Оглавление

[1 тема. “Поддержка и тестирование программных модулей”. (12.01.23) 2](#_Toc135292560)

[2 тема. “Методы и виды тестирования. Анализ требований к ПО”. (26.01.23) 9](#_Toc135292561)

[3 тема. “Тестовая документация. Тест-план. Тест-дизайн”. (19.02.23) 18](#_Toc135292562)

[4 тема. “Тестовая документация. Test Case. Отчёт о прохождении тестов”. (17.03.23) 24](#_Toc135292563)

[4 тема. “Дефект”. (14.04.23) 26](#_Toc135292564)

[5 тема. “Техники тестирования”. (12.05.23) 28](#_Toc135292565)

# 1 тема. “Поддержка и тестирование программных модулей”. (12.01.23)

Жизненный цикл ПО. Жизненный цикл начинается с идеи о задумке, конец — жизни- это когда мы выводим из эксплуатации.

Стадии:

1.Анализ требований 2.Проектирование

3. Разработка/кодирование 4 Тестирование

5 Ввод в эксплантацию. (Доработка и освобождение)

Тестирование не является каким-либо действием, оно заключает в себе комплекс действий.

Тестирование – это процесс, позволяющий убедится в том, что в программе нет ошибок, при этом невозможно отыскать абсолютно все ошибки в программном продукте, из этого выходит, что тестирование – это процесс, позволяющий убедиться, что программа выполняет свой процесс.

Тестирование — это наблюдения финиширование ПО с целью определению с требованиями ПО.

Тестирование – это одна из дисциплин НРП. Она ориентирована на оценку качества с помощью следующих методов:

1.Поиск и документирование дефектов качества.

2.Общие рекомендации относительно качества.

3.Проверка выполнения основных предположений и требований на конкретных примерах.

4.Проверка что продукт функционирует так как было запроектировано.

5.Проверка, что требования были выполнены соответственным образом.

*Тестирование - это проверка соответствия программы требованиям осуществляемая путём наблюдения за её работой специально искусственно созданных ситуаций, выбранных с целью гарантии требованного качества продукта.*

***Качество программного обеспечения.***

Способность программного продукта при заданных условиях удовлетворять установленным или предполагаемым потребностям (ISO/IEC 25000:2014).

Весь объём признаков и характеристик программ, который относится к их способности удовлетворять установленным или предполагаемым потребностям (ГОСТ Р ИСОМ/МЭК 9126-93, ISO 8402:94).

Степень, в которой система, компонент или процесс удовлетворяют потребностям или ожиданиям заказчика, или пользователя (IEEE Std 610.12-1990).



***Гост – каталоги государственных стандартов.***

ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению.

ГОСТ 28195-89.Оценка качества программных средств. Общие положения.

ГОСТ 28806-90. Качество программных средств. Термины и определения.

***Тест.***

Это специальная, искусственно созданная ситуация, выбранная определённым образом и описание того, какие наблюдения за работой программы нужно сделать для проверки её соответствия некоторому требованию.

***Тестовый случай***

-Это артефакт, описывающий совокупность шагов,конкретных условий и параметров, необходимость для проверки реализации тестируемой функции или её части.

Тестовый случай используется для ручного или автоматического тестирования.

***Ошибка-Дефект-Отказ***

Ошибка

Возникает в результате деятельности людей, связанной с отработкой программного обеспечения (Ошибка в требованиях, ошибка в дизайне, ошибка в коде).

Дефект

Несоответствие поведения программы требованиям или ожидаемому поведению, или несоответствие документации требованиям.

Отказ

Проявление дефекта в ходе эксплуатации программы. Аварийный отказ – невозможность продолжать эксплуатацию программы.

Верификация vs Валидация

Верификация – это процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа. Т.е. выполняются ли наши цели сроки, задачи по разработке проекта, определенные в начале текущей фазы.

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

13.01.23

***Цели и задачи тестирования***

Предоставить информацию:

* Руководителю проекта
* Разработчику
* Заказчику

Что:

* Обнаруженные несоответствия и рекомендации к их направлению.
* Рекомендации к усовершенствованию

Как:

* Средства управления тестированием
* Отчёты (E-mail)
* Встречи

***Задачи тестировщика.***

* Определение миссии тестирования
* Проверка подхода к тестированию
* Проверка стабильности выпуска
* Тестирование и оценка
* Достижение приемлемого результата миссии
* Совершенствование методов и средств тестирования

***Планирование тестов***

* Определение требований к тестам
* Оценка рисков
* Выбор стратегии тестирования
* Определение ресурсов
* Создание расписания/последовательностей
* Разработка Плана тестирования
* Определение критериев завершения и успешности тестирования.

***Дизайн и разработка тестов***

* Анализ объёма работ
* Определение тестовых случаев -> Дизайн
* Обзор и оценка тестового покрытия
* Определение тесто-критичной функциональности в Дизайне и Модели реализации
* Запись тест-кейсов или программирование тестовых скриптов
* Создание/подготовка внешних наборов данных

***Процесс выполнения тестов***

* Выполнение тестов
* Оценка выполнения тестов
* Восстановление после сбойных тестов
* Проверка результатов
* Исследование неожиданных результатов
* Запись ошибок

***Оценка тестов***

* Анализ дефектов
* Оценка критериев завершения и успешности тестирования

***Эффективность тестирования***

Фаган (1976)

(Число ошибок, обнаруженных в ходе инспекции/

Общие число ошибок в продукте до инспекции) \*100%

Джонс(1986)

(Дефекты, обнаруженные тестированием или инспекцией/Дефекты, имеющиеся во время тестирования или инспекции) \* 100% = (Обнаруженные дефекты/Обнаруженные + необнаруженные дефекты (обнаруженные позже)) \* 100%

***Участники процесса тестирования***

Менеджер проекта – обеспечение ресурсами, координация работ.

Разработчик, тех., 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 – несёт ответственность за процесс тестирования в целом.

***Роли участников группы тестирования***

Руководитель группы тестирования (Test manager) – представляет ключевую роль тестировщика в рабочей группе, несет ответственность за организацию процесса тестирования в проекте, планирование и контроль действий по тестированию.

Тестировщик-аналитик (Test analyst) – несет ответственность за формирование тестовых спецификаций, и анализ итогов тестирования.

Разработчик тестов (Test developer) – несет ответственность за разработку автоматизированных тестов, предусмотренных в плане тестирования, установку и сопровождение инфраструктуры тестирования, создание стенда для проведения тестирования в соответствии с планом тестирования.

Исполнитель тестов (Test operator/ Engineer) – несет ответственность за фактическое исполнение тестов и документирование выявленных дефектов.

***Определения модели жизненного цикла программной системы по ГОСТ***

Модель жизненного цикла – структура, состоящая из

процессов, работ и задач, включающих в себя разработку,

эксплуатацию и сопровождение программного продукта,

охватывающая жизнь системы от установления

требований к ней до прекращения ее использования [ГОСТ

12207, 1999]

Жизненный цикл автоматизированной системы (АС) –

совокупность взаимосвязанных процессов создания и

последовательного изменения состояния АС, от

формирования исходных требований к ней до окончания

эксплуатации и утилизации комплекса средств

автоматизации АС [ГОСТ 34, 1990].

***Циклы тестирования***

Полный цикл тестирования обычно совпадает с итерацией разработки или соответствует ее определенной части.

Частный цикл тестирования, как правило, проводится для

конкретной сборки объекта тестирования (системы, подсистемы или отдельного компонента).

***Полный цикл тестировани***

Начало -> Планирование тестирования -> Разработка тестов -> Выполнение тестов -> Анализ и отчёт о результатах тестирования -> Завершения.

# 2 тема. “Методы и виды тестирования. Анализ требований к ПО”. (26.01.23)

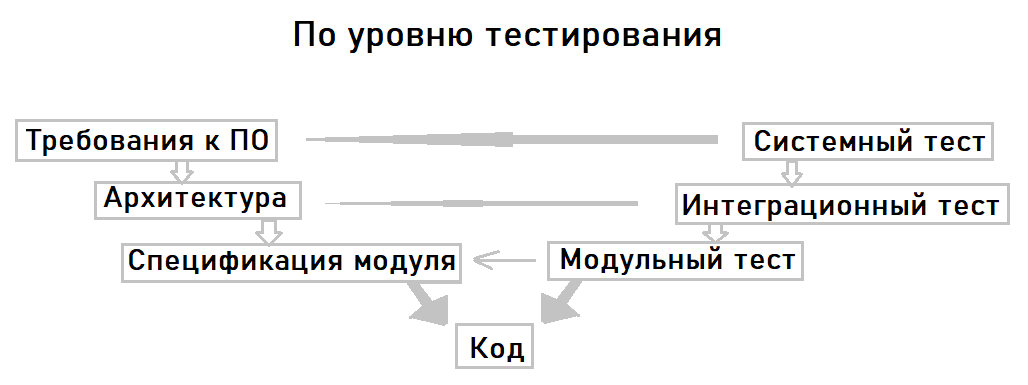
Виды тестирования.

По знанию системы:

Чёрный ящик – тестируем, когда мы не имеем доступа к коду.

Белый ящик –тестируя, когда мы имеем доступ к коду.

По уровню тестирования.



Системное тестирование (System)

• Проверка соответствия системы установленным требованиям. • Задача: выявление дефектов, связанных с работой системы в целом:

* отсутствующая или неверная функциональность
* неверное использование ресурсов системы
* непредусмотренные комбинации данных пользовательского уровня
* несовместимость с окружением
* непредусмотренные сценарии использования
* неудобство в применении и тому подобное.
* Метод «черного ящика»

Интеграционное тестирование (Integration)

Проверка связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами).

Задача: поиск дефектов, связанных с ошибками в реализации и интерпретации взаимодействия между модулями.

* Требует создания тестового окружения (н/р внешние системы).
* Метод «белого ящика»

Модульное тестирование (Unit)

• Проверка отдельно взятых модулей, функций или классов.

• Задача: выявлении локализованных в модуле ошибок в алгоритмах или реализации алгоритмов.

• Требует создания тестового окружения (заглушки).

• Метод «белого ящика».

Методы сборки модулей:

* Монолитный;
* Инкрементальный: 1 - Сверху вниз; 2- Снизу-вверх;

Сравнение методов

- Монолитное тестирование требует больших трудозатрат, связанных со сложностью идентификации сшибок, проявляющихся в пространстве собранного кода.

+ Монолитное тестирование предоставляет большие возможности распараллеливания работ, особенно на начальной фазе тестирования

+ Инкрементальное тестирование связано с меньшей трудоемкостью идентификации ошибок за счет постепенного наращивания объема тестируемого кода и соответственно локализации добавленной области тестируемого кода

- Инкрементальное тестирование требует затрат на разработку заглушек и драйверов

Недостатки нисходящего тестирования

• Проблема разработки достаточно интеллектуальных заглушек, т.е. заглушек, способных к использованию при моделирование различных режимов работы комплекса, необходимых для тестирования

• Сложность организации и разработки среды для реализации исполнения модулей в нужной последовательности

• Параллельная разработка модулей верхних и нижних уровней приводит к не всегда эффективной реализации модулей из-за подстройки (специализации) еще не тестированных модулей нижних уровней к уже оттестированным модулям верхних уровней

Недостатки восходящего тестирования

Запаздывание проверки концептуальных особенностей тестируемого комплекса.

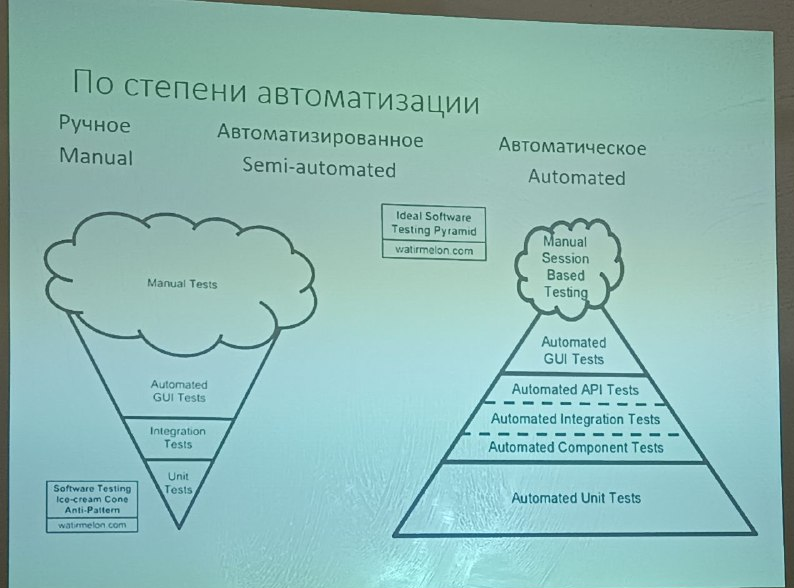
Комбинирование уровней тестирования

• В каждом конкретном проекте должны быть определены задачи, ресурсы и технологии для каждого уровня тестирования.

• Задача тестировщиков и менеджеров - оптимально распределить ресурсы между тремя уровнями тестирования так, чтобы каждый из возможных типов дефектов был «адресован» (в наборе тестов должны иметься тесты, направленные на выявление дефектов этого типа).

• Например, перенесение 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



*По целям*: делится на Не функциональное и Функциональное:

*Не функциональное:*

* Производительности
* Инсталляционное
* Конфигурационное
* Безопасности
* Локализации
* Удобство использования

03.02.23

Тестирование производительности:

Автоматизированное тестирование, имитирующее работу определенного количества бизнес-пользователей на каком-либо общем ресурсе.

Производительности и стабильности – с какой скоростью работает приложение с каким-то количеством пользователей

Объемное – сможет ли система работать так же, как сейчас, если будет расширяться?

Стрессовое – какое кол-во пользователей необходимо до отказа?

Нагрузочное – сможет ли работать система при неожиданном потоке пользователей.

Инсталляционное тестирование:

• Направленно на проверку успешной установки и настройки, обновления или удаления программного обеспечения при различном программном и аппаратном окружении, а также призванное оценить работоспособность системы после завершения работы инсталляции.

• 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 конфигураций:

• Направленно на проверку работы программного обеспечения при различных конфигурациях системы:

1. Заявленных платформах

2. Поддерживаемых драйверах

3. При различных конфигурациях ПК

4. При различных конфигурациях системного программного обеспечения

• Обычно автоматизированное

Тестирование безопасности:

• Для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного прохода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Тестирование локализации:

• Проверка правильности перевода элементов интерфейса пользователя, правильности перевода системных сообщений и ошибок, перевода раздела «Помощь»/«Справка» и сопроводительной документации.

1. Встроен ли текст в программный код

2. Перевод длиннее исходного текста

3. Символы, не входящие в ANSI

4. Мертвые клавиши

5. Фильтрация ввода

6. Сборные сообщения

7. Идентификатор сообщения об ошибке

8. Формат данных, единицы измерения

9. Сортировка, правила перевода регистра

Тестирование удобства использования:

• Направленно на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей продукта в контексте заданных условий

• Способы проведения

1. Экспертная оценка

2. Анкеты, пользовательские тестирования

3. Математические оценки

4. Чит-листы, гайдлайны и стандарты

Тестирование по хронологии выполнения:

• Дымовое – короткий цикл тестов, выполняемый для подтверждения того, что после сборки коды устанавливаемое приложение, стартует и выполняет основные функции (автоматическое)

• Основное – круг выполнения всех запланированных проверок для новой и измененной разработки. Может совмещать в себе функциональное тестирование, тестирование пользовательского интерфейса и удобства использования, тестирование производительности или безопасности

• Регрессионное – направленно на то, чтобы подтвердить, что существующая функциональность работает как прежде (автоматическое)

• Приемочное – формальный процесс тестирования, который проверяет соответствие системы потребностям, требованиям и бизнес-процессами пользователя. Проводиться для вынесения решения заказчиком или другими уполномоченным лицом, принимается приложение или нет

Критерии покрытия тестирования:

• Метрика для оценки качества

• Измеряет долю классов ситуаций, представители которых попали в тестовый набор

• Основан на:

1. Критерий БЯ (коде) – не выявит нереализованные требования

a) Покрытия операторов

b) Покрытия ветвей (решений/условий)

c) Покрытия путей

d) Комбинаторного покрытия условий

2. Критерии ЧЯ (спецификации) – может оставить непроверенными некоторые участки кода

a) Тестирование функций

b) Тестирование классов входных/выходных данных

c) Тестирование области допустимых значений

d) Тестирование длины набора данных

e) Тестирование упорядоченности набора данных

Что такое требования?

• Условие или возможность, требуемая пользователем для решения задач или достижения целей

• Условие или возможность, которые должны удовлетворяться системой/компонентами системы или которыми система/компонент системы должна обладать для обеспечения условий документами

• Документальная репрезентация условий или возможностей, перечисленных в предыдущих пунктах

Источники требований:

• Стандарты (HTML, CSS, XML, URL и др)

• Федеральное и муниципальное отраслевое законодательство (конституция, распоряжение)

• Нормативное обеспечение организации (регламент, уставы, приказы)

• Представления и ожидания потребителей и пользователей системы

• Конкурирующие программные продукты

Методы выявления требований:

• Общение с заказчиком

• Общение с авторами продукта

• Анализ стандартов, нормативной документации

• Анализ выявленных проблем в баг-трекере

Показатели качества требований:

• Корректность или адекватность

• Целостность

• Недвусмысленность

• Полнота

• Непротиворечивость

• Упорядоченность по приоритету и стабильность

• Проверяемость

• Модифицируемость

• Прослеживаемость в ходе разработки

Тестирование требований:

• Когда? – как только появилось хотя бы одно требование

• Зачем?

1. Уменьшение количества доработок и изменений

2. Сокращение рисков

3. Ознакомление и согласованность задач между разработчиками

• Доколе? – достаточно информации для начала разработки/тестирования

• Кто?

1. Тестировщик

2. Аналитик

3. Менеджер

4. Разработчик

Тест №1. Проверяем требования на полноту.

• Содержит ли требования выражения типа «подлежит определению», «и так далее», «и прочее», «но»…

• Ссылается ли требования на несуществующие источники

• Ссылается ли на еще не определенные источники

Тест №2. Выявить неполные, неизмеримые требования.

• Определить меру качества для каждого требования:

1. Верно ли, что каждое требование имеет критерий качества, который можно использовать для проверки того, удовлетворяет ли какое-либо решение требованию

2. Обращаем внимание на неоднозначные требования («быстро», «нормально», «оптимально»)

Тест №3. Проверяем отслеживаемость требований.

• Рассматриваем каждое требование как отдельно распознаваемую, измеряемую сущность

• Каждое ли требование однозначно распознаваемое

Тест №4. Однозначно ли требование?

• Всякая ли ссылка на термин, определенный в спецификации требований, согласуются с этим определением

• Используется ли один и тот же термин для одних и тех же понятий

Проблемы требований:

• Неполные

• Частные изменения

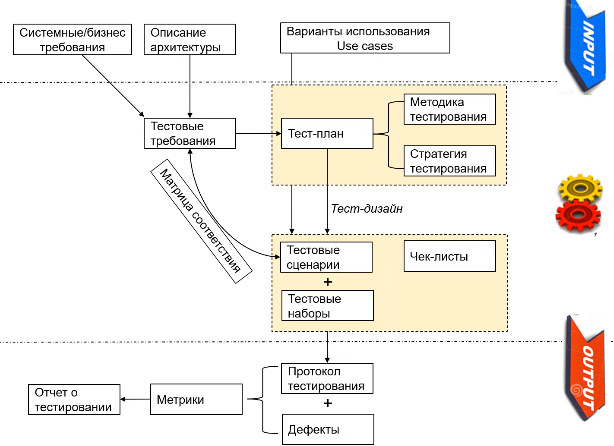
• Изменения в последний момент

• Неверно трактовали

# 3 тема. “Тестовая документация. Тест-план. Тест-дизайн”. (19.02.23)

Для чего нужна тестовая документация?

* Упрощение работы Разработчика автоматизированных тестовых скриптов, Тестировщика, Тест-менеджера.
* Ускорение работы Тестировщика, Тест-менеджера.
* Синхронизация действий тестировщиков различных уровней.
* Повышение надежности результатов работы (тестирование выполняется в соответствии с выбранными критериями оценки качества, все аспекты поведения системы протестированы).
* База знаний: Используется при внесении изменений в систему для проверки того, что как старая, так и новая функциональность работает корректно.



Тестовая Стратегия

это общий, не(очень)детализированный план контроля качества ПО, охватывающий длительный период времени, главная цель которого достижение и обеспечение высокого качества ПО.

Мы будем использовать тестирование черным ящиком, причинно- следственные графы и тестирование граничных значений при тестировании на основании спецификации.

1. Конкретная
2. Практичная
3. Основательная

Стратегия включает

* + Типы тестов, которые нужно выполнять для данного функционала системы
  + Описание необходимых подходов с точки зрения целей тестирования
  + Описания или требования к необходимым для проведения тестирования инструментам и инфраструктуре.

Стратегия отвечает на вопросы:

1. Как, каким образом тестирование даст ответ, что данный функционал работает?
2. Что нужно сделать и чем пользоваться из инструментальных средств, для достижения целей тестирования?
3. Когда определённый функционал будет тестироваться и соответственно когда ожидать получения результатов?
4. Как минимизировать затраты на тестирование?



Стратегия тестирования шара желаний

1. Что за продукт?

• Приложение, которое помогает людям/группам людей принимать важные решения

2. Ключевые риски?

|  |  |
| --- | --- |
| • Шар предлагает неверные решения  • Люди используют его неверно | • Шар неверно определяет сценарий  • Шар не сможет обрабатывать сложные вопросы |

3. Как тестировать, что бы минимизировать риски

• Понимать алгоритм

• Использовать оракул

• Полное тестирование каждой функции с приоритетом больше Major

• Создать большое количество таблиц решений

• Включить сложные сценарии

• Тестировать документацию

• Тестировать «ошибки на дурака»

17.02.23

Методика тестирования

* Способ, который создаются тесты
* Более общая, чем тестовая стратегия
* Часто не привязана к проектору
* Может быть/не быть применимой на практике вне конкретного случая

План тестирования

Это документ, котором определены объём, ресурсы, а также описан календарный план работ по тестированию. В нём определяются выполняемые тесты, тестируемые элементы, задачи тестирования, сотрудники, ответственные за выполнение каждой из задач, а также указывается вероятность возникновения непредвиденных обстоятельств и описывается, какие меры нужно при этом принимать.

Роли, ответственные за тест план: тест-менеджер/руководитель группы тестирования.

Цели плана тестирования:

* Понять что, как, кем будет/не будет проверяться (составить календарный план, определится с инструментарием)
* Донести эту информацию до заказчика/команды

А надо ли?

<Да>;

* Процесс разработки ПО сертифицирован и вам нужно показывать его проверяющим
* Ресурсоёмкий проект
* И т.п.

<Нет>;

* Тест-план у вас записан в рамках какого-либо другого документа
* Вас на проекте 1,2,3 членов.
* У вас частный случай бардака. (Когда не имеешь планов определённых и делаешь всё как получится).

Что писать в плане тестирования:

* Общее описание тест-плана
* Трассировка на другие документы
* Определение тестируемых областей системы
* Определение подходов к тестированию
* Чего НЕ тестируем, ограничения тестирования
* Критерий успешности/не успешности прохождения тестов
* Критерии начала/окончания тестирования
* Тестовые документы
* Требования к среде тестирования
* Человеческие ресурсы и уровень их подготовки
* План-график тестирования (последовательности) + Приоритеты
* Риски

Критерии начала/окончания тестирования Критерии начала:

* Готовность функционала или его части
* Готовность тестовой среды
* Предоставлена актуальная документация
* Предоставлены ресурсы

Критерии окончания:

* Выполнены все запланированные тесты
* Все тесты успешны
* Зарегистрированы все обнаруженные дефекты, нет открытых дефектов приоритета Критичный или Высокий
* Производительность не упала
* Безопасность в пределах допустимого

Успешный план тестирования

* Понимание целей: не «для галочки»
* Согласование с заинтересованными лицами
* Поддержка в актуальном состоянии
* Опыт

Тест-дизайн

Тест-дизайн — это этап процесса тестирования ПО, на котором

проектируются и создаются тестовые случаи (тест-кейсы) соответствия с определенными ранее критериями качества и целями тестирования

Роли, ответственные за тест- дизайн

Тест-аналитик - "ЧТО тестировать

Тест-дизайнер - "КАК тестировать

Цели тест-дизайна

* Придумать тесты, которые обнаружат наиболее серьёзные ошибки продукта.
* Минимизировать количество тестов, необходимых для нахождения большинства серьёзных ошибок.

Задачи тест-дизайна

* Анализ имеющихся проектных артефактов: документация (спецификации, требования, планы), модели, исполняемый код и т.д
* Составление списка функций продукта
* Построение таблиц принятия решений
* Написание спецификации по тест-дизайну
* Написание тест-кейсов
* Создание приемочных проверок
* Анализ жалоб пользователей
* Анализ рисков
* Описание процесса тестирования
* Расстановка приоритетов тестирования

Форма тест-дизайна

* Текстовый документ
* Система тест-менеджера
* Таблица
* Майнд-карта
* Конечный автомат

Основные источники ненадёжного тест-дизайна:

* Низкое качество требований
* Отсутствие контроля качества требований
* Неготовность проекта к изменениям
* Незнание предметной области
* Отсутствие контроля процесса тест-дизайна





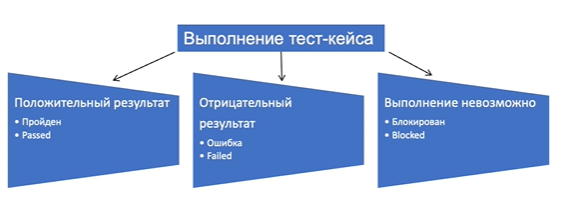
# 4 тема. “Тестовая документация. Test Case. Отчёт о прохождении тестов”. (17.03.23)

Тест-кейс (Test Case), Тестовый случай.

- это артефакт тест-дизайна, описывающий совокупность:

* конкретных условий, параметров (среды, сущности)
* шагов,
* используемых данных необходимых для проверки реализации тестируемой функции или её части.

Бывает позитивный или негативный.



Атрибуты тест-кейса

• Уникальный идентификатор тест-кейса

• Название

• Предусловия

• Шаги

• Ожидаемый результат

* Статус(необязательно)

• Ссылка на требование

• История изменений

* Ссылка на дефекты
* Ссылка на протокол выполнения

• Постусловия

Не должно быть в тест-кейсе

• Зависимостей от других тест-кейсов

• Нечеткой формулировки шагов или ожидаемого результата

• Создайте клиента.

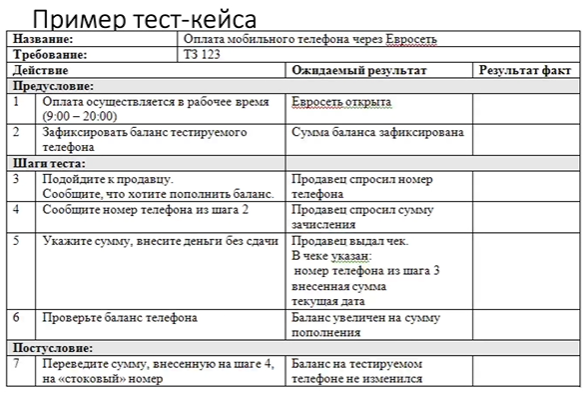
• Заявка создана.

• Отсутствия необходимой для прохождения тест-кейса информации

• Заполните поле корректными данными.

• Залогиньтесь.

• Излишней детализации.



Плюсы и минусы тест-кейсов

+ Тест-кейсы можно доверить выполнять новичку.

+ Являются базой знаний.

- Очень много копипасты.

-Сложно поддерживать.

# 4 тема. “Дефект”. (14.04.23)

Дефект – это несоответствие между техническим заданием в программе и реальным поведением системы или любая другая проблема тестированного продукта.

Какие ошибки могут быть:

1. Ошибка вследствие недостаточной квалификации
2. Ошибка квалификация
3. Инсталляционная ошибка
4. Ошибка управления

В хорошем дефекте есть:

1. Принцип, что? где? когда?.
2. Однозначные указания на номер сборки, в котором данный дефект обнаружен.
3. Тестовое окружение
4. Компонент предположительно в котором произошла ошибка
5. Степень критичности дефекта
6. краткое описание объекта
7. шаги воспроизведения
8. описание ожидаемого результата
9. описание фактического результата.
10. Возможные последствия дефекта

Минимально-достаточная жизненная модель дефекта

1. Открыт
2. Исправлен
3. Закрыт

Приоритет, критичность или важность

Что такое критичность – это атрибут характеризующий влияние дефекта на работоспособность программы (приложения).

Приоритет – это атрибут, указывающий на очерёдность выполнения задачи или устранения дефекта.

Градация важности дефекта

1. Блокирующая
2. Критическая
3. Значительная
4. Незначительная
5. Тривиальная

Градация приоритета дефекта

1. Высокий
2. Средний
3. Низкий.

# 5 тема. “Техники тестирования”. (12.05.23)

Стандартные проверки на поля экранных.

Текстовое поле

|  |  |
| --- | --- |
| Можно:  А-я A-z Спецсимволы  а-я a-z Пустая строка | нельзя |

Числовое поле

|  |  |
| --- | --- |
| Можно:  -5 5 0 0,5 ,Max, min | нельзя  А-я A-z Спецсимволы  а-я a-z пробелы |

Выпадающий список

|  |  |
| --- | --- |
| Можно:  Состав полей, Пустая строка Сортировка | нельзя  Рукописный ввод |

Дата

|  |  |
| --- | --- |
| Можно:  29 февраля 2012  С клавиатуры при наличии календаря  формат | нельзя  29 фев 2019 |



Стандартные проверки GUI

• состав элементов управления (полей ввода, подписей, кнопок, ссылок и т.д.)

• взаимное расположения элементов управления

• условное отображения элементов управления

• порядка обхода элементов управления по Tab

• значения по умолчанию в полях ввода

• состав значений полей списочных типов

• доступность полей для редактирования

• условная доступность полей для редактирования

• обязательность заполнения полей

• условная обязательность заполнения полей

• ограничения на форматы значений полей

• ограничения на диапазоны значений полей (граничные значения)

• сложные ограничения на значения полей (вычисляемые и условные ограничения)

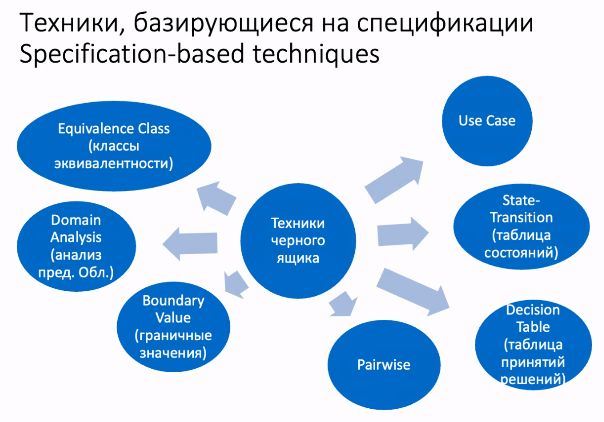
• расчет вычисляемых значений полей на типичных данных

• сохранение значений полей в системе

отображение значений полей из системы

• функциональности кнопок и других активных элементов управления

• нестандартные элементы управления и сложная презентационная логика (диаграммы, мультимедиа и динамические элементы управления и другие)



Множество возможных входных данных программного продукта, как правило, практически бесконечно.

Выбор конечного подмножества, достаточного для проведения исчерпывающего тестирования, является ключевой задачей тестировщика.

Классы эквивалентности

Калькулятор.

Для операции сложения (2 числа по 8 знаков):

10^8\*10^8= 10^16 комбинаций по 10 секунд -10^17 секунд, то есть больше 3 миллиардов лет непрерывных тестов.

Два теста считаются эквивалентными, если:

• Они тестируют одну и ту же вещь (функцию, модуль, часть системы).

• Если один из тестов ловит ошибку, то другой скорее всего тоже её поймает.

• Если один из них не ловит ошибку, то другой скорее всего тоже не поймает.

Класс эквивалентности

• Класс эквивалентности - это набор значений переменной, который считается эквивалентным. Тестовые сценарии эквиваленты, если

• они тестируют одно и тоже; • если один из них выявляет ошибку, то и остальные выявят ее;

• если одни из них не выявляет ошибку, то и остальные не выявят.

Если определен класс эквивалентности, протестируйте только одно-два значения из него

Эквивалентное разбиение - это разработка тестов методом черного ящика, в которой тестовые сценарии создаются для проверки элементов эквивалентной области. Как правило, тестовые сценарии разрабатываются для покрытия каждой области как минимум один раз.

Алгоритм использования анализа классов эквивалентности.

* Определить классы эквивалентности.
* Выбрать одного представителя от каждого класса
* Выполнить тесты.

Советы при определении классов эквивалентности

• Не забывайте о классах, охватывающих заведомо неверные или недопустимые входные данные.

• Организуйте формируемый перечень классов в виде таблицы или плана.

• Определите диапазоны числовых значений.

• Для полей или параметров, принимающих фиксированные перечни значений, выясните, какие из значений входят в перечень.

• Проанализируйте возможные результаты выбора из списков и меню.

• Поищите переменные, значения которых должны быть равными.

• Поищите классы значений, зависящих от времени.

• Выявите группы переменных, совместно участвующих в определенных вычислениях, результат которых ограничивается конкретным набором или диапазоном значений.

• Посмотрите, на какие действия программа отвечает эквивалентными событиями.

• Продумайте варианты операционного окружения.

Анализ граничных значений Boundary-value analysis

Тесты строятся с ориентацией на использование тех величин, которые определяют предельные характеристики тестируемой системы.

1. выделить классы эквивалентности.

2. определить граничные значения этих классов.

3. понять, к какому классу будет относиться каждая граница.

4. Для каждой границы нам нужно провести тесты по проверке значения до границы, на границе, и сразу после границы.

Количество тестов = количество границ \* 3

Предположение ошибок

Составить список, который перечисляет возможные ошибки и ситуации, в которых эти ошибки могли появится. 🡪 Составить тесты на основе списка.

