

## Основы программной инженерии (ПИ)

### Технологии разработки ПО. Управление требованиями

#### План лекции:

- понятие требования к ПО;
- виды и уровни требований;
- процесс разработки требований;
- методы сбора и анализа требований;
- документирование требований.

#### На прошлой лекции:

### 1. Жизненный цикл разработки программного обеспечения

#### Определение:

**Жизненный цикл разработки программного обеспечения –**

это период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент полного его изъятия из эксплуатации.



это ряд событий, происходящих с ПО в процессе его создания и использования



## 2. Инженерия требований

### *Цели разработки требований*

- обеспечение наиболее полного и точного отражения условий или возможностей, необходимых заказчику для решения его проблем и достижения бизнес-целей;
- снижение затрат на разработку, обслуживание и поддержку сложного программного обеспечения.

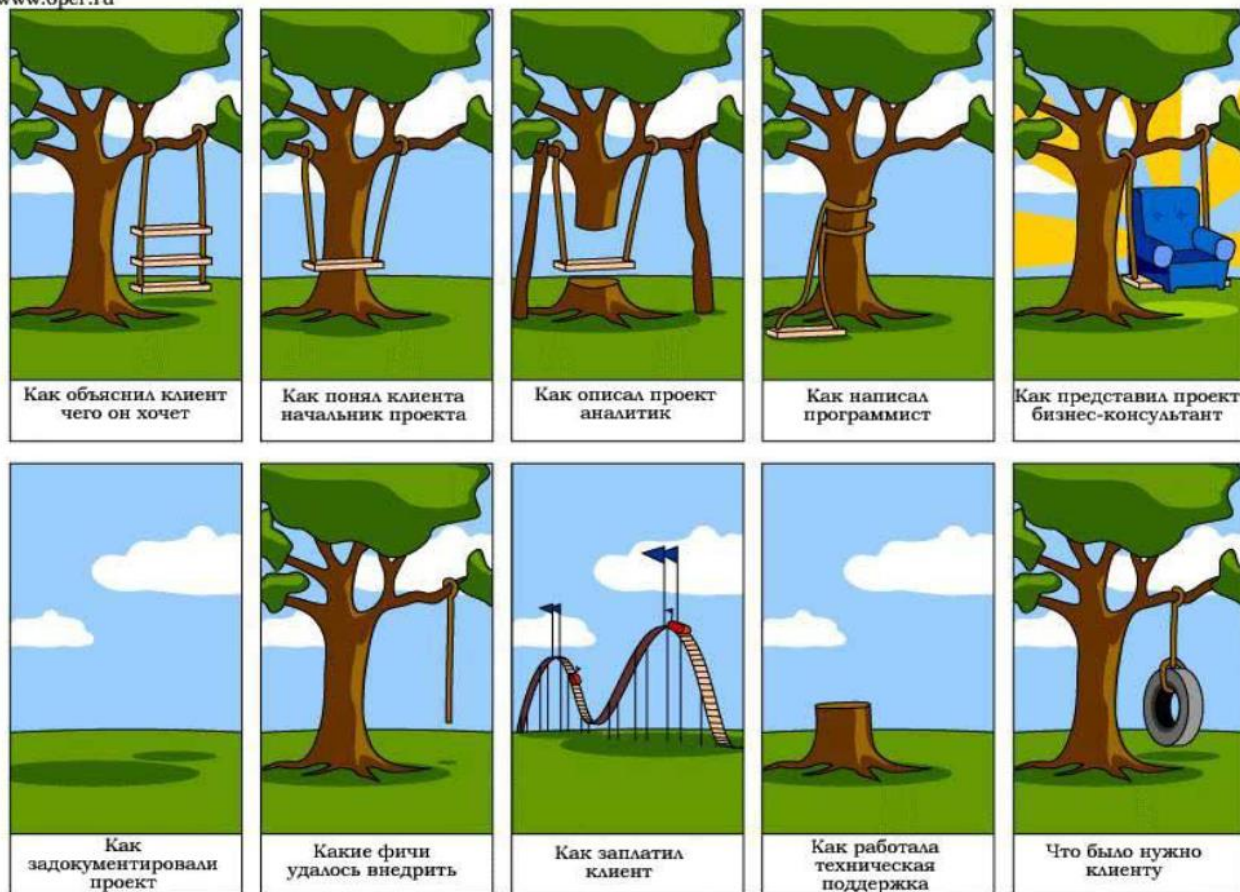
### *Анализ требований к системе:*

- определение требований;
- разработка требований;
  - выявление требований;
  - анализ требований;
- документирование и организация требований;
- изменение требований;
- планирование и управление требованиями.

### *Основная сложность:*

**«Самой сложной задачей при создании программной системы является точное определение того, что требуется создать...  
Ни одна задача не приносит такого же вреда конечной системе в случае ошибки.  
И нет ни одной задачи настолько же сложной в исправлении последствий.»**

***Фредерик Брукс***



**10 слайд** – что хотел клиент.

**1 слайд** – как он это объяснил.

**2 слайд** – как понял руководитель проекта.

**! Цена ошибки напрямую зависит от этапа, на котором она возникла.**

## Проблемы определения требований

### Проблемы определения требований

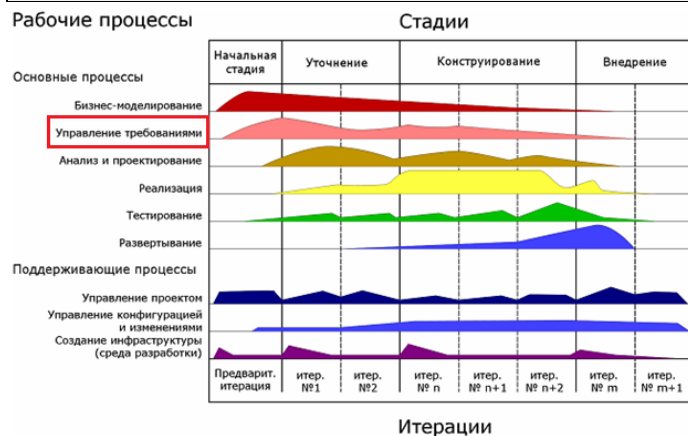
- ✓ *разработка требований – самая сложная часть проектирования ПО;*
- ✓ *требования постоянно меняются;*
- ✓ *требования могут быть;*
  - *неясны;*
  - *двусмысленны;*
  - *противоречивы;*
- ✓ *спецификации могут быть неполны;*
- ✓ *пользователи, излагающие требования, могут быть непредставительны (некомпетентны).*

### 3. Определение требования

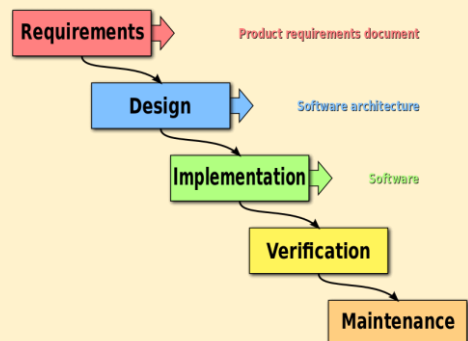
#### Определение:

##### Требование –

это утверждение, которое идентифицирует эксплуатационные, функциональные параметры, характеристики или ограничения проектирования продукта или процесса, которое однозначно, проверяемо и измеримо.



##### Место требований в жизненном цикле ПО



#### Определение (IEEE 1990)

##### Требование

- ✓ *Условие или возможность, необходимые пользователю для решения его задач или достижения цели (1)*
- ✓ *Условие или возможность, которым должна отвечать или которыми должна обладать система или ее компонента, чтобы удовлетворить контракт, стандарт, спецификацию или иной формальный документ (2)*
- ✓ *Документированное представление условия или возможности, указанное в (1) или (2)*

## Управление требованиями

<b>Управление требованиями</b>	<i>процесс, включающий:</i> ✓ <i>идентификацию, выявление, документацию, анализ, отслеживание, приоритизацию требований, достижение соглашений по требованиям и затем управление изменениями и уведомление заинтересованных лиц.</i>
--------------------------------	---

**Управление требованиями** – непрерывный процесс на протяжении всего жизненного цикла продукта.

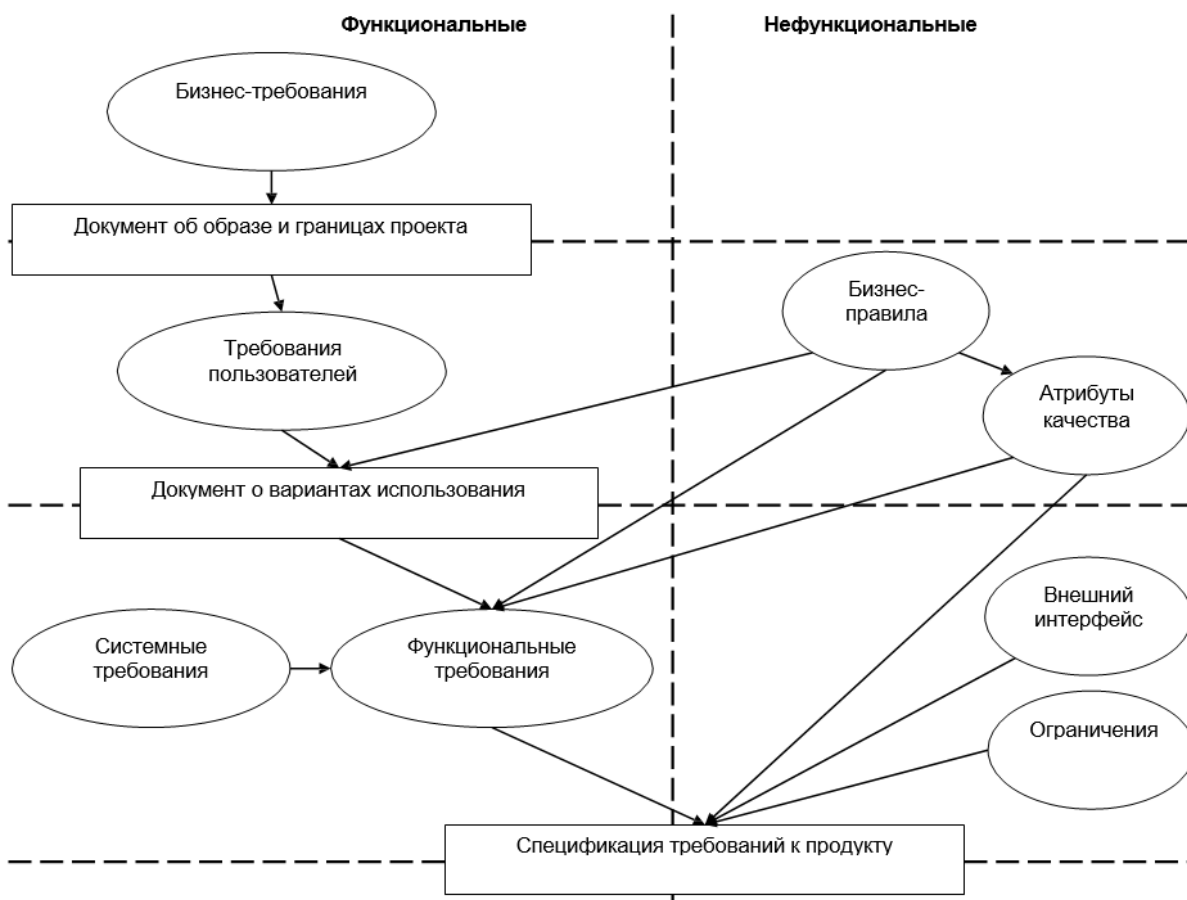
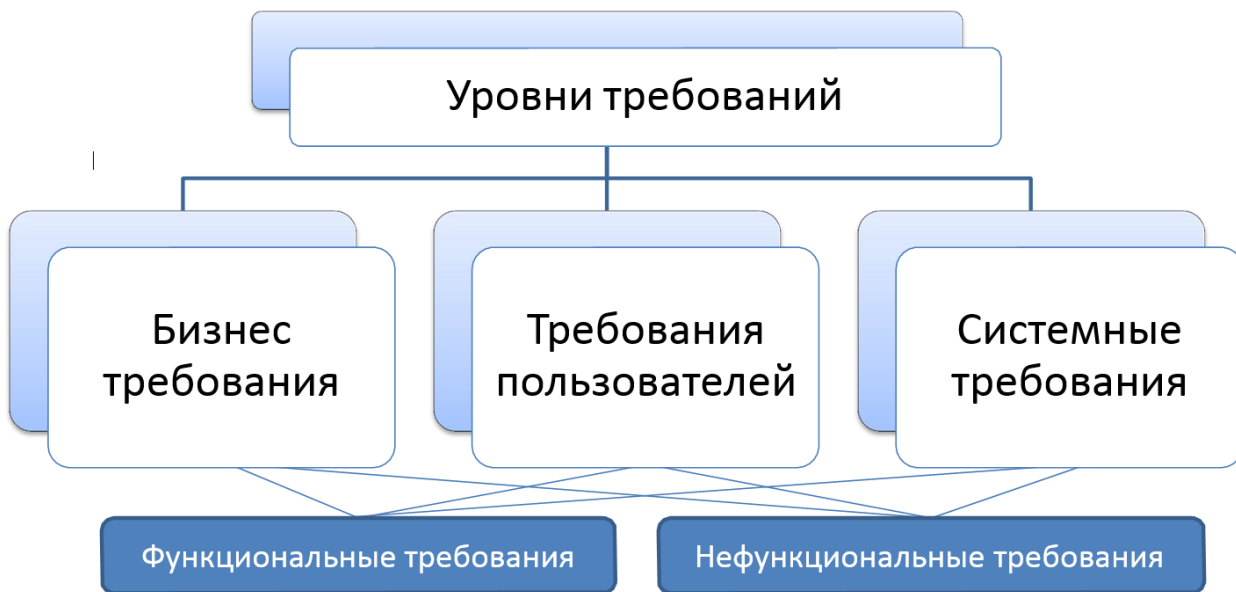
### *Свойства требований:*

- *корректность (correct);*
- *однозначность (unambiguous);*
- *полнота (complete);*
- *непротиворечивость (consistent);*
- *приоритезация (prioritized);*
- *проверяемость (verifiable);*
- *модифицируемость (modifiable);*
- *отслеживаемость (traceable).*

### *Цели разработки требований*

- обеспечение наиболее полного и точного отражения условий или возможностей, необходимых заказчику для решения его проблем и достижения бизнес-целей;
- снижение затрат на разработку, обслуживание и поддержку сложного заказного программного обеспечения.

## Классификация требований



### **Бизнес-требования**

- содержат высокоуровневые задачи и цели организации-разработчика или заказчиков системы.

Отвечает на вопрос **«Зачем?»**

*Пример, для приложения «Калькулятор» - «Приложение должно сократить время, необходимое на расчеты для курсового проекта»*

### **Требования пользователей**

- требования пользователей описывают цели и задачи, которые пользователям позволит решить система.
- пользовательские требования – описание на естественном языке функций, выполняемых системой и ограничений, накладываемых на нее.

Отвечает на вопрос: **«КТО и ЧТО?»**

### **Системные требования**

- системные требования определяют функциональность и характеристики системы, которую должны построить разработчики, для того чтобы пользователи смогли выполнить свои задачи (в рамках бизнес-требований)
- термином системные требования обозначают высокоуровневые требования к продуктам, которые содержат многие подсистемы (программное обеспечение, оборудование и т.д. Люди – часть системы, поэтому некоторые функции системы могут распространяться и на людей).



### Функциональные требования – «Что делает?»

- бизнес-требования
  - формулируются заказчиками
  - описывают цели, которые требуется достичь с данной системой
- требования пользователей
  - какие задачи можно решить с помощью системы
- собственно, сами функциональные требования
  - определяются функциональность, которую необходимо реализовать

**Функциональные требования** определяют функции, которые выполняет система, и зависят от потребностей пользователей и типа решаемой задачи.

**Функциональные пользовательские требования** описывают функции в обобщенном виде. Выполняя детализацию этих требований, разработчики формируют более подробное и точное описание сервисов системы – функциональные системные требования.

### Нефункциональные требования – «Как делает?»

- требования к характеристикам качества
  - требования к надежности
  - требования к совместимости
  - требования к эффективности
  - требования к гибкости
  - требования к эргономике
- ограничения
  - соответствия стандартам и правилам
  - бюджет
  - сроки
  - предопределенные архитектурные решения
  - и т.д.

Нефункциональные требования определяют характеристики и ограничения системы и не связаны непосредственно с функциональными требованиями. Они формируются на основе имеющихся атрибутов качества, требований к внешнему интерфейсу и ограничений.



Пример:



### Что является функциональными требованиями?


1. Работает в режимах: «Обычный», «Инженерный» и «Программист»
2. Выполняет арифметические операции
3. Совместим с Windows
4. Выполняет логические операции
5. Вычисляет сложные функции, ...
6. Время вычисления тригонометрических функций меньше 1 минуты
7. Наличие графического пользовательского интерфейса
8. Наличие справки
9. Справка выводится в формате Windows
10. Память, отводимая на одно число равна ...
11. Реализация памяти
12. Поддержка скобок

**Функциональные требования выделены:**

- 1. Работает в режимах: «Обычный», «Инженерный» и «Программист»**
- 2. Выполняет арифметические операции**
3. Совместим с Windows
- 4. Выполняет логические операции**
- 5. Вычисляет сложные функции, ...**
6. Время вычисления тригонометрических функций меньше 1 минуты
- 7. Наличие графического пользовательского интерфейса**
- 8. Наличие справки**
9. Справка выводится в формате Windows
10. Память, отводимая на одно число равна ...
- 11. Реализация памяти**
- 12. Поддержка скобок**

## Пример:

Для каких целей?

 <p>Интерфейс: для заполнения</p> <p>Компонент: ручка</p> <p>Компонент: чаша</p> <p>Интерфейс: для жидкости</p> <p>Интерфейс: для руки</p> <p>Интерфейс: для стола</p> <p>Окружение: гравитация</p>	<p><b>Пользователь:</b></p>  <p><b>Оптимистичный разработчик:</b> - Это понятная и интуитивная фича, документация тут не нужна.</p>
	

Способность простой чашки удовлетворять сформулированной цели зависит от

- свойств, которые обусловлены взаимодействием ее компонентов;
- соответствующих интерфейсов;
- ее корректного включения в общую систему – чашка удерживается и переносится человеческой рукой;
- внешних условий – в условиях невесомости для достижения цели явно потребуется другое решение.

### ***Что не является требованиями:***

- Детали архитектуры
- Детали реализации
- Сведения о планировании
- Сведения о тестировании
- Проектная информация:
  - Инфраструктура разработки
  - Процесс разработки
  - Команда разработки

#### **Треугольник ограничений**

Мы сделаем проект:

- ***быстро***
- ***качественно***
- ***недорого***

**! Выберите 2 из 3-х !**

**С чего же начать разработку требований?** Начинать нужно с цели – для чего вообще нам что-то делать.

**Ответить на основные вопросы:**

**1. Зачем?** – Начинать надо с **цели**: зачем это делать?

*Например:*

- процесс заказа товаров/услуг считается автоматизированным, если >90% компаний-партнеров делают заказы через систему;
- разработка приложения «Калькулятор» позволит автоматизировать процесс вычислений.

**2. Что?** – Что конкретно мы будем делать, чтобы прийти к цели.

**3. Как?** – Как мы это реализуем?

**4. Когда?** – Полезно всю эту информацию документировать и представлять в виде таблиц и диаграмм.

**Шаблон полного описания варианта использования:**

**Название** <краткая фраза в виде глагола в неопределенной форме совершенного вида отражающая цель>

**Контекст использования** <уточнение цели, при необходимости - условия ее нормального завершения>.

**Область действия** <ссылка на рамки проекта>. Например - подсистема бухгалтерского учета.

**Уровень** <один из трех: обобщенный, цели пользователя, подфункции>.

**Основное действующее лицо** <имя роли основного актора или его описание>.

**Участники и интересы** <список других акторов-участников прецедента с указанием их интересов>.

**Предусловие** <то, что ожидается, уже имеет место>.

**Минимальные гарантии** <что гарантируется акторам-участникам>.

**Гарантии успеха** <что получают акторы-участники в случае успешного достижения цели>.

**Триггер** <то, что «запускает» вариант использования, обычно - событие во времени>.

**Основной сценарий** <здесь перечисляются шаги основного сценария, начиная от триггера и вплоть до достижения гарантии успеха>.

**Формат описания** <Номер шага> <Описание действия>

**Расширения** <здесь последовательно описываются все альтернативные сценарии>. Каждая из альтернатив привязана к шагу основного сценария.

**Формат описания** <Номер шага.Номер расширения> <Условие>:<Действие или ссылка на подчиненный вариант использования>.

В случае, если альтернативный сценарий не удастся описать одной строкой - применяется следующий формат.

Начиная со строки, следующей после описания расширения, идет описание его действий в формате основного сценария:

<Номер шага.Номер расширения.Номер шага расширения> <Действие>

**Список изменений в технологии и данных** <что гарантируется актерам-участникам>. Например - в случае неудавшейся транзакции все данные, имевшиеся в системе до ее начала, сохраняются неизменными.

**Вспомогательная информация** <дополнительная информация, полезная при описании варианта использования>.

## Табличные представления варианта использования

Таблица в 2 колонки:	
Актер	Действие
Пользователь	Формирует запрос на поиск заказов
Система	Отображает список заказов
Пользователь	Выбирает требуемый заказ
Система	Показывает подробную информацию по заказу

Таблица в 3 колонки:		
№ шага	Пользователь	Система
1	Делает запрос на поиск заказов	Отображает список заказов
2	Выбирает требуемый заказ	Показывает подробную информацию по заказу

**Схема требований** (*Алистер Коберн. Современные методы описания функциональных требований к системам.*)

**Приемлемая схема требований**

---

**Раздел 1. Цель и область действия**

- 1a. Что представляют собой общая область действия и цель?
- 1b. Участники (Кого это интересует?)
- 1c. Что входит в область действия и что нет?

**Раздел 2. Используемые термины/Глоссарий**

**Раздел 3. Варианты использования**

- 3a. Основные действующие лица и их общие цели
- 3b. Варианты использования для бизнес-процессов
- 3c. Системные варианты использования

**Раздел 4. Используемая технология**

- 4a. Какие технологические требования предъявляются к данной системе?
- 4b. С какими системами будет взаимодействовать данная, каковы требования?

**Раздел 5. Другие требования**

**5a. Процесс разработки**

- Q1. Кто участвует в проекте?
- Q2. Какие оценки проекта будут отражены (простой, ранний, быстрый или гибкий)?
- Q3. Какая обратная связь или прозрачность проекта нужна пользователям и организаторам?
- Q4. Что мы можем купить, что должны построить, с кем конкурируем?
- Q5. Какие еще существуют технологические требования (тестирование, установка и т.д.)?
- Q6. От чего зависит развитие проекта?

**5b. Бизнес-правила**

**5c. Производительность**

**5d. Эксплуатация, безопасность, документация**

**5e. Использование (простота использования)**

**5f. Сопровождение и мобильность**

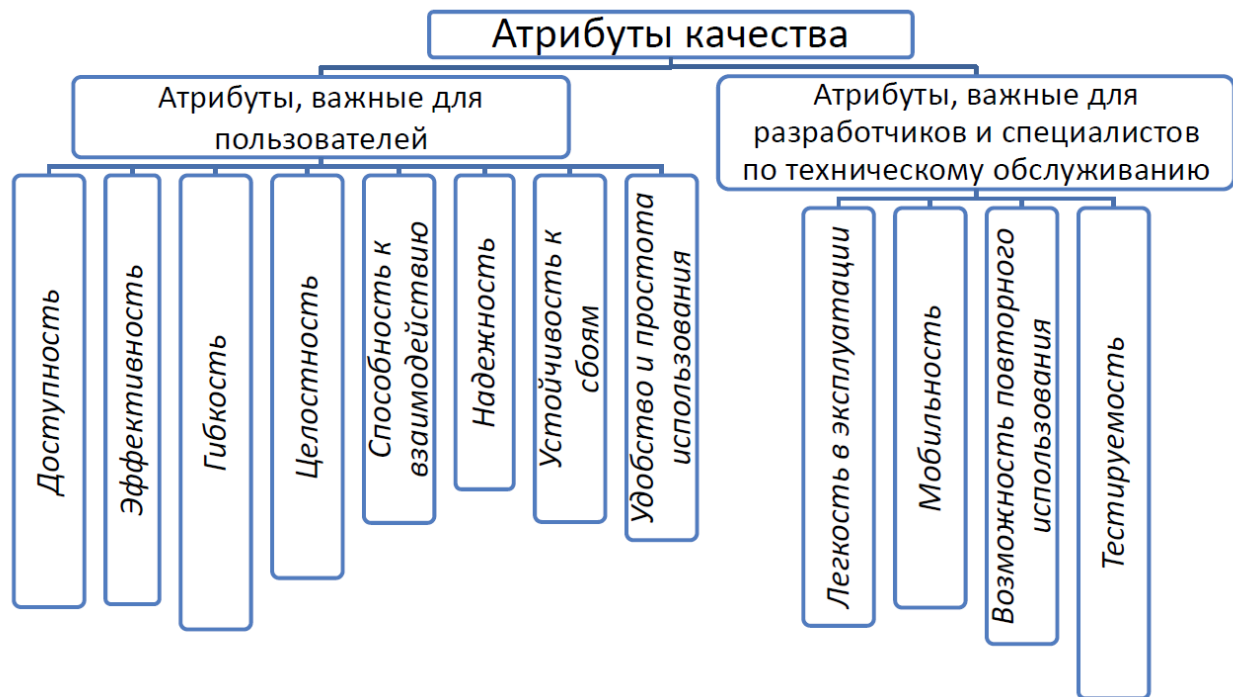
**5g. Нерешенные или отложенные вопросы**

**Раздел 6. Людские резервы, правовые, политические, организационные вопросы**

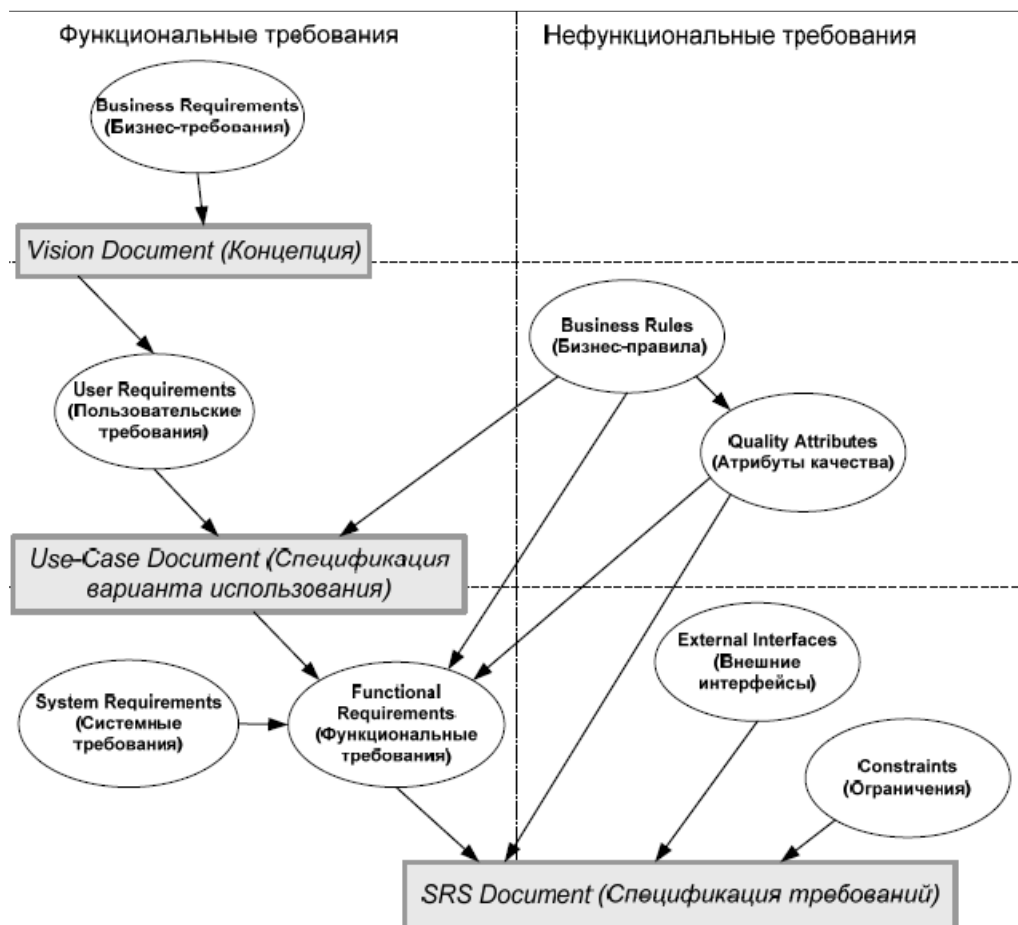
- Q1. Как влияют людские резервы на функционирование системы?
- Q2. Какие существуют правовые и политические требования?
- Q3. Какие последствия для людей будет иметь создание этой системы?
- Q4. Каковы требования к обучению?
- Q5. Какое влияние оказывает система на окружающее сообщество?



## Классификация атрибутов качества



## Варианты формализации требований



## Разработка требований

<b>Разработка требований</b> <ul style="list-style-type: none"><li>– выявление требований</li><li>– анализ требований</li></ul>	<b>Результат</b> <ul style="list-style-type: none"><li>– спецификация требований</li></ul>
---	--

## Источники сбора требований

<b>Выявление требований</b>  <b>! ВАЖНО:</b> заказчик $\neq$ пользователь	<b>Заинтересованные лица</b> <ul style="list-style-type: none"><li>– заказчики</li><li>– менеджеры</li><li>– пользователи<ul style="list-style-type: none"><li>○ операторы</li><li>○ менеджеры</li><li>○ ...</li></ul></li><li>– разработчики</li><li>– служба поддержки</li><li>– другие лица</li></ul>
<b>Выявление требований</b>	<b>Планирование:</b> <ul style="list-style-type: none"><li>– цели выявления требований</li><li>– стратегии и процессы выявления требований</li><li>– результаты усилий по выявлению требований</li><li>– оценки календарного плана и ресурсов</li><li>– риски, связанные с выявлением требований</li></ul>
	<b>Проблемы определения требований:</b> <ul style="list-style-type: none"><li>– ожидания пользователей</li><li>– умение оценить противоречивые требования</li><li>– недостаточные требования</li><li>– умение понять требования пользователей</li></ul>
<b>Методы выявления требований:</b>	<ul style="list-style-type: none"><li>– собеседование (интервьюирование);</li><li>– анкетирование;</li><li>– проведение совещаний («разъясняющие встречи»);</li><li>– сессии по выявлению требований (мозговой штурм);</li><li>– наблюдения («on-site customer» – «присутствующий заказчик»);</li><li>– раскадровка (storyboard);</li><li>– создание и демонстрация работающих прототипов;</li><li>– ролевые игры.</li></ul>

Уровень требований	Область	Точка зрения	Цель
Пользовательские требования	Область проблем	Пользователь (представитель заинтересованной стороны)	Определяет - <b>что</b> пользователь желает достичь с помощью создаваемой системы. Следует избегать формулировки конкретных решений.
Системные требования	Область решения	Аналитик	Абстрактно определяет - <b>как</b> система будет удовлетворять пользовательским требованиям. Следует избегать точных описаний реализации предлагаемых решений.
Системные спецификации (архитектура системы)	Область решения	Архитектор	Определяет - <b>как</b> конкретная архитектура системы будет удовлетворять системным требованиям.

## ПРОЦЕСС РАЗРАБОТКИ ТРЕБОВАНИЙ

Разработка требований – это первый из основных процессов создания программных систем. Этот процесс состоит из следующих основных этапов



*Анализ предметной области:*



# Детально о требованиях в регламентирующих документах

## 1. Требования в своде знаний SWEBOK



## 2. Разработки IEEE:

- IEEE 1362 “Concept of Operations Document”.
- IEEE 1233 «Guide for Developing System Requirements Specifications».
- IEEE Standard 830-1998, «IEEE Recommended Practice for Software Requirements Specifications»
- IEEE Standard Glossary of Software Engineering Terminology/IEEE Std 610.12-1990
- IEEE Guide to the Software Engineering Body of Knowledge (1) - SWEBOK®, 2004.

## 3. ГОСТы:

- ГОСТ 34.601-90. Информационная технология. Автоматизированные системы. Стадии создания.
- ГОСТ 34.602-89. Информационная технология. Техническое задание на создание автоматизированной системы
- ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению