## Лекция 5. Проектирование ПО при структурном подходе

Разработка любого ПО начинается с анализа требований к будущему программному продукту. В результате анализа получают спецификации разрабатываемого ПО: выполняют декомпозицию и содержательную постановку решаемых задач, уточняют их взаимодействие и эксплуатационные ограничения. В процессе определения спецификаций строят общую модель предметной области, как некоторой части реального мира, с которой будет взаимодействовать разрабатываемое ПО, и конкретизируют его основные функции.

Спецификации - *полное* и *точное* описание функций и ограничений разрабатываемого ПО. Одна часть спецификаций (*функциональные*) описывает функции разрабатываемого ПО, а другая часть (*эксплуатационные*) определяет требования к техническим средствам, надежности, информационной безопасности и т.д.

Определение отражает главные требования к спецификациям. Применительно к функциональным спецификациям подразумевается, что:

• требование *полноты* - спецификации должны содержать всю существенную информацию, где отсутствует несущественная информация;

• требование *точности* - спецификации должны однозначно восприниматься заказчиком и разработчиком.

Последнее требование выполнить достаточно сложно в силу того, что естественный язык для описания спецификаций не подходит: даже подробные спецификации на естественном языке не обеспечивают необходимой точности. Точные спецификации можно определить, только разработав некоторую *формальную модель* разрабатываемого ПО.

Формальные модели на этапе определения спецификаций разделяют на две группы: *зависящие от подхода к разработке* (структурного или ОО) и *не зависящие* от него. Диаграммы переходов состояний и математические модели предметной области используют при любом подходе к разработке.

В рамках структурного подхода на этапе анализа и определения спецификаций используют модели: ориентированные на функции, ориентированные на данные и ориентированные на потоки данных. Каждую модель целесообразно использовать для своего класса программных разработок. На рис. 5.1 показана классификация моделей разрабатываемого ПО, используемых на этапе определения спецификаций.

Все функциональные спецификации описывают одни и те же характеристики разрабатываемого ПО: перечень функций и состав обрабатываемых данных.


Рис.5.1. Классификация моделей разрабатываемого ПО на этапе спецификаций

Они различаются только системой приоритетов (акцентов), которая используется разработчиком в процессе анализа требований и определения спецификаций. Диаграммы переходов состояний определяют основные аспекты поведения ПО во времени, диаграммы потоков данных – направление и структуру потоков данных, а концептуальные диаграммы классов – отношение между основными понятиями предметной области.

Поскольку разные модели описывают проектируемое ПО с разных сторон, рекомендуется использовать сразу несколько моделей и сопровождать их текстами: словарями, описаниями и т.п., которые поясняют соответствующие диаграммы.

Методологии *структурного анализа и проектирования*, основанные на моделировании потоков данных, используют комплексное представление проектируемого ПО в виде совокупности моделей:

• диаграмм потоков данных (DFD – Data Flow Diagrams), описывающих взаимодействие источников и потребителей информации через процессы, которые должны быть реализованы в системе;

• диаграмм «сущность-связь» (ERD – Entity-Relationship Diagrams), описывающих базы данных разрабатываемой системы;

• диаграмм переходов состояний (STD – State Transition Diagrams), характеризующих поведение системы во времени;

• спецификаций процессов;

• словаря терминов.

Взаимосвязь элементов такой обобщенной модели показана на рис. 5.2.

*Спецификации процессов* представляют в виде краткого текстового описания, схем алгоритмов, псевдокодов и др. Описание процесса должно быть кратким и понятным разработчику и заказчику, поэтому для их спецификации часто используют псевдокоды.

*Словарь терминов* - краткое описание основных понятий. Он включает: определение основных понятий предметной области, описание структур элементов данных, их типов и форматов, сокращений и условных обозначений. Предназначен для повышения степени понимания предметной области.

Описание термина в словаре выполняют по схеме: термин, категория (понятие предметной области, элемент данных, условное обозначение), краткое описание.

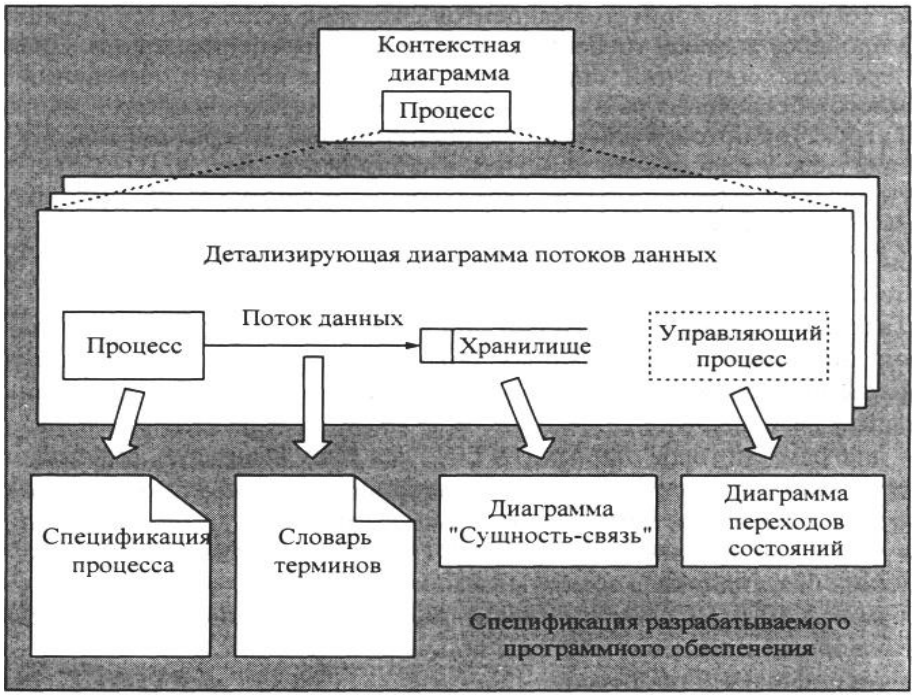



Рис.5.2. Элементы полной модели методологий структурного анализа и проектирования ПО

Рассмотрим перечисленные модели.

***Диаграмма переходов состояний*** является графической формой предоставления *конечного автомата* – математической абстракции, используемой для моделирования детерминированного поведения технических объектов или объектов реального мира.

На этапе анализа требований и определения спецификаций диаграмма переходов состояний демонстрирует *поведение* разрабатываемой программной системы при получении управляющих воздействий (сигналов) - управляющей информации, получаемой системой извне: команды пользователя и сигналы датчиков. Получив такое управляющее воздействие, разрабатываемая система должна выполнить определенные действия и или остаться в том же состоянии, или перейти в другое состояние взаимодействия с внешней средой.

Для построения диаграммы переходов состояний необходимо определить: основные состояния, управляющие воздействия, выполняемые действия и варианты переходов из одного состояния в другое. Условные обозначения диаграммы переходов состояний - на рис. 5.3.

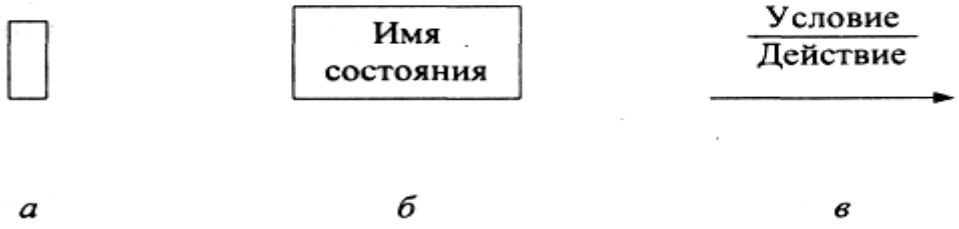



Рис.5.3. Условные обозначения диаграмм переходов состояний:
а – терминальное состояние, б – промежуточное состояние, в – переход.

**Пример 5.1.** Рассмотрим диаграмму переходов состояний для программы построения графиков функций одной переменной (ТЗ на нее представлено в лк.3).

Программа относится к классу интерактивных, соответственно на этапе анализа и определения спецификаций целесообразно уточнить поведение программы на уровне интерфейса с пользователем. Один из возможных вариантов диаграммы переходов состояний программы представлен на рис. 4.6.

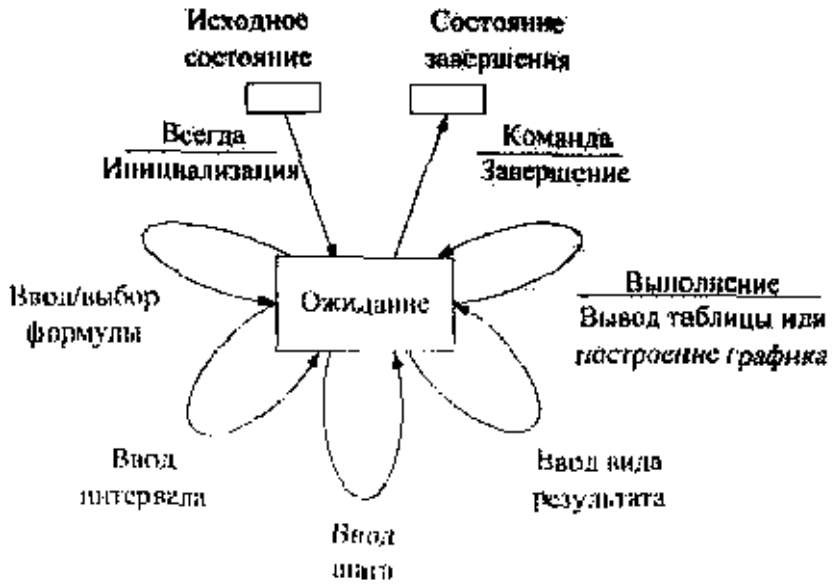



Рис. 5.4. Диаграмма переходов состояний программы построения графиков / таблиц функций

**Метод функционального моделирования SADT**

***Функциональными*** называют диаграммы, отражающие *взаимосвязи функций* разрабатываемого ПО. Пример - активностная модель, предложенная Д. Россом в составе методологии функционального моделирования ***SADT*** (Structured Analysis and Design Technique - технология структурного анализа и проектирования) в 1973 г.

Методология SADT предполагает, что модель может основываться либо на функциях системы, либо на ее предметах (данных, оборудовании, информации). В обоих случаях используют схожие графические нотации, но в первом случае блок соответствует функции, а во втором - элементу данных. Соответствующие модели называют *активностными моделями и моделями данных*. Полная модель включает построение обеих моделей, обеспечивающих полное описание ПО, однако широкое распространение получили только активностные (функциональные) модели. На основе методологии SADT построена известная методология описания сложных систем IDEFO (Icam DEFinition - нотация ICAM), которая является основной частью программы ICAM (Integrated Computer-Aided Manufacturing - интегрированная компьютеризация производства), проводимой по инициативе ВВС США.

Отображение взаимосвязи функций активностной модели осуществляется построением *иерархии функциональных диаграмм,* представляющих взаимосвязи нескольких функций. Каждый блок диаграммы соответствует некоторой функции, для которой должны быть определены: исходные данные, результаты, управляющая информация и механизмы ее осуществления (человек или технические средства).

Все перечисленные связи функции представляются дугами, тип связи и ее направление строго регламентированы. Дуги, изображающие каждый тип связей, должны подходить к блоку с определенной стороны (рис. 5.7), а направление связи должно указываться стрелкой в конце дуги.

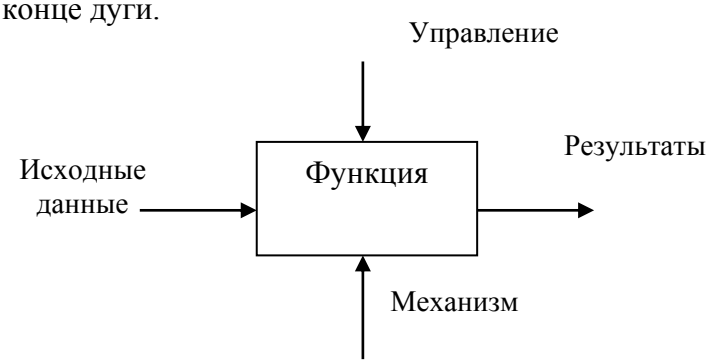
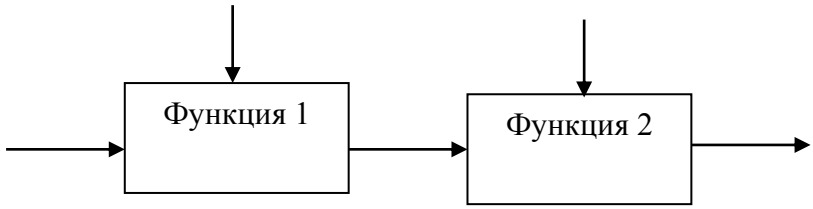


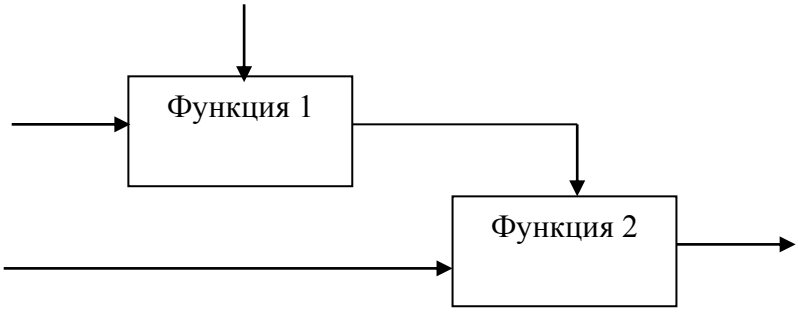

Рис.5.7. Функциональный блок и интерфейсные дуги

Физически дуги исходных данных, результатов и управления представляют собой наборы данных, передаваемые между функциями. Дуги, определяющие механизм выполнения функции, в основном используются при описании спецификаций сложных информационных систем, которые включают как автоматизированные, так и ручные операции. Блоки и дуги маркируются текстами на естественном языке.

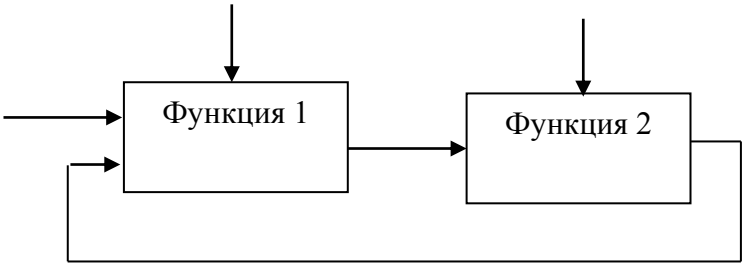
Блоки на диаграмме размещают по «ступенчатой» схеме в соответствии с последовательностью их работы или *доминированием,* которое понимается как влияние, оказываемое одним блоком на другие (рис.5.8).



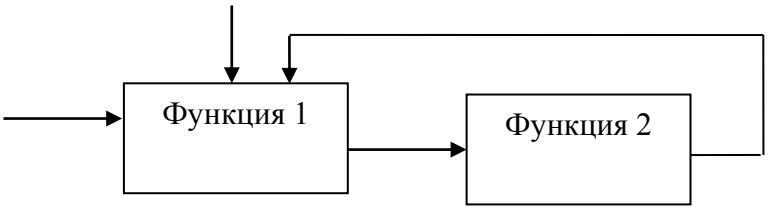

а



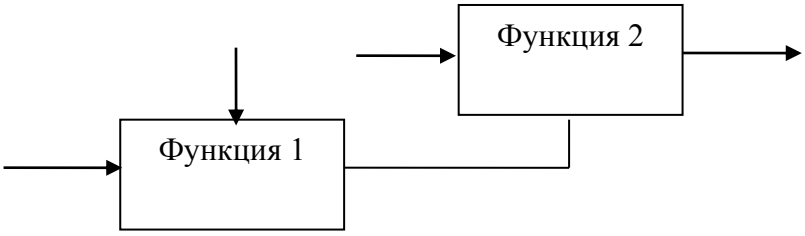

б




в




г




д

Рис.5.8. Типы влияния блоков: а –вход; б – управление; в – обратная связь по входу; г – обратная связь по управлению; д – выход-исполнитель

В функциональных диаграммах SADT - пять типов влияний блоков друг на друга:

• вход - выход блока подается на вход блока с меньшим доминированием, т. е. следующего (рис. 5.8, *а*);

• управление - выход блока используется как управление для блока с меньшим доминированием (следующего) (рис. 5.8, *б*);

• обратная связь по входу - выход блока подается на вход блока с большим доминированием (предыдущего) (рис. 5.8, в);

• обратная связь по управлению - выход блока используется как управляющая информация для блока с большим доминированием (предыдущего) (рис. 5.8, *г*);

• выход-исполнитель - выход блока используется как механизм для другого блока (рис. 5.8, *д*).

Дуги могут разветвляться и соединяться вместе различными способами. Разветвление означает, что часть или вся информация может использоваться в каждом ответвлении дуги. Дуга всегда помечается до ветвления, чтобы идентифицировать передаваемый набор данных. Если ветвь дуги после ветвления не помечена, то непомеченная ветвь содержит весь набор данных. Каждая метка ветви уточняет, что именно содержит данная ветвь (рис. 5.9).

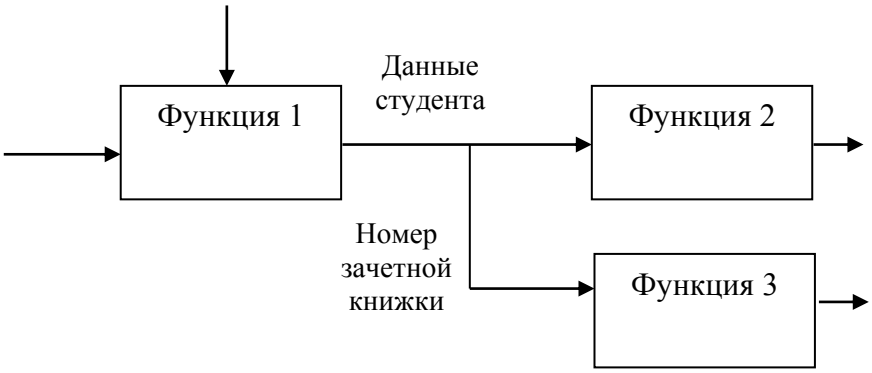



Рис.5.9. Пример обозначения дуг при ветвлении

Построение иерархии функциональных диаграмм ведется поэтапно с увеличением уровня детализации: диаграммы каждого следующего уровня уточняют структуру родительского блока. Построение модели начинают с единственного блока, для которого определяют исходные данные, результаты, управление и механизмы реализации. Затем он последовательно детализируется с использованием метода пошаговой детализации. Рекомендуется каждую функцию представлять 3-7-ю блоками. Во всех случаях *каждая подфункция может использовать или продуцировать только те элементы данных, которые использованы или продуцируются родительской функцией,* причем никакие элементы не могут быть опущены, что обеспечивает непротиворечивость построенной модели.

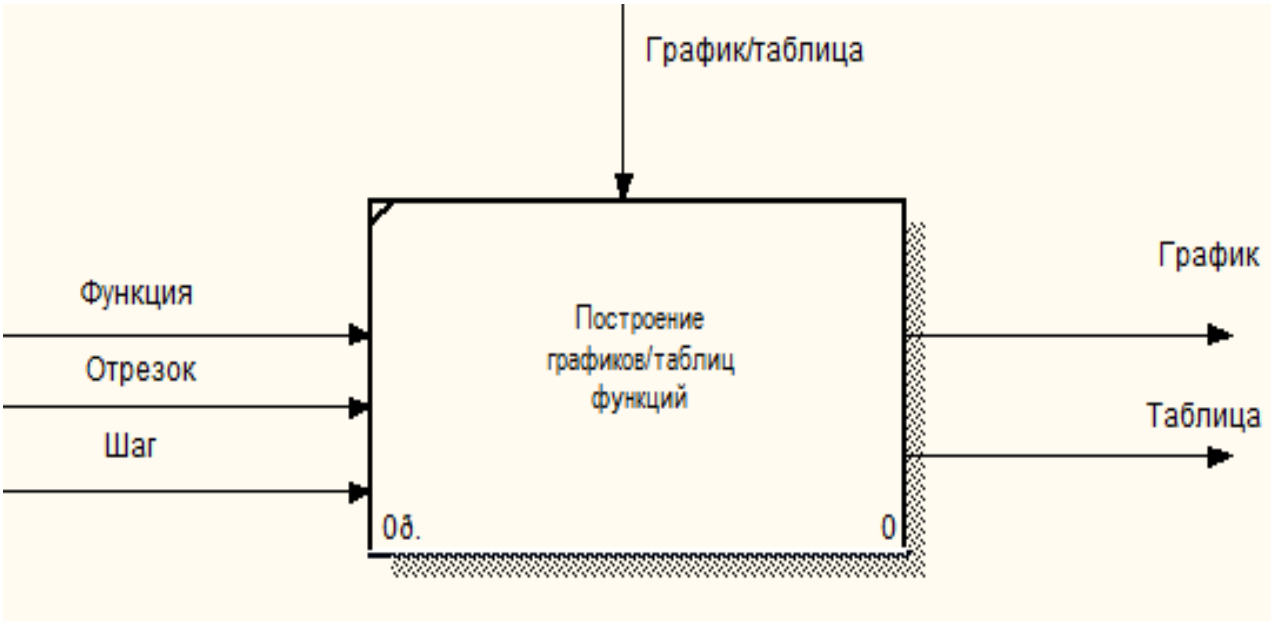
Стрелки, приходящие с родительской диаграммы или уходящие на нее, нумеруют, используя символы и числа. Символ обозначает тип связи: I - входная, C – управляющая, M – механизм, R - результат. Число - номер связи по соответствующей стороне родительского блока, считая сверху вниз и слева направо.

Все диаграммы связывают друг с другом иерархической нумерацией блоков: первый уровень - AO, второй - Al, A2 и т. п., третий - A11, A12, A13 и т.п., где первые цифры - номер родительского блока, а последняя - номер конкретного субблока родительского блока.

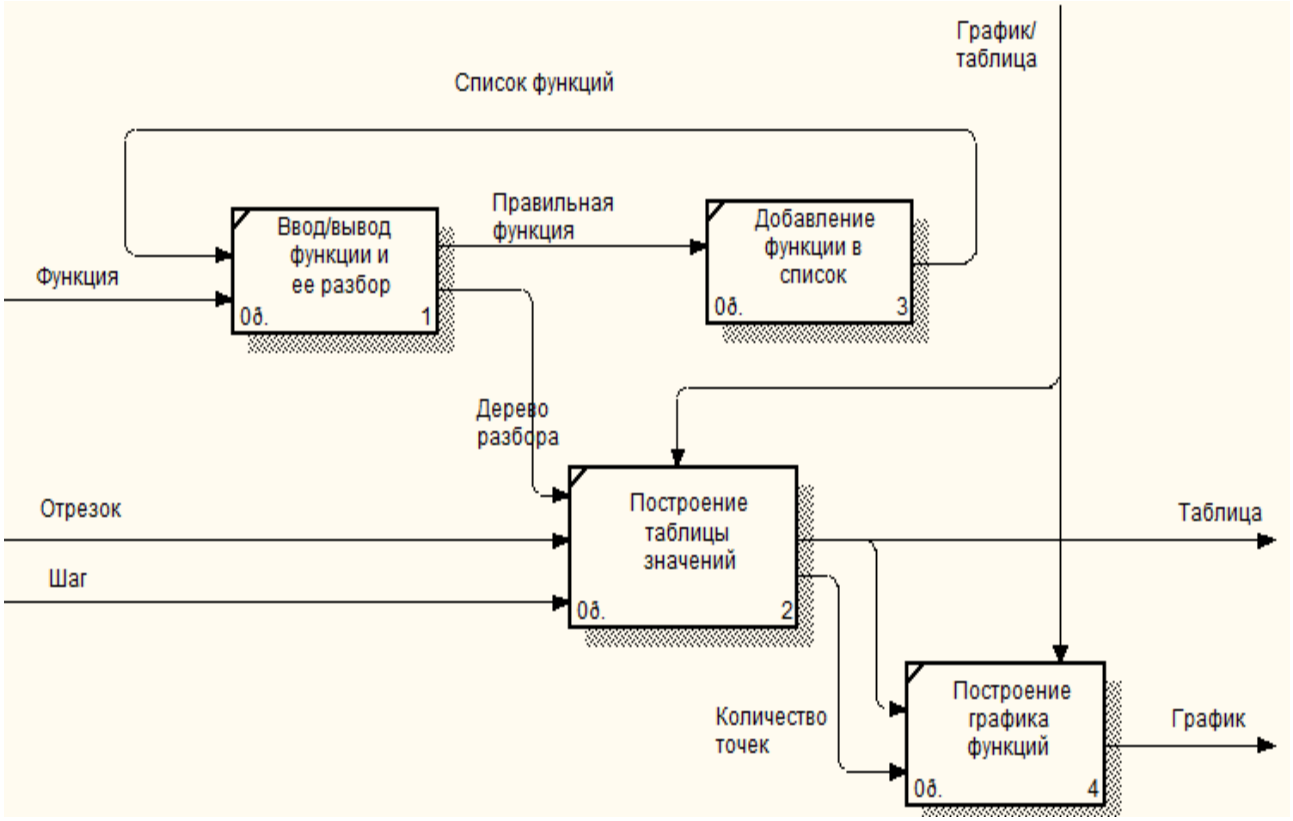
Детализацию завершают после получения функций, назначение которых хорошо понятно как заказчику, так и разработчику. В процессе построения иерархии диаграмм фиксируют всю уточняющую информацию и строят словарь данных, в котором определяют структуры и элементы данных, показанных на диаграммах.

В результате получают спецификацию, состоящую из иерархии функциональных диаграмм, спецификаций функций нижнего уровня и словаря со ссылками друг на друга.

**Пример 4.2.** Разработку функциональных диаграмм продемонстрируем на примере уточнения спецификаций программы построения таблиц/графиков функций одной переменной (рис. 5.10).




а




б

Рис.5.10. Функциональные диаграммы для системы исследования функций:
а – диаграмма верхнего уровня; б – уточняющая диаграмма

Диаграмма на рис. 5.10, *а*, является диаграммой верхнего уровня. На ней хорошо видно, что является исходными данными для программы. и каких результатов работы от нее ожидают. Диаграмма на рис, 5.10. *б,* уточняет функции программы. На ней показаны четыре блока: Ввод/выбор функции и ее разбор. Добавление функции в список. Построение таблицы значений и Построение графика функции.

Для каждого блока определены исходные данные, управляющие воздействия и результаты. Словарь в этом случае должен содержать описание всех данных, используемых в системе. Функциональную модель целесообразно применять для определения спецификаций ПО, не предусматривающего работу со сложными структурами данных, так как она ориентирована на декомпозицию функций.

***Диаграммы потоков данных*** позволяют специфицировать функции разрабатываемого ПО и обрабатываемые им данные. При использовании этой модели систему представляют в виде иерархии диаграмм потоков данных, описывающих асинхронный процесс преобразования информации с момента ввода в систему до выдачи пользователю. На каждом следующем уровне иерархии происходит уточнение процессов, пока очередной процесс не будет признан элементарным.

Модели потоков данных были независимо предложены сначала Е Йорданом (1975), затем Ч. Генном и Т. Сарсоном (1979). На этих моделях основаны классические методологии структурного анализа и проектирования программного обеспечения соответственно Йордана-Де Марка и Геина-Сарсона. Та же модель используется в методологии структурного анализа и проектирования SSADM (Structured Systems Analysis and Design Method). принятой в Великобритании в качестве национального стандарта разработки информационных систем.

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

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

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

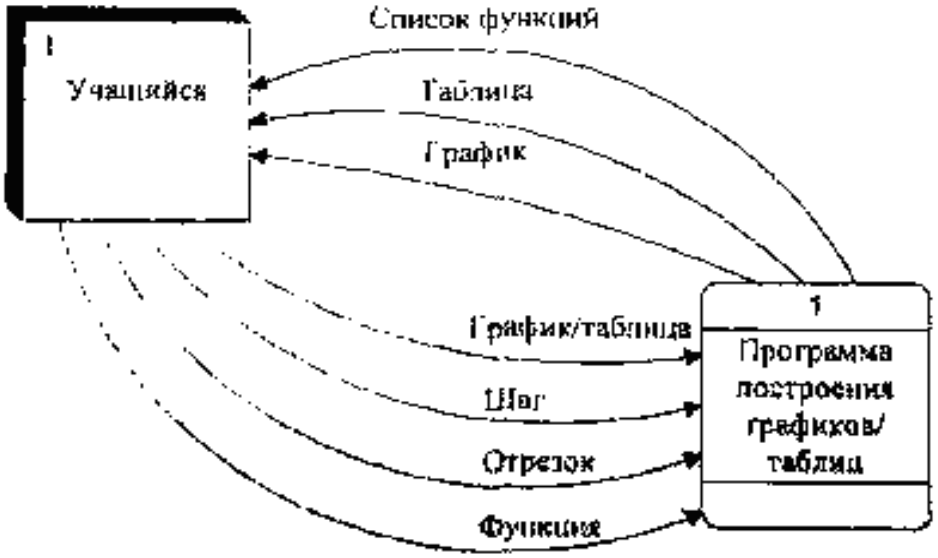
*Хранилище данных* - абстрактное устройство для хранения информации. Тип устройства и способы помещения, извлечения и хранения для такого устройства не детализируют. Физически это может быть БД, файл, таблица в ОП, картотека на бумаге, т.п.

*Поток данных* - процесс передачи информации от источника к приемнику. Физически процесс передачи информации может происходить по кабелям под управлением программы или вручную при участии устройств или людей вне проектируемой системы.

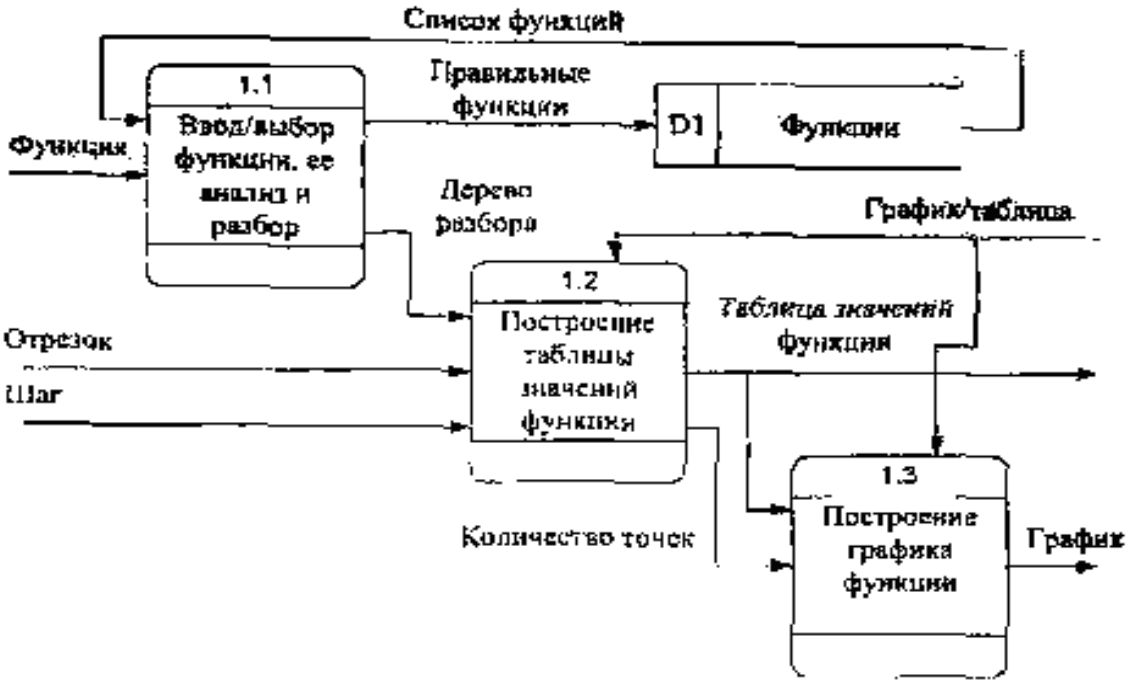
Таким образом, диаграмма иллюстрирует как потоки данных, порожденные некоторыми внешними сущностями, трансформируются соответствующими процессами (или подсистемами), сохраняются накопителями данных и передаются другим внешним сущностям - приемникам информации. В результате получают сетевую модель хранения/обработки информации. Для изображения диаграмм потоков данных традиционно используют два вида нотаций: нотации Йордана и Гейна-Сарсона.

Иерархия диаграмм потоков данных программы построения графиков/таблиц функций представлена на рис. 5.11: рис.5.11, а - контекстная диаграмма системы, рис.5.11, б - детализирующая диаграмма потоков данных системы.

Разработку начинают с построения контекстной диаграммы. Определяют внешние сущности и потоки данных между программой и внешними сущностями. У данной системы единственная внешняя сущность - Учащийся. Он вводит или выбирает из списка функцию, задает интервал и количество точек, а затем получает таблицу значений функции и ее график. Детализируя диаграмму, получаем три процесса: Ввод/выбор функции и ее разбор, Построение таблицы значений функции и Построение графика функции. Для хранения функций добавляем хранилище функций. Затем определяют потоки данных.




а




б

Рис.5.11. Диаграммы потоков данных программы построения графиков/таблиц функций: а – контекстная диаграмма системы; б – детализирующая диаграмма потоков данных системы

Если сравнить полученную детализирующую диаграмму потоков данных (рис. 5.11, б) и функциональные диаграммы для той же системы (см. рис. 5.10), то можно отметить некоторые различия в представлении одной и той же информации. Например, на диаграмме потоков данных можно показать хранилище данных, что очень существенно для систем, включающих базы данных. Кроме того, диаграммы потоков данных позволяют точно адресовать функции системы при наличии нескольких категорий пользователей.