Тестирование ПО

Цели и задачи тестирования

Кулаков Кирилл Александрович

Информация о курсе

- Лекции в дистанте
- Контроль успеваемости
  - 2 лабораторные работы
  - зачет
- Помощь
  - Сайт курса (http://cs.petrsu.ru/~kulakov/courses/testing)
  - График консультаций кафедры ИМО (215 ауд.)
  - Электронная почта (kulakov@cs.petrsu.ru)

Литература

- В.П. Котляров Основы тестирования программного обеспечения http://www.intuit.ru/department/se/testing/
- С. Канер, Д.Фолк Тестирование ПО
- Э. Дастин, Д. Рэшка, Д. Пол "Автоматизированное тестирование программного обеспечения"
- Р. Калбертсон, К. Браун, Г. Кобб "Быстрое тестирование"
- Д. Макгрегор, Д. Сайкс "Тестирование объектно-ориентированного программного обеспечения"
- Л. Тамре "Введение в тестирование программного обеспечения"
- Р. Савин "Тестирование Дот Ком, или пособие по жесткому обращению с багами в интернет-стартапах"
- Э. Хант, Д. Томас "Программист-прагматик. Путь от подмастерья к мастеру"
- ...

Будни программиста

- Идеальный вариант:
  - Получил задачу от заказчика/менеджера проекта
  - Написал код, проверил работу
  - Сдал код заказчику/менеджеру проекта
  - Все счастливы
- Действительность:
  - Получил задачу от заказчика/менеджера проекта
  - Написал код, проверил работу
  - Сдал код заказчику/менеджеру проекта
  - Обнаружились ошибки
  - Заказчик/менеджер недоволен

Плохой программист Джон

Плохой программист Джон сделал ошибку в коде, из-за которой каждый пользователь программы был вынужден потратить в среднем 15 минут времени на поиск обхода возникшей проблемы. Пользователей было 10 миллионов. Всего впустую потрачено 150 миллионов минут = 2.5 миллиона часов. Если человек спит 8 часов в сутки, то на сознательную деятельность у него остается 16 часов. То есть Джон уничтожил 156250 человеко-дней ≈ 427.8 человеко-лет. Средний мужчина живет 64 года, значит Джон убил примерно 6 целых 68 сотых человека.

Как тебе спится, Джон — серийный программист?

Стоимость исправления ошибки

Цель работы

- Цель работы — получить качественное ПО
- Что такое качество:
  - Качество — набор свойств которые определяют насколько продукт хорош.
  - У каждого участника проекта различное представление качества
- Задача обеспечения качества:
  - определение заинтересованных лиц
  - определение критериев качества заинтересованных лиц
  - нахождения оптимального решения, удовлетворяющего этим критериям
- Тестирование — способ обеспечения качества

Тестирование — это:

- 1980 - Процесс выполнения программы с намерением найти ошибки
- 1987 - Процесс наблюдения за выполнением программы в специальных условиях и вынесения на этой основе оценки каких-либо ее аспектов
- 1990 - Интеллектуальная дисциплина, имеющая целью получение надежного программного обеспечения без излишних усилий на его проверку
- 1999 - Техническое исследование программы для получения информации о ее качестве с точки зрения определенного круга заинтересованных лиц
- 2004 - Проверка соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранном определенным образом

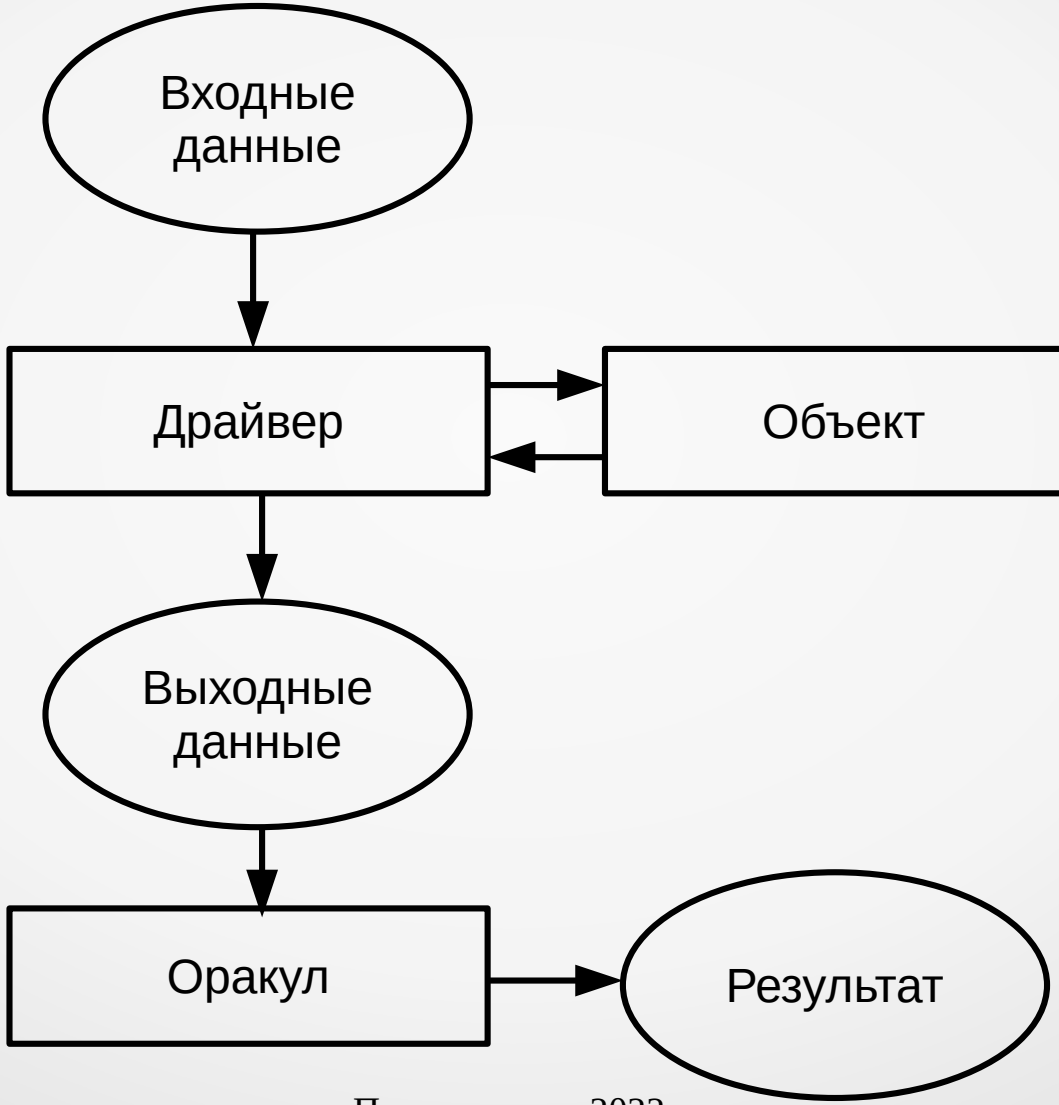
Терминология

- **Тестирование** обеспечивает выявление (констатацию наличия) фактов расхождений с требованиями (ошибок)
  - **Статическое тестирование** выявляет формальными методами анализа без выполнения тестируемой программы неверные конструкции или неверные отношения объектов программы (ошибки формального задания) с помощью специальных инструментов контроля кода
  - **Динамическое тестирование** (собственно тестирование) осуществляет выявление ошибок только на выполняющейся программе с помощью специальных инструментов автоматизации тестирования
- **Отладка** (debug, debugging) – процесс поиска, локализации и исправления ошибок в программе

Отладка

- "Выполнение программы в уме" (deskchecking).
- Вставка операторов протоколирования (печати) промежуточных результатов (logging).
- Пошаговое выполнение программы (single-step running).
- Выполнение с заказанными остановками (breakpoints), анализом трасс (traces) или состояний памяти - дампов (dump).
- Реверсивное (обратное) выполнение (reversible execution).

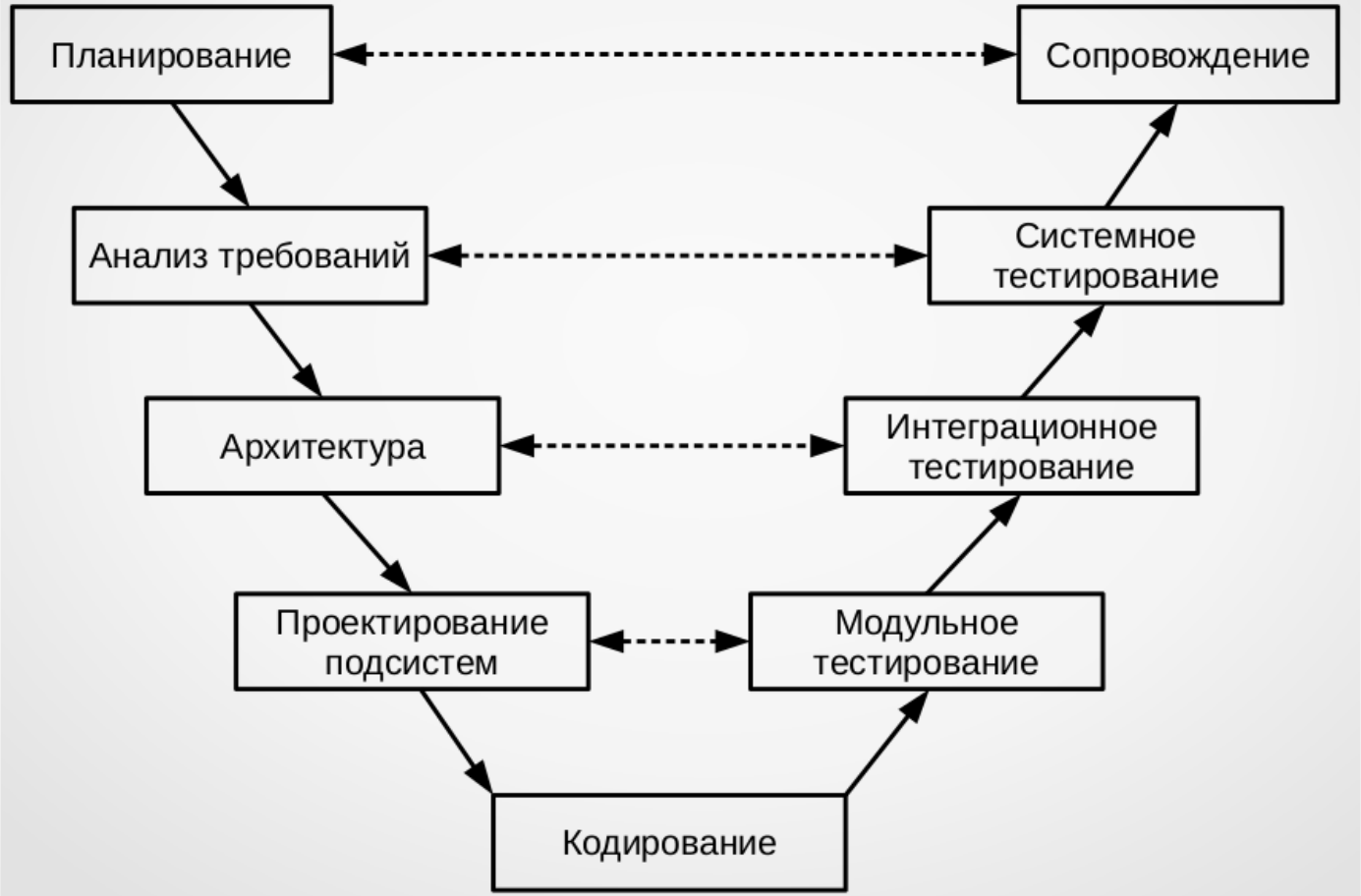
Схема тестирования



Когда тестировать?

- Тестирование нового кода
  - наличие заявленной функциональности
  - отсутствие ошибок
  - возможность интеграции в проект
- Тестирование изменений
  - изменение может быть неправильным
  - изменение может отобразить скрытые ошибки
  - изменение может привести к потере работоспособности
- Тестирование демонстрацией
  - эффект демо: "Вчера все работало, а тут …"
- Тестирование использованием
  - реальное применение разработанного ПО

Когда тестировать?



Как тестировать?

- Запуск всей программной системы
  - дорогостоящая операция
  - неизвестно где ошибка
- Запуск компоненты системы (приложения)
  - надо много окружения
  - неизвестно где ошибка
- Запуск модуля приложения
  - сам модуль не запустится — нужен драйвер
  - модуль использует внутренние форматы данных

Чем тестировать?

- Каждый тест — отдельная программа
  - как это все запускать?
- Скрипты запуска тестовой программы с параметрами
  - что будет если надо добавить тест?
  - А если надо временно отключить группу тестов
- Система контроля и автоматического запуска тестов
  - много магии
  - Требуется «управление» не только разработкой, но и процессом тестирования

Проблемы тестирования

- Тестирование программы на всех входных значениях невозможно.
- Невозможно тестирование и на всех путях.
- Надо отбирать конечный набор тестов, позволяющий проверить программу на основе наших интуитивных представлений
- В теории алгоритмов доказано, что не существует общего метода для решения этого вопроса, а также вопроса, достигнет ли программа на данном тесте заранее фиксированного оператора.

- **Основная проблема тестирования** — определение достаточности множества тестов для истинности вывода о правильности реализации программы, а также нахождения множества тестов, обладающего этим свойством.

Этапы тестирования

1) Создание тестового набора (test suite) путем ручной разработки или автоматической генерации для конкретной среды тестирования (testing environment)
2) Прогон программы на тестах, управляемый тестовым монитором (test monitor, test driver) с получением протокола результатов тестирования (test log).
3) Оценка результатов выполнения программы на наборе тестов с целью принятия решения о продолжении или остановке тестирования.

Характеристики хорошего теста

- Цель тестирования выявление ошибок
- Ошибка — отклонение от эталона
- Варианты эталонов:
  - неформальное представление того, «как ПО должно работать»;
  - формальная техническая спецификация;
  - набор тестовых примеров;
  - корректные результаты работы программы;
  - другая (априори корректная) реализация той же исходной спецификации.

Характеристики хорошего теста

- достижение (Reachibility) — тест должен выполнить место в исходном коде, где присутствует программная ошибка;
- повреждение (Corruption) — при выполнении ошибки состояние программы должно испортиться с появлением сбоя;
- распространение (Propagation) — сбой должен распространиться дальше и вызвать неудачу в работе программы.

Методы разработки тестов

Критерий	Черный ящик	Белый ящик
Основной уровень применимости	Приемочное тестирование	Модульное тестирование
Ответственный	Независимый тестировщик	Разработчик
Знание программирования	Необязательно	Необходимо
Знание реализации	Необязательно	Необходимо
Знание сценариев использования	Необходимо	Необязательно
Основа тестовых сценариев	Спецификации	Код

Методы разработки тестов

- Серый Ящик
  - Комбинация черного и белого ящиков
  - Знаем частично или полностью внутреннее устройство тестируемого объекта
  - Тестировщик находится на уровне пользователя
  - Пример: Зная особенности реализации модуля, создаем тестовые сценарии пользовательского уровня, которые покрывают потенциально проблемную область
  - Основная область применения: интеграционное тестирование

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.

Позитивные и негативные тесты

- Негативные тесты:
  - тесты для проверки устойчивости ПО к негативным входным данным;
  - тесты на проверку устойчивости ПО к ошибкам пользователя;
  - тесты на то, что у программы нет неожиданных побочных эффектов;
  - тестирование с целью «сломаем это!».

Классы эквивалентности и граничные значения

- Если от выполнения двух тестов ожидается один и тот же результат, они считаются эквивалентными.
- Группа тестов представляет собой класс эквивалентности, если выполняются следующие условия
  - Все тесты предназначены для выявления одной и той же ошибки.
  - Если один из тестов выявит ошибку, остальные, скорее всего, тоже это сделают.
  - Если один из тестов не выявит ошибки, остальные, скорее всего, тоже этого не сделают.

Классы эквивалентности и граничные значения

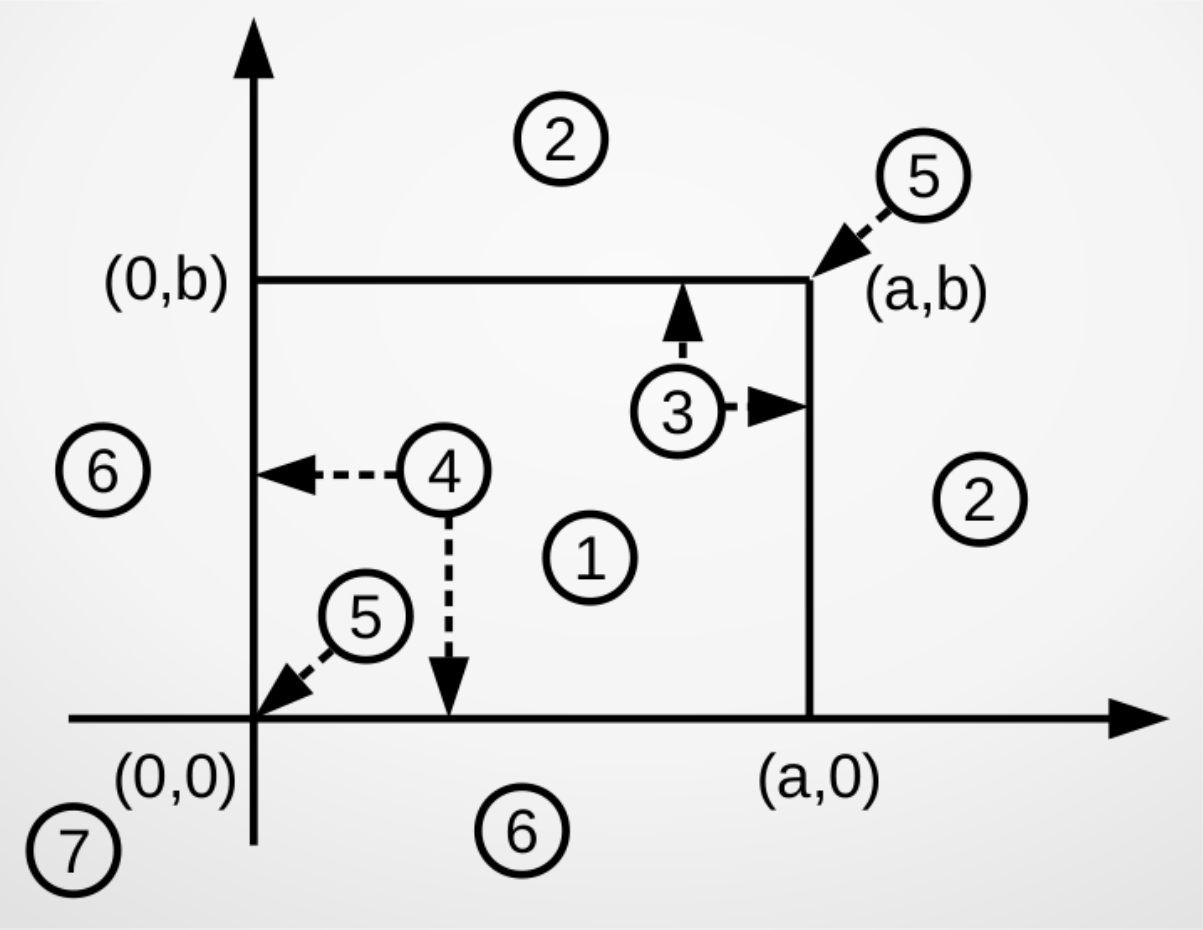
- Практические критерии определения классов эквивалентности:
  - Тесты включают значения одних и тех же входных данных.
  - Для их проведения выполняются одни и те же операции программы.
  - В результате всех тестов формируются значения одних и тех же выходных данных.
  - Либо ни один из тестов не вызывает выполнения блока обработки ошибок программы, либо выполнение этого блока вызывается всеми тестами группы.

Классы эквивалентности и граничные значения

- Рекомендации для поиска классов эквивалентности:
  - Не забывайте о классах, охватывающих заведомо неверные или недопустимые входные данные.
  - Организуйте формируемый перечень классов в виде таблицы или плана.
  - Определите диапазоны числовых значений.
  - Для полей или параметров, принимающих фиксированные перечни значений, выясните, какие из значений входят в перечень.
  - Проанализируйте возможные результаты выбора из списков и меню.
  - Поищите переменные, значения которых должны быть равными.
  - Поищите классы значений, зависящих от времени.
  - Выявите группы переменных, совместно участвующих в определенных вычислениях, результат которых ограничивается конкретным набором или диапазоном значений.
  - Посмотрите, на какие действия программа отвечает эквивалентными событиями.
  - Продумайте варианты операционного окружения.

Классы эквивалентности и граничные значения

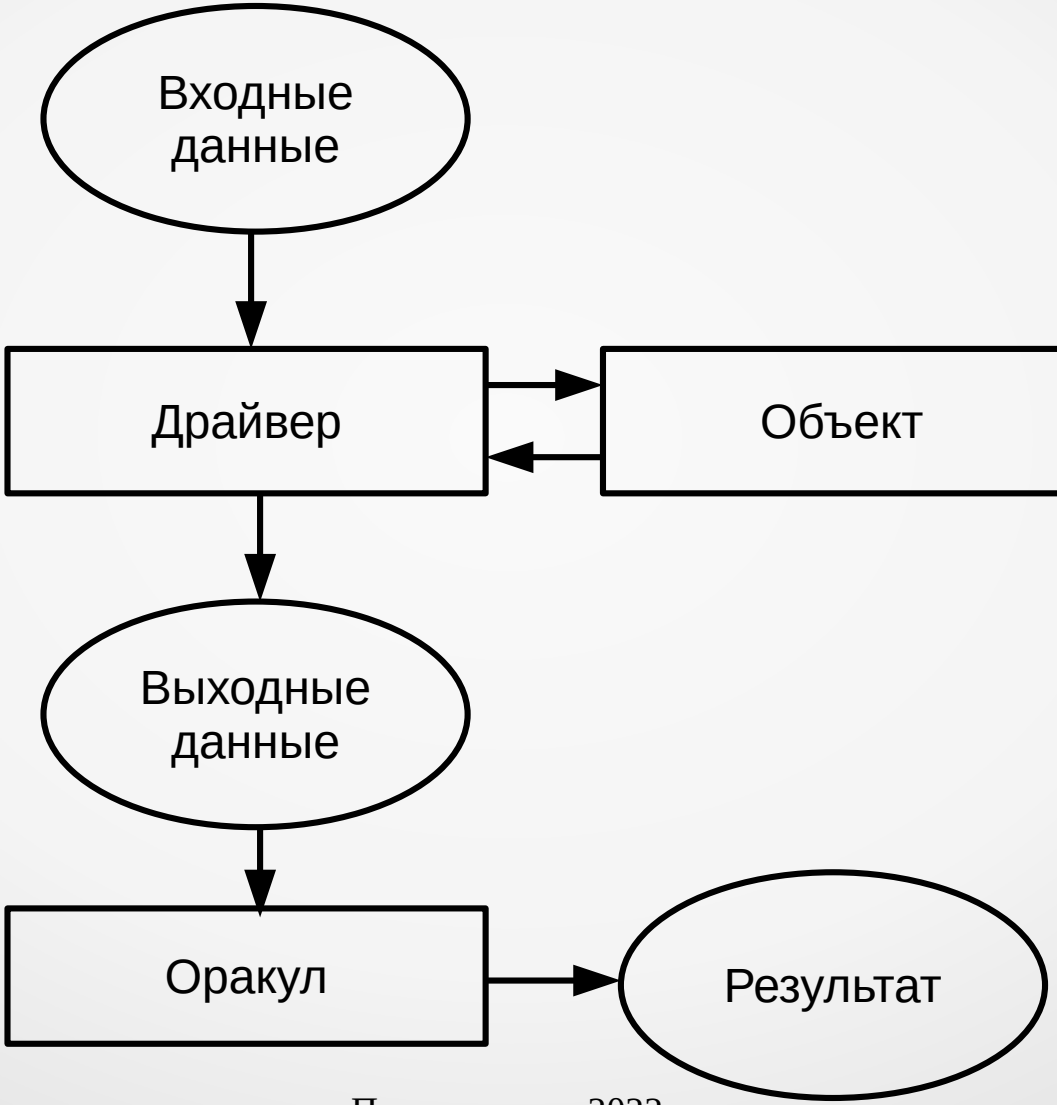
- Пример: функция от двух аргументов



Модульное тестирование

- Структурное тестирование (белый ящик)
- Поиск дефектов
  - алгоритмические ошибки
  - ошибки кодирования алгоритмов
  - выполнение условных и циклических операторов
  - использование переменных и ресурсов
- Использование заглушек, эмуляторов и других вспомогательных инструментов, заменяющих полностью или частично реальные компоненты ПО

Модульное тестирование



Интеграционное тестирование

- Смешанная модель разработки (Серый ящик)
- Поиск ошибок
  - трактовка данных
  - реализация интерфейса взаимодействия
  - совместимость
- Схема интеграции
  - восходящая (от малого к большому)
  - нисходящая (подключение к основному модулю)
  - смешанная (сборка блоков и подключение к основному модулю)

Интеграционное тестирование

	Восходящее	Нисходящее
Преимущества	• Возможность ранней проверки корректности низкоуровневого поведения • Не требуется написание заглушек • Просто определить требования ко входам/выходам модулей	• Возможность ранней проверки корректности высокоуровневого поведения • Модули могут добавляться по одному, независимо друг от друга • Не требуется разработка множества драйверов • Можно разрабатывать систему как в глубину, так и в ширину
Недостатки	• Отложенная проверка высокоуровневого поведения • Требуется разработка драйверов • При замене драйвера на модуль высокого уровня может произойти "мини-Большой Взрыв" числа найденных ошибок	• Отложенная проверка низкоуровневого поведения • Требуется разработка "заглушек" • Крайне сложно корректно сформулировать требования ко входам/выходам частичной системы

Системное тестирование

- Функциональное тестирование (черный ящик)
- Проверка на соответствие стандартам
- Оценка характеристик качества ПО
  - устойчивость
  - надежность
  - безопасность
  - производительность
- Приемочное тестирование
- Демонстрация
- Тестирование пользователями

Переходы между состояниями

- Переход м/у состояниями в следствие действий пользователя или внешнего окружения
- Переходов может быть очень много!
  - Протестируйте все наиболее вероятные последовательности действий пользователей.
  - Если можно предположить, что действия пользователя в одном режиме могут воздействовать на представление данных или набор предоставляемых программой возможностей в другом режиме, протестируйте эту зависимость.
  - Кроме проведения самых необходимых тестов — из тех, что описаны выше, — стоит поработать с программой в произвольном режиме, случайным образом выбирая путь ее выполнения.
- Построение схем меню и форм

Регрессионное тестирование

- Исправление ошибки зачастую порождает новые ошибки
  - Проверка исправления ошибки
  - Поиск связанных ошибок
  - Проверка остальной части программы
- Использование библиотеки регрессионных тестов
  - Удаление тестов эквивалентных другим тестам библиотеки.
  - Уменьшение количества тестов, объектом которых является уже исправленная ошибка
  - Комбинации тестов
  - Автоматизация тестирования
  - Периодическое выполнение

Анализ чувствительности

- Поиск тестов, вызывающих наибольшие возмущения
  - Общее представление о поведении функции, на основе ее значений для ряда параметров, располагающихся вдоль всей области определения
  - Поиск участков области определения, на которых небольшие изменения аргументов вызывают значительные скачки результирующих значений
- Оценка погрешностей получаемых значений (операции с плавающей запятой)
- Использование теории вероятности для тестирования мат.функций

Нагрузочные испытания

- Нагрузочное испытание — один из видов пограничного тестирования
- Проверка максимально допустимых ресурсов (размер файлов, количество соединений, устройств и т.п.)
- Условия гонок
  - Проверка работы в нагруженном и ненагруженном режиме
  - Воздействия на приложение во время работы
  - Проверка на быстрых и медленных машинах

Пользовательское тестирование

- Пользовательское тестирование
- Альфа-тестирование
- Бета-тестирование
- Произвольное использование (bag bush)
- Тестирование экспертом
- Парное тестирование
- Использование внутри компании (Eat your own dogfood)
- Тестирование локализации