Методологии проектирования программных и информационных средств

Лекция 5

Проектирование и разработка программных средств

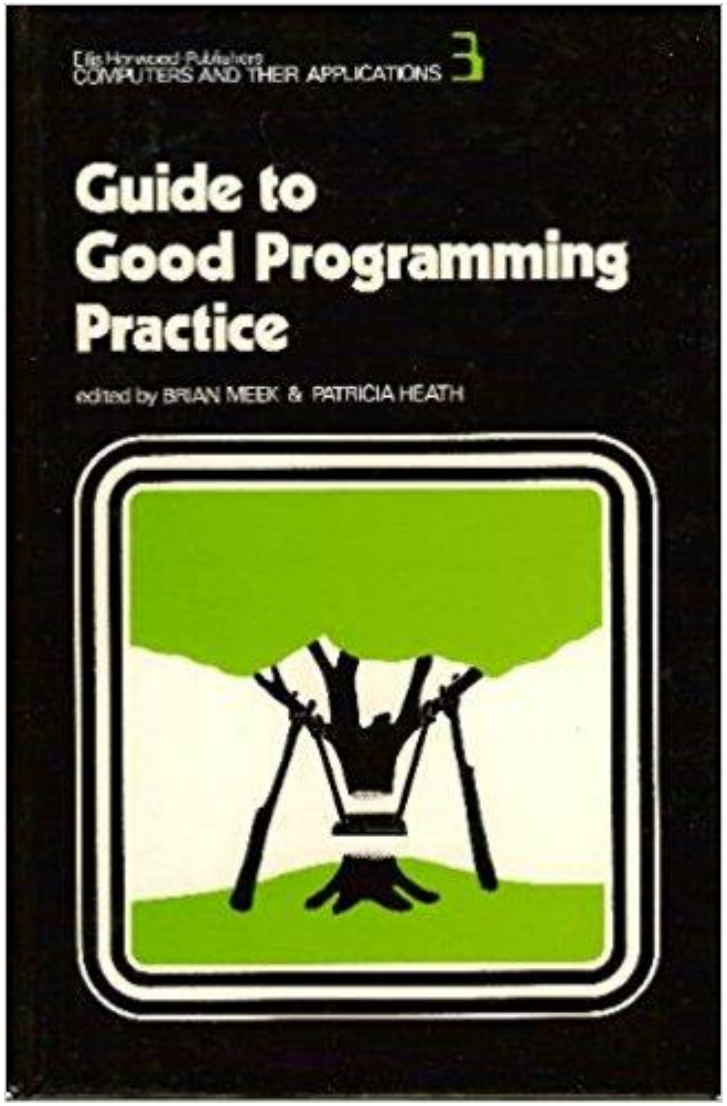
Овчинников П.Е.

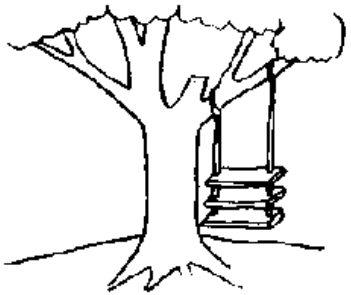
МГТУ «СТАНКИН»,

ст.преподаватель кафедры ИС

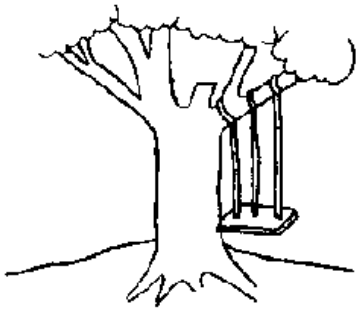
15.10.2020

Проблематика

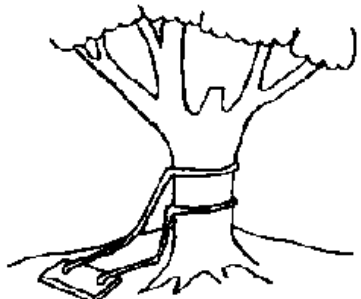




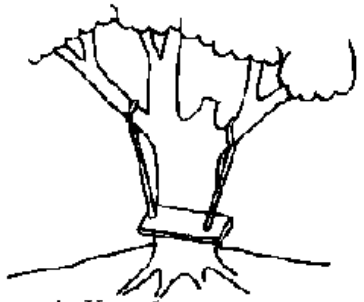
1. Как было предложено организатором разработки



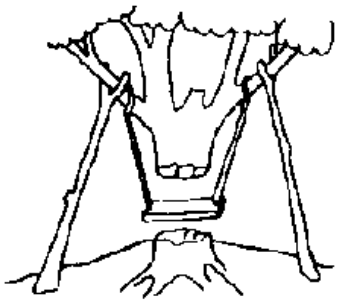
2. Как было описано в техническом задании



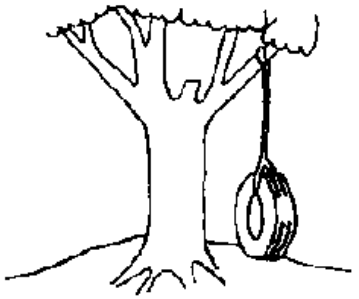
3. Как было спроектировано ведущим системным специалистом



4. Как было реализовано программистами



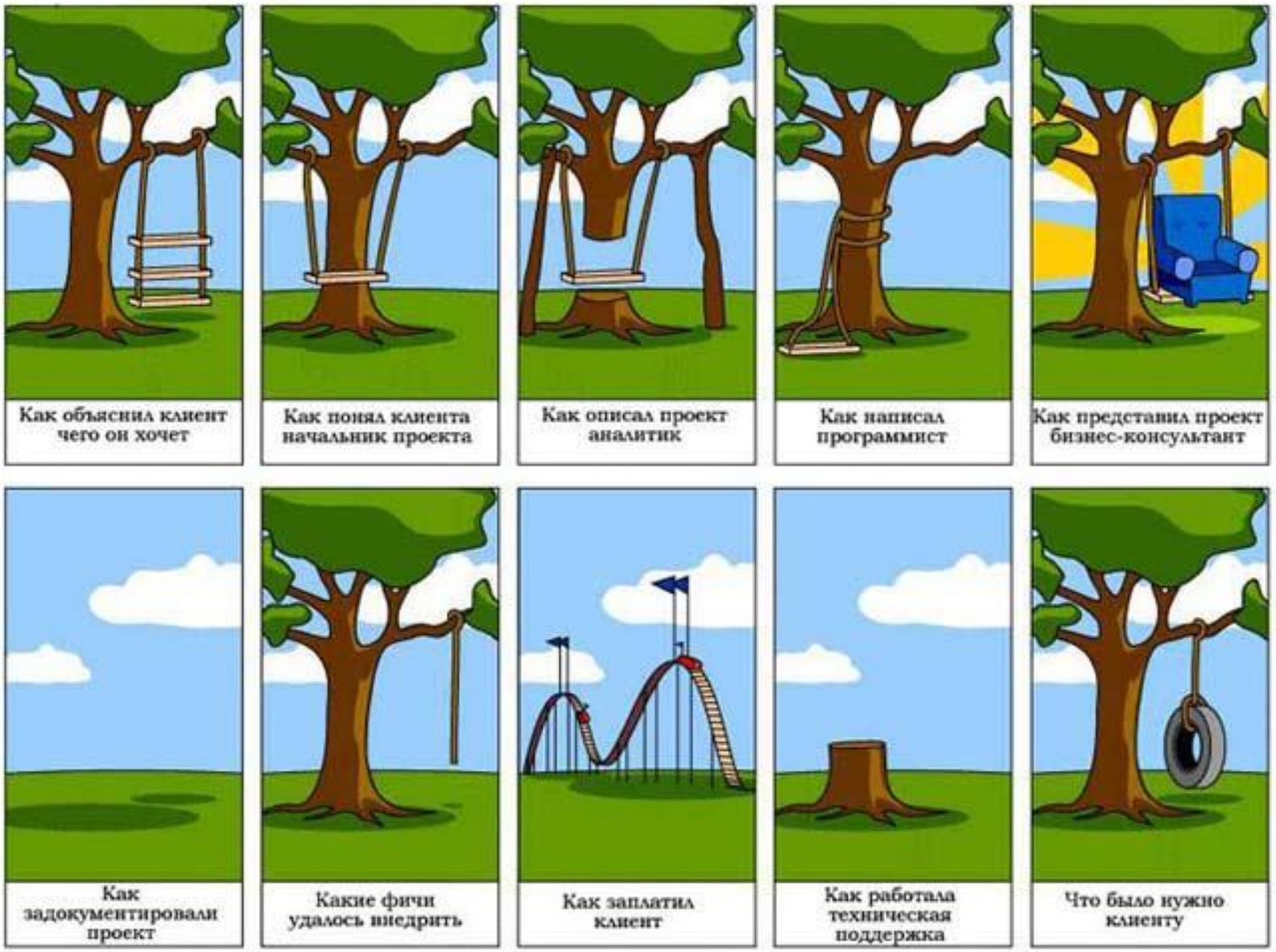
5. Как было внедрено



6. Чего хотел пользователь

Ellis Horwood Ltd , Publisher (June 11, 1980)
ГОСТ Р 57193-2016

Проблематика



ТЗ на сайт
ГОСТ Р ИСО/МЭК 12207-2010

Терминология: методология

Ме́тод (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.

Алгори́тм — 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.

Тео́рия управле́ния — 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

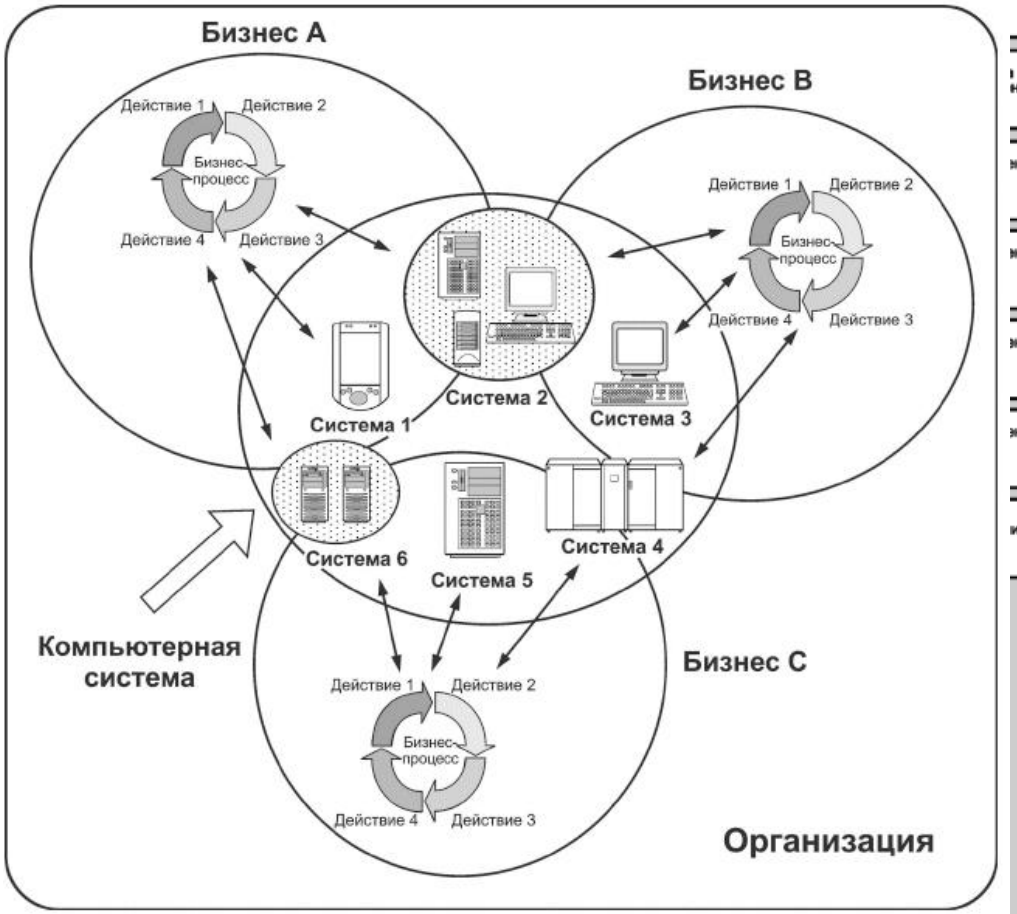
Настоящий стандарт группирует различные виды деятельности, которые могут выполняться в течение жизненного цикла программных систем, в семь групп процессов. Каждый из процессов жизненного цикла в пределах этих групп описывается в терминах цели и желаемых выходов, списков действий и задач, которые необходимо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

ГОСТ Р ИСО/МЭК 12207-2010
ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011

Методология ГОСТ Р ИСО/МЭК 12207-2010



Процессы в контексте системы

Процессы соглашения

Процессы проекта

Технические процессы

- Процесс определения требований правообладателей (6.4.1)
- Процесс анализа системных требований (6.4.2)
- Процесс проектирования архитектуры системы (6.4.3)
- Процесс реализации (6.4.4)
- Процесс комплексирования системы (6.4.5)
- Процесс квалификационного тестирования системы (6.4.6)
- Процесс инсталляции программных средств (6.4.7)
- Процесс поддержки приемки программных средств (6.4.8)
- Процесс функционирования программных средств (6.4.9)
- Процесс сопровождения программных средств (6.4.10)
- Процесс прекращения применения программных средств (6.4.11)

Специальные процессы программных средств

Процессы реализации ПС

- Процесс реализации программных средств (7.1.1)
- Процесс анализа требований программных средств (7.1.2)
- Процесс проектирования архитектуры программных средств (7.1.3)
- Процесс детального проектирования программных средств (7.1.4)
- Процесс конструирования программных средств (7.1.5)
- Процесс комплексирования программных средств (7.1.6)
- Процесс квалификационного тестирования программных средств (7.1.7)

Процессы поддержки ПС

- Процесс менеджмента программной документации (7.2.1)
- Процесс менеджмента конфигурации (7.2.2)
- Процесс обеспечения гарантий качества программных средств (7.2.3)
- Процесс верификации программных средств (7.2.4)
- Процесс валидации программных средств (7.2.5)
- Процесс ревизии программных средств (7.2.6)
- Процесс аудита программных средств (7.2.7)
- Процесс решения проблем в программных средствах (7.2.8)

Процессы повторного применения программных средств

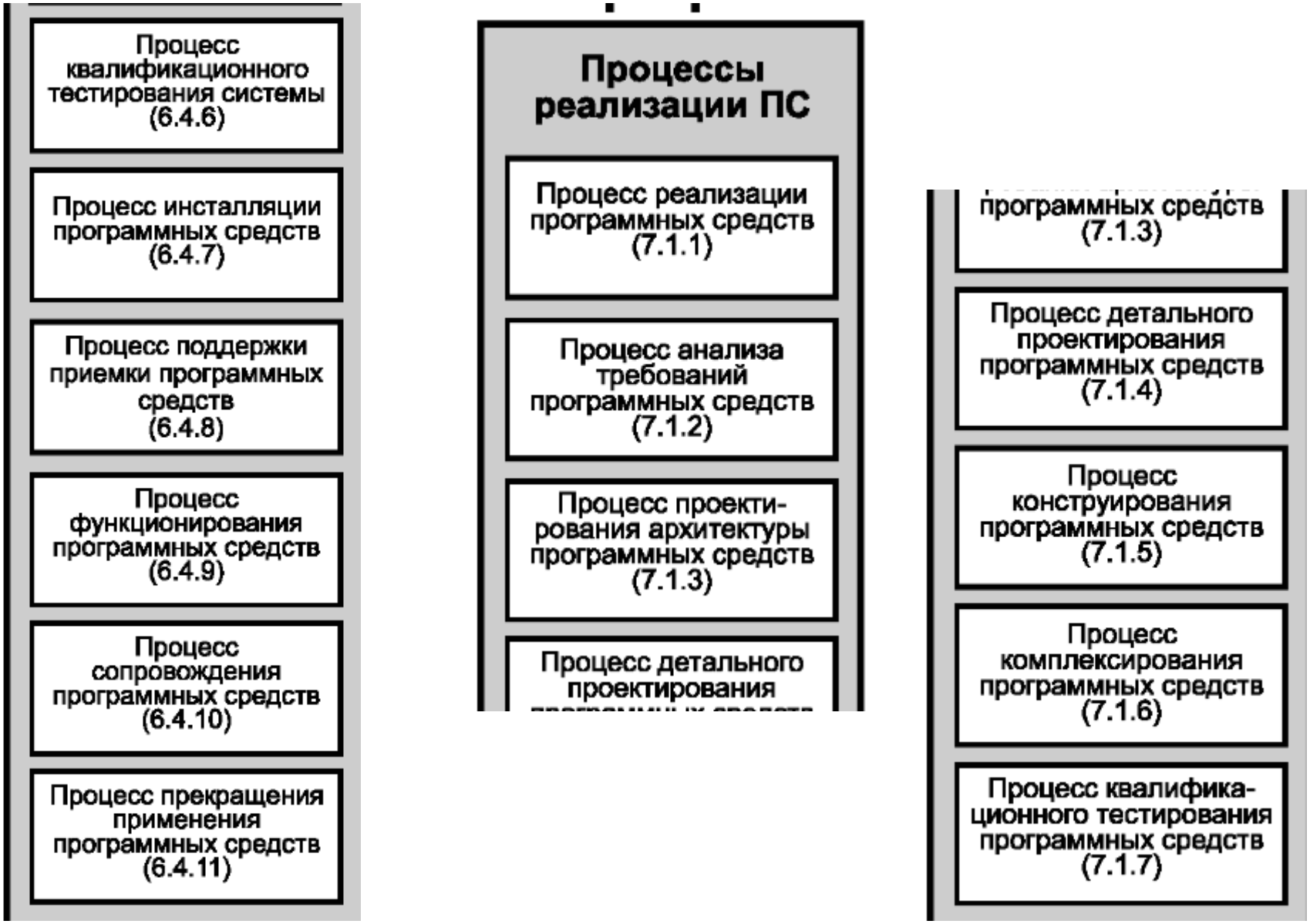
- Процесс проектирования доменов (7.3.1)
- Процесс менеджмента повторного применения активов (7.3.2)
- Процесс менеджмента повторного применения программ (7.3.3)

ГОСТ Р ИСО/МЭК 12207-2010
ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011

Методология ГОСТ Р ИСО/МЭК 12207-2010

ГОСТ Р ИСО/МЭК 12207-2010
ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011

Методология ГОСТ Р ИСО/МЭК 12207-2010



ГОСТ Р ИСО/МЭК 12207-2010
ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011

Методология ГОСТ Р ИСО/МЭК 12207-2010

ГОСТ Р ИСО/МЭК 12207-2010
ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011

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:

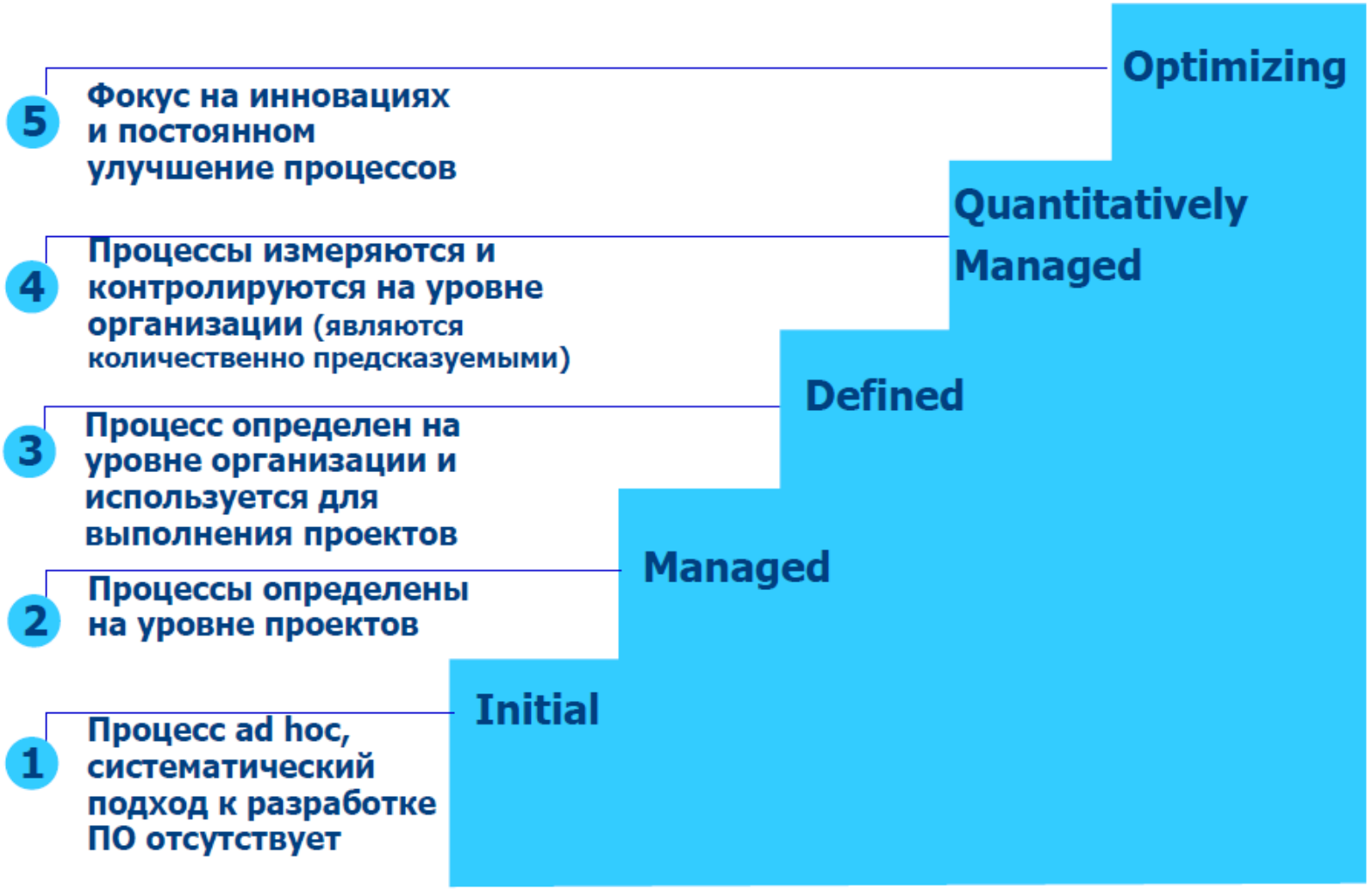
- CMMI for Development (CMMI-DEV),
- CMMI for Services (CMMI-SVC) и
- CMMI for Acquisition (CMMI-ACQ).

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

CMMI
ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011

Методология CMMI

Концепция зрелости процессов



CMMI
ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011

Методология RUP

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

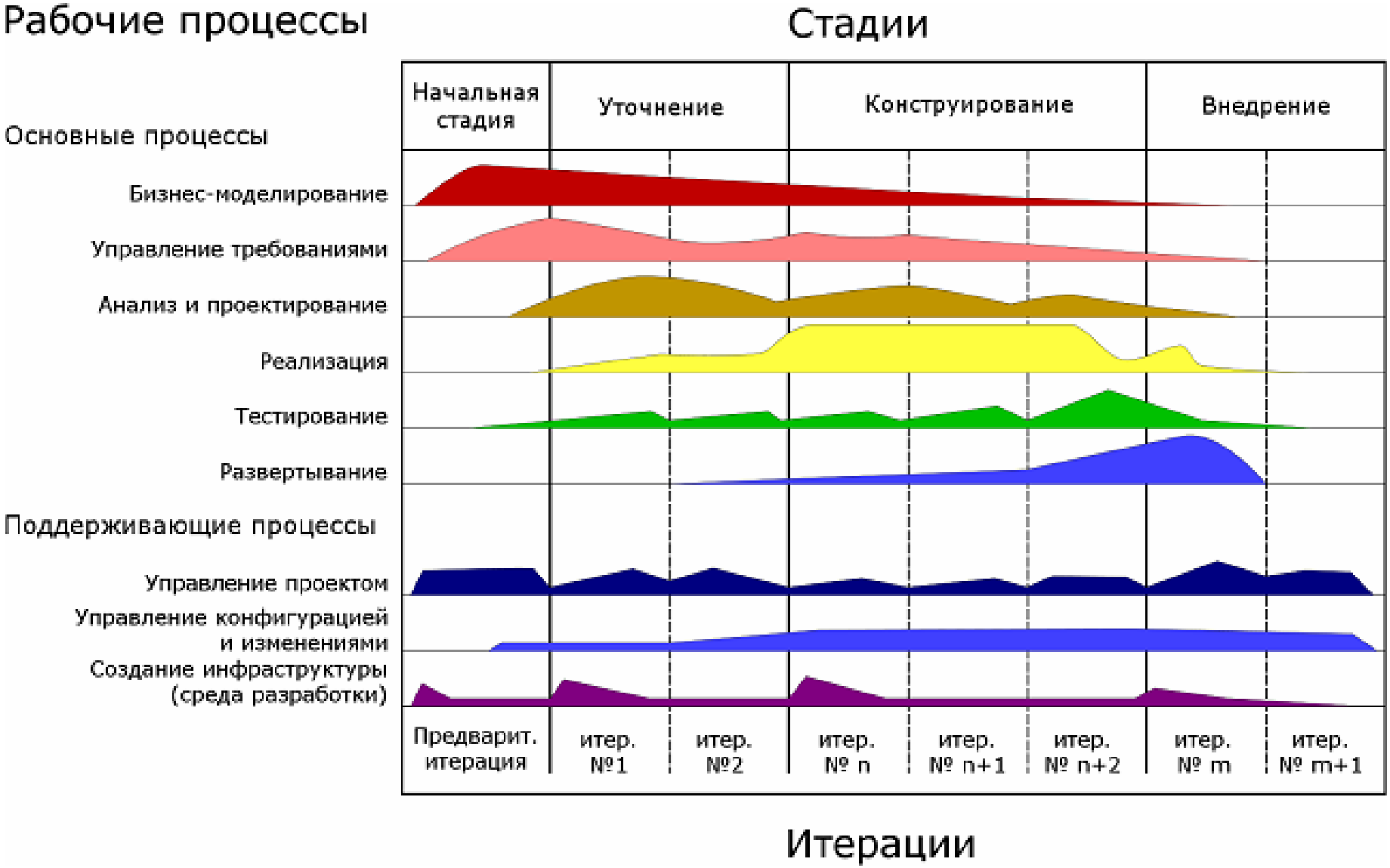
В основе RUP лежат следующие принципы:

- 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

RUP
UML

Методология RUP



RUP
UML

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:

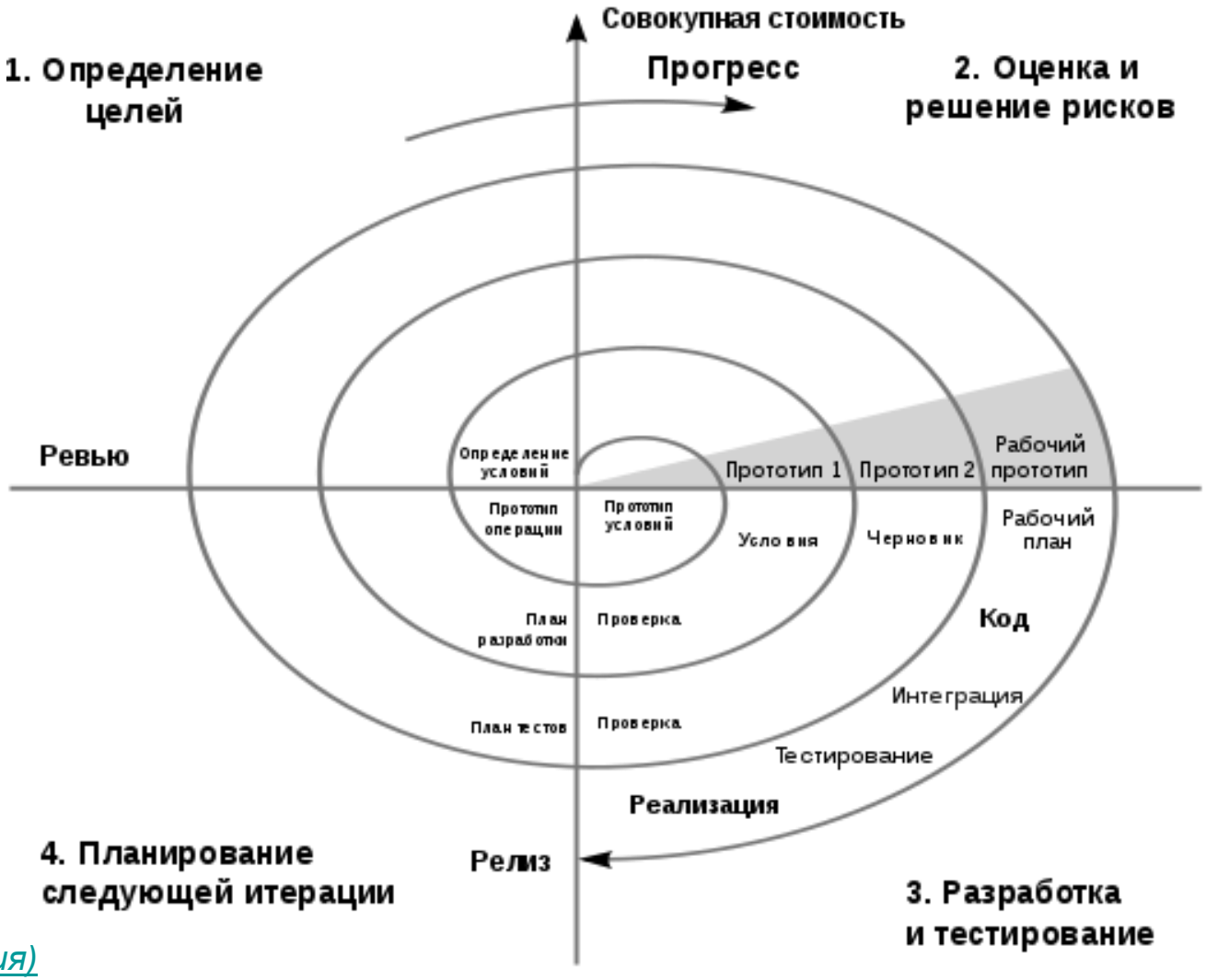
- быстро реагировать на меняющиеся требования,
- обнаруживать и **устранять риски** на ранних стадиях проекта, а также
- эффективно контролировать качество создаваемого продукта

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

RUP (Википедия)

RUP: итеративная модель

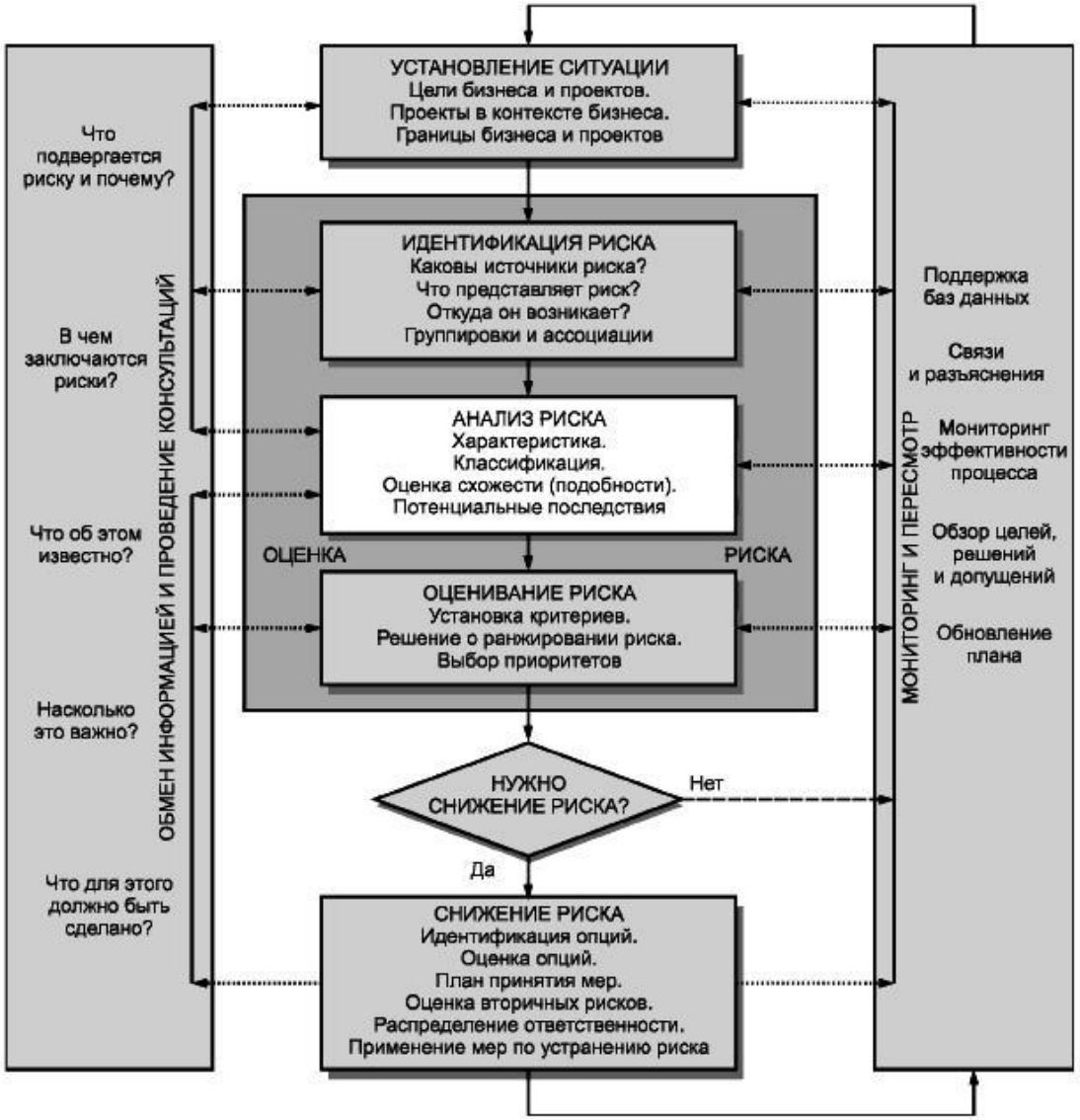
Спира́льная модель, предложенная Барри Боэмом в 1986 году, представляет собой процесс разработки программного обеспечения, сочетающий в себе как итеративность, так и этапность.



RUP (Википедия)

RUP: управление рисками

ГОСТ Р 52806-2007 Менеджмент рисков проектов. Общие положения



ТЕСТ

ГОСТ Р 52806-2007

Терминология: риск

Риск (Risk)

вероятное для проекта **событие**, наступление которого может как **отрицательно**, так и **положительно** отразиться на результатах проекта.

Требование (Requirement)

определенные условия или характеристики, которым должны соответствовать или которые должны иметь система, продукт, услуга, результат или элемент

Допущение (Assumption)

фактор, который **считается верным** для проекта без привлечения доказательств

Ограничение (Constraint)

сдерживающий фактор, влияющий на ход исполнения проекта

Дефект (Defect)

Несовершенство или упущение в элементе проекта, из-за которого этот элемент не соответствует требованиям или характеристикам и должен быть **либо исправлен, либо заменен**

Проблема (Issue)

Обсуждаемая или еще **не решенная** проблема, или проблема, по которой существуют **противоположные мнения** и **разногласия**

Терминология: стратегия управления рисками

В управлении **неблагоприятными рисками (threats)** применяются следующие основные **стратегии** (методы):

Уклонение (Avoidance)	**P(X) = 0**
Снижение (Mitigation)	**P(X) < P(T)**
Передача (Transference)	**P(X) = P(X1)*P(X2) < P(T)**
Принятие (Acceptance)	**P(X) = P(T)**

В управлении **благоприятными рисками (opportunity)** применяются следующие основные стратегии (методы):

Использование (Exploit)	**P(X) = 1**
Усиление (Enhance)	**P(X) > P(O)**
Совместное использование (Share)	**P(X) = P(X1)*P(X2) > P(O)**
Игнорирование (Ignore)	**P(X) = P(O)**

P(T) = **исходная** вероятность **угрозы**
P(O) = **исходная** вероятность **возможности**
P(X) = **целевая** вероятность риска **после применения** стратегии

Лайфхак (противоположные события):

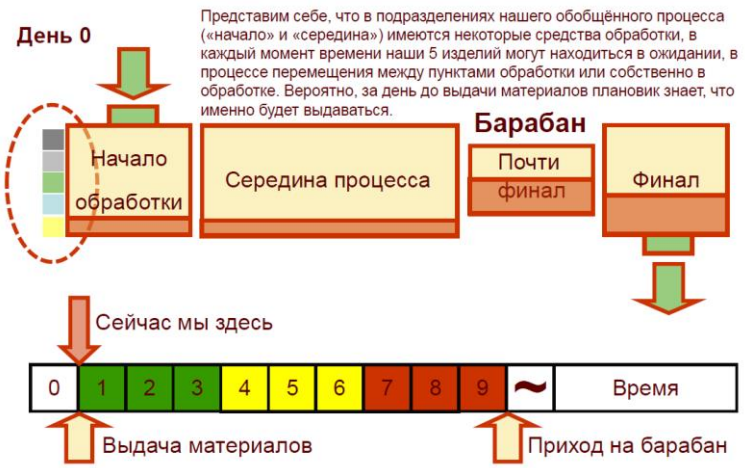
Для любой угрозы **T** существует такая возможность **O**, что

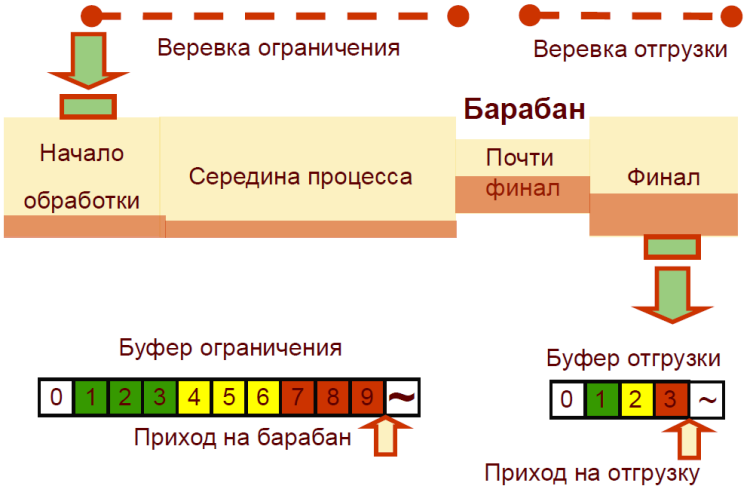
P(O) + P(T) = 1

Планирование запасов: принятие риска

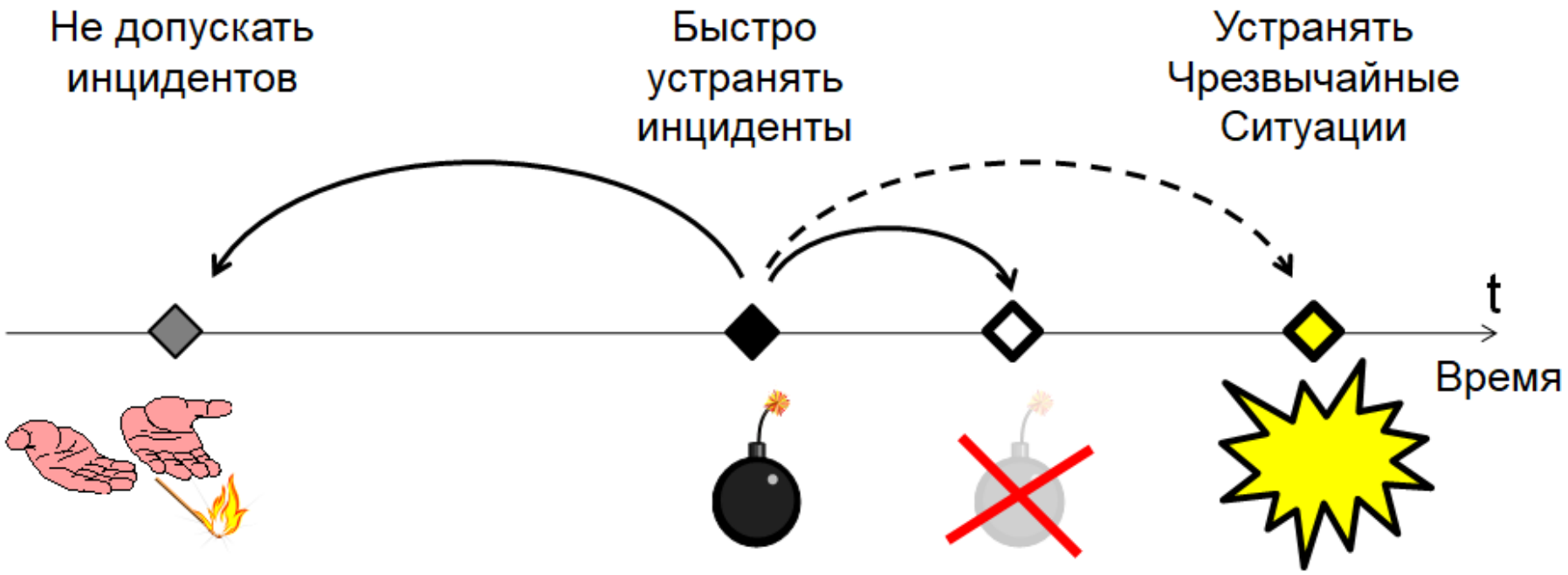
В **управлении рисками** используются следующие основные виды **работ (действий, мероприятий)**:

- Заблаговременное воздействие на **вероятность возникновения** риска
- Заблаговременное воздействие на **степень влияния** риска на
  - **содержание** проекта (Scope)
  - **бюджет** проекта (Cost)
  - **сроки** проекта (Schedule) или на
  - **качество результата** проекта (Quality)
- **Планирование реакции** на неблагоприятное событие (Contingency Plan)
- **Обход** неожиданно возникших неблагоприятных событий (Workaround)



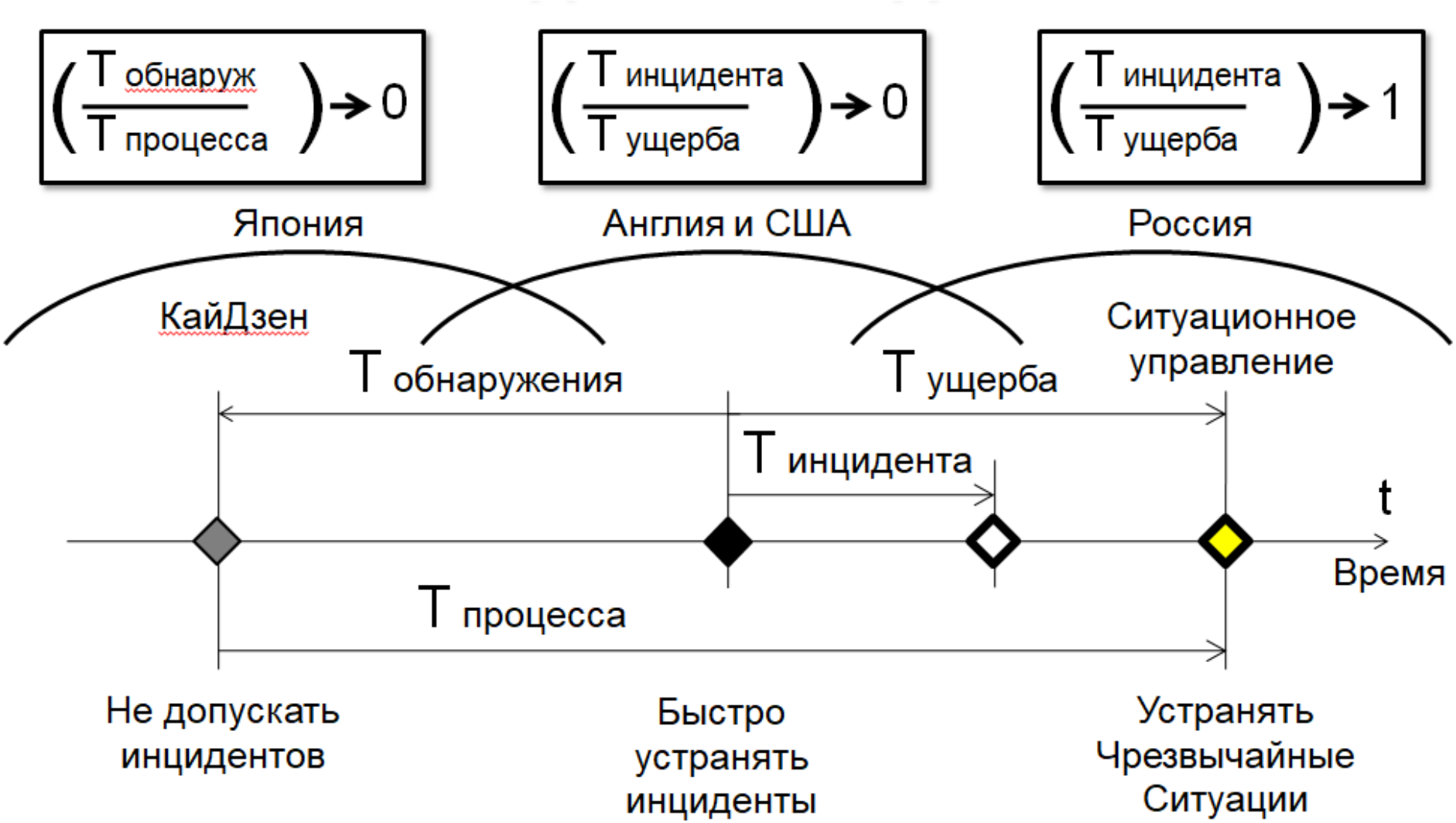


Планирование рисков: стереотипы



Расширение понятия инцидент
(Сервис, Проект, Процесс)

Планирование рисков: стереотипы



RUP: наиболее распространенные риски

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: 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

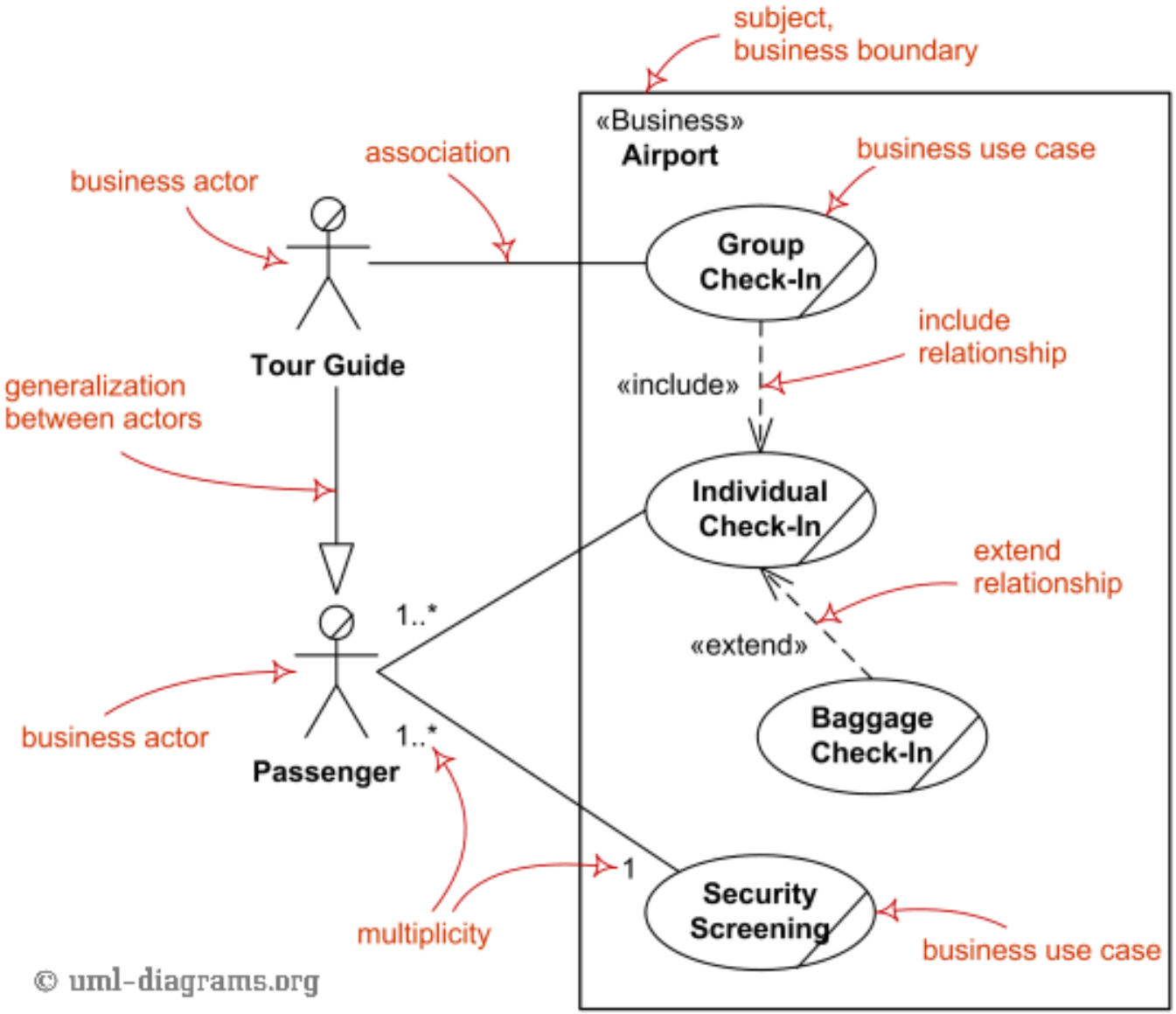
Другими словами, сценарий использования описывает, «кто» и «что» может сделать с рассматриваемой системой, или что система может сделать с «кем» или «чем»

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

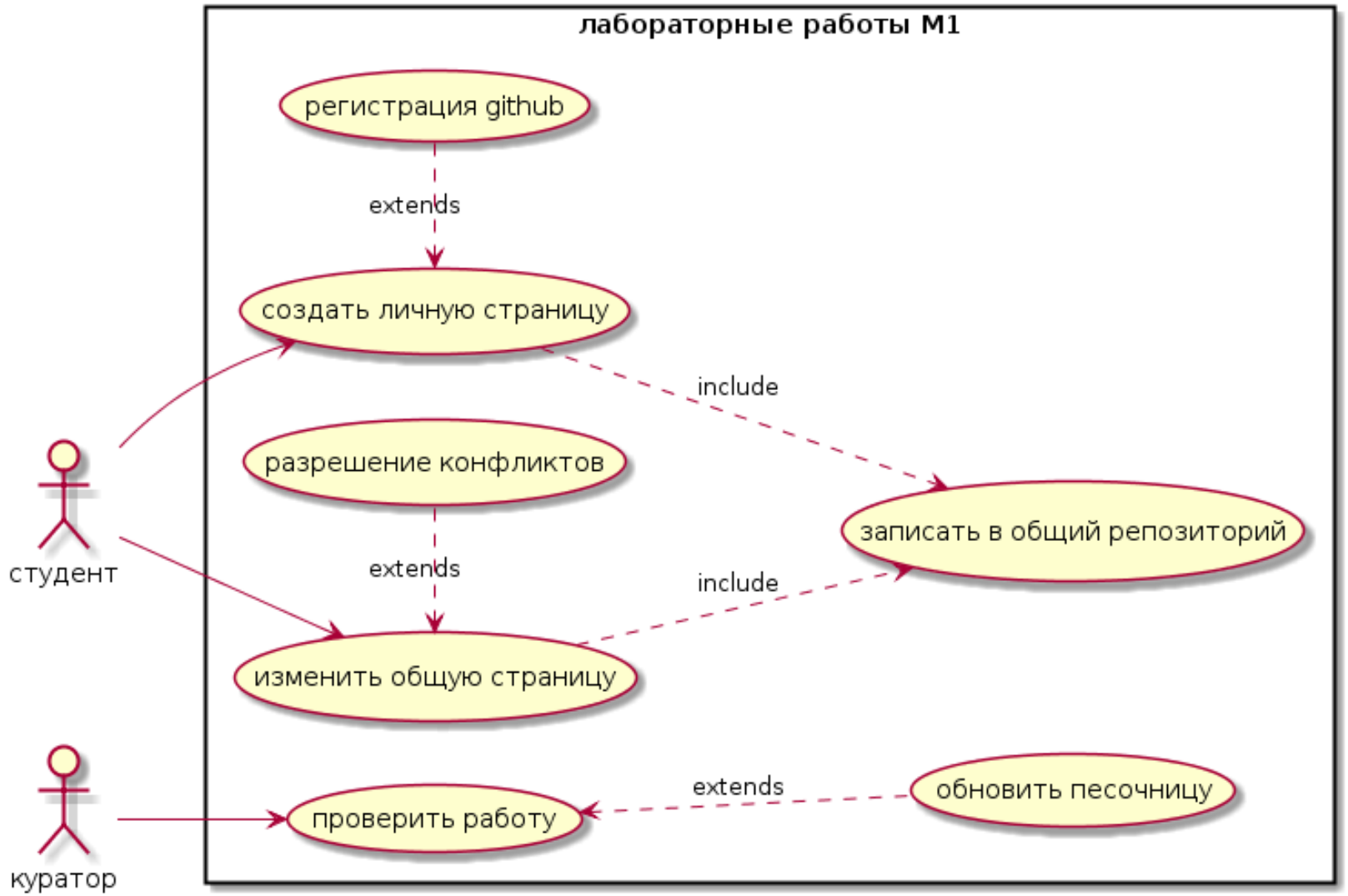
Пользовательские истории (англ. *User Story*) — способ описания **требований** к разрабатываемой системе, сформулированных как одно или более предложений на повседневном или деловом языке пользователя

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

Нотации RUP: Use Case

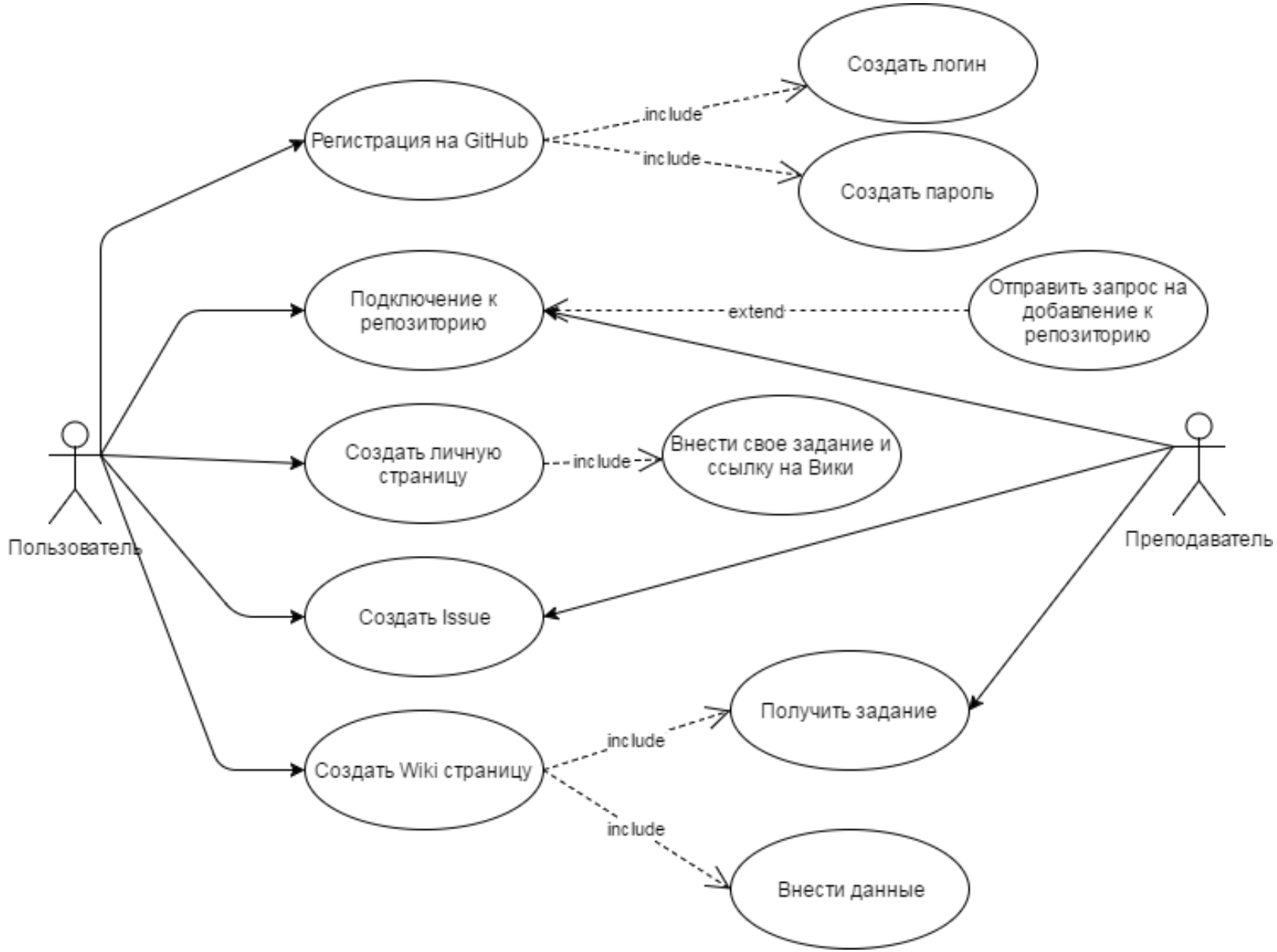


Нотации RUP: Use Case



RUP
UML Use Case (пример)

Нотации RUP: Use Case



RUP
UML Use Case (пример)

Прецеденты: IDEF0 → Use Case

Описание решения

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

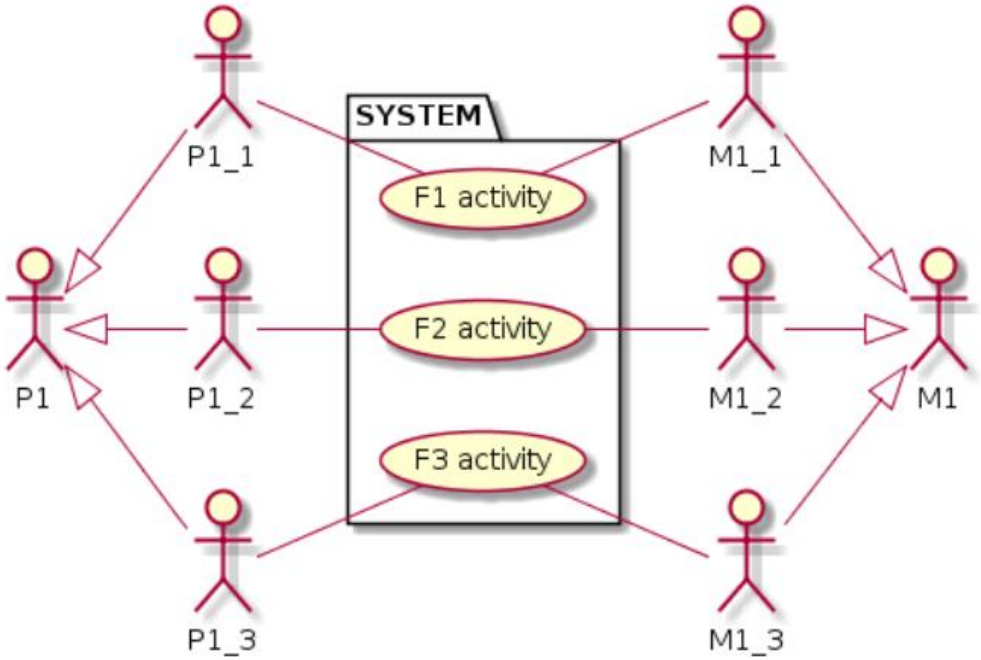
- стрелки механизмов преобразуются в "actor";
- декомпозируемые механизмы становятся родительскими "actor";
- имена блоков становятся именами прецедентов;
- все блоки дочерней диаграммы объединяются в один пакет с именем родительской.

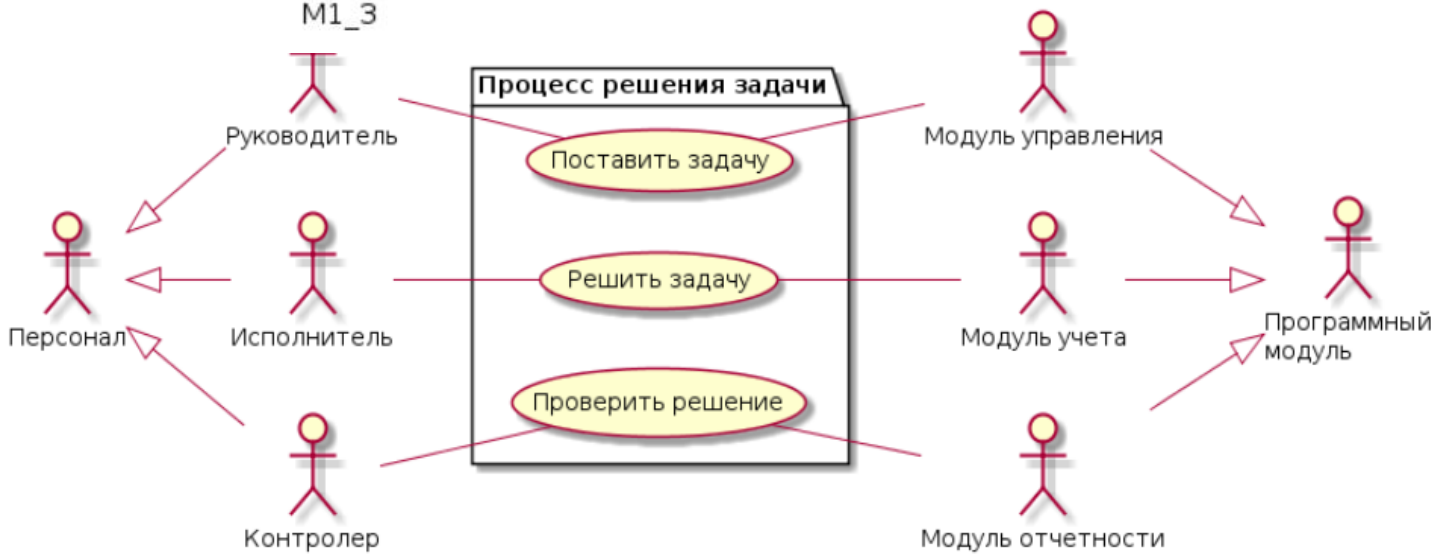
Особенности преобразования

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

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

Манифест (2001) Agile

Методология XP

Экстрема́льное программи́рование (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)

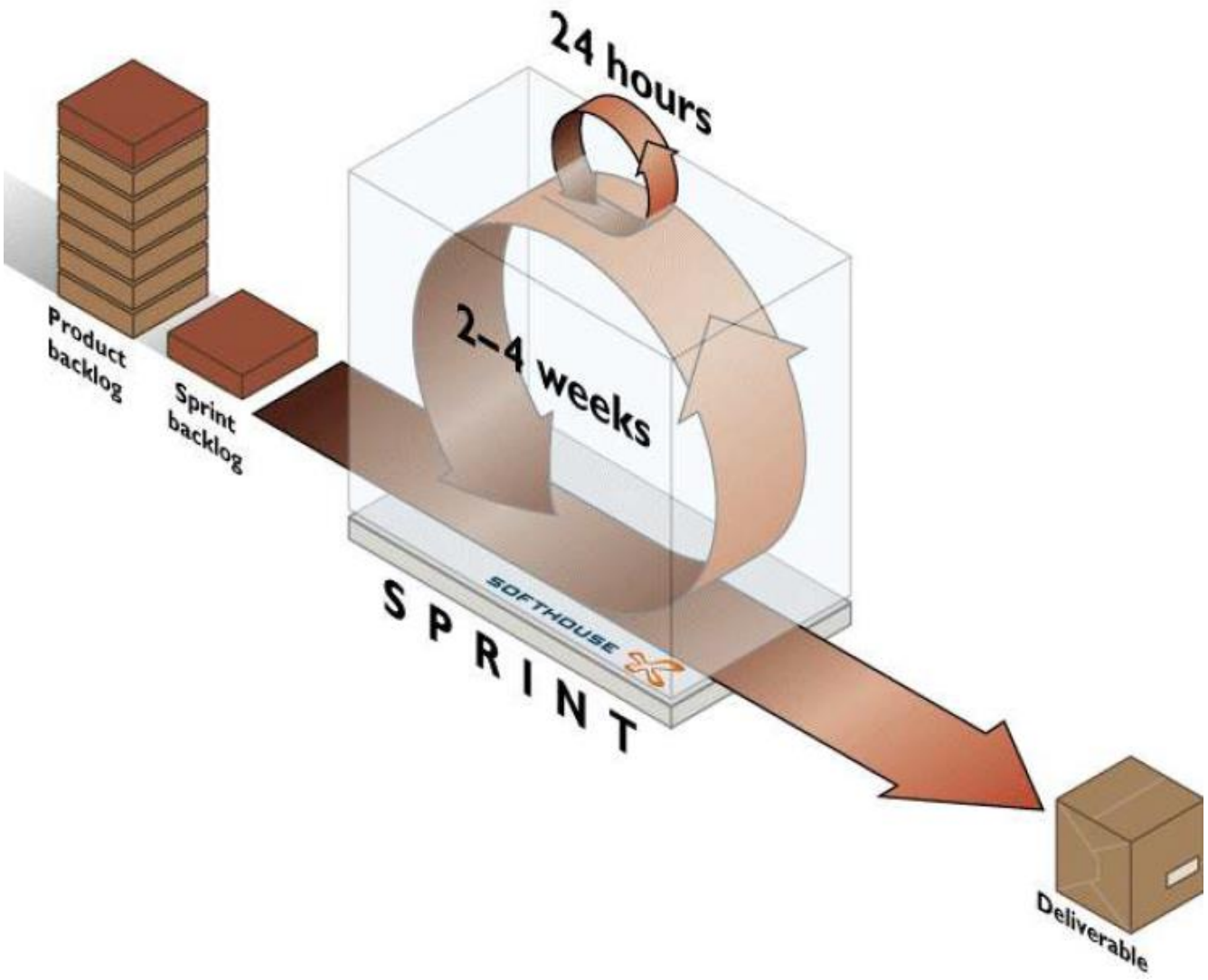
Понимание, разделяемое всеми
- Простота проектирования (Simple design)
- Метафора системы
- Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)
- Стандарт оформления кода (Coding standard or Coding conventions)

Социальная защищённость программиста (Programmer welfare):
- 40-часовая рабочая неделя (Sustainable pace, Forty-hour week)

XP (1999)
Agile

Методология Scrum

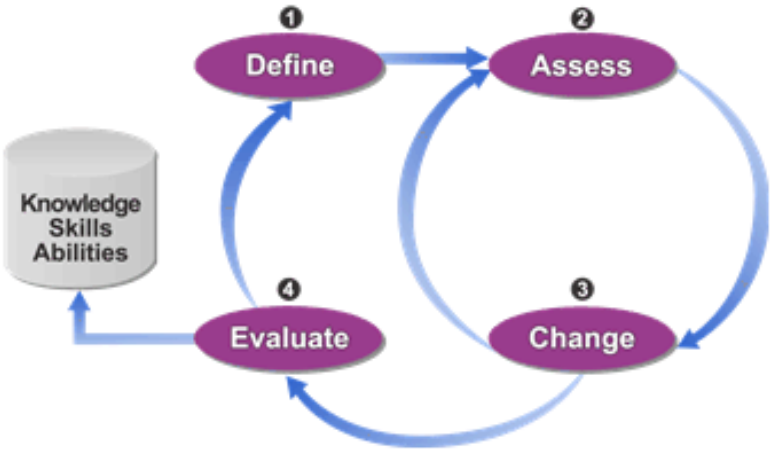
Scrum (/skrʌm/[1][2]; англ. *scrum* «схватка») — методология гибкой разработки ПО. Методология делает акцент на качественном контроле процесса разработки.



Scrum (1986)
Agile

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.



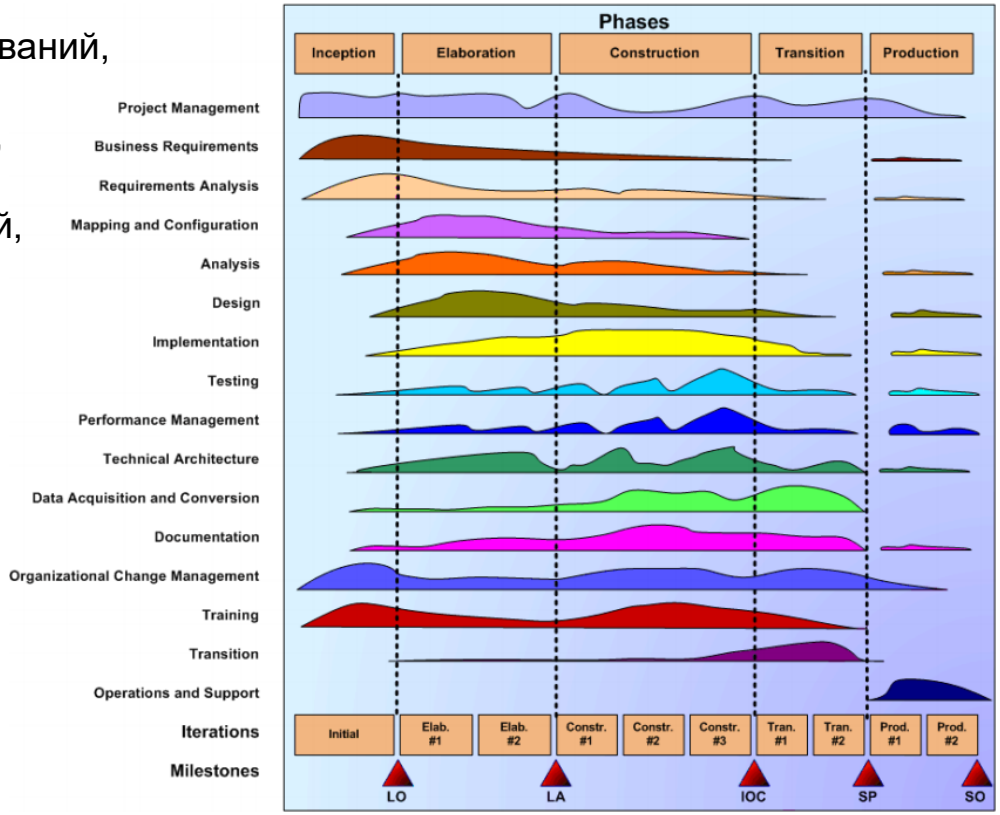
MSF (1994)
Microsoft

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

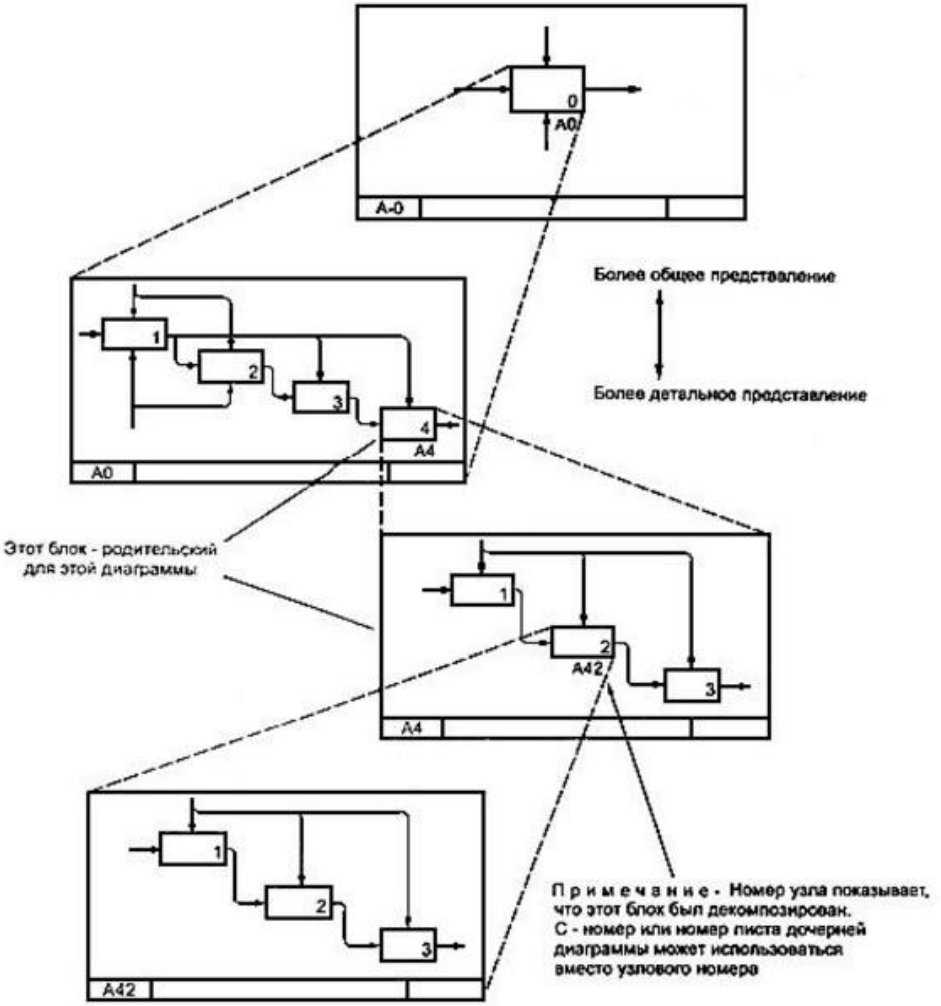
RD - Определение производственных требований,
ES - Исследование существующих систем,
TA - Определение технической архитектуры,
DB - Проектирование и построение БД,
MD - Проектирование и реализация модулей,
CV - Конвертирование данных,
DO - Документирование,
TE - Тестирование,
TR - Обучение,
TS - Переход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



OUM (2006) Oracle

Методология SADT

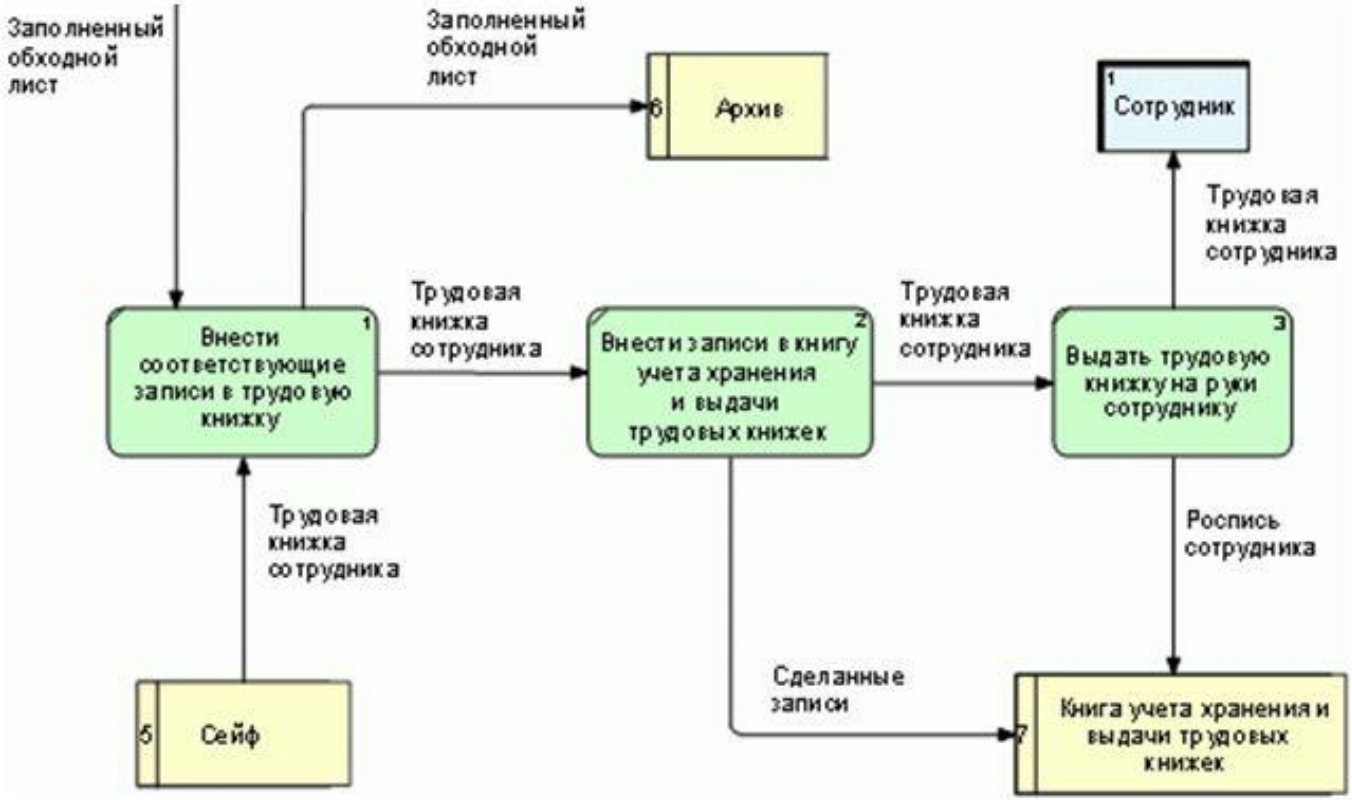
Р 50.1.028-2001 Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования



Р 50.1.028-2001
BPWin, RAMUS

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.



DFD (1970)
BPWin, RAMUS