Тестирование ПО

Основы тестирования

Оглавление

Основные понятия и определения

- **Качество –** способность программы делать то, что от нее ждет пользователь.
- **Тестируемость –** степень, до которой могут быть запланированы объективность и реализуемость тестирования, проверяющего соответствие требованию
- **Тестовые данные–** входы, которые используются для проверки системы.
- **Тестовый сценарий (TEST CASE) –** входы для проверки системы и предполагаемые выходы в зависимости от входов, если система работает в соответствии с ее спецификацией требований.
- **Тест-план –** документ, описывающий весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и завершения тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами из минимизации.

Основные понятия и определения

- **Ошибка** – действие программиста на этапе разработки, которое приводит к тому, что в программном обеспечении содержится внутренний дефект, который в процессе работы программы может привести к неправильному результату.

- **Отказ (failure)** – непредсказуемое поведение системы, приводящее к неожидаемому результату, которое могло быть вызвано дефектами, содержащимся в ней

- **Дефект (fault)**– возможная причина ошибки.

- **Ошибочное действие (mistake)**– действие пользователя, приводящее к неверному результату.

Основные понятия и определения

- **Среда тестирования(test environment)** - инфраструктура для подготовки и проведения тестирования и интерпретации результатов
- **Покрытие (test coverage) –** часть функционала, тестируемая данным набором проверок
- **Позитивный тест** - проверка на правильных данных, реакция на корректные действия.
- **Негативный тест**- проверка на неправильных данных, реакция на некорректные действия
- **PASS/FAIL** - установленное правило, позволяющее определить удачно или неудачно прошла система данный тест

Тестирование. Основные цели и задачи

Что значит тестирование?

Какое тестирование?

Зачем он необходимо?

Тестирование – это …

Бытует мнение, что тестирование – это:

«… выполнение прогона тестов»

Но, активности по тестированию существуют

как до, так и после выполнения тестов.

Активности:

- Планирование и управление
- Выбор тестовых условий
- Разработка и выполнение тестовых сценариев
- Проверка результатов
- Оценка критериев выхода
- Подготовка отчета

Тестирование:

- Часть процесса разработки ПО
- Обеспечивает информацией о ПО
- Находит дефекты для исправления
- Очень креативная и интелектуальнозахватывающая деятельность (Г. Майерс, 1982)

Тестирование – это …

…не отладка!

Тестирование:

- Ответственность на команде тестирования
- Определяет отказы, вызванные дефектами
- Подтверждает, что изменения, предпринятые для устранения отказа, принесли необходимый результат

Отладка:

- Ответственность на команде разработки
- Анализируют отказ
- Устраняют причину отказа

Различие в тестировании

Работа специалистов по тестированию отличается от работы разработчиков.

Задачи разработчика:
- Созидание
- Нахождение верного решения

Задачи тестировщика:
- Разрушение
- Нахождение различных вариантов получения результата

Цели тестирования

Обнаружение дефектов:

- В процессе создания требований
- При кодировании
- На этапе тестирования

Повышение уровня качества ПО:

- Обеспечение работоспособности ПО
- Снижение количества дефектов путем повышения качества

Цели тестирования

Предоставление информации для принятия решения:

- Сформировать и передать заинтересованным сторонам риски выпуска системы
- Определить готовность продукта к внедрению

Предотвращение дефектов:

- Тестирование на ранних этапах ЖЦПО

Тестирование нужно начинать как можно раньше!!!

Задачи тестирования

Общее представление:

- Нахождение ошибок

НО, как насчет:

- Планирования и отслеживания тестов?
- Подготовки тестовых условий и тестовых данных?
- Разработки тестовых сценариев?
- Управления версиями, подготовки отчетности?
- Распределение ресурсов, автоматизация?
- Определения критериев завершения тестирования?
- Рецензирование документов?

Задачи тестирования

Динамическое тестирование:

- Реальное выполнение действий с программной

Статическое тестирование:

- Рецензирование
- Ревью кода

Все это помогает находить ошибки в ПО!

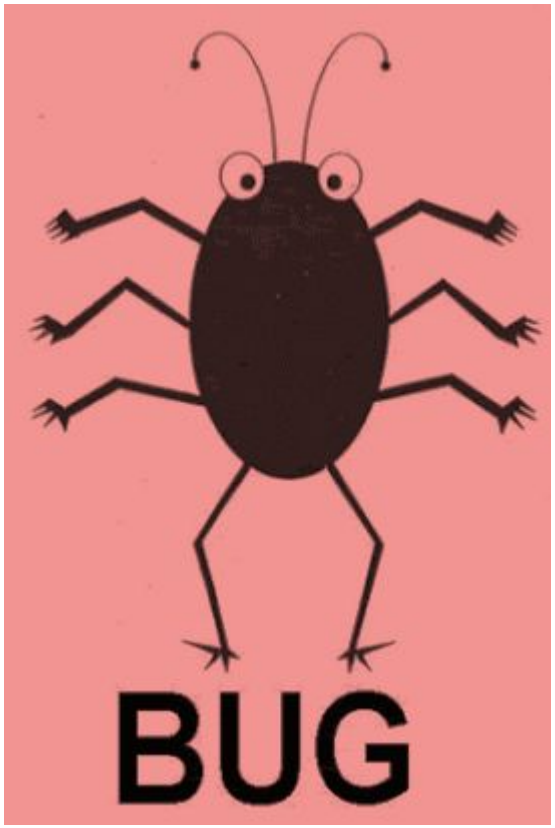
Задачи тестирования

Включают:

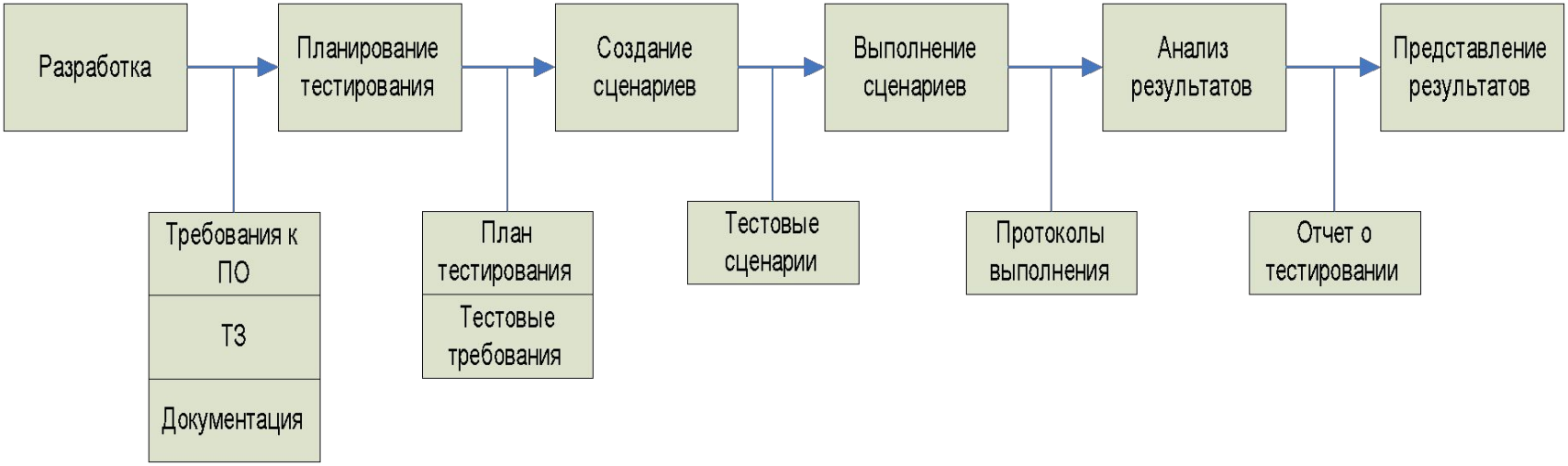
- Планирование и контроль
- Анализ и проектирование тестов
- Выполнение тестов
- Оценка выполнения тестов и отчетность
- Критерии завершения тестирования

а **НЕ** только выполнение тестовых сценариев.

Баги…

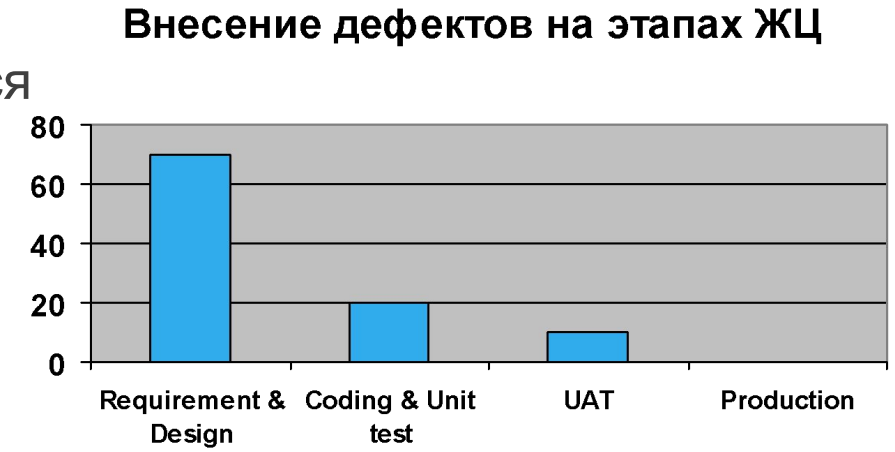


Процесс тестирования

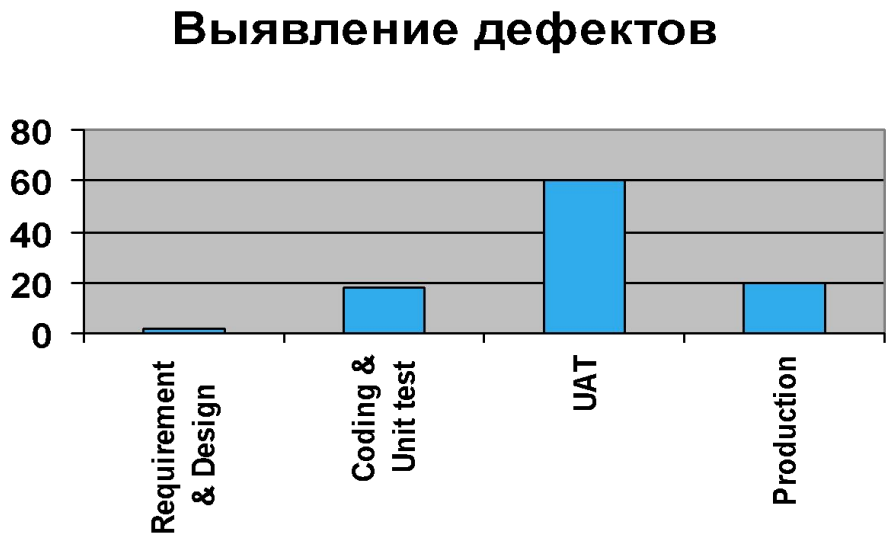


Важность тестирования

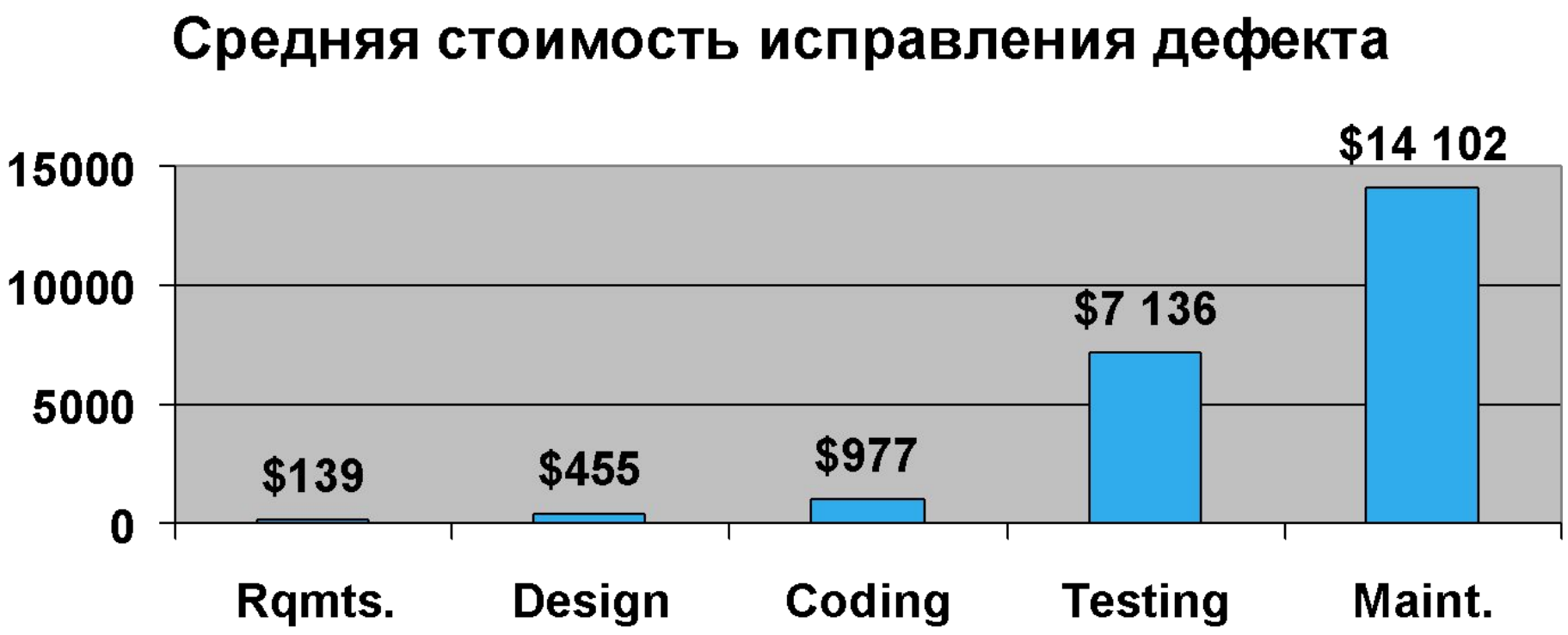
- Большинство дефектов вносятся на этапе сбора требований и разработки



- … а обнаруживаются на стадии приемочных испытаний.



Важность тестирования



Основные принципы по Майерсу

- Описание предполагаемых значений выходных данных или результатов должно быть необходимой частью тестового набора
- Следует избегать тестирования программы ее автором
- Необходимо досконально изучать результаты применения каждого теста
- Тесты для неправильных и непредусмотренных входных данных следует разрабатывать так же тщательно, как для правильных и предусмотренных
- Необходимо проверять не только, делает ли программа то, для чего она предназначена, но и ни делает ли она то, что не должна делать
- Не следует выбрасывать тесты, даже если программа уже не нужна
- Нельзя планировать тестирование в предположении, что ошибки не будут обнаружены
- Вероятность наличия необнаруженных ошибок в части программы пропорциональна числу ошибок, уже обнаруженных в этой части
- Тестирование - процесс творческий. Вполне вероятно, что для тестирования большой программы требуется больший творческий потенциал, чем для ее проектирования

Причины дефектов в программном обеспечении

Причинами возникновения ошибок (просчетов) являются дефекты (недочеты/помехи) в программном коде или документе, что может быть вызвано:

1. Человеком:

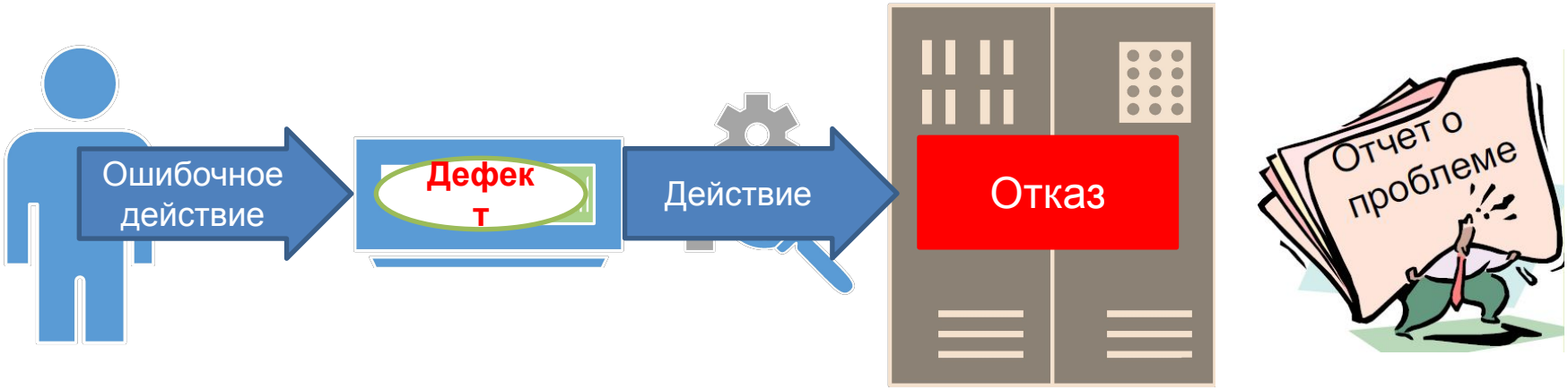
- Склонность человека ошибаться
- Чем выше давление на человека, тем выше риск возникновения ошибки
- Слабая способность учиться

2. Окружением:

- Радиация, электромагнитные поля, загрязнения
- Любые другие внешние факторы

Причины дефектов в программном обеспечении

Ошибка ->Дефект->Отказ



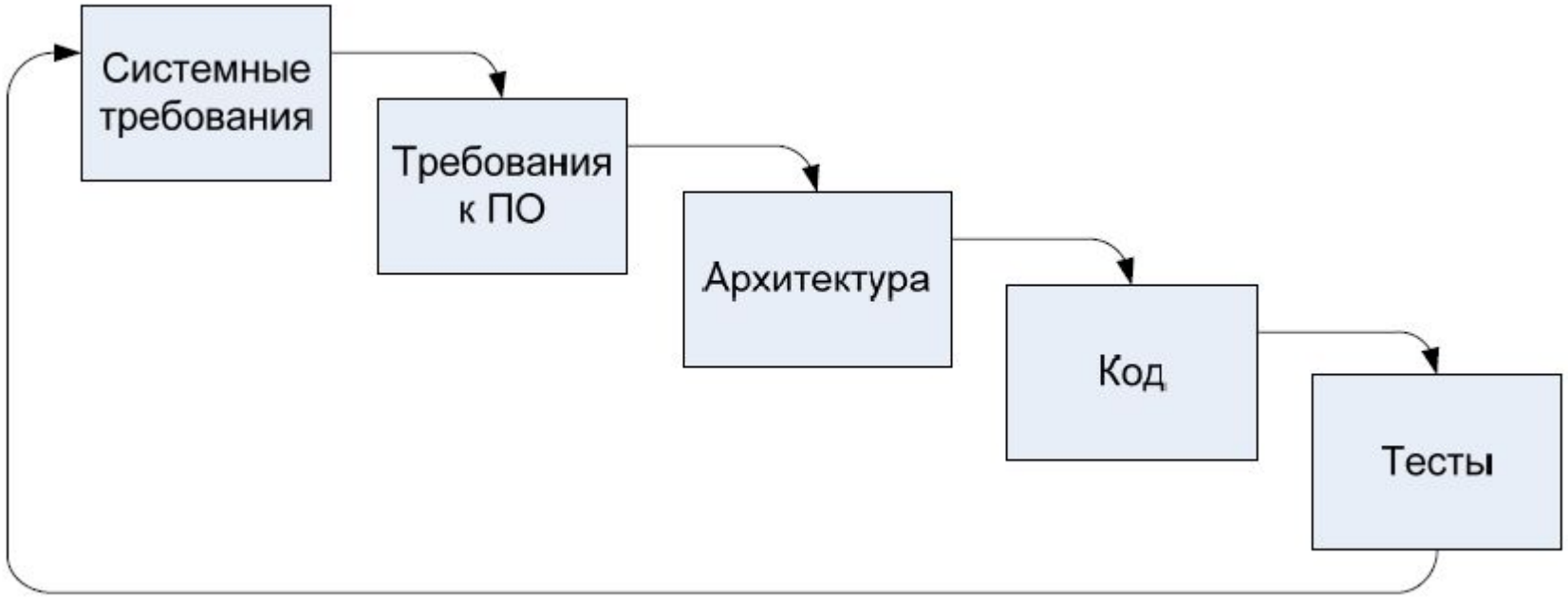
Причины дефектов в программном обеспечении

Почему появляются дефекты?

- Специфика проекта
- Неоднозначные спецификации/требования
- Низкий уровень коммуникаций и взаимодействия
- Низкий уровень управления требованиями
- Недооценка сложности задач проекта
- Разные команды, работающие над одной задачей (разработчики, тестировщики)
- Тесты выполняются в конце процесса разработки
- Нет процесса обеспечения качества на всех этапах жизненного цикла ПО

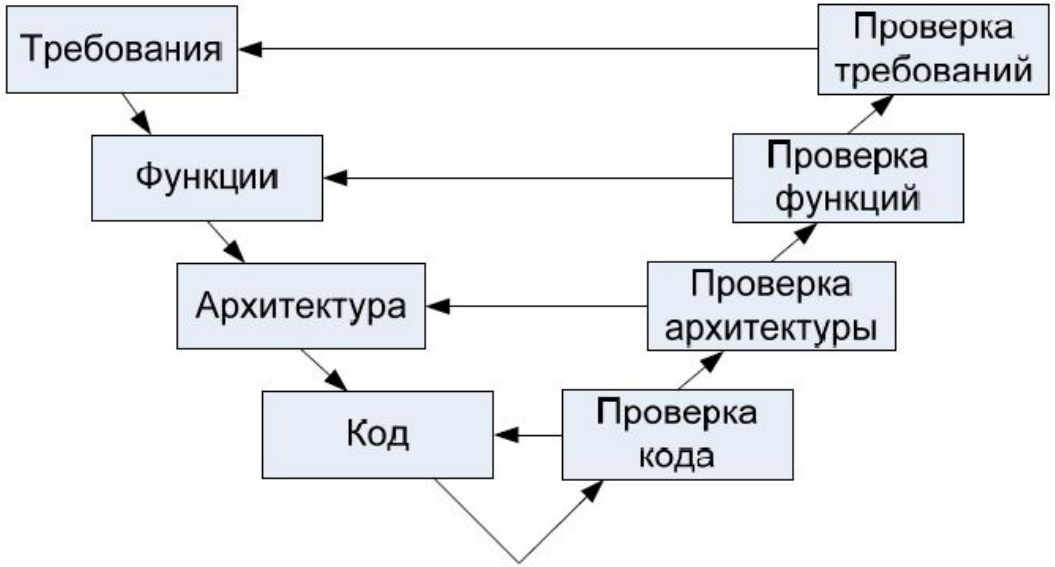
Жизненный цикл разработки ПО

Каскадный жизненный цикл (водопадный) - основан на постепенном увеличении степени детализации описания всей разрабатываемой системы. Каждое повышение степени детализации определяет переход к следующему состоянию разработки [Синицын 2006].



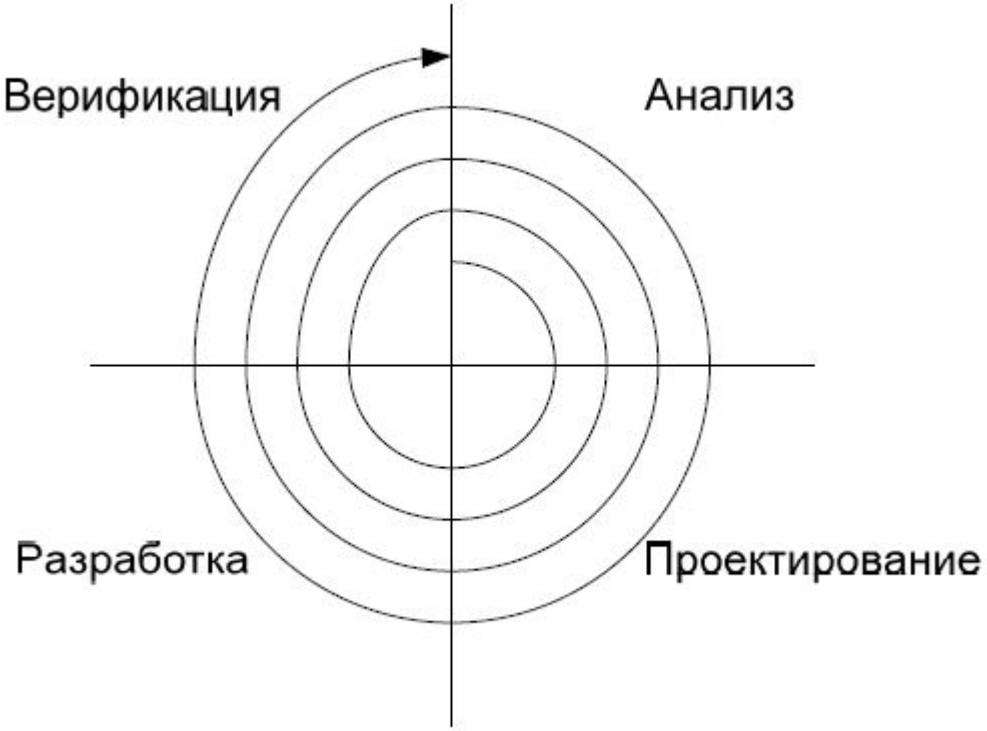
Жизненный цикл разработки ПО

V-образный жизненный цикл– это улучшенная версия классической каскадной модели, представляющая собой цепь обратной связи по отношению к основным процессам.



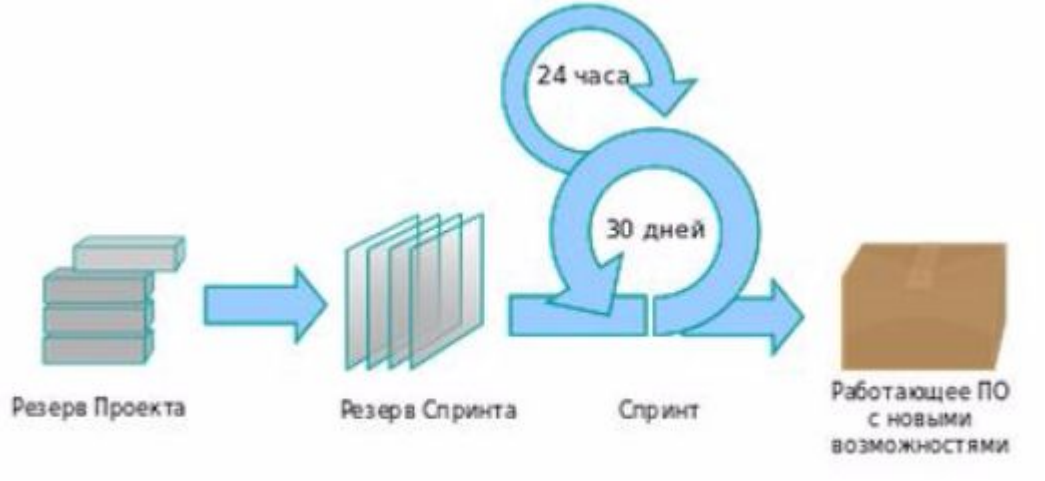
Жизненный цикл разработки ПО

Спиральный жизненный цикл - подразумевает разработку в виде последовательности версий, но в начале проекта определены не все требования. Требования уточняются в результате разработки версий.



Жизненный цикл разработки ПО

Гибкий жизненный цикл - серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Существует несколько методик, относящихся к классу гибких методологий разработки, в частности экстремальное программирование, Scrum, FDD.



Жизненный цикл разработки ПО

Сравнение подходов

Тип жизненного цикла	Длина цикла	Верификация и внесение изменений	Интеграция
Каскадный	Все этапы разработки системы. Длинный	В конце разработки всей системы. Редко.	Четко определенные до начала кодирования интерфейсы.
V - образный	Все этапы разработки системы. Длинный	В конце полной разработки каждого из этапов системы. Средне.	Редко изменяемые интерфейсы.
Спиральный	Разработка одной версии системы. Средний.	В конце разработки каждого из этапов версии системы. Средне.	Периодически изменяемые интерфейсы, редко меняемые в пределах версии.
Гибкий	Разработка одной истории. Короткий.	В конце разработки каждой истории. Очень часто	Часто изменяемые интерфейсы.

Роли в тестировании

Заказчик – представитель организации, заказавшей разрабатываемую систему. Обычно заказчик общается только с менеджерами проекта и специалистом по сертификации или внедрению. Заказчик имеет право изменять требования к продукту (при взаимодействии с менеджерами), читать проектную и сертификационную документацию, затрагивающую нетехнические особенности разрабатываемой системы.

Роли в тестировании

Первый кто необходим на проекте – это **тест-менеджер (менеджер проекта)**. Он занимается планированием тестирования, оценкой затрат, подбором команды, ставит цели и разрабатывает стратегию тестирования, а также занимается подбором инструментов, решений и подходов тестирования.

Роли в тестировании

Менеджер проекта– обеспечивает связь между заказчиком и проектной группой. Менеджер проекта управляет ожиданиями заказчика, разрабатывает и поддерживает бизнес-контекст проекта. Его задача – определить и обеспечить требования заказчика. Менеджер проекта имеет право изменять требования к продукту и финальную документацию на продукт.

Роли в тестировании

Разработчик – принимает технические решения, которые могут быть реализованы и использованы, создает продукт, удовлетворяющий спецификациям и ожиданиям заказчика. Он участвует в обзорах, реализует возможности продукта, участвует в создании функциональных спецификаций, отслеживает и исправляет ошибки. Разработчик имеет доступ ко всей проектной документации, включая документацию по тестированию, имеет право на изменение программного кода системы в рамках своих служебных обязанностей.

Роли в тестировании

Специалист по тестированию, это человек, который проходит сценарии в ручную, документирует дефекты, создает запросы на улучшения, а также коммуници

Роли в тестировании

Другие роли:

- Специалист по контролю качества
- Специалист по верификации
- Специалист по внедрению и сопровождению
- Специалист по безопасности
- Инструктор
- Технический писатель
- Релиз-менеджер

Тестирование и качество

Тестирование - это возможный способ оценки качества программного обеспечения в терминах найденных дефектов, как для функциональных требований, так и для нефункциональных требований и характеристик программного обеспечения (например, надежность, практичность, эффективность, сопровождаемость и переносимость).

Тестирование и качество

Контроль качества

1. Конструктивный:
- Методики и стандарты, покрывающие все стадии, имеющие цель предотвращение дефектов и отказов

2. Аналитический:
- Поиск дефектов и отказов
- Измерение количества вскрытых сбоев
- Измерение уровня соответствия заданным показателям
- Измерение соответствия атрибутам качества (удобство, производительность и т.д.)

Тестирование и качество

**ISO/IEC 25010:2011 – System and software engineering*

Тестирование и качество

Функциональная пригодность (functional suitability)

Степень, в которой продукт или система обеспечивают выполнение функции в соответствии с заявленными и подразумеваемыми потребностями при использовании в указанных условиях. *(ISO 25010)*

Включает:

- **функциональную полноту** (functional completeness): Степень покрытия совокупностью функций всех определенных задач и целей пользователя
- **функциональную корректность** (functional correctness): Степень обеспечения продуктом или системой необходимой степени точности корректных результатов
- **функциональную целесообразность** (functional appropriateness): Степень функционального упрощения выполнения определенных задач и достижения целей

Тестирование и качество

Уровень производительности (performance efficiency)

Производительность относительно суммы использованных при определенных условиях ресурсов. *(ISO 25010)*

Включает:

- **временные характеристики** (time behaviour): Степень соответствия требованиям по времени отклика, времени обработки и показателей пропускной способности продукта или системы
- **использование ресурсов** (resource utilization): Степень удовлетворения требований по потреблению объемов и видов ресурсов продуктом или системой при выполнении их функций
- **потенциальные возможности** (capacity): Степень соответствия требованиям предельных значений параметров продукта или системы

Тестирование и качество

Совместимость (compatibility)

Способность продукта, системы или компонента обмениваться информацией с другими продуктами, системами или компонентами, и/или выполнять требуемые функции при совместном использовании одних и тех же аппаратных средств или программной среды. *(ISO 25010)*

Включает:

- **сосуществование (совместимость)** (co-existence): Способность продукта совместно функционировать с другими независимыми продуктами в общей среде с разделением общих ресурсов и без отрицательного влияния на любой другой продукт
- **функциональная совместимость** (интероперабельность) (interoperability): Способность двух или более систем, продуктов или компонент обмениваться информацией и использовать такую информацию

Тестирование и качество

Удобство использования (usability)

Степень, в которой продукт или система могут быть использованы определенными пользователями для достижения конкретных целей с эффективностью, результативностью и удовлетворенностью в заданном контексте использования. *(ISO 25010)*

Включает:

- **определимость пригодности** (appropriateness recognizability): Возможность пользователей понять, подходит ли продукт или система для их потребностей, сравним ли с функциональной целесообразностью (functional appropriateness)
- **изучаемость** (learnability): Возможность использования продукта или системы определенными пользователями для достижения конкретных целей обучения для эксплуатации продукта или системы с эффективностью, результативностью, свободой от риска и в соответствии с требованиями в указанном контексте использования
- **управляемость** (operability): Наличие в продукте или системе атрибутов, обеспечивающих простое управление и контроль
- **защищенность от ошибки пользователя** (user error protection): Уровень системной защиты пользователей от ошибок
- **эстетика пользовательского интерфейса** (user interface aesthetics): Степень "приятности" и "удовлетворенности" пользователя интерфейсом взаимодействия с пользователем
- **доступность** (accessibility): Возможность использования продукта или системы для достижения определенной цели в указанном контексте использования широким кругом людей с самыми разными возможностями

Тестирование и качество

Надежность (reliability)

Степень выполнения системой, продуктом или компонентом определенных функций при указанных условиях в течение установленного периода времени. *(ISO 25010)*

Включает:

- **завершенность** (maturity): Степень соответствия системы, продукта или компонента при нормальной работе требованиям надежности
- **готовность** (availability): Степень работоспособности и доступности системы, продукта или компонента (ISO 24765)
- **отказоустойчивость** (fault tolerance): Способность системы, продукта или компонента работать как предназначено, несмотря на наличие дефектов программного обеспечения или аппаратных средств
- **восстанавливаемость** (recoverability): Способность продукта или системы восстановить данные и требуемое состояние системы в случае прерывания или сбоя

Тестирование и качество

Защита, защищенность (security)

Степень защищенности информации и данных, обеспечиваемая продуктом или системой путем ограничения доступа людей, других продуктов или систем к данным в соответствии с типами и уровнями авторизации. *(ISO 25010)*

Включает:

- **конфиденциальность** (confidentiality): Обеспечение продуктом или системой ограничения доступа к данным только для тех, кому доступ разрешен
- **целостность** (integrity): Степень предотвращения системой, продуктом или компонентом несанкционированного доступа или модификации компьютерных программ или данных (ISO 24765)
- **неподдельность** (non-repudiation): Степень, с которой может быть доказан факт события или действия таким образом, что этот факт не может быть отвергнут когда-либо позже
- **отслеживаемость** (accountability): Степень, до которой действия объекта могут быть прослежены однозначно
- **подлинность** (authenticity): Степень достоверности тождественности объекта или ресурса требуемому объекту или ресурсу

Тестирование и качество

Сопровождаемость, модифицируемость (maintainability)
Результативность и эффективность, с которыми продукт или система могут быть модифицированы предполагаемыми специалистами по обслуживанию. *(ISO 25010)*

Включает:

- **модульность** (modularity): Степень представления системы или компьютерной программы в виде отдельных блоков таким образом, чтобы изменение одного компонента оказывало минимальное воздействие на другие компоненты (ISO 24765)
- **возможность многократного использования** (reusability): Степень, в которой актив может быть использован в нескольких системах или в создании других активов
- **анализируемость** (analysability): Степень простоты оценки влияния изменений одной или более частей на продукт или систему или простоты диагностики продукта для выявления недостатков и причин отказов, или простоты идентификации частей, подлежащих изменению
- **модифицируемость** (modifiability): Степень простоты эффективного и рационального изменения продукта или системы без добавления дефектов и снижения качества продукта
- **тестируемость** (testability): Степень простоты эффективного и рационального определения для системы, продукта или компонента критериев тестирования, а также простоты выполнения тестирования с целью определения соответствия этим критериям

Тестирование и качество

Переносимость, мобильность (portability)

Степень простоты эффективного и рационального переноса системы, продукта или компонента из одной среды (аппаратных средств, программного обеспечения, операционных условий или условий использования) в другую.*(ISO 25010)*

Включает:

- **адаптируемость** (adaptability): Степень простоты эффективной и рациональной адаптации для отличающихся или усовершенствованных аппаратных средств, программного обеспечения, других операционных сред или условий использования
- **устанавливаемость** (installability): Степень простоты эффективной и рациональной, успешной установки и/или удаления продукта или системы в заданной среде
- **взаимозаменяемость** (replaceability): Способность продукта заменить другой конкретный программный продукт для достижения тех же целей в тех же условиях

Когда заканчивать тестирование?

Если тестирование завершить рано:

- Есть риск того, что ошибки останутся в ПО
- Не рационально будут потрачены время и ресурсы

Как понять, что тестирование можно завершить:

- Уровень риска достаточен для внедрения ПО
- Получено достаточно данных для принятия решения о внедрении

Стоимость тестирования должна быть меньше ожидаемой стоимости ошибок

Когда заканчивать тестирование?

Заключение:

- Ошибки в ПО могут привести в серьезным финансовым потерям, тестирование позволяет снизить этот риск
- Тестирование не может гарантировать, что ошибок в ПО нет, но может найти эти ошибки
- Интенсивность и глубина тестирования должна определяться на основании рисков
- Исключить ненужные тесты

Необходимо расставлять приоритеты так, чтобы к концу тестирования было сделано максимум из возможного!

Статическое тестирование

Статическое тестирование проводится без запуска программы или программного кода продукта (ревью).

Цель: выявление ошибок и потенциальных проблем на ранних этапах разработки ПО.

Ревью может подлежать:

- спецификации требований
- спецификации дизайна
- код
- планы тестирования
- спецификации тестирования
- тестовые сценарии
- руководства пользователя
- веб- страницы, и .т.д.

Динамическое тестирование

Динамическое тестирование проводится путем запуска ПО и проверки его функционала.

Цель: выявление ошибок на этапе реализации ПО

Динамическое тестирование делиться на подтипы (**способы тестирования**):

- Белый ящик
- Черный ящик
- Серый ящик

Черный ящик

Полностью покрыты все …

- •… входные данные
- •… комбинации входных данных
- •… последовательности комбинаций входных данных

Белый ящик

Полностью покрыты все …

- •… строки кода программы
- •… ветви в коде программы
- •… пути в коде программы

Виды тестирования

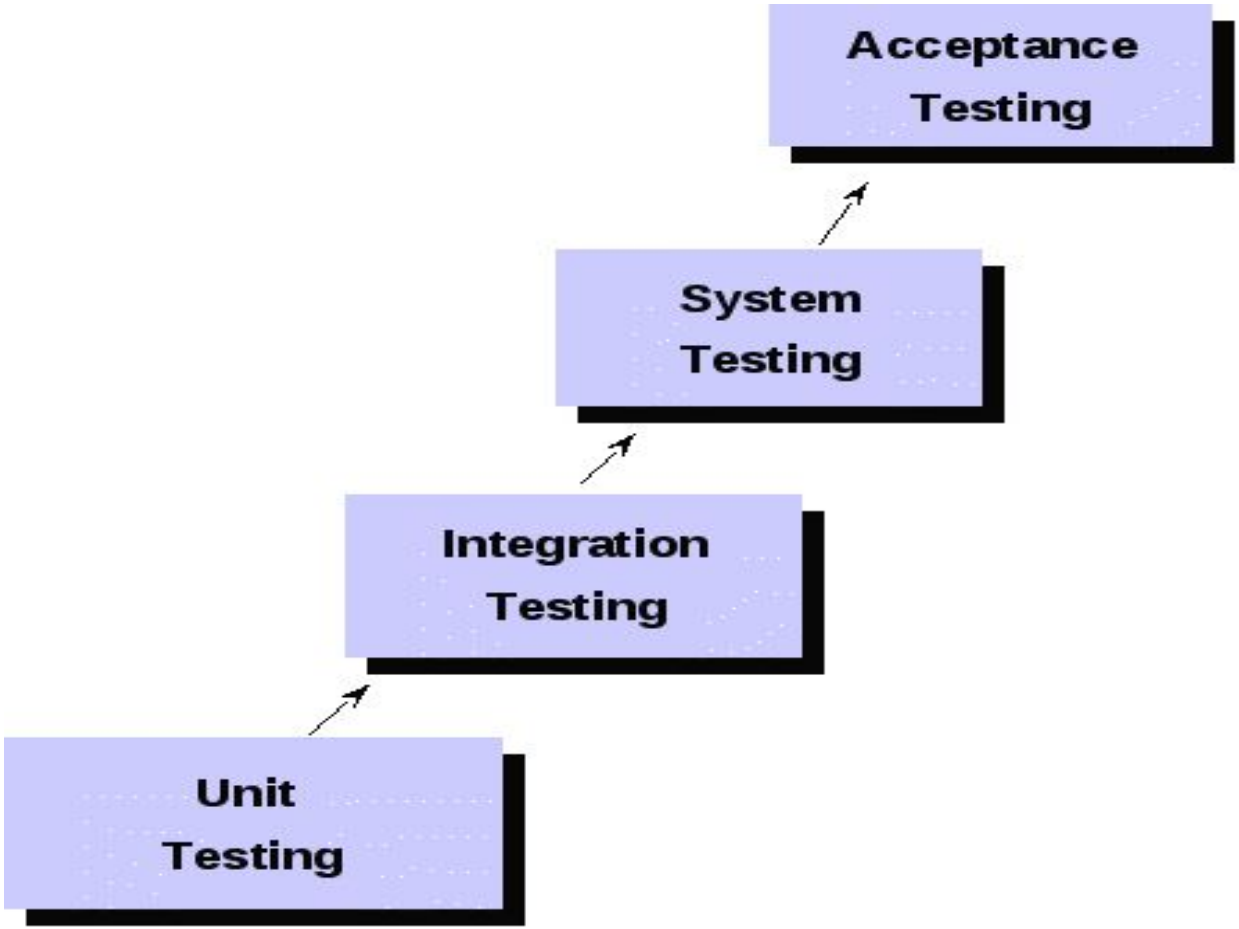
Функциональные виды тестирования– базируются на функциях и особенностях, а также взаимодействии с другими системами. Рассматривают внешнее поведение системы.

- Функциональное тестирование
- Тестирование безопасности
- Тестирование взаимодействия

Не функциональные виды тестирования– описывают ситуации, необходимые для определения характеристик ПО, которые могут быть измерены разными величинами.

- Все виды нагрузочного тестирования
- Тестирование удобства использования
- Тестирование на отказ и восстановление
- Конфигурационное тестирование

Уровни тестирования



Модульное тестирование (unit testing)

Модульное тестирование (или компонентное) – проверка корректности работы отдельных компонентов системы, выполнения ими своих функций и предполагаемых проектом характеристик.

Цель: выявление локализованных в модуле ошибок при реализации алгоритмов, а также определение степени готовности системы к переходу на следующий уровень тестирования.

A simple unit test

```
# test_port1.py

import unittest
from portfolio1 import Portfolio

class PortfolioTest(unittest.TestCase):
    def test_buy_one_stock(self):
        p = Portfolio()
        p.buy("IBM", 100, 176.48)
        assert p.cost() == 17648.0
```

```
$ python -m unittest test_port1
.
----------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

Интеграционное тестирование

Интеграционное тестирование – проверка работоспособности ПО между его компонентами, а также при взаимодействия с различными частями (модулями) системы.

Подходы к интеграционному тестированию:

- **Снизу вверх (Bottom Up Integration).** Все низкоуровневые модули, процедуры или функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования.
- **Сверху вниз (Top Down Integration).** Вначале тестируются все высокоуровневые модули, и постепенно один за другим добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем по мере готовности они заменяются реальными активными компонентами.
- **Большой взрыв ("Big Bang" Integration).** Все или практически все разработанные модули собираются вместе в виде законченной системы или ее основной части, и затем проводится интеграционное тестирование.

Интеграционное и модульное тестирование

Системное тестирование

Системное тестирование – проверка работоспособности ПО в соответствии с предъявляемыми функциональными и нефункциональными требованиями в системе в целом.

Можно выделить два подхода к системному тестированию:

- **на базе требований** (requirements based). Для каждого требования пишутся тестовые сценарии (test cases), проверяющие выполнение данного требования.
- **на базе случаев использования** (use case based). На основе представления о способах использования продукта создаются случаи использования системы (Use Cases). По конкретному случаю использования можно определить один или более сценариев. На проверку каждого сценария пишутся тестовые сценарии (test cases), которые должны быть протестированы.

Приемочное тестирование

Приемочное тестирование – проверяет поведение системы на предмет удовлетворения требований Заказчика.

Можно выделить 3 основных подхода к приемочному тестированию:

- **Пользовательское тестирование (UAT)**. Тестирование, которое проводится конечными пользователями системы с целью принятия решения о внедрении.
- **Альфа-тестирование**. Ручное тестирование потенциальными пользователями, заказчиками или независимой командой тестирования на стенде разработки. Альфа-тестирование часто используется как форма внутреннего приемочного тестирования перед проведением бета-тестирования.
- **Бета-тестирование**. Проводится после альфа-тестирования и может использоваться как приемочное тестирование внешними пользователями.

Виды тестирования

- **По проверяемым свойствам**
- ISO 9126
- Функциональность
- Надежность
- Производительность
- Нагрузочное
- Переносимость
- Удобство использования
- Удобство сопровождения
- Регрессионное
- Аттестационное (соответствия)

- **По исполнителю**
- При разработке
- Альфа
- Бета
- Приемочное

- **По уровню**
- Модульное (Компонентное)
- Интеграционное
- Системное

- **По способу**
- «Черного ящика»
- «Белого ящика»
- «Серого ящика»

- **На отказ**
- «Дымовое»
- Стрессовое

Особенности требований к программному обеспечению

IEEE Standard Glossary определяет требования как:

Условия или возможности, необходимые пользователю для решения проблем или достижения целей;

Требования к ПО состоят из трех уровней —бизнес-требования, требования пользователей и функциональные требования. Вдобавок каждая система имеет свои нефункциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы.

Требования пользователей описывают цели и задачи, которые пользователям даст система.

Функциональные требования определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований. Они содержат положения с традиционным «должен» или «должна»: «Система должна по электронной почте отправлять пользователю подтверждение о заказе». Функциональные требования документируются в спецификации требований к ПО (software requirements specification, SRS), где описывается так полно, как необходимо, ожидаемое поведение системы.

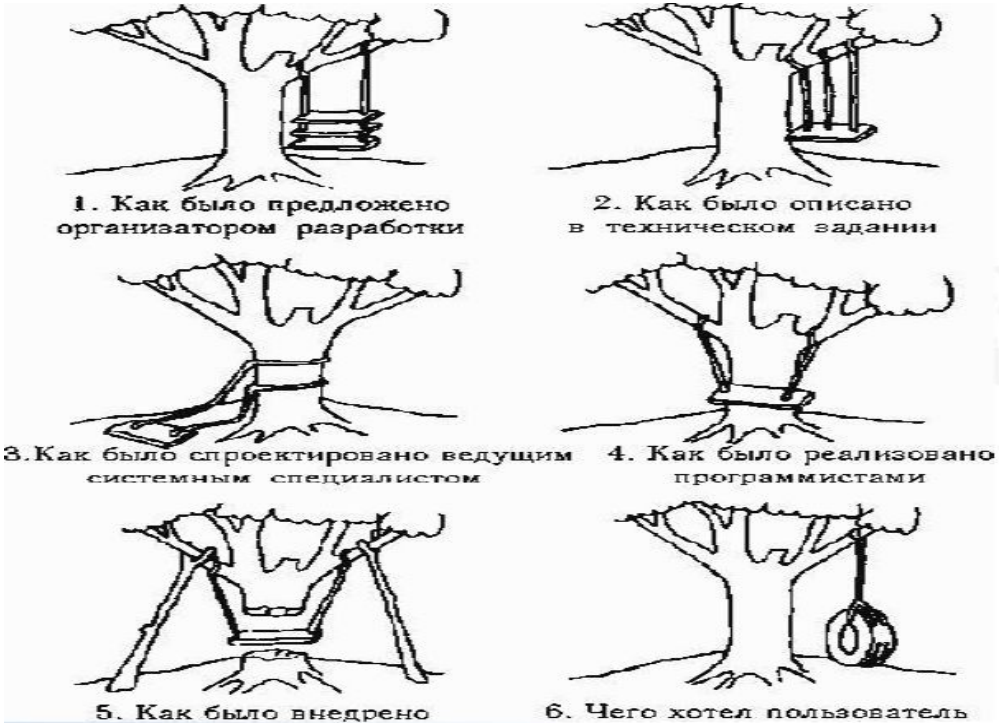
Особенности требований к программному обеспечению

Системные требования(system requirements) -это высокоуровневые требования к продукту, которые содержат многие подсистемы. Говоря о системе, мы подразумеваем программное обеспечение или подсистемы ПО и оборудования. Люди —часть системы, поэтому определенные функции системы могут распространяться и на людей

Нефункциональные требованияописывают цели и атрибуты качества. Атрибуты качества (quality attributes) представляют собой дополнительное описание функций продукта, выраженное через описание его характеристик, важных для пользователей или разработчиков. К таким характеристикам относятся:

- легкость и простота использования
- легкость перемещения
- целостность
- эффективность и устойчивость к сбоям
- внешние взаимодействия между системой и внешним миром
- ограничения дизайна и реализации. Ограничения (constraints) касаются выбора возможности разработки внешнего вида и структуры продукта

В реальной жизни



Анализ требований (тестирование требований)

- **Тестопригодные** требования –степень выраженности требований в терминах, допускающих начало работы над разработкой тестов (и в последствии над тестовыми сценариями) и выполнение тестов для определения соответствия заявленным требованиям [IEEE 610]

- корректность
- однозначность;
- полнота;
- непротиворечивость;
- упорядоченность по важности и стабильности;
- проверяемость (верифицируемость или тестопригодность);
- модифицируемость;
- трассируемость;
- понятность.

Анализ требований (тестирование требований)

- **Корректность.** SRS является корректной, если, и только, если каждое требование, изложенное в ней, является требованием, которому должно удовлетворять программное обеспечение.
- **Однозначность.** SRS является однозначной, если и только, если каждое изложенное в ней требование может интерпретироваться только однозначно.
- **Завершенность(полнота).**
- **Непротиворечивость.** Внутренняя непротиворечивость.

а) Могут входить в конфликт заданные характеристики реальных объектов.

б) Между двумя заданными действиями может существовать логический или временной конфликт.

в) Два или более требований могут описывать один и тот же реальный объект, но использовать для этого объекта различные условия.

- **Упорядочивание по значимости.** SRS является упорядоченной по значимости, если каждое требование в ней имеет идентификатор, указывающий или значимость или устойчивость этого конкретного требования

Выделение требований

Ранжирование функционала по важности (или по рискам):

- Обязательно
- Важно
- Полезно
- Иногда пригождается
- На всякий случай

По частоте использования / востребованности:

- Для всех
- Для большинства
- Для многих
- Для некоторых
- Для единиц

Систематизация и описание требований

Проверяемость. Требование является проверяемым, если и только, если существует некий конечный эффективный процесс, используя который пользователь или машина могут убедиться, что программное изделие удовлетворяет этому требованию.

Непроверяемые требования включают формулировки типа "работает хорошо", "хороший интерфейс с пользователем" и "обычно должно происходить".

Выходные данные программы должны вырабатываться в пределах 20 секунд в течение 60 % временного интервала события; и должны вырабатываться в пределах 30 секунд в течение 100 % временного интервала события.

Систематизация и описание требований

Время отклика системы должно находится в приемлемых рамках

Время отклика	(Отклика какой операции?)
системы	(Что такое система?)
должно находиться в	(Условия? Нагрузка?)
приемлемых рамках	(Цифры?)

Систематизация и описание требований

Пример тестопригодного требования

- Время отклика системы с точки зрения конечного пользователя (end-to-end) во время продуктивной нагрузки (50 пользовательских сессий в режиме «менеджер» / 15 пользовательских сессий в режиме «аналитик») при загруженности пропускного канала от клиентской системы до сервера приложений в пределах 50% для сети 100 Mb/secи утилизации ресурсов сервера приложений (CPU, RAM) в рамках 70-80%, а клиентской машины в рамках 40-60%, не должно превышать 1 секунды для операций создания записи (сущности) и 3 секунд для операций поиска.
- Время выполнения аналитических отчётов определяется отдельно для каждого отчёта

Систематизация и описание требований

Примеры требований: проанализируйте следующие требования с точки зрения тестопригодности.

Решение квадратного уравнения. Программа получает на вход три вещественных числа и интерпретирует их как коэффициенты квадратного уравнения. Результатом работы программы являются два числа —корни этого квадратного уравнения.

***Распознавание треугольника*.** Программа получает на вход три натуральных числа и интерпретирует их как длины сторон треугольника. Результатом работы программы будет сообщение является ли треугольник разносторонним, равнобедренным или равносторонним.

Банкомат должен выполнять следующие операции:

- Прием пластиковой карты
- Проверка ПИН-кода
- Выдача наличных
- Выдача чека с информацией об остатке на счете

Валидация и верификация требований

Валидация представляет собой проверку корректности и полноты требований, то есть проверяет, что зафиксированные в требованиях ограничения и пожелания действительно представляют потребности пользователей, заказчиков и других заинтересованных лиц, а также, что все их существенные потребности нашли соответствующее отражение в требованиях.

Верификация проверяет внутреннюю согласованность, непротиворечивость и однозначность требований, а также их проверяемость и возможность проследить связи требований друг с другом, с кодом, тестами и другими проектными документами.

Составление тестов на основе требований

- Связывание рисков и требований выявляет плохо сформулированные или отсутствующие требования.

- Тестирование может быть сфокусировано на наиболее важных частях информационной системы, «атакуя» самые опасные технологические риски.

- Работа и коммуникация идет «на едином языке» заказчика и других заинтересованных лиц. Например, заказчику будут понятны отчеты о ходе тестирования проекта, и будет легче принимать решение, инвестировать ли ресурсы в дополнительное тестирование, или риск уже приемлем и можно запускать систему в эксплуатацию.

Документы процесса тестирования

Перечень основных артефактов тестирования:

- Системные требования
- Результат ревью системных требований
- Варианты использования (Use cases)
- План(ы) тестирования
- Стратегия тестирования (Test strategy)
- Тест план (Test Plan)
- Тестовые сценарии (Test cases)
- Матрица покрытия требований тестовыми сценариями (Traceabilitymatrix)
- «Список проверки» (Checklist)
- Описание ошибки (Bug report)
- Отчет о тестировании (Test result report)

IEEE 829

1. **План тестирования**. Цель плана тестирования – обеспечить полноту процесса тестирования. План тестирования разрабатывается на основе ТЗ – требований к продукту. В плане тестирования описываются способы, виды и критерии тестирования для всех требований, необходимые ресурсы и порядок выполнения тестирования.
2. **Спецификация тест-дизайна.** Этот документ определяет, как будет тестироваться каждая функция или группа функций программы.
3. **Спецификация тестов** : Разрабатывается отдельно для тестирования каждой функциональности
4. **Спецификация проектирования процедуры тестирования**: устанавливает этапы, необходимые для выполнения набора тестов. Разрабатывается, если последовательность действий имеет какие-либо специфичные особенности, не описанные в плане тестирования.

IEEE 829

5. *Отчет о передаче тестируемого элемента*: определяет тестируемый элемент, указывает локализацию элемента, среду развертывания, а также имя специалиста, ответственного за тестирование этого элемента.

6. *Журнал тестирования*: содержит важные детали, касающиеся выполнения теста.

7. *Отчет о происшествиях*, возникших в ходе тестирования: документировать обнаруженные во время тестирования события, которые требуют дополнительного изучения. Система составления отчетов о неполадках — это один из способов регистрации сведений.

8. Резюмирующий отчет о тесте: дает общую оценку работе по тестированию

Планирование работ по тестированию

Скриптовое

- Тест-дизайнер делает тесты
- Тестировщик их проходит

Плюсы:

- Узкая специализация
- Экономия ресурсов
- Высокая планируемость и предсказуемость
- Качественная отчетность

Минусы:

- Эффект «пестицида»
- Рутина
- Негибкость

Исследовательское

- Тестировщик сам придумывает тесты
- Проходит
- По результатам придумывает новые тесты

Плюсы:

- Гибкость, адаптивность к изменениям
- Скорость начала работы
- Интереснее задачи

Минусы:

- Зависимость от квалификации
- Отсутствие гарантий
- Сложности с отчетностью
- Невозможность планирования

Выбор подхода к тестированию

- Критичность продукта;
- Используемые методологии разработки;
- Сроки выпуска продукта;
- Статус проекта;
- Требования к качеству;
- Приоритеты компании;
- Количество сотрудников на проекте;
- Человеческий фактор

Модель Алистера Коберна

Что должно быть в тест-плане

- Что тестировать;
- Как это тестировать;
- Что НЕ тестировать;
- Трудозатраты на тестирование;
- Приоритеты;
- Последовательности;
- Риски;
- Всякие формальности ☺️

Назначение и состав тест-плана

Тест-план позволяет определить объемы, подходы и методы к выполнению тестирования, заранее определить ограничения, риски, критерии начала и завершения тестирования, а также жестко зафиксировать сроки и команду для выполнения поставленных задач по тестированию.

Назначение и состав тест-плана

Тест-План включает в себя:

- Подходы к тестированию, методы (стратегия тестирования)
    - Область тестирования
    - Задачи на тестировании
    - Критерии начала/завершения тестирования
    - Критерии прохождения/не прохождения тест-кейса
    - Ожидаемые результаты тестирования
- Тестовые среды
- Ограничения тестирования
- Состав команды и компетенции
- Ответственность заинтересованных сторон
- План проведения тестирования

Стратегия тестирования

Стратегия тестирования. Первое действие в планировании испытаний предусматривает разработку стратегии тестирования на высоком уровне. В общем случае стратегия тестирования должна определять объемы тестовых работ, типы методик тестирования, которые должны применяться для обнаружения дефектов, процедуры, уведомляющие об обнаружении и устраняющие дефекты, критерии входа и выхода из испытаний, которые управляют различными видами тестирования.

Стратегия тестирования

Область тестирования – это компоненты системы, автоматизированные бизнес процессы, которые подвергаются тестированию в новом релизе/проекте.

Чтобы определить область тестирования нужно ответить на вопрос:

- **Что надо тестировать?**
- **Что будем тестировать?**

Основные разделы:

- Тестируемые системы
- Тестируемые бизнес процессы
- Элементы тестирования
- Главные тестовые сценарии

Стратегия тестирования

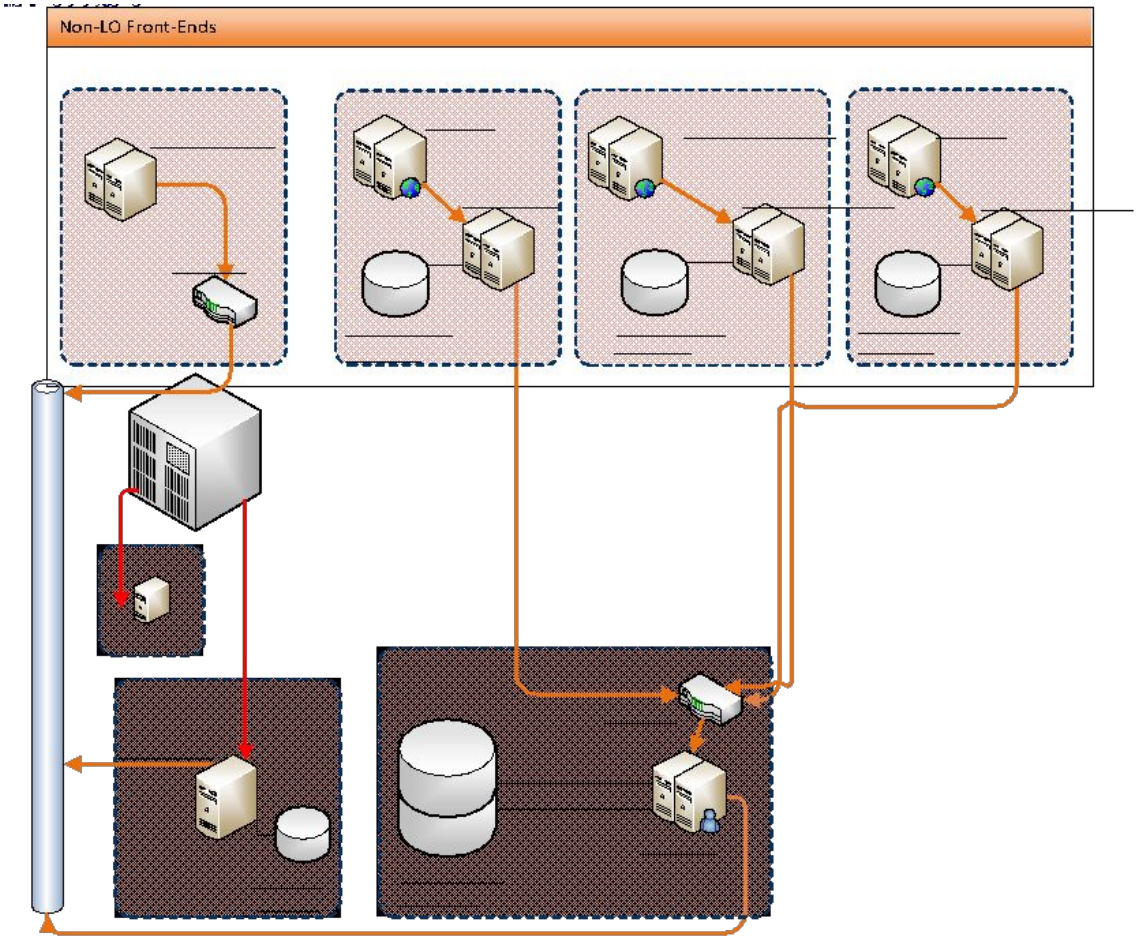
- Определить подход к тестированию:
  - Виды тестирования
  - Тестируемая функциональность с приоритетами
  - Риски тестирования
  - Особые условия тестирования
- Определить критерии входа и выхода для каждой стадии тестирования, равно как и все точки контроля качества, для чего потребуется участие специалистов по тестированию.
- Определить стратегию автоматизации в случае, если планируется использование автоматизации какого-либо вида тестовой деятельности.
- Определить перечень документов, формирующийся по результатам тестирования

Тестовые среды

В описание тестовых сред входит:

- Описание тестовой среды, ее назначение
- Описание систем, входящих в тестовую среду
- Конфигурация
- Критерии приемки/готовности тестовой среды
- Архитектурная схема взаимодействия

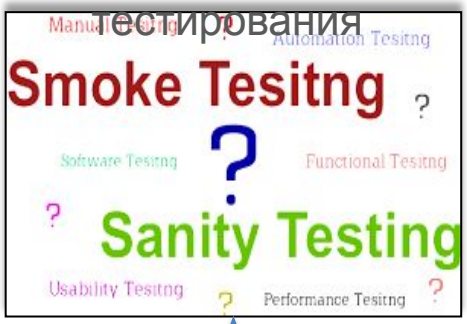
Тестовые среды



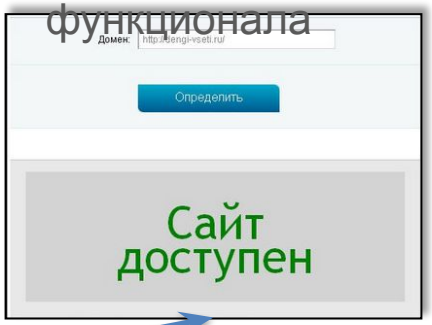
Ограничения тестирования

По разрабатываемым артефактам

По видам тестирования



По доступности функционала



Ограничения тестирования

По работе тестовых стендов

По способу взаимодействия

По ролевой модели

Состав команды и компетенции

Роль	Количество	Обязанности
Руководитель Проекта	1	• Оперативное управление всем проектом;
Ведущий инженер по обеспечению качества (SQE)	1	• Взаимодействие с Заказчиком по организационным в опросам; • Организация и операционный контроль работы кома нды; • Планирование и контроль выделенной части работ; • Взаимодействие с техническим персоналом Заказчика; • Разработка методики тестирования; • Формирование отчетов о статусе проекта; • Руководство за выполнением тестов; • Анализ результатов тестирования; • Отчетность по работам;
Инженер по обеспечению качес тва (QE)	2	• Контроль разработки тестовой модели; • Разработка тестовых сценариев высокой сложности; • Выполнение сложных видов тестирования; • Оценка критичности дефектов; • Определение источника возникновения дефектов.
Дизайнер тестов (TD)	4	• Разработка тестовых сценариев; • Выполнение тестирования; • Фиксирование дефектов • Перепроверка дефектов

Тест-дизайн

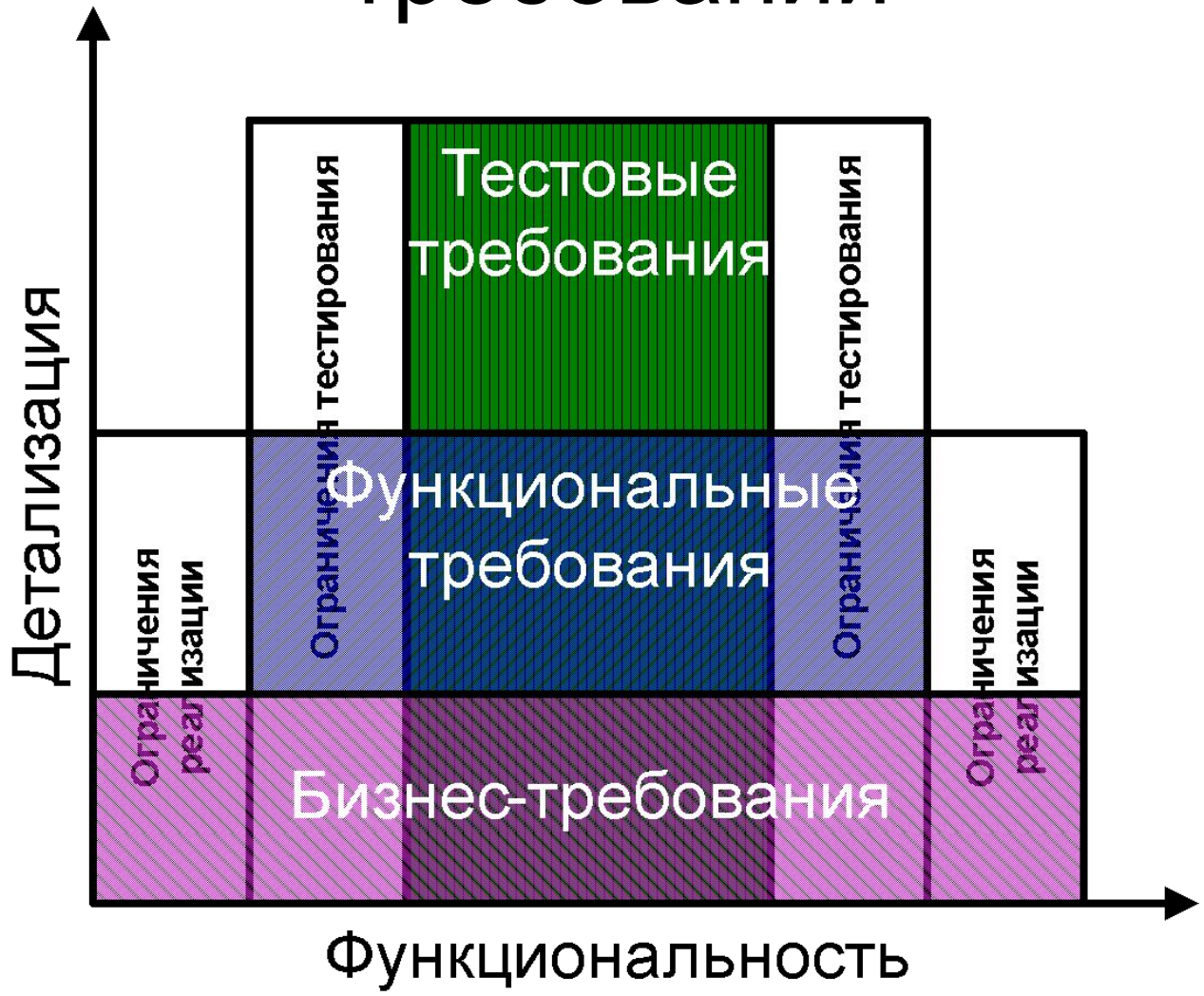
Тест – дизайн – это этап процесса тестирования ПО, который включает создание/проектирование тестовых сценариев и определение необходимых типов тестов, для достижения заданного уровня тестового покрытия приложения или системы под тестом.

Тест-дизайн задает объемы работ и границы проекта по тестированию (что влияет на стоимость тестирования)

Тест-дизайн – это попытка найти баланс между трудозатратами на тестирование и приемлемым уровнем доверия к результатам тестирования.

Тестовые требования

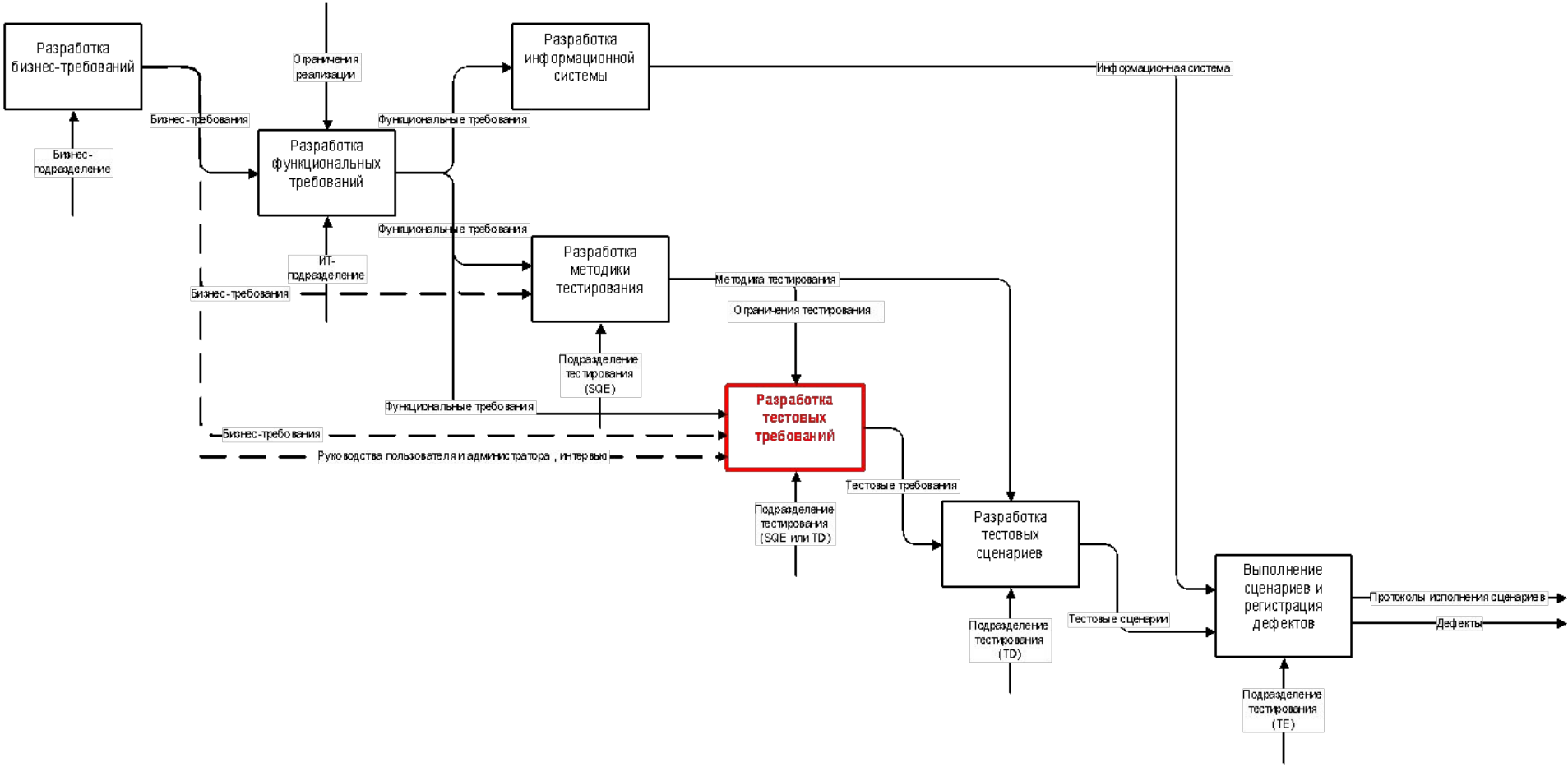
Диаграмма детализации требований



Назначение тестовых требований

- Использовать их, как исходный документ для написания тестовых сценариев
- Ограничить тестируемую функциональность.
- Описать систему с учетом критериев качества
  - Структурирование описания системы
  - Детализация описания системы
  - Устранение неоднозначности описания системы
  - Устранение неполноты описания системы
  - Обеспечение тестируемости описания системы
- Изучить и проанализировать систему с точки зрения тестирования.
- Оценить трудоемкость.
- Установить приоритеты в тестировании.
- Отслеживать тестовое покрытие

Процесс тест-дизайна



Этапы тест-дизайна

- **Выявление доступной документации описывающей систему.**
- **Получение выявленной документации.**
- **Выяснение степени актуальности полученной документации.**
  В общем случае документация может не соответствовать тестируемой версии приложения, поэтому не может быть использована либо полностью, либо в какой-то своей части.
- **Изучение полученной документации.** Данный шаг включает чтение и анализ полученной документации на предмет полноты.
- **Составление списка лиц для устного получения информации о системе.** Этот шаг необходимо выполнить обязательно, если полученной документации недостаточно для написания полноценных тестовых требований.
- **Проведение интервью с компетентными специалистами.** Устное общение с лицами из списка, подготовленного на предыдущем пункте, с целью выявления и уточнения функциональных требований к системе и последующим написанием тестовых требований. Этот шаг необходимо выполнить обязательно, если полученной документации недостаточно для написания полноценных тестовых требований.
- **Подготовка иерархической структуры тестовых требований.** Полученная документация анализируется с целью выделения структурных элементов. Структурные элементы обобщаются в группы, которые также могут быть сгруппированы. В результате формируется иерархическая структура (см. п.6.1 Иерархическая структура тестовых требований).
- **Анализ полноты иерархической структуры тестовых требований.** Иерархическая структура должна полностью покрывать тестируемую функциональность.
- **Наполнение требований описаниями.**
- **Анализ требований на предмет соответствия критериям качества и внесение изменений.**

Матрица покрытия требований

	ФТ 1	ФТ 2	ФТ 3	ФТ 4
ТТ 1	X	X		
ТТ 2			X	
ТТ 3				X
ТТ 4				X

Тестовый сценарий

Тест-кейс – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. Проще говоря, тест-кейс – это сценарий с шагами, которые ведут к определенному поведению системы.

2 типа тест-кейсов:

- Каскадные
- Независимые

Тестовый сценарий

Тест-кейс должен содержать:

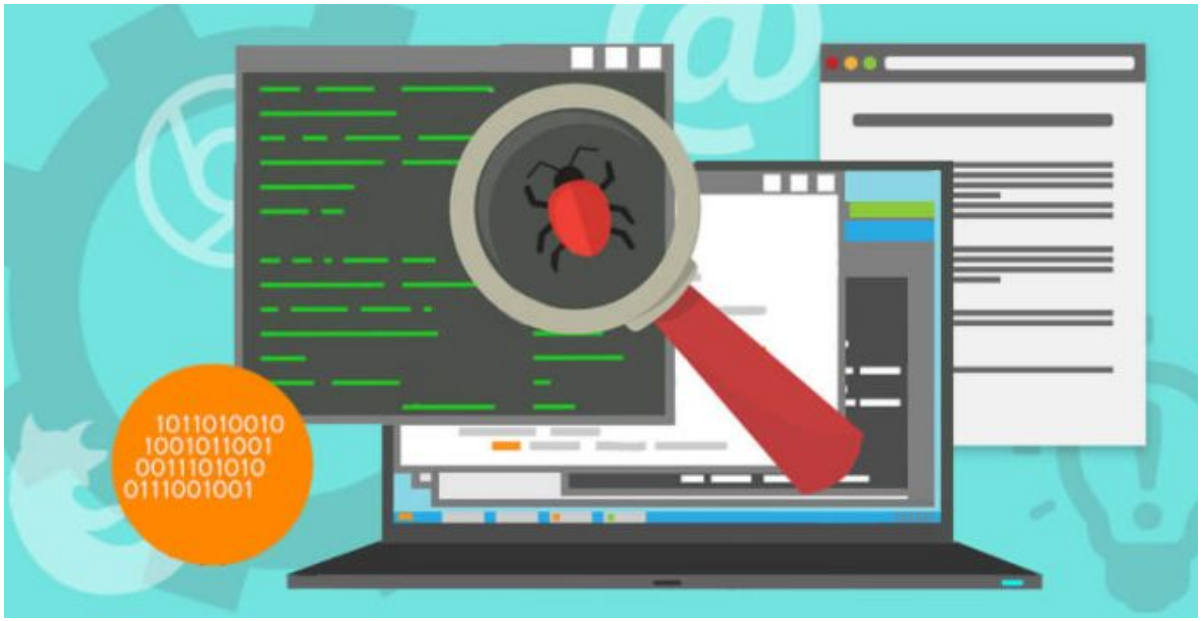
- заголовок (title)
- номер (ID)
- предусловие (preconditions)
- тестовые данные (test data)
- шаги (steps)
- ожидаемый результат (expected result)

А также:

- дата создания
- автор
- дата последнего изменения
- назначен на…
- время прохождения
- статус

Дефект

Дефект или баг-репорт - это документ, описывающий ситуацию или последовательность действий, приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.



Артибуты дефекта

Наименование	Содержание	Обязательность
Инициатор	ФИО и контактные данные сотрудника, обнаружившего ошибку	**Да**
Дата и время	Дата и время выявления ошибки	**Да**
Система	ИС в которой проявляется ошибка	**Да**
Релиз	Номер релиза, при тестировании которого выявлена ошибка.	**Да**
Номер задачи	Ссылка на задачу, которая тестируется (для регресса не заполняется)	**Нет**
Приоритет	Приоритет ошибки в соответствии с правилами	**Да**
Срочность	Срочность ошибки в соответствии с правилами	**Да**
Окружение (Environment New)	Тестовая среда, на которой смоделирована ошибка из актуального списка ТС. Пример:	**Да**
Этап тестирования	Указать этап тестирования	**Да**
Логин	Логин пользователя, под которым воспроизводится ошибка	**Нет**
Тема	Краткое описание сути ошибки	**Да**
Описание и вложения	Подробное описание ошибки, включающее в себя: входные данные, последовательность действий, приводящих к ошибке, скриншоты, ссылки на документы с требованиями, которым противоречит реализация, логи, протокол тестирования и другая доступная информация для локализации ошибки.	**Да**
Ожидаемое поведение системы	Описание ожидаемого поведения ИС на основании требований	**Да**
Реальное поведение системы	Описание реального поведения системы, противоречащего ожидаемому.	**Да**

Приоритет дефекта

Приоритет ошибки - определяет критичность ошибки с точки зрения функционала системы/задачи.

Принимает следующие значения:

Ошибки (функциональные):
Блокирующая (Blocker) – Неработоспособна ИС или неправильно работающая ключевая функция ИС.
Критическая (Critical) – Неправильно работающая ключевая функция задачи.
Серьезная (Major) - Неправильно работающая ключевая функция задачи, но есть возможность для работы с рассматриваемой функцией, используя альтернативные пути штатными функциями системы через интерфейс пользователя.
Незначительная (Minor) - ошибка, не нарушающая ключевые функции задачи или ИС, позволяющая выполнять соответствующие бизнес-процессы.
Незначительная (Trivial) – опечатка или недочет в работе приложения, не оказывающий влияния на работоспособность ИС и всех ее бизнес-процессов.

Срочность дефекта

Приоритет	Этап Тестирования				
	ФТ первая неделя	ФТ после первой недели	ПТ	BAT	Регресс 1
Blocker	Срочная	Срочная	Срочная	Срочная	Срочная
Critical	Высокая	Срочная	Срочная	Срочная	Срочная
Major	Высокая	Высокая	Высокая	Высокая	Срочная
Minor	Низкая	Низкая	Низкая	Низкая	Низкая
Trivial	Низкая	Низкая	Низкая	Низкая	Низкая

Жизненный цикл дефекта

Жизненный цикл бага (bug workflow) – последовательность этапов, которые проходит баг на своём пути с момента его создания до окончательного закрытия. Для лучшего восприятия изображается в виде схемы с возможными статусами и действиями, которые приводят к смене этих статусов.

Дефекты могут быть найдены на любом этапе ЖЦ ПО.

- Во время анализа требований – неверные, недостающие, противоречивые или неоднозначные требования
- Во время проектирования - сложная архитектура, недостающие процессы
- Во время программирования – отступ от рекомендаций, стандартов и лучших практик кодирования
- Во время тестирования – ошибочные тестовые условия, сценарии тестирования или тестовые данные. Ошибочная реализация или автоматизация тестов

Жизненный цикл дефекта

Отчетность по тестированию

Виды отчетности:

Оперативная отчётность

- Показать текущий статус тестирования
- Показать прогресс по тестированию за отчётный период
- Показать текущее состояние тестируемого ПО
- Сформировать ожидания о сроках завершения тестирования

Итоговая отчетность

- Предоставить результаты проведённых тестов
- Предоставить рекомендации по установке в продуктив
- Предоставить выводы о качестве протестированного ПО
- Помощь в принятии решения об установке в продуктив

Оперативная отчетность

- Количество выполненных тестов с разбивкой по задачам/функционалу
- Результат выполнения тестов – passed, failed
- Обнаруженные дефекты
- Статус исправления дефектов
- Количество тестов, доступных для выполнения
- Количество тестов, недоступных для выполнения
- Процент выполненных тестов с разбивкой по задачам/функционалу
- Плановые сроки завершения тестирования

Итоговая отчетность

- ЧТО тестировали?
- КАК тестировали?
- КОГДА тестировали?
- КАКИЕ результаты?
- КАКИЕ проблемы были при тестировании?

Содержание итогового отчета

СОДЕРЖАНИЕ ИТОГОВОГО ОТЧЁТА:

- Предмет тестирования
- Описание тестируемой функциональности
- Ограничения тестирования
- Описание тестового стенда с указанием версий компонент
- Сроки тестирования
- Виды тестирования
- Результаты проведённых тестов
- Описание отклонений от изначального плана тестирования с указанием причин
- Описание отклонений от изначального плана тестирования с указанием причин
- Количество выполненных тестов с разбивкой по итерациям тестирования
- Перечень непроведённых запланированных тестов с указанием причины
- Перечень итоговых failed тестов с указанием причины
- Обнаруженные дефект
- Перечень неустраненных дефектов и рекомендации что с ними делать
- Выводы о качестве ПО на момент завершения тестирования
- Рекомендации для установки
- Риски

Пример оперативной отчетности

Тестирование по Релизу 3.3 Гермес									
Группа тестов	**Всего**	**Не запускались**	**%**	**Пройдены успешно**	**%**	**Пройдены не успешно**	**%**	**Блокированы**	**%**
LET-582 "Получение выписки ПФР"	20	4	*20%*	13	*65%*	3	*15%*	4	*20%*
LET-608 "Развитие телемаркетинга"	29	0	*0%*	23	*79%*	2	*7%*	4	*14%*
LET-619 "Сервис информирования клиентов"	63	37	*59%*	26	*41%*	0	*0%*	0	*0%*
LET- 615 "WEBCASH 2.0"	38	38	*100%*	0	*0%*	0	*0%*	0	*0%*
LET-610 "Обработка претензий"	25	25	*100%*	0	*0%*	0	*0%*	0	*0%*
Мелкие задачи	21	0	*0%*	21	*100%*	0	*0%*	0	*0%*
Регрессионное тестирование-1 итерация	146	6	*4%*	139	*95%*	1	*1%*	0	*0%*
Всего	*342*	*110*	*32%*	*222*	*65%*	*6*	*2%*	*8*	*2%*

Регрессионное тестирование

Регрессионное тестирование–это любой вид тестирования, который преследует цель обнаружение регресса системы.

Регрессионное тестирование –это тестирование, предназначенное для повторной проверки свойств приложения или продукта с целью убедиться в том, что после внесения изменений или добавления новых возможностей приложение по-прежнему работает.

Когда проводить? При любых изменениях:

- Изменения в коде
- Изменение настроек, конфигурационных файлов
- При эксплуатации - накапливается «мусор» (в оперативной памяти, в БД…)

Повторение, но не регресс

- Приёмочное тестирование
- Конфигурационное тестирование
  - тестирование в разных окружениях
  - тестирование с разными настройками

Повторное выполнение тестов

Рациональные поводы:

- Тесты могут снова сработать (перезарядка)
- «Плавающие» дефекты
- Недоверие к предыдущему запуску
- Увеличение пула тестовых данных
- Сравнение характеристик (например, производительность)

«Иррациональные» поводы:

- Это дёшево
- Ошибка появляется снова и снова
- Очень (!) важно, чтобы ошибки не было
- Цель –не тестирование

Повторное выполнение тестов

«Парадокс пестицида» - повторное применение одних тех же тестов, и даже повторное применение одних и тех же методов тестирования, приводит к тому, что в программе остаются дефекты, против которых эти методы неэффективны.

- У нас много регрессионных тестов, нам не хватает времени на то, чтобы их все выполнить, может быть их автоматизировать?
- Может быть. Но сначала скажите, эти тесты часто обнаруживают дефекты?
- Да практически никогда! Поэтому и хотим автоматизировать.
- А новые тесты в этот набор часто добавляются?
- Только по дефектам, которые пользователи нашли.
- То есть пользователи в этих модулях обнаруживают дефекты, а тесты их не обнаруживают?
- Ну-у-у, да...
- А почему?
- ??? (молчание) ... (понимание) Так это же парадокс пестицида!!!!

Полный набор

«Иррациональный» критерий полноты:

- Все тесты, которые когда-либо были придуманы и выполнены

Рациональные критерии полноты:

- Обеспечивающий покрытие требований
- Обеспечивающий покрытие кода
- Обеспечивающий отлов любых мутаций

Распространенные практики

- Простое накопление тестов

результат –«гора мусора»

- Создание тестов для каждого дефекта, запроса на изменение, патча

результат –«лоскутное одеяло»

Как с этим бороться

- Продумывать архитектуру тестового набора
- Минимизировать тестовый набор
  –выбрасывать лишние тесты
  –рефакторинг тестов
- Делать разные наборы тестов в соответствии с разными критериями полноты

Минимизация набора сценариев

Лишний тест –не увеличивающий покрытие по какому-либо критерию

Рефакторинг тестов:

- похожие тесты можно «склеить» (пример: одинаковые действия, но разные проверки)
- крупные тесты можно «раздробить», лишние куски выкинуть
- мелкие тесты можно «склеить»

Разные наборы тестов

- Тесты для изменившихся частей
- Тесты для частей, зависящих от изменившихся частей
- Тесты для критичной функциональности
- Тесты для основной функциональности
- Тесты определённого типа
- Наборы с иными заданными характеристиками

Регрессионное тестирование

- **Недавнее** (Recent)
- **Основное** (Core)
- **Рисковое** (Risky)
- **Чувствительное к конфигурации** (Configuration sensitive)
- **Исправленное** (Repaired)
- **Хроническое** (Chronic)

Регрессионное тестирование

Недавнее

Что было недавно добавлено в приложение? Недавние изменения –от очевидных до едва заметных –возможная причина появления дефектов. Не забудьте проанализировать запросы на изменение требований, сообщения об ошибках, изменения модели данных и относящуюся к делу внутреннюю переписку.

Основное

Определите области приложения, которые обязаны работать «в любом случае», и вы получите ключевые функциональные возможности. Вспомните значения слова «регресс»: движение назад, упадок, ухудшение. Это то, что должно предотвращать регрессионное тестирование.

Регрессионное тестирование

Рисковое

Вспомнить области повышенного риска, имеющиеся в приложении –будь то новые функциональные возможности или старые.

Чувствительное к конфигурации

Код, который зависит от параметров окружения, т.е. чувствительный к конфигурации, более уязвим.

Исправленное

Некоторые функции никак не удается реализовать или поправить с первой попытки, их постоянно сопровождает шлейф дефектов, для устранения которых приходится выпускать несколько внутренних релизов.

Хроническое

Если что-то может сломаться, оно обязательно сломается. Нет ли у продукта областей, в которых часто возникают проблемы?

Управление рисками

- *идентификация риска:* первоначальный мозговой штурм по выявлению риска, последующая сортировка и определение какого-либо механизма для обеспечения постоянного действия данного процесса.
- *анализ воздействия риска:* количественная оценка каждого риска в терминах вероятности его наступления и потенциального ущерба.

Управление рисками

- *планирование реагирования на риски:* что вы собираетесь делать, если и когда данный риск наступит.
- *ослабление риска:* меры, которые должны быть приняты предварительно, чтобы обеспечить возможность и эффективность проведения запланированных действий, если они потребуются.
- *мониторинг и управление рисками:* отслеживание рисков, выделенных в качестве объектов управления, выявление материализации рисков.

Управление рисками

Пример:

Риск -выход из строя разработческого сервера вследствие стихийных бедствий.

Вероятность проявления – низкая

Обоснование: работы ведутся в средней полосе России. Это не сейсмоопасный район, так же по близости нет крупных водоемов, подверженных штормам и цунами и т.п.

Степень влияния на результат – низкая. Обоснование: офис компании находится в железобетонном бункере на глубине 50 метров под землей.

Вывод -можем проигнорировать этот риск.

Действия: нет необходимости в проведении минимизирующих мероприятий.

Управление рисками

Пример:

Риск -не соответствие реализации требованиям.

Вероятность – средняя

Обоснование: тестирование ведется без подробной спецификации продукта, такая же ситуация и у разработчиков. Функционал разрабатывается со слов бизнес аналитика.

Степень влияния на результат – высокая

Обоснование: выявить не соответствие может только заказчик, что повлечет необходимость доработки или переделки готовой части продукта.

Вывод -при средней вероятности возникновения расхождений между спецификацией и реализацией степень влияния данного факта очень высока, что безусловно выльется в дополнительные расходы по доработке. Действия: поставить менеджера в известность.

Где есть неопределенность, там появляется риск

1. *Требования к системе:* Что именно должна делать система?
2. *Обеспечение стандартов взаимодействия:* Как будет система взаимодействовать с людьми-операторами и другими системами того же уровня?
3. *Влияние изменяющейся среды:* Как во время разработки будут изменяться потребности и цели?
4. *Ресурсы:* Какие ключевые навыки и знания исполнителей возможно будет (при необходимости) привлечь по мере продвижения работы над проектом?
5. *Управление:* Хватит ли у руководства таланта, чтобы создать эффективные команды, поддерживать боевой дух, обеспечивать низкую текучесть кадров и координировать сложные комплексы взаимосвязанных задач?

Где есть неопределенность, там появляется риск.

Где есть неопределенность, там появляется риск

6. Сеть поставок: Будут ли другие участники проекта действовать так, как ожидалось?

7. Политика: Каков может быть результат использования политической силы для навязывания ограничений, несовместимых с успехом проекта?

8. Конфликты: Как различные участники проекта найдут компромисс между своими, зачастую несовместимыми, целями?

9. Инновации: Как уникальные для данного проекта технологии и методы влияют на возможный результат?

10. Масштаб: Как повлияет на осуществление проекта увеличение масштаба работ, если раньше у разработчика не было соответствующего опыта?

Top 10 Рисков

1. Дефицит специалистов.
2. Нереалистичные сроки и бюджет.
3. Реализация несоответствующей функциональности.
4. Разработка неправильного пользовательского интерфейса.
5. “Золотая сервировка”, перфекционизм, ненужная оптимизация и оттачивание деталей.
6. Непрекращающийся поток изменений.
7. Нехватка информации о внешних компонентах, определяющих окружение системы или вовлеченных в интеграцию.
8. Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами.
9. Недостаточная производительность получаемой системы.
10. “Разрыв” в квалификации специалистов разных областей знаний.

Барри Боэмом [Boehm, 1988]

Стандарты качества и тестирования

- ISO 9000 - серия международных стандартов, содержащих термины и определения, основные принципы менеджмента качества, требования к системе менеджмента качества организаций и предприятий, а также руководство по достижению устойчивого успеха
- ISO/IEC 12207 – Процессы жизненного цикла программных средств
- IEEE 829 Standard for Software and System Test Documentation
- IEEE 1008 Standard for Software Unit Testing
- BS 7925-1 Glossary of Software Testing Terms
- BS 7925-2 Standard for Software Component Testing
- ISO 29119 Software Testing