Лекция №2

Модульное тестирование

Модульное тестирование – это тестирование программы на уровне отдельно взятых модулей (минимальная компилируемая единица: ООП – класс, процедурное – функция).

Целью модульного тестирования является выявление ошибок, локализованных в конкретном модуле (как правило, это алгоритмические ошибки), и проверка готовности системы к следующему уровню разработки и тестирования. Модульное тестирование, как правило, основывается на подходе белого или прозрачного ящика (учитывается внутренняя структура программы). Использование подхода белого ящика позволяет использовать подходы (?).

Модульное тестирование предполагает создание вокруг модуля тестовой среды. Тестовое окружение создаёт заглушки для всех интерфейсов тестируемого модуля. Эти заглушки используются для подачи входных данных и анализа выходных. Также есть заглушки, имитирующие взаимодействие с внешними модулями.

На этапе модульного тестирования обнаруживаются алгоритмические ошибки внутри одного модуля. Но не обнаруживаются ошибки взаимодействия с другими модулями, неверной трактовки данных, некорректной реализации интерфейса, связанные с производительностью и расходом разных видов ресурсов.

Модульные тесты строятся на основе одного из следующих принципов:

1. Анализ потоков управления – граф потоков управления
   Может применятся критерий покрытия функций, в соответствии с которым, каждая функция должна быть протестирована хоть раз. Критерий покрытия вызовов – каждый вызов должен быть произведен хотя бы один раз.
2. Анализ потоков данных – информационный граф программы
   Выявление аномалий потока данных (ссылки на не инициализированные переменные, избыточные присваивания и т.д.)

Для построения тестов модульного тестирования с требуемой степенью тестируемости необходимо:

1. Построить граф потоков управления – на основе статического анализа программного кода, при этом учитывается критерий
2. Определить на нём тестовые пути – определяются тестовые пути на основе статических, динамических или с помощью метода реализуемых путей
   2.1. Статические методы – путь строится от входной вершины графа, постепенно добавляя по одной дуге к пути, пока не будет достигнута выходная вершина графа. Основной недостаток – не учитываются не реализуемые пути
   + Не высокие требования к ресурсам как для построения тестов и самого тестирования
   – Непредсказуем процент бракованных тестовых случаев (не реализуемых путей)
   – Переход от покрывающего множества путей к множеству тестовые необходимо произвести вручную
   2.2. Динамические методы – предполагают построение полной системы тестов, удовлетворяющих заданному критерию, путём одновременного решения задачи построения покрывающего множества путей и тестовых данных, при этом автоматически учитывается реализуемость или не реализуемость рассматриваемых путей на графе.
   В отличии от статических предполагается, что путь на графе строится на основе постепенного добавления дуг к пути, при этом не нарушая реализуемость и покрывая тестами структурные элементы программы
   – Требуется больше ресурсов как при разработке, так и при тестирования
   + Более качественный набор тестов (без не реализуемых путей)
   2.3. Методы реализуемых путей – из множества возможных путей на графе потоков управления выделяется множество реализуемых путей, из которых выбирается подмножество, покрывающее все необходимые структурные критерии
   – Ещё больше ресурсов

3. Сгенерировать тесты, соответствующие каждому из тестируемых путей – для известных тестовых путей определяются данные, которые обеспечивают прохождение этих путей

Интеграционное тестирование

Интеграционное тестирование – это тестирование части системы, состоящей из двух и более модулей.

Цель – поиск дефектов, связанные с взаимодействием модулей, включая ошибки реализации и интерпретации интерфейсов.

Вокруг тестируемой части программы создаётся тестовое окружение, которое оперирует интерфейсами модулей, отсутствующие модули заменяются программными заглушками, посредством интерфейса подаются какие-то данные на вход интерфейса.

Интеграционное тестирование отличается от модульного целями тестирования (типами обнаруживаемых дефектов), а разные типы обнаруживаемых дефектов в свою очередь определяют порядок выбора тестовых данных. Применяются методы, связанные с покрытием интерфейсов. Используется подход белого ящика, но до модульного уровня. Поскольку интеграционное тестирование выполняется на этапе сборки модулей, необходимо учитывать основные методы интеграции: метод большого взрыва (всё интегрируется одновременно) и инкрементарный (пошаговый) метод (после добавления очередного модуля выполняется модульное тестирование).

При интеграции методом большого взрыва требуются большие затраты на интеграционное тестирование, связанные со следующим: необходимо разработать большое количество тестовых драйверов и заглушек, повышается сложность идентификации ошибок (слишком большое пространство кода).

Пошаговый метод может быть либо сверху-вниз (нисходящее – сначала модули первого уровня при заглушках второго уровня и так далее), либо снизу-вверх (восходящее – сначала разрабатывается модули нижнего уровня, заменяя заглушками модули на уровень выше, постепенно доходя до модулей первого уровня). Трудоёмкость ниже.

Нисходящее тестирование. При тестировании модулей верхнего уровня нет модулей уровнем ниже, что требует создавать их сложные программные заглушки. Проблема определения приоритетов модулей. Затрудняется параллельная разработка модулей разных уровней.

Восходящее тестирование. Тестирование концептуальных особенностей системы в целом выполняется на последнем этапе, когда что-то менять уже поздно.