Метод SADT (IDEF))

На сегодняшний день в программной инженерии существуют два основных подхода к разработке программных продуктов и АИС, принципиальное различие между которыми обусловлено разными способами декомпозиции систем. Первый подход называют *функционально – модульным или структурным*. В его основу положен принцип функциональной декомпозиции, при котором структура системы описывается в терминах иерархии её функций и передачи информации между отдельными функциональными элементами.

Второй, *объектно – ориентированный* подход использует объектную декомпозицию. При этом структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами.

Базовыми принципами структурного подхода являются:

- принцип «разделяй и властвуй»;
- принцип иерархического упорядочения;
- принцип абстрагирования;
- принцип непротиворечивости;
- принцип структурирования данных.

В структурном подходе используется в основном две группы средств, описывающих функциональную структуру системы и отношения между данными. Каждой группе средств соответствуют определенные типы моделей (диаграмм), наиболее распространенными из которых являются:

- DFD – диаграммы потоков данных;
- SADT (метод структурного анализа и проектирования) – функциональные модели (диаграммы);
- ERD – диаграммы «Сущность - связь».

Метод SADT представляет собой совокупность 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. Основные элементы этого метода основываются на следующих

концепциях.

- Графическое представление блочного моделирования. Графика блоков и дуг SADT – диаграммы отображает функцию в виде блока, а интерфейсы входа – выхода представляются дугами, соответственно входящим в блок и выходящим из него. Взаимодействие блоков друг с другом описывается посредством интерфейсных дуг, выражающих «ограничения», которые, в свою очередь, определяют когда и каким образом функции выполняются и управляются.
- Строгость и точность. Выполнение правил SADT требует достаточно строгости и точности, не накладывая в тоже время чрезмерных ограничений на действия аналитика. Правила SADT включают: ограничения количества блоков на каждом уровне декомпозиции (правило 3 – 6 блоков), связанность диаграмм (номера блоков), уникальность меток и наименований (отсутствие повторяющихся имён), синтаксические правила для графики (блоков и дуг), разделение входов и управлений (правило определения роли данных).
- Отделение организации от функций, т.е. исключение влияния административной структуры организации на функциональную модель.

Метод SADT может использоваться для моделирования самых разнообразных систем и определения требований и функций с последующей разработкой информационной системы, удовлетворяющей этим требованиям и реализующей эти функции. В существующих системах метод SADT может применяться для анализа функций, выполняемых системой, и указания механизмов, посредством которых они осуществляются.

Основными компонентами SADT – моделей являются диаграммы, все функции информационной системы (подсистемы) и интерфейсы на которых представлены как блоки и дуги соответственно. Управляющая информация входит в блок сверху, в то время как входная информация, которая подвергается обработке, показана с левой стороны, а результирующая информация – справа. Механизм (человек или прикладная программа), который выполняет функцию, представляется дугой, входящей в блок снизу. Элемент SADT – диаграммы показан на рис. 1.

Одним из важных моментов при моделировании бизнес – процессов организации с помощью метода SADT является точная согласованность типов

связей между функциями. Различают по крайней мере связи семи типов: случайные, логические, временные, процедурные, коммуникационные, последовательные, функциональные.

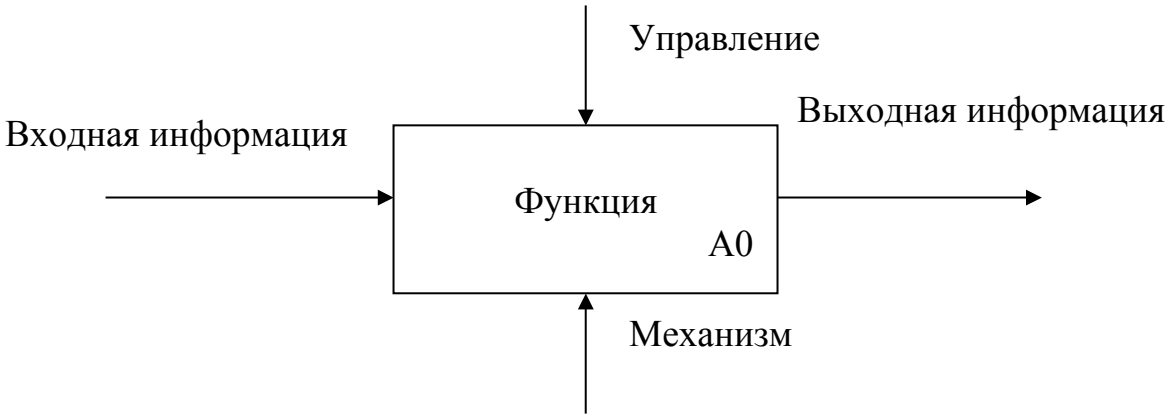

Рис. 1. Функциональный блок и интерфейсные дуги.

Построение SADT – диаграммы некоторой предметной области начинается с запуска программного продукта BPwin v 4.1 (Computer Associates), для этого необходимо выполнить поиск и запуск соответствующего программного продукта. При стандартной установке CASE – средства в среде операционной системы Windows' 2000 (XP) ярлык BPwin v 4.1 находится по схеме: «Пуск» - «Программы» - «Computer Associates» - «Allfusion» - «Process Modeler» - «BPwin» (см. рис. 2).

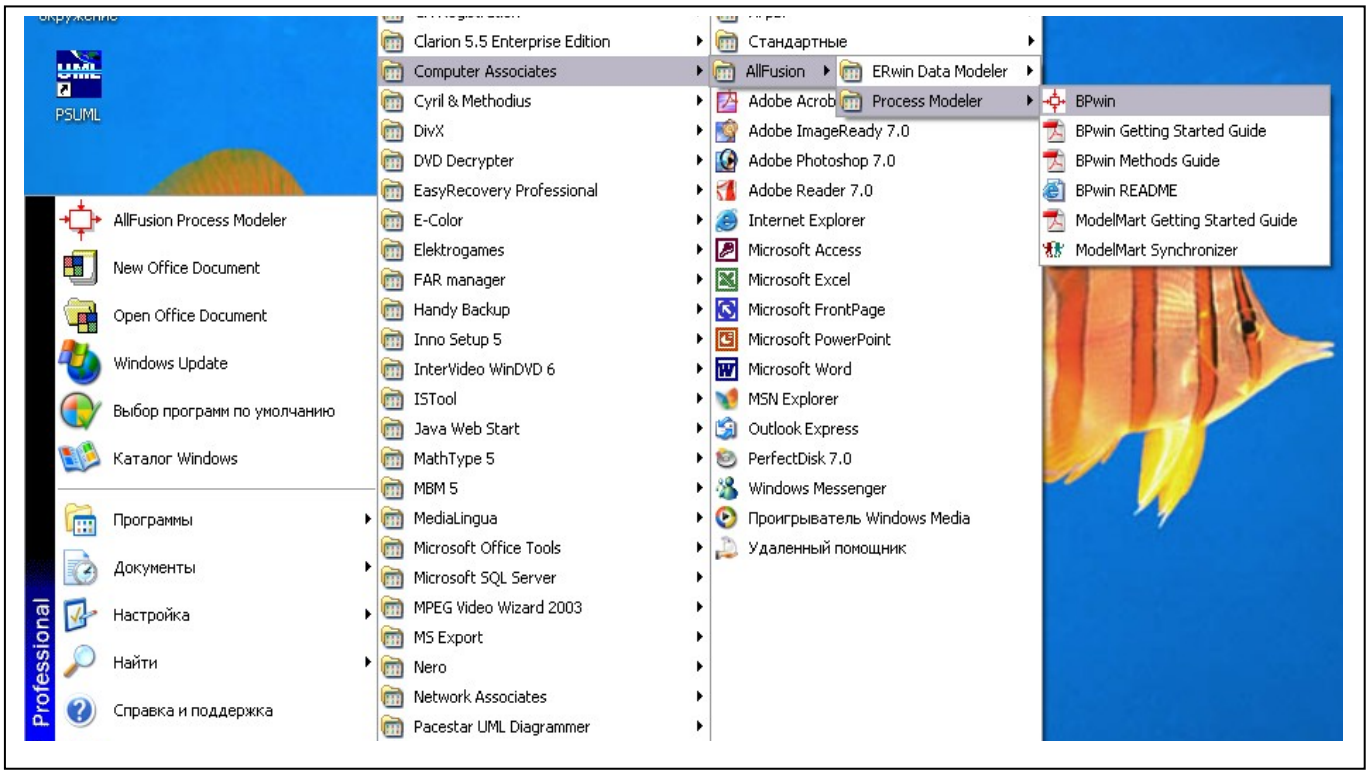

Рис. 2. Запуск CASE – средства BPwin v 4.1 (Computer Associates)

После запуска программного продукта BPwin v 4.1 в случае, если его пользователю необходимо создать новую SADT – модель, необходимо выбрать опцию «File» закладку «New» после чего в появившемся окне ввести имя будущей модели после слова «Name» и выбрать её тип после слова «Type». За словом «Type» следует последовательность поддерживаемых CASE – средством BPwin v 4.1 типов диаграмм: IDEF0, IDEF3, DFD (см. рис. 3). При создании SADT – модели тип диаграммы должен быть IDEF0. После определения имени и типа модели можно приступить к созданию SADT – модели верхнего иерархического уровня.

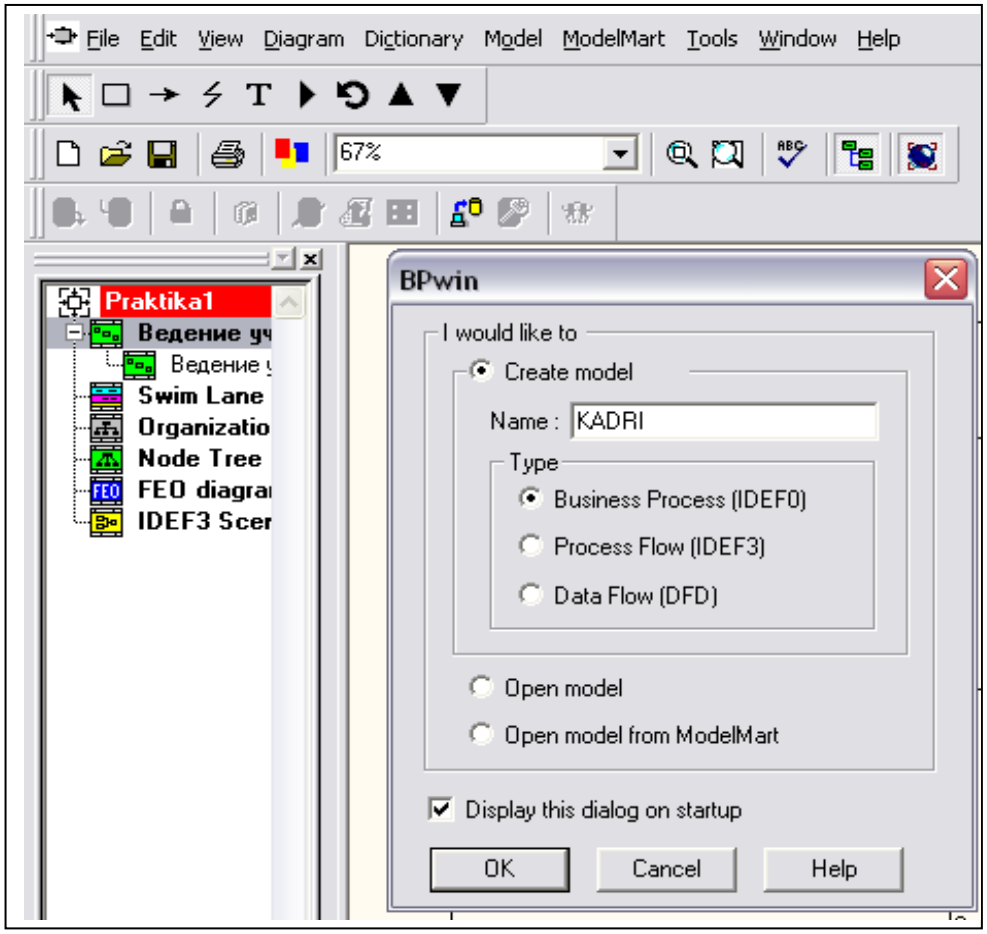

Рис. 3. Выбор типа модели при создании SADT - диаграммы

В соответствии с заданием данной практической работы на первой диаграмме необходимо создать единственный функциональный блок «Ведение учёта кадров» нажатием инструмента «Прямоугольник» на панели инструментов BPwin v 4.1. Далее нажатием инструмента «→» построить стрелки, символизирующие входную, выходную, информацию, управляющее воздействие, механизм (см. рис. 4). Диаграмма первого иерархического уровня не даёт представление о функциях, реализуемых исследуемой системой или предметной областью, поэтому необходимо произвести декомпозицию построенной диаграммы. Декомпозицию

функционального блока имеющейся диаграммы можно выполнить нажатием символа «». После чего появляется окно «Activity Box Count» (рис. 4), на котором необходимо определить тип дочерней диаграммы и предполагаемое количество функциональных блоков на ней, в рассматриваемой задаче тип диаграммы второго уровня будет также как и родительской IDEF0, количество функциональных блоков – 6.

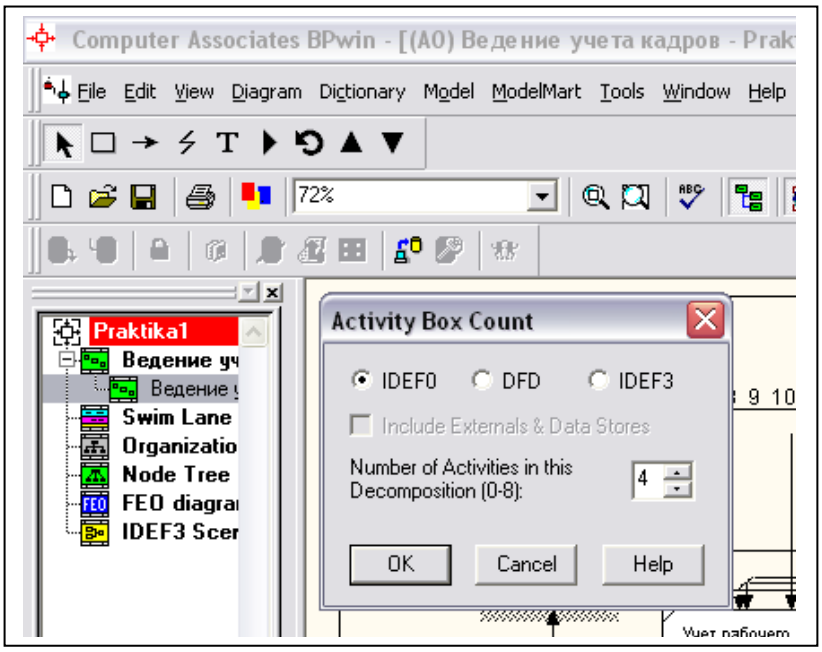

Рис. 4. Окно, позволяющее определить количество функциональных блоков в дочерней SADT – диаграмме.

На основе диаграммы второго уровня можно определить основные функции, реализуемые сотрудниками отдела кадров:

- ведение личных карточек и личных дел работников;
- учёт рабочего времени;
- формирование распорядительной документации;
- ведение трудовых книжек;
- формирование организационных документов;
- регистрация организационной, распорядительной документации.

Декомпозицию диаграмм различных иерархических уровней можно выполнять до тех пор, пока не будет достигнута требуемая степень детализации функций системы и количество функциональных блоков на диаграмме очередного иерархического уровня будет более двух. В рамках настоящей практической работы студентам рекомендуется самостоятельно декомпозировать SADT – диаграмму второго уровня.

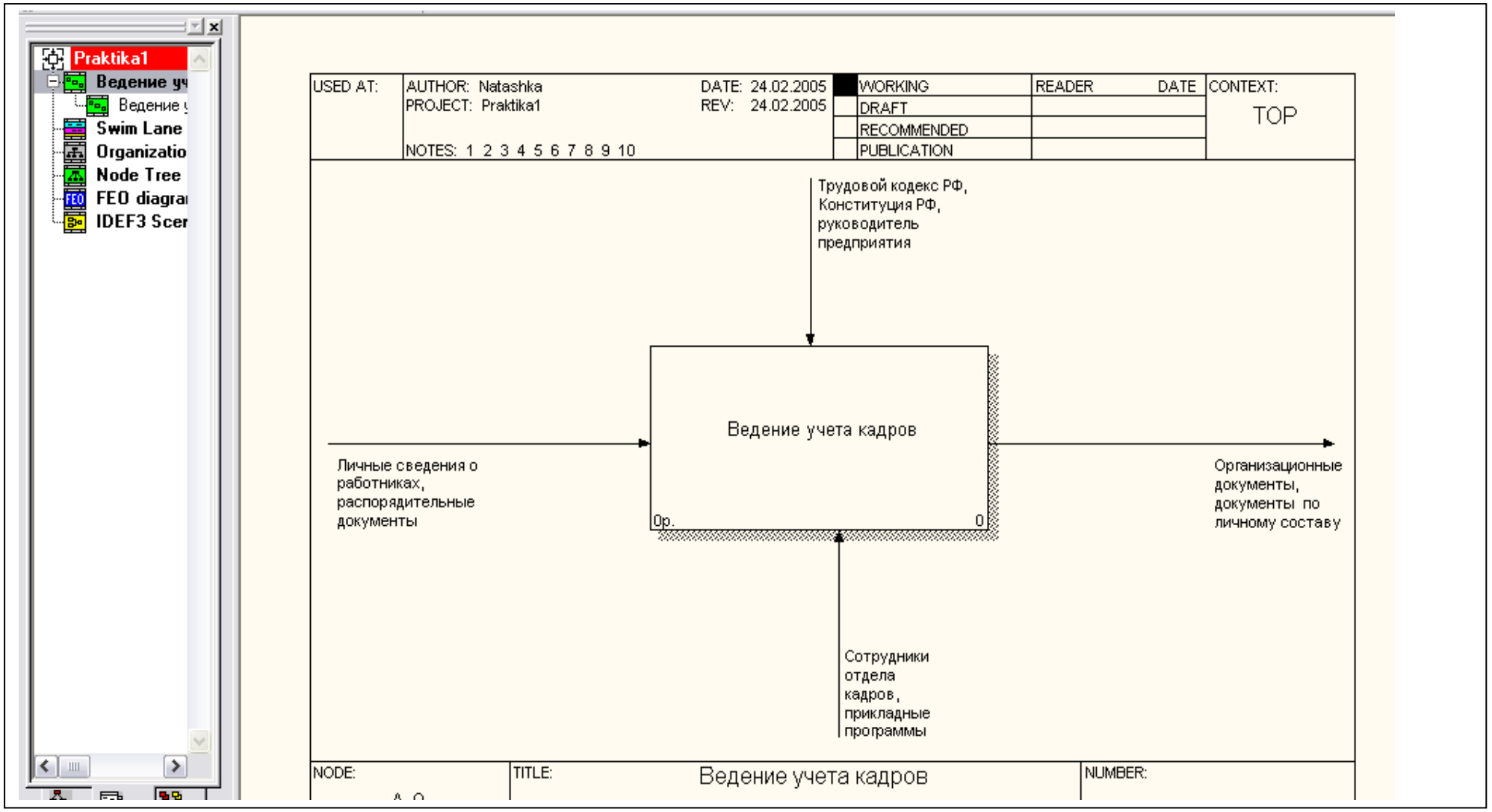

Рис. 5. SADT - диаграмма первого иерархического уровня

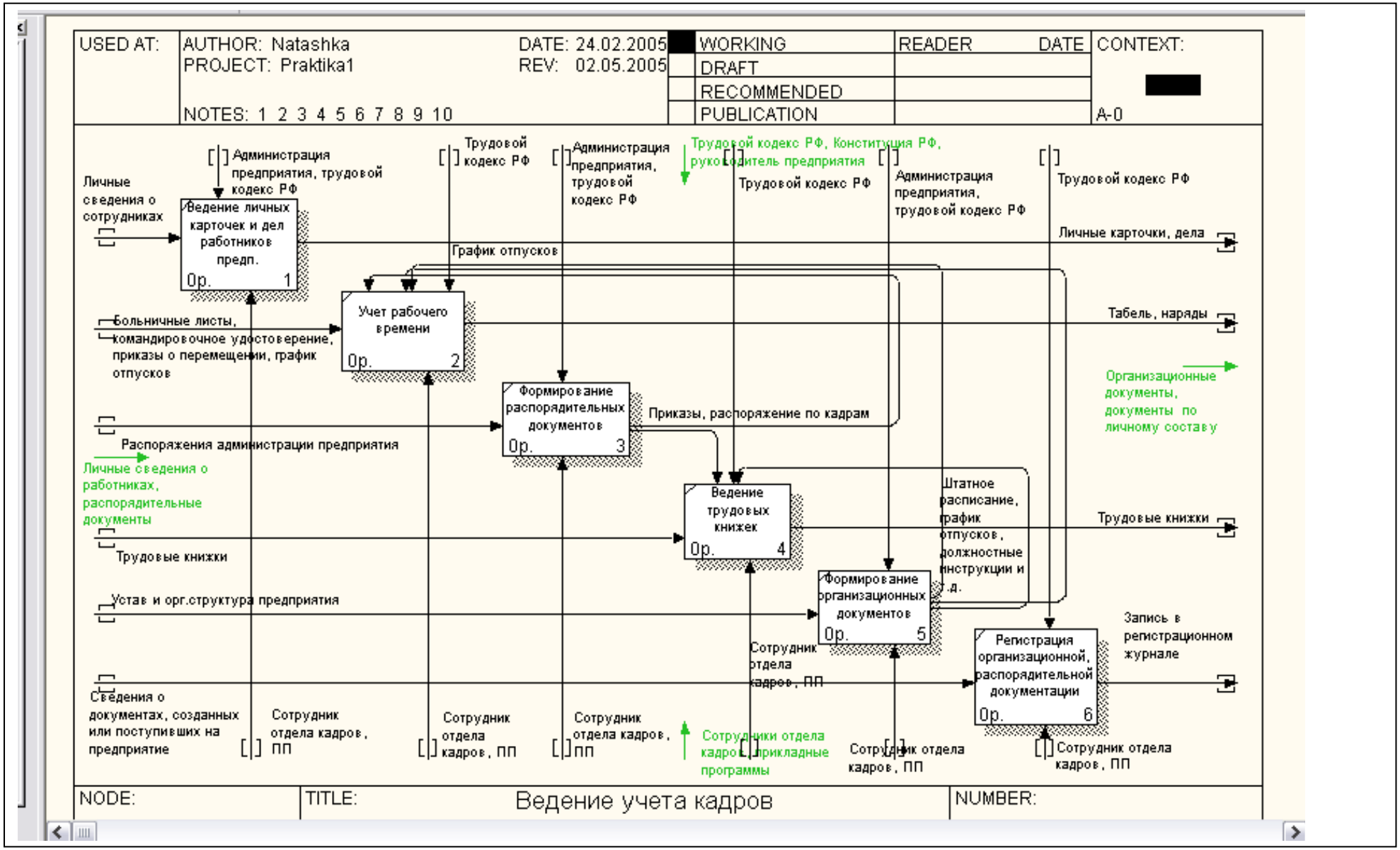

Рис. 6. SADT - диаграмма второго иерархического уровня

Иерархия диаграмм

Построение SADT-модели начинается с представления всей системы в виде простейшей компоненты - одного блока и дуг, изображающих интерфейсы с функциями вне системы. Поскольку единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг - они также представляют полный набор внешних интерфейсов системы в целом.

Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых представлена как блок, границы которого определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления.

Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию. Кроме того, модель не может опустить какие-либо элементы, т.е., как уже отмечалось, родительский блок и его интерфейсы обеспечивают контекст. К нему нельзя ничего добавить, и из него не может быть ничего удалено.

Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других диаграммах. Каждая детальная диаграмма является декомпозицией блока из более общей диаграммы. На каждом шаге декомпозиции более общая диаграмма называется родительской для более детальной диаграммы.

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, потому что блок и диаграмма представляют одну и ту же часть системы.

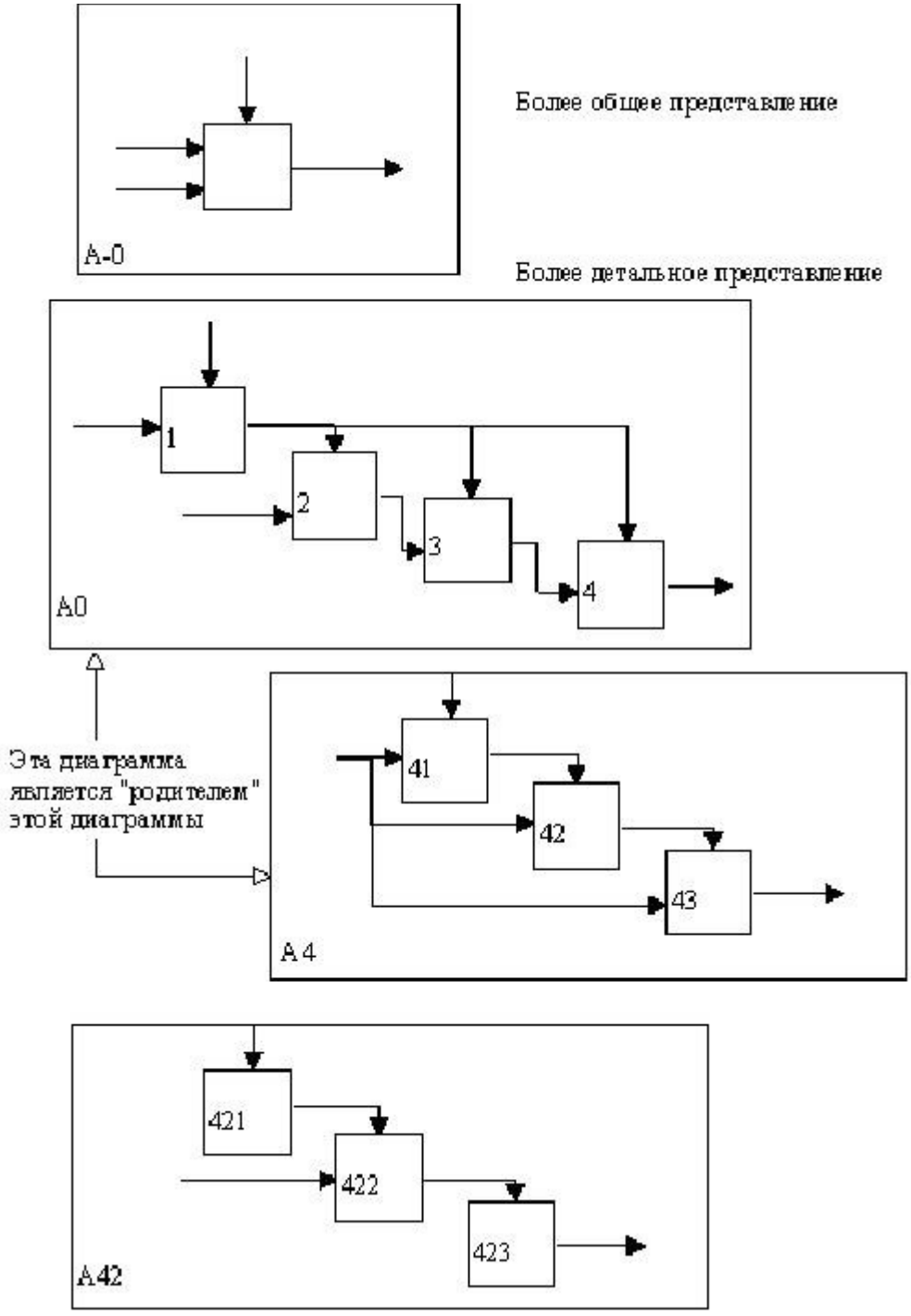

Рис. 1. Структура SADT-модели. Декомпозиция диаграмм

На рисунках 2-4 представлены различные варианты выполнения функций и соединения дуг с блоками.

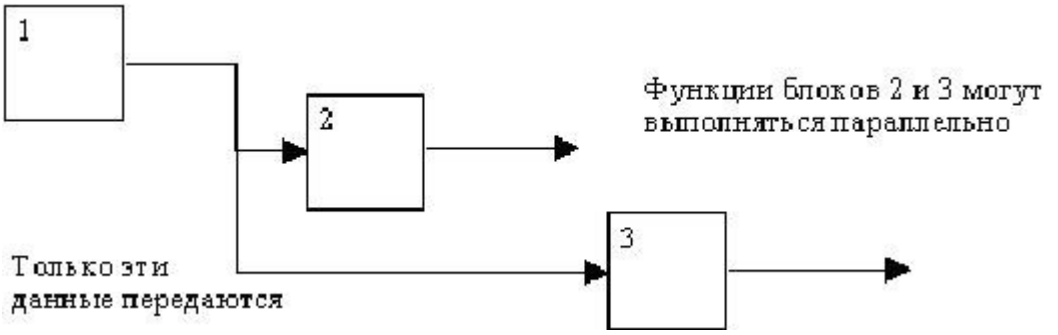

Рис. 2. Одновременное выполнение

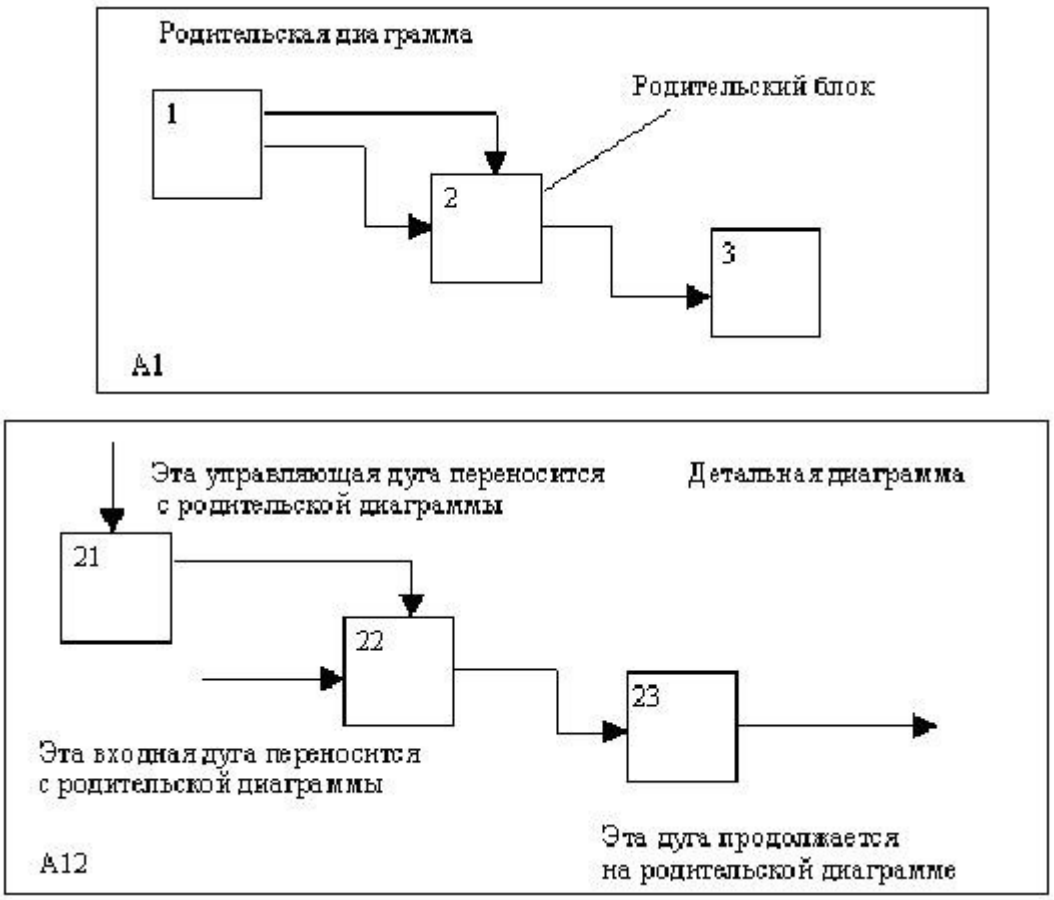

Рис. 3. Соответствие должно быть полным и непротиворечивым

Некоторые дуги присоединены к блокам диаграммы обоими концами, у других же один конец остается неприсоединенным. Неприсоединенные дуги соответствуют входам, управлениям и выходам родительского блока. Источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме. Неприсоединенные концы должны соответствовать дугам на исходной диаграмме. Все граничные дуги должны продолжаться на родительской диаграмме, чтобы она была полной и непротиворечивой.

На SADT-диаграммах не указаны явно ни последовательность, ни время. Обратные связи, итерации, продолжающиеся процессы и перекрывающиеся (по времени) функции могут быть изображены с помощью дуг. Обратные связи могут выступать в виде комментариев, замечаний, исправлений и т.д. (рисунок 4).

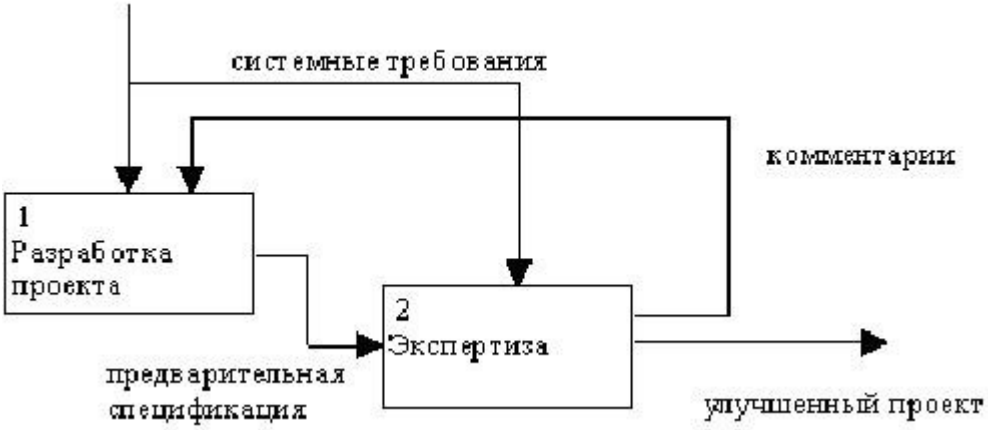

Рис. 4. Пример обратной связи

Как было отмечено, механизмы (дуги с нижней стороны) показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию (рисунок 5).

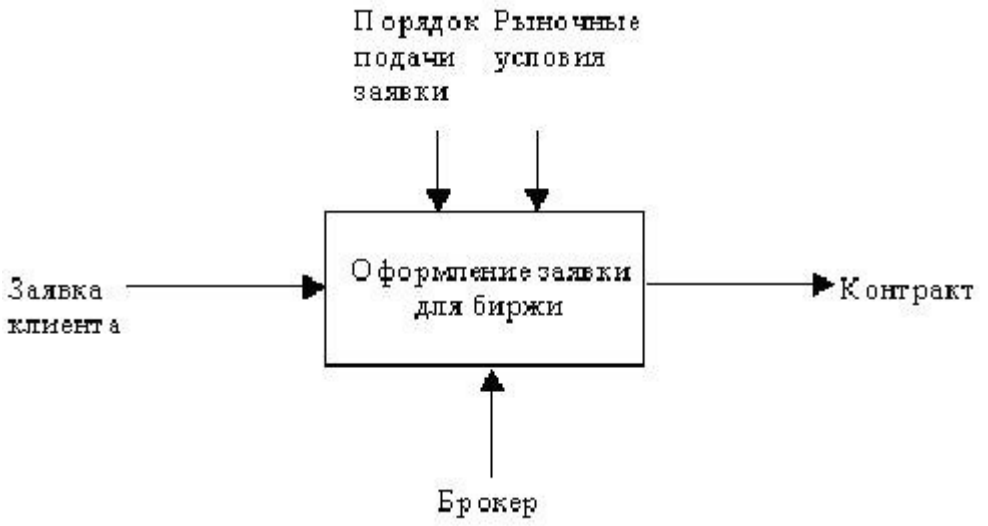

Рис. 5. Пример механизма

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть далее описан диаграммой нижнего уровня, которая, в свою

очередь, может быть далее детализирована с помощью необходимого числа диаграмм. Таким образом, формируется иерархия диаграмм.

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. На рисунке 6 показано типичное дерево диаграмм.

Рис. 6. Иерархия диаграмм

Типы связей между функциями

Одним из важных моментов при проектировании ИС с помощью методологии SADT является точная согласованность типов связей между функциями. Различают по крайней мере семь типов связывания:

Тип связи	Относительная значимость
Случайная	0
Логическая	1

Временная	2
Процедурная	3
Коммуникационная	4
Последовательная	5
Функциональная	6

Ниже каждый тип связи кратко определен и проиллюстрирован с помощью типичного примера из SADT. (0) Тип случайной связности: наименее желательный. Случайная связность возникает, когда конкретная связь между функциями мала или полностью отсутствует. Это относится к ситуации, когда имена данных на SADT-дугах в одной диаграмме имеют малую связь друг с другом. Крайний вариант этого случая показан на рисунке 6.12.

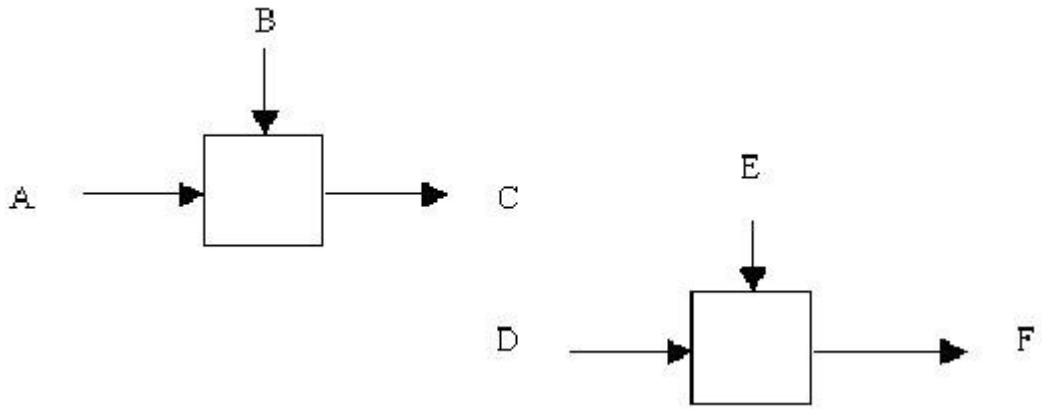

Рис. 7. Случайная связность

(1) Тип логической связности. Логическое связывание происходит тогда, когда данные и функции собираются вместе вследствие того, что они попадают в общий класс или набор элементов, но необходимых функциональных отношений между ними не обнаруживается.

(2) Тип временной связности. Связанные по времени элементы возникают вследствие того, что они представляют функции, связанные во времени, когда данные используются одновременно или функции включаются параллельно, а не последовательно.

(3) Тип процедурной связности. Процедурно-связанные элементы появляются сгруппированными вместе вследствие того, что они выполняются в

течение одной и той же части цикла или процесса. Пример процедурно-связанной диаграммы приведен на рисунке 8.

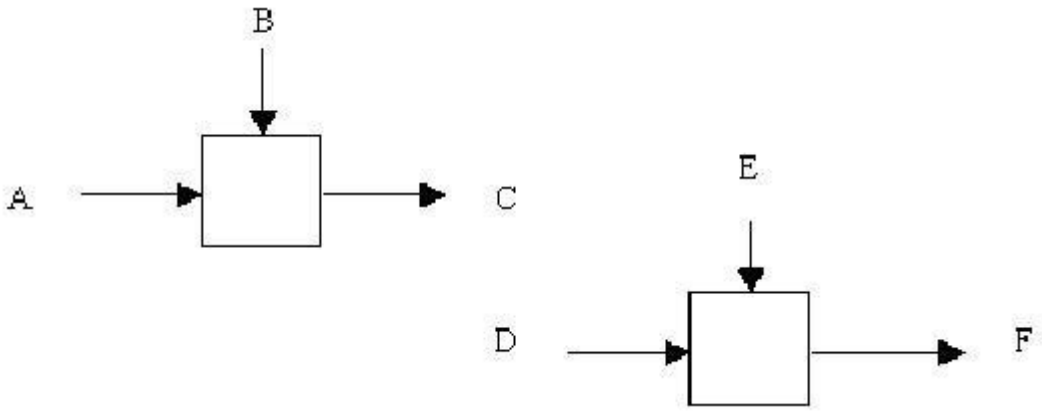

Рис. 8. Процедурная связность

(4) Тип коммуникационной связности. Диаграммы демонстрируют коммуникационные связи, когда блоки группируются вследствие того, что они используют одни и те же входные данные и/или производят одни и те же выходные данные (рисунок 9).

(5) Тип последовательной связности. На диаграммах, имеющих последовательные связи, выход одной функции служит входными данными для следующей функции. Связь между элементами на диаграмме является более тесной, чем на рассмотренных выше уровнях связок, поскольку моделируются причинно-следственные зависимости (рисунок 10).

(6) Тип функциональной связности. Диаграмма отражает полную функциональную связность, при наличии полной зависимости одной функции от другой. Диаграмма, которая является чисто функциональной, не содержит чужеродных элементов, относящихся к последовательному или более слабому типу связности. Одним из способов определения функционально-связанных диаграмм является рассмотрение двух блоков, связанных через управляющие дуги, как показано на рисунке 11.

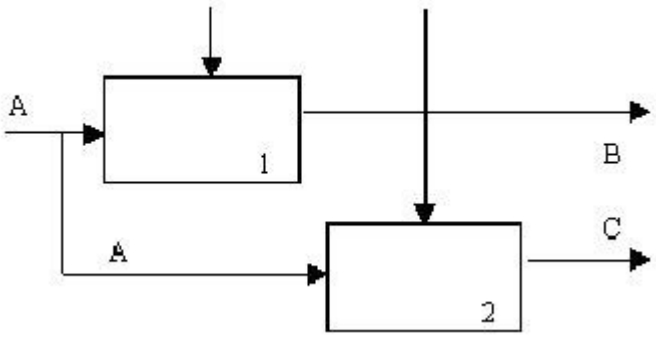

Рис. 9. Коммуникационная связность

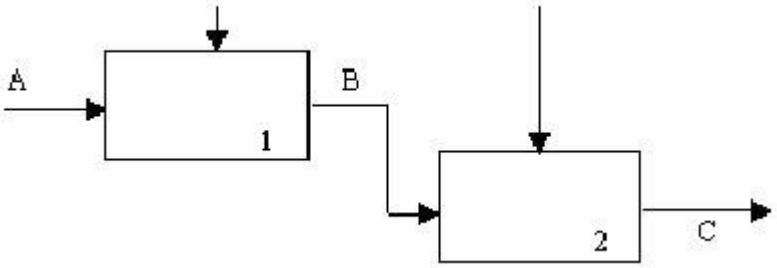

Рис. 10. Последовательная связность

В математических терминах необходимое условие для простейшего типа функциональной связности, показанной на рисунке 6.15, имеет следующий вид:

$$C = g(B) = g(f(A))$$

Ниже в таблице представлены все типы связей, рассмотренные выше. Важно отметить, что уровни 4-6 устанавливают типы связностей, которые разработчики считают важнейшими для получения диаграмм хорошего качества.

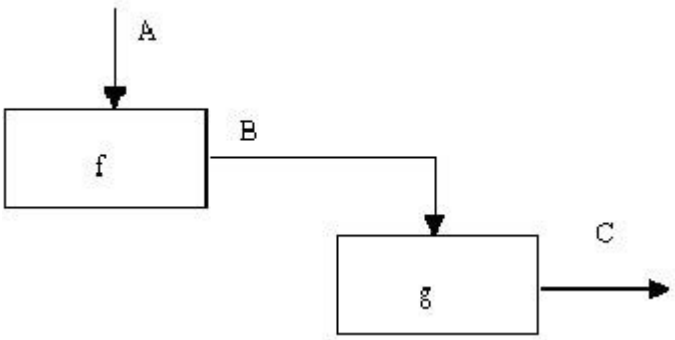

Рис. 11. Функциональная связность

Значимость	Тип связности	Для функций	Для данных
0	Случайная	Случайная	Случайная
1	Логическая	Функции одного и того же множества или типа (например, "редактировать все входы")	Данные одного и того же множества или типа
2	Временная	Функции одного и того же периода времени (например, "операции инициализации")	Данные, используемые в каком-либо временном интервале
3	Процедурная	Функции, работающие в одной и той же фазе или итерации (например, "первый проход компилятора")	Данные, используемые во время одной и той же фазы или итерации
4	Коммуникационная	Функции, использующие одни и те же данные	Данные, на которые воздействует одна и та же деятельность
5	Последовательная	Функции, выполняющие последовательные преобразования одних и тех же данных	Данные, преобразуемые последовательными функциями
6	Функциональная	Функции, объединяемые для	Данные, связанные с

		выполнения одной функции	одной функцией