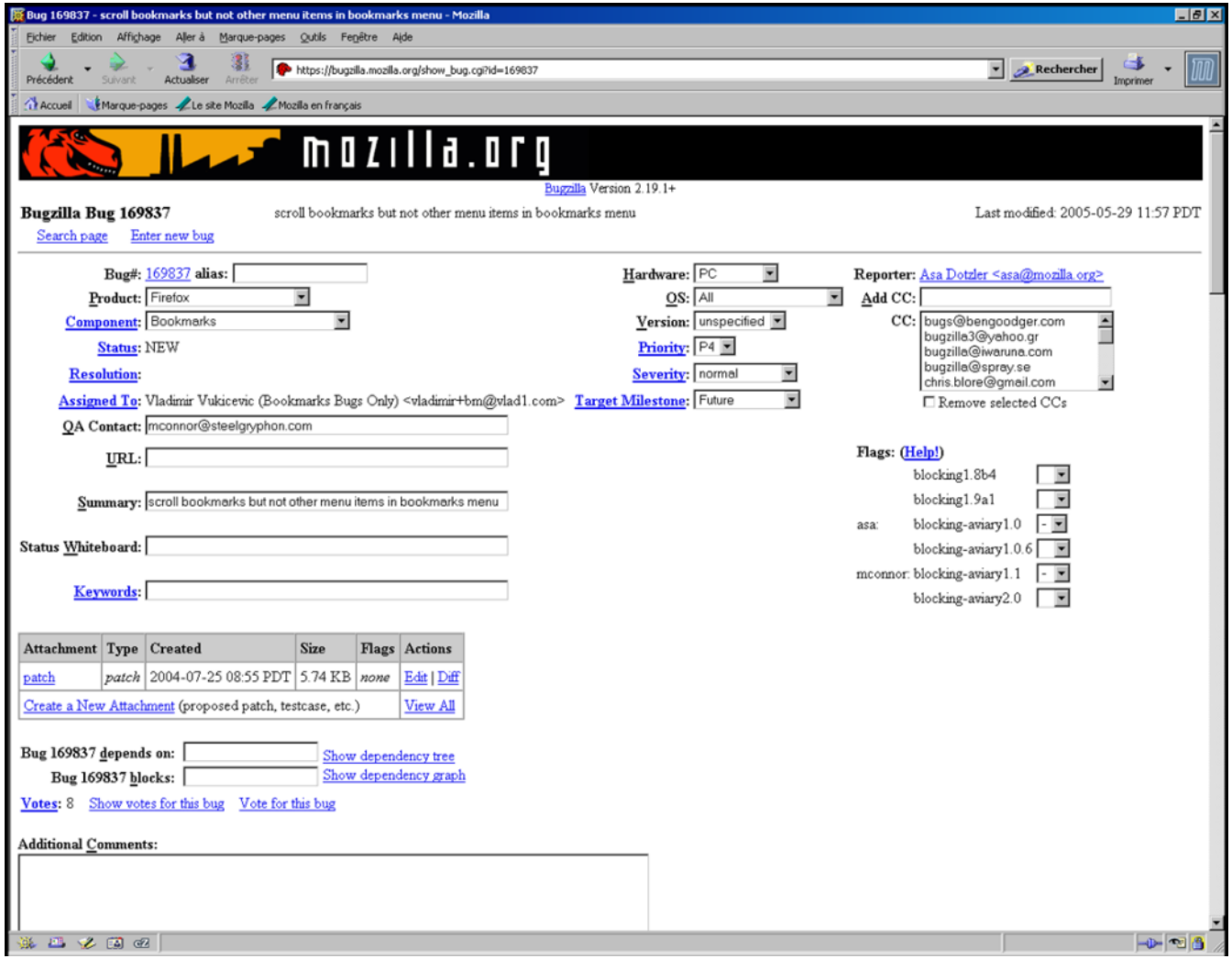
Тренинги Intel Delta Course

«Введение В Управление Качеством ПО»

Введение в Управление Качеством ПО
Тема 2:
Жизненный цикл дефекта. Виды тестирования. Основы разработки тестовых сценариев

Виценко А.Ю., Крюков А.К., Моренов О.А., Пряхин И.В., Семенов Д.С., Чиликин Е.В. Intel

Bug Tracker


Bug 169837 - scroll bookmarks but not other menu items in bookmarks menu - Mozilla
Fichier Edition Affichage Aller à Marque-pages Outils Fenêtre Aide
Précédent Suivant Actualiser Arrêter
https://bugzilla.mozilla.org/show_bug.cgi?id=169837
Rechercher
Imprimer
Accueil Marque-pages Le site Mozilla Mozilla en français
mozilla.org
Bugzilla Version 2.19.1+
Bugzilla Bug 169837
scroll bookmarks but not other menu items in bookmarks menu
Last modified: 2005-05-29 11:57 PDT
Search page Enter new bug
Bug#: 169837 alias:
Product: Firefox
Component: Bookmarks
Status: NEW
Resolution:
Assigned To: Vladimir Vukicevic (Bookmarks Bugs Only) <vladimir+bm@vlad1.com>
QA Contact: mconnor@steelgryphon.com
URL:
Summary: scroll bookmarks but not other menu items in bookmarks menu
Status Whiteboard:
Keywords:
Hardware: PC
OS: All
Version: unspecified
Priority: P4
Severity: normal
Target Milestone: Future
Reporter: Asa Dotzler <asa@mozilla.org>
Add CC:
CC: bugs@bengoodger.com bugzilla3@yahoo.gr bugzilla@iwaruna.com bugzilla@spray.se chris.blore@gmail.com
Remove selected CCs
Flags: (Help!)
blocking1.8b4
blocking1.9a1
asa: blocking-aviary1.0
blocking-aviary1.0.6
mconnor: blocking-aviary1.1
blocking-aviary2.0
Attachment Type Created Size Flags Actions
patch patch 2004-07-25 08:55 PDT 5.74 KB none Edit | Diff
Create a New Attachment (proposed patch, testcase, etc.) View All
Bug 169837 depends on:
Bug 169837 blocks:
Show dependency tree
Show dependency graph
Votes: 8 Show votes for this bug Vote for this bug
Additional Comments:

Дефекты. Жизненный цикл

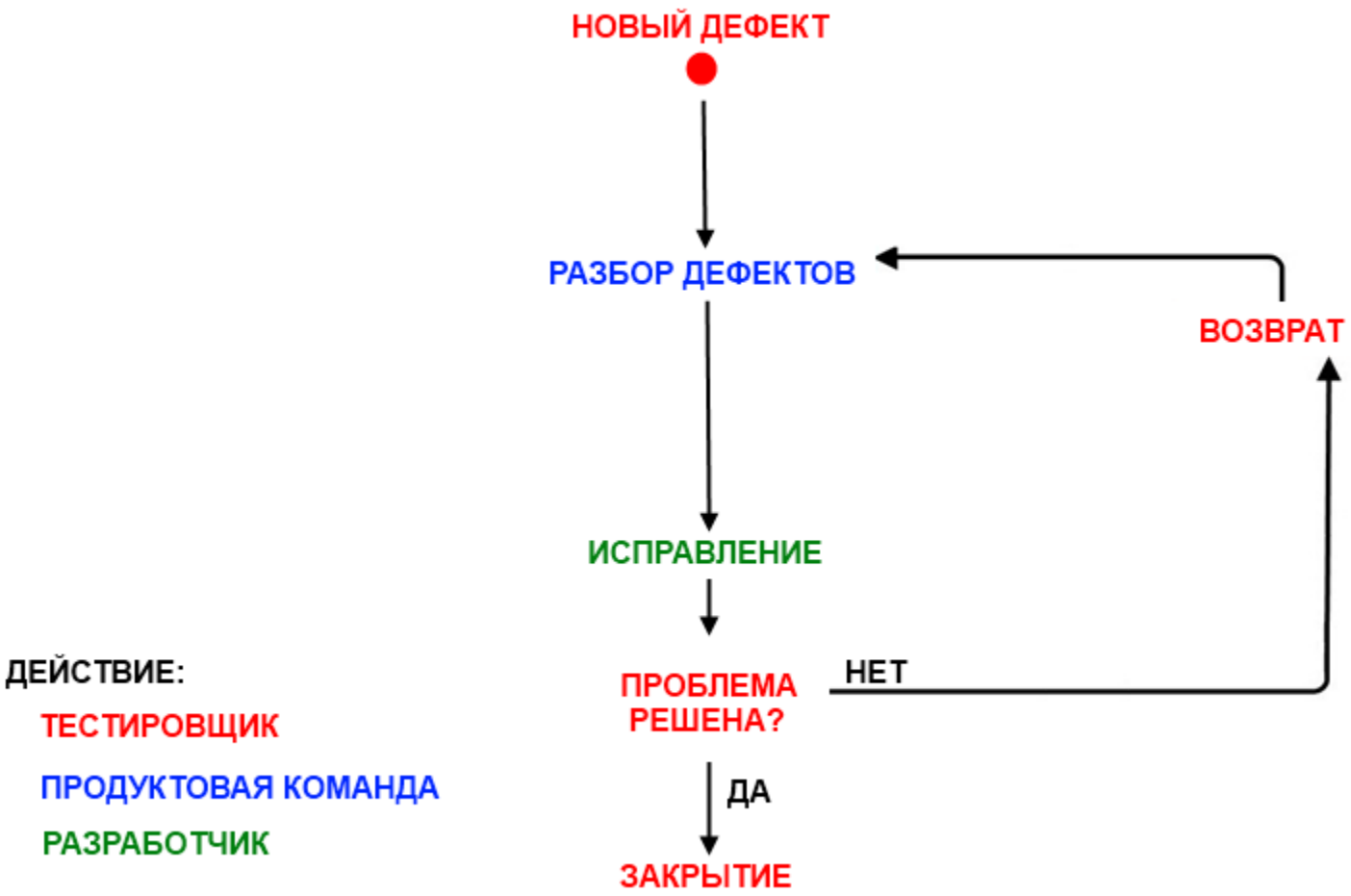
ДЕЙСТВИЕ:

ТЕСТИРОВЩИК

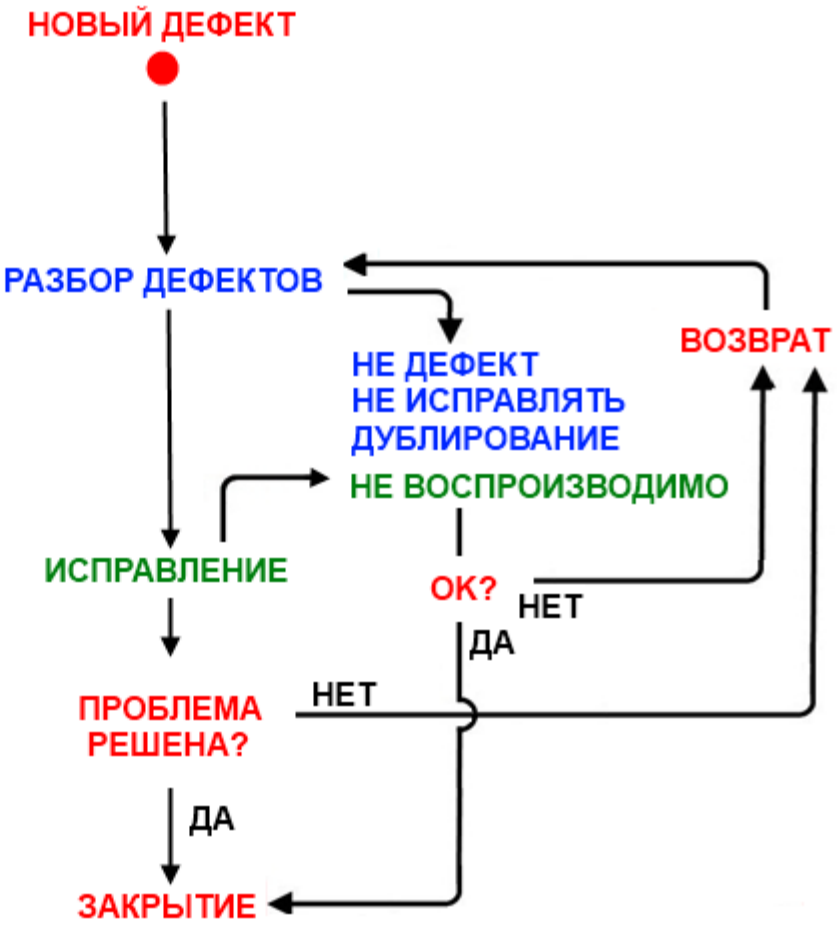
ПРОДУКТОВАЯ КОМАНДА

РАЗРАБОТЧИК

Дефекты. Жизненный цикл



Дефекты. Жизненный цикл



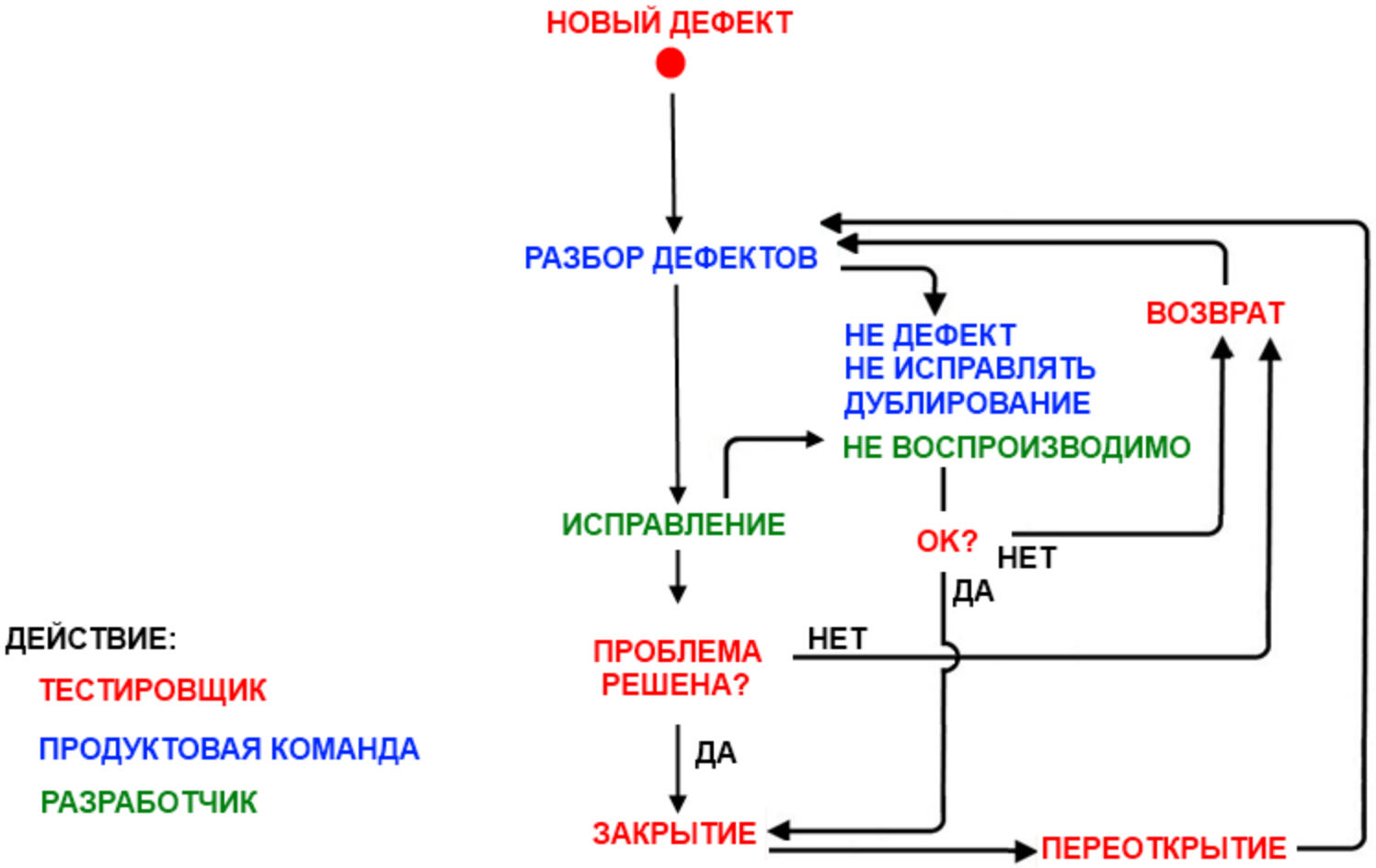
ДЕЙСТВИЕ:

ТЕСТИРОВЩИК

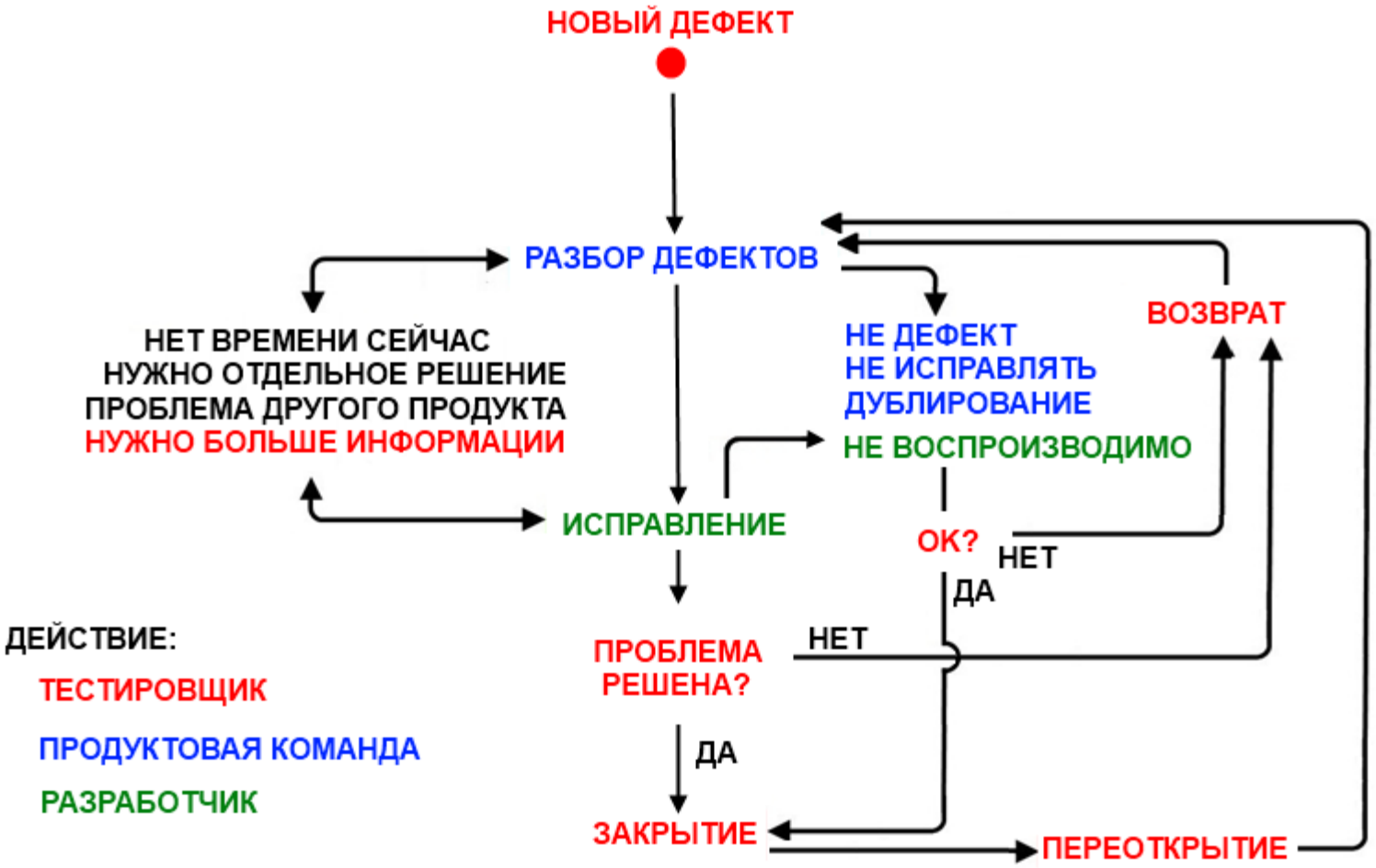
ПРОДУКТОВАЯ КОМАНДА

РАЗРАБОТЧИК

Дефекты. Жизненный цикл



Дефекты. Жизненный цикл



Позитивные и негативные тесты

Позитивные тесты

- Тесты, предназначенные для проверки, что программа выполняет свое основное предназначение
- Тесты на основании «правильных» входных данных
- Тестирование с целью проверки соответствий требованиям

Негативные тесты

- Тесты для проверки устойчивости программы к негативным входным данным
- Тесты на проверки устойчивости программы к ошибкам пользователя
- Тесты на то что у программы нет неожиданных побочных эффектов
- Тестирование с целью «сломаем это!»

Тестирование Черного Ящика

- Не знаем/Игнорируем устройство тестируемого объекта
- Можем управлять входными параметрами
- Среда, в которой проводим эксперименты, может считаться входным параметром
- Можем измерять выходные параметры

Черный Ящик. Когда применять.

Области для поиска ошибок:

Неправильные или пропущенные функции

Ошибки интерфейсов

Инициализация и завершение

Производительность

Структура данных

Стадии применения:

Unit-тестирование

Интеграционное тестирование

Системное тестирование

Приемочное тестирование

Тестирование Черного Ящика - Шаги

1. Изучение спецификаций и требований
2. Выбор входных значений
3. Определение ожидаемых выходных значений

Входные значения	Ожидаемые выходные значения
Вход 1	Выход 1
Вход 2	Выход 2
…	….
Вход N	Выход N

4. Исполнение тестов
5. Сравнение полученных результатов с ожидаемыми

Тестирование Черного Ящика. Стратегии

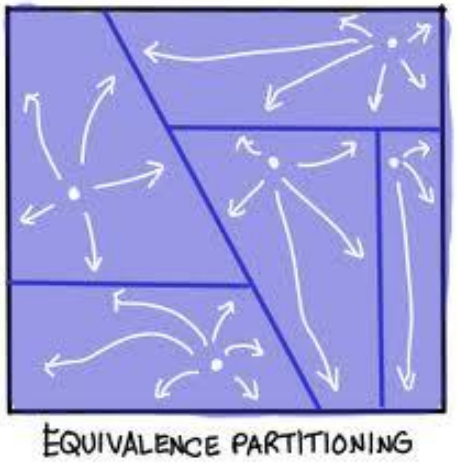
Число тестов определяется числом входов и диапазоном возможных значений входных параметров

Перебор всех возможных входных параметров, как правило, невозможен

Пример: Сложение двух 4хбайтовых целых -
2^{64} входных параметров

Стратегии уменьшения числа тестов:

Классы эквивалентности



Граничные значения

Черный Ящик. Преимущества и недостатки

Преимущества:

- Тестирование с точки зрения пользователя
- Не требует специальных знаний (например конкретного языка программирования)
- Позволяет найти проблемы в спецификациях
- Можно создавать тесты параллельно с кодом
- Тестировщик может быть отделен от разработчиков

Недостатки:

- Эффективность зависит от выбора конкретных тестовых значений
- Необходимость наличия четких и полных спецификаций
- Невозможность сконцентрироваться на особо сложных частях кода
- Трудность локализации причины дефекта
- Возможность не протестировать часть кода

Тестирование Белого Ящика

- Используем знание об устройстве тестируемого объекта
- В случае ПО – имеем полный доступ к тестируемому коду

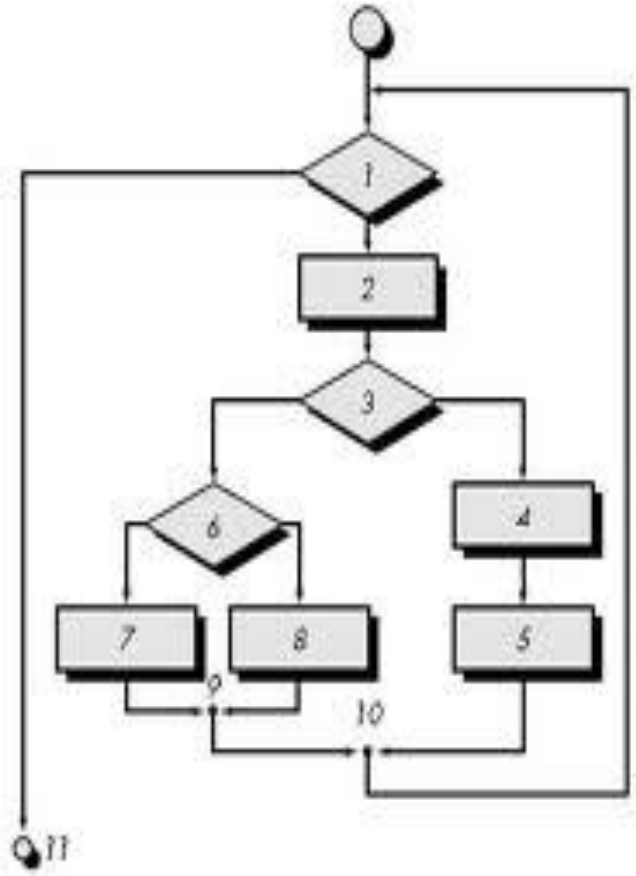
Стадии применения:

- Unit-тестирование
- Интеграционное тестирование

Белый Ящик. Шаги

Представляем программу в виде графа

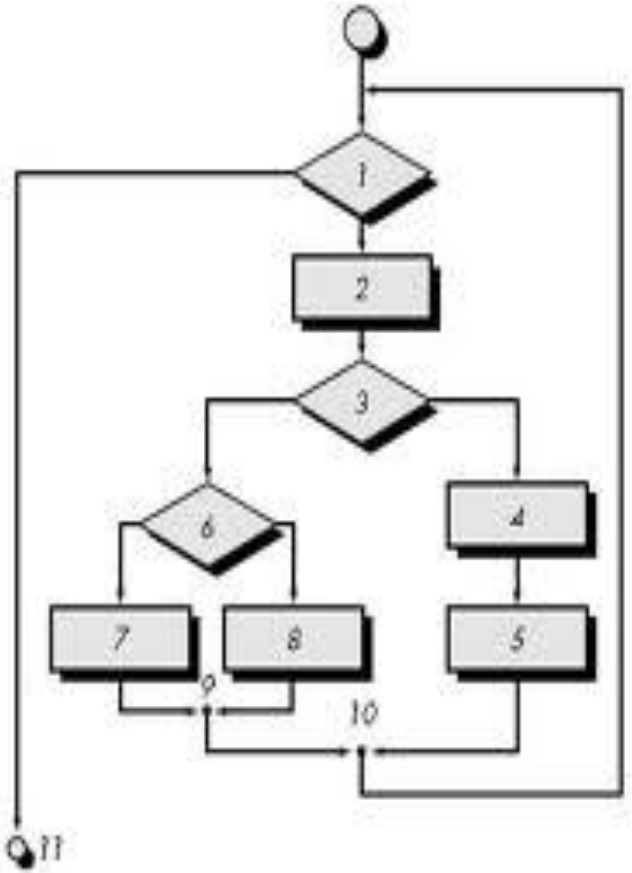
```
PROCEDURE average;

  * This procedure computes the average of 100 or fewer
    numbers that lie between bounding values; it also computes the
    sum and the total number valid.

  INTERFACE RETURNS average, total.input, total.valid;
  INTERFACE ACCEPTS value, minimum, maximum;

  TYPE value[1:100] IS SCALAR ARRAY;
  TYPE average, total.input, total.valid;
    minimum, maximum, sum IS SCALAR;
  TYPE i IS INTEGER;
1 i = 1;
  total.input = total.valid = 0;   2
  sum = 0;
  DO WHILE value[i] <> -999 AND total.input < 100   3
    4 increment total.input by 1;
    IF value[i] >= minimum AND value[i] <= maximum   6
  5   THEN increment total.valid by 1;
  7         sum = sum + value[i]
      ELSE skip
    ENDIF
  8 increment i by 1;
  9 ENDDO
    IF total.valid > 0   10
      11 THEN average = sum / total.valid;
  12   ELSE average = -999;
  13 ENDIF
END average
```



Белый Ящик. Шаги

Создаем тестовые сценарии чтобы:

- Попасть в каждое ветвление
- Пройти хоть раз через все вершины
- Пройти всеми возможными путями
- Пройти через вновь добавленные участки
- Пройти через известные проблемные участки



Белый Ящик. Преимущества и Недостатки

Преимущества:

- Позволяет найти «скрытые» в коде дефекты
- Позитивные побочные эффекты (например, обучение команды)
- Нахождение проблем производительности
- Более надежное разбиение на классы эквивалентности
- Как правило, ускорение цикла нахождение-исправление

Недостатки:

- Не найдем пропущенное в коде
- Дорого

Отличия черного и белого ящиков

Критерий	Черный Ящик	Белый Ящик
Основной уровень применимости	Приемочное тестирование	Юнит-тестирование
Ответственный	Независимый тестировщик	Разработчик
Знание программирования	Не обязательно	Необходимо
Знание реализации	Не обязательно	Необходимо
Знание сценариев использования	Необходимо	Не обязательно
Основа тестовых сценариев	Спецификации	Код

Упражнение

- Есть консольное приложение, получающее на вход 3 числа.
- Смысл чисел – длины отрезков
- На вход выдается сообщение о возможности построить треугольник из этих отрезков:
  - «можно»
  - «нельзя»
  - «ошибка»

Методы выбора входных значений

Бессистемный выбор входных значений не позволит найти большое количество дефектов. Необходимо использование методов для выбора набора входных значений.

Основные методы выбора входных значений:

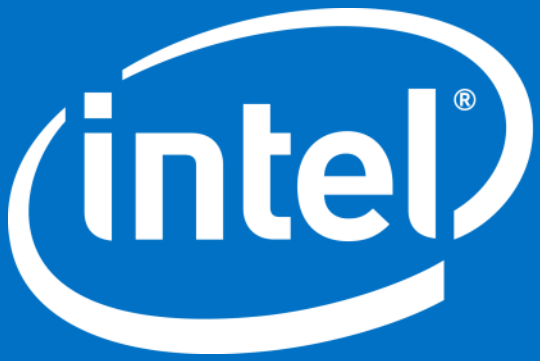
- Перебор всех возможных значений
- Случайные входные данные
- Предугадывание ошибки
- Построение графов «причина-следствие»
- Использование классов эквивалентности
- Исследование граничных значений

Методы проверки результата

Проблема выбора оракула - эвристического принципа или механизма идентификации потенциальной проблемы

Некоторые варианты оракулов:

- Спецификации
- «Пока не упадет»
- Эксперт
- Сравнение с эталоном
- Ограничения
- Похожесть на известную проблему


intel

Домашнее Задание

- Возьмите программу соседа, разработанную для курса «Многопоточность C++»
- Составьте тестовый план для тестирования данной программы по методу черного ящика. Нужны только позитивные тесты
- Выполните ваши тесты. Если нашлись ошибки, то сообщите о них соседу