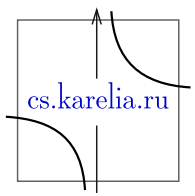
Петрозаводский государственный университет
Институт Математики и информационных технологий
Кафедра Информатики и математического обеспечения

Тестирование. Процесс, уровни, типы.

Лектор:

к.т.н., доцент Богоявленский Ю. А.
ybgv@cs.karelia.ru



Разделы

- Основы тестирования
- Процесс тестирования
- Тестирование в течение жизненного цикла разработки ПО
- Уровни тестирования
- Типы тестирования

Основы тестирования. Содержание

- Что такое тестирование?
  - Основные цели тестирования.
  - Тестирование и отладка.
- Почему тестирование необходимо?
  - Вклад тестирования в успех.
  - Обеспечение качества и тестирование.
  - Ошибки, дефекты и отказы.
  - Дефект, первопричина и эффект.
- Семь принципов тестирования.

Что такое тестирование?

Системы с программным обеспечением являются неотъемлемой частью нашей жизни, от бизнес-приложений до потребительских товаров. Многие люди имели опыт использования программного обеспечения, которое не работало так, как ожидалось.

Программное обеспечение, которое не работает корректно, может привести ко многим проблемам, включая потерю денег, времени или деловой репутации, и стать причиной травмы или смерти. Тестирование программного обеспечения – это способ оценить качество программного обеспечения и снизить риск отказа программного обеспечения.

Основные цели тестирования

Для любого проекта цели тестирования могут включать:

- Оценку рабочих продуктов, таких как требования, пользовательские истории, проектирование и код.
- Проверку, все ли указанные требования выполнены.
- Проверку, завершен ли объект тестирования и работает, как ожидают пользователи и заинтересованные лица.
- Создание уверенности в уровне качества объекта тестирования.
- Предотвращение дефектов.
- Обнаружение отказов и дефектов.

Основные цели тестирования

- Предоставление заинтересованным лицам достаточной информации, позволяющей им принять обоснованные решения, особенно в отношении уровня качества объекта тестирования.
- Снижение уровня риска ненадлежащего качества программного обеспечения.
- Соблюдение договорных, правовых или нормативных требований, или стандартов и/или проверка соответствия объекта тестирования таким требованиям и стандартам.

Основные цели тестирования

Цели тестирования могут варьироваться в зависимости от контекста тестируемого компонента или системы, уровня тестирования и модели жизненного цикла разработки программного обеспечения. Эти различия могут включать:

- При компонентном тестировании цель может заключаться в том, чтобы найти как можно больше сбоев, чтобы выявить и устранить основные дефекты на ранних стадиях.
- При приемочном тестировании цель может заключаться в том, чтобы подтвердить, что система работает, как ожидалось, и удовлетворяет требованиям.

Тестирование и отладка

Стоит различать отладку и тестирование. Выполнение тестов может показать сбои, вызванные дефектами в программном обеспечении. Отладка – это деятельность разработки для нахождения, анализа и исправления таких дефектов. Последующее подтверждающее тестирование проверяет, устранены ли исправленные дефекты.

Почему тестирование необходимо?

Тщательное тестирование компонентов и систем, а также связанной с ними документации может помочь снизить риск отказов во время работы. Когда дефекты обнаружены и исправлены, это способствует качеству компонентов или систем. Также может потребоваться тестирование программного обеспечения на соответствие договорным или правовым требованиям, а также отраслевым стандартам.

Вклад тестирования в успех

На протяжении всей истории компьютеризации довольно часто поставка программного обеспечения и систем в эксплуатацию из-за наличия дефектов приводила к сбоям или иным образом не отвечала потребностям заинтересованных лиц.

Однако, использование соответствующих методов тестирования может снизить частоту таких проблемных поставок, если эти методы применяются с соответствующим уровнем опыта тестирования, на соответствующих уровнях тестирования и соответствующих этапах жизненного цикла разработки программного обеспечения.

Вклад тестирования в успех

- Наличие тестировщиков, участвующих в рецензировании требований или уточнении пользовательских историй, может обнаружить дефекты в этих рабочих продуктах. Выявление и устранение дефектов требований снижает риск разработки неправильной или нетестируемой функциональности.
- Наличие тесного сотрудничества тестировщиков с проектировщиками системы во время проектирования системы может повысить понимание архитектуры системы и способов тестирования каждой стороной. Такое более глубокое понимание снижает риск возникновения фундаментальных дефектов проектирования и даёт возможность определить тесты на ранней стадии.

Вклад тестирования в успех

- Наличие тесного взаимодействия тестировщиков с программистами во время разработки кода может улучшить понимание кода и способов его тестирования каждой стороной. Такое более глубокое понимание снижает риск возникновения дефектов в коде и тестах.
- Наличие тестировщиков для валидации и верификации программного обеспечения до релиза может обнаружить сбои, которые в противном случае могли быть пропущены, и поддержать процесс устранения дефектов, вызвавших отказы (т.е. отладку). Это повышает вероятность того, что программное обеспечение удовлетворяет потребностям заинтересованных лиц и соответствует требованиям.

Обеспечение качества и тестирование

- Обеспечение качества, как правило, сосредоточено на соблюдении соответствующих процессов для обеспечения уверенности, что будут достигнуты соответствующие уровни качества.
- Работы по тестированию являются частью общего процесса разработки и сопровождения программного обеспечения.

Ошибки, дефекты и отказы

- Человек может совершить ошибку, которая может привести к появлению дефекта в коде программы или в каком-либо другом сопутствующем продукте.
- Ошибка, приводящая к появлению дефекта в одном рабочем продукте, может вызвать ошибку, приводящую к появлению дефекта в связанном рабочем продукте.
- Если дефект в коде выполняется, это может привести к отказу.

Ошибки, дефекты и отказы

Ошибки могут возникать по многим причинам. Например,

- Нехватка времени.
- Человек может ошибаться.
- Неопытные или недостаточно квалифицированные участники проекта.
- Недопонимание между участниками проекта, включая недопонимание требований и проектирования.
- Сложность кода, проектирования, архитектуры, основной проблемы, которую надо решить, и или используемых технологий.

Ошибки, дефекты и отказы

- Непонимание внутрисистемных и межсистемных интерфейсов, особенно когда таких внутрисистемных и межсистемных взаимодействий много.
- Новые, незнакомые технологии.

Кроме отказов, вызванных дефектами в коде, отказы также могут быть вызваны условиями окружающей среды.

- Ошибочные негативные тесты – это тесты, которые должны были обнаружить, но не обнаружили дефекты.
- Ошибочные позитивные тесты – это тесты, которые обнаружили дефекты, не являющиеся реальными дефектами.

Дефект, первопричина и эффект

Первопричины дефектов – это самые ранние действия или условия, которые способствовали созданию дефектов. Дефекты можно проанализировать для поиска первопричины, чтобы уменьшить возникновение подобных дефектов в будущем. Обращая внимание на наиболее существенные первопричины, анализ первопричин может привести к улучшению процессов, которые пре- дотвратят появление значительного числа будущих дефектов.

Семь принципов тестирования

За последние пятьдесят лет был предложен ряд принципов тестирования, которые являются общим руководством для тестирования в целом.

- Тестирование демонстрирует наличие дефектов, а не их отсутствие.
- Исчерпывающее тестирование недостижимо.
- Раннее тестирование сохраняет время и деньги.
- Кластеризация дефектов.
- Парадокс пестицида.
- Тестирование зависит от контекста.
- Заблуждение об отсутствии ошибок.

Процесс тестирования. Содержание

Введение

В проектах по разработке программного обеспечения (ПО) помимо основной задачи по реализации заявленной функциональности существует не менее важная задача по обеспечению качества ПО. Одним из устоявшихся способов контроля качества является тестирование. Тестирование ПО — проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.

Нет универсального процесса тестирования программного обеспечения, но есть общие наборы тестовых активностей, без которых тестирование вряд ли достигнет поставленных целей. Эти наборы тестовых активностей и есть процесс тестирования. Правильный соответствующий процесс тестирования программного обеспечения в любой конкретной ситуации зависит от многих факторов.

Процесс тестирования в контексте

Контекстные факторы, которые влияют на процесс тестирования:

- используемые модель жизненного цикла разработки программного обеспечения и проектные методологии
- рассматриваемые уровни и типы тестирования
- продуктовые и проектные риски
- предметная область
- ограничения, включая, в частности:
  - бюджеты и ресурсы
  - сроки
  - сложность
  - договорные и нормативные требования
- организационные политики и практики
- необходимые внутренние и внешние стандарты

Активности и задачи в тестировании

Процесс тестирования состоит из следующих основных групп активностей:

- планирование тестирования
- мониторинг и контроль тестирования
- анализ тестирования
- проектирование тестов
- реализация тестов
- выполнение тестов
- завершение тестирования

Каждая группа активностей состоит из отдельных активностей. Каждая активность в свою очередь может состоять из нескольких отдельных задач, которые будут варьироваться от одного проекта или релиза к другому.

Активности и задачи в тестировании

Планирование тестирования

Планирование тестирования состоит из активностей, которые определяют цели тестирования и подход к достижению целей тестирования с ограничениями, налагаемыми контекстом (например, определение подходящих методов тестирования и задач, а также формирование графика тестирования для соблюдения крайнего срока). Планы тестирования могут быть пересмотрены на основе обратной связи от мониторинга и контроля.

Мониторинг и контроль тестирования

Мониторинг тетирования предполагает непрерывное сравнение фактического хода работы с планом тестирования, используя любые метрики мониторинга тестирования, определённые в плане тестирования. Контроль тестирования подразумевает принятие мер, необходимых для достижения целей плана тестирования.

Активности и задачи в тестировании

Анализ тестирования

Базис тестирования анализируют для определения тестируемых функций и установление соответствующих тестовых условий. Другими словами, анализ тестирования решает “что тестировать” с точки зрения измеримых критериев покрытия.

Анализ тестирования состоит из следующих основных активностей:

- анализ базиса тестирования, применимого к рассматриваемому уровню тестирования
- оценка базиса тестирования и элементов тестирования для выявления дефектов различных типов
- определение свойств и совокупность свойств для тестирования
- установление и приоритизация тестовых условий для каждого свойства на основе анализа базиса тестирования
- обеспечение двунаправленной трассируемости между каждым элементом базиса тестирования и соответствующими тестовыми условиями

Активности и задачи в тестировании

Проектирование тестов

Во время проектирования тестов тестовые условия воплощаются в высокоуровневые тестовые сценарии, наборы высокоуровневых тестовых сценариев и другое тестовое обеспечение. Проектирование тестов отвечает на вопрос “как тестировать?”.

Проектирование тестов состоит из следующих основных активностей:

- проектирования и приоритизации тестовых сценариев и наборов тестовых сценариев
- определения необходимых тестовых данных для поддержки тестовых условий и тестовых сценариев
- проектирования тестового окружения и определения необходимой инфраструктуры и инструментов
- отражения двунаправленной трассируемости между базисом тестирования, тестовыми условиями, тестовыми сценариями и процедурами тестирования

Активности и задачи в тестировании

Реализация тестов

Во время реализации тестов создается и/или подготавливается необходимое тестовое обеспечение для выполнения тестов, включая упорядочивание тестовых сценариев в процедурах тестирования.

Таким образом, проектирование тестов отвечает на вопрос “как проверить?”, в то время как реализация тестов отвечает на вопрос: “у нас теперь есть все для запуска тестов?”.

Реализация тестов — это активность, во время которой процедуры или сценарии тестирования выстраиваются в определенном порядке, чтобы облегчить выполнение тестов.

Активности и задачи в тестировании

Реализация тестов состоит из следующих основных активностей:

- разработка и расстановка приоритетов процедур тестирования и, возможно, создание автоматизированных сценариев тестирования
- создание наборов тестов из процедур тестирования и автоматизированных сценариев тестирования
- организация наборов тестов с расписанием выполнения тестов таким образом, чтобы тесты выполнялись эффективно
- построение тестового окружения и проверка правильности настройки всего необходимого
- подготовка тестовых данных и правильная загрузка их в тестовое окружение
- проверка и обновление двунаправленной трассируемости между базисом тестирования, тестовыми условиями, тестовыми сценариями, процедурами тестирования и наборами тестов

Активности и задачи в тестировании

Выполнение тестов

Во время выполнения тестов, наборы тестов запускаются в соответствии с расписанием выполнения тестов.

Выполнение тестов состоит из следующих основных активностей:

- запись идентификаторов и версий элементов тестирования или объекта тестирования, инструментов тестирования, и тестового обеспечения
- выполнение тестов вручную или с помощью инструментов выполнения тестов
- сравнение фактических и ожидаемых результатов
- анализ отклонений для установления их вероятных причин
- составление отчетов о дефектах на основе наблюдаемых отказов
- протоколирование результатов выполнения тестов
- повторение тестовых действий, результаты которых привели к каждому из отклонений, либо в рамках запланированного тестирования

Активности и задачи в тестировании

Завершение тестирования

Активности по завершению тестирования собирают данные из выполненных активностей тестирования для обобщения опыта, тестового обеспечения и любой другой соответствующей информации.

Активности по завершению тестирования происходят например по завершению выпуска релиза программного обеспечения системы, завершению (отмене) проекта тестирования, окончанию итерации проекта с гибкой методологией (например, как часть итогового митинга), завершению уровня тестирования, или завершению сопровождения релиза.

Рабочие продукты тестирования

Рабочие продукты планирования тестирования

Рабочие продукты планирования тестирования — это один или несколько планов тестирования. План тестирования содержит информацию о базисе тестирования, с которым другие рабочие продукты тестирования будут связаны через информацию о трассируемости, а также критерии выхода, которые будут использоваться во время мониторинга и контроля тестирования.

Рабочие продукты мониторинга и контроля тестирования

Рабочие продукты мониторинга и контроля тестирования — это обычно различные типы отчетов о тестировании, включая отчеты о ходе тестирования и сводные отчеты о тестировании. Все отчеты о тестировании должны обеспечить соответствующей аудитории подробные сведения о ходе тестирования на дату отчета, включая обобщение результатов выполнения тестирования после их получения.

Рабочие продукты тестирования

Рабочие продукты проектирования тестов

Результатом проектирования тестов являются тестовые сценарии и наборы тестовых сценариев для выполнения тестовых условий, определенных в анализе тестирования. В идеале, каждый тестовый сценарий двунаправленно прослеживается к тестовым условиям, которые он покрывает.

Рабочие продукты реализации тестов

Рабочие продукты реализации тестов включают:

- процедуры тестирования и последовательности этих процедур тестирования
- наборы тестов
- расписания выполнения тестов

Рабочие продукты тестирования

Рабочие продукты выполнения тестов

Рабочие продукты выполнения тестов включают:

- документацию о состоянии отдельных тестовых сценариев или процедур тестирования
- отчеты о дефектах
- документацию о том, какие элементы теста, объекты тестирования, инструменты тестирования, и тестовое обеспечение были задействованы в тестировании

Рабочие продукты завершения тестирования

Рабочие продукты завершения тестирования состоят из сводных отчетов тестирования, мероприятий по улучшению последующих проектов или итераций, запросов на изменение или набора задач продукта, и окончательного тестового обеспечения.

Тестирование в течение жизненного цикла разработки ПО. Содержание

- Введение
- Разработка и тестирование программного обеспечения
- Последовательные модели разработки
- Итерационные и инкрементальные модели разработки
- Выбор модели жизненного цикла разработки в зависимости от ситуации

Введение

Модель жизненного цикла разработки программного обеспечения описывает виды активностей, выполняемых на каждом этапе процесса разработки программного обеспечения, а также то, как эти активности связаны друг с другом логически и хронологически.

Существует несколько различных моделей жизненного цикла разработки программного обеспечения, каждый из которых требует различных подходов к тестированию.

Разработка и тестирование программного обеспечения

Важной частью работы тестировщика является знание распространенных моделей жизненного цикла разработки ПО, на основании которых происходит выбор соответствующих активностей по тестированию. Независимо от того, какая модель жизненного цикла разработки выбрана, активности тестирования следует начинать на ранних стадиях жизненного цикла, придерживаясь принципа раннего тестирования.

Эта программа обучения классифицирует распространенные модели жизненного цикла разработки следующим образом:

- Последовательные модели разработки.
- Итерационные и инкрементальные модели разработки.

Разработка и тестирование программного обеспечения

Для любой модели жизненного цикла разработки существуют несколько показателей качественного тестирования:

- Для каждой активности разработки существует соответствующая активность тестирования.
- У каждого уровня тестирования есть цели, характерные для данного уровня.
- Анализ и проектирование тестов для определенного уровня тестирования начинаются на стадии соответствующих активностей разработки.
- Тестировщики должны участвовать в обсуждениях для определения и уточнения требований и дизайна, а также вовлекаться в рецензирование рабочих продуктов(например, требований, дизайна, пользовательских историй и т.п.), как только будут доступны первые их черновые версии.

Последовательные модели разработки

Последовательная модель разработки описывает процесс разработки программного обеспечения в качестве линейного, последовательного потока активностей. Это означает, что любую фазу процесса разработки следует начинать после того, как предыдущая фаза завершена. Теоретически фазы не пересекаются друг с другом, но на практике бывает полезным получать раннюю обратную связь от следующей фазы.

Последовательные модели разработки

Водопад

При работе по модели «Водопад», активности разработки (например, анализ требований, дизайн, написание кода, тестирование) выполняются каждая по окончании предыдущей. В данной модели активности тестирования осуществляются только после того, как все активности разработки выполнены.

Последовательные модели разработки

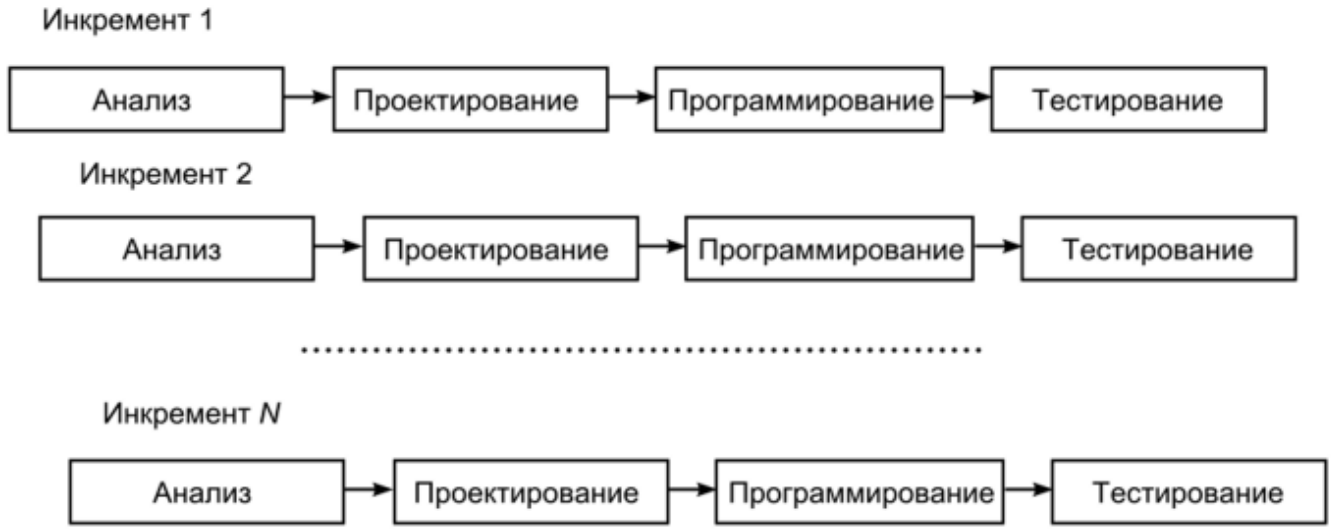
V-модель

В отличие от модели «Водопад», V-модель интегрирует тестовый процесс на протяжении процесса разработки, реализуя принцип раннего тестирования. Более того, V-модель включает уровни тестирования, относящиеся к соответствующей фазе разработки, что также поддерживает раннее тестирование. В этой модели выполнение тестов, связанных с каждым уровнем тестирования, продолжается последовательно, но в некоторых случаях происходят пересечения.

Результатом работы с применением последовательных моделей разработки является ПО, которое содержит полный набор функциональности, но для его поставки заказчику и пользователям зачастую уходят месяцы и даже годы разработки.

Итерационные и инкрементальные модели разработки

Инкрементальная разработка подразумевает определение требований, дизайн, сборку и тестирование системы по частям. Это приводит к тому, что функциональность ПО растет постепенно. Размер таких функциональных приращений разнится в большую или меньшую сторону в зависимости от выбранных методов работы. Приращение функциональности может включать всего одно изменение в пользовательском интерфейсе или новую опцию запроса.



Итерационные и инкрементальные модели разработки

Итеративная разработка используется в случае, когда разрабатываемый функционал должен быть определен, спроектирован, построен и протестирован в течение нескольких этапов, зачастую с фиксированной продолжительностью каждого из них. Итерации могут включать в себя как набор изменений функционала, реализованного в предыдущих итерациях, так и изменения всего разрабатываемого продукта в целом. На выходе каждой итерации получается рабочий программный продукт, который, по сути, является растущим набором общего функционала, и продолжается это до тех пор, пока не будет выпущена финальная версия этого продукта или разработка не будет остановлена.

Итерационные и инкрементальные модели разработки

В качестве примера можно привести следующие методологии:

- RUP(RationalUnifiedProcess): методология: каждая итерация может быть относительно долгой (например, от двух до трех месяцев), а приращения функциональности соответственно большими, к примеру, две или три группы связанных функциональностей.
- Скрам: каждая итерация может быть относительно короткой (например, часы, дни или несколько недель), а приращения функциональности соответственно небольшими, как, например, несколько улучшений и/или две или три новые функции.

Итерационные и инкрементальные модели разработки

В качестве примера можно привести следующие методологии:

- Канбан: осуществляется с использованием или без использования итераций фиксированной продолжительности, по завершению которых выпускается либо единственная доработка или функциональность, либо группа функциональностей, объединенных вместе.
- Спиральная модель разработки (прототипирование): подразумевает создание экспериментальных приращений, некоторые из которых могут быть значительно переработаны или даже отвергнуты в последующей работе по разработке.

Итерационные и инкрементальные модели разработки

Компоненты или системы, разрабатываемые с использованием этих моделей, часто подразумевают пересечение и повторение уровней тестирования на протяжении процесса разработки.

В идеале, каждая функциональность тестируется на нескольких уровнях тестирования по мере приближения к выпуску продукта. В некоторых случаях, команды используют непрерывную поставку или непрерывное развертывание, которые подразумевают существенную автоматизацию многих уровней тестирования в процессе поставки.

Предпринимаемые усилия по разработке ПО с использованием этих методов, также включают концепцию самоорганизующихся команд, что может изменить как организацию работ по тестированию, так и взаимоотношения между тестировщиками и разработчиками.

Вывод

Применение этих методов в итоге формирует готовый продукт, который может быть выпущен конечным пользователям на основе постепенного приращения функциональности или более традиционным способом изготовления основного релиза. Независимо от того, поставлялись ли приращения конечным пользователям, важность регрессионного тестирования растет с ростом разрабатываемой системы.

В отличие от результатов работы с применением последовательных моделей разработки, результатом работы с применением итеративных и инкрементальных моделей является ПО, готовое к использованию через недели или даже дни, но до поставки полного набора требований продукта могут уйти многие месяцы и даже годы.

Выбор модели жизненного цикла разработки в зависимости от ситуации

Модели жизненного цикла разработки программного обеспечения должны выбираться и быть адаптированы к контексту характеристик проекта и продукта. Соответствующая модель жизненного цикла разработки должна быть выбрана и адаптирована на основе цели проекта, типа разрабатываемого продукта, бизнес-приоритетов (например, срок вывода продукта на рынок), и выявленных рисков продукта и проекта.

Например, небольшую внутреннюю административную систему следует разрабатывать и тестировать иначе, нежели критически важную для безопасности систему, например, систему управления тормозами автомобиля.

Как другой пример, в некоторых случаях организационные и культурные аспекты могут осложнять общение между членами команды, что может препятствовать итеративной разработке.

Выбор модели жизненного цикла разработки в зависимости от ситуации

В зависимости от контекста проекта может потребоваться объединение или реорганизация уровней тестирования и/или тестовых активностей.

Например, для интеграции готового коммерческого решения в более крупную систему покупатель этого решения может выполнить тестирование возможности взаимодействия на уровне системного интеграционного тестирования (например, интеграцию с инфраструктурой и другими системами) и на уровне приемочного тестирования (функциональное и/или нефункциональное, наряду с пользовательским приемочным и эксплуатационным приемочным тестированием).

Выбор модели жизненного цикла разработки в зависимости от ситуации

Кроме этого, сами модели жизненного цикла разработки могут сочетаться одна с другой.

Например, V-модель может быть использована для разработки и тестирования систем серверного уровня и их интеграции, тогда как методология гибкой разработки может быть использована для разработки и тестирования пользовательского интерфейса (ПИ) и функциональности. На ранних стадиях проекта может быть использовано прототипирование, с переходом на инкрементальную модель разработки после завершения экспериментальной фазы.

Выбор модели жизненного цикла разработки в зависимости от ситуации

Системы с общим названием «Интернет вещей», которые состоят из множества различных объектов, таких как устройства, продукты и сервисы, часто разрабатываются с применением разных моделей жизненного цикла разработки для каждого объекта.

Это приводит к определенной сложности для разработки версий таких систем. Кроме того, сильный акцент делается уже на поздние фазы жизненного цикла, после внедрения (например, фазы использования, обновления и вывода из эксплуатации).

Уровни тестирования. Содержание

- Уровни тестирования:
  - Компонентное тестирование.
  - Интеграционное тестирование.
  - Системное тестирование.
  - Приемочное тестирование.
- Признаки уровней тестирования:
  - Конкретные цели.
  - Базис тестирования.
  - Объект тестирования.
  - Типичные дефекты и отказы.
  - Специфические подходы и зоны ответственности.

Компонентное тестирование

Компонентное (модульное) тестирование фокусируется на компонентах, которые могут быть проверены отдельно.

Цели компонентного тестирования включают:

- Снижение риска
- Проверка, соответствует ли функциональное поведение компонентов установленным требованиям
- Укрепление уверенности в качестве компонента
- Обнаружение дефектов в компоненте
- Предотвращение пропуска дефектов на более высокие уровни тестирования

Примеры рабочих продуктов, которые используются в качестве **базиса тестирования** включают:

- Детальный дизайн
- Код
- Модель данных
- Спецификации компонента

Типичными **объектами** для компонентного тестирования являются:

- Компоненты, модули
- Код и структуры данных
- Классы
- Модули БД

Примеры типичных **дефектов и отказов** при компонентном тестировании включают:

- Неправильная работа функциональности (например, не так, как описано в спецификации)
- Проблемы с потоками данных
- Неправильные код и логика

Дефекты исправляются после их обнаружения, в соответствующей системе управления дефектами. При оформлении отчетов о дефектах, создается важная информация для анализа первопричин дефектов и улучшения процесса.

Специфические подходы и зоны ответственности

- Компонентное тестирование обычно выполняется разработчиком. Разработчики могут чередовать разработку компонентов с обнаружением и устранением дефектов.
- Разработчики часто пишут и выполняют тесты после написания кода для компонента. Тем не менее, написание автоматизированных тестовых сценариев для компонентов может предшествовать написанию кода приложения.

Интеграционное тестирование

Интеграционное тестирование фокусируется на взаимодействии между компонентами или системами. **Цели** интеграционного тестирования включают:

- Снижение риска
- Проверка, соответствует ли функциональное и нефункциональное поведение интерфейсов установленным проектным требованиям
- Повышение уверенности в качестве интерфейсов
- Обнаружение дефектов (в самих интерфейсах или внутри компонентов)
- Предотвращение пропуска дефектов на более высокие уровни тестирования

Как и при компонентном тестировании, в некоторых случаях автоматизированные тесты поддерживают уверенность в том, что изменения не повредили существующие интерфейсы, компоненты или системы.

Есть два разных уровня интеграционного тестирования, которые могут выполняться на тестовых объектах:

- **Компонентное интеграционное тестирование** фокусируется на взаимодействиях и интерфейсах между интегрированными компонентами.
- **Системное интеграционное тестирование** фокусируется на взаимодействиях и интерфейсах между системами, пакетами и микросервисами. Системное интеграционное тестирование также может охватывать взаимодействия и интерфейсы, предоставляемые сторонними организациями (например, веб-сервисы).

Примеры рабочих продуктов, которые могут использоваться в качестве **базиса тестирования** для интеграционного тестирования, включают:

- Дизайн продукта и системы
- Диаграммы последовательности
- Спецификации интерфейса и протокола связи
- Сценарии использования системы
- Архитектура на уровне компонентов или системы
- Рабочие процессы
- Спецификации, описывающие внешние интерфейсы

Типичными **объектами тестирования** при интеграционном тестировании являются:

- Подсистемы
- Базы данных
- Инфраструктура
- Интерфейсы
- Программные интерфейсы приложения (API)
- Микросервисы

Примеры **типичных дефектов и отказов** при интеграционном тестировании включают:

- Некорректные (отсутствующие) данные, неверная кодировка данных
- Неверная последовательность или временные характеристики обращения к интерфейсам
- Несовместимость интерфейсов
- Сбои связи между компонентами
- Необработанные или неправильно обработанные сбои связи между компонентами
- Неправильные предположения о назначении, единицах или границах данных, передаваемых между компонентами

Специфические подходы и зоны ответственности

- Компонентные интеграционные и системные интеграционные тесты должны быть сосредоточены на интеграции. Например, если интегрировать модуль(систему) A с модулем(системой) B, тесты должны быть сосредоточены на связи между модулями(системами). А функциональности отдельных модулей(систем) должны быть проверены во время компонентного(системного) тестирования.
- Компонентное интеграционное тестирование является обязанностью разработчиков. А системное интеграционного тестирование – обязанность тестировщиков. В идеале тестировщики, должны понимать архитектуру системы и влиять на интеграционное планирование.

Системное тестирование

Системное тестирование фокусируется на поведении и возможностях целой системы или продукта, часто учитывая сквозные задачи, которые может выполнять система, и нефункциональное поведение, которое она демонстрирует при выполнении этих задач.

Цели системного тестирования включают:

- Снижение риска
- Проверка, соответствует ли функциональное и нефункциональное поведение системы установленным проектным требованиям, дизайну и спецификациям
- Проверка, что система реализована полностью и будет работать, как ожидалось
- Повышение уверенности в качестве системы в целом
- Обнаружение дефектов
- Предотвращение попадания дефектов на более высокие уровни тестирования или в среду эксплуатации

Для определенных систем целью может быть проверка качества данных. Системное тестирование часто дает информацию, которая

Примеры рабочих продуктов в качестве **базиса тестирования** для системного тестирования, включают:

- Системные требования и требования к продукту (функциональные и нефункциональные)
- Отчеты об анализе рисков
- Сценарии использования
- Бизнес-потребности и пользовательские истории
- Модели поведения системы
- Диаграммы состояний
- Системные и пользовательские руководства

Типичными **объектами тестирования** являются:

- Приложения
- Аппаратные/программные системы
- Тестируемая система
- Операционные системы
- Конфигурация системы и конфигурация данных

Примеры **типичных дефектов и отказов** при системном тестировании включают:

- Некорректные вычисления
- Некорректное/неожиданное функциональное поведение системы
- Некорректное управление и/или передача данных внутри системы
- Невозможность правильно и полностью выполнить функциональные задачи конечными пользователями
- Неспособность системы работать правильно в среде эксплуатации
- Неспособность системы работать так, как описано в системных и пользовательских руководствах

Специфические подходы и зоны ответственности

- Системное тестирование должно быть сосредоточено на общем поведении системы с точки зрения конечных пользователей.
- Системное тестирование часто проводит независимая группа тестировщиков. Ошибки в спецификациях (например, отсутствие пользовательских историй) могут привести к отсутствию понимания или разногласиям насчет ожидаемого поведения системы. Такие ситуации могут вызывать ложно позитивные/негативные результаты тестирования, которые отнимают время и снижают эффективность обнаружения дефектов.

Приемочное тестирование

Приемочное тестирование, как и системное тестирование, обычно фокусируется на поведении и возможностях системы или продукта в целом. **Цели** приемочного тестирования включают:

- Продемонстрировать уверенность в качестве системы в целом
- Проверить, что система завершена и будет работать как ожидалось
- Проверить, соответствует ли функциональное и нефункциональное поведение системы установленным проектным требованиям

Данное тестирование может дать информацию для оценки готовности системы к развертыванию и использованию конечным пользователем.

Типичными формами приемочного тестирования являются:

- Пользовательские:
  - Основная цель: соответствие системы требованиям и ее способности выполнять поставленные задачи с минимальными трудностями, затратами и рисками.
- Эксплуатационные:
  - Основная цель: поддержание работоспособности системы для пользователей в среде эксплуатации или сложных условиях
- Контрактные и нормативные
  - Основная цель: достижение соответствий контрактных или нормативных требований.
- Альфа-тестирование и бета-тестирование
  - Основные цели: использование системы в повседневных условиях для достижения своих целей. Обнаружение дефектов, связанных с условиями и средой эксплуатации, в которых система будет использоваться.

Примеры рабочих продуктов в качестве **базиса тестирования** для любой формы приемочного тестирования, включают:

- Бизнес-процессы
- Пользовательские и бизнес-требования
- Нормативы, юридические контракты и стандарты
- Сценарии использования системы
- Системные требования
- Системная или пользовательская документация
- Процедуры установки программного обеспечения
- Отчеты анализа риска

Объекты тестирования для любой формы тестирования являются:

- Тестируемая система
- Конфигурация системы и конфигурационные данные
- Бизнес-процессы для полностью интегрированной системы
- Операционные и эксплуатационные процессы
- Формы; Отчеты
- Существующие и преобразованные производственные данные

Примеры **типичных дефектов** для любой формы тестирования:

- Системные рабочие процессы не отвечают бизнес-требованиям или требованиям пользователей
- Бизнес-требования некорректно реализованы
- Система не соответствует контрактным и/или нормативным требованиям
- Нефункциональные сбои, такие как уязвимости в системе безопасности, недостаточная производительность при высоких нагрузках или неправильная работа на поддерживаемой платформе

Специфические подходы и зоны ответственности

Приемочное тестирование часто является обязанностью клиентов, бизнес-пользователей, владельцев продуктов или операторов системы, а также других заинтересованных сторон.

В последовательном жизненном цикле разработки приемочное тестирование часто считается **последним** уровнем тестирования, но оно может проводиться и в другое время, например:

- Приемочное тестирование коммерческого готового программного обеспечения может проводиться, когда ПО установлено или интегрировано
- Приемочное тестирование нового функционального улучшения может производиться перед системным тестированием

При итеративной разработке проектные группы могут использовать различные формы приемочного тестирования как во время итерации, так и в конце итерации.

Типы тестирования. Содержание

- Цели тестирования.
- Функциональное тестирование.
- Нефункциональное тестирование.
- Типы и уровни тестирования.
- Тестирование в период сопровождения:

Цели тестирования

Тип тестирования – это совокупность активностей тестирования, направленных на тестирование заданных характеристик системы или ее части, основываясь на конкретных целях.

К таким целям можно отнести следующие:

- Оценку функциональных характеристик качества системы (полнота, корректность, целесообразность).
- Оценку нефункциональных характеристик качества (надежность, продуктивность работы, безопасность, совместимость и удобство использования).
- Оценку правильности, полноты структуры или архитектуры компонента или системы, их соответствие спецификации.
- Оценку влияния изменений, например, подтверждение того, что дефекты были исправлены и поиск непреднамеренных изменений в поведении, вызванных изменениями в программном обеспечении.

Функциональное тестирование

Функциональное тестирование системы включает тесты по оценке функций, которые должна выполнять система. Функциональные требования могут быть описаны в рабочих продуктах (требования, спецификация, бизнес-потребность, пользовательская история, сценарий использования) или в функциональной спецификации, а могут быть вообще не задокументированы.

Функции системы дают ответ на вопрос «что делает система».

Функциональные тесты должны выполняться на всех уровнях тестирования (например, тесты для компонентов системы могут основываться на спецификации компонента), поэтому фокус тестирования различается для каждого уровня (Компонентное тестирование; Интеграционное тестирование; Системное тестирование).

Функциональное тестирование

Покрытие функциональности – это мера, с которой конкретный тип элемента функциональности был охвачен тестами. Уровень покрытия вычисляется в процентах от общего количества элементов (или типов элементов). Например, используя трассируемость тестов и соответствующих требований, можно вычислить процент требований, которые покрыты тестами, и выявить пробелы в покрытии.

Проектирование и выполнение функциональных тестов может потребовать специальных навыков и знаний, относящихся к конкретной области бизнес задачи, решаемой программным обеспечением (например, знания в области программного обеспечения геологического моделирования в нефте- и газодобывающей отрасли) или специфической предметной области (например, программное обеспечение компьютерных игр, которые предоставляют интерактивный развлекательный контент).

Нефункциональное тестирование

Нефункциональное тестирование системы выполняется для оценки таких характеристик системы и программного обеспечения, как удобство использования, производительность или безопасность.

Нефункциональное тестирование – это проверка того, «насколько хорошо работает система».

Нефункциональное тестирование может и, чаще всего, должно выполняться на всех уровнях тестирования, и как можно раньше. Несвоевременное обнаружение нефункциональных дефектов может быть угрозой успеха всего проекта.

Нефункциональное тестирование

Нефункциональное покрытие – это степень, с которой какая-либо нефункциональная характеристика покрыта тестами. Например, используя трассируемость тестов к соответствующим им поддерживаемым устройствам для мобильного приложения, можно вычислить процент поддерживаемых устройств, используемых при тестировании совместимости, тем самым определив потенциальные пробелы в покрытии.

Проектирование и выполнение нефункциональных тестов может потребовать специальных навыков и знаний, например, знаний о слабых сторонах, свойственных той или иной структуре проекта или технологии (это могут быть уязвимости безопасности, связанные с конкретным языком программирования), знание специфической базы пользователей (например, представления о пользователях управленческих систем в медицинских учреждениях).

Типы и уровни тестирования

Примерами функциональных тестов являются:

- Компонентное тестирование: тесты разрабатываются на основе того, как компонент должен вычислять сложные проценты.
- Тестирование интеграции компонентов: тесты разрабатываются на основе того, как информация учетной записи, созданная в пользовательском интерфейсе, передается модулю бизнес-логики.
- Системное тестирование: тесты разрабатываются на основе того, как владельцы счетов могут подать заявку на кредитную линию на своих расчетных счетах.
- Системное интеграционное тестирование: тесты разрабатываются на основе того, как система использует внешний микросервис для проверки кредитного рейтинга владельца счета.
- Приемочное тестирование: тесты разрабатываются на основе того, как банкир обрабатывает одобрение или отклонение заявки на получение кредита.

Примерами нефункциональных тестов являются:

- Компонентное тестирование: тесты разрабатываются для оценки количества циклов ЦП, необходимых для выполнения расчета сложных процентов.
- Тестирование интеграции компонентов: разрабатываются тесты на защищенность от уязвимостей при переполнении буфера данными, переданными из пользовательского интерфейса в модуль бизнес-логики.
- Системное тестирование: разрабатываются тесты на переносимость для проверки того, работает ли слой представления во всех поддерживаемых браузерах и мобильных устройствах.
- Системное интеграционное тестирование: разрабатываются тесты надежности для оценки работоспособности системы, если микросервис кредитной оценки не отвечает.
- Приемочное тестирование: разрабатываются тесты практичности для оценки доступности интерфейса обработки банковских процессов для людей с ограниченными возможностями.

Примерами тестов, связанных с изменением являются:

- Компонентное тестирование: для каждого компонента разрабатываются автоматические регрессионные тесты и включаются в среду непрерывной интеграции.
- Тестирование интеграции компонентов: разрабатываются тесты, для подтверждения исправлений дефектов, связанных со связями, по мере того как исправления проверяются в репозитории кода.
- Системное тестирование: повторно выполняются все тесты для конкретного рабочего процесса, если любой из аспектов этого процесса меняется.
- Системное интеграционное тестирование: ежедневно повторяются тесты для приложения, взаимодействующего с микросервисом кредитной оценки, в рамках непрерывного развертывания этого микросервиса.
- Приемочное тестирование: все ранее неудавшиеся тесты повторно выполняются после устранения дефекта, обнаруженного при приемочных испытаниях.

Тестирование в период сопровождения

Сопровождение необходимо для сохранения или улучшения нефункциональных характеристик качества компонента или системы в течение всего срока службы, особенно эффективности производительности, совместимости, надежности, безопасности и переносимости.

В процессе сопровождения системы после внесения в нее изменений должно выполняться и тестирование как с целью оценки успеха, ради которого были сделаны изменения, так и для проверки возможных побочных эффектов в частях системы, которые остаются неизменными.

Тестирование в период сопровождения фокусируется на тестировании изменений в системе, а также на тестировании неизмененных частей, на которые могли повлиять изменения.

Тестирование в период сопровождения

Релиз на этапе сопровождения системы может потребовать проведения тестирования в период сопровождения на нескольких уровнях тестирования с использованием различных типов тестов в зависимости от его объема. Объем тестирования в период сопровождения зависит от:

- Степени риска изменения, например, степени, в которой измененная область программного обеспечения взаимодействует с другими компонентами или системами.
- Размера существующей системы.
- Величины внесенных изменений.

Необходимые условия для тестирования в период сопровождения

- Модификации, такие как запланированные улучшения (например, базирующиеся на графике выпуска обновлений), корректирующие и аварийные изменения, изменения среды эксплуатации (например, запланированные обновления операционной системы или базы данных), обновления коммерческого готового программного обеспечения и исправления дефектов и уязвимостей.
- Миграция, например, с одной платформы на другую, которая может потребовать проведения эксплуатационных тестов новой среды, а также измененного программного обеспечения или тестов преобразования данных, когда данные будут перенесены в поддерживаемую систему из другого приложения.
- Снятие с эксплуатации, например, когда заканчивается жизненный цикл приложения.

Необходимые условия для тестирования в период сопровождения

Когда приложение или система снимаются с эксплуатации, то может потребоваться тестирование переноса или архивирования данных, если требуются длительные периоды хранения данных. Также может потребоваться тестирование процедур восстановления после архивирования для случаев длительных периодов хранения. Кроме того, может потребоваться регрессионное тестирование, чтобы гарантировать, что все функциональные возможности, которые остаются в эксплуатации, все еще работают.

Для «Интернета вещей» тестирование в период сопровождения может быть вызвано внедрением в общую систему совершенно новых или модифицированных элементов, таких как аппаратные устройства и программные службы.

Анализ влияния для тестирования в период сопровождения

Анализ влияния оценивает изменения, которые были сделаны для обновленной эксплуатируемой версии для определения предполагаемых последствий, а также ожидаемых и возможных побочных эффектов изменения, и определяет области в системе, на которые может повлиять изменение.

Анализ влияния может быть выполнен до внесения изменений, чтобы решить, следует ли вносить изменения, исходя из возможных последствий в других областях системы.

Анализ влияния может быть затруднен если:

- Спецификации устарели или отсутствуют.
- Тестовые сценарии не задокументированы или устарели.
- Во время разработки уделялось недостаточно внимания сопровождаемости программного обеспечения.

Спасибо за внимание.