Лекции 1-2
«Основные задачи и методы проектирования программных средств»

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

06.02.2019

Дальше

План дисциплины

Лекции

Семинары

Семинар 1 Методология RUP, прецеденты и язык UML
Семинар 2 Управление разработкой ПО, методы Канбан и Скрам
Семинар 3 Прототипирование в разработке ПО
Семинар 4 Автоматизированное тестирование в разработке ПО

Лабораторные работы

Лабораторная 1 Разметка личной страницы, разработка UML Use Case
Лабораторная 2 Заполнение личной страницы, разработка UML Class
Лабораторная 3 Реализация мини-приложения SPA, разработка UML Sequence
Лабораторная 4 Реализация приложения на Angular 7, разработка UML Component
Лабораторная 5 Разработка алгоритма и описания процедуры в выбранном проекте
Лабораторная 6 Разработка процедуры на JS в выбранном проекте
Лабораторная 7 Разработка unit-теста процедуры в выбранном проекте

Экзамен

Вопросы к экзамену

История

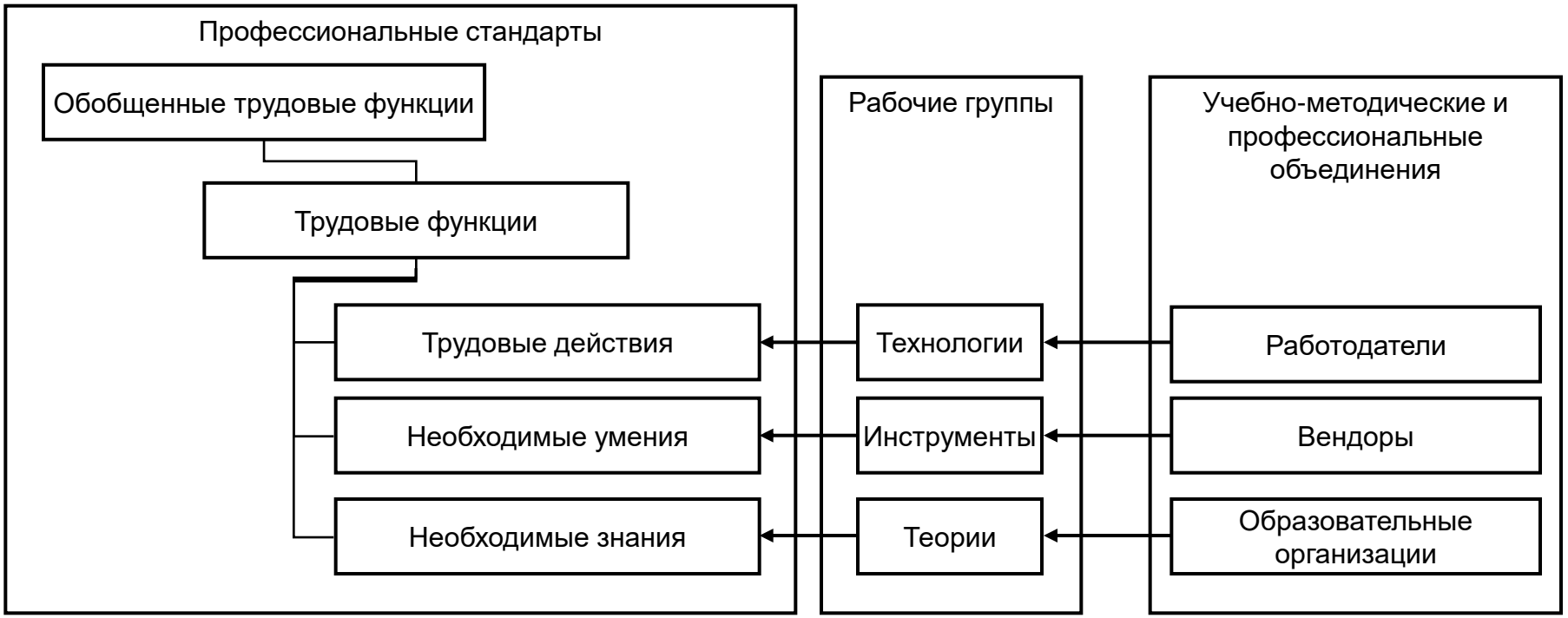
Материалы 2017
Материалы 2018

Лекция 1
«Роль и обязанности архитектора программных средств»

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

06.02.2019

Структура профессиональных стандартов



Предыдущий семестр: бизнес-анализ

ПРОФЕССИОНАЛЬНЫЙ СТАНДАРТ

Архитектор программного обеспечения

67

Регистрационный номер

I. Общие сведения

Проектно-конструкторская деятельность | 06.003

(наименование вида профессиональной деятельности) | Код

Основная цель вида профессиональной деятельности:

Создание и сопровождение архитектуры программных средств, заключающейся
- в синтезе и документировании решений о структуре;
- компонентном устройстве;
- основных показателях назначения;
- порядке и способах реализации программных средств в рамках системной архитектуры;
- реализации требований к программным средствам;
- контроле реализации и ревизии решений

Архитектор: основные обязанности

Основная цель вида профессиональной деятельности - создание и сопровождение архитектуры программных средств, заключающейся в синтезе и документировании решений о:

- структуре
- компонентном устройстве
- основных показателях назначения
- порядке и способах реализации программных средств в рамках системной архитектуры
- реализации требований к программным средствам
- контроле реализации и ревизии решений

Синтез + документирование = проектирование!

Предыдущий семестр: сохранение связей

Модель процессов

Модель архитектуры ИС

Модель данных ИС

Архитектор: определяет показатели назначения

ГОСТ 34.602-89 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы

2.6.1.3. В требованиях к показателям назначения АС приводят значения **параметров**, характеризующие степень **соответствия** системы ее **назначению**.

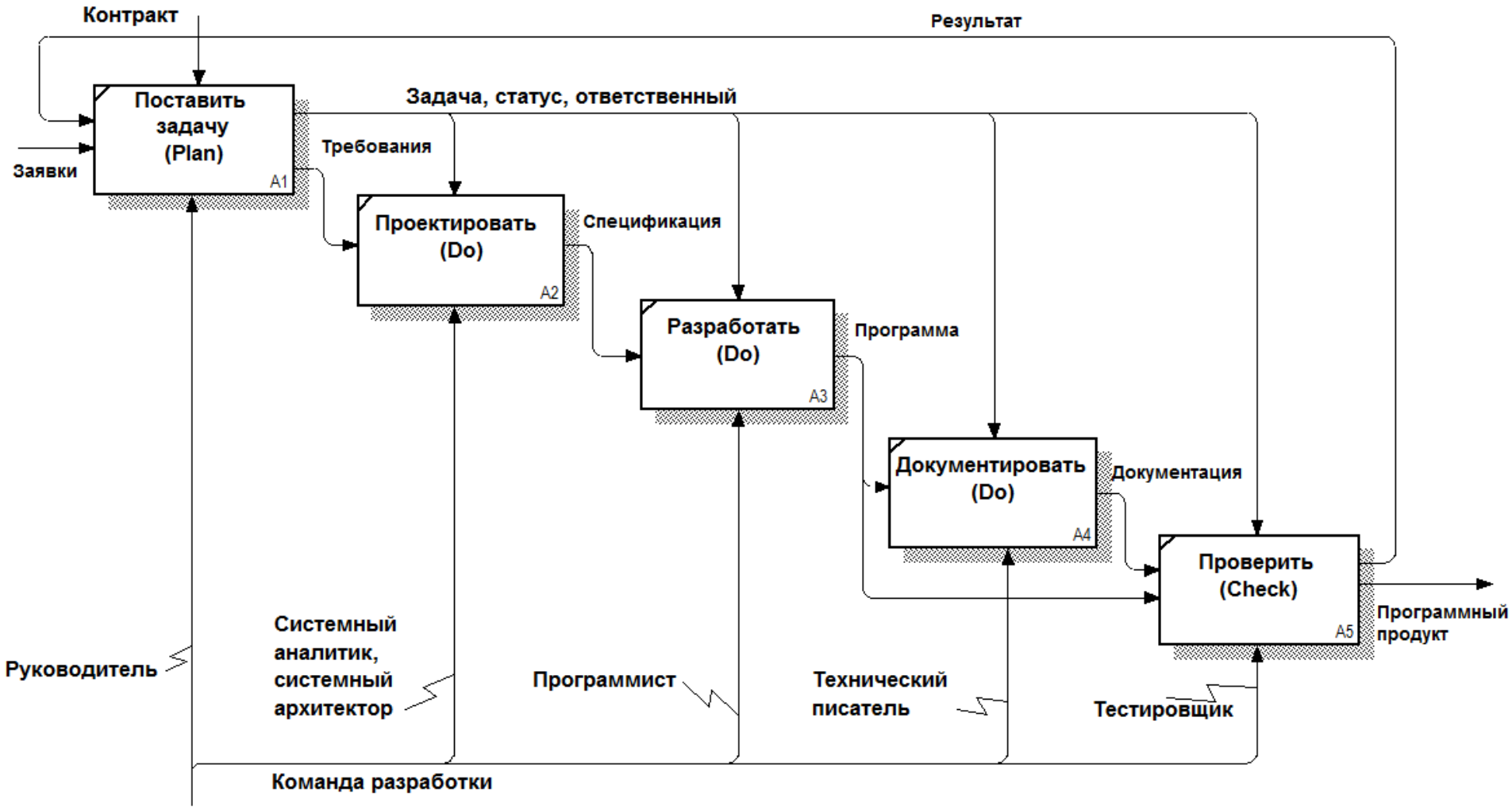
Для АСУ указывают:

- **степень приспособляемости** системы к изменению процессов и методов управления, к отклонениям параметров объекта управления;

- **допустимые пределы модернизации** и развития системы;

- **вероятностно-временные характеристики**, при которых **сохраняется целевое назначение** системы.

Роль архитектора: модель разработки и команда



Позиция исполнителя ИТ: «Программист»

II. Описание трудовых функций, входящих в профессиональный стандарт (функциональная карта вида профессиональной деятельности)

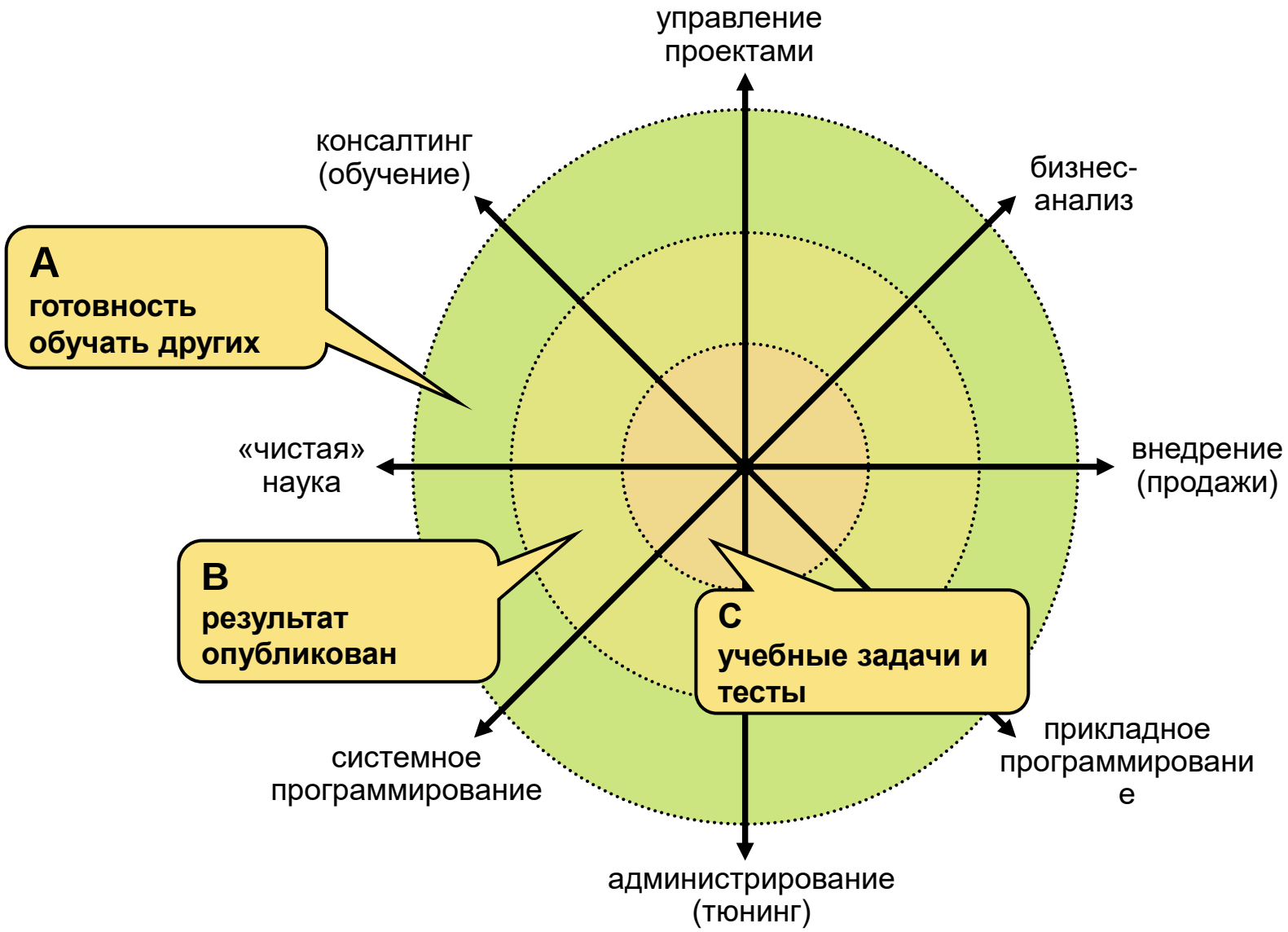
Обобщенные трудовые функции			Трудовые функции		
код	наименование	уровень квалификации	наименование	код	уровень (подуровень) квалификации
A	Разработка и отладка программного кода	3	Формализация и алгоритмизация поставленных задач	A/01.3	3
			Написание программного кода с использованием языков программирования, определения и манипулирования данными	A/02.3	3
			Оформление программного кода в соответствии с установленными требованиями	A/03.3	3
			Работа с системой контроля версий	A/04.3	3
			Проверка и отладка программного кода	A/05.3	3
B	Проверка работоспособности и рефакторинг кода программного обеспечения	4	Разработка процедур проверки работоспособности и измерения характеристик программного обеспечения	B/01.4	4
			Разработка тестовых наборов данных	B/02.4	4
			Проверка работоспособности программного обеспечения	B/03.4	4
			Рефакторинг и оптимизация программного кода	B/04.4	4
			Исправление дефектов, зафиксированных в базе данных дефектов	B/04.5	4

Позиция заказчика ИТ: «Менеджер по информационным технологиям»

II. Описание трудовых функций, входящих в профессиональный стандарт (функциональная карта вида профессиональной деятельности)

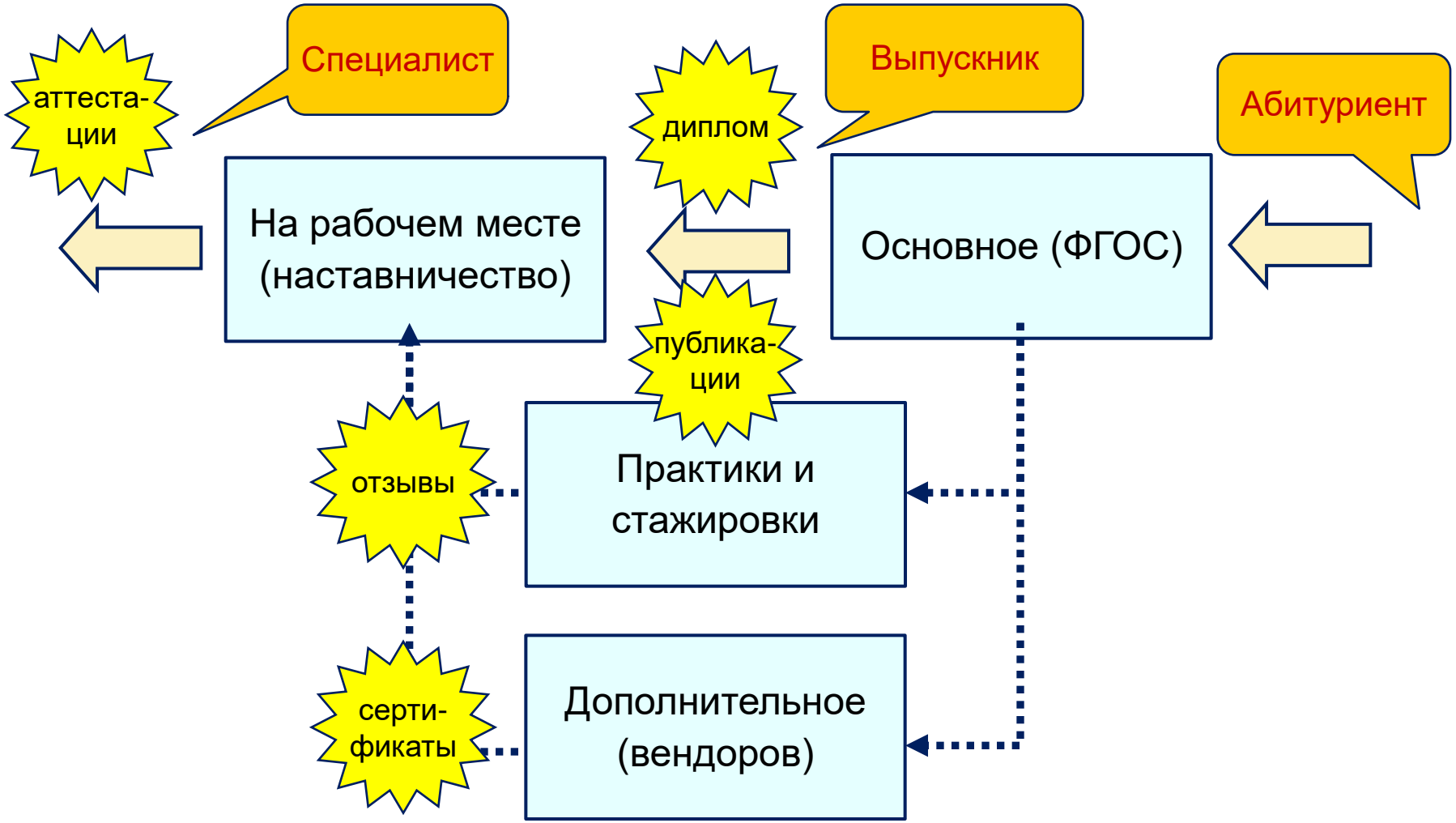
Обобщенные трудовые функции			Трудовые функции		
код	наименование	уровень квалификации	наименование	код	уровень (подуровень) квалификации
A	Управление ресурсами ИТ	6	Управление качеством ресурсов ИТ	A/01.6	6
			Управление ИТ-инфраструктурой	A/02.6	6
			Управление расходами на ИТ	A/03.6	6
			Управление изменениями ресурсов ИТ	A/04.6	6
			Управление отношениями с поставщиками и потребителями ресурсов ИТ	A/05.6	6
			Управление персоналом, обслуживающим ресурсы ИТ	A/06.6	6
			Управление информационной безопасностью ресурсов ИТ	A/07.6	6
B	Управление сервисами ИТ	7	Управление договорами об уровне предоставления сервисов ИТ (SLA)	B/01.7	7
			Управление ИТ-проектами	B/02.7	7
			Управление моделью предоставления сервисов ИТ	B/03.7	7
			Управление изменениями сервисов ИТ	B/04.7	7
			Управление отношениями с пользователями и поставщиками сервисов ИТ	B/05.7	7
			Управление персоналом, осуществляющим предоставление сервисов ИТ	B/06.7	7
			Управление непрерывностью сервисов ИТ	B/07.7	7
C	Управление информационной средой	8	Управление стратегией ИТ	C/01.8	8
			Управление программами и портфелями ИТ-проектов	C/02.8	8
			Управление формированием и внедрением системы показателей оценки эффективности ИТ	C/03.8	8
			Управление изменениями информационной среды	C/04.8	8
			Управление отношениями с поставщиками и потребителями	C/05.8	8

Многообразие профессий: упрощаем задачу

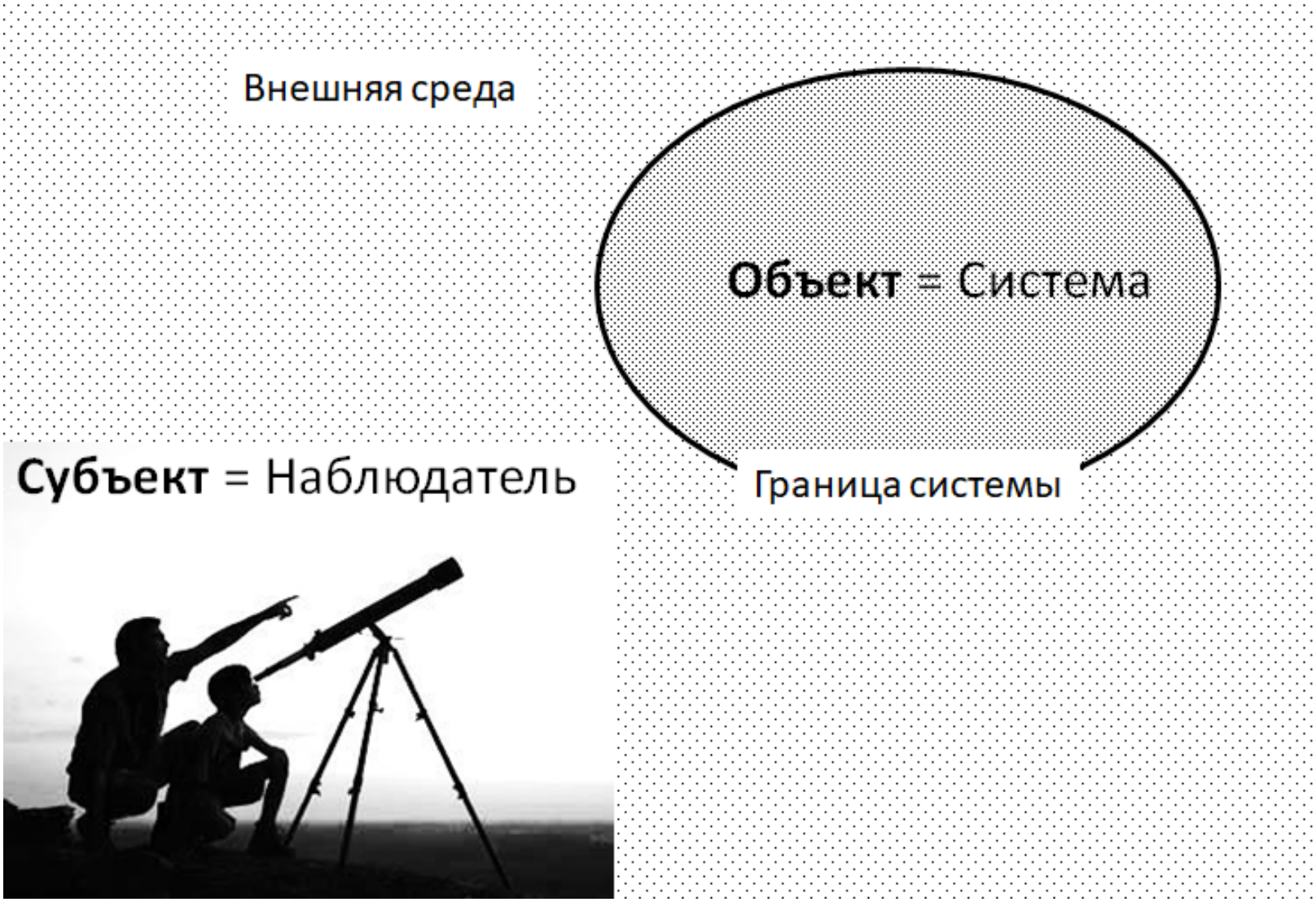


Путь к профессионализму: электронное портфолио

ISO/IEC 20013



Понятия аспекта и сложности: система



Р 50.1.028-2001

Понятия аспекта и сложности: метрики

Показа́тель — обобщённая характеристика какого-либо объекта, процесса или его результата, понятия или их свойств, обычно, выраженная в числовой форме

Крите́рий (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

Ме́ра — философская категория, означающая единство качественной и количественной определённостей 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

Ме́трика — *матем.* правило определения расстояния между любыми двумя точками , *техн.* вычисляемая величина, характеризующее какое-либо явление

Критерий (Википедия)

Понятие аспекта и сложности: система

Р 50.1.028-2001 Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования

4.1 Модель - искусственный объект, представляющий собой отображение (образ) системы и ее компонентов.

Считается, что **М** (модель) *моделирует* **А** (оригинал),
если **М** *отвечает на вопросы относительно* **А**

Модель разрабатывают для:
- понимания,
- анализа и
- принятия решений

Модель разрабатывают при:
- реконструкции (реинжиниринге) или
- замене существующей, либо
- проектировании новой системы.

Р 50.1.028-2001

Понятие аспекта и сложности: система

Система представляет собой совокупность **взаимосвязанных и взаимодействующих частей**, выполняющих **некоторую полезную работу**

Частями (элементами) системы могут быть любые комбинации разнообразных **сущностей**, включающие:

- людей
- информацию
- программное обеспечение
- оборудование
- изделия
- сырье или
- энергию (энергоносители)

Модель описывает:

- что происходит в системе
- как ею управляют
- что она преобразует
- какие средства использует для выполнения своих функций и
- что производит

Р 50.1.028-2001

Понятие аспекта и сложности: система

Аспект - точка зрения, сторона, с которой рассматривается какое-либо явление, понятие

Р 50.1.028-2001 Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования

2.39 точка зрения: Указание на **должностное лицо** или **подразделение** организации, с позиции которого разрабатывается модель

Для каждой модели точка зрения **единственная**

Точка зрения определяет, что и в каком разрезе можно увидеть **в пределах контекста** модели.

Изменение точки зрения приводит к рассмотрению **других аспектов** объекта

Аспекты, важные с одной точки зрения, могут не появиться в модели, разрабатываемой с другой точки зрения **на тот же самый объект**

Р 50.1.028-2001

Понятие аспекта и сложности: алгоритмы

В рамках классической теории, осуществляется **классификация задач** по их сложности (P-сложные, NP-сложные, экспоненциально сложные и другие):

«**P**» — могут быть решены за время, полиномиально **зависящее от объёма исходных данных**, с помощью детерминированной вычислительной машины (например, «машина Тьюринга»);

«**NP**»:

Задачи, решение которых осуществимо за полиномиально выраженное время с помощью недетерминированной вычислительной машины (следующее состояние которой не всегда однозначно определяется предыдущими). Её работу можно представить как разветвляющийся на каждой неоднозначности процесс: задача решена, если хотя бы одна ветвь достигла ответа;

Задачи, решение которых с помощью дополнительной информации полиномиальной длины, данной нам свыше, мы можем проверить за полиномиальное время. В частности, к классу «**NP**» относятся все задачи, решение которых можно *проверить* за полиномиальное время.

Класс «**P**» содержится в «**NP**».

Классическим примером **NP-задачи** является *«Задача о коммивояжёре»*.

Понятие аспекта и сложности: IFPUG

Function points are now a standard **sizing** metric

Function points are now a standard **productivity** metric

Function points are now a powerful **quality** metric

Function points are now a powerful **schedule** metric

Function points are now a powerful **staffing** metric

Function points are now used in software **litigation**

Function points are now used for **outsource contracts**

Function points can be used for **cost analysis** (with care)

Function points can be used for **value analysis** (with care)

Понятие аспекта и сложности: IFPUG

Function points used for **portfolio analysis**

Function points used for **backlog analysis**

Function points used for **risk analysis**

Function points used for **real-time requirements changes**

Function points used for **software usage studies**

Function points used for **delivery analysis**

Function points used for **COTS analysis**

Function points used for **occupation group analysis**

Function points used for **maintenance analysis**

Понятие сложности: эволюция индустрии

Moving from software development to software delivery

Development rates < 12.5 function points per staff month

Delivery rates > 100 function points per staff month

Delivery issues: quality, security, band width

Delivery methods:

- Service Oriented Architecture (SOA)
- Software as a Service (SaaS)
- DevOps and other new methodologies
- Mashups from reusable components

Понятие сложности: эволюция индустрии

Measure **delivered features** as well as development

Measure and **catalog** reusable features

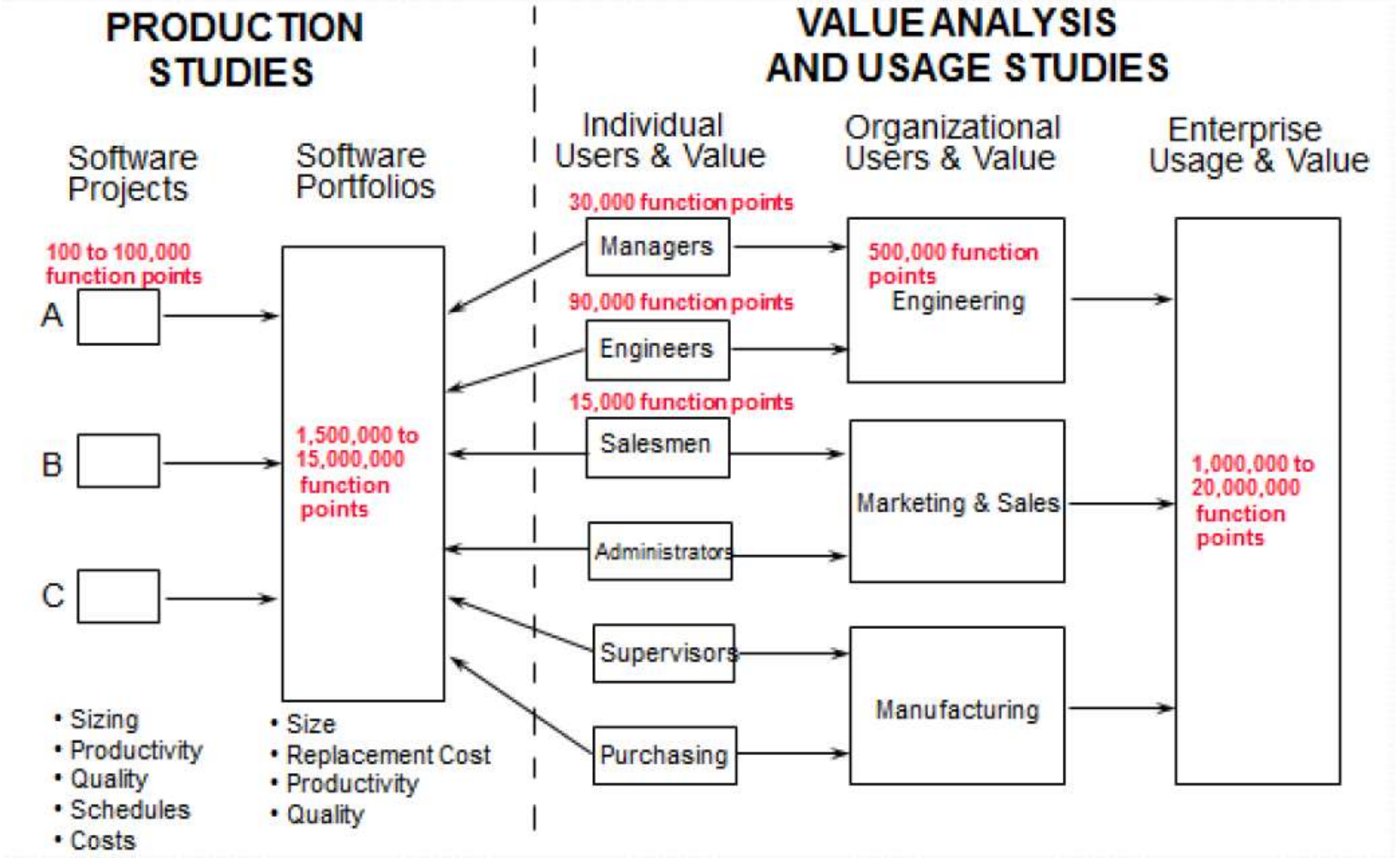
Measure **deployment**, **installation**, and **usage**

Measure **quality** and **security**

New measurements needed:

- Sizes of reusable components
- Quality, security of reusable components
- Sizes of delivered applications
- Deployment and usage of delivered applications

Понятие сложности: эволюция индустрии



Функциональные точки

Понятие сложности: примеры

Applications	Approximate Size in Function Points
Star Wars Missile Defense	300,000
ERP (SAP, Oracle, etc.)	250,000
Microsoft Windows 10	198,000
Microsoft Office	98,000
Airline Reservations	50,000
NASA space shuttle	25,000

Понятие сложности: примеры

Application = 1,000 Function points

Activity	Staff	Effort (months)	Schedule (months)
Requirements	2	5	2.5
Design	2	6	3.0
Coding	5	40	8.0
Testing	7	25	7.0
Documentation	1	7	7.0
Management	1	25	24.0
OVERALL	4.5	108	24.0

Понятие сложности: примеры

Function points ^ 0.40 power = calendar months in schedule

Function points ^ 1.15 power = pages of paper documents

Function points ^ 1.20 power = number of test cases

Function points ^ 1.25 power = software defect potential

Function points / 150 = development technical staff

Function points / 1,500 = maintenance technical staff

Понятие сложности: примеры

The Global JXDM endeavor began in March 2001 as a reconciliation of data definitions and evolved into a broad two-year effort to develop an XML-based framework that would enable the entire justice and public safety communities to effectively share information at all levels--laying the foundation for local, state, tribal, and national justice interoperability.

Approximately 16,000 justice and public safety-related data elements were collected from various local and state government sources.

These were analyzed and reduced to around 2,000 unique data elements that were then incorporated into about **300 data objects**, or reusable components, resulting in the Global Justice XML Data Dictionary (Global JXDD).

GJXDM
GJXDM (статистика)

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

ГОСТ 7.0-99 *ГОСТ 34.321-96* *Знание*

Объектно-ориентированный подход (инженерия знаний)

[ГОСТ 7.0-99](#) [ГОСТ 34.321-96](#) [Знание](#)

Объектно-ориентированный подход (инженерия знаний)

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

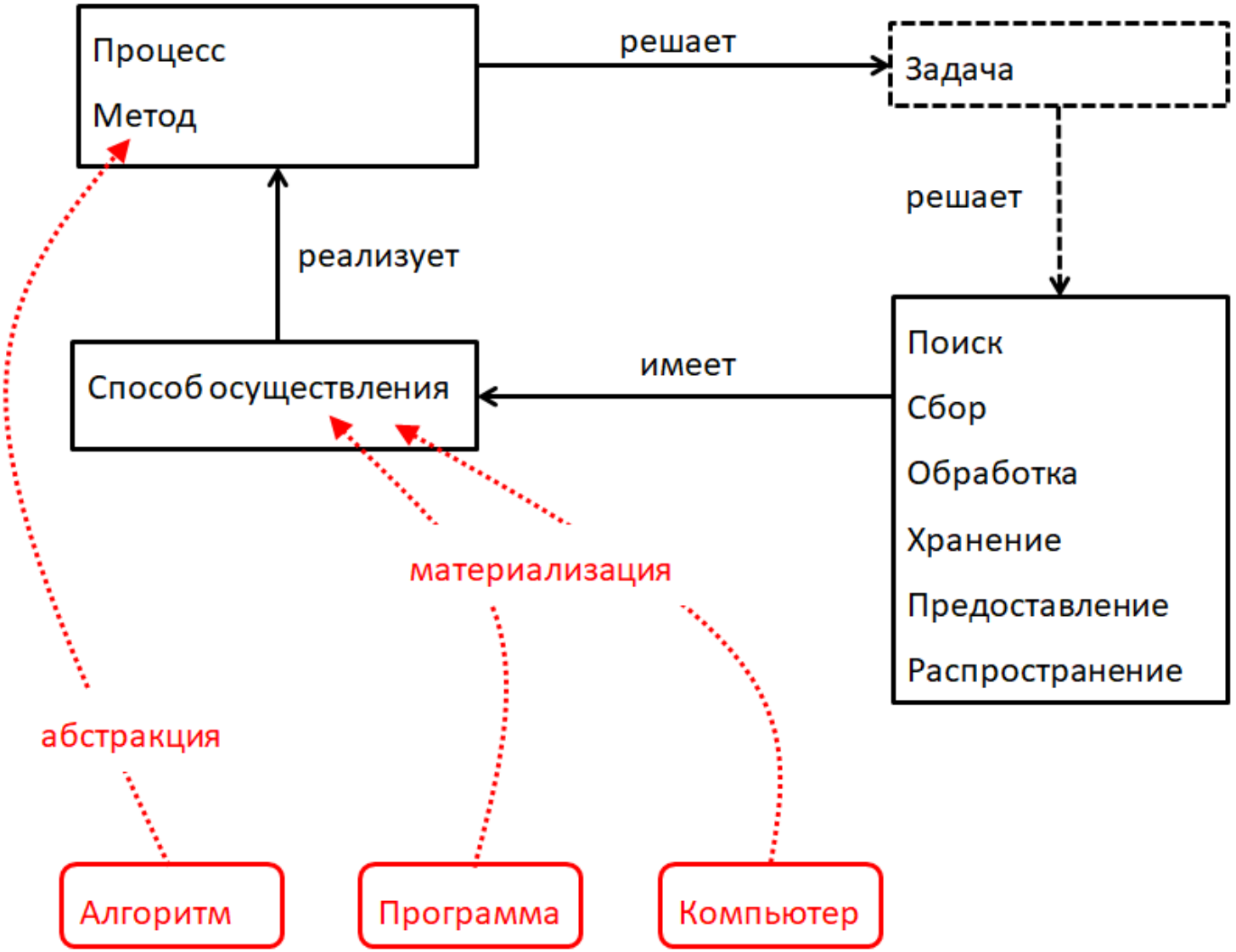
Научные знания могут быть:
- **эмпирическими** (на основе опыта или наблюдения)
- **теоретическими** (на основе анализа **абстрактных моделей**)

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

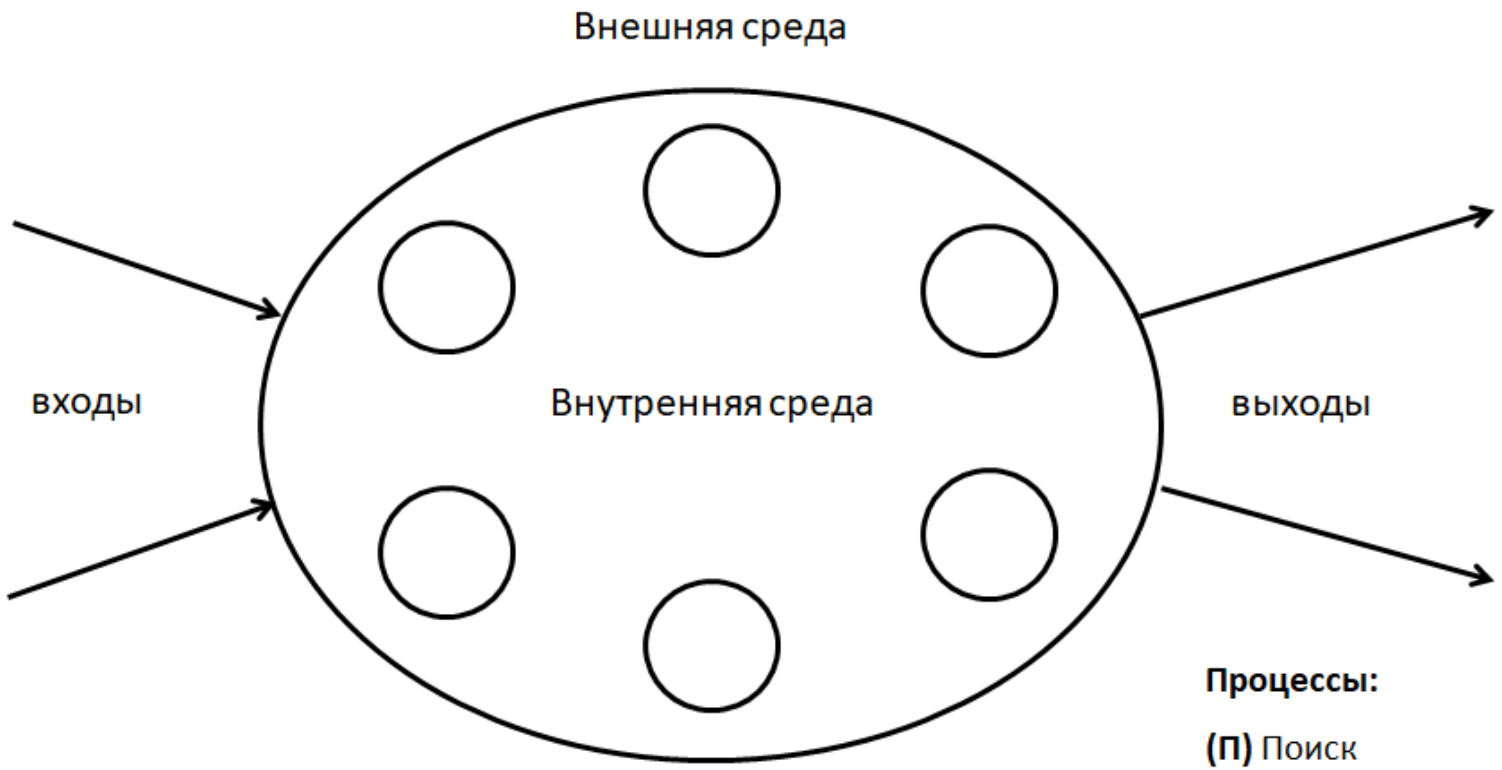
Теоретические знания — абстракции, аналогии, схемы, отображающие структуру и природу процессов изменения объектов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для **прогнозирования** поведения объектов

ГОСТ 7.0-99 *ГОСТ 34.321-96*
Знание

Объектно-ориентированный подход (инженерия знаний)



Объектно-ориентированный подход (инженерия знаний)



Процессы:

(П) Поиск

(С) Сбор

(О) Обработка

(Х) Хранение

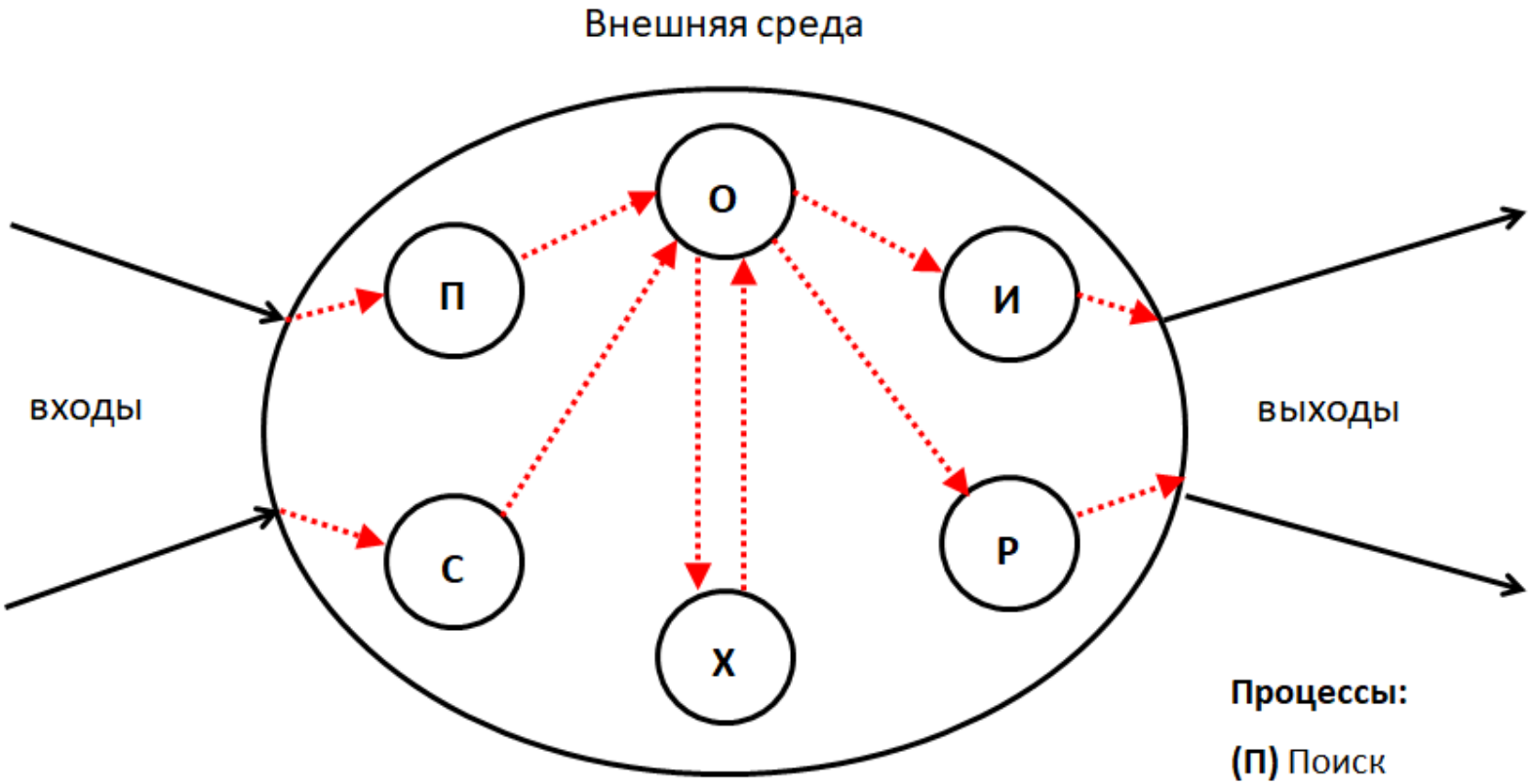
(И) Предоставление

(Р) Распространение

Задача:

1. *Подписать все процессы внутренней среды*
2. *Соединить все процессы стрелками, избегая противоречий*

Объектно-ориентированный подход (инженерия знаний)



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*

Процессы:

(П) Поиск

(С) Сбор

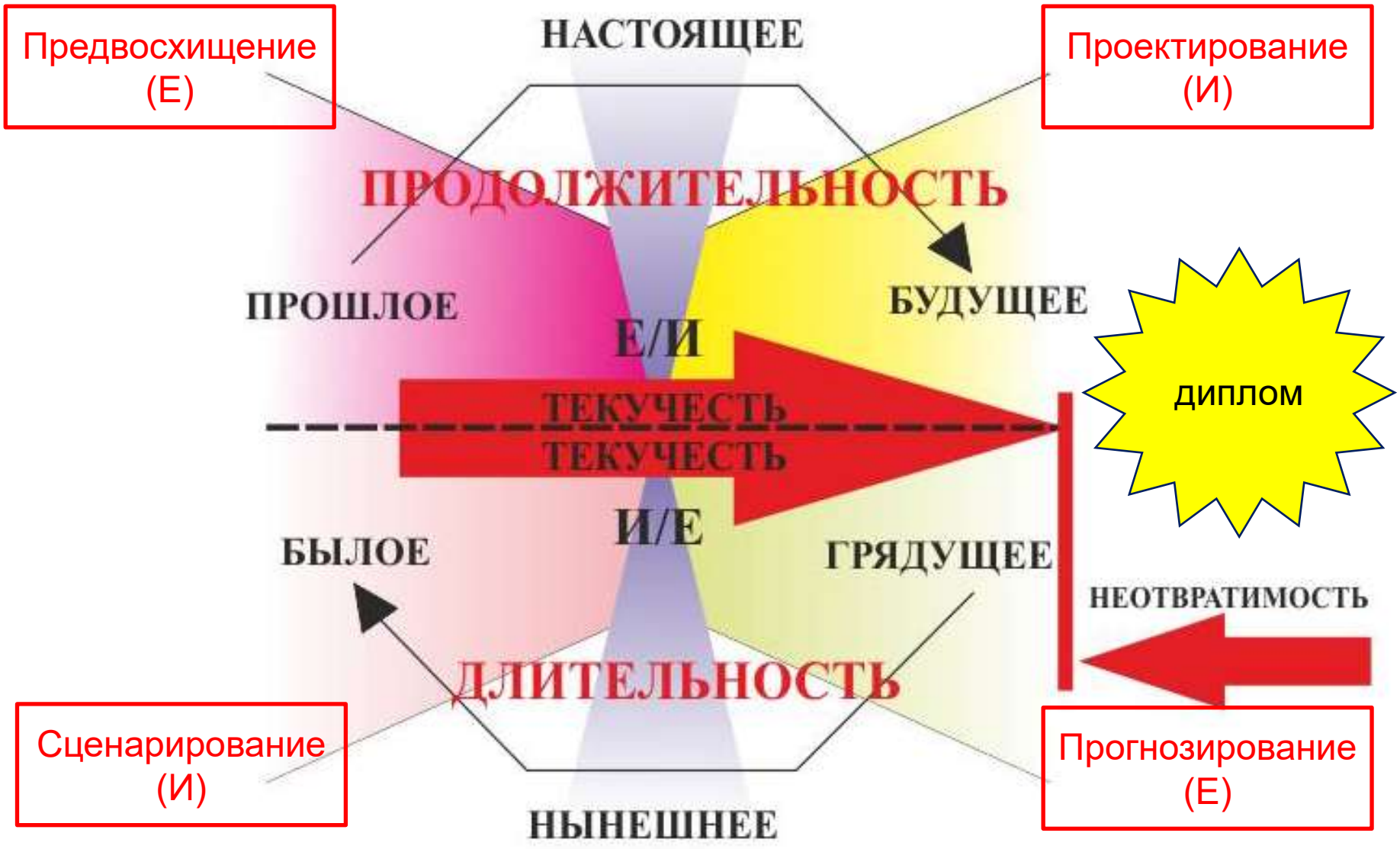
(О) Обработка

(Х) Хранение

(И) Предоставление

(Р) Распространение

Объектно-ориентированный подход (инженерия знаний)



Объектно-ориентированный подход (инженерия знаний)

Фрейм (англ. frame — «каркас» или «рамка») — способ представления знаний в искусственном интеллекте, представляющий собой схему действий в реальной ситуации

Первоначально термин «фрейм» ввёл Марвин Минский в 70-е годы XX века для обозначения структуры знаний для восприятия пространственных сцен.

Фрейм — это модель **абстрактного образа**, минимально возможное описание сущности какого-либо:

- **объекта**
- **явления**
- **события**
- **ситуации**
- **процесса**

Фрейм

Объектно-ориентированный анализ (OOA) и проектирование (OOD)

Объектно-ориентированный анализ — это методология, при которой требования к системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области

Объектно-ориентированное проектирование — это методология проектирования, соединяющая в себе процесс **объектной декомпозиции** и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы

Объектно-ориентированное проектирование:

1) основывается на объектно-ориентированной декомпозиции
2) использует многообразие приемов представления моделей, отражающих:
   логическую (классы и объекты) и
   физическую (модули и процессы) структуру системы,

а также ее

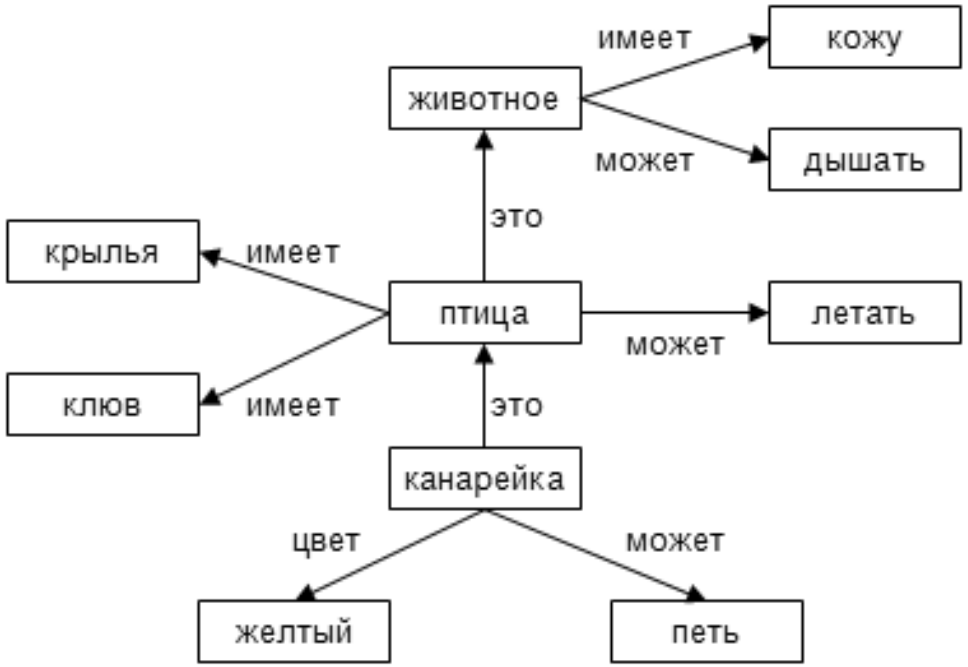
статические и
динамические аспекты

Объектно-ориентированный подход (инженерия знаний)

Семанти́ческая сеть — информационная модель предметной области, имеющая вид ориентированного графа, вершины которого соответствуют объектам предметной области, а дуги (рёбра) задают отношения между ними. Объектами могут быть понятия, события, свойства, процессы. Таким образом, семантическая сеть является одним из способов представления знаний.

Семантическую сеть может образовывать система связанных фреймов, при этом различают:

- фреймы-образцы
- фреймы-экземпляры
- фреймы-структуры
- фреймы-роли
- фреймы-сценарии
- фреймы-ситуации



Семантическая сеть

Минский М. Фреймы для представления знаний. М.: Мир, 1979.

Объектно-ориентированный подход (инженерия знаний)

Базовыми видами отношений для представления фреймов в виде семантической сети являются:

• отношение **AKO** (англ. a kind of), являющееся отношением **классификации** и позволяющее строить иерархические связи между объектами, реализующие основные принципы **наследования** свойств объектов

общее <- (AKO) – частное (только абстракции)

• отношение **HasPart** (англ. has part), являющееся отношением **вхождения** и позволяющее **декомпозировать** сложные объекты на их составляющие

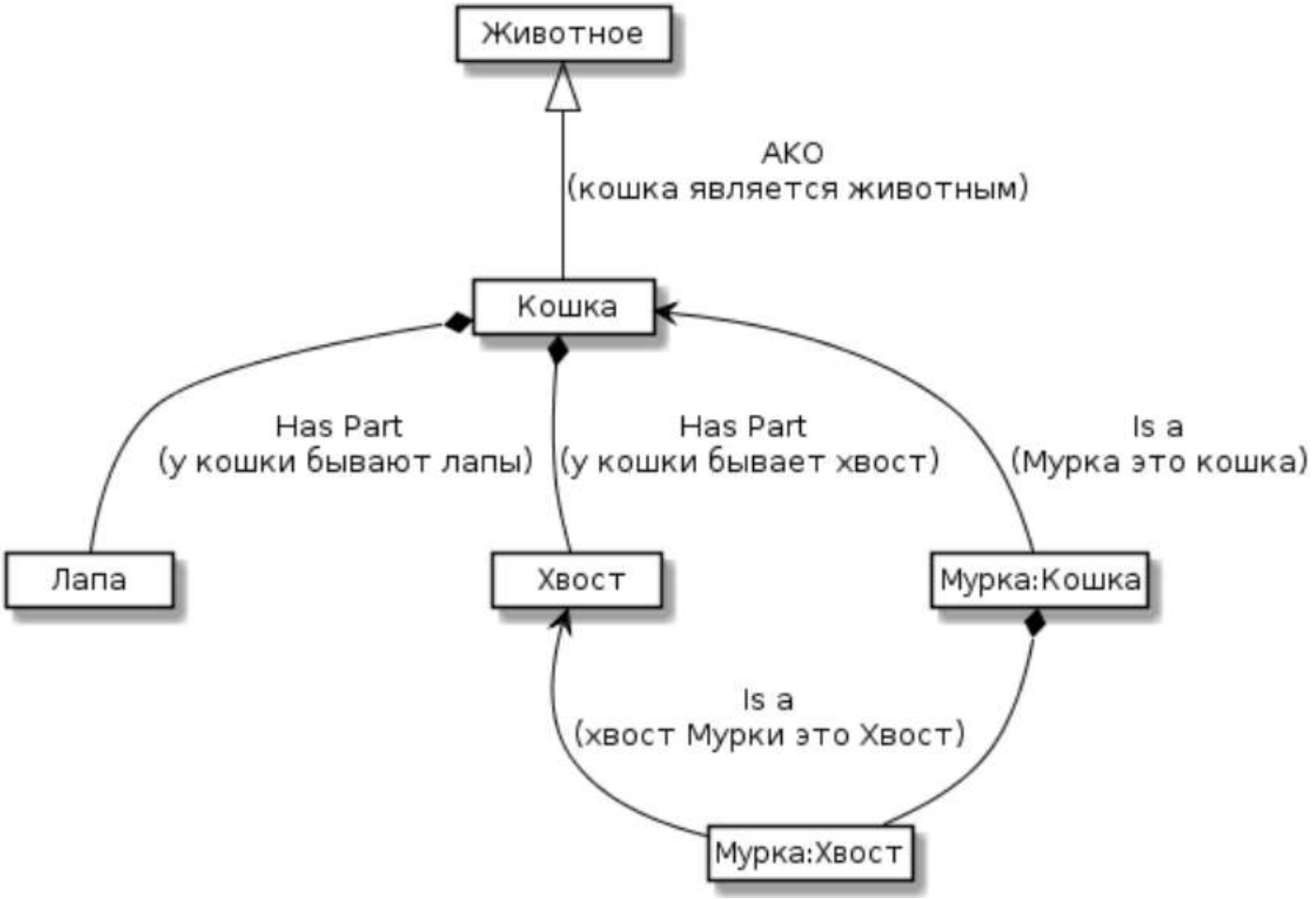
целое <- (HasPart) – часть (только абстракции)

• отношение **ISA** (англ. is a), являющееся отношением между фреймом-**образцом** и фреймом-**экземпляром** , появляющимся **в результате классификации** (отнесения конкретного объекта к абстрактному классу)

абстрактное <- (ISA) – конкретное

Семантическая сеть Минский М. Фреймы для представления знаний. М.: Мир, 1979.

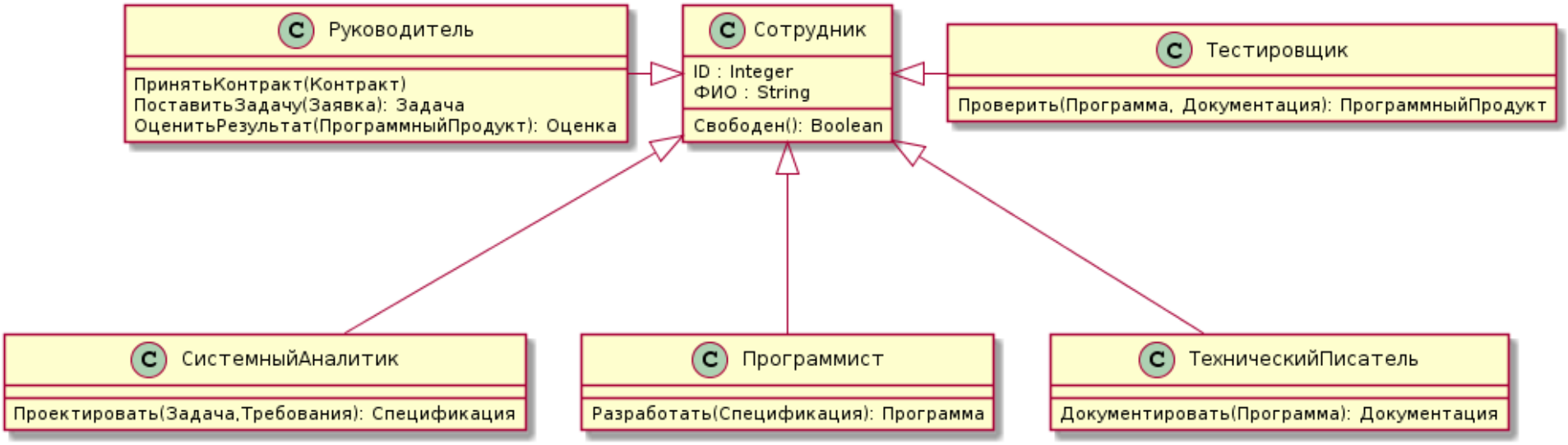
Объектно-ориентированный подход (UML)



Семантическая сеть

Минский М. Фреймы для представления знаний. М.: Мир, 1979.

Объектно-ориентированный подход (UML)



[2017](#) [2018](#)

Объектно-ориентированное программирование

О системах программирования, основанных на фреймах, говорят, что они являются *объектно-ориентированными*

Каждый **фрейм** соответствует некоторому **объекту** предметной области, а **слоты содержат** описывающие этот объект **данные**, то есть в слотах находятся значения признаков объектов. Фрейм может быть представлен в виде **списка свойств**, а если использовать средства базы данных, то в виде **записи**

Более узкое определение: язык программирования является **объектно-ориентированным** тогда и только тогда, когда выполняются следующие условия:

- **поддерживаются объекты**, то есть абстракции данных, имеющие интерфейс в виде именованных операций и собственные данные, с ограничением доступа к ним
- **объекты** относятся к соответствующим **типам (классам)**
- типы (классы) могут **наследовать** атрибуты супертипов (суперклассов)

Фрейм

ООП: языки разметки

HTML (HyperText Markup Language)

```
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title>HTML Document</title>
   </head>
   <body>
      <p>
         <b>
            Этот текст будет полужирным,
            <i>а этот — ещё и курсивным</i>
         </b>
      </p>
   </body>
</html>
```

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>

<p><a href="http://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>
```

ООП: языки разметки

HTML5 (HyperText Markup Language 5)

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>
         (Это title) Пример страницы на HTML5
      </title>
   </head>
   <body>
      <header>
         <hgroup>
            <h1>
               Заголовок "h1" из hgroup
            </h1>
            <h2>
               Заголовок "h2" из hgroup
            </h2>
         </hgroup>
      </header>
      <nav>
         <menu>
            <li>
```

```
<video poster="movie.jpg" controls>
        <source src='movie.webm' type='video/webm; codecs="vp8.0, vorbis"'/>
        <source src='movie.ogv' type='video/ogg; codecs="theora, vorbis"'/>
        <source src='movie.mp4' type='video/mp4; codecs="avc1.4D401E, mp4a.40.2"'/>
        <p>This is fallback content</p>
</video>
```

ООП: управление отображением

CSS (Cascading Style Sheets, каскадные таблицы стилей)

```css
p.Big {
   font-family: arial, helvetica, sans-serif;
   color: maroon;
}
div#First {
   background-color: silver;
}
```

```css
p {
   font-family: arial, helvetica, sans-serif;
}
h2 {
   font-size: 20pt;
   color: red;
   background: white;
}
.note {
   color: red;
   background-color: yellow;
   font-weight: bold;
}
p#paragraph1 {
   padding-left: 10px;
}
a:hover {
   text-decoration: none;
}
#news p {
   color: blue;
}
[type="button"] {
   background-color: green;
}
```

ООП: функции клиентской части

JavaScript (JS)

```
window.onload = function() {
    var linkWithAlert = document.getElementById("alertLink");
    linkWithAlert.onclick = function() {
        return confirm('Вы уверены?');
    };
};
```

```
var fruits, text, fLen, i;

fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "<ul>";
for (i = 0; i < fLen; i++) {
    text += "<li>" + fruits[i] + "</li>";
}
```

```
var obj;
obj = document.getElementById("demo");
obj.innerHTML = "Hello";
```

```
xhttp.open("GET", "ajax_info.txt", false);
xhttp.send();
document.getElementById("demo").innerHTML = xhttp.responseText;
```

```
<script>
window.onload = downScripts;

function downScripts() {
    var element = document.createElement("script");
    element.src = "myScript.js";
    document.body.appendChild(element);
}
</script>
```

ООП: Управление поведением

JQuery

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js">
</script>
<script src="my_jquery_functions.js"></script>
</head>
```

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("[href]").hide();
});
</script>
</head>
<body>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<a href="http://www.w3schools.com/html/">HTML Tutorial</a>
<a href="http://www.w3schools.com/css/">CSS Tutorial</a>

</body>
```

```
$("button").click(function(){
    $("h1, h2, p").addClass("blue");
    $("div").addClass("important");
});
```

```
$("button").click(function(){
    $.get("demo_test.asp", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

ООП: определение данных

XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="country">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="country_name" type="xs:string"/>
        <xs:element name="population" type="xs:decimal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
   <start>
      <element name="book">
         <oneOrMore>
            <ref name="page"/>
         </oneOrMore>
      </element>
   </start>
   <define name="page">
      <element name="page">
         <text/>
      </element>
   </define>
</grammar>
```

XSD

ООП: определение данных

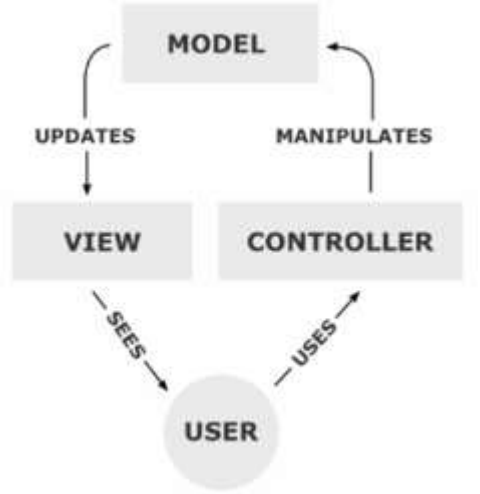
XML

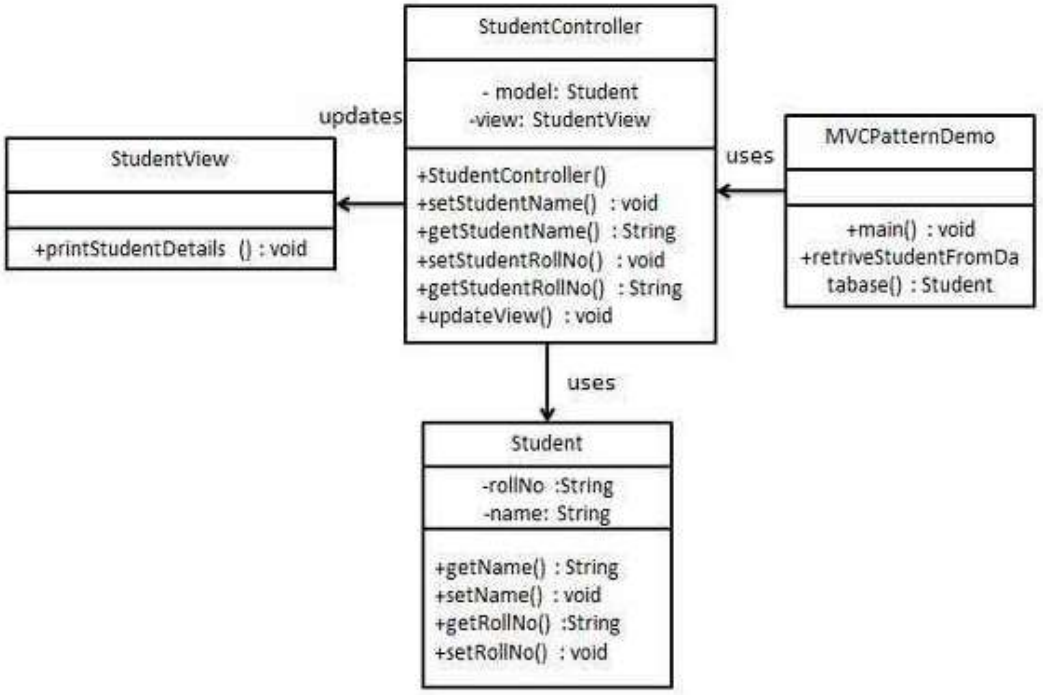
```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE recipe>
<recipe name="хлеб" preptime="5min" cooktime="180min">
   <title>
      Простой хлеб
   </title>
   <composition>
      <ingredient amount="3" unit="стакан">Мука</ingredient>
      <ingredient amount="0.25" unit="грамм">Дрожжи</ingredient>
      <ingredient amount="1.5" unit="стакан">Тёплая вода</ingredient>
   </composition>
   <instructions>
     <step>
        Смешать все ингредиенты и тщательно замесить.
     </step>
     <step>
        Закрыть тканью и оставить на один час в тёплом помещении.
     </step>
```

```xml
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example text.</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

ООП: фреймворк

Model-view-controller (MVC, «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, *пользовательский интерфейс* и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данная схема проектирования часто используется для построения *архитектурного каркаса*, когда переходят от теории к реализации в конкретной предметной области





https://ru.wikipedia.org/wiki/Model-View-Controller

Лекция 2
«**История развития итеративной разработки программных средств**»

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

06.02.2019

История: RUP

Rational Unified Process (RUP) — методология разработки программного обеспечения, созданная компанией Rational Software (1996-2003)

В основе RUP лежат следующие **принципы**:

- Ранняя идентификация и непрерывное (до окончания проекта) устранение **основных рисков**
- Концентрация на выполнении **требований заказчиков** к исполняемой программе - анализ и построение **модели прецедентов (вариантов использования**)
- Ожидание **изменений в требованиях**, проектных решениях и реализации в процессе разработки
- Компонентная архитектура, реализуемая и **тестируемая на ранних стадиях** проекта
- Постоянное **обеспечение качества** на всех этапах разработки проекта (продукта)
- Работа над проектом в сплочённой команде, **ключевая роль** в которой принадлежит **архитекторам**

RUP (Википедия)

История: борьба за качество

ГОСТ Р ИСО 9000-2015 Системы менеджмента качества. Основные положения и словарь

3.6.2 **качество (quality):** Степень соответствия совокупности присущих *характеристик* (3.10.1) *объекта* (3.6.1) *требованиям*

3.6.4 **требование (requirement):** Потребность или ожидание, которое установлено, обычно предполагается или является обязательным.

Примечания
1 Слова "обычно предполагается" означают, что это общепринятая практика *организации* и *заинтересованных сторон* , что рассматриваемые потребности или ожидания предполагаются.
2 Установленным является такое требование, которое определено, например, в *документированной информации*

3.6.1 **объект (object), сущность (entity), элемент (item):** Что-либо воспринимаемое или воображаемое.

Примечание - Объекты могут быть материальными (например, двигатель, лист бумаги, алмаз), нематериальными (например, коэффициент конверсии, план проекта) или воображаемыми (например, будущее положение организации).

ГОСТ Р ИСО 9000-2015

История: борьба за качество

ГОСТ Р ИСО/МЭК 25010-2015 Информационные технологии (ИТ). Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов

ГОСТ Р ИСО/МЭК 25010-2015

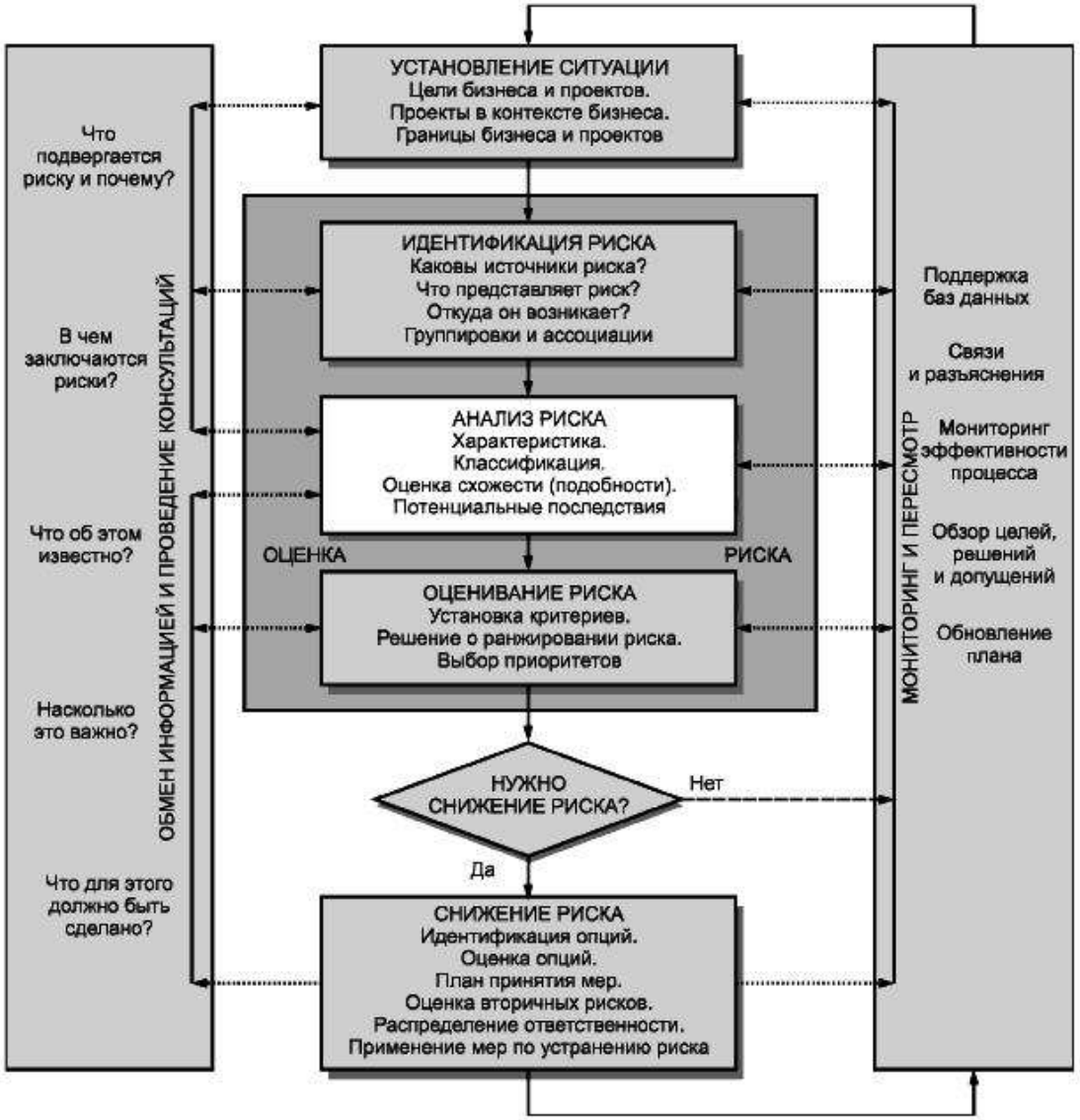
История: борьба за качество

ГОСТ Р ИСО/МЭК 25010-2015 Информационные технологии (ИТ). Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов

ГОСТ Р ИСО/МЭК 25010-2015

История: управление рисками

ГОСТ Р 52806-2007 Менеджмент рисков проектов. Общие положения



ТЕСТ

[ГОСТ Р 52806-2007]

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:

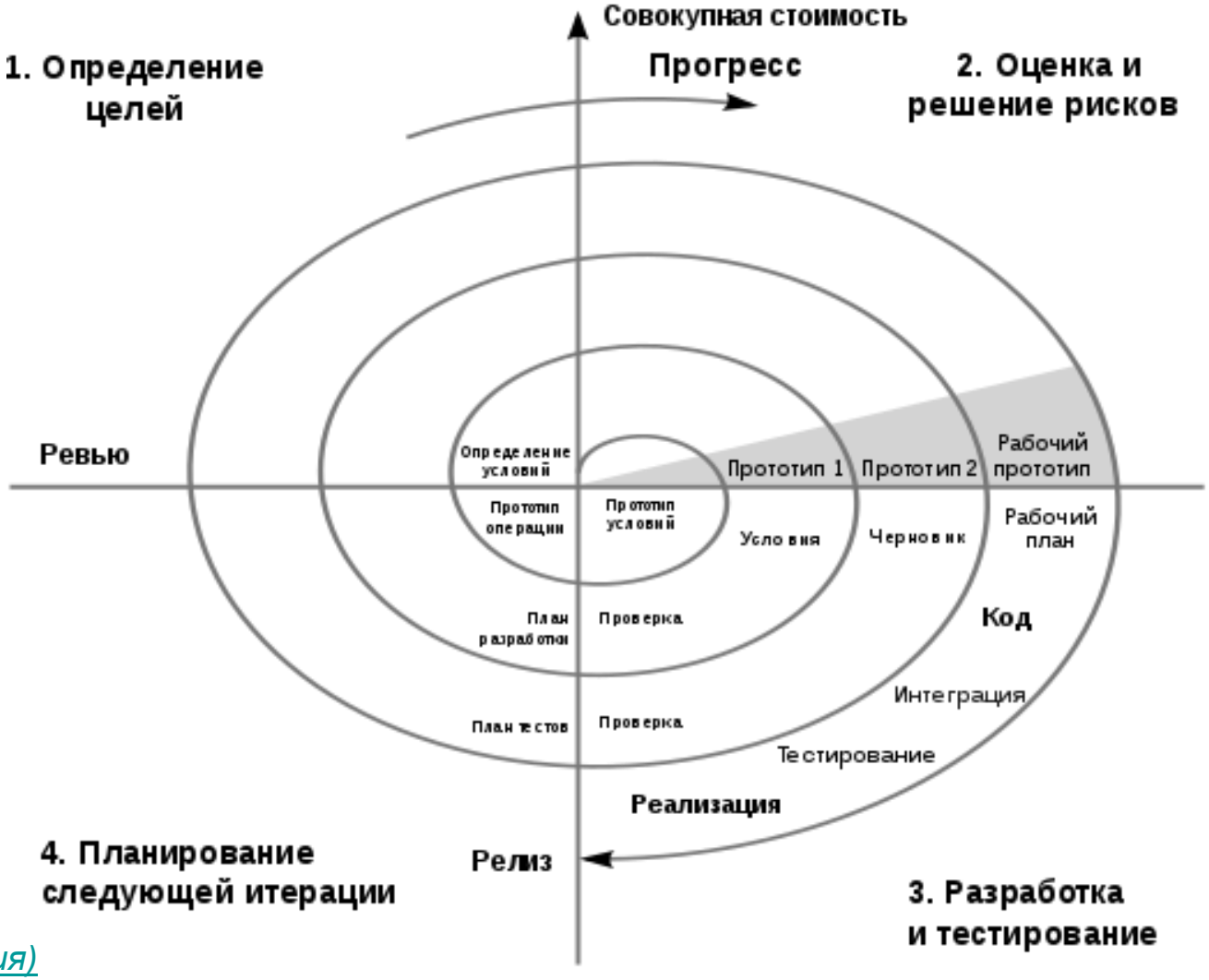
- быстро реагировать на меняющиеся требования,
- обнаруживать и устранять риски на ранних стадиях проекта, а также
- эффективно контролировать качество создаваемого продукта

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

RUP (Википедия)

RUP: итеративная модель

Спира́льная модель, предложенная Барри Боэмом в 1986 году, представляет собой процесс разработки программного обеспечения, сочетающий в себе как итеративность, так и этапность.



RUP (Википедия)

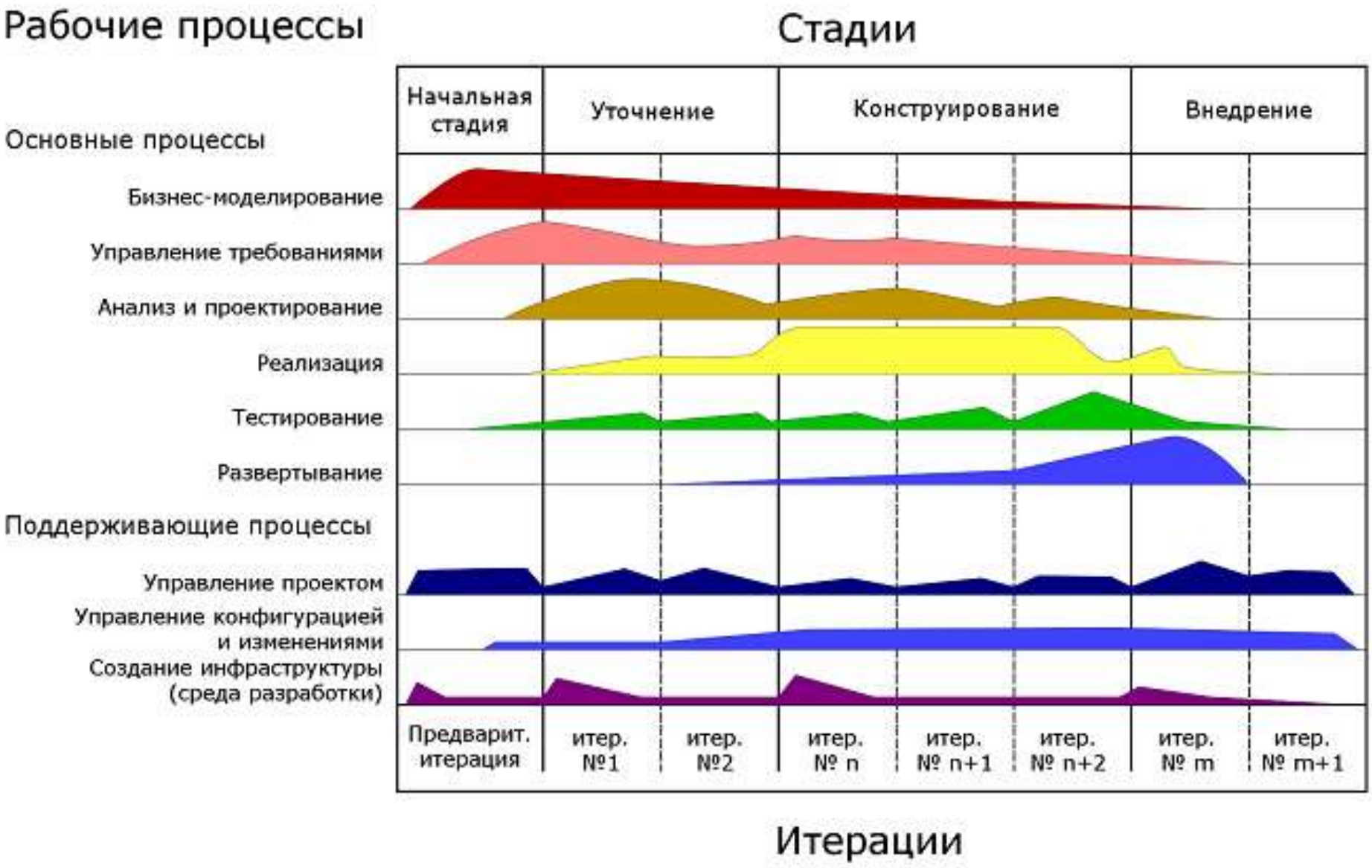
RUP: итеративная модель

Боэм формулирует десять **наиболее распространённых (по приоритетам) рисков**:

1. Дефицит специалистов
2. Нереалистичные сроки и бюджет
3. Реализация несоответствующей функциональности
4. Разработка неправильного пользовательского интерфейса
5. «Золотая сервировка», перфекционизм, ненужная оптимизация и оттачивание деталей
6. Непрекращающийся поток изменений
7. Нехватка информации о внешних компонентах, определяющих окружение системы или вовлечённых в интеграцию
8. Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами
9. Недостаточная производительность получаемой системы
10. Разрыв между квалификацией специалистов и требованиями проекта

RUP (Википедия)

RUP: фазы



[RUP (Википедия)]

RUP: фазы

1. **Начальная стадия (Inception)**:

- Формируются видение и границы проекта
- Создается экономическое обоснование (business case)
- Определяются основные требования, ограничения и ключевая функциональность продукта
- Создается базовая версия модели прецедентов
- Оцениваются риски

After these are completed, the project is checked against the following criteria:

- Stakeholder concurrence on scope definition and cost/schedule estimates.
- Requirements understanding as evidenced by the fidelity of the primary use cases.
- Credibility of the cost/schedule estimates, priorities, risks, and development process.
- Depth and breadth of any **architectural prototype** that was developed.
- Establishing a baseline by which to compare actual expenditures versus planned expenditure

RUP (Википедия)

RUP: фазы

2. Уточнение (Elaboration) - производится анализ предметной области и построение исполняемой архитектуры. Это включает в себя:

- Документирование требований (включая детальное описание для большинства прецедентов)
- Спроектированную, реализованную и оттестированную исполняемую архитектуру
- Обновленное экономическое обоснование и более точные оценки сроков и стоимости
- Сниженные основные риски

The outcome of the elaboration phase is:

- A use-case model in which the use-cases and the actors have been identified and most of the use-case descriptions are developed. The use-case model should be 80% complete.
- A description of the software architecture in a software system development process.
- An executable architecture that realizes architecturally significant use cases.
- Business case and risk list which are revised.
- A development plan for the overall project.
- **Prototypes** that demonstrably mitigate each identified technical risk.
- A preliminary user manual (optional)

RUP (Википедия)

RUP: фазы

3. Построение (Construction) - происходит реализация большей части функциональности продукта.

Фаза завершается первым внешним релизом системы и вехой начальной функциональной готовности (Initial Operational Capability)

4. Внедрение (Transition) - создается финальная версия продукта и передается от разработчика к заказчику.
Это включает в себя:

- программу бета-тестирования,
- обучение пользователей, а также
- определение качества продукта

В случае, если качество не соответствует ожиданиям пользователей или критериям, установленным в фазе Начало, фаза Внедрение повторяется снова

Выполнение всех целей означает достижение вехи готового продукта (Product Release) и завершение полного цикла разработки

RUP (Википедия)

RUP: строительные блоки

RUP is based on a set of building blocks and content elements, describing:

- **what** is to be produced,
- the necessary **skills** required and
- the **step-by-step** explanation describing how specific development goals are to be achieved.

The main building blocks, or content elements, are the following:

- **Roles (who)** – A role defines a set of related skills, competencies and responsibilities
- **Work products (what)** – A work product represents something resulting from a task, including all the documents and models produced while working through the process
- **Tasks (how)** – A task describes a unit of work assigned to a Role that provides a meaningful result

RUP (Wikipedia)

RUP: строительные блоки

Within each iteration, the tasks are categorized into nine **disciplines**:

Six engineering disciplines

1. Business modelling
2. Requirements
3. Analysis and design
4. Implementation
5. Test
6. Deployment

Three supporting disciplines

1. Configuration and change management
2. Project management
3. Environment

RUP (Wikipedia)

RUP: прецеденты

Для отражения модели прецедентов на диаграмме используются:

- **рамки** системы (англ. *system boundary*) — **прямоугольник** с названием в верхней части и эллипсами (прецедентами) внутри. Часто может быть опущен без потери полезной информации
- **актёр** (англ. *actor*) — стилизованный **человечек**, обозначающий набор ролей пользователя (понимается в широком смысле:
  - человек
  - внешняя сущность
  - класс
  - другая система

  взаимодействующего с некоторой сущностью (системой, подсистемой, классом)

  Актёры не могут быть связаны друг с другом (за исключением отношений обобщения/наследования)
- **прецедент** — **эллипс** с надписью, обозначающий выполняемые системой действия (могут включать возможные варианты), приводящие к наблюдаемым актёрами результатам.

RUP: прецеденты

Сценарий использования, вариант использования, **прецедент использования** (англ. *use case*) — в разработке программного обеспечения и системном проектировании это описание поведения системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды

Другими словами, сценарий использования описывает, «кто» и «что» может сделать с рассматриваемой системой, или что система может сделать с «кем» или «чем»

Методика сценариев использования применяется для выявления требований к поведению системы, известных также как **пользовательские и функциональные требования**

Пользовательские истории (англ. *User Story*) — способ описания **требований** к разрабатываемой системе, сформулированных как одно или более предложений на повседневном или деловом языке пользователя

Пользовательские истории используются гибкими методологиями разработки программного обеспечения для спецификации требований (вместе с приёмочными испытаниями

Прецеденты: IDEF0 → Use Case

Описание решения

Общее решение состоит в следующей ассоциации элементов диаграммы IDEF0 с элементами диаграммы прецедентов:

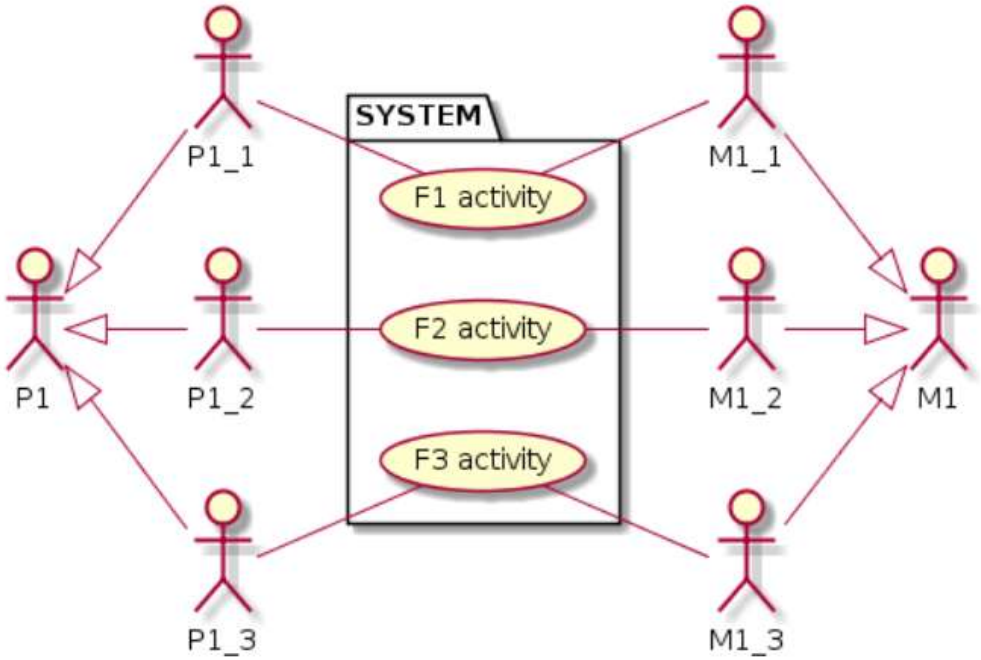
- стрелки механизмов преобразуются в "actor";
- декомпозируемые механизмы становятся родительскими "actor";
- имена блоков становятся именами прецедентов;
- все блоки дочерней диаграммы объединяются в один пакет с именем родительской.

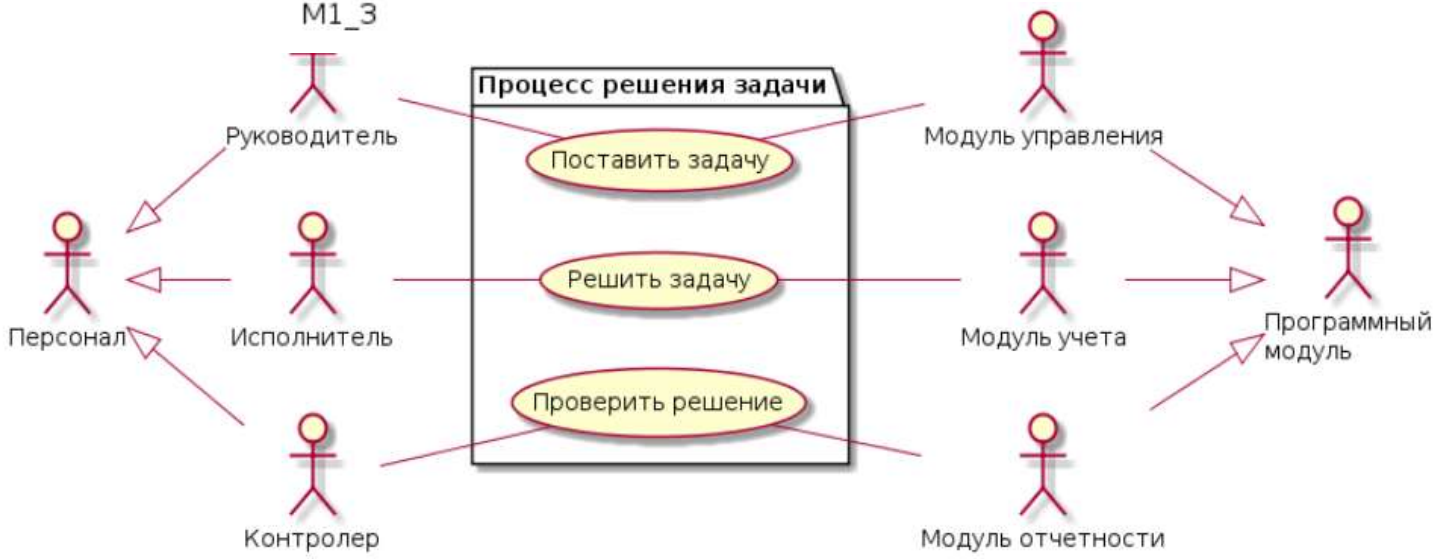
Особенности преобразования

При преобразовании диаграмм IDEF0 в диаграммы прецедентов UML теряется информация обо всех информационных и материальных потоках - о входах, выходах и управлении.

При обратном преобразовании все эти потоки должны быть восстановлены или спроектированы заново.

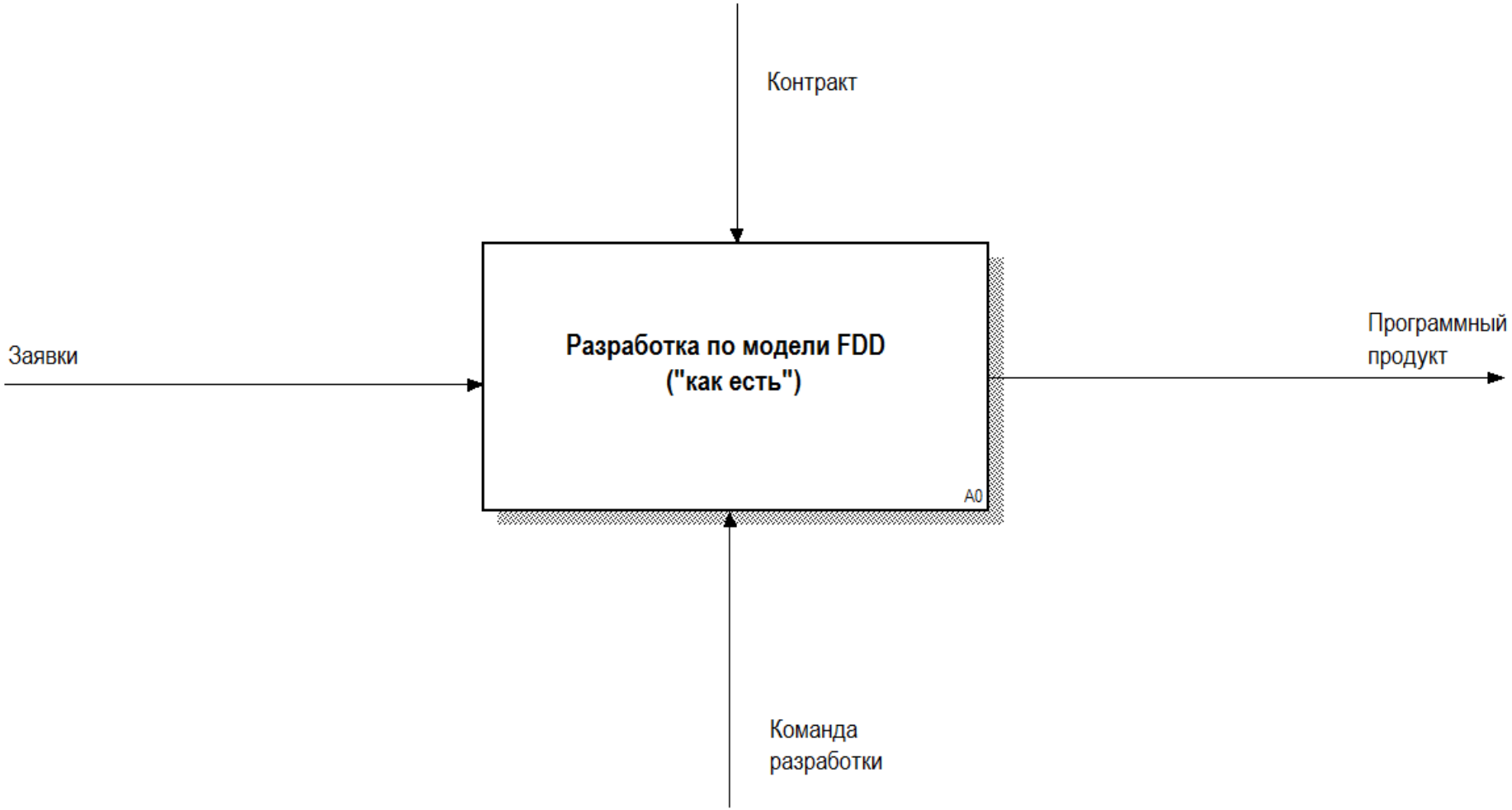
Прецеденты: IDEF0 → Use Case





[Преобразование IDEF0 в Use Case]

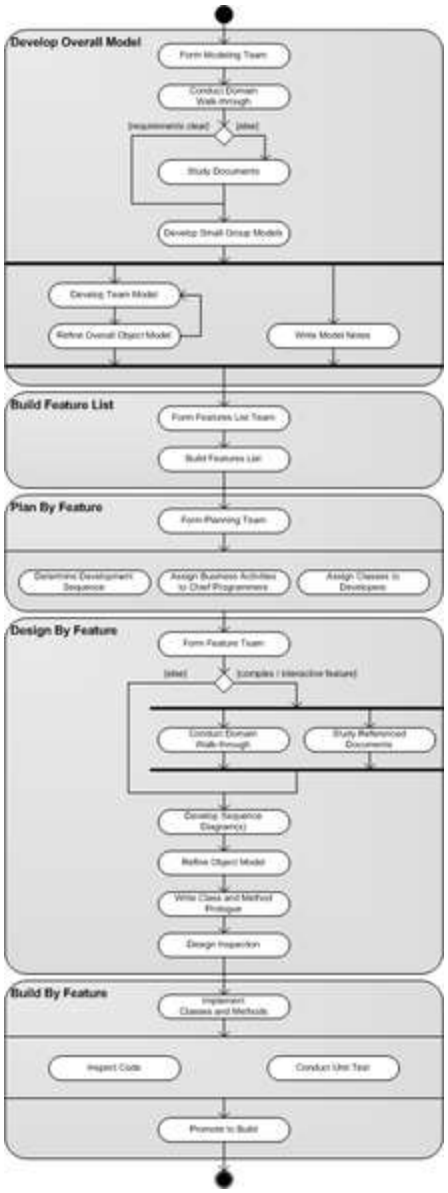
Гибкая модель разработки: FDD



Гибкая разработка: модель FDD

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

FDD представляет собой попытку объединить наиболее признанные в индустрии разработки программного обеспечения методики, принимающие за основу важную для заказчика функциональность (свойства) разрабатываемого программного обеспечения. Основной целью данной методологии является разработка реального, работающего программного обеспечения систематически, в поставленные сроки.

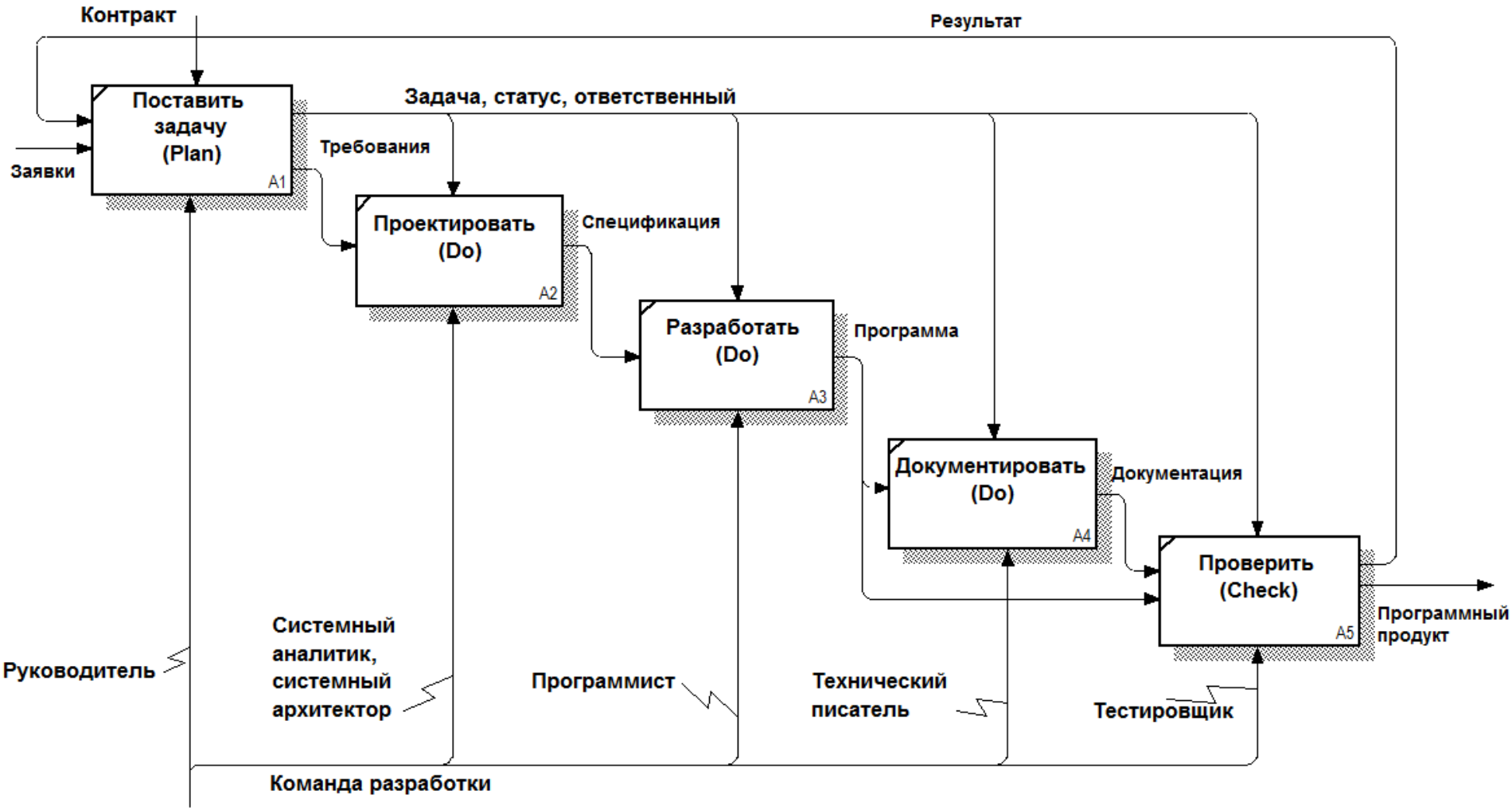


FDD включает в себя пять базовых видов деятельности:

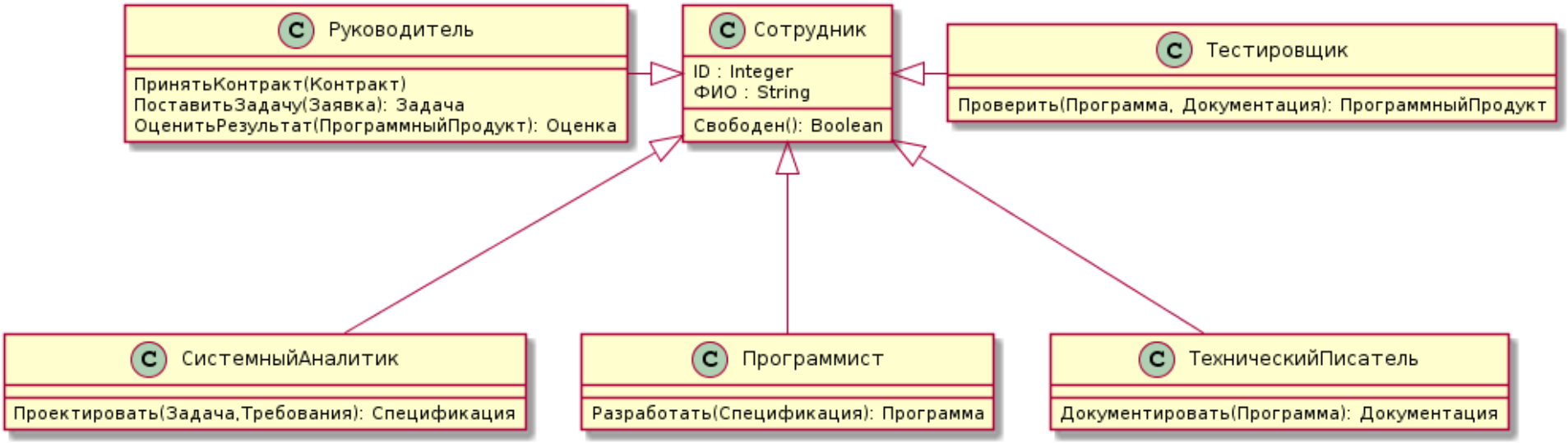
1. разработка **общей модели**
2. составление **списка необходимых функций** системы
3. **планирование** работы над каждой функцией
4. **проектирование** функции
5. **реализация** функции

https://ru.wikipedia.org/wiki/Feature_driven_development
http://www.intuit.ru/studies/courses/3505/747/lecture/26289?page=1#sect3
http://www.intuit.ru/studies/courses/2188/174/lecture/4730?page=2

Модель FDD и команда

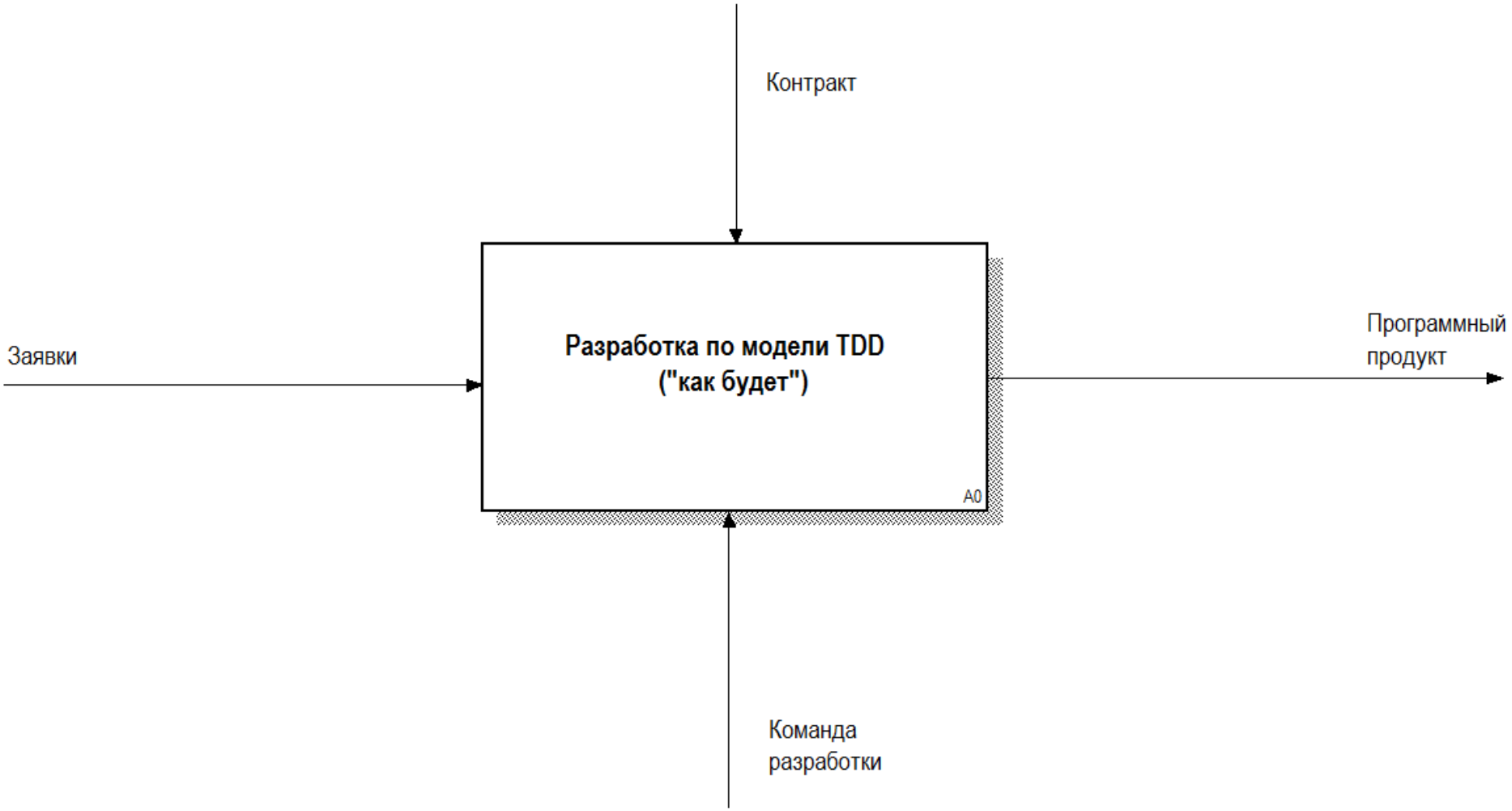


Модель FDD и команда



http://www.intuit.ru/studies/courses/3467/709/lecture/21825
http://www.intuit.ru/studies/courses/3467/709/lecture/21825?page=2

Гибкая разработка: от FDD к TDD



Гибкая разработка: модель TDD

Разработка через тестирование (*test-driven development, **TDD***) — техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода к соответствующим стандартам.

TDD цикл включает в себя пять основных шагов:

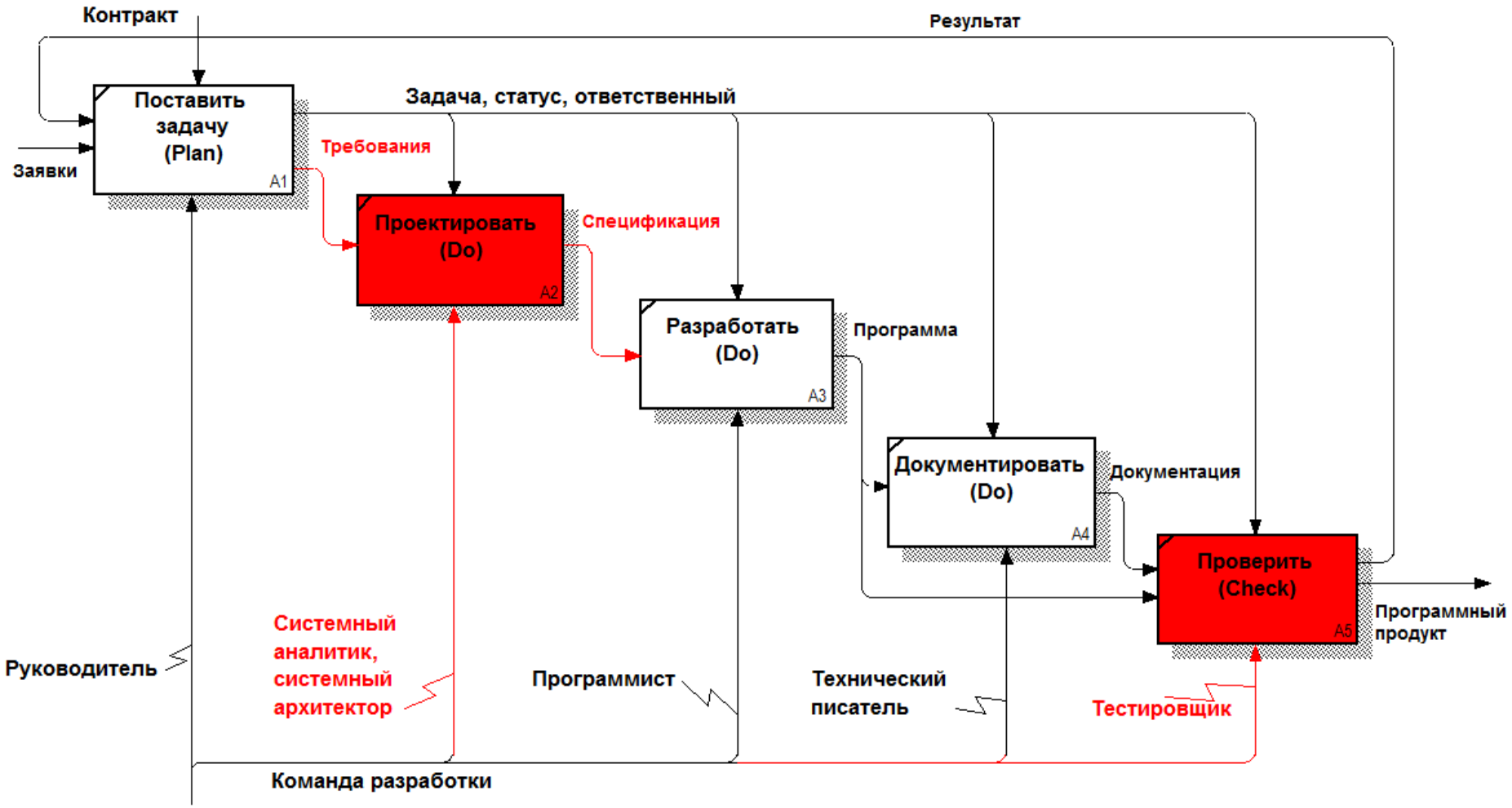
1. Быстро **добавить тест**
2. Выполнить все тесты и увидеть, что **новый тест "падает"**
3. Выполнить **небольшое изменение** системы
4. Убедиться, что **все тесты проходят**
5. Выполнить **рефакторинг** , удаляя дублирование

В модели TDD тест всегда **пишется прежде чем создается** соответствующий программный элемент

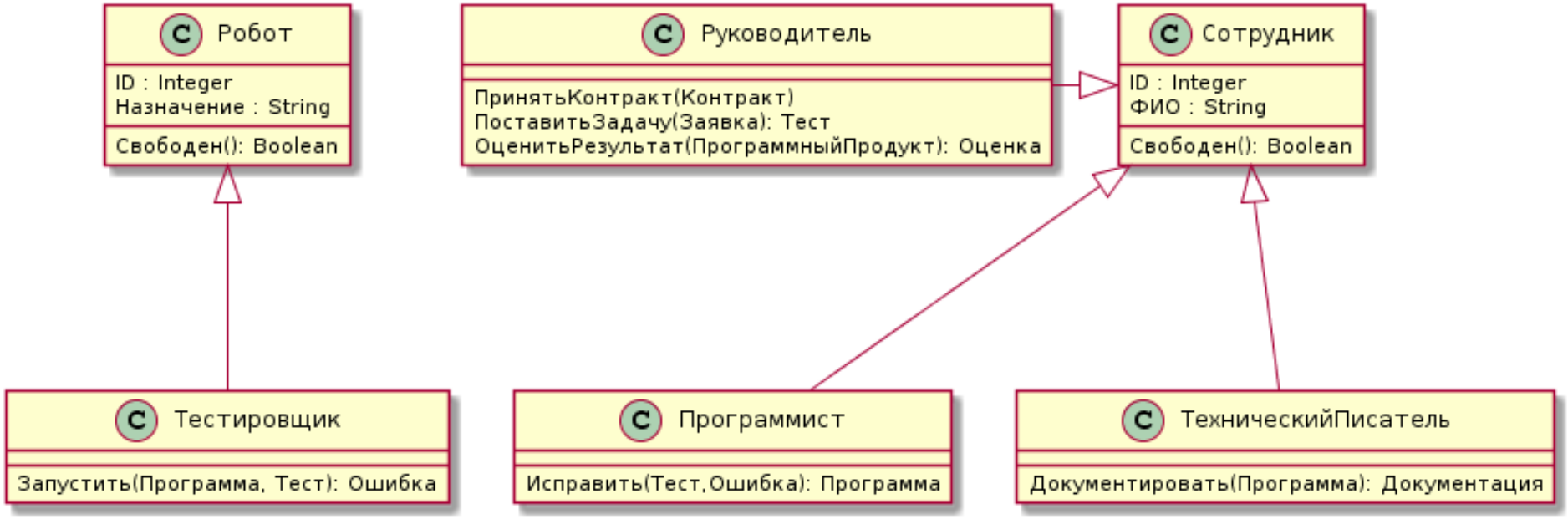
Если далее не выполнять шаги 2, 4, 5 то получится модель **TFD** (разработка "вначале тест«, test first development)

https://en.wikipedia.org/wiki/Test-driven_development
http://www.intuit.ru/studies/courses/3505/747/lecture/26301?page=2

Модель TDD и команда

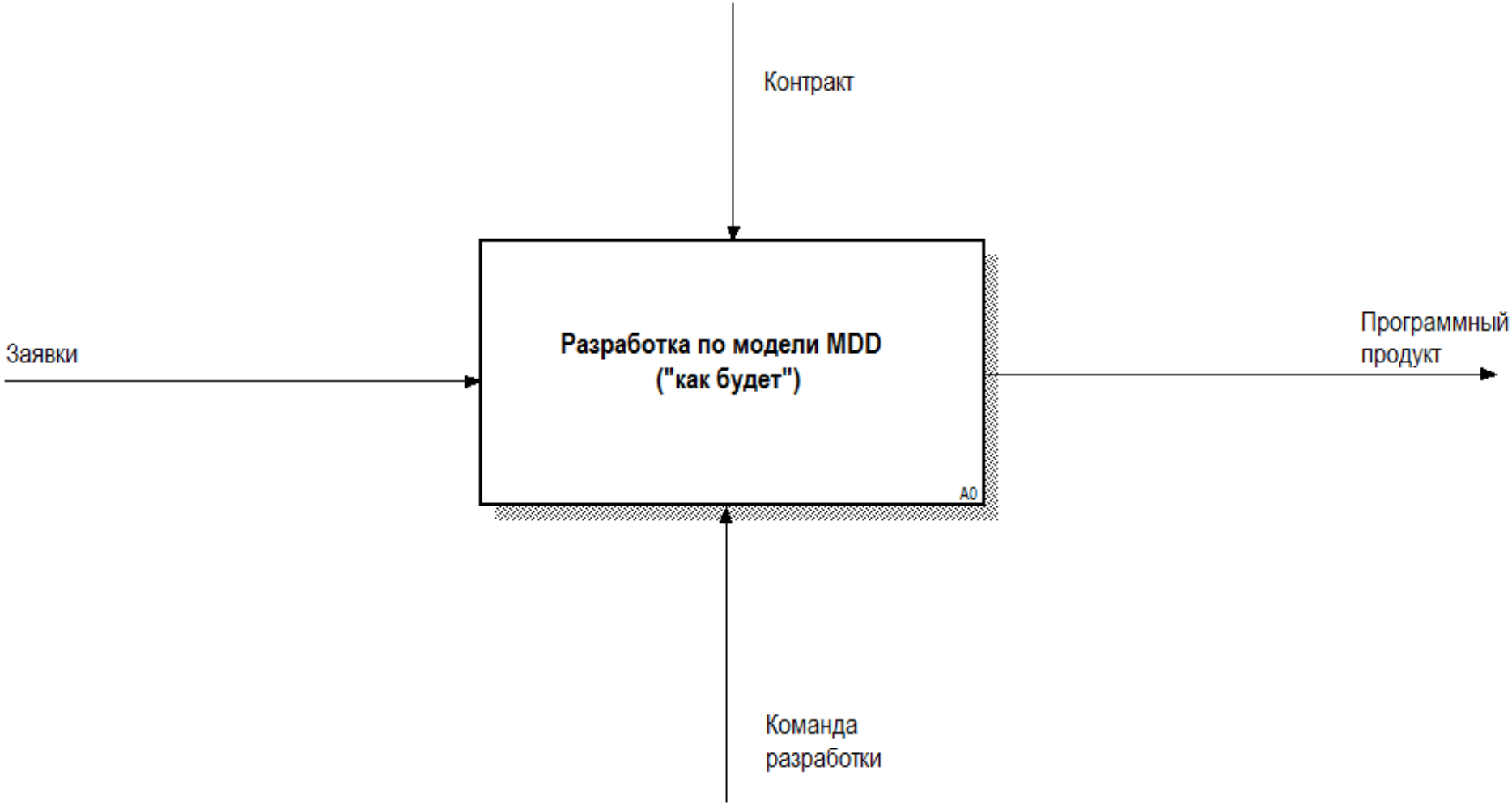


Модель TDD и команда



http://www.intuit.ru/studies/courses/3467/709/lecture/21825
http://www.intuit.ru/studies/courses/3467/709/lecture/21825?page=2

Модель MDD


Контракт
Заявки
Разработка по модели MDD
("как будет")
A0
Программный
продукт
Команда
разработки

Гибкая разработка: модель MDD

Разработка, управляемая моделями (*model-driven development,* **MDD***, Model-driven engineering*, **MDE**) — это стиль разработки программного обеспечения, когда модели становятся основными артефактами разработки, из которых генерируется код и другие артефакты.

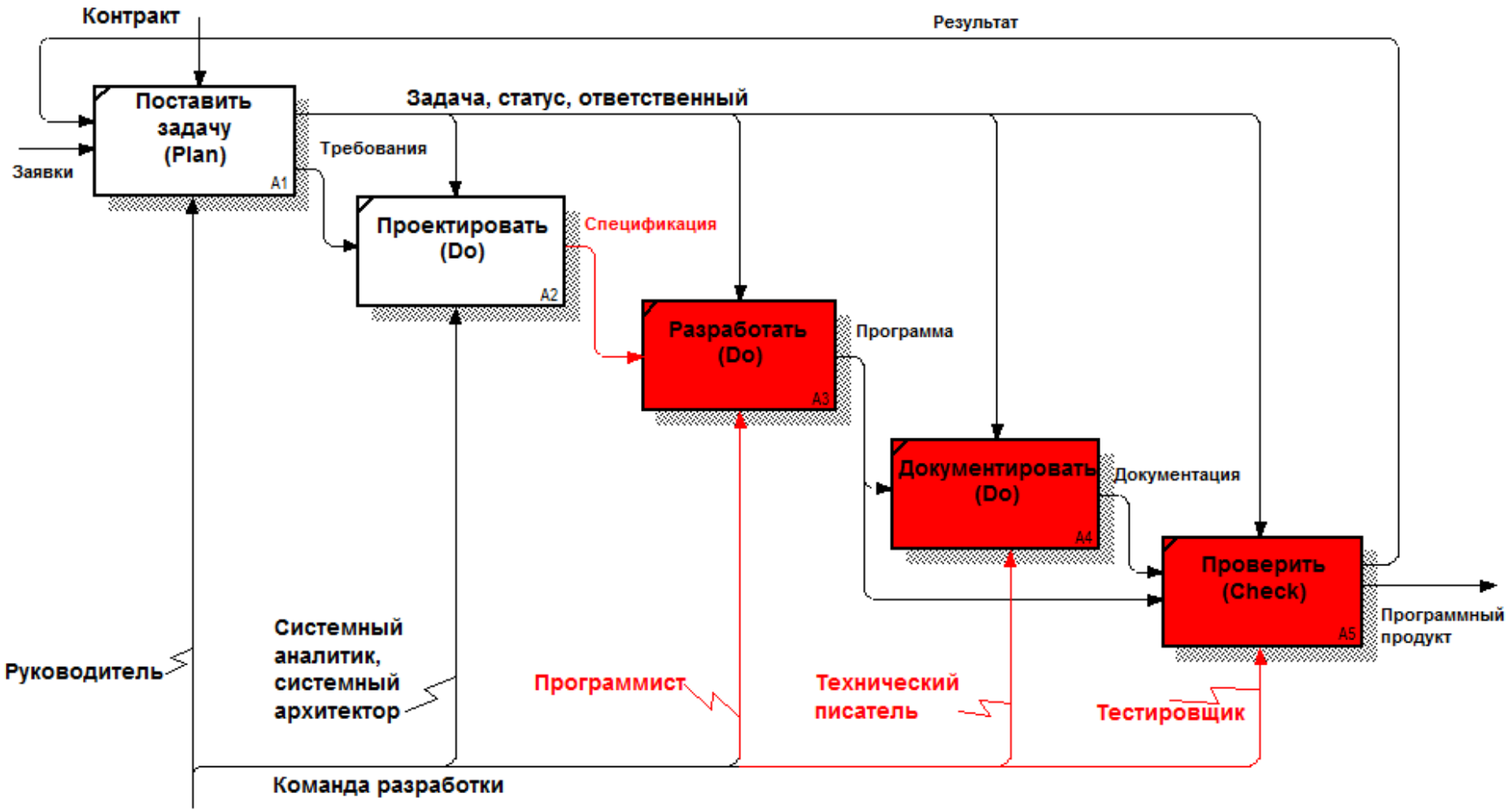
Модель — это абстрактное описание программного обеспечения, которое скрывает информацию о некоторых аспектах с целью представления упрощенного описания остальных. Модель может быть исходным артефактом в разработке, если она фиксирует информацию в форме, пригодной для интерпретаций людьми и обработки инструментальными средствами. Модель определяет **нотацию** и **метамодель**. Нотация представляет собой совокупность графических элементов, которые применяются в модели и могут быть интерпретированы людьми. Метамодель описывает используемые в модели понятия и фиксирует информацию в виде метаданных, которые могут быть обработаны инструментальными средствами.

Наиболее известными современными MDE-инициативами являются:

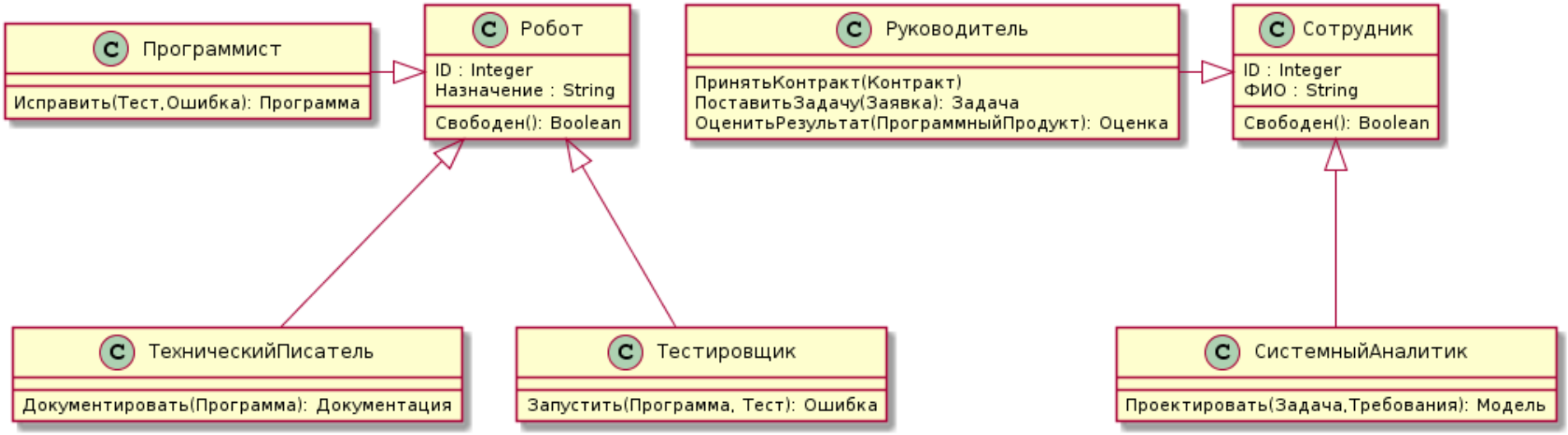
1. разработка Object Management Group (OMG) под названием model-driven architecture (MDA)
2. экосистема Eclipse для инструментов моделирования и программирования (Eclipse Modeling Framework)

https://en.wikipedia.org/wiki/Model-driven_engineering
http://1c-dn.com/1c_enterprise/getting_started/

Модель MDD и команда



Модель MDD и команда



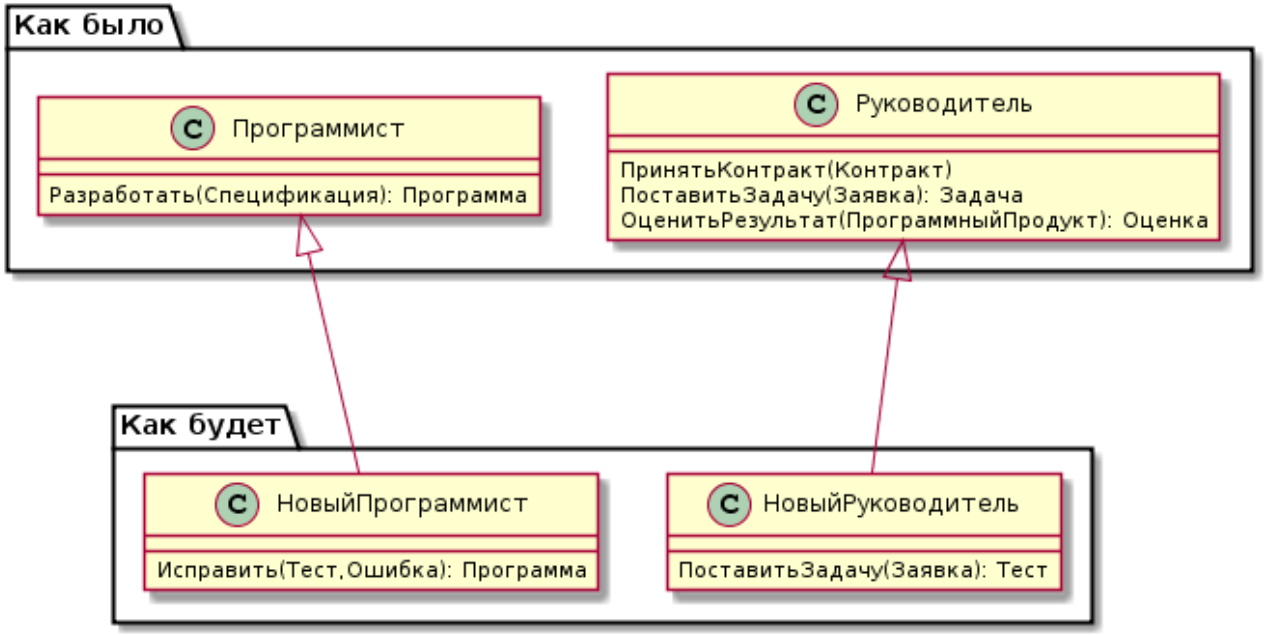
http://www.intuit.ru/studies/courses/3467/709/lecture/21825
http://www.intuit.ru/studies/courses/3467/709/lecture/21825?page=2

Планирование организационных инноваций

Федеральный закон "О науке и государственной научно-технической политике" от 23.08.1996 N 127-ФЗ

Инновации - введенный в употребление новый или значительно улучшенный продукт (товар, услуга) или процесс, новый метод продаж или новый организационный метод в деловой практике, организации рабочих мест или во внешних связях.

Путем сравнения моделей «как есть» (As Is) и «как должно быть» (To Be) составляется модель «что сделать» (To Do).



http://www.consultant.ru/document/cons_doc_LAW_11507/

Планирование организационных инноваций

Федеральный закон "О науке и государственной научно-технической политике" от 23.08.1996 N 127-ФЗ

Инновационный проект - комплекс направленных на достижение экономического эффекта мероприятий по осуществлению инноваций, в том числе по коммерциализации научных и (или) научно-технических результатов.

Инвестиционный проект — экономический или социальный проект, основывающийся на инвестициях; обоснование экономической целесообразности, объёма и сроков осуществления прямых инвестиций в определённый объект, включающее проектно-сметную документацию, разработанную в соответствии с действующими стандартами.

http://www.consultant.ru/document/cons_doc_LAW_11507/
http://www.aup.ru/books/m79/5_3.htm

Инновации: какие они бывают

В соответствии с определением:

- **продуктовые** (пример: электронные пропуска)
- **процессные** (пример: цифровое проектирование)
- **маркетинговые** (пример: аренда лицензий)
- **организационные** (пример: социальные сети в учебном процессе)

Принято рассматривать инновации с позиции «***новое для предприятия***», поскольку именно деятельность конкретных предприятий способна сформировать ***макроинновации*** в отраслях, регионах и целых государствах

Инновации же, в свою очередь, могут состоять из ряда ***микроинноваций***, которые проще всего объяснить как изменения в деятельности отдельного сотрудника предприятия или отдельного потребителя

Инновации: история термина

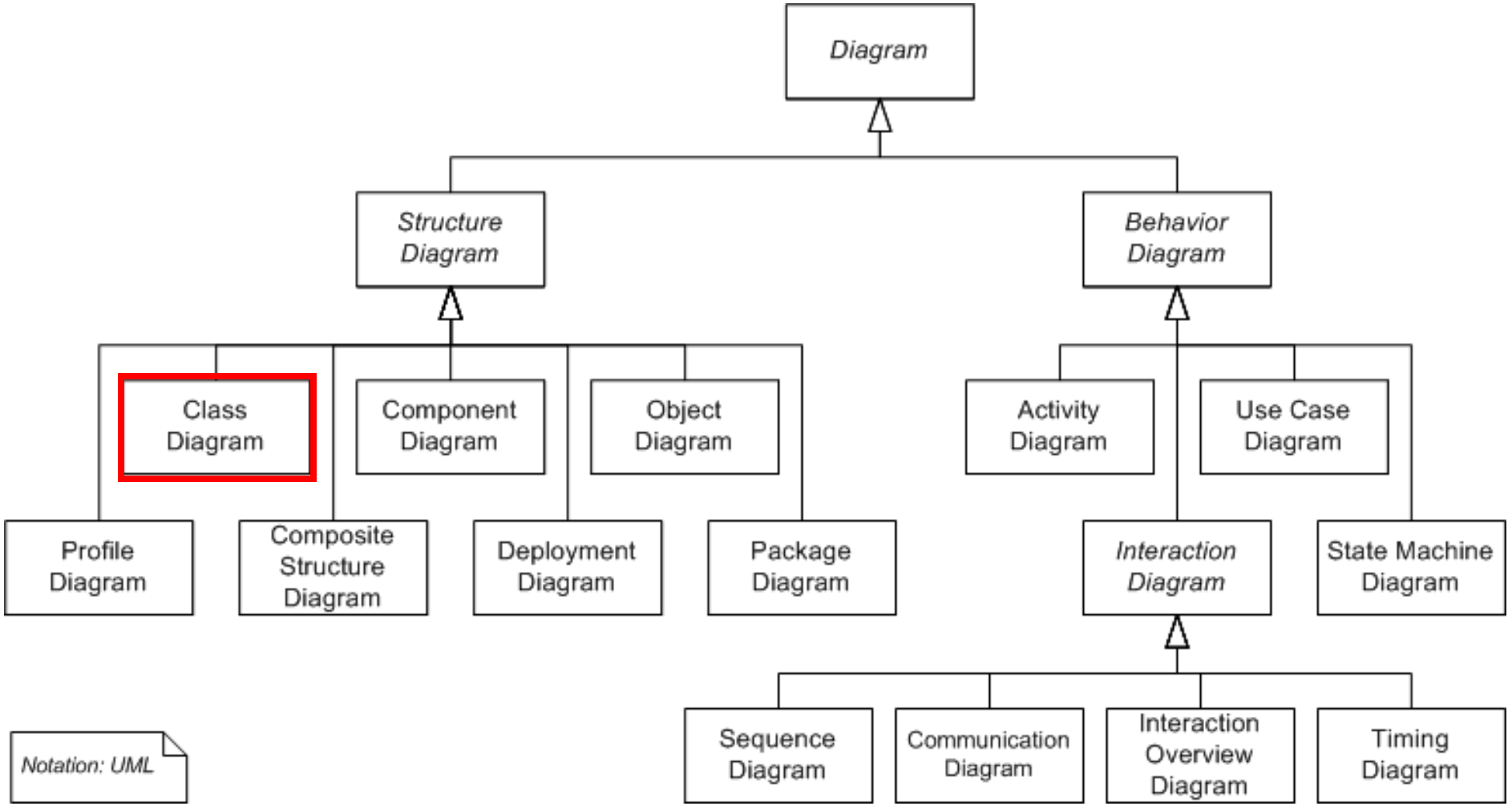
В своей работе «Теория экономического развития» (1911) Йозеф Шумпетер впервые рассмотрел вопросы «новых комбинаций» изменений в развитии и дал полное описание инновационного процесса

Термин «инновация» Йозеф Шумпетер стал использовать в 30-е гг. XX века, понимая при этом под инновацией изменение с целью внедрения и использования новых видов потребительских товаров, новых производственных, транспортных средств, рынков и форм организации в промышленности

Согласно Йозефу Шумпетеру, инновация является главным источником прибыли: «прибыль, по существу, является результатом выполнения новых комбинаций», «без развития нет прибыли, без прибыли нет развития»

Инновации – это ***экономический*** термин!

Прогресс в UML



https://ru.wikipedia.org/wiki/UML
http://www.intuit.ru/studies/courses/1007/229/lecture/5950