

**ПРОЦЕССЫ УПРАВЛЕНИЯ  
ПРОЕКТОМ,  
ОРГАНИЗАЦИОННАЯ  
СТРУКТУРА**

**ЛЕКЦИЯ 2**

**МОСКВА, 2014**



# ПРОЦЕССЫ УПРАВЛЕНИЯ ПРОЕКТОМ

**Управление проектами** – это приложение знаний, навыков, инструментов и методов к работам проекта для удовлетворения требований, предъявляемых к проекту.

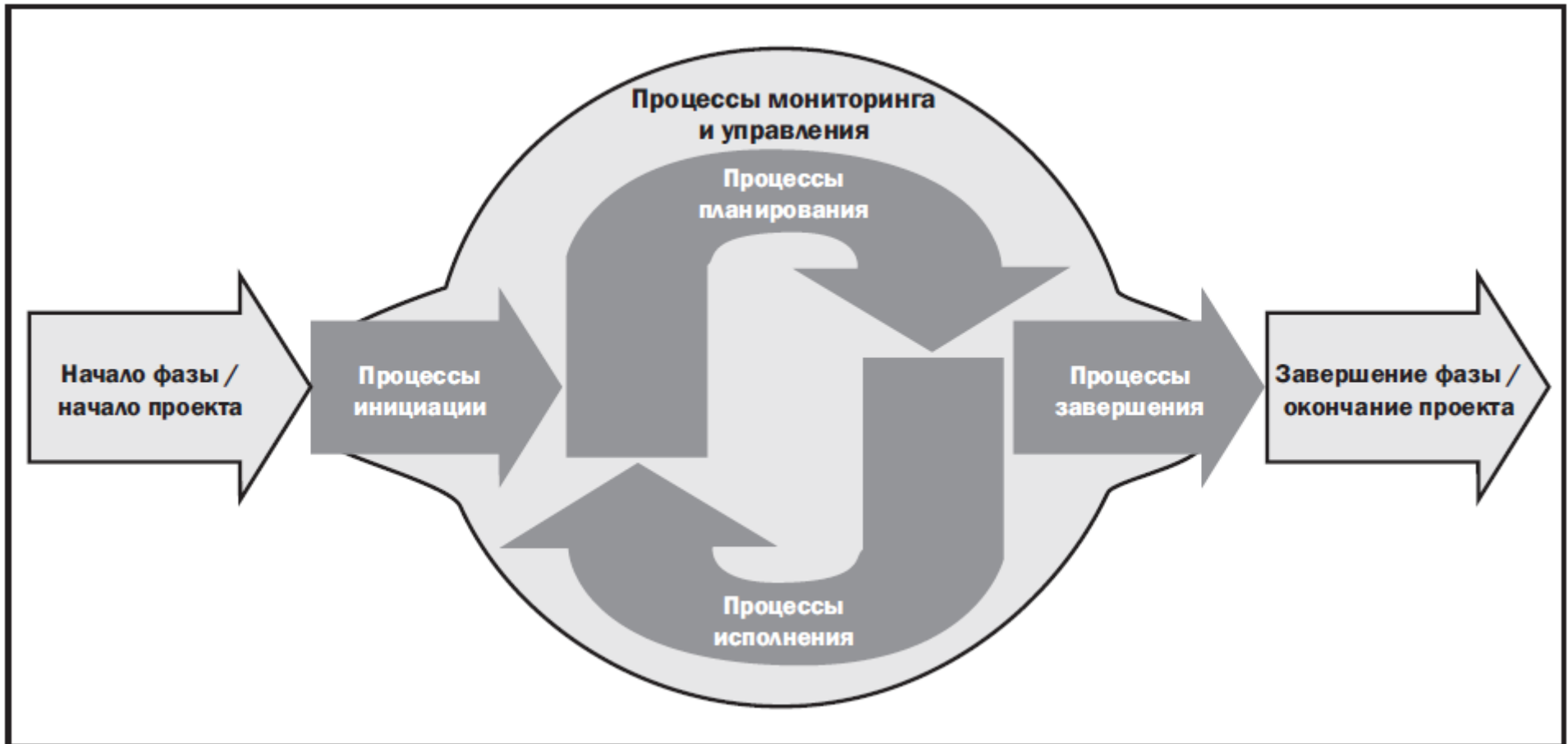
Это приложение знаний требует эффективного управления соответствующими процессами.

# ОПРЕДЕЛЕНИЕ ПРОЦЕССА

**Процесс** – это набор взаимосвязанных действий и операций, осуществляемых для получения заранее определенного продукта, результата или услуги.

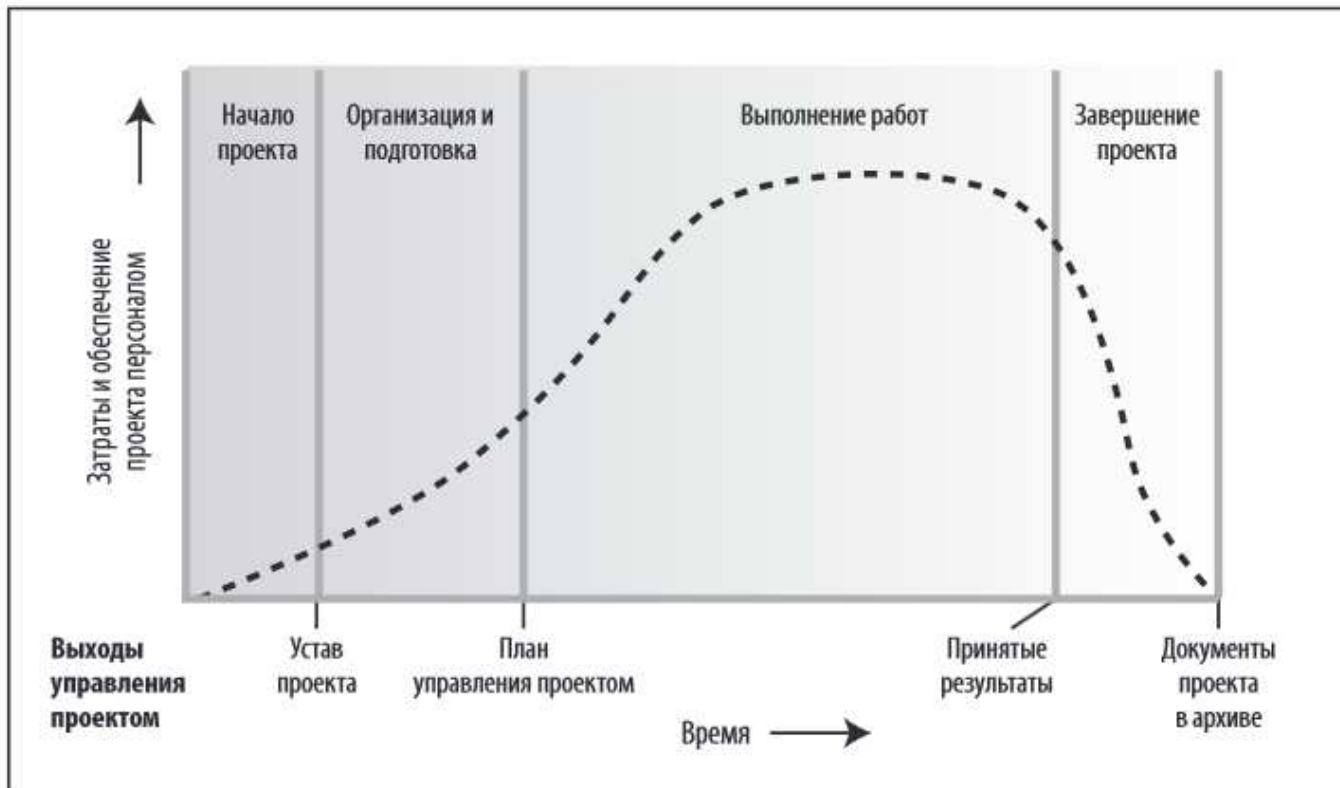
Каждый процесс характеризуется своими **входами**, **инструментами** и **методами**, которые могут быть применены, а также конечными **выходами**.

# ГРУППЫ ПРОЦЕССОВ УПРАВЛЕНИЯ ПРОЕКТАМИ



# ЖИЗНЕННЫЙ ЦИКЛ ПРОЕКТА

Жизненный цикл проекта – это набор, как правило, последовательных и иногда перекрывающихся фаз проекта, названия и количество которых определяются потребностями в управлении и контроле организации или организаций, вовлеченных в проект, характером самого проекта и его прикладной областью.



# ВЛИЯНИЕ ПЕРЕМЕННОЙ, ОСНОВАННОЙ НА СРОКАХ ПРОЕКТА



# ФАЗЫ ПРОЕКТА

Фазы проекта – это отдельные части в рамках проекта, требующие дополнительного контроля для эффективного управления достижением основного результата проекта. Фазы проекта обычно выполняются последовательно, но в некоторых случаях могут перекрываться.

Структура фаз позволяет разделить проект на логические подгруппы для более легкого управления, планирования и контроля.



Рис. 2-3. Пример однофазного проекта

# ФАЗЫ ПРОЕКТА (ПРОДОЛЖЕНИЕ)

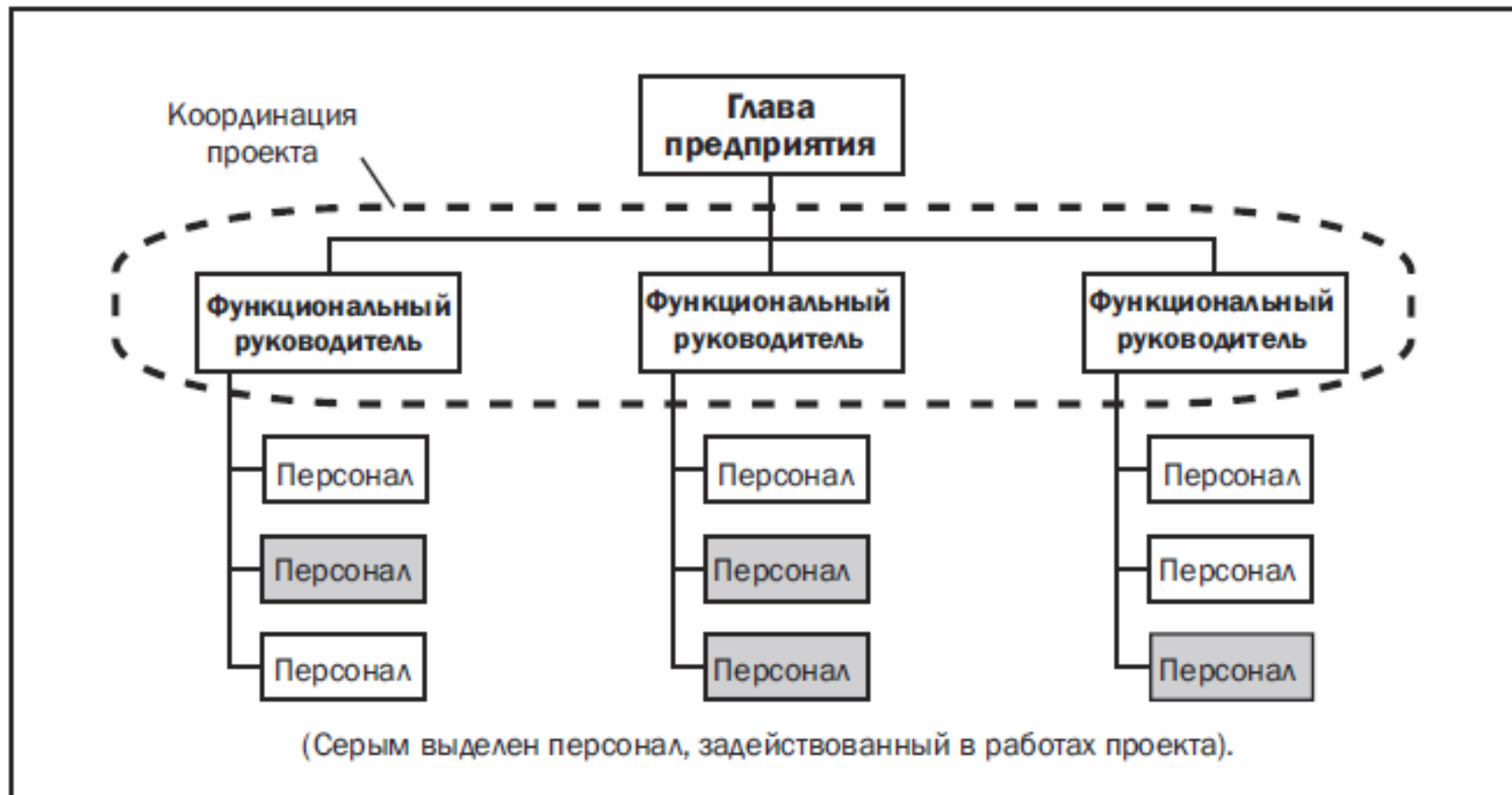


Рис. 2-4. Пример трехфазного проекта

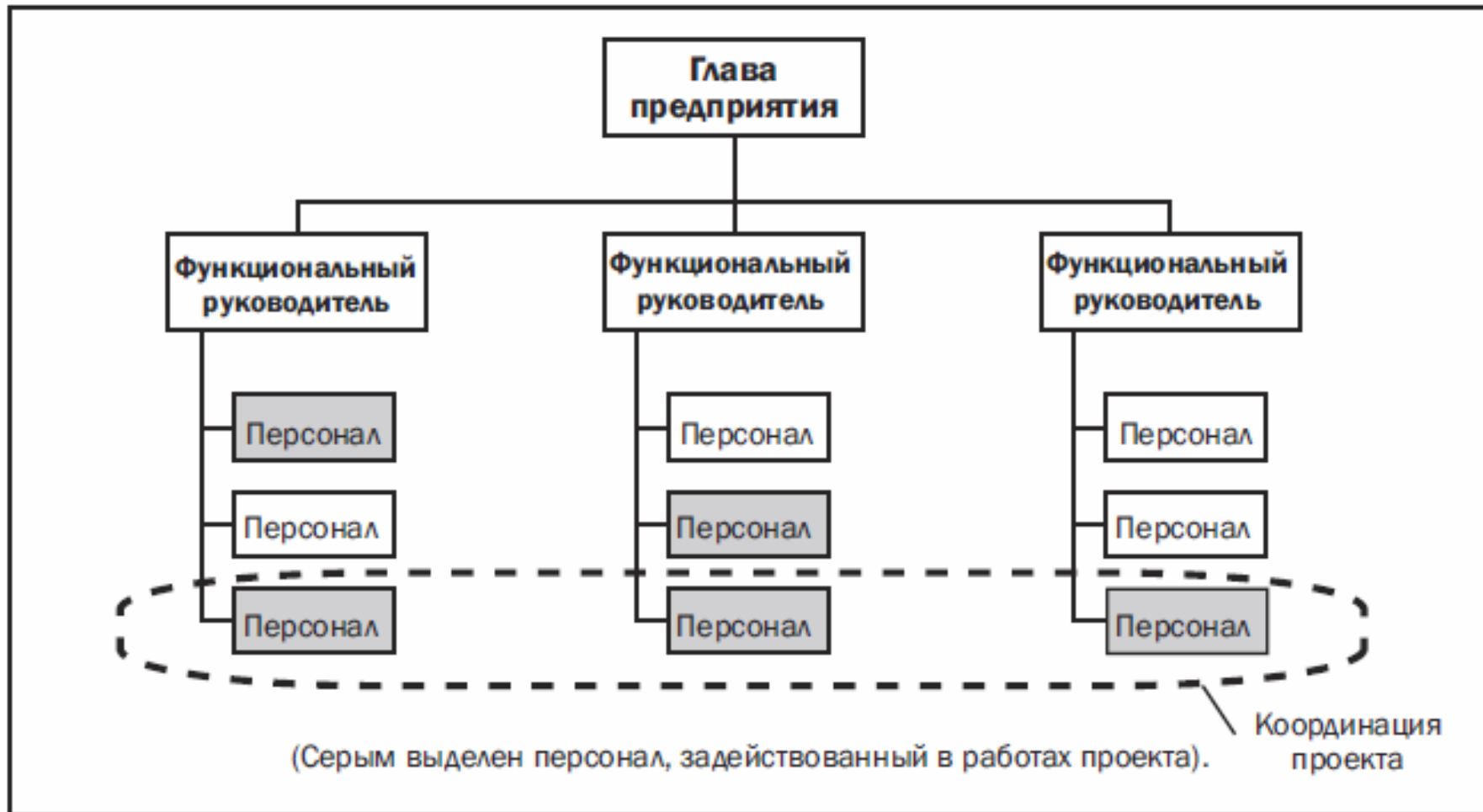




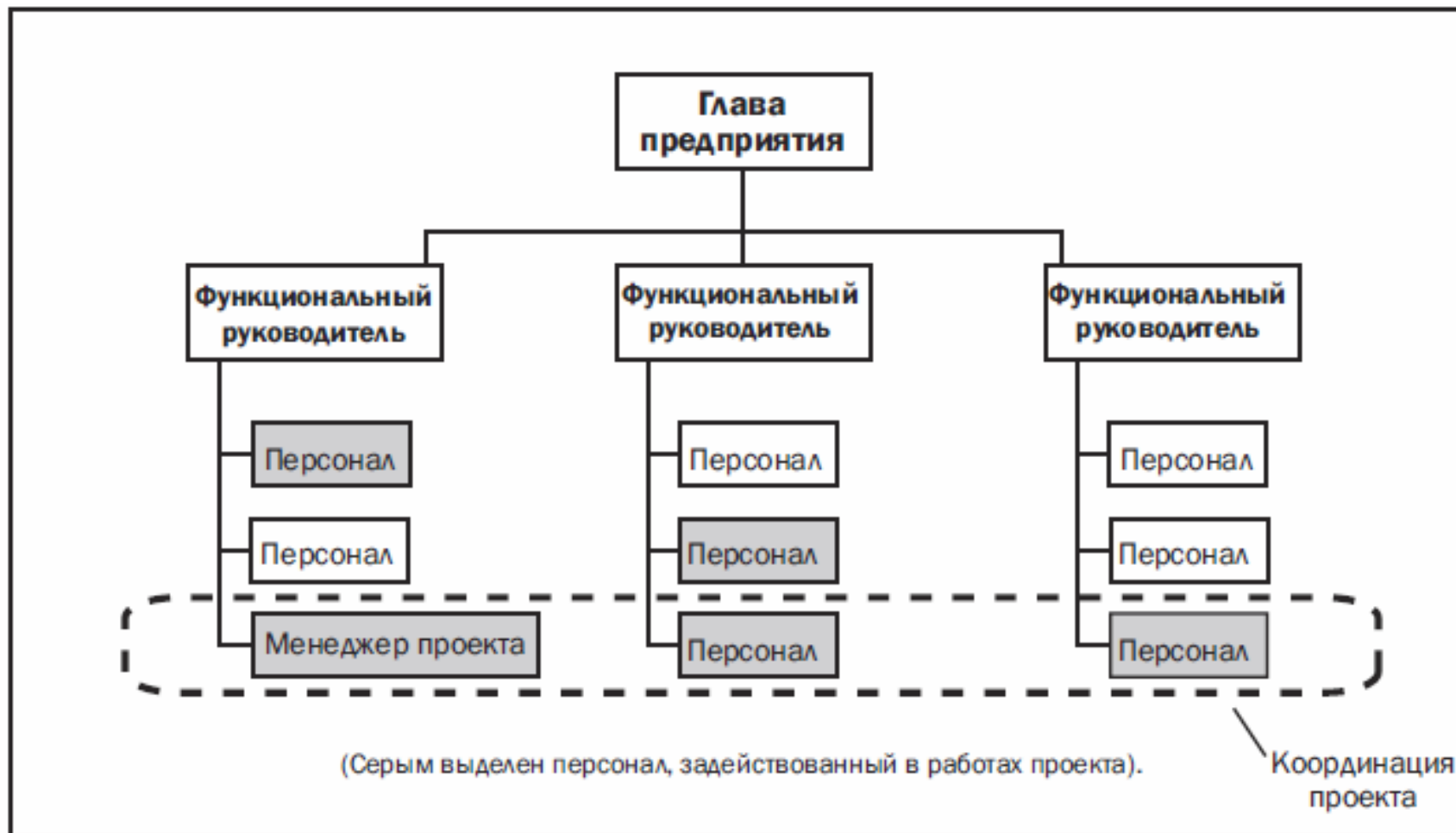
# ОРГАНИЗАЦИОННАЯ СТРУКТУРА. ФУНКЦИОНАЛЬНАЯ ОРГАНИЗАЦИЯ



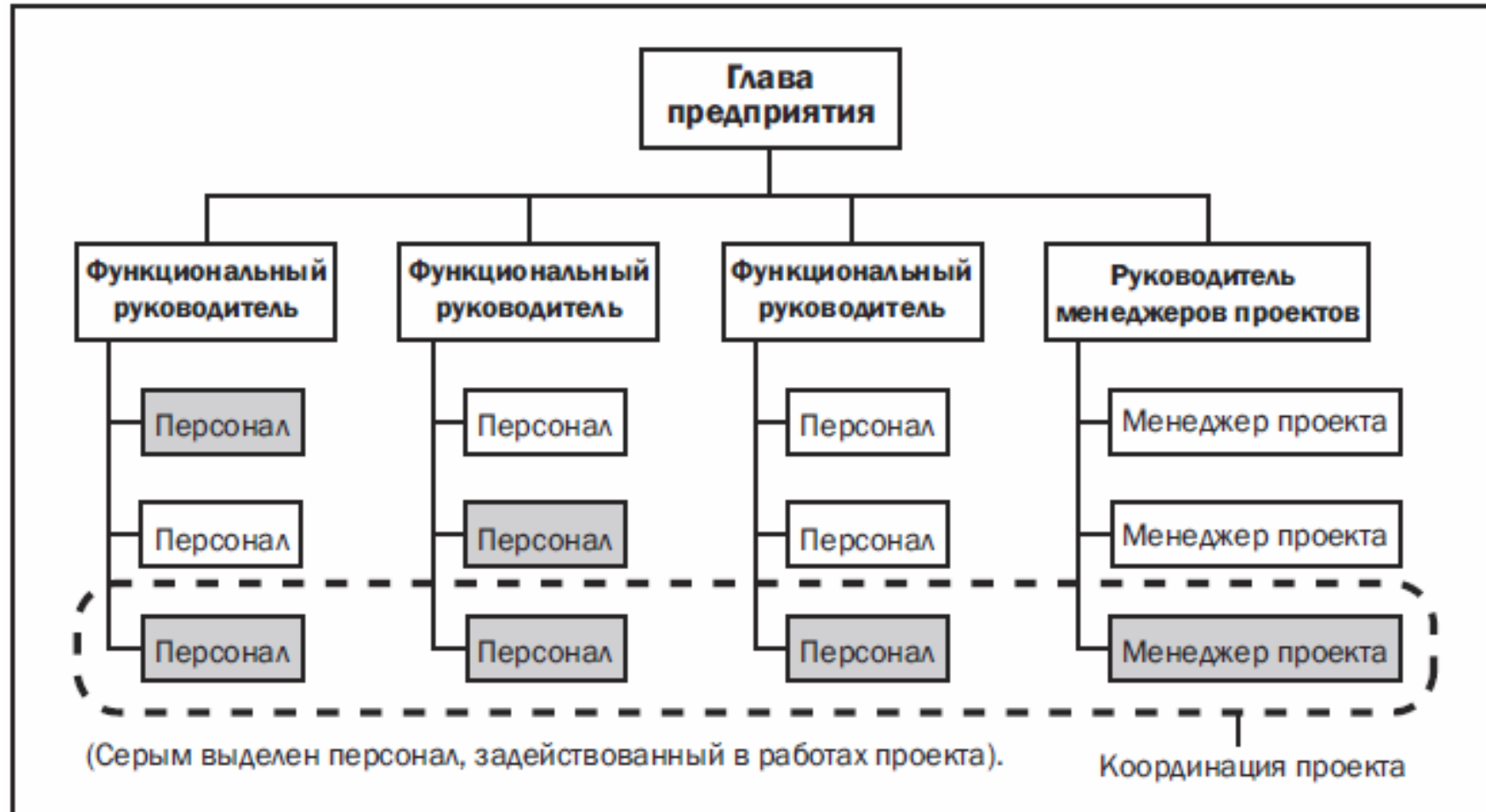
# ОРГАНИЗАЦИОННАЯ СТРУКТУРА. СЛАБАЯ МАТРИЧНАЯ ОРГАНИЗАЦИЯ



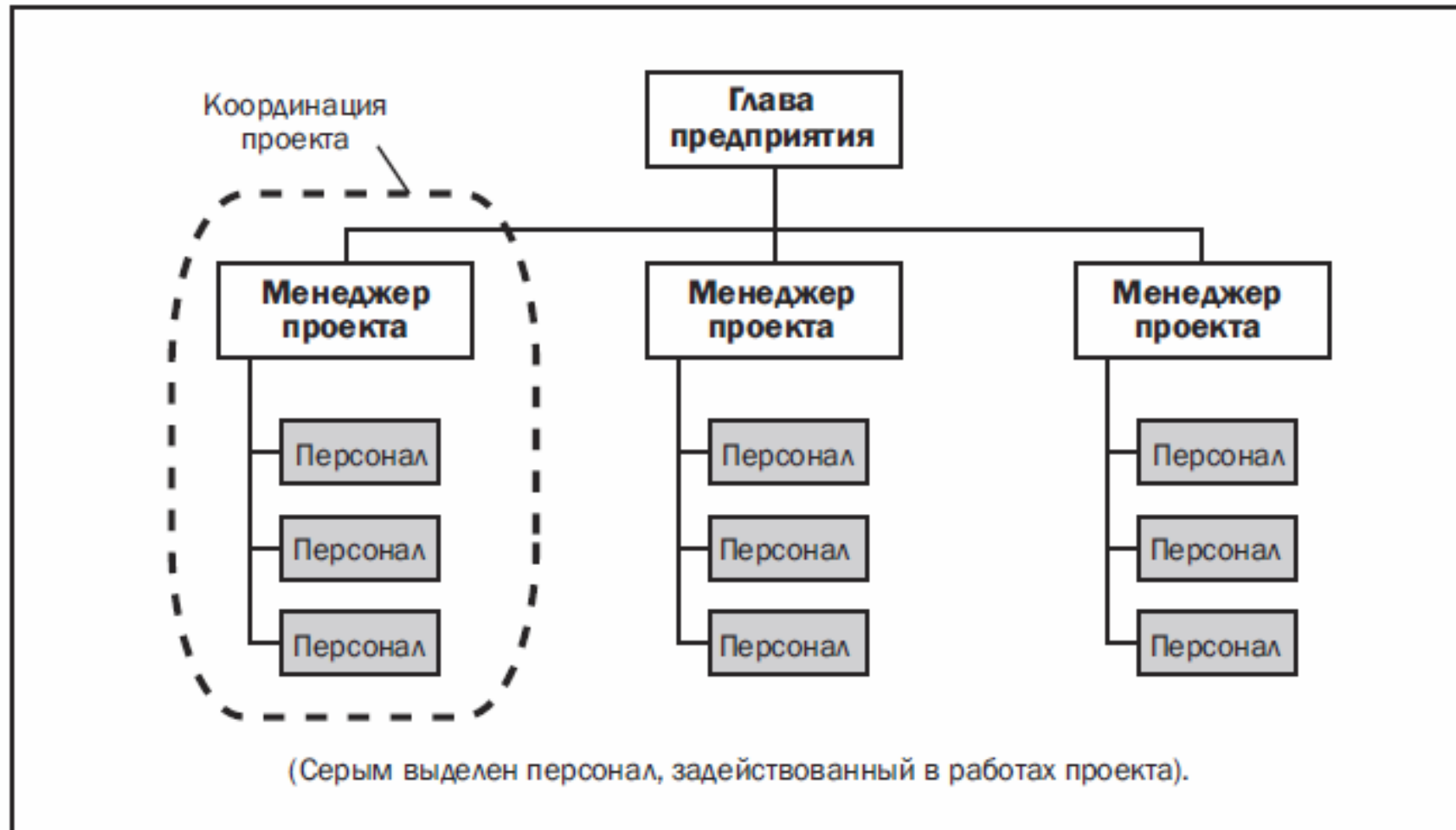
# ОРГАНИЗАЦИОННАЯ СТРУКТУРА. СБАЛАНСИРОВАННАЯ МАТРИЧНАЯ ОРГАНИЗАЦИЯ



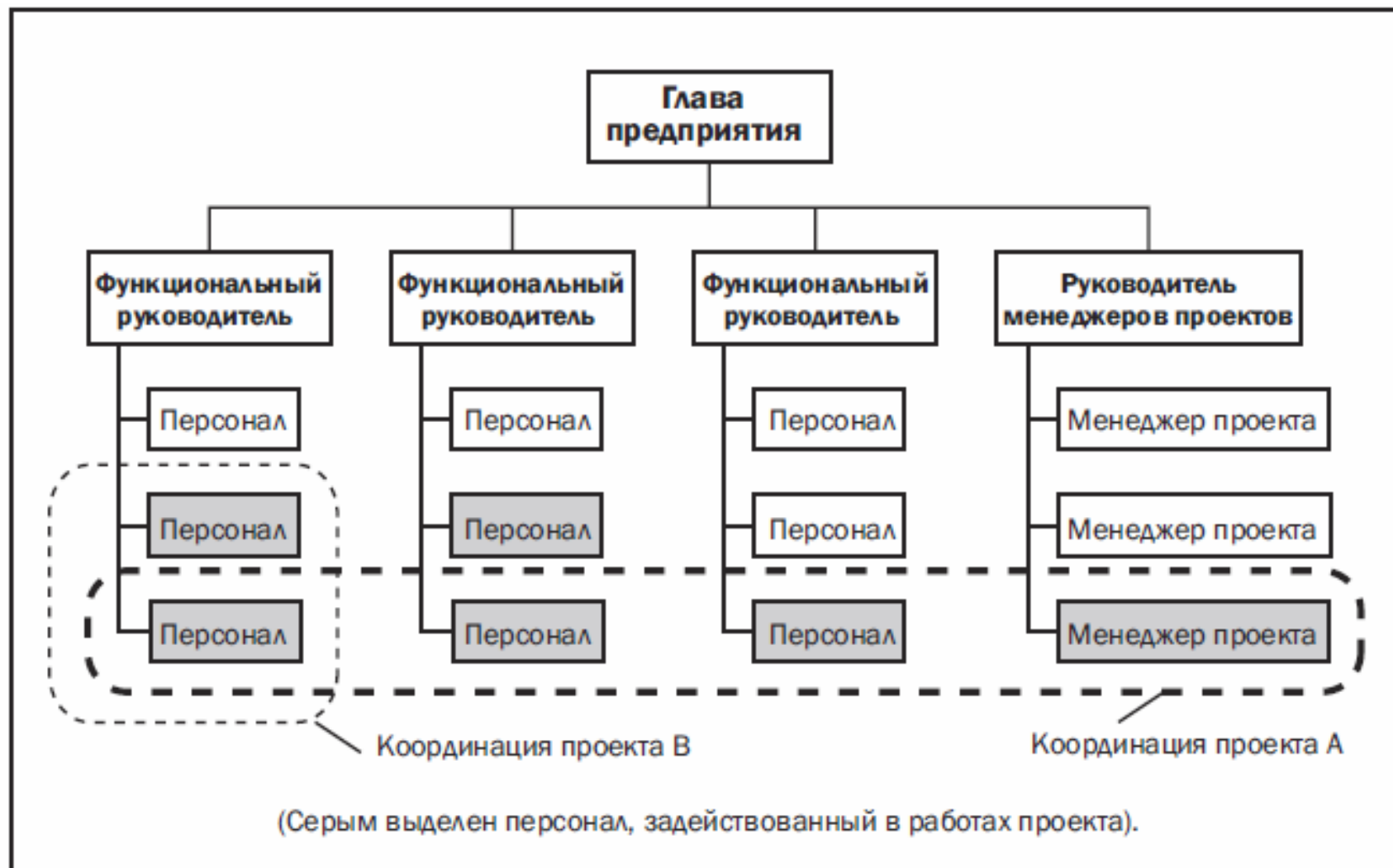
# ОРГАНИЗАЦИОННАЯ СТРУКТУРА. СИЛЬНАЯ МАТРИЧНАЯ ОРГАНИЗАЦИЯ



# ОРГАНИЗАЦИОННАЯ СТРУКТУРА. ПРОЕКТНАЯ ОРГАНИЗАЦИЯ



# ОРГАНИЗАЦИОННАЯ СТРУКТУРА. КОМБИНИРОВАННАЯ ОРГАНИЗАЦИЯ



# ВЛИЯНИЕ ОРГАНИЗАЦИИ НА ПРОЕКТЫ

Характеристики проекта / Организационная структура	Функциональная	Матрица			Проектная
		Слабая матрица	Сбалансированная матрица	Сильная матрица	
Власть менеджера проекта	Незначительная или отсутствует	Ограниченная	От слабой до умеренной	От умеренной до высокой	От высокой до практически абсолютной
Доступность ресурсов	Незначительная или отсутствует	Ограниченная	От слабой до умеренной	От умеренной до высокой	От высокой до практически абсолютной
Лицо, контролирующее бюджет проекта	Функциональный менеджер	Функциональный менеджер	Оба менеджера	Менеджер проекта	Менеджер проекта
Роль менеджера проекта	Частичная загрузка	Частичная загрузка	Полная загрузка	Полная загрузка	Полная загрузка
Административный персонал, управляющий проектом	Частичная загрузка	Частичная загрузка	Частичная загрузка	Полная загрузка	Полная загрузка

# **АКТИВЫ ПРОЦЕССОВ ОРГАНИЗАЦИИ**

Активы процессов организации включают все без исключения активы, относящиеся к процессам, во всех организациях, участвующих в проекте, которые могут быть использованы для оказания влияния на успех проекта.

Активы процессов организации могут быть разбиты на две категории:

- 1 Процессы и процедуры**
- 2. Корпоративная база знаний**



# ПРОЦЕССЫ И ПРОЦЕДУРЫ

Включают в себя, среди прочего:

- стандартные процессы организации, такие как стандарты, правила, стандартные жизненные циклы продуктов и проектов, а также правила и процедуры контроля качества
- типовые приказы, рабочие инструкции, критерии оценки предложений и критерии измерения исполнения;
- шаблоны (например, описание рисков)
- приказы и критерии для подгонки набора стандартных процессов организации с целью удовлетворения конкретных потребностей проекта;
- требования организации к обмену информацией
- приказы или требования к завершению проекта
- процедуры финансового контроля

# КОРПОРАТИВНАЯ БАЗА ЗНАНИЙ

Корпоративная база знаний организации для хранения и извлечения информации включает, среди прочего:

- базы данных измерений процессов;
- файлы проекта (например, содержание, стоимость, сроки);
- историческая информация и базы накопленных знаний
- базы данных по управлению открытыми вопросами и дефектами;
- базы знаний по управлению конфигурацией
- финансовые базы данных, содержащие такую информацию, как данные о человеко-часах, понесенных затратах, бюджете и любом перерасходе средств по проекту.

# МЕТОДОЛОГИИ УПРАВЛЕНИЯ ПРОЕКТАМИ ПО РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**Waterfall**

**RUP**

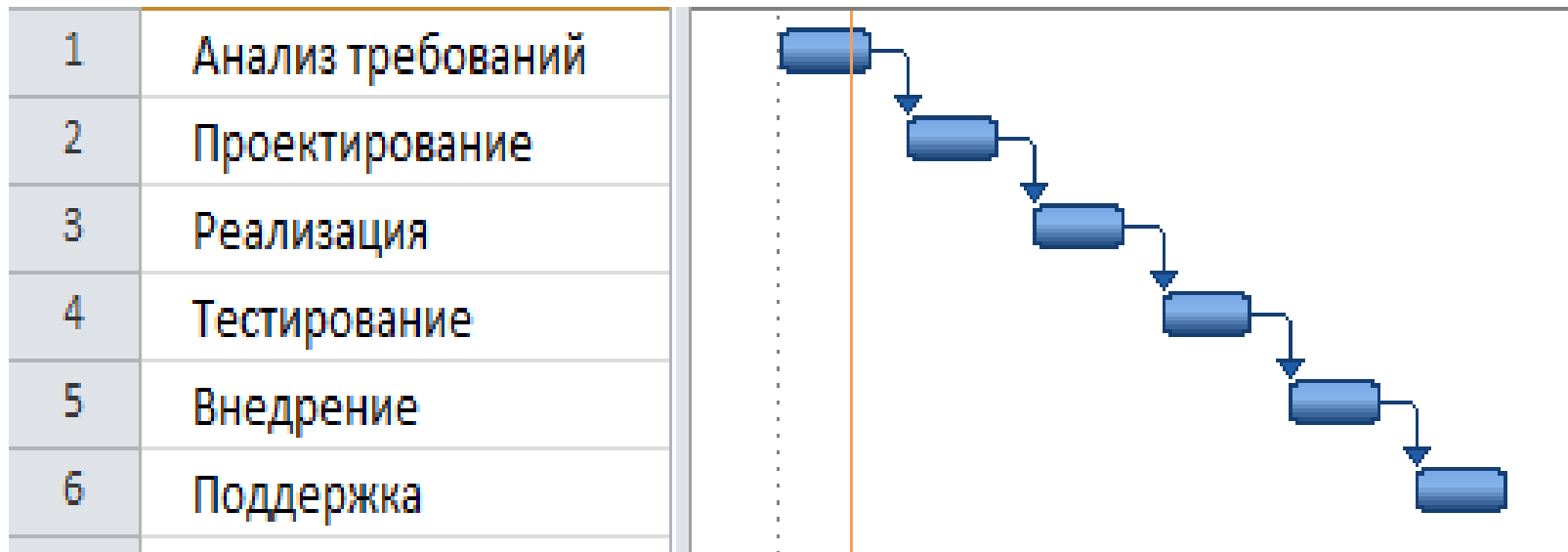
**MSF**

**Agile:**

- XP
- Lean
- Scrum
- TDD (Test-driven design)
- .....

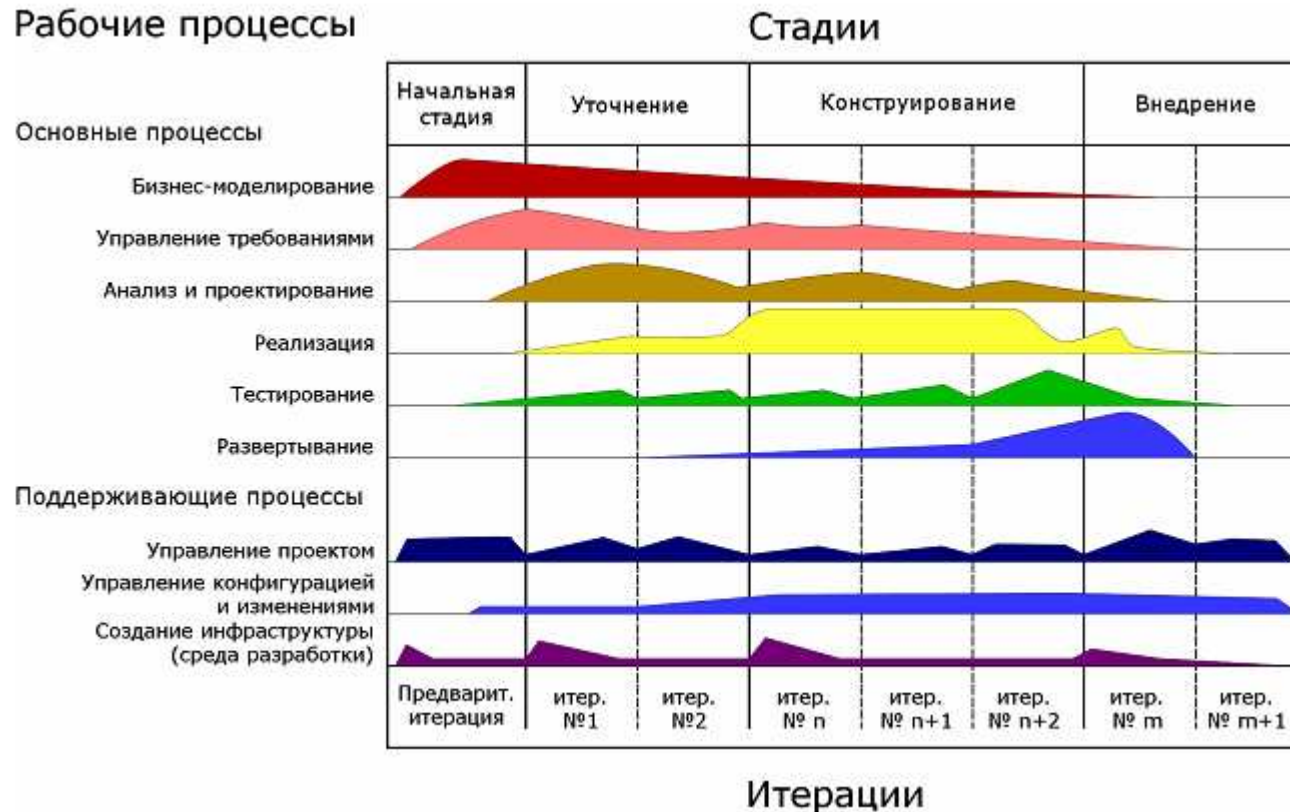
# WATERFALL (ВОДОПАД)

Модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, внедрения и поддержки.



# RUP (RATIONAL UNIFIED PROCESS)

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.



# НАЧАЛЬНАЯ ФАЗА (INCEPTION)

В фазе Inception:

Формируются видение и границы проекта.

Создается экономическое обоснование (**business case**).

Определяются основные требования, ограничения и ключевая функциональность продукта.

Создается базовая версия модели прецедентов.

Оцениваются риски.

При завершении фазы **Inception** оценивается достижение *вехи целей жизненного цикла (Lifecycle Objective Milestone)*, которое предполагает соглашение заинтересованных сторон о продолжении проекта.

# УЛУЧШЕНИЕ (ELABORATION)

В фазе **Elaboration** производится анализ предметной области и построение исполняемой архитектуры. Это включает в себя:

**Документирование требований (включая детальное описание для большинства прецедентов).**

**Спроектированную, реализованную и оттестированную исполняемую архитектуру.**

**Обновленное экономическое обоснование и более точные оценки сроков и стоимости.**

**Сниженные основные риски.**

Успешное выполнение фазы **Elaboration** означает достижение *вехи архитектуры жизненного цикла (Lifecycle Architecture Milestone)*.

# РАЗРАБОТКА (CONSTRUCTION)

Во время этой фазы происходит реализация **большой части функциональности продукта.**

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



# ВНЕДРЕНИЕ (TRANSITION)

Во время фазы **Transition** создается **финальная версия продукта** и передается от разработчика к заказчику. Это включает в себя:

- программу бета-тестирования,
- обучение пользователей
- определение качества продукта.

В случае, если качество не соответствует ожиданиям пользователей или критериям, установленным в фазе **Inception**, фаза **Transition** повторяется снова.

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

# MSF (MICROSOFT SOLUTION FRAMEWORK)



# AGILE

**Agile** — семейство процессов разработки, а не единственный подход в разработке программного обеспечения, и определяется **Agile Manifesto**. Agile не включает **практик**, а определяет **ценности и принципы**, которыми руководствуются успешные команды.

Agile Manifesto разработан и принят 11-13 февраля 2001 года. Agile Manifesto содержит 4 основные идеи и 12 принципов. Примечательно, что Agile Manifesto не содержит практических советов

# ОСНОВНЫЕ ИДЕИ AGILE

- Личности и их взаимодействия важнее, чем процессы и инструменты;
- Работающее программное обеспечение важнее, чем полная документация;
- Сотрудничество с заказчиком важнее, чем контрактные обязательства;
- Реакция на изменения важнее, чем следование плану

# ОСНОВНЫЕ ПРИНЦИПЫ AGILE

- удовлетворение клиента за счёт ранней и бесперебойной поставки ценного ПО;
- приветствие изменения требований, даже в конце разработки ( это может повысить конкурентоспособность полученного продукта);
- частая поставка рабочего ПО (каждый месяц или неделю или ещё чаще);
- тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта;
- проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
- рекомендуемый метод передачи информации — личный разговор (лицом к лицу);
- работающее ПО — лучший измеритель прогресса;
- спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределенный срок;
- постоянное внимание на улучшение технического мастерства и удобный дизайн;
- простота — искусство НЕ делать лишней работы;
- лучшие технические требования, дизайн и архитектура получаются у самоорганизованной команды;
- постоянная адаптация к изменяющимся обстоятельствам

# XP (EXTREME PROGRAMMING)

12 основных приёмов экстремального программирования:

## 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, Refactor)
- Частые небольшие релизы (Small Releases)

## Понимание, разделяемое всеми

- Простота (Simple design)
- Метафора системы (System metaphor)
- Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)
- Стандарт кодирования (Coding standard or Coding conventions)

## Социальная защищённость программиста (Programmer welfare):

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

# SCRUM

Scrum — это набор принципов, на которых строится процесс разработки, позволяющий в жёстко фиксированные небольшие промежутки времени (*спринты* от 2 до 4 недель) предоставлять конечному пользователю работающее ПО с новыми возможностями, для которых определён наибольший приоритет. Возможности ПО к реализации в очередном спринте определяются в начале спринта на этапе планирования и не могут изменяться на всём его протяжении. При этом строго-фиксированная небольшая длительность спринта придаёт процессу разработки предсказуемость и гибкость

