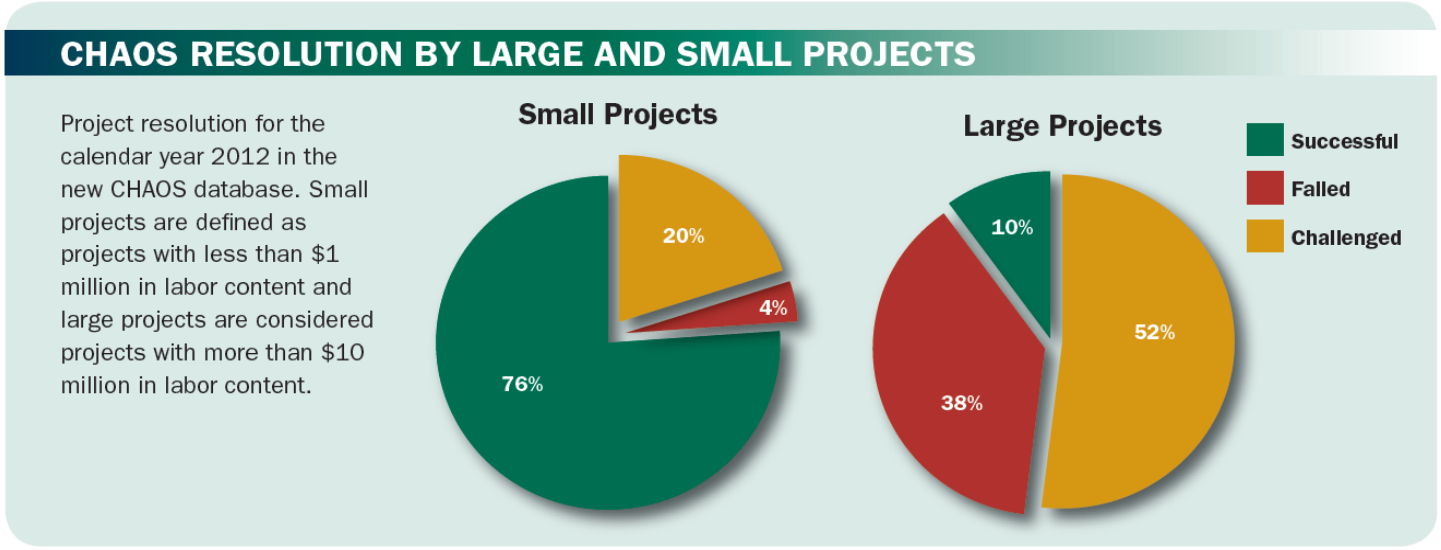
Студия информационных технологий
WebMax.BY

УПРАВЛЕНИЕ IT-ПРОЕКТАМИ
ПОНЯТИЯ, МЕТОДЫ И ПРОЦЕССЫ

www.webmax.by

Успешность IT-проектов

	2004	2006	2008	2010	2012	2013
УСПЕШНЫЕ	29%	35%	32%	37%	39%	36%
ПРОВАЛЬНЫЕ	18%	19%	24%	21%	18%	16%
СПОРНЫЕ	53%	46%	44%	42%	43%	48%



*По данным **Standish Group** в 2012 году*

Составляющие менеджмента

- **Менеджмент** (англ. *management*) – управление, руководство, администрирование, дирекция, регулирование

Составляющие

Крупная IT-компания одновременно ставит и решает комплекс взаимосвязанных задач, для чего создается несколько подсистем в системе менеджмента:

- Менеджмент **программной инженерии**:
  - Управление жизненным циклом программного продукта
  - Управление персоналом
  - Управление качеством
- Маркетинг
- Финансовый менеджмент
- Стратегический менеджмент
- Инвестиционный менеджмент
- Инновационный менеджмент
- Информационный менеджмент
- Экологический менеджмент

Программная инженерия

- ***Программная инженерия*** **есть применение определенного систематического измеримого подхода при разработке, эксплуатации и поддержке программного обеспечения** [1].

Термин *software engineering* впервые появился в названии конференции НАТО, состоявшейся в Германии в 1968 году и посвященной так называемому кризису программного обеспечения. С 1990-го по 1995 год велась работа над международным стандартом, который должен был дать единое представление о процессах разработки программного обеспечения. В результате был выпущен стандарт ISO/IEC 12207 [2].

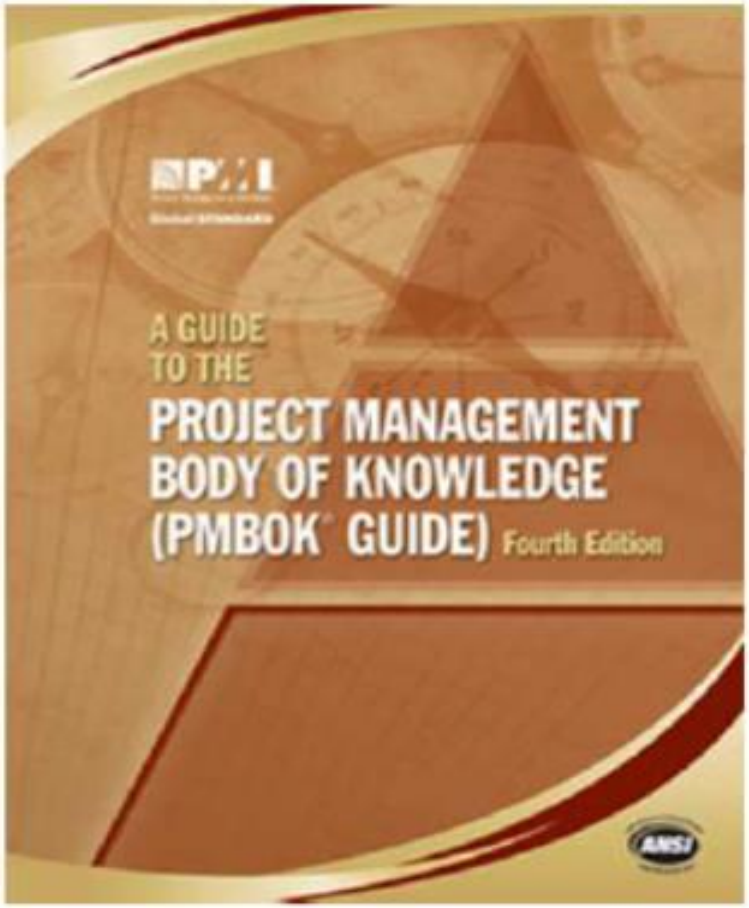
Центры стандартизации

Наиболее известные центры компетенции:

- PMI, Project Management Institute, PMBOK — американский национальный стандарт ANSI/PMI 99-001-2004.
- IPMA, International Project Management Association. В России — СОВНЕТ.

ЧАСТЬ I

ТРАДИЦИОННЫЙ ПОДХОД к управлению проектом



Основные понятия

- ***Проект*** *(Project)* **— это уникальный комплекс взаимосвязанных мероприятий, направленных на достижение конкретной цели при определенных требованиях к срокам, бюджету и характеристикам ожидаемых результатов.**
- ***Управление проектами*** *(Project Management)* **— это процесс планирования, организации и контроля за состоянием задач и ресурсов проекта, направленный на своевременное достижение цели проекта при балансировании между объемом работ, ресурсами (такими как деньги, труд, материалы, энергия, пространство и др.), временем, качеством и рисками.**

Основные понятия

Любой проект в процессе своей реализации проходит различные стадии и фазы, называемые в совокупности ***жизненным циклом проекта***.

Жизненный цикл программного обеспечения (ПО) — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации[1].

Для реализации различных функций управления проектом необходимы действия, которые в дальнейшем именуются ***процессами*** управления проектами.

Логика, цели и критерии

Управление проектами должно подчиняется определенной ***логике***, которая связывает между собой различные области знаний и процессы управления проектами.

У каждого проекта обязательно имеются *одна или несколько* ***целей***.

Достижение целей проекта может быть реализовано различными способами. Для сравнения этих способов необходимы ***критерии*** успешности достижения поставленных целей.

Логика, цели и критерии

Основные критерии оценки различных вариантов исполнения проекта: ***сроки и стоимость*** достижения результатов.

При этом запланированные ***цели и качество*** служат основными ограничениями при рассмотрении и оценке различных вариантов.

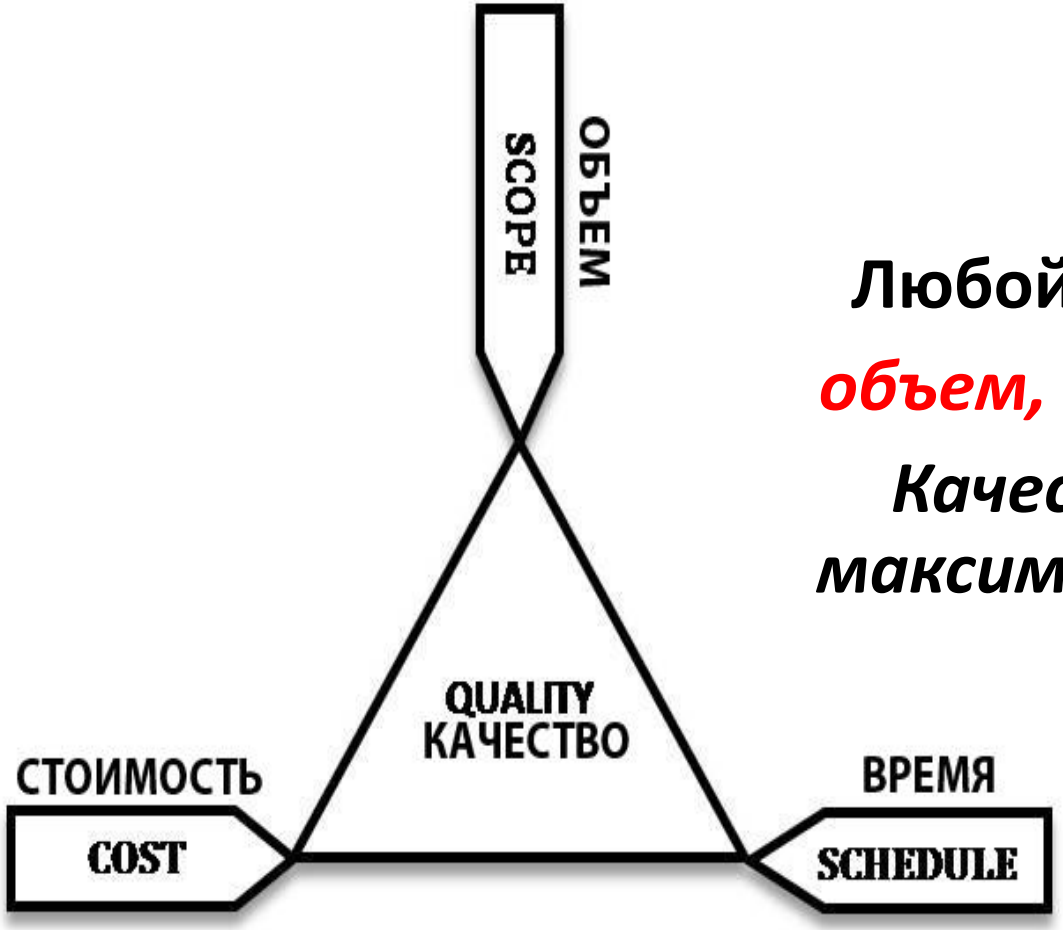
Для управления проектами необходимы ***рычаги***. К основным рычагам управления можно отнести ***ресурсы*** проекта и используемые ***технологии***.

Логика Управления Проектами

Кроме этих основных существуют и вспомогательные средства, предназначенные для управления основными.

К таким средствам можно отнести контракты и организацию работ.

Тройственная ограниченность



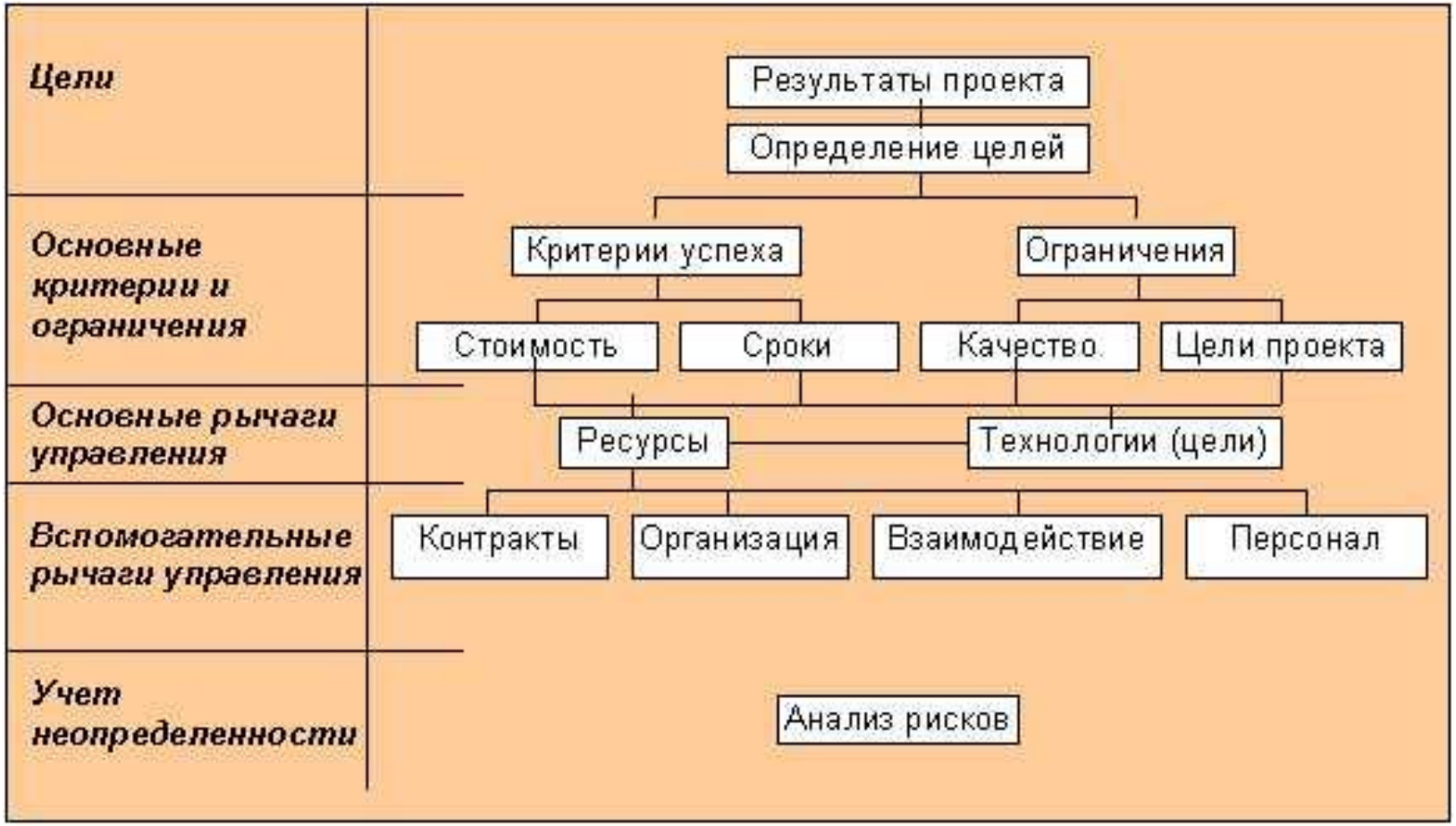
Любой проект ограничен:

объем, время, стоимость

Качество должно быть максимально-возможным

Рис.1. Треугольник управления проектами

Логика Управления Проектами



Жизненный цикл проекта

Организации, выполняющие проекты, обычно разбивают их на несколько стадий и фаз. Вся совокупность фаз носит общее название *жизненный цикл проекта*.

Каждая фаза характеризуется достижением одного или нескольких результатов. Результат - это *измеримый* продукт работы.

Каждая фаза обычно разбивается на отдельные работы, чтобы обеспечить наилучшую управляемость.

Жизненный цикл проекта

Жизненному циклу IT-проектов, обычно, присущи следующие особенности:

- Стоимость и число участников на старте невелики, возрастают к концу и резко уменьшаются перед завершением проекта;

- Вероятность успешного завершения проекта на старте наименьшая, но возрастает по мере исполнения проекта;

- Способность заказчика повлиять на результаты и стоимость проекта наивысшая на старте и уменьшается по мере исполнения проекта.

Ключевые Участники проекта (традиционный подход)

- ***Менеджер проекта*** - лицо, ответственное за управление проектом;
- ***Потребитель*** - лицо или организация, использующая продукт проекта;
- ***Исполняющая организация*** (или исполнитель) - организация, чьи сотрудники непосредственно вовлечены в исполнение проекта;
- ***Инвестор*** - лицо, группа или организация, предоставляющая финансовые ресурсы для исполнения проекта.

Процессы Управления Проектами

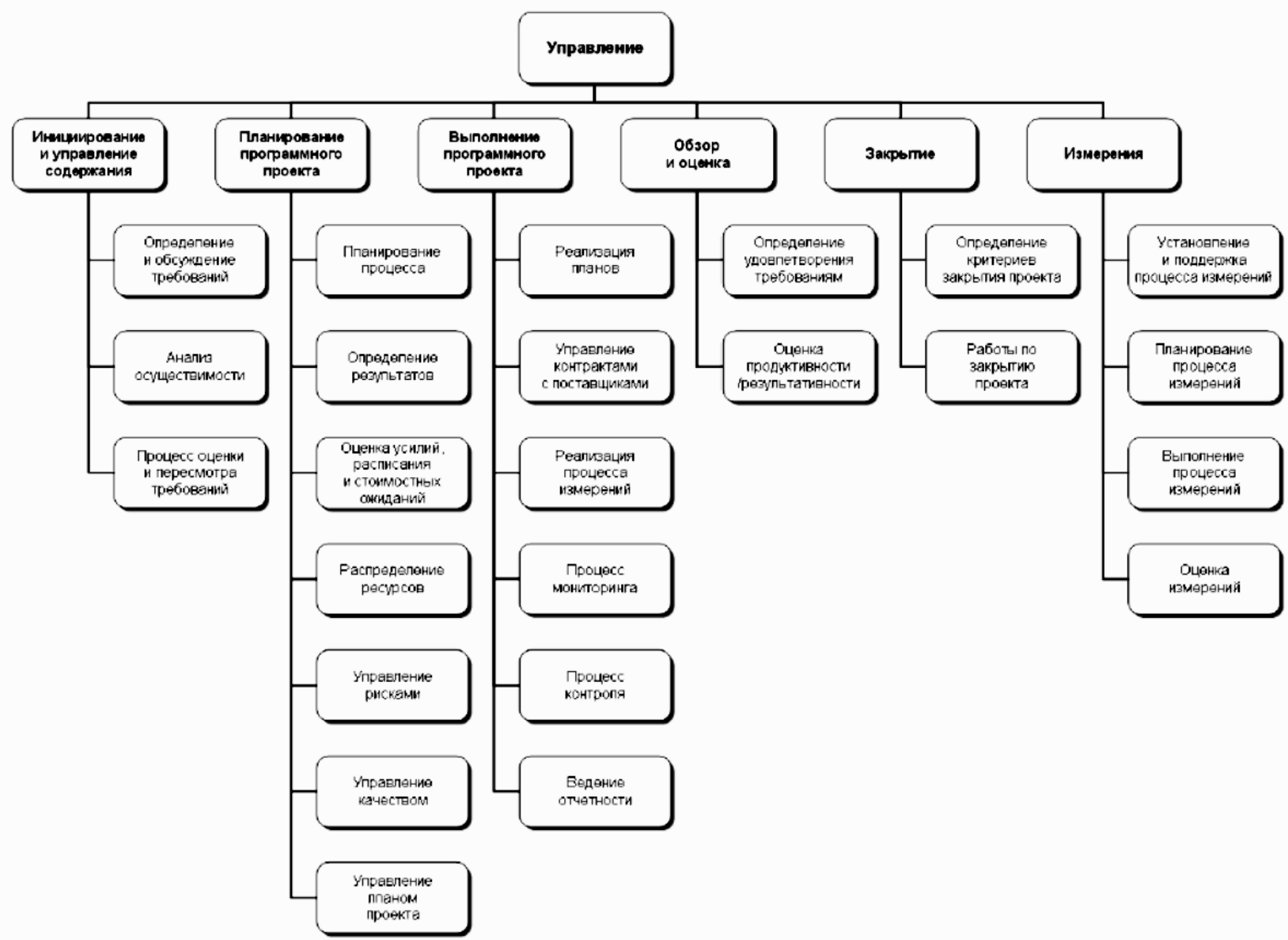
Проект состоит из процессов. ***Процесс*** **-- это совокупность действий, приносящая результат.**

Процессы управления проектами могут быть разбиты на шесть основных групп:

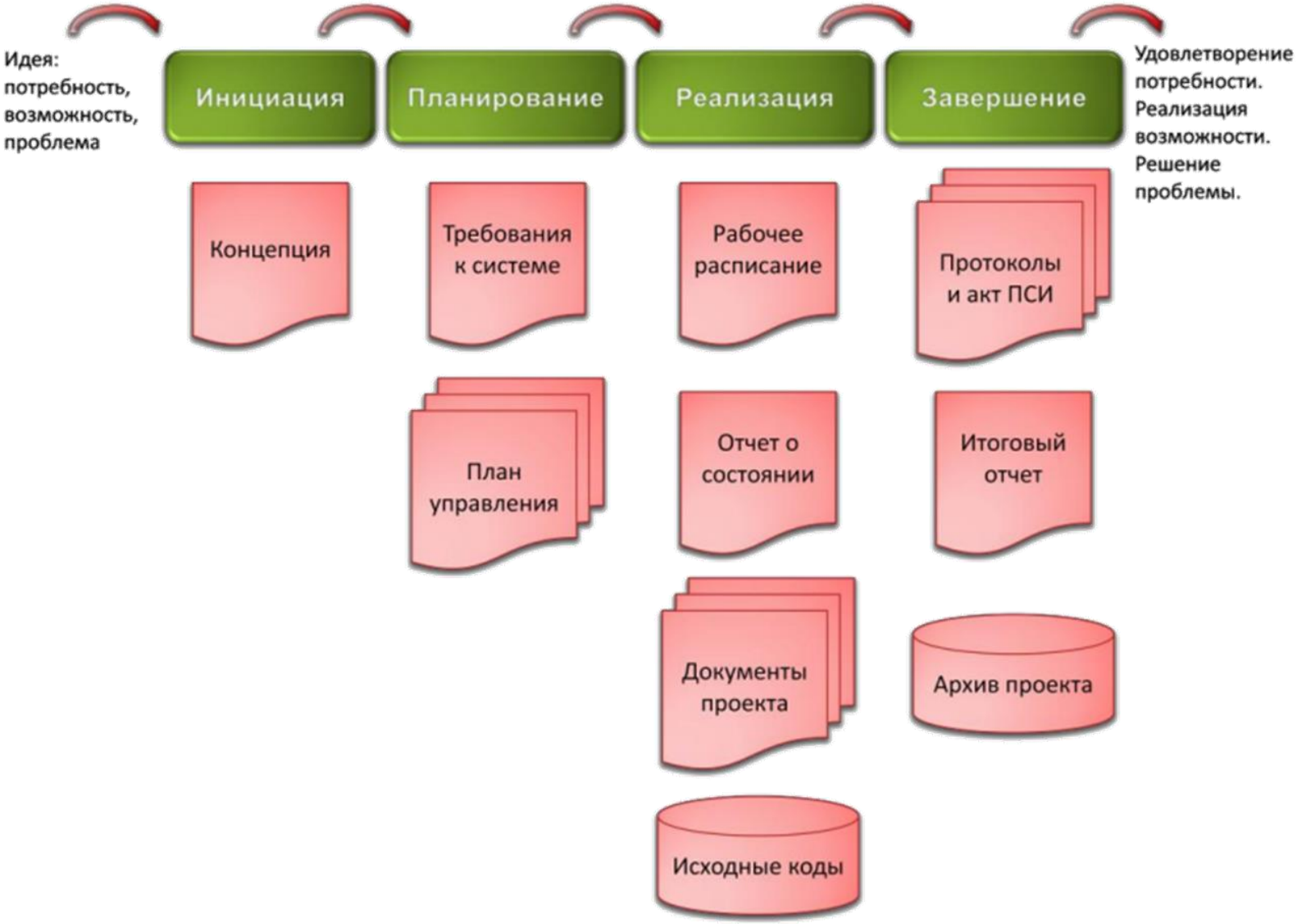
- ***процессы инициации*** **--** принятие решения о начале выполнения проекта;
- ***процессы планирования*** -- определение целей и критериев успеха проекта и разработка рабочих схем их достижения;
- ***процессы исполнения*** -- координация людей и других ресурсов для выполнения плана;
- ***процессы анализа*** -- определение соответствия плана и исполнения проекта поставленным целям и критериям и принятие решений о корректирующих воздействий;
- ***процессы управления*** -- определение корректирующих воздействий, их согласование, утверждение и применение;
- ***процессы завершения*** -- формализация выполнения проекта и подведение его к упорядоченному финалу.

Процессы Управления Проектами

Процессы Управления Проектами



Жизненный цикл проекта и основные документы



Инициация проекта

- Описание проблемы
- Формулировка концепции:
  - ❑ Разработка концепции (Vision);
  - ❑ Границы системы (Scope);
  - ❑ Ключевые пользовательские функции (User Requirements или Business Use Cases);
  - ❑ Экономическое обоснование (Business Case):
  - ❑ Является ли проект прибыльным?
  - ❑ Является ли он реальным?
- Предложить возможное решение (бизнес-архитектуру);
- Перечень рисков;
- Основные участники проекта;
- План работ (диаграмма Ганта);
- Смета работ

Инициация проекта

Для чего нужна КОНЦЕПЦИЯ ПРОЕКТА (Vision)?

Концепция проекта разрабатывается на основе анализа потребностей бизнеса. Главная функция документа — подтверждение и согласование единого видения целей, задач и результатов всеми участниками проекта. Концепция определяет *что и зачем* делается в проекте.

Концепция проекта это ключевой документ, который используется для принятия решений в ходе всего проекта, а также на фазе приемки — для подтверждения результата. Она содержит, как правило, следующие разделы:

Инициация проекта

Стандартное содержание документа Концепции

- Название проекта
- Цели проекта
- Результаты проекта
- Допущения и ограничения
- Ключевые участники и заинтересованные стороны
- Ресурсы проекта
- Сроки
- Риски
- Критерии приемки
- Обоснование полезности проекта

Планирование проекта

Что нужно планировать?
Во что это выливается?

- Уточнение содержания и состава работ
- Планирование организационной структуры
- Планирование управления конфигурациями
- Планирование управления качеством
- Управление рисками, оценка и контроль проекта

Планирование проекта

План управления проектом

ВСПОМОГАТЕЛЬНЫЕ ПЛАНЫ

- План управления содержанием проекта (Scope Management Plan)
- Календарный план (Schedule Plan)
- План управления стоимостью (Cost Management Plan)
- План управления качеством (Quality Management Plan)
- План управления персоналом (Staffing Management Plan)
- План управления взаимодействием (Communication Management Plan)
- План управления рисками (Risk Management Plan)
- План управления поставками (Procurement Management Plan)
- План усовершенствования процессов (Process Improvement Plan)

ДРУГИЕ ВЫХОДЫ ПРОЦЕССОВ ПЛАНИРОВАНИЯ

- Содержание проекта (Scope Statement)
- Состав операций (Activity List)
- Иерархическая структура работ (WBS)
- Сетевая диаграмма (Project network diagram)
- Расписание проекта (Project Schedule)
- Требования к ресурсам (Resourse Requirements)
- Базовый план по стоимости (Cost Baseline)
- Матрица ответственности (Responsibility Matrix)
- План усовершенствования процессов (Process Improvement Plan)

Планирование проекта

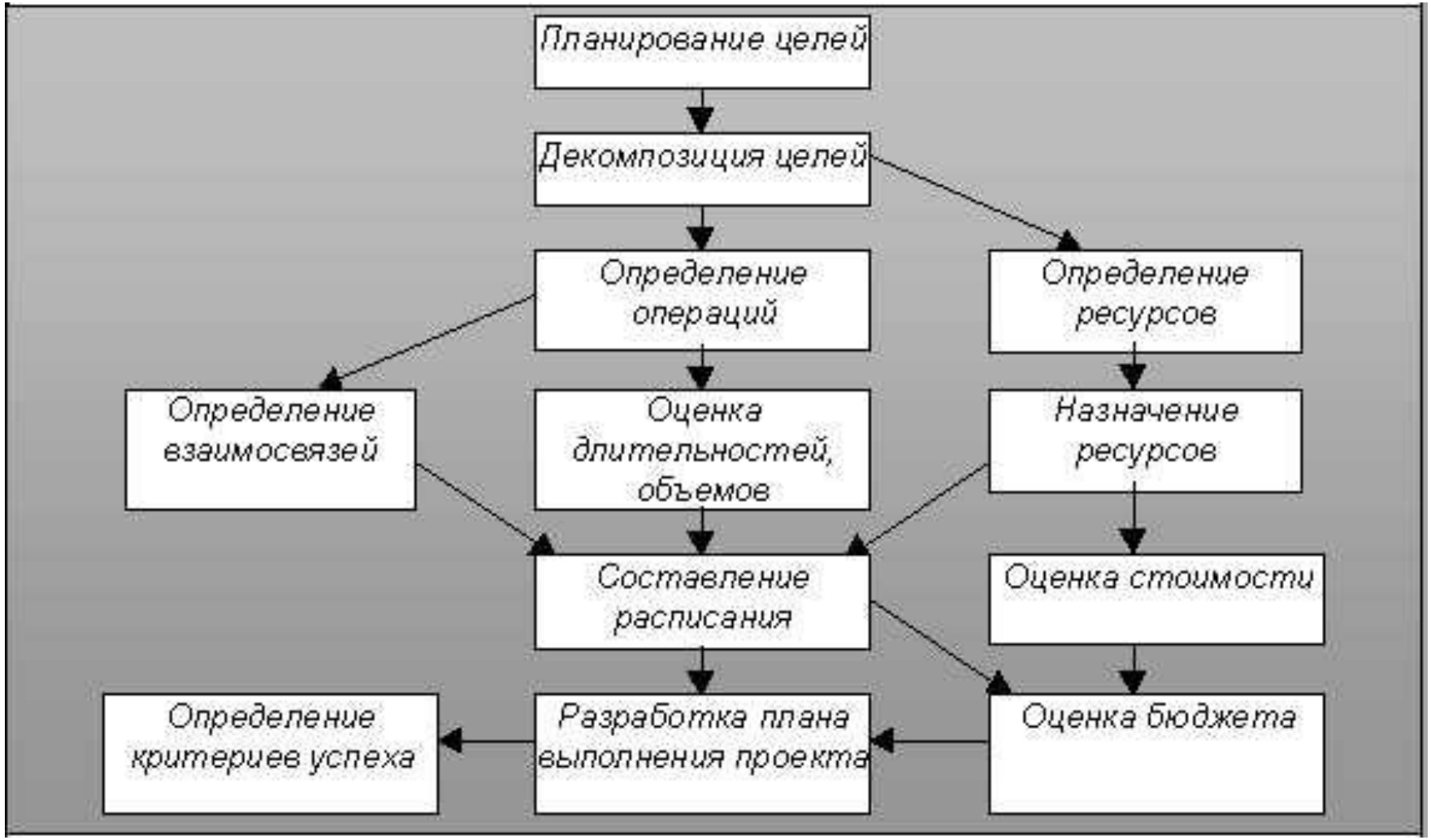
О возможном весе документации…

«Вес документации по проекту разработки самолета Боинг-747» ПРЕВЫСИЛ !!! вес самого самолета»

Дж.Фокс "Программное обеспечение и его разработка".

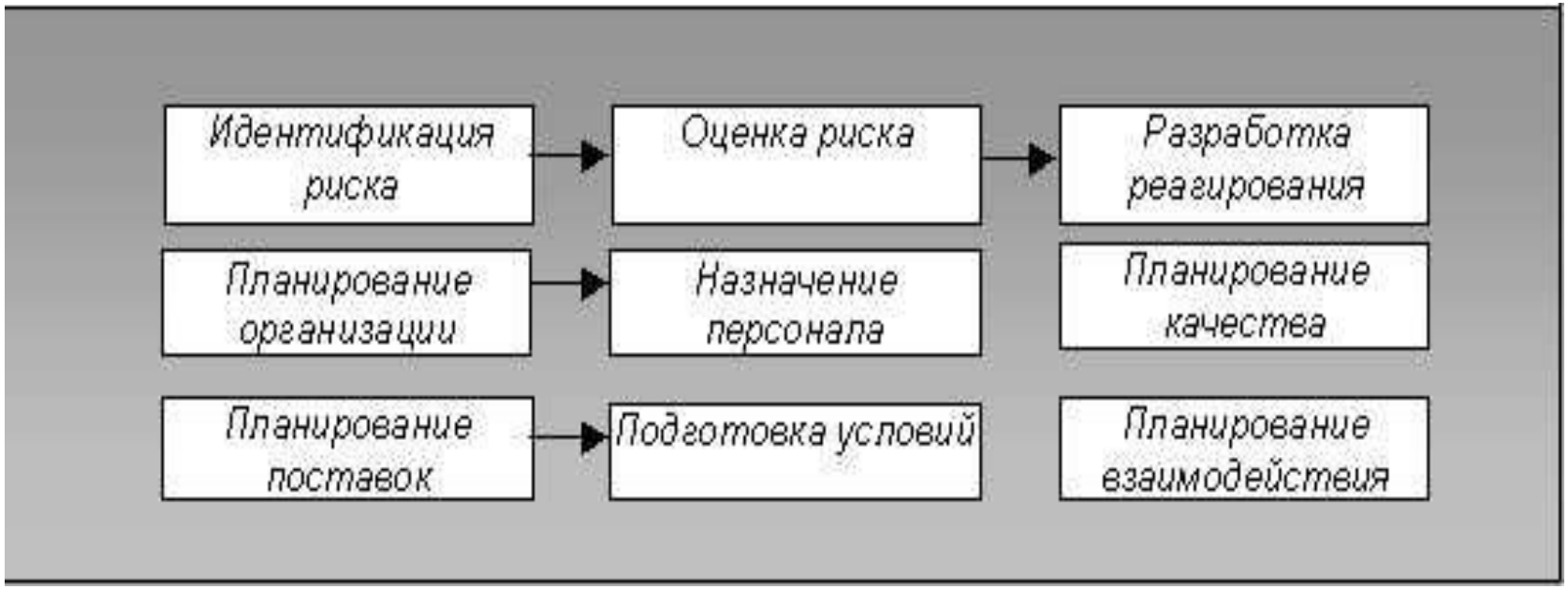
Планирование проекта

Основные процессы планирования



Планирование проекта

Вспомогательные процессы планирования



Планирование проекта

Методы планирования

- **Program (Project) Evaluation and Review Technique** (сокращенно **PERT**) — техника оценки и анализа программ (проектов), в особенности, анализа времени, требуемого для выполнения каждой отдельной задачи, а также определение минимального необходимого времени для выполнения всего проекта.
- Самая известная часть PERT — это диаграммы взаимосвязей работ и событий. Предлагает использовать диаграммы-графы с работами на узлах, с работами на стрелках (сетевые графики), а также диаграммы Ганта.
- Вероятностная оценка трудоёмкости проекта (диапазон неопределённости характеризуется оптимистической, пессимистической и наиболее вероятной оценками трудозатрат).

Планирование проекта

Создается *иерархическая структура работ (ИСР) (**Work /Breakdown Structure, WBS**)** — ориентированная на результат иерархическая декомпозиция работ, выполняемых командой проекта для достижения целей проекта и необходимых результатов.

С ее помощью структурируется и определяется все содержание проекта. Каждый следующий уровень иерархии отражает более детальное определение элементов проекта.

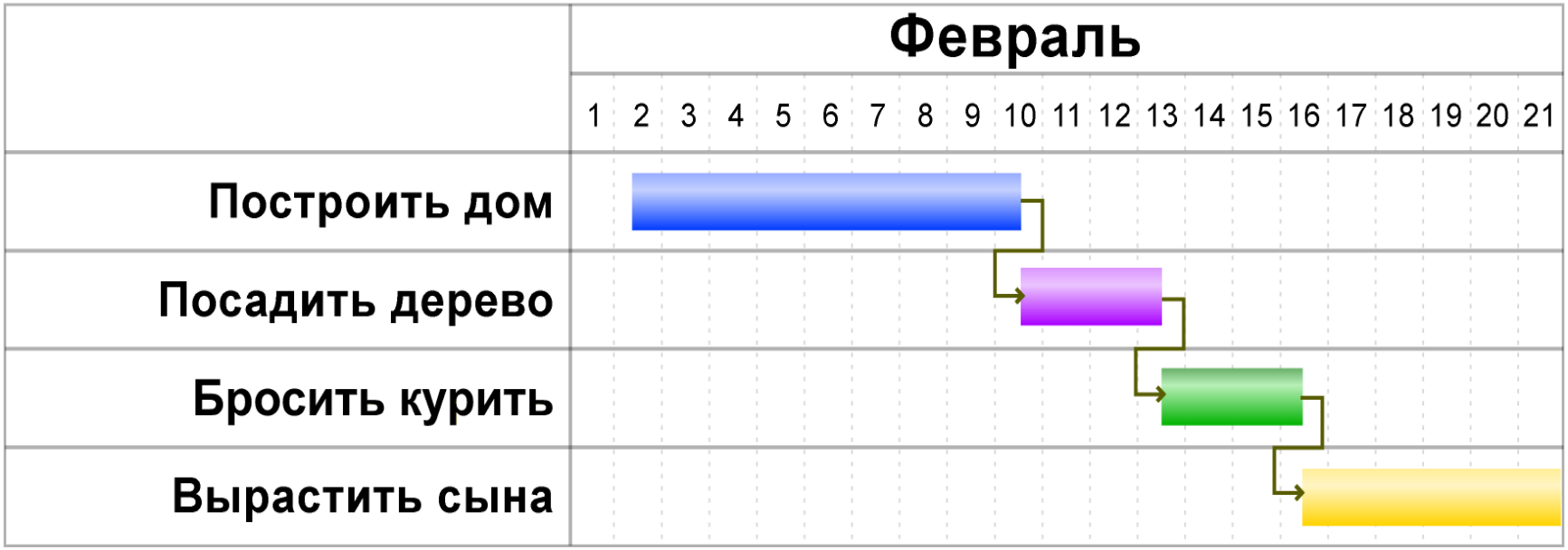
**Чаще всего WBS создаётся в виде диаграммы Ганта*

Планирование проекта

- **Диагра́мма Га́нта** (англ. *Gantt chart*, также **ленточная диаграмма**, **график Ганта**) — это популярный тип столбчатых диаграмм (гистограмм), который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов.
- Первый формат диаграммы был разработан Генри Л. Гантом в 1910 году.

Планирование проекта

Диаграмма Ганта



Каждая полоса на диаграмме представляет отдельную задачу в составе проекта (вид работы), её концы — моменты начала и завершения работы, её протяженность — длительность работы. Вертикальной осью диаграммы служит перечень задач. Кроме того, на диаграмме могут быть отмечены совокупные задачи, проценты завершения, указатели последовательности и зависимости работ, метки ключевых моментов (вехи), метка текущего момента времени «Сегодня» и др.

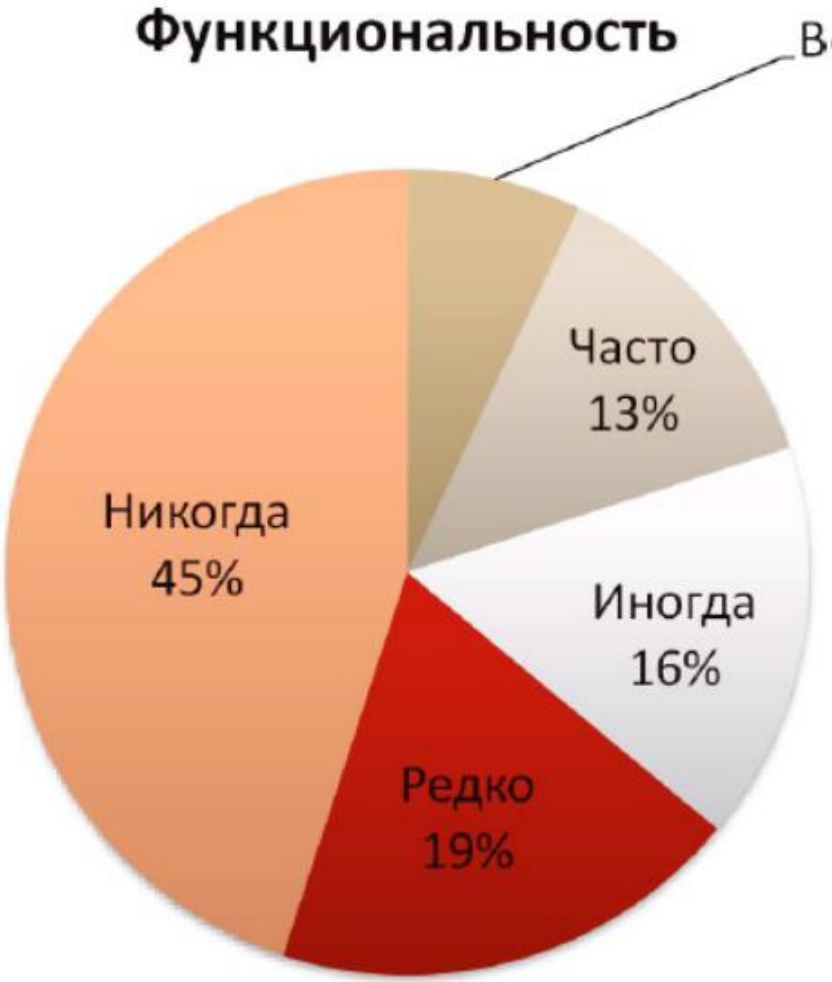
Планирование проекта

- **Модель Noriaki Kano**

Рассматривает 5 типов характеристик:

1. обязательные;
2. одномерные;
3. привлекательные;
4. неважные;
5. нежелательные

Планирование проекта



Функциональность, используемая в типичной системе

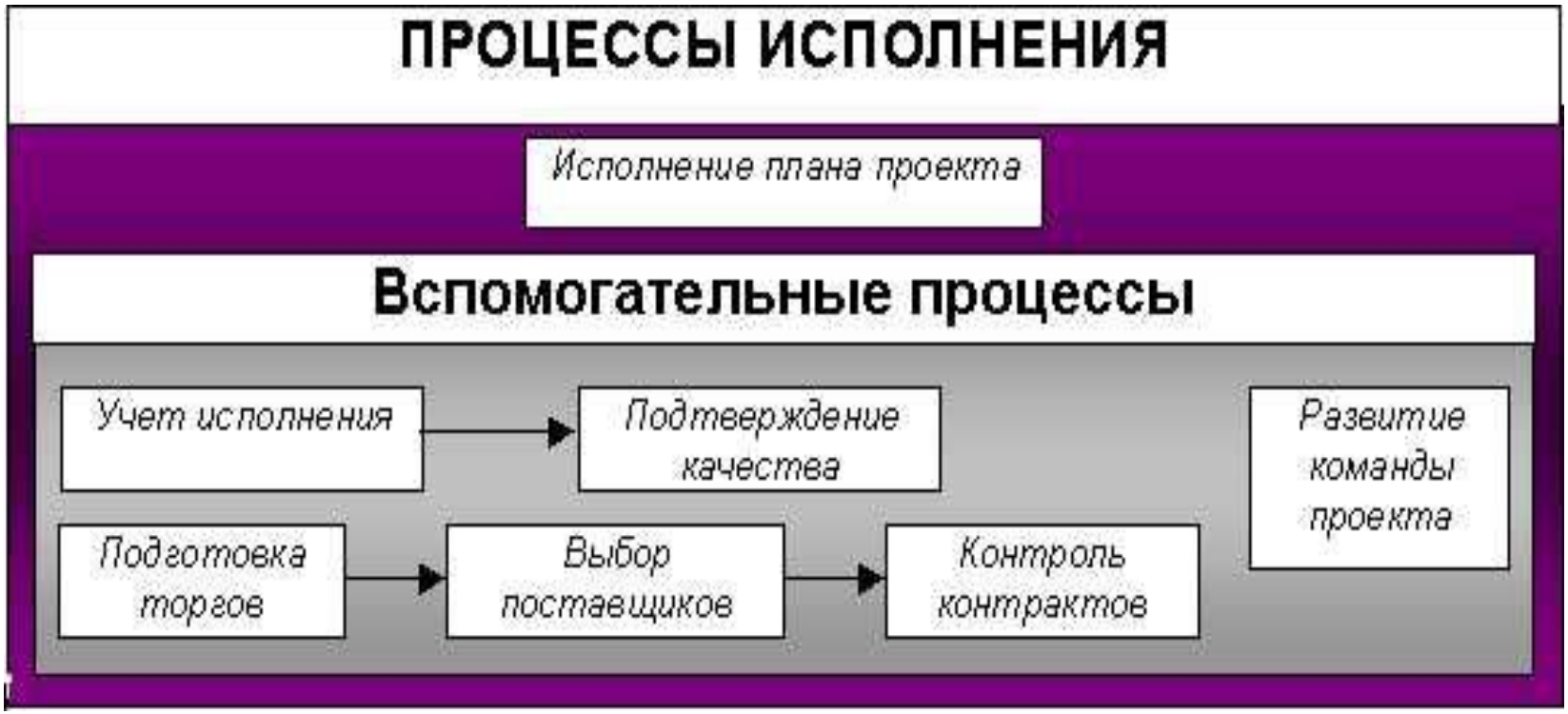
Standish Group Study Report

Планирование проекта

Что не так?

- Длинные циклы разработки документов проекта;
- Документы статичны и официальны;
- Содержание собирается по кусочкам;
- Авторы документов часто изолированы от разработчиков;
- Возникают бесконечные итерационные цепи: документы пишутся-согласуются-корректируются.

Исполнение проекта



Под *исполнением* подразумеваются процессы реализации составленного плана. Исполнение проекта должно регулярно измеряться и анализироваться для того, чтобы выявить отклонения от намеченного плана и оценить их влияние на проект.

Анализ исполнения проекта

- ***Анализ плана*** означает определение того, удовлетворяет ли составленный план исполнения проекта предъявляемым к проекту требованиям и ожиданиям участников проекта.

Управление проектом

- ***Управление исполнением*** **проекта - это определение и применение необходимых управляющих воздействий с целью успешной реализации проекта.**

Управление проектом

Управление рисками

10 наиболее распространенных рисков программного проекта (по Барри Боэму):

1. Дефицит специалистов.
2. Нереалистичные сроки и бюджет.
3. Реализация несоответствующей функциональности.
4. Разработка неправильного пользовательского интерфейса.
5. "Золотая сервировка", ненужная оптимизация и оттачивание деталей.

Управление проектом

Управление рисками

6. Непрекращающийся поток изменений.

7. Нехватка информации о внешних компонентах, определяющих окружение системы или вовлеченных в интеграцию.

8. Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами.

9. Недостаточная производительность получаемой системы.

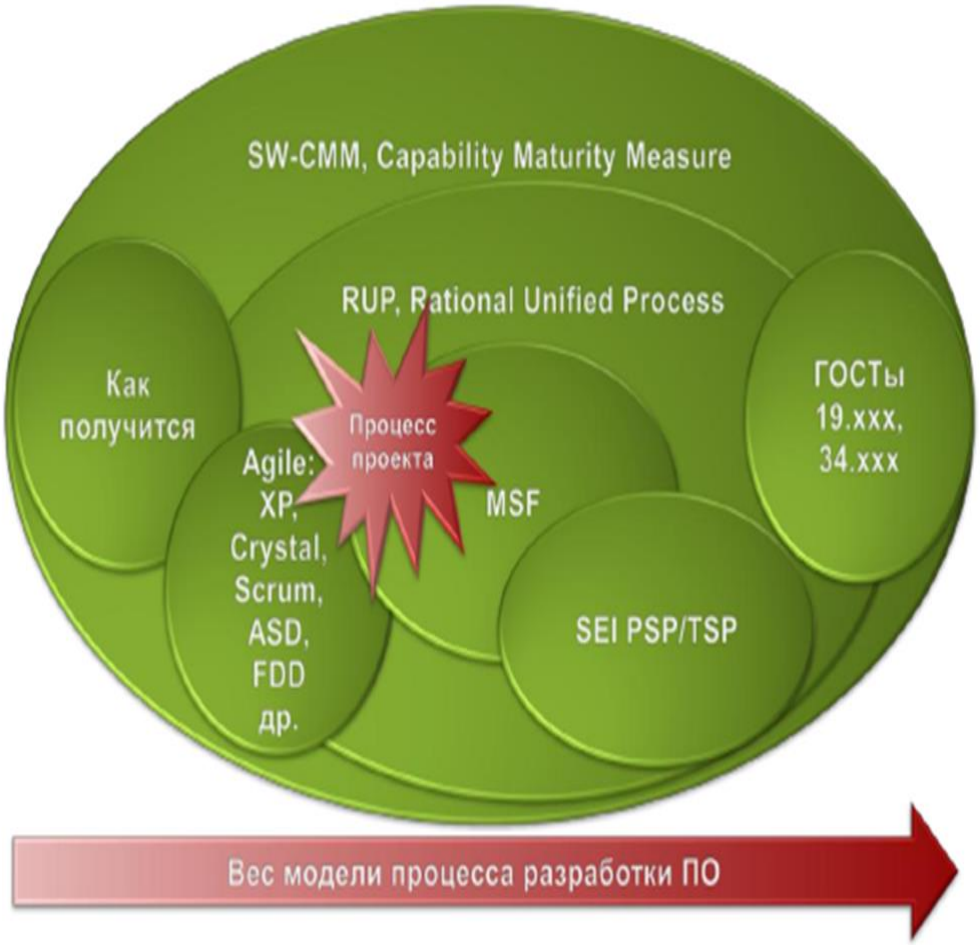
10. "Разрыв" в квалификации специалистов разных областей знаний.

Завершение проекта

- ***Процессы завершения*** **- формализация выполнения проекта и подведение его к упорядоченному финалу.**

Стандарты и модели ЖЦ

Модели (методологии) процессов разработки ПО принято классифицировать по «весу» — количеству формализованных процессов (большинство процессов или только основные) и детальности их регламентации. Чем больше процессов документировано, чем более детально они описаны, тем больше «вес» модели.



Стандарты и модели ЖЦ

- **СТБ ИСО 9001-2009**
- **CMMI**
- **Отраслевые стандарты**
- **ITIL**
- **PMBOOK** (Guide to the Project Management Body of Knowledge) - это проект «Project Management Institute», вобравший в себя накопленные знания в области управления проектами.

ЕСКД (ГОСТ) и ISO

- **ГОСТ 19 «Единая система программной документации»**
- **ГОСТ 34 «Стандарты на разработку и сопровождение автоматизированных систем»**

Ориентированы на последовательный подход к разработке ПО. Разработка в соответствии с этими стандартами проводится по этапам, каждый из которых предполагает выполнение строго определенных работ, и завершается выпуском достаточно большого числа весьма формализованных и обширных документов. Строгое следование гостам приводит к водопадному подходу и требует высокой степени формализованности разработки.

ЕСКД (ГОСТ) и ISO

- **СТБ ИСО 9001-2009**

Предусматривают, с одной стороны, построение организационной системы "сверху вниз": от целей предприятия и его политики - к организационной структуре и формированию бизнес процессов, и с другой - итеративное развитие организационной системы через механизмы измерения и улучшения.

Упрощенно "качество", согласно серии стандартов ISO 9000, это ситуация, при которой потребители получают от производителя продукцию соответствующую их прямым требованиям и подспудным ожиданиям.

ЕСКД (ГОСТ) и ISO

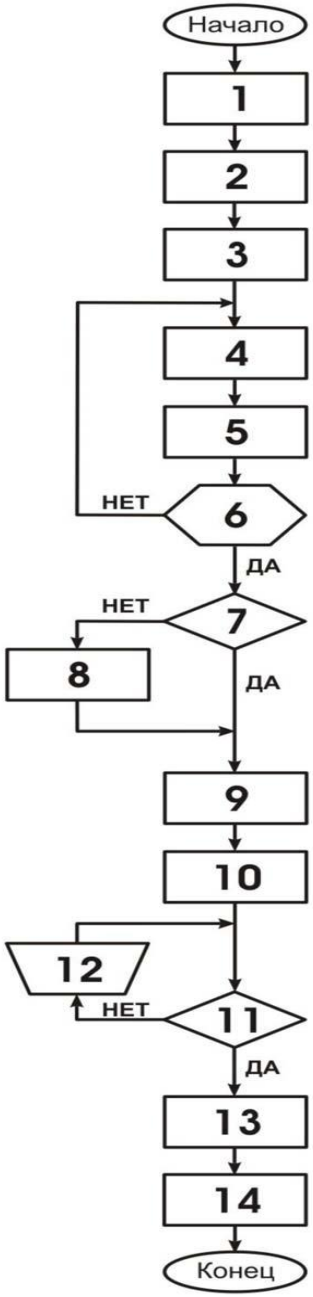
Управление качеством, в соответствии с ISO 9000, предполагает применение т.н. "процессного подхода", когда моделируется и внедряется наиболее оптимальная цепь "преобразований-процессов", гарантирующая, что потребности потребителей воспринимаются производителем и воплощаются в любой продукт без искажений.

Такие понятия, как процессный подход, анализ и измерения, совершенствование процессов, введены в версиях ISO 9000 с 2000 года и ранее в них отсутствовали.

Стандарты и модели

Пример водопадной модели разработки в комбинации с итерационными циклами

Модель разрабатывалась в рамках сертификации по ISO 9001



Карта процесса «Разработка и тестирование программного обеспечения»

Этап выполнения процесса	Ответственный	Результат
1. Анализ ТЗ на создание АСУ ТП и док-тов технического проекта системы. Определение требований к ПО, вкл. входные и выходные данные для разработки	Руководитель сектора ПО, ГИП	Спецификация входных и выходных данных для разработки ПО
2. Составление и утверждение плана разработки ПО по конкретному проекту с указанием сроков и исполнителей работ	Руководитель сектора ПО, ГИП, управляющий	Утвержденный план разработки ПО
3. Заимствование готовых программных компонентов из программного фонда	Руководитель сектора ПО, разработчик ПО	Программы и программная документация
4. Разработка недостающих программных компонентов ПО	Руководитель сектора ПО, разработчик ПО	Программы и программная документация
5. Разраб-ка программ тестовых испытаний и контрольных тестов	Разработчик ПО	Программы тестовых испытаний и контрольные тесты
6. Сборка, наладка и выполнение тестовых испытаний комплекта ПО. Несоответствия имеются?	Разработчик ПО	Перечень несоответствий
7. Оценка функциональности комплекта ПО (верификация). Комплект ПО соответствует треб-ниям ТЗ и ТНПА?	Руководитель сектора ПО, ГИП	Перечень несоответствий
8. Действия с несоответствующей пр-цией	В соответствии с СТП СМК 8.3.0-01 – 2009 «Управление несоответствующей продукцией»	
9. Утверждение результатов тестирования	Руководитель сектора ПО, ГИП	Решение о соответствии комплекта ПО треб-ниям
10. Разработка эксплуатационной документации по ПО	Разработчик ПО	Эксплуатационная документация в соотв. с ТЗ
11. Установка пакета ПО в сис-му. Проверка функционирования пакета ПО при комплексной отладке АСУ ТП (валидация)	Руководитель сектора ПО, ГИП	Перечень несоответствий
12. Анализ, выявление и устранение несоответствий ПО в процессе опыт. экспл-ции АСУ ТП. Корректировка программной док-ции	Разработчик ПО	Доработанные ПО и эксплуатационная док-ция по результатам валидации
13. Ввод ПО в промышленную экспл-цию и передача заказчику	Руководитель сектора ПО, ГИП	Акт ввода АСУ ТП в промышленную экспл-цию
14. Управление изменениями и сопровождение ПО	Руководитель сектора ПО, ГИП	Запросы на изменения. ПО с изменениями

Модель зрелости процессов CMMI

- **CMMI (Capability Maturity Model Integration)**

Это относительно новый стандарт в области менеджмента качества, который описывает модель зрелости процессов разработки программного обеспечения на предприятиях {3}.

Использование данной модели позволяет организации оценить эффективность бизнес-процессов, связанных с разработкой ПО, установить приоритетные направления их усовершенствования, а также внедрить данные усовершенствования.

Модель зрелости процессов CMMI

Модель CMMI, а точнее ее версия 1.1, появилась в марте 2002 года. Она внедрена Институтом программной инженерии Университета Карнеги-Меллона (Software Engineering Institute, SEI) специально для отрасли разработки программного обеспечения.

До этого в конце 80-х годов для ПО по заказу Министерства обороны США была разработана SW-CMM.

Основным компонентом модели CMMI является **область процессов**, определяющая цели и действия для их достижения.

Модель зрелости процессов CMMI

- **Область процессов** (Process Area) – набор связанных практик данной области, исполняются для достижения ряда целей, которые считаются важными в контексте улучшения процессов в данной области.

Важно понимать, что область процесса не является процессом. Один процесс может пересекаться с несколькими областями процесса, а одна область процесса может включать несколько процессов.

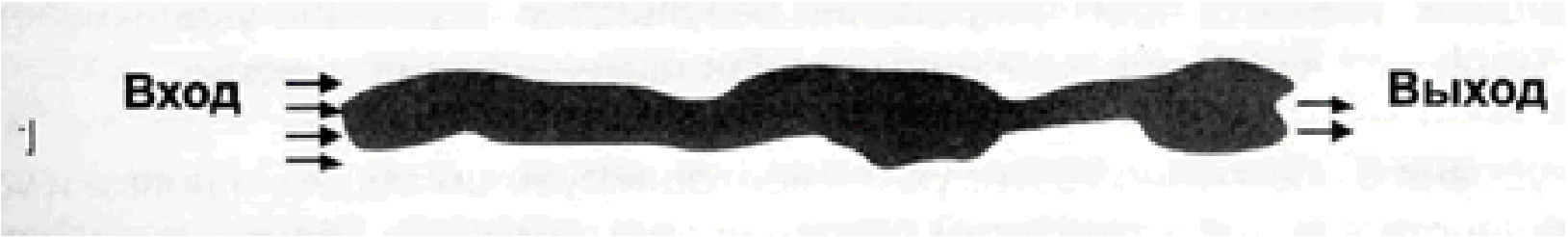
В ходе оценки организации определяется уровень ее работы, который характеризует способность организации управлять рисками и, следовательно, выполнять взятые на себя обязательства.

Модель зрелости процессов CMMI

- **Процессы первого уровня зрелости,** характеризуются хаотичностью, реактивностью, непредсказуемостью.

Несмотря на это, очень часто организации, находящиеся на данном этапе развития, производят довольно качественные продукты. При этом, как правило, превышается бюджет и время разработки данных продуктов. Качественные продукты данных организаций производятся не за счет устойчивых и отлаженных процессов, а благодаря титаническим усилиям отдельных личностей. На данном этапе очень тяжело предсказать производительность процессов, протекающих в организации.

Модель зрелости процессов CMMI

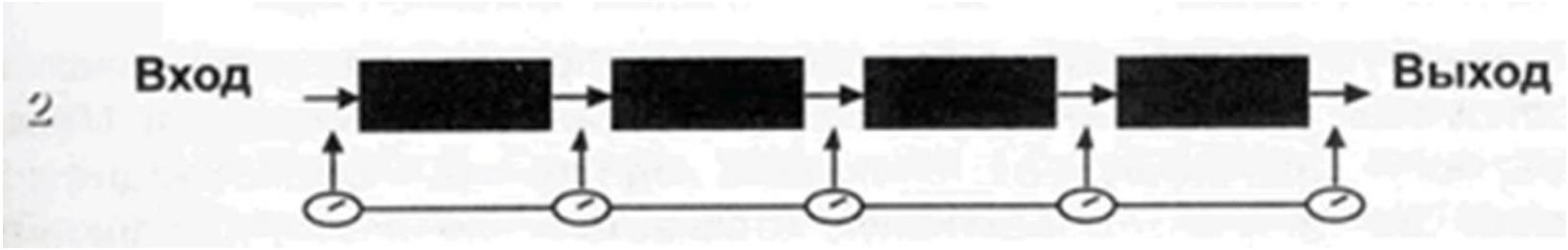


На уровне 1 производственный процесс (а вместе с ним и все процессы) представляется аморфной сущностью, практически «черным ящиком», представление о процессах очень ограниченное, чрезмерно много усилий тратится на выяснение статуса развития проекта и текущего хода работ.

Модель зрелости процессов CMMI

- **Уровень зрелости 2 – управляемый уровень**. На данном этапе основные процессы описаны, их, возможно, использовать неоднократно.

Проекты, выполняемые организацией, отвечают требованиям. Процессы управляемы, они планируются, выполняются, измеряются и контролируются. Однако они все же имеют некоторую долю реактивности в своей сущности.



Модель зрелости процессов CMMI

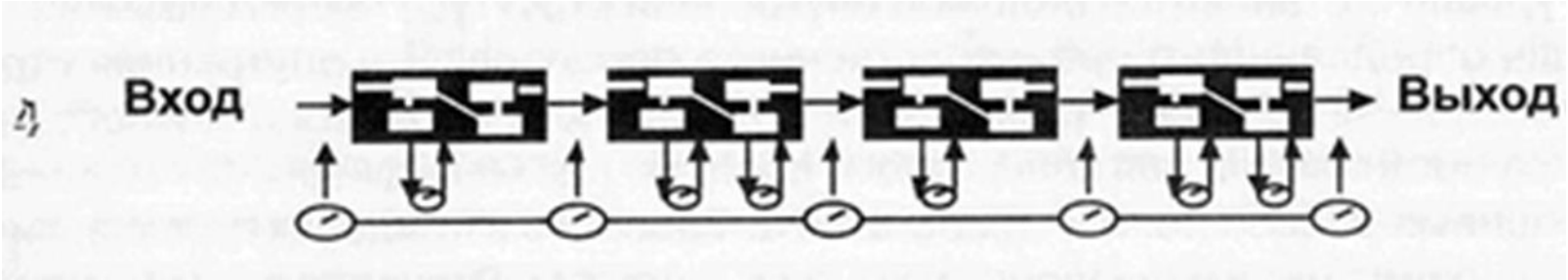
- **Уровень зрелости 3 – определенный уровень.** В этом случае процессы определены. Установлены стандарты в пределах организации. На данном этапе процессы описаны не на уровне отдельного проекта, а на уровне всей организации.

Присутствует более детальное описание всех процессов, в котором лучше раскрываются связи и зависимости, знание которых позволяет улучшить управление.

Модель зрелости процессов CMMI

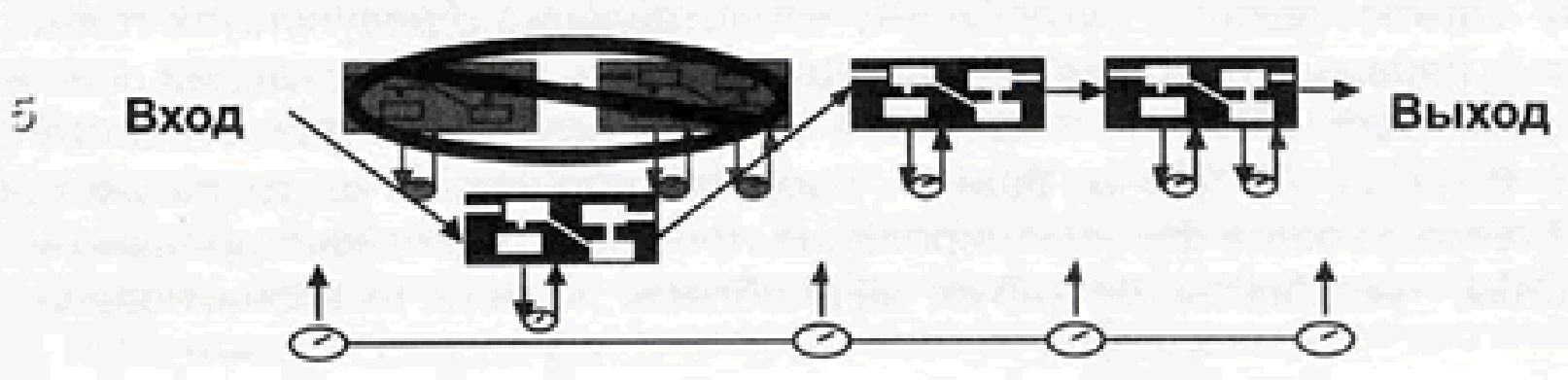
- **Уровень зрелости 4 – количественно-управляемый уровень.** На данном этапе достигнуты все цели предыдущих уровней. Выбраны субпрактики, которые при использовании статистических методов и других количественных техник позволяют контролировать качество выполнения процессов.

Самое главное отличие этого этапа от предыдущего заключается в предсказуемости эффективности процессов и возможности ею (эффективностью) управлять.



Модель зрелости процессов CMMI

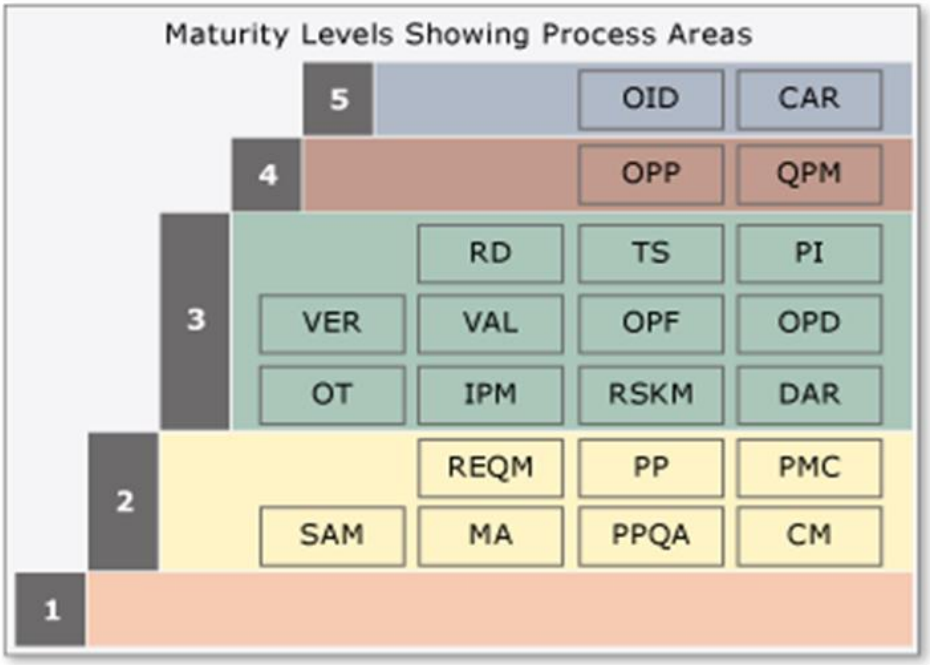
- **Уровень зрелости 5 – уровень постоянного улучшения (оптимизации) процессов.** На данном этапе мы имеем точные характеристики оценки эффективности бизнес процессов, что позволяет нам постоянно и эффективно улучшать бизнес процессы путем развития существующих методов и техник и внедрения новых.



Акроним	Область процесса
CAR	Анализ и разрешение причин
CM	Управление конфигурацией
DAR	Анализ и разрешение решений
IPM	Интегрированное управление проектами
MA	Измерения и анализ
OID	Организационные инновации и развертывание
OPD	Определение организационных процессов
OPF	Фокус организационных процессов
OPP	Производительность организационных процессов
OT	Organizational Training
PI	Интеграция продуктов
PMC	Мониторинг и контроль проектов
PP	Планирование проектов
PPQA	Контроль качества процессов и продуктов
QPM	Количественное управление проектами
RD	Определение требований
REQM	Управление требованиями
RSKM	Управление рисками
SAM	Управление соглашениями с поставщиками
TS	Техническое решение
VER	Проверка
VAL	Проверка

Уровни CMMI

Уровень зрелости	Название уровня
1	Начальный уровень (**Initial**)
2	Управляемый уровень (**Managed**)
3	Определенный уровень (**Defined**)
4	Количественно-управляемый уровень (**Quantitatively Managed**)
5	Оптимизированный уровень (**Optimizing**)



Уровни CMMI

Уровни 4 и 5 называют уровнями высокой зрелости. Деятельность зрелых организаций отличается низкой вариативностью процессов и характеризуется использованием ключевых показателей в рамках реализации статистически оправданного метода управления. Более зрелые организации отличаются большей предсказуемостью и более высокой скоростью реагирования на новую информацию (при отсутствии бюрократических механизмов).

Менее зрелые организации бросают на борьбу с экстренными ситуациями все силы, в то время как более зрелые следуют устоявшимся процедурам и не всегда могут понять, что изменение некоторых процедур станет более адекватным ответом на сложившуюся ситуацию.

CMMI в Беларуси

Центр программного производства EPAM Systems в Минске в конце сентября 2003 г. первым в Европе успешно прошел международную аттестацию соответствия 4-му уровню CMMI® (SEI CMMI Maturity Level 4).

Группа компаний IBA 2003 г. прошла международную аттестацию соответствия 4-му уровню CMMI® (SEI CMMI Maturity Level 4).

Уровень зрелости	Область процессов	Сокращение	Цель
Уровень 2	Менеджмент требований (Requirements Management)	REQM	Управление требованиями предъявляемым к продуктам проекта или компонентам продукта, с целью выявления несоответствия между требованиями и планами проекта.
	Планирование проекта (Project Planning)	PP	Разработка и поддержание планов определяющих развитие проекта
	Мониторинг и контроль проекта (Project Monitoring and Control)	PMC	Обеспечить понимание стадии разработки проекта с целью принятия корректирующих действий в случае серьезного отклонения от плана
	Менеджмент договоров с поставщиками (Supplier Agreement Management	SAM	Управление приобретением товаров и услуг от внешних поставщиков, с которыми заключены договоры
	Измерение и анализ (Measurement and Analysis)	M&A	Разработка и поддержание возможности измерения, используемой для поддержки нужд информационного менеджмента
	Оценка (гарантирование) качества товаров и процессов (Process and Product Quality Assurance)	PPQA	Обеспечение поддержки и управления в соответствии с целями процессов и связанными с ними продуктами работы
	Конфигурационный менеджмент (Configuration Management)	CM	Установка и поддержание целостности продуктов работы (work products) в результате использования идентификации конфигураций, конфигурационного контроля и конфигурационного аудита
Уровень 3	Разработка требований (Requirements Development)	RD	Сбор и анализ требований потребителей к продуктам и компонентам продуктов
	Техническое решение (Technical Solution)	TS	Разработка, дизайн и внедрение решений по соответствующим требованиям. Решения, дизайн и внедрения выражены продуктами, компонентами продуктов и связанными с данными продуктами процессами.
	Интеграция продукта (Product Integration)	PI	Сборка (монтирование) продукта из его составляющих, проверка качества интеграции, ее функциональности и выпуск продукта.
	Верификация (Verification)	Ver	Гарантирование того, что выбранные продукты работы отвечают предъявляемым требованиям
	Валидация (Validation)	Val	Демонстрация того, что продукт и его компоненты соответствуют его предполагаемому использованию в предполагаемой среде.
	Фокусирование на процессах организации (Organization Process Focus)	OPF	Установление и поддержание понимания процессов организации и процессных активов, идентификация, планирование и внедрение улучшений связанных с данными областями.
	Описание процессов организации (Organization Process Definition)	OPD	Установление и поддержание возможного к использованию массива процессов организации
	Организационный тренинг (Organizational Training)	OT	Повышение знаний и способностей людей для выполнения ими своих ролей эффективно и рационально
	Менеджмент интеграции проектов (Integrated Project Management)	IPM	Установка и управление проектом и вовлечение всех заинтересованных лиц в интегрированный и определенный процесс. Данная область также затрагивает общее видение проекта командой разработчиков
	Менеджмент рисков (Risk Management)	RSKM	Определение потенциальных проблем до их появления. В связи с этим процессы по снижению рисков могут планироваться и осуществляться на любом этапе разработки продукта или процесса.
	Интегрированные команды (разработчиков)Integrated Teaming	IT	Формирование и поддержание интегрированных команд для разработки продуктов работы (work products)
	Интегрированное управление поставщиками (Integrated Supplier Management)	ISM	Мониторинг новых продуктов, оценка источников продуктов, которые могут удовлетворить требованиям к проекту и использование данной информации для выбора поставщиков
	Анализ решений и разрешение(Decision Analysis and Resolution	DAR	Разработка решений на основе структурированного подхода, который позволяет оценить альтернативные решения на основе установленных критериев
	Организационная среда для интеграции (Organizational Environment for Integration)	OEI	Предоставление инфраструктуры для интегрированной разработки продуктов и процессов и управление людьми (персоналом) в целях интеграции
Уровень 4	Производительный организационный процесс (Organizational Process Performance)	OPP	Установление и поддержание количественного понимания производительности набора стандартизированных процессов организации и обеспечение информацией о производительности процессов и моделей для количественного управления проектами организации.
	Количественный менеджмент проекта (Quantitative Project Management)	QPM	Количественно управлять определенным процессом в целях достижения установленного в рамках проекта качества и целей производительности.
Уровень 5	Организационные инновации и внедрение(Organizational Innovation and Deployment)	OID	Выбор и внедрение инноваций и улучшений, которые измеряемо, улучшают организационные процессы и технологии.
	Анализ причин и разрешение (Causal Analysis and Resolution)	CAR	Идентификация причин дефектов и других проблем и принятие действий предотвращающих их появление в будущем

Процессы разработки

Для успешного выполнения программного проекта недостаточно выбрать эффективные средства разработки, обеспечить необходимый бюджет и найти квалифицированных разработчиков.

В любой организации существуют правила и методики, по которым участники проекта (заказчики, разработчики, тестеры, технические писатели) распределяют между собой задачи, взаимодействуют друг с другом, создают проектные артефакты (спецификации, исходный код, документацию).

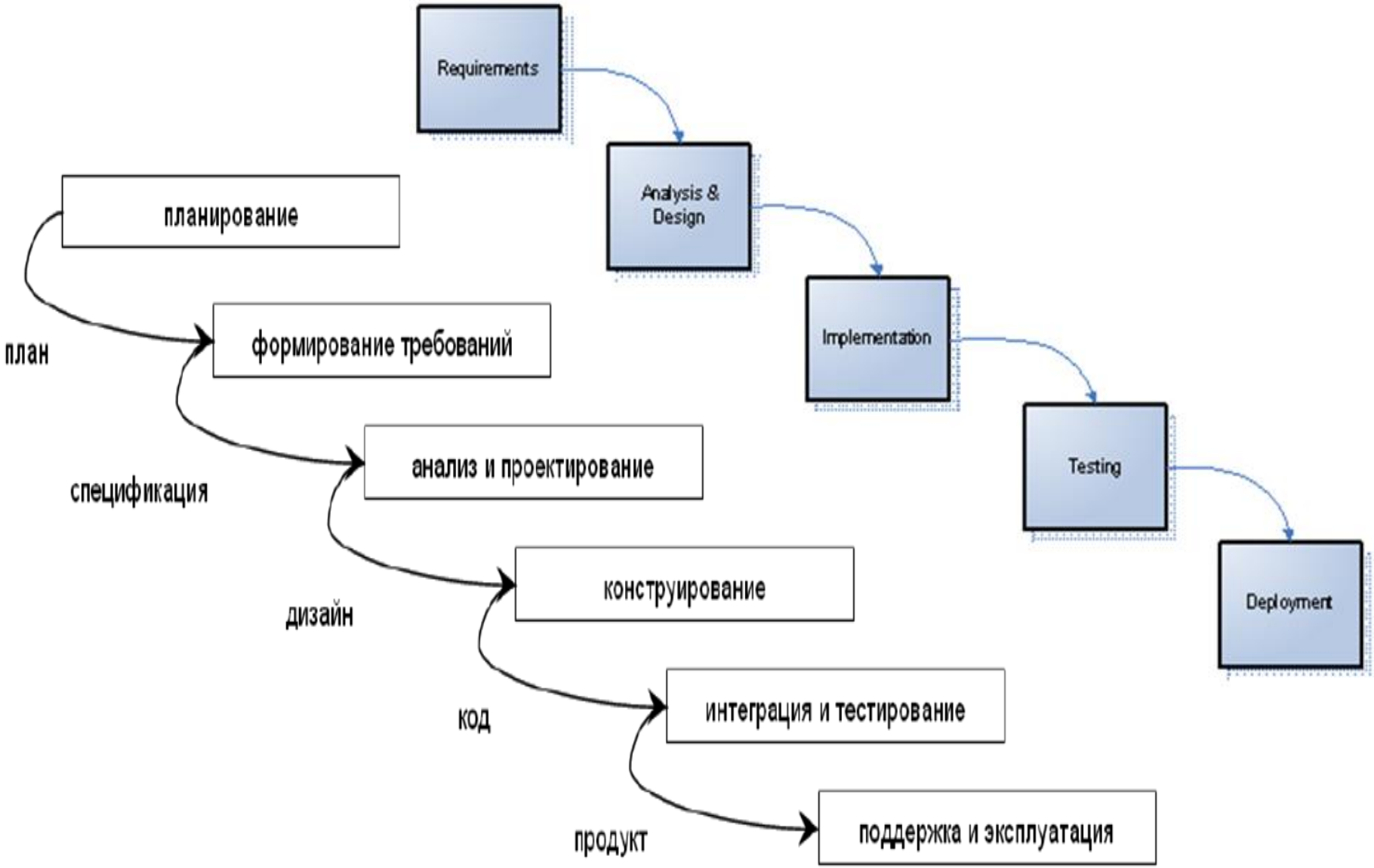
Эти правила могут быть четко организованными или хаотичными, быть формально документированными или существовать в головах проектной команды, но именно их совокупность называется процессом разработки.

Процессы разработки

Жизненный цикл разработки программного обеспечения (ЖЦ проекта) – проектная деятельность по разработке и развертыванию программных систем

Жизненный цикл программной системы – включает разработку, развертывание, поддержку и сопровождение

Последовательная разработка



Последовательная разработка

Модель водопада (*waterfall model* или последовательная разработка) – основная идея: процесс разработки делится на четко определенные фазы, выполняемые строго последовательно, переход от одной фазы проекта к другой предполагает полную корректность результата (выхода) предыдущей фазы.

Фазы проекта включают следующие этапы:

- **Разработка требований** (*requirements*): выявление пользователей системы и других заинтересованных лиц, сбор бизнес- и пользовательских требований, их преобразование в функциональные требования к программному продукту, сбор нефункциональных требований.
- **Анализ и дизайн** (*analysis and design*): разработка моделей предметной области (*domain model*) и системного анализа (*application model*), проектирование классов приложения и логической структуры базы данных, разработка объектной модели, проетирование пользовательского интерфейса и т. п.

Последовательная разработка

- **Реализация** (*implementation*): создание продукта по спецификациям, разработанным на предыдущем этапе.
- **Тестирование** (*testing*): включает проверку соответствия функциональности программного продукта потребностям пользователей (validation), а также поиск дефектов в реализации.
- **Развертывание** (*deployment*): обучение пользователей, инсталляция системы, перевод в промышленную эксплуатацию.

Неточность какого-либо требования или некорректная его интерпретация, в результате, приводит к тому, что приходится “откатываться” к ранней фазе проекта и требуемая переработка не просто выбивает проектную команду из графика, но приводит часто к качественному росту затрат и, не исключено, к прекращению проекта в той форме, в которой он изначально задумывался.

Последовательная разработка

Недостатки модели:

1. Процесс плохо работает в проектах с нечеткими требованиями. Даже если проектная команда считает, что полностью проработала и документировала функциональный дизайн системы, он может значительно отличаться от ожиданий пользователей. С большой вероятностью это расхождение будет обнаружено не на этапе рецензирования функциональной спецификации (редкий заказчик способен представить поведение реальной системы, читая документ с описанием ее функциональности), а во время внедрения продукта.

2. Процесс неэффективен при постоянных изменениях требований (часто случается в проектах, длящихся несколько месяцев). Каждое изменение заставляет возвращаться к фазе определения требований и повторять весь процесс с начала

Последовательная разработка

3. Сложно управлять рисками некоторых типов (таких, как риски, связанные с использованием новых технологий или риски некорректного определения требований). Подобные риски могут проявить себя только на этапе реализации (если не тестирования), когда число возможных путей исправления ситуации намного меньше, чем в начале проекта.
4. Ограничены возможности оценки и корректировки важных атрибутов проекта – скорости разработки, качества продукта, обоснованности принятых архитектурных решений. Адекватно оценить эти атрибуты становится возможным только на поздних этапах проекта.

Последовательная разработка

Фредерик Брукс во втором издании своего классического труда “Мифический человеко-месяц” так описывает главную беду каскадной модели [Брукс, 1995, с.245]:

“Основное заблуждение каскадной модели состоит в предположениях, что проект проходит через весь процесс один раз, архитектура хороша и проста в использовании, проект осуществления разумен, а ошибки в реализации устраняются по мере тестирования. Иными словами, каскадная модель исходит из того, что все ошибки будут сосредоточены в реализации, а потому их устранение происходит равномерно во время тестирования компонентов и системы.”

Последовательная разработка

Практика показывает, что в реальном мире, особенно в мире бизнес-систем, каскадная модель не должна применяться. Специфика таких систем (если можно говорить о “специфике” для подавляющего большинства создаваемых систем) - требования характеризуются высокой динамикой корректировки и уточнения, невозможностью четкого и однозначного определения требований до начала работ по реализации (особенно, для новых систем) и быстрой изменчивостью в процессе эксплуатации системы.

[«Основы программной инженерии» (перевод SWEBOK), Сергей Орлик, 2005-2010, http://swebok.sorlik.ru]:

Итеративная и инкрементальная модель как эволюционный подход

Итеративная разработка

Итеративная (инкрементальная) разработка – это эволюционное развитие модели водопада. Процесс состоит из серии повторяющихся итераций (их число зависит от конкретного проекта), каждая из которых фактически является полноценным мини-проектом с фазами определения требований, анализа, дизайна и т.д.

В результате очередной итерации продукт приобретает новую функциональность или улучшения в существующей.

Полный набор требований, зафиксированный границами проекта, оказывается реализованным после завершения финальной итерации.

Итеративная разработка (RUP)

- **RUP (Rational Unified Process)**

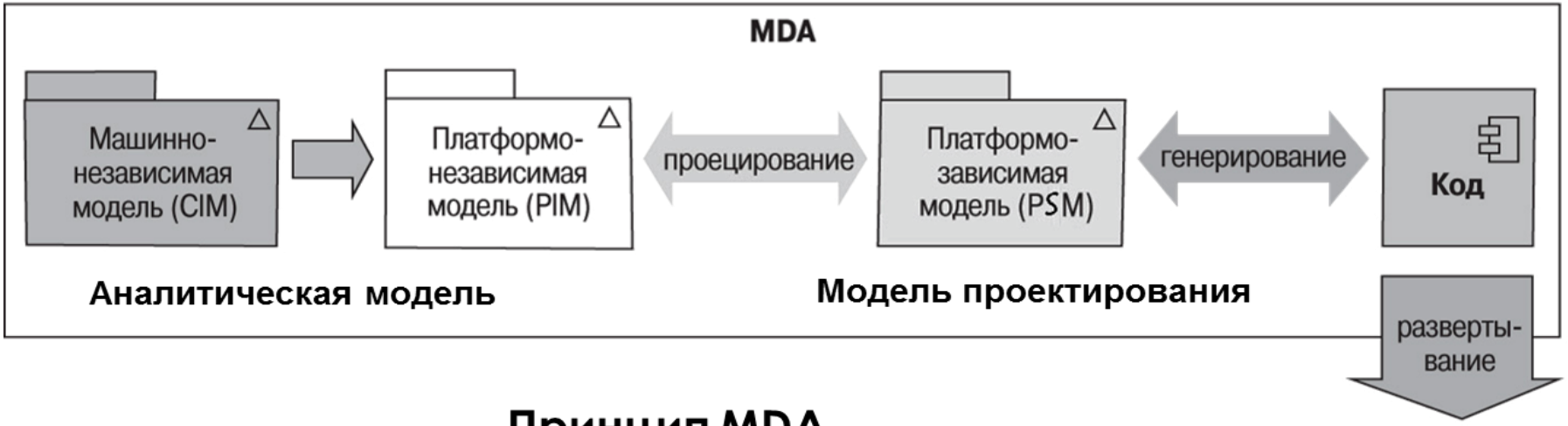
Одной из самых известных процессов, использующих итеративную модель разработки. Он был создан во второй половине 1990-х годов в компании Rational Software. Основными разработчиками были Филипп Крачтен (Philippe Kruchten), Грейди Буч (Grady Booch), Джеймс Рамбо (James Rumbaugh) и Айвар Якобсон (Ivar Jacobson).

Термин RUP означает методологию разработки и продукт компании IBM (ранее – Rational) для управления процессами разработки.

Итеративная разработка

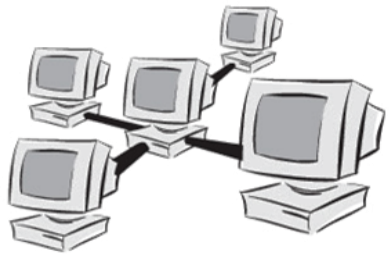
Итеративная разработка

RUP – первая методология с реализацией MDA



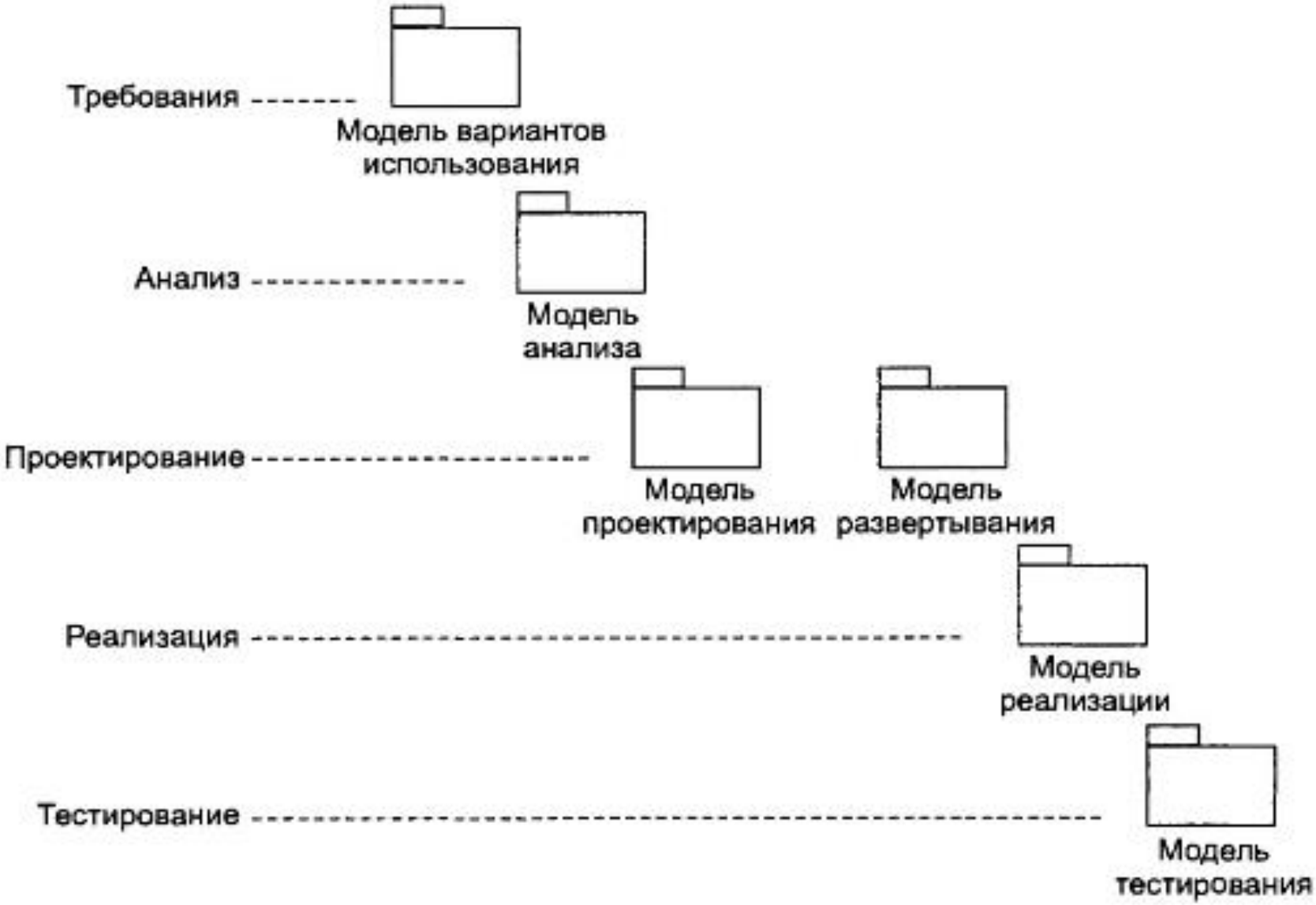
Принцип MDA

Архитектура, управляемая моделью
(Model Driven Architecture, MDA)



- Программное обеспечение создаётся в результате ряда трансформаций модели при поддержке инструмента моделирования.
- Аналитическая модель описывает предметную область и программную систему с разных точек зрения (представлений).
- Все представления модели направлены на единое целое – проект.

Итеративная разработка (RUP)



Итеративная разработка (RUP)

Методология RUP описывает абстрактный общий процесс, на основе которого организация или проектная команда должна создать специализированный процесс, ориентированный на ее потребности.

В терминах RUP участники проектной команды создают так называемые **артефакты** (*work products*), выполняя **задачи** (tasks) в рамках определенных **ролей** (roles). Артефактами являются спецификации, модели, исходный код и т.п.

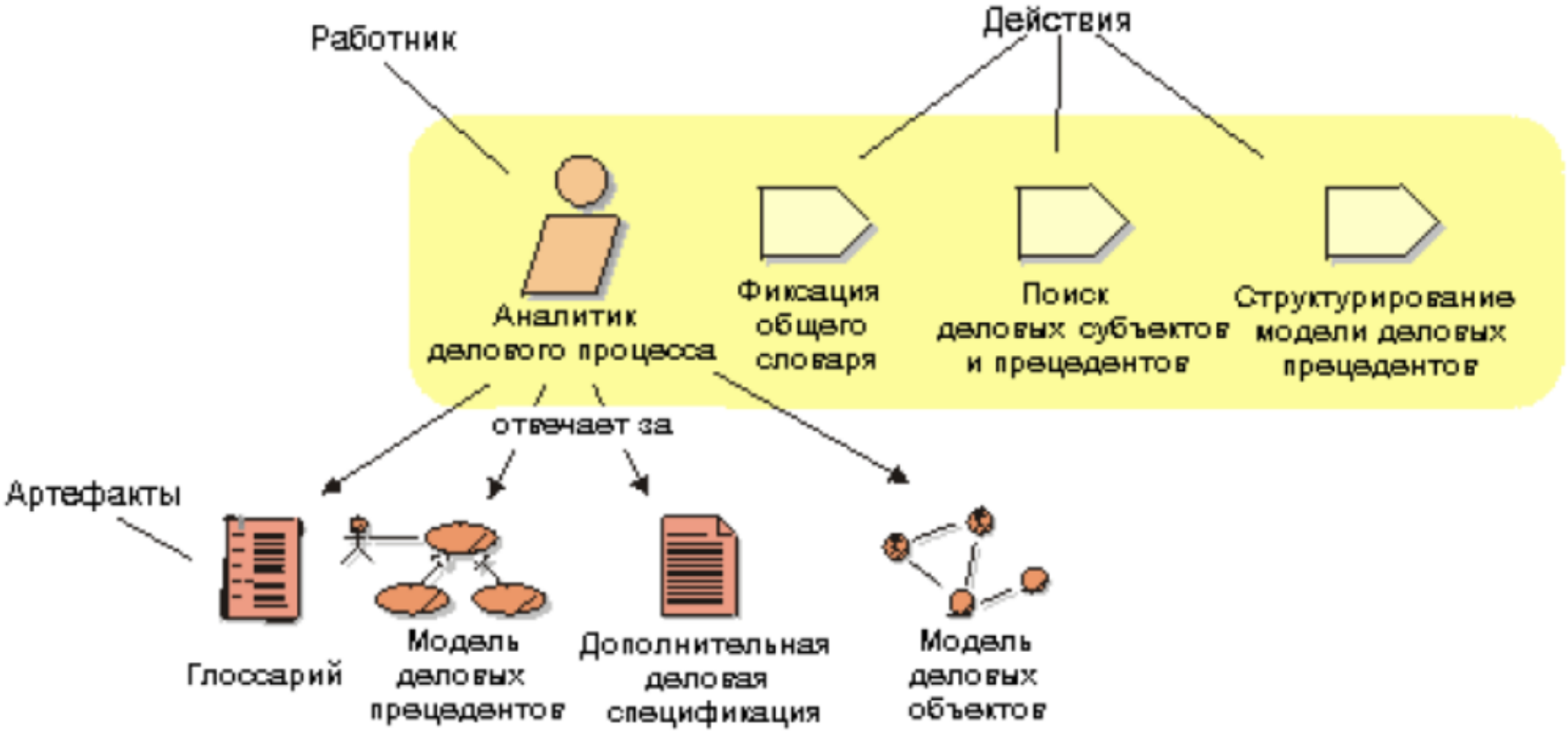
Задачи разделяются по девяти процессным областям, называемым **дисциплинами** (discipline).

Итеративная разработка (RUP)

Участники RUP

Итеративная разработка (RUP)

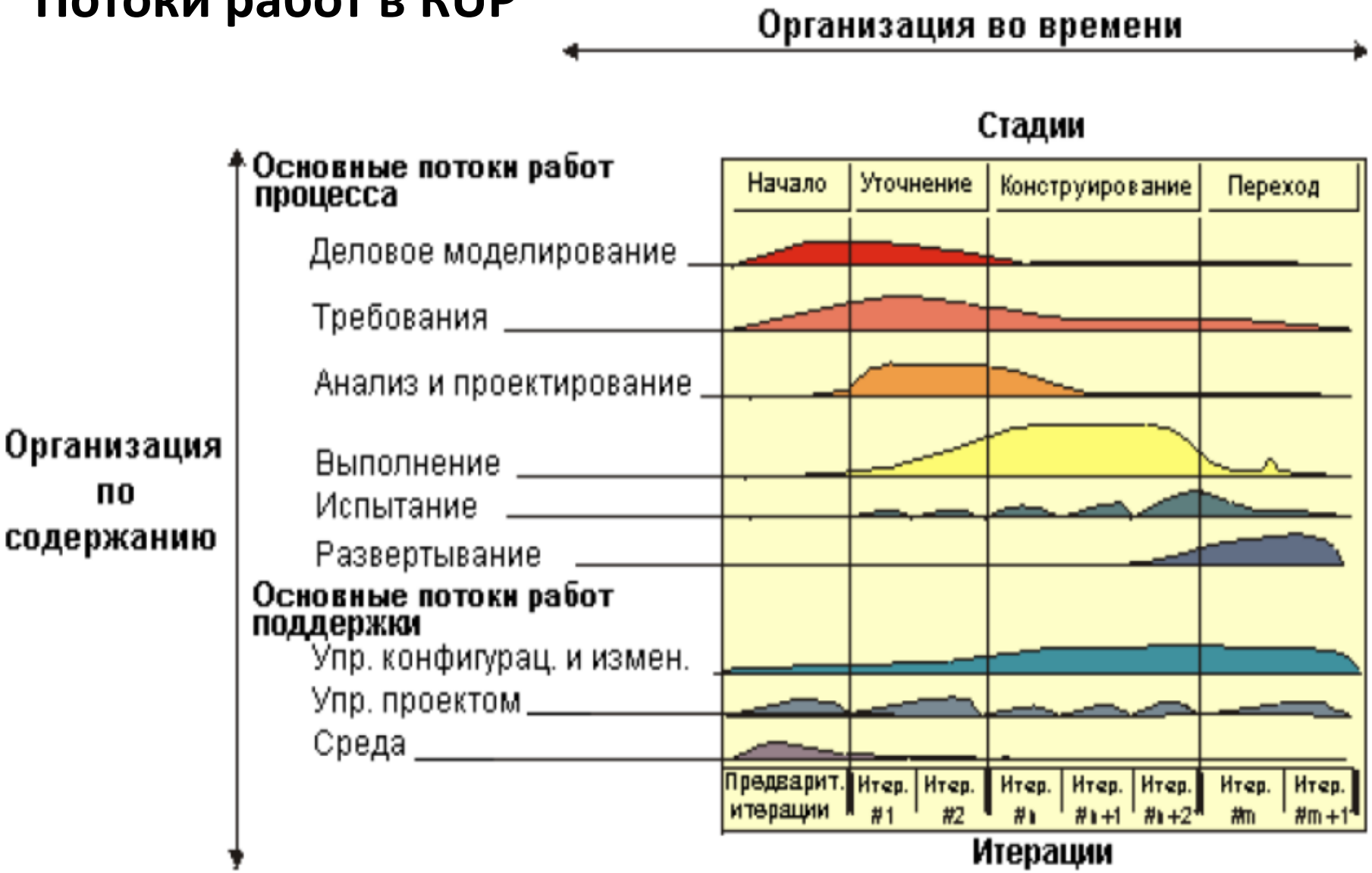
Действия работников в RUP



Каждый работник имеет свой собственный набор действий и артефактов

Итеративная разработка (RUP)

Потоки работ в RUP



Итеративная разработка (RUP)

В RUP определены шесть инженерных и три вспомогательные дисциплины:

• **Бизнес-моделирование** (*Business Modeling*) – определение и описание бизнес-целей, моделирование сущностей предметной области (сущности и их атрибуты, структура связей, взаимодействие внешних и внутренних сущностей), моделирование функциональных обязанностей бизнес-актеров, анализ и моделирование существующих бизнес-процессов заказчика, а также поиск их возможных улучшений.

• **Разработка и Управление требованиями** (*Requirements Engineering & Requirements Management*) – выявление требований, определение границ проекта, разработка функционального дизайна будущей системы и его согласование с заказчиком.

Итеративная разработка (RUP)

- **Анализ и проектирование** (*Analysis and Design*) – проектирование архитектуры системы на основе функциональных требований и ее развитие на протяжении всего проекта.

  Анализ (*Analysis)*: выявление объектов программной системы, отвечающих за ее бизнес-логику, путем анализа потоков событий, описанных в сценариях, моделирование их взаимодействия и структуры связей, моделирование классов бизнес-логики, их иерархий и структуры связей (платформонезависимая модель PIM).

  Проектирование *(Design)*: создание и моделирование платформозависимой архитектуры программной системы (модель PSM).

- **Реализация** *(Implementation*) – разработка, юнит-тестирование и интеграция компонентов системы.

Итеративная разработка (RUP)

- **Тестирование** (*Test)* – поиск и отслеживание дефектов в системе, проверка корректности реализации требований.
- **Развертывание** (*Deployment*) – создание дистрибутива, установка системы, обучение пользователей.
- **Управление конфигурациями и изменениями** (*Configuration and Change Management*) – управление версиями исходного кода и документации, процесс обработки запросов на изменение (*change requests*).
- **Управление проектом** (*Project Management)* – создание проектной команды, планирование фаз и итераций, управление бюджетом и рисками.
- **Среда** (*Environment)* – создание инфраструктуры для выполнения проекта, включая организацию и настройку процесса разработки.

Итеративная разработка (RUP)

Стадии проекта в RUP

Начало (*Inception*)

Цели:

- понять границы проекта:
- разработка экономического обоснования;
- заключение соглашения между заинтересованными сторонами.
- Веха: разработка целей жизненного цикла

Уточнение (*Elaboration*)

Цели:

- свести к минимуму главные риски;
- создать базовую архитектуру;
- понять во что обойдется построение системы
- Веха: архитектура программной системы

Итеративная разработка (RUP)

Стадии проекта в RUP

Конструирование (*Construction*)

Цели:

- Построить первую работающую версию продукта
- Веха: начальная функциональная готовность

Внедрение (*Transition*)

Цели:

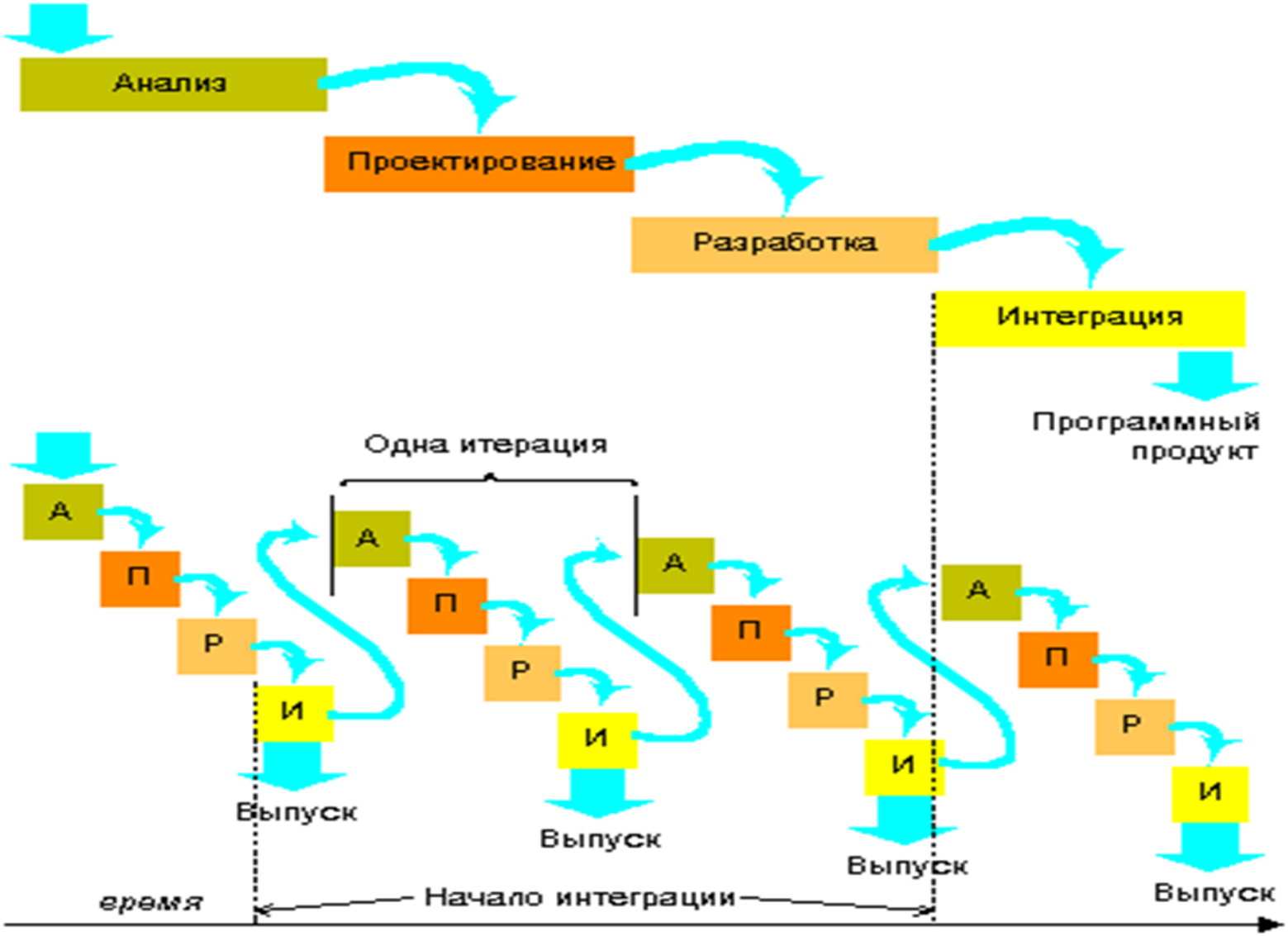
- Создать окончательную версию программного продукта и передать ее Заказчику
- Веха: готовый продукт

Итеративная разработка (RUP)

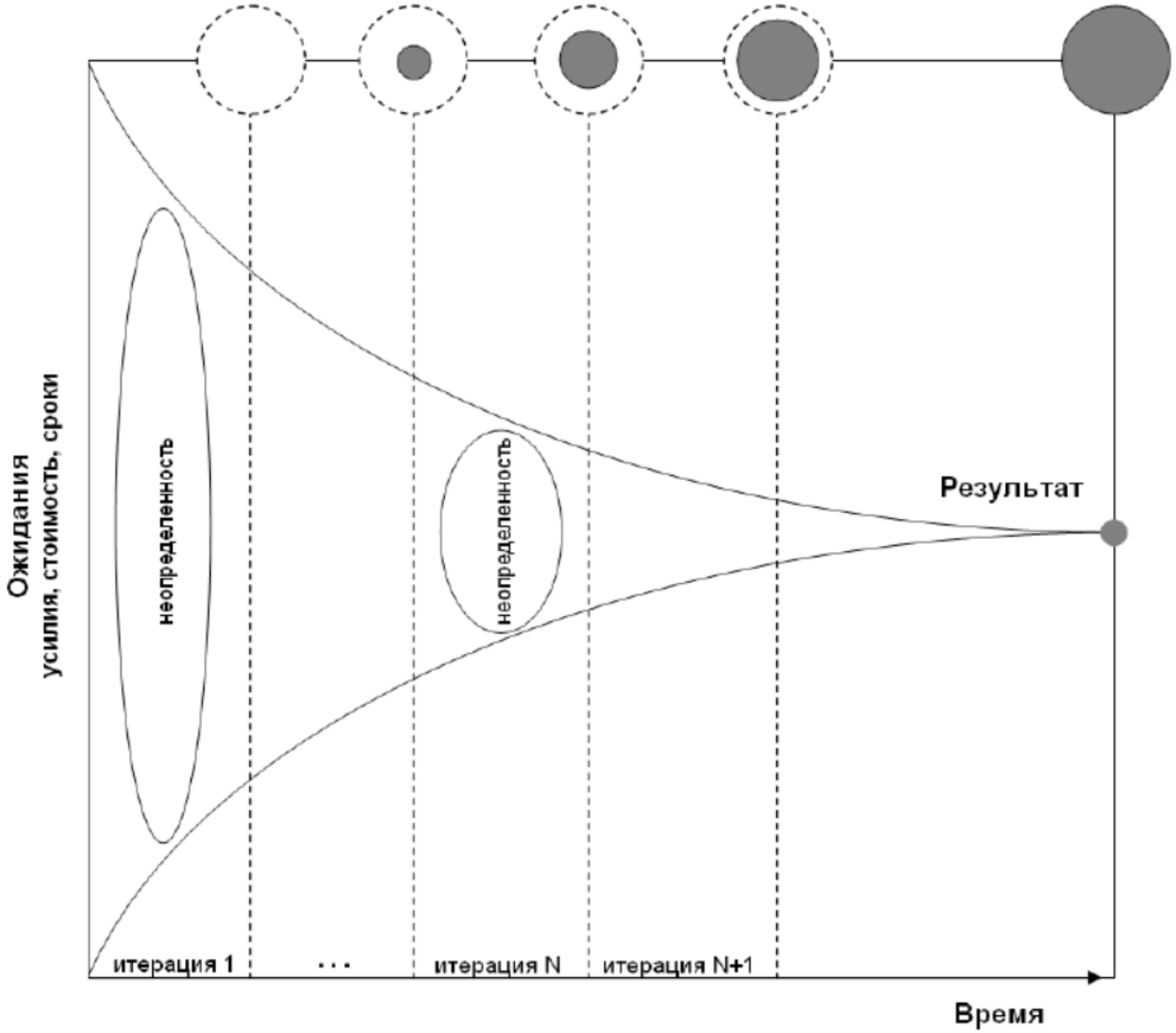
Инструментальные средства в RUP

- Инструмент моделирования: Rational Rose
- Инструмент управления требованиями: Rational Requisite Pro
- Инструмент документирования: Rational SoDA
- Средства программирования: Rational Apex/C++ (Java ready)
- Инструмент поддержки PM: Microsoft Project
- Инструмент управления задачами: Rational ClearQuest
- Инструмент управления конфигурацией: ClearCase
- Инструменты тестирования: Rational SQA Suite, Rational TestMate, Rational Visual Test

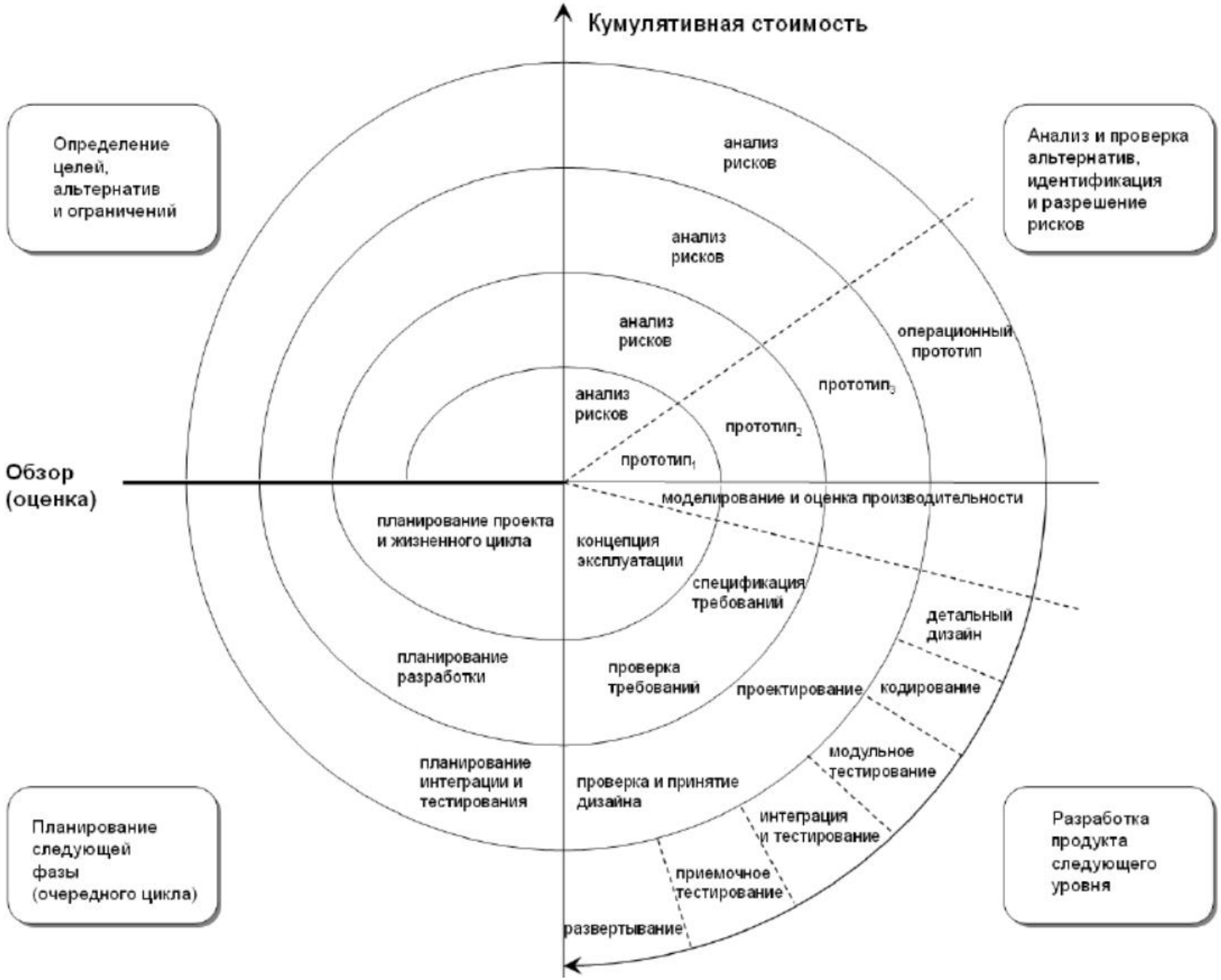
Итеративная разработка (RUP)



Итеративная разработка (RUP)



Спиральная модель (Барри Боэма)



Роли и ответственность участников

Роли и ответственности участников типового проекта разработки ПО можно условно разделить на пять групп:

- **Анализ.** Извлечение, документирование и сопровождение требований к продукту.
- **Управление**. Определение и управление производственными процессами.
- **Производство**. Проектирование и разработка ПО.
- **Тестирование**. Тестирование ПО.
- **Обеспечение**. Производство дополнительных продуктов и услуг.

Роли и ответственность участников

Группа анализа включает в себя следующие роли:

- *Бизнес-аналитик*. Построение модели предметной области (онтологии).
- *Бизнес-архитектор*. Разрабатывает бизнес-концепцию системы. Определяет общее видение продукта, его интерфейсы, поведение и ограничения.
- *Системный аналитик*. Отвечает за перевод требований к продукту в функциональные требования к ПО.
- *Специалист по требованиям*. Документирование и сопровождение требований к продукту.
- *Менеджер продукта* (функциональный заказчик). Представ-ляет в проекте интересы пользователей продукта.

Роли и ответственность участников

Группа управления состоит из следующих ролей:

- *Руководитель проекта*. Отвечает за достижение целей проекта при заданных ограничениях (по срокам, бюджету и содержанию), осуществляет операционное управление проектом и выделенными ресурсами.
- *Куратор проекта*. Оценка планов и исполнения проекта. Выделение ресурсов.
- *Системный архитектор*. Разработка технической концепции системы. Принятие ключевых проектных решений относительно внутреннего устройства программной системы и ее технических интерфейсов.

Роли и ответственность участников

- *Руководитель группы тестирования*. Определение целей и стратегии тестирования, управление тестированием.
- *Ответственный за управление изменениями, конфигурациями*, за сборку и поставку программного продукта.

В производственную группу входят следующие роли:

- *Проектировщик.* Проектирование компонентов и подсис-тем в соответствие с общей архитектурой, разработка архитектурно-значимых модулей.
- *Проектировщик базы данных*.
- *Проектировщик интерфейса пользователя*.
- *Разработчик*. Проектирование, реализация и отладка отдельных модулей системы.

Роли и ответственность участников

Группа тестирования состоит из следующих ролей:

- *Проектировщик тестов*. Разработка тестовых сценариев.
- *Разработчик автоматизированных тестов*.
- *Тестировщик.* Тестирование продукта. Анализ и документирование результатов.

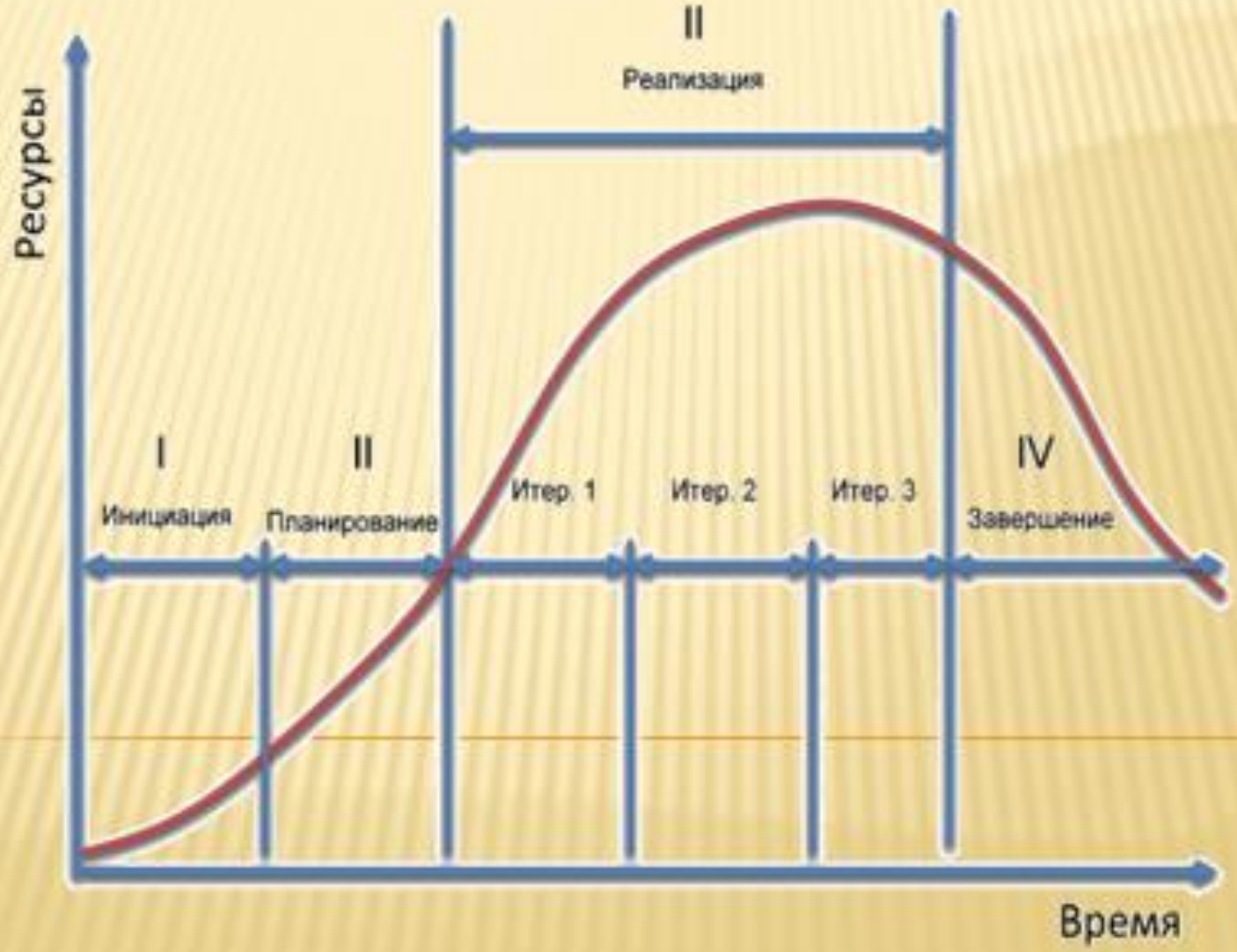
Группа обеспечения может включать следующие проектные роли:

- *Технический писатель*.
- *Переводчик.*
- *Дизайнер графического интерфейса*.

Роли и ответственность участников

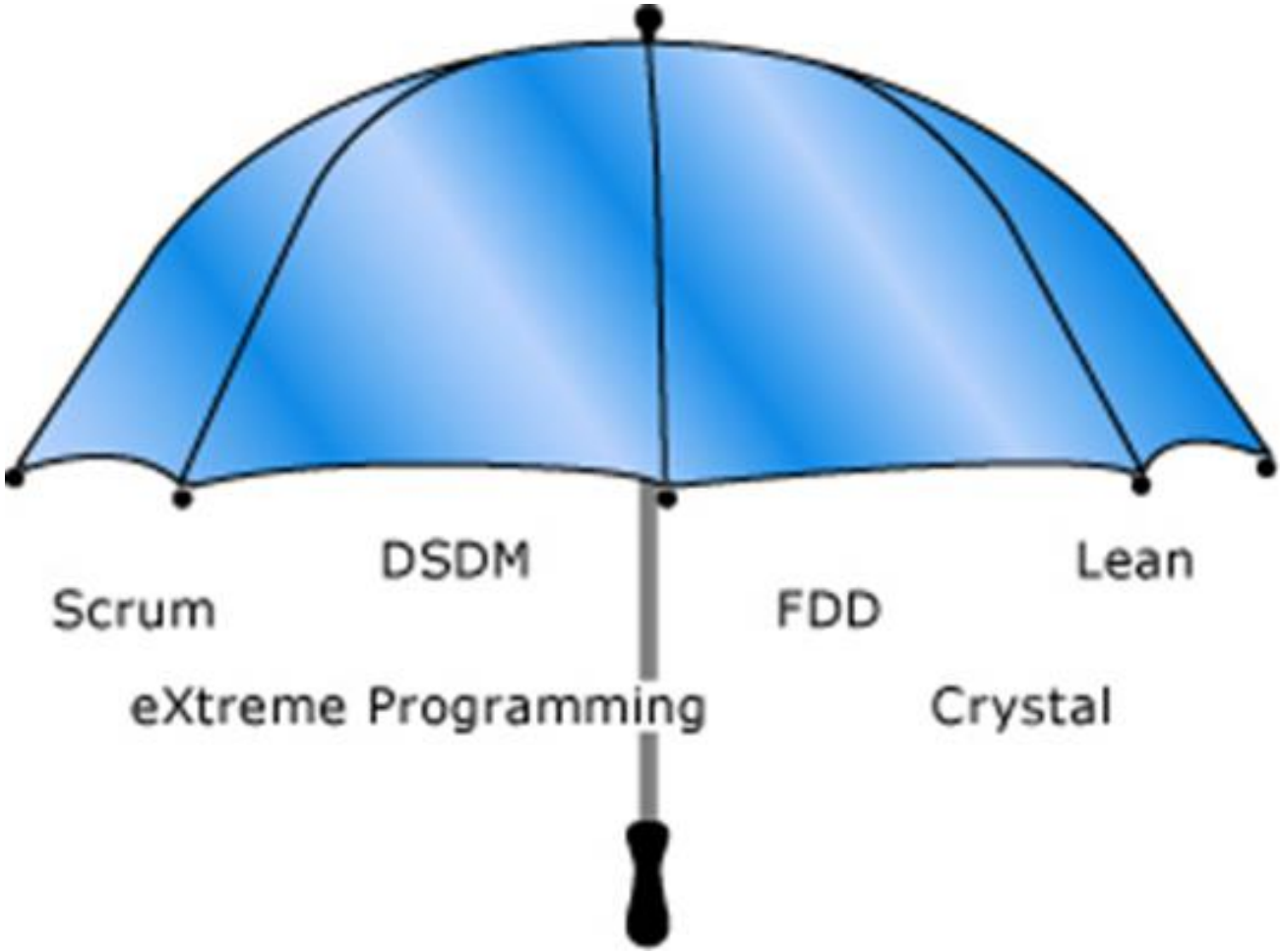
- *Разработчик учебных курсов, тренер*.
- *Участник рецензирования*.
- *Менеджер по продажам и маркетингу*.
- *Системный администратор*.
- *Технолог*.
- *Специалист по инструментальным средствам*.
- Другие.

Распределение ресурсов по фазам проекта



ЧАСТЬ II

ГИБКАЯ РАЗРАБОТКА (Agile)



Гибкая разработка (Agile)

Гибкая разработка — это термин, произошедший от так называемого "**Гибкого манифеста**" (*Agile Manifesto*), написанного в 2001 г. группой, включавшей в себя создателей **Scrum**, экстремального программирования (**XP**), метода разработки динамических систем (**DSDM**) и **Crystal**; представителя разработки, управляемой функциями; а также некоторых других лидеров мысли из отрасли программного обеспечения.

"**Гибкий манифест**" установил общий набор ценностей и принципов для всех отдельных гибких методологий, существовавших на то время. В нем детализированы четыре ключевые ценности, позволяющих командам достигать высокой производительности.

Agile-манифест разработки программного обеспечения

Мы постоянно открываем для себя более совершенные методы разработки программного обеспечения, занимаясь разработкой непосредственно и помогая в этом другим. Благодаря проделанной работе мы смогли осознать, что

- *Люди и взаимодействие **важнее** процессов и инструментов*
- *Работающий продукт **важнее** исчерпывающей документации*
- *Сотрудничество с заказчиком **важнее** согласования условий контракта*
- *Готовность к изменениям **важнее** следования первоначальному плану*

То есть, не отрицая важности того, что справа, мы всё таки больше ценим то, что слева.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Основополагающие принципы Agile-манифеста

Мы следуем таким принципам:

• Наивысшим приоритетом для нас является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения.

• Изменение требований приветствуется, даже на поздних стадиях разработки. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.

• Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.

• На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.

• Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.

• Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.

• Работающий продукт — основной показатель прогресса.

• Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно. Agile помогает наладить такой устойчивый процесс разработки.

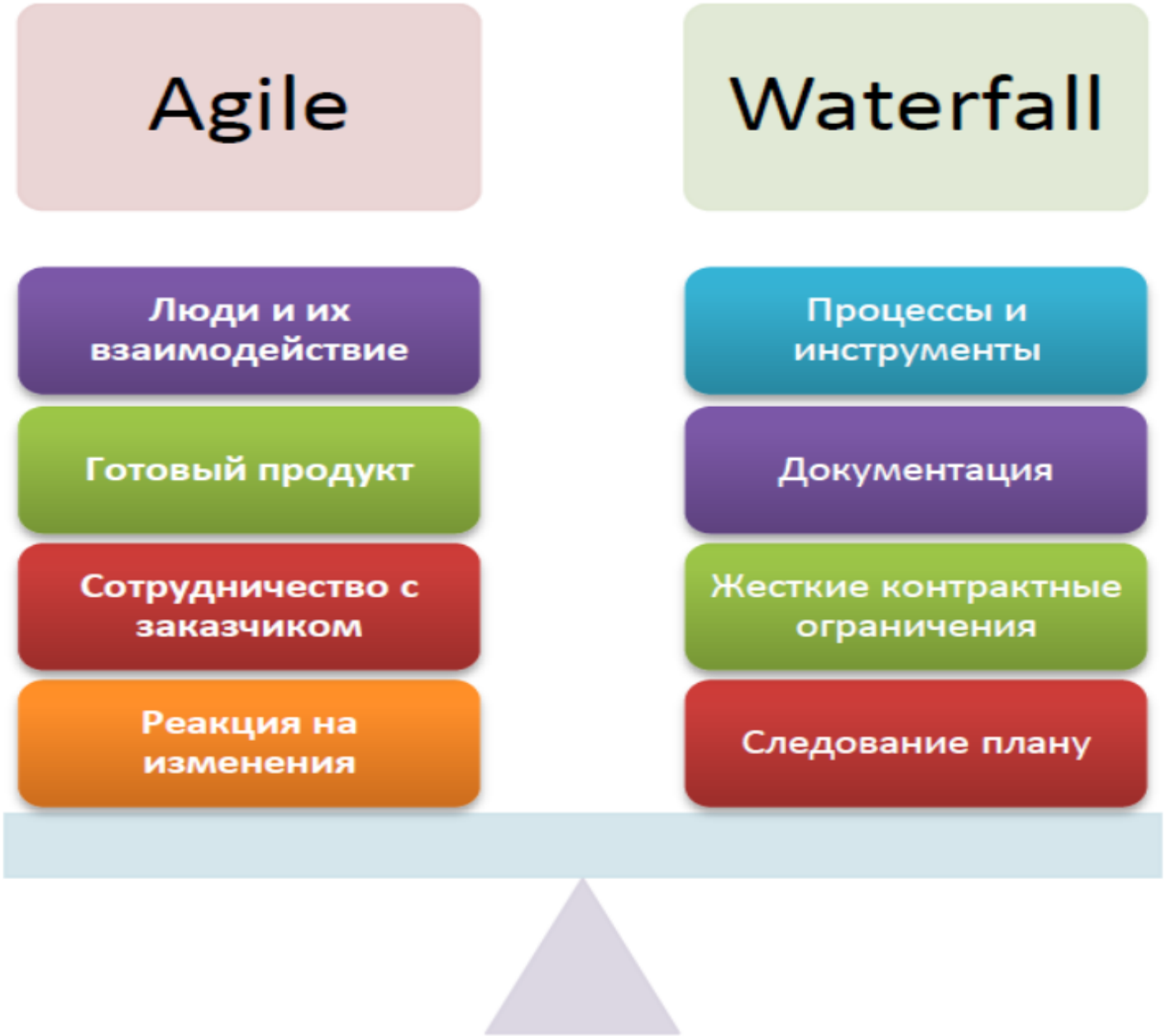
• Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.

• Простота — искусство минимизации лишней работы — крайне необходима.

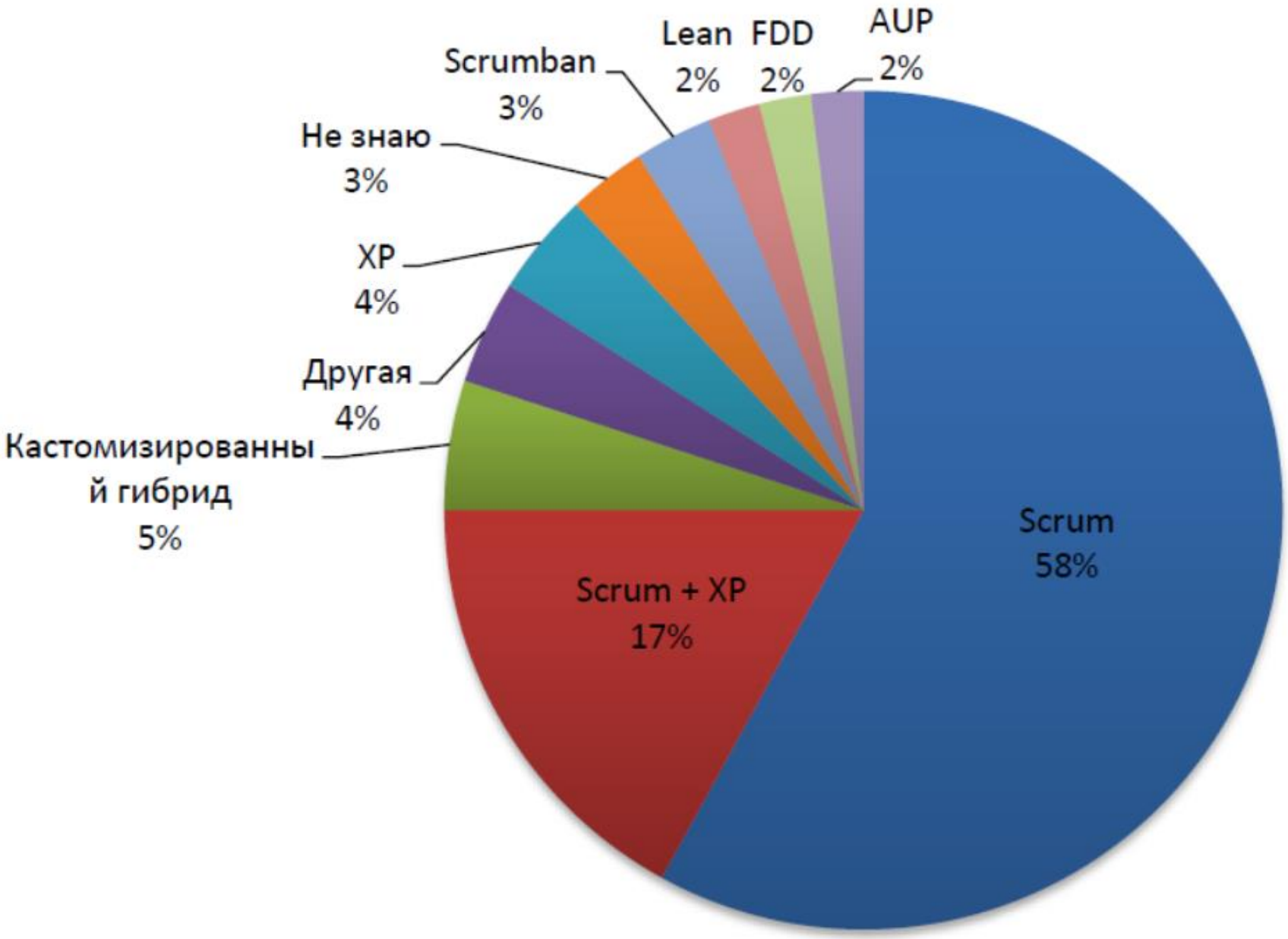
• Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.

• Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

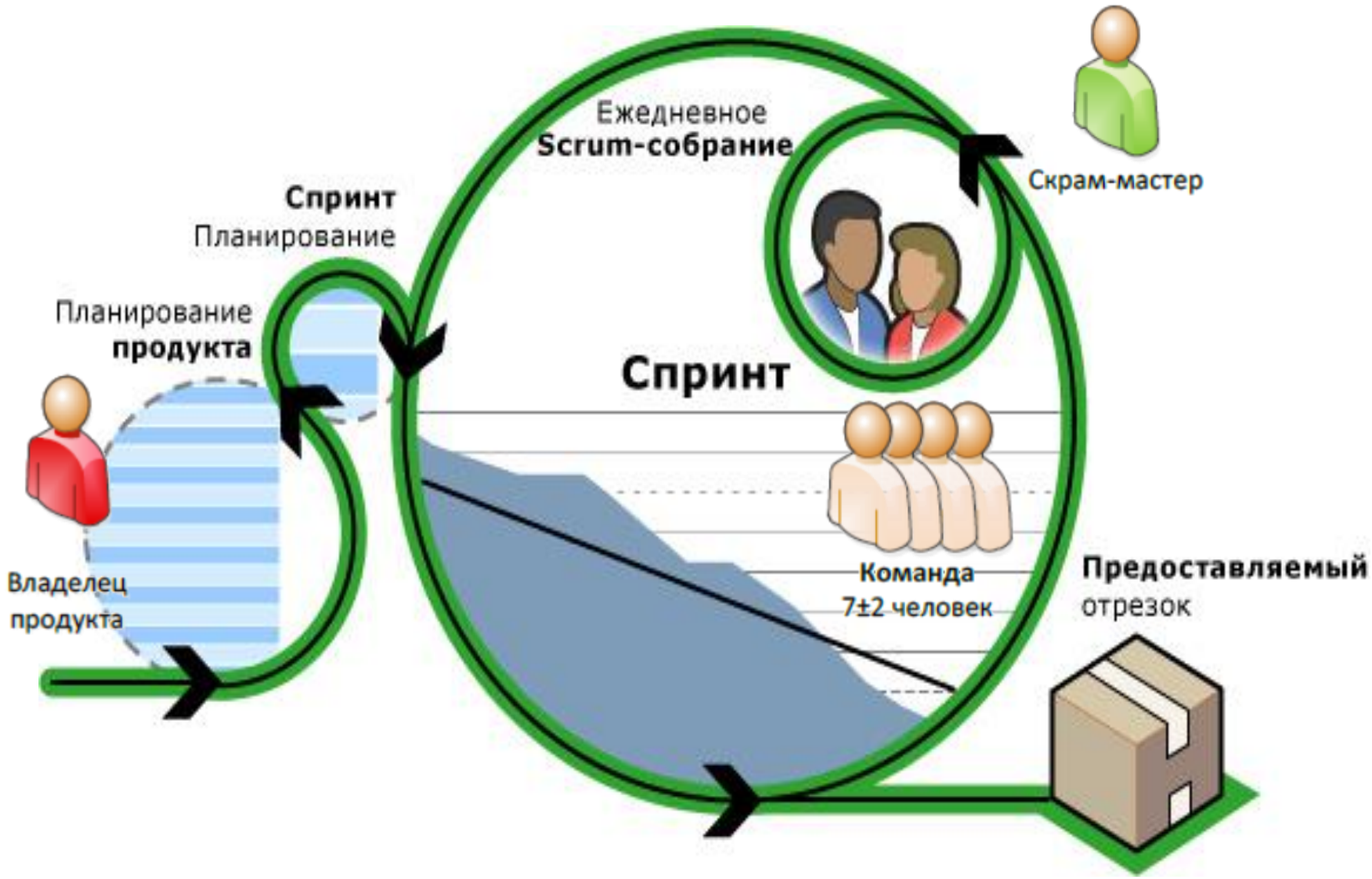
Манифест это:



Популярность гибких методологий



Методология SCRUM



Код; Обзор Спринта;
Ретроспектива

Product backlog

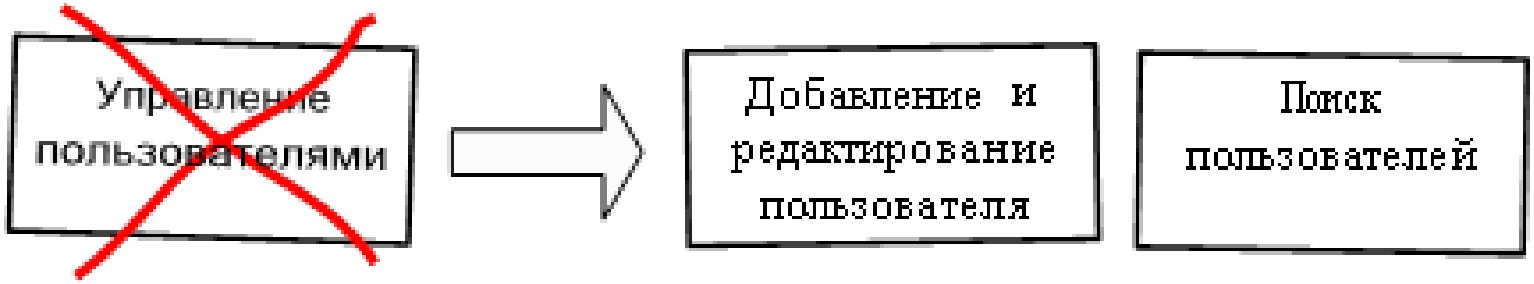
- **Product backlog** – список требований, историй, функциональности (user story), которые размещены по степени важности и описаны на понятном для заказчика языке
- **User story** включает:

1. **Порядковый номер** (ID);
2. **Название** – краткое описание от 2 до 10 слов;
3. **Индекс важности** – степень важности данной задачи по отношению к другим по мнению product owner, например, 10 или 150;
4. **Предварительная оценка объема работ** (оценивает команда используя planning poker);
5. **Сценарий демонстрации** (how to demo);
6. Примечания.

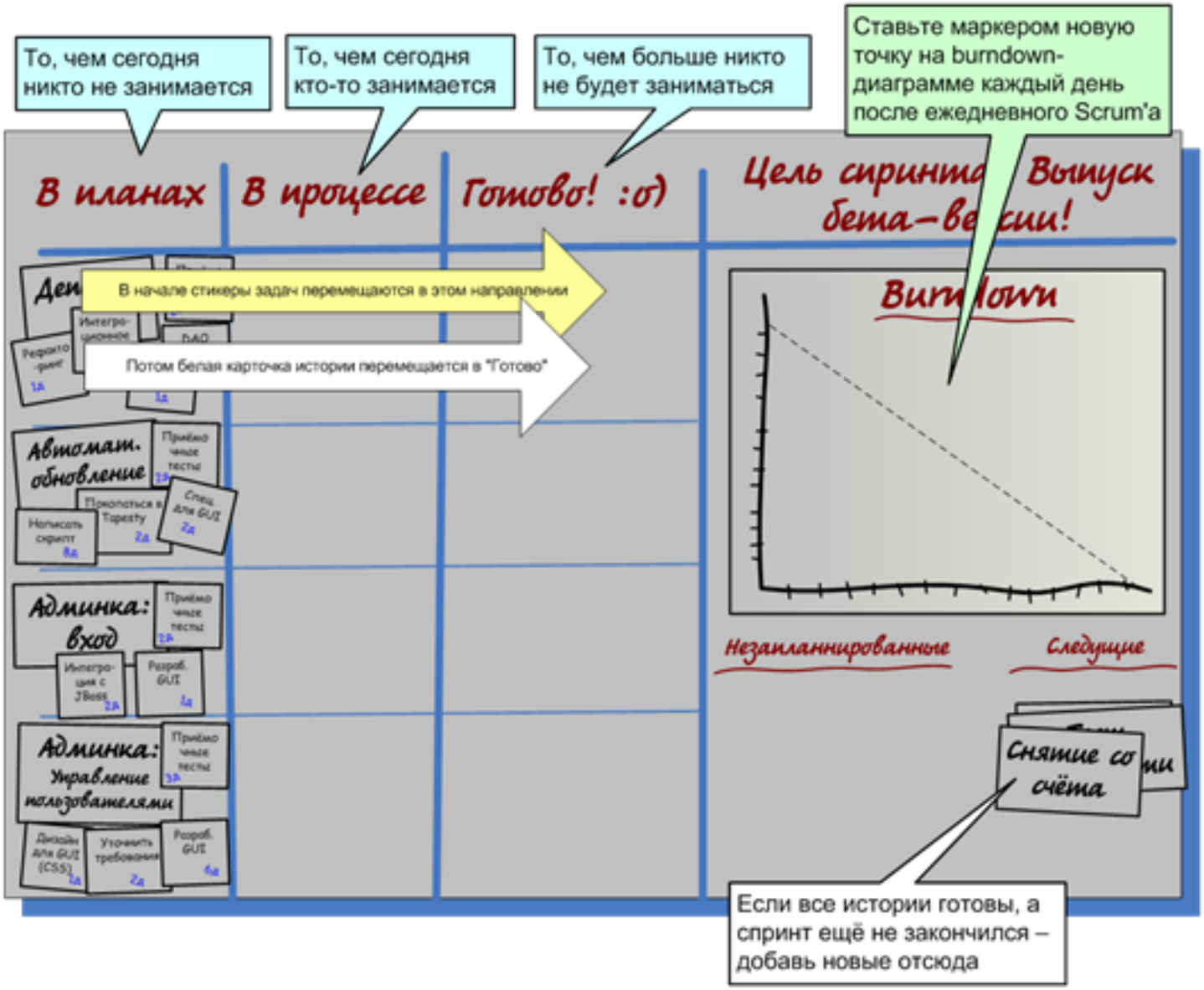
Пример user story

Product backlog (пример)					
ID	Название	Важность	Предварительная оценка	Как продемонстрировать	Примечания
1	Депозит	30	5	Войти в систему, открыть страницу депозита, положить на счет €10, перейти на страницу баланса и проверить, что он увеличился на €10.	Нужна UML диаграмма последовательности. Пока что не стоит беспокоиться про шифрование данных.
2	Просмотр журнала личных транзакций	10	8	Войти в систему; перейти на страницу транзакций; положить деньги на счет; вернуться на страницу транзакций; проверить, что новая транзакция появилась в списке.	Чтобы избежать больших запросов к базе данных, стоит воспользоваться постраничным выводом информации. Дизайн такой же, как и у страницы просмотра пользователей.

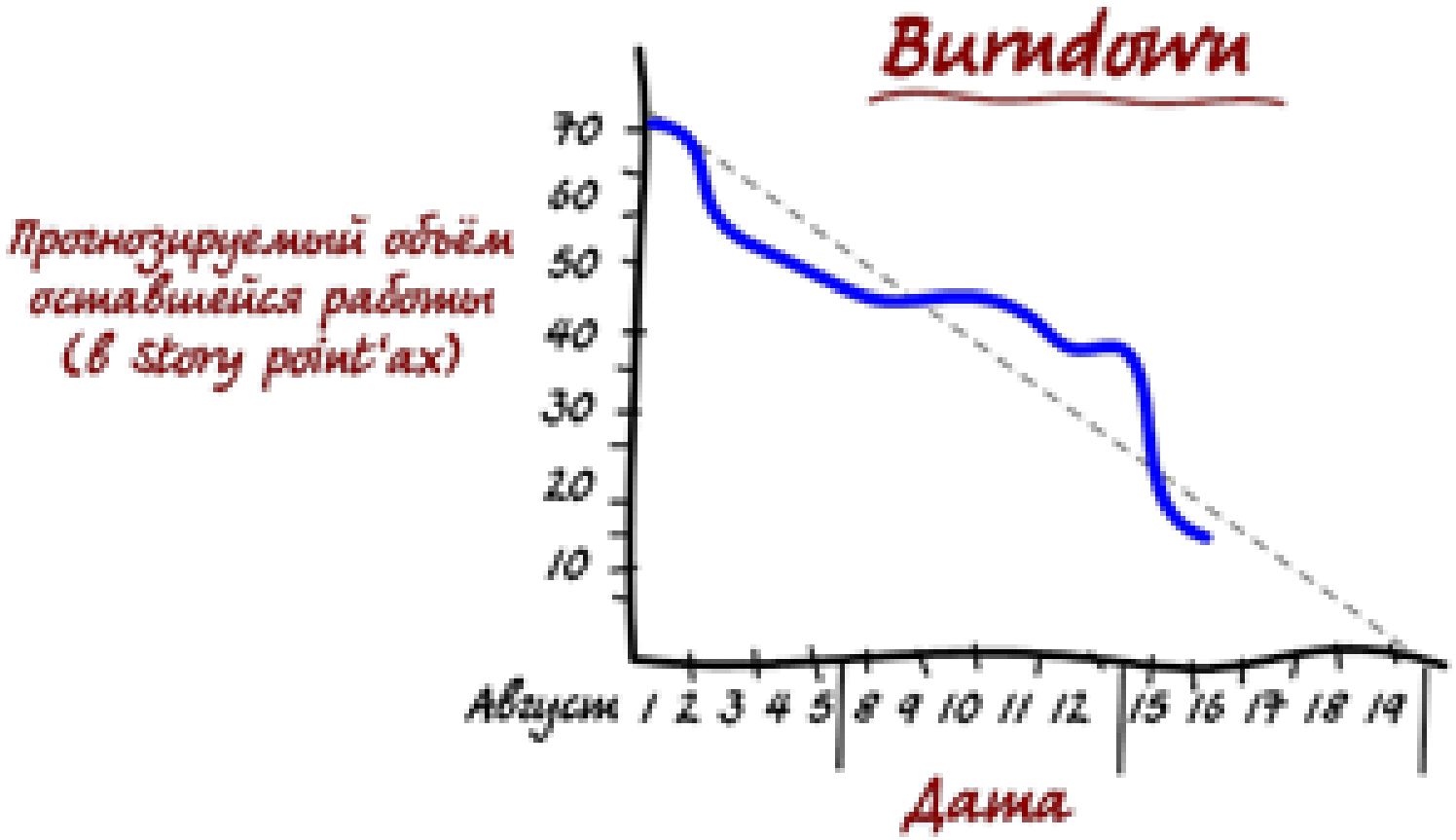
Оценка объема работ



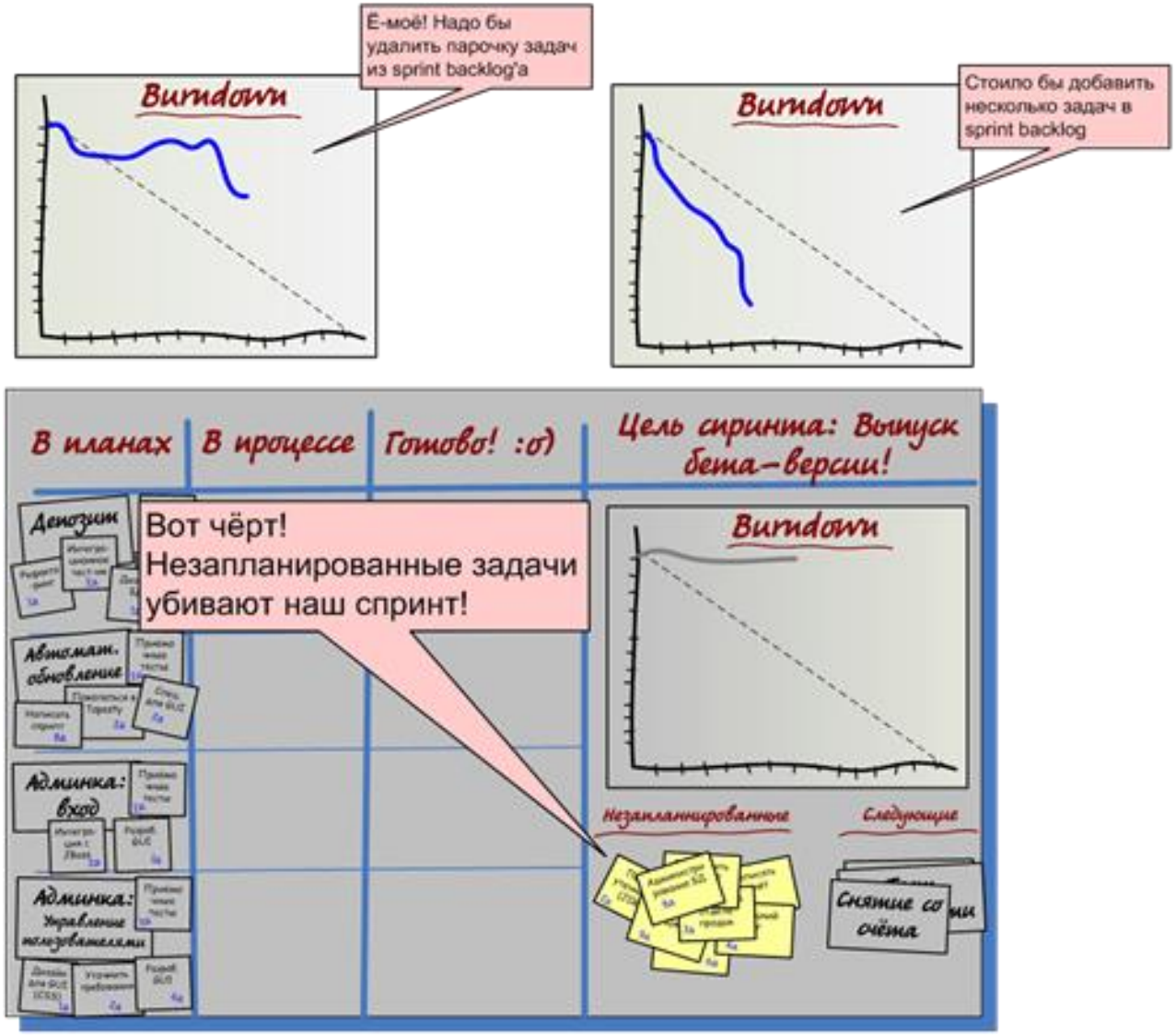
Доска SCRUM


То, чем сегодня никто не занимается
То, чем сегодня кто-то занимается
То, чем больше никто не будет заниматься
Ставьте маркером новую точку на burndown-диаграмме каждый день после ежедневного Scrum'а
В планах
В процессе
Готово! :o)
Цель спринта
Выпуск бета-версии!
В начале стикеры задач перемещаются в этом направлении
Потом белая карточка истории перемещается в "Готово"
Burndown
Автомат. обновление
Приёмочные тесты
Написать скрипт
Спец. для GUI
Админка: вход
Приёмочные тесты
Интеграция с JBoss
Разраб. GUI
Админка: Управление пользователями
Приёмочные тесты
Дизайн для GUI (CSS)
Уточнить требования
Разраб. GUI
Незапланированные
Следующие
Снятие со счёта
Если все истории готовы, а спринт ещё не закончился – добавь новые отсюда

Оценка производительности



Оценка производительности



KANBAN – «конвейерная» разработка

ЦЕЛИ KANBAN:

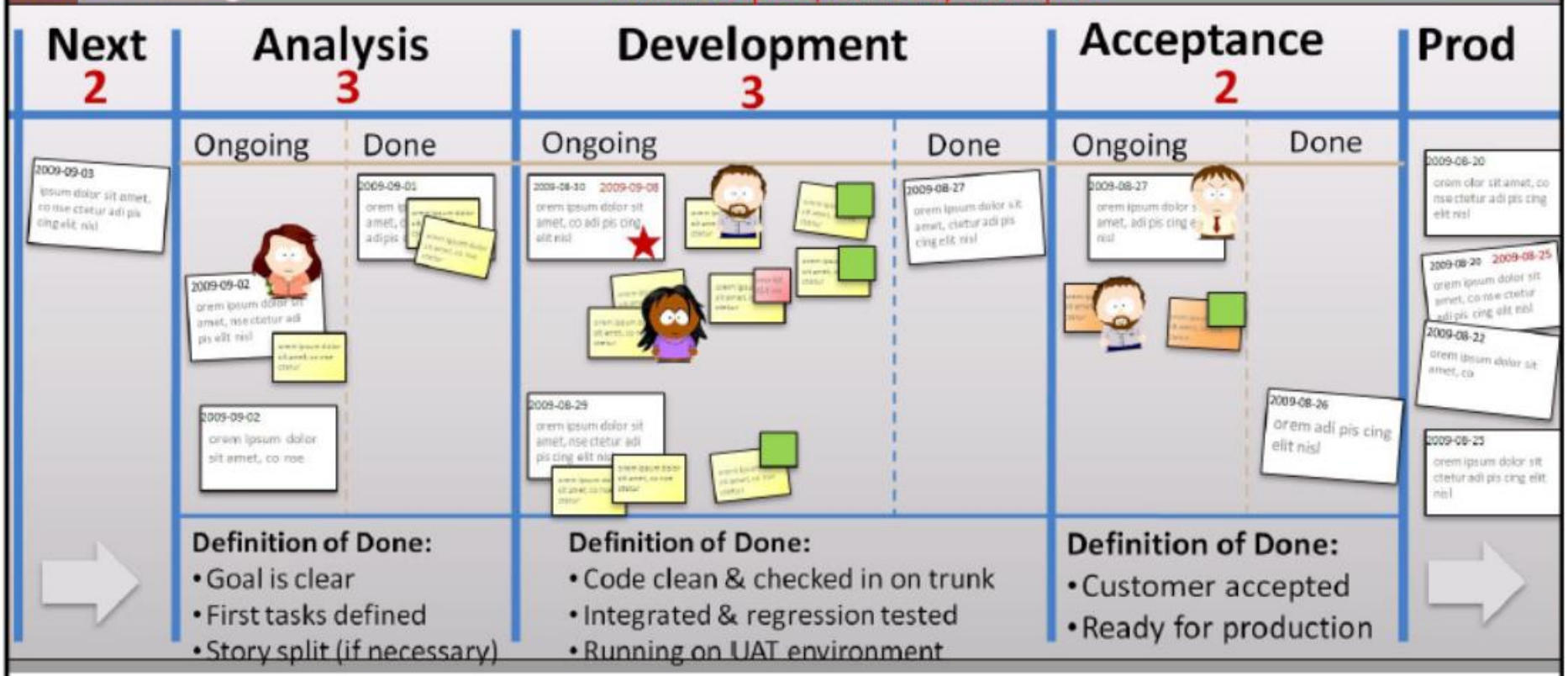
- Визуализация процесса разработки
- Ограничение числа незавершённых задач в процессе разработки
- Снижение среднего времени исполнения задач
- Поиск потерь (оптимизация процесса разработки)

Henrik Kniberg

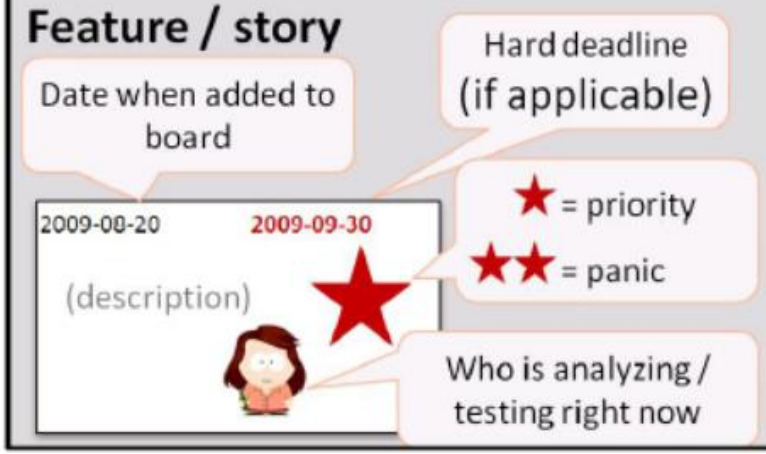
Пример канбан

www.crisp.se/kanban/example

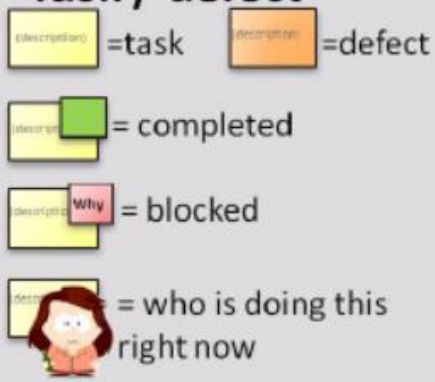
version 1.2
2009-11-16



Feature / story

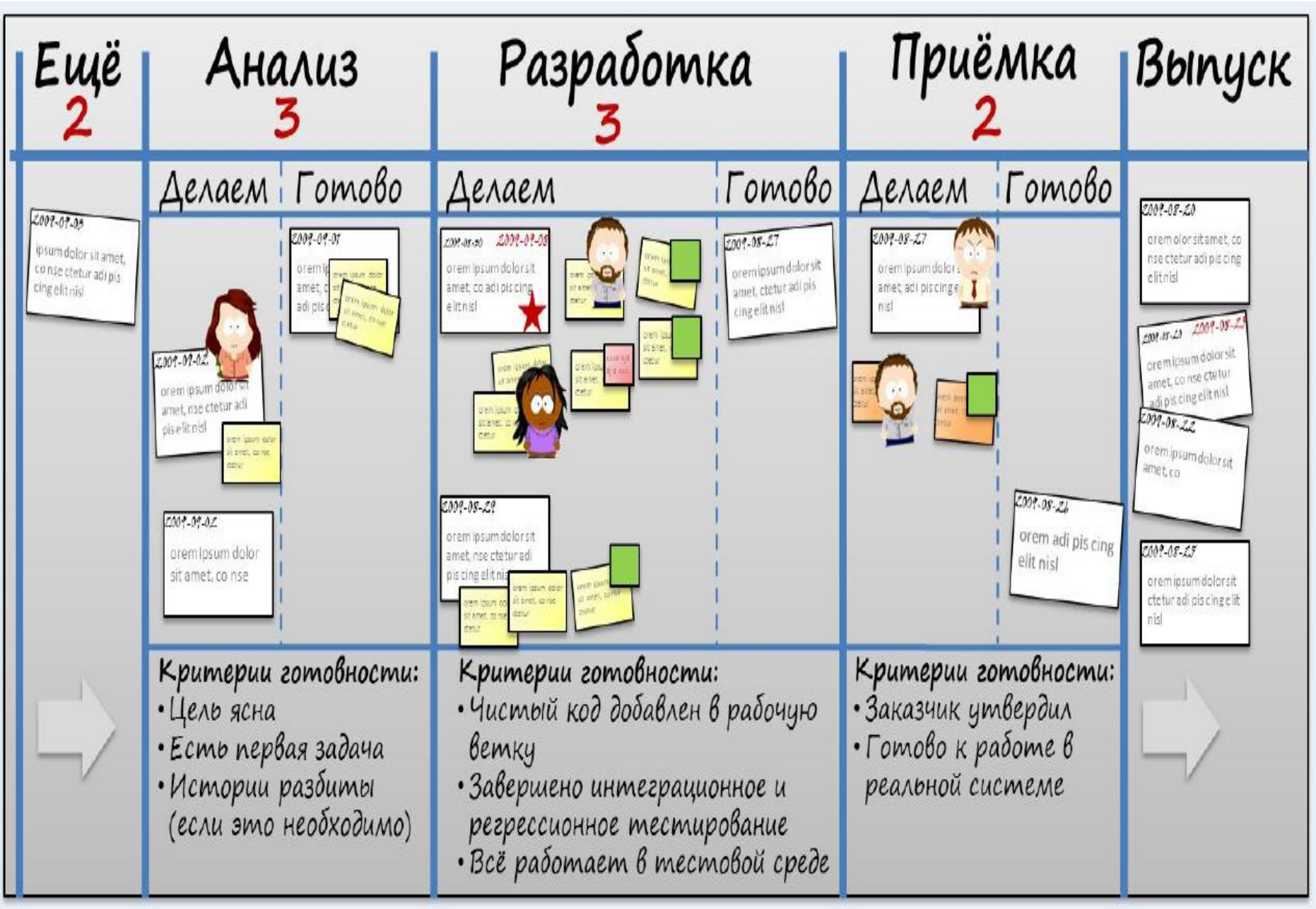


Task / defect



What to pull first

- **Panic**features ★★
  (should be swarmed and kept moving. Interrupt other work and break WIP limits as necessary)
- **Priority** features ★
- Hard **deadline** features
  (only if deadline is at risk)
- **Oldest** features


Ещё
2
Анализ
3
Разработка
3
Приёмка
2
Выпуск
Делаем
Готово
Делаем
Готово
Делаем
Готово
2009-07-05
ipsum dolor sit amet, co nse ctetur adi pis cing elit nisl
2009-07-01
2009-07-02
orem ipsum dolor sit amet, nse ctetur adi pis elit nisl
2009-07-12
orem ipsum dolor sit amet, co nse
2009-08-20
2009-07-08
orem ipsum dolor sit amet, co adi pis cing elit nisl
2009-08-27
orem ipsum dolor sit amet, ctetur adi pis cing elit nisl
2009-08-27
orem ipsum dolor sit amet, adi pis cing elit nisl
2009-08-29
orem ipsum dolor sit amet, nse ctetur adi pis cing elit nisl
2009-08-26
orem adi pis cing elit nisl
2009-08-20
orem olor sit amet, co nse ctetur adi pis cing elit nisl
2009-08-23
2009-08-23
orem ipsum dolor sit amet, co nse ctetur adi pis cing elit nisl
2009-08-22
orem ipsum dolor sit amet, co
2009-08-25
orem ipsum dolor sit ctetur adi pis cing elit nisl
Критерии готовности:
• Цель ясна
• Есть первая задача
• Истории разбиты (если это необходимо)
Критерии готовности:
• Чистый код добавлен в рабочую ветку
• Завершено интеграционное и регрессионное тестирование
• Всё работает в тестовой среде
Критерии готовности:
• Заказчик утвердил
• Готово к работе в реальной системе

Выбор модели процесса

Вес модели	Плюсы	Минусы
Тяжелые	Процессы рассчитаны на среднюю квалификацию исполнителей. Большая специализация исполнителей. Ниже требования к стабильности команды. Отсутствуют ограничения по объему и сложности выполняемых проектов.	Требуют существенной управленческой надстройки. Более длительные стадии анализа и проектирования. Более формализованные коммуникации.
Легкие	Меньше непроизводительных расходов, связанных с управлением проектом, рисками, изменениями, конфигурациями. Упрощенные стадии анализа и проектирования, основной упор на разработку функциональности, совмещение ролей. Неформальные коммуникации.	Эффективность сильно зависит от индивидуальных способностей, требуют более квалифицированной, универсальной и стабильной команды. Объем и сложность выполняемых проектов ограничены.

Выбор модели процесса

Более директивные — Более адаптивные

RUP (120+)

- Architecture Reviewer
- Business Designer
- Business-Model Reviewer
- Business-Process Analyst
- Capsule Designer
- Change Control Manager
- Code Reviewer
- Configuration Manager
- Course Developer
- Database Designer
- Deployment Manager
- Design Reviewer
- Designer
- Graphic Artist
- Implementer
- Integrator
- Process Engineer
- Project Manager
- Project Reviewer
- Requirements Reviewer
- Requirements Specifier
- Software Architect
- Stakeholder
- System Administrator
- System Analyst
- Technical Writer
- Test Analyst
- Test Designer
- Test Manager
- Tester
- Tool Specialist
- User-Interface Designer
- Architectural analysis
- Assess Viability of architectural proof-of-concept
- Capsule design
- Class design
- Construct architectural proof-of-concept
- Database design
- Describe distribution
- Describe the run-time architecture
- Design test packages and classes
- Develop design guidelines
- Develop programming guidelines
- Identify design elements
- Identify design mechanisms
- Incorporate design elements
- Prioritize use cases
- Review the architecture
- Review the design
- Structure the implementation model
- Subsystem design
- Use-case analysis
- Use-case design
- Analysis model
- Architectural proof-of-concept
- Bill of materials
- Business architecture document
- Business case
- Business glossary
- Business modeling guidelines
- Business object model
- Business rules
- Business use case
- Business use case realization
- Business use-case model
- Business vision
- Change request
- Configuration audit findings
- Configuration management plan
- Data model
- Deployment model
- Deployment plan
- Design guidelines
- Design model
- Development case
- Development-organization assessment
- End-user support materials
- Glossary
- Implementation model
- Installation artifacts
- Integration build plan
- Issues list
- Iteration assessment
- Iteration plan
- Manual styleguide
- Programming guidelines
- Quality assurance plan
- Reference architecture
- Release notes
- Requirements attributes
- Requirements management plan
- Review record
- Risk list
- Risk management plan
- Software architecture document
- Software development plan
- Software requirements specification
- Stakeholder requests
- Status assessment
- Supplementary business specification
- Supplementary specification
- Target organization assessment
- Test automation architecture
- Test cases
- Test environment configuration
- Test evaluation summary
- Test guidelines
- Test ideas list
- Test interface specification
- Test plan
- Test suite
- Tool guidelines
- Training materials
- Use case model
- Use case package
- Use-case modeling guidelines
- Use-case realization
- Use-case storyboard
- User-interface guidelines
- User-interface prototype
- Vision
- Work order
- Workload analysis model

XP (13)

- Whole team
- Coding standard
- TDD
- Collective ownership
- Customer tests
- Pair programming
- Refactoring
- Planning game
- Continuous integration
- Simple design
- Sustainable pace
- Metaphor
- Small releases

Scrum (9)

- Scrum Master
- Product Owner
- Team
- Sprint planning meeting
- Daily Scrum
- Sprint review
- Product backlogt
- Sprint backlog
- BUrndown chart

Kanban (3)

- Visualize the workflow
- Limit WIP
- Measure and optimize lead time

Делай что хочешь (0)

Выбор модели процесса

Процесс в проекте должен определяться в зависимости от проекта, продукта и персонала

Выбор модели процесса

- Основной стандарт в области ЖЦ ПС и систем в Республике Беларусь является стандарт *СТБ ИСО/МЭК 12207–2003*.

В соответствии с данным стандартом выбор модели ЖЦ должен осуществляться при выполнении первой работы процесса разработки.

В соответствии с положениями стандарта, если модель ЖЦ ПС не определена в договоре, то разработчик должен определить или выбрать модель, соответствующую области реализации, величине и сложности проекта.

Выбор модели процесса

- В соответствии с положениями стандартов *СТБ ИСО/МЭК 12207–2003* и *ГОСТ Р ИСО/МЭК ТО 15271–2002* вопросы структурирования в выбранной модели работ и задач процесса разработки должны решаться с учетом следующих ***характеристик проекта:***

1. *организационные подходы* (например, связанные с защитой, безопасностью, конфиденциальностью, управлением рисками, использованием независимого органа по верификаци и и аттестации, использованием конкретного языка программирования, обеспечением техническими ресурсами);

Выбор модели процесса

2. политика заказа (например, тип договора, необходимость использования процесса поставки, использование услуг сторонних разработчиков или субподрядчиков, которых необходимо контролировать);

3. политика сопровождения программных средств (например ожидаемые период сопровождения и периодичность внесения изменений, критичность применения, квалификация персонала сопровождения, необходимая для сопровождения среда);

Выбор модели процесса

4. вовлеченные стороны (например заказчик, поставщик, разработчик, субподрядчик, посредники по верификации и аттестации, персонал сопровождения; численность сторон);

5. работы жизненного цикла системы (например подготовка проекта заказчиком, разработка и сопровождение поставщиком);

6. характеристики системного уровня (например количество подсистем и объектов конфигурации, межсистемные и внутрисистемные интерфейсы, интерфейсы пользователя, оценка временных ограничений, наличие реализованных техническими средствами программ);

Выбор модели процесса

- *7. характеристики программного уровня* (например, количество программных объектов, типы документов, характеристики качества ПС по *ISO/IEC 9126–1:2001* и СТБ ИСО/МЭК 9126–2003, типы и объемы программных продуктов);
- *8. объем проекта* (в больших проектах, в которые вовлечены десятки илисотни лиц, необходим административный надзор и контроль с применением процессов совместного анализа, аудита, верификации, аттестации, обеспечения качества; данные процессы для малых проектов такие методы контроля могут быть излишними);

Выбор модели процесса

- *9. критичность проекта* (значительная зависимость работы системы от правильного функционирования ПС и своевременности выдачи результатов);
- *10. технические риски* (например, создание уникального или сложного программного средства, которое трудно сопровождать и использовать, неправильное, неточное или неполное определение требований);
- *11. другие характеристики* (например, усиленный административный контроль за критичными или большими программными продуктами, требующий применения постоянных оценок).

Оценка стоимости проекта

- **Экспертная оценка**. Делается на основании списка работ. Дается оптимистическая, пессимистическая, вероятностная (средняя или на основании статистики) оценки.

Метод «Дельфи». Его суть в том, чтобы использовать группу экспертов и с помощью серии последовательных итераций добиться максимального консенсуса при определении группового решения.
После двух-трёх туров оценки сближаются и фиксируются (оптимистическая, пессимистическая, средняя)

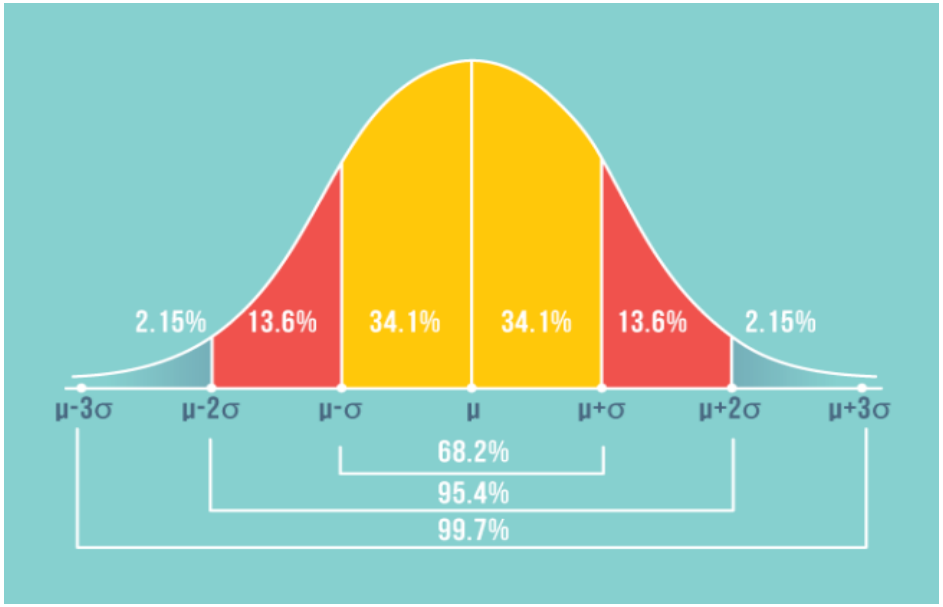
Planning Poker

Оценка стоимости проекта

На основании трёх оценок выбирается вид распределения вероятности и формулы для математических вычислений оценки.

Н.Талеб в книге **«Черный лебедь»** доказал, что распределение Гаусса способно прогнозировать только те события и величины, которые принадлежат пространству измерений, где ни одно наблюдение не может оказывать значительного влияния на результаты.

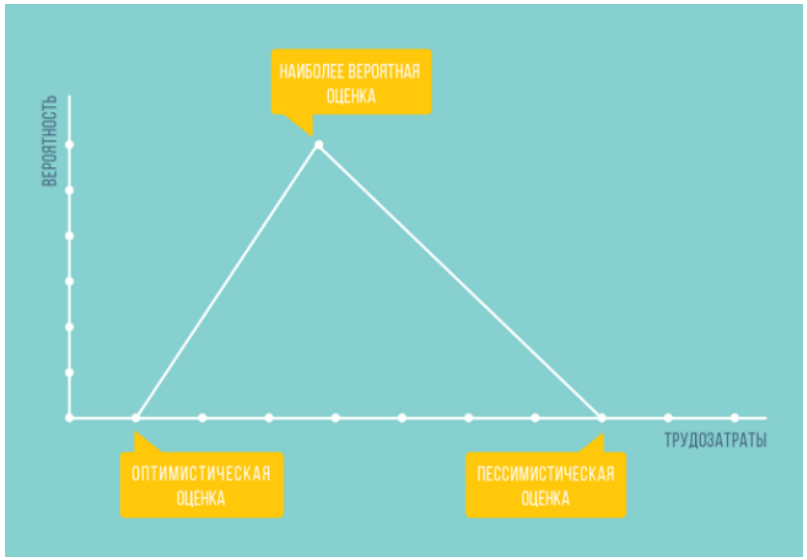
Нормальное распределение (Гаусса)



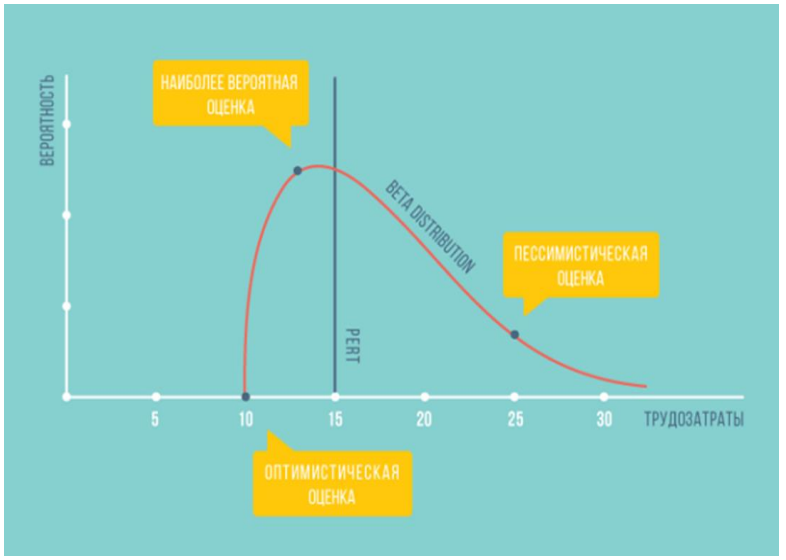
Оценка стоимости проекта

Проектные работы относятся к той категории явлений, что не описываются нормальным распределением. Поэтому математики и предложили для прогнозирования трудозатрат, сроков и бюджетов использовать бета-распределение или треугольное распределение вероятности.

Треугольное распределение

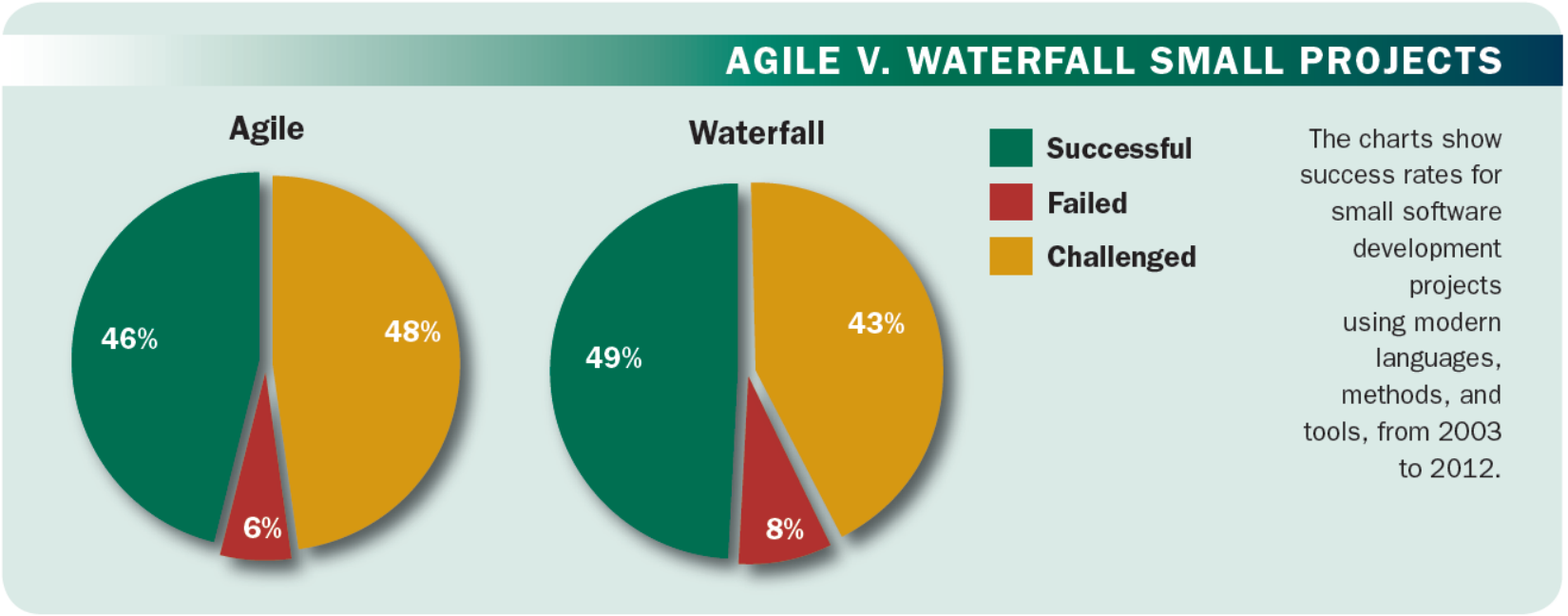


BETA-распределение



Сравнение успешности процессов

Гибкие методологии против каскадной разработки для малых проектов



Выводы

- Не существует единственного правильного процесса разработки ПО.
- Эффективный производственный процесс должен основываться на итеративности, инкрементальности, самоуправляемости команды и адаптивности.
- Главный принцип: не люди должны строиться под выбранную модель процесса, а модель процесса должна подстраиваться под конкретную команду, чтобы обеспечить ее наивысшую производительность.

Тест шансов проекта

Стив Макконнелл (*гл.редактор IEEE Software*) приводит тест программного проекта на выживание. Руководитель программного проекта должен его периодически использовать для внутреннего аудита своих процессов.

Чтобы программный проект стал успешным, необходимо:

- 1. Четко ставить цели.
- 2. Определять способ достижения целей.
- 3. Контролировать и управлять реализацией.
- 4. Анализировать угрозы и противодействовать им.
- 5. Создавать команду.

Тест шансов проекта

- 1. **Ставим цели**
- 1.1. Концепция определяет ясные недвусмысленные цели.
- 1.2. Все члены команды считают концепцию реалистичной.
- 1.3. У проекта имеется обоснование экономической эффективности.
- 1.4. Разработан прототип пользовательского интерфейса.
- 1.5. Разработана спецификация целевых функций программного продукта.
- 1.6. С конечными пользователями продукта налажена двухсторонняя связь.

Тест шансов проекта

- **2. Определяем способ достижения целей**
- 2.1. Имеется детальный письменный план разработки продукта.
- 2.2. В список задач проекта включены «второстепенные» задачи (управление конфигурациями, конвертация данных, интеграция с другими системами).
- 2.3. После каждой фазы проекта обновляется расписание и бюджет.
- 2.4. Архитектура и проектные решения документированы.
- 2.5. Имеется план обеспечения качества, определяющий тестирование и рецензирование.
- 2.6. Определен план многоэтапной поставки продукта.
- 2.7. В плане учтены обучение, выходные, отпуска, больничные.
- 2.8. План проекта и расписание одобрен всеми участниками команды.

Тест шансов проекта

- **3. Контролируем и управляем реализацией**
- 3.1. У проекта есть куратор. Это такой топ-менеджер исполняющей компании, который лично заинтересован в успехе данного проекта.
- 3.2. У проекта есть менеджер, причем только один!
- 3.3. В плане проекта определены «бинарные» контрольные точки.
- 3.4. Все заинтересованные стороны могут получить необходимую информацию о ходе проекта.
- 3.5. Между руководством и разработчиками установлены доверительные отношения.

Тест шансов проекта

- 3.6. Установлена процедура управления изменениями в проекте.
- 3.7. Определены лица, ответственные за решение о принятии изменений в проекте.
- 3.8. План, расписание и статусная информация по проекту доступна каждому участнику.
- 3.9. Код системы проходит автоматическое рецензирование.
- 3.10. Применяется система управления дефектами.

Тест шансов проекта

- **4. Анализируем угрозы**
- 4.1. Имеется список рисков проекта. Осуществляется его регулярный анализ и обновление.
- 4.2. Руководитель проекта отслеживает возникновение новых рисков.
- 4.3. Для каждого подрядчика определено лицо, ответственное за работу с ним.
- **5. Работаем над созданием команды**
- 5.1. Опыт команды достаточен для выполнения проекта.
- 5.2. У команды достаточная компетенция в прикладной области.
- 5.3. В проекте имеется технический лидер.
- 5.4. Численность персонала достаточна.
- 5.5. У команды имеется достаточная сплоченность.
- 5.6. Все участники привержены проекту.

Тест шансов проекта

- **Оценка и интерпретация теста**

Оценка: сумма баллов, каждый пункт оценивается от 0 до 3:

0 – даже не слышали об этом;

1 – слышали, но пока не применяем;

2 – применяется частично;

3 – применяется в полной мере.

- **Поправочные коэффициенты:**
- для малых проектов (до 5 человек) - 1.5;
- для средних (от 5 до 20 человек) – 1.25.

Тест шансов проекта

- **Результат:**

< 40 – завершение проекта сомнительно.

40-59 – средний результат. В ходе проекта следует ожидать серьезные проблемы.

60-79 – хороший результат. Проект, скорее всего, будет успешным.

80-89 – отличный результат. Вероятность успеха высока.

> 90 – великолепный результат. 100% шансов на успех.

Заключение

Имеется достаточно много методологий разработки ПО, от полного хаоса до крайне формализованных. Каждая из них имеет свои плюсы и минусы, а её конкретный выбор часто зависит от «закон четырёх П»: **Процесс** в проекте определяется **Продукт**ом (который мы создаём), **Проект**ом (в рамках которого должны работать) и **Персонал**ом (который имеется у нас в наличии).

При этом полезно помнить одно условие: *при разработке ПО, всё без исключения, должно делаться для того, чтобы ускорить процесс создания программного кода, обеспечив приемлемое для заказчика качество продукта*.

Литература

1. «PMBOK. Руководство к Своду знаний по управлению проектами», 4-е изд., PMI, 2008.
2. «Руководство к своду знаний по программной инженерии». The Guide to the Software Engineering Body of Knowledge, SWEBOK, IEEE Computer Society Professional Practices Committee, 2004.
3. Сергей Архипенков «Лекции по управлению программными проектами», 2009.
4. А. Якобсон, Г. Буч, Дж. Рамбо Унифицированный процесс разработки программного обеспечения. – СПб.: ПИТЕР, 2002. – 496с. ISBN 5-318-00358-3
5. Борис Вольфсон «Гибкие методологии разработки», 2014, Интернет-издание. http://www.facebook.com/borisvolfson
6. Алексей Просницкий, Владимир Иванов, «Управление проектами в Microsoft Project 2010», 2012, Интернет-издание, www.microsoftproject.ru