Конструирование программного обеспечения (Технология программирования)

СПИНТех
Доцент Федоров Алексей Роальдович

Конструирование –процесс поиска варианта структуры, форм, размеров, взаимного расположения и параметров частей и элементов конструкции, материалов и взаимосвязи совокупности отдельных элементов, предназначенных для выполнения заданных функций в соответствии с требованиями технического задания, с учетом достижений науки и техники, патентных оценок и перспектив.

Результат – техническая документация

Технология программирования - совокупность методов и средств, используемых в процессе разработки ПО. Включает:

- указание последовательности выполнения технологических операций;
- перечисление условий, при которых выполняется та или иная операция;
- описание самих операций, где для каждой операции определены исходные данные, результаты, а также инструкции, нормативы, стандарты, критерии и методы оценки и т. п.

Вес — 27 тонн.
Потребляемая мощность — 174 кВт.
Объём памяти — около 1 кбайт.
Тактовая частота — 100 кГц.

iPhone 11 - шестиядерный процессор 1,4 ГГц и 128 Гб!


"Apple II"
disk II
2
disk II
1
apple II

iMac27

суперкомпьютер Blue Gene/L
Энергопотребление 1,6 мегаватт
280 терафлопс

Производительность суперкомпьютеров

Название	год	флопсы
флопс	1941	10^0
килофлопс	1949	10^3
мегафлопс	1964	10^6
гигафлопс	1987	10^9
терафлопс	1997	10^{12}
петафлопс	2008	10^{15}
эксафлопс	2018 или позже	10^{18}
зеттафлопс	не ранее 2030	10^{21}
иоттафлопс	не скоро	10^{24}
ксерафлопс	не скоро	10^{27}


天河
天河

90 серверов Power7 750

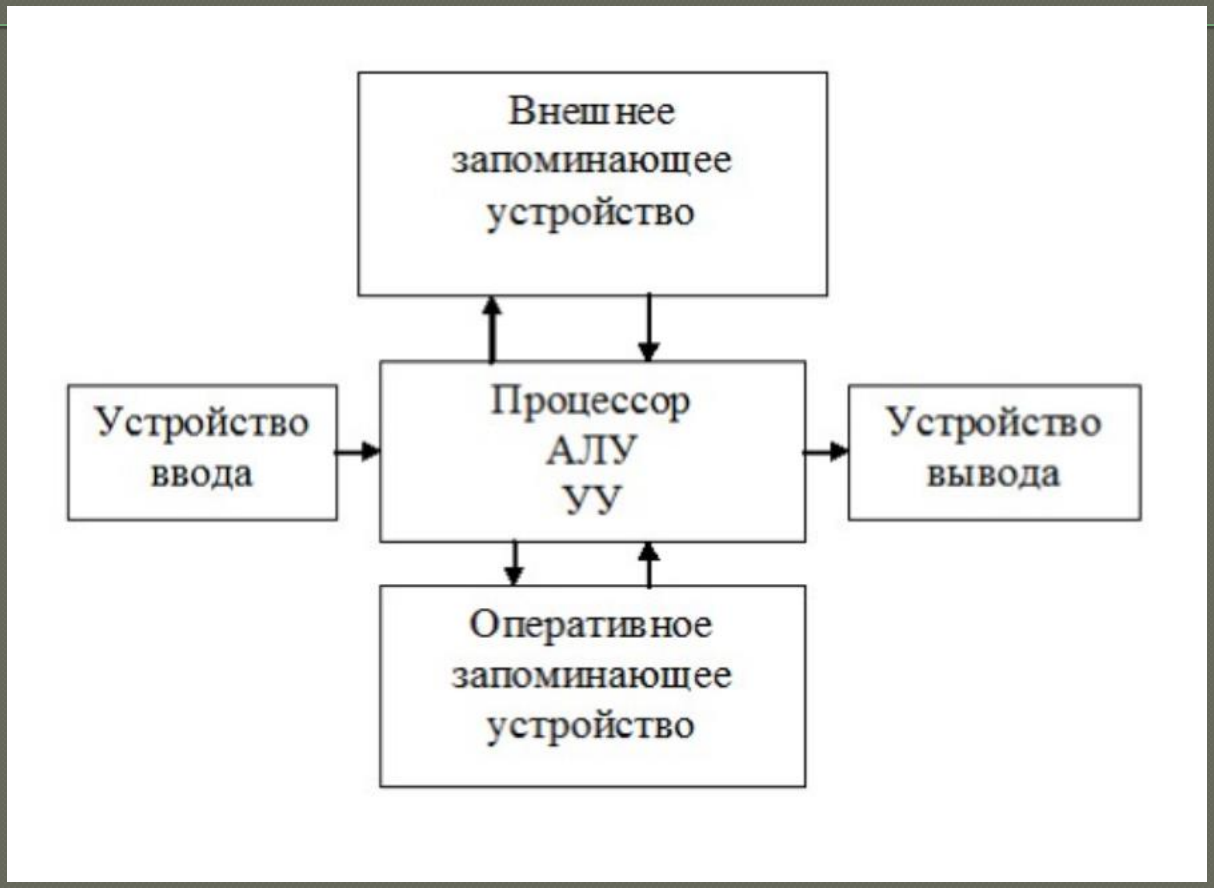
Оперативная память более 15 терабайт.

А.А. Марков: алгоритм - *точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату.*

*М*атематически строго алгоритм был определен на основе понятия *рекурсивной функции* (А. Чёрч) и на основе описания *алгоритмического процесса (А*. Тьюринг)

Рекурсивная функция – это функция, для которой существует алгоритм вычисления ее значений для произвольного значения аргумента на основе известных предыдущих значений.

Алгоритмический процесс определяется как дискретный процесс, осуществимый на абстрактной машине, называемой «машиной Тьюринга»..

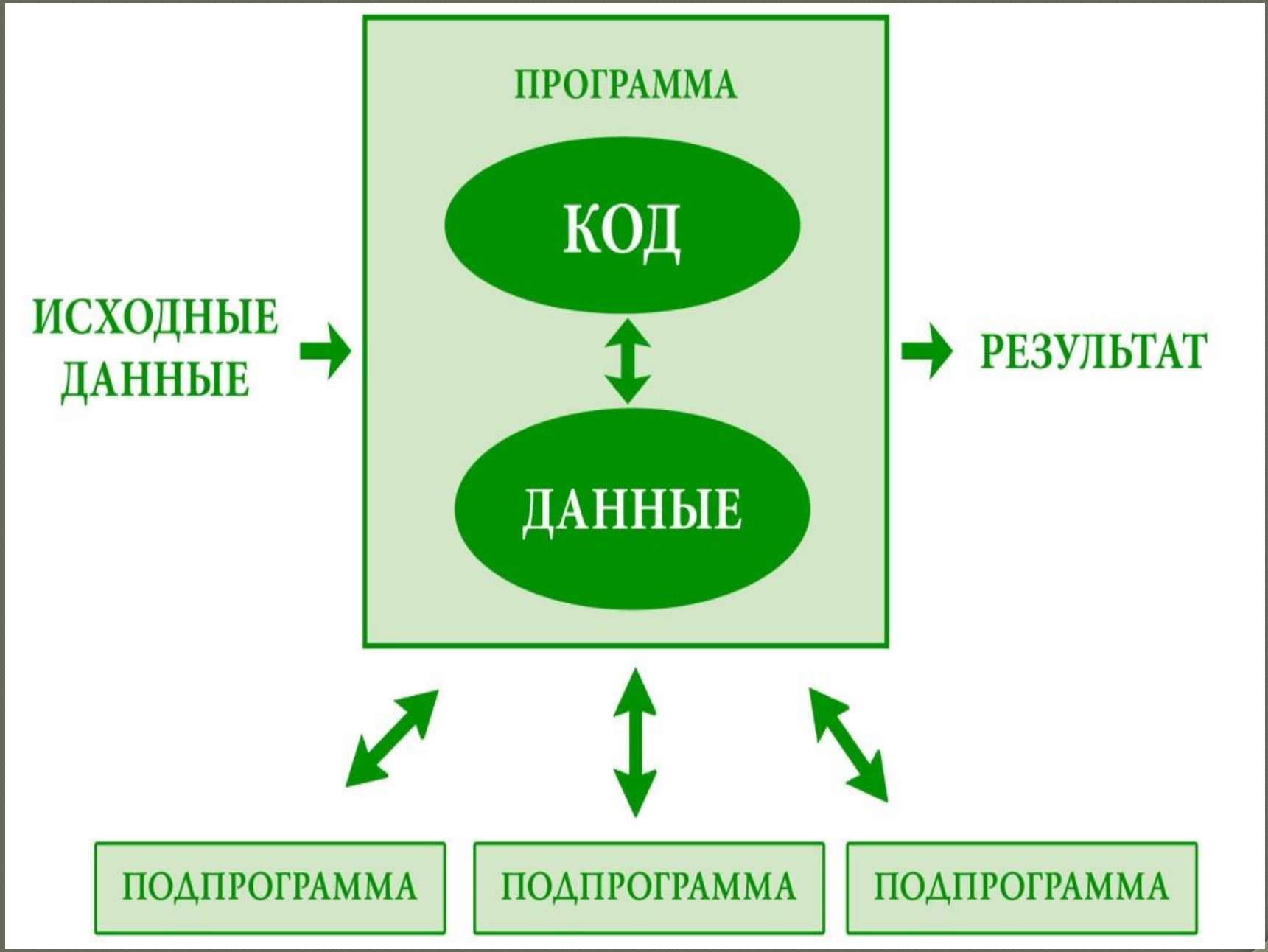

Внешнее запоминающее устройство
Устройство ввода
Процессор АЛУ УУ
Устройство вывода
Оперативное запоминающее устройство

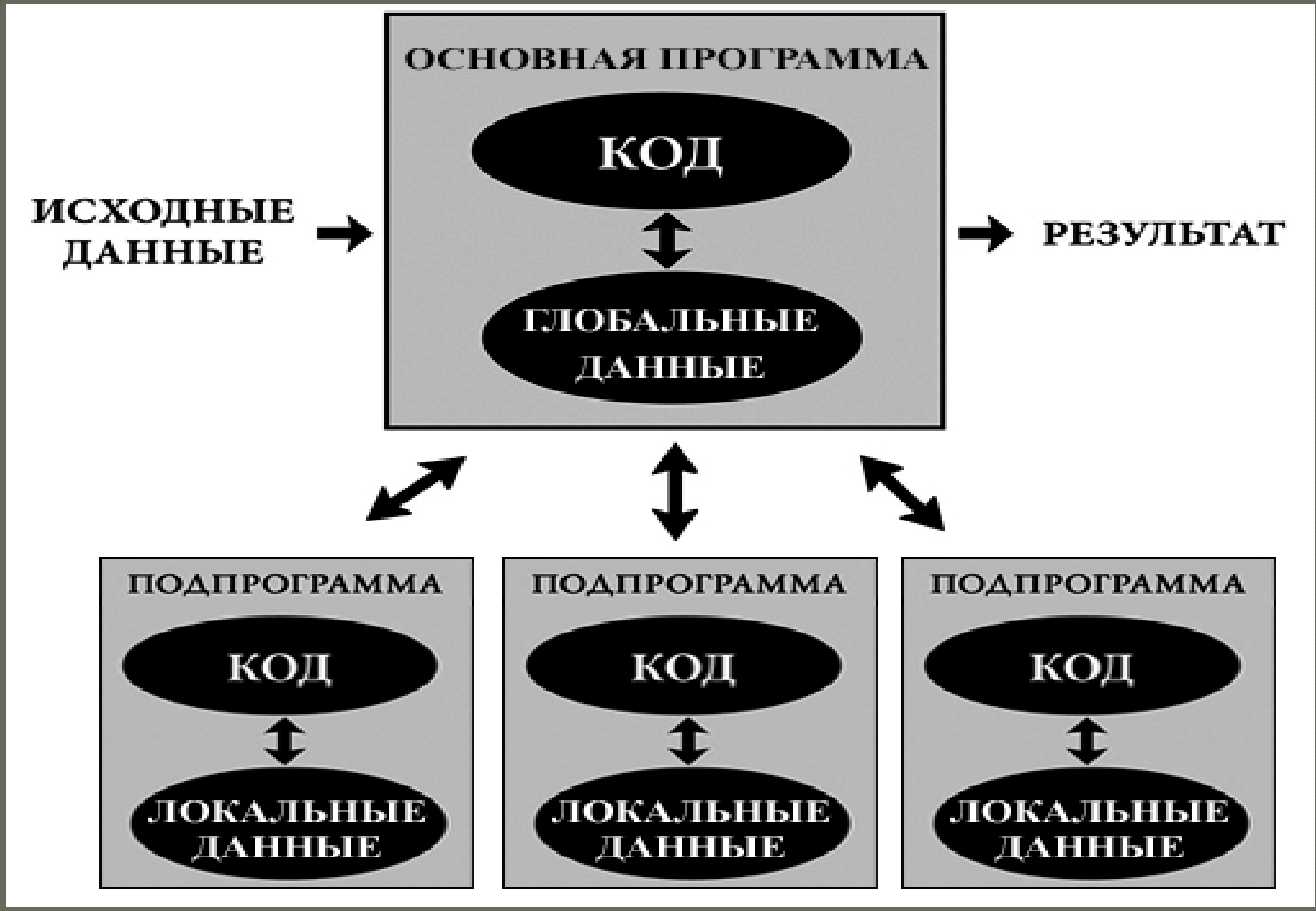
Под *программным обеспечением* (ПО) будем понимать множество развивающихся во времени *логических предписаний*, с помощью которых некоторый коллектив людей управляет и использует *многопроцессорную* и *распределенную* систему *вычислительных устройств*.

Свойства.

- *Сложность*
- *Нематериальность*
- *Согласованность*
- *Изменяемость*

- ЭВМ использовались исключительно для математических расчетов . На этом этапе ценились в основном простота алгоритмов и компактность кода.
- Каждая программа уникальна и предназначена исключительно для использования на конкретной ЭВМ.
- Организация данных простая и сложная одновременно: есть область памяти машины для данных, есть для кода. Никто даже и не думал сравнивать программирование с архитектурой. Скорее с кухней. Программа – как мясорубка: на входе исходные данные, на выходе готовый фарш – результат.


ПРОГРАММА
КОД
ДАННЫЕ
ИСХОДНЫЕ ДАННЫЕ
РЕЗУЛЬТАТ
ПОДПРОГРАММА
ПОДПРОГРАММА
ПОДПРОГРАММА


ОСНОВНАЯ ПРОГРАММА
КОД
ГЛОБАЛЬНЫЕ ДАННЫЕ
ИСХОДНЫЕ ДАННЫЕ
РЕЗУЛЬТАТ
ПОДПРОГРАММА
КОД
ЛОКАЛЬНЫЕ ДАННЫЕ
ПОДПРОГРАММА
КОД
ЛОКАЛЬНЫЕ ДАННЫЕ
ПОДПРОГРАММА
КОД
ЛОКАЛЬНЫЕ ДАННЫЕ

Теорема о структурном программировании

Любая программа, заданная в виде блок-схемы, может быть представлена с помощью трех управляющих структур:

- **последовательность —** *обозначается:* f THEN g,
- **ветвление** — *обозначается:* IF p THEN f ELSE g,
- **цикл** — *обозначается:* WHILE p DO f,

где f, g — блок-схемы с одним входом и одним выходом, p — условие,
THEN, IF, ELSE, WHILE, DO — ключевые слова.

Основные принципы выполнения разработки:

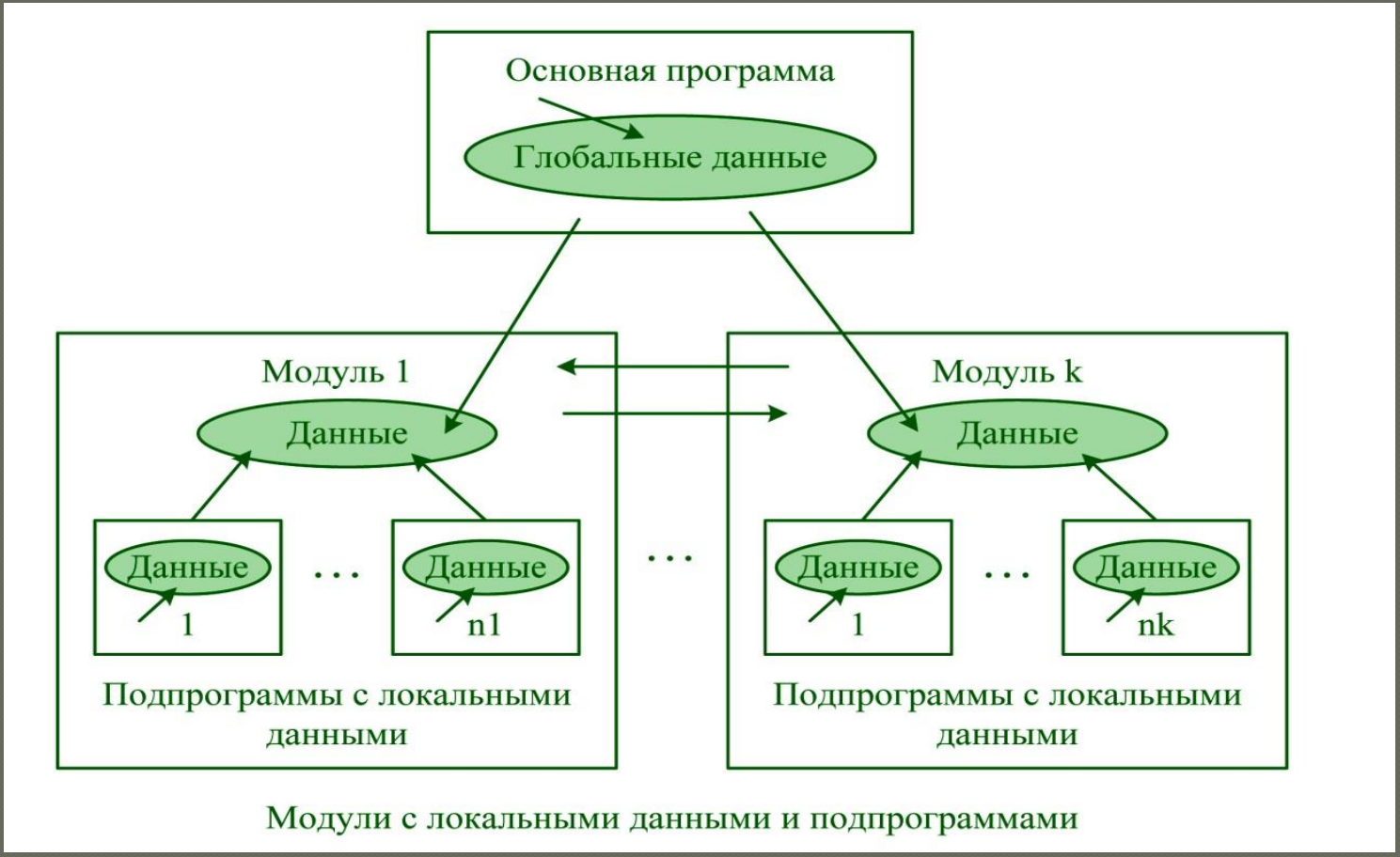
- Отказ от использования оператора безусловного перехода goto.
- Только три управляющих конструкции: последовательность, ветвление, цикл.
- управляющие конструкции могут быть вложены друг в друга произвольным образом.
- Повторяющиеся фрагменты программы следует оформлять в виде подпрограмм.
- Все конструкции должны иметь один вход и один выход.
- Наличие четкой структуры алгоритма и соблюдение стиля культурного программирования;
- Разработка программы пошагово, методом «сверху вниз» с поэтапной детализацией;
- сквозной структурный контроль

технология модульного программирования

основные концепции:

Принцип утаивания информации Парнаса.

Аксиома модульности Коуэна



А.А. Марков: алгоритм - *точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату.*

*М*атематически строго алгоритм был определен на основе понятия *рекурсивной функции* (А. Чёрч) и на основе описания *алгоритмического процесса (А*. Тьюринг)

Рекурсивная функция – это функция, для которой существует алгоритм вычисления ее значений для произвольного значения аргумента на основе известных предыдущих значений.

Алгоритмический процесс определяется как дискретный процесс, осуществимый на абстрактной машине, называемой «машиной Тьюринга»..

Наиболее распространенные парадигмы программирования:

- Императивное программирование;
- Структурное программирование;
- Декларативное программирование;
- Функциональное программирование;
- Логическое программирование;
- Объектно-ориентированное программирование;
- Компонентно-ориентированное программирование;
- Агентно-ориентированное программирование.

Императивное программирование:

- использование именованных переменных;
- использование оператора присваивания;
- использование составных выражений.

Структурное программирование

- Basic
- Pascal
- PL/M
- Алгол
- Модула
- Оберон
- Фокал
- Фортран

Декларативное программирование:

- Логическое программирование;

Mercury
Prolog
Curry

- Функциональное программирование;

функциональное программирование

это стиль программирования, использующий только композиции функций.
Другими словами, это программирование в выражениях, а не в императивных командах

Языки функционального программирования

- **Lisp**
- **ISWIM** (If you See What I Mean)
- **Scheme**
- **ML** (Meta Language)
- **Standard ML**
- **Caml Light** и **Objective Caml**
- **Miranda**
- **Haskell**
- **Gofer**
- **Clean**

Объектно-ориентированное программирование

технология создания сложного программного обеспечения,
основанная на представлении программы в виде совокупности ***объектов***,
каждый из которых является экземпляром определенного типа (***класса***),
классы образуют иерархию с ***наследованием*** свойств.

ВАЖНО!:
ООП использует в качестве основных логических конструктивных элементов ***объекты***, а не ***алгоритмы***;
каждый ***объект*** является экземпляром определенного ***класса***;
классы образуют ***иерархии***.

Кроме перечисленных понятий ООП оперирует с терминами
АБСТРАКЦИЯ ДАННЫХ , ИНКАПСУЛЯЦИЯ, ПОЛИМОРФИЗМ.

Объектно-ориентированное программирование

набор основных принципов :

- Всё является объектом.
- Вычисления осуществляются путём взаимодействия (обмена данными) между объектами, при котором один объект требует, чтобы другой объект выполнил некоторое действие. Объекты взаимодействуют, посылая и получая сообщения. Сообщение — это запрос на выполнение действия, дополненный набором аргументов.
- Каждый объект имеет независимую память, которая состоит из других объектов.
- Каждый объект является представителем класса, который выражает общие свойства объектов.
- В классе задаётся поведение (функциональность) объекта. Тем самым все объекты, которые являются экземплярами одного класса, могут выполнять одни и те же действия.
- Классы организованы в единую древовидную структуру с общим корнем, называемую иерархией наследования. Память и поведение, связанное с экземплярами определённого класса, автоматически доступны любому классу, расположенному ниже в иерархическом дереве.

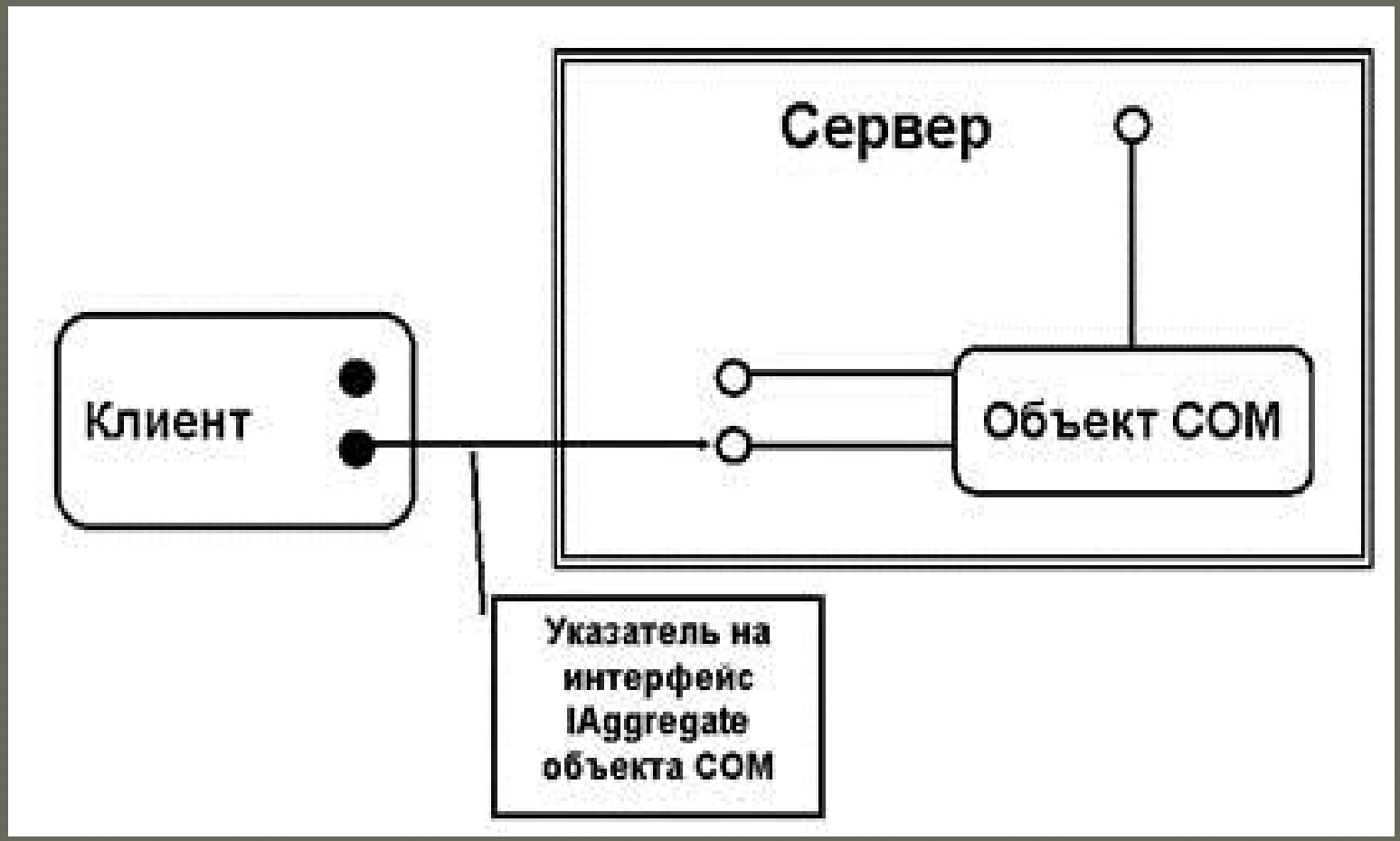
Объектно-ориентированное программирование

Главный недостаток - низкая производительность.

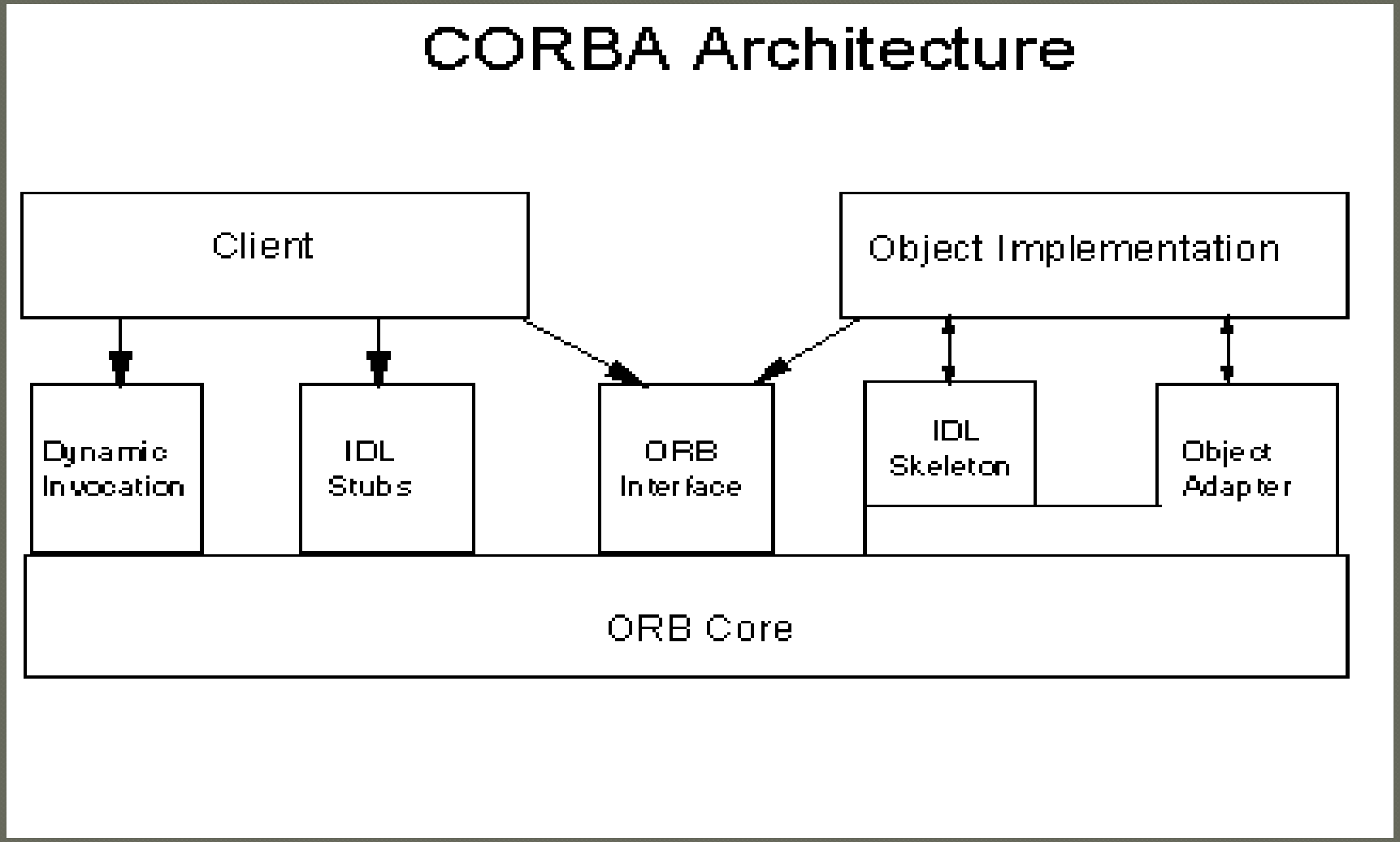
Причины:

- *Динамическое связывание методов*.
- *Значительная глубина абстракции.*
- *Наследование «размывает» код*.
- *Инкапсуляция снижает скорость доступа к данным.*
- *Динамическое создание и уничтожение объектов.*

компонентный подход
COM-технология



компонентный подход



API —
программный интерфейс приложения,
интерфейс прикладного программирования —
набор готовых классов, процедур, функций,
структур и констант,
предоставляемых приложением (библиотекой,
сервисом) или операционной системой для
использования во внешних программных
продуктах

DirectX, OpenGL, GDI+, SDL, GTK, Qt.

Фреймворк — заготовки, шаблоны для программной платформы, определяющие структуру программной системы функциональной гибкости.

- HTML/CSS-фреймворки: *Bootstrap, Foundation, Semantic UI, Uikit, Pure by Yahoo!*
- PHP-фреймворки: *Yii, Laravel, Symfony, CodeIgniter, Phalcon PHP.*
- Python-фреймворки: *Django, Flask, TurboGears, Tornado, Web2spy.*
- Ruby-фреймворки: *Ruby on Rails (RoR), Sinatra, «Mongrel (HTTP сервер) + Erb», Hanami (Lotus).*
- Java-фреймворки: *Spring, Vaadin, Google Web Toolkit (GWT), JavaServer Faces (JSF), Play.*
- JavaScript-фреймворки – *AngularJS*, Ember.js.

CASE-технологии (Computer-Aided Software/System Engineering)

CASE-технология - методология проектирования ПО, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ПО и разрабатывать приложения в соответствии с информационными потребностями пользователей.

CASE-инструменты:

- ***средства анализа***;
- ***средства проектирования баз данных;***
- ***средства разработки приложений;***
- ***средства реинжиниринга процессов;***
- ***средства планирования и управления проектом;***
- ***средства тестирования;***
- ***средства документирования.***

Три типа моделей при *структурном* подходе:

- функциональные
- информационные
- структурные.

Системы:

- SADT (Structured Analysis and Design Technique) - модели и соответствующие функциональные диаграммы;
- DFD (Data Flow Diagrams) - диаграммы потоков данных;
- ERD (Entity-Relationship Diagrams) - диаграммы "сущность-связь".

Основной инструмент *объектно-ориентированного* подхода - язык **UML**

CASE-инструменты:

- средства анализа
- средства проектирования баз данных;
- средства разработки приложений;
- средства реинжиниринга процессов;
- средства планирования и управления проектом;
- средства тестирования;
- средства документирования.

Система – *множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность и единство элементов*.

- *Элемент*
- *Компонент, подсистема*
- *Связь, отношение*
- *Структура*
- ***Цель***

Всякая система характеризуется :

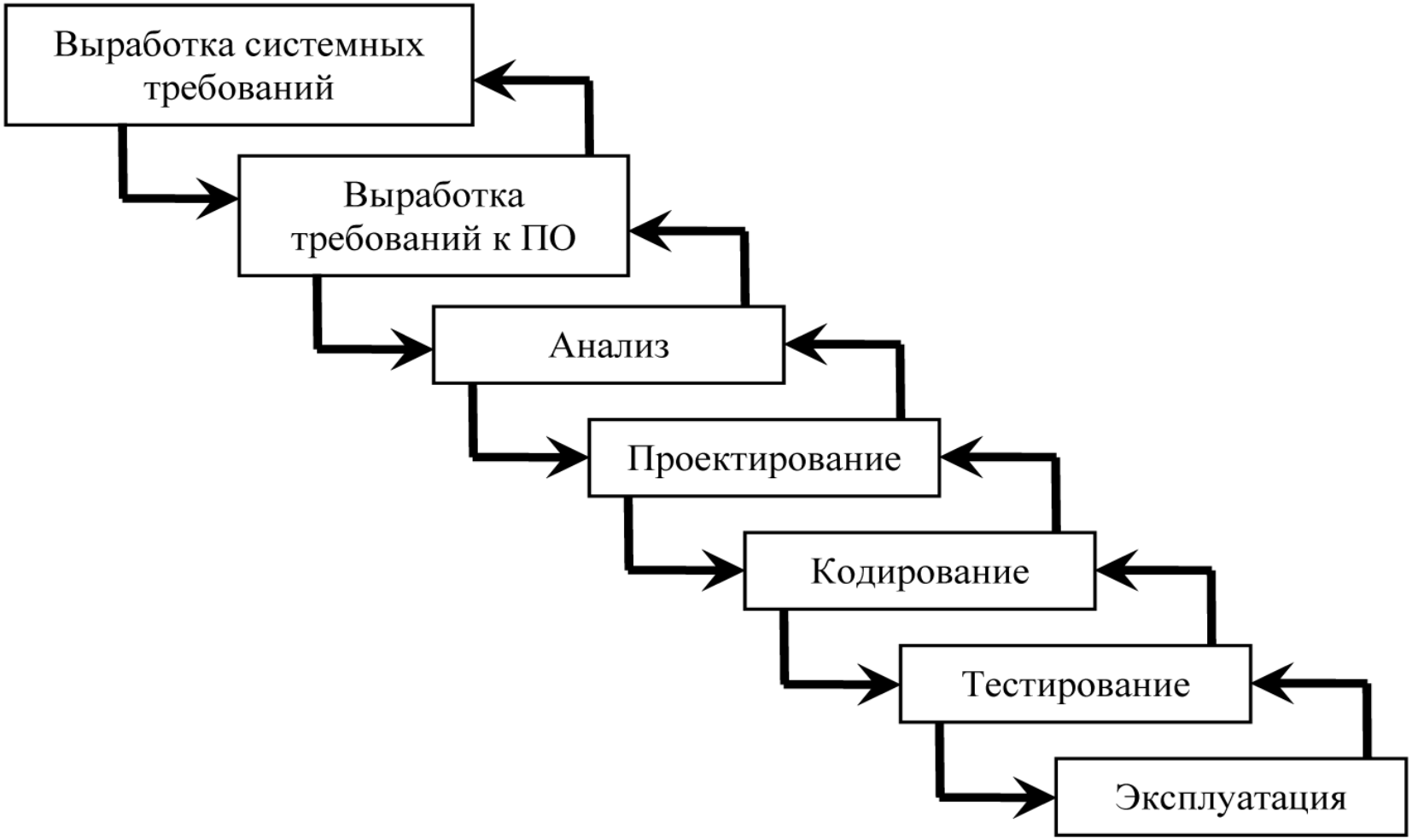
- *Состояние*
- *Поведение*
- *Развитие, эволюция*
- *Жизненный цикл*
- *Внешняя среда*

ЖИЗНЕННЫЙ ЦИКЛ И ЭТАПЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

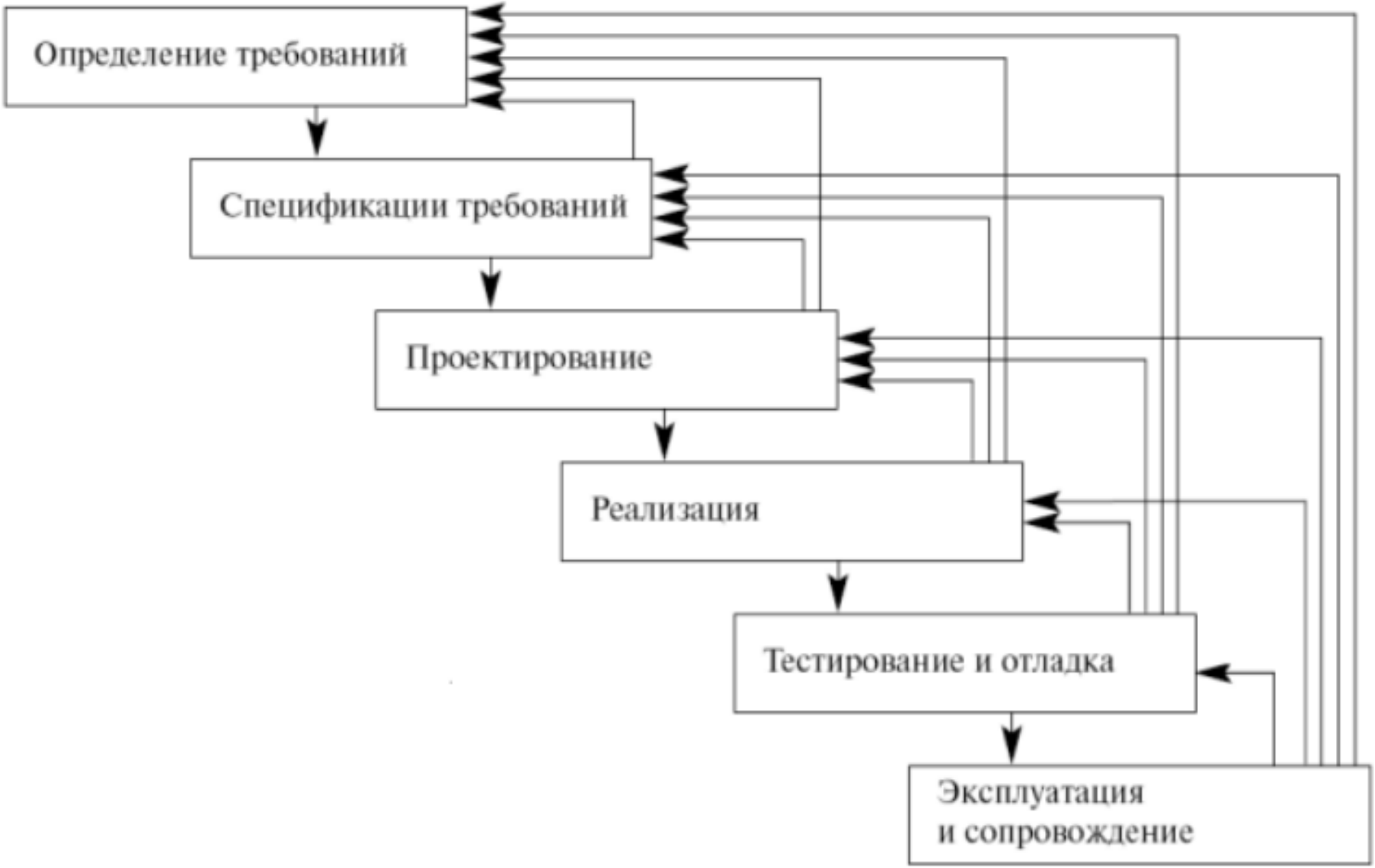
Жизненным циклом программного обеспечения называют период от момента появления идеи создания некоторого программного обеспечения до момента завершения его поддержки фирмой-разработчиком или фирмой, выполнявшей сопровождение. В ходе жизненного цикла ПО проходит через анализ предметной области, сбор требований, проектирование, кодирование, тестирование, сопровождение и др.

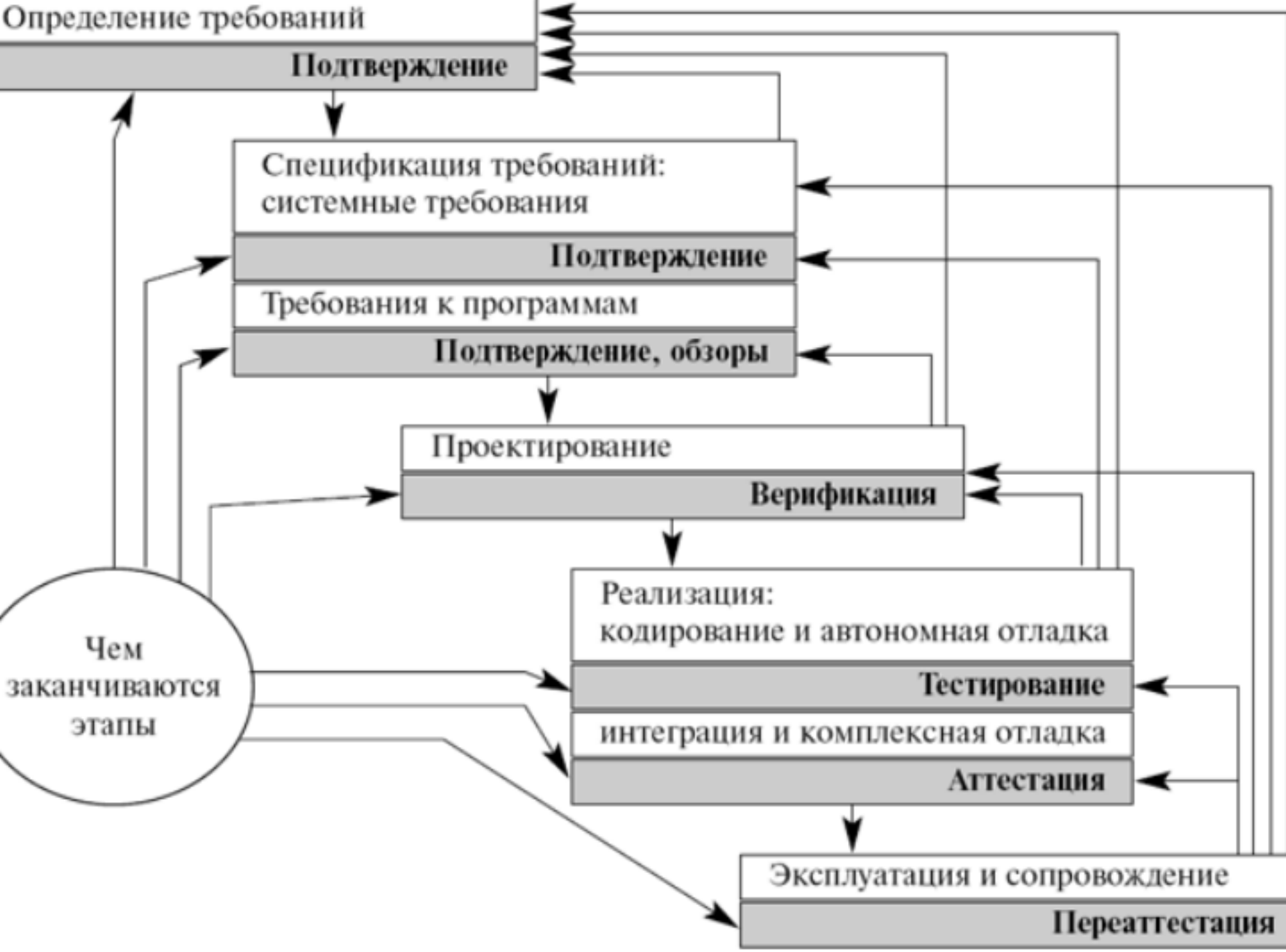
КАСКАДНАЯ МОДЕЛЬ РАЗРАБОТКИ ПО

МОДЕЛЬ С ПРОМЕЖУТОЧНЫМ КОНТРОЛЕМ

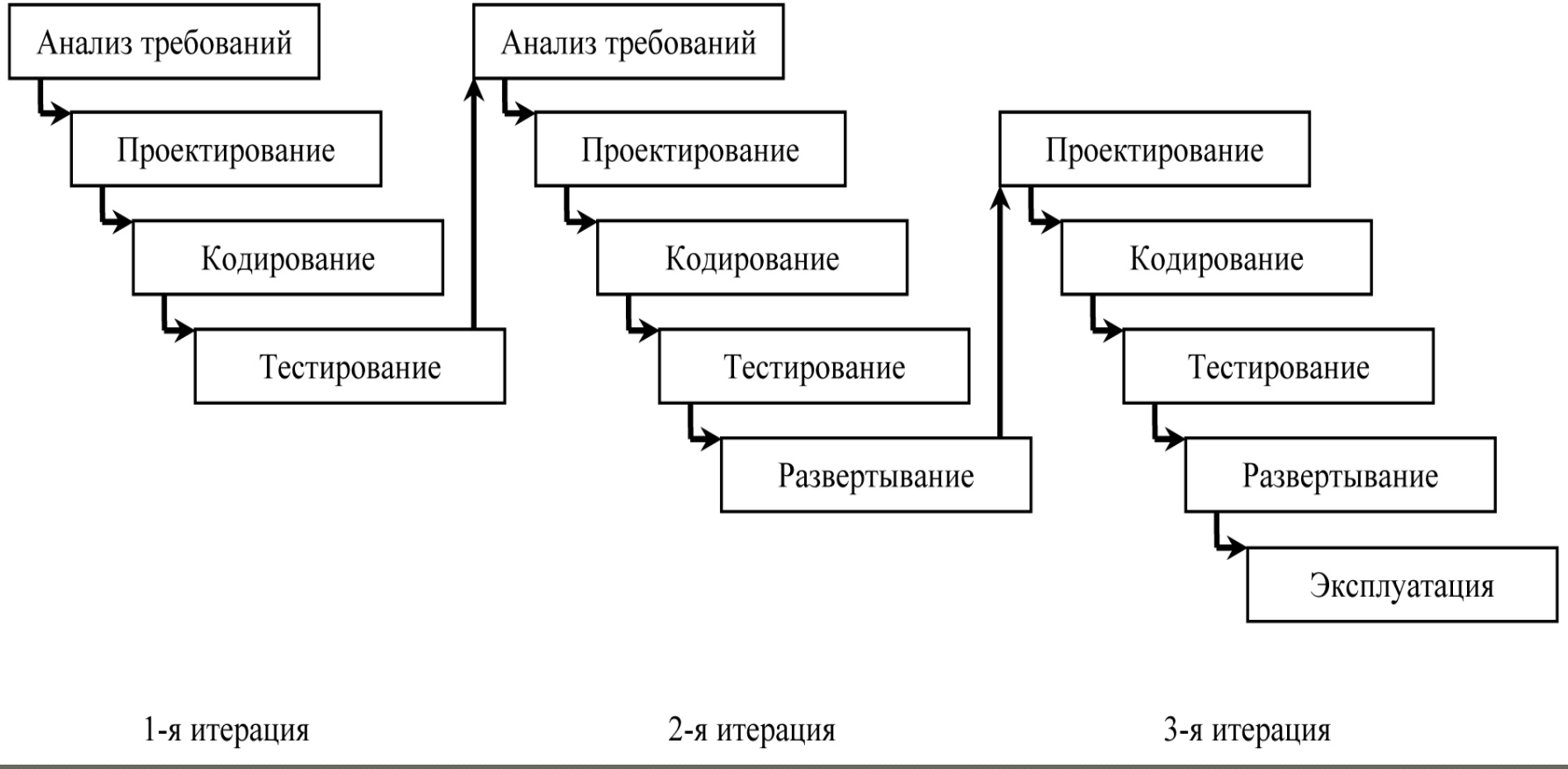


МОДЕЛЬ С ПРОМЕЖУТОЧНЫМ КОНТРОЛЕМ

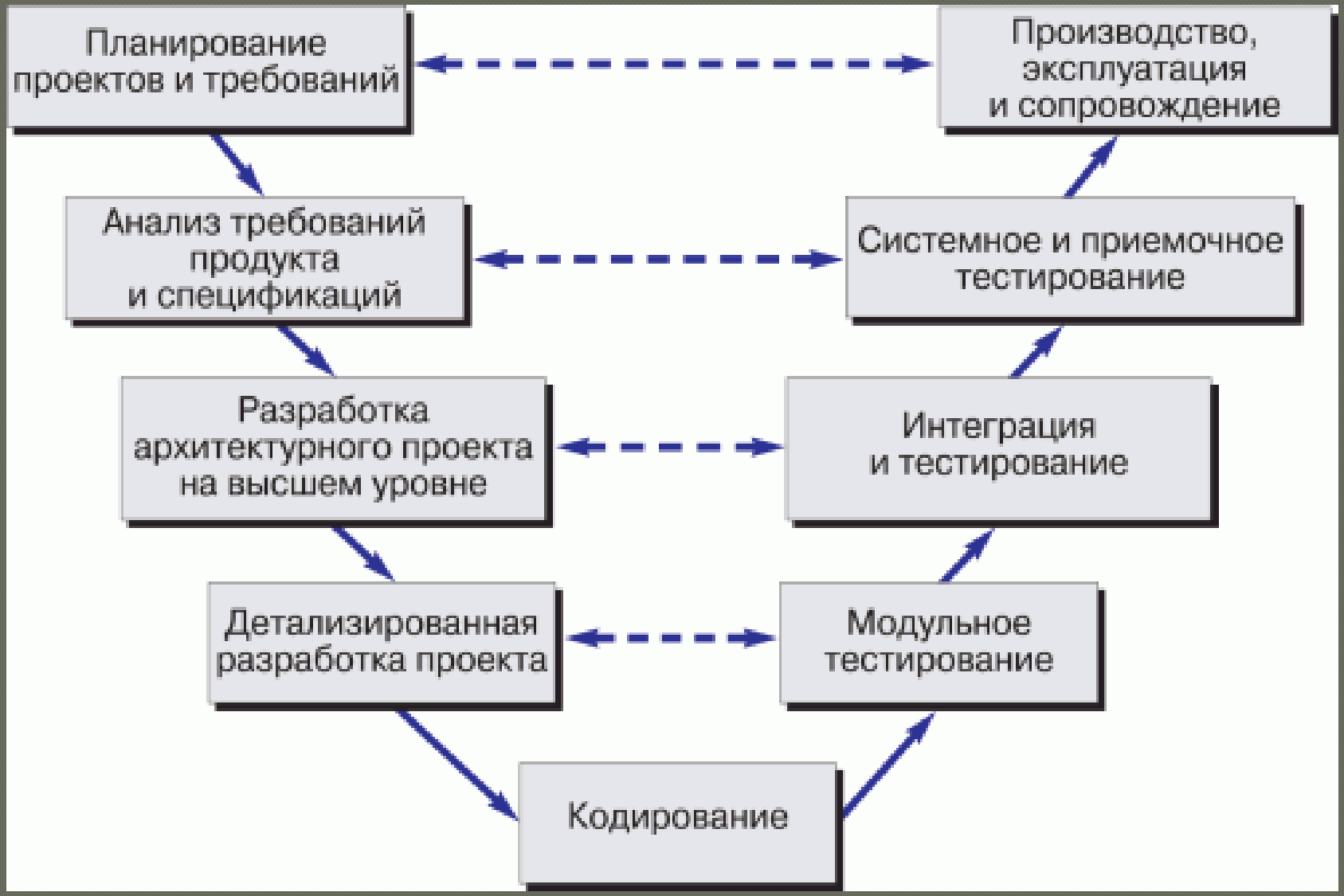



Определение требований
Подтверждение
Спецификация требований:
системные требования
Подтверждение
Требования к программам
Подтверждение, обзоры
Проектирование
Верификация
Реализация:
кодирование и автономная отладка
Тестирование
интеграция и комплексная отладка
Аттестация
Эксплуатация и сопровождение
Переаттестация
Чем
заканчиваются
этапы

Итеративная или инкрементная модель ЖЦ



V-образная модель жизненного цикла ПО



Спиральная модель ЖЦ

Модель фазы—функции

Фазовое измерение модели фазы—функции

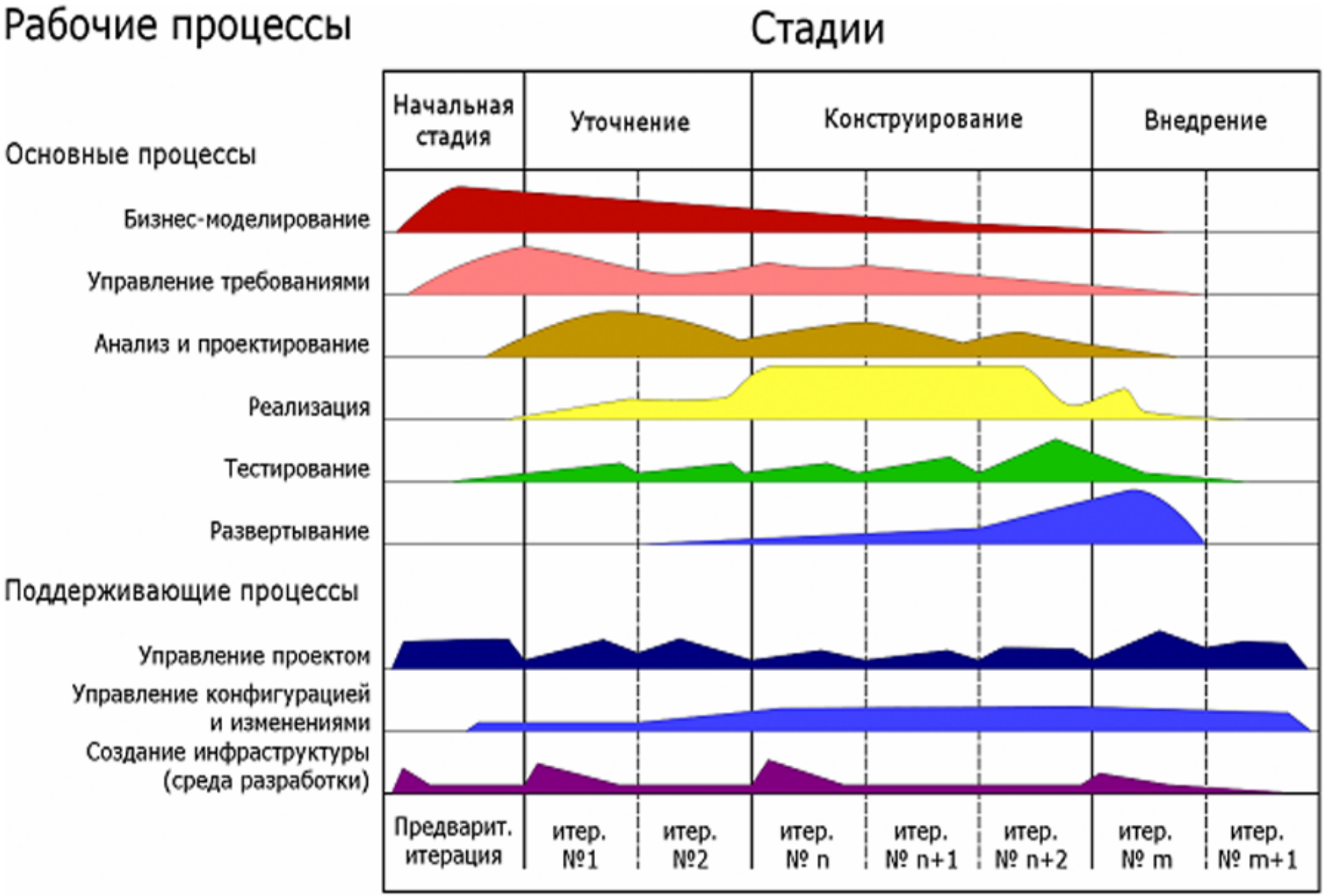
Модель фазы—функции

Матрица фазы-функции модели Гантера

Модель фазы—функции

Развитие модели фазы-функции

RUP (Rational Unified Process).



Манифест Agile

4 базовых идеи
12 принципов эффективного управления.

Идеи Agile:

- люди и их взаимодействие важнее, чем процессы и инструменты;
- работающее ПО важнее, чем исчерпывающая документация;
- сотрудничество с заказчиком важнее согласования условий контракта;
- готовность изменениям важнее, чем первоначальный план.

12 ПРИНЦИПОВ AGILE:

- удовлетворение клиента за счёт непрерывной поставки ПО;
- изменять требования к конечному продукту даже в конце разработки;
- частая поставка рабочего программного обеспечения;
- ежедневное общение заказчика с разработчиками;
- проектом должны заниматься мотивированные личности;
- метод передачи информации — личный разговор;
- работающее ПО — лучший измеритель прогресса;
- спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный высокий темп разработки;
- уделять внимание дизайну и техническим, постоянного совершенствовать продукт;
- стараться сделать рабочий процесс максимально простым, а ПО – простым и понятным;
- позволять членам команды самостоятельно принимать решения;
- постоянно адаптироваться к меняющейся среде.

Самыми известные варианты проект-менеджмента по Agile-методологии:

Scrum (Скрам) и **Kanban** (Канбан).

Методологии

- Agile Modeling
- Agile Unified Process (AUP)
- Agile Data Method
- DSDM
- Essential Unified Process (EssUP)
- Экстремальное программирование (XP)
- Feature driven development (FDD)
- Getting Real
- OpenUP

Модель хаоса

- В целом проект должен быть определен, реализован и интегрирован.
- Системы должны быть определены, реализованы и интегрированы.
- Модули должны быть определены, реализованы и интегрированы.
- Функции должны быть определены, реализованы и интегрированы.
- Строки кода должны быть определены, реализованы и интегрированы.

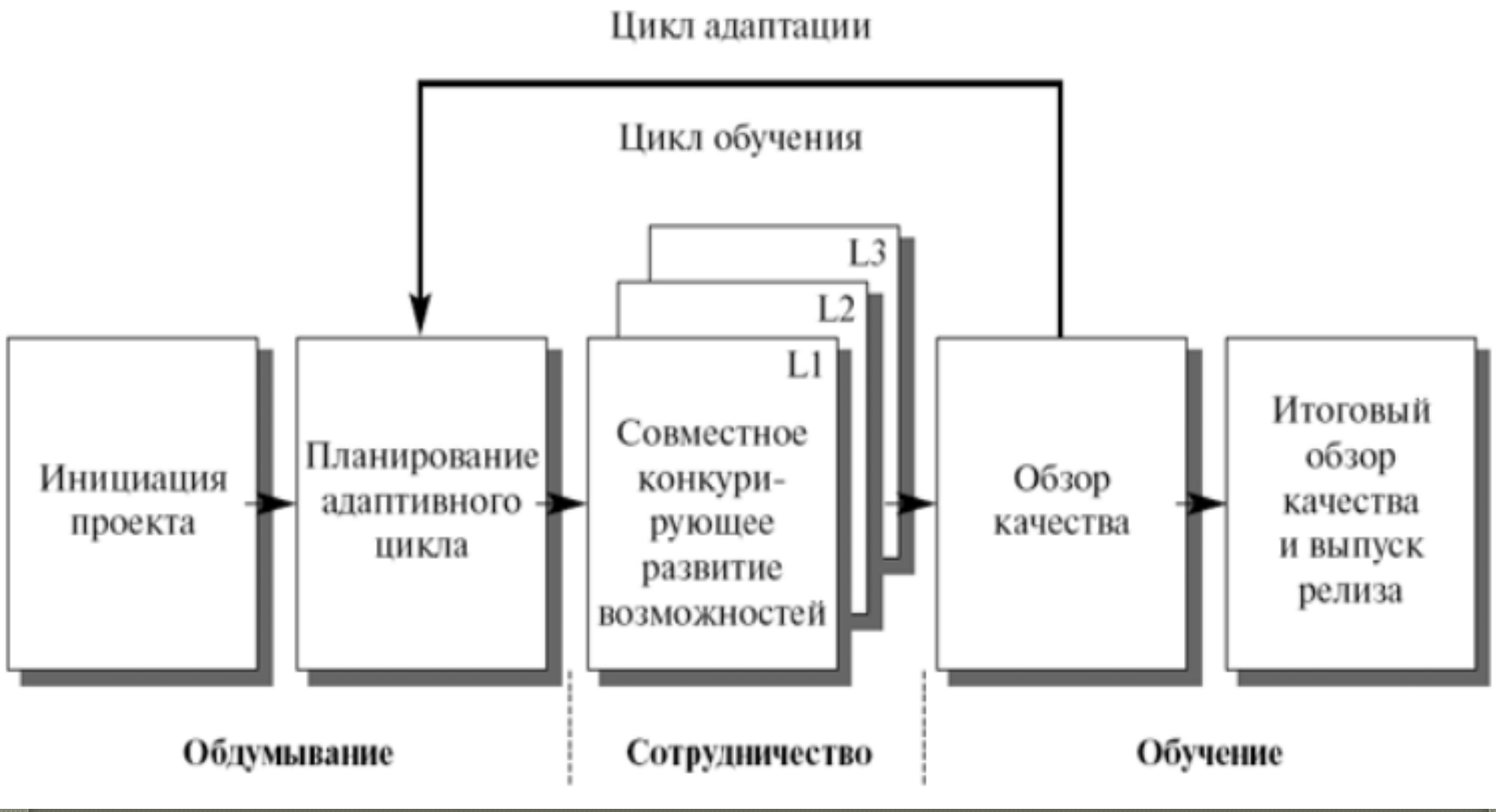
всегда решать наиболее важную задачу первой

- ***Задача*** **—** это незавершенная частная задача программирования.
- ***Наиболее важная задача*** **—** это комбинация *большого размера*, *срочности* и *устойчивости*.
- **Задачи *большого размера*** ценны для пользователей настолько, насколько они функциональны.
- ***Срочные*** **задачи** своевременны настолько, насколько должны быть, иначе задерживается остальная работа.
- ***Устойчивые*** **задачи** проверены и испытаны. Разработчики могут благополучно сфокусироваться на другом.
- ***Решить*** означает привести в ***состояние стабильности***.

ЭКСТРЕМАЛЬНОЕ ПРОГРАММИРОВАНИЕ

- Короткий цикл обратной связи
- Разработка через тестирование
- Игра в планирование
- Заказчик всегда рядом
- Парное программирование
- Непрерывный, а не пакетный процесс
- Непрерывная интеграция
- Рефакторинг
- Частые небольшие релизы
- Понимание, разделяемое всеми
- Простота проектирования
- Метафора системы
- Коллективное владение кодом
- Стандарт оформления кода
- Социальная защищённость программиста:
- 40-часовая рабочая неделя

Адаптивная разработка (ASD) по Хайсмиту




Разработка
Тестирование
DevOps
Эксплуатация

РАЗРАБОТЧИКИ СТАНДАРТОВ, РЕГЛАМЕНТИРУЮЩИХ РАЗЛИЧНЫЕ АСПЕКТЫ ЖИЗНЕННОГО ЦИКЛА:

• **IEEE** — Институт инженеров по электротехнике и электронике;
• **ISO** —Международная организация по стандартизации;
• **EIA** —Ассоциация электронной промышленности;
• IEC —Международная комиссия по электротехнике;
• ANSI —Американский национальный институт стандартов;
• SEI —Институт программной инженерии;
• ECMA —Европейская ассоциация производителей компьютерного оборудования

Процессы жизненного цикла ПО по ISO 12207

Основные процессы	Поддерживающие процессы	Организационные процессы	Адаптация
• Приобретение ПО; • Передача ПО в использование; • Разработка ПО; • Эксплуатация ПО; • Поддержка ПО	• Документирование; • Управление конфигурациями; • Обеспечение качества; • Верификация; • Валидация; • Совместные экспертизы; • Аудит; • Разрешение проблем	• Управление проектом; • Управление инфраструктурой; • Усовершенствование процессов; • Управление персоналом	• Адаптация описываемых стандартом процессов под нужды конкретного проекта

Процессы жизненного цикла ПО по ISO 12207

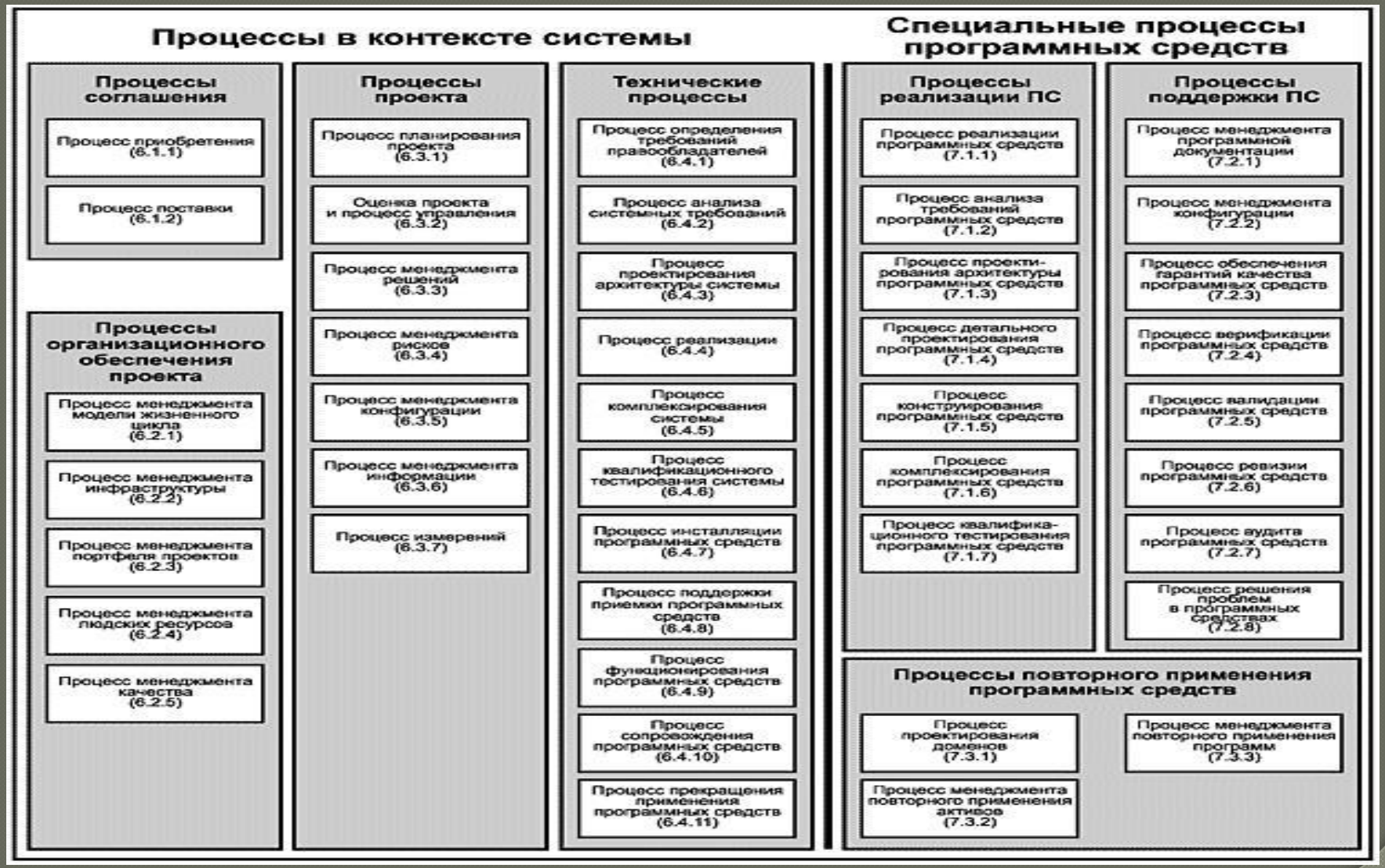
Процессы выработки соглашений	Процессы уровня организации	Процессы уровня проекта	Технические процессы	Специальные процессы
Приобретение системы; Поставка системы	Управление окружением; Управление инвестициями; Управление процессами; Управление ресурсами; Управление качеством	Планирование; Оценивание; Мониторинг; Управление рисками; Управление конфигурацией; Управление информацией; Выработка решений	Определение требований; Анализ требований; Проектирование архитектуры; Реализация; Интеграция; Верификация; Валидация; Передача в использование; Эксплуатация; Поддержка; Изъятие из эксплуатации	Адаптация описываемых стандартом процессов под нужды конкретного проекта

Процессы жизненного цикла ПО. Группа стандартов IEEE

Процесс разработки включает:

- ***подготовительную работу;***
- ***анализ требовании к системе;***
- ***проектирование архитектуры;***
- ***анализ требований к программному обеспечению;***
- ***проектирование архитектуры программного обеспечения;***
- ***детальное проектирование программного;***
- ***кодирование и тестирование программного обеспечения;***
- ***интеграцию программного;***
- ***квалификационное тестирование программного обеспечения;***
- ***интеграцию системы;***
- ***квалификационное тестирование системы;***
- ***установку программного обеспечения;***
- ***приемку программного обеспечения .***

Национальный стандарт РФ
ГОСТ Р ИСО/МЭК 12207-2010



ГОСТ 19.102-77 основные этапы разработки ПО:

- постановка задачи (стадия «Техническое задание»);
- анализ требований и разработка спецификаций (стадия «Эскизный проект»);
- проектирование (стадия «Технический проект»);
- реализация (стадия «Рабочий проект»).

***ГОСТ Р ИСО/МЭК 12207-2010*)**
устанавливает 11 процессов:

a) определение требований правообладателей;
b) анализ системных требований;
c) проектирование архитектуры системы;
d) процесс реализации;
e) процесс комплексирования системы;
f) процесс квалификационного тестирования системы;
g) процесс инсталляции программных средств;
h) процесс поддержки приемки программных средств;
i) процесс функционирования программных средств;
j) процесс сопровождения программных средств;
k) процесс изъятия из обращения программных средств.

Особенности создания программного продукта

Принципы работы с требованиями к программному обеспечению.

Требование — это любое условие, которому должна соответствовать разрабатываемая система или программное средство. *Требованием* может быть возможность, которой система должна обладать и ограничение, которому система должна удовлетворять.

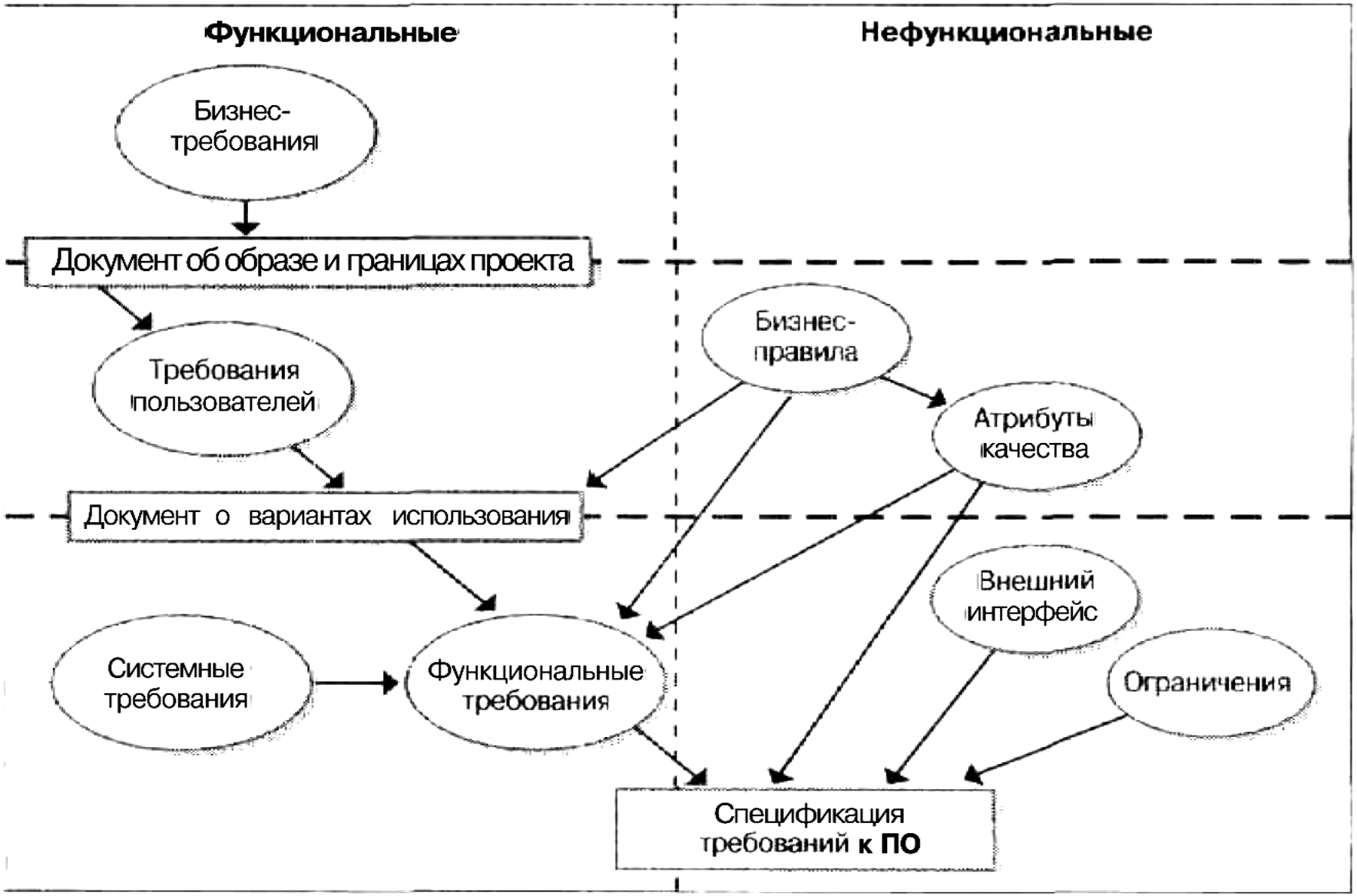
1. Условия или возможности, необходимые пользователю для решения проблем или достижения целей;
2. Условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;
3. Документированное представление условий или возможностей для пунктов 1 и 2.

***Требование* должно обладать следующими характеристиками:**

- *Единичность*
- *Завершенность*
- *Последовательность*
- *Атомарность*
- *Отслеживаемость*
- *Актуальность*
- *Выполнимость*
- *Недвусмысленность*
- *Обязательность*
- *Проверяемость*

Управление требованиями — процесс, включающий идентификацию, выявление, документирование, анализ, отслеживание, определение приоритетов требований, достижение соглашений по требованиям и затем управление изменениями и уведомление заинтересованных лиц. Управление требованиями — непрерывный процесс на протяжении всего жизненного цикла создания ПО.

Модель Вигерса



Программные требования

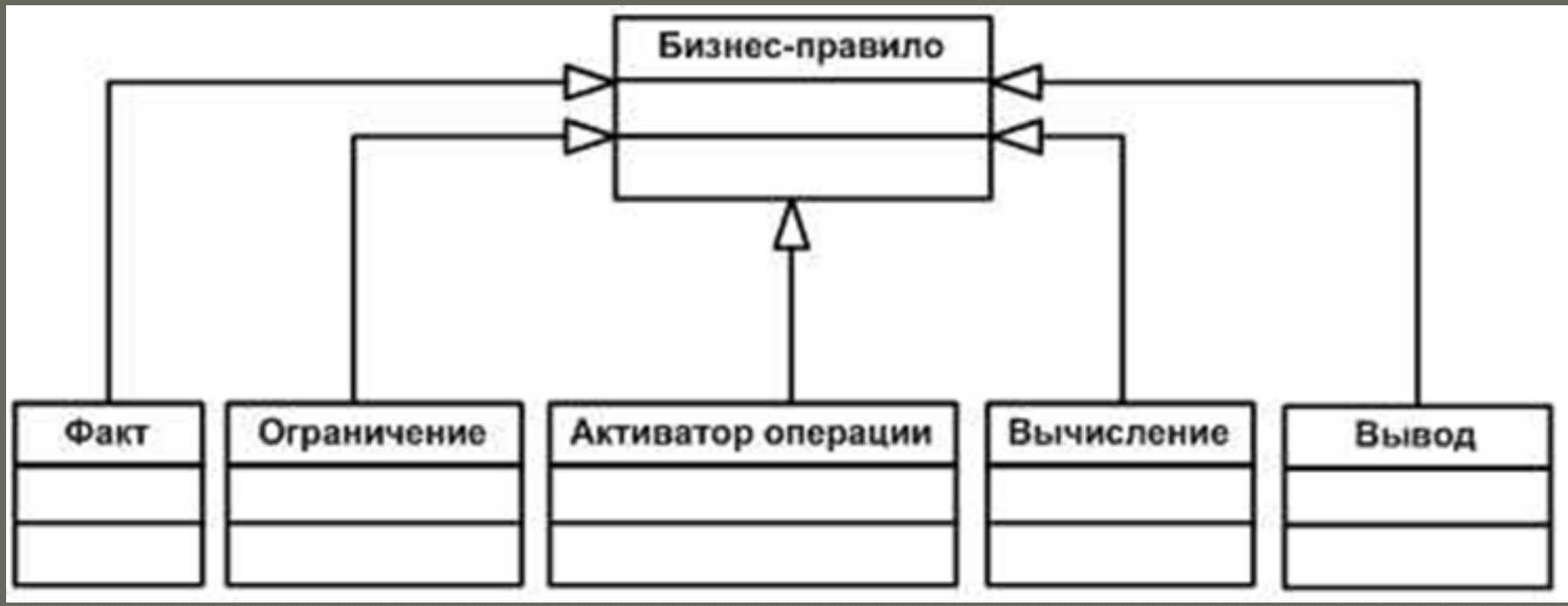
Функциональные и нефункциональные требования

- *Потребности*

- *Группа функциональных требований*
  - *Бизнес-требования*
  - *Пользовательские требования*
  - *Функциональные требования*

- *Группа нефункциональных требований*
  - *Бизнес-правила*
  - *Внешние интерфейсы*
  - *Атрибуты качества*
  - *Ограничения*

Программные требования



Процесс работы с требованиями

- *Неформальная постановка требований в переписке по электронной почте*
- *Требования в виде документа*
- *Требования в виде графа с зависимостями*
- *Формальная модель требований*

Извлечение требований

- *Источники требований*
- *Техники извлечения требований*
  - *Интервьюрирование*
  - *Совещания*
  - *Мозговой штурм*
  - *Прототипы*
  - *Разъясняющие встречи*
  - *Наблюдение*

Анализ требований

- Обнаружение и разрешение конфликтов между требованиями;
- Определение границ задачи, решаемой создаваемым ПО, границ и содержания программного проекта;
- Детализация системных требований для установления программных требований;

Проверка требований

- Определение системы
- Системные требования
- Программные требования

Границы системы

Программные требования.
инструментальные средства – системы управления требованиями

	IBM Rational RequisitePro	IBM Rational/ Telelogic DOORS	Borland Caliber RM	Redmine
Web-интерфейс	+	+	+	+
Трассировка требований	+	+	+	-
Авторизация доступа	+	+	+	+
Права доступа	+	+	+	-
Интеграция со средствами визуального моделирования UML	+	+	+	+
Интеграция с системами управления версиями	+	+	+	+
Работа с нетекстовыми объектами (изображения и др.)	-	+	+	+
Оповещения об изменении требований (документов)	+	+	+	-
Бесплатность	-	-	-	+
Легковесность	-	-	-	+