Тестирования с полным покрытием недостаточно

Март 2018

1 Тестирование и полное покрытие

Охват является полезным показателем тестирования, но высокий охват *не* является целью. Код может иметь 100% покрытие и при этом содержать ошибки. Например, если какой-то код *отсутствует* и, таким образом, программное обеспечение не делает то, что оно должно делать.

1.1 100% покрытый код с ошибкой

В модуле, имевшем 100% покрытие, как линейных, так и ответвленных, был довольно серьезный баг.

Модуль имеет дело со списками таймеров, реализованными в виде (дважды) связанных списков. Каждый элемент содержит информацию о том, «сколько тиков должно пройти после истечения срока действия предыдущего узла, пока мы не истедем». Например, в какой-то момент список может выглядеть следующим образом (отображается только вперед указатели):

```
T1(100) -> T2(20) -> T3(38) -> T4(2)
```

что означает, что у нас есть 4 активных таймера, срок действия которых истекает:

- T1 в 100 клещах/тиков
- T2 в 120 клещах/тиков
- T3 в 158 клещах/тиков
- T4 в 160 клещах/тиков

Это делает работу со списками таймеров довольно эффективной. «Тик» — это единица измерения времени, используемая в приложении (обычно мс).

Ошибка была проста - если мы отменяем таймер, тем самым удаляя его из списка, а это тот самый *первый* таймер в списке (следующий по истечении срока действия), он был обработан плохо. Время, по истечении которого истекает этот таймер, *не* добавляется к следующему таймеру.

Чтобы проиллюстрировать это, предположим, что у нас есть два таймера, оба начали на 100 тиков, но 1 тик «врозь». Итак, после запуска второго таймера, список выглядел так:

```
T1(99) -> T2(1)
```

Допустим, прошло 40 тиков, список будет таким:

```
T1(59) -> T2(1)
```

Затем T1 отменили. Список *должен* быть:

```
T2(60)
```

Но из-за ошибки список был:

```
T2(1)
```

Таким образом, срок действия T2 истечет через 41 тик, а не через 100 тиков.

Существовала ветвь кода, которая обрабатывала этот случай. Но код для добавления времени до истечения срока действия отсутствовал. Это был Ц, и эта строка отсутствовала:

```
list->timeout_left_ms += to_remove->timeout_left_ms;
```

1.2 Почему испытания этого не “поймали”?

Тесты проверили удаление первого элемента. Но, так как «timeout_left_ms» был внутренним по отношению к модулю, его не проверяли. Они также не проверили «что, если пройдет какое-то время после удаления». В нашем примере, что, если, скажем, пройдет 40 тиков - T2 не должен *истечь* (но он истек).

Очевидно, что этот простой сценарий не был предусмотрен при тестировании, но каждый признак (функция) тестировался независимо. Тест на удаление Только проверенное удаление. Тест на «время проходит» (и таймеры, истекающие - или нет) только проверенное время. Ошибки часто скрыты в сценариях, которые Комбинируйте разные функции.

1.3 Урок

Чтобы протестировать с хорошим качеством, нужно рассмотреть сценарии использования и протестировать их (все, если это возможно). Не следует *должно* учитывать показатели охвата. Более того, метрики охвата *не* должны быть целью. Мы не хотим покрывать код, мы хотим его протестировать. Показатель охвата - это всего лишь показатель - если покрытие плохое, маловероятно, что тесты очень хорошие.