Программная инженерия

начало 6 Раздела. Подходы к разработке ПС: 6.1. Классический подход

Расписание:

11.01 Пн	14:30-18:25	Г-2082	Лекция
12.01 Вт	14:30-18:25	Г-2082	Лекция
13.01 Ср	14:30-18:25	Г-2023	Лабораторная работа
14.01 Чт	14:30-18:25	Г-2023	Лабораторная работа
21.01 Чт	18:30-21:40	Г-2023	Экзамен

6. Подходы к разработке ПС

Два подхода к разработке ПС

Структурный (функционально-модульный) классический

(основан на принципе функциональной декомпозиция)

Структура системы описывается в терминах иерархии ее функций и передачи информации между отдельными функциональными элементами.

В структурных технологиях функциональная и информационная модели строятся отдельно, чаще всего в виде диаграмм потоков данных и диаграмм "сущность-связь"

В области создания бизнес-систем лидируют структурные технологии, т. к. они максимально приспособлены для взаимодействия с заказчиками и пользователями, не являющимися специалистами в области информационных технологий.

Объект-ориентированный

(объектная декомпозиция)

рассматривают информацию неотъемлемо от процедур ее обработки. Модели объектно-ориентированных технологий описывают структуру, поведение и реализацию систем в терминах классов объектов.

Объектно-ориентированные технологии доминируют в области создания операционных систем, средств разработки и исполнения приложений, систем реального времени. Концепция объекта помогает бороться с быстро растущей сложностью систем. Кроме того, взаимодействующие электронные устройства, как и элементы программ, естественно представляются объектами.

Задача выбора подхода

- Важно понимать, что **выбор подхода определяется целями проекта и в значительной мере влияет на весь его дальнейший ход.**
- Между сторонниками структурного(СП) и объектно-ориентированного подходов(ООП) в настоящее время ведутся ожесточенные споры. При этом не существует решающих аргументов, доказывающих несостоятельность того или иного из методов, так как **каждый из них имеет свои преимущества и недостатки**.

Сравнение подходов

СП

- **Достоинства**

Традиционность (проверенность временем)

Наглядность и однозначность

Много CASE-средств

- **Недостатки**

Сложность исправления ошибок проектирования

Сложность моделирования динамики

Строгий регламент методологии

ООП

- **Достоинства**

Автоматическое кодирование

Открытость UML для дополнения

Простота масштабирования

- **Недостатки**

Возможность неоднозначной трактовки диаграмм

Выше требования к квалификации – трудности для экспертов

Отличия подходов

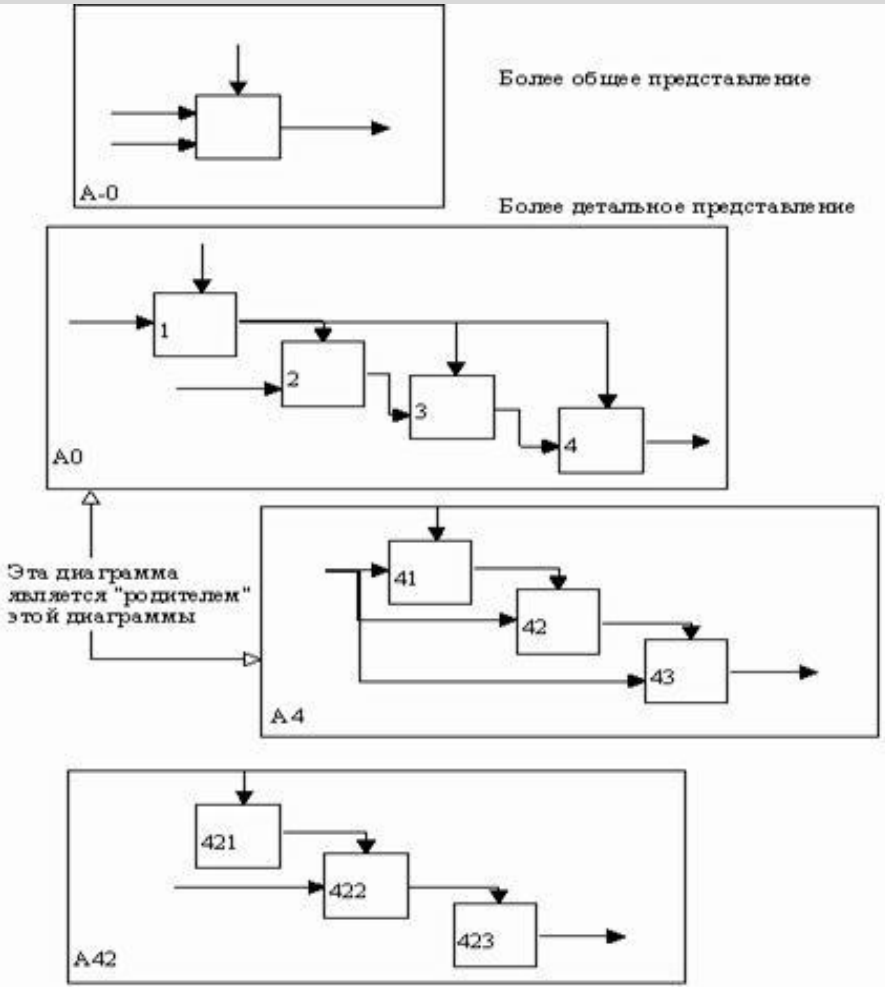
Первое отличие(определяющее) подходов друг от друга - в принципах декомпозиции(разбиения) и структурной организации элементов (компонентов, модулей) системы.

Согласно этим принципам система представляет собой структуру, состоящую из четко выраженных модулей, связанных между собой определенными отношениями.

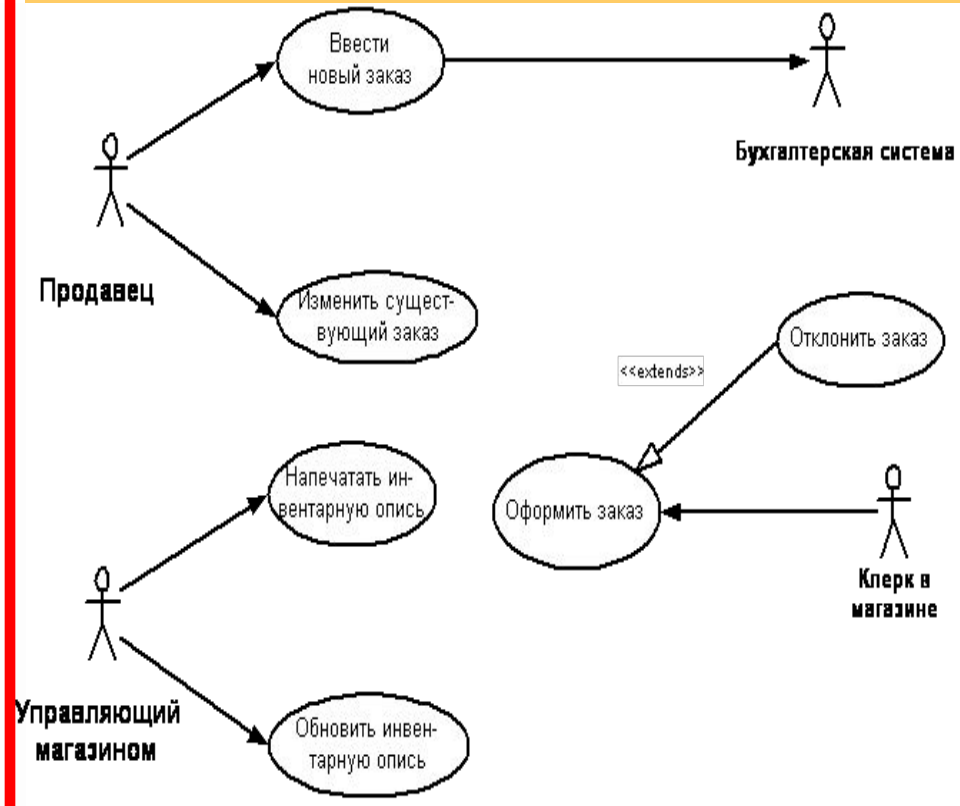
- При использовании **СП** (1-ый вид декомпозиции) - выполняется **функциональная** (процедурная, алгоритмическая) **декомпозиция** системы, т. е. **она представляется в виде иерархии (дерева) взаимосвязанных функций.**
- При **ООП** (2-ой вид декомпозиции). Система разбивается на **набор объектов**, соответствующих объектам реального мира, **взаимодействующих между собой путем посылки сообщений**.

Отличие подходов – 1 - принципы декомпозиции:

СП (функция)



ООП (объект)



Примеры используемых диаграмм для двух подходов

Отличия подходов

Второе отличие:

- В ООП объединение в объекте как атрибутивных данных (характеристики, свойства), так и поведения (функции, методы). Каждый объект системы обладает своим собственным поведением, моделирующим поведение объекта реального мира.
- В СП функции и данные хранятся (существуют) отдельно.

Отличия подходов

Третье отличие - в структурной организации внутри модулей системы.

- В СП модуль состоит из функций, иерархически связанных между собой отношением композиции (англ. “part-of” – “часть-целое”), т. е. функция состоит из подфункций, подфункция из подподфункций и т.д.
- В ООП иерархия выстраивается с использованием двух отношений:

  - композиции и
  - наследования (англ. “is-a” – “это есть”).

При этом в ООП «объект-часть» может включаться сразу в несколько «объектов-целое».

Таким образом, модуль

в СП представляется в виде дерева,

в ООП – в виде орграфа, т. е. с помощью более общей структуры.

Орграф – ориентированный граф

6.1. Классический структурный подход

Сущность СП к разработке ИС:

- заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее.
- Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны.
- При разработке системы **«снизу-вверх»** от отдельных задач ко всей системе целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов. Этой проблемы использование СП позволяет избежать.

Сущность СП к разработке ИС

- Кроме того СП дает возможность рассмотреть логику процессов компании и приблизить организацию бизнеса к оптимуму.
- Т. Е. использование данного подхода наиболее эффективно в том случае, когда речь идет, прежде всего, **об оптимизации бизнеса, а не просто об его автоматизации**.

Базовые принципы структурного подхода

Все наиболее распространенные методологии структурного подхода базируются на ряде общих принципов.

В качестве **двух базовых принципов** используются следующие:

- **«разделяй и властвуй»** – решение сложных проблем производится путем **их разбиения на множество меньших независимых задач**, легких для понимания и решения;
- **иерархического упорядочивания** – организация составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

Принципы структурного подхода:

- **абстрагирования** – выделение существенных аспектов системы и отвлечения от несущественных;
- **формализации** – необходимость осуществления строгого методического подхода к решению проблемы;
- **непротиворечивости** – обоснованность и согласованность элементов;
- **структурирования данных** – данные должны быть структурированы и иерархически организованы.

Достоинства структурного подхода

- возможность проведения **глубокого анализа бизнес-процессов, выявления узких мест:** комплексное применение позволяет выявить все возможные рассогласования и неточности;
- **применение универсальных графических языков моделирования** IDEF0, IDEF3 и DFD обеспечивает логическую целостность и полноту описания, необходимую для достижения точных и непротиворечивых результатов;
- **проверенность временем и широкое распространение** среди аналитиков и разработчиков.

Недостатки структурного подхода

- **низкая наглядность для неподготовленных пользователей модели**: при увеличении количества уровней представления, анализа и модификации моделей становится затруднительными;
- **сложность восприятия** иерархически упорядоченной информации;
- **необходимость следования жесткой (не всегда необходимой) структуре**.

применение структурного подхода рекомендуется для правильного, точного и полного определения требований на начальных этапах разработки ПС.

Обзор методологий структурного анализа и проектирования

Методологии структурного анализа и проектирования(СА и П) ИС

- Методологии СА и П ИС появились позже фактического использования этих принципов на практике (**структурного программирования**).
- В конце 60-х гг. XX в. стали появляться и применяться первые методологии, ориентированные на СП.
- В СА и П используются в основном две группы средств, иллюстрирующих:

  - функции, выполняемые системой и
  - отношения между данными.
- Большинство методологий СА и П основано на представлении моделей разрабатываемых систем в виде диаграмм.

Основные из них представлены в таблице ниже.

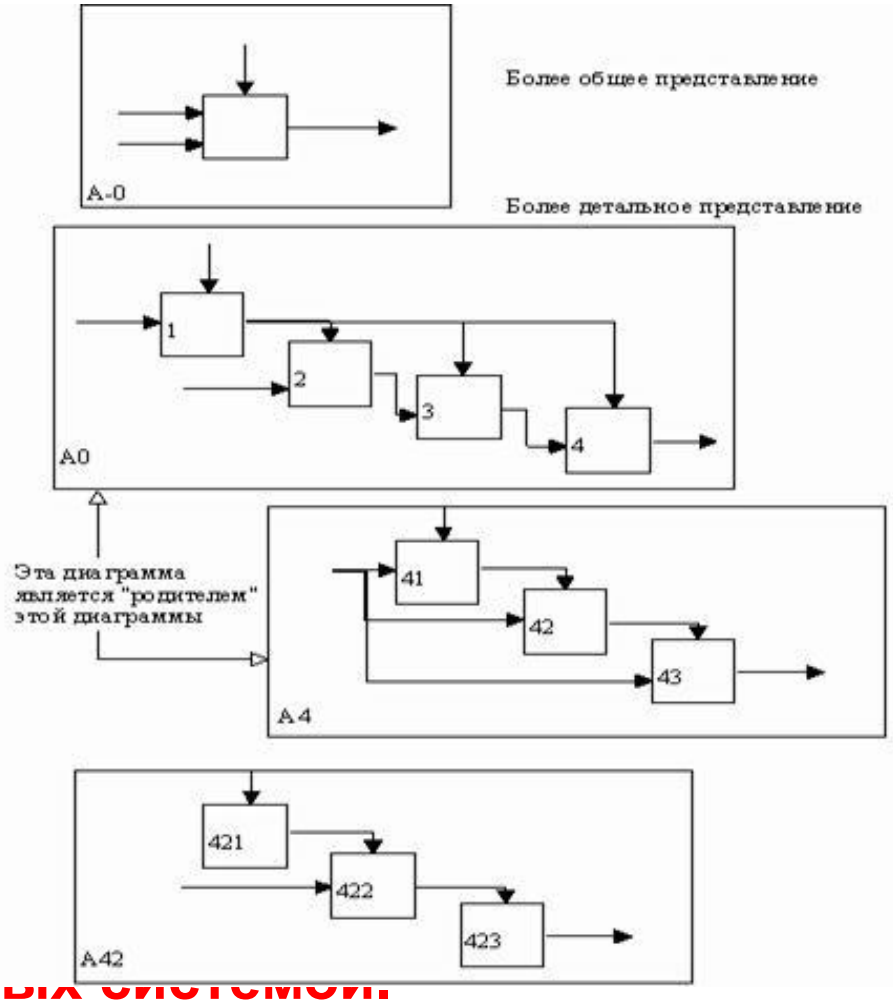
Идея структурного программирования заключается в том, что структура программы должна отражать структуру решаемой задачи, чтобы алгоритм решения был ясно виден из исходного текста.
В разработке идей структурного программирования (IBM) участвовали известные ученые Э. Дейкстра, Х. Милс, Э. Кнут, С. Хоор

Наиболее распространенные виды диаграмм (методологий СА и П):

Методология	Тип разрабатываемой модели	Примечание
SADT (Structured Analysis and Design Technique, методология структурного анализа и проектирования)	Функциональная	Разрабатывалась в 1969 – 1973 г. Автор Дуглас Росс На ее основе разработана методология IDEF0
DFD (Data Flow Diagrams, диаграммы потоков данных)	Смешанная (функциональная, информационная, компонентная)	Используются две нотации, соответствующие методам Йордона-ДеМарко и Гейна-Сарсона, различающиеся графическим изображением символов
ERD (Entity-Relationship Diagrams, диаграммы «сущность-связь»)	Информационная	Модель «сущность-связь» была предложена в 1976 г. Питером Пин-Шен Ченом
STD (State Transition Diagrams, диаграммы изменения состояний)	Поведенческая (еще наз-ся динамическая)	Если много состояний + матрицы переходов состояний
Flowcharts (блок-схемы)	Смешанная (поведенческая, информационная, компонентная)	Построение блок-схем алгоритмов регламентируется ГОСТ 19.701-90 (ИСО 5807-85)

SADT:

- Для ***новых систем*** SADT (IDEF0) применяется **для определения требований (функций) для разработки системы**, реализующей выделенные функции.
- Для ***уже существующих*** методология IDEF0 может быть использована **для анализа функций, выполняемых системой.**



Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм.

Вершина - самое общее описание системы.

DFD

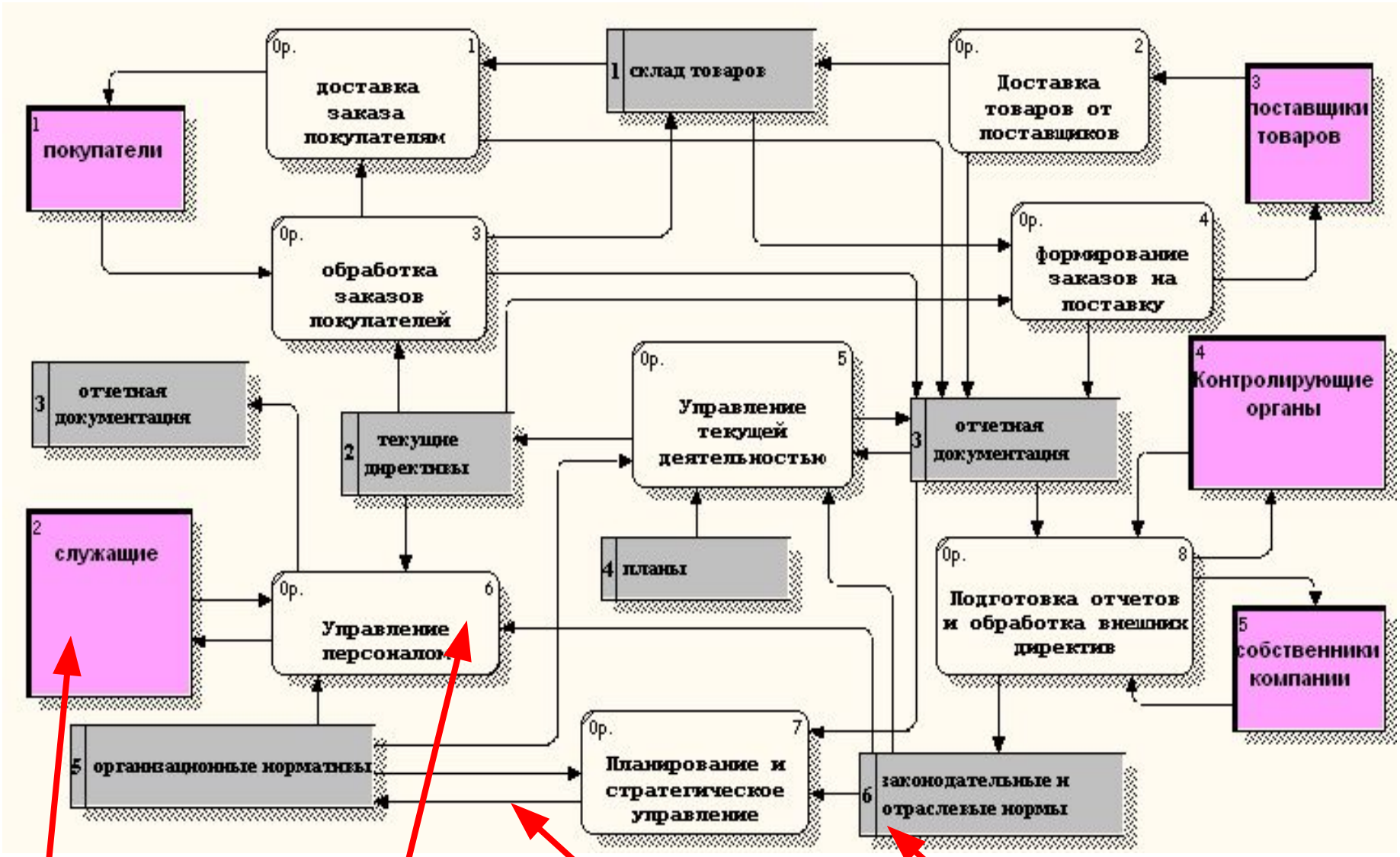
Диаграммы DFD обычно строятся для **наглядного изображения текущей работы системы документооборота** организации. Как правило, диаграммы DFD используют **в качестве дополнения модели бизнес-процессов**, выполненной в IDEF0.

Контекстная DFD-диа

Элемент DFD-диаграммы, построенной в нотации Обозначение процесса в нотации Гейна — Сарсона

http://vernikov.ru/component/k2/item/212.html

Элементы DFD диаграммы



Внешние сущност

Процесс

Поток данных

Хранилище (накопитель

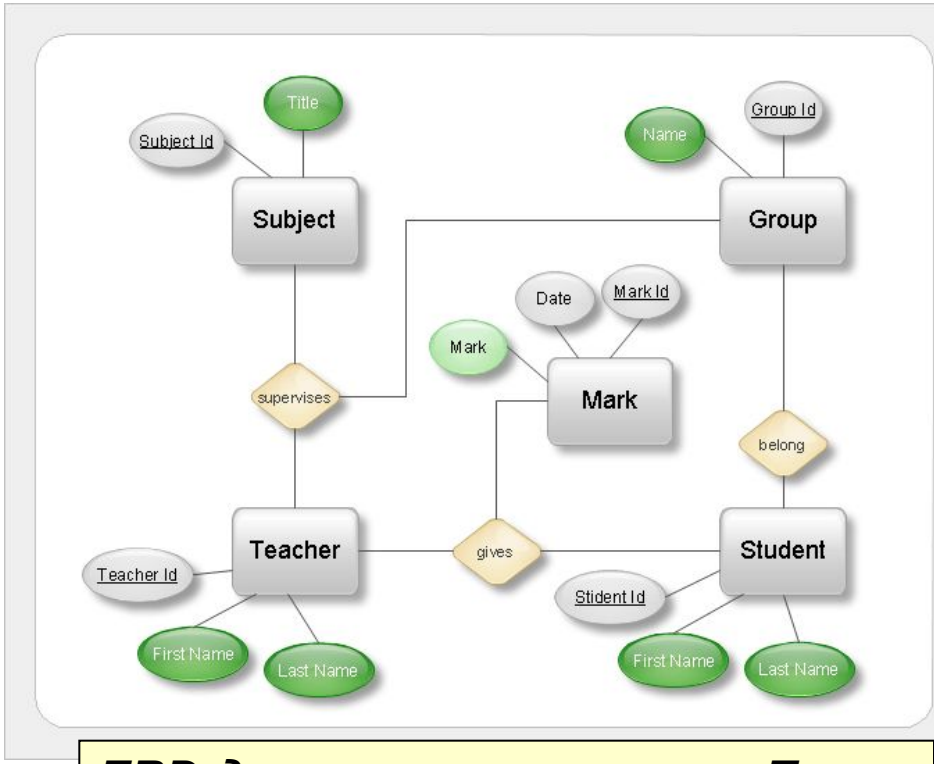
ERD

Для проектирования БД в н.в. используются CASE-средства ориентированные на использование *ERD (диаграммы «сущность–связь»)*.

С их помощью определяются:

- важные для предметной области объекты (**сущности**)
- отношения друг с другом (**связи**) и
- их свойства (**атрибуты**).

Средства проектирования ERD в основном ориентированы на реляционные БД (РБД), и если существует необходимость проектирования другой системы, скажем объектно-ориентированной, то лучше избрать др. методы проектирования.



ERD диаграмма в нотации П. Чена

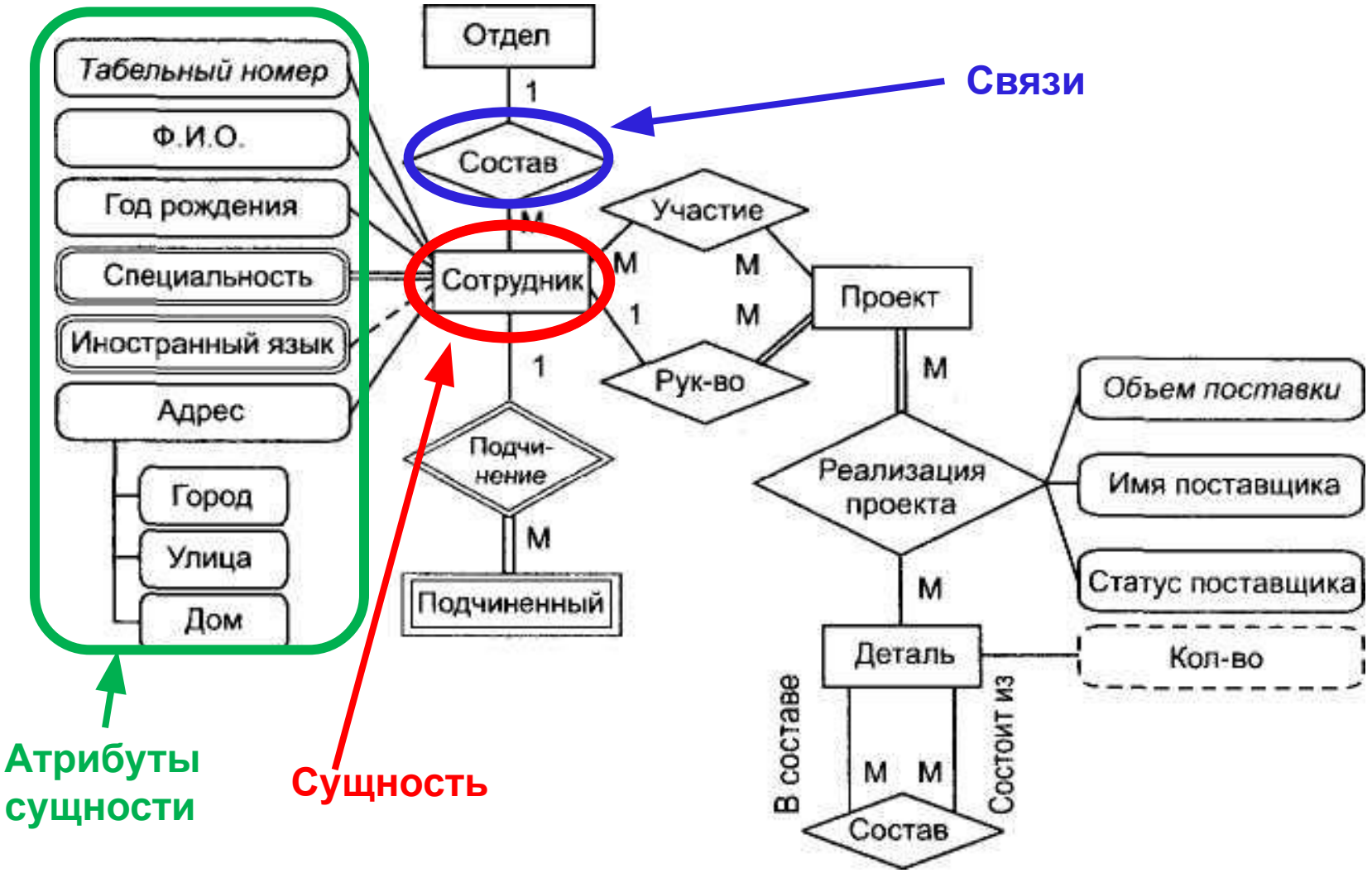
Элементы ERD-диаграммы:

- сущности

- связи

- атрибуты сущностей

Пример ER диаграммы:



STD

- предназначены для моделирования и документирования аспектов **систем, зависящих от времени или реакции на событие**.
- позволяют осуществлять декомпозицию управляющих процессов и описывают отношения между входными и выходными управляющими потоками для управляющего процесса-предка.
- **Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний**. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены.
  Элементы STD (диаграммы изменения состояний)
- **Узлы** – соот-ют состояниям динамической системы. Состояния бывают промежуточные и терминальные (одно начальное и одно или несколько конечных). Имя состояния должно отражать реальную ситуацию, в которой находится система, н-р, Нагревание, Ускорение и пр.
- **Дуги** соот-ют переходу системы из одного состояния в другое

Подробнее про STD

Правила построения STD:

- Строить STD на как можно более высоком уровне детализации DFD.

- Строить как можно более простые STD.

- Использовать те же принципы именования состояний, событий и действий, что и при именовании процессов и потоков. (DFD)

Применяются два способа построения STD:

1) идентификация всех возможных состояний и дальнейшее исследование всех небессмысленных связей (переходов) между ними.
2) сначала строится начальное состояние, затем следующие за ним и т. д.

Результат в обоих случаях – предварительная STD, для которой затем осуществляется контроль состоятельности.

STD

Контроль состоятельности, заключается в ответе на следующие вопросы:

Все ли состояния определены и имеют уникальное имя?

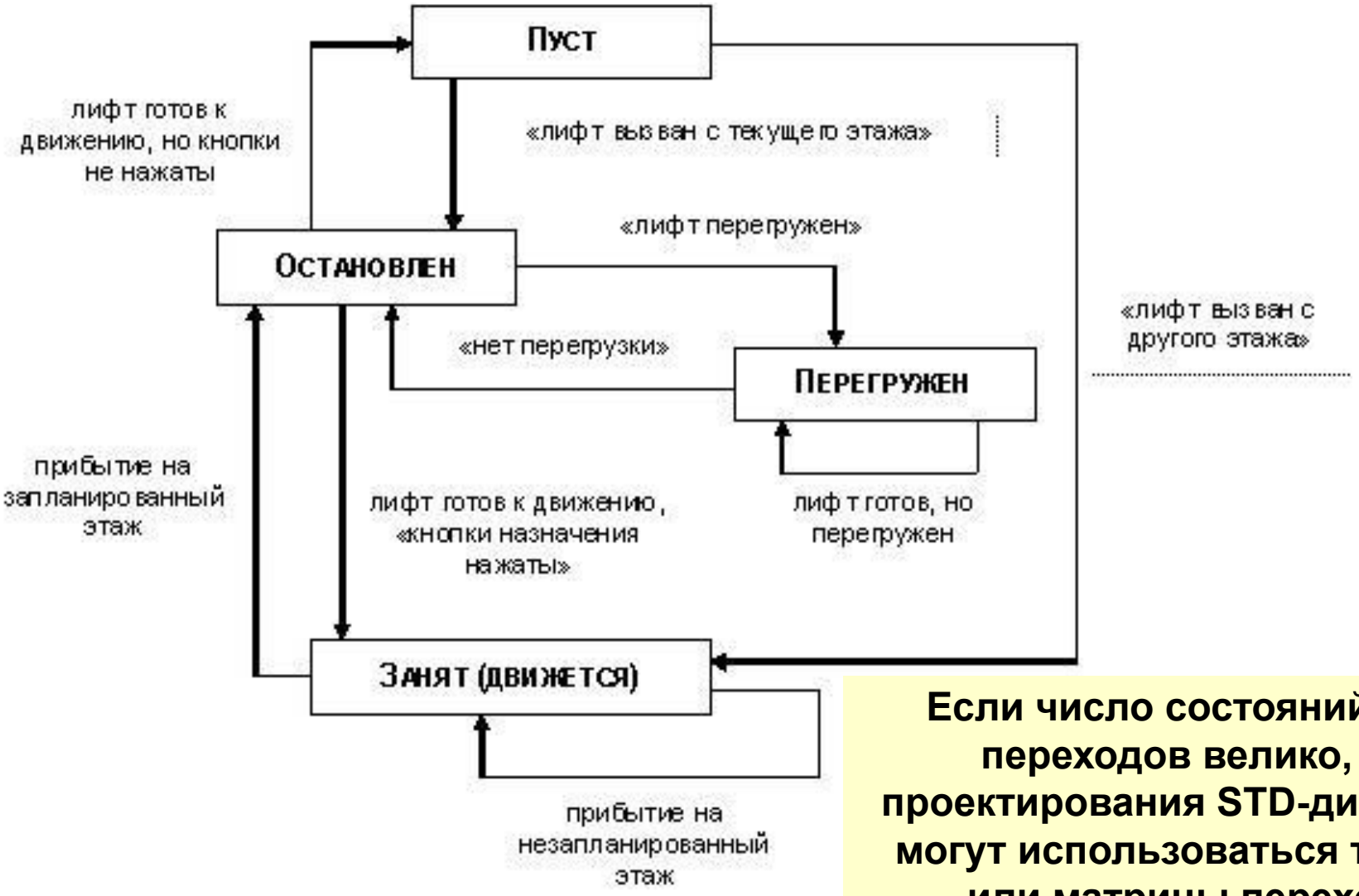
Все ли состояния достижимы?

Все ли состояния имеют выход?

Для каждого состояния – реагирует ли система соответствующим образом на все возможные условия, особенно на ненормальные?

Все ли входные (выходные) потоки управляющего процесса отражены в условиях (действиях) на STD?

Пример STD-диаграммы(диаграмма переходов состояний для системы управления лифтом)



Если число состояний и/или переходов велико, для проектирования STD-диаграммы могут использоваться таблицы или матрицы переходов состояний.

Источник:http://5fan.ru/wievjob.php?id=286

Пример таблицы или матрицы переходов состояний (ECS) для рассмотренной STD-диаграммы

	Занят (движется)	Остановлен	Пуст (стоит)	Перегружен (стоит)
Занят (движется)	прибытие на незапланированный этаж	прибытие на запланированный этаж		
Остановлен	лифт готов к движению, «кнопки назначения нажаты»		лифт готов к движению, но кнопки не нажаты	«лифт перегружен»
Пуст (стоит)	«лифт вызван с другого этажа»	«лифт вызван с текущего этажа»		
Перегружен (стоит)		«нет перегрузки»		Лифт готов, но перегружен

Источник:
http://5fan.ru/wievjob.php?id=2867

Flowcharts (блок-схемы)

Для построения поведенческой модели обычно используются блок-схемы алгоритмов. Как правило, их строят для функций (процессов), показываемых на последних уровнях диаграмм декомпозиции IDEF0 и DFD.

Построение блок-схем алгоритмов регламентируется

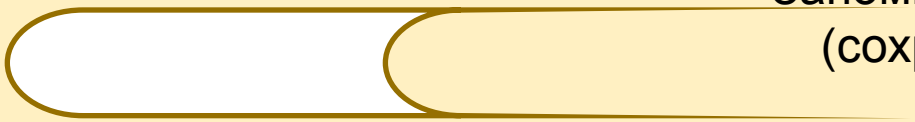
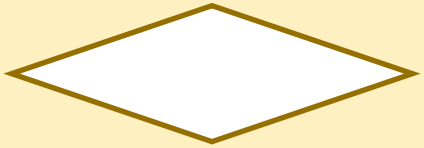
- **ГОСТ 19.701-90** «Единая система программной документации. Схемы алгоритмов программ, данных и систем. Условные обозначения и правила выполнения».
- Данный государственный стандарт составлен на основе международного стандарта «**ISO 5807-85.** Information processing – Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts».

Из ГОСТ 19.701-90

Под *схемой* понимается графическое представление определения, анализа или метода решения задачи. С помощью схем можно отобразить как статические, так и динамические аспекты системы. Символы, приведенные в гос стандарте, могут использоваться в следующих *типах схем*:

- **схемы данных** – определяют последовательность обработки данных и их носители;
- **схемы программ** – отображают последовательность операций в программе (по сути, это и есть блок-схемы алгоритмов в традиционном понимании);
- **схемы работы системы** – отображают управление операциями и потоки данных в системе;
- **схемы взаимодействия программ** – отображают путь активации программ (модулей) и их взаимодействие с соответствующими данными;
- **схемы ресурсов системы** – отображают конфигурацию блоков данных и обрабатывающих блоков.

Некоторые условные обозначения на блок-схемах:

Символ	Наименование
	Данные
	Типовой (предопределенный) процесс
	Процесс
	Запоминающее устройство (сохраненные данные)
	Решение

Семейство IDEF (Integration Definition for Function Modeling):

- **IDEF** - методологии создавались в рамках предложенной ВВС США программы компьютеризации промышленности - ICAM, в ходе реализации которой выявилась потребность в разработке методов анализа процессов взаимодействия в производственных (промышленных) системах.

Принципиальным требованием при разработке семейства методологий была возможность эффективного обмена информацией между всеми специалистами - участниками программы ICAM (отсюда название: **Icam DEFinition** - IDEF другой вариант - Integrated DEFinition).

После опубликования стандарта он успешно применялся в самых различных областях бизнеса, показав себя эффективным средством анализа, конструирования и отображения бизнес-процессов.

Definition(англ.) - определение, формулирование, формулировка, описание

История развития методологий моделирования бизнес-процессов

Период	Методология моделирования бизнес-процессов	Методология (стандарты) управления качеством
40 – 60-е гг.	Появление алгоритмических языков описания	Национальные стандарты
60-е гг.	Появление методологии SADT (структурного анализа и проектирования)	Развитие стандартов в различных областях, в частности в области контроля качества продукции
70 – 80-е гг.	Появление методологий серии IDEF (IDEF0, IDEF3, IDEF1X), DFD, ERD	Принятие МС ИСО серии 9000 версии 1988 г.
90-е гг.	Появление методологий ARIS (архитектура интегрированных информационных систем), UML (универсальный язык моделирования), методологий компаний Oracle, Baan, Rational и др.	Принятие МС ИСО серии 9000 версии 1994 г. (в стандартах закладываются основы процессного подхода)
2000 г.	Принятие МС ИСО серии 9000 версии 2000 г., четкое определение процессного подхода к управлению организацией	

Волков О. Стандарты и методологии моделирования бизнес-процессов. Режим доступа: http://www.connect.ru/article.asp?id=5710.

Семейство IDEF
(для справки)

Семейство IDEF(для справки)

IDEF0 - Function Modeling - методология функционального моделирования. С помощью наглядного графического языка IDEF0, изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков - в терминах IDEF0). **Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы.** Методология IDEF0 - развитие хорошо известного графического языка описания функциональных систем SADT (Structured Analysis and Design Teqnique);

В 2000 г. – IDEF0 был принят в качестве стандарта и в Российской Федерации (РД IDEF0-2000).

Семейство IDEF(для справки)

- **IDEF1 - Information Modeling - методология моделирования информационных потоков** внутри системы, позволяющая отображать и анализировать их структуру и взаимосвязи;
- **IDEF1X (IDEF1 Extended) - Data Modeling - методология построения реляционных структур (баз данных),** относится к типу методологий "Сущность-взаимосвязь" (ER - Entity-Relationship) и, как правило, используется для моделирования реляционных баз данных, имеющих отношение к рассматриваемой системе;

Семейство IDEF(для справки)

- **IDEF2 - Simulation Model Design - методология динамического моделирования развития систем.** В связи с весьма серьезными сложностями анализа динамических систем от этого стандарта практически отказались, и его развитие приостановилось на самом начальном этапе. В н в. существуют алгоритмы и их компьютерные реализации, позволяющие превращать набор статических диаграмм IDEF0 в динамические модели, построенные на базе "раскрашенных сетей Петри" (CPN - Color Petri Nets);
- **IDEF3 - Process Description Capture - Документирование технологических процессов,** IDEF3 - методология документирования процессов, происходящих в системе (например, на предприятии), описываются сценарий и последовательность операций для каждого процесса**. IDEF3 имеет прямую взаимосвязь с методологией IDEF0 - каждая функция (функциональный блок) может быть представлена в виде отдельного процесса средствами IDEF3;**

Семейство IDEF(для справки)

- **IDEF4 - Object-Oriented Design - методология построения объектно-ориентированных систем,** позволяют отображать структуру объектов и заложенные принципы их взаимодействия, тем самым позволяя анализировать и оптимизировать сложные объектно-ориентированные системы;
- **IDEF5 - Ontology Description Capture - Стандарт онтологического исследования сложных систем.** С помощью методологии IDEF5 онтология системы может быть описана при помощи определенного словаря терминов и правил, на основании которых могут быть сформированы достоверные утверждения о состоянии рассматриваемой системы в некоторый момент времени. На основе этих утверждений формируются выводы о дальнейшем развитии системы и производится её оптимизация;

Семейство IDEF(для справки)

- **IDEF6 - Design Rationale Capture - Обоснование проектных действий.** Назначение IDEF6 состоит в облегчении получения "знаний о способе" моделирования, их представления и использования при разработке систем управления предприятиями. Под "знаниями о способе" понимаются причины, обстоятельства, скрытые мотивы, которые обуславливают выбранные методы моделирования. Проще говоря, "знания о способе" интерпретируются как ответ на вопрос: "почему модель получилась такой, какой получилась?" Большинство методов моделирования фокусируются на собственно получаемых моделях, а не на процессе их создания. Метод IDEF6 акцентирует внимание именно на процессе создания модели;
- **IDEF7 - Information System Auditing - Аудит информационных систем.** Этот метод определён как востребованный, однако так и не был полностью разработан;

Семейство IDEF(для справки)

- **IDEF8 - User Interface Modeling - Метод разработки интерфейсов взаимодействия оператора и системы (пользовательских интерфейсов).** Современные среды разработки пользовательских интефейсов в большей степени создают внешний вид интефейса. IDFE8 фокусирует внимание разработчиков интерфейса на программировании желаемого взаимного поведения интефеса и пользователя на трех уровнях: выполняемой операции (что это за операция); сценарии взаимодействия, определяемом специфической ролью пользователя (по какому сценарию она должна выполняться тем или иным пользователем); и, наконец, на деталях интерфейса (какие элементы управления, предлагает интерфес для выполнения операции);

Семейство IDEF(для справки)

- **IDEF9 - Scenario-Driven IS Design (Business Constraint Discovery method) - Метод исследования бизнес ограничений** создан для облегчения обнаружения и анализа ограничений в условиях которых действует предприятие. Обычно, при построении моделей описанию ограничений, оказывающих влияние на протекание процессов на предприятии уделяется недостаточное внимание. Знания об основных ограничениях и характере их влияния, закладываемые в модели, в лучшем случае остаются неполными, несогласованными, распределенными нерационально, но часто их вовсе нет. Это не обязательно приводит к тому, что построенные модели нежизнеспособны, но возможно их реализация столкнется с непредвиденными трудностями, в результате чего их потенциал будет не реализован. Тем не менее в случаях, когда речь идет именно о совершенствовании структур или адаптации к предсказываемым изменениям, знания о существующих ограничениях имеют критическое значение;

Семейство IDEF(для справки)

- IDEF10 - Implementation Architecture Modeling - Моделирование архитектуры выполнения. Этот метод определён как востребованный, однако так и не был полностью разработан;
- IDEF11 - Information Artifact Modeling. Этот метод определён как востребованный, однако так и не был полностью разработан;
- IDEF12 - Organization Modeling - Организационное моделирование. Этот метод определён как востребованный, однако так и не был полностью разработан;

Семейство IDEF(для справки)

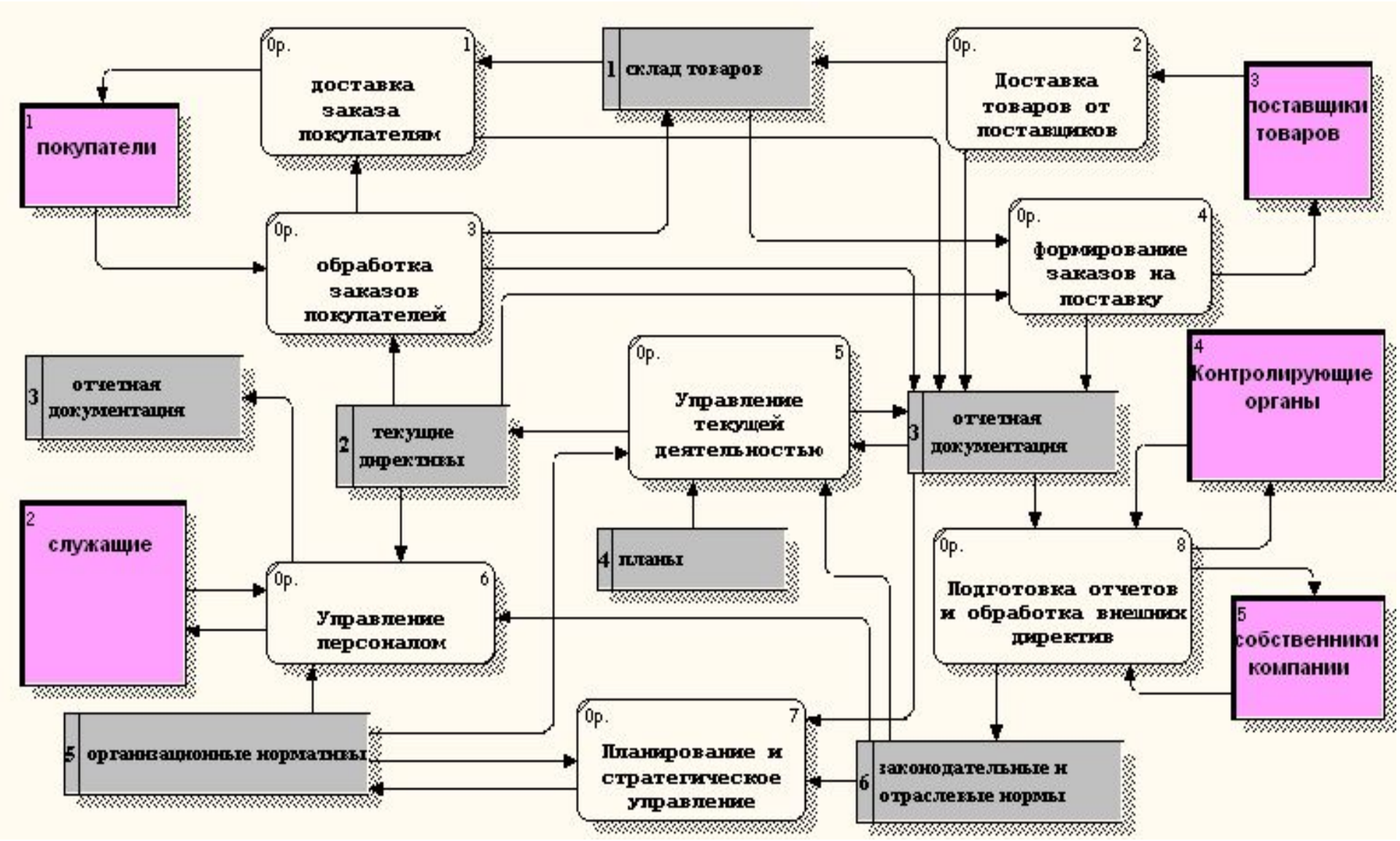
- **IDEF14 - Network Design - Метод проектирования компьютерных сетей, основанный на анализе требований, специфических сетевых компонентов, существующих конфигураций сетей.** Также он обеспечивает поддержку решений, связанных с рациональным управлением материальными ресурсами, что позволяет достичь существенной экономии;

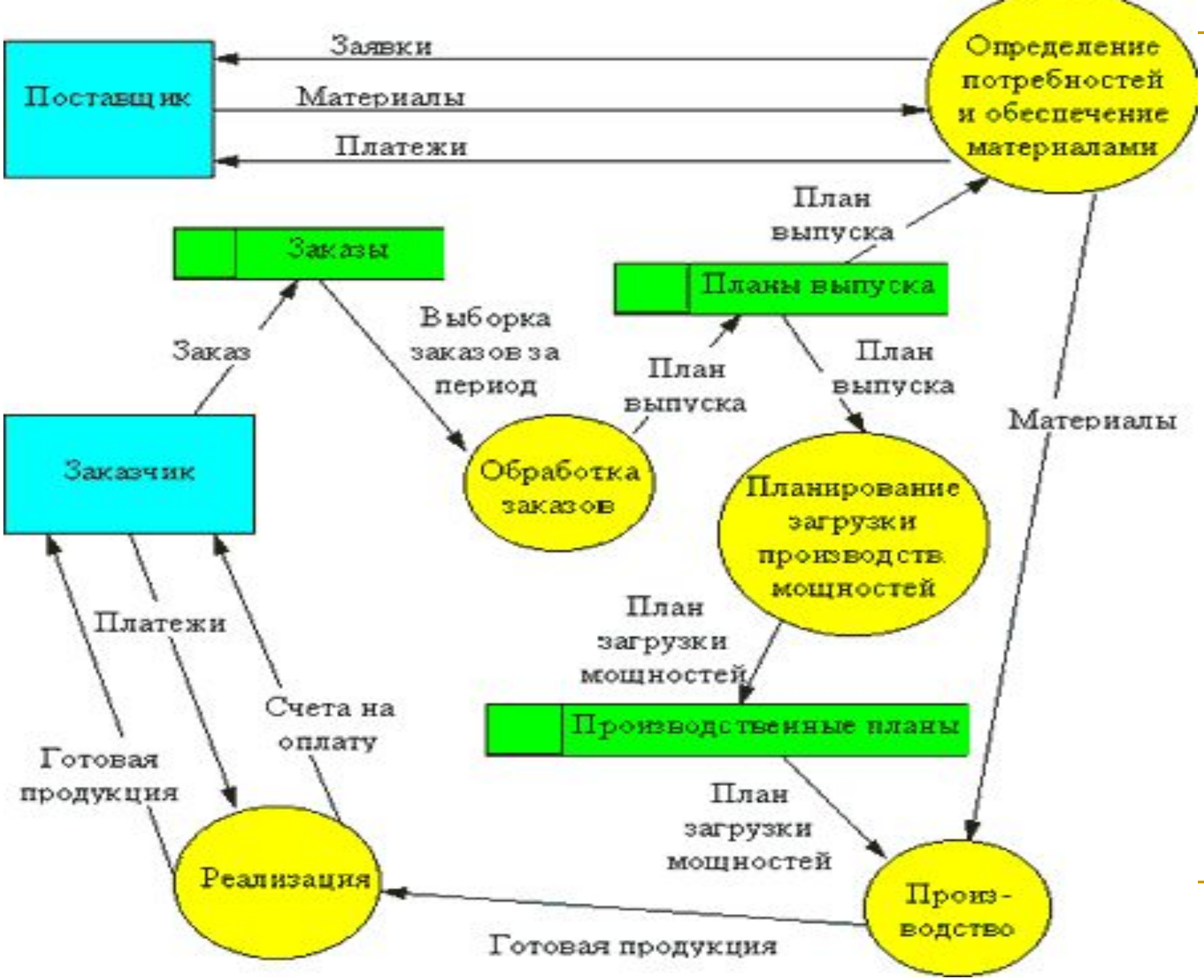
Нотация IDEF0 (Integration Definition for Function Modeling)

Диаграммы потоков данных

В DFD модель системы определяется как иерархия диаграмм потоков данных, описывающих процессы преобразования информации от момента ее ввода в систему до выдачи конечному пользователю.

Примеры DF диаграмм:




Поставщик
Заявки
Материалы
Платежи
Определение потребностей и обеспечение материалами
План выпуска
Заказы
Планы выпуска
Выборка заказов за период
Заказ
План выпуска
План выпуска
Материалы
Заказчик
Обработка заказов
Планирование загрузки производств. мощностей
Платежи
План загрузки мощностей
Производственные планы
Счета на оплату
Готовая продукция
План загрузки мощностей
Реализация
Произ-водство
Готовая продукция

Особенности

Диаграммы потоков данных используются для описания движения документов и обработки информации. В отличие от IDEF0, где система рассматривается как взаимосвязанные функциональные блоки, а дуги представляют собой жесткие взаимосвязи, стрелки в DFD показывают лишь то, как объекты (включая данные) движутся от одной работы к другой. DFD отражает функциональные зависимости значений, вычисляемых в системе, включая входные значения, выходные значения и внутренние хранилища данных.

DFD - это граф, на котором показано движение значений данных от их источников через преобразующие их процессы к их потребителям в других объектах.

DFD содержит процессы, которые преобразуют данные, потоки данных, которые переносят данные, активные объекты, которые производят и потребляют данные, и хранилища данных, которые пассивно хранят данные.

На практике

при создании моделей процессов полезно использовать несколько способов описания:

- Сначала - модель в нотации IDEF0, выявляем функции, входящие в процесс. Затем: декомпозиция процесса.

- При достижении некоторого уровня детализации (три-четыре) целесообразно сформировать для каждого процесса несколько схем в различных форматах: управление — IDEF0, а потоки данные и материалов — в DFD.

Нотация DFD может использоваться в качестве основной нотации функционального моделирования, однако, часто она применяется как дополнительная по отношению к IDEF0.

Использование DFD

- Нотация DFD может использоваться в качестве основной нотации функционального моделирования, однако, часто она применяется как дополнительная по отношению к IDEF0.

2 вида DFD нотаций

Нотация Гейна-Сарсона

(Gane-Sarson)

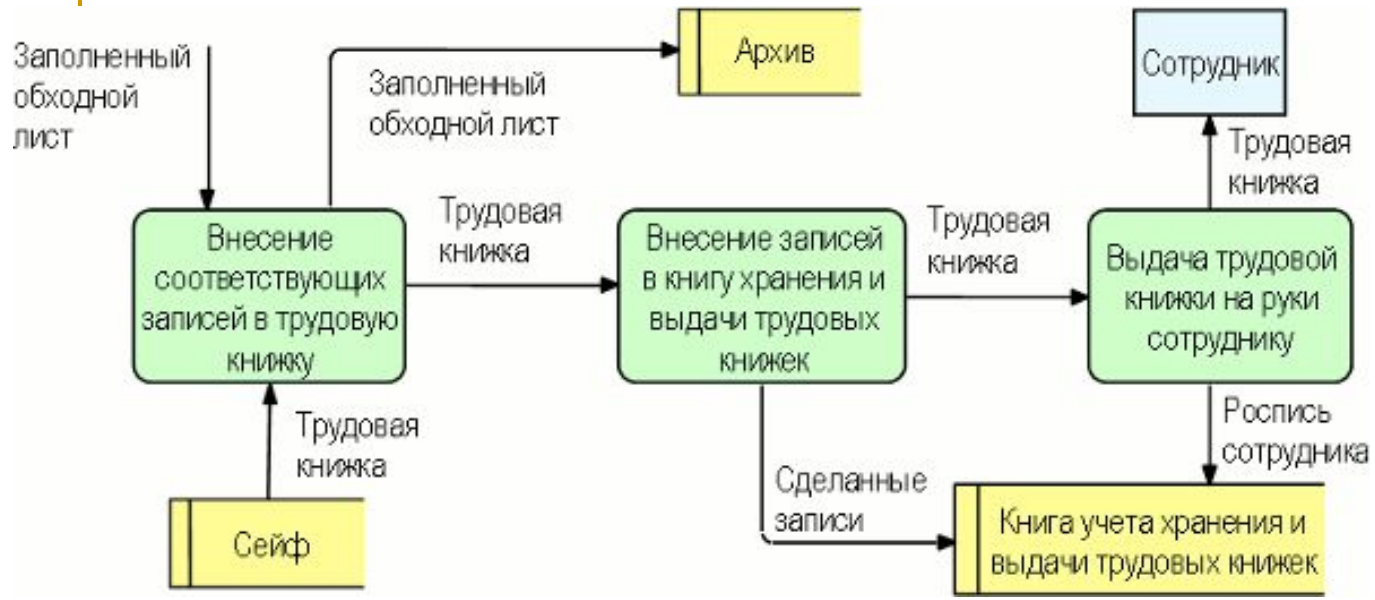
Нотация Йордана/де Марко

(Yourdon)

Авторами одной из первых графических нотаций DFD (1979 г.) стали Эд Йордан (Yourdon) и Том де Марко (DeMarko).

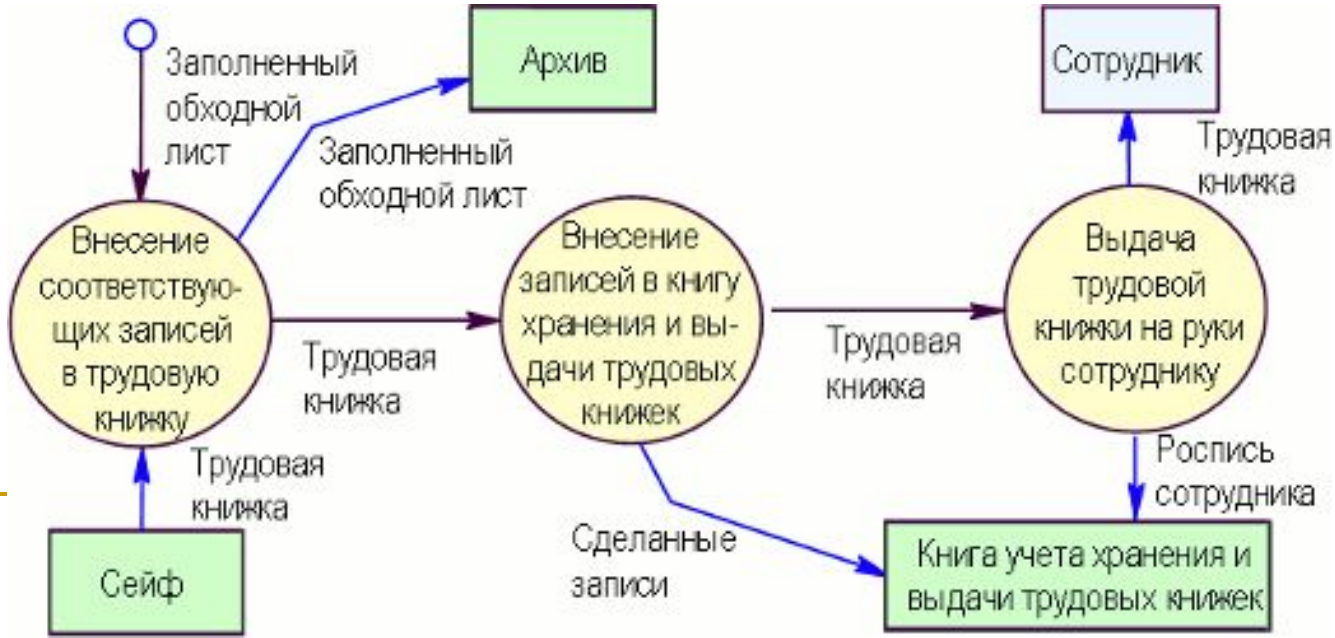
Эти нотации отличаются только тем набором графических примитивов, которые используются для построения функциональных моделей

Методология DFD в различных нотациях



Нотация Гейна-Сарсона

Нотация Йордана/ де Марко



Для чего служат нотации DFD?

Они нужны для описания реально существующих в организации потоков данных. Описания могут создаваться как по **процессному**, так и по **функциональному** признаку.

В этом случае мы получаем модели бизнес-процессов в формате DFD.

В этом случае получаем — схему обмена данными между подразделениями.

Для чего служат нотации DFD?

Созданные модели потоков Данных организации могут быть использованы при решении таких задач, как:

- определение существующих хранилищ данных (текстовые документы, файлы, Система управления базой данных — СУБД);
- определение и анализ данных, необходимых для выполнения каждой функции процесса;
- подготовка к созданию модели структуры данных организации (ERD-модель IDEF1X);
- выделение основных и вспомогательных бизнес-процессов организации.

Нотация DFD может быть эффективно применена для описания потоков документов или потоков материальных ресурсов

Основные элементы DFD и их назначение

Словарь данных

Структуры потоков данных и точные определения их компонент хранятся и анализируются в словаре данных.

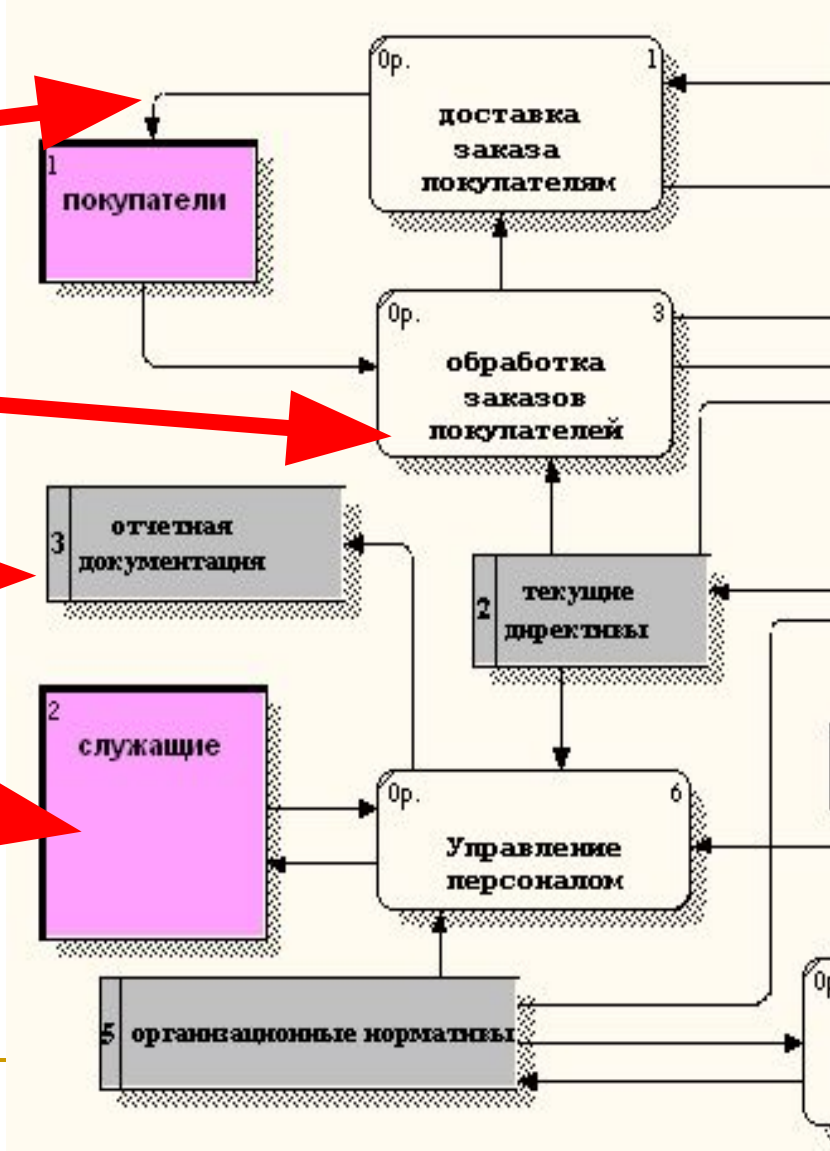
Определения элементов данных в словаре осуществляются следующими видами описаний:

- описанием значений потоков и хранилищ, изображенных на DFD;
- описанием композиции агрегатов данных, движущихся вдоль потоков, т.е. комплексных данных, которые могут расчленяться на элементарные символы (например, АДРЕС ПОКУПАТЕЛЯ содержит ПОЧТОВЫЙ ИНДЕКС, ГОРОД, УЛИЦУ и т.д.);
- описанием композиции групповых данных в хранилище;
- специфицированием значений и областей действия элементарных фрагментов информации в потоках данных и хранилищах;
- описанием деталей отношений между хранилищами

http://www.myshared.ru/slide/766786/

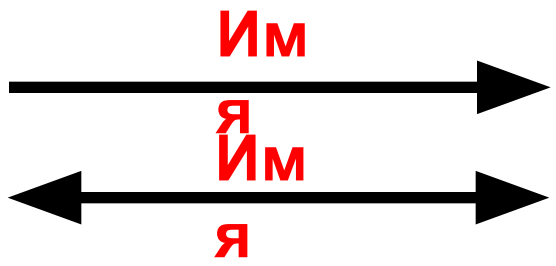
Основные элементы DFD:

- *поток данных*
- *процесс*
- *хранилище*
- *внешняя сущность*



Поток данных -

- соединяет выход объекта (или процесса) с входом другого объекта (или процесса). Он представляет промежуточные данные вычислений. Поток данных изображается в виде стрелки между производителем и потребителем данных, помеченной именами соответствующих данных. ***ПОТОКИ ДАННЫХ*** являются механизмами, использующимися для моделирования передачи информации (или физических компонент) из одной части системы в другую.



Потоки на диаграммах изображаются стрелками (обычно именованными), ориентация которых указывает направление движения информации.

Слияние и разветвление стрелок: В DFD стрелки могут сливаться и разветвляться, что позволяет описать декомпозицию стрелок. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя.

Процесс

преобразует значения данных.

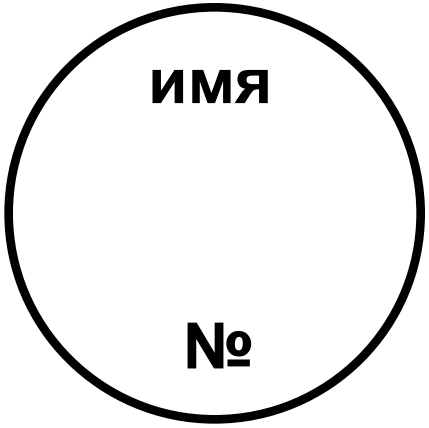
Процесс - преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом.

В реальной жизни процесс может выполняться некоторым подразделением организации, выполняющим обработку входных документов и выпуск отчетов, отдельным сотрудником, программой, установленной на компьютере, специальным логическим устройством и т. п.

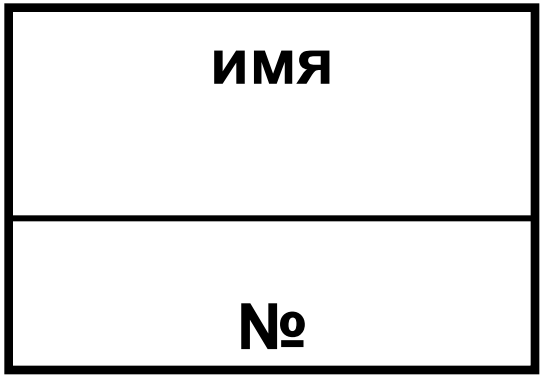
В отличие от SADT(IDEF0), в DFD все стороны блока процесса равнозначны !!!

Обозначение процесса в DFD

Примечание: нет единого стандарта и объекты DFD могут иметь разное обозначение.



Нотация Йордана/ де Марко



Нотация Гейна-Сарсона

BPWin

Хранилища = Накопители данных

позволяет на определенных участках определять данные, которые будут сохраняться в памяти между процессами.

Информация, которую содержит хранилище, может использоваться в любое время после ее определения, при этом данные могут выбираться в любом порядке.

Хранилище имеет имя, которое идентифицирует его содержимое.

В случае, когда поток данных входит или выходит в/из хранилища, и его структура соответствует структуре хранилища, он должен иметь то же самое имя.

Обозначение хранилища в DFD

Накопители данных являются неким прообразом базы данных ИСорганизации.

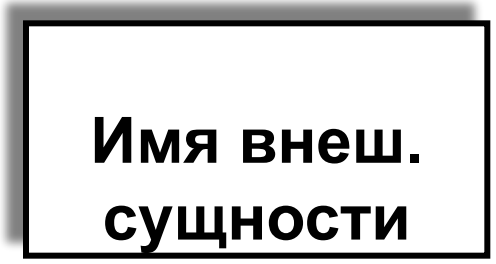
Нотация Йодана/де Марко

Нотация Гейна-Сарсона

Внешняя сущность (Терминатор)

- сущность вне контекста системы **материальный объект**, являющийся источником или приемником системных данных.
- Имя внешней сущности должно содержать существительное, например, Список студентов,
- Предполагается, что объекты, представленные такими узлами, не должны участвовать ни в какой обработке.

Нотация Йодана/де Марко



Нотация Гейна-Сарсона

Основные компоненты DFD

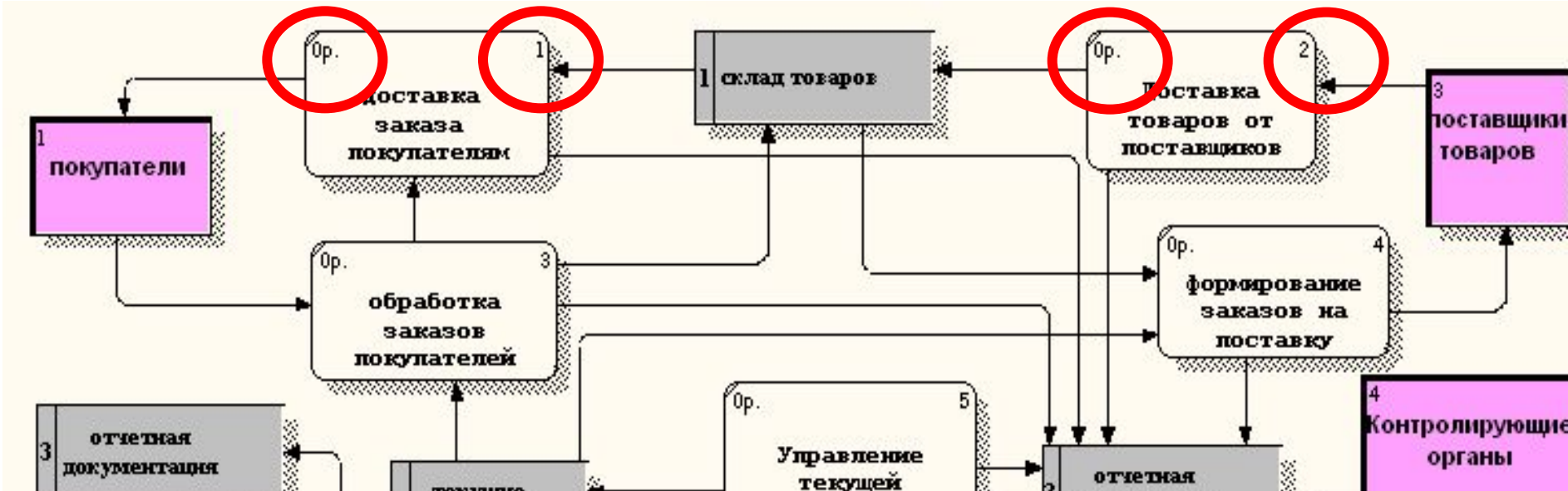
Нотация Гейна-Сарсона		Нотация Йордана/ де Марко
План выпуска →	*- Поток данных -*	Информация о клиентах →
Тестирование изделия / №	*- Процесс -*	Доставить заказ / №
Данные клиента	*-Хранилище-*	Список заказов
Клиент	*- Внешняя сущность -*	Сотрудник

Нумерация объектов

В DFD номер каждой работы(процесса) может включать префикс, номер родительской работы (А) и номер объекта. Номер объекта - это уникальный номер работы на диаграмме. Н-р, работа

может иметь номер А. 12.4.

Пример нумерации в BPwin:

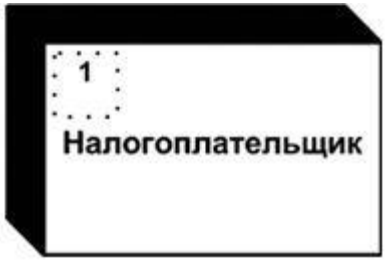


Нумерация объектов

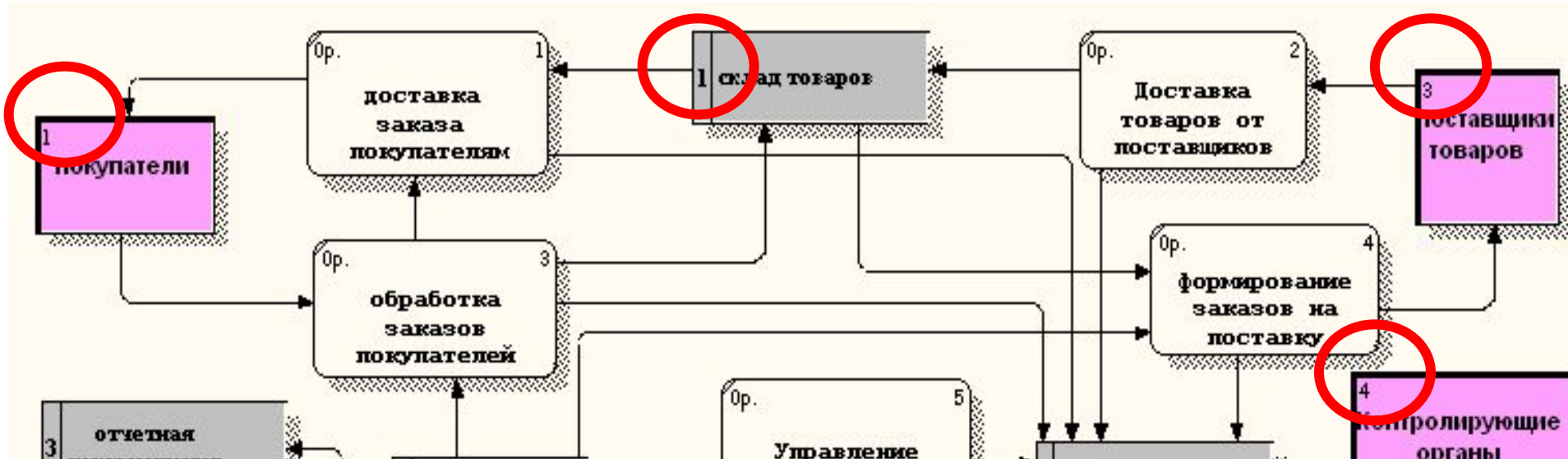
Уникальный номер имеют хранилища данных и внешние сущности независимо от их расположения на диаграмме.

- Каждое хранилище данных имеет префикс D и уникальный номер, например D5.

- Каждая внешняя сущность имеет префикс E и уникальный номер, например E5.



Пример нумерации в BPwin:



Правила и рекомендации построения DFD:

- Правила и рекомендации построения модели DFD в основном совпадают с принятыми в IDEF0.
- По аналогии с IDEF0 у каждого процесса (подсистемы) на диаграмме потоков данных(DFD) должен быть как минимум один входящий и один выходящий поток.
- Процесс должен запускаться на выполнение либо через обрабатываемый, либо через управляющий поток данных. Работа каждого процесса должна завершаться конкретным результатом.

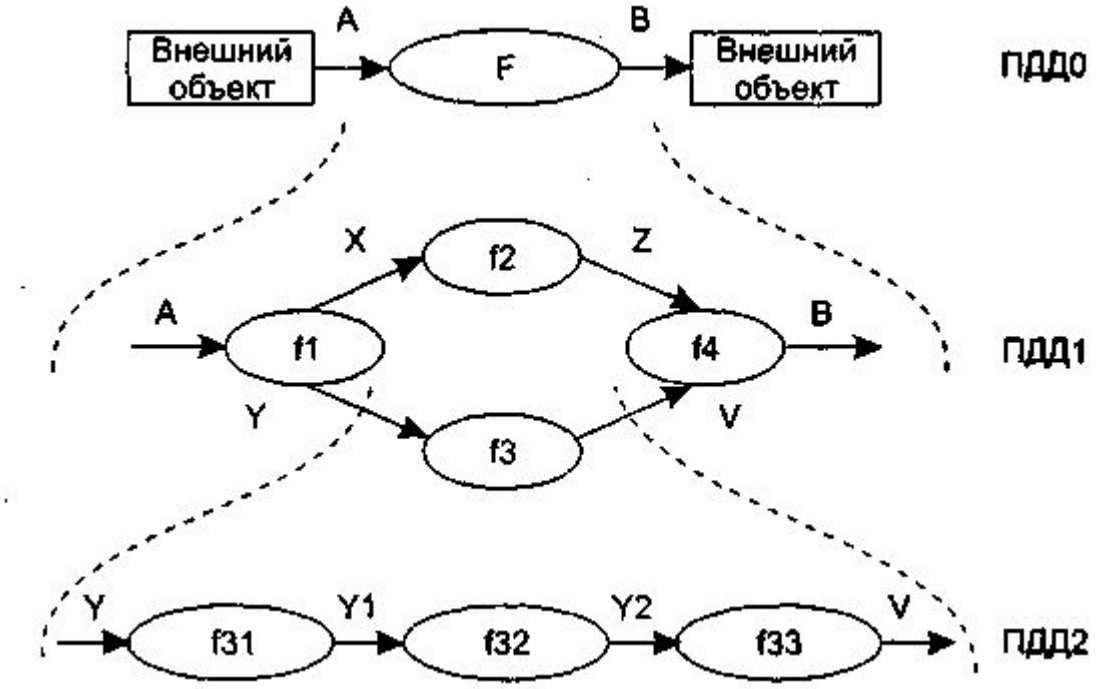
Правила и рекомендации построения DFD:

- **Каждый накопитель данных также должен иметь как минимум один входящий и один выходящий поток.** ***Наличие только входящих потоков в накопитель означает, что информация накапливается, но не используется.***
- ***Наличие только выходящих потоков из накопителя также является ошибкой.*** **Прежде чем использовать данные из накопителя, они должны там появиться в результате работы какого-либо процесса (подсистемы, внешней сущности).** ***Исключением из правил считается случай, когда накопитель является внешней сущностью.*** **Тогда допускается наличие либо только вх. стрелок, либо только вых. стрелок.**

Построение иерархии диаграмм потоков данных

Цель построения иерархии DFD **сделать описание системы ясным и понятным на каждом уровне детализации**, а также разбить его на части с точно определенными отношениями между ними.

Система взаимосвязанных диаграмм потоков данных



ПДД - Диаграмма потоков данных

Для достижения этой цели

целесообразно пользоваться следующими рекомендациями:

- Размещать на каждой диаграмме **от 3 до 6-7 процессов** (аналогично SADT). (Верхняя граница соответствует человеческим возможностям одновременного восприятия и понимания структуры сложной системы с множеством внутренних связей, нижняя граница выбрана по соображениям здравого смысла: нет необходимости детализировать процесс диаграммой, содержащей всего 1 или 2 процесса.)

- **Не загромождать диаграммы несущественными** на данном уровне деталями.

- **Декомпозицию потоков данных осуществлять параллельно с декомпозицией процессов**. Эти две работы должны выполняться одновременно, а не одна после завершения другой.

- Выбирать ясные, отражающие суть дела имена процессов и потоков, при этом стараться не использовать аббревиатуры.

Порядок построения DFD*

Сначала необходимо построить контекстную диаграмму.

Пример контекстной диаграммы. Топология - звезда

На контекстной диаграмме отображается основной процесс (сама система в целом) и ее связи с внешней средой (внешними сущностями)

*** - источник: Космачев С.Н. Автоматизированные информационные системы http://5fan.ru/wievjob.php?id=39990**

Контекстная диаграмма(DFD) для более сложной ИС (для системы определения допускаемых скоростей)

БД АРМ-П или СБД-П по отношению к системе являю внешними сущностями, они, в целях лучшего воспр показаны в виде накопителя данных.

NODE:	TITLE: Система определения допускаемых скоростей	NUMBER:

Построение контекстной диаграммы

- Перед построением контекстной DFD необходимо проанализировать внешние события (внешние сущности), оказывающие влияние на функционирование системы. Количество потоков на контекстной диаграмме должно быть по возможности небольшим, поскольку каждый из них может быть в дальнейшем разбит на несколько потоков на следующих уровнях диаграммы.
  Для проверки контекстной диаграммы можно составить список событий.
- Список событий должен состоять из описаний действий внешних сущностей (событий) и соответствующих реакций системы на события.
- Каждое событие должно соответствовать одному или более потокам данных: вх. потоки интерпретируются как воздействия, а вых. потоки — как реакции системы на вх. потоки.

Построение контекстной диаграммы

- Для сложных систем строится иерархия контекстных диаграмм. *Признаками сложности могут быть:*
  - *наличие большого количества внешних сущностей (10 и более),*
  - *распределенная природа системы*
  - *многофункциональность системы*
- При этом контекстная диаграмма верхнего уровня содержит не единственный главный процесс, а набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем.

Порядок построения DFD

После того как построена контекстная диаграмма

- Каждый процесс на DFD, в свою очередь, может быть детализирован при помощи DFD или (если процесс элементарный) спецификации.
- Спецификация процесса должна формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу.
- Спецификация является конечной вершиной иерархии DFD.

Порядок построения DFD

При детализации должны выполняться следующие правила:

- правило балансировки - означает, что при детализации подсистемы или процесса детализирующая диаграмма в качестве внешних источников/приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных), с которыми имеет информационную связь детализируемая подсистема или процесс на родительской диаграмме;
- правило нумерации - означает, что при детализации процессов должна поддерживаться их иерархическая нумерация. Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2 и т.д.

Порядок построения DFD

Решение о завершении детализации процесса и использовании спецификации принимается аналитиком исходя из следующих критериев:

• наличия у процесса относительно небольшого кол-ва входных и выходных потоков данных (2-3 потока);

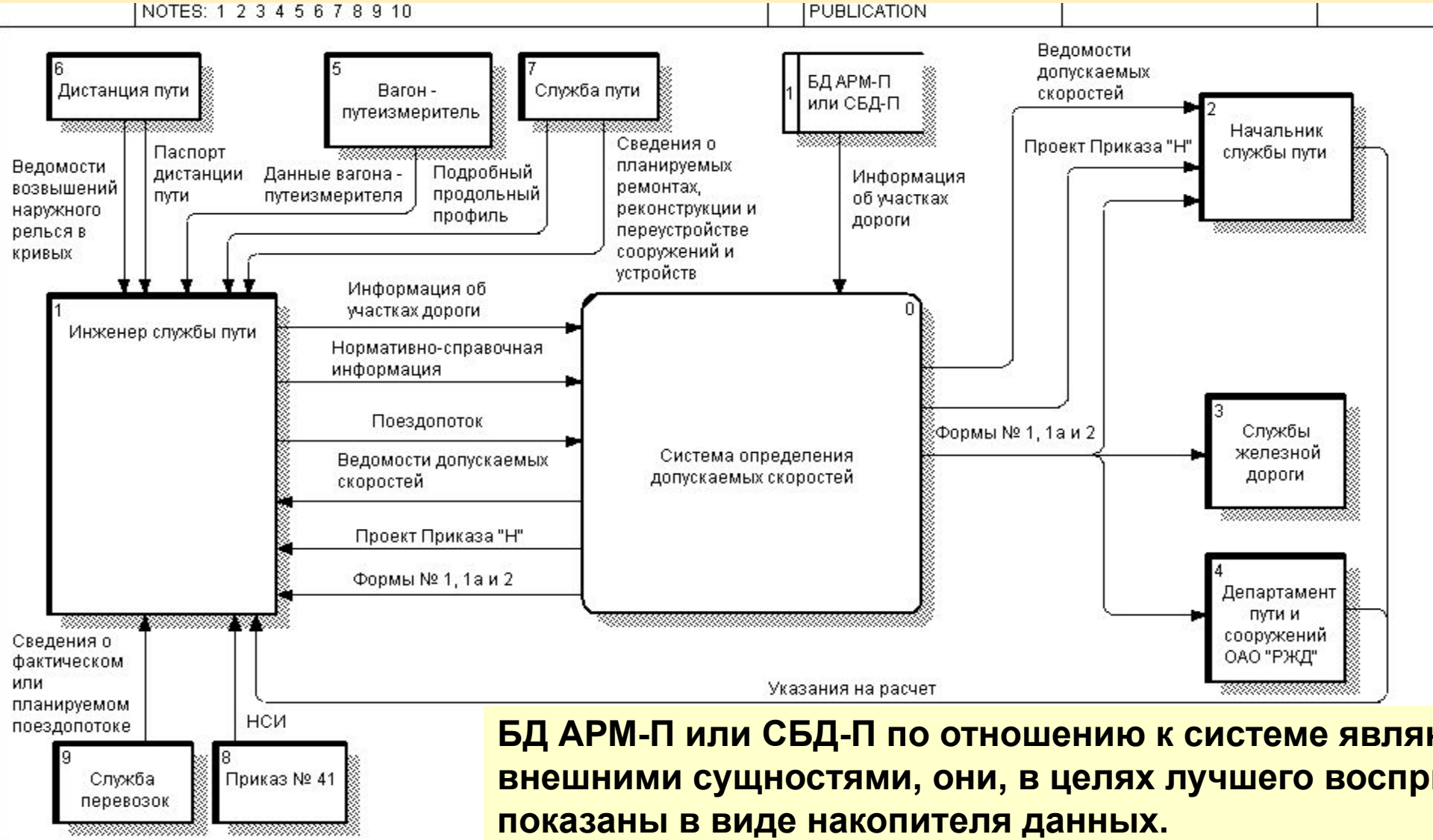
• возможности описания преобразования данных процессов в виде последовательного алгоритма;

• выполнения процессом единственной логической функции преобразования входной информации в выходную;

• возможности описания логики процесса при помощи спецификации небольшого объема (не более 20-30 строк) – миниспецификации.

Пример

Контекстная диаграмма(DFD) (для системы определения допускаемых скоростей)



БД АРМ-П или СБД-П по отношению к системе являю внешними сущностями, они, в целях лучшего воспр показаны в виде накопителя данных.

Диаграмма декомпозиции первого уровня (DFD)