### Структурный подход к проектированию ИС

#### Сущность структурного подхода

Сущность структурного подхода 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.

Все наиболее распространенные методологии структурного подхода [9,11,12,13] базируются на ряде общих принципов [3]. В качестве двух базовых принципов используются следующие:

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

Выделение двух базовых принципов не означает, что остальные принципы являются второстепенными, поскольку игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к провалу всего проекта). Основными из этих принципов являются следующие:

* принцип абстрагирования - заключается в выделении существенных аспектов системы и отвлечения от несущественных;
* принцип формализации - заключается в необходимости строгого методического подхода к решению проблемы;
* принцип непротиворечивости - заключается в обоснованности и согласованности элементов;
* принцип структурирования данных - заключается в том, что данные должны быть структурированы и иерархически организованы.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются следующие:

* SADT (Structured Analysis and Design Technique) модели и соответствующие функциональные диаграммы (подраздел 2.2);
* DFD (Data Flow Diagrams) диаграммы потоков данных (подраздел 2.3);
* ERD (Entity-Relationship Diagrams) диаграммы "сущность-связь" (подраздел 2.4).

На стадии проектирования ИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру программного обеспечения: архитектуру ПО, структурные схемы программ и диаграммы экранных форм.

Перечисленные модели в совокупности дают полное описание ИС независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

#### Методология функционального моделирования SADT

Методология SADT разработана Дугласом Россом и получила дальнейшее развитие в работе [4]. На ее основе разработана, в частности, известная методология IDEF0 (Icam DEFinition), которая является основной частью программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой по инициативе ВВС США.

Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями. Основные элементы этой методологии основываются на следующих концепциях:

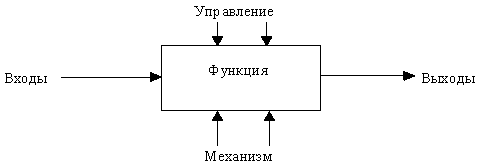
* графическое представление блочного моделирования. Графика блоков и дуг SADT-диаграммы отображает функцию в виде блока, а интерфейсы входа/выхода представляются дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описываются посредством интерфейсных дуг, выражающих "ограничения", которые в свою очередь определяют, когда и каким образом функции выполняются и управляются;
* строгость и точность. Выполнение правил SADT требует достаточной строгости и точности, не накладывая в то же время чрезмерных ограничений на действия аналитика. Правила SADT включают:
* ограничение количества блоков на каждом уровне декомпозиции (правило 3-6 блоков);
* связность диаграмм (номера блоков);
* уникальность меток и наименований (отсутствие повторяющихся имен);
* синтаксические правила для графики (блоков и дуг);
* разделение входов и управлений (правило определения роли данных).
* отделение организации от функции, т.е. исключение влияния организационной структуры на функциональную модель.

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

#### *Состав функциональной модели*

Результатом применения методологии SADT является модель, которая состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга. Диаграммы - главные компоненты модели, все функции ИС и интерфейсы на них представлены как блоки и дуги. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке, показана с левой стороны блока, а результаты выхода показаны с правой стороны. Механизм (человек или автоматизированная система), который осуществляет операцию, представляется дугой, входящей в блок снизу (рисунок 2.1).

Одной из наиболее важных особенностей методологии SADT является постепенное введение все больших уровней детализации по мере создания диаграмм, отображающих модель.



*Рис. 2.1. Функциональный блок и интерфейсные дуги*

На рисунке 2.2, где приведены четыре диаграммы и их взаимосвязи, показана структура SADT-модели. Каждый компонент модели может быть декомпозирован на другой диаграмме. Каждая диаграмма иллюстрирует "внутреннее строение" блока на родительской диаграмме.

#### *Иерархия диаграмм*

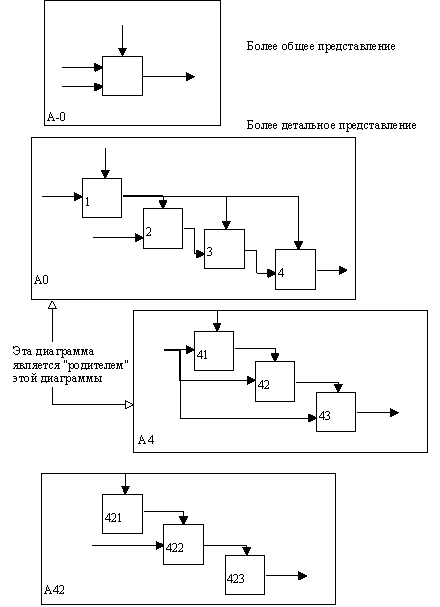
Построение SADT-модели начинается с представления всей системы в виде простейшей компоненты - одного блока и дуг, изображающих интерфейсы с функциями вне системы. Поскольку единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных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

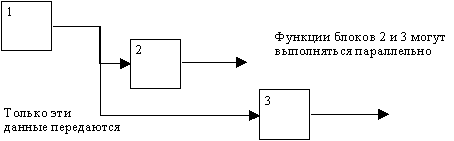
Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других диаграммах. Каждая детальная диаграмма является декомпозицией блока из 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.

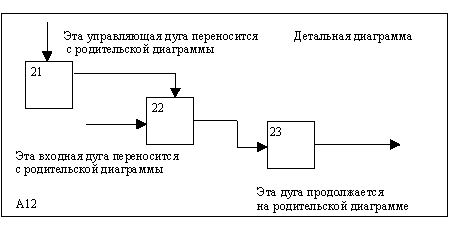
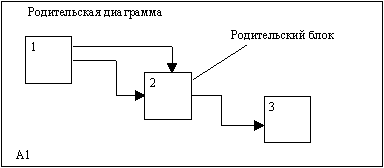


*Рис. 2.2. Структура SADT-модели. Декомпозиция диаграмм*

На рисунках 2.3 - 2.5 представлены различные варианты выполнения функций и соединения дуг с блоками.



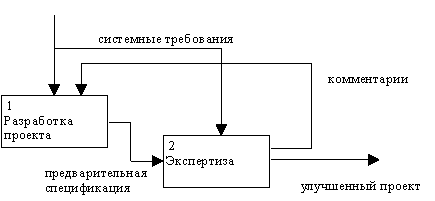
*Рис. 2.3. Одновременное выполнение*



*Рис. 2.4. Соответствие должно быть полным и непротиворечивым*

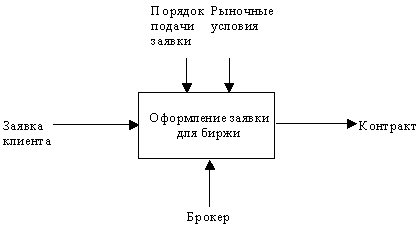
Некоторые дуги присоединены к блокам диаграммы обоими концами, у других же один конец остается неприсоединенным. Неприсоединенные дуги соответствуют входам, управлениям и выходам родительского блока. Источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме. Неприсоединенные 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 2.5).



*Рис. 2.5. Пример обратной связи*

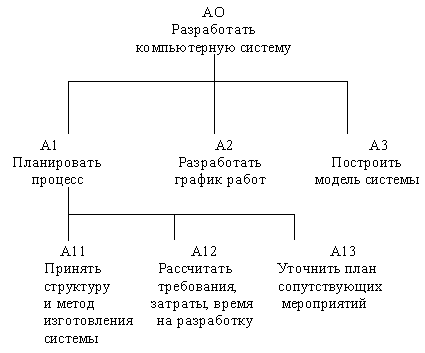
Как было отмечено, механизмы (дуги с нижней стороны) показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию (рисунок 2.6).



*Рис. 2.6. 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, А21 является диаграммой, которая детализирует блок 1 на диаграмме А2. Аналогично, А2 детализирует блок 2 на диаграмме А0, которая является самой верхней диаграммой модели. На рисунке 2.7 показано типичное дерево диаграмм.



*Рис. 2.7. Иерархия диаграмм*

#### *Типы связей между функциями*

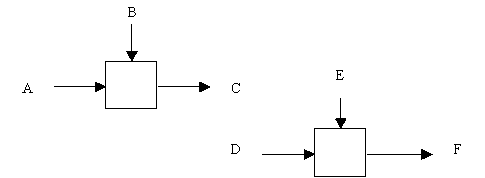
Одним из важных моментов при проектировании ИС с помощью методологии SADT является точная согласованность типов связей между функциями. Различают по крайней мере семь типов связывания:

|  |  |
| --- | --- |
| **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 2.8.

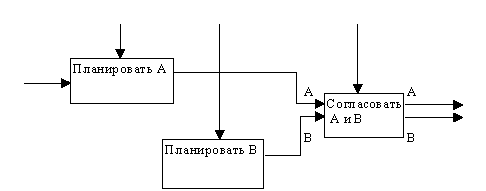


*Рис. 2.8. Случайная связность*

**(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 2.9.

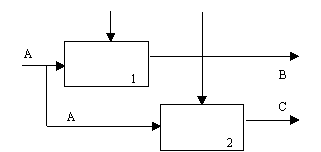


*Рис. 2.9. Процедурная связность*

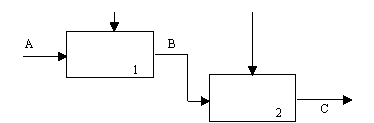
**(4) Тип коммуникационной связности.** Диаграммы демонстрируют коммуникационные связи, когда блоки группируются вследствие того, что они используют одни и те же входные данные и/или производят одни и те же выходные данные (рисунок 2.10).

**(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 2.11).

**(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 2.12.



*Рис. 2.10. Коммуникационная связность*

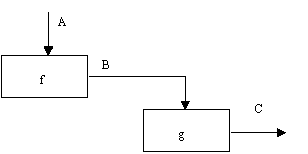


*Рис. 2.11. Последовательная связность*

В математических терминах необходимое условие для простейшего типа функциональной связности, показанной на рисунке 2.12, 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. 2.12. 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** | Коммуникационнная | 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 выполнения одной функции | Данные, связанные с одной функцией |

### Литература

1. Вендров А.М. Один из подходов к выбору средств проектирования баз данных и приложений. "СУБД", 1995, №3.
2. Зиндер Е.З. Бизнес-реинжиниринг и технологии системного проектирования. Учебное пособие. М., Центр Информационных Технологий, 1996
3. Калянов Г.Н. CASE. Структурный системный анализ (автоматизация и применение). М., "Лори", 1996.
4. Марка Д.А., МакГоуэн К. Методология структурного анализа и проектирования. М., "МетаТехнология", 1993.
5. Международные стандарты, поддерживающие жизненный цикл программных средств. М., МП "Экономика", 1996
6. Создание информационной системы предприятия. "Computer Direct", 1996, N2
7. Шлеер С., Меллор С. Объектно-ориентированный анализ: моделирование мира в состояниях. Киев, "Диалектика", 1993.
8. Barker R. CASE\*Method. Entity-Relationship Modelling. Copyright Oracle Corporation UK Limited, Addison-Wesley Publishing Co., 1990.
9. Barker R. CASE\*Method. Function and Process Modelling. Copyright Oracle Corporation UK Limited, Addison-Wesley Publishing Co., 1990.
10. Boehm B.W. A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes, Aug. 1986
11. Chris Gane, Trish Sarson. Structured System Analysis. Prentice-Hall, 1979.
12. Edward Yourdon. Modern Structured Analysis. Prentice-Hall, 1989.
13. Tom DeMarco. Structured Analysis and System Specification. Yourdon Press, New York, 1978.
14. Westmount I-CASE User Manual. Westmount Technology B.V., Netherlands, 1994.
15. Uniface V6.1 Designers' Guide. Uniface B.V., Netherlands, 1994.