Подходы к тестированию

Технологии разработки программного обеспечения

Виноградова М.В.
МГТУ им. Н.Э. Баумана
Кафедра СОИУ (ИУ5)

Тестирование ПО

- Процесс выполнения ПО с целью обнаружения ошибок.

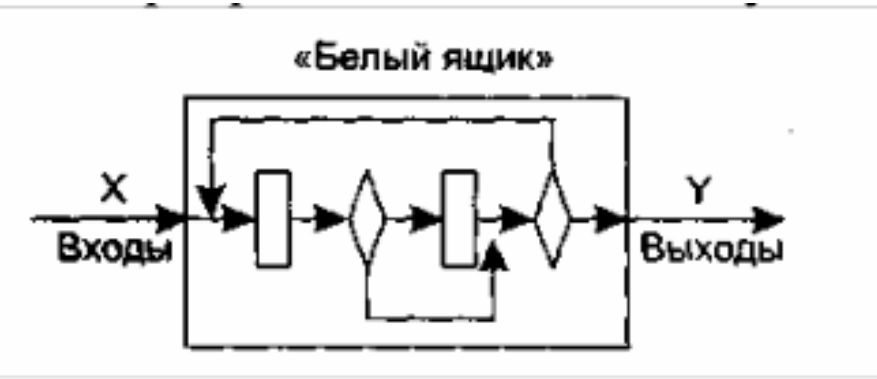
- Шаги процесса задаются тестами.
- Тест (Тестовый вариант):
  - набор исходных данных + условия запуска ПО,
  - набор ожидаемых результатов работы ПО.

- Хороший тестовый вариант
  - с большой вероятностью обнаружение новой ошибки.
- Успешный тест – нашел новую ошибку.

- Цель разработки тестов
  - систематическое обнаружение различных классов ошибок при min затратах времени и стоимости.

Структурное тестирование

- Объект тестирования:
  - внутреннее поведение ПО;
  - корректность построения всех элементов ПО и правильность их взаимодействия;
- Анализ управляющих и информационных потоков:
  - Метод базового пути;
  - Тестирование условий;
  - Тестирование потоков данных;
  - Тестирование циклов.
- Характеризуется степенью покрытия логики (исходного кода) ПО.



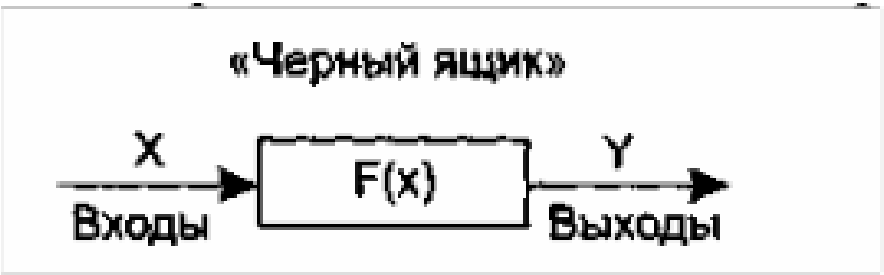
Особенности тестирования «Белого ящика»

- Анализ управляющей структуры ПО.
- ПО полностью проверено <= исчерпывающее тестирование путей (маршрутов) ее графа управления.

- Тестовые варианты:
  - Гарантируется проверка всех независимых маршрутов ПО;
  - Проходятся ветви True, False для всех логических решений.
  - Выполняются все циклы (в пределах их границ).
  - Анализируется правильность внутренних структур данных.

Функциональное тестирование

- Демонстрирует:
  - Как выполняются функции ПО;
  - Как принимаются исходные данные;
  - Как вырабатываются результаты;
  - Как сохраняется целостность внешней информации.
- Рассматриваются системные характеристики ПО. Игнорируется внутренняя логическая структура.
- Методы:
  - Разбиение на классы эквивалентности;
  - Анализ граничных значений;
  - Диаграммы причинно-следственных связей.

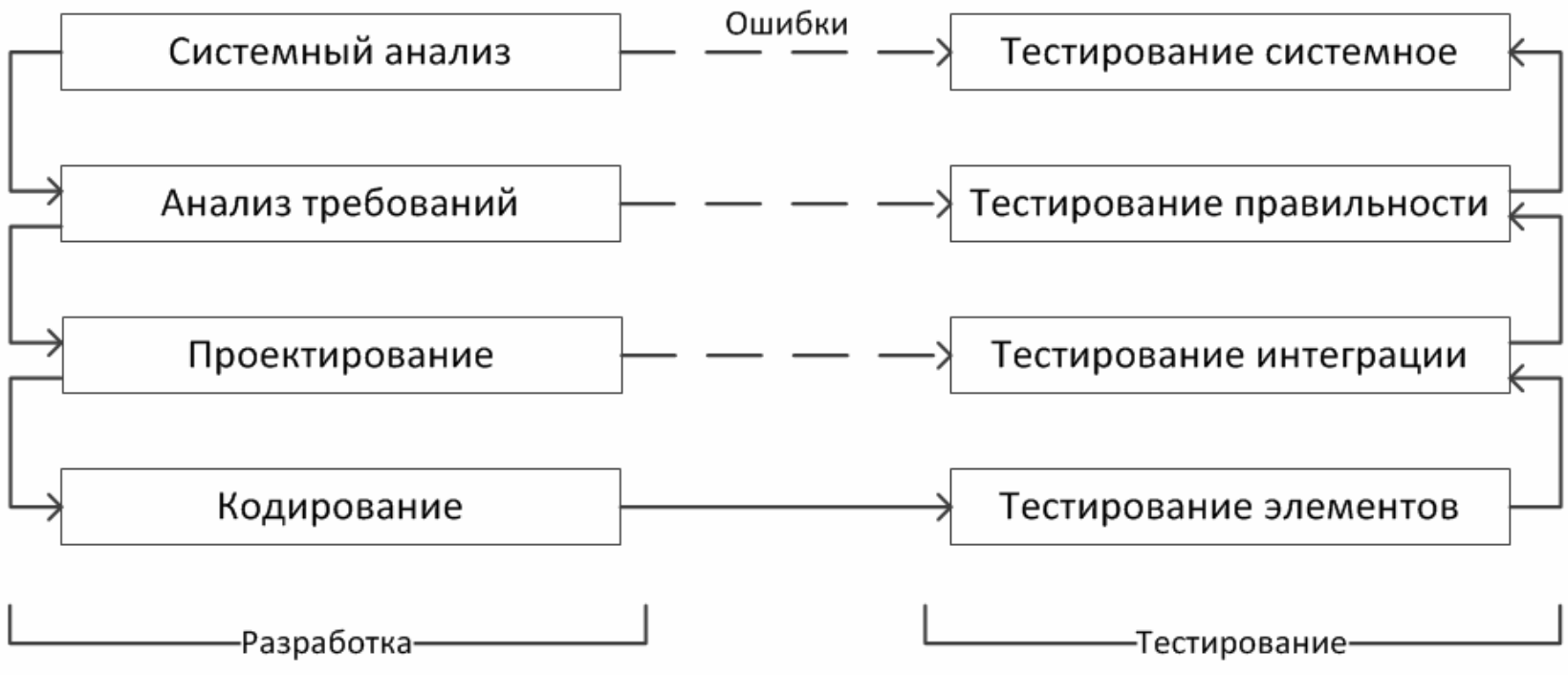


Процесс тестирования

- Тестирование — всеобъемлющая операция, которая применяется ко всем стадиям жизненного цикла.
- Процесс развивается по спирали.

- Принцип окончания тестирования:
  - В случае, когда вероятность безотказной работы центрального процессора (ЦП) с программным обеспечением (ПО) в течение 1000 часов больше или равно 0,995, то с вероятностью 95% в программе отсутствуют ошибки.

Тестирование в процессе разработки ПО



Тестирование элементов

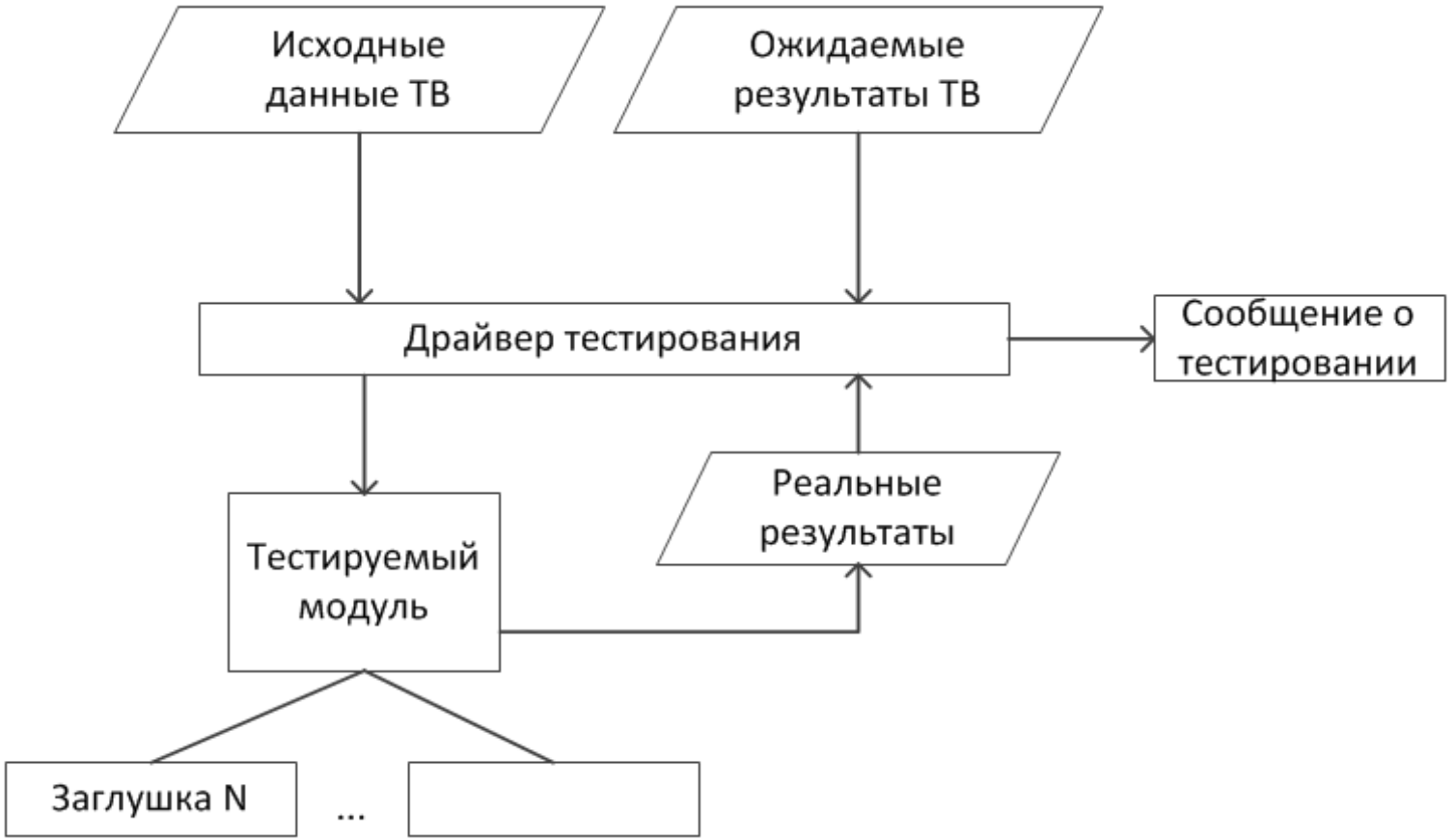
- Индивидуальная проверка любого модуля (единицы проектирования);
- «Белый ящик»;
- Параллельное тестирование модулей.
- Тестирование элементов дополняет этап кодирования.

- Объекты тестирования:
  - Интерфейс модуля
  - Внутренняя структура данных
  - Независимые пути
  - Пути обработки ошибок
  - Граничные условия

Объект тестирования	**Цель тестирования**
Интерфейс модуля	Проверка правильности вводов / выводов данных (иначе прочие тесты бесполезны)
Внутренняя структура данных	Проверка целостности сохраняемых данных
Пути обработки ошибок	Выявление следующих ошибок: – Непонятные сообщения; – Сообщение не соответствует ошибке; – Системное исключение до обработки в модуле; – Некорректная обработка ошибки; – По сообщению не понятна основная причина.
Граничные условия	Часто встречаются в следующих ситуациях: – При обработке n-го элемента n-го массива; – m итерация из m итераций цикла; – При минимальном/максимальном значении.

Объект тестирования	**Цель тестирования**
Независимые пути	Однократное выполнение всех операторов. **Виды ошибок вычислений:** – Неправильный приоритет арифметической операции; – Смешанная форма операций; – Некорректная инициализация; – Несогласованное представление точности; – Некорректное символьное представление выражения. **Ошибки сравнения и потоков управления:** – Сравнение различных типов данных; – Некорректные логические операции и приоритет; – Ожидание равенства при несоответствующей точности; – Некорректное сравнение переменных; – Неправильное прекращение цикла; – Отказ в выходе при отклонении итерации; – Неправильное изменение переменных цикла.

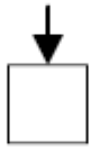
Программная среда для тестирования модуля

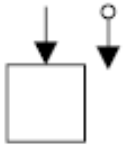


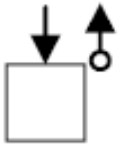
Заглушки

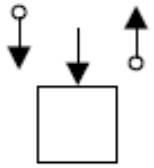
- **Заглушка** моделирует поведение отсутствующего компонента.
- Особенности заглушек:
  - замещают модули, вызываемые тестируемым;
  - реализуют интерфейс подчинённого модуля;
  - минимальная обработка данных;
  - имитация приёма / возврата данных.

Типы заглушек

А) Отображает сообщение; 

Б) Отображает входные данные; 

В) Возвращает величину из таблицы; 

Г) Возвращает данные из таблицы по ключу. 

Типы драйверов

А) Вызывает подчинённые модули;

Б) Посылает данные из внутренней таблицы в подчинённый модуль;

В) Отображает действие из подчинённого модуля;

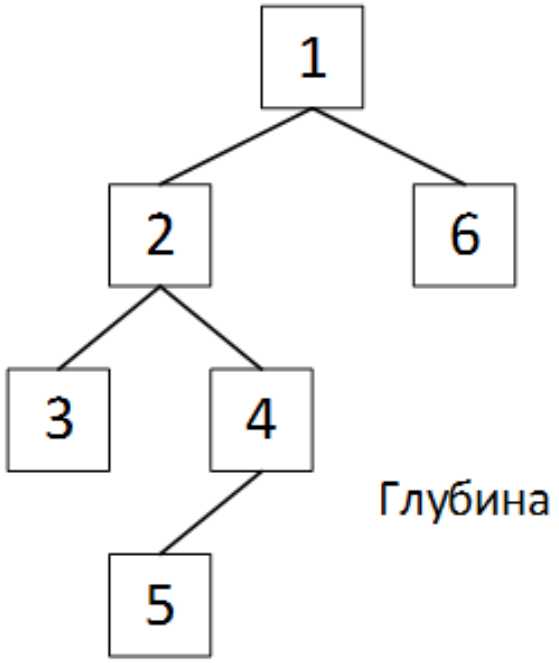
Г) Сочетание действий Б и В.

Тестирование интеграции

- Тестирование сборки протестированных модулей в программную систему;
- «Чёрный ящик»;
- Поиск ошибок интерфейса:
  - Потеря данных;
  - Отсутствие в модуле ссылки;
  - Объединение подфункций не дают функцию;
  - Объединение допустимых неточностей больше допустимых;
  - Проблемы использования глобальной структуры данных;
  - Неблагоприятное влияние модулей друг на друга

Нисходящее тестирование интеграции - шаги

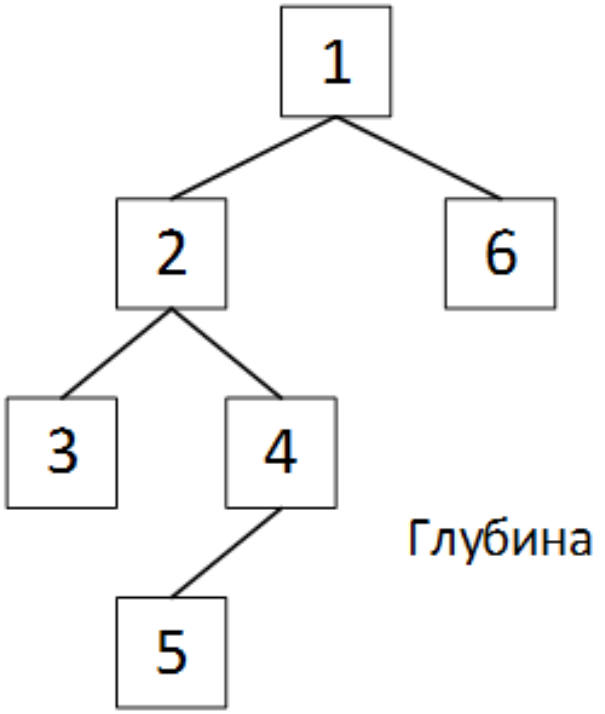
1. Тестовым драйвером будет главный модуль;
2. Вместо подчинённых модулей используются заглушки;
3. Проводим тесты;
4. Заменяем одну заглушку на реализованный модуль (либо в глубину, либо в ширину);
5. Повторяем все тесты.
6. Если заглушек больше нет, то один из подчиненных модулей станет тестовым драйвером.



Нисходящее тестирование интеграции

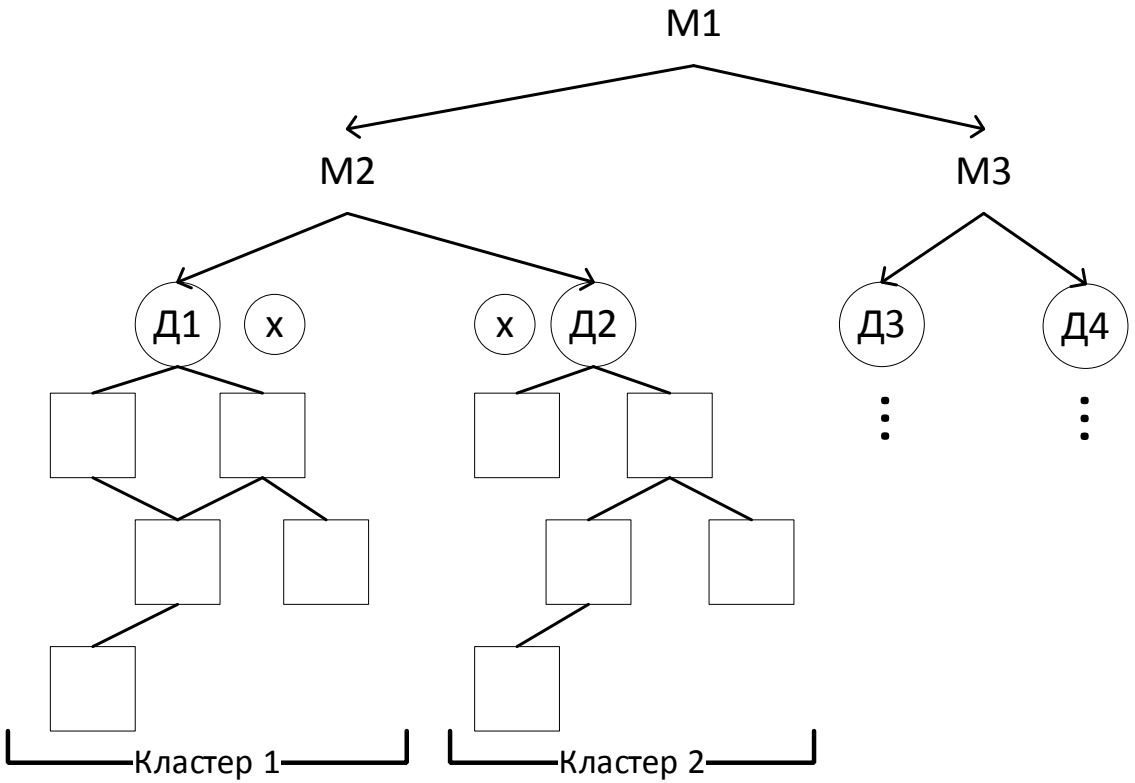
- От верхних к нижним по управляющей иерархии;
- Поиск в глубину и в ширину

- (+) Подчинённые главного модуля – заглушки;
- (+) Раннее тестирование главных управляющих модулей;
- (-) Заглушки приводят к увеличению затрат.



Восходящее тестирование интеграции - шаги

1. Объединяем модули нижнего уровня в кластер (по функции);
2. Создаём драйвер для кластера;
3. Тестируем кластер;
4. Заменяем драйвер на объединение кластеров (восходящее).



Восходящее тестирование интеграции

- Сборка и тестирование системы с атомарными модулями (нижний уровень);
- Подключение модулей снизу-вверх;
- Нет заглушек.

- (+) Упрощаются тестовые варианты;
- (+) Нет заглушек;
- (-) Система как объект может быть оценена только в конце.

Комбинация восходящего и нисходящего тестирования

- Верхние уровни – нисходящее тестирование.
- Нижние уровни – восходящее тестирование.
- **Критические модули** (раннее и регрессионное тестирование, которое бывает полным и частным).
- Признаки критических модулей:
  - Реализует несколько требований к ПО;
  - Высокий уровень управления (верхние уровни иерархии);
  - Высокая сложность (максимальная цикломатическая сложность равна 10);
  - Имеет требования к производительности работы.

Тестирование правильности

- После интеграционного тестирования;
- Проверяет все функциональные и другие требования к ПО;
- Только «чёрный ящик».
- Из тестирования правильности следует список недостатков, который приводит к изменению сроков разработки.

- Тестирование конфигурации
- Альфа – тестирование
- Бета – тестирование

Проверка конфигурации (наличие, полнота, непротиворечивость)

- Системная спецификация;
- План программного проекта;
- Спецификация требований (макет);
- Предварительное руководство пользователя;
- Спецификация проектирования;
- Листинг;
- Программа и методика испытаний (ПМИ), тестовые варианты (ТВ) и полученные результаты;
- Руководство по работе и инсталляции;
- Исполняемый код программного обеспечения;
- Описание базы данных (БД);
- Руководство пользователя по настройке;
- Документация сопровождения (отчёты о проблемах и т.д.)
- Стандарты разработки, которые должны быть разработаны, учтены и детализированы.

- **Альфа-тестирование**
- Альфа-тестирование выполняется заказчиком у разработчика.
- Разработчик фиксирует ошибки и проблемы.

- **Бета-тестирование**
- Бета-тестирование выполняется пользователем у заказчика.
- Формируется список проблем и отправляется разработчику.
- Приводит к изменению ПО.
- Бета-тестирование продолжается примерно 1 год.

Системное тестирование

- Происходит не только при разработке;
- Поиск причин при взаимных обвинениях;
- Доказательство правильности работы и объединения системных элементов.
- Виды:
  - Тестирование восстановления.
  - Тестирование безопасности.
  - Стрессовое тестирование.
  - Тестирование чувствительности.
  - Тестирование производительности.

Тестирование восстановления

- Восстановление в пределах времени (после отказа);
- Восстановление после сбоев;
- Автоматическое:
  - Проверка повторения инициализации;
  - Проверка восстановления данных;
  - Перезапуск;
  - Копирование контрольных точек.
- Ручное:
  - Проверка, что среднее время восстановления меньше, чем предельно допустимое .
- Отказоустойчивость
  - Отказ обработки не приводит к прекращению работы ПО.

Тестирование безопасности

- При тестировании безопасности происходит проверка фактической реакции системы защиты ПО при проникновении.

- **Методы:**
  - Взлом пароля;
  - Утилиты атаки;
  - Стресс системы (отказ);
  - Неверный ввод (отказ ведёт к восстановлению);
  - Просмотр открытых данных для поиска скрытых данных.
- Цель в том, чтобы стоимость проникновения была больше, чем стоимость информации.

Стрессовое тестирование

- Проверка поведения ПО при ненормальном повышении запросов на ресурсы системы (количество, частота, размер-объём):
  - Оперативная память, виртуальная оперативная память, максимальная скорость чтения/записи диска, увеличение частоты запросов.
- Главный вопрос: Когда система откажет?

- **Тестирование чувствительности**
  - При малом диапазоне корректных исходных данных возможен некорректный результат или снижение производительности. Их надо найти.

Тестирование производительности

- Для встроенного ПО и ПО реального времени важно время ответа:
  - Проверка скорости работы ПО;
  - На всех шагах тестирования;
  - Белый ящик используется для проверки производительности модуля;
  - Тестирование всей системы для оценки реального времени обработки.
- Необходимо специальное аппаратное обеспечение (АО) / программное обеспечение (ПО) (инструменты):
  - Скорость центрального процессора (ЦП);
  - Прерывания;
  - События.

Регрессионное тестирование

- Регрессионное тестирование является повторным.
- После изменения или объединения ПО.

Эволюция тестирования

- Блочное (модульное);
- Компонентное (несколько блоков);
- Комплексное (объединение компонентов);
- Под нагрузкой (масштабирование ПО) (расчётная нагрузка);
- Пиковыми нагрузками (Пропускная способность; способность корректировки отказов и восстановления);
- Комплексное под нагрузкой «прогон» (пиковое долгое время приводит к накоплению ошибок).