

Лекции 3-4

«Математические методы в объектно-ориентированном проектировании»

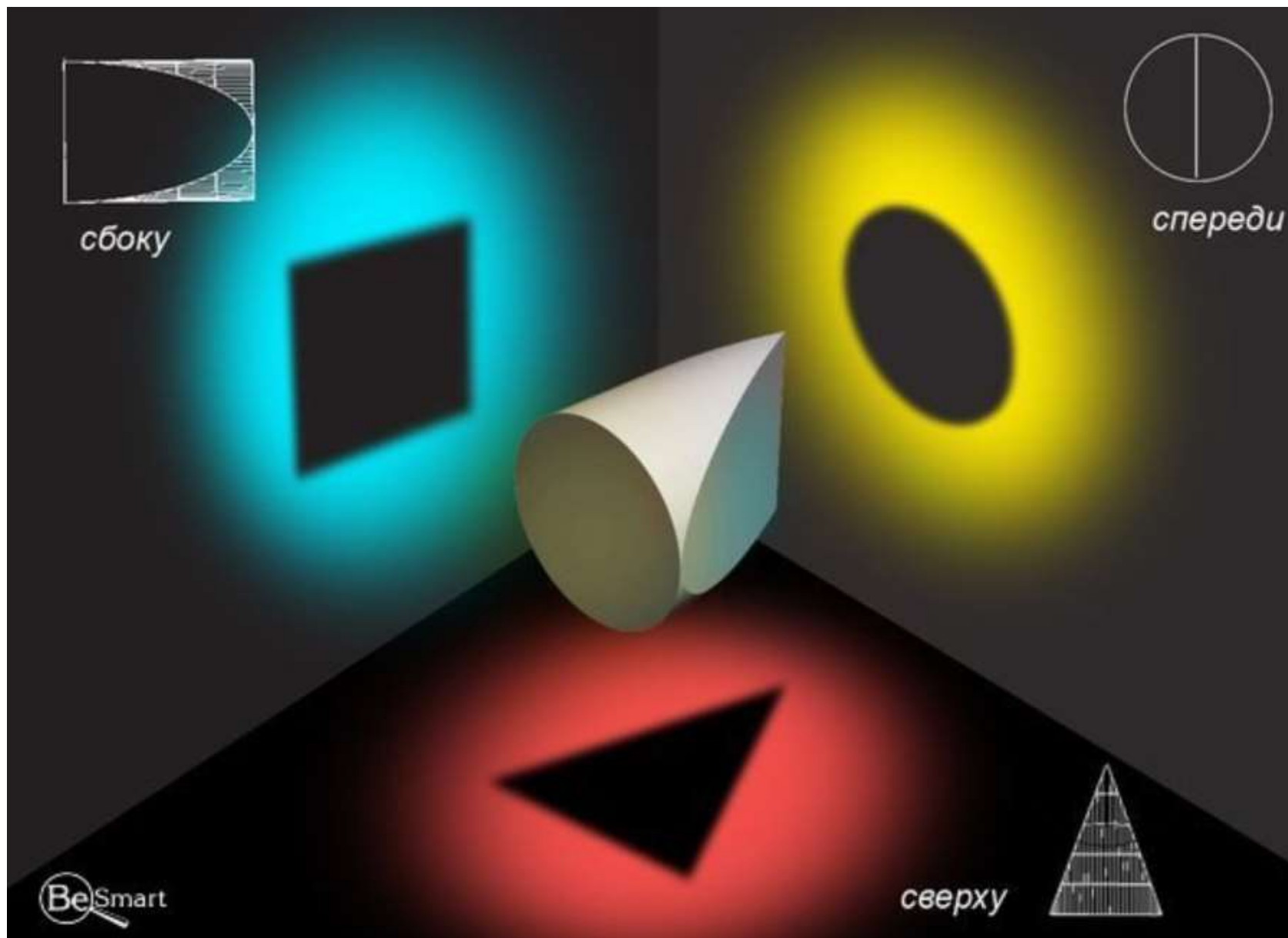
Овчинников П.Е.
МГТУ «СТАНКИН»,
ст.преподаватель кафедры ИС

Лекция 3

«Информационные системы как системы массового обслуживания»

Овчинников П.Е.
МГТУ «СТАНКИН»,
ст.преподаватель кафедры ИС

Точка зрения определяет все



Понятия аспекта и сложности: метрики

Показатель — обобщённая [характеристика](#) какого-либо [объекта](#), [процесса](#) или его результата, [понятия](#) или их [свойств](#), обычно, выраженная в числовой форме

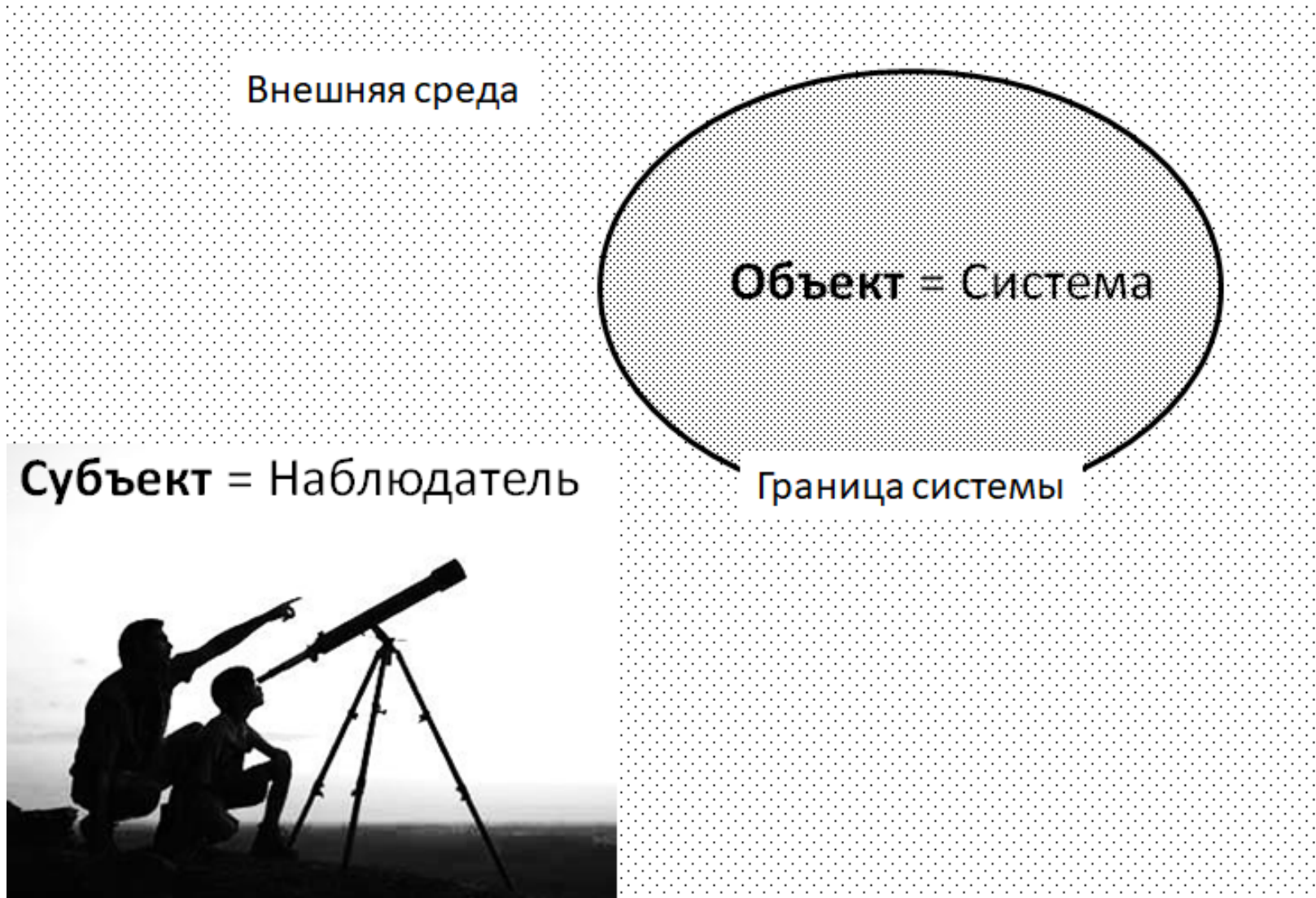
Критерий ([др.-греч.](#) κριτήριον — способность различения, средство суждения, мерило) — признак, основание, правило принятия решения по оценке чего-либо на соответствие предъявленным требованиям (мере)

Критерий в [квалиметрии](#) — условие, накладывающееся на показатель свойства предмета исследования

Мера — [философская категория](#), означающая единство [качественной](#) и [количественной](#) определённости некоторого предмета. Эта категория обобщает способы и результаты [измерения](#) предметов. Анализ меры исходит из важности интервала изменений количественных величин, в рамках которого можно говорить о сохранении качества предмета

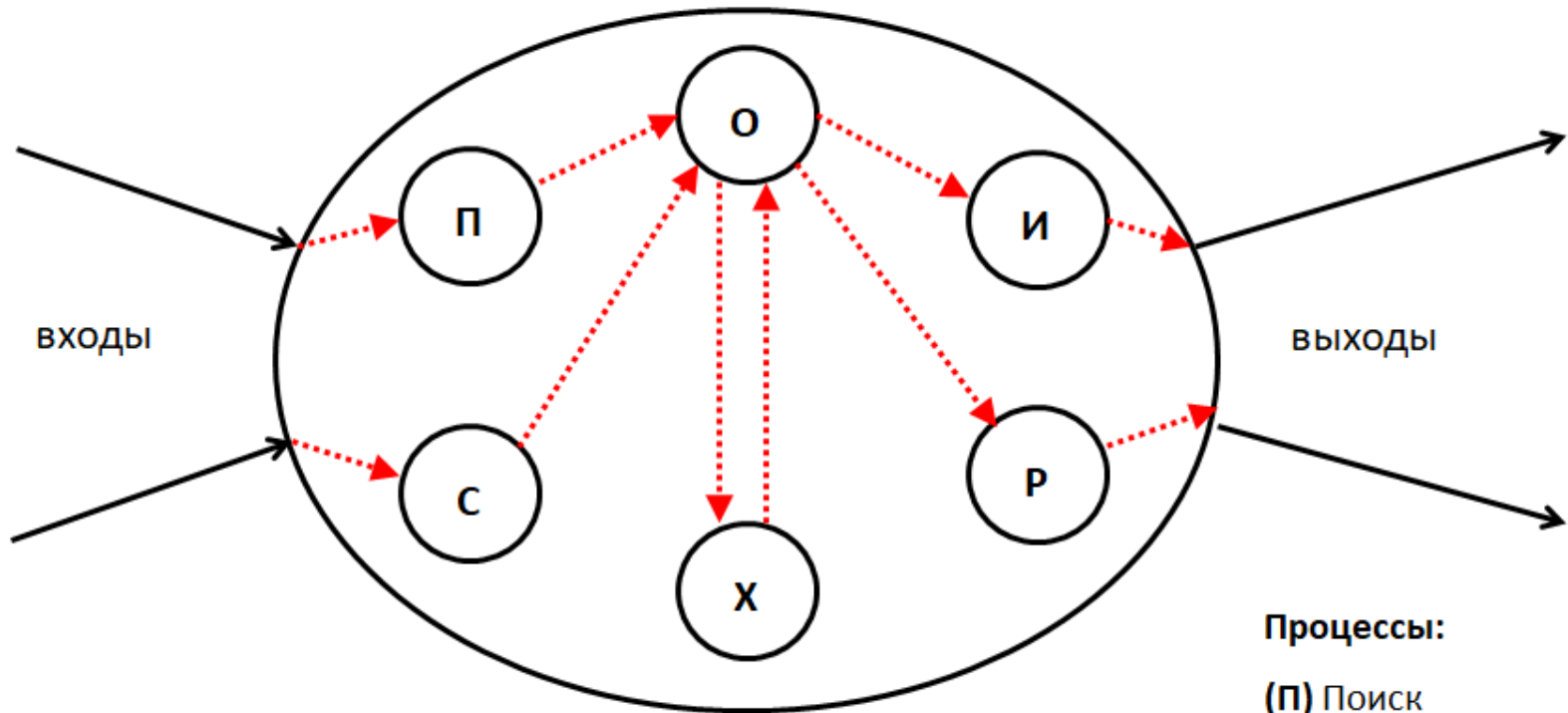
Метрика — [матем. правило](#) определения расстояния между любыми двумя точками, [техн.](#) вычисляемая [величина](#), характеризующее какое-либо явление

Понятия аспекта и сложности: система



Объектно-ориентированный подход (инженерия знаний)

Внешняя среда



Процессы:

(П) Поиск

(С) Сбор

(О) Обработка

(Х) Хранение

(И) Предоставление

(Р) Распространение

Одно из решений:

- 1. Поиск и сбор получают информацию из внешней среды*
- 2. Предоставление и распространение отправляют информацию во внешнюю среду*
- 3. Хранение взаимодействует только с обработкой*

Терминология: поток (программирование)

Поток данных ([англ. stream](#)) в программировании — абстракция, используемая для [чтения или записи файлов](#), [сокетов](#) и т. п. в единой манере.

Потоки являются удобным унифицированным [программным интерфейсом](#) для чтения или записи [файлов](#) (в том числе [специальных](#) и, в частности, связанных с [устройствами](#)), [сокетов](#) и передачи данных между [процессами](#)

Пото́к выполне́ния (тред; от [англ. thread](#) — нить) — наименьшая единица обработки, исполнение которой может быть [назначено ядром операционной системы](#).

Несколько потоков выполнения могут существовать в рамках одного и того же процесса и совместно использовать ресурсы, такие как [память](#), тогда как процессы не разделяют этих ресурсов.

В частности, потоки выполнения разделяют инструкции процесса (его код) и его контекст (значения переменных, которые они имеют в любой момент времени). В качестве аналогии потоки выполнения процесса можно уподобить нескольким вместе работающим поварам. Все они готовят одно блюдо, читают одну и ту же кулинарную книгу с одним и тем же рецептом и следуют его указаниям, причём не обязательно все они читают на одной и той же странице.

Терминология: поток (производство)

ГОСТ Р ИСО 15531-43-2011 Системы промышленной автоматизации и интеграция. Данные по управлению промышленным производством. Часть 43. Информация для управления производственными потоками. Модель данных для мониторинга и обмена производственной информацией

поток (flow): Движение множества физических или информационных объектов в пространстве и времени



Методология SADT (IDEF0)

Информационный поток - множество информационных объектов, распределенное во времени

Информация, участвующая в процессах, операциях, действиях и деятельности в целом, может быть классифицирована на **три группы:**

- 1. ограничительная**
- 2. описательная**
- 3. предписывающая (управляющая)**

Ограничительная информация - сведения о том, **что нельзя делать:**

- а) никогда, ни при каких обстоятельствах (кроме, быть может, форс-мажорных), в любой фазе жизненного цикла и на любом этапе функционирования системы в целом;
- б) в рамках функционирования конкретного блока.

Ограничительная информация содержится в законах, подзаконных актах, международных, государственных и отраслевых стандартах, а также в специальных внутренних положениях и документах предприятия, в частности, в технических требованиях, условиях, регламентах и т.д.

Методология SADT (IDEF0)

Описательная информация - сведения **об атрибутах объекта** (потока), преобразуемого функциональным блоком

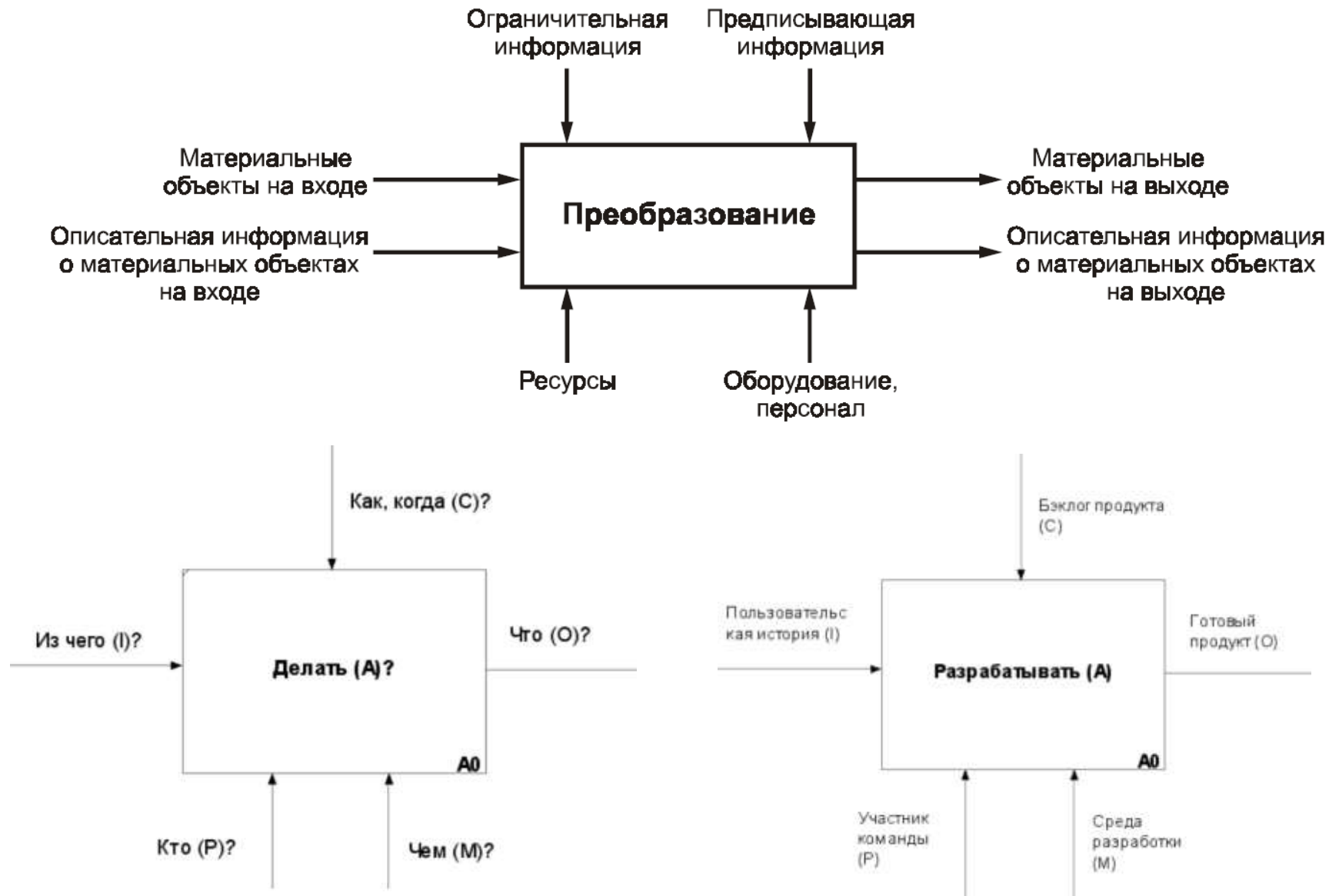
Содержится в чертежах, технических и иных описаниях, реквизитах и других документах, являясь неотъемлемым компонентом объекта в течение всего жизненного цикла

Эта информация сама преобразуется (изменяется) в результате выполнения функции.

Предписывающая (управляющая) информация - сведения о том, как, при каких условиях и по каким правилам следует **преобразовать объект** (поток) **на входе** **в объект** (поток) **на выходе** блока

Содержится в технологических (в широком смысле) инструкциях, руководствах, документах, определяющих «настройки» и характеристики блока

Методология SADT (IDEF0)



Информация и энтропия

Энтро́пия (от [др.-греч. *ἐν*](#) «в» + [τροπή](#) «поворот; превращение») — широко используемый в [естественных](#) и [точных науках](#) [термин](#).

Впервые введён в рамках [термодинамики](#) как [функция состояния термодинамической системы](#). Энтропия определяет меру необратимого рассеивания [энергии](#) или бесполезности энергии, ибо не всю энергию системы можно использовать для превращения в какую-нибудь полезную работу. Для понятия энтропии в данном разделе физики используют название термодинамическая энтропия.

В [статистической физике](#) энтропия характеризует [вероятность](#) осуществления какого-либо [макроскопического состояния](#).

Кроме физики, термин широко употребляется **в математике**: [теории информации](#) и [математической статистике](#). В этих областях знания энтропия определяется **статистически** и называется информационной (или статистической) энтропией.

Данное определение энтропии известно также как [энтропия Шеннона](#) (в математике) и [энтропия Гиббса](#) (в физике).

Информация и энтропия

Энтропия может интерпретироваться как **мера неопределённости** (неупорядоченности) некоторой системы, например, **какого-либо опыта** (испытания), который может иметь разные исходы

Математический смысл информационной энтропии — это логарифм числа **доступных состояний системы**

Выражение для информационной энтропии Шеннона:

$$H = \log \bar{N} = - \sum_{i=1}^N p_i \log p_i.$$

Основание логарифма может быть различным, но большим 1, оно определяет **единицу измерения** энтропии:

- бит (двоичный логарифм, основание = 2)
- нат (натуральный логарифм, основание = число e ≈ 2,718281828459045...)
- трит (троичный логарифм, основание = 3)
- децит (десятичный логарифм, основание = 10)
- байт (основание = 256)

Информация и энтропия

Собственная информация — статистическая функция дискретной случайной величины

Собственная информация сама является случайной величиной, которую следует отличать от её среднего значения — информационной энтропии

Для случайной величины X , имеющей конечное число значений:

$$P_X(x_i) = p_i, \quad p_i \geq 0, i = 1, 2, \dots, n, \quad \sum_{i=1}^n p_i = 1$$

собственная информация определяется как

$$I(X) = -\log P_X(X)$$

Количество информации является **мерой снятой неопределенности**:
числовое значение количества информации о некотором объекте равно **разности априорной и апостериорной энтропии**:

$$I(X, Y) = H(Y) - H(Y/X)$$

Объектно-ориентированный подход (инженерия знаний)

ГОСТ 7.0-99 СИБИД. Информационно-библиотечная деятельность, библиография. Термины и определения

Информация - сведения, воспринимаемые человеком и (или) специальными устройствами как отражение фактов материального или духовного мира в процессе коммуникации

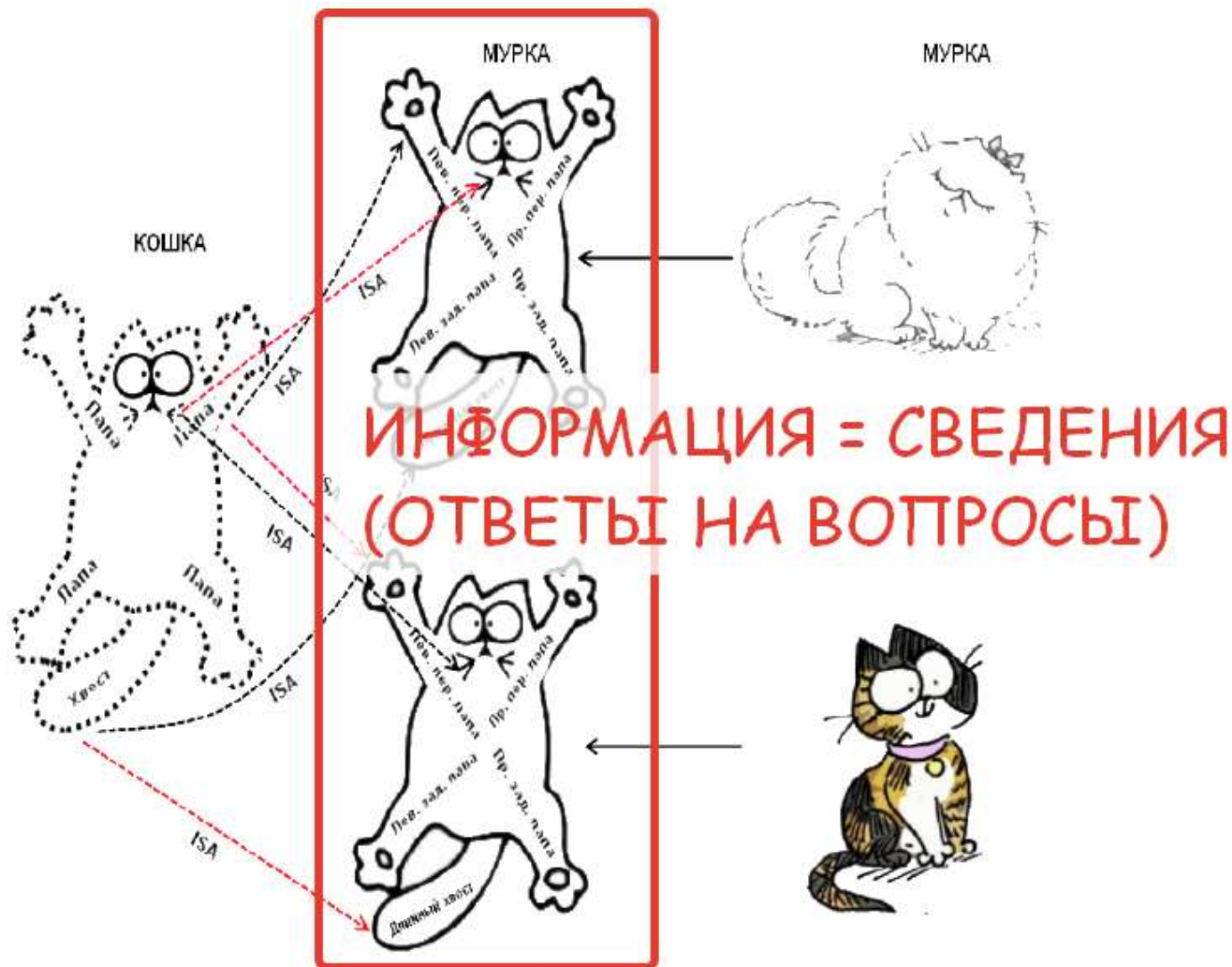
Федеральный закон от 27.07.2006 N 149-ФЗ (ред. от 21.07.2014) «Об информации, информационных технологиях и о защите информации»

Информация - сведения (сообщения, данные) независимо от формы их представления

ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными

Данные - информация, представленная в формализованном виде, пригодном для передачи, интерпретации или обработки с участием человека или автоматическими средствами

Объектно-ориентированный подход (инженерия знаний)



Измерения в ИТ

ГОСТ 8.417-2002 Государственная система обеспечения единства измерений (ГСИ). Единицы величин (с Поправкой)

Наименование величины	Единица				Примечание
	Наименование	Обозначение		Значение	
		международное	русское		
Количество информации ¹⁾	бит ²⁾	bit	бит	1	Единица информации в двоичной системе счисления (двоичная единица информации)
	байт ^{2), 3)}	В (byte)	Б (байт)	1 Б = 8 бит	

Измерения в ИТ

ГОСТ 8.417-2002 Государственная система обеспечения единства измерений (ГСИ). Единицы величин (с Поправкой)

1. Термин "количество информации" используют в устройствах цифровой обработки и передачи информации, например в цифровой вычислительной технике (компьютерах), для записи объема запоминающих устройств, количества памяти, используемой компьютерной программой.
2. В соответствии с международным стандартом МЭК 60027-2 единицы "бит" и "байт" применяют с приставками СИ (таблица 8 и раздел 7) [7].
3. Исторически сложилась такая ситуация, что с наименованием "байт" некорректно (вместо $1000=10^3$ принято $1024=2^{10}$) использовали (и используют) приставки СИ:
 - 1 Кбайт=1024 байт,
 - 1 Мбайт=1024 Кбайт,
 - 1 Гбайт=1024 Мбайт и т.д.При этом обозначение Кбайт начинают с прописной буквы в отличие от строчной буквы "к" для обозначения множителя 10^3

Измерения в ИТ

Десятичный множитель	Приставка	Обозначение приставки		Десятичный множитель	Приставка	Обозначение приставки	
		междуна- родное	русское			междуна- родное	русское
10 ²⁴	иотта	Y	И	10 ⁻¹	деци	d	Д
10 ²¹	зетта	Z	З	10 ⁻²	санتي	c	с
10 ¹⁸	экса	E	Э	10 ⁻³	милли	m	м
10 ¹⁵	пета	P	П	10 ⁻⁶	микро	μ	мк
10 ¹²	тера	T	Т	10 ⁻⁹	нано	n	н
10 ⁹	гига	G	Г	10 ⁻¹²	пико	p	п
10 ⁶	мега	M	М	10 ⁻¹⁵	фемто	f	ф
10 ³	кило	k	к	10 ⁻¹⁸	атто	a	а
10 ²	гекто	h	г	10 ⁻²¹	зепто	z	з
10 ¹	дека	da	да	10 ⁻²⁴	иокто	y	и

Объектно-ориентированный подход (инженерия знаний)

Знания - структурированная информация,
связанная причинно-следственными отношениями и образующая системы.

Научные знания могут быть:

- **эмпирическими** (на основе опыта или наблюдения)
- **теоретическими** (на основе анализа **абстрактных моделей**)

Научные знания в любом случае должны быть **обоснованными** на эмпирической или теоретической доказательной основе.

Теоретические знания — абстракции, аналогии, схемы, отображающие структуру и природу процессов изменения объектов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для **прогнозирования** поведения объектов

Имитационное моделирование

Теорема Байеса является фундаментальной теоремой в байесовской статистике, так как она используется байесовскими методами для обновления вероятностей, которые являются степенью доверия, после получения новых данных

Если даны два события **A** и **B**, условная вероятность **A**, при условии, что **B** верно, выражается формулой

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)}$$

$P(B | A)$ является функцией правдоподобия, которую можно интерпретировать как вероятность свидетельства **B**, при условии, что произошло событие **A**

Правдоподобие даёт количественное значение степени, насколько свидетельство **B** поддерживает утверждение **A**

Имитационное моделирование

Мéтоды Мóнте-Кáрло (ММК) — группа численных методов для изучения случайных процессов

Суть метода заключается в следующем: процесс моделируется при помощи генератора **случайных величин**

Это повторяется **много раз**, а потом на основе полученных случайных данных вычисляются вероятностные характеристики решаемой задачи

Например, чтобы узнать, какое в среднем будет расстояние между двумя случайными точками в круге, методом Монте-Карло, нужно:

- взять много случайных пар точек,
- для каждой пары найти расстояние, а потом
- усреднить

Имитационное моделирование

Погрешность измерения — отклонение измеренного значения величины от её истинного (действительного) значения. Погрешность измерения является характеристикой точности измерения

Выяснить с абсолютной точностью истинное значение измеряемой величины, как правило, невозможно, поэтому невозможно и указать величину отклонения измеренного значения от истинного

При использовании численных методов возникает еще несколько видов погрешностей:

- при приближении одного числа другим возникает погрешность **округления**
- погрешность, связанная с неточными начальными **данными** называется неустранимой
- в связи с заменой исходной задачи на приближённую существует погрешность **метода**

Полная погрешность при этом складывается из погрешности метода и погрешности вычислений

Имитационное моделирование

Рассмотрим задачу разработки **прикладной программы**, демонстрирующей действие **центральной предельной теоремы** (Ц.П.Т.) и **закона больших чисел** [1], а также основные понятия информационных методов статистической обработки данных [2].

Как и во всяком проекте разработки программных средств, до начала непосредственного написания кода требуемой программы должны быть определены и специфицированы требования к результатам, проведено эскизное и техническое проектирование, составлен план разработки, тестирования и сдачи работы заказчику.

В крупных и масштабных проектах для выполнения указанных работ формируются специальные проектные команды, в состав которых кроме программистов включаются специалисты для заполнения необходимых проектных ролей [3,4,5,6,7,8]:

- менеджер продукта (руководитель разработки);
- ведущий программист (**системный архитектор**);
- системный аналитик;
- дизайнер;
- тестировщик;
- технический писатель и др.

Имитационное моделирование

432 lines (416 sloc) | 23.9 KB

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title>Пример ЦПТ</title>
6   <link rel="stylesheet" type="text/css" href="gauss12.css"/>
7   <script type="text/javascript">
```

```
8 </head>
9 <body>
```

```
10   <table>
11     <tr>
12       <td class="form">
13         <div class="input">
14           <input type="text"/>
15         </div>
16       </td>
17     </tr>
18   </table>
```

154 lines (122 sloc) | 2.4 KB

```
1 table {
2   border-collapse: collapse;
3   width: 100%;
4   table-layout: fixed;
5 }
```

```
6
7 .blue {
8   color: rgba(255,255,255,0.5);
9   text-shadow: #2e7eb3 2px 2px 0px;
10  border-color: #60a3d8;
11  background: #60a3d8;
12  box-shadow: inset 0 0 0 2px #2e7eb3;
13 }
```

```
14
15 .orange {
16   color: rgba(255,255,255,0.5);
```

249 lines (211 sloc) | 8.2 KB

```
1 var diagramColor = 'blueDiagram';
2 var diagramBorder = '';
3 var startTime;
4
5
6 /**
7  * Demonstrate Central limit theorem.
8  */
9 function count() {
10   startTime = new Date().getTime();
11
12   var n = parseInt(document.getElementById("number_of_sources").value) || 0;
13   var k = parseInt(document.getElementById("number_of_elements").value) || 0;
14
15   clearHistogram();
16   clearResult();
```

Имитационное моделирование

Основные параметры

Количество источников:

Количество элементов в выборке:

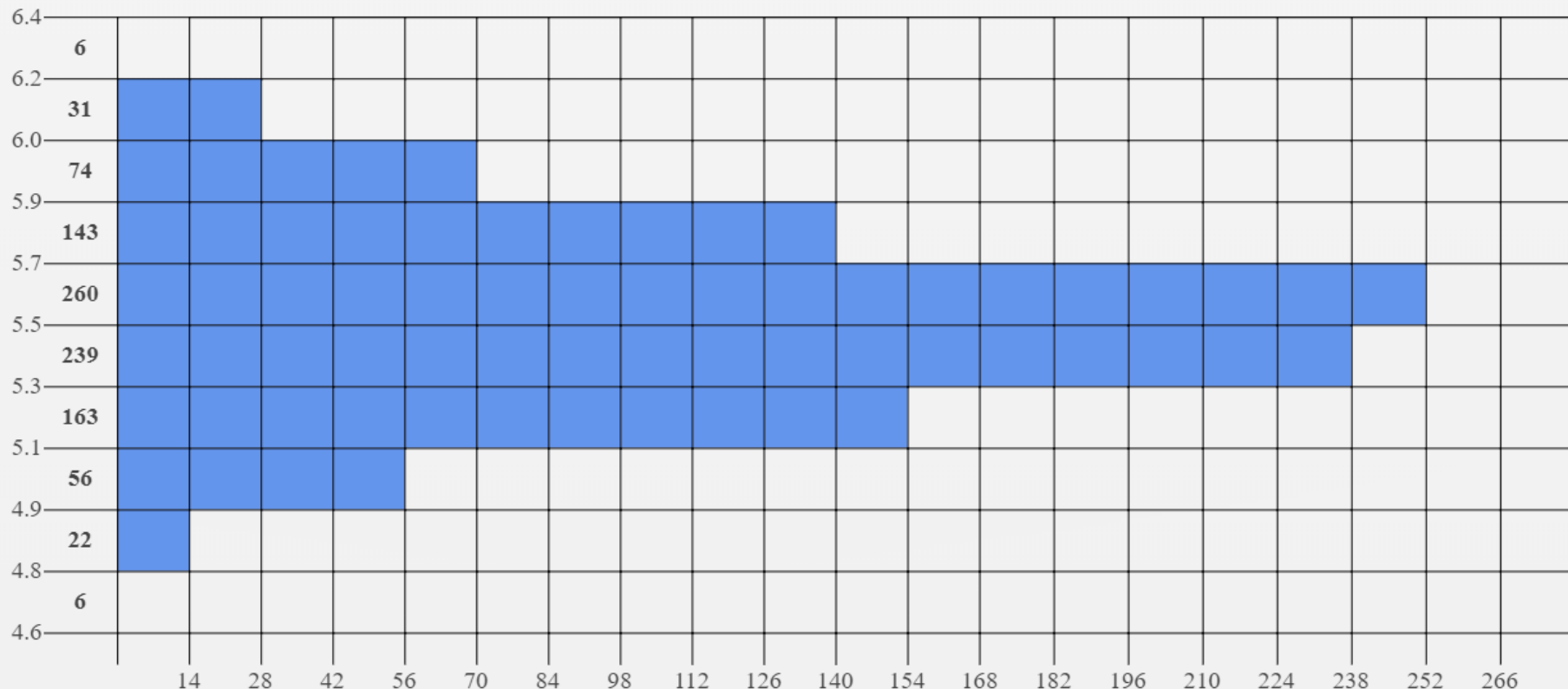
Дополнительно

☒ Показывать сетку

Цвет фона:

☒ Посчитать энтропию

Цвет диаграммы:



Энтропия (нат) = 1.87 Мат. ожидание = 5.51 Среднеквадратическое отклонение = 0.29
Время работы (мс) = 72

[Текст практикума](#)

[Практикум \(приложение в работе\)](#)

Лекция 4

«Понятия исполнительного устройства и очереди в системе массового обслуживания»

Овчинников П.Е.

МГТУ «СТАНКИН»,

ст.преподаватель кафедры ИС

Информационная система как СМО

Система массового обслуживания (СМО) — система, которая производит обслуживание **поступающих** в неё **требований**

Обслуживание требований в СМО осуществляется **обслуживающими приборами**

В зависимости от наличия **возможности ожидания** поступающими требованиями начала обслуживания СМО подразделяются на:

- **системы с потерями**, в которых требования, не нашедшие в момент поступления ни одного свободного прибора, теряются
- **системы с ожиданием**, в которых имеется накопитель бесконечной ёмкости для буферизации поступивших требований, при этом ожидающие требования образуют **очередь**
- **системы с накопителем** конечной ёмкости (ожиданием и ограничениями), в которых длина очереди не может превышать ёмкости накопителя; при этом требование, поступающее в переполненную СМО (отсутствуют свободные места для ожидания), теряются

Информационная система как СМО

Выбор требования **из очереди** на обслуживание производится с помощью так называемой дисциплины обслуживания, например:

- FCFS/FIFO (пришедший первым обслуживается первым)
- LCFS/LIFO (пришедший последним обслуживается первым)
- random (случайный выбор)

Основные понятия СМО:

- **требование** (заявка) — запрос на обслуживание.
- **входящий поток требований** — совокупность требований, поступающих в СМО
- **время обслуживания** — период времени, в течение которого обслуживается требование
- **математическая модель** — это совокупность математических выражений, описывающих:
 - входящий поток требований,
 - процесс обслуживания и
 - их взаимосвязь

Очередь (ИТ)

Очередь сообщений (или **почтовый ящик**) — в информатике — программно-инженерный компонент, используемый для межпроцессного или межпотокowego взаимодействия внутри одного процесса

Для обмена сообщениями используется очередь

Парадигма очереди сообщений сродни шаблону издатель-подписчик и обычно является частью более крупной системы промежуточного программного обеспечения, ориентированной на обработку сообщений

Большинство систем обмена сообщениями в своих API поддерживают модели как очереди сообщений, так и «издатель-подписчик»

Очередь и стек (программирование)

Очередь — абстрактный тип данных с дисциплиной доступа к элементам «первый пришёл — первый вышел» (FIFO, англ. *first in, first out*)

Добавление элемента (принято обозначать словом **enqueue** — поставить в очередь) возможно лишь в конец очереди, выборка — только из начала очереди (что принято называть словом **dequeue** — убрать из очереди), при этом выбранный элемент из очереди удаляется

Стек (англ. *stack* — стопка; читается *стэк*) — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. *last in — first out*, «последним пришёл — первым вышел»)

В цифровом вычислительном комплексе стек называется магазином — по аналогии с магазином в огнестрельном оружии (стрельба начнётся с патрона, заряженного последним)

Очередь и стек (программирование)

Очередь с приоритетом ([англ. priority queue](#)) — [абстрактный тип данных](#) в [программировании](#), поддерживающий две обязательные операции — добавить элемент и извлечь максимум (минимум)

Предполагается, что для каждого элемента можно вычислить его **приоритет** — действительное число или в общем случае элемент [линейно упорядоченного множества](#)

В качестве примера очереди с приоритетом можно рассмотреть **список задач работника**. Когда он **заканчивает** одну задачу, он **переходит** к очередной — самой приоритетной (ключ будет величиной, обратной приоритету) — то есть выполняет операцию извлечения максимума

Начальник добавляет **задачи в список**, указывая их приоритет, то есть выполняет операцию добавления элемента

Модно: Agile

Гибкая методология разработки ([англ. Agile software development, agile-методы](#)) — серия подходов к [разработке программного обеспечения](#), ориентированных на использование [итеративной](#) разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля

Существует несколько методик, относящихся к классу гибких методологий разработки, в частности [экстремальное программирование](#), [DSDM](#), [Scrum](#), [FDD](#).

- ❑ **Люди и взаимодействия** важнее чем процессы и инструменты
- ❑ **Работающий код** важнее совершенной документации
- ❑ **Сотрудничество с заказчиком** важнее контрактных обязательств
- ❑ **Реакция на изменения** важнее следования плану

Методология XP

Экстремальное программирование ([англ. Extreme Programming, XP](#)) — одна из [гибких методологий разработки программного обеспечения](#).

Двенадцать основных приёмов экстремального программирования (по первому изданию книги *Extreme programming explained*) могут быть объединены в четыре группы:

Короткий цикл обратной связи (Fine-scale feedback)

- [Разработка через тестирование](#) (Test-driven development)

- Игра в планирование (Planning game)

- Заказчик всегда рядом (Whole team, Onsite customer)

- [Парное программирование](#) (Pair programming)

Непрерывный, а не пакетный процесс

- [Непрерывная интеграция](#) (Continuous integration)

- [Рефакторинг](#) (Design improvement, Refactoring)

- Частые небольшие релизы (Small releases)

Понимание, разделяемое всеми

- Простота проектирования (Simple design)

- Метафора системы

- Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)

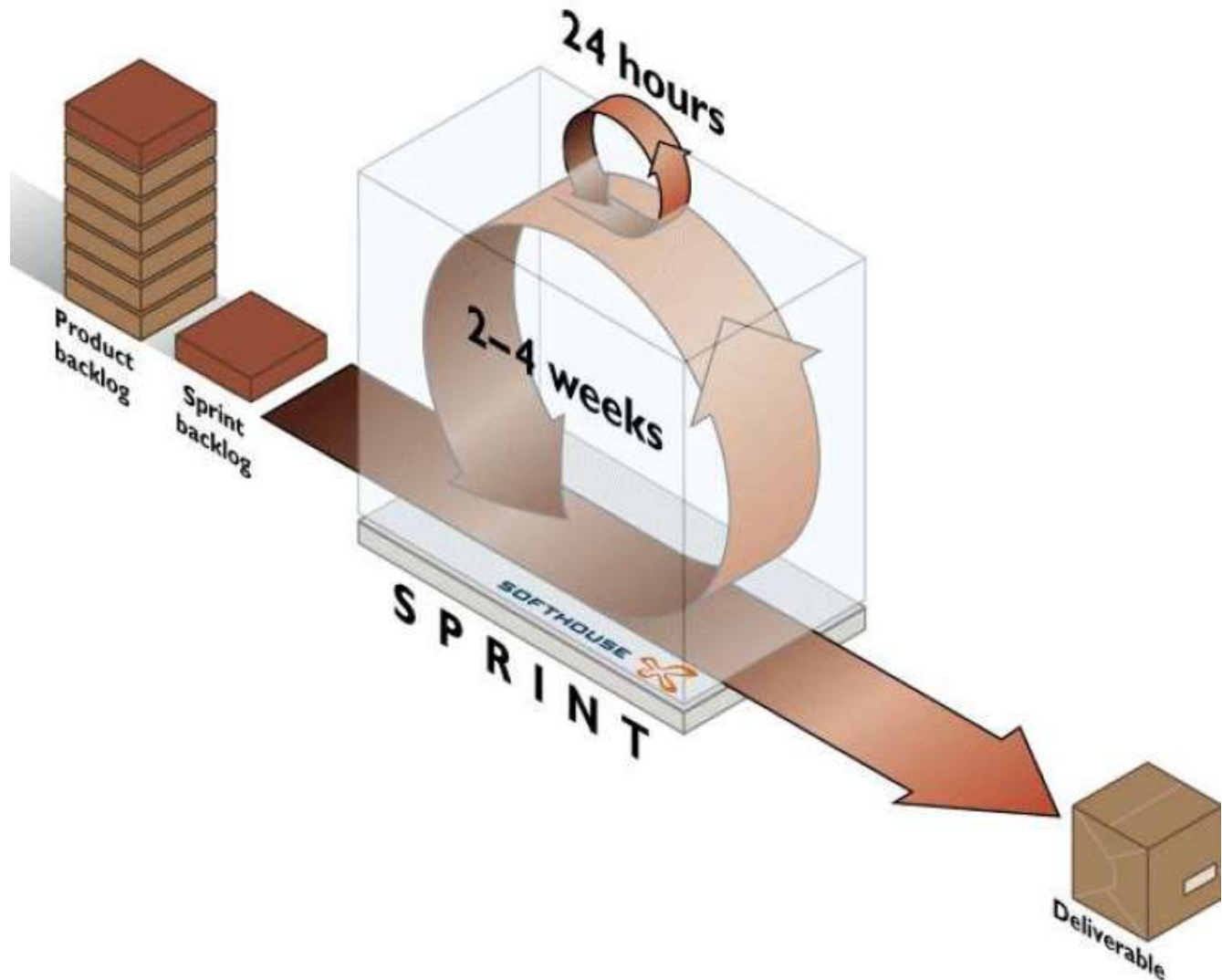
- [Стандарт оформления кода](#) (Coding standard or Coding conventions)

Социальная защищённость программиста (Programmer welfare):

- 40-часовая рабочая неделя (Sustainable pace, Forty-hour week)

Методология Scrum

Scrum ([/skrʌm/](#); [англ.](#) *scrum* «схватка») — методология [гибкой разработки](#) ПО. Методология делает акцент на качественном контроле процесса разработки.



Метод Канбан

Канбан ([яп.](#) カンバン *камбан*)¹ — система **организации производства** и снабжения, позволяющая реализовать принцип «[точно в срок](#)». Слово «канбан» по-японски означает «рекламный щит, вывеска» ([яп.](#) 看板), в финансовой среде устоялся вариант с ошибочной транскрипцией латинской записи японского слова (kanban)

Канбан — **метод управления разработкой**, реализующий принцип «[точно в срок](#)» и способствующий равномерному распределению нагрузки между работниками. При данном подходе весь процесс разработки прозрачен для всех членов команды.

Задачи по мере поступления **заносятся в отдельный список**, откуда каждый разработчик может **извлечь требуемую задачу**

RUP: итеративная модель

RUP использует **итеративную модель разработки**.

В конце каждой итерации (в идеале продолжающейся от 2 до 6 недель) проектная команда должна:

- достичь запланированных на данную итерацию целей,
- создать или доработать проектные артефакты и
- получить промежуточную, но функциональную версию конечного продукта.

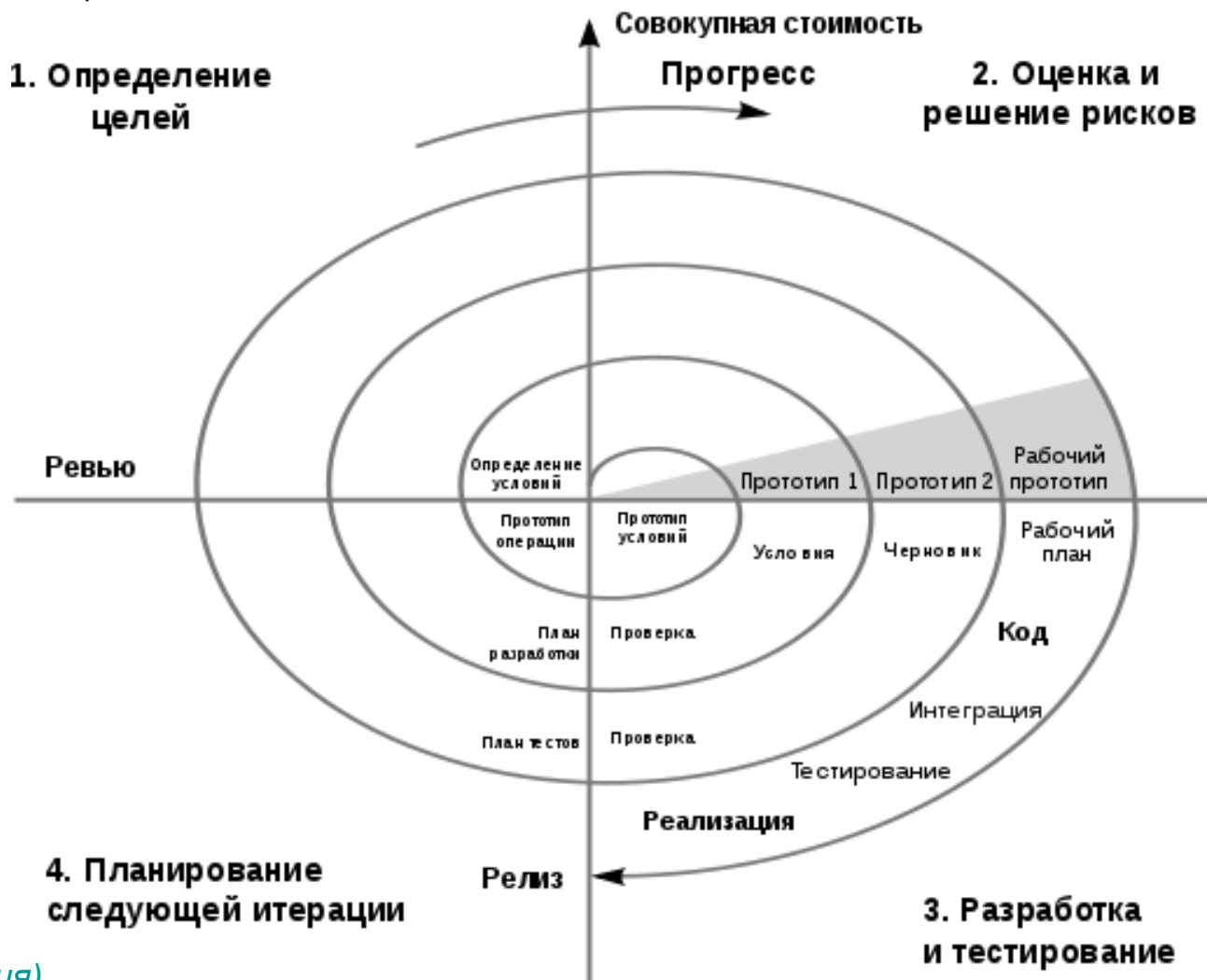
Итеративная разработка позволяет:

- быстро реагировать на меняющиеся требования,
- обнаруживать и устранять риски на ранних стадиях проекта, а также
- эффективно контролировать качество создаваемого продукта.

Первые идеи итеративной модели разработки были заложены в "спиральной модели"

RUP: итеративная модель

Спиральная модель, предложенная Барри Боэмом в [1986 году](#), представляет собой [процесс разработки программного обеспечения](#), сочетающий в себе как итеративность, так и этапность.



FDD

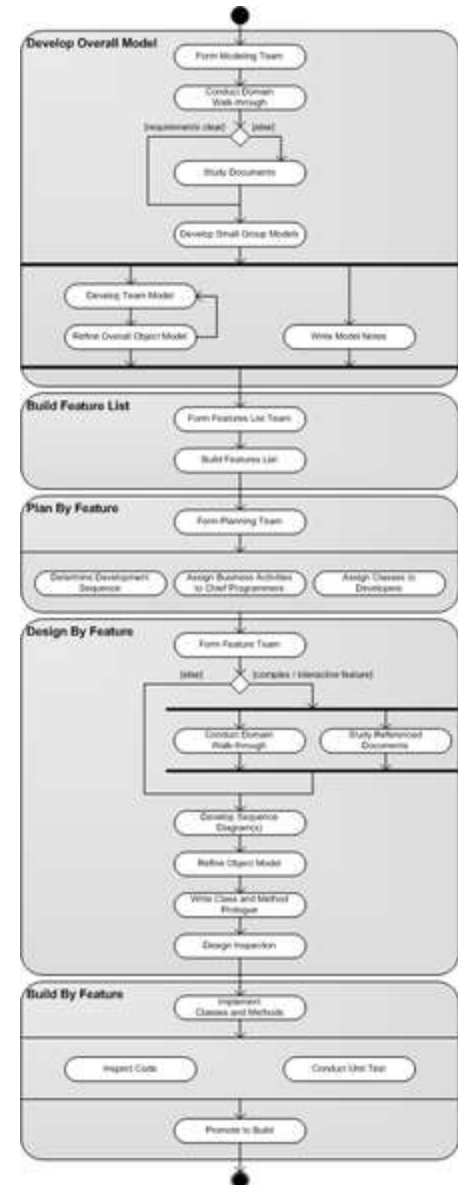
Feature driven development (FDD, разработка, управляемая функциональностью) — итеративная методология разработки программного обеспечения, одна из гибких методологий разработки (agile)

FDD включает в себя пять базовых видов деятельности:

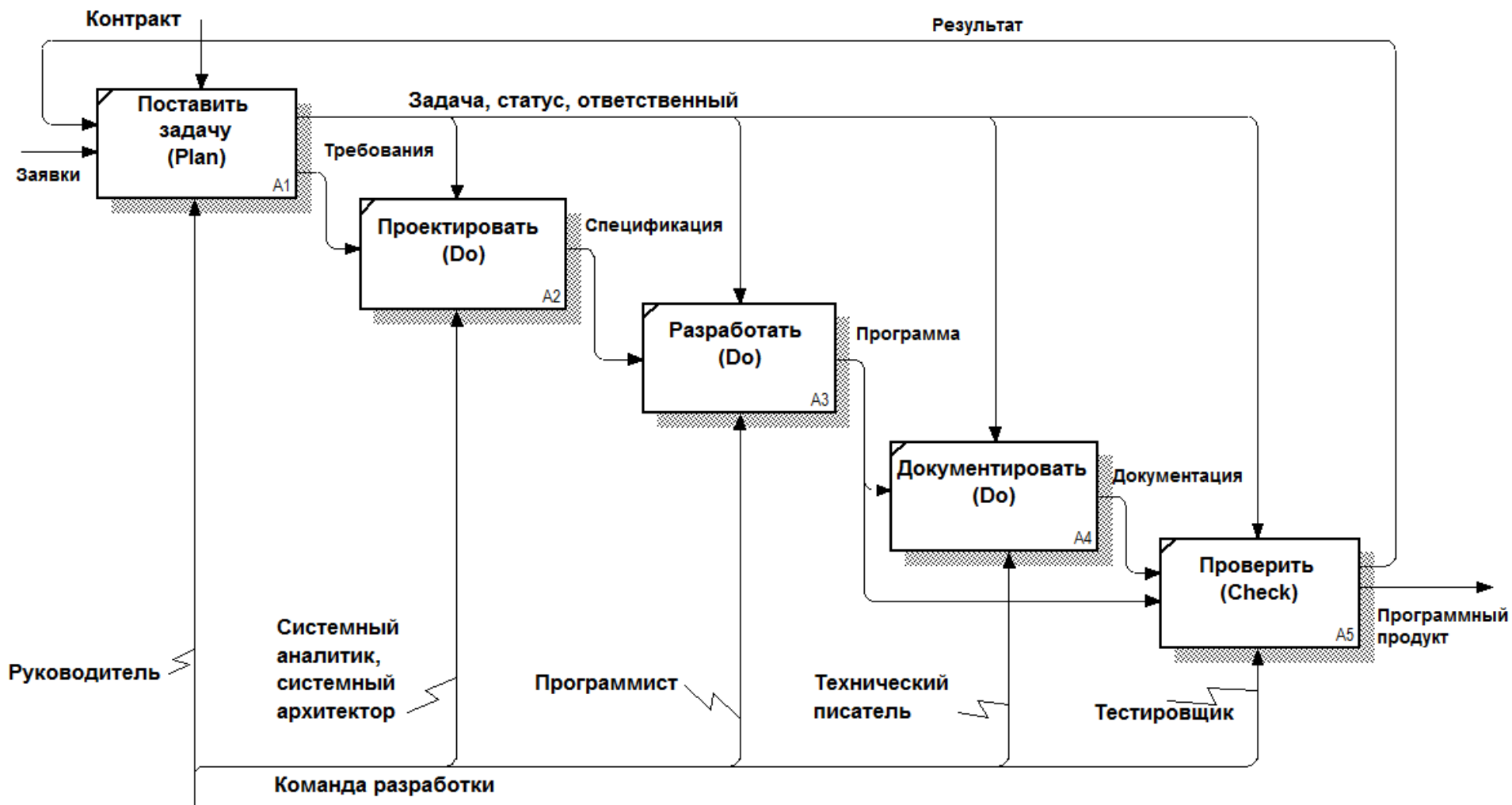
1. разработка общей модели
2. составление списка необходимых функций системы
3. планирование работы над каждой функцией
4. проектирование функции
5. реализация функции

FDD

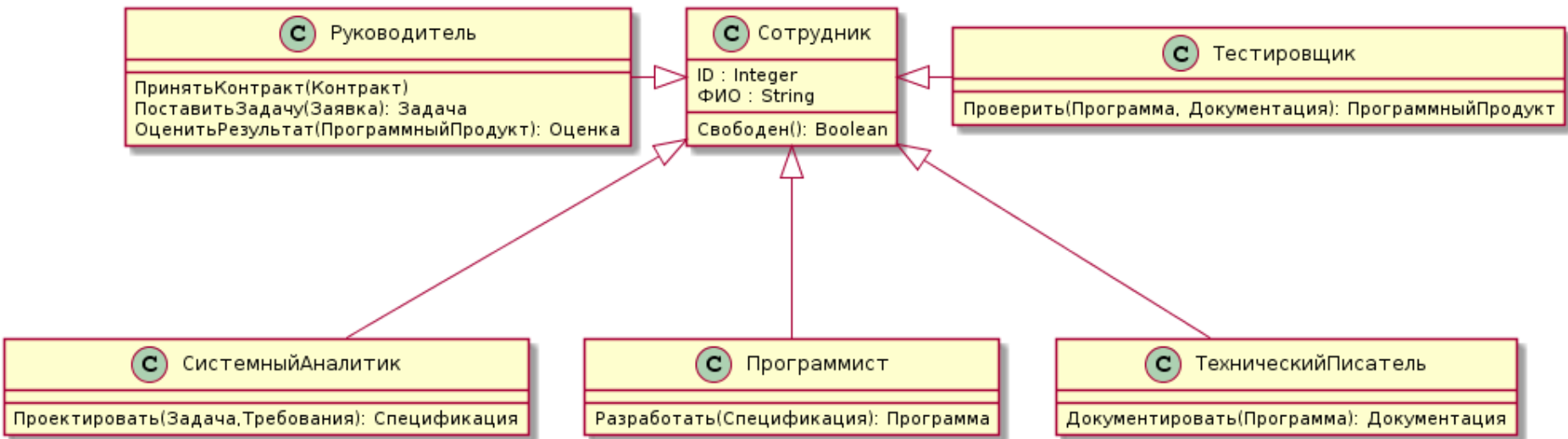
Анализ и оценка методов разработки программного обеспечения (Agile)
Анализ требований к автоматизированным информационным системам



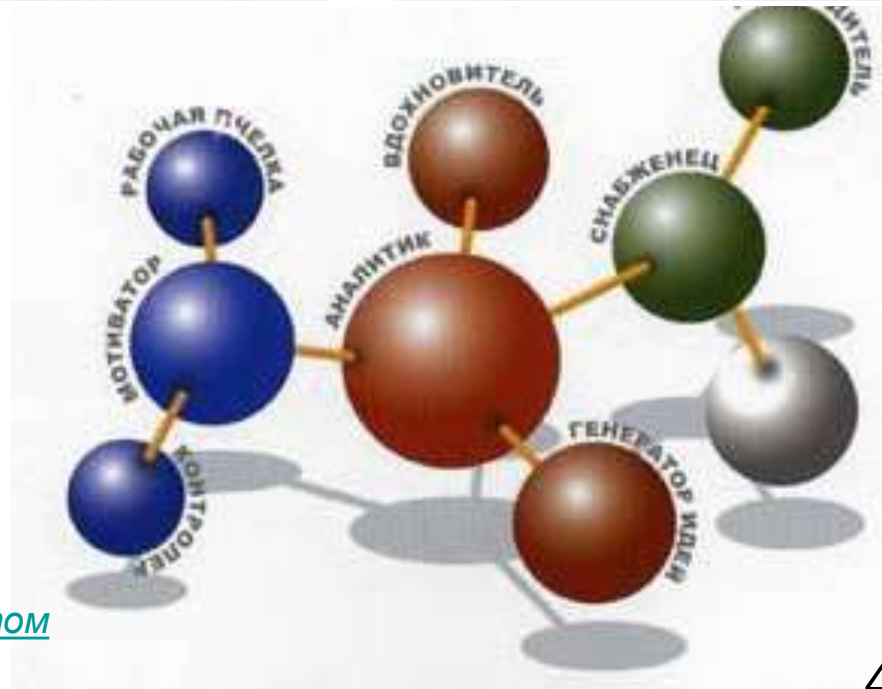
FDD и цикл Деминга (PDCA)



Модель FDD и команда



Аналитик
Вдохновитель
Генератор идей
Контролер
Мотиватор
Рабочая пчелка
Руководитель



Формирование команды и управление проектом
Роли в команде (теория М.Белбина)

Веб-разработка: AJAX

AJAX, Ajax (['eɪdʒæks](#), от [англ. Asynchronous Javascript and XML](#) — «асинхронный [JavaScript](#) и [XML](#)») — подход к построению интерактивных [пользовательских интерфейсов веб-приложений](#), заключающийся в «фоновом» [обмене данными браузера](#) с [веб-сервером](#)

В результате, при обновлении данных [веб-страница](#) не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. При использовании AJAX:

- Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент
- [Скрипт](#) (на языке [JavaScript](#)) определяет, какая информация необходима для обновления страницы
- [Браузер](#) отправляет соответствующий запрос на [сервер](#)
- Сервер возвращает только ту часть документа, на которую пришёл запрос
- [Скрипт](#) вносит изменения с учётом полученной информации (без полной перезагрузки страницы)

Веб-разработка: AJAX

AJAX базируется на использовании технологий:

- динамического обращения к [серверу](#) «на лету», без перезагрузки всей страницы полностью, например с использованием [XMLHttpRequest](#) и
- использования [DHTML](#) для динамического изменения содержания страницы

XMLHttpRequest (XMLHTTP, XHR) — [API](#), доступный в [скриптовых языках браузеров](#), таких как [JavaScript](#)

Использует запросы [HTTP](#) или [HTTPS](#) напрямую к [веб-серверу](#) и загружает данные ответа сервера напрямую в вызывающий скрипт

Информация может передаваться в любом [текстовом формате](#), например, в [XML](#), [HTML](#) или [JSON](#). Позволяет осуществлять HTTP-запросы к серверу без перезагрузки страницы

Веб-разработка: AJAX

Метод	Описание
<code>abort()</code>	Отменяет текущий запрос, удаляет все заголовки, ставит текст ответа сервера в null.
<code>getAllResponseHeaders()</code>	Возвращает полный список HTTP-заголовков в виде строки. Заголовки разделяются знаками переноса (CR+LF). Если флаг ошибки равен true, возвращает пустую строку. Если статус 0 или 1, вызывает ошибку <code>INVALID_STATE_ERR</code> .
<code>getResponseHeader(headerName)</code>	Возвращает значение указанного заголовка. Если флаг ошибки равен true, возвращает null. Если заголовок не найден, возвращает null. Если статус 0 или 1, вызывает ошибку <code>INVALID_STATE_ERR</code> .
<code>open(method, URL, async, userName, password)</code>	Определяет метод, URL и другие опциональные параметры запроса; параметр <code>async</code> определяет, происходит ли работа в асинхронном режиме. Последние два параметра необязательны.
<code>send(content)</code>	Отправляет запрос на сервер.
<code>setRequestHeader(label, value)</code>	Добавляет HTTP-заголовок к запросу.
<code>overrideMimeType(mimeType)</code>	Позволяет указать mime-type документа, если сервер его не передал или передал неправильно.

Веб-разработка: AJAX

Свойство	Тип	Описание
onreadystatechange	EventListener	Обработчик события, которое происходит при каждой смене состояния объекта. Имя должно быть записано в нижнем регистре.
readyState	unsigned short	Текущее состояние объекта (0 — не инициализирован, 1 — открыт, 2 — отправка данных, 3 — получение данных и 4 — данные загружены)
responseText	DOMString	Текст ответа на запрос. Если состояние не 3 или 4, возвращает пустую строку.
responseXML	Document	Текст ответа на запрос в виде XML, который затем может быть обработан посредством DOM . Если состояние не 4, возвращает null.
status	unsigned short	HTTP-статус в виде числа (404 — «Not Found», 200 — «OK» и т. д.)
statusText	DOMString	Статус в виде строки («Not Found», «OK» и т. д.). Если статус не распознан, браузер пользователя должен вызвать ошибку INVALID_STATE_ERR.

jQuery ajax() Method

[← jQuery AJAX Methods](#)

Example

Change the text of a <div> element using an AJAX request:

```
$("button").click(function(){  
    $.ajax({url: "demo_test.txt", success: function(result){  
        $("#div1").html(result);  
    }});  
});
```

[Try it Yourself »](#)