

Методологии проектирования программных и информационных средств

Лекция 5

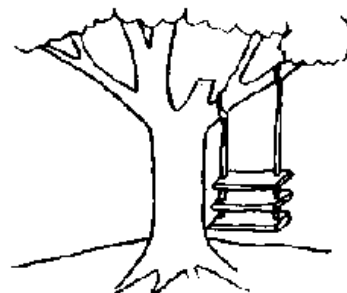
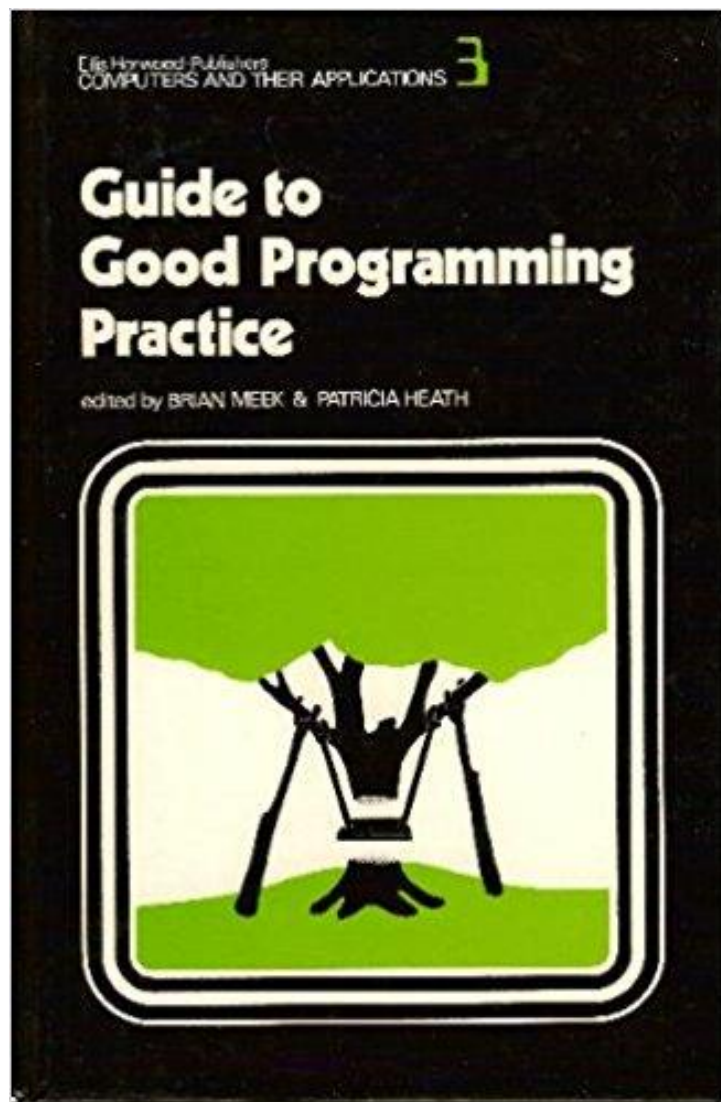
Проектирование и разработка программных средств

Овчинников П.Е.

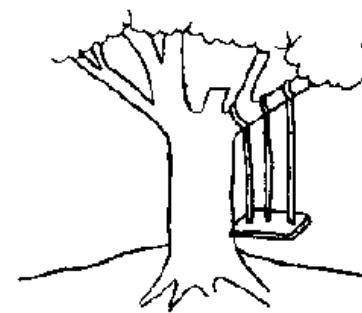
МГТУ «СТАНКИН»,

ст.преподаватель кафедры ИС

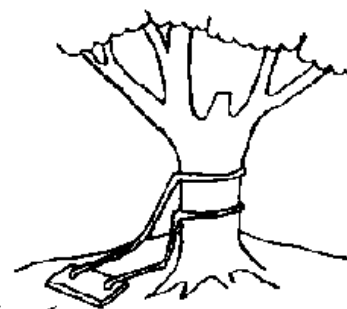
Проблематика



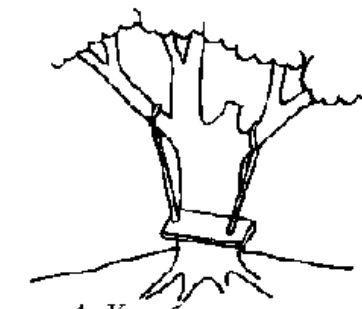
1. Как было предложено организатором разработки



2. Как было описано в техническом задании



3. Как было спроектировано ведущим системным специалистом



4. Как было реализовано программистами

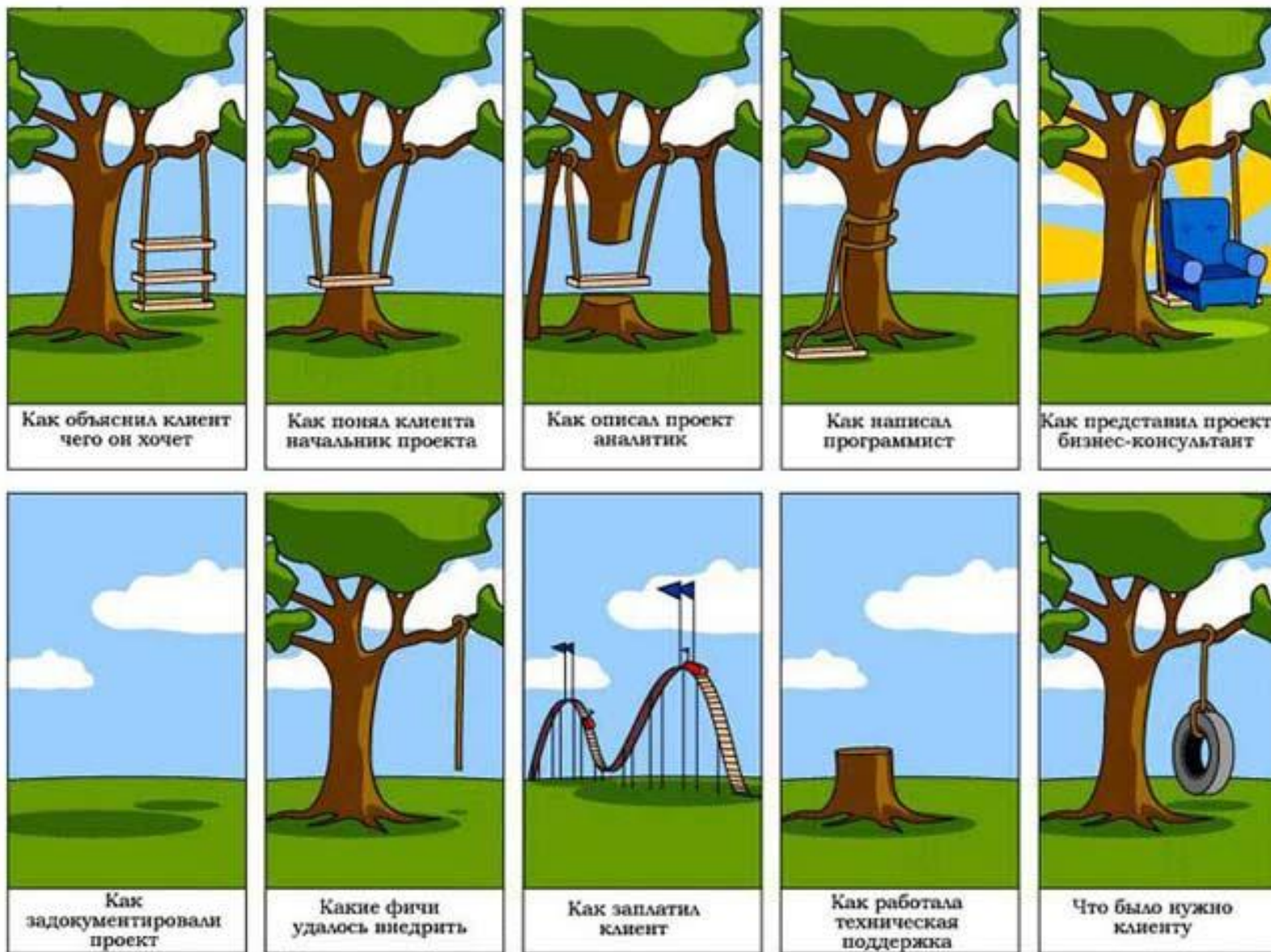


5. Как было внедрено



6. Чего хотел пользователь

Проблематика



Терминология: методология

Метод (от др.-греч. μέθοδος — путь исследования или познания, от μετά- + ὁδός «путь») — систематизированная совокупность шагов, действий, которые нацелены на решение определённой задачи или достижение определённой цели.

В отличие от области знаний или исследований, является авторским, то есть созданным конкретной персоной или группой персон, научной или практической школой. В силу своей ограниченности рамками действия и результата, методы имеют тенденцию устаревать, преобразовываясь в другие методы, развиваясь в соответствии со временем, достижениями технической и научной мысли, потребностями общества. Совокупность однородных методов принято называть подходом. Развитие методов является естественным следствием развития научной мысли.

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата.

В старой трактовке вместо слова «порядок» использовалось слово «последовательность», но по мере развития параллельности в работе компьютеров слово «последовательность» стали заменять более общим словом «порядок».

Независимые инструкции могут выполняться в произвольном порядке, параллельно, если это позволяют используемые исполнители.

Часто в качестве исполнителя выступает компьютер, но понятие алгоритма необязательно относится к компьютерным программам, так, например, чётко описанный рецепт приготовления блюда также является алгоритмом, в таком случае исполнителем является человек (а может быть и некоторый механизм, ткацкий станок, и пр.).

[Метод \(Википедия\)](#)

[Алгоритм \(Википедия\)](#)

Терминология: методология

Методоло́гия (от [греч.](#) μεθοδολογία — учение о [способах](#); от [др.-греч.](#) μέθοδος из [μετά-](#) + ὁδός, букв. «путь вслед за чем-либо» и [др.-греч.](#) λόγος — [мысль](#), [причина](#)) — учение о [методах](#), способах и стратегиях исследования предмета.

В методологии можно выделить следующую структуру:

- основания методологии: [философия](#), [логика](#), [системология](#), [психология](#), [информатика](#), [системный анализ](#), [науковедение](#), [этика](#), [эстетика](#);
- характеристики деятельности: особенности, принципы, условия, нормы деятельности;
- логическая структура деятельности: [субъект](#), [объект](#), [предмет](#), [формы](#), [средства](#), [методы](#), результат деятельности, [решение задач](#);
- временная структура деятельности: фазы, стадии, этапы.
- [технология](#) выполнения работ и решения задач: средства, методы, способы, приемы.

Теория управлénия — [наука](#) о принципах и методах управления различными системами, процессами и объектами.

Теоретической базой теории управления являются [кибернетика](#) и [теория информации](#). Суть теории управления состоит в построении на основе [анализа](#) данной системы, процесса или объекта такой [абстрактной модели](#), который позволит получить [алгоритм](#) управления ими в динамике, — для достижения системой, процессом или объектом состояния, которое требуется целями управления.

[Методология \(Википедия\)](#)

[Теория управления \(Википедия\)](#)

Методология ГОСТ 34

ГОСТ 34.601-90 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания

1. Формирование требований к АС
2. Разработка концепции АС
3. Техническое задание
4. Эскизный проект
5. Технический проект
6. Рабочая документация
7. Ввод в действие
8. Сопровождение АС

РД 50-34.698-90 Методические указания. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов

1.1. Требования к содержанию документов, разрабатываемых при создании АС, установлены настоящими указаниями, а также соответствующими государственными стандартами Единой системы программной документации (ЕСПД), Единой системы конструкторской документации (ЕСКД), Системы проектной документации для строительства (СПДС) и [ГОСТ 34.602](#).

Виды и комплектность документов регламентированы [ГОСТ 34.201](#).

[ГОСТ 34.601-90](#)

[РД 50-34.698-90](#)

Методология ГОСТ Р ИСО/МЭК 12207-2010

ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология (ИТ). Системная и программная инженерия. Процессы жизненного цикла программных средств

5.1.12 Модели и стадии жизненного цикла

Процесс жизни любой системы или программного продукта может быть описан посредством модели жизненного цикла, состоящей из стадий. Модели могут использоваться для представления всего жизненного цикла от замысла до прекращения применения или для представления части жизненного цикла, соответствующей текущему проекту. Модель жизненного цикла представляется в виде последовательности стадий, которые могут перекрываться и (или) повторяться циклически в соответствии с областью применения, размером, сложностью, потребностью в изменениях и возможностях.

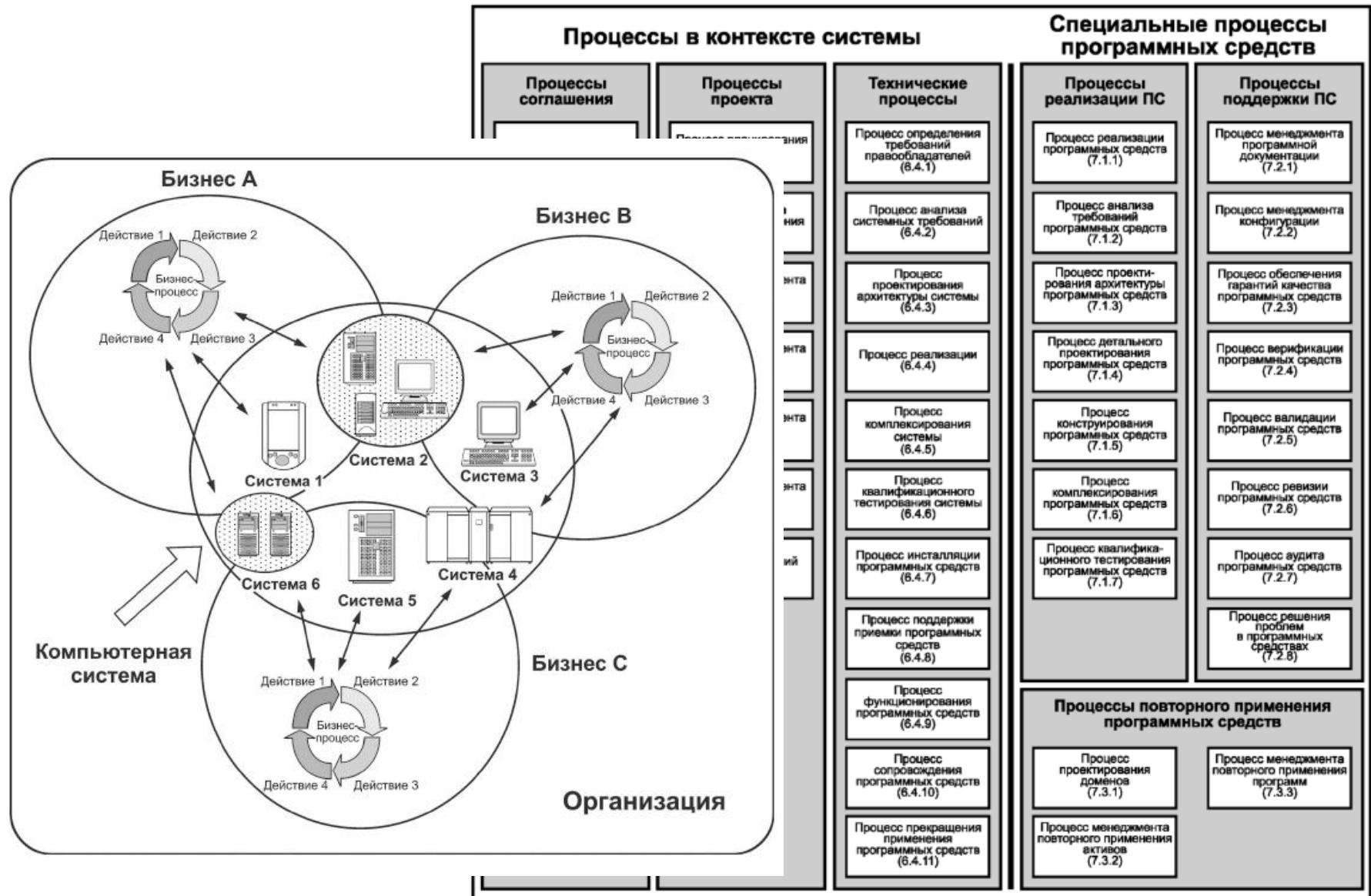
5.2.1 Категории процессов жизненного цикла

Настоящий стандарт группирует различные виды деятельности, которые могут выполняться в течение жизненного цикла программных систем, в семь групп процессов. Каждый из процессов жизненного цикла в пределах этих групп описывается в терминах цели и желаемых выходов, списков действий и задач, которые необходимо выполнять для достижения этих результатов.

- a) процессы **соглашения** - два процесса (см. 5.2.2.1.1 и 6.1);
- b) процессы **организационного обеспечения проекта** - пять процессов (см. 5.2.2.1.2 и 6.2);
- c) процессы **проекта** - семь процессов (см. 5.2.2.1.3 и 6.3);
- d) **технические** процессы - одиннадцать процессов (см. 5.2.2.1.4 и 6.4);
- e) процессы **реализации программных средств** - семь процессов (см. 5.2.2.2.1 и 7.1);
- f) процессы **поддержки программных средств** - восемь процессов (см. 5.2.2.2.2 и 7.2);
- g) процессы **повторного применения программных средств** - три процесса (см. 5.2.2.2.3 и 7.3).

Цели и результаты процессов жизненного цикла образуют эталонную модель процессов.

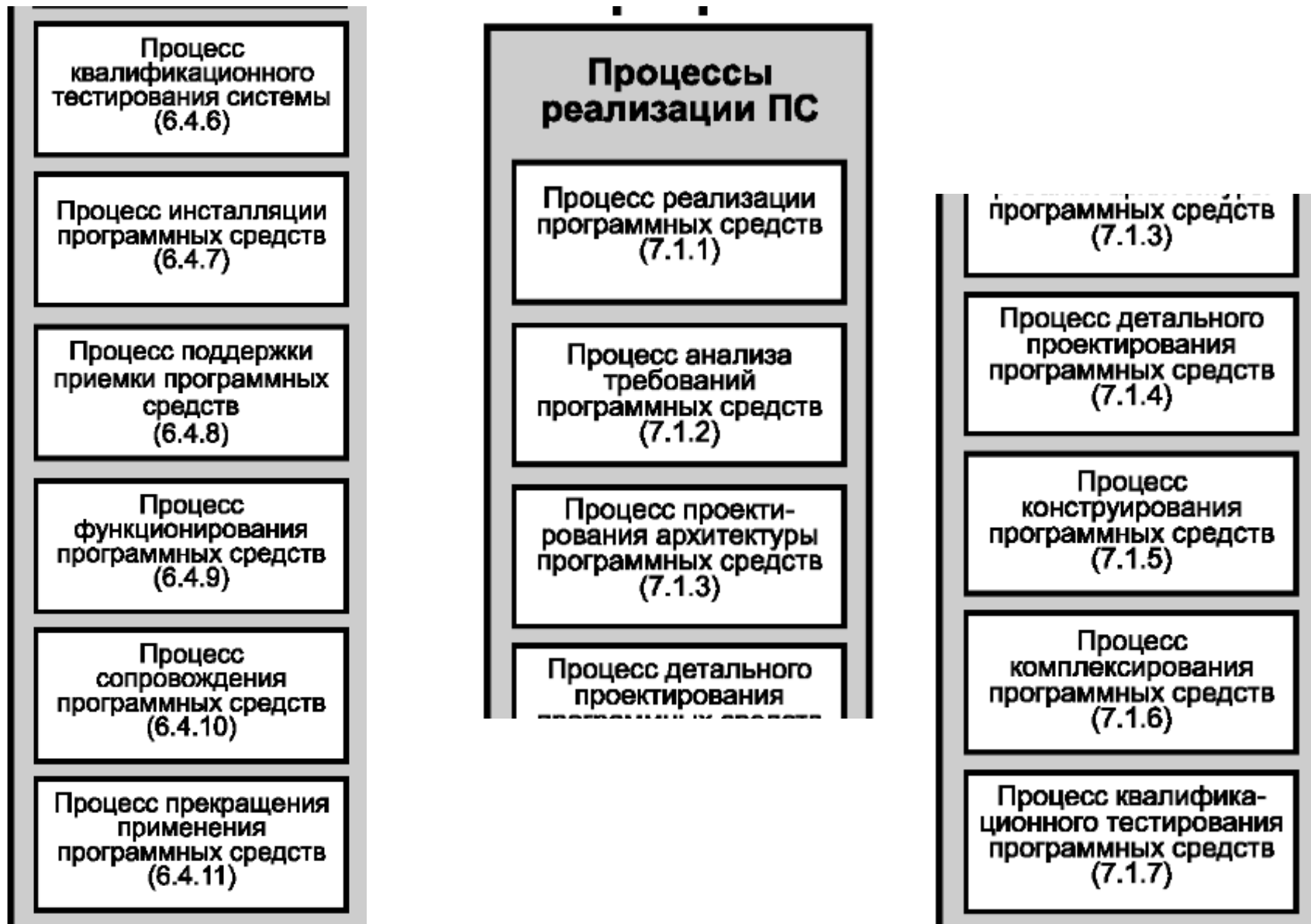
Методология ГОСТ Р ИСО/МЭК 12207-2010



Методология ГОСТ Р ИСО/МЭК 12207-2010



Методология ГОСТ Р ИСО/МЭК 12207-2010



Методология ГОСТ Р ИСО/МЭК 12207-2010



Методология CMMI

Capability Maturity Model Integration (CMMI) — набор моделей (методологий) совершенствования процессов в организациях разных размеров и видов деятельности. CMMI содержит набор рекомендаций в виде практик, реализация которых, по мнению разработчиков модели, позволяет реализовать цели, необходимые для полной реализации определённых областей деятельности.

Набор моделей CMMI включает три модели:
CMMI for Development (CMMI-DEV),
CMMI for Services (CMMI-SVC) и
CMMI for Acquisition (CMMI-ACQ).

Наиболее известной является модель CMMI for Development, ориентированная на организации, занимающиеся разработкой программного обеспечения, аппаратного обеспечения, а также комплексных систем

Концепция зрелости процессов



Методология RUP

Rational Unified Process (RUP) — [методология](#) разработки программного обеспечения, созданная компанией [Rational Software](#).

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**

1996 – старт

1999 – UML 1.3

2003 – IBM

2006 - [OpenUP](#)

Методология RUP

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).

Методология RUP

Рабочие процессы

Стадии

Основные процессы

Бизнес-моделирование

Управление требованиями

Анализ и проектирование

Реализация

Тестирование

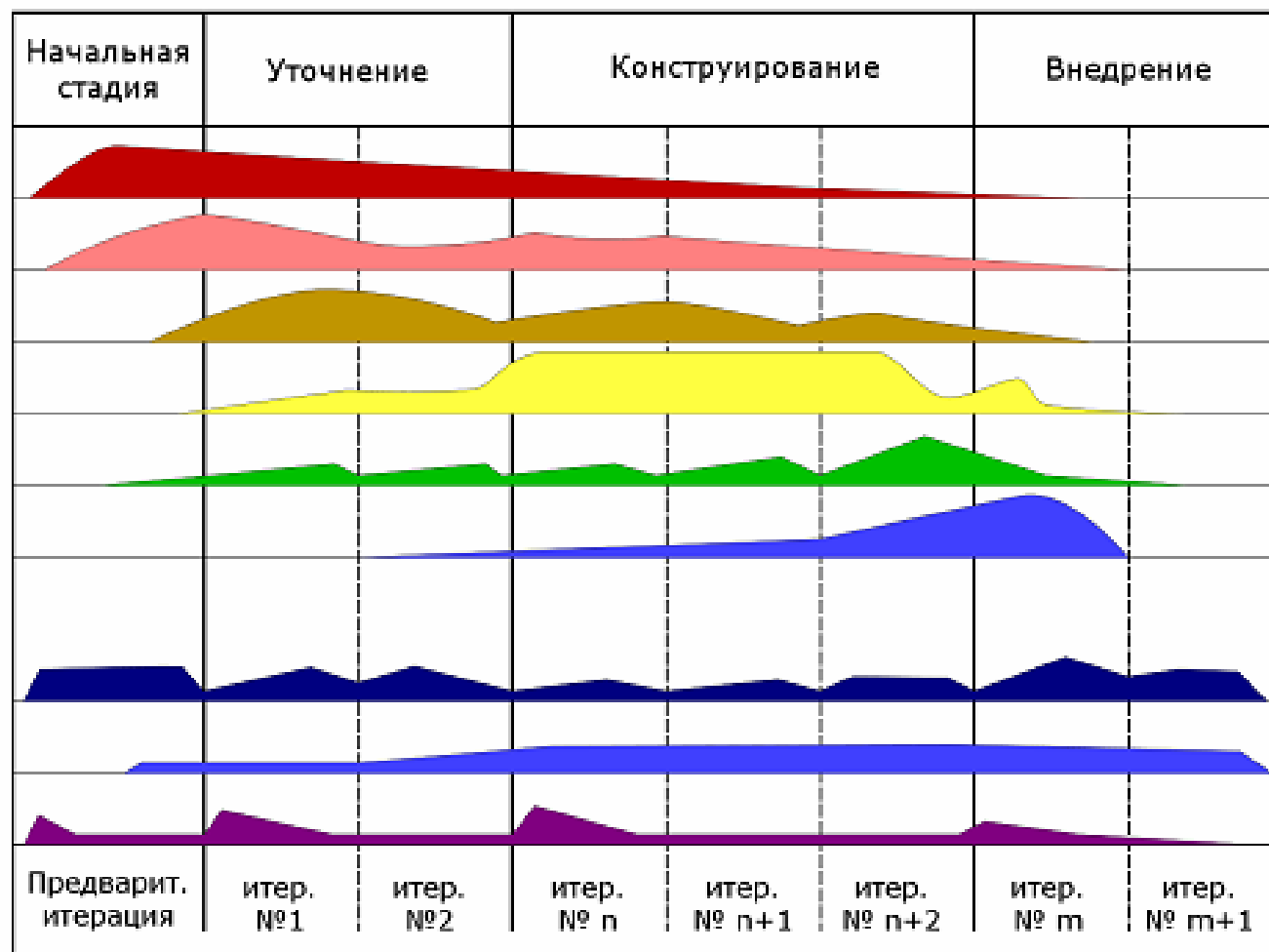
Развертывание

Поддерживающие процессы

Управление проектом

Управление конфигурацией
и изменениями

Создание инфраструктуры
(среда разработки)



Итерации

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

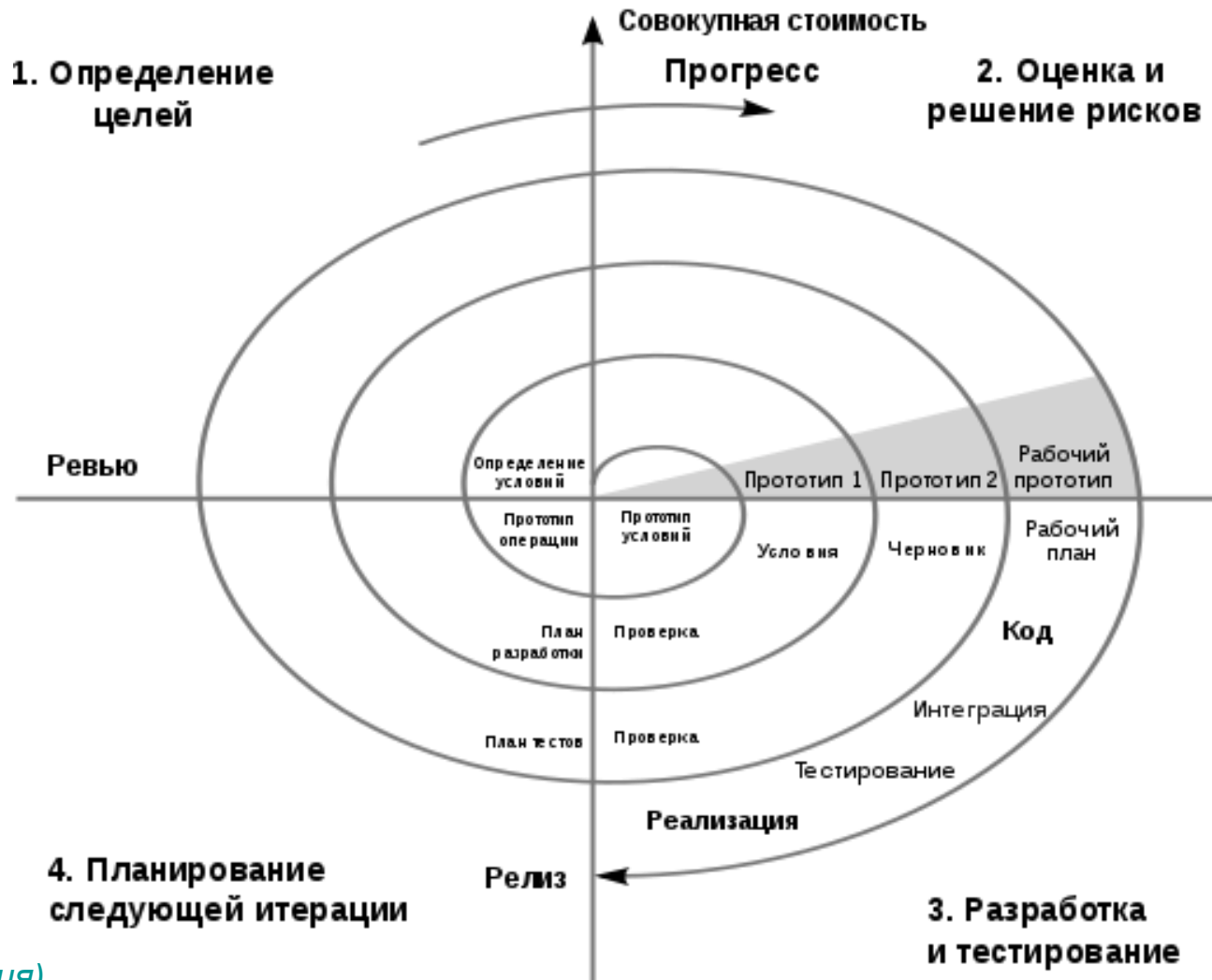
Итеративная разработка позволяет:

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

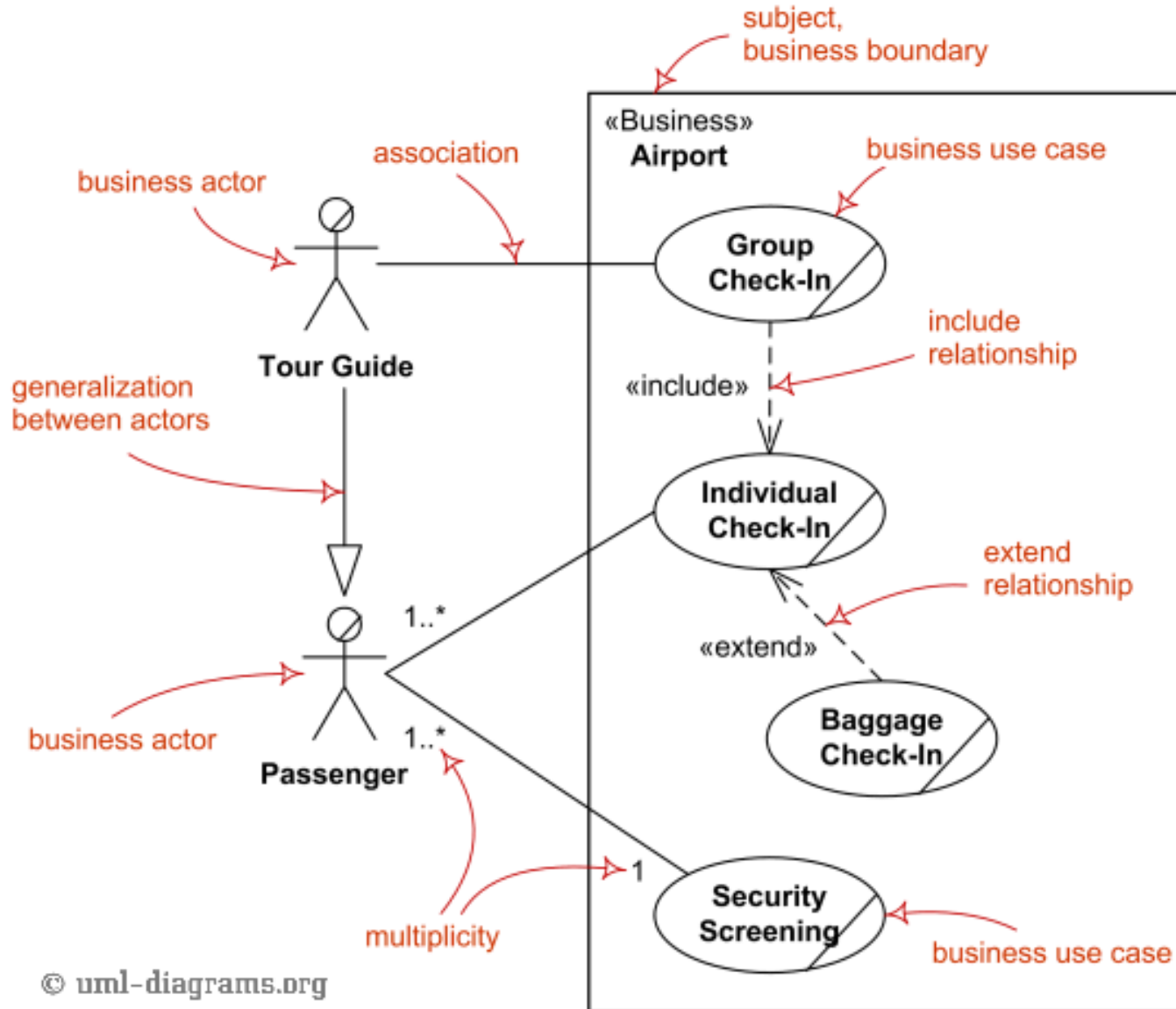
Первые идеи итеративной модели разработки были заложены в "спиральной модели"

RUP: итеративная модель

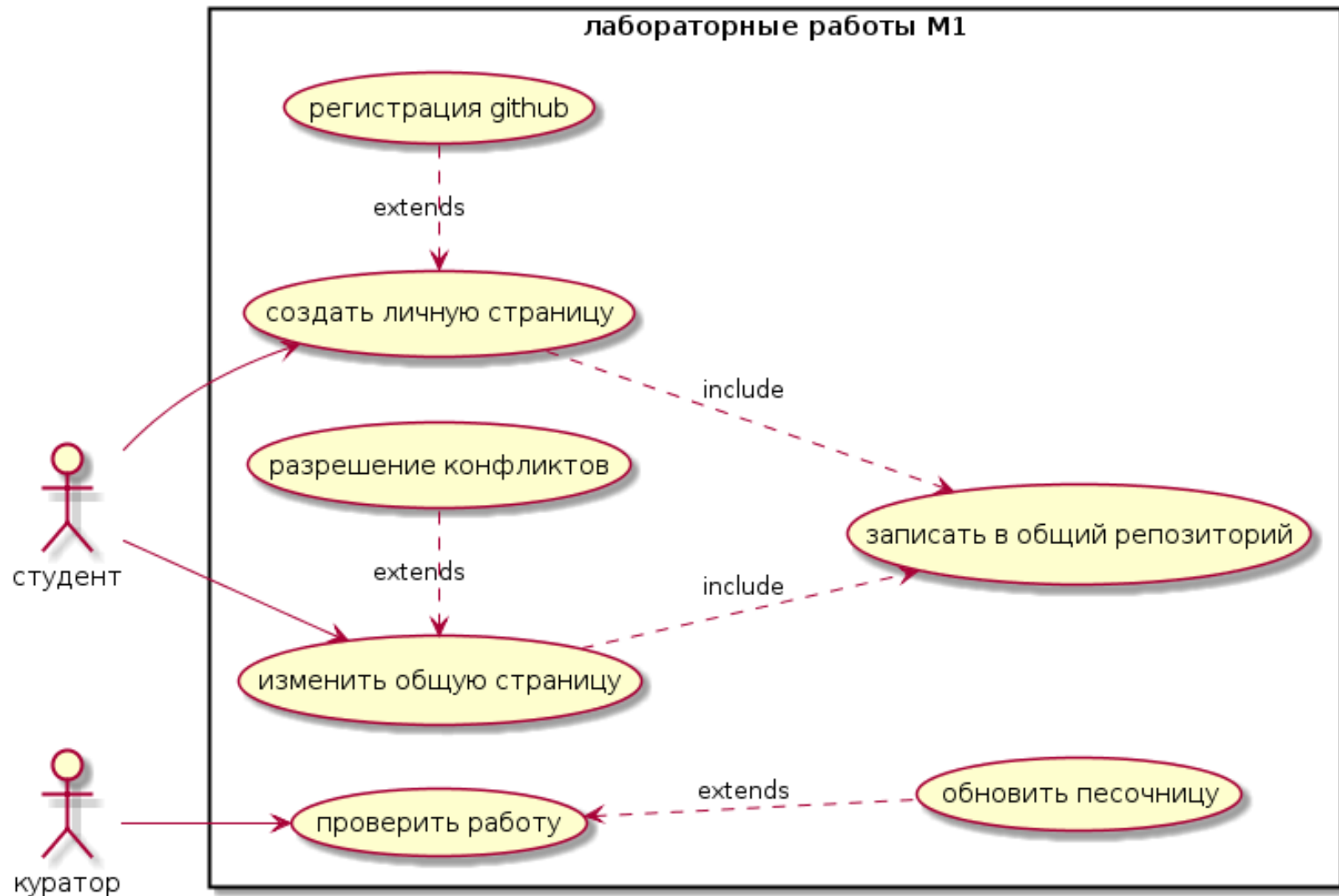
Спиральная модель, предложенная Барри Боэмом в [1986 году](#), представляет собой [процесс разработки программного обеспечения](#), сочетающий в себе как итеративность, так и этапность.



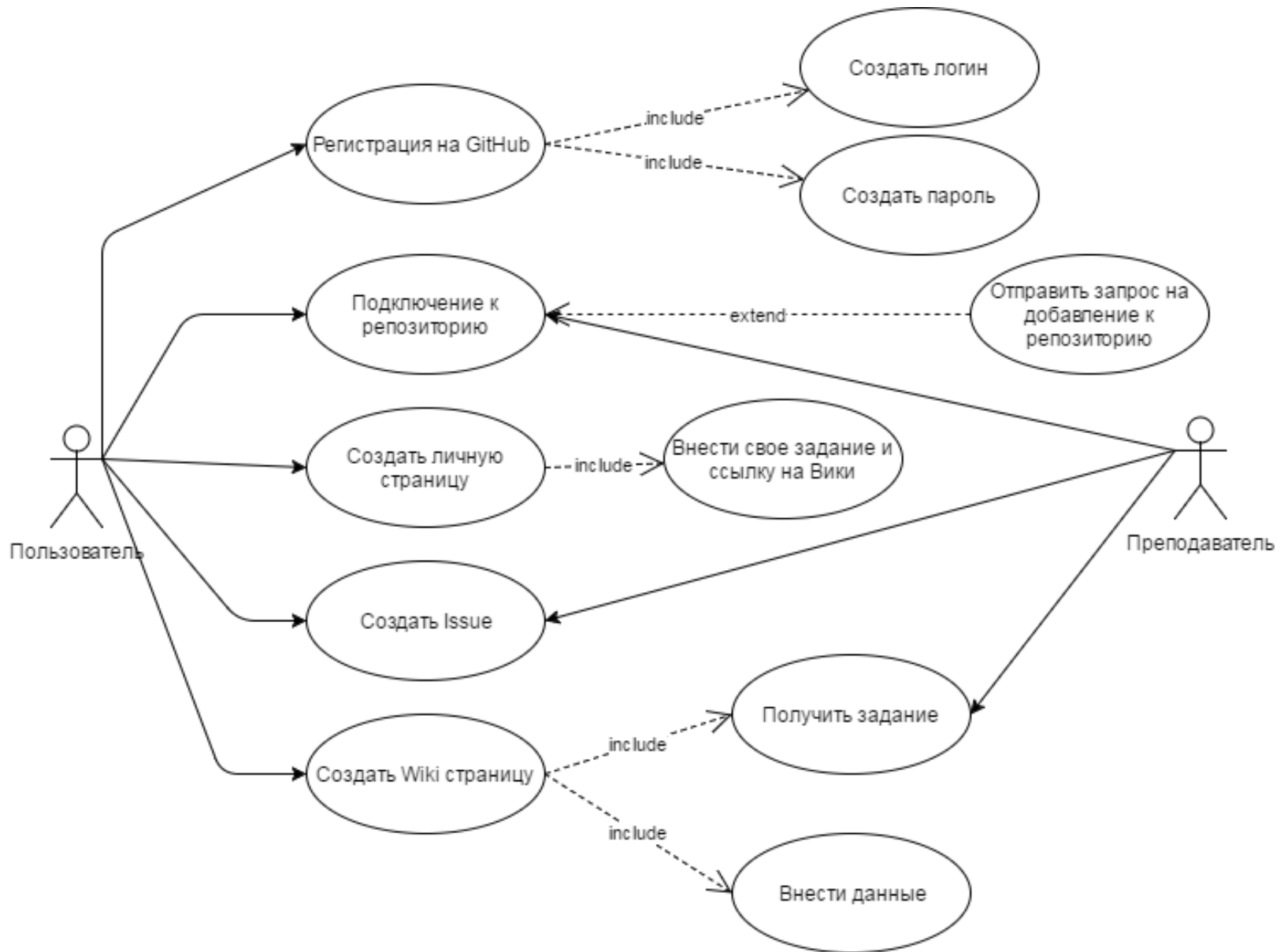
Нотации RUP: Use Case



Нотации RUP: Use Case

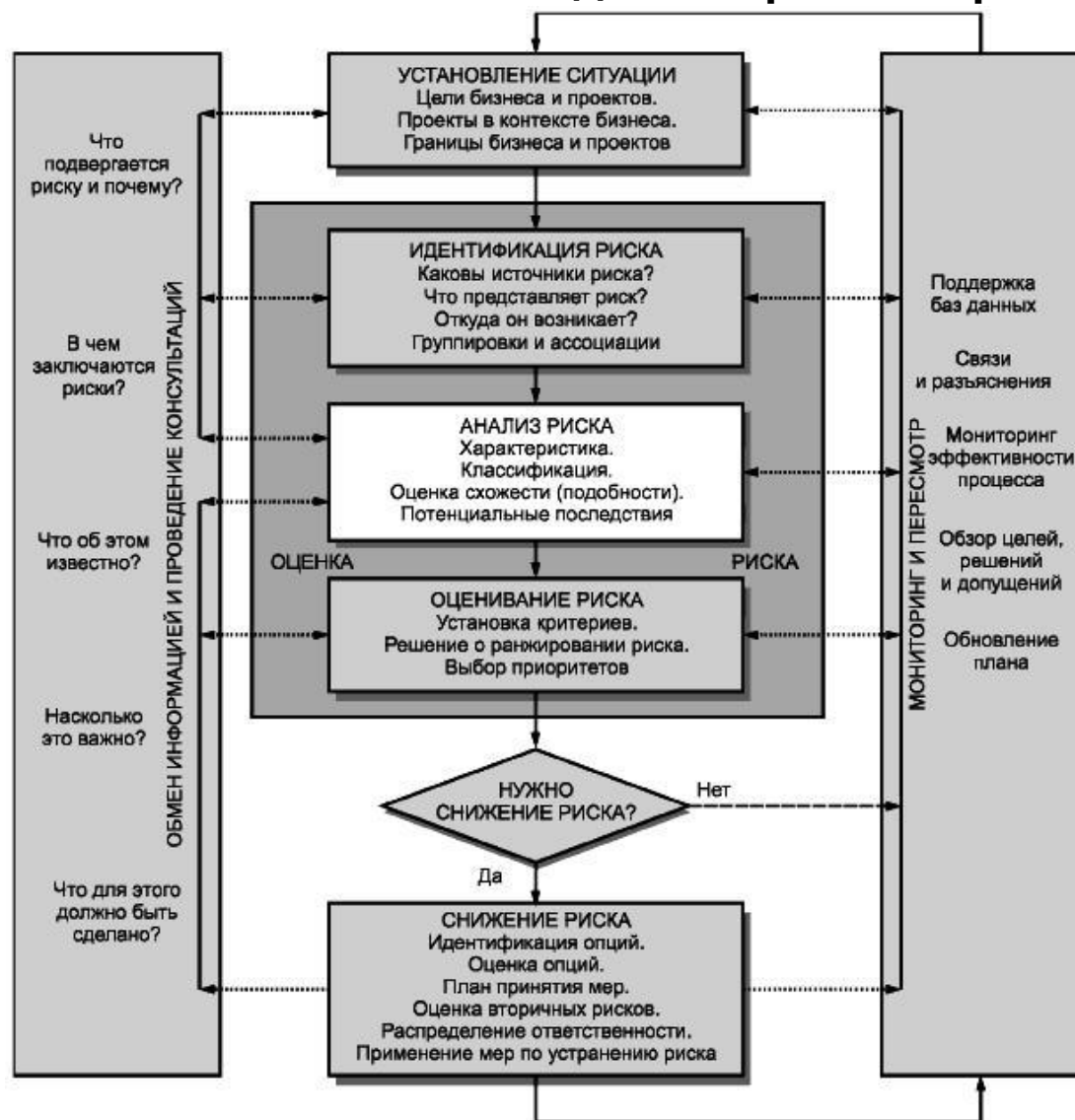


Нотации RUP: Use Case



RUP: управление рисками

ГОСТ Р 52806-2007 Менеджмент рисков проектов. Общие положения



ТЕСТ

Методологии 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^[1].

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 ХР

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)

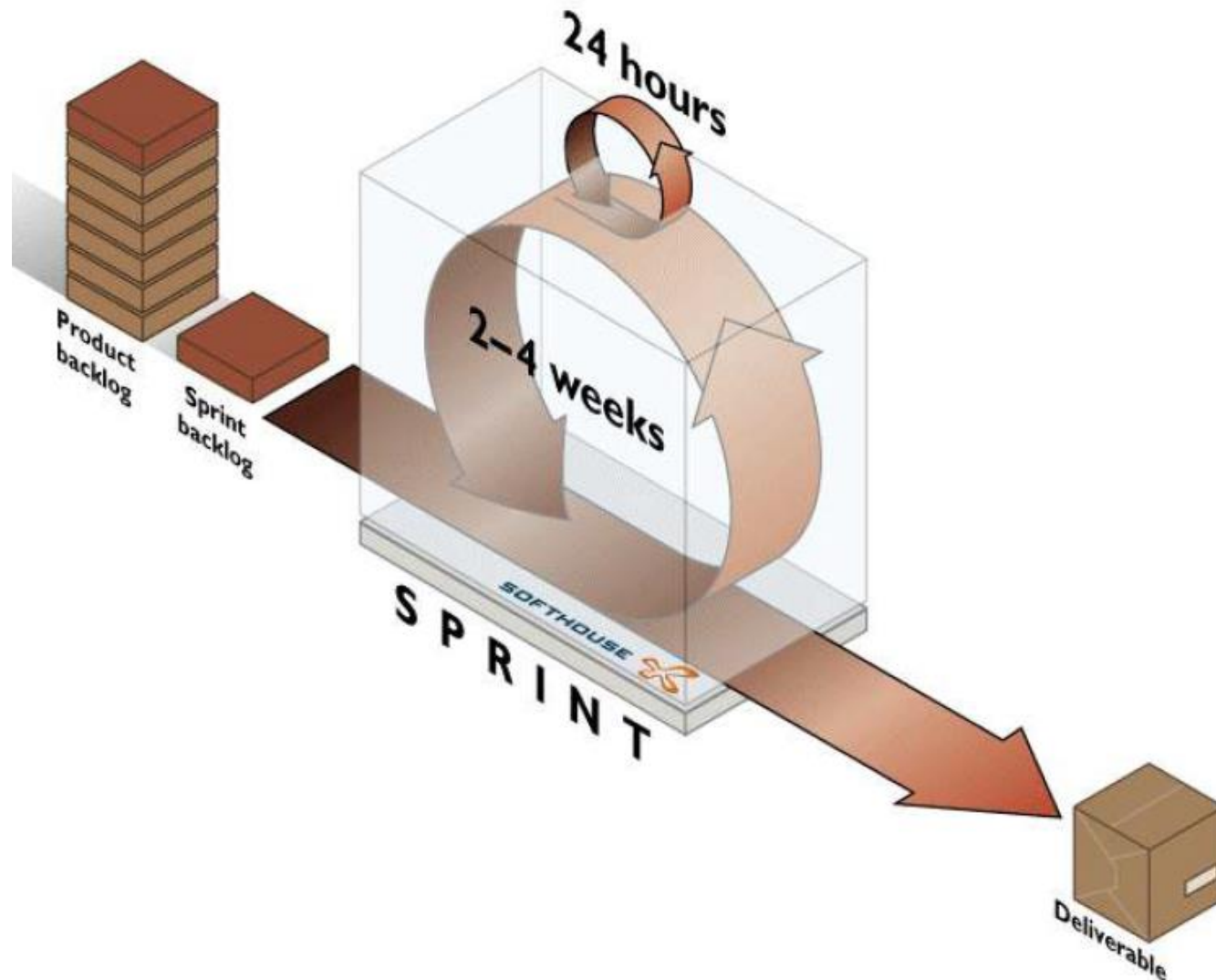
- [Стандарт оформления кода](#) (Coding standard or Coding conventions)

Социальная защищённость программиста (Programmer welfare):

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

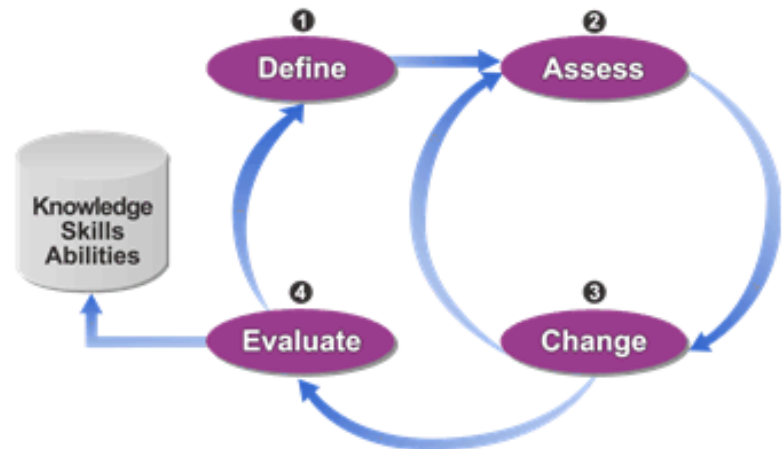
Методология Scrum

Scrum ([/skrʌm/\[1\]\[2\]](#); [англ.](#) *scrum* «схватка») — методология [гибкой разработки](#) ПО. Методология делает акцент на качественном контроле процесса разработки.



Методология MSF

Microsoft Solutions Framework (MSF) — методология разработки программного обеспечения, предложенная корпорацией Microsoft. MSF опирается на практический опыт Microsoft и описывает управление людьми и рабочими процессами в процессе разработки решения.

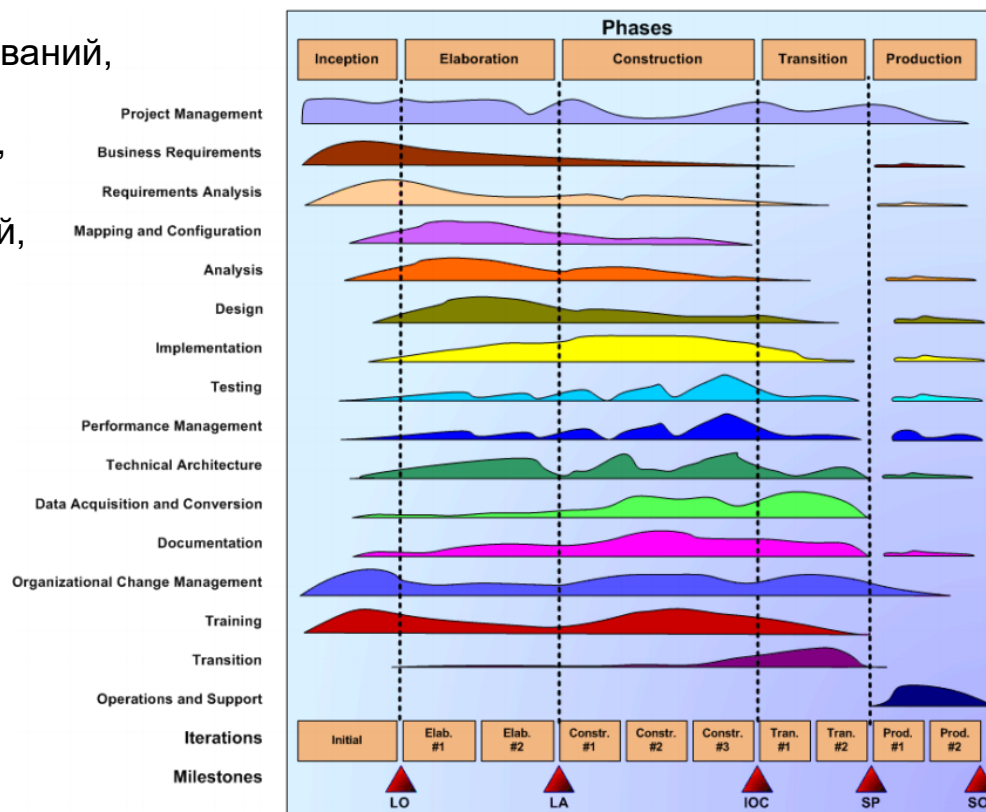


Методология OUM

Oracle Unified Method (унифицированный метод Oracle, сокр. OUM) - фреймворк для итеративного и инкрементального процесса разработки ПО, разработанный корпорацией Oracle для реализации своей точки зрения на поддержку успешной реализации каждого продукта Oracle – приложений, промежуточного ПО и баз данных.

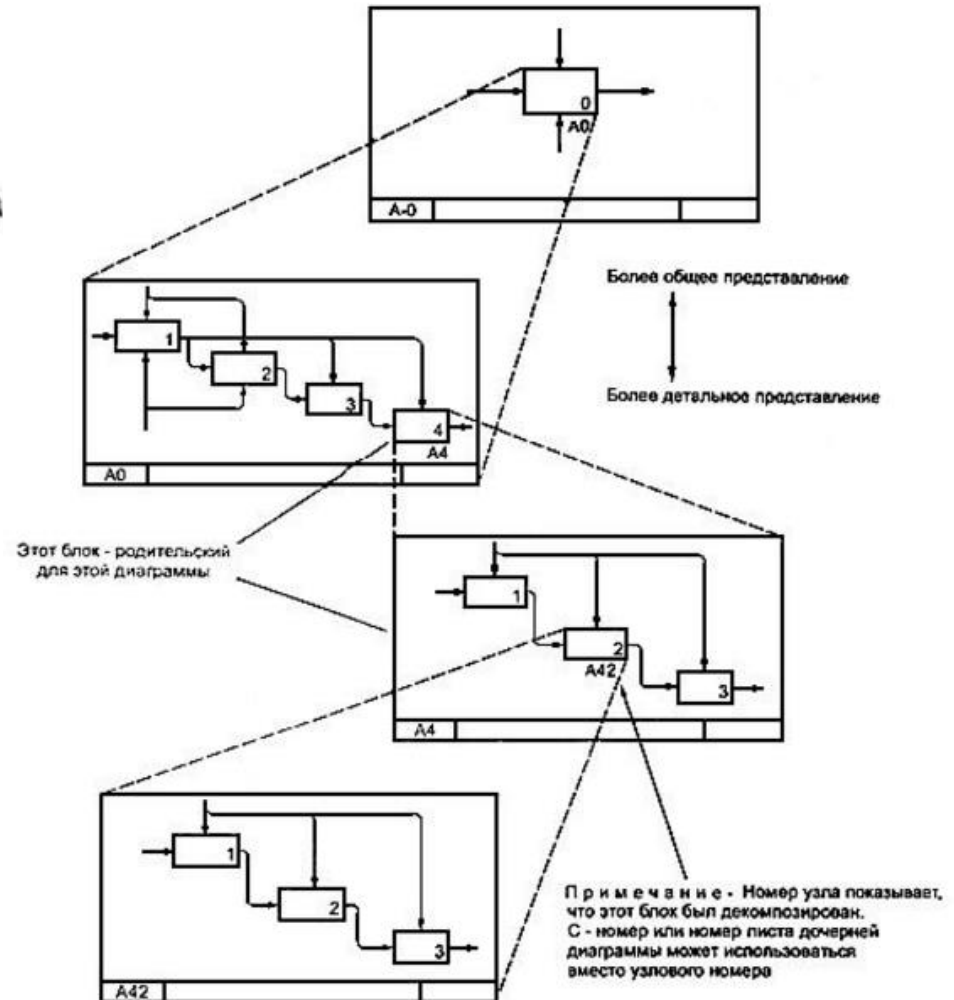
Процессы разработки

RD - Определение производственных требований,
ES - Исследование существующих систем,
TA - Определение технической архитектуры,
DB - Проектирование и построение БД,
MD - Проектирование и реализация модулей,
CV - Конвертирование данных,
DO - Документирование,
TE - Тестирование,
TR - Обучение,
TS - Переход к новой системе,
PS - Поддержка и сопровождение.



Методология SADT

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



Методология DFD

DFD — общепринятое сокращение от [англ. data flow diagrams](#) — диаграммы потоков данных. Так называется [методология](#) графического структурного [анализа](#), описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

