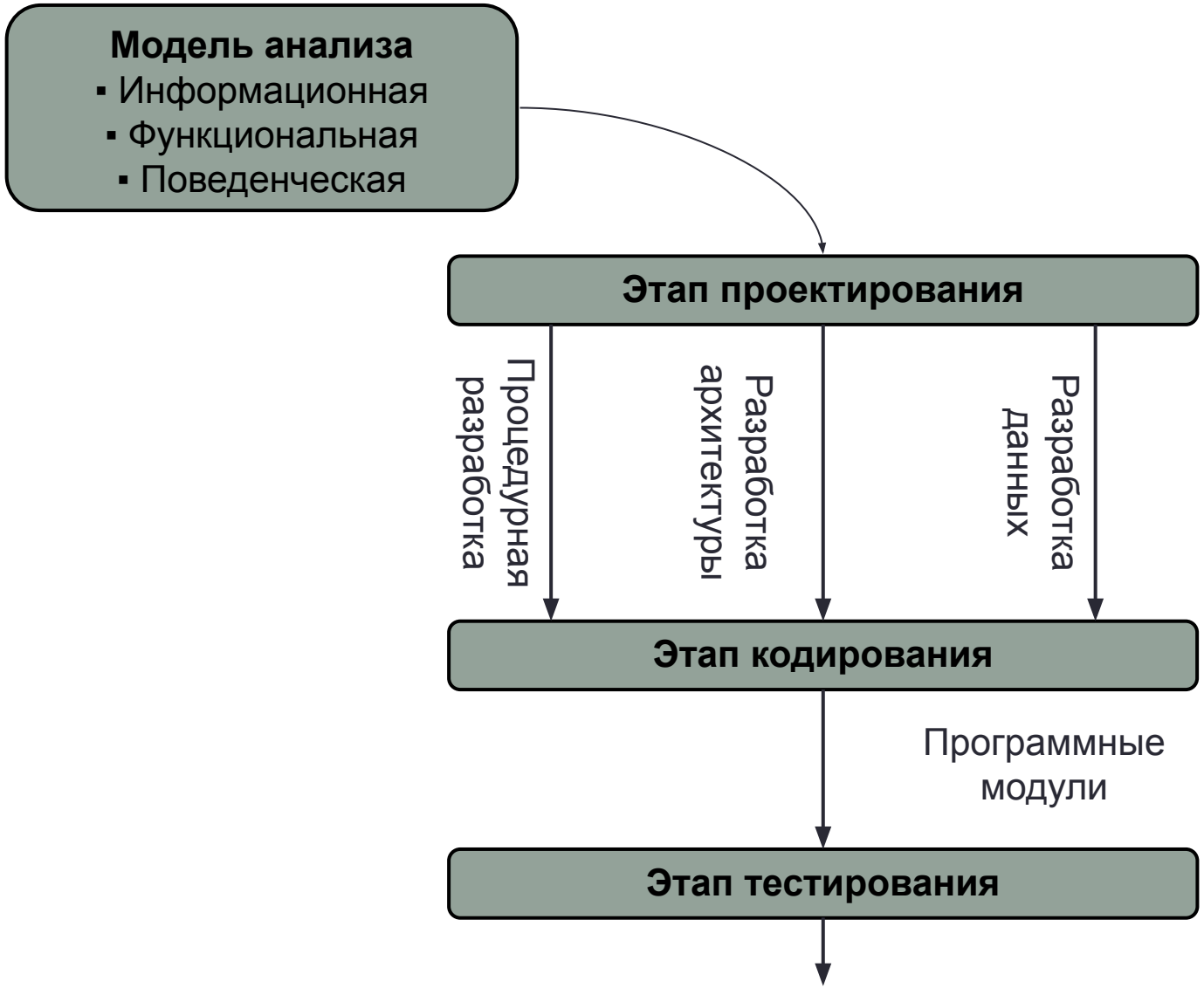
ЛЕКЦИЯ 5

Моделирование архитектуры и управления

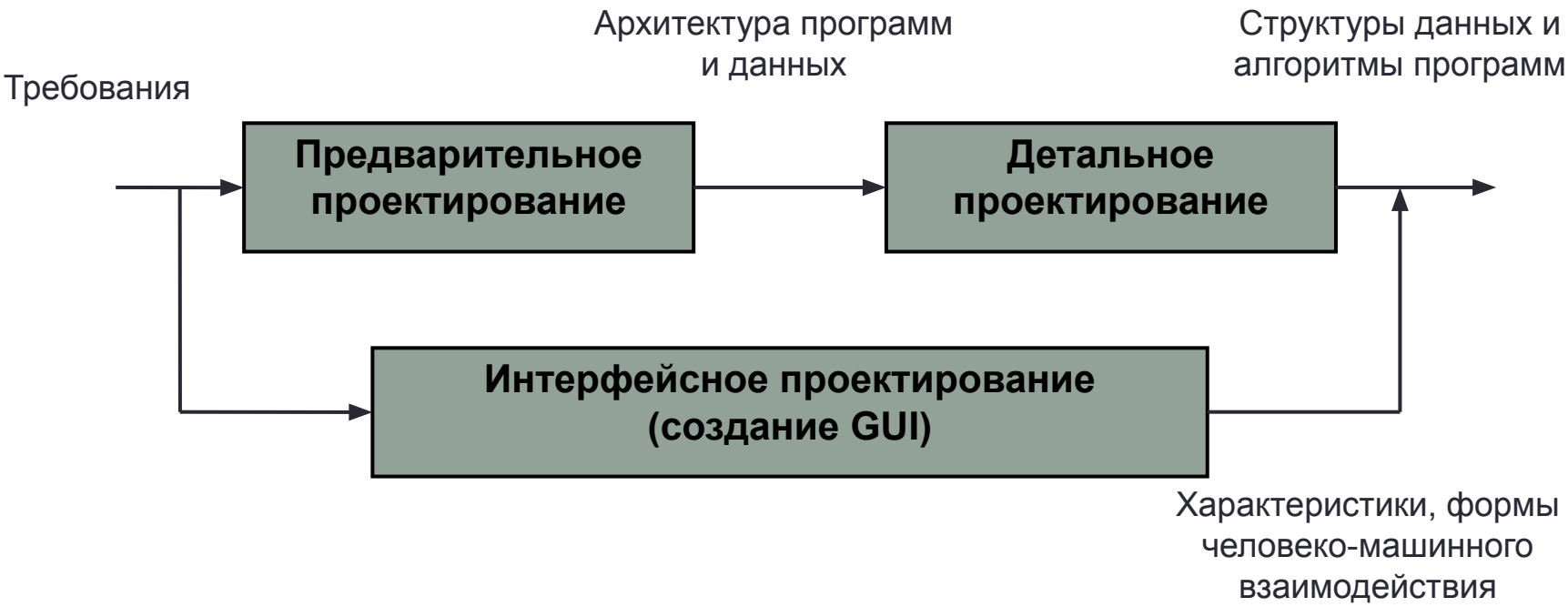
Модули и модульная декомпозиция

Оценка сложности программной системы

Особенности процесса синтеза программных систем (ПС)



Особенности этапа проектирования



Предварительное проектирование

Предварительное проектирование обеспечивает:

- идентификацию подсистем;
- определение основных принципов управления подсистемами, взаимодействия подсистем.

Предварительное проектирование включает:

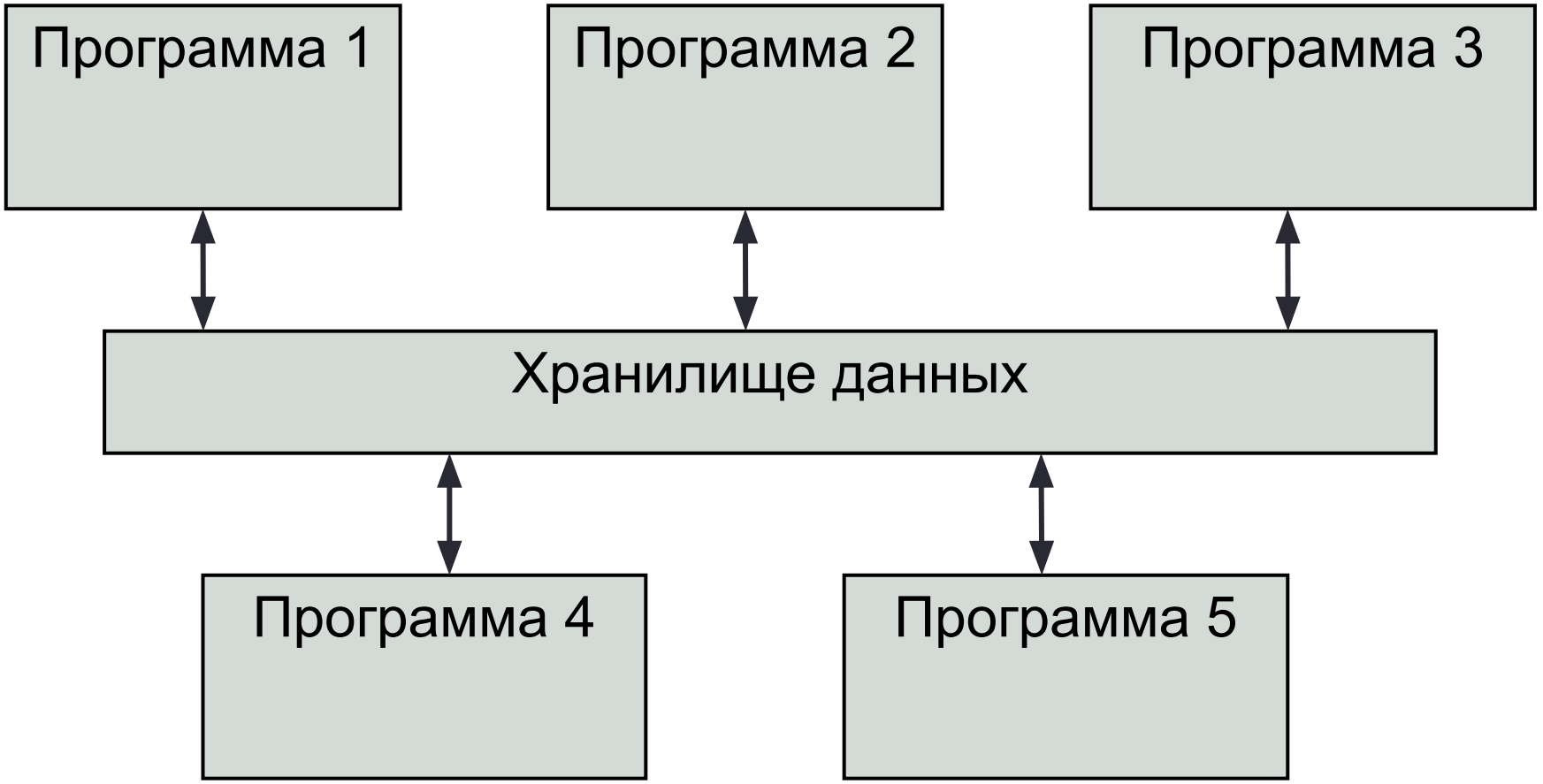
1. Структурирование системы.
2. Моделирование управления.
3. Декомпозиция подсистем на модули.

Структурирование системы

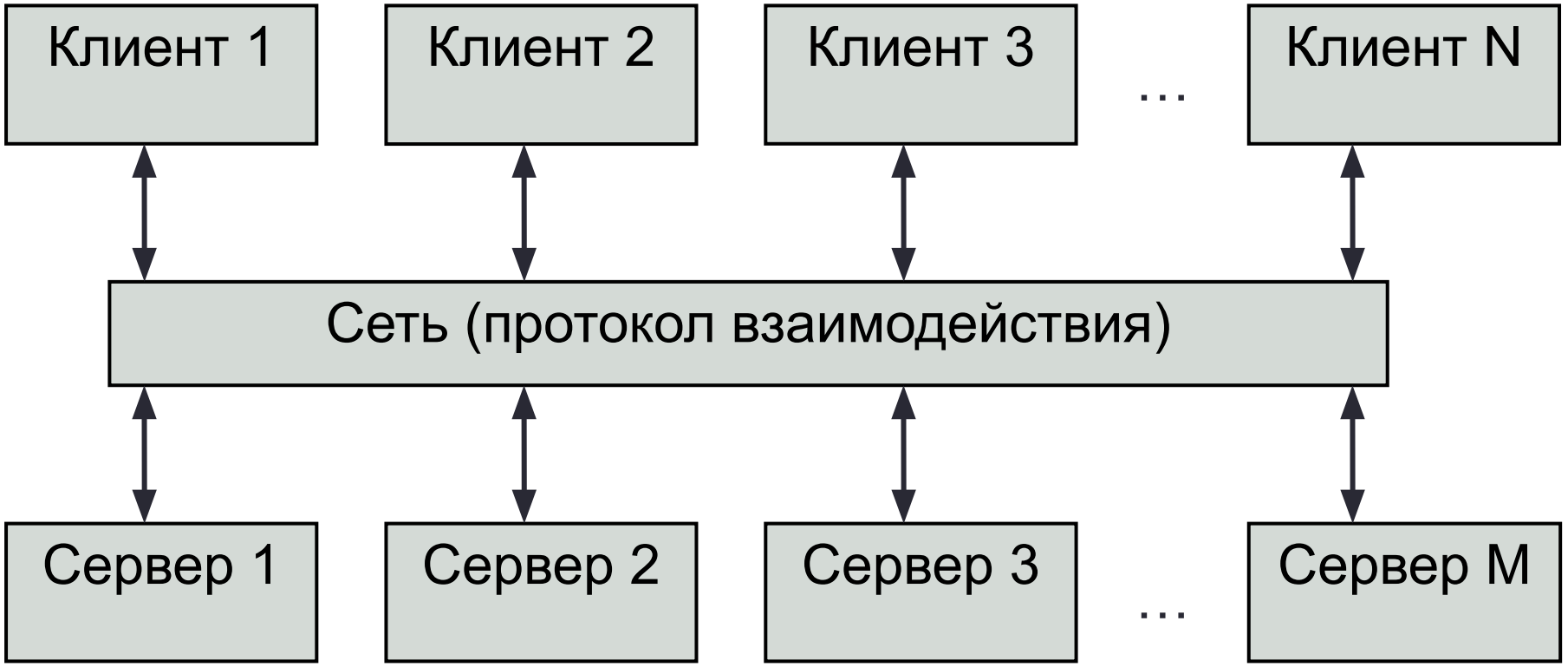
Три модели системного структурирования:

- модель хранилища данных;
- модель клиент-сервер;
- трехуровневая модель.

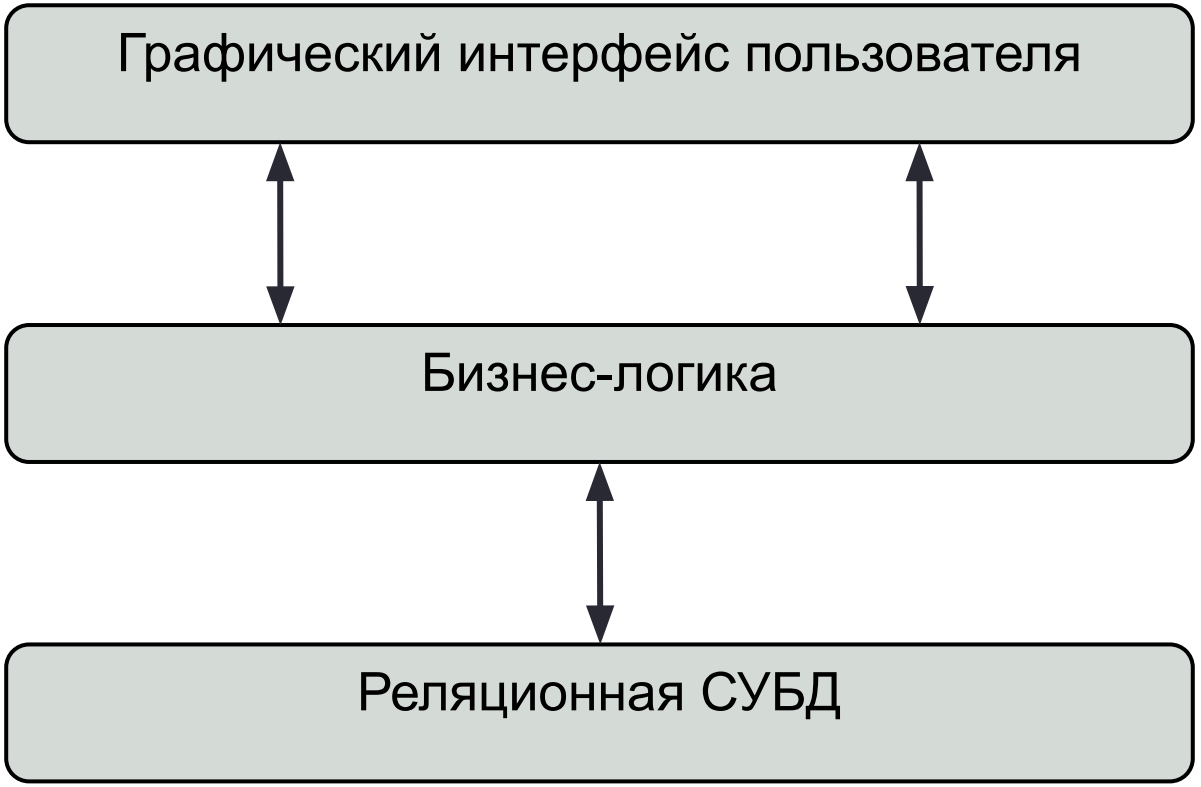
Модель хранилища данных



Модель клиент-сервер



Трехуровневая модель



Моделирование управления

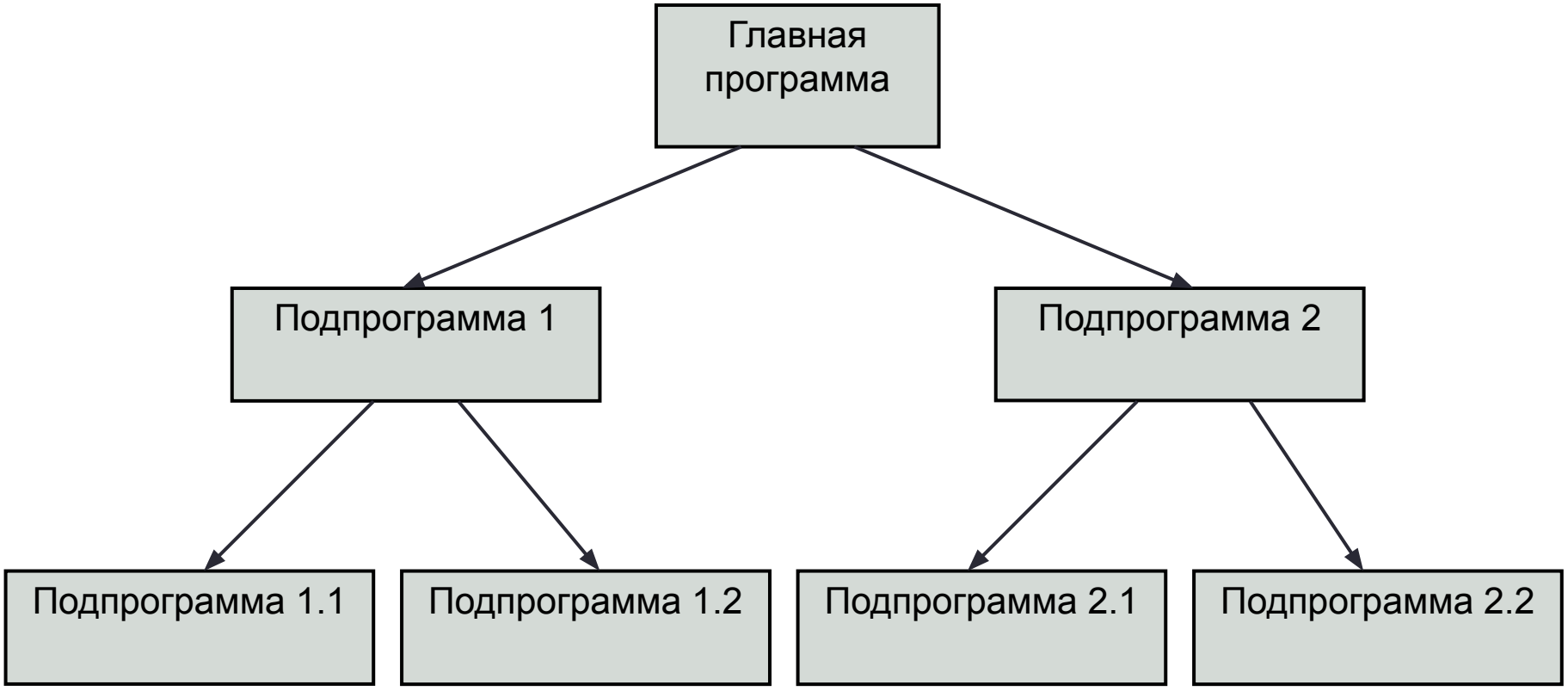
1 Модель централизованного управления:

- Модель вызов-возврат,
- Модель менеджера;

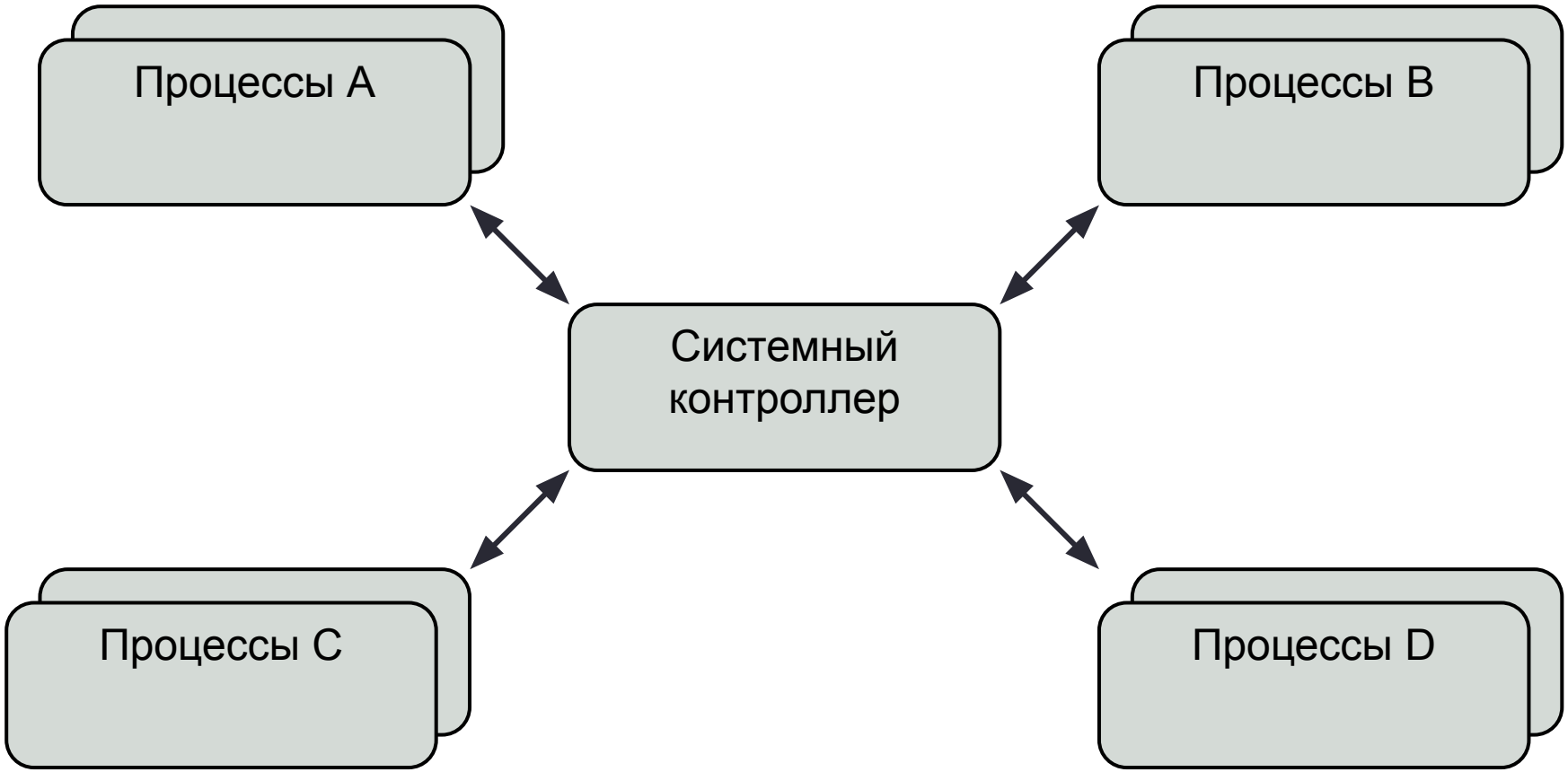
2 Модель событийного управления:

- Широковещательная модель,
- Модель, управляемая прерываниями.

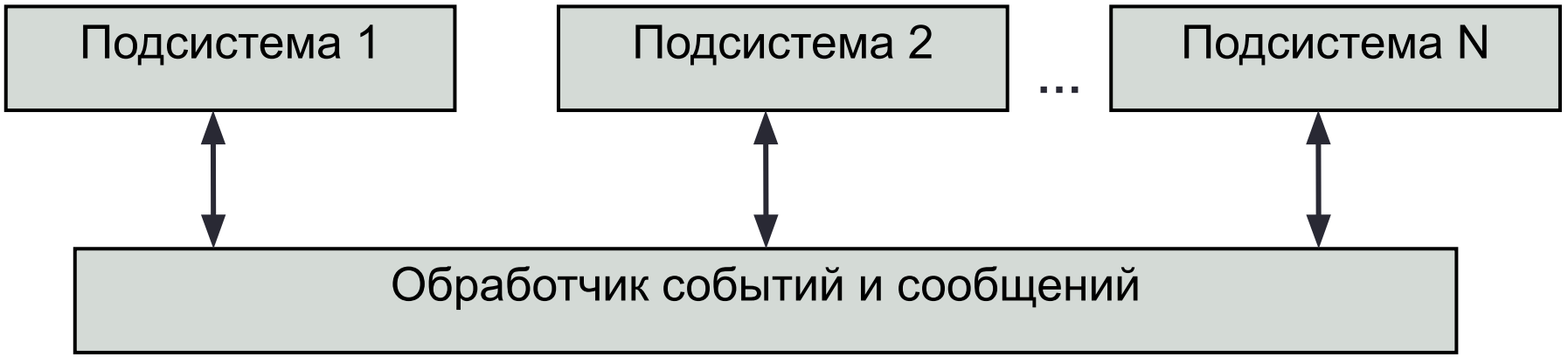
Модель вызов-возврат



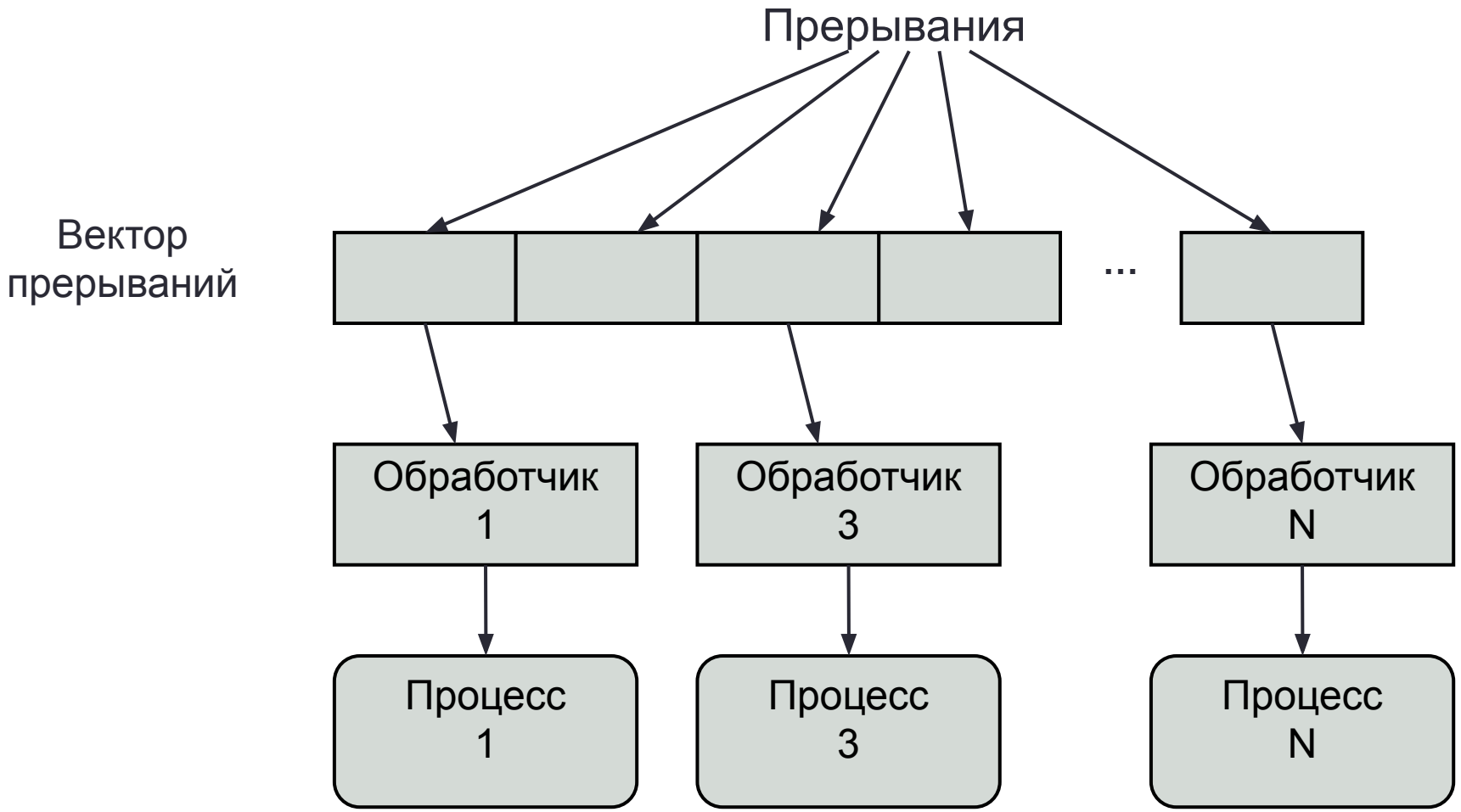
Модель менеджера



Широковещательная модель



Модель, управляемая прерываниями



Декомпозиция подсистем на модули

Два типа моделей декомпозиции:

- Модель потока данных.
- Модель объектов.

Модульность

Модуль – фрагмент программного текста, являющийся строительным блоком для физической структуры системы.

Модульность – свойство системы, которая может подвергаться декомпозиции на ряд внутренне связанных и слабо зависящих друг от друга модулей.

Модульность

Пусть

- C(x) – функция сложности решения проблемы,
- T(x) – функция затрат времени на решение проблемы.

Для двух проблем p_1 и p_2 из соотношения

$$C(p_1) > C(p_2)$$

следует, что

$$T(p_1) > T(p_2).$$

Модульность

Из практики решения проблем человеком следует:

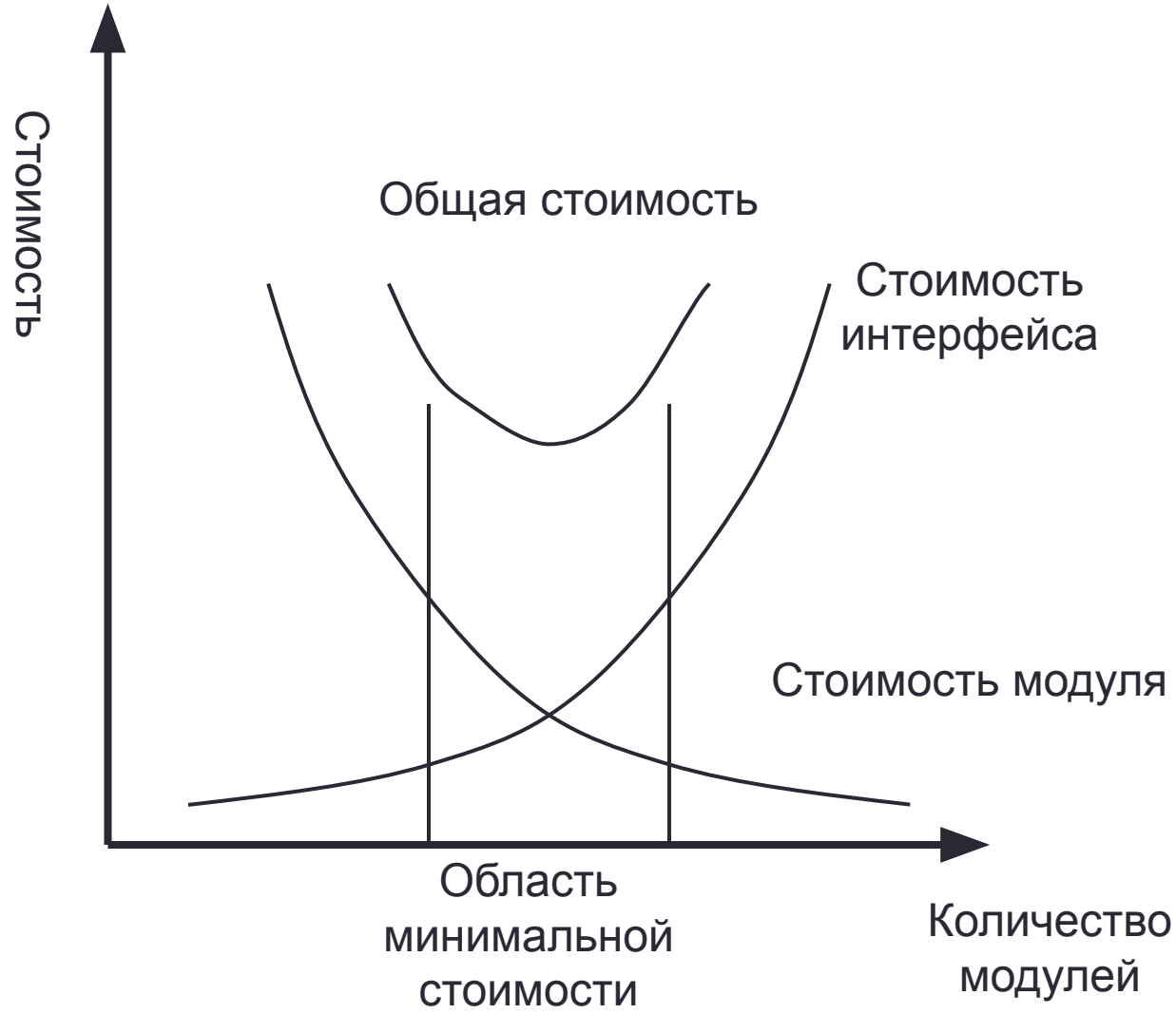
$$C(p_1+p_2) > C(p_1) + C(p_2).$$

Учитывая ранее сказанное:

$$T(p_1+p_2) > T(p_1) + T(p_2).$$

Это и есть принцип «разделяй и властвуй».

Модульность

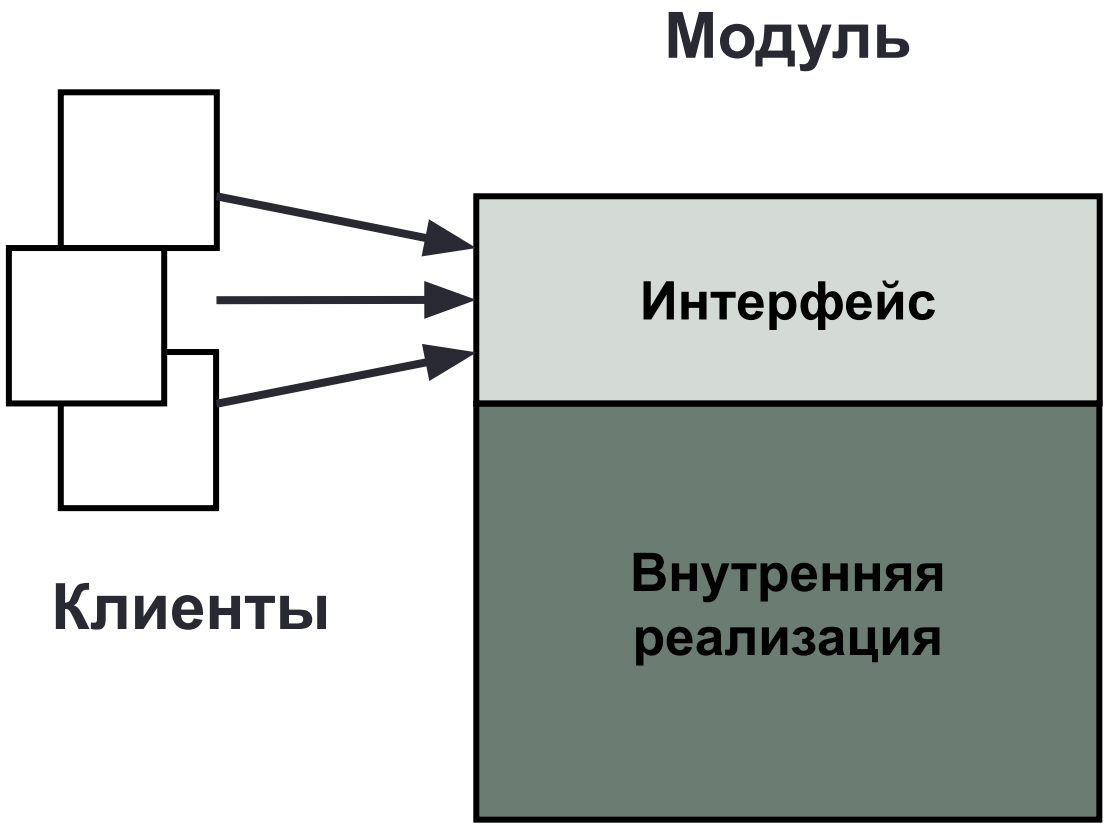


Модульность

Оптимальный модуль должен удовлетворять двум критериям:

- снаружи он проще чем внутри,
- его проще использовать, чем построить.

Информационная закрытость



Информационная закрытость

Информационная закрытость означает следующее:

1. все модули независимы, обмениваются только информацией, необходимой для работы;
2. доступ к операциям и структурам данных модуля ограничен.

Достоинства информационной закрытости:

- обеспечивает возможность разработки модулей различными, независимыми коллективами;
- обеспечивается легкая модификация системы.

Связность модуля

Связность модуля (внутренняя характеристика) – это мера зависимости его частей.

Семь типов связности:

1. Связность по совпадению.
2. Логическая связность.
3. Временная связность.
4. Процедурная связность.
5. Коммуникативная связность.
6. Информационная (последовательная) связность.
7. Функциональная связность.

Характеристика связности модуля

Тип связности	Сопровождаемость	Роль модуля
Функциональная	Лучшая	«Черный ящик»
Информационная		Не совсем «черный ящик»
Коммуникативная		«Серый ящик»
Процедурная	Худшая	«Просвечивающий ящик»
Временная		«Белый ящик»
Логическая		
По совпадению		

Функциональная связность

Функционально связанный модуль содержит элементы, участвующие в выполнении одной и только одной проблемной задачи.

Вычислить синус угла.

Вычислить координаты цели.

Читать запись файла.

Проверять орфографию.

Информационная связность

Элементы-обработчики модуля образуют конвейер для обработки данных – результаты одного обработчика используются как исходные данные для следующего обработчика.

Модуль: Прием и проверка записи

- *прочитать запись из файла*
- *проверить контрольные данные в записи*
- *удалить контрольные данные в записи*
- *вернуть обработанную запись*

Конец модуля

Коммуникативная связность

Элементы-обработчики модуля используют одни и те же данные, например внешние данные.

Модуль: Отчет и средняя зарплата

- *используется таблица зарплата служащих*
- *сгенерировать отчет по зарплате*
- *вычислить параметр средняя зарплата*
- *вернуть отчет по зарплате, средняя зарплата*

Конец модуля

Процедурная связность

Модуль состоит из элементов, реализующих независимые действия, для которых задан порядок работы, то есть порядок передачи управления.

Модуль: Вычисление средних значений

- *используется таблица-А, таблица-Б*
- *вычислить среднее по таблице-А*
- *вычислить среднее по таблице-Б*
- *вернуть среднее табл-А, среднее табл-Б*

Конец модуля

Временная связность

Элементы-обработчики модуля привязаны к конкретному периоду времени (из жизни программной системы).

Модуль: Инициализировать систему

- *перемотать магнитную ленту 1*
- *счетчик магнитной ленты 1 = 0*
- *перемотать магнитную ленту 2*
- *счетчик магнитной ленты 2 = 0*
- *переключатель 1 = вкл.*
- *переключатель 2 = выкл.*

Конец модуля

Логическая связность

Элементы модуля принадлежат одной категории, и из этой категории клиент выбирает выполняемое действие.

Модуль: Пересылка сообщения

- *переслать по e-mail*
- *переслать по факсу*
- *послать в телеконференцию*
- *переслать по ftp-протоколу*

Конец модуля

Связность по совпадению

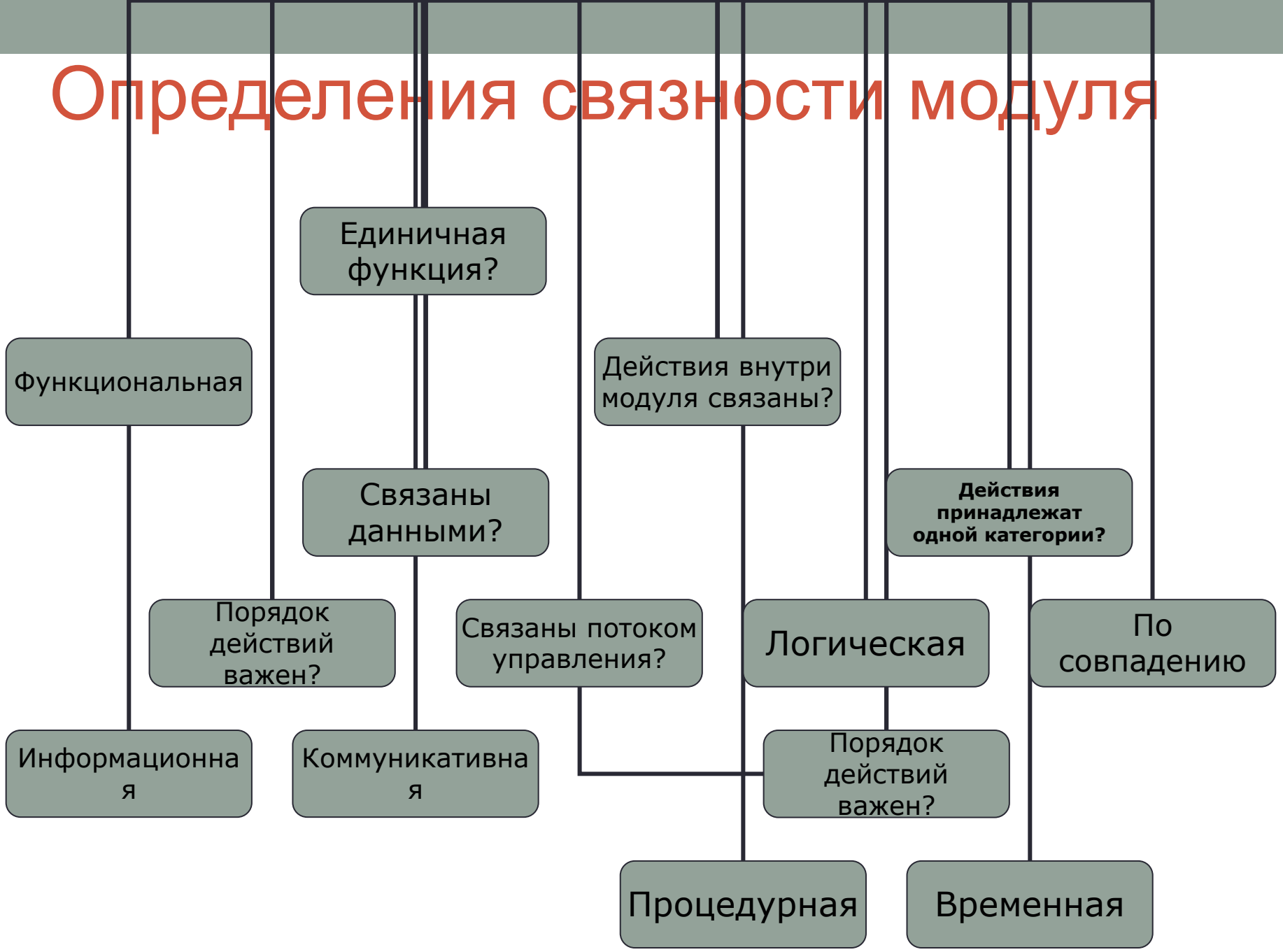
Элементы модуля вообще не имеют никаких отношений друг с другом.

Модуль: Разные функции

- *поздравить с Новым годом*
- *вывести собаку на прогулку*
- *измерить температуру*
- *запастись продуктами*

Конец модуля

Определения связности модуля



Дополнительные правила определения типа связности

- правило параллельной цепи. Если все действия модуля имеют несколько уровней связности, то модулю присваивают самый сильный уровень связности;

- правило последовательной цепи. Если действия в модуле имеют разные уровни связности, то модулю присваивают самый слабый уровень связности.

Сцепление модулей

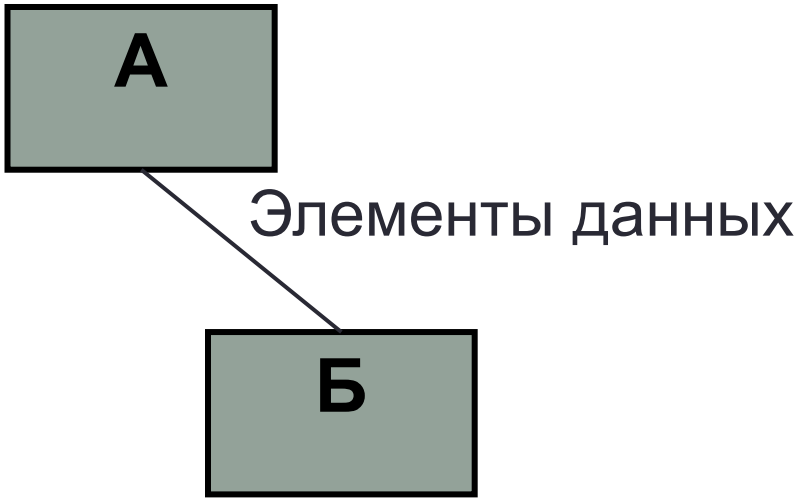
Сцепление модулей (внешняя характеристика) – это мера взаимозависимости модулей по данным.

Шесть типов сцепления:

1. Сцепление по данным
2. Сцепление по образцу
3. Сцепление по управлению
4. Сцепление по внешним ссылкам
5. Сцепление по общей области
6. Сцепление по содержанию

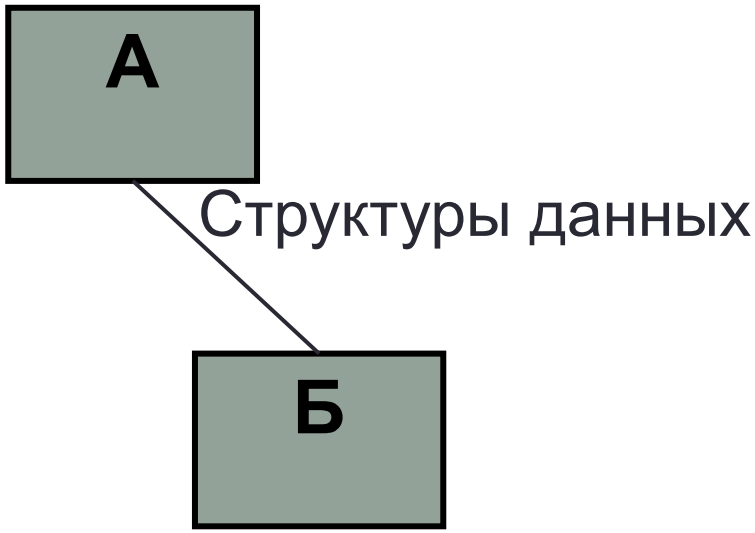
Сцепление по данным

Модуль А вызывает модуль Б. Все входные и выходные параметры вызываемого модуля – простые элементы данных.



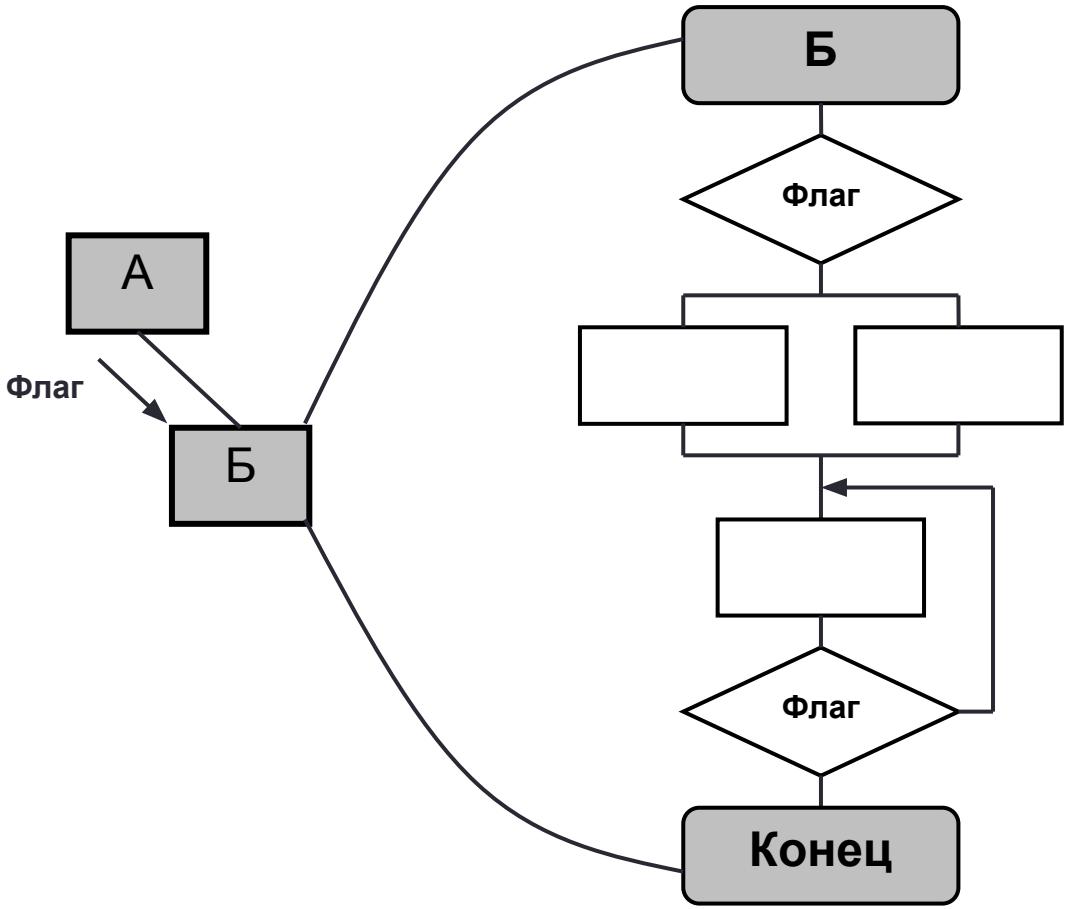
Сцепление по образцу

В качестве параметров используются структуры данных.



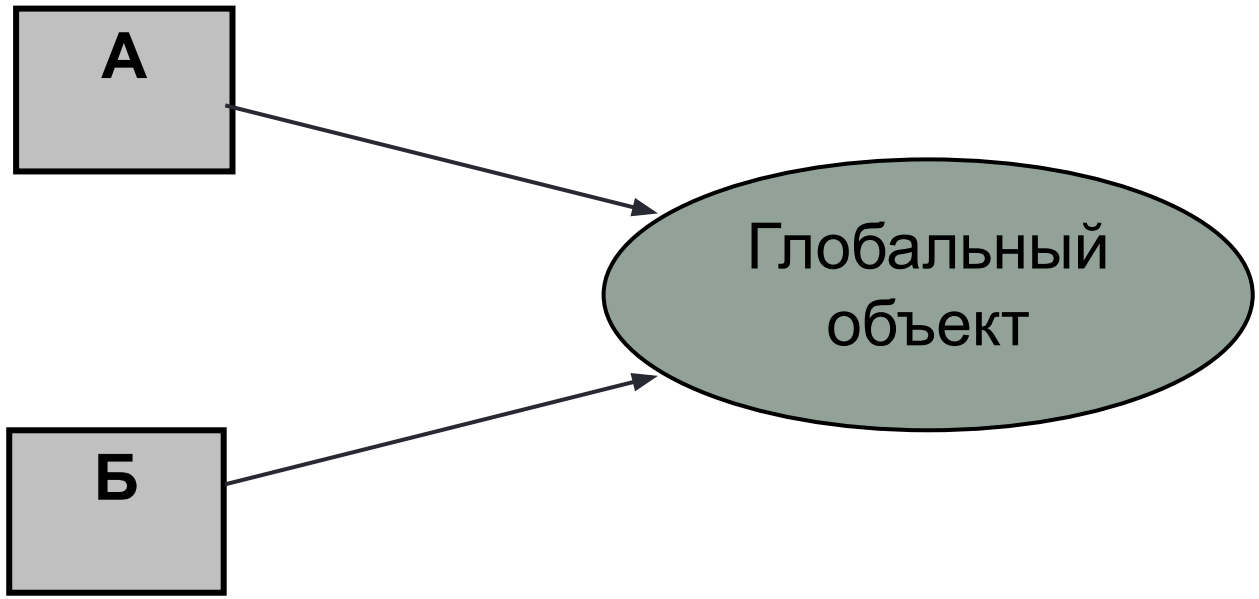
Сцепление по управлению

Модуль А явно управляет функционированием модуля Б, посылая ему управляющие данные.



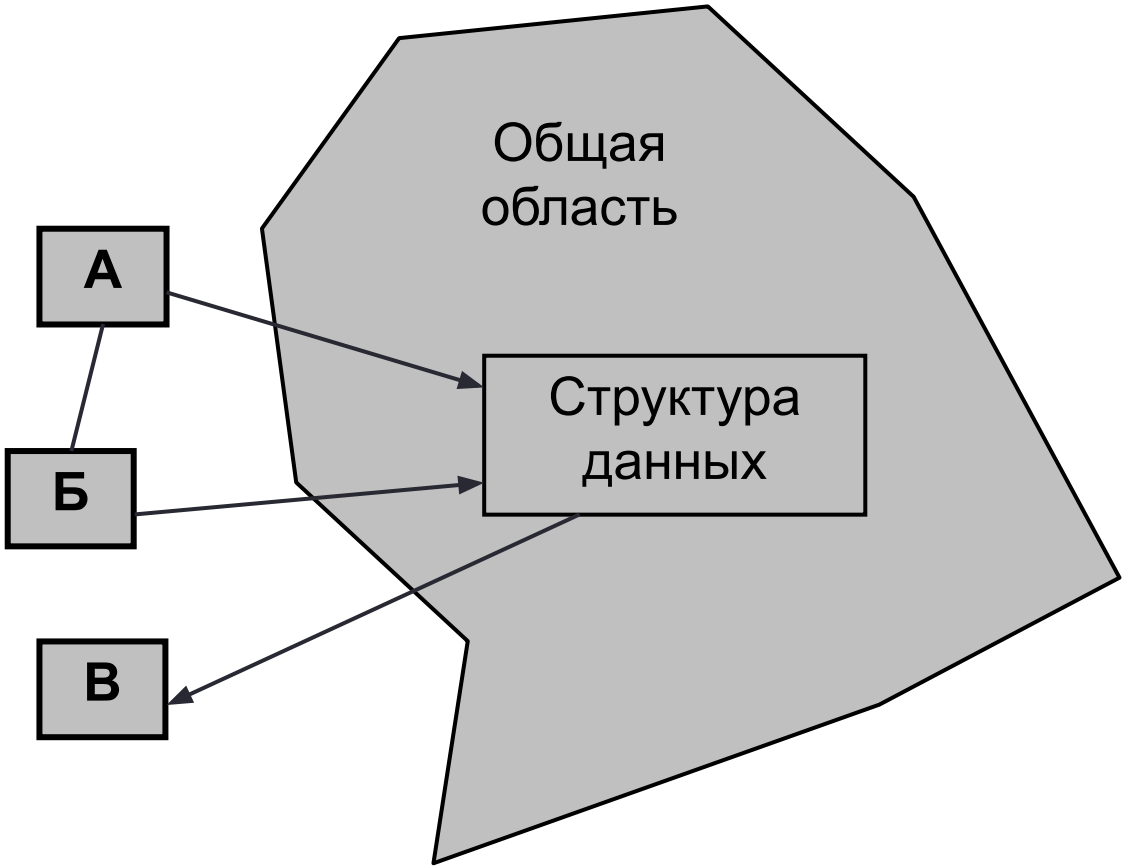
Сцепление по внешним ссылкам

Модули А и Б ссылаются на один и тот же глобальный элемент данных.



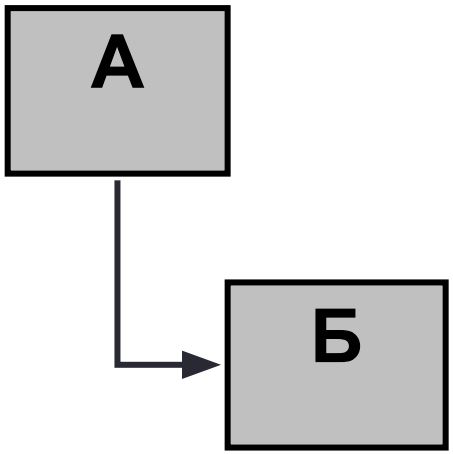
Сцепление по общей области

Модули разделяют одну и ту же глобальную структуру данных



Сцепление по содержанию

Один модуль прямо ссылается на содержание другого модуля.



Сложность программной системы

Автор: М. Холстед (1977)

Длина модуля:

$$N=n_1 \cdot \log_2(n_1)+n_2 \cdot \log_2(n_2)$$

n_1 – число различных операторов,

n_2 – число различных операндов.

Объем модуля:

$V = N \times \log_2(n_1+n_2)$

Сложность программной системы

Автор: Том МакКейб (1976)

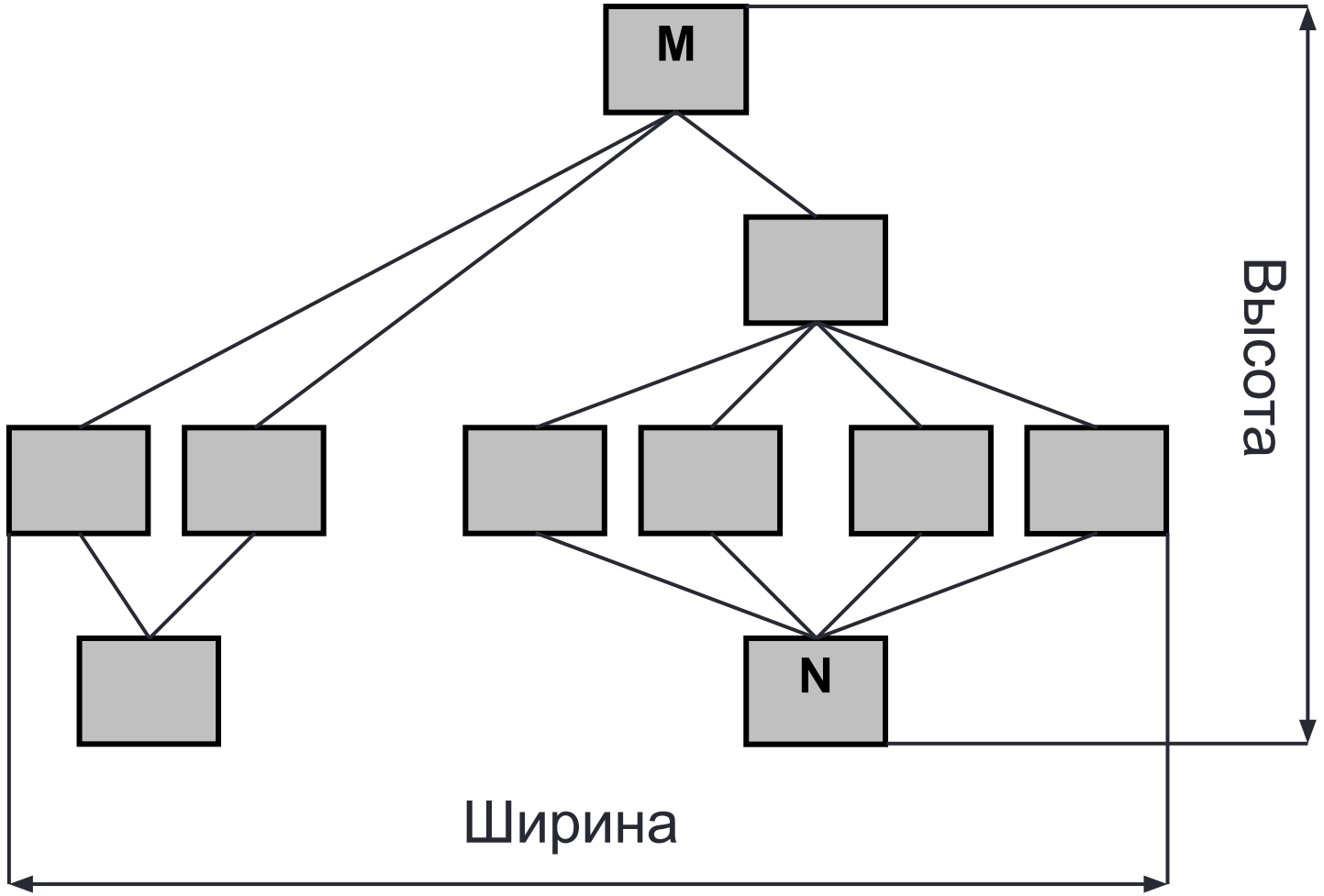
Метрика цикломатической сложности:

$$V(G) = E - N + 2$$

E – количество дуг,

N – количество вершин в управляющем графе программной системы.

Иерархическая структура программной системы



Характеристики ИС

Первичные:

- Количество вершин (модули)
- Количество ребер (связи между модулями)

Глобальные:

- Высота – количество уровней управления
- Ширина – максимальное из количеств модулей, размещенных на уровнях управления

Локальные:

- Коэффициент объединения по входу: Fan_in(i)
- Коэффициент разветвления по выходу: Fan_out(i)

Невязка проектной структуры

$$Nev = \frac{2 \times (e - n + 1)}{(n - 1) \times (n - 2)}$$

n – количество вершин, e – количество ребер проектной структуры.

Nev = 1 – полный граф.

Nev = 0 – дерево.

Использование коэффициентов объединения и разветвления

Полные значения коэффициентов:
Fan_in(i) = sfan_in(i) + ifan_in(i)
Fan_out(j) = sfan_out(j) + ifan_out(j)

Метрика общей сложности структуры:

$$S = \sum_{i=1}^{n} length(i) \times \left(Fan\_in(i) + Fan\_out(i) \right)^2$$

length(i) – оценка размера i-го модуля (в виде LOC- или FP-оценки)