Исследование программного кода на предмет ошибок и отклонения от алгоритма.

Тестирование программного кода – процесс выполнения программного кода, направленный на выявление существующих в нем дефектов. Под дефектом здесь понимается участок программного кода, выполнение которого при определенных условиях приводит к неожиданному поведению системы (т.е. поведению, не соответствующему требованиям). Неожиданное поведение системы может приводить к сбоям в ее работе и отказам, в этом случае говорят о существенных дефектах программного кода. Некоторые дефекты вызывают незначительные проблемы, не нарушающие процесс функционирования системы, но несколько затрудняющие работу с ней. В этом случае говорят о средних или малозначительных дефектах.

Задача тестирования при таком подходе – определение условий, при которых проявляются дефекты системы и протоколирование этих условий. В задачи тестирования обычно не входит выявление конкретных дефектных участков программного кода и никогда не входит исправление дефектов – это задача отладки, которая выполняется по результатам тестирования системы.

Цель применения процедуры тестирования программного кода – минимизация количества дефектов, в особенности существенных, в конечном продукте. Тестирование само по себе не может гарантировать полного отсутствия дефектов в программном коде системы. Однако, в сочетании с процессами верификации и валидации, направленными на устранение противоречивости и неполноты проектной документации (в частности – требований на систему), грамотно организованное тестирование дает гарантию того, что система удовлетворяет требованиям и ведет себя в соответствии с ними во всех предусмотренных ситуациях.

При разработке систем повышенной надежности, например, авиационных, гарантии надежности достигаются при помощи четкой организации процесса тестирования, определения его связи с остальными процессами жизненного цикла, введения количественных характеристик, позволяющих оценивать успешность тестирования. При этом, чем выше требования к надежности системы (ее уровень критичности), тем более жесткие требования предъявляются.

Таким образом, в первую очередь мы рассматриваем не конкретные результаты тестирования конкретной системы, а общую организацию процесса тестирования, используя подход «хорошо организованный процесс дает качественный результат». Такой подход является общим для многих международных и отраслевых стандартах качества, о которых более подробно будет рассказано в конце данного курса. Качество разрабатываемой системы при таком подходе является следствием организованного процесса разработки и тестирования, а не самостоятельным неуправляемым результатом.

Поскольку современные программные системы имеют весьма значительные размеры, при тестировании их программного кода используется метод функциональной декомпозиции. Система разбивается на отдельные модули (классы, пространства имен и т.п.), имеющие определенную требованиями функциональность и интерфейсы. После этого по отдельности тестируется каждый модуль – выполняется модульное тестирование. Затем выполняется сборка отдельных модулей в более крупные конфигурации – выполняется интеграционное тестирование, и наконец, тестируется система в целом – выполняется системное тестирование.

С точки зрения программного кода, модульное, интеграционное и системное тестирование имеют много общего, поэтому в данной теме основное внимание будет уделено модульному тестированию, особенности интеграционного и системного тестирования будут рассмотрены позднее.

В ходе модульного тестирования каждый модуль тестируется как на соответствие требованиям, так и на отсутствие проблемных участков программного кода, могущих вызвать отказы и сбои в работе системы. Как правило, модули не работают вне системы – они принимают данные от других модулей, перерабатывают их и передают дальше. Для того, чтобы с одной стороны, изолировать модуль от системы и исключить влияние потенциальных ошибок системы, а с другой стороны – обеспечить модуль всеми необходимыми данными, используется тестовое окружение.

Задача тестового окружения – создать среду выполнения для модуля, эмулировать все внешние интерфейсы, к которым обращается модуль. Об особенностях организации тестового окружения пойдет речь в данной теме.

Типичная процедура тестирования состоит в подготовке и выполнении тестовых примеров (также называемых просто тестами). Каждый тестовый пример проверяет одну «ситуацию» в поведении модуля и состоит из списка значений, передаваемых на вход модуля, описания запуска и выполнения переработки данных – тестового сценария, и списка значений, которые ожидаются на выходе модуля в случае его корректного поведения. Тестовые сценарии составляются таким образом, чтобы исключить обращения к внутренним данным модуля, все взаимодействие должно происходить только через его внешние интерфейсы.

Выполнение тестового примера поддерживается тестовым окружением, которое включает в себя программную реализацию тестового сценария. Выполнение начинается с передачи модулю входных данных и запуска сценария. Реальные выходные данные, полученные от модуля в результате выполнения сценария сохраняются и сравниваются с ожидаемыми. В случае их совпадения тест считается пройденным, в противном случае – не пройденным. Каждый не пройденный тест указывает либо на дефект в тестируемом модуле, либо в тестовом окружении, либо в описании теста.

Совокупность описаний тестовых примеров составляет тест-план – основной документ, определяющий процедуру тестирования программного модуля. Тест-план задает не только сами тестовые примеры, но и порядок их следования, который также может быть важен. Структура и особенности тест-планов будут рассмотрены в данной теме, проблемы, связанные с порядком следования тестовых примеров – в теме «Повторяемость тестирования».

При тестировании часто бывает необходимо учитывать не только требования к системе, но и структуру программного кода тестируемого модуля. В этом случае тесты составляются таким образом, чтобы детектировать типичные ошибки программистов, вызванные неверной интерпретацией требований. Применяются проверки граничных условий, проверки классов эквивалентности. Отсутствие в системе возможностей, не заданных требованиями, гарантируют различные оценки покрытия программного кода тестами, т.е. оценки того, какой процент тех или иных языковых конструкций выполнен в результате выполнения всех тестовых примеров. Обо всем этом пойдет речь в завершение данной темы.

- <https://intuit.ru/studies/courses/1040/209/lecture/5385> - методы тестирования



