**Слайд 2**

Все мы знаем, что каждая программа выдает баги, то есть ошибки. Иногда они оказываются реальными особенностями программы, то есть фичами. Но чаще программисты с трудом признают свои ошибки, выдавая баги за фичи. В мире программистов даже появилось крылатое выражение "Это не баг, это фича!"  
Для отслеживания и искоренения багов необходимо проводить тестирование программы.

**Слайд 3**

Тестирование программного обеспечения - проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.

Существует несколько признаков, по которым принято производить классификацию видов тестирования. Обычно выделяют следующие классификации :

**Слайд 4**

**Функциональное тестирование** —тестирование в целях проверки реализуемости функциональных [требований](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E)

**Тестирование производительности**

**Конфигурационное тестирование** (**Configuration Testing**) — это вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы

**Юзабилити-тестирование**— [исследование](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), выполняемое с целью определения, удобен ли некоторый объект (такой как [веб-страница](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0), [пользовательский интерфейс](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) или устройство) для его предполагаемого применения.

**Тестирование безопасности** — оценка уязвимости [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) к различным атакам.

**Локализа́ция програ́ммного обеспече́ния** — процесс адаптации программного обеспечения к культуре какой-либо страны.

И

**Тестирование совместимости** — вид **тестирования** основной целью которого является проверка корректной работы продукта в определенном окружении

**Слайд 5**

**Тестирование чёрного ящика** - стратегия тестирования функционального поведения объекта с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве тестируемого объекта.

**Тестирование белого ящика** — тестирование, которое учитывает внутренние механизмы системы

**Тестирование серого ящика –** комбинация первых двух вариантов

**Слайд 6**

**Ручное тестирование** —Оно производится [тестировщиком](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D1%89%D0%B8%D0%BA) без использования программных средств, для проверки программы путём моделирования действий пользователя. В роли тестировщиков могут выступать и обычные пользователи, сообщая разработчикам о найденных ошибках.

**Автоматизированное тестирование программного обеспечения** —Данный вид тестирования использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

**Полуавтоматизированное тестирование –** Комбинирование первых двух вариантов

**Слайд 7**

**Модульное тестирование** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *unit testing*) — процесс, позволяющий проверить на корректность единицы [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4).

**Интеграцио́нное тести́рование** — это процесс, при которой отдельные программные модули объединяются и тестируются в группе.

**Систе́мное тести́рование програ́ммного обеспече́ния** — это тестирование, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям.

Так же есть еще тестирование **По времени проведения тестирования**

* **Альфа-тестирование**
* **Бета-тестирование**

**По признаку позитивности сценариев**

* **Позитивное тестирование**
* **Негативное тестирование**

**Слайд 8**

Непрерывная интеграция нацелена на ускорение и облегчение процесса выявления проблем, возникающих в процессе разработки программного обеспечения. При регулярной интеграции изменений единовременный объем проверок уменьшается с помощью CI. В результате на отладку тратится меньше времени, оставшееся время можно перераспределить на добавление новых функций.

Были рассмотрены популярные CI платформы такие как

1. **CircleCI**
2. **Travis CI**
3. **Jenkins**

**Слайд 9**

Фреймворки автоматизированного тестирования - это набор приложений, концепций и инструментов, которые обеспечивают поддержку для автоматизированного тестирования программного обеспечения. Основным преимуществом такой структуры является низкая стоимость обслуживания.

Функции фреймворка автоматизированного тестирования

* создание механизма для подключения или выполнения тестируемого приложения;
* выполнение тестов;
* отчетность о результатах.

Были рассмотренны такие фрейворки как Xunit, Nunit и Visual Studio Unit Testing Framework

**Слайд 10**

Были рассмотрены и проанализированы системы версионного контроля такие системы как SVN GIT Mercurial. Были выявлены их достоинства и недостатки.

**Слайд 11**

**“И собственно сама программа для которой планируется разработка автоматизированных тестов”**

Приложение Alpha.Alarms используется в пунктах автоматизации и мониторинга технологических процессов. Применяется для отслеживания событий и тревог, которые появляются при изменении состояний технологических объектов. Основные функции приложения:

1. отображение сообщений о событиях и тревогах в режиме реального времени (оперативный режим);
2. отображение истории сообщений о событиях и тревогах за прошедшие периоды (исторический режим).

Так же Alpha.Alarms может использоваться как встраиваемый компонент или работать как самостоятельное приложение. В качестве встраиваемого компонента Alpha.Alarms может использоваться в Alpha.HMI, а также в HMI SCADA систем сторонних разработчиков - Genesis, iFix и других.

Слайд 12

В ходе преддипломной практики были изучены аспекты процессов тестирования. Были описаны концепция непрерывной интеграции и популярные CI-платформы. Так же были изучены различные фреймворки тестирования и проведен обзор систем контроля версий. Была описана система TestLink которая используется в настоящий момент для проведения ручного тестирования. Так же была описана программа Alpha.Alarms.

Для Alpha.Alarms существует ряд ручных тестов в системе TestLink. В будущем планируется автоматизация ручных тестов с использованием библиотеки NUnit.