

Способы декомпозиции предметной области

Современные языки - несколько стилей разработки.

Стиль и способ декомпозиции предметной области.

Способ декомпозиции и концептуальная основа.

Процедурная декомпозиция

(структурный подход)

Общий алгоритм программы и отдельные фрагменты
(подпрограммы)

Последовательности действий в предметной области задачи и
зависимые функциональные части (структуры)

Функциональные части и повторное использование в текущем
и других проектах

Декомпозиция и метод пошаговой детализации

Метод пошаговой детализации

Этап 1

- Основная задача (основное действие) разбивается на подзадачи.
- Определяется последовательность выполнения подзадач.
- Определяются возможные альтернативы подзадач.
- Определяется частота выполнения подзадач.

Метод пошаговой детализации

Этап 2

Каждая подзадача, разбивается на подзадачи с использованием тех же структур.

Этап 3

Процесс продолжается, пока на очередном уровне не получится подзадача, которая достаточно просто реализуется средствами выбранного языка

(конкретно это зависит от уровня языка и библиотеки).

Сложная система

верхний уровень

Иерархия подпрограмм

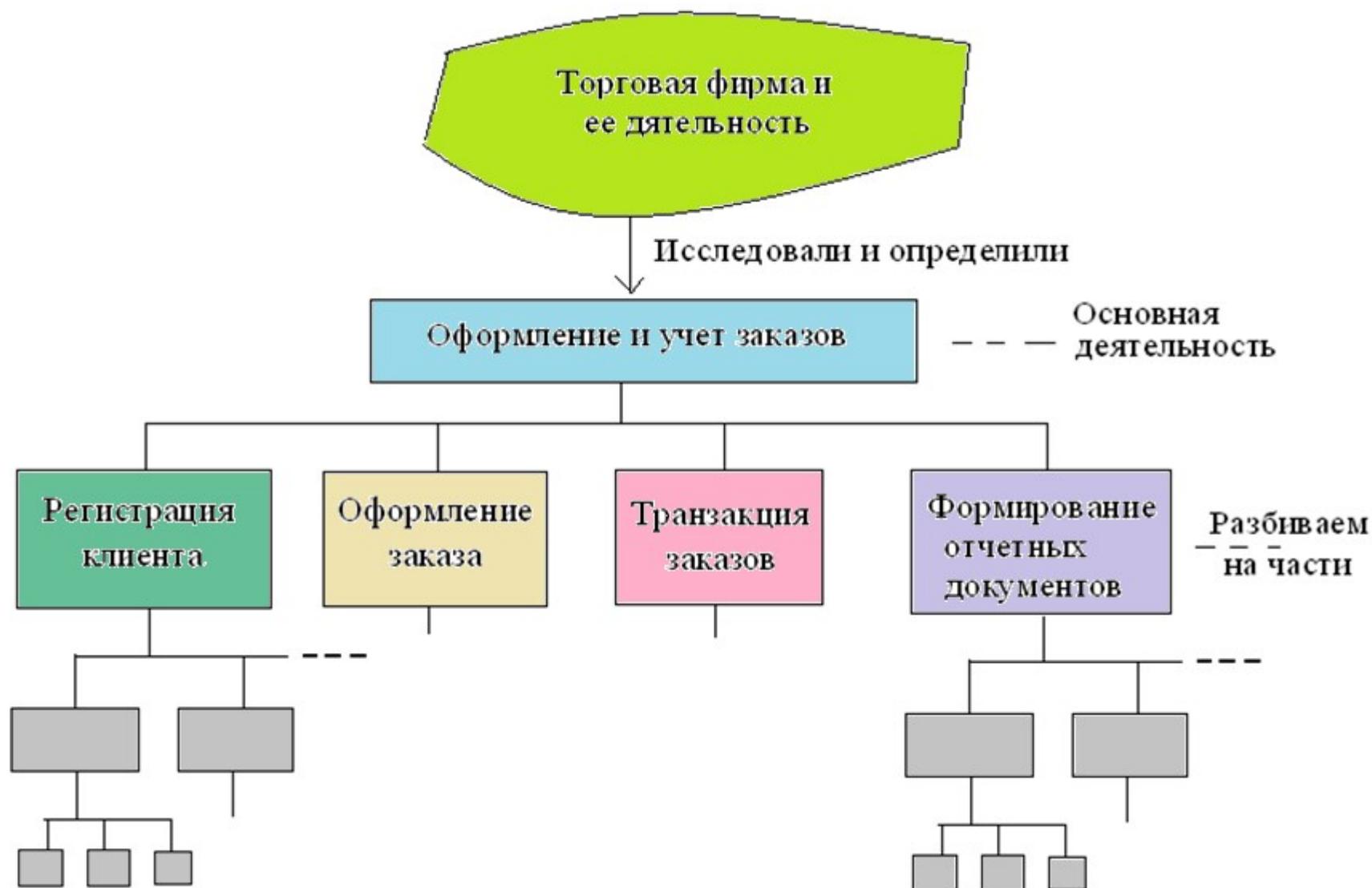
нижний уровень

Простейшие процедуры и функции (иерархия)

Пример.

Задача – автоматизировать деятельность торговой фирмы, т.е. создать на компьютере программную модель.

Предметная область - торговая фирма и ее деятельность.



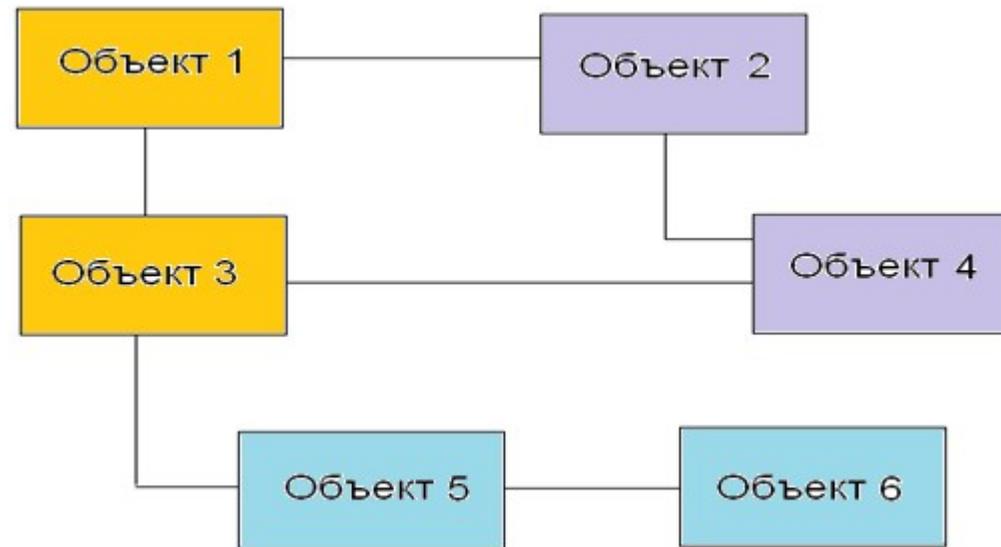
Объектно-ориентированная декомпозиция

(объектно-ориентированный подход)

- **Выделение объектов предметной области.**
(Объект - сущность (реальная или абстрактная), имеющая четко определенное функциональное назначение в данной предметной области).
- **Определение свойств** (статических, динамических, количественных, качественных) **объектов.**
- **Определение отношений между объектами.**
(Определяется поведение объектов и отношения между объектами).

В итоге:

- выделенные объекты предметной области могут стать объектами-классами;
- выделенные объекты предметной области могут представлять классы сущности программной модели (например, в виде контекстной диаграммы классов).



Логическая декомпозиция

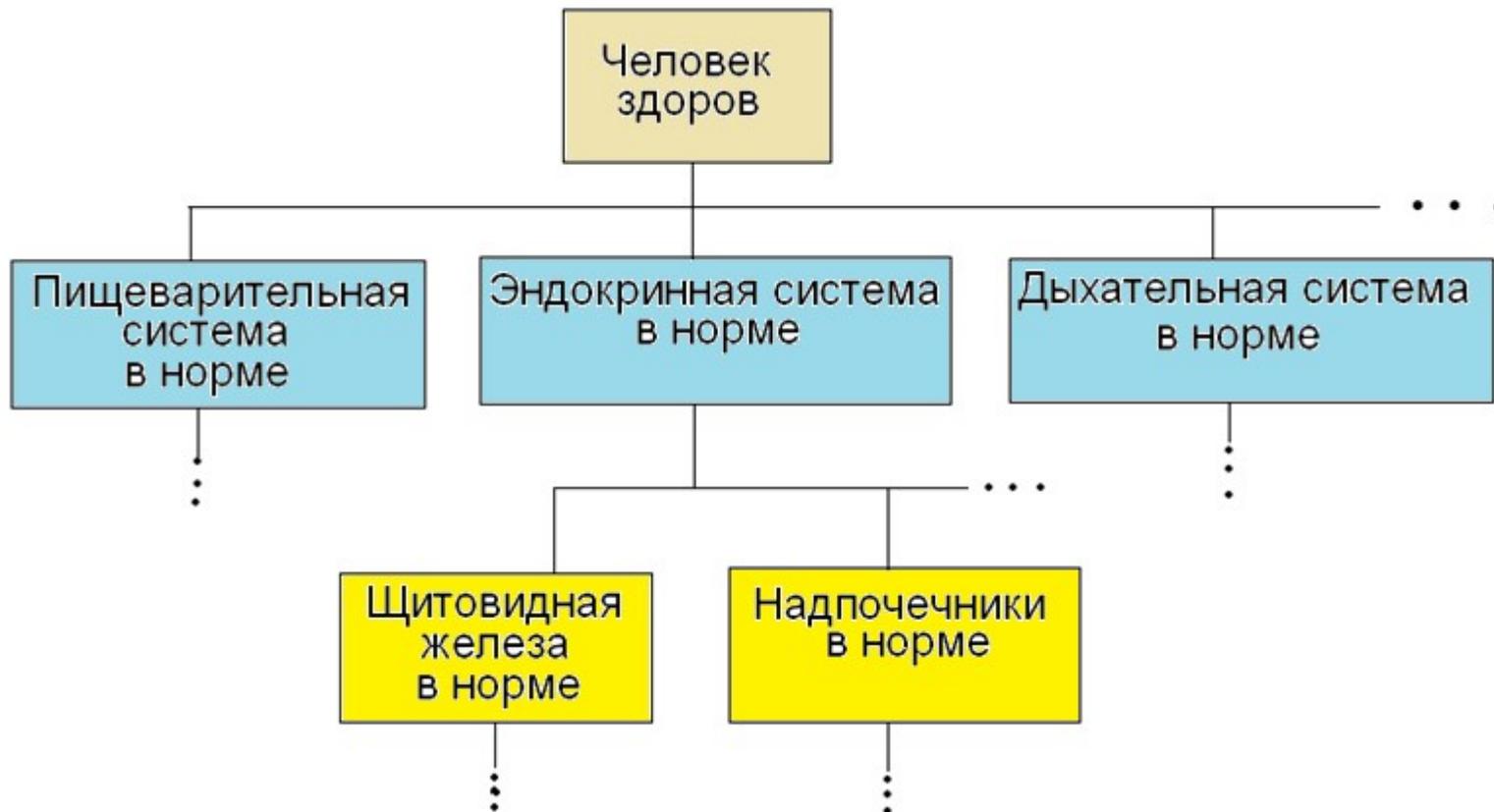
(логически-ориентированный подход)

- Предполагает выделение целей и подцелей в предметной области задачи.
- Используется при разработке систем с целью, т.е. интеллектуальных систем.

Например:

Задача - необходимо создать диагностическую медицинскую экспертную систему, которая помогала бы пользователям определять состояние здоровья.

Другими словами, система должна пытаться достичь цели, например, что человек здоров. В этом случае результатом логической декомпозиции может быть следующая схема (фрагмент).



Далее

на основе результатов логической декомпозиции
(+ другие виды декомпозиций)
могут быть построены:

- модель представления знаний
- схемы механизмов логического вывода
- и др.

Декомпозиция по правилам продукции

(подход, ориентированный на правила)

Предполагает формирование правил вида:

«Если ...То»

- служит основой для продукционной модели;
- модель можно реализовать на продукционных языках, например, **ПРОЛОГ**.

О Прологе

- Является *декларативным языком, т.е.* представляет собой набор логических взаимосвязанных описаний, определяющих цель, ради которой она написана.
- Освобождает программиста от составления программы в виде последовательности действий.
- Для работы с правилами используется стратегия поиска решений в глубину и обратный порядок вывода.

Инвариантная декомпозиция

(подход, ориентированный на ограничения)

Предполагает описание предметной области в виде инвариантных соотношений.

Рассмотрим на примере Пролога.

Общая форма записи правила имеет вид:

<заголовок правила>:- <тело правила>.

Правило - утверждение о связи некоторого факта с другими фактами.

Заголовок представляет собой предикат.

Тело состоит из термов, которые могут быть связаны между собой "," или ";". (","- означает И, ";"- означает ИЛИ).

Между телом и заголовком стоит символ ":-", который означает ЕСЛИ.

Например, правило состоящее из двух термов может описано следующим образом:

likes(tom,kathy) :- likes(kathy,computer), likes(kathy,apples)

Пример 1.

Predicates

z a(integer) b(string)

Goal z.

Clauses

z:-a(X),write("Ok1").

z:-b(X),write("Ok2").

z:-write("no Ok").

a(X):-readln(Y),str_int(Y,X).

b(X):-readint(Y),str_int(X,Y).

В итоге у предиката `str_int(X,Y)` три рабочих варианта

ИСПОЛЬЗОВАНИЯ.

Пример 2.

Predicates

z a b c

Clauses

z:-a,b,c,fail.

a:-write("a1").

a:-write("a2").

b:-write("b1").

b.

c:-write("c1").

c:-write("c2").

Goal z.

ВЫВОДЫ

Нельзя назвать лучшего стиля, так как каждый стиль ориентирован на свою область применения.

Например:

- Логически-ориентированный подход - для проектирования экспертных систем;
- Объектно-ориентированный - для широкого круга задач, связанных с разработкой сложных систем и др.

