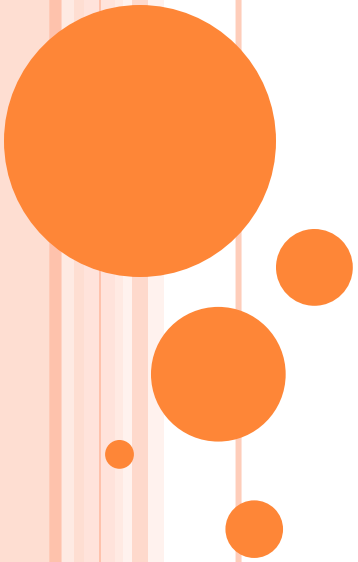
Доклад по первым двум главам книги С. Канера «Тестирование программного обеспечения»



Февраль 2013 г.
Мат-мех СПбГУ
332 гр.
Данилова ЕМ

Для чего нужно тестирование ПО

Для надежной работы программ, так как от этого зависит, например, развитие бизнеса или даже человеческая жизнь.

На практике часто встречается

- При создании определенных типов ПО отступление программистов и управляющего персонала от стандартизированной методологии *абсолютно недопустимо*.

- При разработке ПО тестирование проводится, как правило, в условиях нехватки бюджета, сотрудников, согласия между ними и времени.

Глава 1 Примеры серии тестов

Пусть у нас есть программа, назначение которой - сложить два введенных вами числа, и ее описание:

- в каждом из чисел должна быть одна или две цифры.
- программа выполняет эхо-отображение вводимых чисел, а затем выводит их сумму.
- ввод каждого числа завершается нажатием клавиши <Enter>.
- запускается программа с помощью команды ADDER.

Первый цикл тестирования. Проверяется стабильность работы программы.

Шаги 1 - 3.

- Следует провести 4 основных типа ввода для тестирования:
  - допустимые входные значения;
  - недопустимые входные значения;
  - ввод с редактированием;
  - несвоевременный ввод;

В случае, если проблемы в работе найдены, составляется отчет:

НАЗВАНИЕ КОМПАНИИ КОНФИДЕНЦИАЛЬНО ОТЧЕТ О ПРОБЛЕМЕ №___

ПРОГРАММА_____ ВЫПУСК_____ ВЕРСИЯ_____

ТИП ОТЧЕТА (1-6)__
1 - Ошибка кодирования
2 - Ошибка проектирования
3 - Предложение
4 - Расхождение с документацией
5 - Взаимодействие с аппаратурой
6 - Вопрос

СТЕПЕНЬ ВАЖНОСТИ (1-3)__
1 - Фатальная
2 - Серьезная
3 - Несущественная

ПРИЛОЖЕНИЯ (Д/Н)__
Если да, какие

ПРОБЛЕМА _____

МОЖЕТЕ ЛИ ВЫ ВОСПРОИЗВЕСТИ ПРОБЛЕМНУЮ СИТУАЦИЮ? (Д/Н) __

ПОДРОБНОЕ ОПИСАНИЕ ПРОБЛЕМЫ И КАК ЕЕ ВОСПРОИЗВЕСТИ _____

ПРЕДЛАГАЕМОЕ ИСПРАВЛЕНИЕ (НЕОБЯЗАТЕЛЬНО) _____

ОТЧЕТ ПРЕДСТАВЛЕН СОТРУДНИКОМ_____ ДАТА_/_/_

Следующие строки предназначены только для менеджеров

ФУНКЦИОНАЛЬНАЯ ОБЛАСТЬ_____ ОТВЕТСТВЕННЫЙ_____

КОММЕНТАРИИ _____

СОСТОЯНИЕ(1-2)__
1 - Открыто 2 - Закрыто

ПРИОРИТЕТ (1-5)__

РЕЗОЛЮЦИЯ (1-6)__
1 - Рассматривается
2 - Исправлено
3 - Не воспроизводится
4 - Отложено
5 - Соответствует проекту
6 - Не может быть исправлено

ИСПРАВЛЕННАЯ ВЕРСИЯ _____
7 - Отложено составителем
8 - Нужна дополнительная информация
9 - Не согласен с приоритетом

РАССМОТРЕНО_____ ДАТА_/_/_

ПРОКОНТРОЛИРОВАНО_____ ДАТА_/_/_

СЧИТАТЬ ОТЛОЖЕННЫМ (Д/Н)__

РИСУНОК 1.3. *Форма документа "Отчет о проблеме"*

Первый цикл тестирования.

Шаг 1. Начнем с простого и наиболее очевидного теста

- С программой нужно познакомиться: посмотреть, оценить стабильность.
- В программах, предоставленных для первого формального тестирования, часто сразу же происходит сбой. Не стоит тратить на них много времени.

Что вы делаете	*Что происходит*
Вводите ADDER и нажимаете клавишу <Enter>	Экран мигает. Вверху экрана вы видите знак вопроса.
Нажимаете 2	За знаком вопроса появляется цифра 2.
Нажимаете <Enter>	В следующей строке появляется знак вопроса.
Нажимаете 3	За вторым знаком вопроса появляется цифра 3.
Нажимаете <Enter>	В третьей строке появляется цифра 5. На несколько строк ниже появляется еще один знак вопроса.

РИСУНОК 1.1. *Первый тест программы*

Первый цикл тестирования

Отчет о проблемах, выявленных первым тестом

- Программа работает — она приняла числа 2 и 3 и вернула 5. Но проблемы все же есть. Для их описания составляется отчет.

- 1. *Ошибка проектирования. Нет никаких указаний на то, с какой программой* вы работаете. Откуда вам знать, что именно с той, которая нужна?

Первый цикл тестирования

Отчет о проблемах, выявленных первым тестом (продолжение)

- 2. *Ошибка проектирования. На экране нет никаких инструкций. Откуда* вам знать, что нужно делать? Что, если вы вводите недопустимые числа?

- Отобразить инструкцию на экране не трудно, и она всегда будет перед глазами, в то время как печатная документация может потеряться.

Первый цикл тестирования

Отчет о проблемах, выявленных первым тестом (продолжение)

- 3. *Ошибка проектирования. Как остановить* программу? Эта инструкция тоже должна быть на экране.
- 4. *Ошибка кодирования. Сумма (число* 5) выведена в стороне от слагаемых.

Первый цикл тестирования

Шаг 2. Составим заметки о том, что еще должно быть протестировано

- Выполнив первые, и самые очевидные тесты, следует подумать о том, что еще следует протестировать.
- Свои соображения нужно записать: одни из записей примут форму заметок, другие же могут представлять собой достаточно строго формализованные описания серий тестов.

Первый цикл тестирования
Шаг 2 (продолжение)

- Такие документированные группы тестов в дальнейшем могут послужить для проверки следующих версий программы.

- Пример на рис. 1.4.

Входные данные	*Ожидаемый результат*	*Замечания*
99 + 99	198	Пара наибольших чисел, которые может складывать программа.
-99 + -99	-198	В документации не сказано, что нельзя складывать отрицательные числа.
99 + -14	85	Большое первое число может повлиять на интерпретацию программой второго.
-38 + 99	61	Проверим сложение отрицательного числа с положительным.
56 + 99	155	Проверим, не влияет ли слишком большое второе число на интерпретацию первого.
9 + 9	18	9 является наибольшим числом из одной цифры.
0 + 0	0	Программы часто сбоят на нулях.
0 + 23	23	Программа может особым образом обрабатывать 0, поэтому его нужно проверить и в виде первого, и в виде второго слагаемого.
-78 + 0	-78	

РИСУНОК 1.4. *Тест на допустимые входные данные*

Первый цикл тестирования

Шаг 2. (продолжение)

- Эти тесты охватывают все допустимые входные данные программы — пары чисел, которые ей полагается складывать правильно.

- В первом тесте вы ввели два числа и проверили результат. На рис. 1.4 для тестирования программы предлагается восемь примеров.

Первый цикл тестирования

Шаг 2. (продолжение)

- Прежде всего, тесты были подобраны так, чтобы каждая цифра встречалась в них хотя бы один раз. Мы подобрали по одной комбинации чисел на каждую из вероятных проблем.

- А чтобы определить, на каких данных вероятнее всего возникнут проблемы, эффективнее всего проверить граничные условия.

Первый цикл тестирования

Шаг 3. Проверим допустимые значения и посмотрим, что происходит

- Серия тестов, приведенных на рис. 1.4, охватывает только допустимые значения входных данных программы.
- На следующем этапе тестирования можно создать такую же серию тестов для недопустимых значений.
- Еще одна серия тестов может быть предназначена для проверки редактирования чисел: вы вводите значение, затем изменяете его и только после этого нажимаете <Enter>.

Первый цикл тестирования

Шаг 4.

Формальное тестирование окончено. В дело вступает интуиция. Пробуйте все, что придет вам в голову.

Шаг 5.

Самое время окинуть взглядом программу в целом, увидеть ее недостатки и продумать стратегию будущего тестирования.

Первый цикл тестирования

Шаг 4. Немного тестирования в режиме "свободного полета"

- Разрабатывать новые тесты пока нет никакого смысла, поскольку к новой версии они могут просто не подойти.
- Поэтому можно провести несколько чисто исследовательских экспериментов — все, что придет в голову.

Первый цикл тестирования

Замечания

- *Из двух тестов, от которых ожидается один и тот же результат, проводите только один*

- *Для выполнения всегда выбирайте из класса те тесты, на которых вероятнее всего ожидается сбой программы*

Так, часто это – граничные значения, на которых программа меняет свое поведение. С двух сторон проверять.

Шаг 5. Подведем итоги тому, что мы узнали о программе и ее недостатках

- Эта последняя работа — только для вас. Она не всегда необходима, но часто оказывается очень полезной.
- Самое время мысленно отступить немного назад и окинуть взглядом программу в целом, увидеть ее недостатки и продумать стратегию будущего тестирования.

Первый цикл тестирования. Итоги

- Вы начали с простейшего из возможных тестов. Программа его прошла, вы разработали серию формальных тестов, чтобы проверить, как она работает с допустимыми данными.
- Часть проверок программа не прошла, вы провели несколько неформальных экспериментов и выяснили, что программа вообще очень нестабильна.
- Если бы программа успешно прошла первую серию тестов, вы бы разработали вторую, более обстоятельную. И далее продолжили бы ее тестирование, пока не исчерпались бы идеи или отведенное время.

Второй цикл тестирования. Добиваемся исправления ошибок.

Шаг 1.

Программист должен составить резолюцию на ваших отчетах об ошибках. Из резолюций на ваших отчетах вы увидите, какие тесты больше проводить не нужно, а какие нужно заменить новыми.

Теперь у вас появятся новые идеи для тестирования. ☺️

Второй цикл тестирования

Пример резолюции программиста

1. Ошибка проектирования: Резолюция:	На экране нет названия программы. Не будет исправлена.
2. Ошибка проектирования: Резолюция:	На экране нет инструкций. Не будет исправлена. Примечание: Замечание верное, но вывод инструкций замедлит работу программы.
3. Ошибка проектирования: Резолюция:	Как остановить программу? Исправлена. На экране отображается подсказка: "Для выхода нажмите <Ctrl+C>".
4. Ошибка кодирования: Резолюция:	Сумма (5) выводится в стороне от слагаемых. Исправлена.
5. Ошибка кодирования: Резолюция:	Программа "зависает" на отрицательных числах. Исправлена. Программа будет складывать и отрицательные числа.
6. Ошибка кодирования: Резолюция:	Программа интерпретирует третий введенный символ как нажатие <Enter>. В работе (еще не исправлена).
7. Ошибка кодирования: Резолюция:	Сбой при вводе нечисловых данных. Не проблема. Комментарий: "Не делайте этого".
8. Ошибка кодирования: Резолюция:	Сбой при вводе управляющих символов. Не проблема. Комментарий: "См. отчет 7".
9. Ошибка кодирования: Резолюция:	Сбой при нажатии функциональных клавиш. Не проблема. Комментарий: "См. отчет 7".

РИСУНОК 1.7. *Резолюции на отчетах первого цикла тестирования*

Второй цикл тестирования

Шаг 2. Готовьтесь поспорить с автором

- Проанализируйте комментарии к ошибкам, которые не будут исправлены. Возможно, следует провести дополнительное тестирование

- Чтобы добиться у программиста исправления ошибки, нужно продемонстрировать ситуацию, в которой ее появление абсолютно недопустимо (например, программа «зависает»)

Второй цикл тестирования

Шаг 3

- Просмотрите записи, которые вы сделали в прошлый раз, добавьте к ним новые замечание, и приступайте к тестированию

После исправлений ошибок снова повторите старые и нудные тесты и убедитесь, что программа по-прежнему работает на них правильно.

Итог второго цикла тестирования

- По мере дальнейшей разработки программы вы будете создавать новые серии формальных тестов и выполнять их снова и снова.
- В последнем цикле тестирования все тесты нужно снова выполнить в полном объеме.
- Когда работа над проектом приблизится к концу, придет время самых строгих и самых сложных тестов.

Ваша главная задача — добиться, чтобы те, кто отвечает за ход разработки, до конца понимали серьезность каждой описанной вами проблемы.

Глава 2. Желаемое и действительное в жизни тестировщика

- Полностью протестировать программу невозможно.
- Хорошее тестирование – не исправление всех ошибок, а гарантирование того, что программа будет работать правильно.

Правильный подход к проверке научной теории заключается в поиске не подтверждающих, а опровергающих ее фактов — попытке доказать, что в ней есть ошибки. И чем более тщательное тестирование выдерживает выдвинутая теория, тем больше у нас уверенности в том, что она верна.

Глава 2 Желаемое и действительное в жизни тестировщика.

Причины, по которым полное тестирование не может быть выполнено никогда

- Невозможно проверить реакцию программы на каждую комбинацию входных данных

- Пользовательский интерфейс программы обычно слишком сложен для полного тестирования.

На обнаружение и исправление ошибок тратится от 40 до 80 процентов общей стоимости разработки ПО.

Глава 2. Желаемое и действительное в жизни тестировщика.

Замечания

Результаты проверки очень сильно зависят от ваших ожиданий (психологический фактор).

Чем больше вы выведите ошибок и чем серьезнее они будут, тем лучше.

Поэтому приступайте к тестированию, надеясь на наличие ошибок – они всегда есть.

Глава 2 Желаемое и действительное в жизни тестировщика.

Замечания

- **Невозможно проверить реакцию программы на каждую комбинацию входных данных:**

  в предыдущей главе описывалась совсем простенькая программка. И даже для нее объем возможных входных данных был огромным.

  Чтобы проверить эту программу полностью, нужно провести множество разнообразных тестов.

Глава 2 Желаемое и действительное в жизни тестировщика.

Замечания

- **Следует проверить все недопустимые входные значения:**

Вам следует поверить, как обрабатывает программа не только допустимые входные данные, но и вообще все, что пользователь может ввести с клавиатуры

Глава 2 Желаемое и действительное в жизни тестировщика. Замечания

- **Следует проверить все способы редактирования входных данных:**

  если программа позволяет редактировать вводимые числа, нужно убедиться, что она это делает правильно.

  Проверьте, сможете ли вы изменить любой введенный символ.

  Протестируйте повторное редактирование: введите число, измените его, потом измените еще раз.

Глава 2 Желаемое и действительное в жизни тестировщика.

Замечания

- **Следует проверить реакцию программы на ввод в каждый момент ее работы:**

нужно попробовать ввести данные, когда программа их совсем не ждет.

Вводите числа, пока программа еще обрабатывает предыдущие данные, выводит на экран результат или отображает сообщение.

Итак, для чего же тестируют программы?

- Программу тестируют для того, чтобы найти в ней ошибки
- Смысл тестирования заключается в поиске проблем. Ваша цель — найти их как можно больше, и чем серьезнее найденные проблемы, тем лучше.
- Помните, что времени всегда очень мало, и старайтесь использовать его как можно эффективнее.
- Если тест позволил выявить проблему, значит, он успешный. А тест, не выявивший проблем, был потерей времени.

Итак, для чего же тестируют программы? (продолжение)

- Ошибки ищут для того, чтобы их исправить

- В конечном счете большинство найденных ошибок исправляют, и качество программного продукта улучшается. Это и есть настоящая цель тестирования .