: «Сделать под себя»

(«Customise»)

Огромная красная кнопка «ВСЁ!»

(«Finish»)

Пункт меню «Ясные печенья»

(«Clear cookies»)

Виды тестирования (продолжение) (6/12)

Компонентное тестирование (unit testing)

С термином unit testing связана некоторая путаница. Во-первых, на русский его переводят и как «модульное тестирование», и как «компонентное тестирование». Во-вторых, под этим термином кроется два достаточно различных вида деятельности:

1) Если мы говорим и «компонентном тестировании» (здесь уместнее этот перевод) в общем смысле (в контексте «тестирования вообще»), мы имеем в виду тестирование отдельных частей приложения. Причём эти части могут быть достаточно крупными.
2) Если мы говорим о «модульном тестировании» (здесь уже лучше использовать такой перевод) в контексте автоматизации тестирования, мы имеем в виду тестирование отдельных участков кода, классов, методов.

Виды тестирования (продолжение) (7/12)

Системное тестирование (system testing)

Системное тестирование охватывает целиком всю систему. Большинство функциональных сбоев должно быть идентифицировано еще на уровне модульных и интеграционных тестов. В свою очередь, системное тестирование, обычно фокусируется на нефункциональных требованиях – безопасности, производительности, точности, надёжности т.п. На этом уровне также тестируются интерфейсы к внешним приложениям, аппаратному обеспечению, операционной среде и т.д.

Виды тестирования (продолжение) (8/12)

Исследовательское тестирование (exploratory testing)

Как правило, многие об этом тестировании слышали, но мало кто видел. Причина очень проста: на него почти никогда не хватает времени. Это тестирование относится к категории «невоспроизводимого» («non-reproducible testing», «on-the-fly testing»), т.е. производится интуитивно, без заранее подготовленных тест-кейсов.

Цель такого тестирования – на основе профессиональной интуиции тестировщика найти в приложении ошибки, которые могли быть пропущены классическими подходами к тестированию.

В процессе исследовательского тестирования также можно изучить особенности работы приложения в тех или иных условиях и внести поправки в документацию.

Виды тестирования (продолжение) (9/12)

Тестирование документации (documentation testing)

Вид тестирования, с которого начинается почти любой проект. Призвано обнаружить ошибки в документации. Эти ошибки опасны тем, что они, как маленький комок снега могут вызвать лавину проблем, вырастая на более поздних стадиях работы с проектом в очень сложноустранимые и дорогостоящие последствия.

Виды тестирования (продолжение) (10-12/12)

Тестирование производительности (performance testing)

Нагрузочное тестирование (load testing)

Стрессовое тестирование (stress testing)

Эти три вида тестирования тесно связаны между собой и перетекают в некоторых случаях друг в друга.

Тестирование производительности проверяет способность программы выполнять заданное количество операций в заданный промежуток времени.

Нагрузочное тестирование проверяет способность приложения работать при запланированной нагрузке.

Стрессовое тестирование проверяет поведение приложения в условиях, выходящих за рамки оговоренных для эксплуатации приложения.

Уровни тестирования (1/3)

Приёмочный тест (smoke test)

Название этого теста (smoke test) пошло к нам из времён тестирования различных электроприборов. К примеру, радиоприёмник включался в сеть и, если дым не пошёл, проводилось дальнейшее тестирование. Таким образом, приёмочный тест – это самый первый и короткий тест, проверяющий работу основной функциональности программного продукта.

Данный тест длится от получаса до 2-3-х часов максимум в зависимости сложности программы. По результатам этого теста принимается решение о целесообразности дальнейшего тестирования.

Если программа не прошла приёмочный тест, она отправляется на доработку к программистам. Как можно детально тестировать функциональность, если ПО в принципе не работает? Именно поэтому нет смысла выполнять дальнейшее тестирование, пока серьёзные проблемы не будут устранены.

Давайте обсудим, какие функции наиболее критичны, например, для текстового редактора типа Microsoft Word.

Уровни тестирования (2/3)

Тест критического пути (critical path test)

Уровень тестирования, во время которого проверяется основная функциональность программного продукта, критичная для конечного пользователя, при стандартном его использовании.

В рамках данного тестирования, как правило, проверяется большинство требований, предъявляемых к программному продукту.

Давайте обсудим, какие функции уже упомянутого Microsoft Word будут относиться к этому уровню тестирования.

Уровни тестирования (3/3)

Расширенный тест (extended test)

Это углубленный тест, при котором проверяется нестандартное использование программного продукта. Прогоняются сложные, логически запутанные сценарии, совершаются действия, которые конечный пользователь будет совершать редко. Зачем выполнять такие тесты? Ответ прост: потому, что программа должна корректно реагировать на любые, даже самые случайные действия пользователя и работать надёжно в любой ситуации.

Другое дело, что не для всех программ можно выполнить подобные тесты, особенно, если нет времени, поскольку степень надёжности определяется спецификой области, целей и задач, поставленных перед продуктом.

Например, программы медицинской, финансовой, военной, космической направленности должны быть максимально надёжными. А программы, предназначенные для домашнего использования, могут и не быть столь надёжными, и соответственно нет смысла тратить слишком много времени на разработку и прогон изощрённых тестовых сценариев.

Давайте приведём парочку примеров функций Microsoft Word, которые будут относиться к этому уровню тестирования.

Стадии процесса тестирования

Выделяют следующие стадии процесса тестирования:

- Инициирование
- Планирование
- Разработка тестов
- Выполнение тестов
- Анализ результатов и написание отчётов
- Завершение

Стадии процесса тестирования (1/6)

Инициирование

Процесс начинается с момента получения ресурс-менеджерами письменного запроса от менеджера проекта на выделение требуемых ресурсов для тестирования программного продукта. После согласования указанных ресурсов, ресурс-менеджер назначает специалистов, ответственных за тестирование этого проекта.

Назначенные специалисты по функциональному тестированию принимают участие во вступительном собрании и приступают к анализу проектной документации, на основании которого ведущий специалист по тестированию, при необходимости, разрабатывает рекомендации по корректировке документации для обеспечения возможности полноценного тестирования программного продукта и посылает их менеджеру проекта.

Стадии процесса тестирования (2/6)

Планирование

Ведущий специалист по тестированию разрабатывает на основе анализа проектной документации тестовый план и согласовывает его с менеджером проекта. После чего он посылает план на утверждение руководителю группы и публикует утвержденный план. Начальная версия тестового плана должна быть направлена на утверждение руководителю группы до начала тестирования программного продукта.

Стадии процесса тестирования (3/6)

Разработка тестов

Ведущий специалист по тестированию распределяет обязанности по тестированию программного продукта среди команды специалистов, выделенных для данного проекта, в соответствии с которыми, все участники тестирования разрабатывают тестовые сценарии для закреплённых за ними областей тестирования программного продукта. Ведущий специалист на их основе формирует тестовый сценарий для программного продукта в целом и согласует его с менеджером проекта.

Тестовый сценарий должен быть направлен на утверждение руководителю группы до начала тестирования программного продукта. Тестовый сценарий может выполняться как вручную, так и с использованием средств автоматического тестирования. Часто приёмочный тест выполняться автоматическим способом.

Тестовые сценарии, как правило, учитываю следующие, уже известные нам, уровни тестов:

- Приёмочный тест (smoke test)
- Тест критического пути (critical path test)
- Расширенный тест (extended test)

Стадии процесса тестирования (4/6)

Выполнение тестов

При получении сообщения о выпуске новой версии программного продукта ведущий специалист по тестированию проверяет соответствие конфигурации версии программного продукта проектной документации, команда тестирования приступает к инсталляции продукта на выделенном тестовом оборудовании, уточняет конфигурационную информацию (проверяет номер версии ПП, наличие требуемых компонентов и т.п.) и проводит приёмочный тест.

В ходе приёмочного и последующих уровней тестов специалисты по тестированию проверяют соответствие исправленных дефектов, заявленных группой разработки программного продукта их реальному состоянию. Кроме того каждый обнаруженный в программном продукте дефект должен незамедлительно документироваться и отправляться в группу разработки.

По результатам приёмочного теста ведущий специалист по тестированию принимает решение о прохождении теста программным продуктом. Если тест не пройден, процесс тестирования данной версии заканчивается. При положительном решении команда тестирования приступает к критическому и расширенному тестам, которые могут длиться продолжительное время (вплоть до нескольких рабочих дней). Как правило, данный процесс продолжается до момента выпуска новой версии программного продукта.

Стадии процесса тестирования (5/6)

Анализ результатов и написание отчётов

Ведущий специалист по тестированию, как правило, в конце каждой рабочей недели создаёт отчёт о результатах тестирования версии (нескольких версий) программного продукта и публикует его. В случае, когда в течение рабочей недели версии ПП не выпускались, допускается не создавать данный отчёт, при этом вся статистическая информация за данный период должна быть включена в следующий отчёт.

Ведущий специалист по тестированию принимает участие в регулярных, как правило, еженедельных (хотя менеджер проекта может установить иную периодичность) обсуждениях состояния проекта, а также в завершающем собрании по проекту.

Стадии процесса тестирования (6/6)

Завершение

После того, как качество программного продукта становится достаточным (это проверяется по принятым метрикам, о которых мы поговорим чуть позже), принимается решение о завершении работы над проектом и передаче его заказчику («выпуске в продакшн»).

Создаётся обобщающий отчёт за весь период работы над проектом, проводится общее заключительное собрание.

Жизненный цикл ПО

Для более глубокого понимания места процесса тестирования в разработке ПО, познакомимся с этапами жизненного цикла ПО. Он включает:

- выработку требований;
- разработку спецификаций;
- общее проектирование;
- проектирование архитектуры;
- детальное проектирование;
- реализацию и кодирование;
- интеграцию;
- сертификацию;
- внедрение;
- сопровождение.

Жизненный цикл ПО: затраты

Аналитики отмечают, что распределение затрат по стадиям жизненного цикла ПО примерно таково:

Анализ требований 3%

Спецификация 3%

Проектирование 5%

Кодирование 7%

Тестирование 15%

Промышленное производство и сопровождение 67%

Жизненный цикл ПО: подробнее (1-3/10)

Выработка требований является начальной стадией работы с любым проектом. Здесь происходит интенсивное взаимодействие с заказчиком, интервьюирование, анкетирование, исследование прототипов продукта (если они есть). Основная цель этой стадии – получить представление о продукте, его задачах, особенностях эксплуатации и т.д., достаточное для начала разработки спецификаций.

Разработка спецификаций – следующий шаг, на котором требования детализируются и описываются формальным образом. Основная цель этой стадии – учесть все требования заказчика в технически грамотной, непротиворечивой, полной и понятной форме.

Общее проектирование подразумевает экспертное исследование путей и возможностей реализации программного продукта, в т.ч. анализ возможных аппаратных и программных платформ, сред разработки, выбор технологий разработки и т.д.

Жизненный цикл ПО: подробнее (4-7/10)

Проектирование архитектуры подразумевает создание общей концепции ПС: разбиение его на модули, определение способов и технологий их взаимодействия, выбор общих моделей и алгоритмов работы ПС.

Детальное проектирование позволяет уточнить архитектуру до уровня чётко расписанной иерархии классов, наборов файлов, конкретных моделей, алгоритмов, протоколов взаимодействия и т.д. и т.п.

Реализация и кодирование подразумевают создание кода, реализующего требования к ПС, на основе полученной выше проектной документации.

Интеграция может быть представлена в двух видах:

А) интеграция отдельных модулей ПС, разработанных на предыдущей стадии;

Б) интеграция разработанного ПС в некий программный или аппаратно-программный комплекс.

Часто, стадия Б следует за стадией А.

Жизненный цикл ПО: подробнее (8-10/10)

Сертификация происходит в форме проверки (как правило, внешними организациями) соответствия разработанного ПС требуемым стандартам.

Внедрение («развёртывание») происходит при совместном участии разработчика и представителей заказчика. Цель этой стадии – удостовериться, что разработанный программный продукт корректно работает в реальном окружении и в полной мере соответствует требованиям заказчика.

Сопровождение ПС выражается в устранении найденных на стадии эксплуатации недостатков ПС, а также в доработке ПС с целью расширения его функциональных возможностей (если таковое было оговорено в контракте).

Отметим, что последние три стадии присутствуют не во всех проектах.

Стадия планирования: подробнее

Стадия планирования включает в себя следующие этапы:

Определение целей. Описывается общее видение продукта. Могут быть специфицированы требования к пользовательскому интерфейсу, надёжности, производительности. Формируется документ под названием «Общее видение» («Scope vision», чтобы дать команде разработчиков возможность в общих чертах представить проект.

Анализ требований. На данном этапе, в основном, речь идёт об общих бизнес-требованиях. Детализированные спецификации разрабатываются чуть позже.

Определение функциональных характеристик ПП. Как и в случае с бизнес-требованиями, на данном этапе не идёт речь о чётком описании реализации функционала. Здесь лишь прописывается общий набор функций, которые может должно приложение, а также даются ключевые характеристики этих функций.

Стадия планирования: тестирование

На этапе планирования, как легко можно догадаться, уже начинается тестирование.

Тестируются на разумность и выполнимость различные идеи.

К тестированию привлекаются специалисты по маркетингу, руководители проекта, главные конструкторы, специалисты по анализу человеческого фактора.

Проводится анализ и оценка требований к продукту:

- Адекватны и выполнимы ли требования.
- Насколько требования полны.
- Совместимы ли требования между собой.

Проводятся собрания, на которых обсуждаются текущие задачи и дальнейшие перспективы работы с проектом.

Стадия планирования: методы тестирования

Проведение сравнительного анализа существующих продуктов.

Изучение рабочих копий, демо-версий и описаний продуктов конкурентов (существующих и анонсированных).

Формирование и пересмотр списков функций с целью устранения из него не согласующихся с концепцией разрабатываемого ПП.

Организация дискуссионных групп.

Привлечение экспертов и аналитиков к анализу бизнес-требований.

Стадия проектирования: подробнее

Стадия проектирования включает в себя следующие этапы:

Разработка пользовательского интерфейса. Разрабатывается описание пользовательского интерфейса («user interface», UI), включая экранные формы. Однако, стоит помнить, что UI в процессе развития проекта может ещё неоднократно измениться.

Разработка архитектуры и дизайна. Определяется набор программных модулей (программная архитектура), структура, взаимосвязи и принципы хранения и обработки данных и алгоритмы работы программы (программный дизайн).

Затем производится декомпозиция (разбиение на меньшие элементы) разработанных моделей до отдельных модулей, наборов классов, отдельных классов и т.д.

Стадия проектирования: тестирование

На данном этапе также тестируются идеи, но они уже гораздо лучше формализованы и описаны намного подробнее. К работе подключаются узкопрофильные специалисты. На этапе проектирования в центре внимания находятся следующие вопросы:

Действительно ли проект хорош? Будет ли создан хороший продукт?

Соответствует ли проект бизнес-требованиям?

Полон ли проект? Описаны ли все модули и взаимосвязи между ними?

Достаточно ли проект реалистичен? Исследуются требования к аппаратным и программным ресурсам, инструментальным средствам и т.д.

Хорошо ли описана в проекте подсистема обработки ошибок?

Что можно улучшить в проекте?

Что мы упустили? :)

Стадия проектирования: методы тестирования

В основном, на данном этапе проводятся совещания аналитиков и специалистов:

Обзорные – демонстрируется модель ПП, показывается, что делает программа с предложенными входными данными.

Инспекционные – анализируется каждый элемент проекта или его аспект на соответствие спецификациям.

Рецензионные – составляется список вопросов, выделяются сомнительные элементы проекта.

Также проводится анализ псевдокода.

Когда начинать и прекращать тестирование?

ПС считается пригодным к выпуску, если в нём устранены все критические ошибки и устранено 85% процентов некритических ошибок. Считается, что в общем случае дальнейшее тестирование экономически нецелесообразно. Таков общепринятый подход.

Экономическая отдача тестирования тем выше, чем на более ранней стадии тестирование выявило дефект. Таким образом, тестирование начинается с анализа требований заказчика. Если удалось выявить противоречия, неточности, неоднозначности и т.д. – а значит, мы сэкономили время работы всей команды.

После того, как требования заказчика выверены, можно передавать их разработчикам и планировать сам процесс тестирования. Пока не готовы первые версии ПС, тестировщики могут использовать время на анализ того, что и как тестировать в данном проекте, и подготовиться к такому тестированию.

Тестирование на ранних этапах призвано, скорее, не искать ошибки в ПО, а искать ошибки в выборе пути написания ПО. Выбор неверной технологии, неверного подхода, неверных инструментальных средств, недостаточное понимание разработчиками требований заказчика – всё это легко устранить в начале процесса разработки, и крайне сложно (и ДОРОГО!) устранять в процессе или в конце работы над ПО.

Основные выводы:

- тестирование даёт тем большую экономическую отдачу, чем на более ранних стадиях работы над проектом оно выявило дефект;
- тестирование имеет смысл прекращать тогда, когда устранены все критические и 85% и более некритических дефектов программы, т.к. дальнейшее тестирование, как правило, является неоправданной статьёй расходов.

Основные сложности тестирования (1/12)

При разработке ПО часто срываются графики работ и наблюдается превышение установленного бюджета. Бывают, к сожалению, даже случаи, когда поставленный программный продукт не отвечает требованиям потребителя, и его никогда не используют. Программные продукты часто просто плохо работают. Как видно из симптомов проблем с ПО, при его разработке возникает ряд принципиальных сложностей.

ПО по своей природе является концептуальным. В отличие от, например, здания или любого другого физического объекта, сложно «посмотреть» на программный продукт и оценить степень его завершённости. Чтобы избежать этой проблемы, фирмы-разработчики создают политики управления конфигурациями (configuration management) и управления изменениями (change management), применение которых позволяет получить достаточно чёткий обобщённый взгляд на все элементы конфигурации, компоненты и подкомпоненты для различных версий продукта.

Основные сложности тестирования (2/12)

Недостаток прозрачности. Программное обеспечение по своей природе является концептуальным. В отличие от, например, здания или любого другого физического объекта, сложно «посмотреть» на программный продукт и оценить степень его завершённости. Чтобы избежать этой проблемы, фирмы-разработчики создают политики управления конфигурациями (configuration management) и управления изменениями (change management), применение которых позволяет получить достаточно чёткий обобщённый взгляд на все элементы конфигурации, компоненты и подкомпоненты для различных версий продукта.

Основные сложности тестирования (3/12)

Недостаток контроля. Поскольку программное обеспечение является нематериальным в физическом смысле, его сложно контролировать. Без точной оценки процесса разработки срываются графики выполнения работ и превышаются установленные бюджеты. Очень сложно оценить объём выполненной и оставшейся работы. Применение уже упомянутых ранее политик управления конфигурациями и изменениями предоставляет механизм управления процессом через определение фактически затраченных и плановых ресурсов и оценку будущих затрат, исходя из объёма выполненной работы.

Основные сложности тестирования (4/12)

Недостаток прослеживаемости (traceability). Отсутствие связи между отдельными событиями проекта приводит к его провалу. Главное преимущество политик управления конфигурациями и изменениями состоит в том, что с их помощью обеспечивается прослеживаемость состояний и взаимных зависимостей версий и семейств продуктов. Ценность прослеживаемости особенно высока в ситуациях, когда в одной из версий продукта возникает проблема, которая оказывает влияние на другие версии этого продукта или на другие продукты. Выполнение одного изменения и его распространение на всю базу ПО экономит много времени и средств.

Отсутствие связи между событиями проекта приводит к тому, что решение одной проблемы увеличивает проблему в другой области или приводит к неудаче в попытке решить аналогичную проблему где-то в другом месте.

Сквозная прослеживаемость («трассировка») выполняемых задач позволяет менеджменту в пределах аудиторской возможности проверить цепочку событий, из за которых возникли сложности, в проекте как в едином процессе. А отслеживание календарного графика выполнения работ позволяет, не затягивая проект, завершать разработку ПО в установленные сроки.

Основные сложности тестирования (5/12)

Недостаток мониторинга. Без «трассировки» и «прозрачности» сложно осуществить мониторинг программных проектов. В такой ситуации руководство не может принимать компетентные решения, поэтому графики продолжают срываться, а затраты продолжают превышать установленный бюджет.

Невозможно выполнить мониторинг проекта, если у менеджера проекта нет инструментальных средств, чтобы следить за фактической разработкой продукта в пределах проекта.

Управления конфигурациями и изменениями позволяет осуществить разносторонний мониторинг процесса. При наличии этих политик мониторинг программных проектов становится простой частью общей задачи управления проектом, менеджеры проектов принимают взвешенные решения, не выбиваясь из графика работ и не превышая бюджет.

Основные сложности тестирования (6/12)

Неконтролируемые изменения. ПО является достаточно гибким, оно представляет результат работы большого коллектива, но у потребителей постоянно возникают новые идеи относительно данного программного продукта. Люди редко просят конструктора моста внести изменения в середине проекта, тогда как пользователи ПО часто обращаются с такими просьбами. Влияние таких изменений может быть колоссально. Поэтому, в крупных проектах рекомендуется использовать инструментальные средства, позволяющие гибко управлять изменениями в проекте, синхронизировать их с уже созданными частями проекта, учитывать в календарном плане и бюджете, а также ясно и чётко доносить их до всей проектной группы.

Основные сложности тестирования (7/12)

Групповой синдром разработчика. Если для разработки проекта требуется более одного разработчика, то возникает проблема группы людей, работающих над одной «базой продукта». Под «базой» может пониматься план проведения испытаний, требования к спецификации или коду, сам код и т.д. Усилия тратятся впустую, если несколько человек работают над одним и тем же файлом, а затем его сохраняют – сохраняются только изменения, записанные тем, кто работал над файлом последним. Самое простое решение этой проблемы лежит в блокировании файла на время работы с ним одним из сотрудников, чтобы предотвратить одновременную работу над ним нескольких человек.

Однако такое решение неприемлемо в больших (десятки, сотни и даже тысячи человек) группах разработчиков. На помощь в такой ситуации приходят «системы контроля версий». Наиболее известными продуктами являются CVS («Concurrent version system») и SVN («Subversion»).

Основные сложности тестирования (8/12)

Множественность версий. Совершенствование базового продукта приводит к выпуску дополнительных версий с самыми последними изменениями. Несмотря на наличие последней модернизированной версии программного продукта, часть пользователей продолжает работать с более ранней версией.

Если в программе обнаружены ошибки, изменения необходимо сделать во всех версиях. Как только в продукте появляются новые свойства, они должны быть доступны для всех пользователей независимо от времени выпуска версии продукта.

К сожалению, из-за колоссального объёма работ, который необходимо выполнить для реализации этого требования, на практике разработчики ограничиваются поддержкой только небольшого набора самых последних версий продукта.

Основные сложности тестирования (9/12)

Семейство программных продуктов. Поскольку программные продукты созданы для того, чтобы предлагать аналогичные функции с помощью неоднородных платформ аппаратного и программного обеспечения, необходимо управлять как программным продуктом вообще, так и ПО на базе определённой платформы.

Если программный продукт работает с четырьмя версиями Windows, тремя версиями Unix, Red Hat Linux и FreeBSD, то для этих девяти платформ необходимы разные процессы установки ПО, настройки, особенности реализации и т.д.

Необходимо чётко осознавать этот момент и делать соответствующие поправки в проектном плане, рассчитывая сроки и стоимость реализации продукта.

Основные сложности тестирования (10/12)

Изменение графика работ. Поскольку технические требования меняются в процессе работы с проектом, должен меняться и график их выполнения. Составление календарного плана с учётом набора функциональных возможностей для версии позволяет менеджерам проектов более точно распределять силы, необходимые для выпуска следующей версии программного продукта.

Статистические данные помогают оценить развитие подобных событий на успешных (или неуспешных) управленческих решений, принятых ранее.

Основные сложности тестирования (11/12)

Изменения штата сотрудников. Во всех организациях сотрудники продвигаются по служебной лестнице, приходят новые работники и увольняются имеющиеся.

Если это происходит в разгар работы с проектом, то с уходом специалиста теряются не только технологические знания, теряются также практические знания по разработке продуктов, на овладение которыми ушло много времени. Новые сотрудники, даже зная технологию, не смогут заниматься разработкой продукта без грамотной документации (требований, описаний процесса и т.д.).

Вывод прост: чем лучше документированы все стороны развития проекта, тем меньше такой проект подвержен рискам, связанным с изменениями состава проектной группы.

Основные сложности тестирования (12/12)

Изменения во внешней среде. Некоторые проекты разрабатываются на протяжении достаточно длительного времени (история знает проекты, разрабатываемые более 15-ти лет). За это время меняются не только требования к проекту, меняется очень многое во всей IT-индустрии. Выходят новые версии операционных систем, появляется новое аппаратное обеспечение, появляются новые технологии.

Для успешного выполнения проекта такие изменения следует учитывать ещё на стадии планирования, чтобы быть готовым принять соответствующие меры, когда такие изменения произойдут.

Психологические аспекты тестирования

Мы познакомились со многими вопросами в контексте тестирования, но пока очень мало сказали о людях, которые им занимаются. Так кто же такой тестировщик? Тестировщик – специалист, который в процессе своей профессиональной деятельности занимается обнаружением ошибок в тестируемом объекте.

Иногда, тестировщиков называют «инженерами по обеспечению качества» (QA engineers). Использование этого термина в данном контексте не совсем верно, так как оно исходит из ошибочного отождествление терминов «тестирование» (testing) и «обеспечение качества» (quality assurance).

Тестирование является одним из проявлений обеспечения качества. Само по себе тестирование призвано выявлять ошибки, в то время как обеспечение качества направлено на устранение ошибок и их предотвращение, что предполагает разработку и соблюдение планов тестирования, стандартов и процессов, которым должны следовать программисты при разработке программного обеспечения, и тестировщики при тестировании программ.

Каков же смысл деятельности тестировщика? Зачем он делает то, что делает?

Тестирование – это не только поиск ошибок, это бесконечное стремление к совершенствованию (нет, не программы!) процесса тестирования. Что приводит к повышению квалификации, т.е. к совершенствованию себя.

Да, на каждой специальности можно повышать квалификацию, но обычно этим занимаются по мере необходимости. Просто именно на данной специальности эта необходимость чувствуется особенно сильно.

Психологические аспекты тестирования

Хороший тестировщик должен обладать следующими психологическими качествами:

- Повышенной ответственностью.
- Хорошими коммуникативными навыками.
- Способностью ясно, быстро, чётко выражать свои мысли.
- Исполнительностью.
- Терпением, усидчивостью, внимательностью к деталям, наблюдательностью.
- Гибким мышлением, хорошей способностью к обучению.
- Хорошим абстрактным и аналитическим мышлением.
- Способностью ставить нестандартные эксперименты.
- Высокими техническими навыками.
- Склонностью к исследовательской деятельности.

Технические навыки тестировщика

Тестировщик (в идеале) должен знать следующие технологии:

Программирование: C/C++/C#, Java, Object Pascal, Visual Basic, JavaScript, VBScript, HTML, .NET.

Администрирование СУБД: Oracle, MS SQL, IBM DB2, Sybase, Informix.

Системное администрирование: Windows, Sun Solaris, HP-UX, IBM AIX, Linux, Free BSD.

Сетевое администрирование: NetWare, Cisco IOS, TCP/IP, IPX/SPX, NetBIOS.

Автоматизированное тестирование: Segue SilkTest and SilkPerformer, Mercury Interactive WinRunner, Quick Test Pro and LoadRunner, JUnit, HTTP Unit.

Рекомендуемая литература (1/3)

1. Основы тестирования программного обеспечения
Котляров В.П., "Интернет-университет информационных технологий - ИНТУИТ.ру" - 2006, 285 стр

2. Автоматизированное тестирование программного обеспечения
Automated Software Testing
Элфрид Дастин, Джефф Рэшка
Мягкая обложка
592 стр., 2005 г. Издательство: Лори. ISBN 5-85582-186-2

3. Тестирование Web-приложений
Automated Web Testing Toolkit. Expert Methods for Testing and Managing Web Applications
Диан Стотлемайер
Мягкая обложка
240 стр., 2003 г. Издательство: Кудиц-образ. ISBN 5-93378-064-2

4. Быстрое тестирование
К. Браун, Р. Калбертсон, Гэри Кобб
Мягкая обложка
384 стр., 2002 г. Издательство: Вильямс. ISBN 5-8459-0336-X

5. Тестирование черного ящика. Технологии функционального тестирования ПО
Black-Box Testing. Techniques for Functional Testing of Software and Systems
Борис Бейзер
Мягкая обложка
320 стр., 2004 г. Издательство: Питер. Серия: Библиотека программиста. ISBN 5-94723-698-2

Рекомендуемая литература (2/3)

6. Отладка приложений
Джон Роббинс
Мягкая обложка
512 стр., 2001 г. Издательство: BHV. Серия: Мастер. ISBN 5-94157-056-2

7. Экстремальное программирование: постановка процесса. С первых шагов и до победного конца
Extreme Programming Applied: playing to win
Кен Ауэр, Рой Миллер
Мягкая обложка
368 стр., 2004 г. Издательство: Питер. Серия: Библиотека программиста. ISBN 5-318-00132-7

8. Практическое руководство по проектированию и разработке пользовательского интерфейса
Practitioner`s Handbook for User Interface Design and Development
Дж. Торрес
Мягкая обложка
400 стр., 2003 г. Издательство: Вильямс. ISBN 5-8459-0367-X

9. Введение в тестирование программного обеспечения
Луиза Тамре
Мягкая обложка
359 стр., 2003 г. Издательство: Вильямс. ISBN 5-8459-0394-7

Рекомендуемая литература (3/3)

10. Экстремальное программирование: разработка через тестирование
Test-driven development: by example
Кент Бек
Мягкая обложка
224 стр., 2003 г. Издательство: Питер. Серия: Библиотека программиста. ISBN 5-8046-0051-6

11. Совершенный код. Мастер-класс
Code complete
Стив Макконнелл
Второе издание. Твердый переплет
896 стр., 2005 г. Издательство: Питер, Русская редакция. Серия: Мастер-класс. ISBN 5-469-00822-3

12. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие
Джон Макгрегор, Девид Сайкс
Диасофт, 2002 г., 432 стр.

13. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений
ДиаСофт, 2001, 544 стр

14. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах
Роман Савин
Дело, 2007, 312 стр.

Практическое задание

Давайте представим себе ситуацию и обсудим её. Заказчик просит разработать графический редактор − аналог Adobe Photoshop.

На каком этапе и с чего вы начнёте процесс тестирования?

Что вы будете проверять в исходной документации к проекту?

Как будут выглядеть smoke test, critical test, extended test?

Когда вы завершите тестирование?

Тест для проверки изученного

1. Дайте определение тестированию программного обеспечения.
2. Что, как правило, можно тестировать в процессе работы над проектом?
3. Перечислите основные методы тестирования и дайте им определения.
4. Назовите основную цель тестирования.
5. Назовите основную цель функционального тестирования.
6. Перечислите основные виды тестирования. Дайте им определения.
7. Перечислите основные уровни тестирования. Дайте им определения.
8. Перечислите основные этапы процесса тестирования.
9. Что включает в себя жизненный цикл программы?
10. Когда тестирование приносит наибольшую экономическую выгоду и когда его целесообразно прекращать?
11. Каковы основные сложности тестирования?
12. Какими психологическими качествами и техническими навыками должен обладать хороший тестировщик?

Помните! :)