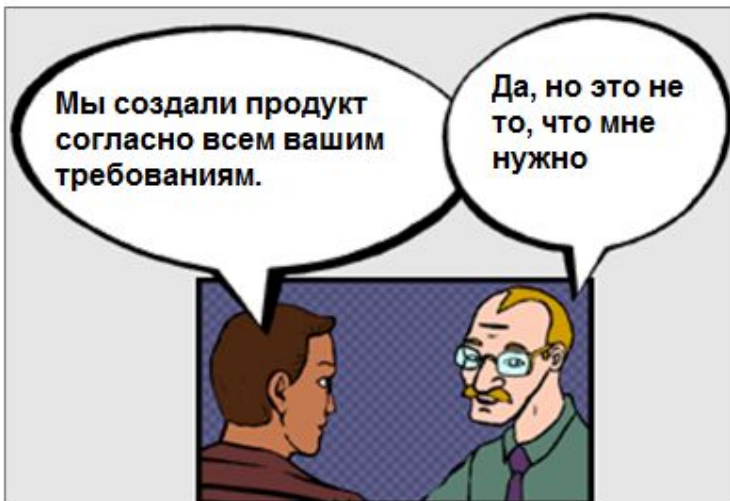


Основы проектирования ПО

Куликова Елена Васильевна

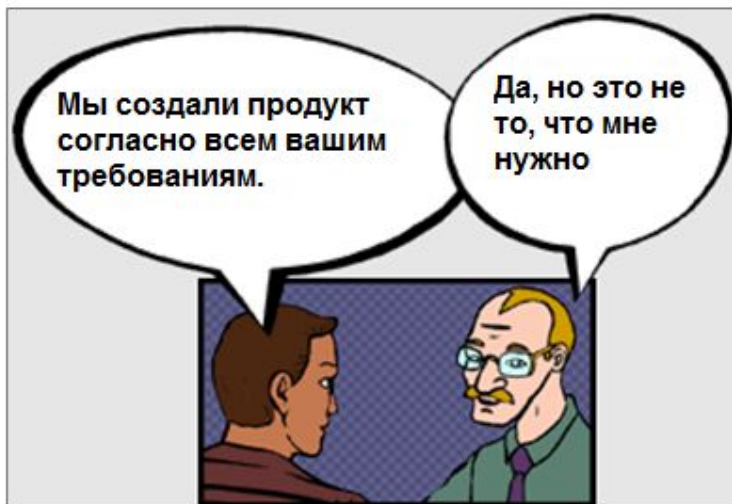
Тема 3. УПРАВЛЕНИЕ ТРЕБОВАНИЯМИ

- *Определение ключевых понятий управления требованиями.*
- *Определение факторов, способствующих и препятствующих успеху проекта.*
 - Описание роли управления требованиями в успешности проекта.
- *Определение характеристик требований.*
 - Проверяемость, трассируемость, однозначность.
- *Описание рабочего процесса, ролей и артефактов управления требованиями в методологии IBM RUP.*



Зачем управлять требованиями?

- ❑ Ошибки и разночтения, которые возникают при выявлении требований к системе, оказываются одними из самых дорогих.
- ❑ Существуют **трудности в понимании между заказчиком и разработчиками**, а еще – в **изменчивости ПО** (требования имеют тенденцию меняться в ходе разработки).



Введение в управление требованиями



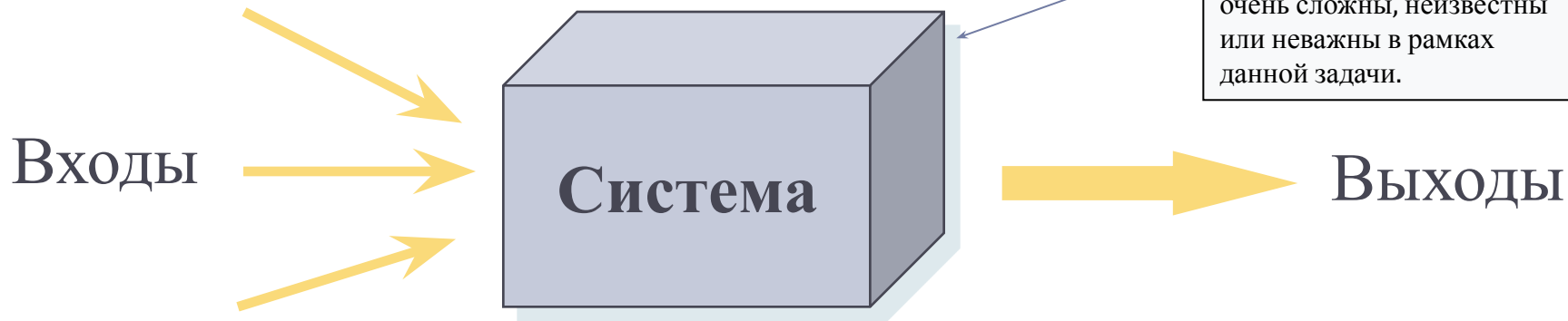
Определения

- **Требование** – это условие или характеристика, которым должна отвечать система.
- **Требования** – это то исходное понимание задачи разработчиками, которое является основой всей разработки.
- **Управление требованиями** – это систематический подход к:
 - Выявлению, организации и документированию требований.
 - Выработка и поддержка согласия между заказчиками/пользователями и проектной командой относительно меняющихся требований.



Что указывают требования к ПО?

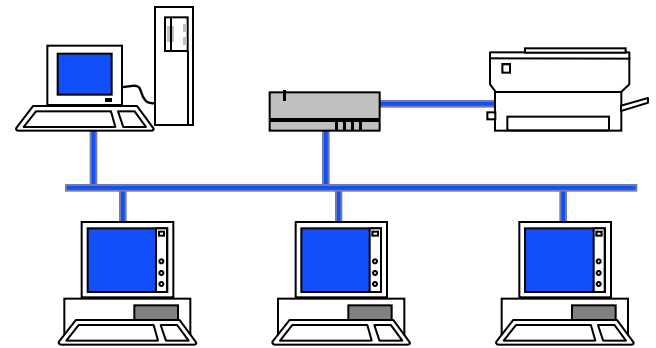
Черный ящик — термин, используемый для обозначения системы, внутреннее устройство и механизм работы которой очень сложны, неизвестны или неважны в рамках данной задачи.



Функция

Нефункциональные требования
(производительность и т.п.)

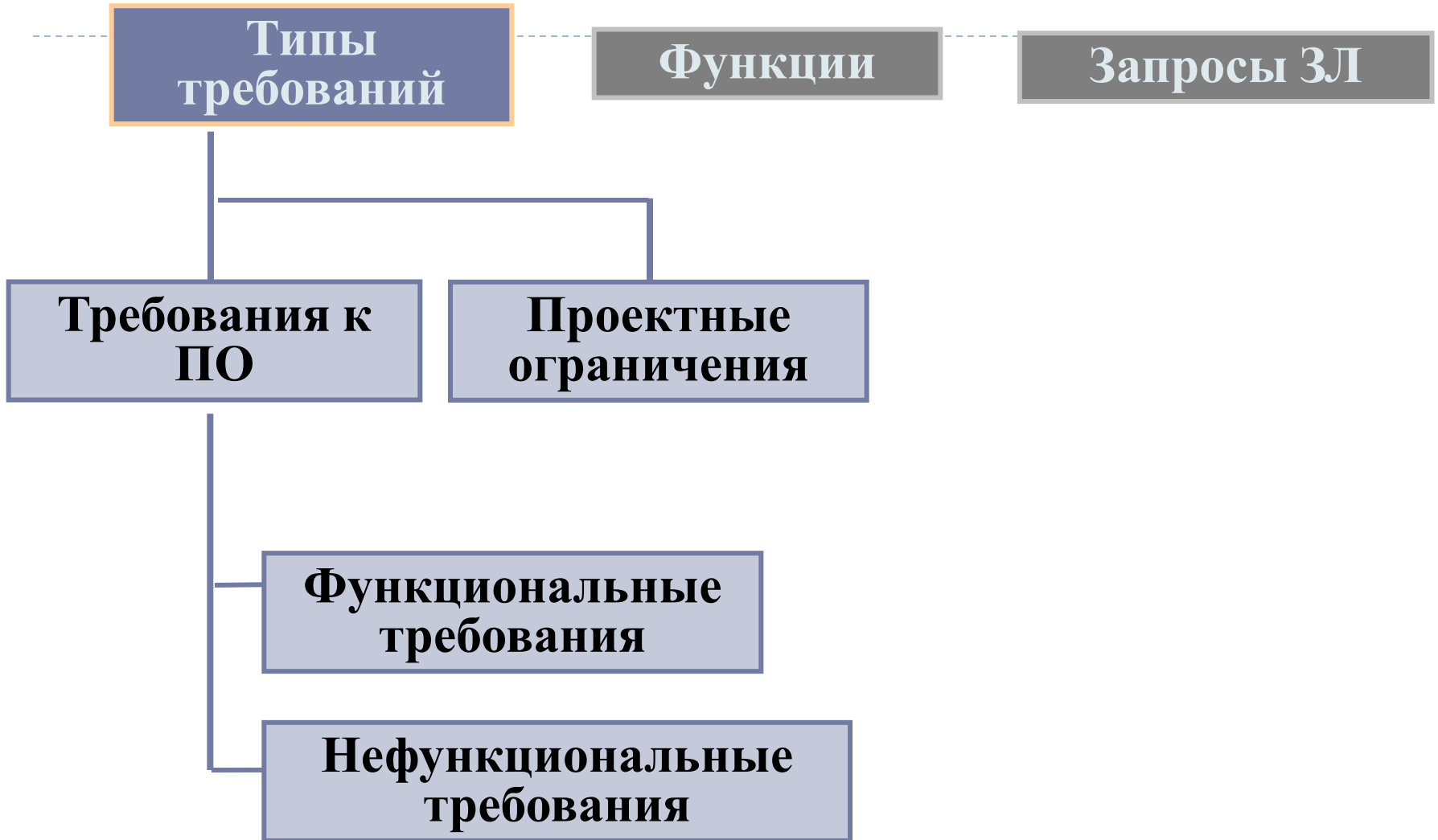
Проектные ограничения



(среда и т.п.)



Категоризация определений



Определения

Запросы заинтересованных лиц (Stakeholder Request)

Пожелания ЗЛ относительно решения.

Функция (Feature)

Внешне наблюдаемая служба, через которую система непосредственно выполняет один или несколько запросов ЗЛ.

Ограничения

Факторы, затрагивающие проектирование системы, или процессы, используемые при ее создании



Виды требований

• Функциональное требование

- Требование, описывающее **взаимодействие** поставляемого решения с **внешним миром** с точки зрения «черного ящика».
- Функциональные требования являются **детальным описанием поведения и сервисов системы, ее функционала**. **Они определяют то, что система должна уметь делать.**

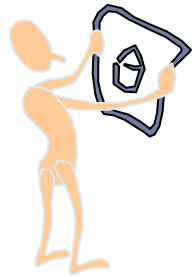
• Нефункциональное требование

- Требование, описывающее **качественные атрибуты** поставляемого решения с точки зрения «черного ящика».
- **Не являются описанием функций системы**. Описывают такие характеристики системы, как: **надежность**, особенности **поставки** (наличие инсталлятора, документации), определенный **уровень качества**, требования к **переносимости, соответствию стандартам** и т.д.
Требования этого вида часто относятся ко всей системе в целом.

Черный ящик — термин, используемый для обозначения системы, внутреннее устройство и механизм работы которой очень сложны, неизвестны или неважны в рамках данной задачи.



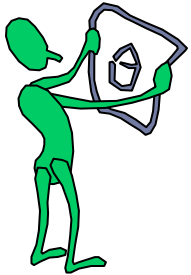
Требования существуют на многих уровнях



Что
Как

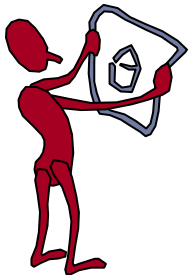
Потребности ЗЛ

Функции системы или продукта



Что
Как

Требования к ПО



Что
Как

Проектные спецификации

Методики тестирования

Планы по документации



Управлять требованиями трудно, так как...

Требования

- Не всегда очевидны.
- Исходят из многих источников.
- Не всегда могут быть четко сформулированы.
- Связаны друг с другом и прочими артефактами процесса разработки ПО.
- Имеют уникальные характеристики или значения характеристик.
- Изменчивы.
- С трудом поддаются контролю в больших объемах.



Причины изменчивости ПО:

- Меняется **ситуация на рынке**, для которого предназначалась система или требования к системе изменяются из-за быстро сменяющихся перспектив продажи еще неготовой системы.
- **В ходе разработки возникают проблемы и трудности**, в силу которых **итоговая функциональность меняется** (видоизменяется, урезается).
- Заказчик может менять свое собственное видение системы:
 - выясняется, что он что-то упустил с самого начала...
 - выясняется, что разработчики его не так поняли...
 -

Свойства требований

1. **Ясность, недвусмысленность** — однозначность понимания требований заказчиком и разработчиками.
2. **Полнота**
 - отражает все существенные требования
3. **Непротиворечивость**
 - требования не противоречат друг другу
4. **Необходимый уровень детализации.** Требования должны обладать ясно осознаваемым уровнем детализации, стилем описания, способом формализации:
 - либо это описание свойств предметной области, для которой предназначается ПО,
 - либо это техническое задание, которое прилагается к контракту,
 - либо это проектная спецификация, которая должна быть уточнена в дальнейшем, при детальном проектировании.
 - либо
5.

Свойства требований

5. **Прослеживаемость (трассируемость)** — происхождение каждого требования легко отследить. Прослеживаемость функциональных требований достигается путем их дробления на отдельные, элементарные требования, присвоение им идентификаторов и создание трассировочной модели, которая в идеале должна протягиваться до программного кода.
- Например, желательно знать, где **нужно изменить код, если какое-то требование изменилось**. На практике полная формальная прослеживаемость труднодостижима, т.к. логика и структура реализации системы могут сильно не совпадать с таковыми для модели требований. В итоге одно требование оказывается сильно «размазано» по коду, а тот или иной участок кода может влиять на много требований.

Свойства требований

6. **Тестируемость и проверяемость** — необходимо, чтобы существовали способы оттестировать и проверить данное требование.
 - Необходимы процедуры проверки –выполнение тестов, проведение инспекций, проведение формальной верификации части требований и др. Необходимы критерии полноты проверок, чтобы выполняющие их и руководители проекта четко осознавали, что именно проверено, а что еще нет.
7. **Модифицируемость.** Определяет процедуры внесения изменений в требования.
 - Изменения не повлияют на структуру и стиль спецификации.
8. **Корректность**
 - Спецификация является корректной, если и только если каждое требование, изложенное в ней, является требованием, которому должно удовлетворять программное обеспечение.

Свойства требований

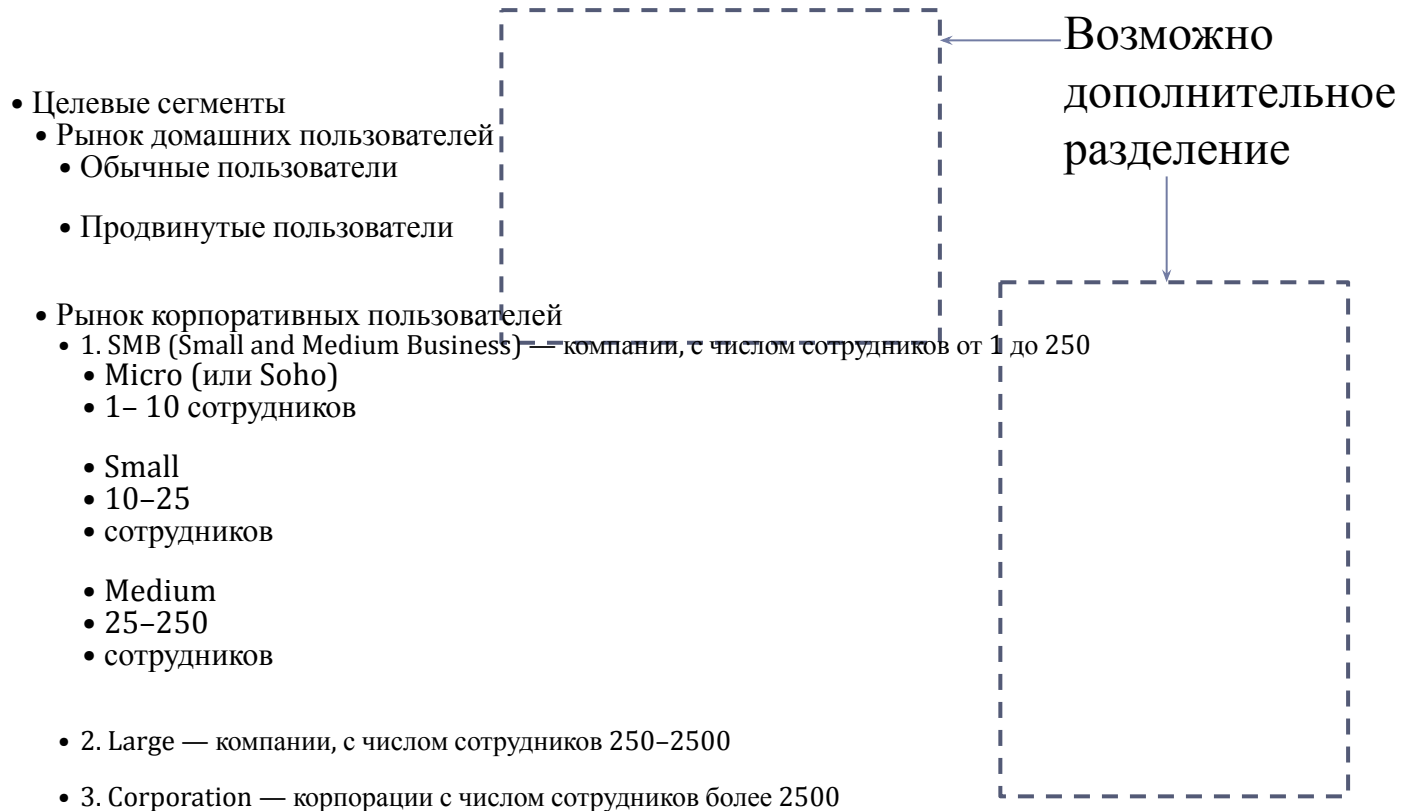
9. Проверимость . Требование является проверяемым, если:

- Существует некий конечный и экономически эффективный процесс, с помощью которого машина или человек может проверить соответствие продукта требованиям.
 - Система поддерживает до 1000 пользователей одновременно.
 - Система должна отвечать на произвольный запрос за 500 миллисекунд.
 - Цветовая гамма должна быть приятного зеленого оттенка.
 - Система должна быть доступна 24 часа в сутки, семь дней в неделю.
 - Система должна экспортировать данные в формате значений, разделенных запятыми, согласно спецификации IEEE.

7.

Пример. Нефункциональные требования ПО (с учетом целевого сегмента рынка)

□ Типовое сегментирование рынка:



Пример. Нефункциональные требования ПО (с учетом целевого сегмента рынка)

	Домашний пользователь	Малый и средний бизнес	Крупная компания
Администрирование	Не требуется.	Интуитивно понятное, не требующее спец. обучения. Интеграция с AD на уровне пользователей.	Удаленное или централизованное (в зависимости от пр. обл.). Интеграция с AD на уровне пользователей и прав. Централизованная или, как минимум, удаленная установка.
Интерфейс	Графический. Красивый, удобный и быстрый.	Графический.	<ul style="list-style-type: none"> • Графический • Командный • SDK
Требования к безопасности.	—	Защита паролем.	Защита сертификатом.
Отказоустойчивость.	В зависимости от прикладной области	В зависимости от прикладной области	В зависимости от прикладной области
Масштабируемость*	—	Поддержка до 250 компьютеров	Поддержка до 2000 компьютеров.
Лицензирование	Лицензии на: <ul style="list-style-type: none"> • одну копию • пользователя 	Лицензии на: <ul style="list-style-type: none"> • одну копию пользователя • группу 	Использование специальных лицензий.
Поддержка	<ul style="list-style-type: none"> • Email • форум 	<ul style="list-style-type: none"> • Email • Форум • телефон 	<ul style="list-style-type: none"> • Email • Форум • Телефон • Выделенный инженер для разворачивания и поддержки продукта.
Способ продажи	<ul style="list-style-type: none"> • Интернет • магазин 	<ul style="list-style-type: none"> • Интернет • магазин 	Дилер

SDK (*software development kit*) — набор средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ

* Требуется тестирование на указанном количестве продуктов.

На примере ИС «Турагенство» выделить требования, разбив их на две группы (см. слайд 9). Учитывать свойства: слайды 13-16

Функциональные требования

Группы требований	Функциональные требования
Формирование отчетов	Формирование отчета «Количество проданных туров за заданный период»
	Формирование отчета «Туры в определенную страну»

Эффективное управление требованиями

- Четкое изложение требований обеспечивается:
 - Высоким качеством требований
 - Применимыми атрибутами для каждого типа требования
 - Трассируемостью к другим требованиям и проектным артефактам

**ЦЕЛЬ – поставить в срок и в рамках бюджета
качественный продукт,
который будет отвечать
реальным потребностям заказчика**



Что такое «качественный продукт»?

❑ Устаревший подход к качеству

- ❑ Соответствие перечню требований.
- ❑ Прохождение системных тестов.
- ❑ Разработка в соответствии с процессом.

❑ Ориентированность на процесс

❑ Современный подход к качеству

- ❑ Понимание потребностей заинтересованных лиц.
- ❑ Непрерывная оценка всех рабочих продуктов с целью удостовериться, что они отвечают этим потребностям.

❑ Ориентированность на результат



Основные атрибуты качества

Применимость

Надежность

Производительность

Эксплуатационная
пригодность

*Атрибуты качества хорошо раскрыты в модели **FURPS** (расширение модели: **FURPS +**)*

(со ссылкой на стандарт [IEEE Std 610.12.1990](#))



FURPS и FURPS+

- ▣ ***FURPS*** (Functionality Usability Reliability Performance Supportability): функциональность, удобство использования, надежность, производительность, удобство сопровождения) — классификация требований к программным системам, разработанная в Hewlett-Packard.
- ▣ Согласно классификации, все требования подразделяются на 5 категорий, непосредственно следующих из составляющих наименования классификации.

Атрибуты качества. Модель FURPS

Компоненты FURPS	Перевод	Описание
Functionality	Функциональность	Функциональные возможности, безопасность, общие вопросы
Usability	Применимость	Эргономические характеристики, согласованность, документация
Reliability	Надежность	Частота/серьезность отказов, восстанавливаемость, предсказуемость, точность
Performance	Производительность	Скорость, использование ресурсов, пропускная способность, время отклика
Supportability	Эксплуатационная пригодность	Тестируемость Расширяемость Адаптируемость Сопровождаемость Совместимость Конфигурируемость Обслуживаемость Устанавливаемость Локализуемость Устойчивость



Задание. Дать определения к характеристикам Supportability

- Тестируемость
- Расширяемость
- Адаптируемость
- Сопровождаемость
- Совместимость
- Конфигурируемость
- Обслуживаемость
- Устанавливаемость
- Локализуемость
- Устойчивость



Расширение модели FURPS+

- В настоящее время FURPS используется, как составная часть более общей классификации FURPS+.
- Символ "+" расширяет FURPS-модель, добавляя к ней новые требования



Модель FURPS+

- ▣ **FURPS+** (Functionality Usability Reliability Performance Supportability +: функциональность, удобство использования, надежность, производительность, удобство сопровождения, дополнительные требования) — расширенная версия классификации требований FURPS.
- ▣ Дополнительно включает в себя ограничения, разделенные на следующие группы требований:
 - ▣ ограничения проектирования (**design**);
 - ▣ ограничения разработки (**implementation**);
 - ▣ ограничения на интерфейсы (**interface**);
 - ▣ физические ограничения (**physical**).



Задание.

Примеры состава требований:

- ограничения проектирования (**design**);
- ограничения разработки (**implementation**);
- ограничения на интерфейсы (**interface**);
- физические ограничения (**physical**).



Цикл работы с требованиями

- В своде знаний по программной инженерии **SWEBOK** определяется следующие **виды деятельности при работе с требованиями:**

1. Выделение (извлечение) требований, нацеленное на выявление всех возможных источников требований и ограничений на работу системы и извлечение требований из этих источников.

2. Анализ требований, целью которого обнаружение и устранение противоречий и неоднозначностей в требованиях, их уточнение и систематизация.

3. Описание требований, деятельность требования должны быть оформлены в виде структурированного набора документов и моделей, который может систематически анализироваться, оцениваться с разных позиций и в итоге должен быть утвержден как официальная формулировка требований к

4. Валидация требований, которая решает задачу оценки понятности сформулированных требований и их характеристик, необходимых, чтобы разрабатывать ПО на их основе, в первую очередь, непротиворечивости и полноты, а также соответствия корпоративным стандартам на техническую документацию.



Варианты формализации требований

- **1. Неформальная постановка требований** (переписка по электронной почте, устная беседа и т.п.). Хорошо работает в небольших проектах, при вовлеченности заказчика в разработку (например, команда выполняет субподряд). Хорошо также при таком стиле, когда есть взаимопонимание между заказчиком и командой, то есть лишние формальности не требуются. Важно уметь вести деловую переписку, подводить итоги, хранить важные письма и пользоваться ими при разногласиях. Важно также вовремя понять, когда такой способ перестает работать и необходимы более формальные подходы.
- **2. Требования в виде документа** – описание предметной области и ее свойств, техническое задание как приложение к контракту, функциональная спецификация для разработчиков и т.д.
- **3. Требования в виде графа с зависимостями** в одном из средств поддержки требований (например, UML-диаграммы).
- **4. Формальная модель требований** для верификации, модельно-ориентированного тестирования и т.д.

Традиционное описание требований



Пример. Описание функциональных требований

1. Система АТМ должна проверять действительность вставленной в банкомат карточки.
2. Система АТМ должна проверять достоверность PIN-кода, введенного пользователем.
3. Система АТМ должна выдавать по одной АТМ-карточке не более \$250 в сутки.
 - *Программа «АТМ» - подключение электронных устройств Банка, мобильных приложений и модуля Интернет-банка к приему платежей в пользу различных поставщиков услуг (сотовые операторы, банки, коммунальные платежи, связь, ТВ, интернет и т.д.)*

Пример. Описание нефункциональных требований

1. Система АТМ должна быть написана на С++.
2. Система АТМ должна обмениваться информацией с банком, используя 256-разрядную кодировку.
3. Система АТМ должна проверять действительность карточки АТМ в течение не более трех секунд.
4. Система АТМ должна проверять достоверность PIN-кода в течение не более трех секунд.

Задание. Привести свой пример (для АИС, АРМ, сайта и т.д.). Представить в виде описания.

- Не менее 3-х **функциональных** требований
- Не менее 3-х **нефункциональных** требований



Пример. Общее представление функционального назначения системы (функциональные требования)

□ Диаграмма вариантов использования (диаграмма прецедентов) – это наиболее общее представление функционального назначения системы.

□ Задачи диаграммы прецедентов:

Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы

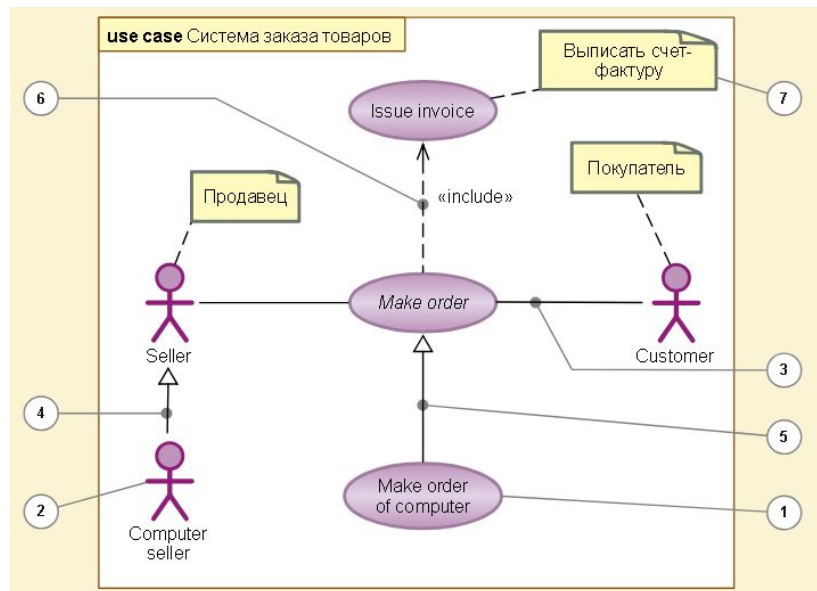
Сформулировать общие требования к функциональному поведению проектируемой системы

Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей

Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями

Пример. Общее представление функционального назначения системы (функциональные требования)

- На диаграмме использования применяются два типа основных элементов:
 - варианты использования - прецеденты (1)
 - действующие лица - актеры (2), между которыми устанавливаются следующие основные типы отношений:
- Между отдельными элементами может создаваться семантическая связь **отношение (relationship)**.
- На диаграмме использования, могут присутствовать комментарии (7).



Отношения в примере:

- ассоциация между действующим лицом и вариантом использования (3);
- обобщение между действующими лицами (4);
- обобщение между вариантами использования (5);
- зависимости (различных типов) между вариантами использования 6.

Стандартные виды отношений:

Отношение *ассоциации* (association relationship)

Отношение *включения* (include relationship)

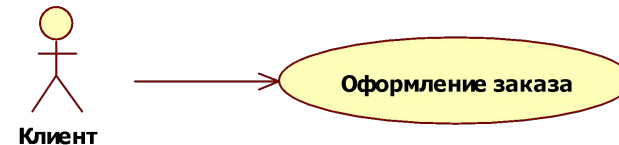
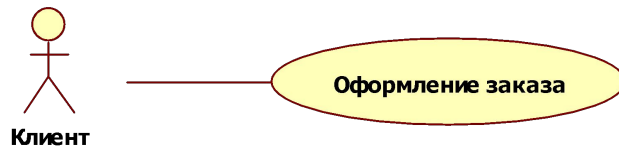
Отношение *расширения* (extend relationship)

Отношение *обобщения* (наследования, генерализации)
(generalization relationship)

Отношение ассоциации (association relationship)

Применительно к диаграммам *вариантов использования* **ассоциация** служит для обозначения **специфической роли актера** при его взаимодействии с **отдельным вариантом использования**.

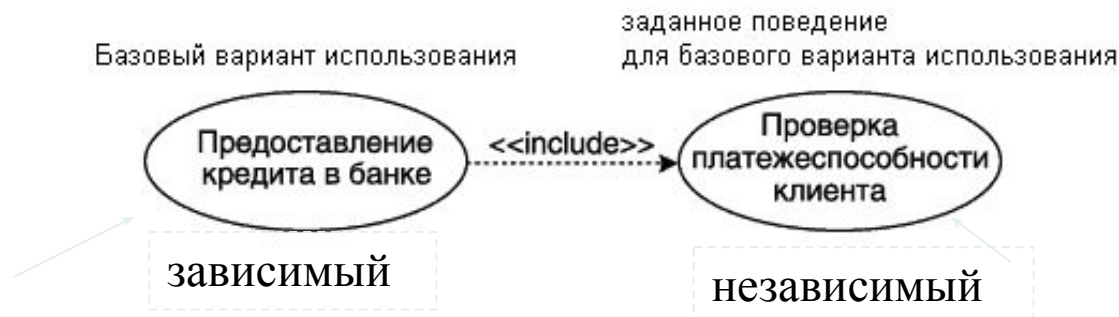
Пример графического представления отношения ассоциации между актером и вариантом использования:



Отношение включения (include relationship)

Включение (include) в языке UML — это разновидность отношения зависимости **только между двумя вариантами использования**: между базовым вариантом использования и его специальным случаем.

- Указывает на то, что заданное поведение для одного *варианта использования* включается в качестве составного фрагмента в последовательность поведения другого *варианта использования*.
- При этом изменение одного элемента (независимого) приводит к изменению другого элемента (зависимого).



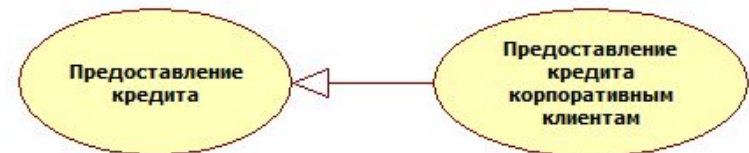
Отношение расширения (extend relationship)

- Отношение *расширения (extend)* определяет взаимосвязь **базового варианта использования** с другим **вариантом использования**, функциональное поведение которого **задействуется базовым не всегда, а только при выполнении дополнительных условий**.

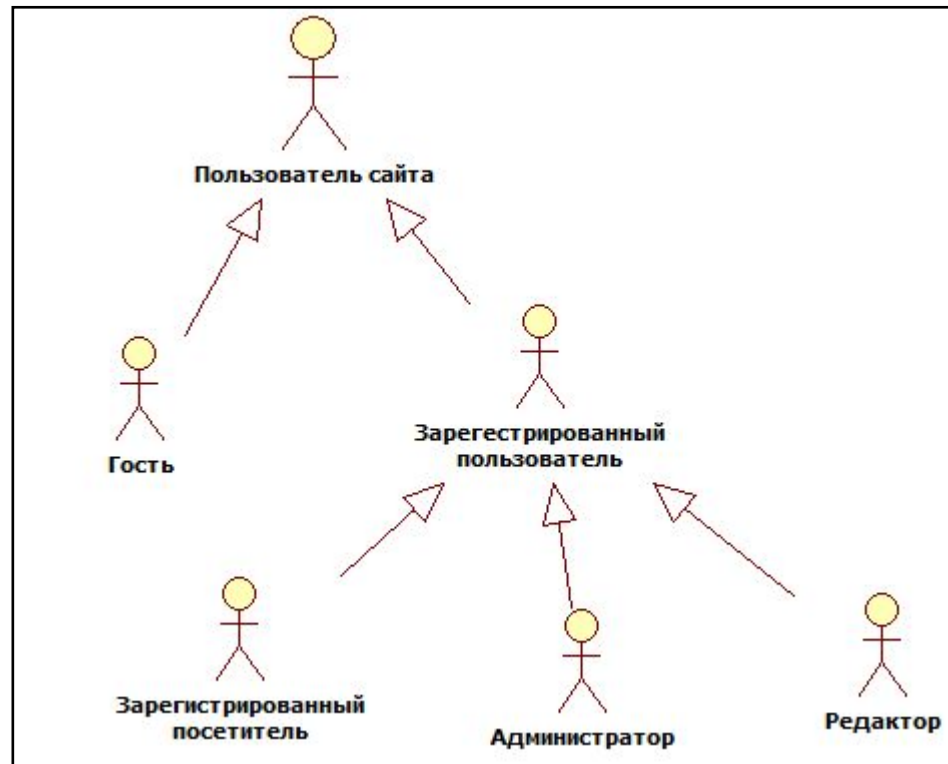


Отношение обобщения (наследования, генерализации) (generalization relationship)

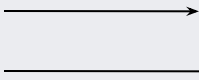


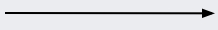
- Два и более *актера* могут иметь общие свойства, т. е. взаимодействовать с одним и тем же множеством вариантов использования одинаковым образом. Такая общность свойств и поведения представляется в виде отношения обобщения с другим, возможно, абстрактным актером, который моделирует соответствующую общность ролей.
- Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми особенностями поведения родительских вариантов.



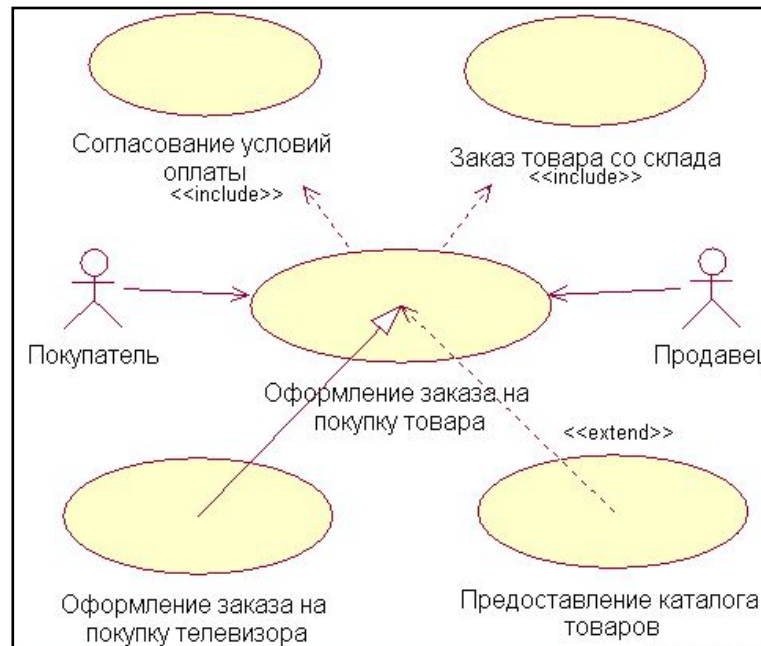
Отношение обобщения между актерами



Сводная таблица «Отношения»

	association	include	extend	generalization
Actor - Actor	нет	нет	нет	да
Actor - UseCase	да	нет	нет	нет
UseCase- UseCase	нет	да	да	да
Обозначение (вид стрелки)				
Стереотип		<<include>>	<<extend>>	

Пример. Диаграмма прецедентов для системы продажи товаров в супермаркете



Хорошая книга!

- https://pmi.ru/profes/Software_Requirements_Khimonin.pdf

*Сбор и анализ требований
к программному продукту*

Версия 1.03

Автор: Химонин Юрий
program manager компании Acronis





1. Выделение требований

- 1) Определение основных профилей пользователей
- 2) Формирование инициативной группы
- 3) Сбор пользовательских историй



1. Выделение требований

□ 1) Определение основных профилей пользователей

- Для **домашних продуктов** сложно определить «среднего пользователя», чтобы на основе его потребностей проектировать продукт.
- Для **корпоративных продуктов** это вообще **невозможно**: директор будет пользоваться одной функциональностью, его секретарь другой, а бухгалтер третьей.

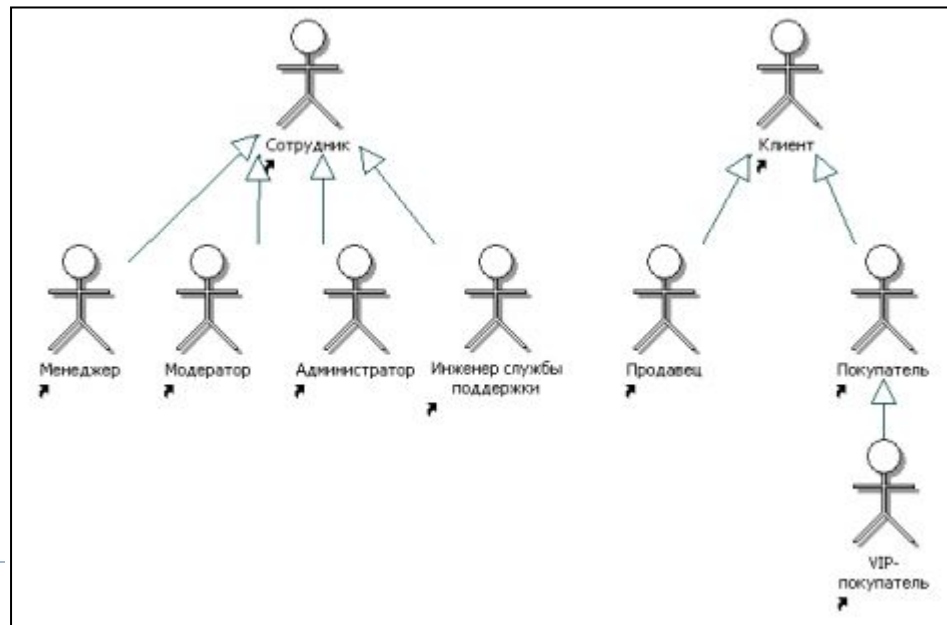


перед началом сбора требований должны быть определены основные профили пользователей!!!



Разделение пользователей

- 1) Для домашних продуктов разделение обычно производится, исходя из уровня подготовленности пользователя и его интересов
- 2) Для корпоративных продуктов основным критерием является специализация работника и круг его обязанностей



Выделение требований

□ 2) Формирование инициативной группы

Когда определены профили пользователей продукта, следует найти людей, соответствующих этим профилям и желающих помочь в разработке продукта. () **это не разработчики! Люди со стороны**

Работа может считаться выполненной, когда есть как минимум один источник для каждого описанного выше профиля, и связь с источниками налажена

Выделение требований

▣ 3) Сбор пользовательских историй

Пользовательская история — это вариант использования будущего продукта в конкретной ситуации с целью достижения измеримого результата.

Пользовательские истории могут содержать как **сложные инструкции с ответвлениями**, так и **конкретные примеры**.

Важно: пользовательская история **не описывает способ работы с конкретным продуктом и использует только термины предметной области**. Она должна быть понятной и привычной людям, которые знают процесс, автоматизируемый продуктом. **Способ реализации всегда должен оставаться за кадром** (он будет определен на более поздних этапах).

Выделение требований

▣ 3) Сбор пользовательских историй

Пользовательская история — это вариант использования будущего продукта в конкретной ситуации с целью достижения измеримого результата.

Пользовательские истории могут содержать как **сложные инструкции с ответвлениями**, так и **конкретные примеры**.

Важно: пользовательская история **не описывает способ работы с конкретным продуктом и использует только термины предметной области**. Она должна быть понятной и привычной людям, которые знают процесс, автоматизируемый продуктом. **Способ реализации всегда должен оставаться за кадром** (он будет определен на более поздних этапах).

Варианты формализации требований

- Формализация требований в проекте может быть очень разной – это зависит от его **величины, принятого процесса разработки, используемых инструментальных средств**, а также тех **задач**, которые решают формализованные требования.