ГЛАВА 5.

Тестирование модулей.

В этой главе мы рассмотрим начальный шаг структурирования – **тестирование модулей**. Оно представляет собой процесс тестирования отдельных подпрограмм или процедур программы. Причины:

1. Появляется возможность управлять комбинаторикой тестирования;
2. Облегчается задача обнаружения места ошибки и её исправления;
3. Возможность одновременного тестирования нескольких модулей.

Процесс тестирования модулей рассматривается в трёх аспектах: способы построения наборов тестов, порядок, в котором тестируются модули, и некоторые практические рекомендации по реализации

Проектирование тестов

При проектировании тестов для тестирования модулей должны быть доступны два источника информации: спецификация модуля и его текст. Спецификация обычно содержит описание входных и выходных параметров модуля и его функций.

Тестирование модулей в основном ориентировано на принцип белого ящика. Это объясняется тем, что принцип белого ящика труднее реализовать при переходе в последующем к тестированию более крупных единиц.

Процедура создания набора тестов для тестирования модулей такова: анализируется логика отдельного модуля с помощью одного или нескольких методов белого ящика, а затем этот набор тестов применяется при тестировании методами черного ящика по спецификации модуля.

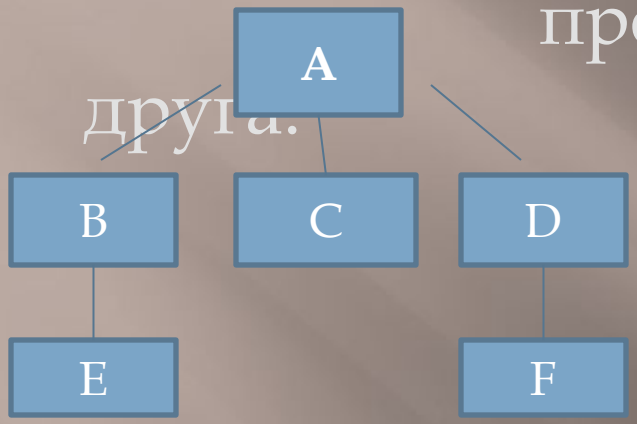
Пошаговое тестирование

Реализация процесса тестирования модулей опирается на два ключевых положения: построение эффективного набора тестов и выбор способа, посредством которого модули комбинируются при построении из них рабочей программы.

В этом разделе мы рассмотрим два подхода к комбинированию модулей: пошаговое и монолитное тестирование.

Монолитный метод

Сначала тестируются шесть модулей, входящих в программу, независимо друг от друга.



Затем модули собираются в программу.

Пошаговый метод

Модули тестируются не изолированно друг от друга, а подключаются поочерёдно для выполнения теста к набору уже ранее оттестированных модулей.

Для тестирования любого модуля требуются специальный

модуль-драйвер (отлаживающий драйвер) и один или

несколько ***модулей-заглушек***.

Например, если тестируется модель B, то первоначально

нужно разработать тесты, а затем написать программу, которая

передаст модулю B входные тестовые данные, необходимые

для его исполнения.

Поскольку в модуле B есть вызов модуля E, нужно сделать

модуль-заглушку, которому будет передано управление при

Допустим, что тестирование пошаговым методом
начинается снизу.

Первоначально можно последовательно или
параллельно выполнить тестирование модулей E,C и
F. При этом для каждого модуля создаётся драйвер,
заглушки здесь не нужны. Следующий шаг –
тестирование модулей B и D, но не независимо, а
совместно с модулями E и F.

Таким образом, разрабатывается драйвер,
включающий тесты, и выполняется тестирование

1. Монолитное тестирование требует больших затрат труда.

Например, для данного случая необходимо создать пять драйверов и пять заглушек (для верхнего модуля драйвер не нужен). При пошаговом тестировании снизу вверх потребуется только пять драйверов, а сверху вниз – только пять заглушек (при тестировании сверху вниз тестируемые модули выполняют функции заглушек, а при тестировании снизу вверх – функции драйверов).

2. При пошаговом тестировании раньше обнаруживаются ошибки в интерфейсах между модулями, поскольку раньше начинается сборка программы.
3. Отладка программ при пошаговом тестировании легче. Если

4. Результаты пошагового тестирования более совершенны.

При тестировании модуля В одновременно с ним исполняется

или модуль А, или модуль Е. Из-за того что оттестированные

ранее модули затем используются в качестве драйверов или

заглушек, они подвергаются дополнительной проверке при

исполнении теста отлаживаемого модуля. При монолитном

тестировании результаты ограничены только этим модулем.

5. Расход машинного времени при монолитном тестировании

меньше. Число исполняемых машинных команд во время

6. Использование монолитного метода предоставляет

большие возможности для параллельной организации

работы на начальной фазе тестирования. Это

положение имеет важное значение при выполнении

больших проектов.

Итак, пункты 1-4 демонстрируют преимущества пошагового тестирования, а 5-6 – его недостатки. Делаем вывод, что пошаговое тестирование является

предпочтительным.

Нисходящее тестирование

Нисходящее тестирование начинается с верхнего модуля. Первый шаг – тестирование модуля A. Для его выполнения необходимо написать модули-заглушки, замещающие модули
B,C и D.

После тестирования верхнего модуля тестирование выполняется в различных последовательностях. Так, если Последовательно тестируются все модули, то возможны следующие варианты: **ABCDEF, ABECDF, ADFCBE**.

При параллельном выполнении тестирования рекомендуется:

1. Если в программе есть критические части (модуль C), то лучше выбирать последовательность, которая включала бы эти части как можно раньше;
2. Модули, включающие операции ввода-вывода, также необходимо подключать в последовательность тестирования как можно раньше

Восходящее тестирование

Тестирование начинается с терминальных модулей (т.е. модулей, не вызывающих другие модули).

Единственное правило состоит в том, чтобы очередной модуль вызывал уже оттестированные модули.

Первым шагом должно быть тестирование E, C и F.

Для каждого из них требуется свой модуль-драйвер, т. е. модуль, который содержит фиксированные тестовые данные, вызывает тестируемый модуль и отображает выходные результаты. В отличие от заглушек, драйвер не должен иметь несколько версий, поэтому он может последовательно вызывать тестируемый модуль несколько раз. В большинстве случаев драйвер проще

Нисходящее тестирование

ПРЕИМУЩЕСТВА

1.Имеет преимущества, если ошибки главным образом в верхней части программы;

2. Представление теста облегчается после подключения функций ввода вывода;

3.Раннее формирование структуры программы позволяет провести её демонстрацию пользователю и служит моральным стимулом.

НЕДОСТАТКИ

1. Необходимо разрабатывать модули-заглушки, которые часто оказываются сложнее, чем кажется вначале;

2. До применения функций ввода-вывода может быть сложно представлять тестовые данные в заглушки;

3. Может оказаться трудным или невозможным создать тестовые условия;

4. Сложнее оценка результатов тестирования;

5. Возможно формирование представления о совмещении тестирования и

Восходящее тестирование

ПРЕИМУЩЕСТВА

1. Имеет преимущества, если ошибки главным образом в модуле нижнего уровня;
2. Легче создавать тестовые условия;
3. Проще оценка результатов.

НЕДОСТАТКИ

1. Необходимо разрабатывать модули-драйверы;
2. Программа как единое целое не существует до тех пор, пока не добавлен последний модуль.

Исполнение теста

Исполнение теста – завершающий этап тестирования модулей. Если исполнение теста приносит результаты, не соответствующие предполагаемым, то это означает, что либо модуль имеет ошибку, либо неверны предполагаемые результаты.

Применение автоматизированных средств позволяет снизить трудоёмкость процесса тестирования. Например, существуют средства, позволяющие избавиться от потребности в драйверах.

Нужно помнить, что необходимой частью тестового набора является описание ожидаемых результатов. При исполнении теста следует обращать внимание на побочные эффекты, например, если модуль делает то, чего он делать не должен.

Также программистам полезно поменяться модулями, чтобы не тестировать свои собственные, однако отладку модуля всегда должен выполнять его автор.

Результаты тестов лучше представлять в такой форме, чтобы можно было повторно воспользоваться ими в будущем.