Тестирование и поддержка программных модулей

# Введение. Основные понятия.

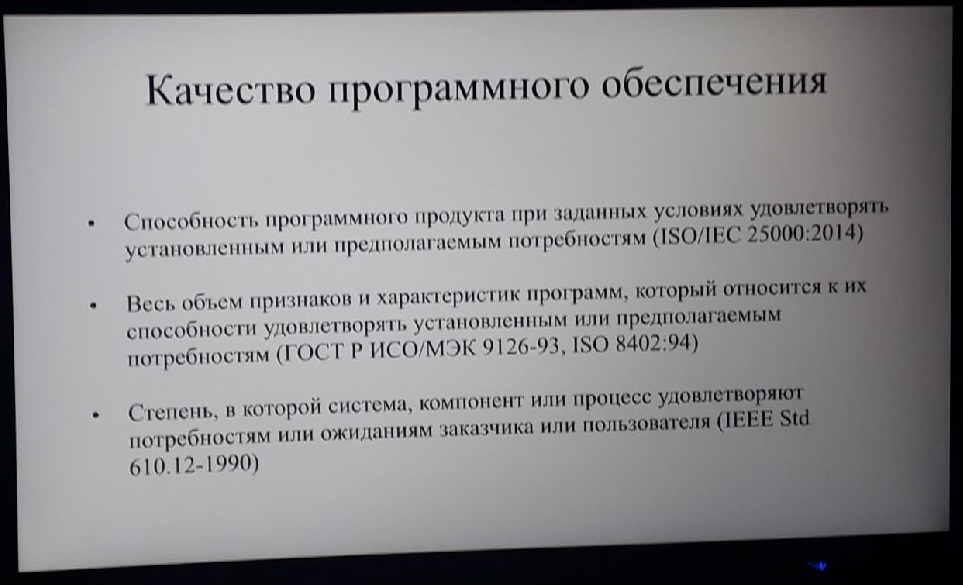
(15.01.2020)

Тестирование – не разработка программного продукта, анализ, управление, техписательство

Невозможно отыскать абсолютно все ошибки в программном продукте. Ошибки будут всегда.

Тестирование – процесс, который позволяет убедиться в том, что программа выполняет свое назначение. (Тестирование – это наблюдение за функционированием ПО в специфических условиях с целью определения степени соответствия ПО в специфических условиях)

Главная цель тестировщика заключается в предоставлении отрицательной обратной связи. Он не отвечает за качество.



Качество ПО

1. Функциональность: функциональная исправность, соответствие стандартам, функциональная совместимость, безопасность, точность;
2. Надежность: завершенность, восстанавливаемость, устойчивость к отказам;
3. Удобство использования: удобство изучения, понятность, удобство и простота использования;
4. Эффективность: эффективность по времени, эффективность использования ресурсов;
5. Удобство сопровождения: стабильность, анализируемость, контролепригодность, изменяемость;
6. Портативность: удобство установки, совместимость, заменимость;

Тест

1. это специальная, искусственно созданная ситуация, выбранная определенным образом
2. описание того, какие наблюдения за работой программы можно сделать
3. для проверки ее соответствия некоторому требованию

Тестовый случай (test case)

* артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или ее части
* Тестовый случай используется для автоматического или ручного тестирования

Ошибка-Дефект-Отказ

* Ошибка (Mistake) – Возникает в результате деятельности людей, связанной с разработкой программного обеспечения и ошибки в требованиях
* Дефект (Defect) – Несоответствие поведения программы требованиям или несоответствие ожидаемому поведению или документации.
* Отказ (Error) – Проявление дефекта в ходе эксплуатации программы

Верификация vs Валидация

* Верификация (Verification) – процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа.
* Валидация (Validation)– определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе

Цели и задачи

Предоставить информацию:

* Руководителю проекта
* Разработчику
* Заказчику

Что:

* Обнаруженные несоответствия и рекомендации к их исправлению
* Рекомендации к усовершенствованию

Как:

* Средства управления тестированием
* Отчеты
* Встречи

Задачи тестировщика

1. Определение миссии тестирования
2. Проверка похода к тестированию
3. Проверка стабильности выпуска
4. Тестирование и оценка
5. Достижение приемлемого результата миссии
6. Совершенствование методов и средств тестирования

Планирование тестов

1. Определение требований к тестам
2. Оценка рисков
3. Выбор стратегии тестирования
4. Определение ресурсов
5. Создание расписания/последовательностей
6. Разработка плана тестирования
7. Определение критериев завершения и успешности тестирования

Оценка тестов состоит из анализа дефектов и оценки критериев завершения и успешности тестирования

Формула Фагана

Участники процесса тестирования

* Менеджер проекта – обеспечивает ресурсами, координация работ
* Разработчик, технический писатель – исправление найденных ошибок
* Архитектор – обеспечение целостности проекта в процессе исправления ошибок
* Интегратор (релиз-менеджер) – контроль и выпуск версий ПО
* Аналитик – установка приоритетов, связанных с необходимостью и сложностью исправления найденных дефектов
* Тестировщик – несет ответственность за тестирования в целом

Четыре обобщенные фазы жизненного цикла

1. Концепция (инициализация, идентификация, отбор)
2. Определение (анализ)
3. Выполнение (практическая реализация или внедрение, производство и развертывание, проектирование или конструирование, сдача в эксплуатацию, инсталляция, тестирование и т.п.)
4. Закрытие (завершение, включая оценивание после завершения)

# Методы и виды тестирования. Анализ и виды ПО.

По знанию системы, тестирование делится на два вида:

1. Метод белого ящика
2. Метод черного ящика

По уровню тестирования

Требования к ПО

Системный тест

Интеграционный тест

Архитектура

Модульный тест

Спецификация модуля

Код

Системное тестирование (System):

1. отвечает за проверку соответствия системы установленным требованиям
2. Задача: выявление дефектов, связанных с работой системы в целом
   1. Отсутствующая или неверная функциональность
   2. Неверное использование ресурсов системы
   3. Непредусмотренные комбинации данных пользовательского уровня
   4. Несовместимость с окружением
   5. Непредусмотренные сценарии использования
   6. Неудобство в применении
3. Метод черного ящика

Интеграционное тестирование (Integration):

1. Проверка связи между компонентами, а также между различными частями системы
2. Задача: поиск дефектов, связанных с ошибками в реализации и интерпретации взаимодействия между модулями.
3. Требует создание тестового окружения

Модульное тестирование (Unit):

1. Проверка отдельно взятых модулей функций или классов
2. Задача: выявление локализованных в модуле ошибок в алгоритмов или реализации алгоритмов
3. Требует создания тестового окружения (заглушки)
4. Метод «Белого ящика»

Методы сборки модулей:

1. Монолитный
2. Инкрементальный
   1. Сверху вниз
   2. Снизу-вверх

Сравнение методов

1. Монолитное тестирование требует больших трудозатрат, связанных с сложностью идентификации ошибок, появляющихся в пространстве собранного кода.
2. Монолитное тестирование предоставляет большие возможности распараллеливания работ, особенно на первых этапах тестирования.
3. Инкрементальное тестирование связано с меньшей трудоёмкостью идентификации ошибок за счет постепенного наращивания объема тестируемого кода.
4. Инкрементальное тестирование требует затрат на создание заглушек.

Недостатки нисходящего тестирования:

1. Проблема разработки достаточно “интеллектуальных” заглушек
2. Сложность организации и разработки среды для реализации исполнения модулей в нужной последовательности
3. Параллельная разработка модулей верхних и нижних уровней приводит к не всегда эффективной реализации модулей из-за подстройки еще не тестированных модулей нижних уровней уже оттестированным модулям верхних уровней

Недостатки восходящего тестирования - Запаздывание проверки концептуальных особенностей тестируемого комплекса.

Комбинирование уровней тестирования

1. В каждом конкретном проекте должны быть определены задачи, ресурсы и технологии для каждого уровня тестирования
2. Задача тестировщиков и менеджеров – оптимально распределить ресурсы между тремя уровнями тестирования, так, чтобы каждый из возможных типов дефектов был «адресован»
3. Например, пересечение усилий на поиск фиксированного типа дефектов из области системного в область модульного тестирования может существенно снизить сложность и стоимость всего процесса тестирования

По степени автоматизации:

1. Ручное
2. Автоматизированное
3. Автоматическое

По целям:

1. Функциональное – рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.
2. Не функциональное
   1. Производительности
   2. Инсталляционное
   3. Конфигурационное
   4. Безопасности
   5. Локализации
   6. Удобства использования

Тестирование производительности – Автоматическое тестирование, имитирующее работу определенного количества бизнес пользователей на каком-либо ресурсе.

1. Производительность
2. Стабильность
3. Объемное
4. Нагрузочное
5. Стрессовое
6. Масштабируемость
7. Инсталляционное

Инсталляционное тестирование направлено на проверку успешной установки и настройки обновления или удаления программного обеспечения при различном программном и аппаратном окружении.

Проверить:

1. Установку
2. Корректность списка файлов
3. Регистрация приложения в ОС
4. Обратная совместимость данных
5. Прерывание установки, установка при запущенном приложении
6. Удаление

Тестирование конфигурации – направлено на проверку работы ПО при различных конфигурациях системы.

1. На заявленных платформах
2. Поддерживаемых драйверах
3. При различных конфигурациях компьютеров

Тестирование безопасности – используется для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостности подхода к защите приложения.

1. XSS
2. XSRF
3. Injections
4. SSI

Тестирование локализации – проверка правильности перевода элементов пользователя.

Тестирование удобства использования (Usability testing) – проверяет степень удобства использования, обучаемости, понятности и привлекательности интерфейса.

Хронология тестирования:

1. Дымовое – подтверждение работы после свежей сборки
2. Основное – проверка другими видами тестирования (функциональное, тестирование интерфейса)
3. Регрессионное – проверка старых возможностей, которые не должны быть удалены при обновлении
4. Приёмочное – правильно-ли работает ПО для заказчика или нет

Критерии черного ящика:

1. Тестирование функций
2. Тестирование классов входных данных
3. Тестирование классов выходных данных
4. Тестирование области допустимых значений
5. Тестирование длины набора данных
6. Тестирование упорядоченности набора данных

Критерии белого ящика

1. Покрытие операторов
2. Покрытие ветвей
3. Комбинаторное покрытие условий
4. Покрытие путей
5. Покрытие решений/условий

Что такое требование?

1. Условие или возможность, требуемая пользователем для решения задач или достижения целей
2. Условие или возможность, которые должны удовлетворяться системой/компонентом системы должна обладать для обеспечения условий контракта, стандартов, спецификаций или др. регулирующими документами.
3. Документальная репрезентаций условий или возможностей, перечисленных в предыдущих пунктах

Источники требований

1. Стандарты
2. Законодательство
3. Конкурирующие программные продукты
4. Внутренние документы организации
5. Ожидание потребителей

# Тестовая документация. Тест-план, тест-дизайн

1. Упрощение работы Разработчика автоматизированных тестовых скриптов.
2. Ускорение работы Тестировщика, Тест-менеджера
3. Синхронизация действий тестировщиков различных уровне й
4. Повышение надежности результатов работы
5. База знаний. Используется при внесении изменений в систему для проверки того, что как старая, так и новая функциональность работает корректно

Тестовая стратегия – общий план контроля качества ПО, охватывающий длительный период времени, главная цель которого достижение и обеспечение высокого качества ПО. Включает в себя:

1. Типы тестов, которые нужно выполнить для данного функционала системы
2. Описание необходимых подходов с точки зрения целей тестирования
3. Описание или требования к необходимым для проведения тестирования инструментам и инфраструктуре

Стратегия тестирования шара желаний:

1. Что за продукт?
   1. Приложение, помогающее людям принимать важные решения
2. Ключевые риски?
   1. Шар предлагает неверные решения
   2. Люди используют его неверно
   3. Шар неверно определяет сценарий
   4. Шар не сможет обрабатывать сложные запросы
3. Как тестировать, чтобы минимизировать риски?
   1. Понимать алгоритм
   2. Использовать оракул
   3. Полное тестирование каждой функции
   4. Создать большое количество таблиц решений
   5. Включить сложные сценарии
   6. Тестировать документацию

План тестирования – документ, в котором определены объем, ресурсы, а также описан календарный план работ по тестированию.

Что писать в плане тестирования:

1. Общее описание тест-плана
2. Трассировка на другие документы
3. Определение тестируемых областей системы
4. Определение подходов к тестированию
5. Чего мы НЕ тестируем, ограничение тестирования
6. Критерий успешности/неуспешности прохождения тестов
7. Критерии начала/окончания тестирования
8. Тестовые документы
9. Требования к среде тестирования
10. Человеческие ресурсы и уровень их подготовки

Критерии начала/окончания тестирования

Критерии начала:

1. Готовность функционала или его части
2. Готовность тестовой среды
3. Предоставлена актуальная документация

Успешный план тестирования:

1. Понимание целей
2. Согласование с заинтересованными лицами
3. Поддержка в актуальном состоянии
4. Опыт

# Отчеты

* Состав команды
* Сроки выполнения
* Описание процессов тестирования
* Изменение тестовой модели, дополнение ТК
* Процент пройденных ТК
* Критичные и блокирующие проблемы
* Результаты регресса
* План на следующую итерацию/план/месяц

Приемы представления информации

Бизнес-пользователи, Продукт-менеджер

* Графики
* Сводные таблицы

Тест-менеджер

* Термины, формулы, профессиональный сленг

# Третий курс

## Unit-тестирование

Unit-тест – блок кода, который вызывает тестируемый блок кода и проверяет его правильную работу. Если результат теста не совпадает с ожидаемым результатом, то тест считается не пройдённым.

Типы тестирования:

1. Модульное тестирование
2. Интеграционное тестирование
3. Системное тестирование

При интеграционном тестировании существует много критических точек, в которых приложение может дать сбой.

Верификация – процесс оценки системы или её компонентов с целью определения того, удовлетворяют ли результаты текущего этапы разработки условиям, сформированным в начале этого этапа.

Валидация – определение соответствия, разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе.

Практика использования Unit-тестирования

Свойства хорошего юниттеста:

1. Автоматизация
2. Повторяемость
3. Простота в реализации
4. Должен остаться для дальнейших тестирований
5. Должен запускаться одним нажатием
6. Кто угодно может запустить
7. Быстрое выполнение

* Тесты должны быть частью контроля версий
* Если приложения монолитные, то все тесты располагаются в одной папке

Именование методов и классов

Когда не нужно создавать тесты

1. Когда нет зависимостей

Test driven Development

Разработка через тестирование. Подход разработки ПО, который заключается в написании юнит-теста перед написанием самого кода.

Способ написания юнит-теста

1. Пишем код
2. Пишем тест
3. Проверка