

# Практический курс тестирования программного обеспечения

## Урок 1



**Test Club 2014**

<http://www.testclub.com.ua>

# План занятия:

1. SDLC (Software Development Life Cycle)
2. Модели жизненного цикла ПО
3. Методологии разработки информационных систем
4. Определение термина «Тестирование ПО» и определение необходимости тестирования ПО
5. QA vs QC, Verification vs Validation
6. Роли и артефакты в проектной команде
7. Зачем нужны тестировщики на проекте?
8. Анализ требований к программному обеспечению



# 1. SDLC (Software Development Life Cycle)

***Software Development Life Cycle (Жизненный цикл программного обеспечения ПО)*** — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации



# SDLC (Software Development Life Cycle)

*Основные стадии и этапы создания программного обеспечения:*

- Разработка концепции к ПО
- Формирование требований к ПО
- Разработка
- Тестирование
- Ввод в эксплуатацию
- сопровождение



## 2. Модели жизненного цикла ПО

***SDLC Model (Модель жизненного цикла ПО)*** - структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла

*Примеры:*

- Каскадная модель
- Итеративная модель
- Спиральная модель



# Модели жизненного цикла ПО

Где почитать:

***Каскадная модель:***

<http://bit.ly/1vbhYPx>

***Итеративная модель:***

<http://bit.ly/1d6KDKE>

***Спиральная модель:***

<http://bit.ly/1ouQpit>





### 3. Методологии разработки ИС

**Методология** - учение о методах, методиках, способах и средствах познания

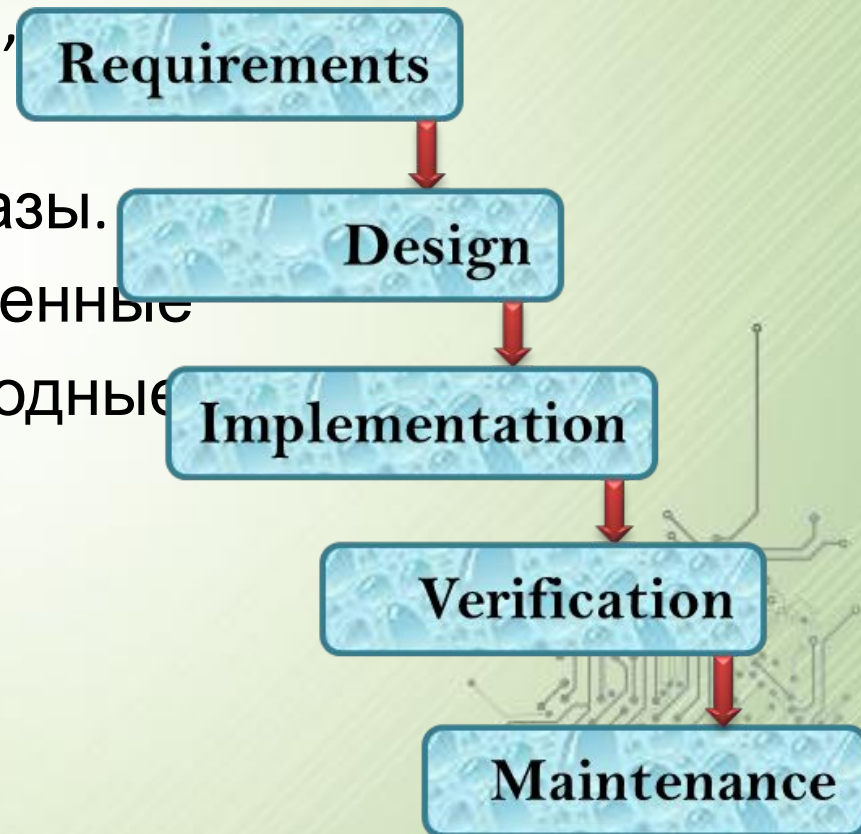
В то время, как SDLC относится к стадиям, через которые система проходит, методология является изобретением человечества. Она показывает подход для контроля событий на стадиях SDLC

**Методология** - это набор шагов, инструкций, действий и принципов, которыми следует пользоваться в той или иной ситуации

# Методологии разработки ИС

## *Каскадные методологии:*

**Waterfall:** Предусматривает, что каждая последующая фаза начинается лишь тогда, когда полностью завершено выполнение предыдущей фазы. Каждая фаза имеет определенные критерии входа и выхода: входные и выходные данные.



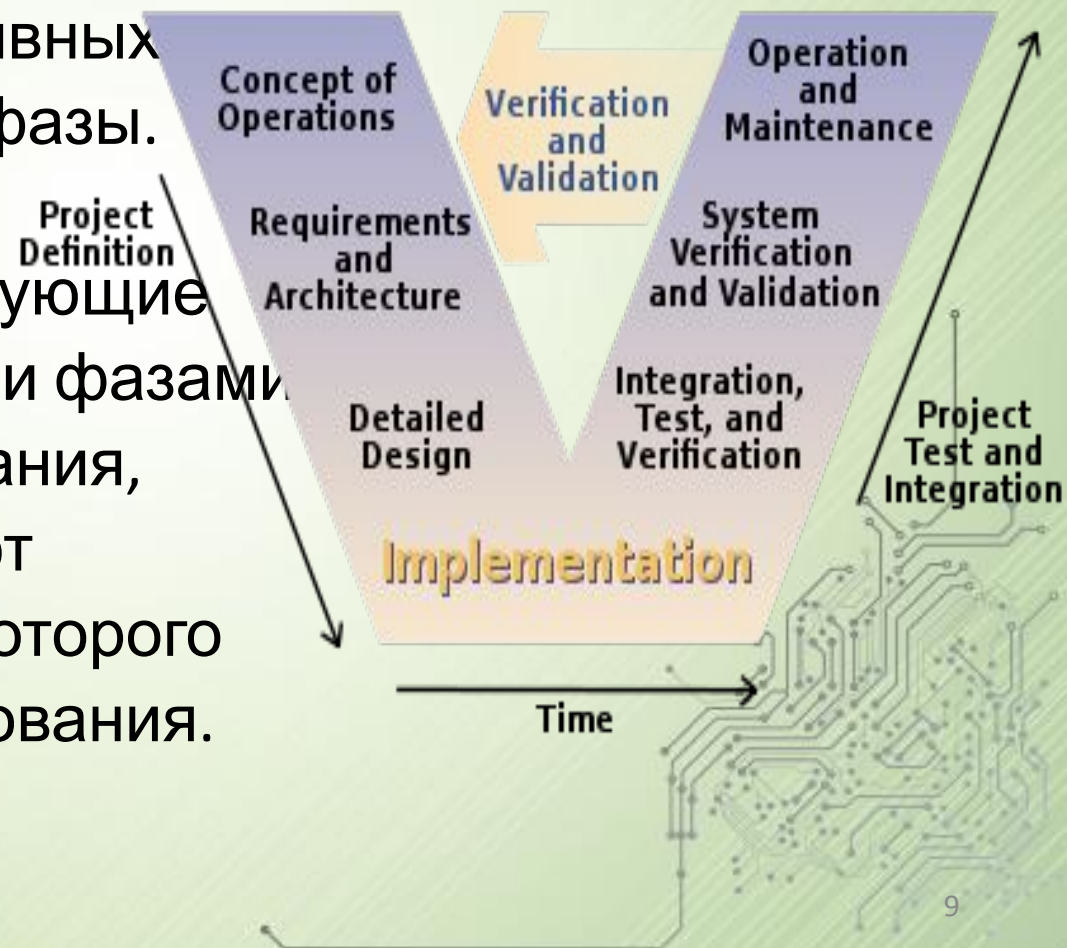


# Методологии разработки ИС

## *Каскадные методологии:*

**V-model:** Разновидность каскадной модели. Каждая последующая фаза начинается по завершению получения результативных данных предыдущей фазы.

В ней подчеркнуты взаимосвязи, существующие между аналитическими фазами и фазами проектирования, которые предшествуют кодированию, после которого следуют фазы тестирования.



# Методологии разработки ИС

## *Итерационные методологии:*

**Agile:** это семейство гибких процессов разработки (SCRUM, Extreme programming, Kanban, etc).

Ценности и принципы Agile методологии

закреплены в документе 'Agile Manifesto' (<http://agilemanifesto.org>)

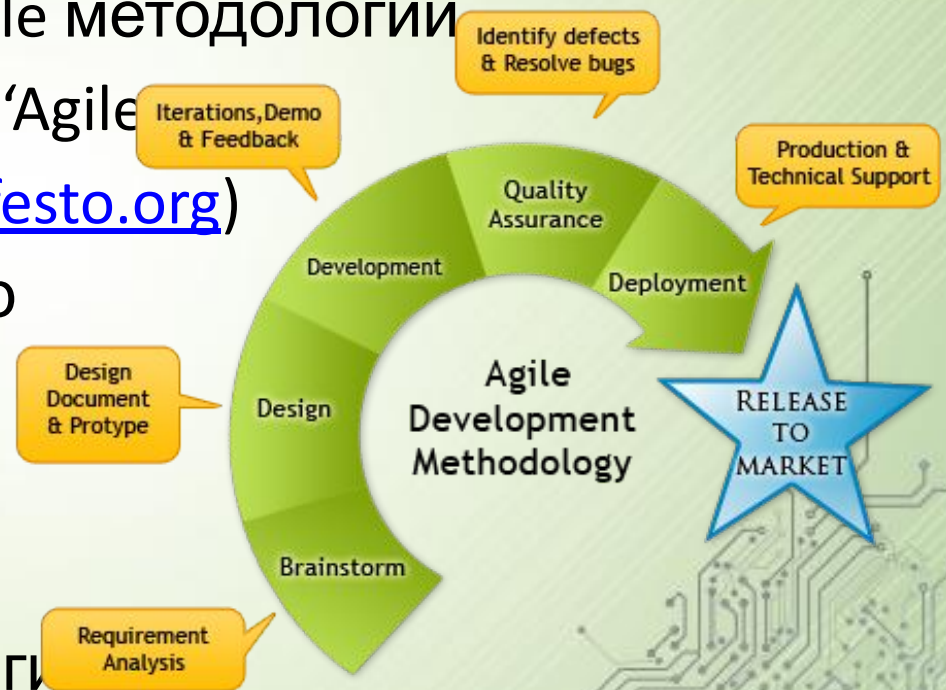
Agile-методы делают упор

на непосредственное общение лицом к лицу.

Основным результатом

работы по agile-методологии

является работающий программный продукт.



# Методологии разработки ИС

## *Итерационные методологии:*

**Rational Unified Process (RUP)** — создана компанией Rational Software. Основные принципы RUP:

- Ранняя идентификация и устранение основных рисков.
- Концентрация на выполнении требований заказчиков к исполняемой программе
- Компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта
- Постоянное обеспечение качества на всех этапах разработки проекта (продукта)
- Работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам.

# Методологии разработки ИС

**Waterfall** - есть документация, требования будут мало меняться, ведётся вся документация, весь процесс разбит на стадии и рабочие процессы

<http://bit.ly/1vbhYPx>

**Agile** - нет документации в формальных документах, часто меняющиеся требования, короткие этапы жизненного цикла

<http://bit.ly/18zgT6F>

**Waterfall vs Agile?**

<http://bit.ly/1rBa0dR>





# 4. Определение термина «Тестирование ПО»

## *Тестирование ПО – 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**

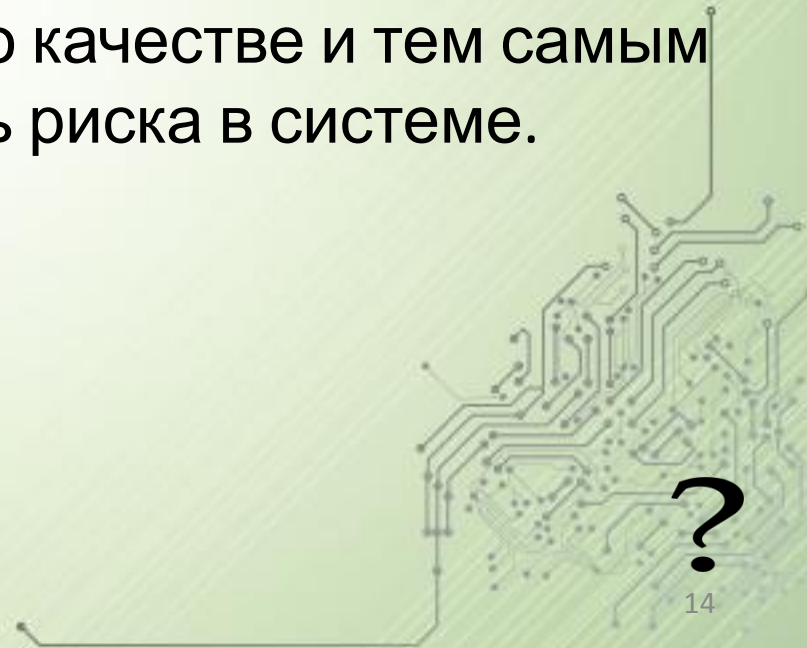


# Определение необходимости тестирования ПО

В процессе тестирования обнаруживаются дефекты в работе системы.

Анализ найденных дефектов дает возможность оценить качество программного продукта.

Таким образом руководство проекта получает необходимую информацию о качестве и тем самым уменьшается общий уровень риска в системе.



## 5. QA vs QC, Verification vs Validation

**QA** aims to prevent defects with a focus on the process used to make the product. It is a proactive quality process. The goal of **QA** is to improve development and test processes so that defects do not arise when the product is being developed.

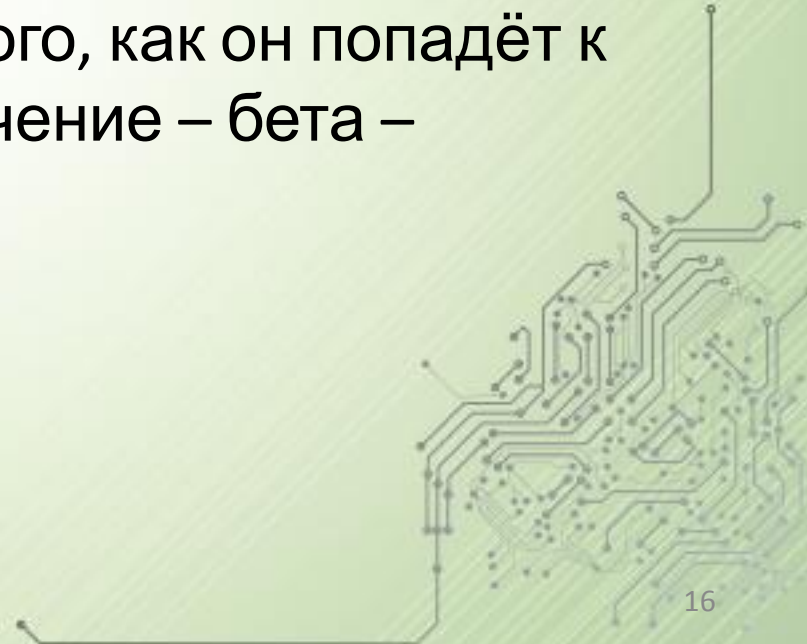
**Задача QA** – предотвратить ошибки в процессе, который используется для построения программного продукта. Это – выбор подходов, методологий, инструментов, команды, построение процессов.

# QA vs QC, Verification vs Validation

**QC** aims to identify defects in the finished product. Quality control, therefore, is a reactive process

The goal of **QC** is to identify defects after a product is developed and before it's released

**Задача QC** – нахождение ошибок в готовой версии программного продукта до того, как он попадёт к конечному заказчику (исключение – бета – тестирование)



# QA vs QC, Verification vs Validation

*QC включает в себя:*

- подготовку, анализ и тестирование требований
- написание Use Cases в случае сложных систем и необходимости более широкой детализации требования (варианты использования)
- написание Test Case headers
- заполнение Traceability Matrix – покрытие требований тестами
- написание Test Cases
- планирование выполнения Test Cases
- выполнение Test Cases
- занесение ошибок в баг – трекингтовую систему
- повторное тестирование ошибок
- повторное прохождение тест кейсов
- составление отчёта о тестировании



# QA vs QC, Verification vs Validation

*Полный цикл тестирования включает в себя:*

**Verification (верификацию)** - проверка того, что система соответствует установленным требованиям («Делаем ли мы продукт правильно?»)

**Validation (валидацию)** - подтверждение того, что система соответствует ожиданиям заказчика при ее непосредственном применении («Делаем ли мы правильный продукт?»)

Эти два раздела взяты из CMMi - набора моделей (методологий) совершенствования процессов создания программного обеспечения.

<http://ru.wikipedia.org/wiki/CMMI>



# 6. Роли и артефакты в проектной команде

**Project Manager, PM** — это специалист в области управления проектами, который несет ответственность за планирование, подготовку и исполнение конкретного проекта

*Основные артефакты (документы):*

- PMP – Project Management Plan
- WBS – Work Breakdown Structure
- Project Status Report

*Где почитать и посмотреть:*

<http://bit.ly/18ZjTwF>



# Роли и артефакты в проектной команде

**Business Analyst, BA** — это специалист, использующий методы бизнес-анализа для аналитики потребностей деятельности организаций с целью определения проблем бизнеса и предложения их решения

*Основные артефакты (документы):*

- 1) Functional Requirements or BRS
- 2) Technical Requirements
- 3) Use Case documents

*Где почитать и посмотреть:*

<http://bit.ly/1vbpHNE>

# Роли и артефакты в проектной команде

**Software Architect, SA** — это специалист, определяющий начальную структуру системы, основные элементы системы, их особенности и поведение. Также он представляет точку зрения пользователя на то, какой должны быть система в разрезе основных бизнес сценариев и моделей поведения

*Основные артефакты (документы):*

- 1) SyRS – System Requirements Specification
- 2) TD – Technical design

*Где почитать и посмотреть:*

<http://bit.ly/1sxlSOG>



# Роли и артефакты в проектной команде

**Разработчик (Developer, Dev)** — это специалист, кодирующий функциональности программного продукта на выбранном языке программирования с использованием технологий, определённых системным архитектором.

*Основные технологии:*

- 1) Java (Джава)
- 2) .Net (дот нет)

*Основные артефакты (документы):*

- 1) Все документы с требованиями (all requirements documents – functional, non-functional)
- 2) Technical Design
- 3) Coding Guidelines
- 4) Исходный код программного продукта
- 5) Unit tests

# Роли и артефакты в проектной команде

**Руководитель группы тестирования (Test Lead, Test Manager, TL)** — это специалист, отвечающий за внедрение QA и контроль QC активностей на всех этапах разработки программного обеспечения.

Основные артефакты (документы):

- 1) Project Management Plan
- 2) Test Plan
- 3) Traceability Matrix
- 4) Testing Schedule
- 5) Test Execution Summary Report





# Роли и артефакты в проектной команде

**Тестировщик (Software tester)** — это специалист, отвечающий за QC активности.

*Основные артефакты (документы):*

- 1) All requirement documents
- 2) Requirements Check List
- 3) Test Plan
- 4) Technical Design
- 5) Traceability Matrix
- 6) Test Cases
- 7) Test Scripts
- 8) Defects / Enhancements in bug – tracking system
- 9) Test Execution Report

# Роли и артефакты в проектной команде

В зависимости от сложности проекта и квалификации специалиста, тестировщики могут относиться к различным группам, а именно:

- Testers
- Test designers
- Automation test engineers (<http://bit.ly/1k63Q5i>)
- QA engineers (<http://bit.ly/1EXARO>)



# 7. Зачем нужны тестировщики на проекте?

- Предоставляют заинтересованным сторонам информацию, достаточную для принятия обоснованного решения о релизе тестируемого продукта, передаче на следующий этап разработки или в качестве готовой системы пользователям
- Должны найти и задокументировать баги до того как их найдут пользователи
- «Смотрят на продукт глазами пользователя» и проверяют основные сценарии использования продукта
- Обладают знаниями и навыками позволяющими проектировать и выполнять эффективные тесты?

## 8. Анализ требований к программному обеспечению

- **Требования** – это функциональная характеристика системы, необходимая заказчику для того, что бы решить проблему или достигнуть поставленных целей
- **Требования** – это совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации
- **Требования** – это точно сформулированное описание совокупности полезных для пользователя характеристик, ожидаемых от программного продукта

*Где почитать:*

<http://slidesha.re/1qGyW8O>

# Анализ требований к программному обеспечению

Требования принято разделять по характеру использования

## ***Функциональный характер:***

- Бизнес – требования
- Пользовательские требования
- Функциональные требования

## ***Нефункциональный характер:***

- Бизнес – правила
- Системные требования и ограничения
- Атрибуты качества
- Внешние системы и интерфейсы
- Ограничения





# Анализ требований к программному обеспечению

## *Зачем и кому нужны требования?*

**Developer** – согласно требованиям пишется программный код, который реализует требуемые функциональные и нефункциональные требования

**Tester** – согласно требованиям пишутся тест кейсы, которые тестируют функциональные и нефункциональные аспекты работы системы

## *В целом для проекта:*

На основании требований определяются трудоёмкость, сроки и стоимость разработки программного продукта

# Анализ требований к программному обеспечению

## *Как собрать требования:*

- 1) Интервью, собрания (meetings, митинги) с представителями заказчика
- 2) Мозговой штурм, использование навыков участников проекта и их опыта
- 3) Наблюдение за производственной деятельностью
- 4) Анализ нормативной документации
- 5) Анализ моделей деятельности
- 6) Анализ конкурентных продуктов
- 7) Анализ предыдущих версий системы



# Анализ требований к программному обеспечению

## *Что делать, если нет требований?*

- 1) Запросить соответствующий документ
- 2) Запросить источник пожеланий заказчика (backlog)
- 3) Провести серию встреч (митингов) для выяснения требований в телефонном режиме, по Skype или организовать Business trip
- 4) Предоставление заказчику своего видения (vision) требований
- 5) Предоставление нескольких вариантов с плюсами и минусами каждого

# Анализ требований к программному обеспечению

## *Правила работы команды тестирования:*

- 1) Каждый документ должен утверждаться заказчиком – устно или письменно
- 2) После каждого важного митинга должно быть разослано письмо всем участникам с Minutes of Meeting, где кратко описаны основные темы, которые обсуждались, и решения, которые были приняты

# Анализ требований к программному обеспечению

## *Критерии требований:*

- 1) Правильность
- 2) Полнота
- 3) Понятность
- 4) Измеримость
- 5) Тестируемость
- 6) Непротиворечивость

## *Как проверять требования:*

Для проверки требований используется Check List, где по колонкам отмечены основные критерии требований, а в столбик выписаны заголовки требований



# Анализ требований к программному обеспечению

## Правильность

Каждое требование должно точно описывать то, что должно быть разработано

### *Где проверяется?*

На прототипе системы или в документации

### *Пример:*

- 1) Веб – сервисы должны реализовывать функционал передачи данных между клиентскими терминалами
- 2) Front – End сайта должен уметь регистрировать пользователя и показывать данные о его посещении
- 3) Функциональный модуль «Платёжные карты» должен проводить валидацию кредитной карты клиента
- 4) Уровень шума при работе стиральной машины в режиме отжима должен составлять 135 милликельвинов

# Анализ требований к программному обеспечению

## Полнота

- Все требования задокументированы
- Каждое требование содержит всю информацию, необходимую для проектирования, разработки и тестирования

### *Где и как проверяется?*

- На прототипе системы
- На созданной модели системы
- Путём опроса конечных пользователей и экспертов

### *Пример:*

- 1) Система должна уметь решать уравнение  $ax^2+bx+c=0$
- 2) Back End банковской системы должен автоматически обновлять курс валют каждые 10 минут и обновлять БД
- 3) Функциональный модуль «Платёжные карты» должен проводить валидацию кредитной карты клиента

# Анализ требований к программному обеспечению

## Понятность

- Одинаковая интерпретация требования (недвусмысленность) Требование описано - четко, просто, кратко
- Все специальные термины описаны и определены

### *Где и как проверяется?*

- Вычитываются все требования в функциональной и нефункциональной спецификации

### *Пример:*

- 1) Все данные авторизованных пользователей должны отправлять на сервер по защищенному протоколу https
- 2) Сайт должен корректно отображаться в поддерживаемых браузерах: IE9, IE10, IE11, Chrome, FireFox, Safari
- 3) При регистрации пользователь должен выбрать пол Male/Female выбрав соответствующий radio-button

# Анализ требований к программному обеспечению

## Измеримость

- Требование должно быть сформулировано так, что бы можно было доказать соответствие системы предъявленному требованию
- Требование не должно содержать неизмеримых формулировок

### *Где и как проверяется?*

- Вычитываются все требования в функциональной и нефункциональной спецификации на предмет присутствия слов, которые не гарантируют измеримость

### *Пример неизмеримых формулировок:*

Легко, лучше чем, более эффективно, качественно, максимально, минимально

Acceptable, adequate, as much as, between, depends on, better, faster, should work fine, where appropriate

# Анализ требований к программному обеспечению

## Тестируемость

- Требование должно быть сформулировано так, что бы тестировщик, прочитав его, смог написать тест кейс, который протестирует данное требование

### *Где и как проверяется?*

- Совокупность измеримости и понятности в сочетании с доступными механизмами проверки

### *Пример:*

Зерно монитора Samsung SyncMaster S27B350 должно составлять 0,23 мм





# Анализ требований к программному обеспечению

## Непротиворечивость

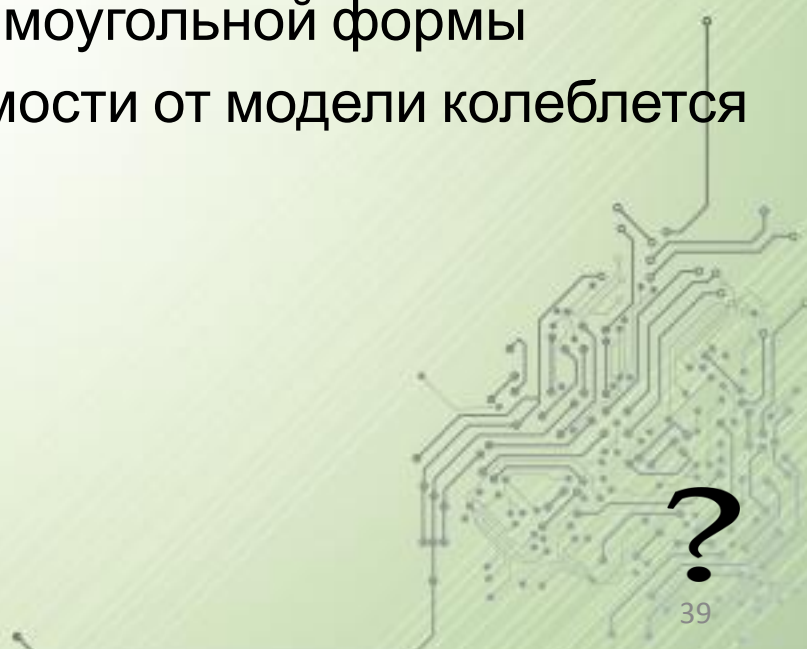
Требование не должно противоречить другим требованиям

***Где и как проверяется?***

- Вычитывание спецификаций

***Пример:***

- 1) Столешница должна быть прямоугольной формы
- 2) Радиус столешницы в зависимости от модели колеблется от 80 см до 1,5 м



# Домашнее задание

**NB! Все, кроме перевода с английского на русский, выполняется на английском языке**

1. Прочитать про модели жизненного цикла ПО: Каскадные, Итерационные, Спиральная. Проанализировать методологии и описать:
  - Waterfall & Agile (сравнительная характеристика)
  - RUP & V-model (сравнительная характеристика)
  - Спиральная модель (основные принципы, преимущества, недостатки)

Формат - Microsoft Power Point, имя файла – Methodologies\_[\[Name\]](#)\_[\[Surname\]](#).ppt

2. Перевести на английский язык слайды 34 – 39 урока №1

Формат - Microsoft Word, имя файла – Slide\_Translation\_[\[Name\]](#)\_[\[Surname\]](#).doc

# Домашнее задание

3. Прочитать и проанализировать презентацию по тестированию требований Testing\_The\_Requirements.pdf, перевести слайды 24 и 25 на русский или украинский язык

Формат - Microsoft Word, имя файла –  
Requirements\_Slide\_Translation\_[Name]\_[Surname].doc

4. Перевести на русский или украинский язык  
<http://bit.ly/1oUuY7M>

Формат - Microsoft Word, имя файла  
–Testing\_Introduction\_Translation\_[Name]\_[Surname].doc

5. Сформулировать пример требований для смартфона, которые
  - Удовлетворяют всем критериям (3 примера)
  - Не удовлетворяют ни одному из критериев (по примеру на каждое требование)

Формат - Microsoft Excel, имя файла –  
smartphone\_Requirements\_[Name]\_[Surname].xls

# Домашнее задание

- 1) Установить Tortoise SVN (ссылка на скачивание <http://bit.ly/1IJZUEw>)
- 2) Создать папку с именем “с:\SVN”, в которой будут храниться файлы из репозитория
- 3) Нажать на неё правой кнопкой и выполнить операцию **SVN Checkout**
- 4) Ввести ссылку на репозиторий  
`svn://134.249.184.92/repo/09092014/trunk`
- 5) При запросе credentials:  
**Login:** user\*(в зависимости от присвоенного номера)  
**Password:** 12
- 6) Зайти в свою папку (“User\*”), создать папку HomeWork1 и скопировать туда пять файлов, которые будут сделаны в процессе выполнения домашнего задания
- 7) Нажать на вашей папке (“User\*”) правой кнопкой и выполнить операцию **SVN commit**

# Домашнее задание

Пять файлов, которые будут сделаны в процессе выполнения домашнего задания, необходимо вложить в ОДНО письмо!

Тема письма: Homework\_Lesson1\_[\[Name\]](#)\_[\[Surname\]](#)

Тело письма:

Hello Artem,

I have completed my homework.

Please find files listed below attached.

Lifecycles – [\[filename.fileextension\]](#)

[\[List ALL you files according to example above here....\]](#)

Faced issues and difficulties:

[\[describe them\]](#)

Thank you, [\[Name\]](#)[\[Surname\]](#)

Домашнее задание отправить на [artem@testclub.com.ua](mailto:artem@testclub.com.ua)