

# **Методы структурного анализа и проектирования ПО**

**Структурный анализ** — один из формализованных методов анализа требований и проектирования ПО (автор Том Де Марко).

Основная задача – описание:

- а) функциональной структуры системы;
- б) последовательности выполняемых действий;
- в) передачи информации между функциональными процессами;
- г) отношений между данными.

Модели структурного анализа и проектирования:

1. Функциональная модель SADT (Structured Analysis and Design Technique);
2. DFD (Data Flow Diagrams) – диаграммы потоков данных;
3. Модель IDEF3;

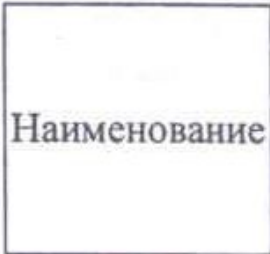



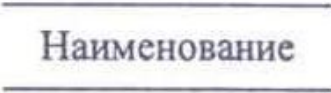
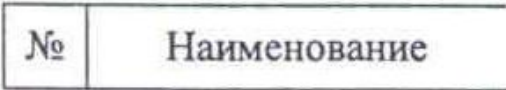


**DFD** (Data Flow Diagrams) – иерархия функциональных процессов, связанных потоками данных.

**Метод SADT** (IDEF0) – совокупность правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области (производимые действия и связи между ними).

**Модель IDEF3** – предназначена для моделирования последовательности выполняемых действий и взаимозависимости между ними, основа модели – сценарий процесса.

# Модель DFD

Для изображения диаграмм потоков данных (DFD) традиционно используют два вида нотаций: нотации Йордана и Гейна-Сарсона

Понятие	Нотация Йордана	Нотация Гейна-Сарсона
Внешняя сущность	 <p>Наименование</p>	 <p>Номер Наименование</p>
Система, подсистема или процесс	 <p>Наименование Номер</p>	 <p>Номер Наименование Механизм</p>
Накопитель данных	 <p>Наименование</p>	 <p>№ Наименование</p>
Поток	 <p>Наименование</p>	 <p>Наименование</p>

В основе модели лежат понятия внешней сущности, процесса, хранилища (накопителя) данных и потока данных.

**Внешняя сущность** - материальный объект или физическое лицо, выступающие в качестве источников или приемников информации, например, заказчики, персонал, поставщики, клиенты, банк и т. п.

**Процесс** - преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом.

Каждый процесс в системе имеет свой номер и связан с исполнителем, который осуществляет данное преобразование.

На верхних уровнях иерархии, когда процессы еще не определены, вместо понятия «процесс» используют понятия «система» и «подсистема», которые обозначают соответственно систему в целом или ее функционально законченную часть.

**Хранилище данных** - абстрактное устройство для хранения информации. Тип устройства и способы помещения, извлечения и хранения для такого устройства не детализируют. *Физически это может быть база данных, файл, таблица в оперативной памяти, картотека на бумаге и т. п.*

**Поток данных** — процесс передачи некоторой информации от источника к приемнику.

Физически процесс передачи информации может происходить по кабелям под управлением программы или программной системы или вручную при участии устройств или людей вне проектируемой системы.

# Пример потока данных (нотация Гейна-Сарсона)



Пример потока данных (нотация Гейна-Сарсона)

Над линией потока, направление которого обозначают стрелкой, указывают, какая конкретно информация в данном случае передается

Построение иерархии диаграмм потоков данных начинают с диаграммы особого вида - **контекстной диаграммы**, которая определяет наиболее общий вид системы.

На такой диаграмме показывают, как разрабатываемая система будет взаимодействовать с приемниками и источниками информации без указания исполнителей, т. е. описывают интерфейс между системой и внешним миром.



# Методология структурного моделирования SADT

Методология SADT (Structured Analysis and Design Technique) была создана и опробована на практике в период с 1969 по 1973 гг. Автором методологии SADT является Дуглас Росс.

Предназначения для моделирования систем на основе принципов структурного анализа. Методология предлагает графический язык проектирования систем, в котором сочетаются декомпозиция и иерархическое упорядочение и для обозначения составляющих системы используется графическая конструкция, называемая SA-блок.

# Предпосылки создания SADT

- **Возрастание сложности проектируемых систем.**
- **Необходимость формализации процесса разработки при создании крупномасштабных систем.**

**Процесс разработки систем был формально разбит на этапы:**

1. **Анализ** – определение того, что система будет делать
2. **Проектирование** – определение подсистем и их взаимодействие
3. **Реализация** – разработка подсистем по отдельности
4. **Объединение** – сборка подсистем в целое
5. **Тестирование** – проверка работы системы
6. **Установка** – введение системы в действие
7. **Функционирование** – использование системы

**Данная последовательность этапов разработки стала традиционной**

# Проблемы традиционного подхода

- Неучастие пользователя в процессе разработки.
- Сложности и отсутствие согласования результатов этапов разработки.
- Сложности в качественной и количественной оценке процесса разработки.
- Трудности в выявлении ошибок, допущенных на ранних этапах разработки системы.
- Неполнота функциональных спецификаций.
- Отсутствие согласованности между спецификациями и результатами проектирования.

# Результат применения традиционного подхода

- Выявление необходимости совершенствования методов анализа как ключа к созданию систем, эффективных по стоимости, производительности и надежности.
- Поиск методологии, применение которой способно было бы преодолеть выявленные недостатки традиционного подхода.
- Появление и совершенствование методологии структурного анализа SADT.

# Преимущества SADT

- Легко отражает такие системные характеристики как управление, обратная связь и исполнители, так как возникла на базе проектирования систем общего вида в отличие от структурных методов, «выросших» из проектирования программного обеспечения.
- Имеет развитые процедуры поддержки коллективной работы.
- Применяется на ранних стадиях создания системы, что позволяет избежать наиболее дорогостоящих ошибок.
- Успешно сочетается с другими структурными методами.

**Разработка и широкое успешное использование ее графического языка превратило SADT в методологию, способную значительно повысить качество продуктов, создаваемых на ранних этапах проектов.**

# Сущность структурного подхода

Система декомпозируется (разбивается) на функциональные подсистемы до нужной степени детализации.

Базовые принципы:

- принцип «разделяй и властвуй».
- принцип иерархического упорядочивания

# Использование SADT

Методология SADT может использоваться для моделирования широкого круга систем и определения требований и функций, а затем для разработки системы, которая удовлетворяет этим требованиям и реализует эти функции. Для уже существующих систем SADT может быть использована для анализа функций, выполняемых системой, а также для указания механизмов, посредством которых они осуществляются.

Методология SADT может быть направлена как для описания функций, выполняемых системой, так и на описание объектов, составляющих систему.

В первом случае методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями.

Во-втором случае методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для описания объектов, входящих в систему, их свойств и взаимосвязей между ними



# Методологии SADT

- IDEF0 (Icam Definition) модели и соответствующие функциональные диаграммы.
- DFD (Data Flow Diagrams) диаграммы потоков данных.
- ERD (Entity-Relationship Diagrams) диаграммы "сущность-связь«.

# Методология функционального моделирования IDEF0

Методология функционального моделирования IDEF0 (Icam DEFinition) была разработана на основе SADT и являлась основной частью программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой по инициативе ВВС США.

# Принципы функционального моделирования. Основные понятия.

- **Система** – совокупность взаимодействующих компонент и взаимосвязей между ними.
- **Моделирование** – процесс создания точного описания системы.
- **SADT модель** – полное, точное и адекватное описание системы, имеющее конкретное назначение, которое называется целью модели. SADT модель может быть сосредоточена либо на функциях системы (функциональная модель), либо на ее объектах (модель данных).
- **Цель модели** – получение ответов на некоторую совокупность вопросов относительно системы. Список вопросов сводится к одной-двум фразам, которые и формулируют цель.

- **Субъект моделирования** – сама система.
- **Границы системы** - точно определяют, что является и что не является субъектом моделирования, что входит в систему и что лежит за ее пределами. SADT-модель всегда имеет единственный субъект.
- **Точка зрения** – позиция, с которой наблюдается система и создается ее модель. Это позиция человека или объекта, в которую нужно встать, чтобы увидеть систему в действии.

В процессе моделирования **субъект** определяет, что включить в модель, а что исключить из нее. **Точка зрения** диктует выбор нужной информации о субъекте и форму ее подачи. **Цель** становится критерием окончания моделирования.

# Концепции IDEF0

- IDEF0-Модель отображает систему в виде иерархии диаграмм.
- Каждая диаграмма содержит блоки и дуги.
- Диаграмма в виде блока отображает функцию. Блоки имеют доминирование;
- Интерфейсы входа/выхода представляются дугами, входящими в блок и выходящими из него;
- Интерфейсные дуги показывают взаимодействие блоков друг с другом;
- Интерфейсные дуги выражают "ограничения", определяющие, когда и каким образом функции выполняются и управляются.

# Правила IDEFO

- Диаграмма, лежащая на вершине иерархии, называется контекстной.  
На этой диаграмме вся система представляется в виде единого функционального блока.
- Следующей в иерархии является диаграмма декомпозиции контекстной диаграммы. На ней функциональный блок контекстной диаграммы декомпозируется на составляющие его функциональные блоки. Каждый из этих блоков может иметь свою диаграмму декомпозиции.
- Количество блоков на каждом уровне декомпозиции ограничено (может быть от 3 до 6);
- Диаграммы связаны по номерам блоков;
- Метки и наименования уникальны;
- Входы и управления разделены по роли данных;
- Исключено влияние организационной структуры на функциональную модель.

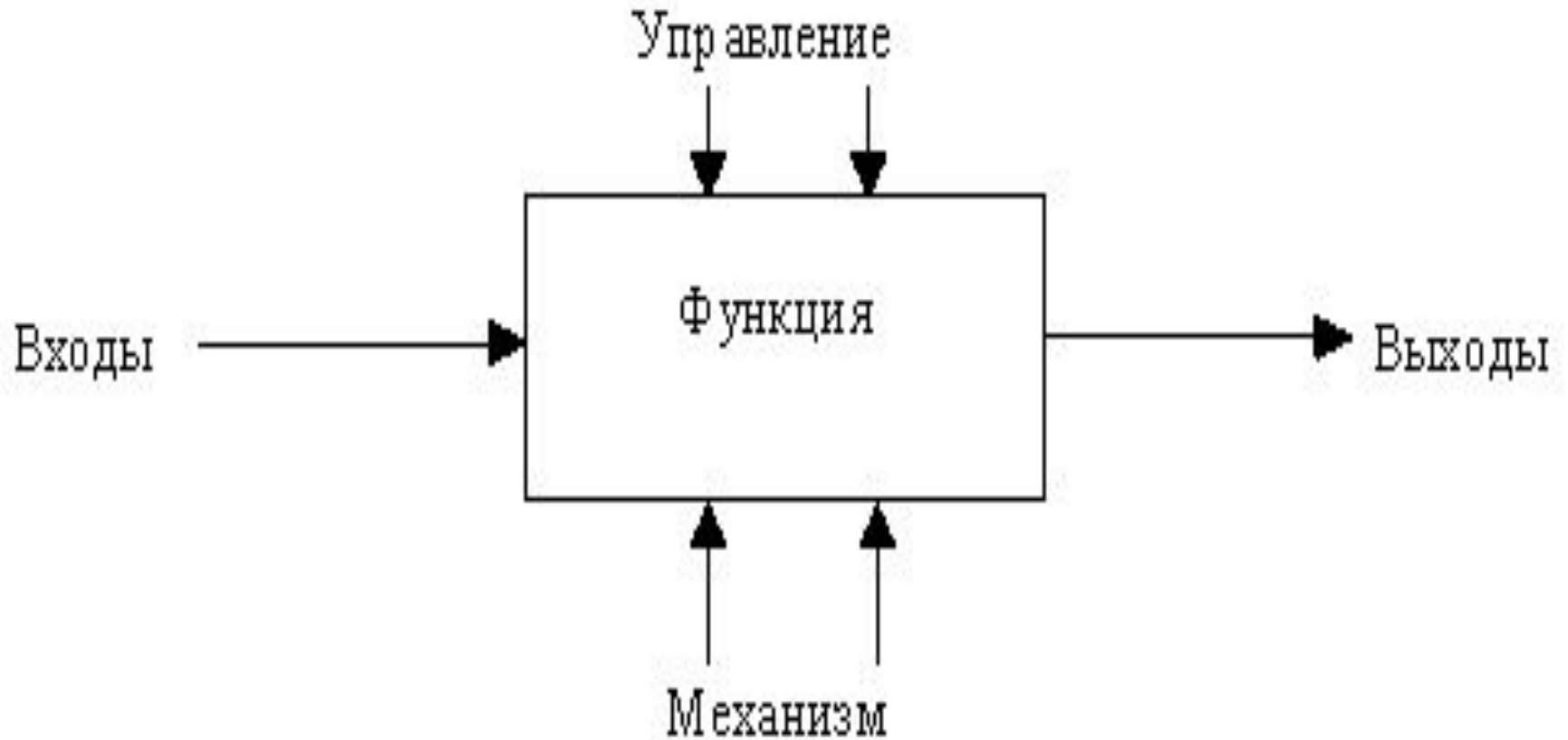
# Состав функциональной модели IDEFO

Функциональная модель состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга.

Диаграммы - главные компоненты модели, все функции системы и интерфейсы на них представлены как блоки и дуги.

Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, информация, которая подвергается обработке, - слева, результаты выхода - справа стороны. Механизм (человек или автоматизированная система), который осуществляет операцию, входит в блок снизу.

# Функциональный блок и интерфейсные дуги





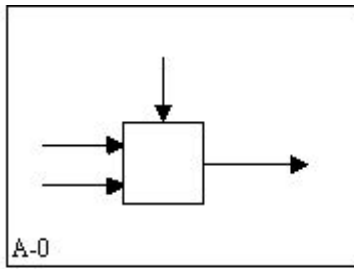
# Иерархия диаграмм

На вершине иерархии находится диаграмма, на которой система представляется в виде единого блока и дуг, изображающих интерфейсы с функциями вне системы. Контекстная диаграмма.

Уровнем ниже находится диаграмма, на которой блок, представляющий систему в целом, детализируется с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции.

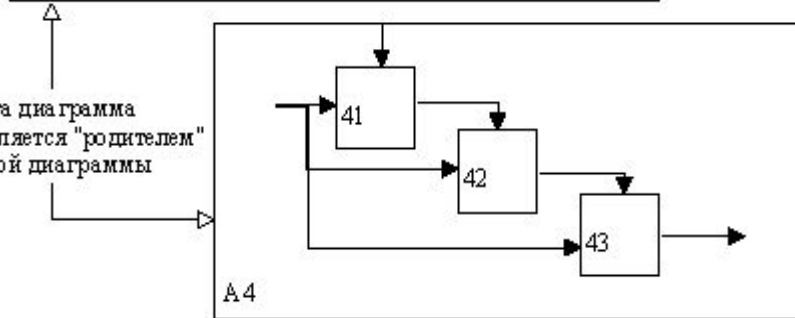
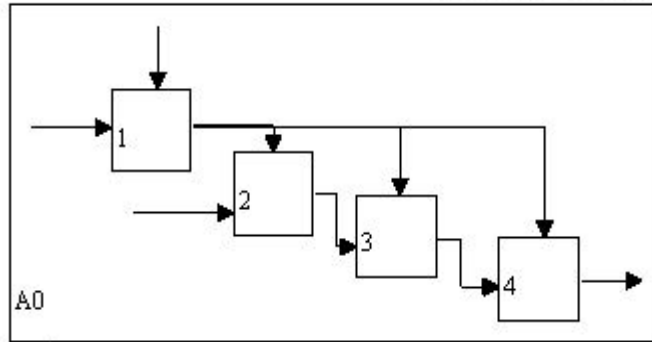
# Правила декомпозиции функциональных блоков

- Каждая функция может быть декомпозирована на подфункции;
- Подфункция может содержать только те элементы, которые входят в исходную функцию;
- Родительский блок и его интерфейсы обеспечивают контекст. Из модели нельзя выбросить какие-либо элементы или добавить их;
- Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются точно теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее.

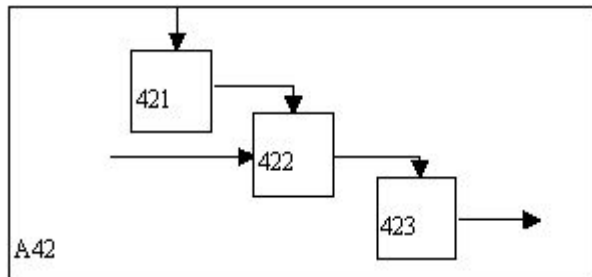


Более общее представление

Более детальное представление



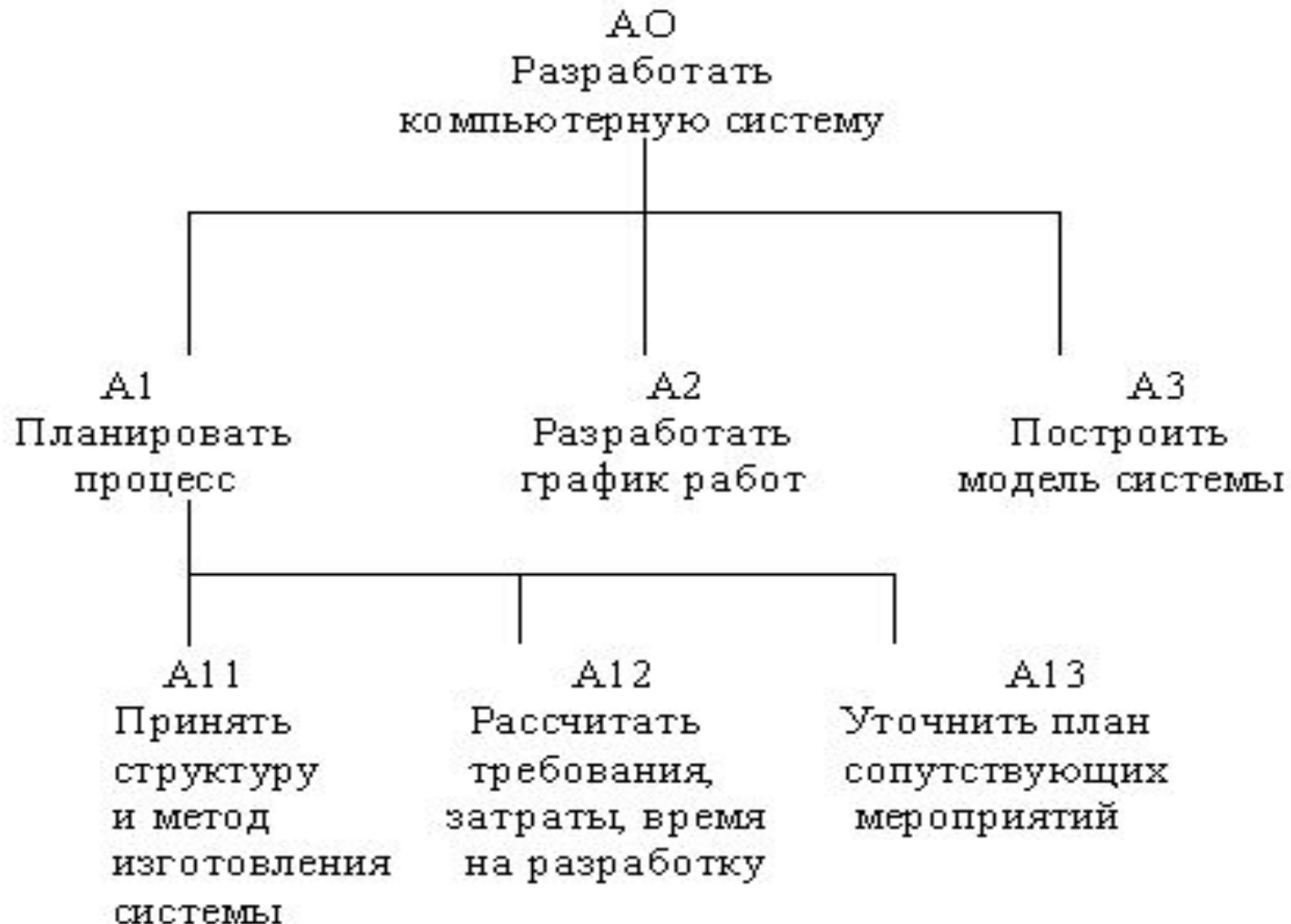
Эта диаграмма является "родителем" этой диаграммы



# Структура IDEF0-модели. Декомпозиция диаграмм

- Каждый блок на диаграмме имеет свой номер.
- Для того, чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм. Например, A21 является диаграммой, которая детализирует блок 1 на диаграмме A2. Аналогично, A2 детализирует блок 2 на диаграмме A0, которая является самой верхней диаграммой модели.

# Иерархия диаграмм



# Что отражает модель IDEF3?

В общем случае, процесс – это упорядоченная последовательность действий.

Следовательно, процессная модель IDEF3 *позволяет:*

- Отобразить последовательность процессов
- Показать логику взаимодействия элементов системы.

**Цель IDEF3** - дать возможность аналитикам описать ситуацию, когда процессы выполняются в определенной последовательности, а также объекты, участвующие совместно в одном процессе.

# Основные компоненты IDEF3-модели


Основными элементами IDEF3-модели являются:

- 1) единицы работ;*
- 2) связи;*
- 3) перекрестки;*
- 4) объекты ссылок.*


# Единицы работ

- Единица работ (UOW, Unit of Work) является центральным компонентом модели.

Номер работы является уникальным, присваивается при ее создании и не меняется никогда



Словосочетание с отглагольным существительным, изображающим действие (выполнение, изготовление,...)  
*Или*  
Инфинитив глагола (изготовить продукцию)



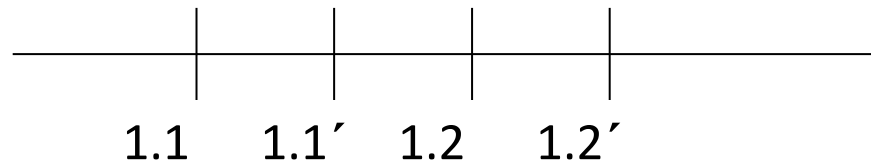


# СВЯЗИ

- Связи показывают *взаимоотношения* работ.
- Связи *однонаправлены* и могут быть направлены куда угодно
- Обычно диаграммы рисуют таким образом, чтобы связи были направлены слева направо
- Различают *3 типа* связей:
  - Старшая стрелка
  - Стрелка отношений
  - Поток объектов.

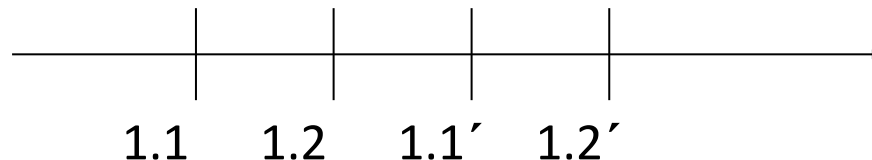
# Связь «старшая стрелка»

- Связь типа «временное предшествование» - *Precedence*
- Соединяет единицы работ
- Показывает, что работа-источник должна быть закончена прежде, чем начнется работа-цель



# Стрелка отношений

- Связь типа нечеткое отношение - *Relational*
- Изображается в виде пунктирной линии, используется для изображения связи между единицами работ, а также между единицами работ и объектами ссылок






# Поток объектов

- Стрелка, изображающая поток объектов - *Object Flow*
- Применяется для описания того факта, что объект используется в двух и более единицах работ, например, когда объект порождается в одной работе и используется в другой



# Перекрестки (соединения)

- Используются для отображения **логики взаимодействия** стрелок при их *слиянии* или *разветвлении*, для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы.
- Различают перекрестки для **слияния** и **разветвления** стрелок.
- Перекрестки не могут быть одновременно использованы для слияния и разветвления стрелок.
- Все перекрестки на диаграммах нумеруются, каждый номер имеет префикс **J**.
- В отличие от других методологий (IDEF0, DFD) стрелки могут сливаться или разветвляться только через перекрестки.

# Типы перекрестков

Обозначение	Наименование	Смысл в случае слияния стрелок	Смысл в случае разветвления стрелок
	<i>Асинхронное «И»</i>	Все предшествующие процессы должны быть завершены	Все последующие процессы должны быть запущены
	<i>Синхронное «И»</i>	Все предшествующие процессы должны быть завершены одновременно	Все последующие процессы запускаются одновременно
	<i>Асинхронное «ИЛИ»</i>	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены

# Типы перекрестков

Обозначение	Наименование	Смысл в случае слияния стрелок	Смысл в случае разветвления стрелок
	<i>Синхронное «ИЛИ»</i>	Один или несколько предшествующих процессов должны быть завершены одновременно	Один или несколько следующих процессов должны быть запущены одновременно
	<i>Эксклюзивное (исключающее) «ИЛИ»</i>	Только один предшествующий процесс должен быть завершен	Только один следующий процесс запускается

# Правила создания перекрестков

1. Каждому *перекрестку для слияния* должен предшествовать *перекресток для разветвления*.
2. **Перекресток для слияния «И»** не может следовать за перекрестком для разветвления типа **синхронного** или **асинхронного «ИЛИ»**



# Правила создания перекрестков

3. Перекресток для слияния «И» не может следовать за перекрестком типа исключительного «ИЛИ»

# Правила создания перекрестков

4. Перекресток для слияния типа **исключительного «ИЛИ»** не может следовать за перекрестком для разветвления типа **«И»**

5. Перекресток, имеющий одну стрелку на одной стороне, должен иметь более одной стрелки на другой.

# Примеры

# Примеры

# Примеры

# Комбинации перекрестков

- Перекрестки могут комбинироваться для создания сложных соединений

# Объект ссылок

- выражает **идею, концепцию данных**, которые нельзя связать со стрелкой, перекрестком, работой
- используется при построении диаграммы для привлечения внимания пользователя к каким-либо важным аспектам модели

# Объект ссылок

- *Официальная спецификация IDEF3 различает 3 стиля объектов ссылок – безусловные (unconditional), синхронные (synchronous), асинхронные (asynchronous).*
- VRWin поддерживает только *безусловные* объекты ссылок.



# Типы объектов ссылок

<b>Тип объекта ссылок</b>	<b>Назначение</b>
1. <b>Object</b>	Используется для описания того, что в действии принимает участие какой-либо заслуживающий отдельного внимания объект
2. Ссылка <b>GOTO</b>	Используется для реализации цикличности выполнения действий. Этот объект также может относиться к перекрестку
3. Единица действий <b>UOB</b> (Unit of Behavior)	Используется для многократного отображения на диаграмме одного и того же действия, но без цикла

# Типы объектов ссылок

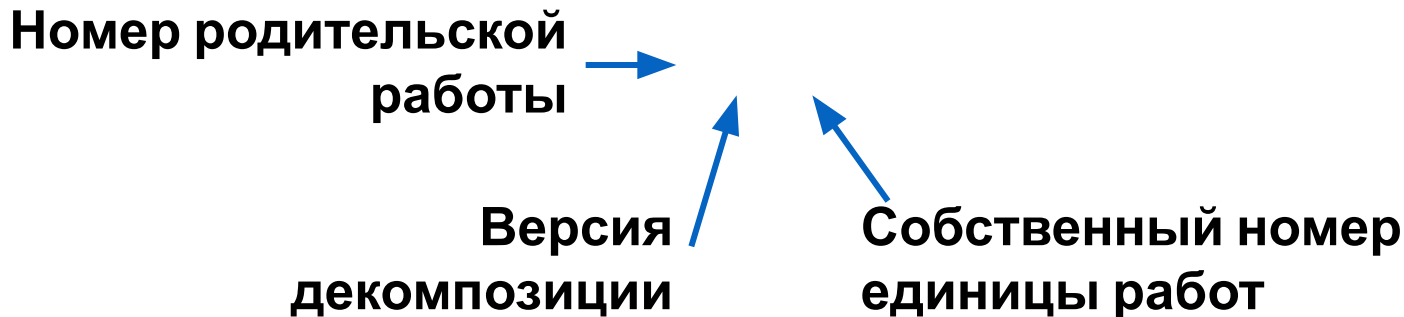
<b>Тип объекта ссылки</b>	<b>Назначение</b>
4. <i>Заметка</i> ( <b>Note</b> )	Используется для документирования какой-либо важной информации общего характера, относящейся к изображаемому на диаграммах. Служит альтернативой методу помещения текстовых заметок непосредственно на диаграммах
5. <i>Уточнение</i> <i>Elaboration</i> ( <b>ELAB</b> )	Для уточнения или более подробного описания изображаемого на диаграмме. Обычно используется для детального описания разветвления или слияния стрелок на перекрестках

# Декомпозиция работ в IDEF3

- В IDEF3 декомпозиция используется для *детализации* работ.
- Методология IDEF3 позволяет декомпонировать работу **множественно**, т.е. работа может иметь множество дочерних работ.
- Это позволяет в одной модели описать **альтернативные потоки**.
- Возможность множественной декомпозиции предъявляет дополнительные требования к нумерации работ

# Нумерация работ в IDEF3

- Номер работы состоит из *номера родительской работы*, *версии декомпозиции* и *собственного номера* работы на текущей диаграмме



# Структура множественной декомпозиции работ

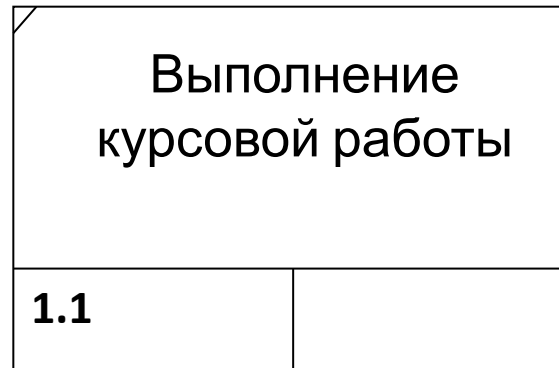
Первая  
декомпозиция  
работы 1.2

Вторая  
декомпозиция  
работы 1.2



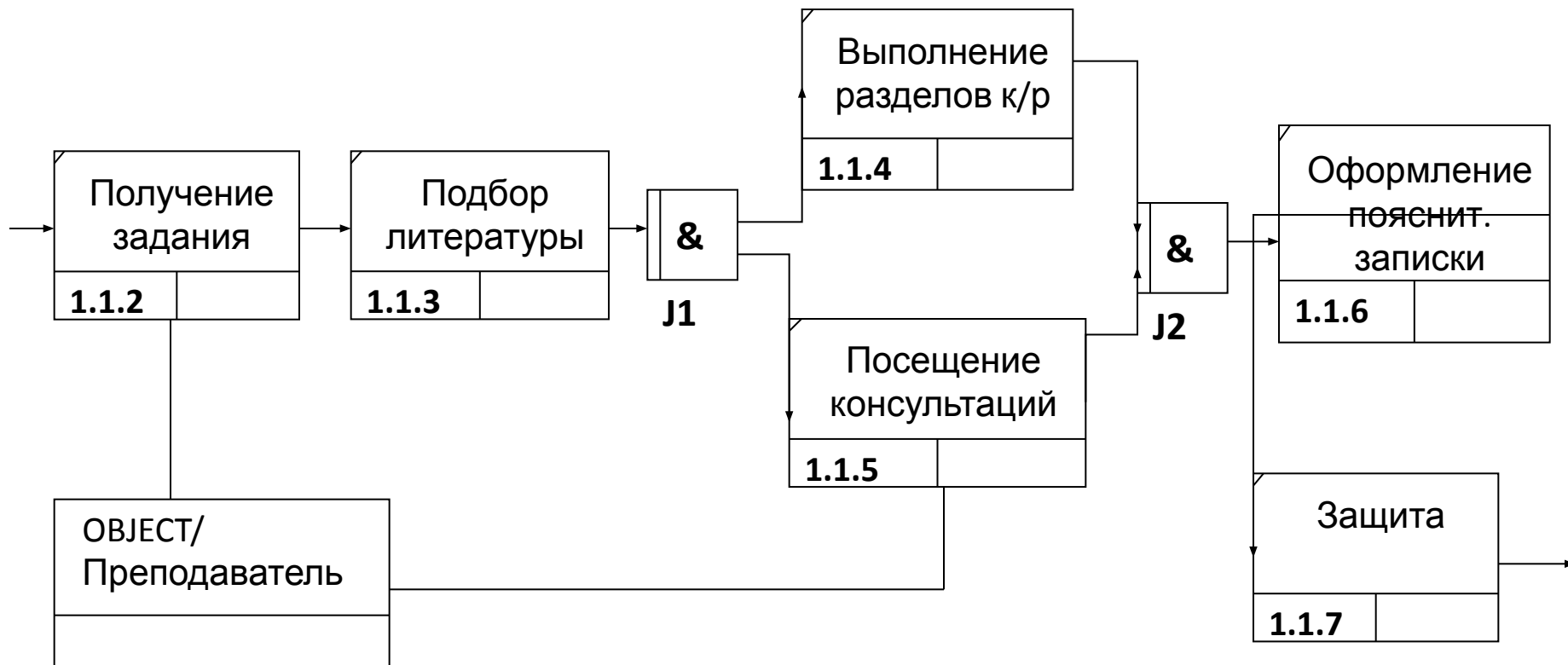
# Пример построения модели IDEF3

- Рассмотрим на примере построения динамической модели процесса «**Выполнение курсовой работы**»
- Начнем с построения контекстной диаграммы



# Пример построения модели IDEF3

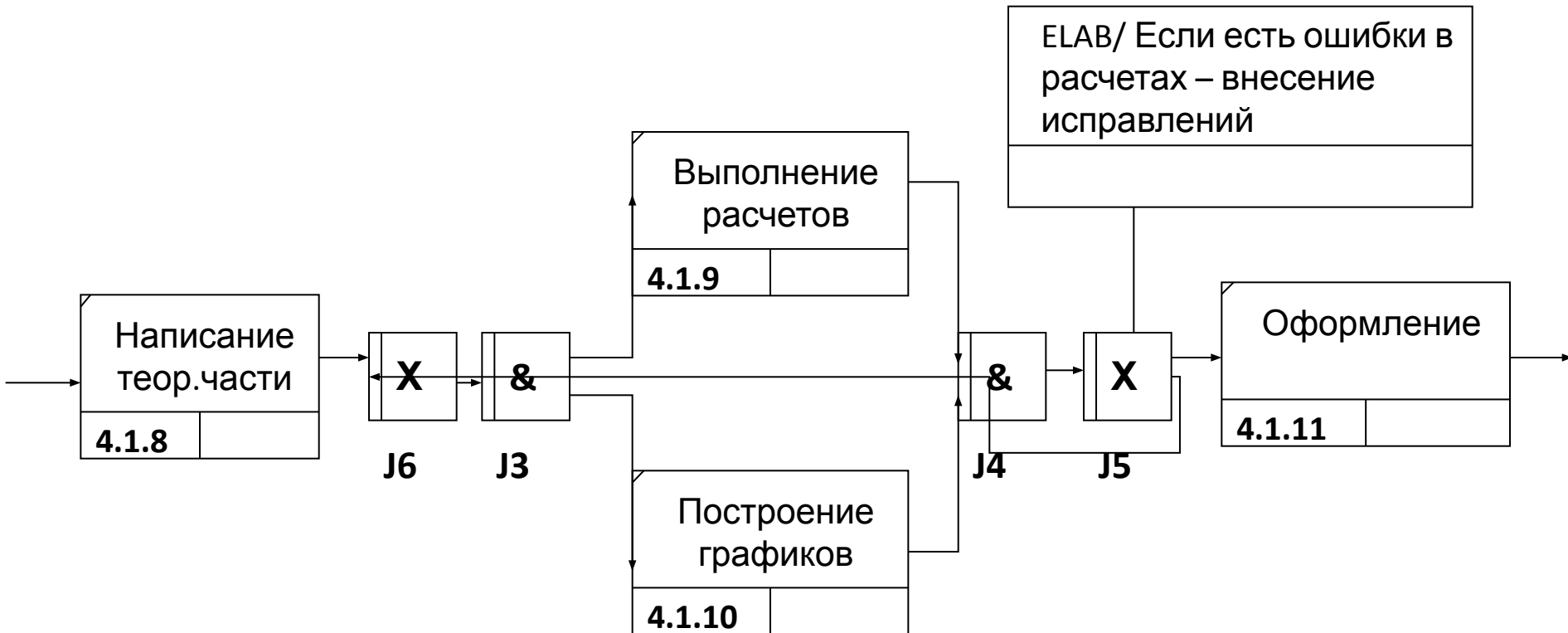
Выполним декомпозицию контекстной диаграммы:



**Примечание:** Обратите внимание на нумерацию единиц работ. Родительской является работа с собственным номером **1**. Она декомпозируется первый раз, следовательно, версия декомпозиции = **1**, далее следует собственный номер единицы работ в рамках модели (**2-7**).

# Пример построения модели IDEF3

Выполним декомпозицию UOW №4 – «Выполнение разделов к/р»





# Пример построения модели IDEF3

- Продекомпозируем повторно контекстную диаграмму (в виде [сценария](#) IDEF3 для выполнения курсовой работы по «Информатике и программированию»)

