

Тестировщик программного обеспечения: с нуля до первых проектов”

“Введение в тестирование ПО”

Раздел № 1

Темы раздела

Тема 1.1. Введение в тестирование ПО

Тема 1.2. Роль тестировщика в ПО

Тема 1.3. Жизненный цикл ПО и тестирования

Тема 1.4. Методологии разработки и тестирования ПО



План занятия

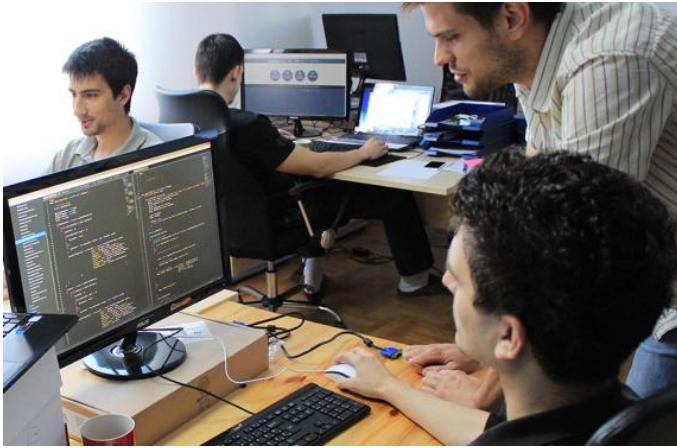
Тема 1.1. Введение в тестирование ПО

Вопросы:

1. О профессии
2. Основные термины в IT
3. Цели и задачи тестирования ПО
4. Принципы тестирования ПО
5. Понятие качество ПО
6. Стандарты качества ПО



Как выглядит профессия ?



Прикладное программное обеспечение

Прикладное ПО (или приложение) — «вспомогательная программа», предназначенная для выполнения определенных задач и рассчитанная на непосредственное взаимодействие с пользователем.



Чем занимается тестировщик программного обеспечения?



Тестировщик — специалист, принимающий участие в тестировании компонента или системы.

В его обязанность входит поиск вероятных ошибок и сбоев в функционировании объекта тестирования

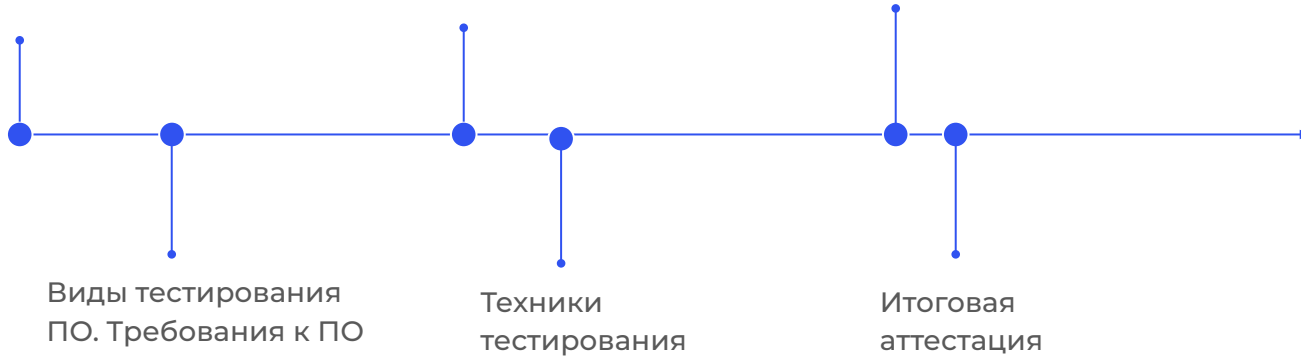
- Создает сценарии тестирования
- Тестирует программный продукт
- Находит и Документирует дефекты
- Контролирует исправление дефектов

Что нужно чтобы освоить профессию?

Введение в
тестирование ПО

Тестовая
документация

Основы автоматизации
тестирования



Карьера
тестировщика

Основные термины

Тестирование программного обеспечения - это процесс который позволяет узнать, делает ли программа то, для чего она была создана, и не делает ли она ничего лишнего

Баг, ошибка или дефект

Сбой (отказ) - отклонение поведения системы от ожидаемого

Дефект (bug, problem) - недостаток компонента или системы, который может привести к сбою или отказу

Ошибочное действие - действие пользователя, приводящее к неверному результату

Отчет об ошибке (bug report) - документ, описывающий действия или условия, которые привели к выявлению дефекта.

Ошибка - действие программиста во время разработки, которое приводит к тому, что в программном обеспечении содержится дефект, который во время работы программы может привести к неправильному результату

Верифицируй, валидируй

Верификация (verification) — проверка, что выполнены требования по наличию чего-либо

Валидация (validation) — проверка, что выполнены требования по конкретному использованию чего-либо

План Тестирования (Test Plan) - это документ, который описывает весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения

Чек-лист (checklist, контрольный список) — перечень параметров, которые нуждаются в проверке.

Тест-кейс (test case, тестовый случай) — своего рода сценарий или описание последовательности шагов при проведении тестирования.

Тестовый набор (test suite) — несколько тест-кейсов, которые объединены по типу тестирования или другим признакам.

Цели и задачи тестирования

- Убедиться, что ПО соответствует предъявленным требованиям
- Предоставление информации о состоянии продукта на текущий момент для принятия решений
- Повышение уверенности в уровне качества продукта
- Обнаружение и локализация дефектов
- Предотвращение дефектов

Принципы тестирования ПО

Семь принципов тестирования

Принцип 1 – Исчерпывающее **тестирование** недостижимо

Принцип 2 – **Тестирование** демонстрирует наличие дефектов, но не их отсутствие

Принцип 3 – Раннее **тестирование** эффективно

Принцип 4 – Скопление дефектов

Принцип 5 – Парадокс пестицида

Принцип 6 – **Тестирование** зависит от контекста

Принцип 7 – Заблуждение об отсутствии ошибок

Понятие качества ПО

Качество программного обеспечения - это степень, в которой ПО обладает требуемой комбинацией свойств.

Качество программного обеспечения - это совокупность характеристик ПО, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

Стандарты

ИСО (ISO) - международная организация по стандартизации

IEEE (ай-трипл-и) - институт инженеров по электротехнике и электронике

ГОСТ Р ИСО/МЭК 2510-2015 - информационные технологии (ИТ). Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов.

ГОСТ Р 56920-2016/ISO/IEC/IEEE 29119:2013- Системная и программная инженерия. Тестирование программного обеспечения.

QA, QC, testing

Обеспечение качества (Quality Assurance - QA) - это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации программного обеспечения (ПО) информационных систем, предпринимаемых на разных стадиях жизненного цикла ПО для обеспечения требуемого уровня качества выпускаемого продукта.

Контроль качества (Quality Control - QC) - это совокупность действий, проводимых над продуктом в процессе разработки для получения информации о его актуальном состоянии в разрезе: "готовность продукта к выпуску", "соответствие зафиксированным требованиям", "соответствие заявленному уровню качества продукта".



Обязанности QA

- формирование критериев качества;
- планирование мероприятий по соблюдению критериев на каждом этапе разработки продукта;
- выбор инструментов тестирования;
- тестирование продукта;
- расчет KPI;
- предотвращение появления ошибок и усовершенствование процесса.

Вопрос/упражнение по теме 1.1

Практическая работа № 1.1

1. Разбор характеристик качества ПО





Домашнее задание № 1.1

Изучение дополнительных материалов:

- словарь тестировщика ПО <https://bytextest.ru/slovar-testirovschika/>
- определение терминов ПО (по списку) в википедии или смежных источниках сети интернет

До следующего вебинара!



План занятия

Тема 1.2. Роль тестировщика в разработке ПО

Вопросы:

1. Кто такой тестировщик. Определение понятия «Тестировщик»
2. Роли тестировщика
3. Обязанности тестировщика ПО
4. Личные качества тестировщика ПО
5. Почему это важно



Кто такой тестировщик?

Тестировщик ПО — специалист, который занимается тестированием программного обеспечения

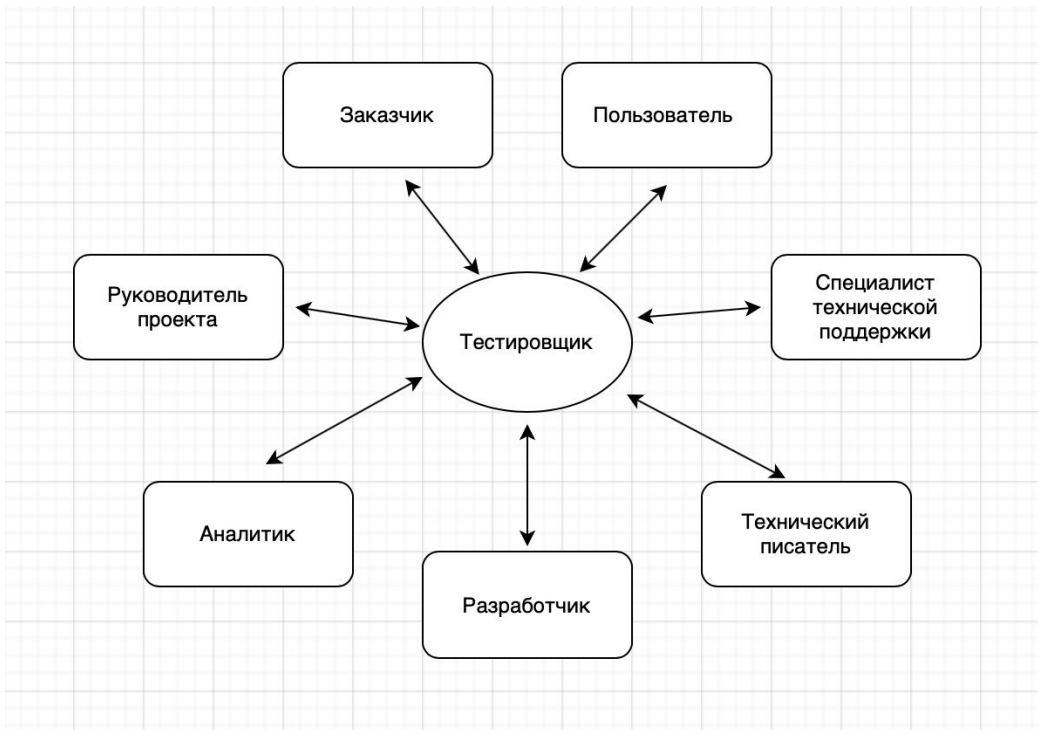
Тестировщик ищет дефекты, оформляет их и передает разработчику для исправления. Контролирует корректность исправления дефекта

Тестировщик проверяет готовую программу на соответствие требованиям

Тестировщик предоставляет информацию о качестве продукта

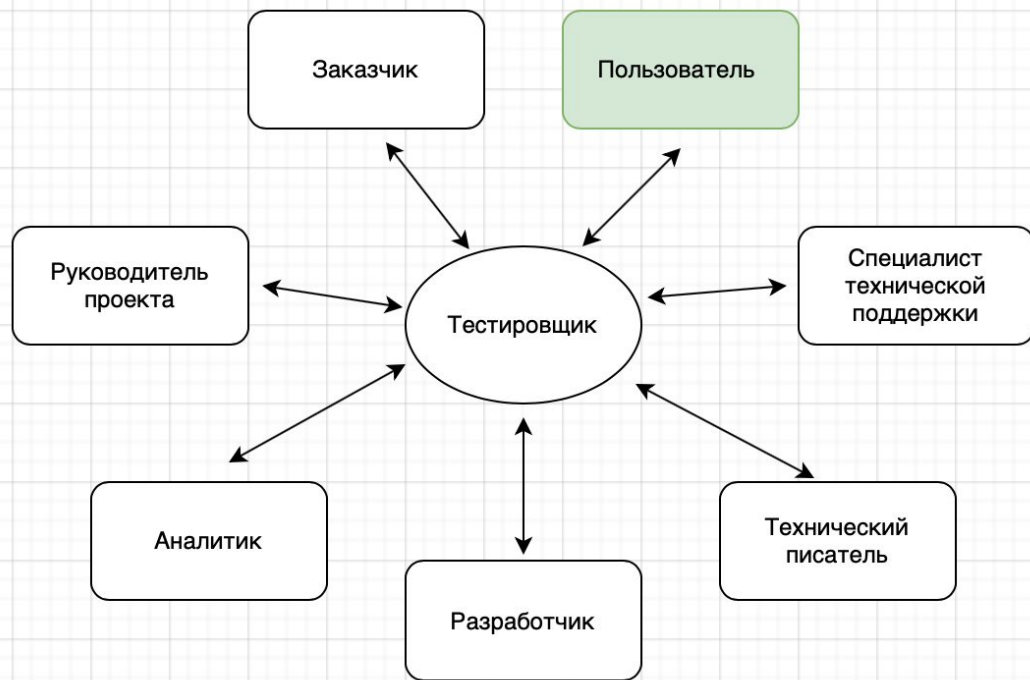
Роль тестировщика

Как тестировщик, вы оказываете услуги по тестированию другим участникам процесса:



Тестировщик и Пользователь

В сущности, тестировщик работает на пользователей продукта. Их удовлетворение является приоритетной задачей всей команды и, конечно же, тестировщика

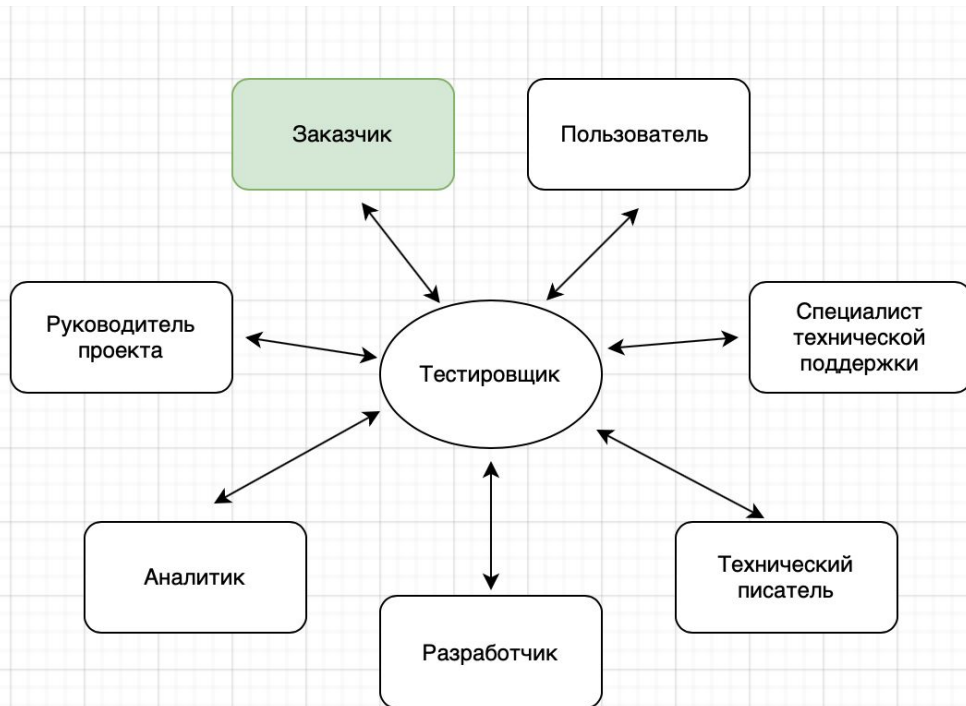


Тестировщик и Заказчик

Заказчик диктует требования

Тестировщик контролирует,
чтобы требования были
выполнены

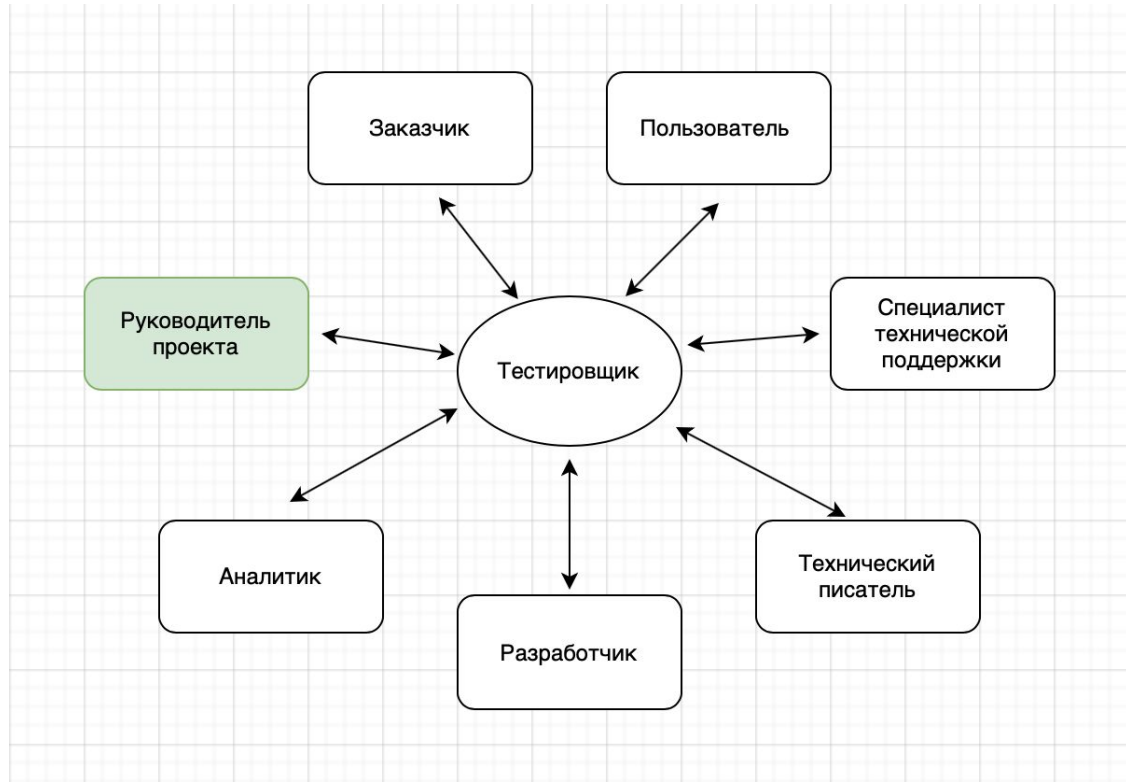
Иногда, Заказчик=Пользователь



Тестировщик и Руководитель проекта

Руководитель проекта обязан быть в курсе ситуации по всем выполняемым работам на проекте.

Тестировщик должен, по запросу извещать РП о статусе тестирования, об обнаруженных серьезных проблемах

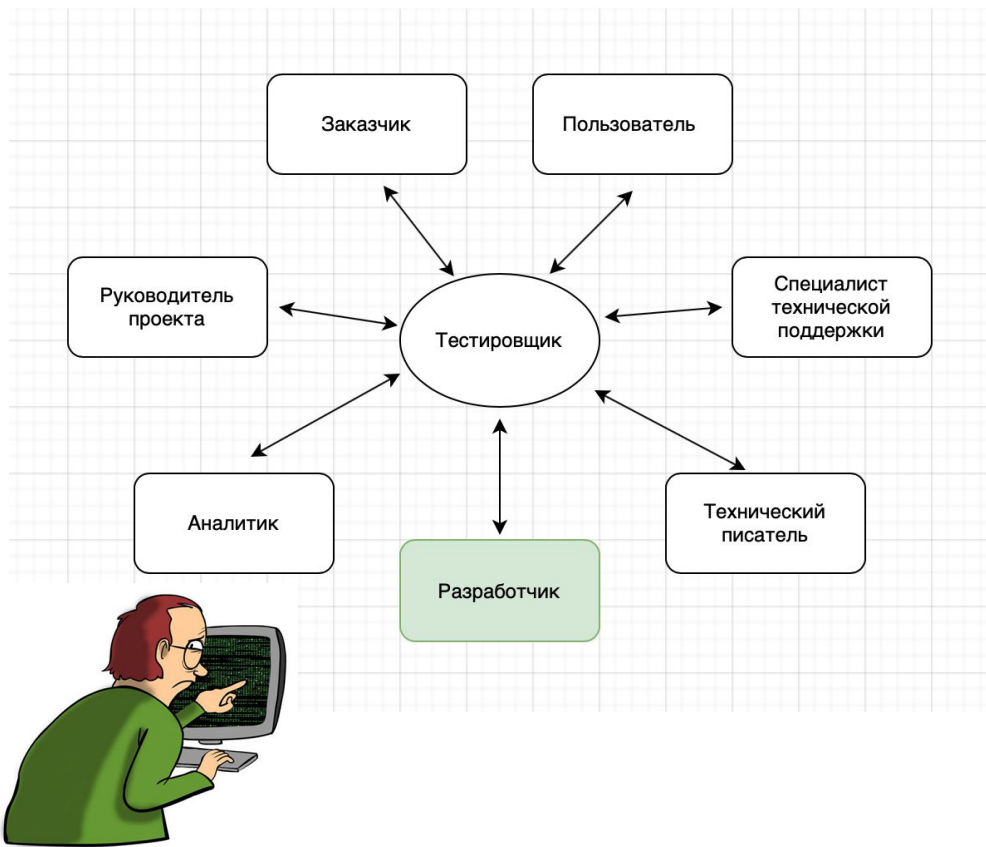


Тестировщик и Разработчик

Тестировщик облегчает работу разработчика, сообщая ему о его ошибках

От тестировщика требуется как можно быстрее и точнее указать место ошибки - оформить баг репорт

Даже если ошибок нет, задача разработчика будет считаться выполненной после подтверждения тестировщиком

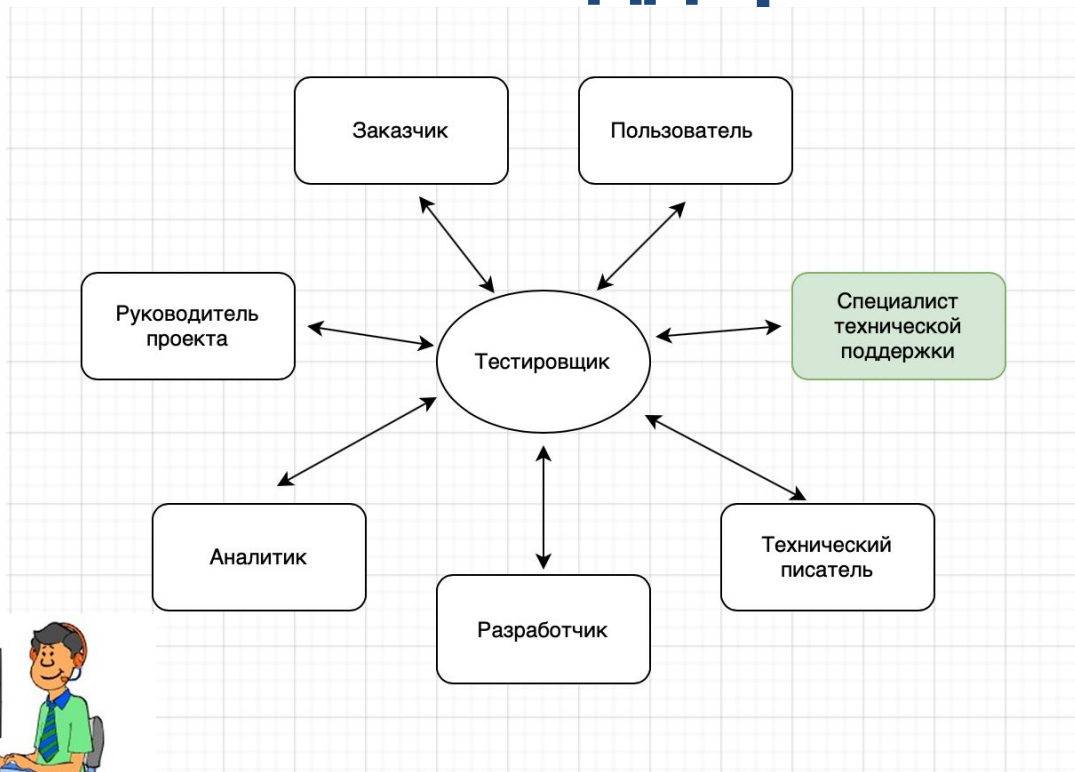


Тестировщик и Техническая поддержка

Тестировщик сообщает тех. поддержке о проблемах, к которым нужно подготовить пользователя

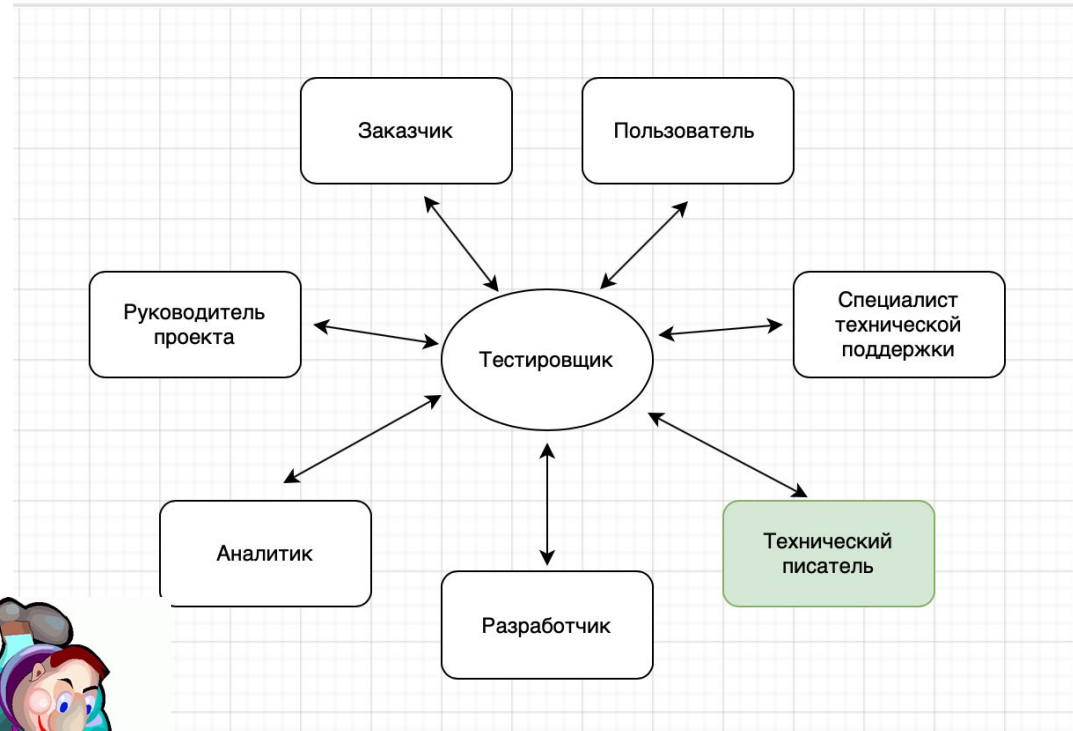
Тестировщик принимает дефекты которые обработала тех. поддержка

Можно вести общение с пользователем через тех. поддержку



Тестировщик и Технический писатель

Специалисты, пишущие руководства, получают неполную информацию о продукте. Тестировщик может лучше объяснить им, как работает программа и предостеречь от тех или иных ошибок в документации.

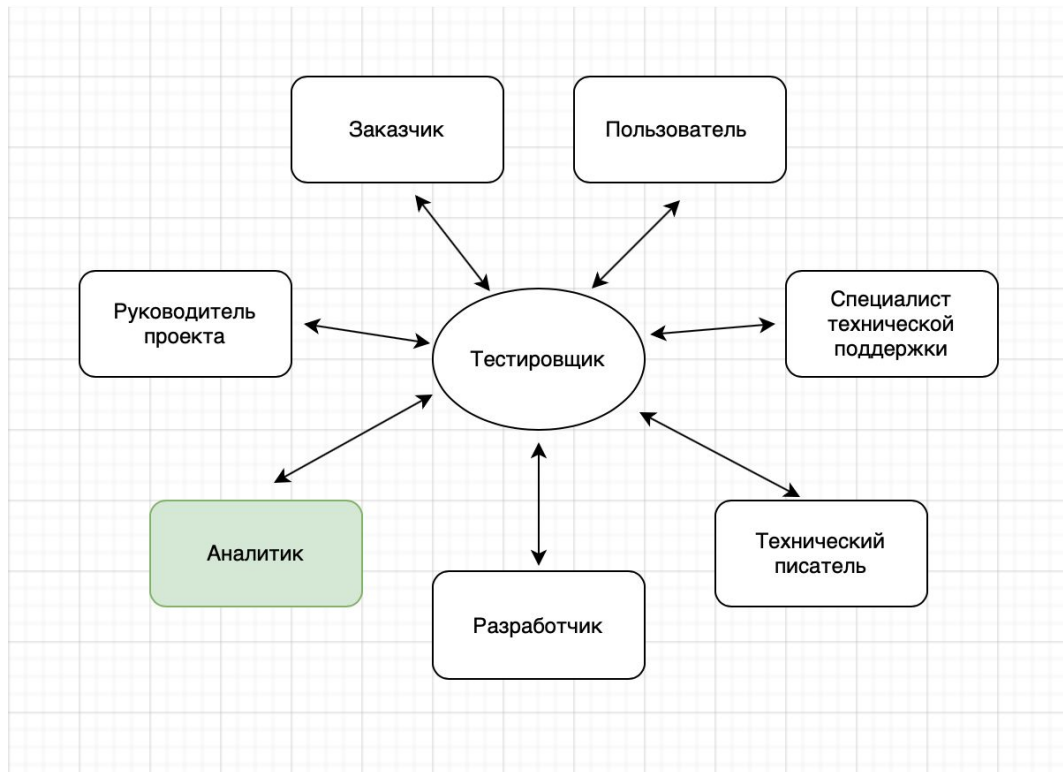


Тестировщик и Аналитик

Тестировщик может уточнить требования у аналитика

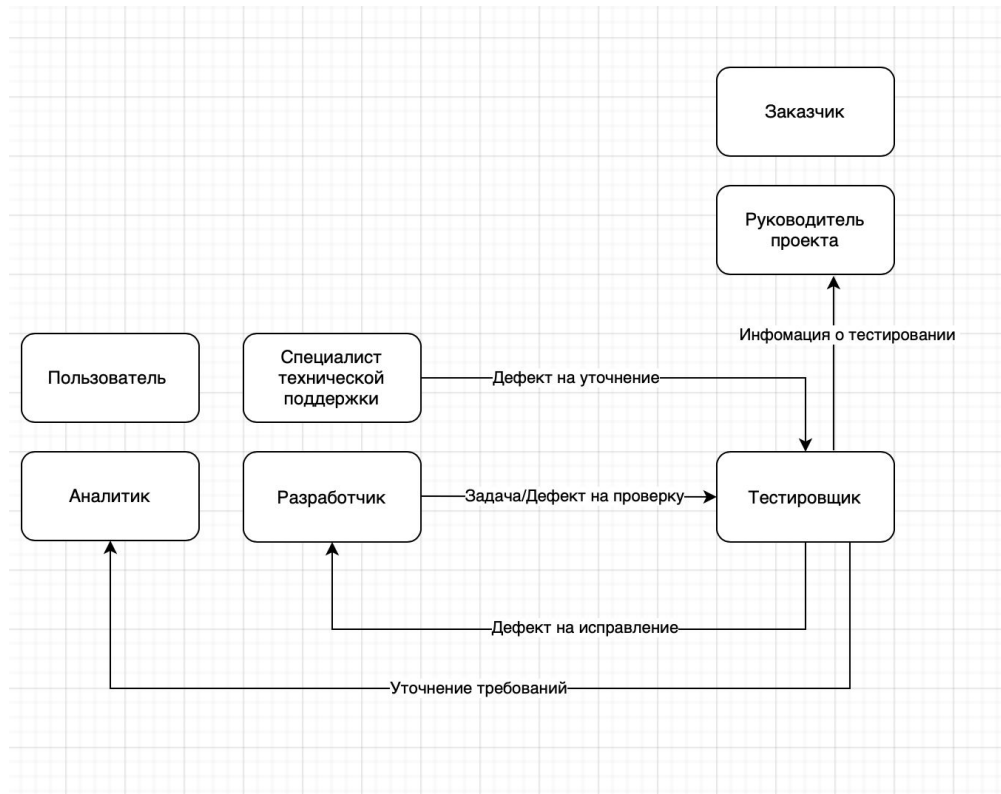
Аналитик может уточнить поведение программы у тестировщика и скорректировать требования

Тестировщик может предложить улучшение



Обязанности тестировщика ПО

- Подготовка проверок
- Тестирование задач
- Поиск и фиксация дефектов
- Проверка исправления дефектов
- Отправка дефектов найденных пользователями в работу
- Предоставление информации о тестировании



Необходимые тестировщику Навыки

Навыки в тестировании (Testings Skills)

Навыки в технической области (Tech Skills)

Социальные навыки (Soft Skills)

Уровень знания языков (Language Skills)

Testings Skills

Ручное тестирование

Тестирование по готовым тестам

Тест-анализ

Тестирование требований

Исследовательское тестирование

Автоматизированное тестирование

Методология автоматизированного тестирования

Разработка автотестов в Selenium

Нагрузочное тестирование

Разработка модели нагрузочного тестирования

Разработка сценариев нагрузочного тестирования на JMeter

Разработка сценариев нагрузочного тестирования в Load Runner

Документирование

Написание чек-листов

Написание тест-кейсов

Разработка тест-планов

Планирование работы и оценка трудозатрат

Локализация и документирование дефектов

Отчётность по результатам тестирования

Оценка тестового покрытия, метрики в тестировании

Тестирование юзабилити

Экспертиза в UX

Проведение юзабилити-тестов

Платформо-ориентированное тестирование

Тестирование веб-сервисов

Тестирование безопасности веб-сервисов

Тестирование мобильных приложений

Тестирование баз данных

Tech Skills

Администрирование Windows

Администрирование Linux, Unix

Администрирование Mac OS X

Основы сетевых технологий

Любые языки программирования

Интернет технологии (HTML, CSS, HTTP , response codes)

Веб-серверы (MS IIS/ Apache HTTP Server/ Apache Tomcat)

Инструменты виртуализации (VMware, VirtualBox, MS Virtual PC)

Непрерывная интеграция

Стандарты обеспечения качества (ISO)

Базовые знания бизнес областей (Finance, Travel, Health Care etc.)

Soft Skills

Ответственность

Коммуникации

Стрессоустойчивость

Сотрудничество и координация действий

Наставничество

Делегирование

Самостоятельность

Тайм-менеджмент

Общее отношение к работе

Почему это важно

Правильно протестированный программный продукт обеспечивает надежность, безопасность и высокую производительность, что в дальнейшем приводит к экономии времени, денег и удовлетворенности клиентов.

Тестирование важно, потому что ошибки в программном обеспечении могут дорого обойтись производителю.

Тестирование необходимо потому, что все мы совершаем ошибки. Некоторые из них могут быть незначительными, в то время как другие – иметь самые разрушительные последствия. ... Именно поэтому любой продукт нуждается в проверке – **тестировании**, прежде чем его можно будет эффективно и безопасно использовать.

Вопрос/упражнение по теме 1.2

Практическая работа № 1.2

1. Работа с профстандартом по компетенции тестировщика



Домашнее задание № 1.2

Изучение дополнительных материалов:

- основные обязанности тестировщика
<https://fktopm.ru/file/113-svyatoslav-kulikov-testirovanie-po-bazovyi-kurs.pdf>
- мифы и заблуждения о тестировании:
<https://habr.com/ru/company/alee/blog/144975/>
- Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах, Савин Р., 2007.

До следующего вебинара!



План занятия

Тема 1.3. Жизненный цикл ПО и тестирования

Вопросы:

1. Понятие «Жизненный цикл ПО»
2. Этапы жизненного цикла ПО
3. Жизненный цикл тестирования



Программа - исполняемый файл

комбинация компьютерных инструкций и данных, позволяющая аппаратному обеспечению вычислительной системы выполнять вычисления или функции управления (стандарт ISO/IEC/IEEE 24765:2010)

```
00000000 fc 31 c0 8e c0 8e d8 8e d0 bc 00 7c 89 e6 bf 00 |.l.....|
00000010 06 b9 00 01 f3 a5 89 fd b1 08 f3 ab fe 45 f2 e9 |.....E..|
00000020 00 8a f6 46 bb 20 75 08 84 d2 78 07 80 4e bb 40 |...F.u...x..N.@|
00000030 8a 56 ba 88 56 00 e8 fc 00 52 bb c2 07 31 d2 88 |.V.V...R...l..|
00000040 6f fc 0f a3 56 bb 73 19 8a 07 bf 87 07 b1 03 f2 |o...V.s.....|
00000050 ae 74 0e b1 0b f2 ae 83 c7 09 8a 0d 01 cf e8 c5 |.t.....|
00000060 00 42 80 c3 10 73 d8 58 2c 7f 3a 06 75 04 72 05 |.B...s.X,...u.r.|
00000070 48 74 0d 30 c0 04 b0 88 46 b8 bf b2 07 e8 a6 00 |Ht.0...F.....|
00000080 be 7b 07 e8 b2 00 8a 56 b9 4e e8 8e 00 eb 05 b0 |.{.....V.N.....|
00000090 07 e8 b0 00 30 e4 cd 1a 89 d7 03 7e bc b4 01 cd |...0.....~....|
000000a0 16 75 0d 30 e4 cd 1a 39 fa 72 f2 8a 46 b9 eb 16 |.u.0...9.r..F...|
000000b0 30 e4 cd 16 88 e0 3c 1c 74 f1 2c 3b 3c 04 76 06 |0....<.t...<.v.|
000000c0 2c c7 3c 04 77 c9 98 0f a3 46 0c 73 c2 88 46 b9 |...<.w....F.s..F.|
000000d0 be 00 08 8a 14 89 f3 3c 04 9c 74 0a c0 e0 04 05 |.....<.t.....|
000000e0 be 07 93 c6 07 80 53 f6 46 bb 40 75 08 bb 00 06 |.....S.F.@.....|
000000f0 b4 03 e8 59 00 5e 9d 75 06 8a 56 b8 80 ea 30 bb |...Y.^..u..V...0.|
00000100 00 7c b4 02 e8 47 00 72 86 81 bf fe 01 55 aa 0f |...G.r.....U...|
00000110 85 7c ff be 85 07 e8 19 00 ff e3 b0 46 e8 24 00 |...f.e3.b0.46.e8.24.00|
00000120 b0 31 00 d0 eb 17 0f ab 56 0c be 78 07 e8 eb ff |.l.....V.x....|
00000130 89 fe e8 03 00 be 85 07 ac a8 80 75 05 e8 04 00 |.....u.....|
00000140 eb f6 24 7f 53 bb 07 00 b4 0e cd 10 5b c3 8a 74 |...$.S.....[.t..|
00000150 01 8b 4c 02 b0 01 56 89 e7 f6 46 bb 80 74 13 66 |...L...V...F..t.f|
00000160 6a 00 66 ff 74 08 06 53 6a 01 6a 10 89 e6 48 80 |j.f.t..Sj.j...H..|
00000170 cc 40 cd 13 89 fc 5e c3 20 20 a0 0a 44 65 66 61 |@...^...Defa|
00000180 75 6c 74 3a a0 0d 8a 00 05 0f 01 06 07 0b 0c 0e |ult.....|
00000190 83 a5 a6 a9 0d 0c 0b 0a 09 08 0a 0e 11 10 01 3f |.....?|
000001a0 bf 44 4f d3 4c 69 6e 75 f8 46 72 65 65 42 53 c4 |.D0.Linu.FreeBS.|
000001b0 66 bb 44 72 69 76 65 20 00 00 80 8f b6 00 00 00 |f.Drive.....|
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 55 aa |.....U..|
00000200
```

Программа - исходный файл

синтаксическая единица, которая соответствует правилам определённого языка программирования, состоящая из определений и операторов или инструкций, необходимых для определенной функции, задачи или решения проблемы (стандарт ISO/IEC 2382-1:1993)

```
1 sub Calculator()  
2   sub addition(self, other)  
3     return self + other  
4   end  
5  
6   /** Create a list of 2 numbers. */  
7   sub makeFraction(umerator, denominator)  
8     return [numerator, denominator]  
9   end  
10  
11  /** Warning: Destroys original fraction! */  
12  sub multiplyFrac(frac, otherFrac)  
13    frac[0] *= otherFrac[0]  
14    frac[1] *= otherFrac[1]  
15    return frac  
16  end  
17 end  
18  
19 sub InfinityCalculator()  
20   inherit Calculator()  
21   /** Create a list of 2 numbers. */  
22   sub makeFraction(umerator, denominator)  
23     if denominator == 0  
24       /* The user is trying to divide by 0.  
25        * Use Java's way of handling this: */  
26       import math into mathematics  
27       return mathematics.INFINITY  
28     end  
29     return [numerator, denominator]  
30   end  
31 end
```

Method

Class

Программное обеспечение (ПО, software)

Программное обеспечение — программа или множество программ, используемых для управления компьютером (ISO/IEC 26514:2008)

Программное обеспечение — компьютерные программы, процедуры и, возможно, соответствующая документация и данные, относящиеся к функционированию компьютерной системы (IEEE Std 829—2008)

Жизненный цикл программного обеспечения

Период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации

Конечное множество типовых фаз и этапов, через которые система может проходить за всю историю своей жизни

- ГОСТ 34.601-90
- ISO/IEC 15288:2015 Systems and software engineering — System life cycle processes
- ISO/IEC/IEEE 12207:2017 System and software engineering — Software life cycle processes

Проект

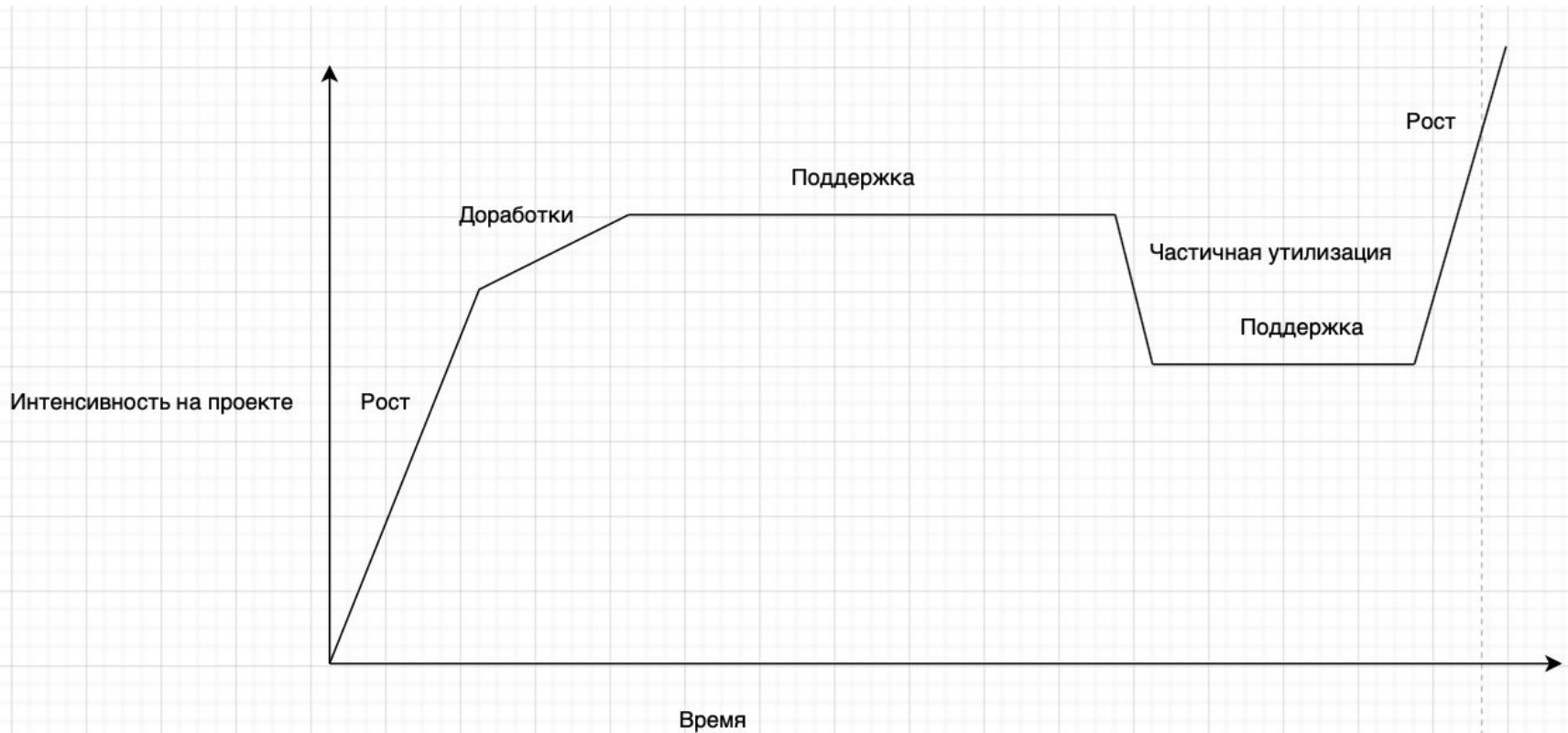
Проект – это целенаправленная, ограниченная во времени деятельность, осуществляемая для удовлетворения конкретных потребностей при наличии внешних и внутренних ограничений и использовании ограниченных ресурсов

Разработка программного обеспечения это Проектная деятельность это ПРОЕКТ

Фазы проекта

- Инициализация (Разработкам концепции проекта)
- Анализ и Планирование
- Исполнение
- Контроль
- Завершение





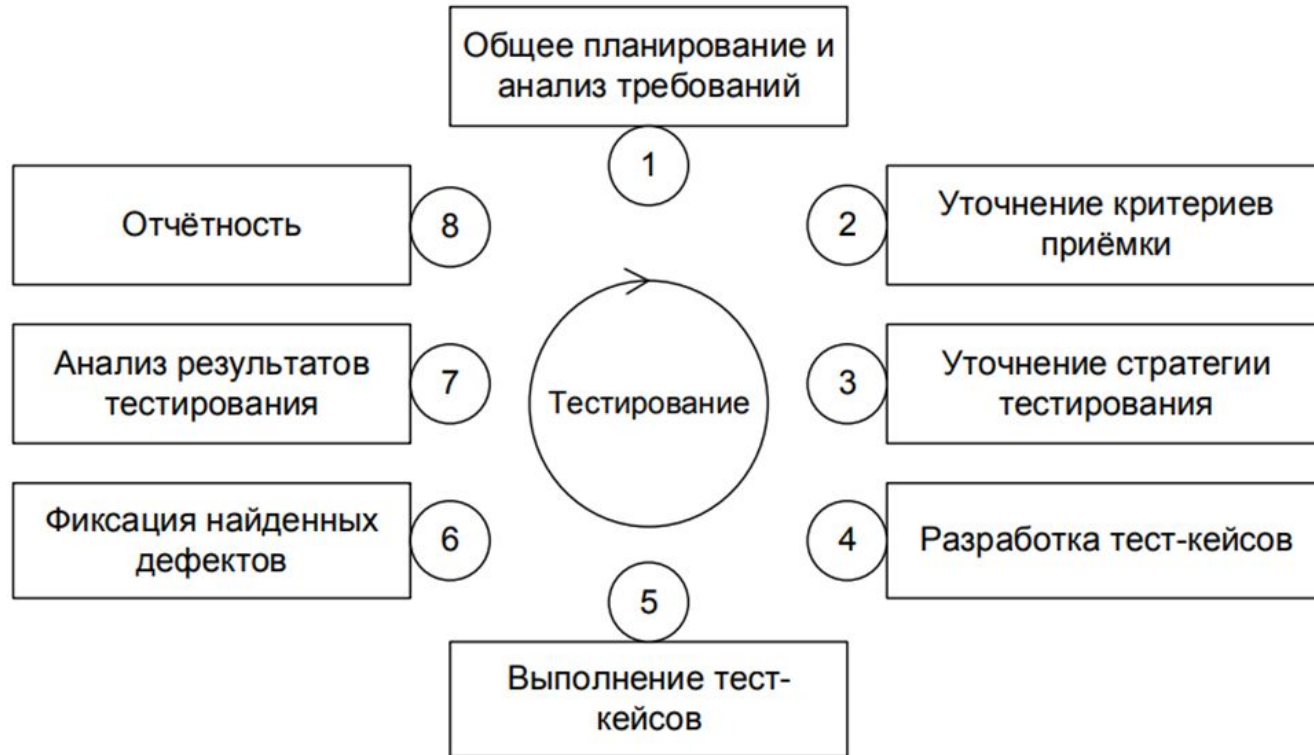
Этапы жизненного цикла ПО

- планирование и анализ требований;
- документирование требований
- проектирование;
- кодирование (разработка);
- тестирование и исправление недостатков;
- внедрение и сопровождение;
- усовершенствование;
- утилизация;

Жизненный цикл тестирования

последовательность действий, проводимых в процессе тестирования, с помощью которых гарантируется качество программного обеспечения и его соответствие требованиям

Этапы жизненного цикла тестирования



Вопрос/упражнение по теме 1.3

Практическая работа № 1.3

1. Работа с кейсом: жизненный цикл ПО



Домашнее задание № 1.3

Изучение дополнительных материалов:

- жизненный цикл ПО

https://ru.wikipedia.org/wiki/Жизненный_цикл_программного_обеспечения

<https://xbsoftware.ru/blog/zhiznennyj-tsykl-testirovaniya-po-preimuschestva-shagi/>

- ЖЗ тестирования

<https://fktopm.ru/file/113-svyatoslav-kulikov-testirovanie-po-bazovyi-kurs.pdf>

(Куликов С. Тестирование программного обеспечения. Базовый курс: практическое пособие стр. 26-28)

-

До следующего вебинара!



План занятия

Тема 1.4. Методологии разработки и тестирования ПО

Вопросы:

1. Понятие “Методология разработки ПО”
2. Виды методологии разработки ПО



Методология разработки ПО

совокупность **методов**, применяемых на различных стадиях жизненного цикла программного обеспечения и имеющих общий философский подход.

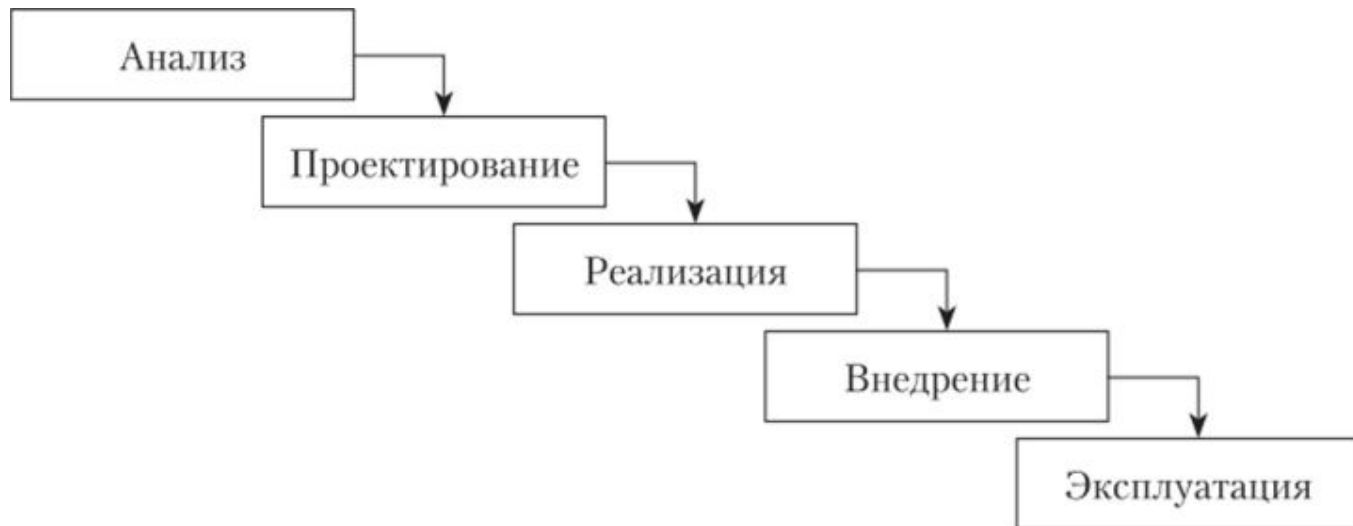
Виды методологии разработки ПО

- Каскадная модель (Waterfall Model, водопадная)
- V-образная модель
- Инкрементная и итеративная модель
- Спиральная модель

Водопадная модель

последовательное прохождение стадий,
каждая из которых должна завершиться
полностью до начала следующей

Каскадная модель (Waterfall Model, водопадная)

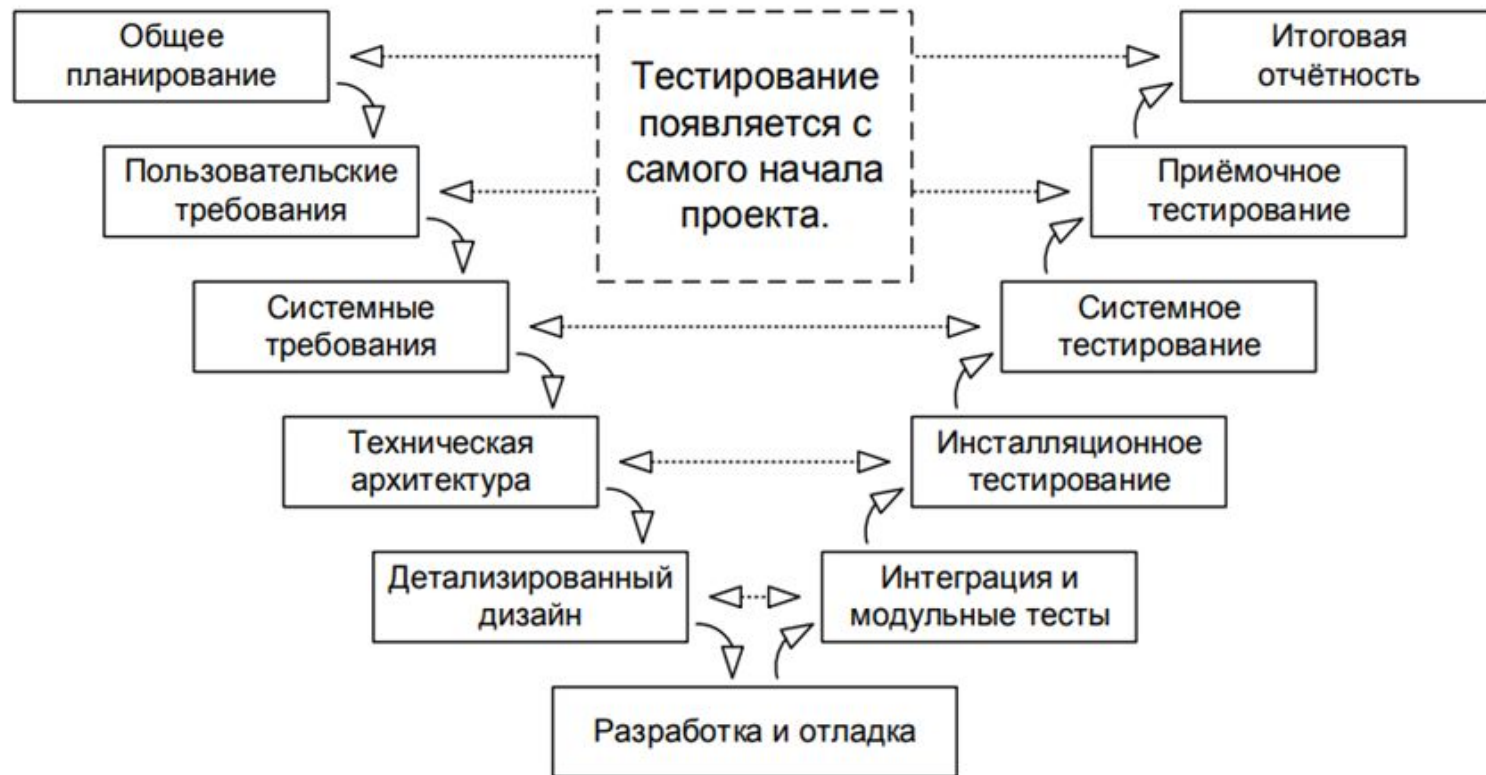


V-образная модель

Унаследовала структуру «шаг за шагом» от каскадной модели.

Особенностью модели можно считать, что она направлена на тщательную проверку и тестирование каждого этапа

V-образная модель

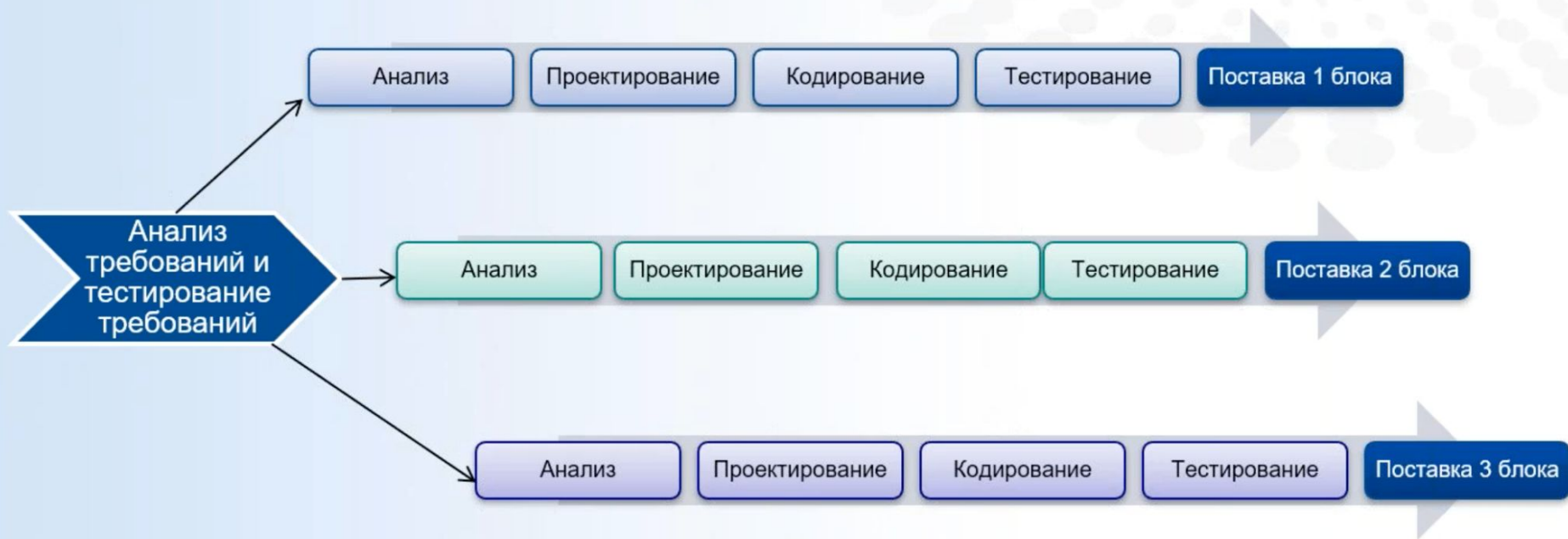


Инкрементальная модель

Итеративные или инкрементальные модели – это модели, в которых система реализуется и тестируется итерационно. В результате каждой итерации появляется рабочий продукт, являющийся частью конечного разрабатываемого продукта

Разновидности: Rapid Application Development (RAD) и гибкие методологии разработки (Agile).

Инкрементальная модель



Спиральная модель

Система разрабатывается на основе ранних прототипов. Разработка движется от прототипа к прототипу, каждый из которых тестируется, затем перепроектируется и повторно прототипируется, после чего снова тестируется. И так до тех пор, пока все рискованные конструктивные решения не пройдут тестирование (или не пройдут и будут отвергнуты) .

Спиральная модель



Спиральная модель

Ключевые характеристики

Спиральная модель сочетает в себе концепцию итеративной разработки с систематикой и контролем водопадной модели:

- Данная модель включает в себя большую часть этапов водопадной модели, и в том же порядке. Однако этапы отделены друг от друга планированием, оценкой рисков, прототипированием и имитацией.
- На каждой итерации по всему циклу продукт является расширением более раннего продукта (как в итеративной модели)
- Расширение модели осуществляется только после анализа рисков: во время каждого цикла проводится поиск крупных рисков и делаются попытки по их устранению

Спиральная модель предназначена для крупных, дорогостоящих и сложных проектов (с высокими проектными рисками)

Спиральная модель

Преимущества

- Лучший способ разработки систем с большим количеством неизвестных величин
- Одна из наиболее гибких моделей: изменения могут быть внесены позже в жизненном цикле
- Управление рисками – одна из встроенных функций данной модели, что делает ее более привлекательной по сравнению с другими моделями

Недостатки

- Стоимость продукта неизвестна
- Чересчур трудный подход для проектов с четкими техническими требованиями к продукту

«RAD Model» (rapid application development model или быстрая разработка приложений)



Гибкие методологии Agile

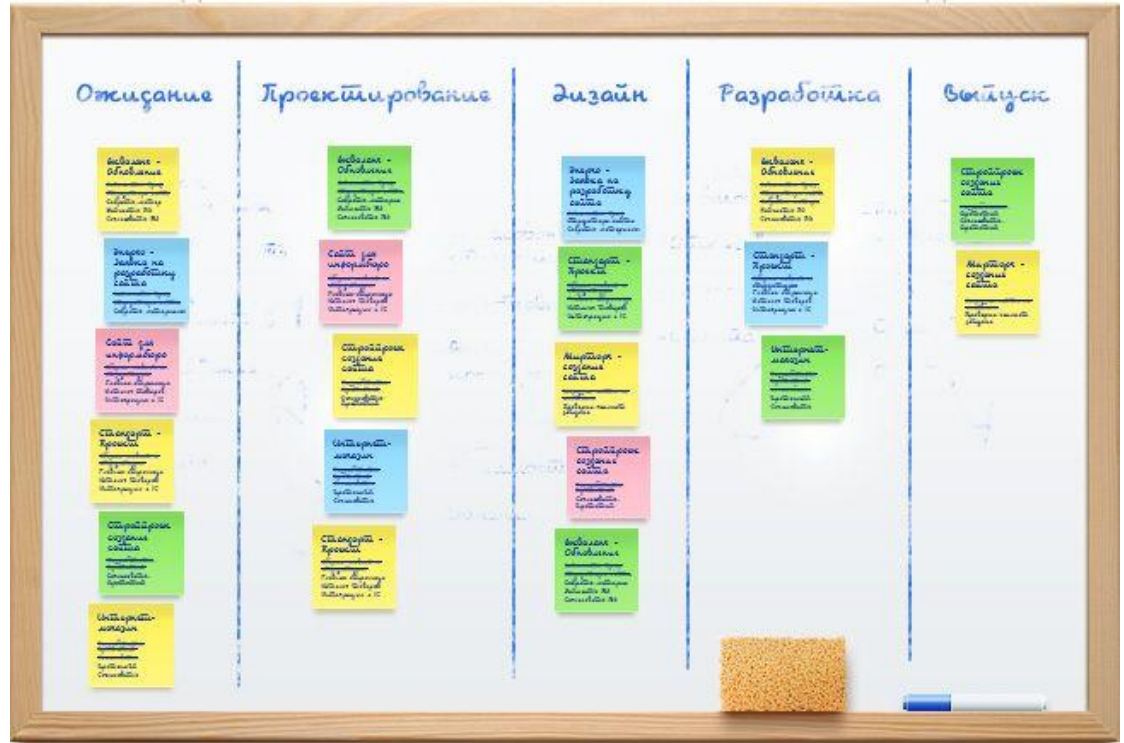
- люди и взаимодействие важнее процессов и инструментов;
- работающий продукт важнее исчерпывающей документации;
- сотрудничество с заказчиком важнее согласования условий контракта;
- готовность к изменениям важнее следования первоначальному плану.

Scrum

набор правил для организации гибкого рабочего процесса, который заключается в командном подходе, работе итерациями, фокусировке на цели каждой итерации и нестандартном распределении обязанностей внутри коллектива.

Kanban

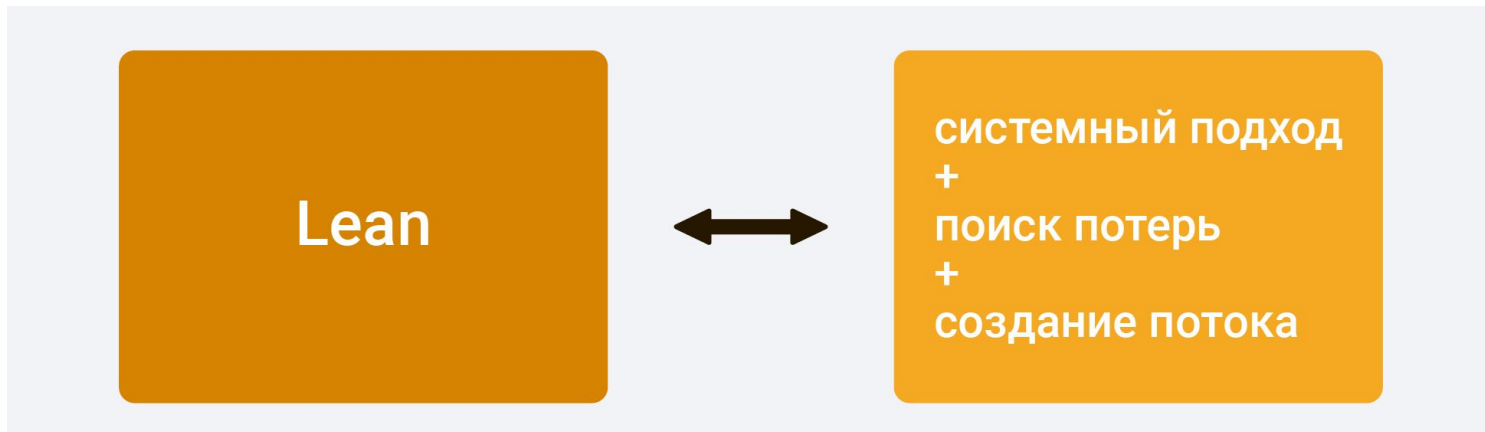
система постановки задач и организации рабочих процессов для эффективного достижения поставленных целей. Главный показатель эффективности в kanban – это среднее время прохождения задачи по доске.



Lean-методология

философия бережливого мышления. Подход, который позволяет экономить ресурсы и получать лучший результат.

Придерживаться Lean — значит всегда использовать системный подход, искать и устранять потери, создавать поток. Поток — это непрерывный процесс создания ценности — не любого продукта, а именно того, который нужен потребителю.



Вопрос/упражнение по блоку 1.4

Практическая работа № 1.4

1. Работа с кейсом по одной из моделей



Домашнее задание № 1.4

Изучение дополнительных материалов:

- методологии разработки ПО

<https://fktpm.ru/file/113-svyatoslav-kulikov-testirovanie-po-bazovy-kurs.pdf>

- модели и методологии разработки ПО

<https://gb.ru/posts/methodologies>

Интерактивный элемент

Пройдите тест № 1 “Базовая терминология по тестированию ПО”



Закрепление темы

**Выполните практическое задание по разделу
№ 1 “Введение в тестирование ПО”**

**Чек-лист практического задания по
разделу №1**



До следующего вебинара!

