

Введение в разработку программного обеспечения (ИС и ЦД)

Технологии разработки ПО.

Управление требованиями. Формализация функциональных требований

План лекции:

- назначение диаграммы вариантов использования;
- компоненты диаграммы вариантов использования;
- примеры.
- понятие требования к ПО;
- виды и уровни требований;
- процесс разработки требований;
- методы сбора и анализа требований;
- документирование требований.

1. 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:

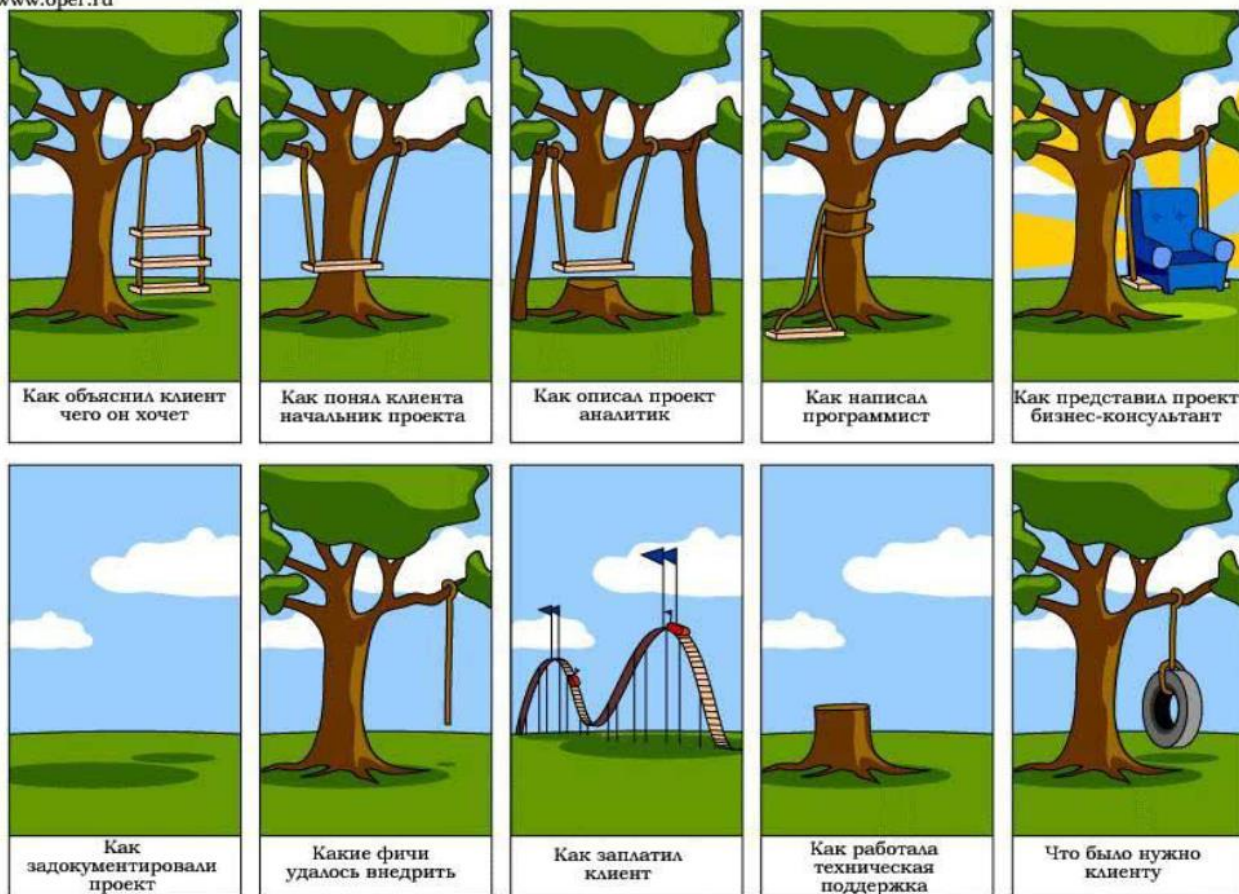
- определение требований;
- разработка требований;
 - выявление требований;
 - анализ требований;
- документирование и организация требований;
- изменение требований;
- планирование и управление требованиями.

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

www.oper.ru



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).*

2. Определение требования

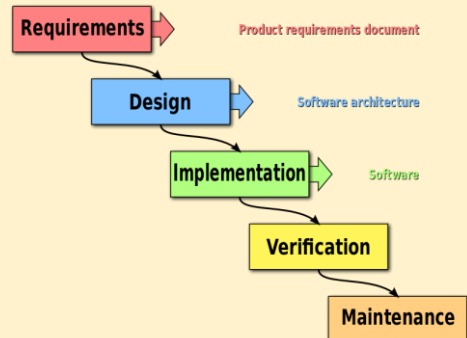
Определение:

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	<ul style="list-style-type: none">✓ <i>Условие или возможность, необходимые пользователю для решения его задач или достижения цели (1)</i>✓ <i>Условие или возможность, которым должна отвечать или которыми должна обладать система или ее компонента, чтобы удовлетворить контракт, стандарт, спецификацию или иной формальный документ (2)</i>✓ <i>Документированное представление условия или возможности, указанное в (1) или (2)</i>
-------------------	--

Управление требованиями

Управление требованиями	<p><i>процесс, включающий:</i></p> <ul style="list-style-type: none">✓ <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.

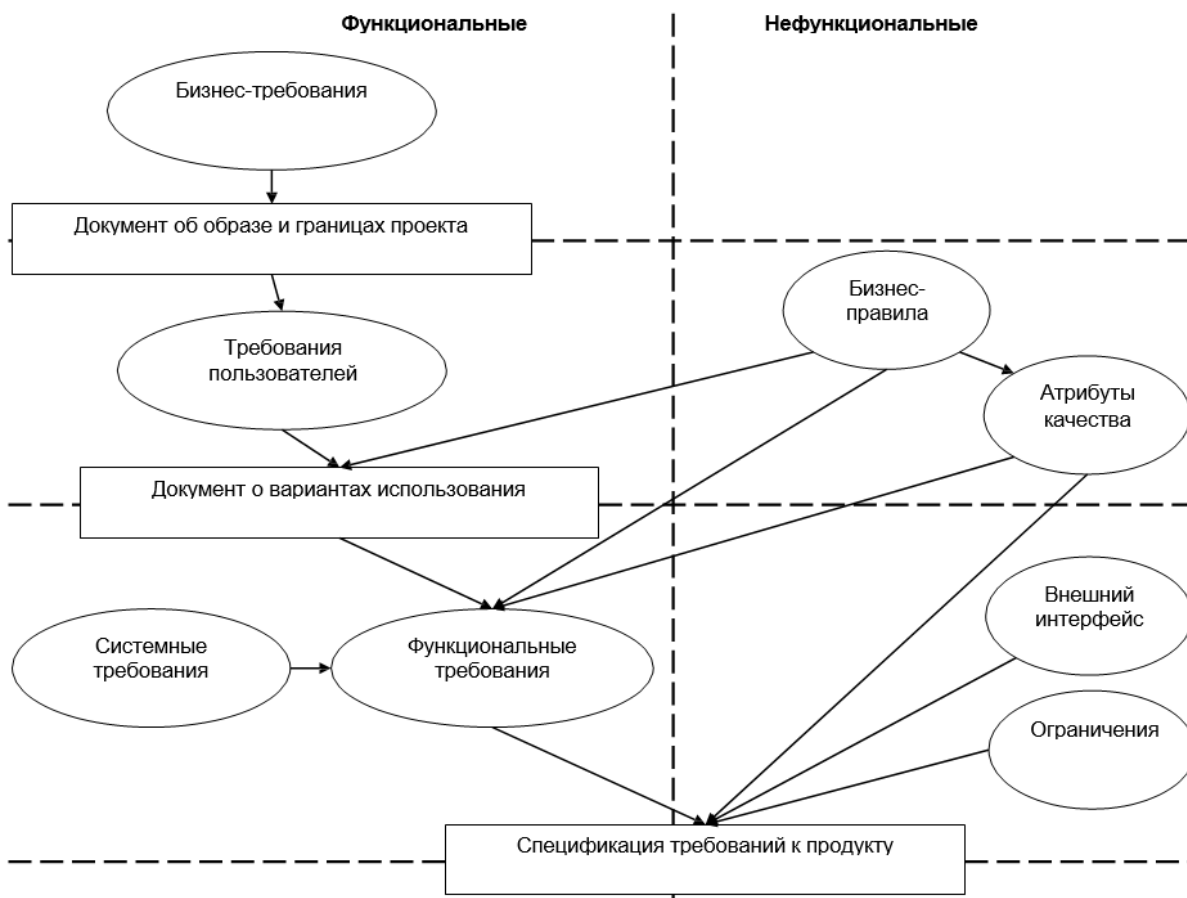
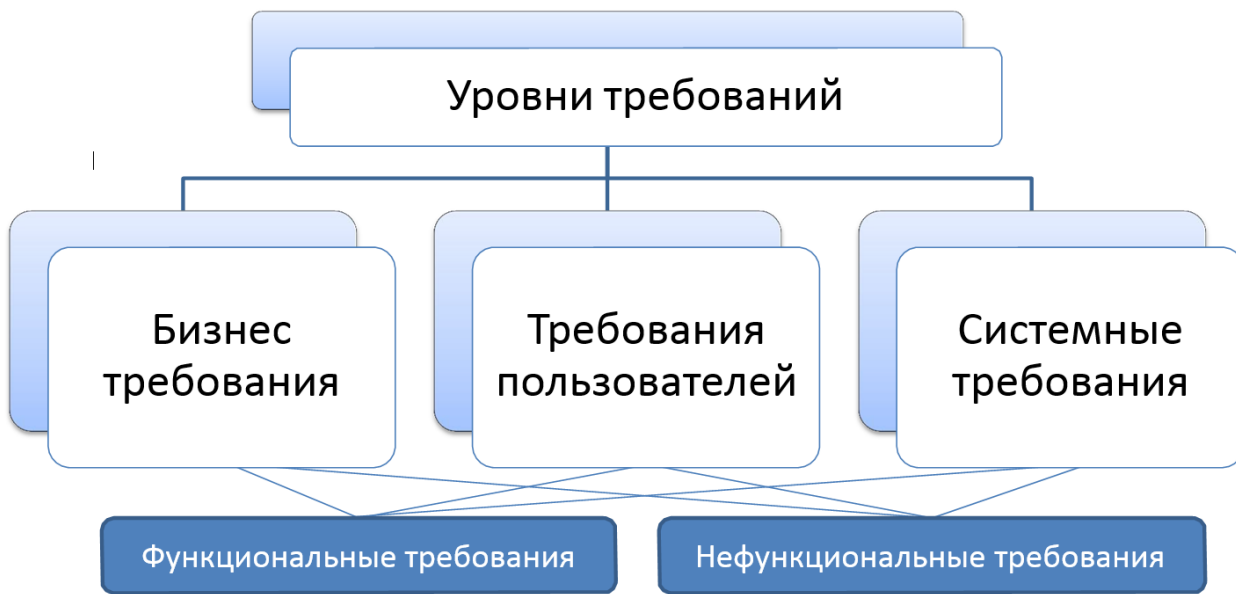
Свойства требований:

- *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.

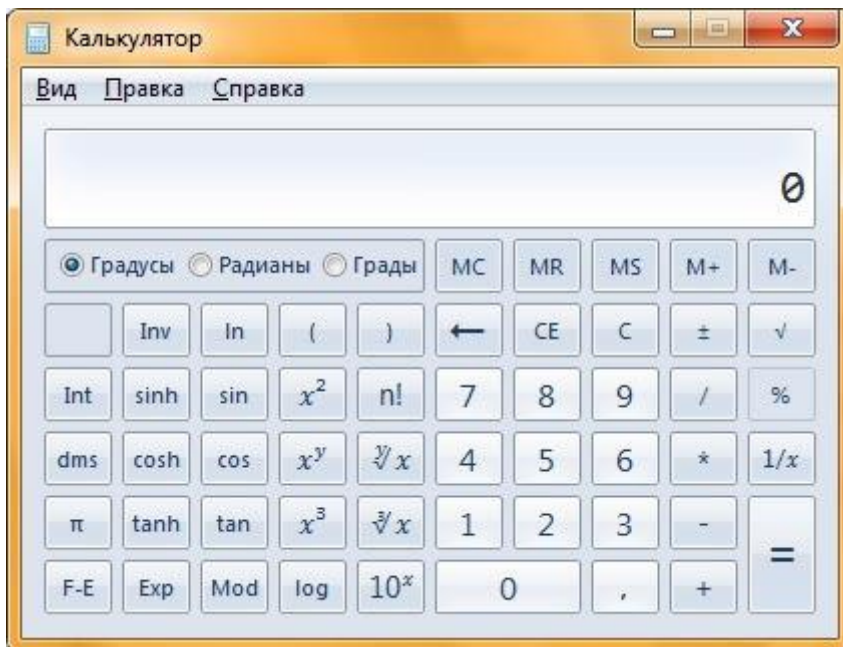
Функциональные пользовательские требования описывают функции в обобщенном виде. Выполняя детализацию этих требований, разработчики формируют более подробное и точное описание сервисов системы – функциональные системные требования.

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
 - predetermined 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>Пользователь:</p> <p>Оптимистичный разработчик: - Это понятная и интуитивная фишка, документация тут не нужна.</p>



Способность простой чашки удовлетворять сформулированную цель зависит от

- свойств, которые зависят от взаимодействия 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:	
Актор	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)

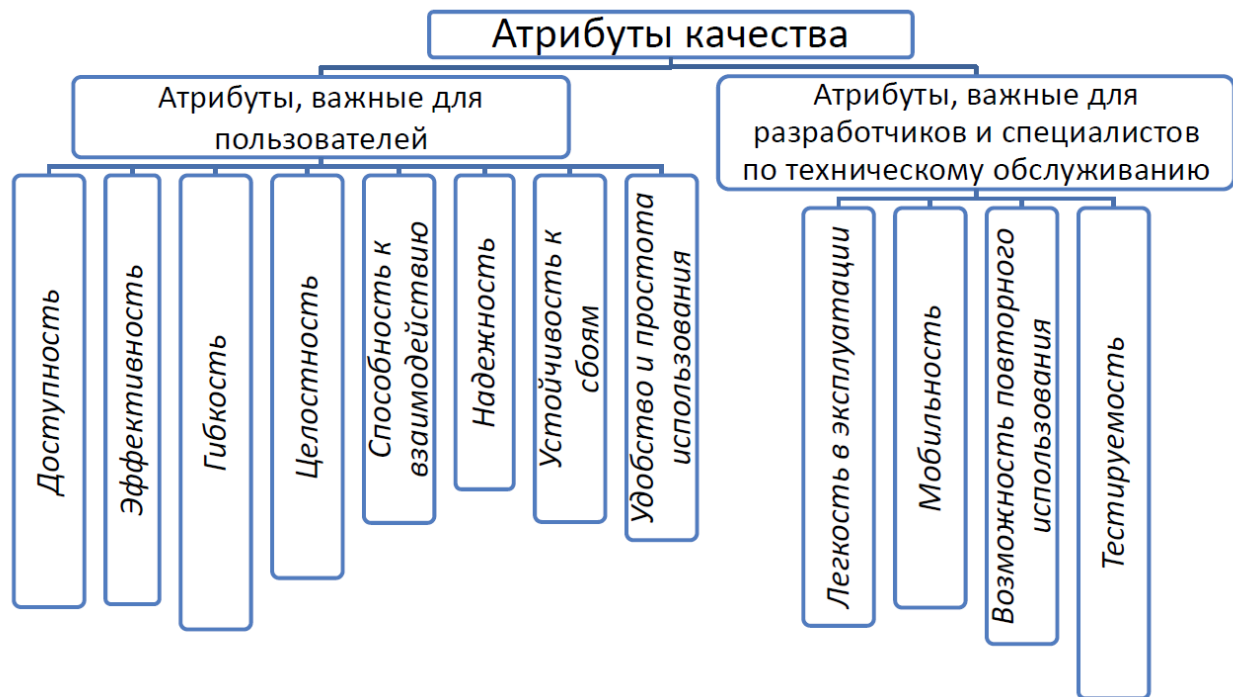
5f. Сопровождение и мобильность

5g. Нерешенные или отложенные вопросы

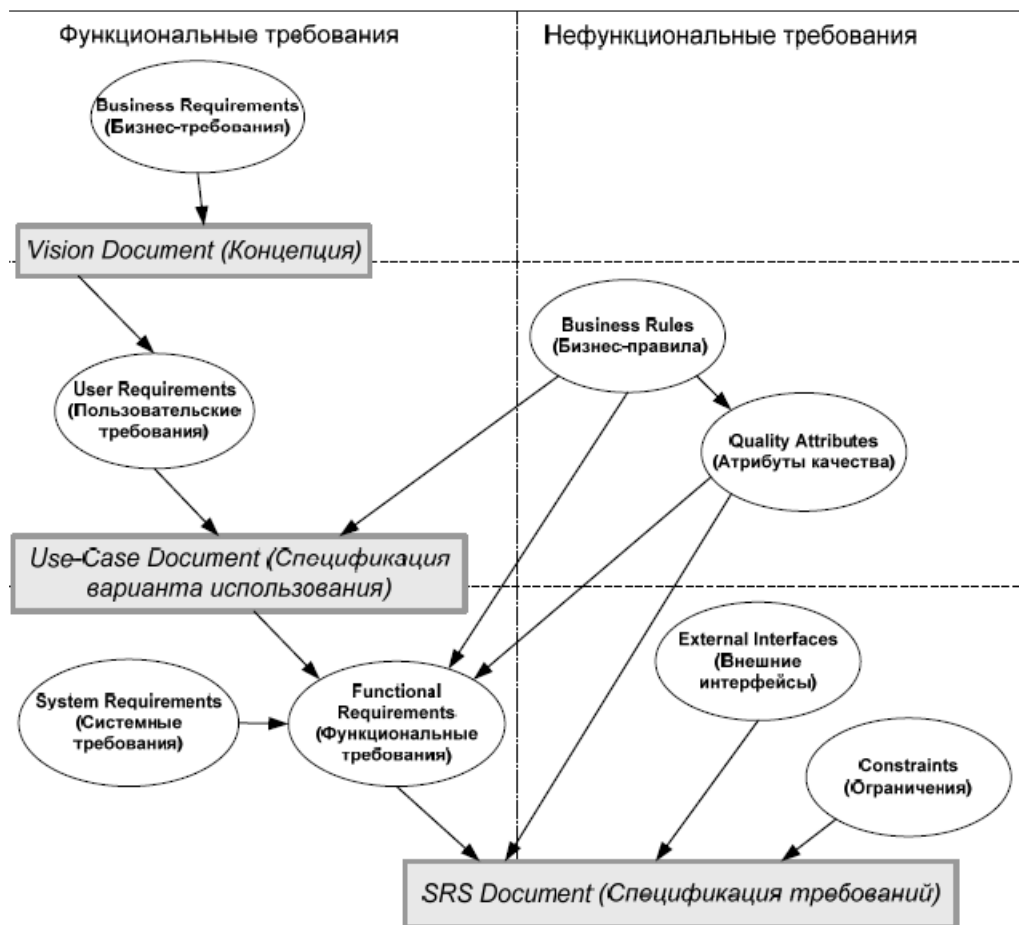
Раздел 6. Людские резервы, правовые, политические, организационные вопросы

- 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

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

Источники сбора требований

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

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

Диаграмма вариантов использования

Диаграмма вариантов использования (англ. use-case diagram) –

диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Диаграмма вариантов использования = Диаграмма прецедентов



- А ты строишь диаграммы при проектировании?

- Да, как видишь

Диаграммы вариантов использования

- ✓ показывают взаимодействия между **вариантами использования** и **действующими лицами**, отражая функциональные требования к системе с точки зрения **пользователя**.
- ✓ являются исходной концептуальной моделью системы в процессе ее проектирования и разработки.

В Microsoft Visio 2016 использовать набор инструментов – «Схема вариантов использования»

Цели построения

Цели построения	<ol style="list-style-type: none"> 1) определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования; 2) сформулировать общие требования к функциональному проектированию системы; 3) разработать исходную концептуальную модель системы для ее последующей реализации; 4) документировать функциональные требования в общем виде для взаимодействия разработчика системы с ее заказчиком и пользователями.
------------------------	---

3. Достоинства модели вариантов использования

Достоинства модели вариантов использования	<ul style="list-style-type: none"> – определяет пользователей и границы системы – определяет системный интерфейс; – удобна для общения пользователей с разработчиками; – используется для написания тестов; – является основой для пользовательской документации; – хорошо вписывается в любые методы проектирования (как объектно-ориентированные, так и структурные).
---	---

Суть диаграммы вариантов использования

Диаграмма вариантов использования

позволяет наглядно представить ожидаемое поведение системы.

Основными понятиями диаграмм вариантов использования являются: действующее лицо, вариант использования и связь.

Основные понятия

- ✓ **действующее лицо;**
- ✓ **вариант использования;**
- ✓ **связь:**
 - ассоциация;
 - отношение расширения;
 - отношение включения;
 - отношение обобщения.

4. Вариант использования

Вариант использования

определяет последовательность действий (сценариев) взаимодействия конкретного актера с проектируемой системой с целью достижения какой-либо цели, значимой для этого актера.

В качестве актера могут выступать не только люди, но и другие системы, устройства и т.п.

Вариант использования = Прецедент = Use Case



Имя **варианта использования** начинается с большой буквы и обозначается оборотом глагола или существительного, обозначающего **действие**

5. Актер

Актер

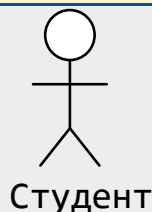
представляет собой внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей и решения частных задач.

Может рассматриваться как некая роль относительно конкретного варианта использования.

Каждый актер – отдельная роль относительно конкретного варианта использования.

Актер = Actor = Действующее лицо = Роль

Стандартное графическое изображение актера:



Актер всегда находится вне системы, его *внутренняя структура* никак не воспринимается.





Примеры актеров: студент, преподаватель, клиент банка, банковский служащий, продавец, сотовый телефон, гость.

Имя ***актера*** основано на использовании имени ***существительного***.

6. Отношения

Один ***актер*** может взаимодействовать с несколькими ***вариантами использования*** и наоборот.

Два варианта использования, определенные для одной и той же сущности, **не могут** взаимодействовать друг с другом, т.к. любой из них самостоятельно описывает законченный вариант использования этой сущности.

<i>Виды отношений</i>	
ассоциативное отношение (отношение ассоциации, association relationship)	1 * 
отношение расширения (extend relationship)	
отношение включения (include relationship)	
отношение обобщения (generalization relationship)	

Отношение ассоциации (association relationship):


отношение между **вариантом использования** и **актером**, отражающее **связь** между ними.

Отношения ассоциации отражают возможность использования актером прецедента.

Отношение устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования.

Обозначение: в виде прямой линии.

Могут быть дополнительные обозначения: кратность связи, направление связи, наименование связи

Пример:	
---------	--

Мощность (кратность, multiplicity) ассоциации определяет количество экземпляров обеих сущностей, которое может участвовать в данной ассоциации. Графически значение мощности отмечается возле линии отношения ассоциации на стороне соответствующей сущности.

В диаграммах вариантов использования определено три типа мощности ассоциации: **один-к-одному**; **один-ко-многим**; **многие-ко-многим**.

Отношение расширения (extend relationship):

определяет взаимосвязь базового варианта использования с некоторым другим более общим вариантом использования, функциональное поведение которого задействуется базовым не всегда, а только при выполнении некоторых дополнительных условий.

Обозначение: ← —<<расширить>>—

Отношение расширения отражает возможное присоединение одного варианта использования к другому в некоторой точке (точке расширения).

Пример:

Пересдать зачет

← —<<расширить>>—

Взять индивидуальную ведомость

Стрелка указывает на базовый вариант использования!

Отношение включения (include relationship):

указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования.

Обозначение: —<<включить>>→

Пример:

Защитить все лабораторные работы

Успешно пройти промежуточные аттестации

Выполнить проект

Сдать зачет по ОПИ (ТРПО)

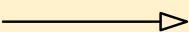
<<включить>>

<<включить>>

<<включить>>

Отношение обобщения (generalization relationship):

Отношение обобщения служит для указания того, что некоторый вариант использования может быть обобщен до другого варианта использования.

Обозначение: 

Пример:



A

B

B – предок по отношению к A

A – потомок B

Потомок наследует все свойства и поведение своего родителя, может быть дополнен новыми свойствами и особенностями поведения.

Стрелка указывает на родительский вариант использования.

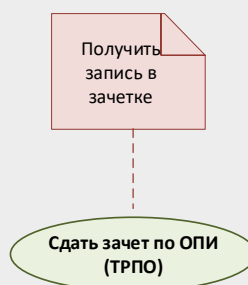
7. Примечание – элемент диаграммы вариантов использования

Примечание:

элемент диаграммы вариантов использования.

Соединяется с элементом *штриховой линией*

Пример:



Примечание (Note) в языке UML предназначено для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта.

Примечание может относиться к любому элементу диаграммы.

8. Пример

Рассмотрим выполнение 12-ой лабораторной работы студентами.

Моделируемая система – «Ознакомление с технологиями разработки ПО».

Цель – см. цель лабораторной работы 12.

Актер – «Студент» (взаимодействует с системой).

Функциональность:

- «Студент» должен ответить на контрольные вопросы по изученному материалу;
- «Студент» определяет цели и назначение проекта;
- «Студент» выбирает название проекта;
- «Студент» выбирает модули, входящие в состав проекта в соответствии с требованиями лабораторной работы 12;
- «Студент» выполняет индивидуальное задание;
- «Студент» выполняет проектирование и разработку проекта с использованием гибких технологий разработки ПО;
- «Студент» должен выполнить тестирование;
- «Студент» должен выполнить рефакторинг;
- «Студент» должен выполнить документирование;
- «Студент» представляет проект.

При взаимодействии с актером «Студент» система должна позволять выполнять набор функциональных требований, при этом важна реакция системы.

Главная последовательность взаимодействия «Студента» с «Системой»:

Функциональность	Реакция системы
1. «Студент» должен ответить на контрольные вопросы по изученному материалу	2. «Система» обеспечивает «Студенту» возможность изучения теории по теме лабораторной работы
3. «Студент» определяет цели и назначение проекта	4. «Система» должна проконтролировать определение «Студентом» цели и назначения проекта

5. «Студент» называет проект	
6. «Студент» выбирает модули, входящие в состав проекта в соответствии с требованиями лабораторной работы 12	
7. «Студент» выполняет индивидуальное задание	8. «Система» контролирует выполнение «Студентом» индивидуального задания
9. «Студент» выполняет проектирование и разработку проекта с использованием гибких технологий разработки ПО	
10. «Студент» должен выполнить тестирование	
11. «Студент» должен выполнить рефакторинг	
12. «Студент» должен выполнить документирование	
13. «Студент» представляет проект	14. «Система» оценивает презентацию проекта скрам-команды и каждого члена команды - сдача лабораторной работы преподавателю

Альтернативная последовательность

7а: «Студент» **не выполняет** индивидуальное задание.

7в: Реакция системы: «Студент» **не** допускается до экзамена.

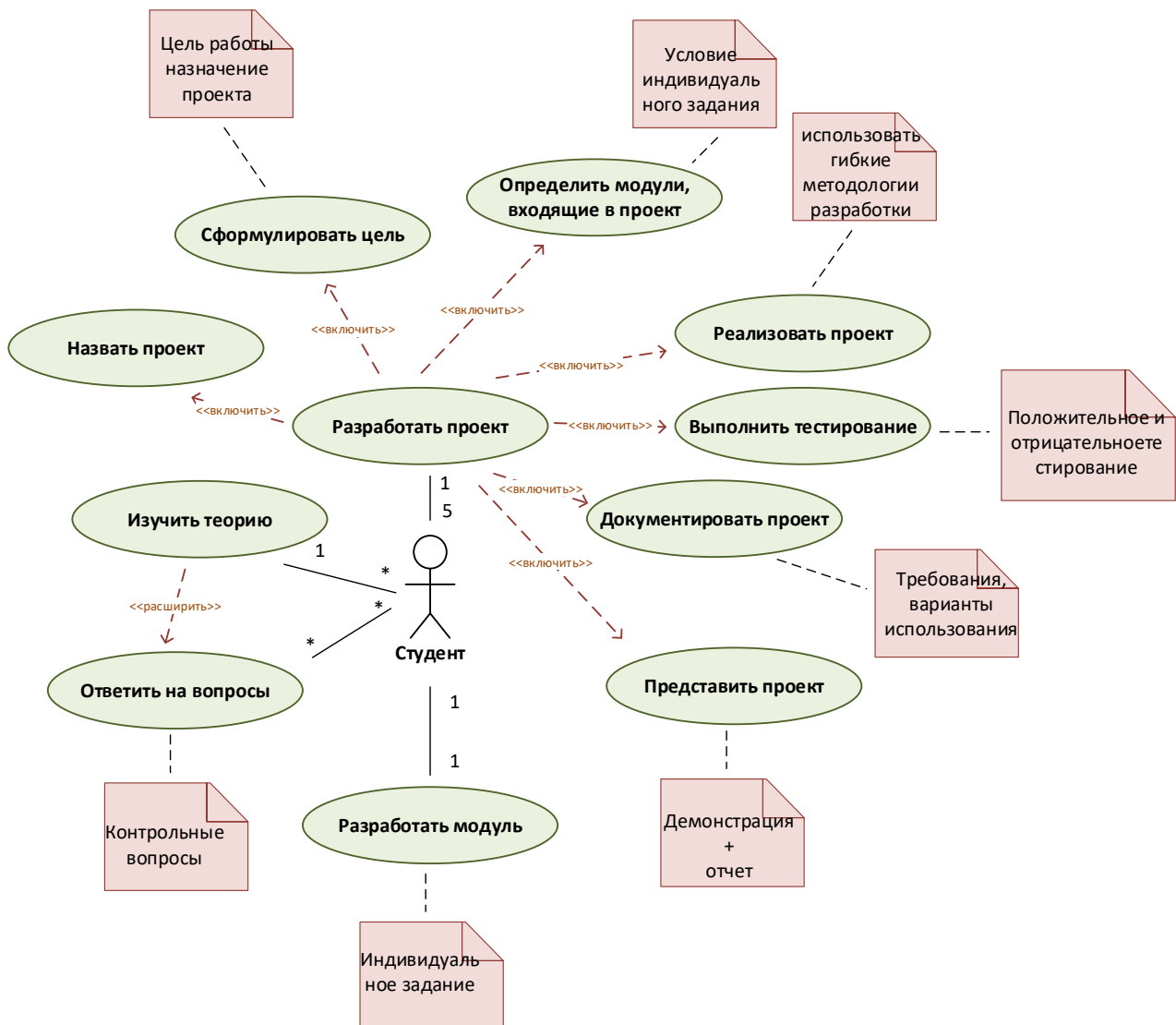
Некоторые пояснения

Помимо актеров и вариантов использования диаграмма вариантов использования содержит примечания – элементы, служащие для размещения на диаграмме поясняющей текстовой информации. Примечание может относиться к любому элементу диаграммы и соединяется с данным элементом штриховой линией.

На диаграмме показано, что у актера «Студент» и соответствующими вариантами использования существуют отношения ассоциации.

Базовый вариант использования «Ответить на вопросы» связан отношением расширения с вариантом использования «Изучить теорию».

Пример диаграммы вариантов использования системы для актера «Студент»:



Можно также расширить функциональное назначение моделируемой системы: ввести актера «Преподаватель» и построить диаграммы вариантов использования системы «Ознакомление с технологиями разработки ПО» для этой роли.

При взаимодействии с актером «Преподаватель» система должна обеспечить возможность выполнения следующих основных функциональных требований:

- «Преподаватель» разрабатывает техническое задание;
- «Преподаватель» должен организовать взаимодействие со скрам-мастером каждой команды;
- «Преподаватель» отвечает на возникающие вопросы;

- «Преподаватель» регистрирует скрам-мастера и состав команды, исходные данные проекта;
- «Преподаватель» осуществляет контроль хода выполнения индивидуального задания;
- «Преподаватель» осуществляет контроль хода выполнения проектирования и разработки проекта;
- «Преподаватель» проверить результаты выполнения индивидуального задания;
- «Преподаватель» принимает лабораторную работу.