лекция 2

Методология проектирования информационных систем

Структурный подход к моделированию систем

Методология функционального моделирования IDEF0

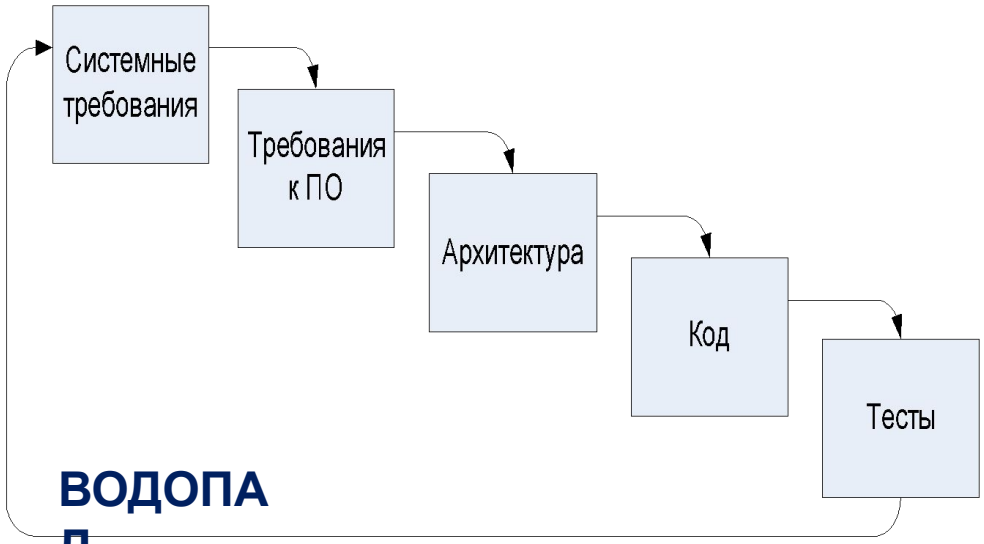
Основные вопросы

- Сущность структурного подхода
- Основные принципы структурного подхода
- Сущность методологии функционального моделирования IDEF0
- Основные понятия методологии IDEF0
- Правила построения моделей IDEF0
- Примеры функциональной модели в нотации IDEF0

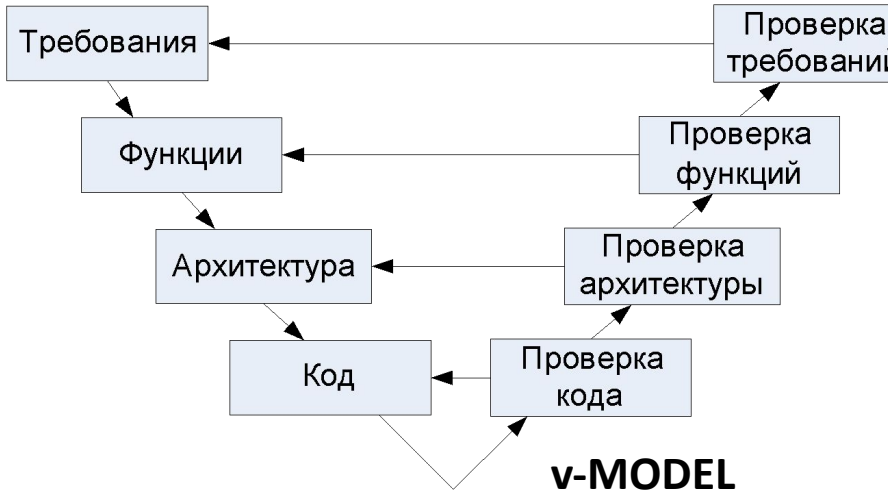
Жизненный цикл проектов разработки ПО

Жизненный цикл – совокупность процедур, связанных с последовательным изменением состояния программного обеспечения от формирования исходных требований к нему до окончания его эксплуатации конечным пользователем.

Жизненные циклы



ВОДОПАД



v-MODEL

СПИРАЛЬ

Роли в проекте

- Менеджер проекта
- Разработчик
- Тестировщик
- Специалист по контролю качества
- Специалист по внедрению и сопровождению
- Технический писатель

Процессы разработки ПО

- **Основные процессы**
  - Разработка требований
  - Разработка программного кода
  - Верификация
- **Поддерживающие процессы**
  - Управление качеством
  - Управление конфигурациями
  - Управление рисками

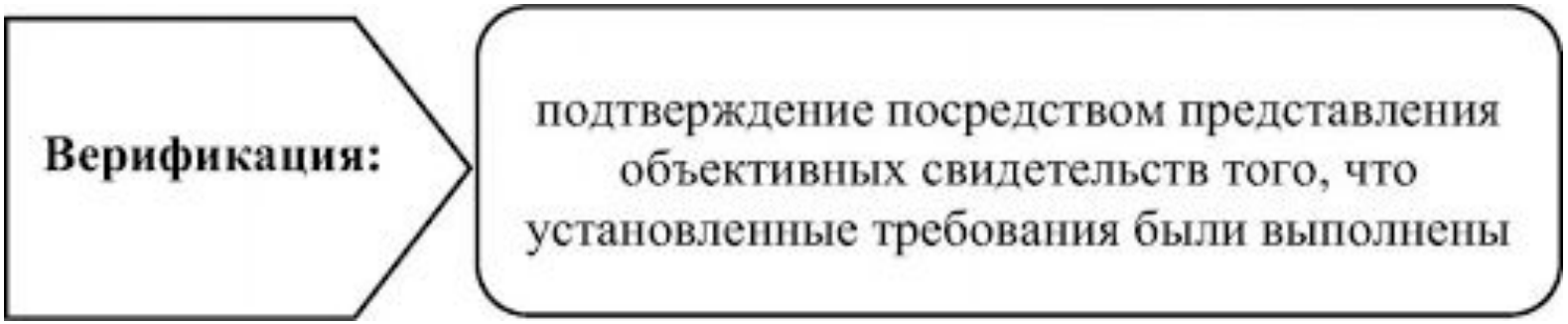
Верификация

- Требования к системе должны быть составлены так, чтобы можно было проверить корректность работы системы
- Верификация – процесс проверки корректности системы по отношению к требованиям

- **Specific**
  - **(Определенными)**
- **Measurable**
  - **(Измеряемыми)**
- **Achievable**
  - **(Достижимыми)**
- **Realistic**
- **(Реалистичными)**
- **Time-limited**
  - **(Ограниченными во времени**

Верификация

- Верификация и отладка – разные вещи
- Верификация – поиск того, что именно не работает в системе
- Отладка – устранение ошибок



Верификация

Пример требования:

«Система должна печатать 2, если на вход подается 3, и 3, если на вход подается 2»

Как может работать система, реализующая требование?

- Что будет, если на вход будет подано число, отличное от 2 и 3?
  - Выдается ошибка
  - Выдается само это число
  - Выдается «5 минус число»

Процессный подход к разработке требований

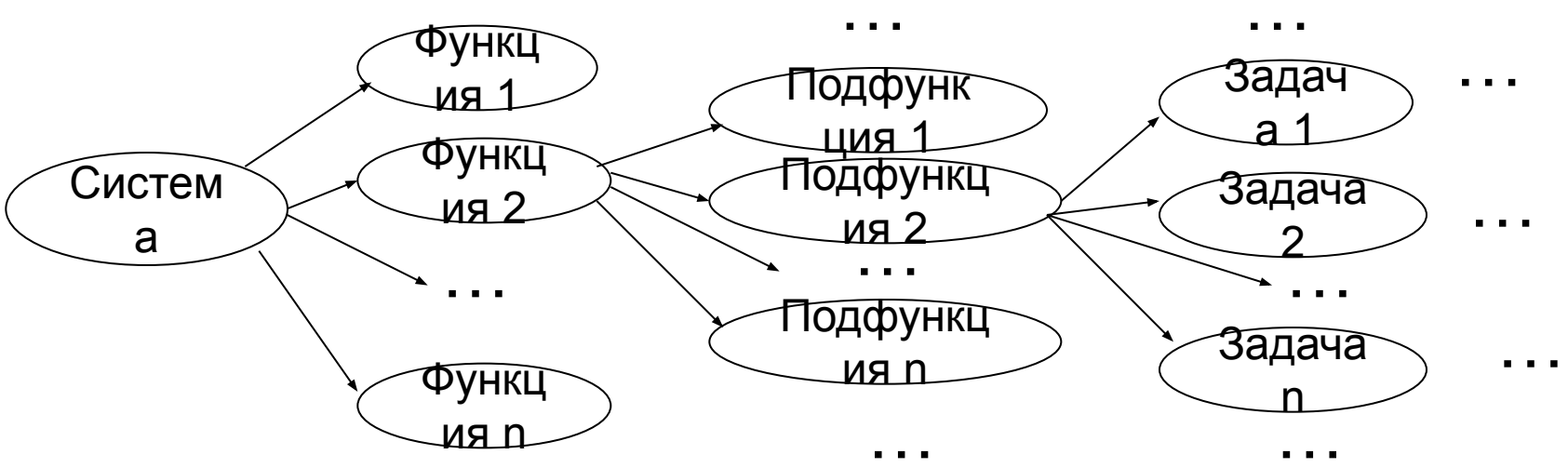
- Процесс – совокупность взаимосвязанной деятельности по преобразованию входа в выход, которая использует определенные ресурсы
- Процесс может быть управляемым и неуправляемым

Основы системного анализа

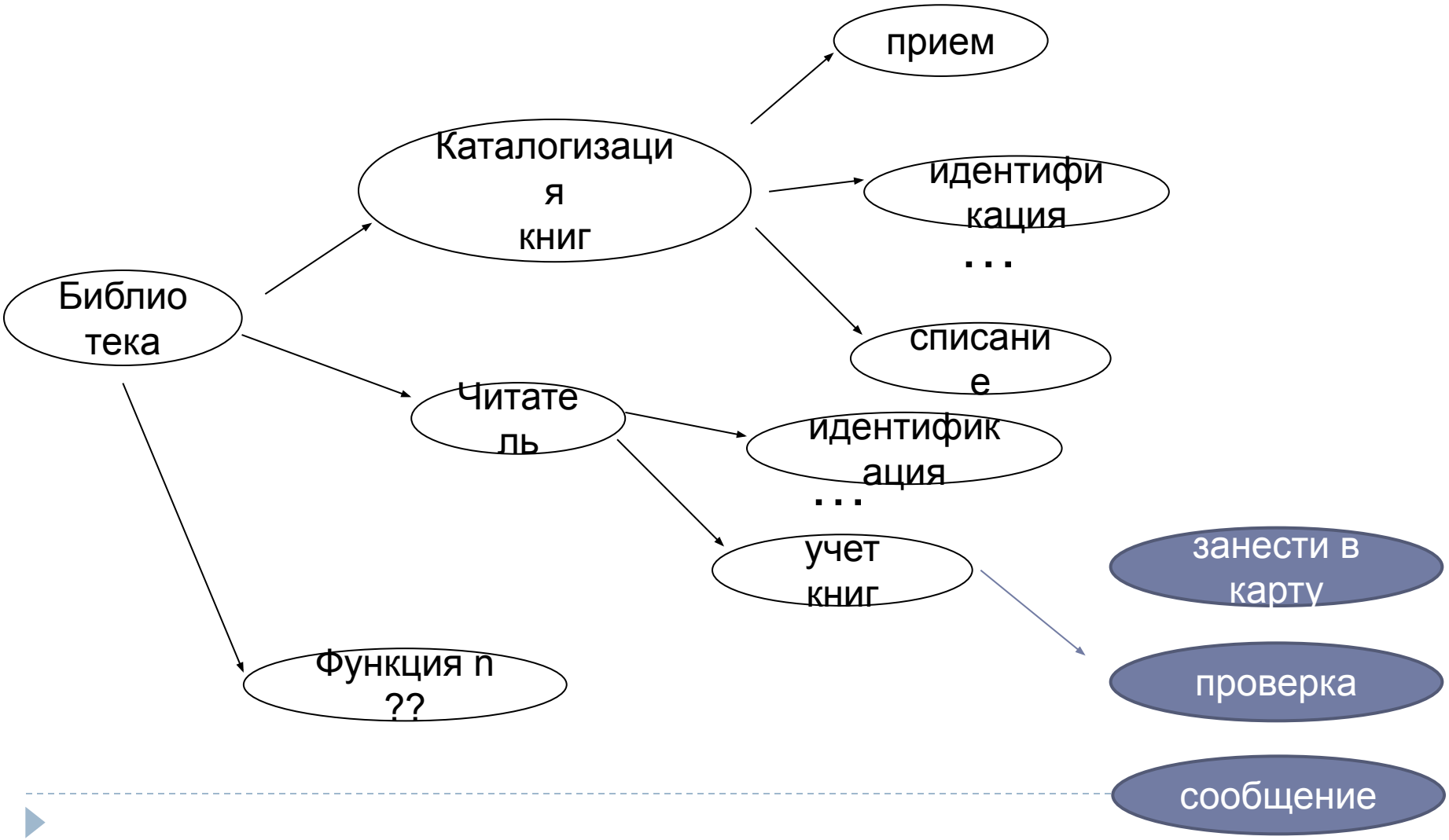
- Системный подход – признание того, что объекты реального мира декомпозируются на взаимодействующие части
- Система – совокупность взаимодействующих элементов, выполняющих общую задачу

Сущность структурного подхода к моделированию систем

Система разбивается *на функциональные подсистемы*, которые, в свою очередь, делятся на *подфункции*, подфункции – на *задачи* и т.д. до конкретных *процедур.*



пример. ИС БИБЛИОТЕКА



Базовые принципы структурного подхода

- принцип *«Разделяй и властвуй»*
- принцип *иерархического упорядочивания*
- принцип *абстрагирования*
- принцип *непротиворечивости*
- принцип *структурирования данных*

Принципы структурного подхода

В основу структурного проектирования положен принцип функциональной декомпозиции. Структура системы описывается в терминах иерархии ее функций.

Функциональная группа программ – это несколько программ, решающих одну функциональную задачу.

Иерархия представляет собой свойство упорядоченного множества компонентов между которыми установлены отношения приоритета.

Многоуровневое иерархическое построение системы позволяет локализовать на каждом из уровней соответствующие ему компоненты. Компоненты, имеющие один приоритет, образуют один иерархический уровень.

Принципы структурного подхода

Существует несколько типов подчиненности:

- иерархия целей ПО и его составляющих;
- иерархия задач и поведения групп программ;
- иерархия структуры ПО;
- иерархия компонентов ПО;
- иерархия данных и др.

Базовыми принципами структурного подхода являются:

- иерархическая декомпозиция на ряд подсистем;
- организация подсистем в древовидную структуру с добавлением новых деталей на каждом иерархическом уровне.

Принципы структурного подхода

Всем иерархическим структурам присущи следующие свойства:

- вертикальная соподчиненность, которая заключается в последовательном упорядоченном расположении взаимодействующих компонентов комплекса,

- право вмешательства и приоритетного воздействия на компоненты любых уровней со стороны компонентов более высоких иерархических уровней,

- зависимость действий компонентов верхних иерархических уровней от реакции на воздействие и от функционирования компонентов нижних уровней, информация о которых передается верхним уровням.

В результате в иерархической системе образуется два потока взаимодействий:

- сверху вниз – координирующий и управляющий,
- снизу вверх – информационный.

Функциональная иерархия данных

Функциональная иерархия данных показывает условное расстояние между расчетом переменной и ее использованием.

Переменные, которые рассчитываются и используются внутри программного модуля, называются **локальными.**

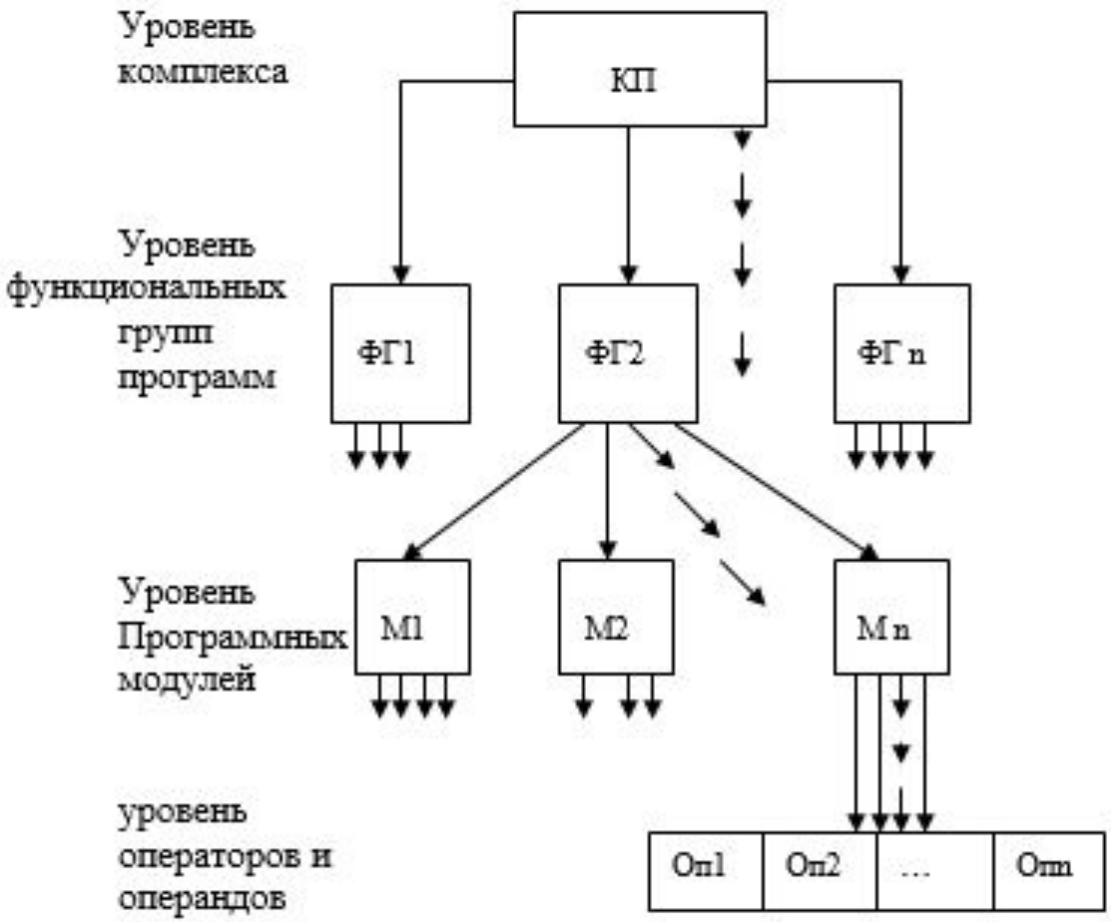
Переменные, которые используются двумя взаимодействующими программами и хранятся в течении всего времени работы этих программ называются **обменными.**

Переменные, которые используются многими программами системы называются **глобальными**. Они соответствуют высшему иерархическому уровню среди данных. . Эти переменны объединяются в информационные модули и определяют сложные связи внутри программного программной системы или внутри функциональной группы программ.

Иерархия программных компонентов

Одной из основных задач структурного подхода является формирование общей структуры программного комплекса. При ее формировании можно выделить следующие компонентные уровни:

1. уровень операторов и операндов, который соответствует компонентам текста программы на ЯП;
2. уровень программных модулей, оформленных, как законченные компоненты текста программ;
3. уровень функциональных групп программ (может быть от нескольких уровней до нескольких десятков уровней в зависимости от сложности решаемой задачи);
4. уровень комплекса, оформленного, как завершенное ПО.



С повышением уровня увеличивается количество реализующих его машинных команд и количество обрабатываемых переменных, а также количество типов, обрабатываемых переменных.

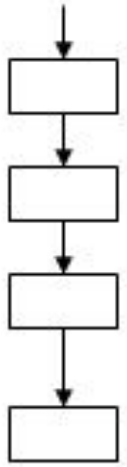
При этом уменьшается возможность использования компонентов в различных комбинациях для решения аналогичных задач, т.е. с повышением уровня программы все более специализируются.

Элементарные базовые конструкции, используемые при создании структурированной программы

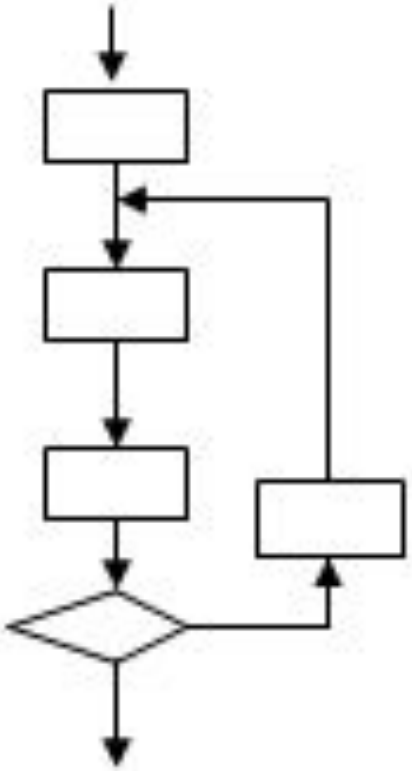
Простота исходных конструкций структурного программирования предотвращает появление сложных информационных связей. Каждая программа может быть создана на основе элементарных базовых конструкций 3-типов:

1. простой вычислительной последовательности;
2. выбора или альтернативы;
3. повторения или итерации.

Простая вычислительная последовательность заключается в последовательном преобразовании исходных данных. При этом операторы конструкции следуют один за другим, причем конец предыдущего оператора замыкается на начало следующего.

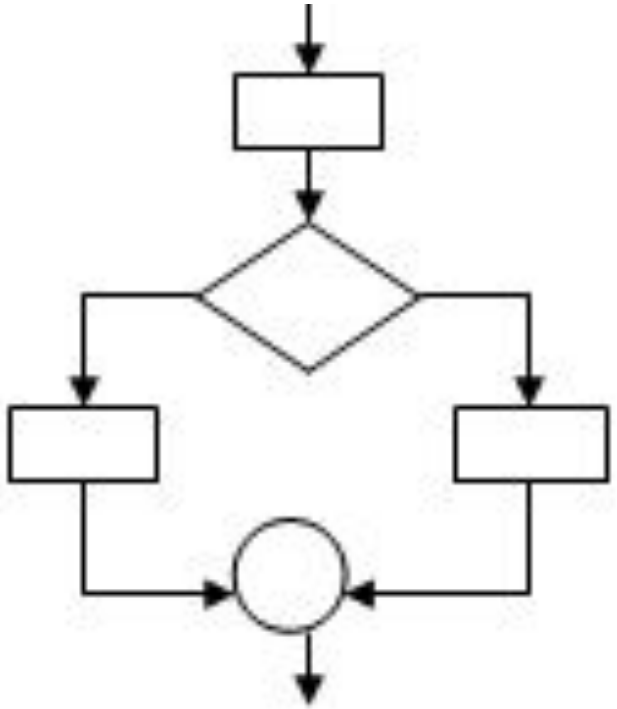


Элементарные базовые конструкции, используемые при создании структурированной программы



Итерация представляет собой конструкцию, в которой оператор или группа операторов повторяется боле одного раза. Для структурированной программы число итераций должно быть задано до входа в цикл.

Элементарные базовые конструкции, используемые при создании структурированной программы



Альтернатива состоит в проверке некоторого условия и в выборе одного из двух операторов преобразования данных. При ветвлении происходит однократный проход по одной из ветвей решения задачи

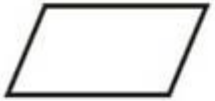
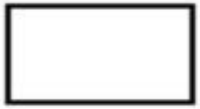
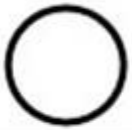
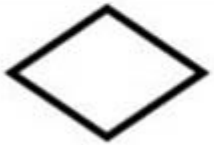
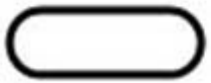
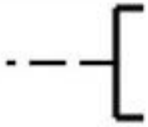
Элементарные базовые конструкции, используемые при создании структурированной программы

Существуют программные конструкции, использование которых рекомендуется максимально ограничивать. При искажении исходных данных они могут привести к непредсказуемым последствиям. Наиболее неустойчивой и трудно контролируемой конструкцией является безусловный переход по содержимому ячейки оперативной памяти (GO TO).

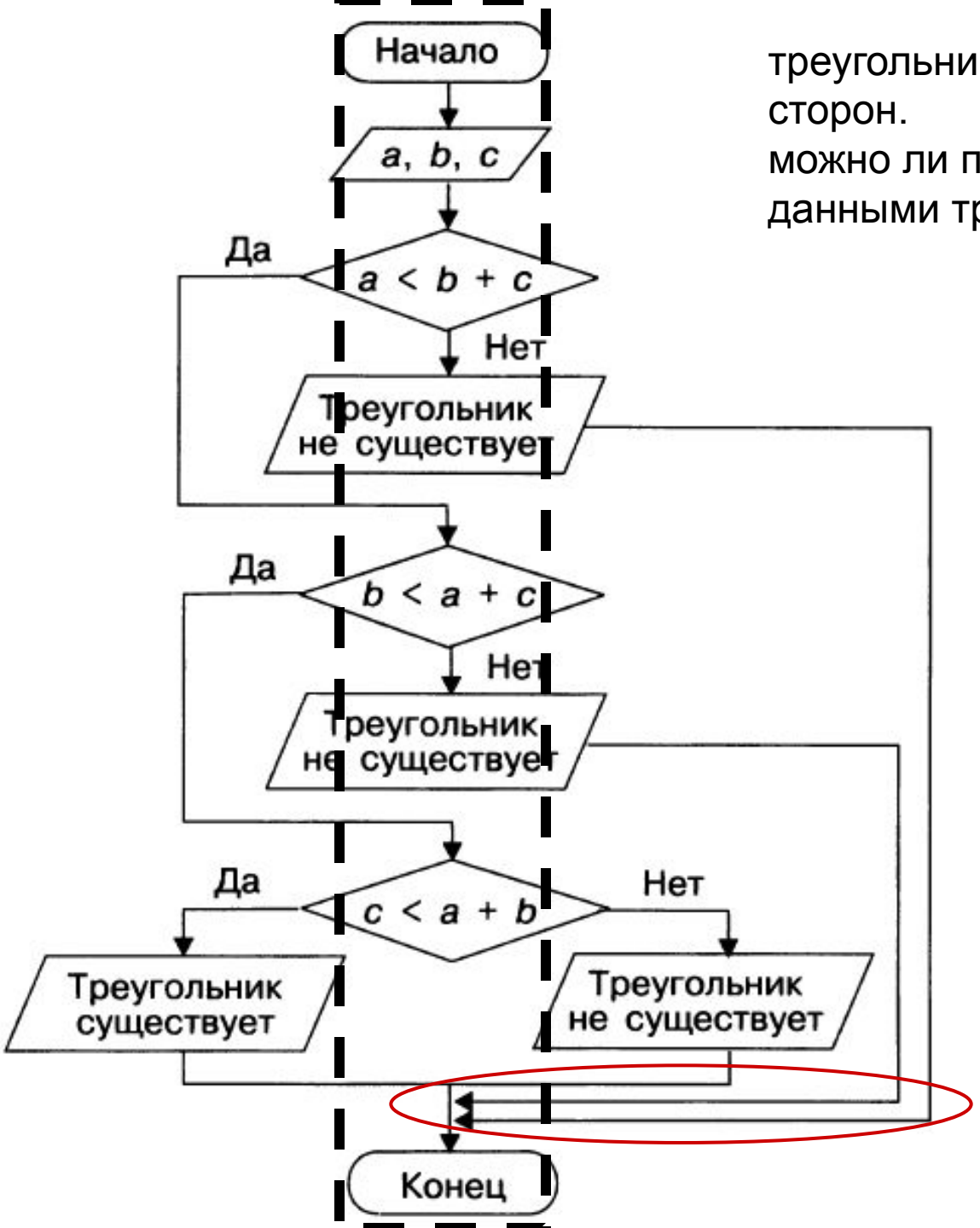
Структурированной считается программа, которая:

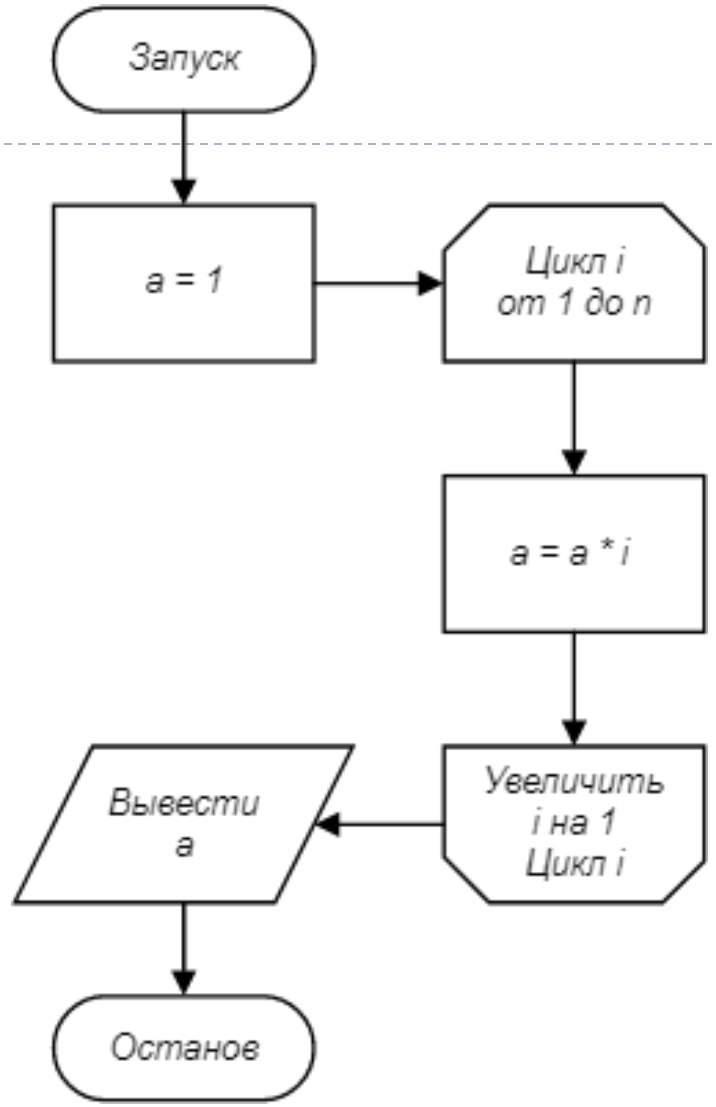
- не имеет переходов внутрь циклов или условных операторов;
- не имеет выходов из внутренней части циклов и условных операторов;
- число итераций должно быть задано до входа в цикл;
- ограничено использование оператора GO TO.

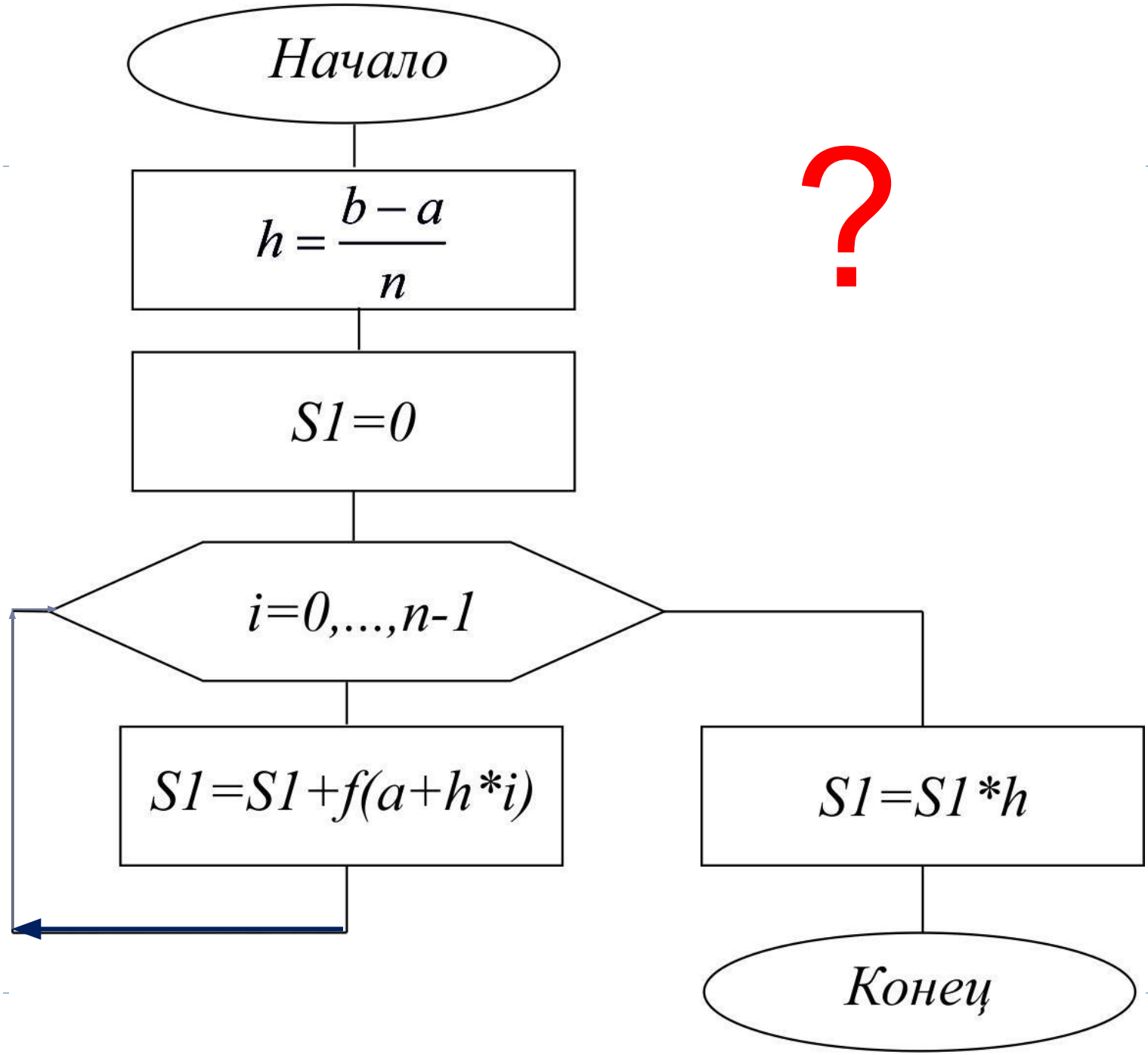
При повышении структурированности снижается сложность программ, возрастает их наглядность, что способствует сокращению числа ошибок. Однако, повышение качества программы может повлечь необходимость в дополнительной памяти и времени реализации.

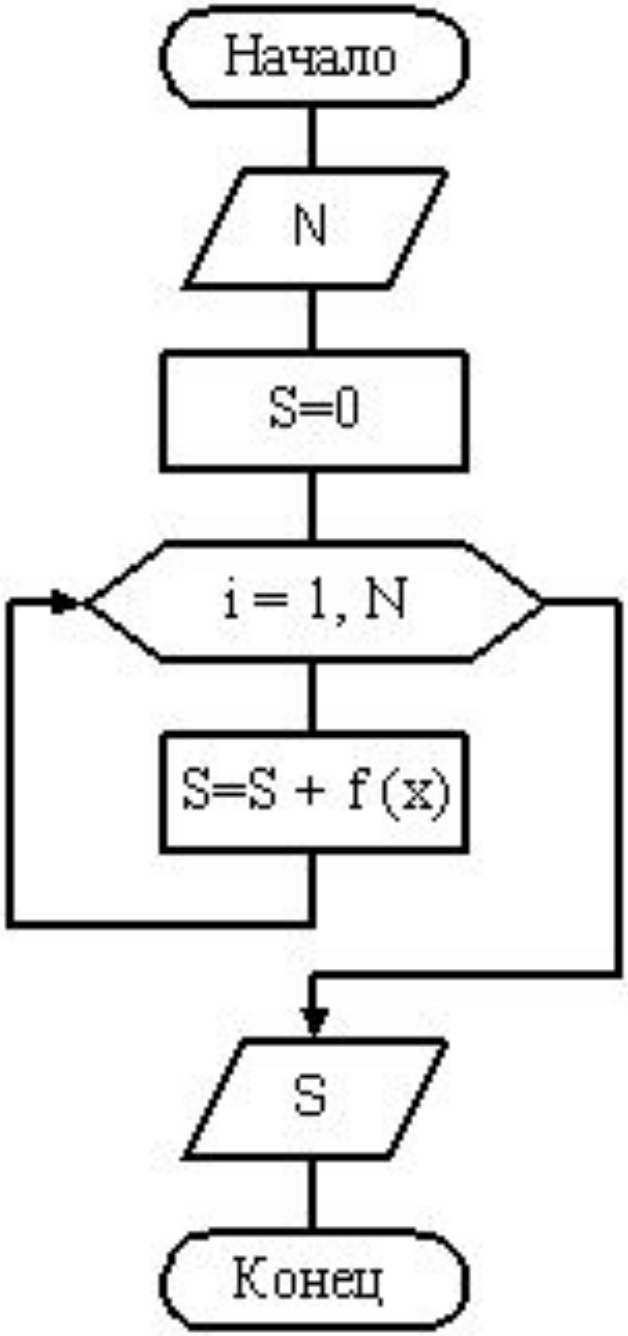
Символ	Название	Назначение
	Данные	Общее обозначение ввода или вывода данных
	Процесс	Обработка данных, операция или группа операций
	Соединитель	Соединение прерванных линий потока
	Предопределенный процесс	Вычисления по подпрограмме (модулю)
	Подготовка	Осуществляет задание изменений параметров цикла
	Решение	Проверка условия
	Терминатор	Вход или выход во внешнюю среду
	Комментарий	Для записи пояснений к алгоритму

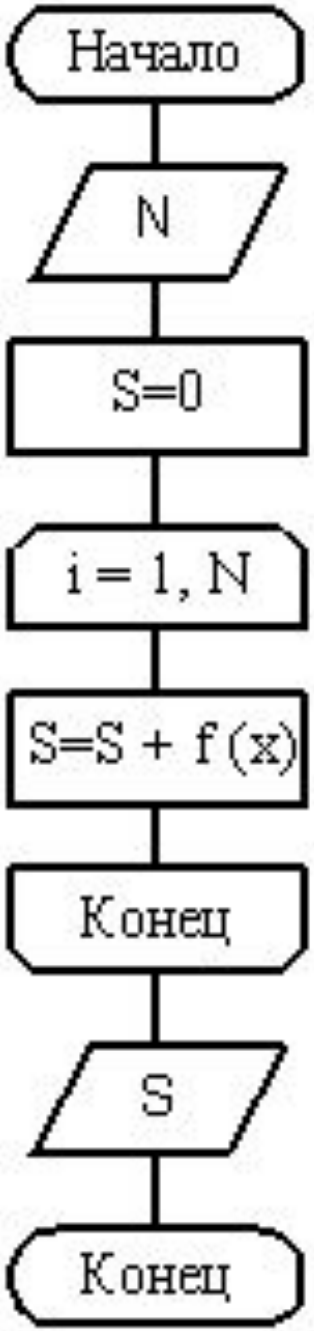
треугольник задан длинами сторон.
можно ли построить с этими данными треугольник?



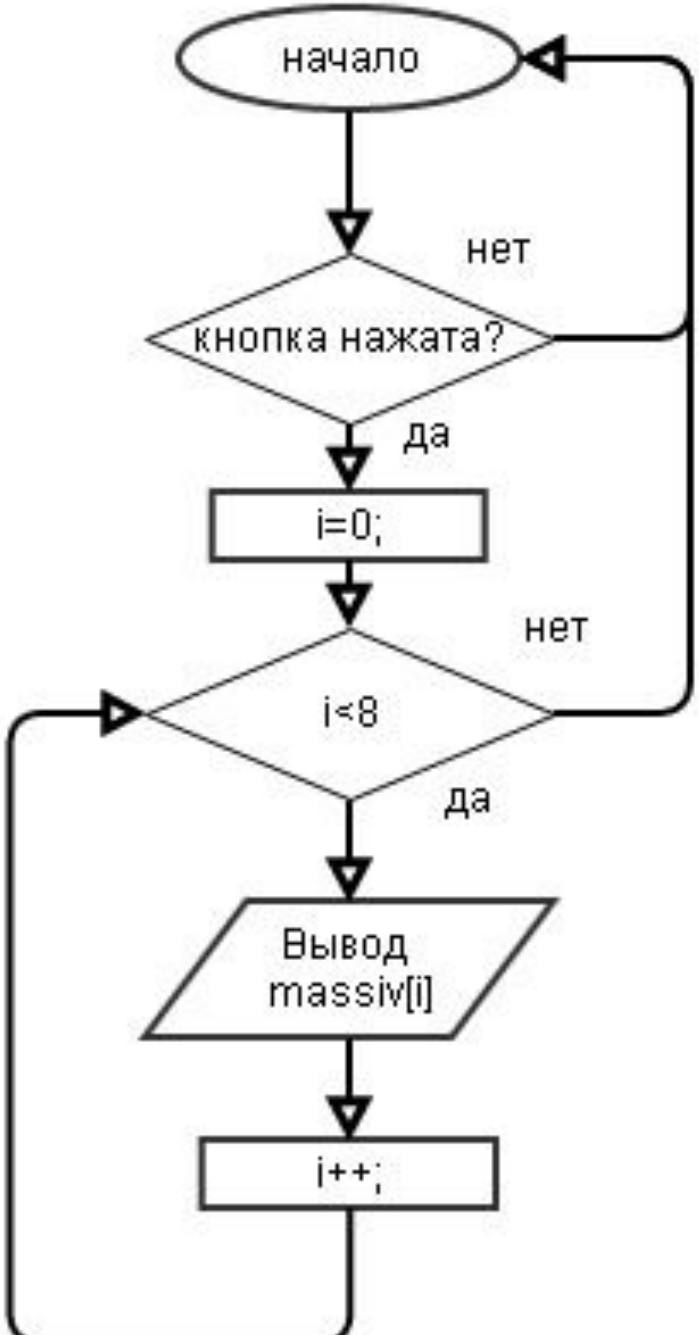

Запуск
a = 1
Цикл i
от 1 до n
a = a * i
Увеличить
i на 1
Цикл i
Вывести
a
Останов


Начало
$h = \frac{b-a}{n}$
S1=0
i=0,…,n-1
S1=S1+f(a+h*i)
S1=S1*h
Конец
?


Начало
N
S=0
i = 1, N
S=S + f(x)
S
Конец


Начало
N
S=0
i = 1, N
S=S + f(x)
Конец
S
Конец

ПЛОХО



Достоинства структурного подхода

- возможность проведения глубокого анализа бизнес-процессов, выявления «узких мест»;
- применение универсальных графических языков моделирования;
- проверенность временем и широкое распространение среди аналитиков и разработчиков.

Управление предприятием

Уровень подсистем	Управление складом	?	Финансы	?
Уровень функции	?	Приходование материалов	?	?
…	?	?	Формирование зарплатной ведомости	?
…	?	?	?	Печать оборотно-сальдовой ведомости

Недостатки структурного подхода

- низкая наглядность для неподготовленных пользователей модели;
- сложность восприятия иерархически упорядоченной информации;
- необходимость следования жесткой (не всегда необходимой) структуре.

IDEF

- (I-CAM DEFinition или Integrated DEFinition) — методологии семейства ICAM (Integrated Computer-Aided Manufacturing) для решения задач моделирования сложных систем, позволяют отображать и анализировать модели деятельности широкого спектра сложных систем в различных разрезах.
- Широта и глубина обследования процессов в системе определяется самим разработчиком, что позволяет не перегружать создаваемую модель излишними данными.

Стандарты IDEF

В настоящий момент к семейству IDEF относятся более 15 стандартов. Основные из них:

- **IDEF0** — Function Modeling — методология функционального моделирования.
- **IDEF1** — Information Modeling — методология моделирования информационных потоков внутри системы. Позволяет отображать и анализировать их структуру и взаимосвязи;
- **IDEF1X** (IDEF1 Extended) — Data Modeling — методология построения реляционных структур (баз данных), используется для моделирования реляционных БД, имеющих отношение к рассматриваемой системе;
- **IDEF2** — Simulation Model Design — методология динамического моделирования развития систем. От этого стандарта практически отказались.
- **IDEF3** — Process Description Capture — Документирование технологических процессов. Методология документирования процессов, происходящих в системе, описываются сценарий и последовательность операций для каждого процесса. IDEF3 имеет прямую связь с IDEF0 — каждая функция может быть представлена в виде отдельного процесса средствами IDEF3;
- **IDEF4** — Object-Oriented Design — методология построения объектно-ориентированных систем, позволяют отображать структуру объектов и заложенные принципы их взаимодействия, позволяя анализировать и оптимизировать сложные объектно-ориентированные системы;
- **IDEF5** — Ontology Description Capture — Стандарт онтологического исследования сложных систем. С помощью методологии IDEF5 системы может быть описана при помощи определенного словаря терминов и правил, на основании которых могут быть сформированы утверждения о состоянии рассматриваемой системы в некоторый момент времени.

Методы моделирования

В **структурном анализе** используются группы средств, иллюстрирующих функции, выполняемые системой, и отношения между данными.

Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются:

–**SADT** (Structured Analysis and Design Technique) - модели и соответствующие функциональные диаграммы (подмножеством SADT является IDEF0);

–**DFD** (Data Flow Diagrams) – диаграммы потоков данных;

–**ERD** (Entity-Relationship Diagrams) – диаграммы«сущность-связь».

Модели структурного подхода

3 типа моделей, используемых в структурном подходе:

- функциональные модели (ФМ)
- информационные модели (ИМ)
- динамические модели (ДМ)

ФМ	SADT (IDEF0)-модели DFD-модели	ПакетыBPWin, Design/IDEF
ИМ	ERD (IDEF1X)	Design/IDEF, ERWin
ДМ	IDEF/CPN IDEF3	Design/IDEF Пакет BPWin

Будем работать в DRAW.IO

Наиболее распространенные виды диаграмм

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** (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)

Методология структурного анализа и проектирования

- 70-е гг. XX века – методология **SADT**
- Предложена *Дугласом Россом* (Douglas Ross)
- Основная идея данной методологии – построение древовидной иерархической модели предприятия(ПП).

SADT.Основные понятия и правила

Стандарт IDEF0

В начале *1990-х* на основе SADT принят стандарт моделирования бизнес-процессов **IDEF0**, являющийся одним из 14 стандартов линейки *IDEF – Integration Definition for Functional Modeling*

- Результат программы автоматизации промышленных предприятий ICAM (Integrated Computer Aided Manufacturing)
- (IDEF=ICAM DEFinition)

Положения методологии зафиксированы в разработанном в США стандарте IDEF0 (*РД IDEF0 – 2000*)

Методология SADT

Методология SADT - совокупность методов, правил и процедур, предназначенных для построения **функциональной модели** объекта какой-либо предметной области.

SADT – **графические** обозначения и подход к **описанию систем**.

Функциональная модель SADT отображает **функциональную структуру** объекта, т.е. производимые им действия и связи между этими действиями.

Методология SADT

- **формализация** и **описание** бизнес-процессов;
- акцент на **соподчинённость объектов;**
- рассматриваются **логические отношения между работами, а не** их **временная последовательность.**

Модель в 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.

Сущность функционального моделирования

Для любой системы определяющим является ее функциональное содержание, так как оно определяет ее основные свойства.

- В основе функционального моделирования - ***функциональное содержание*** системы.
- В качестве отношений между функциями рассматривается ***информация об объектах***, связывающих эти функции.

Функциональное моделирование

Модель может быть сосредоточена либо на *функциях системы*, либо на ее *объектах*.

Модели, ориентированные на функции, принято называть *функциональными моделями*.

Функциональное моделирование является важнейшим элементом анализа, который выполняется **на начальном этапе** проектирования любой автоматизированной информационной системы.

Функциональная модель

Функциональная модель позволяет решать целый ряд *задач*, связанных с:

- оптимизацией ;
- оценкой и распределением затрат;
- оценкой функциональной производительности;
- загрузки и сбалансированности составных частей;

т. е. вопросов *анализа и реинжиниринга бизнес-процессов* (Business Process Reengineering, BPR).

Методология IDEF0

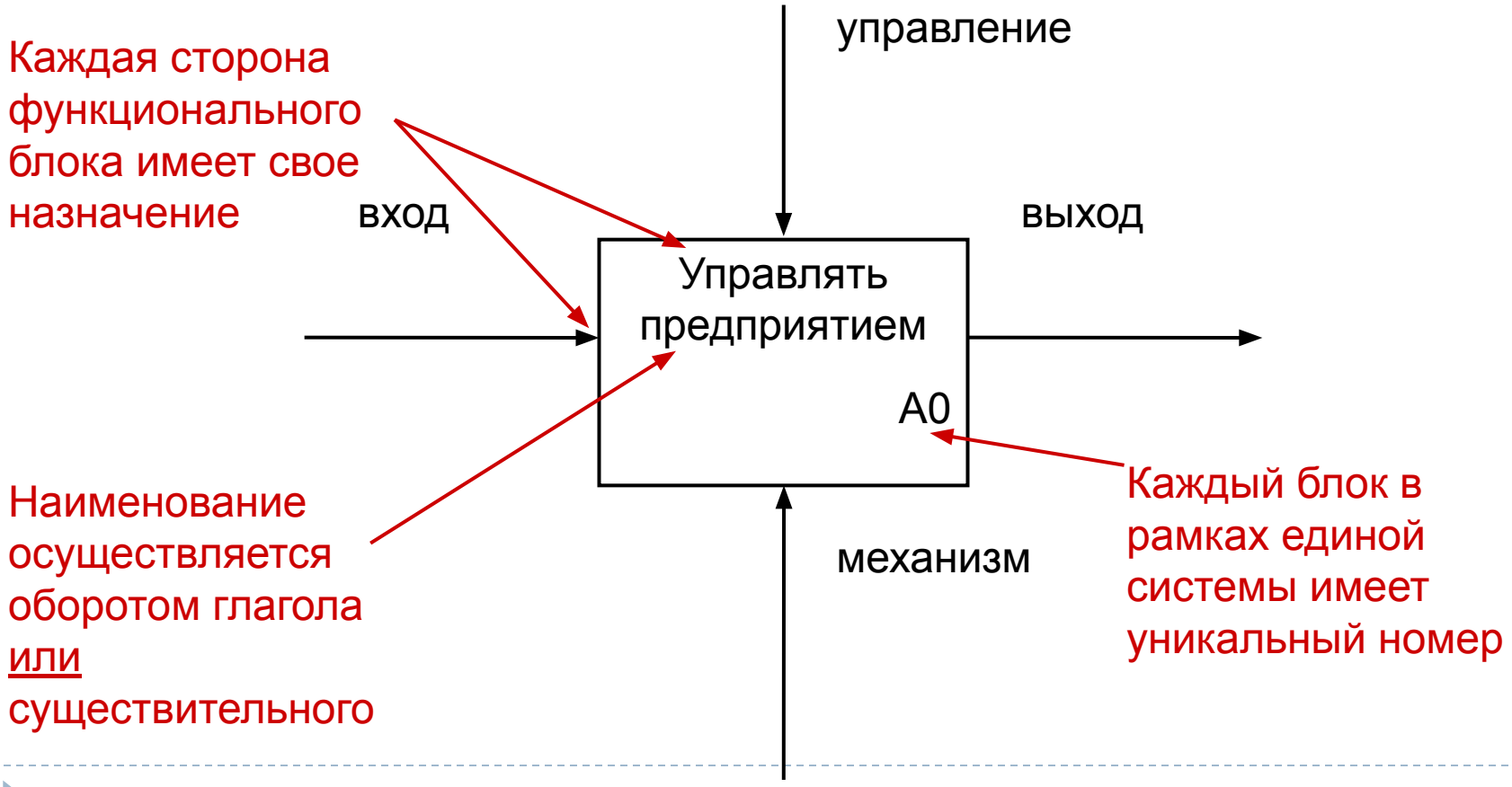
Методология IDEF0

В основе *IDEF0*-методологии лежат **4 основных понятия**:

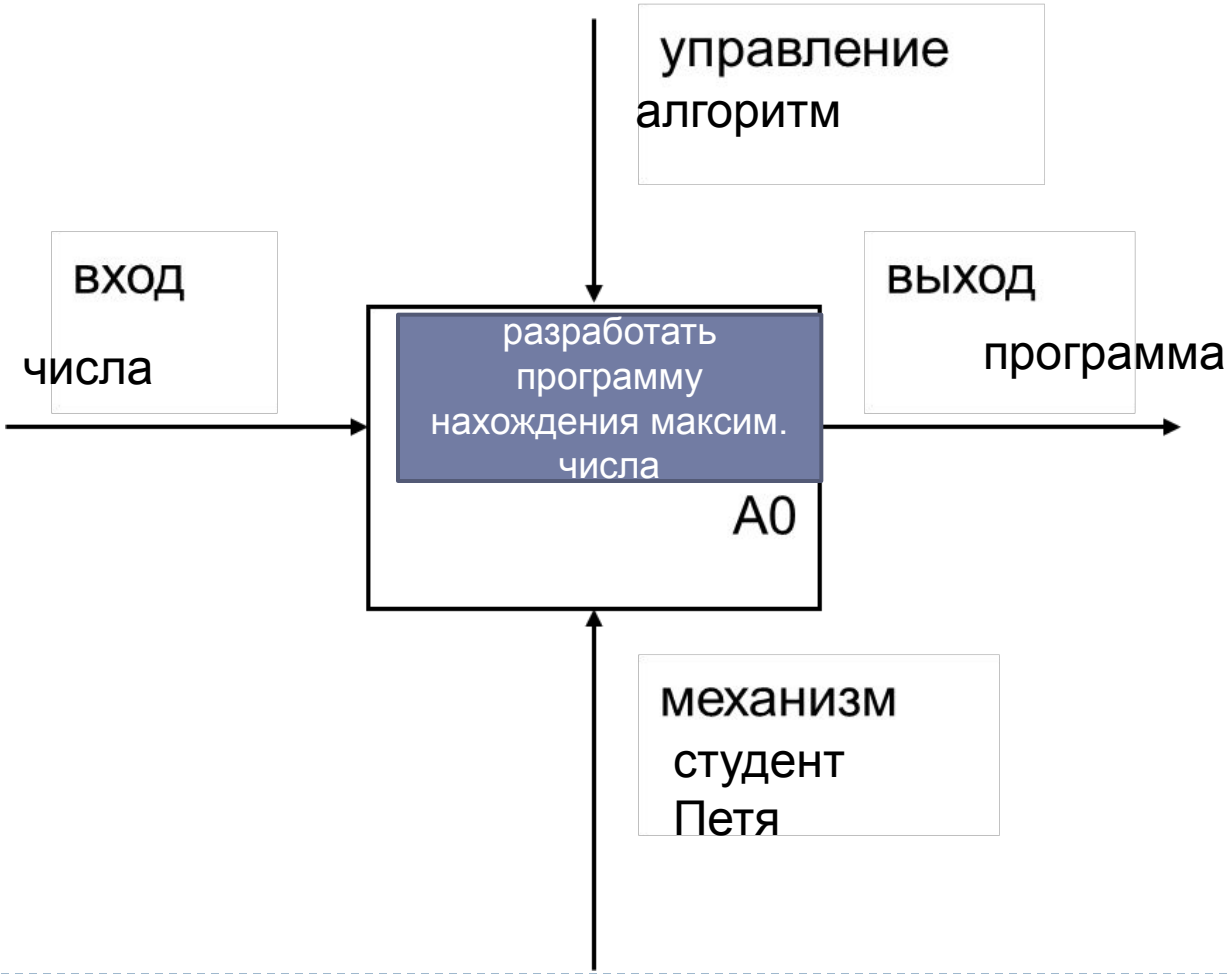
- 1) функциональный блок;
- 2) интерфейсная дуга (стрелка);
- 3) декомпозиция;
- 4) глоссарий.

Функциональный блок (Activity box)

- Олицетворяет некоторую конкретную функцию или работу в рамках рассматриваемой системы
- *РД IDEF0 – 2000*: прямоугольник, содержащий имя и номер и используемый для описания функции



пример


управление
алгоритм
вход
числа
разработать программу нахождения максим. числа
A0
выход
программа
механизм
студент
Петя

Интерфейсная дуга (Arrow)

- Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображаемую функциональным блоком.

- *Графически* изображается в виде однонаправленной стрелки.

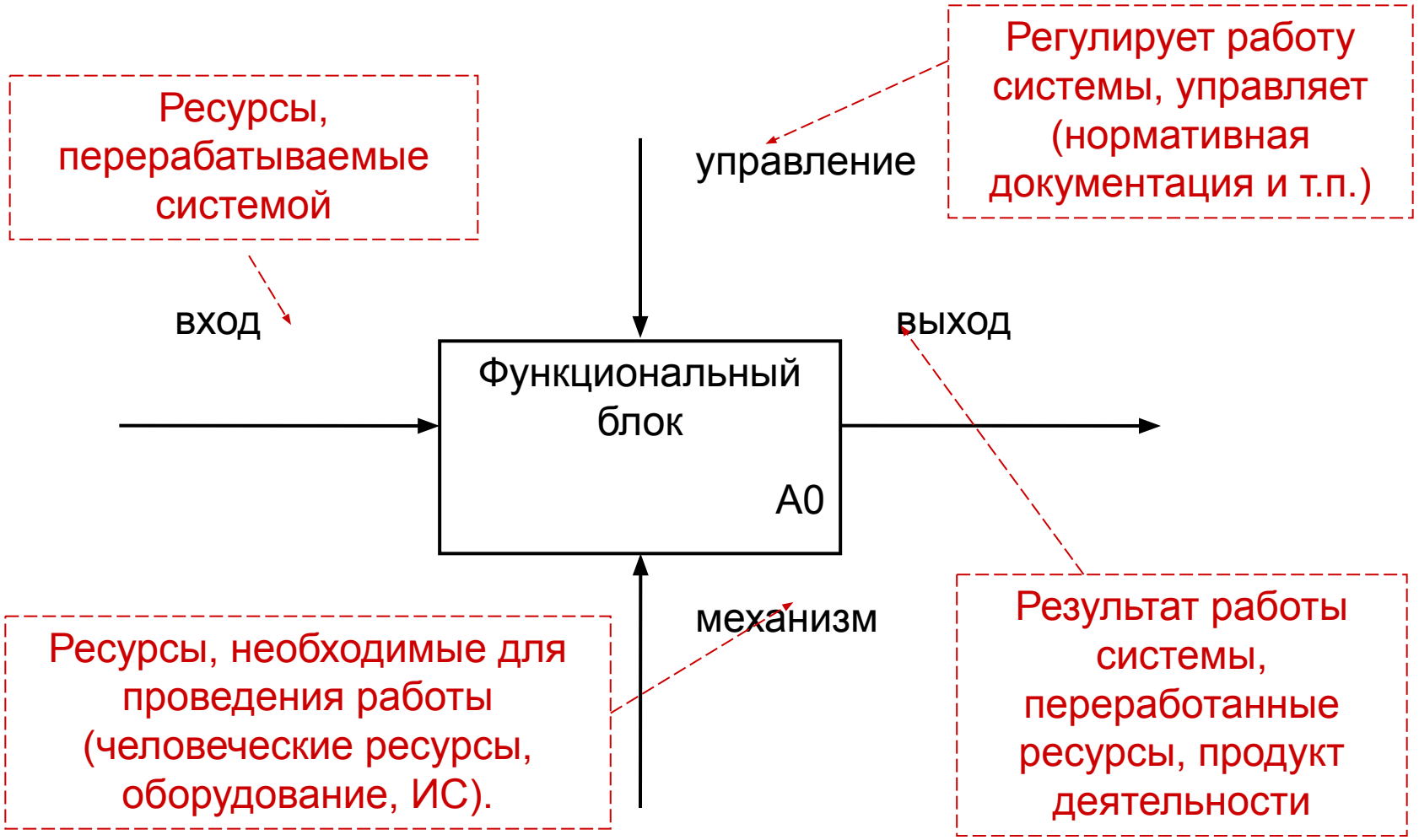
Интерфейсная дуга

- Каждая дуга должна иметь свое уникальное *название*, сформулированное оборотом существительного (должно отвечать на вопросы кто?, что?).

  Примеры: информация, разработчик, документ, обработанная заявка.

- В зависимости от того, к какой стороне блока она подходит, интерфейсная дуга будет являться ***входящей, выходящей, управления, механизма***.

Интерфейсная дуга



Стрелки входа может не быть. Остальные интерфейсные дуги обязательны.

Декомпозиция

Принцип декомпозиции применяется при разбиении сложных процессов на составляющие его функции.

При этом уровень детализации определяется непосредственно разработчиком модели.

Декомпозиция

- Модель IDEF0 всегда начинается с рассмотрения системы как единого целого, т.е. одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области.
- Такая диаграмма называется *контекстной*, она обозначается идентификатором A-0.
- Для определения границ системы на контекстной диаграмме обязательно должны быть цель и точка зрения.

Цель моделирования (Purpose)

Цель моделирования должна отвечать на **вопросы**:

- Почему процесс должен быть смоделирован?
- Что должна показывать модель?
- Что может получить читатель?

Примеры целей:

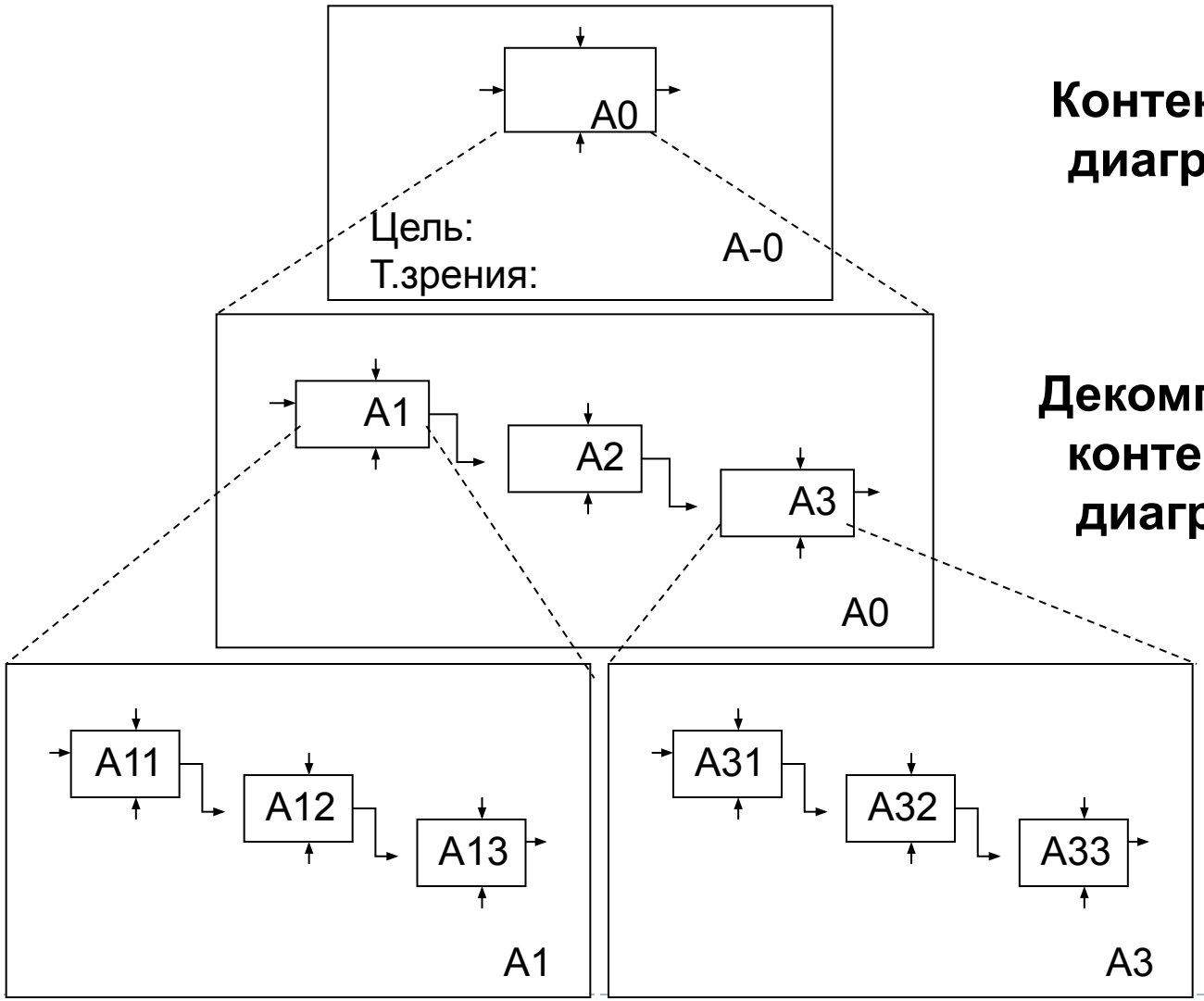
«Идентифицировать слабые стороны процесса сбора данных», «Определить ответственность сотрудников для написания должностных инструкций» и т.п.

Точка зрения (ViewPoint)

- **Точка зрения** – позиция, с которой будет строиться модель.

  В качестве точки зрения берется взгляд человека, который видит систему **в необходимом для моделирования аспекте**.
- Как правило, выбирается точка зрения **человека**, **ответственного за выполнение** моделируемой работы.
- Между **целью и точкой зрения** должно быть **жесткое соответствие**.

Декомпозиция



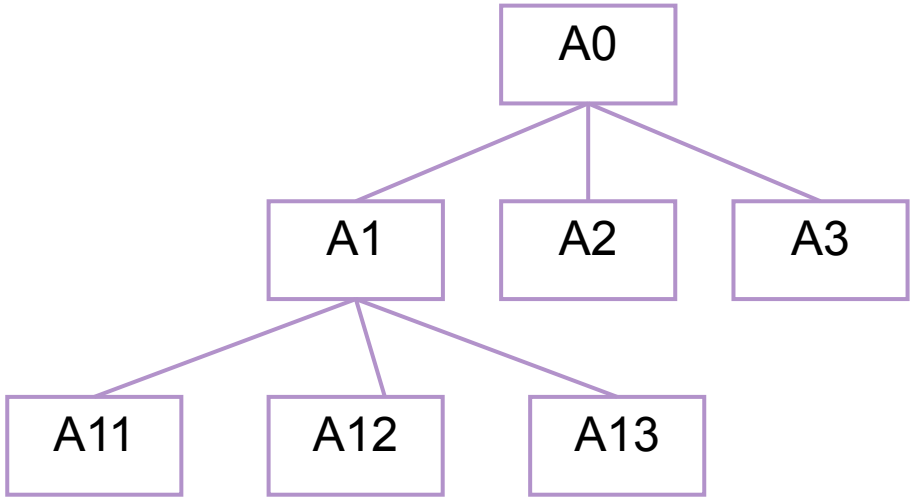
Контекстная диаграмма

Декомпозиция контекстной диаграммы

Декомпозиция блока A1

Декомпозиция блока A3

Декомпозиция

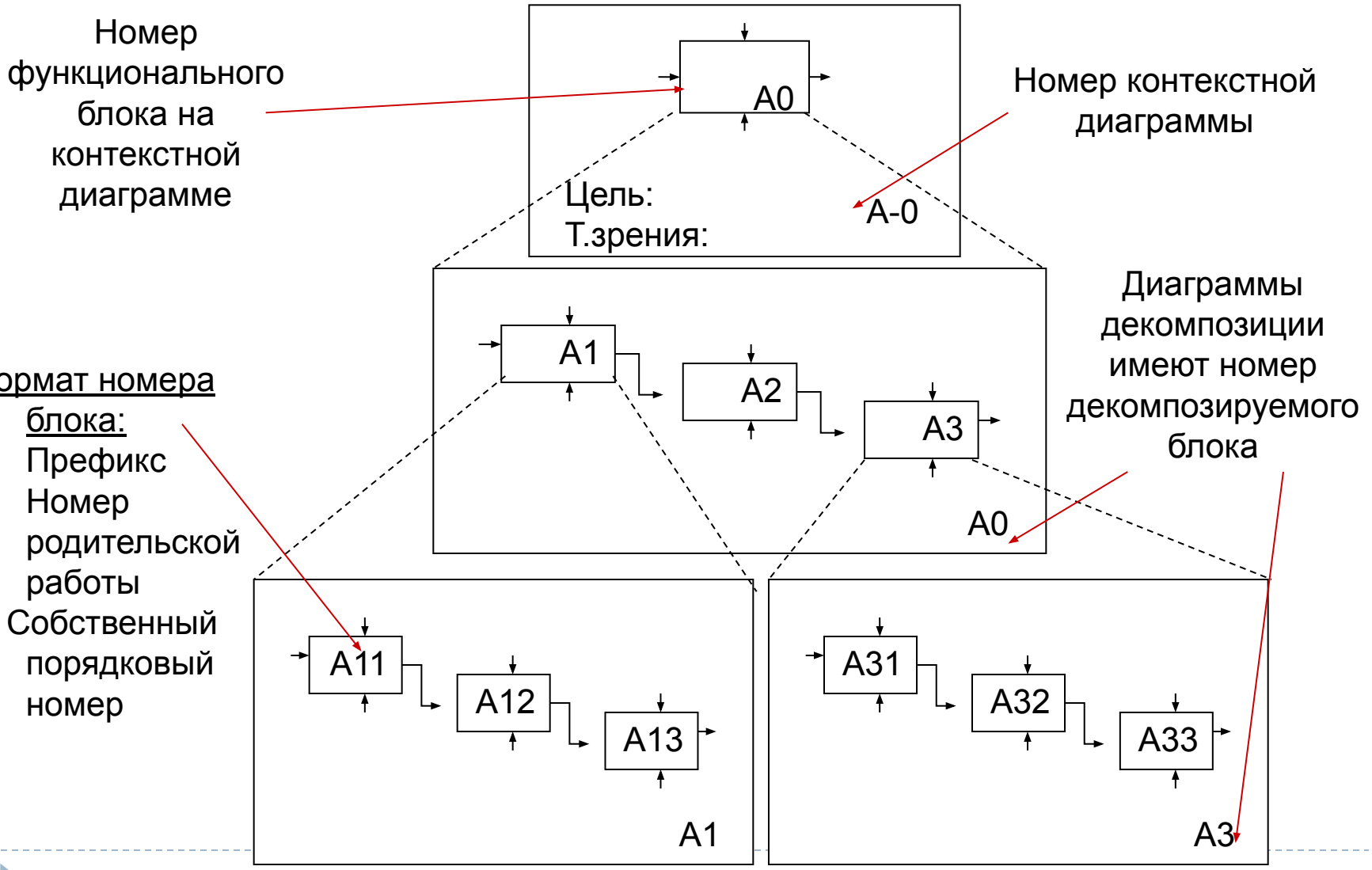


Дерево узлов

A0 _____
A1_____
A11_____
A12_____
A13_____
A2_____
A3_____

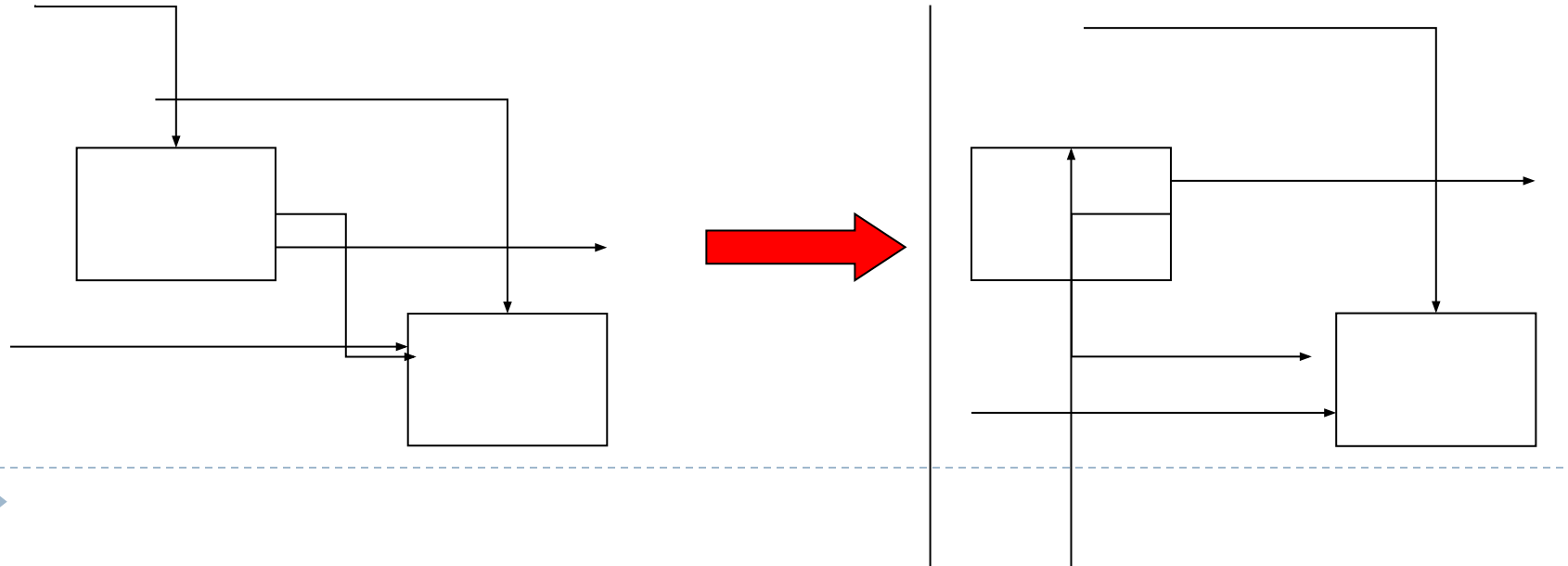
Индекс узлов

Нумерация работ и диаграмм



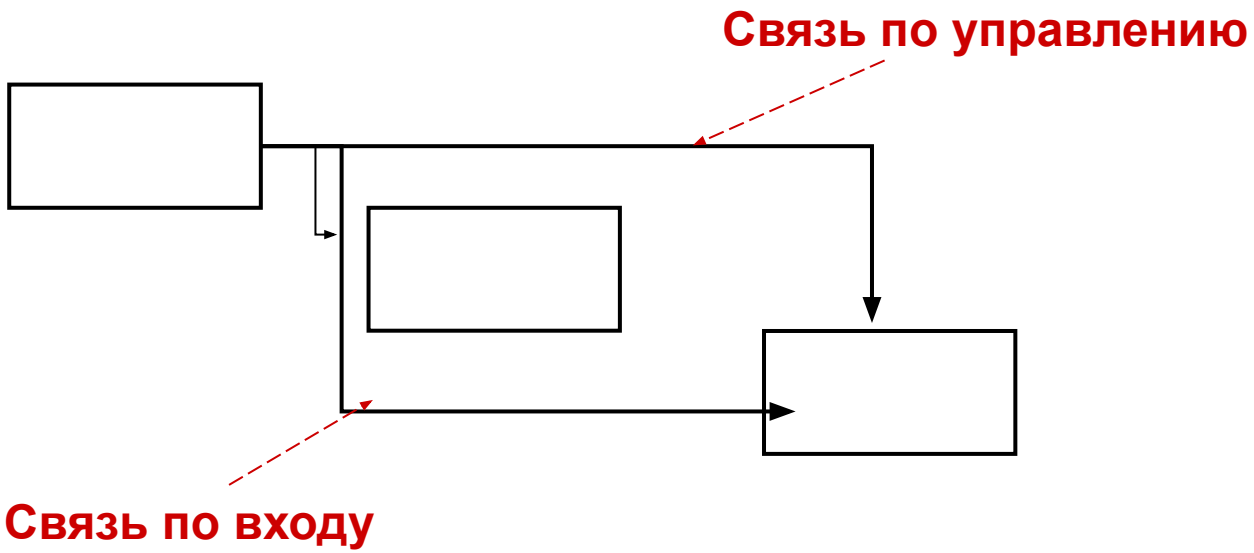
Основные правила построения диаграмм

1. На одной диаграмме рекомендуется рисовать от 3 до 6 блоков. Иначе диаграмма будет плохо читаемой.
2. Функциональные блоки должны располагаться слева направо сверху вниз в порядке доминирования.
3. Следует избегать излишнего пересечения стрелок.

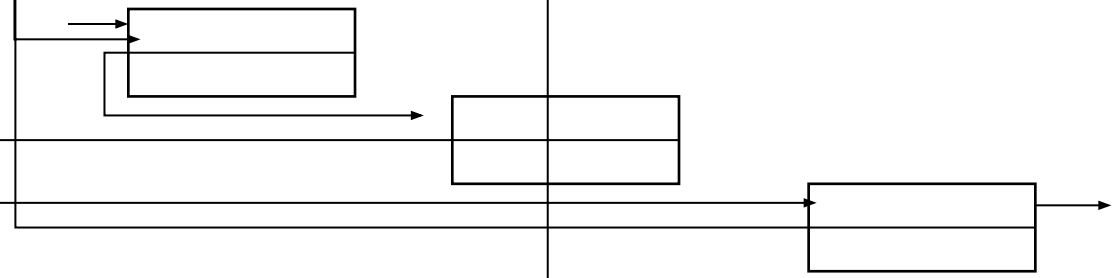


Основные правила построения диаграмм

4. Выход одного блока может являться входом (управлением) для другого. Могут быть и обратные связи по входу и управлению.

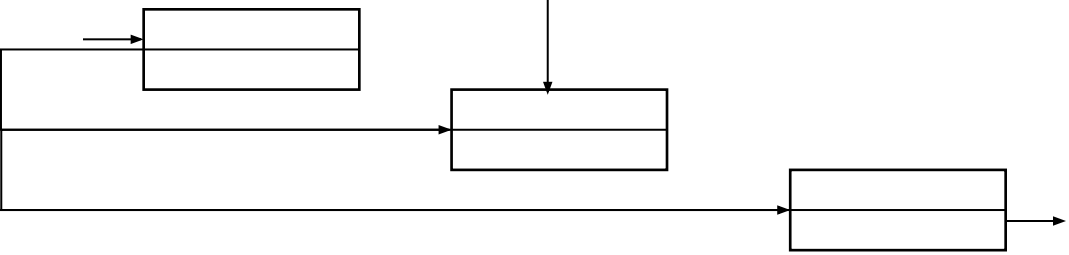


Основные правила построения диаграмм



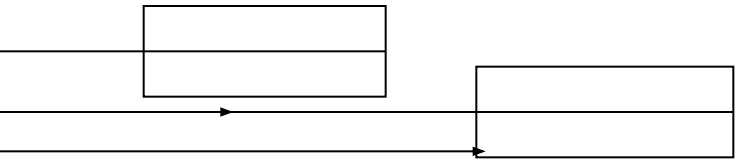
а) обратная связь по входу

Обратная связь по входу, как правило, используется для описания циклов.



б) обратная связь по управлению

Обратная связь по управлению – выход нижестоящей работы передается на управление вышестоящей

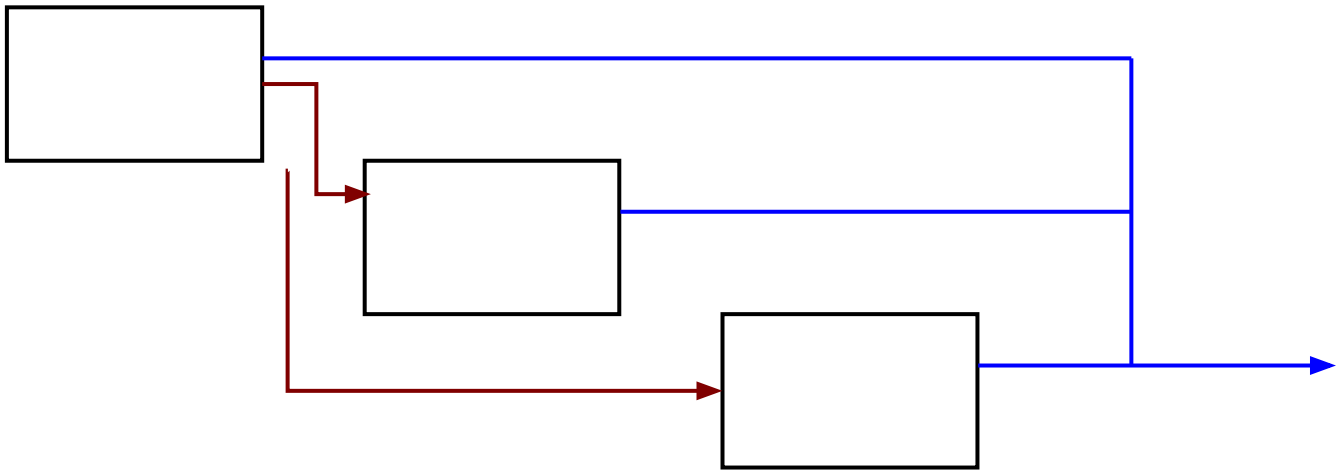


в) обратная связь по механизму

Обратная связь по механизму – выход нижестоящей работы создает ресурсы, выполняющие вышестоящую работу

Основные правила построения диаграмм

5. Стрелки могут быть сливающимися и разветвляющимися



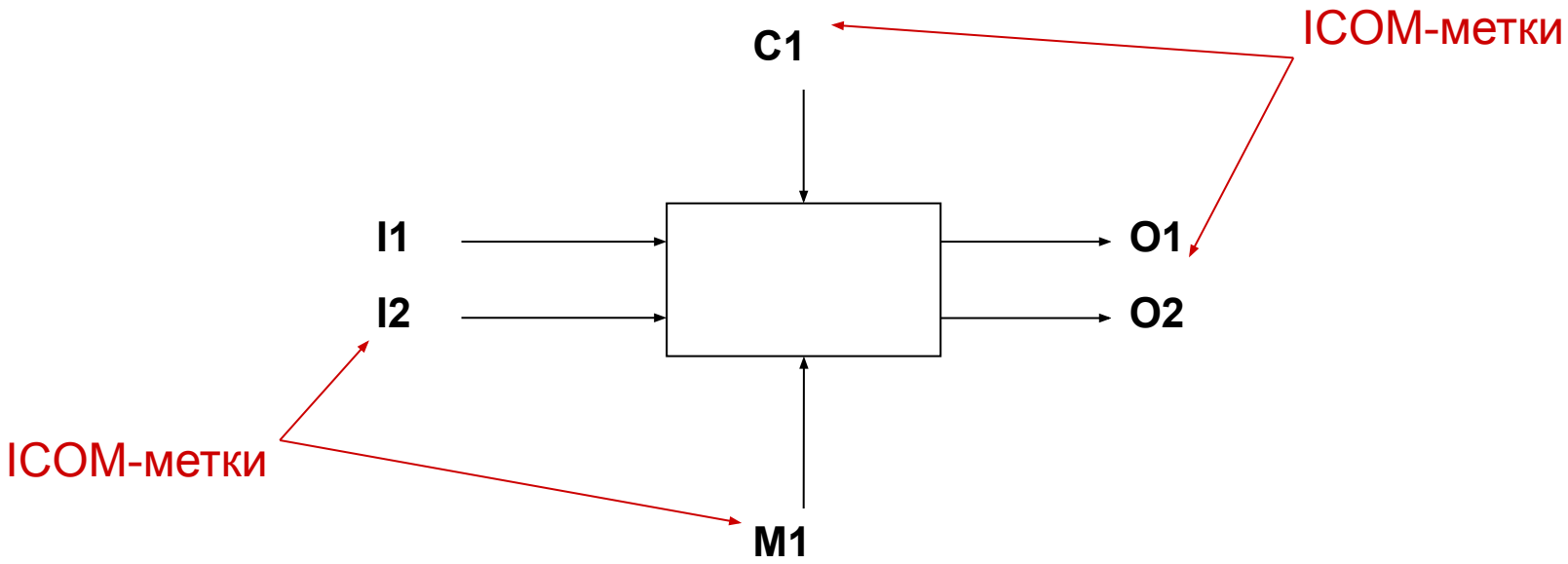
Слияние стрелок

Разветвление стрелок

Граничные стрелки

Стрелки на контекстной диаграмме служат для описания **взаимодействия системы с окружающим миром**. Они могут начинаться у границы диаграммы и заканчиваться у функционального блока и наоборот. Такие стрелки называются ***граничными .***

Граничные стрелки помечаются с помощью **ICOM-меток** (Input, Control, Output, Mechanism)



Тоннельные стрелки (Arrow Tunnel)

Иногда необходимо отобразить граничные стрелки, которые **значимы на данном уровне** и **не значимы на родительской** диаграмме.

Например, некоторые данные используются только на данном уровне и не используются на других.

Без использования механизма **тоннелирования** малозначимая стрелка появится **на всех уровнях** модели, что затруднит чтение диаграмм.

Глоссарий

Для каждого из элементов в IDEF0 существует *стандарт,* подразумевающий создание и поддержку набора соответствующих определений, ключевых слов, повествований, изложений и т.д, которые характеризуют объект, отраженный данным элементом.

Этот набор – **глоссарий**, являющийся *описанием сущности* данного элемента.

FEO-страница

FEO-диаграмма (*For Exposition Only*) – это *диаграмма*, которая поясняет особо интересные и тонкие *аспекты* диаграмм.

Эти диаграммы не ограничены синтаксисом IDEF0. В них может быть текстовая, графическая информация, схемы, альтернативная точка зрения на процесс и т.п.

FEO – диаграмма

FEO – диаграмма может быть использована для:

- упрощения чтения диаграмм модели их потребителями,
- выявления тех или других особенностей диаграммы разработчиком,
- анализа корректности диаграмм и модели в целом – разработчиком
- для других целей

Мастерская страница (каркас диаграммы)

- Стандартный бланк для диаграмм (облегчает подшивку и копирование)
- Разделен на 3 основные части:

1) **поле рабочей информации**

- для отслеживания диаграммы в процессе моделирования

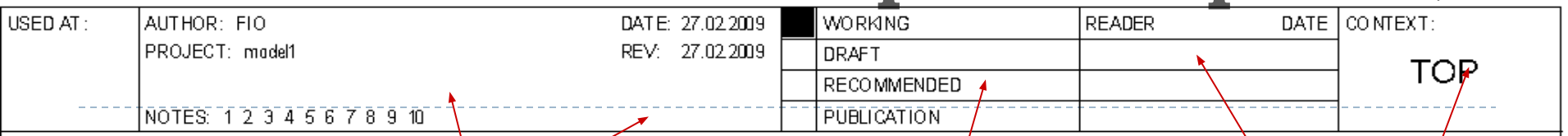
2) **поле сообщений**

- область рисования диаграммы

3) **поле идентификации**

- идентификация диаграммы и ее позиционирование в иерархии

Мастерская страница



Поле рабочей информации

Сведения о модели:

-автор;

-название проекта;

-замечания;

-дата создания и пересмотра.

Статусы проекта:

1) *Рабочая версия* – диаграмма с большим числом изменений на стадии разработки

2) *Эскиз* имеет меньше изменений и свидетельствует о достижении некоторого согласия ряда читателей

3) *Рекомендовано* – сопутствующие тексты утверждены

4) *Публикация* – материал может печататься.

Сведения о читателях и экспертах и дате экспертизы

Сведения о родительской работе

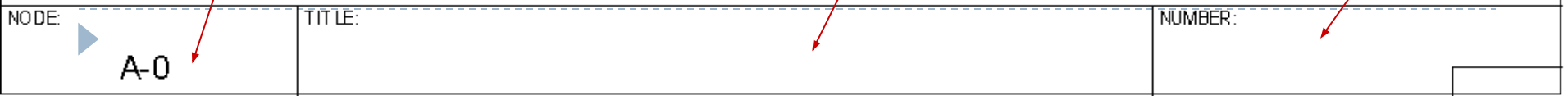
Поле сообщений

Номер диаграммы

Поле идентификации

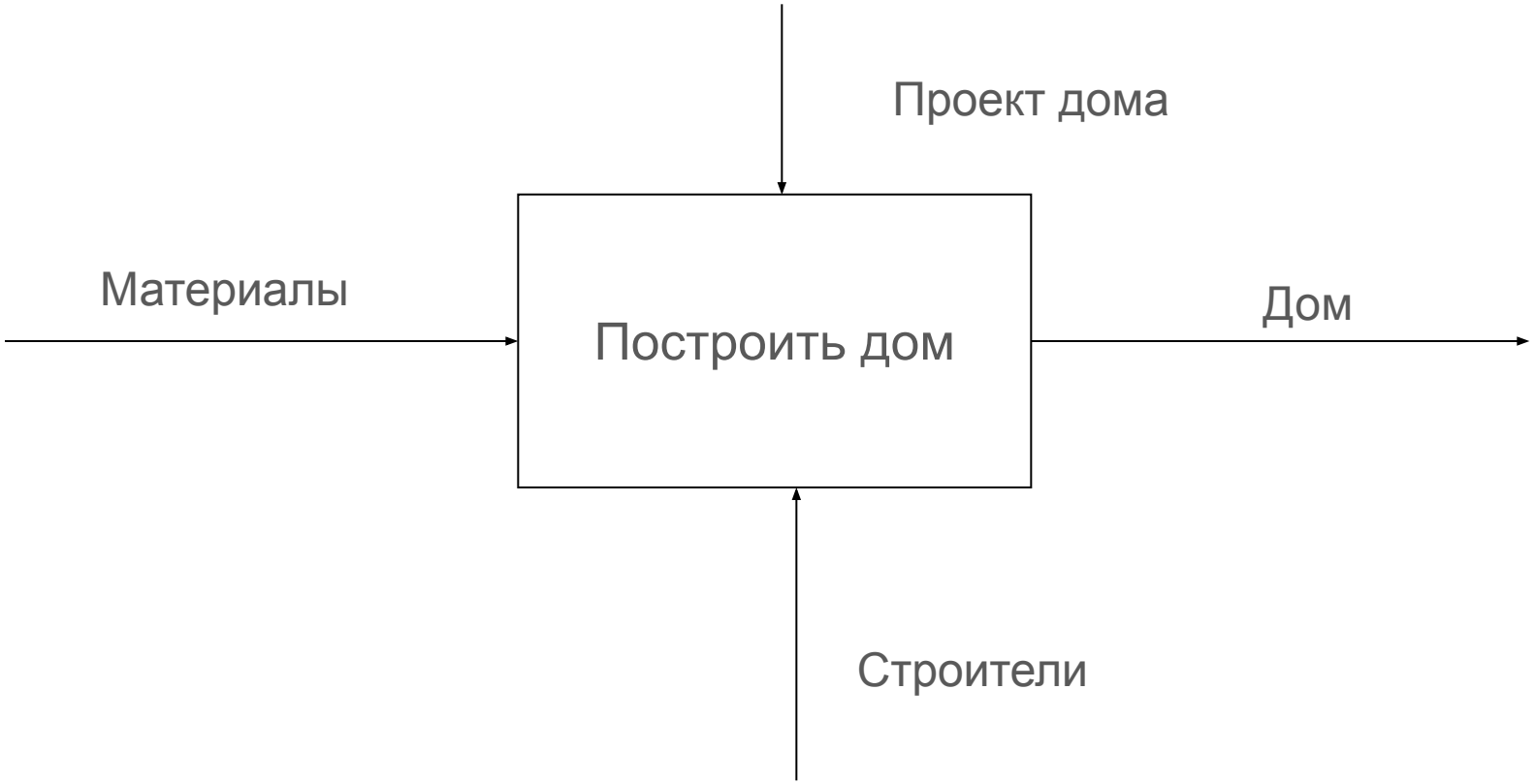
Название диаграммы (совпадает с названием родительской работы)

Уникальный номер версии диаграммы



Пример модели процесса постройки садового домика

1. Строим контекстную диаграмму.

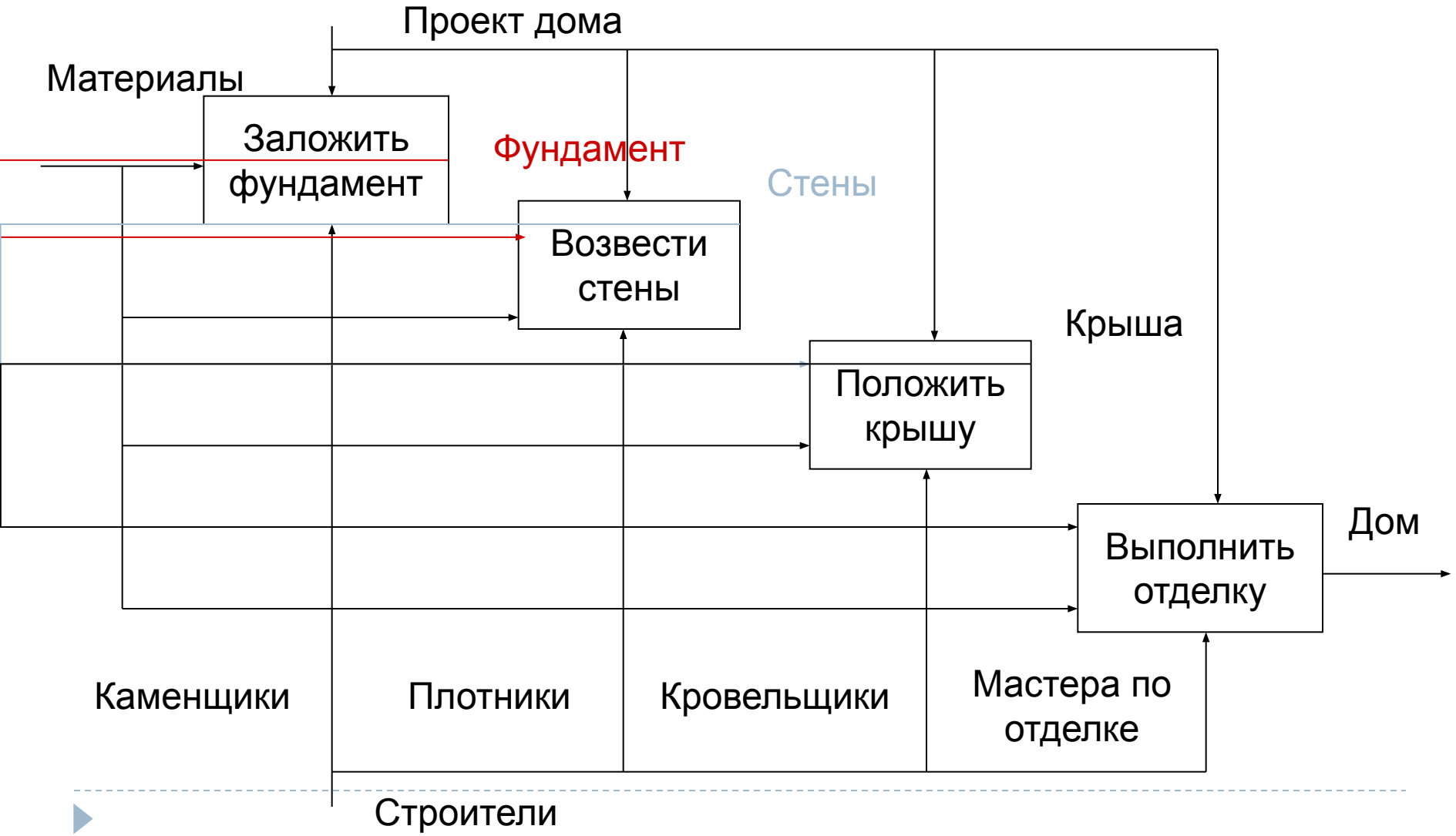


Цель: Определить действия, необходимые для постройки дачного домика

Точка зрения: владельца дачного участка

Пример модели процесса постройки садового домика

2. Декомпозируем контекстную диаграмму



Пример модели, построенной с использованием CASE-средства BPWin

Пример модели, построенной с использованием CASE-средства BPWin

Дерево узлов (Node Tree)

FEO-страница

Пример

Система учета выдачи книг в библиотеке

Описание информационной системы:

Администратор данной системы должен вести учет книжного фонда библиотеки. В его функции входит: управление пользователями системы (создание, удаление, редактирование), управление книжным фондом (ввод данных о поступающих книгах), удаление данных о списанных книгах. Каждый пользователь характеризуется: ФИО, пароль доступа. Каждая книга характеризуется: ФИО автора, название, издательство, год издания, количество страниц, месторасположение. Пользователем системы является библиотекарь, который может создавать записи абонементов библиотеки и осуществлять регистрацию выдачи и возврата книг в библиотеку на абонемент. Абонемент характеризуется следующими полями: ФИО, паспортные данные, адрес, контактный телефон. Акт выдачи или возврата книги описывается датой, абонементом, книгой, и пользователем, осуществившим эту запись. Дополнительно система должна предоставлять: отчет о выдаче определенной книги и отчет по определенному абонементу. Доступ администратора и пользователей к системе осуществляется после процедуры аутентификации. Ввод данных о выдаче и возврате книг должен осуществляться с авторизацией.

Контекстная диаграмма

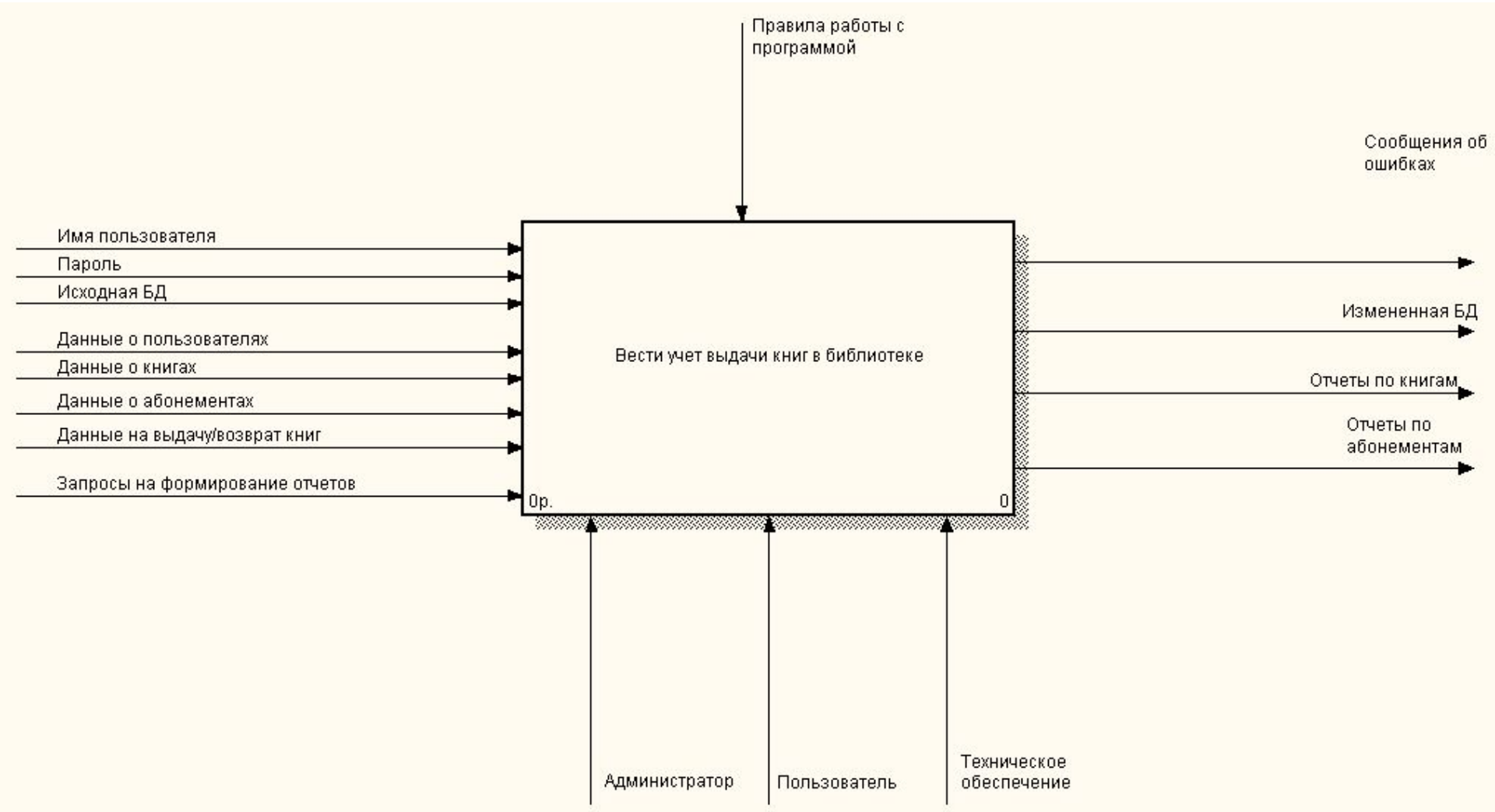


Диаграмма A0

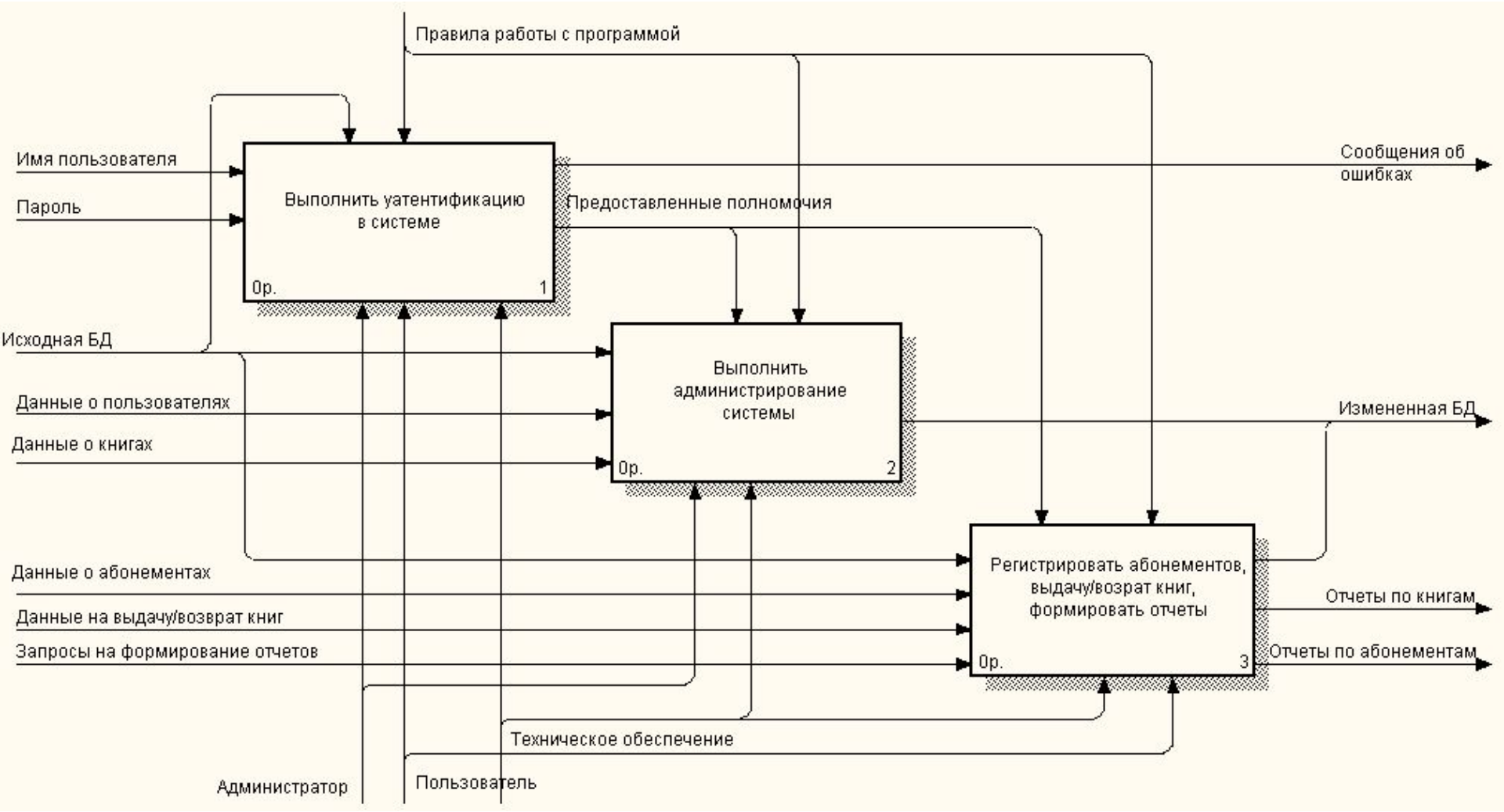


Диаграмма A1

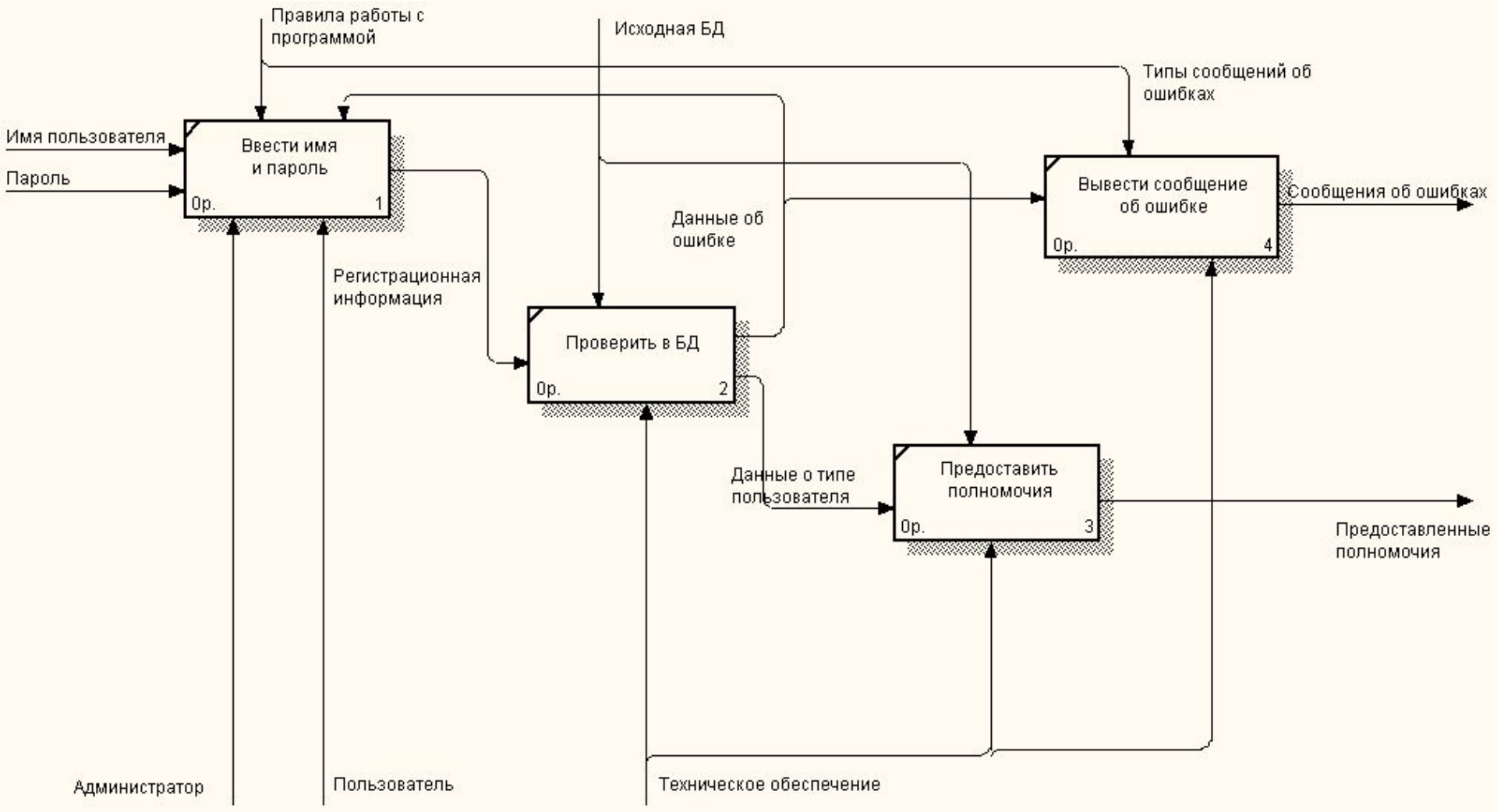


Диаграмма A2

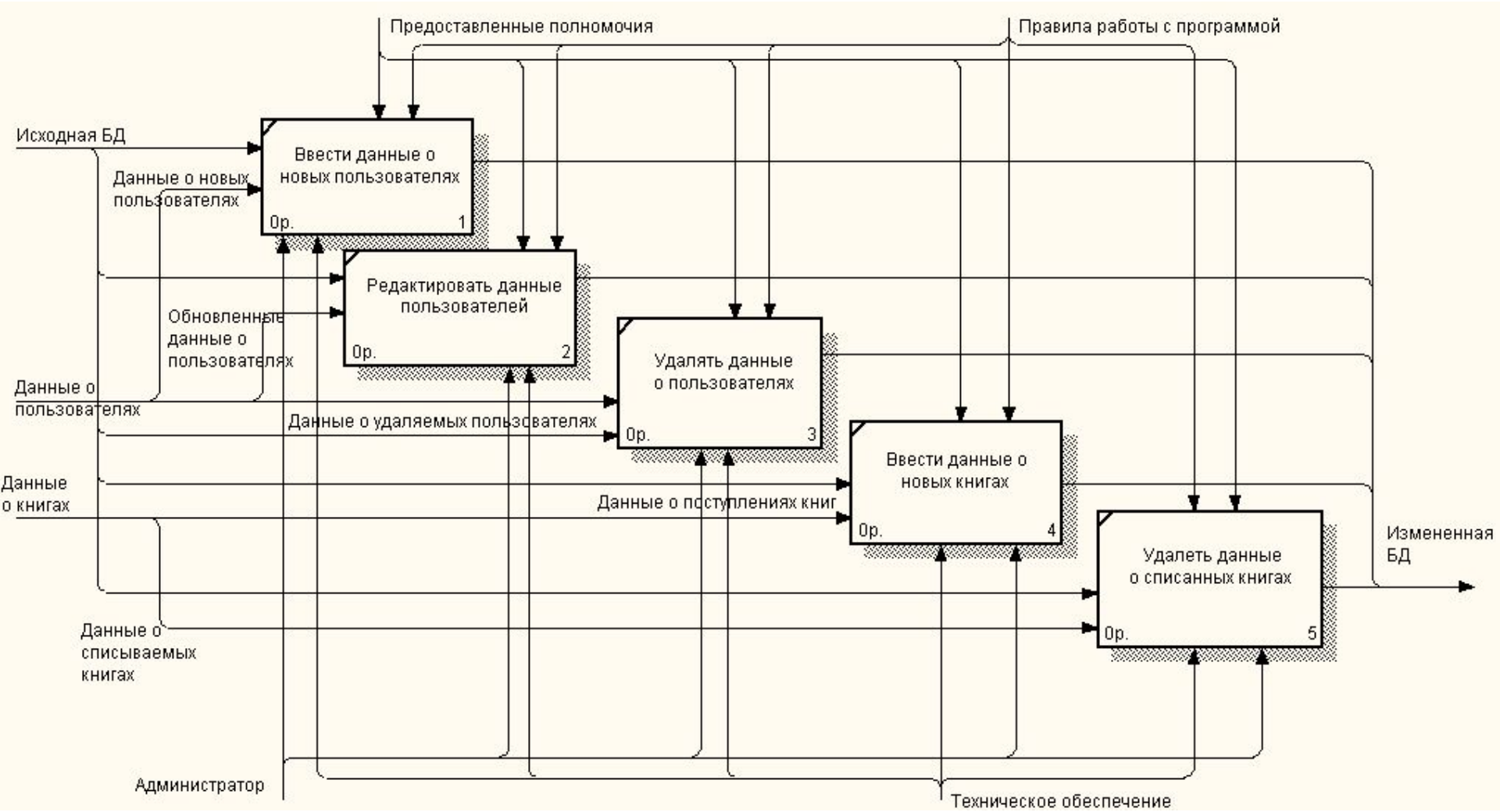
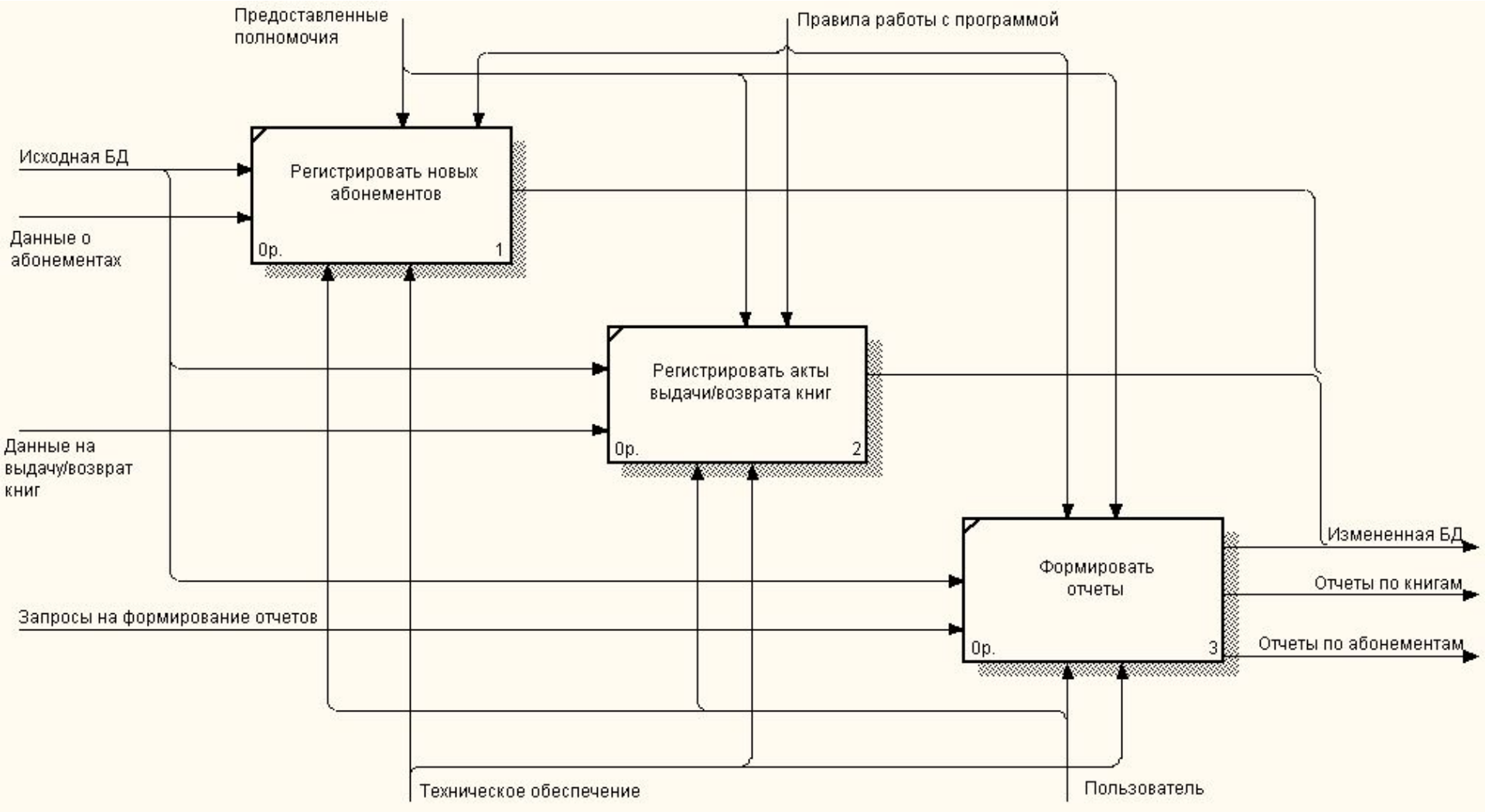
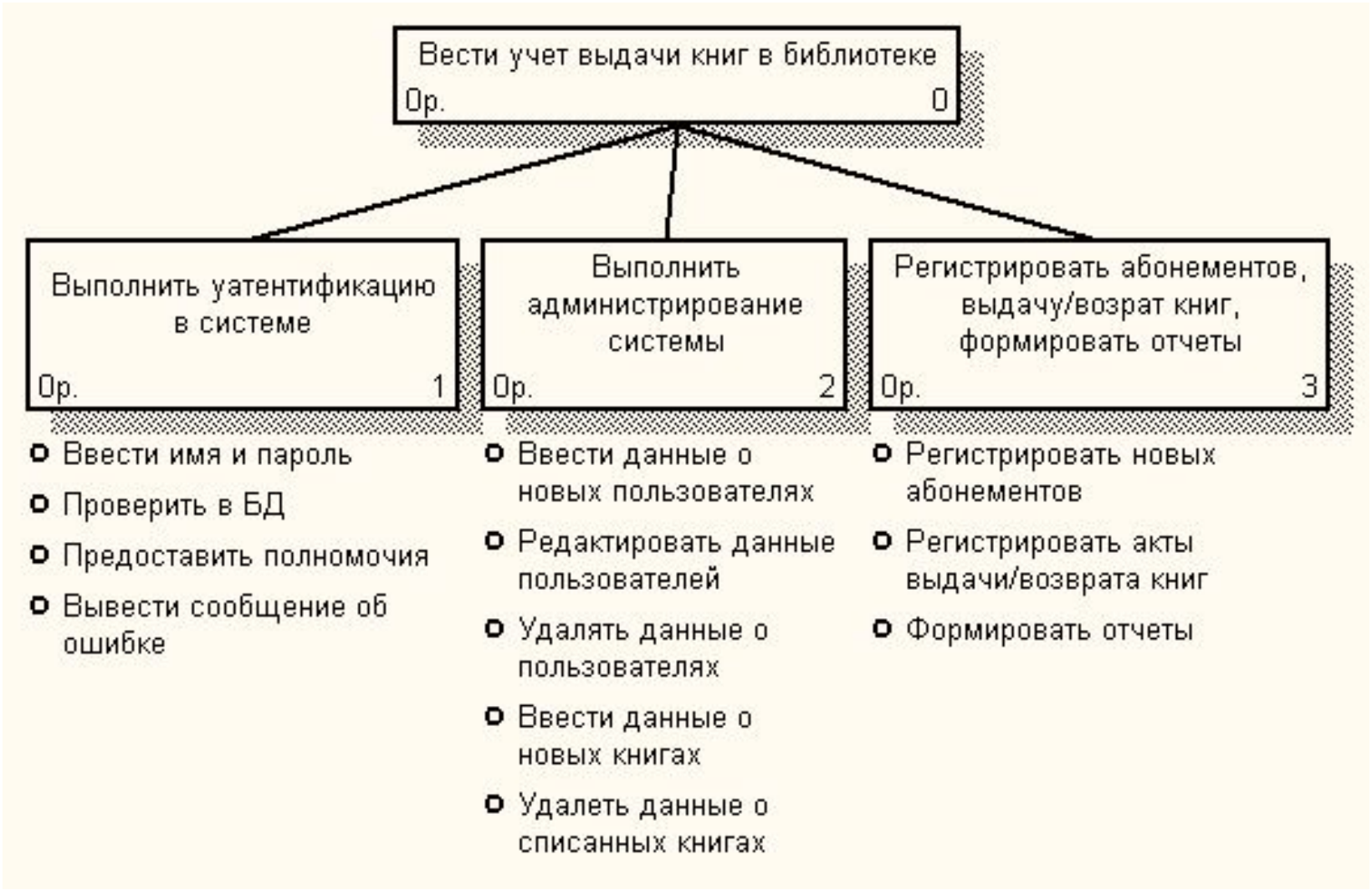


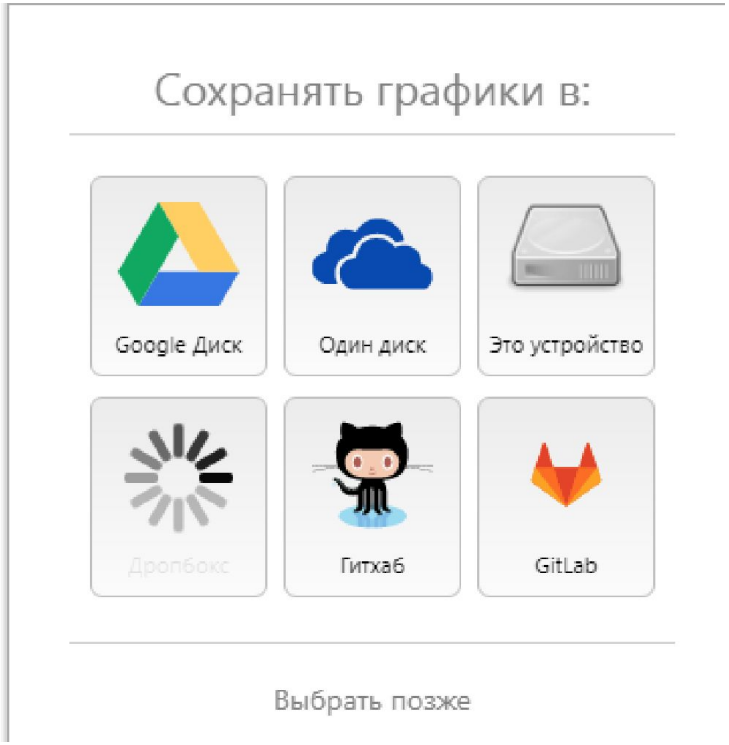
Диаграмма А3

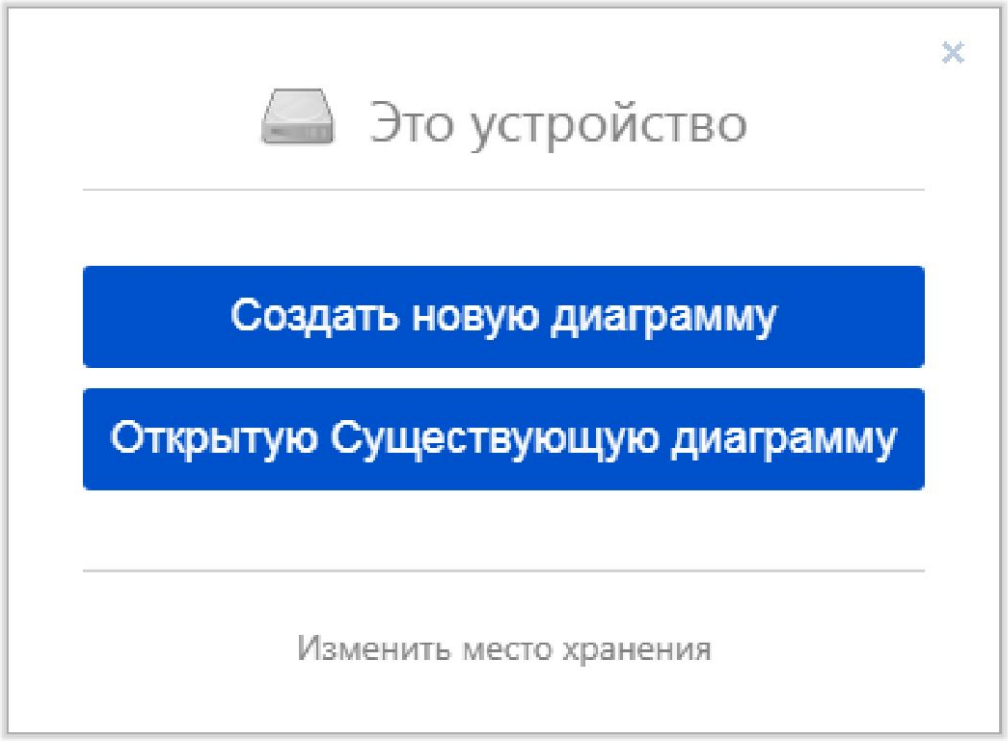


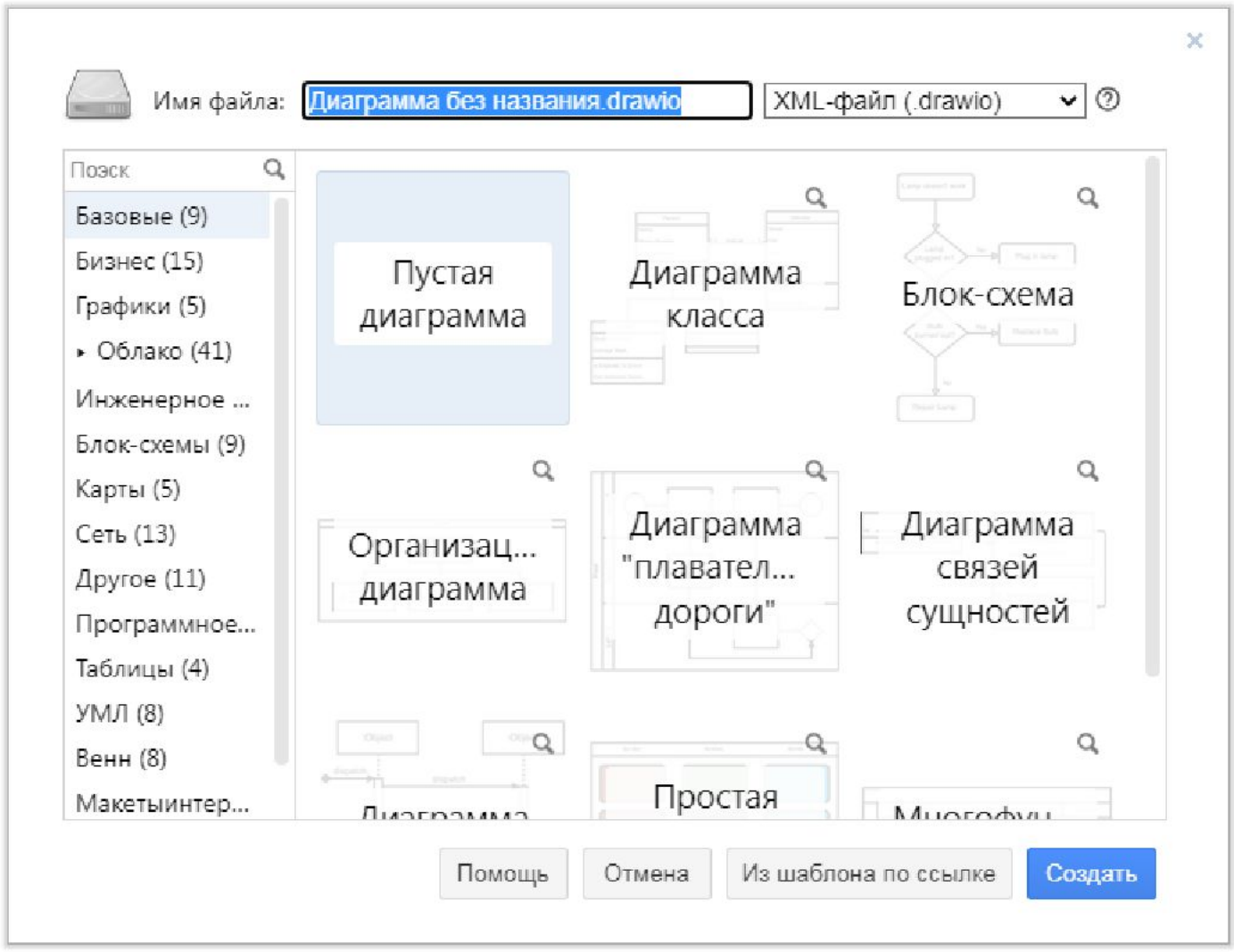
Дерево модели

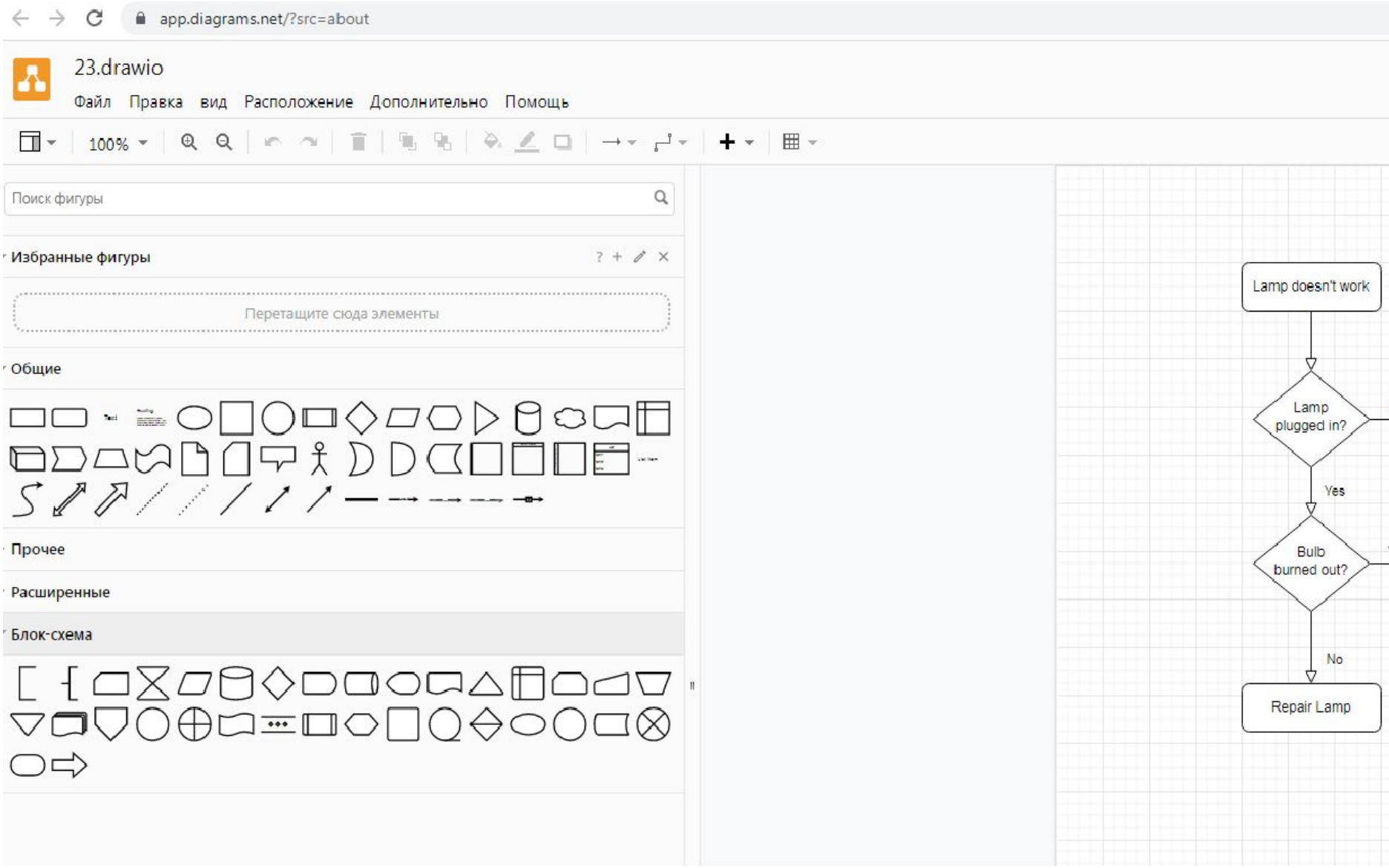


draw.io


Сохранять графики в:
Google Диск
Один диск
Это устройство
Дропбокс
Гитхаб
GitLab
Выбрать позже


Это устройство
Создать новую диаграмму
Открытую Существующую диаграмму
Изменить место хранения


Имя файла:
Диаграмма без названия.drawio
XML-файл (.drawio)
Поэск
Базовые (9)
Бизнес (15)
Графики (5)
Облако (41)
Инженерное ...
Блок-схемы (9)
Карты (5)
Сеть (13)
Другое (11)
Программное...
Таблицы (4)
УМЛ (8)
Венн (8)
Макетыинтер...
Пустая диаграмма
Диаграмма класса
Блок-схема
Организац... диаграмма
Диаграмма "плавател... дороги"
Диаграмма связей сущностей
Простая
Помощь
Отмена
Из шаблона по ссылке
Создать


app.diagrams.net/?src=about
23.drawio
Файл Правка вид Расположение Дополнительно Помощь
100%
Поиск фигуры
Избранные фигуры
Перетащите сюда элементы
Общие
Прочее
Расширенные
Блок-схема
Lamp doesn't work
Lamp plugged in?
Yes
Bulb burned out?
No
Repair Lamp

вопросы….

РАЗДЕЛ 2 МЕТОДЫ ПРОЕКТИРОВАНИЯ И ПРОГРАММИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ТЕМА 2.1 МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ CASE-ТЕХНОЛОГИИ

CASE-ТЕХНОЛОГИИ

Под термином CASE-средства понимаются программные средства, поддерживающие процесс создания и сопровождения ПО, включая:

- анализ и формирование требований;
- проектирование прикладного ПО (приложений) и БД;
- генерацию кода;
- тестирование;
- документирование;
- контроль и обеспечение качества;
- управление проектом;
- и др. процессы.

CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки.

CASE-ТЕХНОЛОГИИ

Современные крупные проекты имеют следующие особенности:

- сложность описания;
- наличие подсистем, решающих автономные задачи;
- отсутствие прямых аналогов;
- необходимость интеграции уже существующих и вновь разрабатываемых приложений;
- функционирование в неоднородной среде на нескольких аппаратных платформах;
- разобщенность и неоднородность различных групп разработчиков по уровню квалификации и использованию различных инструментальных средств;
- значительная временная протяженность проекта.

CASE-ТЕХНОЛОГИИ

Вручную достаточно трудно разработать и графически представить строгие формальные спецификации системы, проверить их на полноту и непротиворечивость и при необходимости внести изменения. Ручная разработка порождает следующие проблемы:

- неадекватная спецификация требований;
- неспособность обнаружения ошибок в проектных решениях;
- низкое качество документирования;
- затяжное и, зачастую, неудовлетворительное тестирование.

CASE-ТЕХНОЛОГИИ

Появлению CASE – технологии способствовали следующие факторы:

- наличие аналитиков и программистов, знакомых с концепциями модульного, структурного и объектно-ориентированного проектирования;
- широкое внедрение и рост производительности компьютеров;
- развитие сетевых технологий, позволяющих объединять усилия отдельных исполнителей в единый процесс.

CASE-ТЕХНОЛОГИИ

CASE-средства, как правило, не дают немедленного эффекта. Он может быть получен только спустя некоторое время. Реальные затраты на внедрение обычно намного превышают затраты на приобретение.

Пользователь, приобретающий CASE – средств, должен быть готов к необходимости долгосрочных затрат на эксплуатацию, к частому появлению новых версий, к быстрому моральному старению средств и к постоянным затратам на обучение и повышение квалификации сотрудников. Для успешного внедрения CASE-средств организация должна обладать:

- **технологией**, т.е. пониманием ограниченности существующих возможностей и способностью принять новую технологию;
- **культурой**, т.е. готовностью к внедрению новых процессов и взаимоотношений между разработчиками и пользователями;
- **управлением**, т.е. четким руководством на наиболее важных этапах в процессе внедрения.

CASE-ТЕХНОЛОГИИ

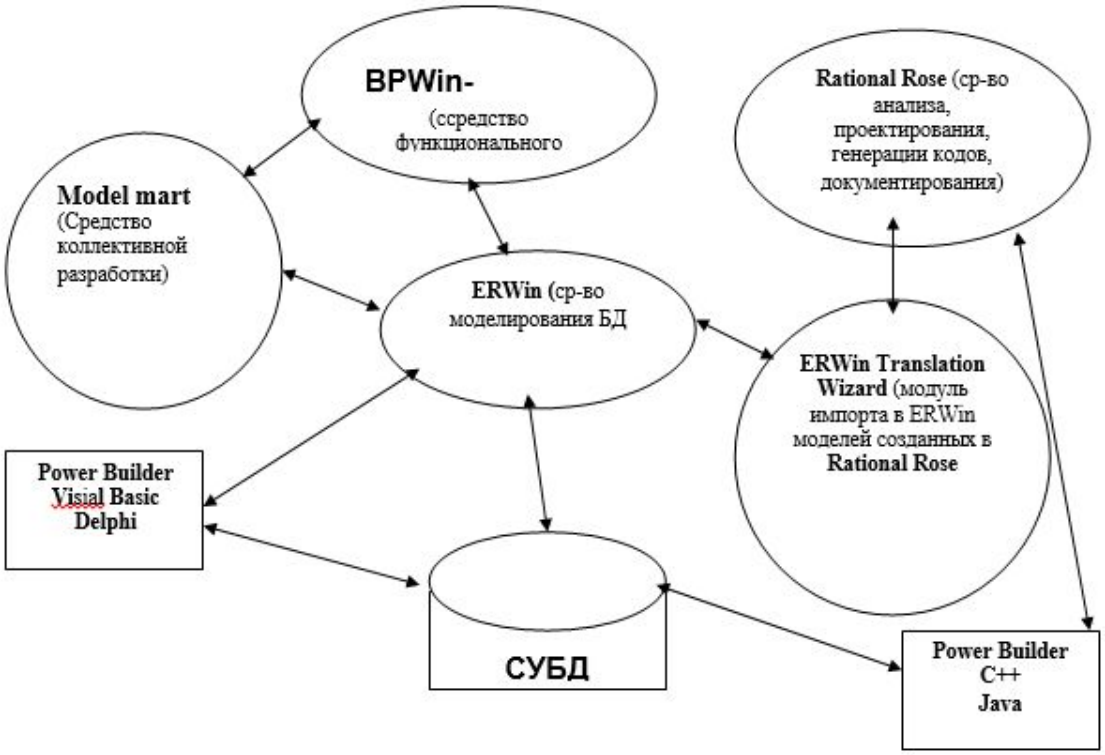
Процесс внедрения CASE – средств состоит из следующих этапов:

- определение потребности в CASE- средствах;
- оценка и выбор CASE- средств;
- выполнение пилотного проекта;
- практическое внедрение CASE – средств.

В качестве основных критериев выбора CASE – средств можно принять следующие:

- поддержка полного ЖЦПО;
- обеспечение целостности проекта и контроля за его состоянием;
- независимость от программно-аппаратной платформы и СУБД;
- открытая архитектура;
- качество, стоимость и опыт успешного использования;
- простота освоения и использования.

ПРИМЕР


BPWin-
(ссредство
функционального
Rational Rose (ср-во
анализа,
проектирования,
генерации кодов,
документирования)
Model mart
(Средство
коллективной
разработки)
ERWin (ср-во
моделирования БД
ERWin Translation
Wizard (модуль
импорта в ERWin
моделей созданных в
Rational Rose
Power Builder
Visial Basic
Delphi
СУБД
Power Builder
C++
Java

ПРИМЕР

Перед внедрением выбранного CASE-средства выполняется пилотный проект, целью которого является проверка правильности принятых на предыдущих этапах решений и подготовка к внедрению.

Пилотный проект – это первоначальное реальное использование CASE – средств в предназначенной для этого среде и, как правило подразумевает более широкий масштаб использования CASE-средства по отношению к тому, который был достигнут во время оценки. Он должен обладать многими из характеристик реальных проектов, для разработки которых приобретается CASE – средство. Он преследует следующие цели:

- подтверждает достоверность результатов этапов оценки и выбора;
- определяет, действительно ли данное средство годится для использования в данной организации и какова область его применения;
- собирает информацию для разработки плана практического внедрения;
- дает возможность приобрести опыт использования выбранного средства.

ПРИМЕР

По результатам выполнения пилотного проекта принимается решение о необходимости приобретения данного CASE – средства. В случае отказа организация несет не значительные убытки, связанные с приобретением небольшого количества лицензий и обучением небольшой группы специалистов.

После успешного завершения пилотного проекта выбранное CASE-средство приобретается, интегрируется в проектную среду и настраивается в соответствии с требованиями пользователя.

В этом случае, как показывает опыт возможно несколько вариантов:

- средство полностью удовлетворяет требованиям пользователя.
- частично удовлетворяет требованиям пользователя. При таком варианте выполняется дополнительный пилотный проект и CASE – средство либо дополняется недостающими компонентами, либо организация отказывается от его использования.

ПРИМЕР

Полный комплект CASE – средств, обеспечивающий полную поддержку ЖЦПО должен содержать следующие компоненты:

- **репозиторий**, - являющийся основой CASE – средства, хранящий версии проекта и его компоненты и обеспечивающий синхронизацию поступления информации от различных разработчиков при групповой разработке, а т.ж. контроль данных на полноту и не противоречивость,
- **графические средства анализа и проектирования**, обеспечивающие создание и редактирование иерархически связанных диаграмм (потоков данных и т.д.), образующих модели проектируемой системы;
- **средства разработки приложений**;
- **средства конфигурационного управления**;
- **средства документирования**;
- **средства тестирования**;
- **средства управления проектом**;
- **средства реинжиниринга**, - обеспечивающие анализ программных кодов и схем БД и формирования на их основе моделей и проектных спецификаций для повторной разработки.

КЛАССИФИКАЦИЯ CASE-СРЕДСТВ

На сегодняшний день рынок ПО предлагает следующие наиболее развитые CASE-средства:

- Vantage Team Builder;
- Designer 2000;
- Silverrun;
- ERwin, Bpwin;
- S-Designer,
- CASE.Аналитик;
- Rational Rose;
- SQL, JAM.

Классифицировать CASE-средства можно по следующим признакам:

- ориентация на этапы ЖЦПО;
- степень независимости от СУБ.Д
- функциональная полнота;
- тип используемой модели разработки.

КЛАССИФИКАЦИЯ CASE-СРЕДСТВ

По ориентации на этапы ЖЦПО можно выделить следующие средства:

- анализа (для построения моделей) - ERwin, BPwin, Rational Rose;
- анализа и проектирования (для создания проектных спецификаций) - Vantage Team Builder, Silverrun, Designer 2000, CASE.Аналитик;
- создания БД (для моделирования и разработки схем к основным СУБД) – SQL, ERwin, S-Designer;
- разработки приложений - SQL, JAM,Unifase, Delphi, Developer/2000;
- генераторы кодов - Vantage Team Builder, Silverrun;
- средства реинжиниринга - Silverrun, Vantage Team Builder, Designer 2000, S-Designer, Rational Rose, Object Team;
- конфигурационного управления – PVCS, SCCS;
- планирования и управления проектом – Microsoft Project, SE Companion;
- тестирования – Quality Works.

КЛАССИФИКАЦИЯ CASE-СРЕДСТВ

По степени независимости от СУБД CASE-средства можно разделить на две группы:

- независимые, которые поставляются в виде автономных систем, не входящих в состав конкретных СУБД. Обычно они поддерживают несколько форматов данных через интерфейс ODBC (ERwin, S-Designer, Silverrun);
- встроенные поддерживают формат БД СУБД, в состав которых они входят (Designer 2000, входящая в состав СУБД Oracke).

КЛАССИФИКАЦИЯ CASE-СРЕДСТВ

По функциональной полноте можно выделить следующие типы:

- средства, используемые для решения частных задач на одном или нескольких этапах ЖЦПО (ERwin, S-Designer, Silverrun, CASE.Аналитик);
- интегрированные системы, поддерживающие полный ЖЦПО (Vantage Team Builder, Designer 2000 с системой разработки приложений Developer/2000).

По типу используемой модели можно выделить три группы:

- структурные (Vantage Team Builder);
- объектно-ориентированные (Rational Rose, Object Team);
- комбинированные (Designer 2000).

Итоги лекции

Изучены следующие понятия:

- Структурный подход
- Функциональная модель
- Методология SADT/IDEF0
- Функциональный блок
- Интерфейсная дуга
- Декомпозиция
- Глоссарий
- FEO-диаграмма
- Дерево узлов
- Мастерская страница