

РОЛЬ ТЕСТИРОВАНИЯ В ОБЕСПЕЧЕНИИ КАЧЕСТВА ПРОГРАММНОГО ПРОДУКТА

1. Уровни, виды и типы тестирования

2. Критерии тестирования

3. Принципы тестирования

УРОВНИ, ВИДЫ И ТИПЫ ТЕСТИРОВАНИЯ

- С технической точки зрения тестирование заключается в выполнении приложения на некотором множестве исходных данных в сверке получаемых результатов с заранее известными (эталонными) с целью установить соответствие различных свойств и характеристик приложения заказанным свойствам.

ТЕСТИРОВАНИЕ

- **Тестирование** — контролируемое выполнение программы на конечном множестве тестовых данных и анализ результатов этого выполнения для поиска ошибок.
- Термин «отладка» в отечественной литературе трактуется двояко: для обозначения активности по поиску ошибок (собственно тестирование), по нахождению причин их появления и исправлению или активности по локализации и исправлению ошибок.

ТЕСТИРОВАНИЕ И ОТЛАДКА

- Тестирование — это процесс выполнения ПО системы или компонента в условиях анализа или записи получаемых результатов в целях проверки (оценки) некоторых свойств тестируемого объекта; анализа пункта требований к ПО в целях фиксации различий между существующим состоянием ПО и требуемым (что свидетельствует о проявлении ошибки) при экспериментальной проверке соответствующего пункта требований.
- Порой термины «тестирование» и «отладка» используют взаимозаменяемо, но внимательные программисты различают два этих процесса. Тестирование — это средство обнаружения ошибок, тогда как отладка является средством поиска и устранения причин уже обнаруженных ошибок.

Шаги процесса задаются тестами.

Каждый тест определяет:

- свой набор исходных данных и условий для запуска программы;
- набор ожидаемых результатов работы программы.

Другое название теста — *тестовый вариант*. Полную проверку программы гарантирует исчерпывающее тестирование. Оно требует проверить все наборы исходных данных, все варианты их обработки и включает в себя большое количество тестовых вариантов. В большинстве случаев исчерпывающее тестирование невозможно, прежде всего из-за ограничения по времени.

Хорошим считают тестовый вариант с высокой вероятностью обнаружения еще не раскрытой ошибки. *Успешным* называют тест, который обнаруживает до сих пор не раскрытую ошибку.

ЦЕЛЬ ПРОЕКТИРОВАНИЯ ТЕСТОВЫХ ВАРИАНТОВ

Целью проектирования тестовых вариантов является систематическое обнаружение различных классов ошибок при минимальных затратах времени и стоимости.

Тестирование обеспечивает:

- обнаружение ошибок;
- демонстрацию соответствия функций программы ее назначению;
- демонстрацию реализации требований к характеристикам программы;
- отображение надежности как индикатора качества программы.

- Целью тестирования является нахождение ошибок. Успешным считается тест, нарушающий работу ПО. Все остальные этапы разработки направлены на предотвращение ошибок и недопущение нарушения работы программы.
- Тестирование никогда не доказывает отсутствие ошибок. Отсутствие ошибок может указывать как на безупречность программы, так и на неэффективность или неполноту тестов.
- Тестирование не повышает качества ПО — оно указывает на качество программы, но не влияет на него.

ВИДЫ ТЕСТИРОВАНИЯ

Тестирование — самая популярная методика повышения качества, подкреплённая многими исследованиями и богатым опытом разработки коммерческих приложений. Существует множество видов тестирования: одни обычно выполняют сами разработчики, а другие — специализированные группы.

Перечислим **виды тестирования**:

- блочное;
- тестирование компонента;
- интеграционное тестирование;
- регрессивное тестирование;
- тестирование системы.

- **блочное** — это тестирование полного класса, метода или небольшого приложения, написанного одним программистом или группой, выполняемое отдельно от прочих частей системы;
- **тестирование компонента** — это тестирование класса, пакета, небольшого приложения или другого элемента системы, разработанного несколькими программистами или группами, выполняемое в изоляции от остальных частей системы;
- **интеграционное тестирование** — это совместное выполнение двух или более классов, пакетов, компонентов или подсистем, созданных несколькими программистами или группами;
- **регрессивное тестирование** — это повторное выполнение тестов, направленное на обнаружение дефектов в программе, уже прошедшей этот набор тестов;
- **тестирование системы** — это выполнение ПО в его окончательной конфигурации, интегрированного с другими программными и аппаратными системами.

РЕАЛИЗАЦИЯ ТЕСТИРОВАНИЯ

- **Этап 1.** Создание тестового набора (test suite) путем ручной разработки или автоматической генерации для конкретной среды тестирования (testing environment).
- **Этап 2.** Прогон программы на тестах, управляемый тестовым монитором (test monitor, test driver) с получением протокола тестирования (test log).
- **Этап 3.** Оценка результатов выполнения программы на наборе тестов в целях принятия решения о продолжении или остановке тестирования.

КРИТЕРИИ ТЕСТИРОВАНИЯ

Можно выделить требования к идеальному критерию тестирования:

- критерий должен быть достаточным, т.е. показывать, когда некоторое конечное множество тестов достаточно для тестирования данной программы;
- критерий должен быть полным, т.е. в случае ошибки должен существовать тест из множества тестов, удовлетворяющих критерию, который раскрывает ошибку;
- критерий должен быть надежным, т.е. любые два множества тестов, удовлетворяющих ему, одновременно должны раскрывать или не раскрывать ошибки программы;
- критерий должен быть легко проверяемым, например, вычисляемым на тестах.

КЛАССЫ КРИТЕРИЕВ

Для нетривиальных классов программ в общем случае не существует полного и надежного критерия, зависящего от программ или спецификаций. Поэтому, как правило, стремятся к идеальному общему критерию через реальные частные.

Выделяют следующие классы критериев:

- структурные критерии — используют информацию о структуре программы (критерии так называемого белого ящика);
- функциональные критерии — формулируются в описании требований к программному изделию (критерии так называемого черного ящика);
- критерии стохастического тестирования — формулируются в терминах проверки наличия заданных свойств у тестируемого приложения, средствами проверки некоторой статистической теории;
- мутационные критерии — ориентированы на проверку свойств программного изделия на основе подхода Монте-Карло.

СТРУКТУРНЫЕ КРИТЕРИИ (КЛАСС I)

Структурные критерии используют модель программы в виде «белого ящика», что предполагает знание исходного текста программы или спецификации программы в виде потокового графа управления. Структурная информация понятна и доступна разработчикам подсистем и модулей приложения, поэтому данный класс критериев часто используется на этапах модульного и интеграционного тестирования.

Структурные критерии базируются на основных элементах: *операторах, ветвях и путях*.

Условие критерия тестирования команд (критерий C0) — набор тестов в совокупности должен обеспечить прохождение каждой команды не менее одного раза. Это слабый критерий, используется в больших программных системах, где другие критерии применить невозможно.

Условие критерия тестирования ветвей (критерий C1) — набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза. Это достаточно сильный и при этом экономичный критерий. Данный критерий часто используется в системах автоматизации тестирования.

Условие критерия тестирования путей (критерий C2) — набор тестов в совокупности должен обеспечить прохождение каждого пути не менее одного раза. Если программа содержит цикл (в особенности с неявно заданным числом итераций), то число итераций ограничивается константой (часто — 2, или числом классов выходных путей).

Структурные критерии не проверяют соответствие спецификации, если оно не отражено в структуре программы.

ФУНКЦИОНАЛЬНЫЕ КРИТЕРИИ (КЛАСС II)

- Функциональный критерий — важнейший для программной индустрии критерий тестирования. Он обеспечивает, прежде всего, контроль степени выполнения требований заказчика в программном продукте. Поскольку требования формулируются к продукту в целом, они отражают взаимодействие тестируемого приложения с окружением. При функциональном тестировании преимущественно используется модель «черного ящика». Проблема функционального тестирования — это, прежде всего, трудоемкость. Дело в том, что документы, фиксирующие требования к программному изделию (Software requirement specification, Functional specification и т.п.), как правило, достаточно объемны, тем не менее, соответствующая проверка должна быть всеобъемлющей.

ВИДЫ ФУНКЦИОНАЛЬНЫХ КРИТЕРИЕВ

ТЕСТИРОВАНИЕ ПУНКТОВ СПЕЦИФИКАЦИИ

- Набор тестов в совокупности должен обеспечить проверку каждого тестируемого пункта не менее одного раза. Спецификация требований может содержать сотни и тысячи пунктов требований к программному продукту и каждое из этих требований при тестировании должно быть проверено в соответствии с критерием не менее чем одним тестом.

ТЕСТИРОВАНИЕ КЛАССОВ ВХОДНЫХ ДАННЫХ

- Набор тестов в совокупности должен обеспечить проверку представителя каждого класса входных данных не менее одного раза при создании тестов. Классы входных данных сопоставляются с режимами использования тестируемого компонента или подсистемы приложения, что заметно сокращает варианты перебора, учитываемые при разработке тестовых наборов. Следует заметить, что, перебирая в соответствии с критерием величины входных переменных (например, различные файлы — источники входных данных), мы вынуждены применять мощные тестовые наборы. Действительно, наряду с ограничениями на величины входных данных, существуют ограничения на величины входных данных во всевозможных комбинациях, в том числе проверка реакций системы на появление ошибок в значениях или структурах входных данных. Учет этого многообразия — процесс трудоемкий, что создает сложности для применения критерия

ТЕСТИРОВАНИЕ ПРАВИЛ

- Набор тестов в совокупности должен обеспечить проверку каждого правила, если входные и выходные значения описываются набором правил некоторой грамматики. Следует заметить, что грамматика должна быть достаточно простой, чтобы трудоемкость разработки соответствующего набора тестов была реальной (вписывалась в сроки и штат специалистов, выделенных для реализации фазы тестирования)

ТЕСТИРОВАНИЕ КЛАССОВ ВЫХОДНЫХ ДАННЫХ

- Набор тестов в совокупности должен обеспечить проверку представителя каждого выходного класса, при условии, что выходные результаты заранее расклассифицированы, причем отдельные классы результатов указывают, в том числе, ограничения на ресурсы или на время (time out); при создании тестов классы выходных данных сопоставляются с режимами использования тестируемого компонента или подсистемы, что заметно сокращает варианты перебора, учитываемые при разработке тестовых наборов

ТЕСТИРОВАНИЕ ФУНКЦИЙ

- Набор тестов в совокупности должен обеспечить проверку каждого действия, реализуемого тестируемым модулем, не менее одного раза. Очень популярный на практике критерий, который, однако, не обеспечивает покрытия части функциональности тестируемого компонента, связанной со структурными и поведенческими свойствами, описание которых не сосредоточено в отдельных функциях (описание рассредоточено по компоненту)

КРИТЕРИЙ ТЕСТИРОВАНИЯ ФУНКЦИЙ

- Объединяет отчасти особенности структурных и функциональных критериев. Он базируется на модели «полупрозрачного ящика», где явно указаны не только входы и выходы тестируемого компонента, но также состав и структура используемых методов (функций, процедур) и классов

КОМБИНИРОВАННЫЕ КРИТЕРИИ ДЛЯ ПРОГРАММ И СПЕЦИФИКАЦИЙ

- Набор тестов в совокупности должен обеспечить проверку всех комбинаций непротиворечивых условий программ и спецификаций не менее одного раза. При этом все комбинации непротиворечивых условий надо подтвердить, а условия противоречий следует обнаружить и ликвидировать

СТОХАСТИЧЕСКИЕ КРИТЕРИИ (КЛАСС III)

Стохастическое тестирование применяется при тестировании сложных программных комплексов, когда набор детерминированных тестов (X, Y) имеет громадную мощность. Когда подобный набор невозможно разработать и исполнить на фазе тестирования, можно применить следующую методику:

- разработать программы-имитаторы случайных последовательных входных сигналов $\{x\}$;
- вычислить независимым способом значения $\{y\}$ для соответствующих входных сигналов $\{y\}$ и получить тестовый набор $\{X, Y\}$;
- протестировать приложение на тестовом наборе $\{X, Y\}$, используя два способа контроля результатов: детерминированный контроль и стохастический контроль

КРИТЕРИИ СТОХАСТИЧЕСКОГО ТЕСТИРОВАНИЯ:

- *статистические методы окончания тестирования* — стохастические методы принятия решений о совпадении гипотез о распределении случайных величин. К ним принадлежат широко известные метод Стьюдента (t), метод Хи-квадрат (χ^2) и т.д.;
- *метод оценки скорости выявления ошибок* — основан на модели скорости выявления ошибок, согласно которой тестирование прекращается, если оцененный интервал времени между текущей ошибкой и следующей слишком велик для фазы тестирования приложения.

МУТАЦИОННЫЙ КРИТЕРИЙ (КЛАСС IV)

- Постулируется, что профессиональные программисты пишут сразу почти правильные программы, отличающиеся от правильных мелкими ошибками или опечатками (например, перестановка местами максимальных значений индексов в описании массивов, ошибки в знаках арифметических операций, занижение или завышение границы цикла на 1 и т.д.). Предлагается подход, позволяющий на основе мелких ошибок оценить общее число ошибок, оставшихся в программе.

Подход базируется на следующих понятиях:

- **мутации** — мелкие ошибки в программе;
- **мутанты** — программы, отличающиеся друг от друга мутациями;
- **метод мутационного тестирования** — в разрабатываемую программу P вносят мутации, т.е. искусственно создают программы-мутанты P_1, P_2, \dots . Затем программа P и ее мутанты тестируются на одном и том же наборе тестов $\{X, Y\}$.

Если на наборе $\{X, Y\}$ подтверждается правильность программы P и, кроме того, выделяются все внесенные в программы-мутанты ошибки, то набор тестов (X, Y) соответствует мутационному критерию, а тестируемая программа объявляется правильной.

Если некоторые мутанты не выявили всех мутаций, то надо расширять набор тестов (X, Y) и продолжать тестирование.

ПРИНЦИПЫ ТЕСТИРОВАНИЯ

- Современные приложения имеют больше возможностей благодаря взаимодействию многих модулей. Число тестовых условий, необходимых для обращения к приложениям этого типа, может превысить бюджет и требования плана. Это происходит из-за неэффективности процесса тестирования, сосредоточенного на исследовании кода законченного приложения. Только организации, способные выдержать такие затраты и гибкий временной график, могут и дальше увеличивать продолжительность тестирования и отношение времени тестирования ко времени разработки.

- Тестирование должно помочь находить и исправлять ошибки на самой ранней возможной стадии. Пересмотр процесса тестирования включает в себя определение концептуальной структуры, организующей различные технологии тестирования. Среда для этого процесса построена на концепции «стадийной локализации» (stage containment), т.е. обнаружении и исправлении ошибок на той стадии, где они и появились.
- Пересмотр процесса тестирования включает в себя сопоставление стадии обнаружения ошибки со стадией, на которой ошибка в системе впервые возникла. В результате мероприятия поиска ошибок сдвигаются на ранние стадии процесса разработки, когда вносить изменения проще и дешевле.

ВЕРИФИКАЦИЯ

- *Верификация* удостоверяет, что объект работы внутренне не противоречив и соответствует стандартам. Преимущество верификации состоит в обнаружении ошибок на ранних этапах разработки до того, как они попадут на следующую стадию. Это уменьшает затраты. Верификация применима ко всем объектам (как к тестовым моделям, так и к спецификациям). Методы верификации включают в себя экспертные оценки, формальный контроль и проверки на непротиворечивость.

ПРОВЕРКА НА КОРРЕКТНОСТЬ

- Здесь проверяется, удовлетворяет ли объект требованиям, специфицированным на предыдущей и более ранних стадиях. Преимущество проверки на корректность заключается в отлавливании ошибок до того, как они перешли в следующую стадию разработки. Эти методы также применимы ко всем объектам (к тестовым моделям и спецификациям). Используемые здесь приемы могут включать в себя обзоры и формальные проверки, а также прототипирование и моделирование

ТЕСТИРОВАНИЕ, ОСНОВАННОЕ НА СПЕЦИФИКАЦИЯХ

- Мероприятия тестирования сосредоточены на проверке определенных спецификаций на каждой стадии разработки. Это уменьшает степень наложения тестов и точно определяет рамки и задачу каждого из них. Проверка на основе спецификаций позволяет отслеживать требования на протяжении всего процесса тестирования и предоставляет основу для управления процессом тестирования.

КОНТРОЛЬНЫЕ ВОПРОСЫ

- Что такое тестирование информационных систем?
- В чем заключается разница процессов тестирования и отладки?
- Какие результаты должно обеспечивать тестирование?
- Перечислите виды тестирования.
- Какие основные критерии тестирования необходимо использовать при разработке программ тестирования?
- Какие классы критериев существуют?
- Что такое верификация?
- В чем суть тестирования, основанного на спецификациях?