



Роль тестирования и обеспечения качества в жизненном цикле разработки ПО

**Зиберов
Александр**
ПИЖ-б-о-21-1

Что такое тестирование?

- Согласно ANSI / IEEE 1059, тестирование в программной инженерии – это процесс оценки программного продукта, позволяющий определить, соответствует ли текущий программный продукт требуемым условиям. Процесс тестирования включает в себя оценку характеристик программного продукта на соответствие требованиям с точки зрения отсутствующих требований, ошибок или дефектов, безопасности, надежности и производительности.
- Тестирование программного обеспечения – это метод проверки соответствия фактического программного продукта ожидаемым требованиям, чтобы убедиться, что продукт не содержит дефектов.





Для чего это всё?

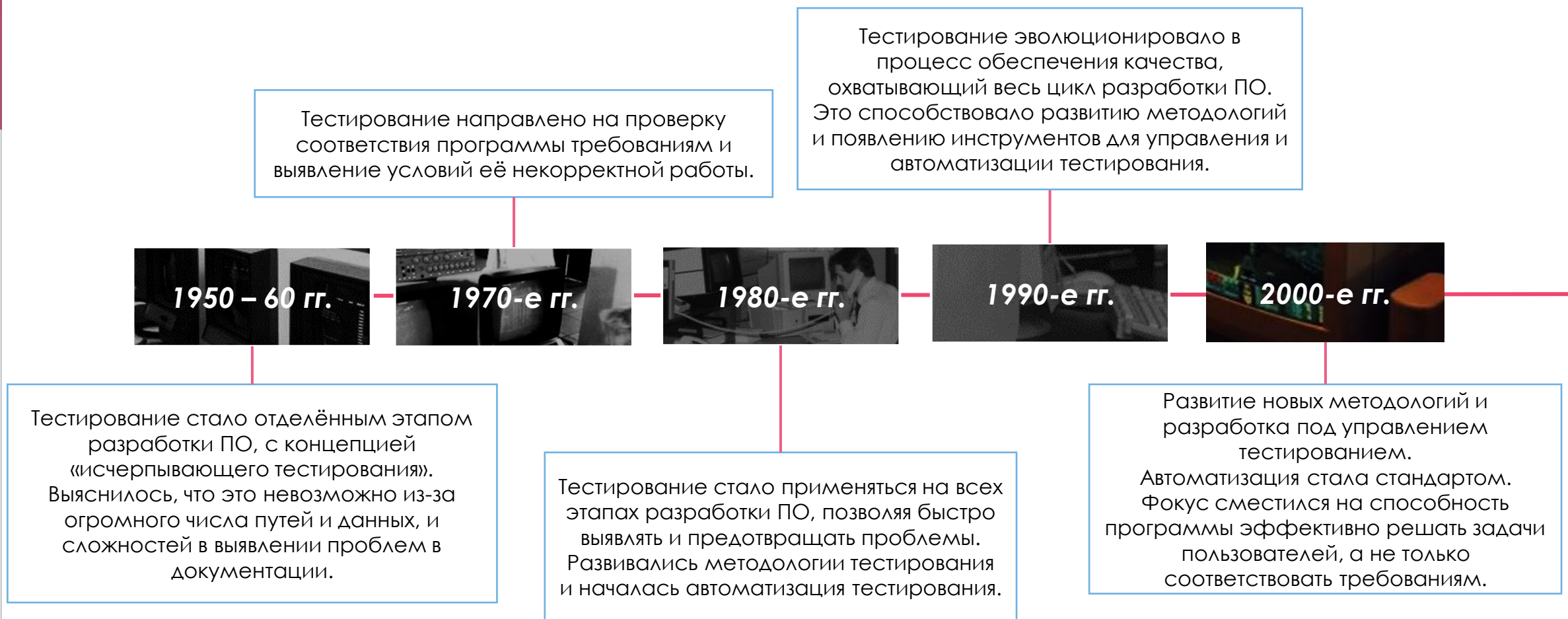
Цели тестирования

- Проверка, все ли указанные требования выполнены (например, по техническому заданию)
- Создание уверенности в уровне качества объекта тестирования (работает как задумано)
- Обнаружение и предотвращение отказов и дефектов
- Предоставление заинтересованным лицам достаточной информации
- Снижение уровня риска

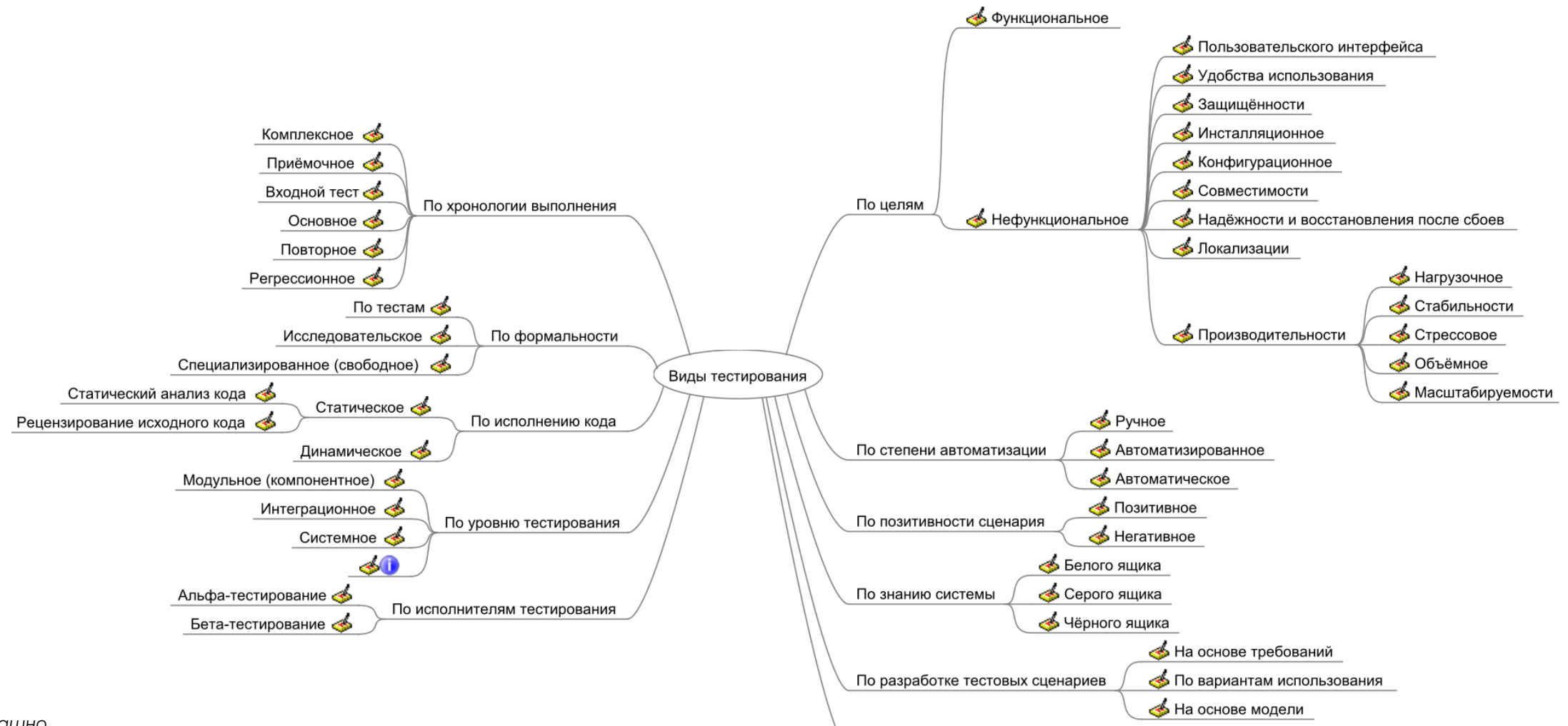
Преимущества тестирования

- Рентабельность (обнаружение и исправление ошибок на раннем этапе тестирования обходится дешевле)
- Безопасность (исключение рисков критических ошибок, например, утечки информации)
- Качество продукта (гарантия соответствия работы программы с требованиями)
- Удовлетворенность клиентов/пользователей

Краткая историческая справка

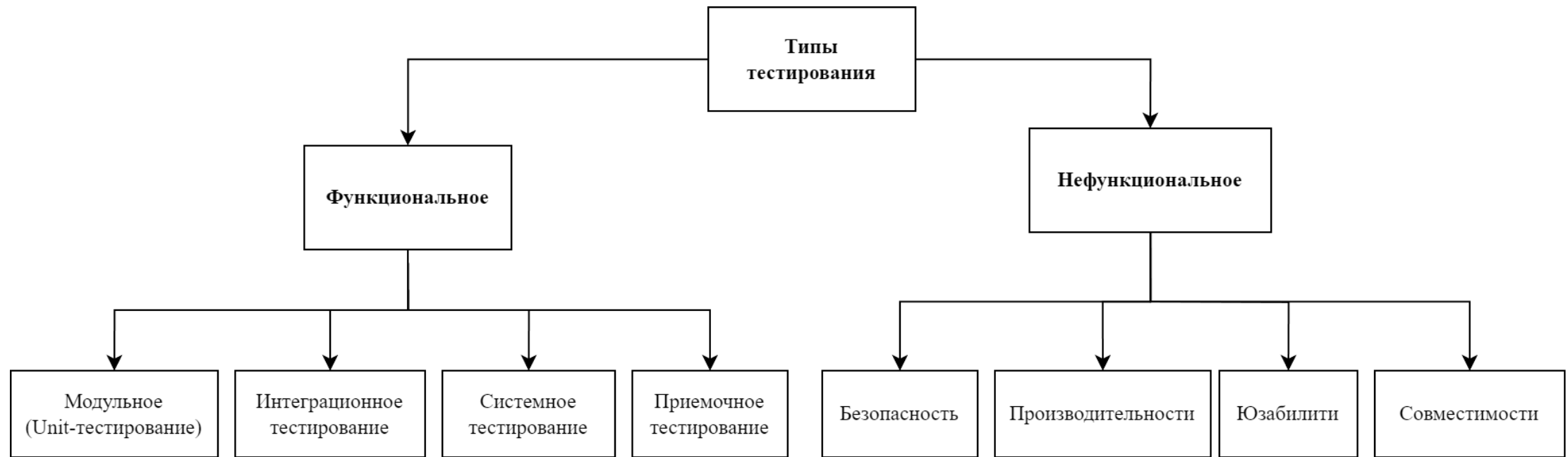


Классификация видов тестирования



Страшно...

Классификация типов тестирования по целям



Функциональное тестирование

Функциональное тестирование (Functional testing) – проводится для оценки соответствия компонента или системы функциональным требованиям.

- Фокусируется на выполнении программой **того, что от неё ожидается**, и что все её **функции работают корректно**.
- **Основной подход в** – создание тест-кейсов* на основе функциональных требований. Эти тест-кейсы затем используются для проверки каждой функции или особенности программы на соответствие заданным критериям.

*Тест-кейс (Test Case) – набор условий или переменных для проверки корректности работы системы. Тест-кейсы включают описание входных данных, шаги выполнения и ожидаемые результаты.



Этапы проведения тестирования:

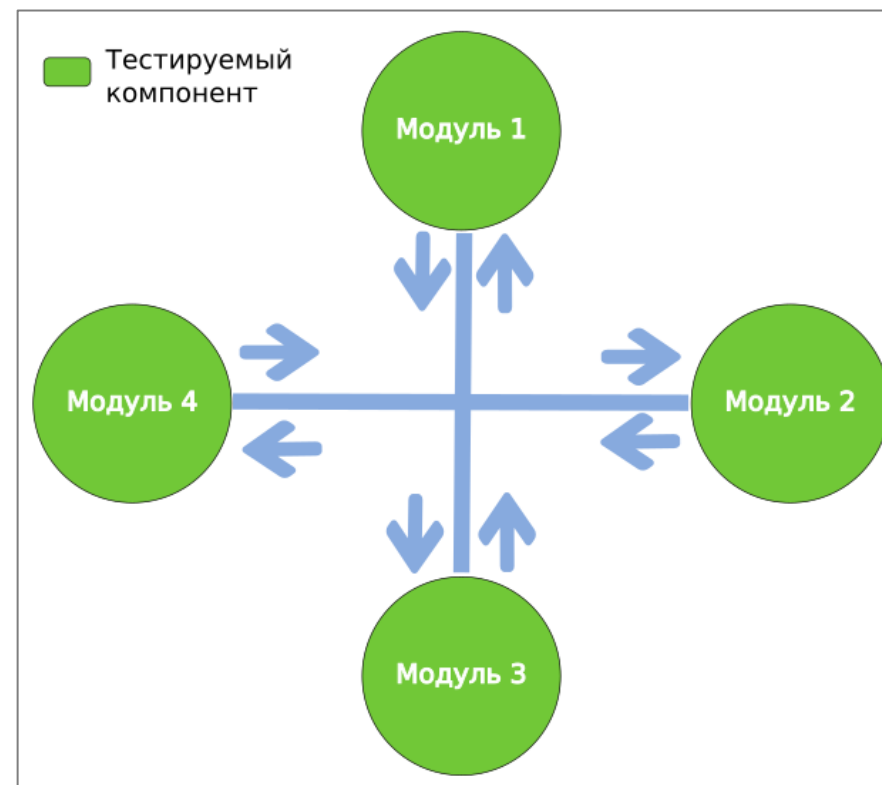
- 1) **Определение входных данных для тестирования:** выбор функций и условий для тестирования.
- 2) **Определение ожидаемых результатов:** создание входных данных и ожиданий по спецификациям.
- 3) **Выполнение тестовых сценариев:** запуск тестов и запись результатов.
- 4) **Сравнение фактических и ожидаемых результатов:** анализ отклонений для проверки корректности работы системы.

Модульное тестирование

Модульное тестирование – процесс проверки отдельных компонентов* (модулей) ПО на корректность их работы.

- **Цель тестирования модуля** – выявление ошибок и оценка готовности системы к следующему этапу разработки и тестирования.
- Проверяются отдельные функции и методы.
- Фокусируется на функциональности и корректности кода на уровне отдельных функций.
- На этом уровне проще всего обнаружить дефекты, связанные с алгоритмическими ошибками.

* В некоторых источниках отдельно упоминается понятие «компонентное тестирование». Практически, является тем же самым, разница лишь в масштабе и количестве тестируемых компонентов.



Модульное тестирование. Продолжение

Пример для операции сложения:

- 1) Создаем тестовый сценарий, который вызывает функцию сложения с двумя входными аргументами.
- 2) Результат сложения **двух положительных чисел правильный**.
- 3) Результат сложения **положительного и отрицательного чисел** правильный.
- 4) Результат сложения **отрицательных чисел** правильный.
- 5) Результат сложения **нуля с положительным числом** равен **положительному числу**.
- 6) Результат сложения **нуля с отрицательным числом** равен **отрицательному числу**.
- 7) Результат сложения **положительного числа с нулем** равен **положительному числу**.
- 8) Результат сложения **отрицательного числа с нулем** равен **отрицательному числу**.

Преимущества

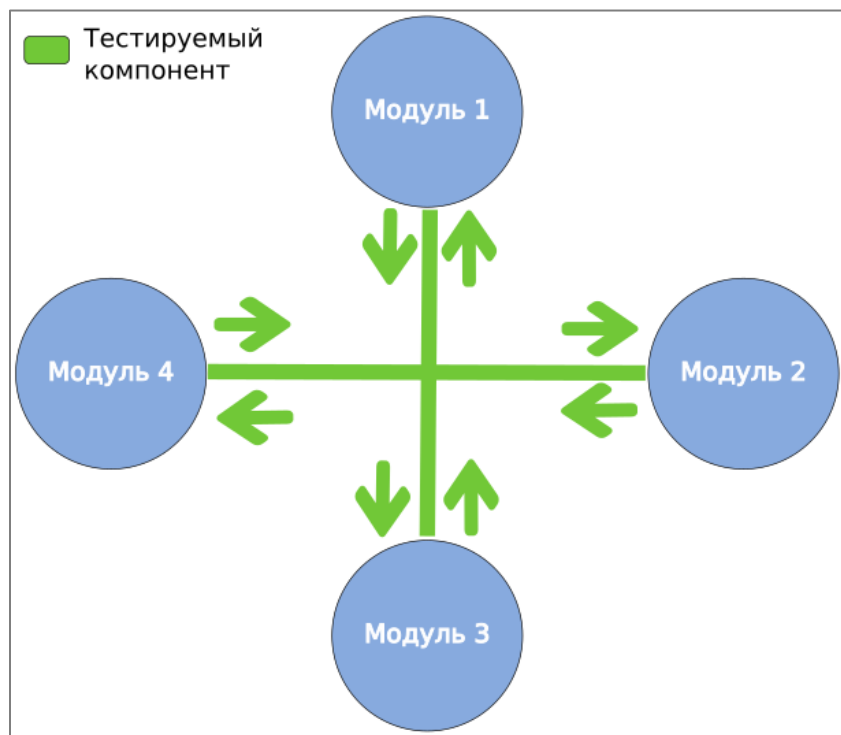
- + Позволяет выявить дефекты на ранних этапах разработки.
- + Упрощает процесс отладки и поиска ошибок.
- + Обеспечивает лучшую структурированность кода.
- + Снижает затраты на исправление ошибок в долгосрочной перспективе.

Недостатки

- Не гарантирует корректную работу взаимодействия между компонентами.
- Требуется дополнительных усилий на написание и поддержание тестового кода.
- Может привести к созданию избыточного количества тестов, особенно при неправильном планировании.

Интеграционное тестирование

Интеграционное тестирование – метод тестирования ПО, который фокусируется на проверке взаимодействия и обмена данными между различными компонентами или модулями программного приложения.



- Проводится после модульного тестирования и перед тестированием системы.
- Целью является выявление дефектов взаимодействия между программными модулями при их интеграции.
- Проверка обеспечения связности и согласованности между модулями.
- Подтверждение функциональности системы в целом перед переходом к финальному тестированию.

Интеграционное тестирование. Продолжение

Стратегии в интеграционном тестировании

подход Большого взрыва

Большинство разработанных модулей соединяются вместе, образуя либо всю необходимую систему либо её большую часть.

инкрементальный подход

Тестирование выполняется путем соединения двух или более логически связанных модулей.

Пример интеграционного тестирования:

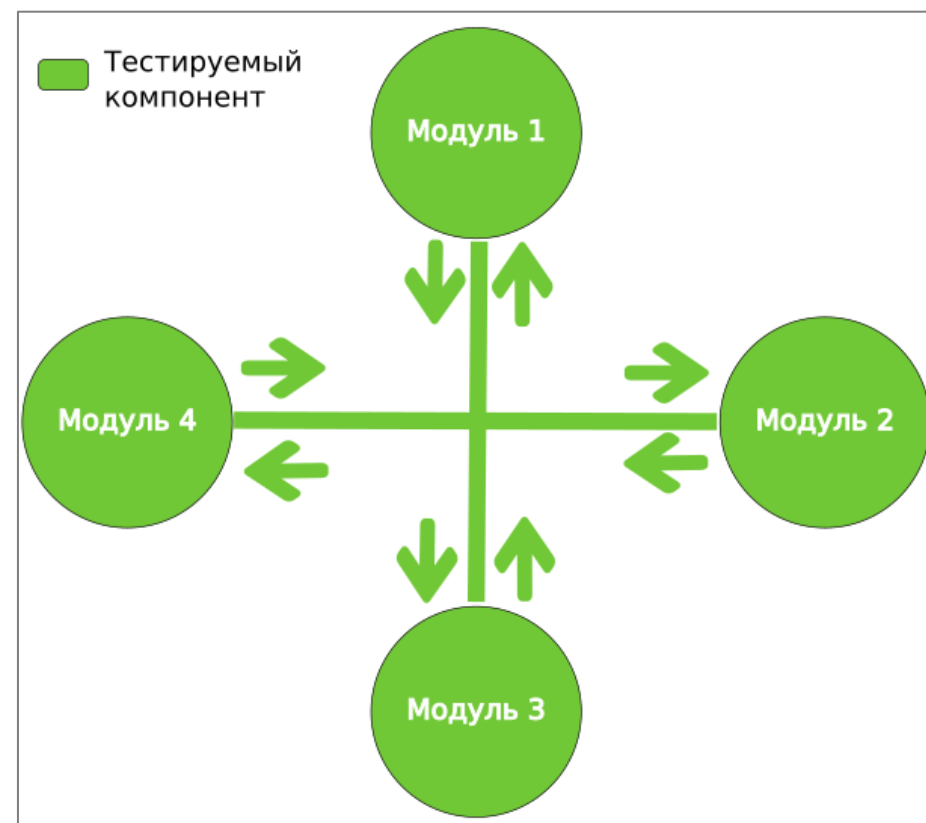
В веб-приложении есть **два модуля: модуль аутентификации и профиля пользователя**. Интеграционное тестирование проверяет взаимодействие этих модулей на соответствие сценария использования приложения.

Например, пользователь регистрируется через модуль аутентификации, и проверяется, что модуль профиля корректно получает и отображает данные о новом пользователе. Затем пользователь обновляет информацию в профиле, и проверяется, что обновленные данные сохраняются и правильно отображаются при повторной аутентификации.

Системное и приемочное тестирования

Системное тестирование – процесс проверки полной интегрированной системы на соответствие требованиям. Оно включает в себя взаимодействие всех компонентов системы и дополнительных элементов, таких как базы данных, сетевые устройства и другие внешние системы.

Приёмочное тестирование – готовая система проверяется на соответствие техническим и бизнес требованиям.



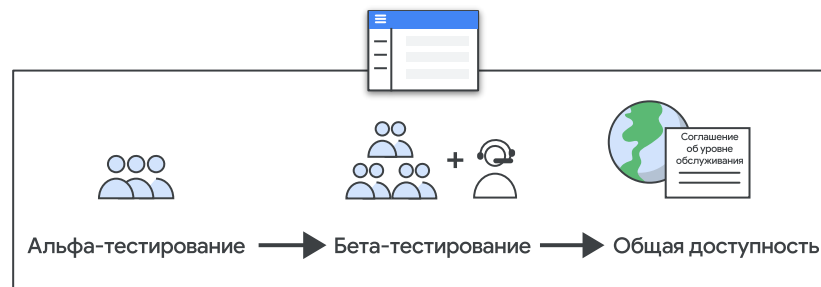
Дополнительные виды тестирования

Дополнительные виды тестирования, не относящиеся напрямую к категориям, но играющие значительную роль при разработке ПО:

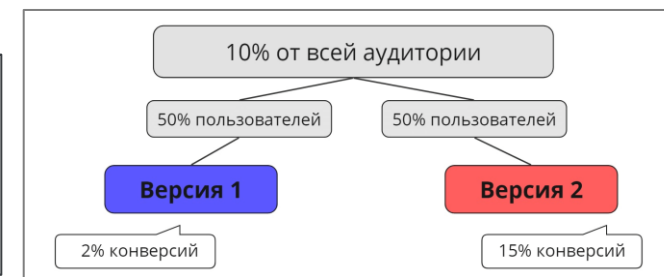
Smoke-тестирование – вид тестирования, который выполняется для быстрой проверки основных функциональных возможностей приложения или системы после каждого крупного изменения или перед выпуском новой версии.



Альфа- и бета- тестирование – эксплуатационное тестирование пользователями/заказчиками или независимой командой тестирования разрабатываемого продукта.



А/Б-тестирование (AB testing) – метод исследования для оценки эффективности двух вариантов одного элемента/функции.



Нефункциональное тестирование





МЕТОДЫ И СПОСОБЫ ТЕСТИРОВАНИЯ

Ручное тестирование

Ручное тестирование – выполнение тестов без использования автоматизированных средств.

Ситуации, когда необходимо ручное тестирование:

- ✓ **Начальная стадия разработки:** Частые изменения в начале разработки легче тестировать вручную
- ✓ **Краткосрочные и небольшие проекты:** Быстрее и дешевле внедряется, чем автоматизация.
- ✓ **Тестирование удобства использования:** Автоматизация не может полностью имитировать непредсказуемое поведение пользователя.
- ✓ **Интуитивное и исследовательское тестирование:** Лучше выполняются вручную из-за зависимости от реального взаимодействия с продуктом.
- ✓ **Работа с физическими устройствами:** Тестирование физических устройств требует гибкости, которую обеспечивает ручное тестирование.



Ручное тестирование. Процесс и недостатки



Шаги проведения ручного тестирования:

- 1) Определение целей, создание плана и тестовых сценариев.
- 2) Разработка тест-кейсов - создание инструкций с шагами и ожидаемыми результатами.
- 3) Последовательное выполнение тест-кейсов
- 4) Фиксация результатов, включая ошибки.
- 5) Анализ результатов для выявления проблем и оценки качества.
- 6) Передача ошибок разработчикам и использование результатов для улучшения процесса.

Преимущества

- + Низкая стоимость эксплуатации, так как не используются программные инструменты
- + Незначительные изменения могут быть исследованы сразу, без написания кода и его исполнения.
- + Возможность исследовательского тестирования (не заранее составленные тест-кейсы, а придуманные на лету сценарии).

Недостатки

- Человеческий фактор. Некоторые ошибки могут остаться незамеченными.
- Провести серию стандартных автоматических тестов проще, чем протестировать проект вручную.
- Нельзя смоделировать большое количество пользователей.

Автоматизированное тестирование

Автоматизированное тестирование – этот метод предполагает использование инструментов и фреймворков* автоматизации.

Ситуации для выбора автоматизированного тестирования:

- ✓ **Повторяющиеся тесты:** Часто выполняемые тесты целесообразно автоматизировать.
- ✓ **Проведение нагрузочное тестирование:** Автоматизация помогает эффективно выявлять уязвимости в производительности.
- ✓ **Большое количество тест-кейсов:** Автоматизация экономит время при выполнении тысяч тестов.
- ✓ **Исключение человеческого фактора:** Автоматизация устраняет ошибки, связанные с человеческим фактором.
- ✓ **Работа с большими объемами данных:** Автоматизированные тесты быстро обрабатывают большие объемы данных, как в случае тестирования баз данных.



*Фреймворк — это набор инструментов и библиотек для упрощения и ускорения разработки программного обеспечения.

Инструменты для автоматизированного тестирования

Для каждого вида разработки существуют свои инструменты для автоматизации тестирования, один из таких – **Selenium**.



Selenium – инструмент с открытым исходным кодом для тестирования.

Наиболее популярной областью применения Selenium является **автоматизация тестирования веб-приложений**.

Однако при помощи Selenium можно автоматизировать любые другие рутинные действия, выполняемые через браузер.

- Поддерживает языки программирования, такие как Java, C#, Python, Ruby, и другие.
- Работает с браузерами Chrome, Firefox и другими
- Предоставляет мощные возможности для создания и выполнения тестов, взаимодействуя напрямую с браузером.

Также, популярные инструменты: TesNG, Jenkins, TestComplete, Cucumber, Appium, JMeter, Jira, Katalon Studio и другие...

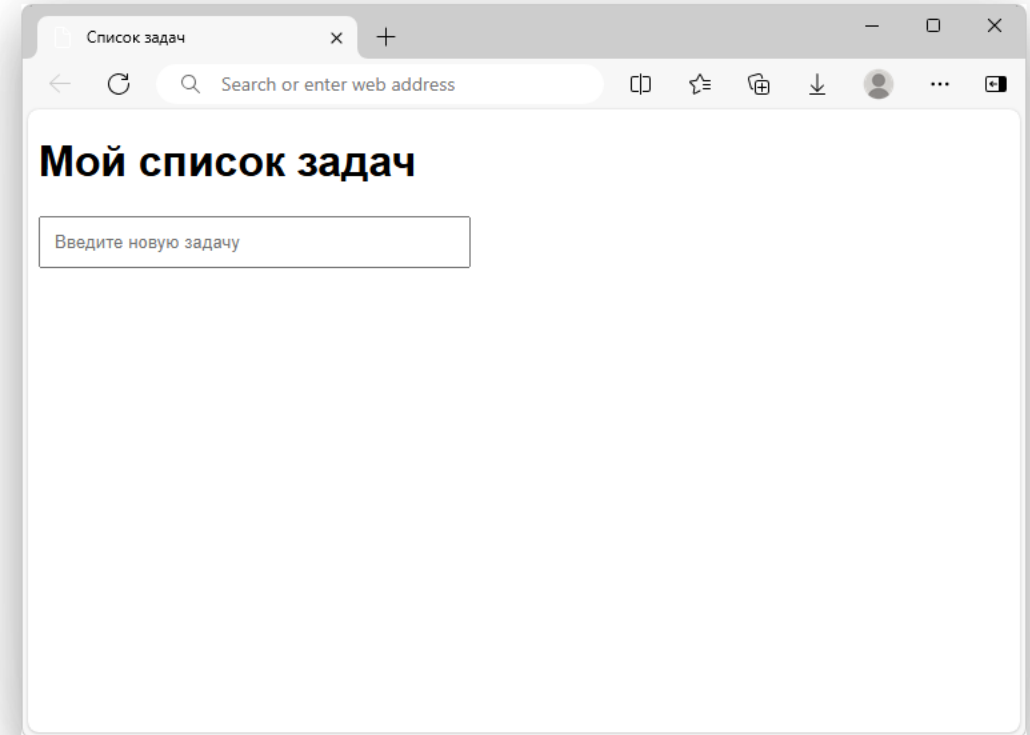
Автоматизированное тестирование. Пример на Selenium

Кейс: Веб-приложение для управления списком задач.

Что необходимо протестировать:
Добавление новой задачи в список задач

Как реализована функция: Ввод в текстовое поле текста задачи, добавление при помощи нажатия клавиши Enter.

Что используем: Библиотека Selenium
Webdriver для языка Python



Автоматизированное тестирование. Пример на Selenium

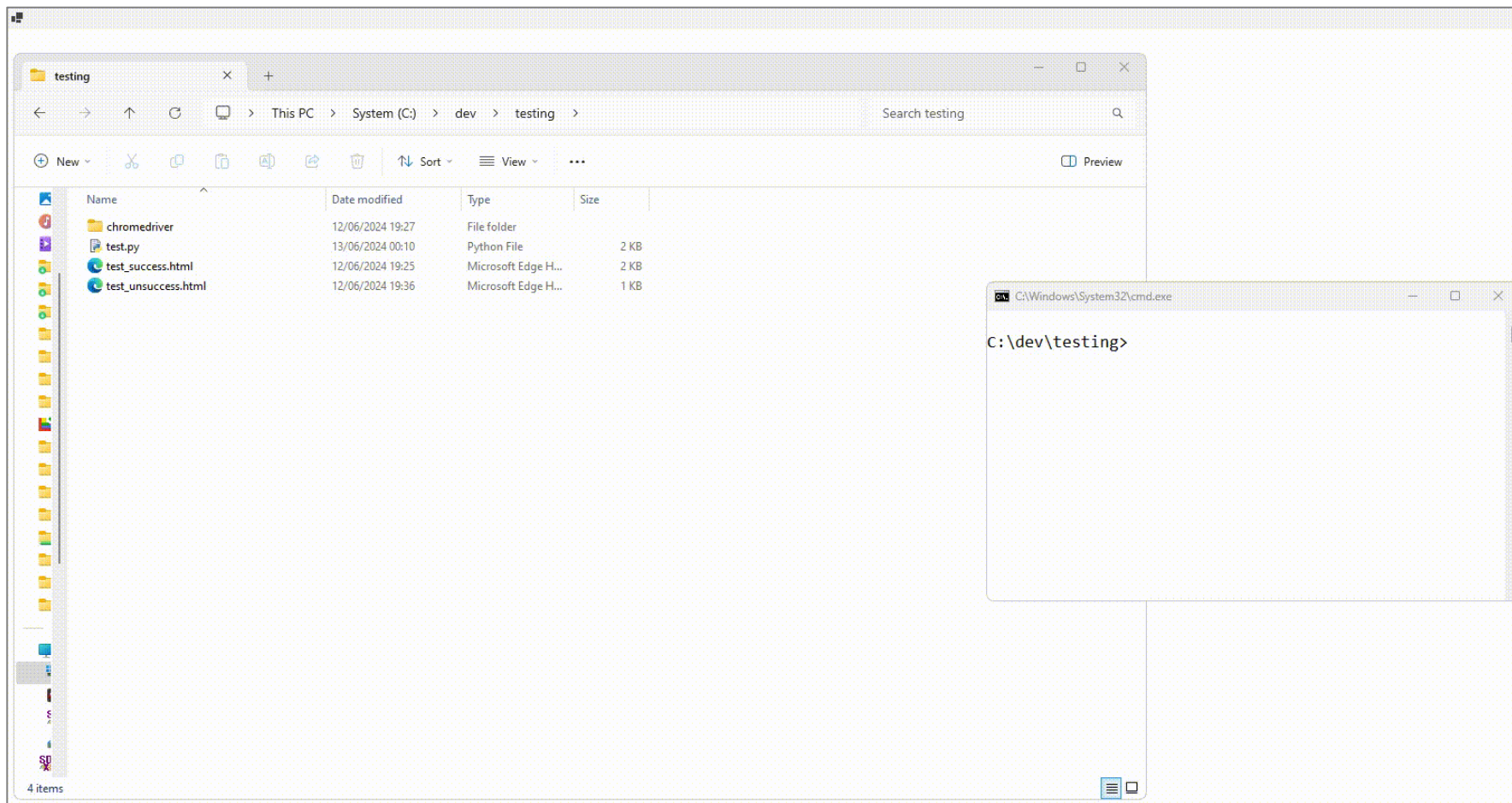
```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.common.keys import Keys
4 from selenium.webdriver.chrome.options import Options
5 from selenium.webdriver.chrome.service import Service
6 import time
7
8 options = Options()
9 options.binary_location = "chromedriver\\GoogleChromePortable\\App\\Chrome-bin\\chrome.exe"
10
11 driver_service = Service(executable_path=r'chromedriver\\chromedriver.exe')
12 driver = webdriver.Chrome(service=driver_service, options=options)
13 driver.set_window_size( width=1024, height=768)
14
15 # Открытие веб-приложения
16 driver.get("file:///C:/dev/testing/test_success.html")
17 # Находим поле для ввода новой задачи
18 input_field = driver.find_element(By.ID, value="task-input")
19
20 # Вводим новую задачу по букве с задержкой
21 new_task = "Тестовая задача"
22 for char in new_task:
23     input_field.send_keys(char)
24     time.sleep(0.2) # Задержка в 0.2 секунды между вводом каждой буквы
25
26 # Нажимаем клавишу Enter для добавления задачи
27 input_field.send_keys(Keys.ENTER)
28
29 # Добавленная задача должна отобразиться в списке задач
30 time.sleep(2) # Ждем, чтобы список успел обновиться
31 task_list = driver.find_element(By.ID, value="task-list")
32 tasks = task_list.find_elements(By.TAG_NAME, value="li")
33
34 # Проверяем, содержится ли новая задача в списке
35 for task in tasks:
36     if task.text == new_task:
37         print("Задача успешно добавлена!")
38         break
39
40 # Закрываем браузер после завершения теста
41 driver.quit()
```

Реализация теста

```
<body>
  <h1>Мой список задач</h1>
  <input type="text" id="task-input" placeholder="Введите новую задачу">
  <ul id="task-list"></ul>
  <script>
    document.getElementById('task-input').addEventListener('keypress', function (e)
    {
      if (e.key === 'Enter') {
        var taskText = e.target.value;
        if (taskText) {
          var li = document.createElement('li');
          li.textContent = taskText;
          document.getElementById('task-list').appendChild(li);
          e.target.value = '';
        }
      }
    });
  </script>
</body>
</html>
```

Код веб-страницы

Автоматизированное тестирование. Пример на Selenium



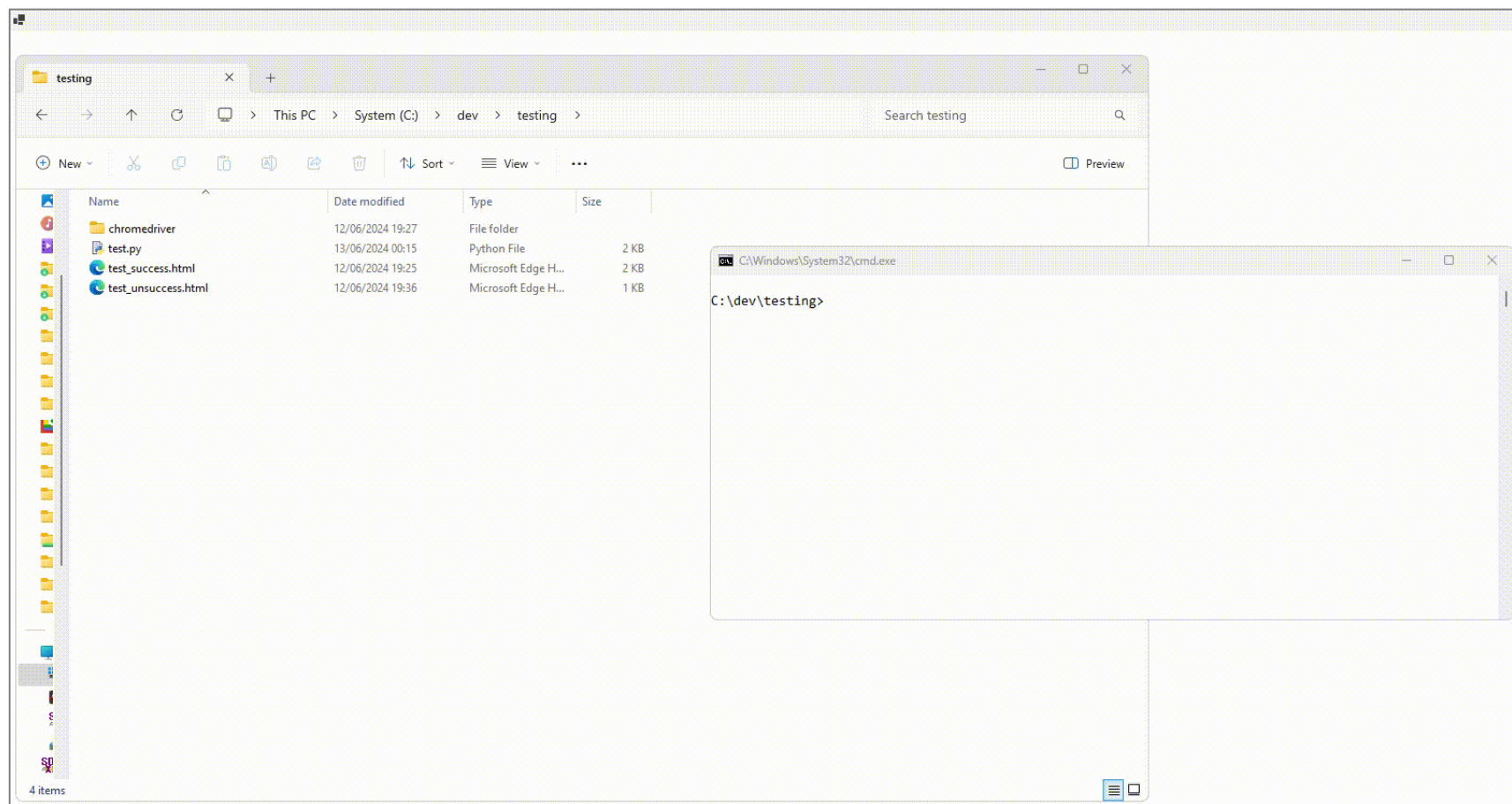
Успешно пройденный автоматизированный тест

Автоматизированное тестирование. Пример на Selenium

Изменим код страницы, удалив список и запустим тест снова...

```
<body>
  <h1>Мой список задач</h1>
  <input type="text" id="task-input" placeholder="Введите новую задачу">
  <script>
    document.getElementById('task-input').addEventListener('keypress', function (e) {
      if (e.key === 'Enter') {
        var taskText = e.target.value;
        if (taskText) {
          // Здесь могло бы быть добавление задачи в список, но списка нет
          console.log("Новая задача:", taskText);
          e.target.value = '';
        }
      }
    });
  </script>
</body>
</html>
```

Автоматизированное тестирование. Пример на Selenium



Автоматизированный тест не пройден

Сравнение ручного и автоматизированного тестирования

Характеристика	Ручное тестирование	Автоматизированное тестирование
Кем выполняется	Ручные QA специалисты и инструменты ручного тестирования.	Специалисты по автоматизированному тестированию со знанием кода и фреймворков тестирования.
Время	Может быть запущено очень быстро.	На настройку могут уйти недели.
Стоимость	Относительно низкая, поскольку ручные QA специалисты оплачиваются не так высоко, как специалисты по автоматизации	Специалисты по автоматизации стоят дороже, и может потребоваться дополнительное оборудование.
Рентабельность	Низкая, поскольку ручные тест-кейсы не всегда можно использовать повторно.	Высокая, так как помогает экономить ресурсы на повторных тестах.
Необходимость навыков программирования	Нет	Да
Повторение	Ручной QA специалист, выполняющий одни и те же тесты раз за разом, может потерять фокус и пропустить ошибки.	Можно повторять снова и снова с одинаковой эффективностью.
Человеческие ошибки	Есть склонность к человеческим ошибкам.	Исключение человеческой ошибки.

A blurred background image of a person with glasses and a bun, sitting at a desk with multiple computer monitors. The person is looking at the screens. The image is overlaid with a semi-transparent dark blue rectangle containing white text. The text is in Russian and reads 'ПРАКТИКИ ОБЕСПЕЧЕНИЯ КАЧЕСТВА ПО'.

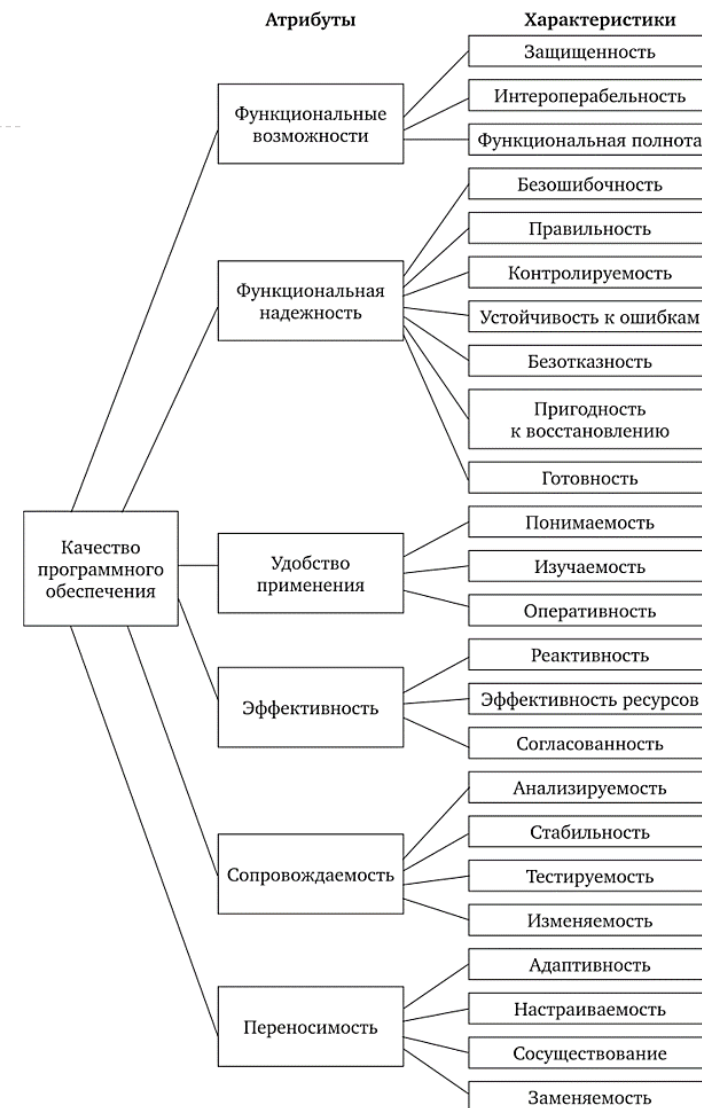
ПРАКТИКИ ОБЕСПЕЧЕНИЯ КАЧЕСТВА ПО

Качество ПО

Качество ПО – совокупность свойств, определяющих полезность программы для пользователей в соответствии с функциональным назначением и предъявленными требованиями.

Согласно стандартам ГОСТ* в модель качества входит **шесть характеристик**, или шесть основных показателей качества, которые перечислим в порядке их значимости для большинства пользователей:

- функциональные возможности;
- функциональная надежность;
- удобство применения;
- эффективность;
- сопровождаемость;
- переносимость.



*ГОСТ Р ИСО/МЭК 9126-93, ГОСТ Р ИСО/МЭК 12119-2000, ГОСТ 28195-89

Метрики качества

Метрика качества представляет собой стандартизированный способ измерения качества разрабатываемого или разработанного ПО.

Метрика (metric) - шкала измерений и метод, используемый для измерений (ISO 14598)

В процессе тестирования метрики используются:

- ✓ для отслеживания прогресса команды по срокам проекта, дедлайнам и другим временным отрезкам;
- ✓ качественной оценки текущего состояния системы;
- ✓ контроля качества процесса тестирования;
- ✓ постановки целей и эффективного планирования исходя из понимания существующих проблем.



Метрики предназначены для **повышения эффективности и результативности** процессов тестирования, а также способствуют **оптимизации будущего тестирования** путем получения точных данных о процессе продолжающегося тестирования.

Метрики качества. Данные

ДАННЫЕ ДЛЯ СНЯТИЯ МЕТРИК



Информация о ресурсах

- Статус задач, трудозатраты
- Статус специалистов, загруженности и хода выполнения задач
- Стоимость тестирования, проекта и другая финансовая информация



Покрытие тестовыми сценариями/чек-листами

- Требований
- Пользовательских историй
- Критериев приемки
- Рисков
- Кода



Метрики выполнения тестов

- Количество пройденных тест кейсов
- Количество проваленных кейсов
- Отношение выполненных кейсов к общему числу кейсов
- Отношение проваленных кейсов к общему числу кейсов
- Среднее время прохождения кейса



Процент написания тестовых сценариев/чек-листов

- Общее количество фичей
- Матрицу трассировки
- User story



Информация о дефектах

- Плотность дефектов
- Количество обнаруженных и исправленных дефектов
- Частота отказов
- Результаты подтверждающих тестов



Процент выполненных работ по подготовке

- Тестовой среды
- Тестовых данных

В настоящее время в мировой практике используется несколько сотен метрик программ.

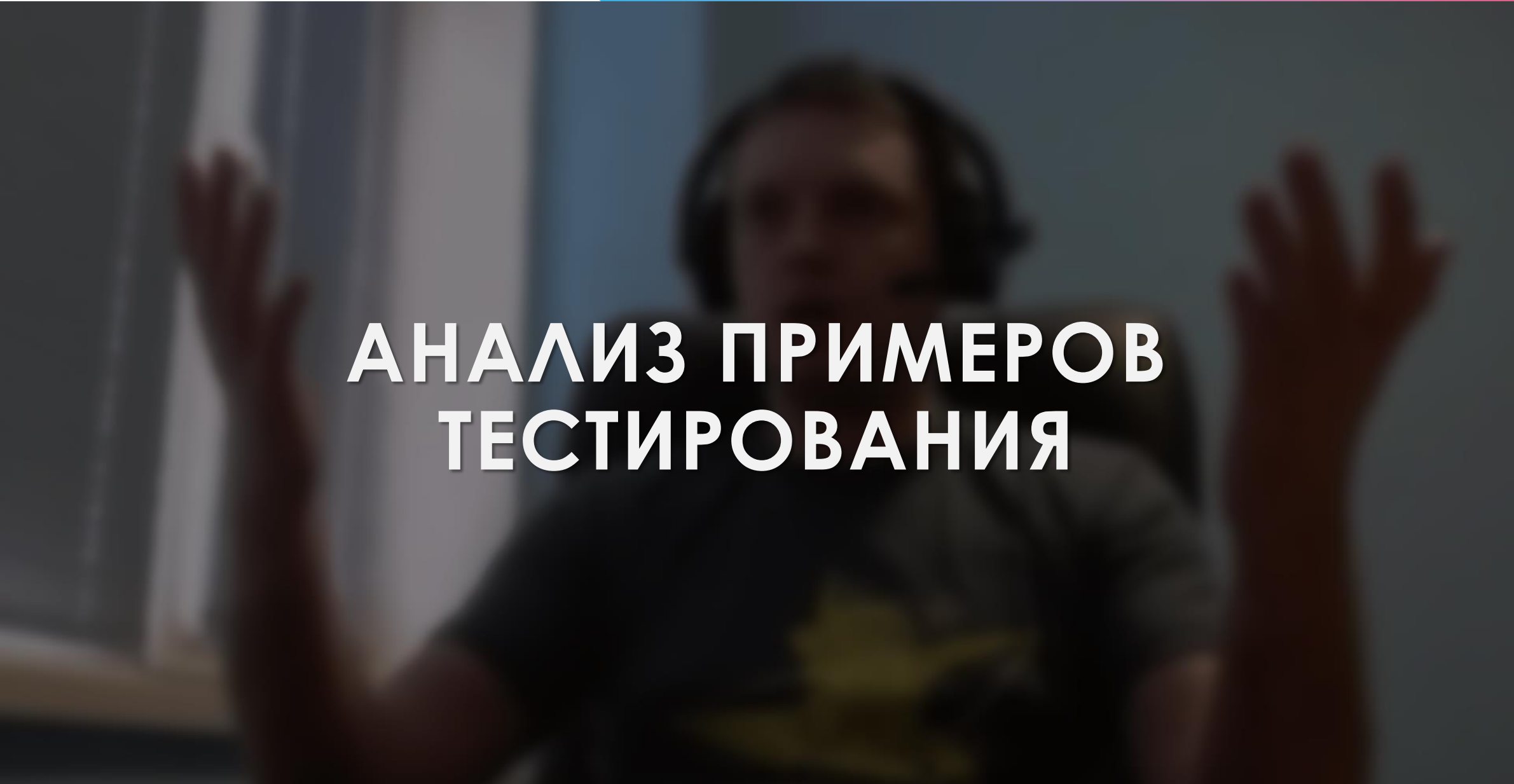
Однако не следует исключать того, что оценка качества ПО в целом может быть связана с субъективной интерпретацией получаемых оценок.

Метрики качества. Пример ведения статистики

Номер метрики	Метрика	Данные, полученные в ходе написания и выполнения тест-кейсов
1	Количество требований	5
2	Среднее количество написанных тест-кейсов на одно требование	40
3	Общее количество тест-кейсов, написанных для всех требований	200
4	Общее количество выполненных тест-кейсов	164
5	Количество зеленых тест-кейсов	100
6	Количество красных тест-кейсов	60
7	Количество заблокированных тест-кейсов	4
8	Количество невыполненных тест-кейсов	36
9	Общее количество выявленных дефектов	20
10	Дефекты, принятые дев-командой	15
11	Дефекты, отложенные на будущие релизы	5
12	Исправлено дефектов	12

Выводы на основе метрик:

- Количество написанных тест-кейсов на требование высокое, что говорит о детальном покрытии функциональности.
- Процент успешных тест-кейсов (зеленых) составляет 50%, что может указывать на некоторые проблемы с качеством разрабатываемого ПО.
- Доля исправленных дефектов среди выявленных не очень высока (60%), что может свидетельствовать о недостаточной эффективности процесса исправления дефектов.
- Дефекты, отложенные на будущие релизы, также нуждаются во внимании, поскольку их число не невелико, но они могут влиять на долгосрочную стабильность и качество продукта.



АНАЛИЗ ПРИМЕРОВ ТЕСТИРОВАНИЯ

Пример успешно проведенного тестирования

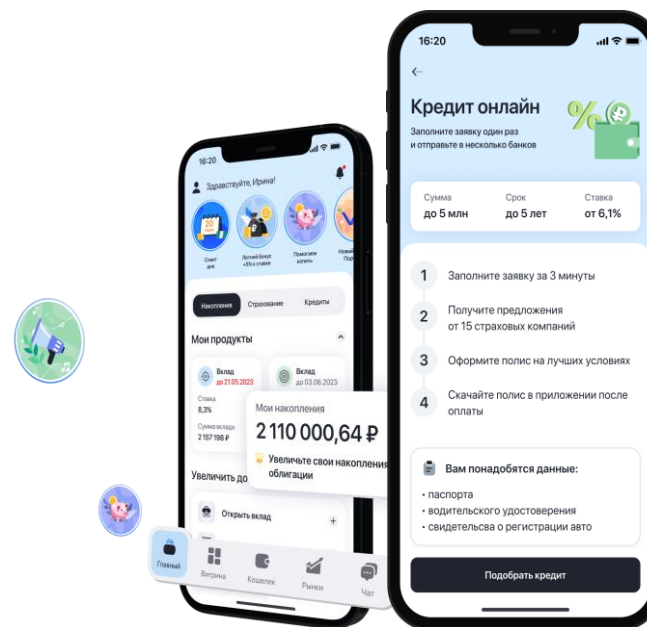
Финансовый маркетплейс «Финуслуги»

Что было сделано?

- 1) Разделение тестирования «спринтами»
- 2) Разработка задач и тест-кейсов к ней
- 3) Тестирование всех уровней
- 4) Использование автотестов
- 5) Оценка качества при помощи метрик

Что получилось в итоге?

- 1) Продукт, удовлетворяющий требованиям пользователей (оценка приложения в App Store 4,8 при 5 тыс. отзывах)
- 2) Платформа получила Премию Рунета в 2022



“Финуслуги — знаковый для нас проект. Наша команда тестировщиков освоила новые технологии, научилась писать автотесты и поучаствовала во всех этапах разработки. Но самой интересной частью стала организация процессов.”

Цитата автора [статьи](#)

Пример неуспешного тестирования

Неудачное обновление Windows 10 и OneDrive

- Обновление Windows 10 от 3 октября 2018 года удаляло файлы из папок с документами и картинками.
- Если локальные файлы не были синхронизированы с OneDrive (не имели копию в облаке), то они безвозвратно удалялись автоматически.

Итоги:

- 1) Потеря файлов некоторых пользователей, что оставило негативный опыт использования
- 2) Microsoft отозвала обновление и исправила ошибку
- 3) Microsoft приняла ряд мер для улучшения тестирования обновлений Windows



RZ

Robert Ziko

Created on October 4, 2018 ▾

windows 10 (October 2018 update) (version 1809) deleted all my files

I have just updated my windows using the October update (10, version 1809) it deleted all my files of 23 years in amount of 220gb. This is unbelievable, I have been using Microsoft products since 1995 and nothing like that ever happened to me.

Files were located at C:/Users/rober/Documents/
This location is still present, with no files. All of files deleted.

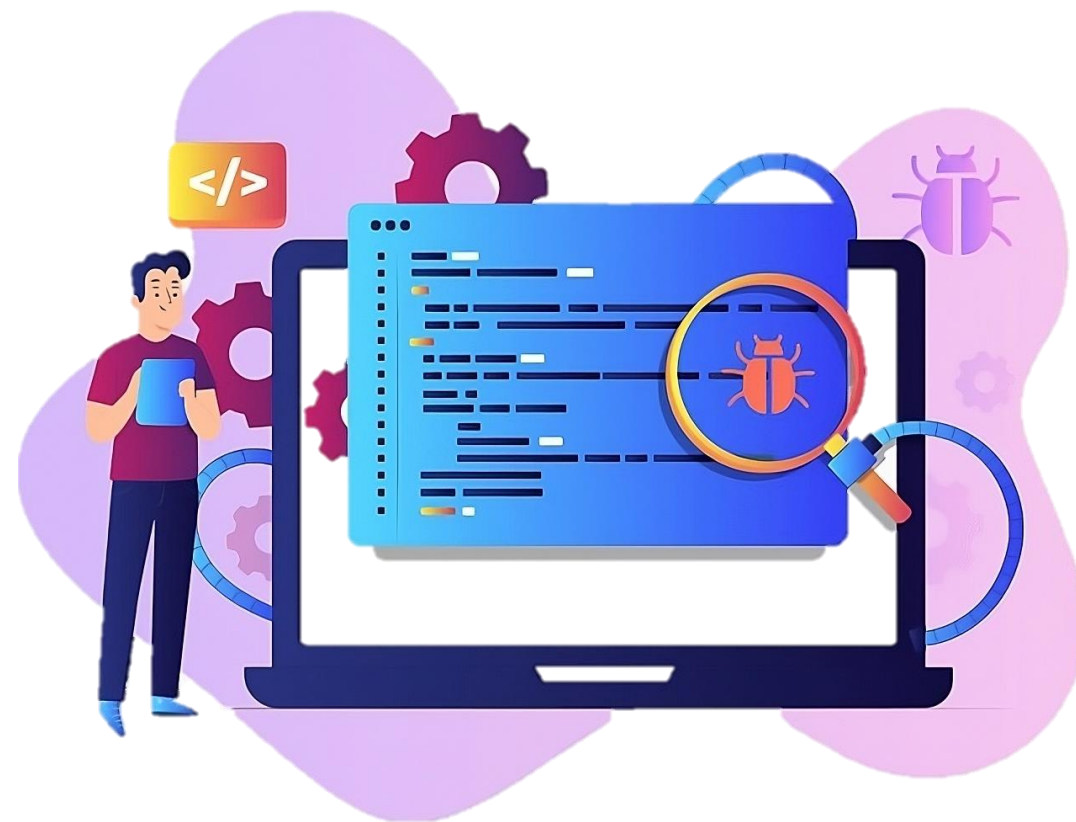
I am extremely upset. Not sure what to do....please let me know.

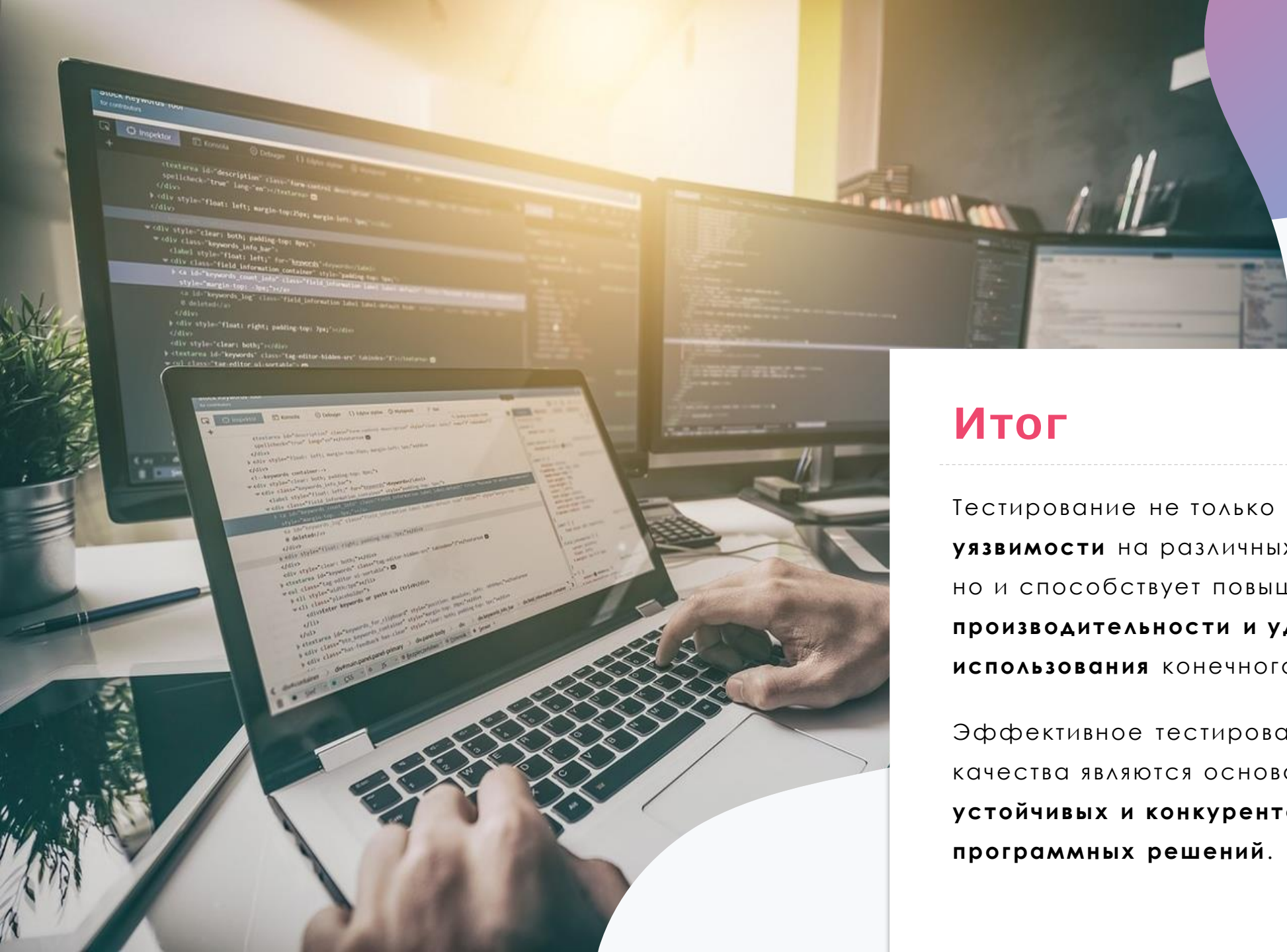
Robert

Пользователь потерял 220ГБ файлов

Будущие направления и перспективы

- **Использование нейросетей и моделей машинного обучения** для создания и оптимизации тестов
- **Автоматическое тестирование без сценариев** для минимизации использования кода
- **Интеграция с DevOps**, что обеспечит непрерывное тестирование, автоматизацию, экономию ресурсов и времени.
- **Фокус на пользовательский опыт**, большой акцент на UX-тестирование





Итог

Тестирование не только **выявляет дефекты и уязвимости** на различных этапах разработки, но и способствует повышению **надежности, производительности и удобства использования** конечного продукта.

Эффективное тестирование и обеспечение качества являются основой для создания **устойчивых и конкурентоспособных программных решений**.

Спасибо за внимание!

