**Лекция №16**

**Содержание лекции**

[Структурный метод разработки ИС 1](#_Toc158640045)

[Теоретические сведения о методологии IDEF0 3](#_Toc158640046)

[Основные определения (понятия) методологии и языка IDEF0 6](#_Toc158640047)

# Структурный метод разработки ИС

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 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 методологии структурного анализа базируются на ряде общих принципов, часть из которых регламентирует организацию работ на начальных этапах ЖЦ, а часть используется при выработке рекомендаций по организации работ. В качестве двух базовых принципов используются следующие: принцип "разделяй и властвуй" и принцип иерархического упорядочивания. Первый является принципом решения трудных проблем путем разбиения их на множество меньших независимых задач, более легких для понимания и ре­шения. Второй принцип декларирует, что устройство этих частей также существенно для понимания. Уровень уяснения проблемы резко повышается при представлении ее частей в виде древовидных иерархических структур, т.е. система может быть понята и построена по уровням, каждый из которых добавляет новые детали.

Отметим основные из неосновных принципов.

1. Принцип **абстрагирования** — заключается в выделении существенных с некоторых позиций аспектов системы и отвлечении от несущественных с целью представления проблемы в простом общем виде.
2. Принцип **формализации** — заключается в необходимости строгого методического подхода к решению проблемы.
3. Принцип "**упрятывания**" — заключается в упрятывании несущественной на конкретном этапе информации: каждая часть "знает" только необходимую ей информацию.
4. Принцип **концептуальной общности** — заключается в следовании единой философии на всех этапах ЖЦ (структурный анализ — структурное проектирование — структурное программирование — структурное тестирование).
5. Принцип **полноты** — заключается в контроле присутствия лишних элементов.
6. Принцип **непротиворечивости** — заключается в обоснованности и согласованности элементов.
7. Принцип **логической независимости** — заключается в концентрации внимания на логическом проектировании для обеспечения независимости от физического проектирования.
8. Принцип **независимости данных** — заключается в том, что модели данных должны быть проанализированы и спроектированы независимо от процессов их логической обработки, а также от их физической структуры и распределения.
9. Принцип **структурирования данных** — заключается в том, что данные должны быть структурированы и иерархически организованы.

10. Принцип **доступа конечного пользователя** — заключается в том, что пользователь должен иметь средства доступа к базе данных, которые он может использовать непосредственно (без программирования).

Соблюдение указанных принципов необходимо при организации работ на начальных этапах ЖЦ независимо от типа разрабатываемой ИС и используемых при этом методологий. Руководствуясь всеми принципами в комплексе, можно на более ранних стадиях разработки понять, что будет представлять собой создаваемая система, обнаружить промахи и недоработки, что, в свою очередь, облегчит работы на последующих этапах ЖЦ и понизит стоимость разработки.

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;
* DFD (Data Flow Diagrams) — диаграммы потоков данных;
* ERD (Entity-Relationship Diagrams) —диаграммы "сущность—связь";
* STD (State Trasition Diagrams) — диаграммы переходов состояний.

На стадии проектирования ИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру ИС.

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

## Теоретические сведения о методологии IDEF0

**Постоянное усложнение** производственно-технических и организационно-экономических **систем** – фирм, предприятий, производств, и др. субъектов производственно-хозяйственной деятельности, и **необходимость их анализа** с целью совершенствования функционирования и повышения эффективности **обусловливают** необходимость применения **специальных средств** описания и анализа таких систем. Эта проблема приобретает особую актуальность в связи с появлением интегрированных компьютеризированных производств и автоматизированных предприятий.

В США это обстоятельство было осознано еще в конце 70-ых годов, когда ВВС США предложили и реализовали Программу интегрированной компьютеризации производства ICAM (**ICAM** - **I**ntegrated **C**omputer **A**ided **M**anufacturing),направленную на увеличение эффективности промышленных предприятий посредством широкого внедрения компьютерных (информационных) технологий.

Реализация программы ICAM потребовала создания **адекватных методов анализа** и проектирования производственных **систем** и способов обмена информацией между специалистами, занимающимися такими проблемами. Для удовлетворения этой потребности в рамках программы ICAM была разработана **методология** IDEF (**I**CAM **Def**inition), позволяющая исследовать структуру, параметры и характеристики производственно-технических и организационно-экономических систем (в дальнейшем, там, где это не вызывает недоразумений – **систем**). Общая методология IDEF состоит из **трех частных методологий** моделирования, основанных на **графическом** представлении систем:

* **IDEF0** используется для создания ***функциональной модели***, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции;
* **IDEF1** применяется для построения ***информационной модели***, отображающей структуру и содержание информационных потоков, необходимых для поддержки функций системы;
* **IDEF2** позволяет построить ***динамическую модель*** меняющихся во времени поведения функций, информации и ресурсов системы.

К настоящему времени наибольшее распространение и применение имеют методологии IDEF0 и IDEF1 (IDEF1X), получившие в США статус федеральных стандартов.

Методология IDEF0, особенности и приемы применения основана на подходе, разработанном **Дугласом Россом** в начале 70–ых годов и получившем название **SADT** (**S**tructured **A**nalysis & **D**esign **T**echnique – метод структурного анализа и проектирования). Основу подхода и, как следствие, методологии IDEF0, составляет **графический язык** описания (моделирования) систем, обладающий следующими свойствами.

* **Графический язык** – полное и выразительное средство, способное наглядно представлять широкий спектр деловых, производственных и других процессов и операций предприятия на любом уровне детализации;
* **Язык** обеспечивает **точное и лаконичное описание** моделируемых объектов, удобство использования и интерпретации этого описания;
* **Язык** облегчает **взаимодействие и взаимопонимание** системных аналитиков, разработчиков и **персонала** изучаемого объекта (фирмы, предприятия), т.е. служит средством **«информационного общения»** большого числа специалистов и **рабочих групп**, занятых в одном проекте, в процессе обсуждения, рецензирования, критики и утверждения результатов;
* **Язык** прошел многолетнюю проверку и продемонстрировал **работоспособность** как в проектах ВВС США, так и в других проектах, выполнявшихся государственными и частными промышленными компаниями;
* **Язык легок и прост** в изучении и освоении;
* **Язык** может генерироваться рядом инструментальных **средств машинной графики**; известны коммерческие программные продукты, поддерживающие разработку и анализ моделей - диаграмм IDEF0, например, продукт Design/IDEF 3.7 (и более поздние версии) фирмы Meta Software Corporation (США).

Перечисленные свойства языка предопределили выбор методологии IDEF0 в качестве базового средства анализа и синтеза производственно-технических и организационно-экономических систем.

Методология **IDEF0 основана** на следующих концептуальных **положениях**.

**1. Модель**– искусственный объект, представляющий собой отображение (**образ**) системы и ее компонентов. **М** *моделирует* **А**, *если* **М** *отвечает на вопросы относительно* **А**. Здесь **М** – модель, **А** – моделируемый объект (оригинал). Модель разрабатывают для понимания, анализа и принятия решений о реконструкции (реинжиниринге) или замене существующей, либо проектировании новой системы. ***Система*** представляет собой совокупность **взаимосвязанных и взаимодействующих частей**, выполняющих некоторую **полезную работу**. Частями (**элементами**) системы могут быть любые комбинации разнообразных сущностей, включающие людей, информацию, программное обеспечение, оборудование, изделия, сырье или энергию (энергоносители). **Модель** **описывает, что происходит в системе**, как ею управляют, какие сущности она преобразует, какие средства использует для выполнения своих функций и что производит.

**2. Блочное моделирование и его графическое представление.**Основной концептуальный **принцип** методологии IDEF – представление любой изучаемой системы в виде набора **взаимодействующих** и **взаимосвязанных блоков**, отображающих **процессы, операции, действия**, происходящие в изучаемой системе. В IDEF0 все, что происходит в системе и ее элементах, принято называть ***функциями***. Каждой функции ставится в соответствие ***блок***. На ***IDEF0 – диаграмме***, основном документе при анализе и проектировании систем, блок представляет собой прямоугольник. Интерфейсы, посредством которых блок взаимодействует с другими блоками или с внешней по отношению к моделируемой системе средой, представляются ***стрелками***,входящими в блок или выходящими из него. Входящие стрелки показывают, какие условия должны быть одновременно выполнены, чтобы функция, описываемая блоком, осуществилась.

**3. Лаконичность и точность.**Документация, описывающая систему, должна быть точной и лаконичной. Многословные характеристики, изложенные в форме традиционных текстов, неудовлетворительны. Графический язык позволяет лаконично, однозначно и точно показать все элементы (блоки) системы и все отношения и связи между ними, выявить ошибочные, лишние или дублирующие связи и т.д.

**4. Передача информации.**Средства IDEF0 облегчают передачу информации от одного участника разработки модели (отдельного разработчика или рабочей группы) к другому. К числу таких средств относятся:

* диаграммы, основанные на простой графике блоков и стрелок, легко читаемые и понимаемые;
* метки на естественном языке для описания блоков и стрелок, а также глоссарий и сопроводительный текст для уточнения смысла элементов диаграммы;
* последовательная декомпозиция диаграмм, строящаяся по иерархическому принципу, при котором на верхнем уровне отображаются основные функции, а затем происходит их детализация и уточнение;
* древовидные схемы иерархии диаграмм и блоков, обеспечивающие обозримость модели в целом и входящих в нее деталей.

**5. Строгость и формализм.**Разработка моделей IDEF0 требует соблюдения ряда строгих формальных правил, обеспечивающих преимущества методологии в отношении однозначности, точности и целостности сложных многоуровневых моделей. Отметим только основное из них: все стадии и этапы разработки и корректировки модели должны строго, формально документироваться с тем, чтобы при ее эксплуатации не возникало вопросов, связанных с неполнотой или некорректностью документации.

**6. Итеративное моделирование.**Разработка модели в IDEF0 представляет собой пошаговую, **итерационную процедуру**. На каждом шаге итерации разработчик предлагает вариант модели, который **подвергают обсуждению**, рецензированию и последующему редактированию, после чего цикл повторяется. Такая организация работы способствует оптимальному использованию знаний системного аналитика, владеющего методологией и техникой IDEF0, и знаний специалистов – экспертов в предметной области, к которой относится объект моделирования.

**7. Отделение «организации» от «функций».**При разработке моделей следует **избегать** изначальной «**привязки**» **функций** исследуемой системы **к** существующей организационной **структуре** моделируемого объекта (предприятия, фирмы). Это помогает избежать субъективной точки зрения, навязанной организацией и ее руководством. Организационная **структура** **должна** **явиться** **результатом использования** (применения) **модели**. Сравнение результата с существующей структурой позволяет, во-первых, оценить адекватность модели, а во-вторых – предложить решения, направленные на совершенствование этой структуры.

### Основные определения (понятия) методологии и языка IDEF0

1. **Блок:**прямоугольник, содержащий имя и номер и используемый для описания функции.

2. **Ветвление:**разделение стрелки на два или большее число сегментов. Может означать «развязывание пучка ».

3. **Внутренняя стрелка:**входная, управляющая или выходная стрелка, концы которой связывают источник и потребителя, являющиеся блоками одной диаграммы. Отличается от граничной стрелки.

4. **Входная стрелка:**класс стрелок, которые отображают вход IDEF0-блока, то есть данные или материальные объекты, которые преобразуются функцией в выход . Входные стрелки связываются с левой стороной блока IDEF0.

5. **Выходная стрелка:**класс стрелок, которые отображают выход IDEF0-блока, то есть данные или материальные объекты, произведенные функцией. Выходные стрелки связываются с правой стороной блока IDEF0.

6. **Глоссарий:**список определений для ключевых слов, фраз и аббревиатур, связанных с узлами, блоками, стрелками или с моделью IDEF0 в целом.

7. **Граничная стрелка:**стрелка, один из концов которой связан с источником или потребителем, а другой не присоединен ни к какому блоку на диаграмме. Отображает связь диаграммы с другими блоками системы и отличается от внутренней стрелки.

8. **Декомпозиция:**разделение моделируемой функции на функции - компоненты.

9. **Дерево узлов:**представление отношений между родительскими и дочерними узлами модели IDEF0 в форме древовидного графа. Имеет то же значение и содержание, что и перечень узлов.

10. **Диаграмма A-0:** специальный вид (контекстной) диаграммы IDEF0, состоящей из одного блока, описывающего функцию верхнего уровня, ее входы, выходы, управления, и механизмы, вместе с формулировками цели модели и точки зрения, с которой строится модель.

11. **Диаграмма:**часть модели, описывающая декомпозицию блока.

12. **Диаграмма** –**(FEO):** графическое описание, используемое, для сообщения специфических фактов о диаграмме IDEF0. При построении диаграмм FEO можно не придерживаться правила IDEF0.

13. **Дочерний блок:**блок на дочерней (порожденной) диаграмме.

14. **Дочерняя диаграмма:**диаграмма, детализирующая родительский (порождающий) блок.

15. **Имя блока:**глагол или глагольный оборот, помещенный внутри блока и описывающий моделируемую функцию.

16. **Интерфейс:**разделяющая граница, через которую проходят данные или материальные объекты; соединение между двумя или большим числом компонентов модели, передающее данные или материальные объекты от одного компонента к другому.

17. **Код ICOM:** аббревиатура (**I**nput – Вход, **C**ontrol – Управление, **O**utput – Выход, **M**echanism – Механизм), код, обеспечивающий соответствие граничных стрелок дочерней диаграммы со стрелками родительского блока; используется для ссылок.

18. **Контекст:**окружающая среда, в которой действует функция (или комплект функций на диаграмме).

19. **Контекстная диаграмма:**диаграмма, имеющая узловой номер A-n (**n>0**), которая представляет контекст модели, Диаграмма A-0, состоящая из одного блока, является необходимой (обязательной) контекстной диаграммой; диаграммы с узловыми номерами A-1, A-2,... – дополнительные контекстные диаграммы.

20. **Метка стрелки**: существительное или оборот существительного, связанные со стрелкой или сегментом стрелки и определяющие их значение.

21. **Модель IDEF0:**графическое описание системы, разработанное с определенной целью и с выбранной точки зрения. Комплект одной или более диаграмм IDEF0, которые изображают функции системы с помощью графики, текста и глоссария.

22. **Номер блока:** число (0 – 8), помещаемое в правом нижнем углу блока и однозначно идентифицирующее блок на диаграмме.

23. **Перечень узлов:**список, часто ступенчатый, показывающий узлы модели IDEF0 в упорядоченном виде. Имеет то же значение и содержание, что и дерево узлов.

24. **Примечание к модели:**текстовый комментарий, являющийся частью диаграммы IDEF0 и используемый для записи факта, не нашедшего графического изображения.

25. **Родительская диаграмма:**диаграмма, которая содержит родительский блок.

26. **Родительский блок:**блок, который подробно описывается дочерней диаграммой.

27. **Связывание/развязывание:** объединение значений стрелок в составное значение (связывание в «пучок»), или разделение значений стрелок (развязывание «пучка»), выраженные синтаксисом слияния или ветвления стрелок.

28. **Сегмент стрелки:** сегмент линии, который начинается или заканчивается на стороне блока, в точке ветвления или слияния, или на границе (конец стрелки).

29. **Семантика:**значение синтаксических компонентов языка.

30. **Синтаксис:**структурные компоненты или характеристики языка и правила, которые определяют отношения между ними.

31. **Слияние:**объединение двух или большего числа сегментов стрелок в один сегмент. Может означать «развязывание пучка»

32. **С-номер:** номер, создаваемый в хронологическом порядке и используемый для идентификации диаграммы и прослеживания ее истории; может быть использован в качестве ссылочного выражения при определении конкретной версии диаграммы.

33. **Стрелка:** направленная линия, состоящая из одного или нескольких сегментов, которая моделирует открытый канал или канал, передающий данные или материальные объекты от источника (точка стрелки), к потребителю (конечная точка с «наконечником»). Имеется 4 класса стрелок: входная стрелка, выходная стрелка, управляющая стрелка, стрелка механизма (включает стрелку вызова).

34. **Стрелка вызова:**вид стрелки механизма, который обозначает обращение из блока данной модели (части модели) к блоку другой модели (или другой части той же модели) и обеспечивает связь между моделями или между разными частями одной модели.

35. **Стрелка механизма:**класс стрелок, которые отображают механизмы IDEF0, то есть средства, используемые для выполнения функции; включает специальный случай стрелки вызова. Стрелки механизмов связываются с нижней стороной блока IDEF0.

36. **Стрелка, помещенная в туннель (туннельная стрелка):**стрелка (со специальной нотацией), не удовлетворяющая обычному требованию, согласно которому каждая стрелка на дочерней диаграмме должна соответствовать стрелкам на родительской диаграмме.

37. **Текст:**любой текстовый (графический) комментарий к графической диаграмме IDEF0.

38. **Тильда:**небольшая ломаная (волнистая) линия, используемая для соединения метки с конкретным сегментом стрелки или примечания модели с компонентом диаграммы.

39. **Точка зрения:**указание на должностное лицо или подразделение организации, с позиции которого разрабатывается модель.

40 **Узел:**блок, порождающий дочерние блоки; родительский блок.

41. **Узловая ссылка:**код, присвоенный диаграмме, для ее идентификации и определения положения в иерархии модели; формируется из сокращенного имени модели и узлового номера диаграммы с дополнительными расширениями.

42. **Узловой номер диаграммы:**часть узловой ссылки диаграммы, которая соответствует номеру родительского блока.

43. **Узловой номер:**код, присвоенный блоку и определяющий его положение в иерархии модели; может быть использован в качестве подробного ссылочного выражения.

44. **Управляющая стрелка:** класс стрелок, которые в IDEF0 отображают управления, то есть условия, при выполнении которых выход блока будет правильным. Данные или объекты, моделируемые как управления, могут преобразовываться функцией, создающей соответствующий выход. Управляющие стрелки связываются с верхней стороной блока IDEF0.

45. **Функция:**деятельность, процесс или преобразование (моделируемые блоком IDEF0), идентифицируемое глаголом или глагольной формой, которая описывает, что должно быть выполнено.

46. **Цель:**краткая формулировка причины создания модели.

**Лекция №17**

**Содержание лекции**

[Средство автоматизированного проектирования BPwin 1](#_Toc326826859)

[Синтаксис языка IDEF0 1](#_Toc326826860)

[Блок 1](#_Toc326826861)

[Стрелка 2](#_Toc326826862)

[Синтаксические правила 2](#_Toc326826863)

[Блоки 2](#_Toc326826864)

[Стрелки 2](#_Toc326826865)

[Семантика языка IDEF0 3](#_Toc326826866)

[Семантика блоков и стрелок 3](#_Toc326826867)

[Имена и метки 4](#_Toc326826868)

[Семантические правила блоков и стрелок 4](#_Toc326826869)

[Диаграммы IDEF0 5](#_Toc326826870)

[Контекстная диаграмма верхнего уровня 5](#_Toc326826871)

[Дочерняя диаграмма 7](#_Toc326826872)

[Родительская диаграмма 7](#_Toc326826873)

[Создание контекстной диаграммы на основе IDEF0 8](#_Toc326826874)

[Средство автоматизированного проектирования BPwin 13](#_Toc326826875)

# Средство автоматизированного проектирования BPwin

## Синтаксис языка IDEF0

Набор структурных компонентов языка, их характеристики и правила, определяющие связи между компонентами, представляют собой синтаксис языка. Компоненты синтаксиса IDEF0 – блоки, стрелки, диаграммы и правила. Блоки представляют функции, определяемые как деятельность, процесс, операция, действие или преобразование (см. ниже). Стрелки представляют данные или материальные объекты, связанные с функциями. Правила определяют, как следует применять компоненты; диаграммы обеспечивают формат графического и словесного описания моделей. Формат образует основу для управления конфигурацией модели.

#### Блок

Блок описывает функцию. Типичный блок показан на рис. 1. Внутри каждого блока помещается его имя и номер. Имя должно быть активным глаголом или глагольным оборотом, описывающим функцию. Номер блока размещается в правом нижнем углу. Номера блоков используются для их идентификации на диаграмме и в соответствующем тексте.

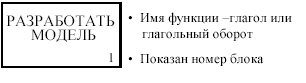


Рис. 1.

#### Стрелка

Стрелка формируется из одного или более отрезков прямых и наконечника на одном конце. Как показано на рис. 2, сегменты стрелок могут быть прямыми или ломаными; в последнем случае горизонтальные и вертикальные отрезки стрелки сопрягаются дугами, имеющими угол 90о. Стрелки не представляют поток или последовательность событий, как в традиционных блок-схемах потоков или процессов. Они лишь показывают, какие данные или материальные объекты должны поступить на вход функции для того, чтобы эта функция могла выполняться. Рис. 2.

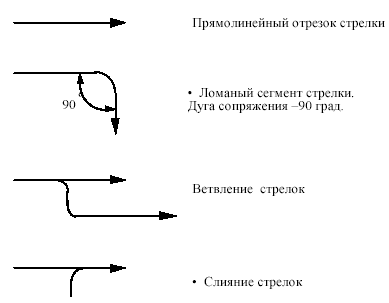


Рис. 2.

### Синтаксические правила

#### Блоки

1.Размеры блоков должны быть достаточными для того, чтобы включить имя блока.

2.Блоки должны быть прямоугольными, с прямыми углами.

3.Блоки должны быть нарисованы сплошными линиями.

#### Стрелки

1. Ломаные стрелки изменяют направление только под углом 90 о.

2. Стрелки должны быть нарисованы сплошными линиями различной толщины.

3. Стрелки могут состоять только из вертикальных или горизонтальных отрезков; отрезки, направленные по диагонали, не допускаются.

4. Концы стрелок должны касаться внешней границы функционального блока, но не должны пересекать ее.

5.Стрелки должны присоединяться к блоку на его сторонах. Присоединение в углах не допускается.

## Семантика языка IDEF0

Семантика определяет содержание (значение) синтаксических компонентов языка и способствует правильности их интерпретации. Интерпретация устанавливает соответствие между блоками и стрелками с одной стороны и функциями и их интерфейсами – с другой.

### Семантика блоков и стрелок

Поскольку IDEF0 есть методология функционального моделирования, имя блока, описывающее функцию, должно быть глаголом или глагольным оборотом; например, имя блока "Выполнить проверку", означает, что блок с таким именем превращает непроверенные детали в проверенные. После присваивания блоку имени, к соответствующим его сторонам присоединяются входные, выходные и управляющие стрелки, а также стрелки механизма, что и определяет наглядность и выразительность изображения блока IDEF0.

Чтобы гарантировать точность модели, следует использовать стандартную терминологию. Блоки именуются глаголами или глагольными оборотами и эти имена сохраняются при декомпозиции Стрелки и их сегменты, как отдельные, так и связанные в «пучок», помечаются существительными или оборотами существительного. Метки сегментов позволяют конкретизировать данные или материальные объекты, передаваемые этими сегментами, с соблюдением синтаксиса ветвлений и слияний.

Каждая сторона функционального блока имеет стандартное значение с точки зрения связи блок/стрелки, В свою очередь, сторона блока, к которой присоединена стрелка, однозначно определяет ее роль. Стрелки, входящие в левую сторону блока - входы. Входы преобразуются или расходуются функцией, чтобы создать то, что появится на ее выходе. Стрелки, входящие в блок сверху - управления. Управления определяют условия, необходимые функции, чтобы произвести правильный выход. Стрелки, покидающие блок справа – выходы, т.е. данные или материальные объекты, произведенные функцией.

Стрелки, подключенные к нижней стороне блока, представляют механизмы. Стрелки, направленные вверх, идентифицируют средства, поддерживающие выполнение функции. Другие средства могут наследоваться из родительского блока. Стрелки механизма, направленные вниз, являются стрелками вызова. Стрелки вызова обозначают обращение из данной модели или из данной части модели к блоку, входящему в состав другой модели или другой части модели, обеспечивая их связь, т.е. разные модели или разные части одной и той же модели могут совместно использовать один и тот же элемент (блок). Стандартное расположение стрелок показано на рис.3.

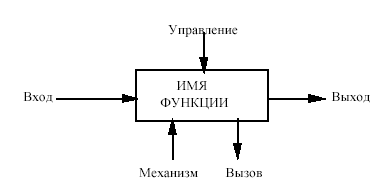


Рис 3.

### Имена и метки

Как указывалось, имена функций – глаголы или глагольные обороты. Примеры таких имен:



Стрелки идентифицируют данные или материальные объекты, необходимые для выполнения функции или производимые ею. Каждая стрелка должна быть помечена существительным или оборотом существительного, например:



Пример размещения меток стрелок и имени блока показан на рис. 4.

### Семантические правила блоков и стрелок

1. Имя блока должно быть активным глаголом или глагольным оборотом.

2. Каждая сторона функционального блока должна иметь стандартное отношение блок/стрелки:

а) входные стрелки должны связываться с левой стороной блока;

б) управляющие стрелки должны связываться с верхней стороной блока;

в) выходные стрелки должны связываться с правой стороной блока;

г) стрелки механизма (кроме стрелок вызова) должны указывать вверх и подключаться к нижней стороне блока.

д) стрелки вызова механизма должны указывать вниз, подключаться к нижней стороне блока, и помечаться ссылкой на вызываемый блок.

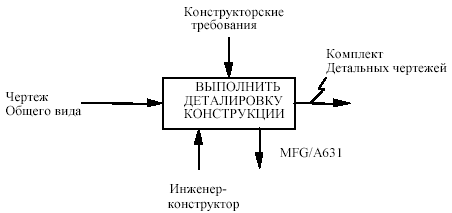


Рис. 4.

3. Сегменты стрелок, за исключением стрелок вызова, должны помечаться существительным или оборотом существительного, если только единственная метка стрелки несомненно не относится к стрелке в целом.

4. Чтобы связать стрелку с меткой, следует использовать "загогулину" ()



5. В метках стрелок не должны использоваться следующие термины: функция, вход, управление, выход, механизм, вызов.

## Диаграммы IDEF0

IDEF0-модели состоят из трех типов документов: графических диаграмм, текста и глоссария. Эти документы имеют перекрестные ссылки друг на друга. Графическая диаграмма – главный компонент IDEF0-модели, содержащий блоки, стрелки, соединения блоков и стрелок и ассоциированные с ними отношения. Блоки представляют основные функции моделируемого объекта. Эти функции могут быть разбиты (декомпозированы) на составные части и представлены в виде более подробных диаграмм; процесс декомпозиции продолжается до тех пор, пока объект не будет описан на уровне детализации, необходимом для достижения целей конкретного проекта. Диаграмма верхнего уровня обеспечивает наиболее общее или абстрактное описание объекта моделирования. За этой диаграммой следует серия дочерних диаграмм, дающих более детальное представление об объекте.

### Контекстная диаграмма верхнего уровня

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком с граничными стрелками. Эта диаграмма называется A-0. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой. Поскольку единственный блок представляет весь объект, его имя общее для всего проекта. Это же справедливо и для всех стрелок диаграммы, поскольку они представляют полный комплект внешних интерфейсов объекта. Диаграмма A-0 устанавливает область моделирования и ее границу. Пример диаграммы A-0 показан на рис. 5.

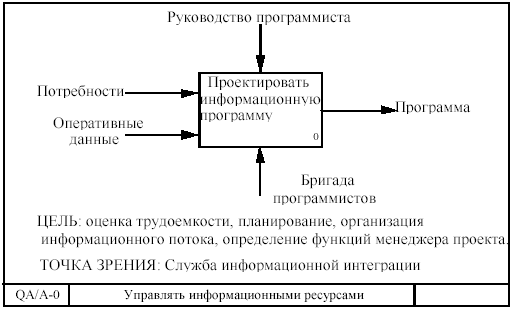


Рис. 5.

Контекстная диаграмма A-0 также должна содержать краткие утверждения, определяющие точку зрения должностного лица или подразделения, с позиций которого создается модель, и цель, для достижения которой ее разрабатывают. Эти утверждения помогают руководить разработкой модели и ввести этот процесс в определенные рамки. Точка зрения определяет, что и в каком разрезе можно увидеть в пределах контекста модели. Изменение точки зрения, приводит к рассмотрению других аспектов объекта. Аспекты, важные с одной точки зрения, могут не появиться в модели, разрабатываемой с другой точки зрения на тот же самый объект.

Формулировка цели выражает причину создания модели, т.е. содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет ее структуру. Наиболее важные свойства объекта обычно выявляются на верхних уровнях иерархии; по мере декомпозиции функции верхнего уровня и разбиения ее на подфункции, эти свойства уточняются. Каждая подфункция, в свою очередь, декомпозируется на элементы следующего уровня, и так происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования. Каждая подфункция моделируется отдельным блоком. Каждый родительский блок подробно описывается дочерней диаграммой на более низком уровне. Все дочерние диаграммы должны быть в пределах области контекстной диаграммы верхнего уровня.

### Дочерняя диаграмма

Единственная функция, представленная на контекстной диаграмме верхнего уровня, может быть разложена на основные подфункции посредством создания дочерней диаграммы. В свою очередь, каждая из этих подфункций может быть разложена на составные части посредством создания дочерней диаграммы следующего, более низкого уровня, на которой некоторые или все функции также могут быть разложены на составные части. Каждая дочерняя диаграмма содержит дочерние блоки и стрелки, обеспечивающие дополнительную детализацию родительского блока.

Дочерняя диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский блок, но описывает ее более подробно. Таким образом, дочерняя диаграмма как бы вложена в свой родительский блок. Эта структура иллюстрируется рис.6.

### Родительская диаграмма

Родительская диаграмма – та, которая содержит один или более родительских блоков. Каждая обычная (неконтекстная) диаграмма является также дочерней диаграммой, поскольку, по определению, она подробно описывает некоторый родительский блок. Таким образом, любая диаграмма может быть как родительской диаграммой (содержать родительские блоки), так и дочерней (подробно описывать собственный родительский блок). Аналогично, блок может быть как родительским (подробно описываться дочерней диаграммой) так и дочерним (появляющимся на дочерней диаграмме). Основное иерархическое отношение существует между родительским блоком и дочерней диаграммой, которая его подробно описывает (рис.6).

То, что блок является дочерним и раскрывает содержание родительского блока на диаграмме предшествующего уровня, указывается специальным ссылочным кодом, написанным ниже правого нижнего угла блока. Этот ссылочный код может формироваться несколькими способами, из которых самый простой заключается в том, что код, начинающийся с буквы А (по имени диаграммы А-0), содержит цифры, определяемые номерами родительских блоков. Например, показанные на рис.7 коды означают, что диаграмма является декомпозицией 1-го блока диаграммы, которая, в свою очередь является декомпозицией 6-го блока диаграммы А0, а сами коды образуются присоединением номера блока.

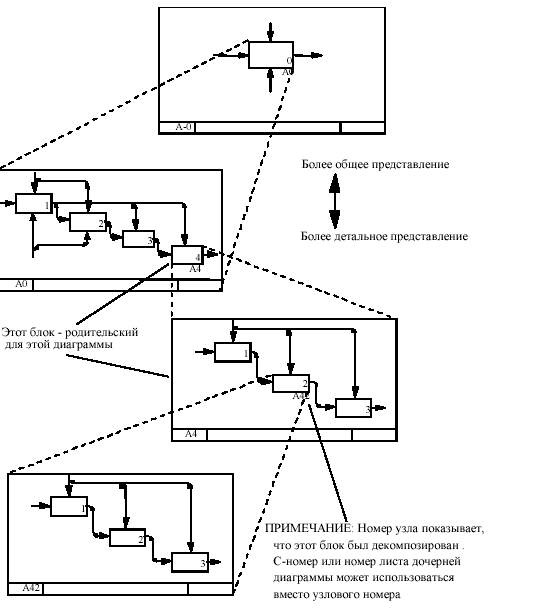


Рис. 6.

Таким образом, код формируется так:

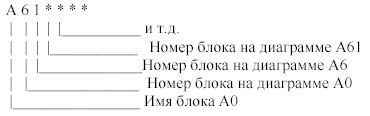


Рис. 7.

## Создание контекстной диаграммы на основе IDEF0

1. В составе модели должна присутствовать контекстная диаграмма A-0, которая содержит только один блок. Номер единственного блока на контекстной диаграмме A-0 должен быть 0.

2. Блоки на диаграмме должны располагаться по диагонали – от левого верхнего угла диаграммы до правого нижнего в порядке присвоенных номеров. Блоки на диаграмме, расположенные вверху слева «**доминируют**» над блоками, расположенными внизу справа. «**Доминирование**» понимается как влияние, которое блок оказывает на другие блоки диаграммы. Расположение блоков на листе диаграммы отражает авторское понимание доминирования.

Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные.

3. Неконтекстные диаграммы должны содержать не менее трех и не более шести блоков. Эти ограничения поддерживают сложность диаграмм на уровне, доступном для чтения, понимания и использования.

Диаграммы с количеством блоков менее трех вызывают серьезные сомнения в необходимости декомпозиции родительской функции. Диаграммы с количеством блоков более шести сложны для восприятия читателями и вызывают у автора трудности при внесении в нее всех необходимых графических объектов и меток.

4. Каждый блок неконтекстной диаграммы получает номер, помещаемый в правом нижнем углу; порядок нумерации - от верхнего левого к нижнему правому блоку (номера от 1 до 8).

5. Каждый блок, подвергнутый декомпозиции, должен иметь ссылку на дочернюю диаграмму; ссылка (например, узловой номер, C-номер или номер страницы) помещается под правым нижним углом блока.

6. Имена блоков (выполняемых функций) и метки стрелок должны быть уникальными. Если метки стрелок совпадают, это значит, что стрелки отображают тождественные данные.

7. При наличии стрелок со сложной топологией целесообразно повторить метку для удобства ее идентификации.

8. Следует обеспечить максимальное расстояние между блоками и поворотами стрелок, а также между блоками и пересечениями стрелок для облегчения чтения диаграммы. Одновременно уменьшается вероятность перепутать две разные стрелки.

9. Блоки всегда должны иметь хотя бы одну управляющую и одну выходную стрелку, но могут не иметь входных стрелок.

10. Если одни и те же данные служат и для управления, и для входа, вычерчивается только стрелка управления. Этим подчеркивается управляющий характер данных и уменьшается сложность диаграммы.

11. Максимально увеличенное расстояние между параллельными стрелками облегчает размещения меток, их чтение и позволяет проследить пути стрелок (рис. 8).

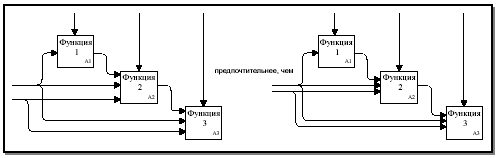


Рис. 8.

12. Стрелки связываются (сливаются), если они представляют сходные данные и их источник не указан на диаграмме (рис. 9).

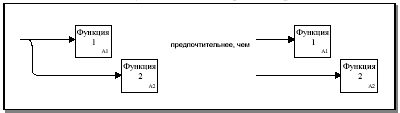


Рис. 9.

13. Обратные связи по управлению должны быть показаны как "вверх и над" (рис. 10, а):

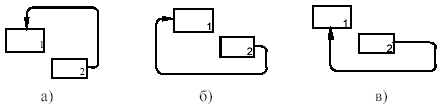


Рис. 10.

Обратные связи по входу должны быть показаны как "вниз и под" (рис. 10,б). Так же показываются обратные связи посредством механизма. Таким образом, обеспечивается показ обратной связи при минимальном числе линий и пересечений.

14. Циклические обратные связи для одного и того же блока изображаются только для того, чтобы их выделить. Обычно обратную связь изображают на диаграмме, декомпозирующей блок. Однако иногда требуется выделить повторно используемые объекты (рис.11).

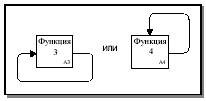


Рис 11.

15. Стрелки объединяются, если они имеют общий источник или приемник, или они представляют связанные данные. Общее название лучше описывает суть данных. Следует минимизировать число стрелок, касающихся каждой стороны блока, если, конечно, природа данных не слишком разнородна (рис.12).

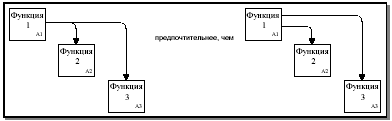


Рис 12.

16. Если возможно, стрелки присоединяются к блокам в одной и той же позиции. Тогда соединение стрелок конкретного типа с блоками будет согласованным и чтение диаграммы упростится.

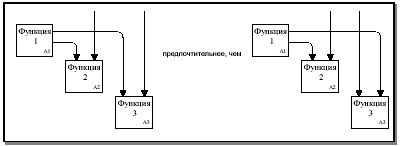


Рис. 13.

17. При соединении большого числа блоков необходимо избегать необязательных пересечений стрелок. Следует минимизировать число петель и поворотов каждой стрелки.

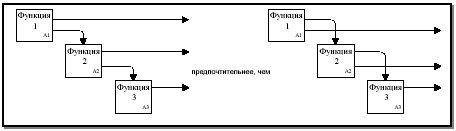


Рис 14.

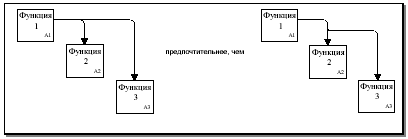


Рис 15.

18. Блоки (функции) являются сопряженными через среду, если они имеют связи с источником, генерирующим данные, без конкретного определения отношения отдельной части данных к какому-либо блоку.

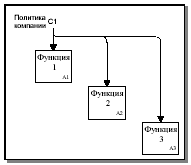


Рис. 16

19. Две или более функций являются сопряженными через запись, если они связаны с набором данных и не обязательно зависят от того, представлены ли все возможные интерфейсы как сопряжение через среду. Тип интерфейса, показанный на рисунке 17, предпочтителен, поскольку определяются отношения конкретных элементов данных к каждому блоку.

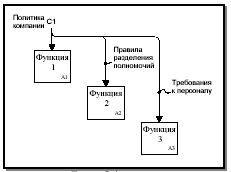


Рис. 17.

20. Необходимо использовать (где это целесообразно) выразительные возможности ветвящихся стрелок.

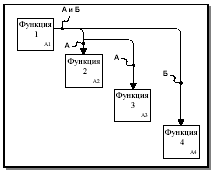


Рис 18.

## Средство автоматизированного проектирования BPwin

BPwin-CASE-средство, позволяющее документировать бизнес-процедуры графически. Описание бизнес-процедур на языке IDEF-диаграмм будет одновременно и формально строгим, и достаточно лаконичным.

PLATINUM BPwin – инструмент моделирования, который используется для анализа, документирования и реорганизации сложных бизнес-процессов. Модель, созданная средствами BPwin, позволяет четко документировать различные аспекты деятельности – действия, которые необходимо предпринять, способы их осуществления, требующиеся для этого ресурсы и др. Таким образом, формируется целостная картина деятельности предприятия от моделей организации работы в маленьких отделах до сложных иерархических структур. При разработке или закупке программного обеспечения модели бизнес-процессов служат прекрасным средством документирования потребностей, помогая обеспечить высокую эффективность инвестиций в информационные технологии.

**Лекция №18**

**Содержание лекции**

[Средство автоматизированного проектирования BPwin 1](#_Toc326829329)

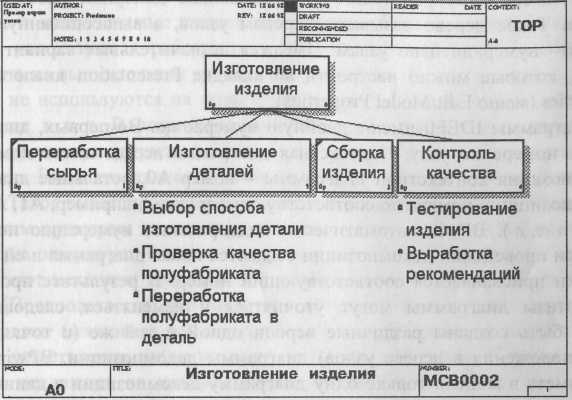
[Построение диаграмм дерева узлов и FEO 1](#_Toc326829330)

[Стоимостный анализ (ABC) 3](#_Toc326829331)

# Средство автоматизированного проектирования BPwin

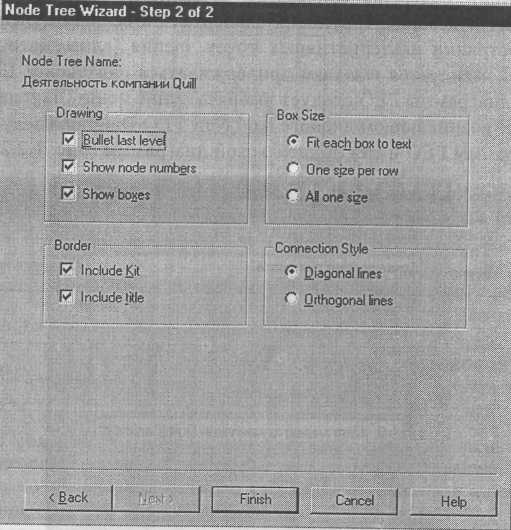
## Построение диаграмм дерева узлов и FEO

Диаграмма дерева узлов показывает иерархию работ в модели и позволяет рассмотреть всю модель целиком, но не показывает взаимосвязи между работами (стрелки) (рис. 1). Процесс создания модели работ является итерационным, следовательно, работы могут менять свое расположение в дереве узлов многократно. Чтобы не запутаться и проверить способ декомпозиции, следует после каждого изменения создавать диаграмму дерева узлов. Впрочем, BPwin имеет мощный инструмент навигации по модели – Model Explorer, который позволяет представить иерархию работ и диаграмм в удобном и компактном виде, однако этот инструмент не является составляющей стандарта IDEF0.



*Рис. 1. Диаграмма дерева узлов*

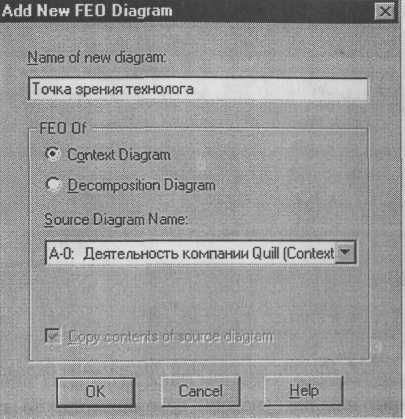
Для создания диаграммы дерева узлов следует выбрать в меню пункт Diagram/Add Node Tree. Возникает эксперт создания диаграммы дерева узлов Node Tree Wizard. В первом диалоге эксперта необходимо внести имя диаграммы дерева узлов, узел верхнего уровня и глубину дерева – Number of Levels (по умолчанию 3). Поскольку дерево узлов не обязательно в качестве верхнего уровня должна иметь контекстную работу и иметь произвольную глубину. В одной модели можно создавать множество диаграмм деревьев узлов. Имя дерева узлов по умолчанию совпадает с именем работы верхнего уровня, а номер диаграммы автоматически генерируется как номер узла верхнего уровня плюс литера "N", например A0N. Если в модели создается два дерева узлов, имеющих в качестве верхнего уровня одну и ту же работу, то по умолчанию диаграммы получат идентичные номер и имя. Поэтому рекомендуется при создании диаграммы дерева узлов внести имя диаграммы, отличное от значения по умолчанию. Второй диалог эксперта Node Tree Wizard (рис. 2) позволяет задать свойства диаграммы дерева узлов.



*Рис. 2. Диалог настройки диаграммы дерева узлов*

По умолчанию нижний уровень декомпозиции показывается в виде списка, остальные работы – в виде прямоугольников. Для отображения всего дерева в виде прямоугольников следует выбрать опцию Bullet Last Level. Труппа Connection Style позволяет выбрать стиль соединительных линий - диагональные (по умолчанию) или ортогональные.

Диаграммы "только для экспозиции" (FEO) часто используются в модели для иллюстрации других точек зрения, для отображения отдельных деталей, которые не поддерживаются явно синтаксисом IDEF0. Диаграммы FEO позволяют нарушить любое синтаксическое правило, поскольку по сути являются просто картинками - копиями стандартных диаграмм и не включаются в анализ синтаксиса. Например, работа на диаграмме FEO может не иметь стрелок управления и выхода. С целью обсуждения определенных аспектов модели с экспертом предметной области может быть создана диаграмма только с одной работой и одной стрелкой, поскольку стандартная диаграмма декомпозиции содержит множество деталей, не относящихся к теме обсуждения и дезориентирующих эксперта. Но если FEO используется для иллюстрации альтернативных точек зрения (альтернативный контекст), рекомендуется все-таки придерживаться синтаксиса IDEF0. Для создания диаграммы FEO следует выбрать пункт меню Diagram/Add FEO Diagram. В возникающем диалоге Add New FEO Diagram следует указать имя диаграммы FEO и тип родительской диаграммы (рис. 3).



*Рис. 3. Диалог создания FEO-диаграммы*

Новая диаграмма получает номер, который генерируется автоматически (номер родительской диаграммы по узлу + постфикс F, например A1F).

## Стоимостный анализ (ABC)

Обычно сначала строится функциональная модель существующей организации работы – AS-IS (Как есть). После построения модели AS-IS проводится анализ бизнес-процессов, потоки данных и объектов перенаправляются и улучшаются, в результате строится модель ТО-ВЕ. Как правило, строится несколько моделей ТО-ВЕ, из которых по какому-либо критерию выбирается наилучшая. Проблема состоит в том, что таких критериев много и непросто определить важнейший. Для того чтобы определить качество созданной модели с точки зрения эффективности бизнес-процессов, необходима система метрики, т. е. качество следует оценивать количественно.

BPwin предоставляет аналитику два инструмента для оценки модели – стоимостный анализ, основанный на работах (Activity Based Costing, ABC), и свойства, определяемые пользователем (User Defined Properties, UDP). ABC является широко распространенной методикой, используемой международными корпорациями и государственными организациями (в том числе Департаментом обороны США) для идентификации истинных движителей затрат в организации.

Стоимостный анализ представляет собой соглашение об учете, используемое для сбора затрат, связанных с работами, с целью определить общую стоимость процесса. Стоимостный анализ основан на модели работ, потому что количественная оценка невозможна без детального понимания функциональности предприятия. Обычно ABC применяется для того, чтобы понять происхождение выходных затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия (Business Process Re-engineering, BPR). С помощью стоимостного анализа можно решить такие задачи, как определение действительной стоимости производства продукта, определение действительной стоимости поддержки клиента, идентификация работ, которые стоят больше всего (те, которые должны быть улучшены в первую очередь), обеспечение менеджеров финансовой мерой предлагаемых изменений, и др.

ABC может проводиться только тогда, когда модель работы последовательная (следует синтаксическим правилам IDEF0), корректная (отражает бизнес), полная (охватывает всю рассматриваемую область) и стабильная (проходит цикл экспертизы без изменений), другими словами, создание модели работы закончено.

ABC включает следующие основные понятия:

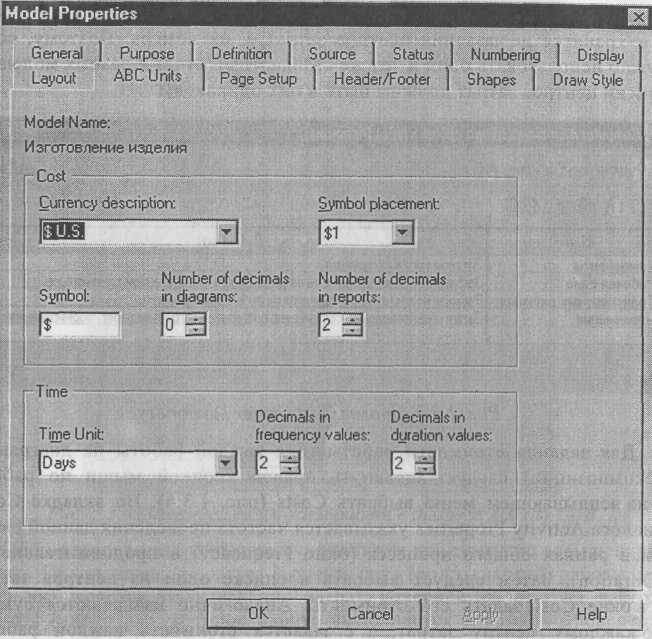
* объект затрат – причина, по которой работа выполняется, обычно, основной выход работы, стоимость работ есть суммарная стоимость объектов затрат ***("Готовое изделие"***, рис. 4).
* движитель затрат – характеристики входов и управлений работы (***"Сырье"***, ***"Чертеж"***,рис. 4),которые влияют на то, как выполняется и как долго длится работа;
* центры затрат, которые можно трактовать как статьи расхода.



*Рис. 4. Иллюстрация терминов ABC*

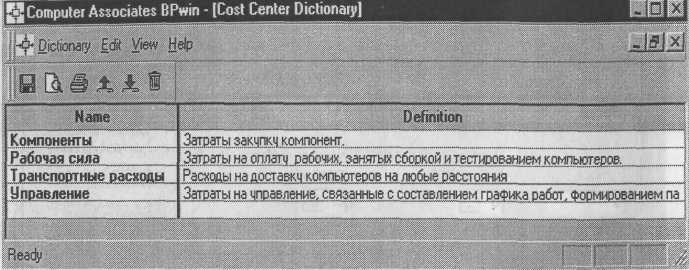
При проведении стоимостного анализа в BPwin сначала задаются единицы измерения времени и денег. Для задания единиц измерения следует вызвать диалог Model Properties (меню Edit/Model Properties), вкладка ABC Units (рис. 5).

Если в списке выбора отсутствует необходимая валюта (например, рубль), ее можно добавить. Символ валюты по умолчанию берется из настроек Windows. Диапазон измерения времени в списке Time Unit достаточен для большинства случаев – от секунд до лет.



*Рис. 5. Настройка единиц измерения валюты и времени*

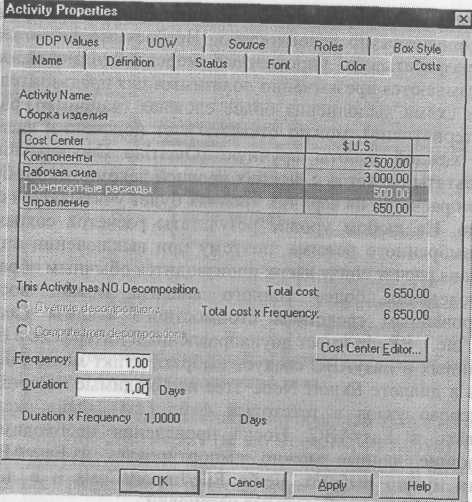
Затем описываются центры затрат (cost centers). Для внесения центров затрат необходимо вызвать диалог Cost Center Dictionary (меню Dictionary /Cost Center (рис. 6).



*Рис. 6. Диалог Cost Center Dictionary*

Каждому центру затрат следует дать подробное описание в окне Definition. Список центров затрат упорядочен. Порядок в списке можно менять при помощи стрелок, расположенных справа от списка. Задание определенной последовательности центров затрат в списке, во-первых, облегчает последующую работу при присвоении стоимости работам, а во-вторых, имеет значение при использовании единых стандартных отчетов в разных моделях. Хотя BPwin сохраняет информацию о стандартном отчете в файле BPWINRPT.INI, информация о центрах затрат и UDP сохраняется в виде указателей, т.е. хранятся не названия центров затрат, а их номера. Поэтому, если нужно использовать один и тот же стандартный отчет в разных моделях, списки центров затрат должны быть в них одинаковы.

Для задания стоимости работы (для каждой работы на диаграмме декомпозиции) следует щелкнуть правой кнопкой мыши по работе и на всплывающем меню выбрать Costs (рис. 7). Во вкладке Costs диалога Activity Properties указывается частота проведения данной работы в рамках общего процесса (окно Frequency) и продолжительность (Duration). Затем следует выбрать в списке один из центров затрат и в окне Cost задать его стоимость. Аналогично назначаются суммы по каждому центру затрат, т. е. задается стоимость каждой работы по каждой статье расхода. Если в процессе назначения стоимости возникает необходимость внесения дополнительных центров затрат, диалог Cost Center Editor вызывается прямо из диалога Activity Cost соответствующей кнопкой.



*Рис. 7. Задание стоимости работ в диалоге Activity Cost*

Общие затраты по работе рассчитываются как сумма по всем центрам затрат. При вычислении затрат вышестоящей (родительской) работы сна чала вычисляется произведение затрат дочерней работы на частоту работы (число раз, которое работа выполняется в рамках проведения родительской работы), затем результаты складываются. Если во всех работах модели включен режим Compute from Decompositions, подобные вычисления автоматически проводятся по всей иерархии работ снизу вверх (рис. 8).



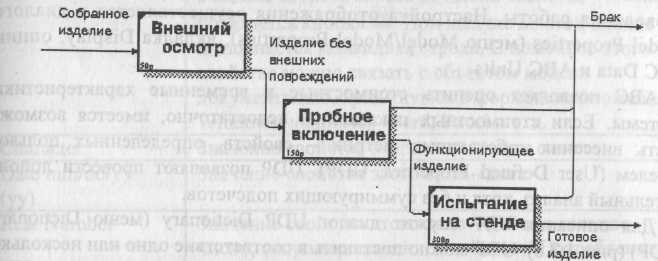
*Рис. 8. Вычисление затрат родительской работы*

Этот достаточно упрощенный принцип подсчета справедлив, если работы выполняются последовательно. Встроенные возможности BPwin позволяют разрабатывать упрощенные модели стоимости, которые тем не менее оказываются чрезвычайно полезными для предварительной оценки затрат. Если схема выполнения более сложная (например, работы производятся альтернативно), можно отказаться от подсчета и задать итоговые суммы для каждой работы вручную (Override Decompositions). В этом случае результаты расчетов с нижних уровней декомпозиции будут игнорироваться, при расчетах на верхних уровнях будет учитываться сумма, заданная вручную. На любом уровне результаты расчетов сохраняются независимо от выбранного режима, поэтому при выключении опции Override Decompositions расчет снизу вверх производится обычным образом.

Для проведения более тонкого анализа можно воспользоваться специализированным средством стоимостного анализа EasyABC (ABC Technology, Inc.). BPwin имеет двунаправленный интерфейс с EasyABC. Для экспорта данных в EasyABC следует выбрать пункт меню File/Export/Node Tree, задать в диалоге Export Node Tree необходимые настройки и экспортировать дерево узлов в текстовый файл (.txt). Файл экспорта можно импортировать в EasyABC. После проведения необходимых расчетов результирующие данные можно импортировать из EasyABC в BPwin. Для импорта нужно выбрать меню File/Import/Costs и в диалоге Import Activity Costs выбрать необходимые установки.

Результаты стоимостного анализа могут существенно повлиять на очередность выполнения работ. Рассмотрим пример, изображенный на рис. 9. Предположим, что для оценки качества изделия необходимо провести три работы:

* внешний осмотр – стоимость 50 руб.;
* пробное включение – стоимость 150 руб.;
* испытание на стенде – стоимость 300 руб.



*Рис. 9. Фрагмент диаграммы декомпозиции работы "Контроль качества"*

Предположим также, что с точки зрения технологии очередность проведения работ несущественна, а вероятность выявления брака одинакова (50%). Пусть необходимо проверить восемь изделий. Если проводить работы в убывающем по стоимости порядке, то стоимость, затраченная на получение готового изделия, составит:

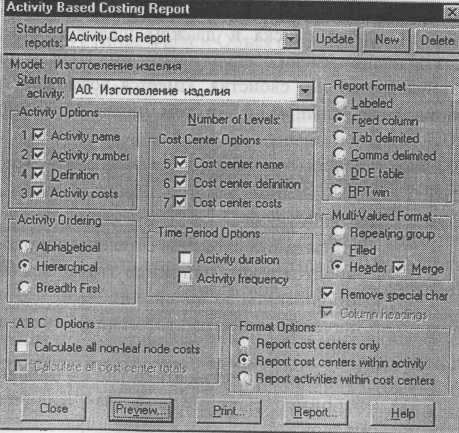
300 руб. (Испытание на стенде)\*8 +150 руб. (Пробное включение)\*4 + 50 руб. (Внешний осмотр)\*2 = 3100 руб.

Если проводить работы в возрастающем по стоимости порядке, то стоимость, затраченная на получение готового изделия, составит:

50 руб. (Внешний осмотр)\*8+150 руб. (Пробное включение)\*4 + 300 руб. (Испытание на стенде)\*2 = 1600 руб.

Следовательно, с целью минимизации затрат первой должна быть выполнена наиболее дешевая работа, затем - средняя по стоимости и в конце -наиболее дорогая (см. рис. 9).

Результаты стоимостного анализа наглядно представляются на специальном отчете BPwin – Activity Cost Report (меню Tools/Report/Activity Cost Report). Отчет позволяет документировать имя, номер, определение и стоимость работ, как суммарную, так и раздельно по центрам затрат (рис. 10).



*Рис. 10. Диалог настройки отчета по стоимости работ*

Результаты отображаются и непосредственно на диаграммах. В левом нижнем углу прямоугольника работы может показываться либо стоимость (по умолчанию), либо продолжительность, либо частота проведения работы. Настройка отображения осуществляется в диалоге Model Properties (меню Model/Model Properties), вкладка Display, опции ABC Data и ABC Units.

ABC позволяет оценить стоимостные и временные характеристики системы. Если стоимостных показателей недостаточно, имеется возможность внесения собственных метрик - свойств, определенных пользователем (User Defined Properties, UDP). UDP позволяют провести дополнительный анализ, хотя и без суммирующих подсчетов.

**Лекция №19**

**Содержание лекции**

[Средство автоматизированного проектирования BPwin 1](#_Toc326835141)

[Дополнение созданной модели процессов, диаграммами DFD 1](#_Toc326835142)

[Моделирование потоков данных (процессов) 4](#_Toc326835143)

# Средство автоматизированного проектирования BPwin

## Дополнение созданной модели процессов, диаграммами DFD

Диаграммы потоков данных (Data flow diagramming, DFD) используются для описания документооборота и обработки информации. Подобно IDEF0, DFD представляет модельную систему как сеть связанных между собой работ. Их можно использовать как дополнение к модели IDEF0 для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации. DFD описывает:

* функции обработки информации (работы);
* документы (стрелки, arrow), объекты, сотрудников или отделы, которые участвуют в обработке информации;
* внешние ссылки (external references), которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы;
* таблицы для хранения документов (хранилище данных, data store).

В BPwin для построения диаграмм потоков данных используется нотация Гейна–Сарсона.

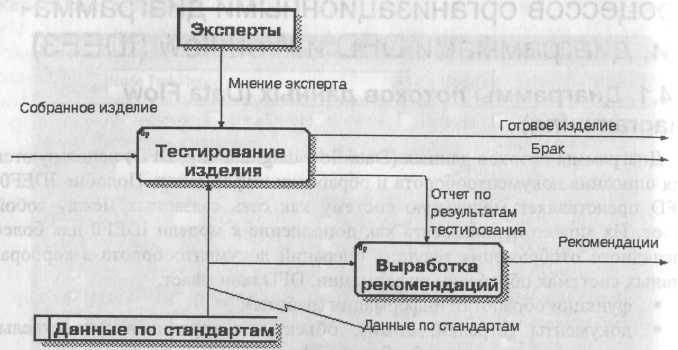
Для того чтобы дополнить модель IDEF0 диаграммой DFD, нужно в процессе декомпозиции в диалоге Activity Box Count "кликнуть" по радиокнопке DFD. В палитре инструментов на новой диаграмме DFD появляются новые кнопки:

– добавить в диаграмму внешнюю ссылку (External Reference). Внешняя ссылка является источником или приемником данных извне модели;



– добавить в диаграмму хранилище данных (Data store). Хранилище данных позволяет описать данные, которые необходимо сохранить в памяти прежде, чем использовать в работах.

В отличие от стрелок IDEF0, которые представляют собой жесткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) двигаются от одной работы к другой. Это представление потоков совместно с хранилищами данных и внешними сущностями делает модели DFD более похожими на физические характеристики системы - движение объектов (data flow), хранение объектов (data stores), поставка и распространение объектов (external entities) (рис. 1.).



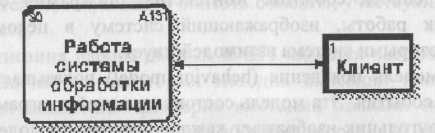
*Рис. 1. Пример диаграммы DFD*

В отличие от IDEF0, где система рассматривается как взаимосвязанные работы, DFD рассматривает систему как совокупность предметов. Контекстная диаграмма часто включает работы и внешние ссылки. Работы обычно именуются по названию системы, например ***"Система обработки информации".*** Включение внешних ссылок в контекстную диаграмму не отменяет требования методологии четко определить цель, область и единую точку зрения на моделируемую систему.

**Работы.** В DFD работы представляют собой функции системы, преобразующие входы в выходы. Хотя работы изображаются прямоугольниками **со** скругленными углами, смысл их совпадает со смыслом работ IDEF0 и IDEF3. Так же как работы IDEF3, они имеют входы и выходы, но не поддерживают управления и механизмы, как IDEF0.

**Внешние сущности**. Внешние сущности изображают входы в систему и/или выходы из системы. Внешние сущности изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах. Обычно такой прием используют, чтобы не рисовать слишком длинных и запутанных стрелок.

**Стрелки (Потоки данных).** Стрелки описывают движение объектов из одной части системы в другую. Поскольку в DFD каждая сторона работы имеет четкого назначения, как в IDEF0, стрелки могут подходить выходить из любой грани прямоугольника работы. В DFD также применяются двунаправленные стрелки для описания диалогов типа "команда-ответ" между работами, между работой и внешней сущностью и между внешними сущностями (рис. 2).



*Рис. 2. Внешняя сущность*

**Хранилище данных.** В отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое (рис. 3).



*Рис. 3. Хранилище данных*

В материальных системах хранилища данных изображаются там, где объекты ожидают обработки, например в очереди. В системах обработки информации хранилища данных являются механизмом, который позволяет сохранить данные для последующих процессов.

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

**Построение диаграмм DFD.** Диаграммы DFD могут быть построены с использованием традиционного структурного анализа, подобно тому как строятся диаграммы IDEF0. Сначала строится физическая модель, отображающая текущее состояние дел. Затем эта модель преобразуется в логическую модель, которая отображает требования к существующей системе. После этого строится модель, отображающая требования к будущей системе. И наконец, строится физическая модель, на основе которой должна быть построена новая система.

Альтернативным подходом является подход, популярный при создании программного обеспечения, называемый событийным разделением (event Partitioning), в котором различные диаграммы DFD выстраивают модель системы. Во-первых, логическая модель строится как совокупность работ и документирования того, что они (эти работы) должны делать.

Затем модель окружения (environment model) описывает систему как объект, взаимодействующий с событиями из внешних сущностей. Модель окружения обычно содержит описание цели системы, одну контекстную диаграмму и список событий. Контекстная диаграмма содержит один прямоугольник работы, изображающий систему в целом, и внешние сущности, с которыми система взаимодействует.

Наконец, модель поведения (behavior model) показывает, как система обрабатывает события. Эта модель состоит из одной диаграммы, в которой каждый прямоугольник изображает каждое событие из модели окружения. Хранилища могут быть добавлены для моделирования данных, которые необходимо запоминать между событиями. Потоки добавляются для связи с другими элементами, и диаграмма проверяется с точки зрения соответствия модели окружения.

Полученные диаграммы могут быть преобразованы с целью более наглядного представления системы, в частности работы на диаграммах могут быть декомпозированы.

**Нумерация объектов.** В DFD номер каждой работы может включать префикс, номер родительской работы (А) и номер объекта. Номер объекта -это уникальный номер работы на диаграмме. Например, работа может иметь номер А. 12.4. Уникальный номер имеют хранилища данных и внешние сущности независимо от их расположения на диаграмме. Каждое хранилище данных имеет префикс D и уникальный номер, например D5. Каждая внешняя сущность имеет префикс Е и уникальный номер, например Е5.

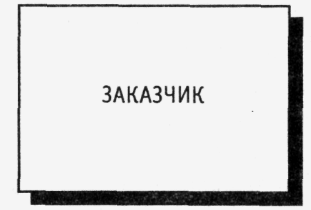
# Моделирование потоков данных (процессов)

В основе данной методологии (методологии Gane/Sarson) лежит построение модели анализируемой ИС – проектируемой или реально существующей. В соответствии с методологией модель системы определяется как иерархия диаграмм потоков данных (DFD), описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы ИС с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. Такая декомпозиция продолжается, создавая многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процессы становятся элементарными и детализировать их далее невозможно.

Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те, в свою очередь, преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных, или внешним сущностям – потребителям информации. Таким образом, основными компонентами диаграмм потоков данных являются: внешние сущности, системы (подсистемы), процессы, накопители данных, потоки данных.

**Внешние сущности.** Внешняя сущность – материальный предмет или физическое лицо, представляющее собой источник или приемник информации, например, заказчики, персонал, поставщики, клиенты, склад. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой ИС. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь диаграммы анализируемой ИС, если это необходимо, или, наоборот, часть процессов ИС может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

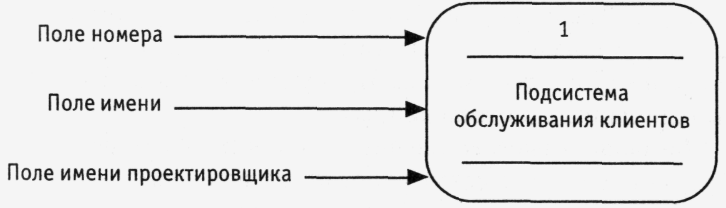
Внешняя сущность обозначается квадратом (рис. 4), расположенным как бы над диаграммой и бросающим на нее тень, для того чтобы можно было выделить этот символ среди других обозначений.



*Рис. 4. Внешняя сущность*

**Системы и подсистемы.** При построении модели сложной ИС она может быть представлена в самом общем виде на так называемой контекстной диаграмме в виде одной системы как единого целого либо может быть декомпозирована на ряд подсистем.

Подсистема (или система) на контекстной диаграмме изображена на рис. 5.

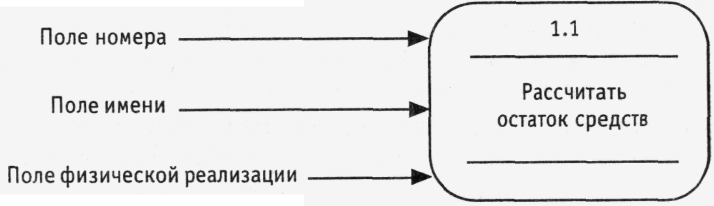


*Рис. 5. Подсистема*

Номер подсистемы служит для ее идентификации. В поле имени вводится наименование подсистемы в виде предложения с подлежащим и соответствующими определениями и дополнениями.

**Процессы.** Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов; программа; аппаратно реализованное логическое устройство, и т.д.

Процесс на диаграмме потоков данных изображен на рис. 6.

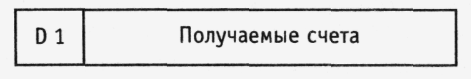


*Рис. 6. Процесс на диаграмме потоков*

Номер процесса служит для его идентификации. В поле имени вводится наименование процесса в виде предложения с активным недвусмысленным глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже. Использование таких глаголов, как обработать, модернизировать или отредактировать, означает, как правило, недостаточно глубокое понимание данного процесса и требует дальнейшего анализа. Информация в поле физической реализации показывает, какое подразделение организации, программа или аппаратное устройство выполняют данный процесс.

**Накопители данных.** Накопитель данных представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

Накопитель данных может быть реализован физически в виде ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носителе и т.д. Накопитель данных на диаграмме потоков данных изображен на рис. 7.



*Рис. 7. Накопитель данных*

Он идентифицируется буквой «D» и произвольным числом. Имя накопителя выбирается из соображений наибольшей информативности для проектировщика.

Накопитель данных в общем случае является прообразом будущей базы данных, и описание хранящихся в нем данных должно быть увязано с информационной моделью.

**Потоки данных.** Поток данных определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой, и т.д.

Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление потока (рис. 8). Каждый поток данных имеет имя, отражающее его содержание.



*Рис. 8. Поток данных*

**Иерархия диаграмм потоков данных (ДПД).** Первым шагом при построении иерархии ДПД является построение контекстных диаграмм. Обычно при проектировании относительно простых ИС строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится так называемый главный процесс, соединенный с приемниками и источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы.

Если же для сложной системы ограничиться единственной контекстной диаграммой, то она будет содержать слишком большое количество источников и приемников информации, которые трудно расположить на листе бумаги нормального формата, и, кроме того, единственный главный процесс не раскрывает структуры распределенной системы. Признаками сложности (в смысле контекста) могут быть:

* наличие большого количества внешних сущностей (десять и более);
* распределенная природа системы;
* многофункциональность системы с уже сложившейся или выявленной группировкой функций в отдельные подсистемы.

Для сложных ИС строится иерархия контекстных диаграмм. При этом контекстная диаграмма верхнего уровня содержит не единственный главный процесс, а набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем.

Иерархия контекстных диаграмм определяет взаимодействие основных функциональных подсистем проектируемой ИС как между собой, так и с внешними входными и выходными потоками данных и внешними объектами (источниками и приемниками информации), с которыми взаимодействует ИС.

Разработка контекстных диаграмм решает проблему строгого определения функциональной структуры ИС на самой ранней стадии ее проектирования, что особенно важно для сложных многофункциональных систем, в разработке которых участвуют разные организации и коллективы разработчиков.

После построения контекстных диаграмм полученную модель следует проверить на полноту исходных данных об объектах системы и на изолированность объектов (отсутствие информационных связей с другими объектами).

Для каждой подсистемы, присутствующей на контекстных диаграммах, выполняется ее детализация при помощи ДПД. Каждый процесс на ДПД, в свою очередь, может быть детализирован при помощи ДПД или миниспецификации. При детализации должны выполняться следующие правила:

* правило балансировки – означает, что при детализации подсистемы или процесса детализирующая диаграмма в качестве внешних источников (приемников) данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных), с которыми имеет информационную связь детализируемая подсистема или процесс на родительской диаграмме;
* правило нумерации – означает, что при детализации процессов должна поддерживаться их иерархическая нумерация. Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2, 12.3 и т.д.

Миниспецификация (описание логики процесса) должна

формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог их выполнить или разработать соответствующую программу.

Миниспецификация является конечной вершиной иерархии ДПД. Решение о завершении детализации процесса и использовании миниспецификации принимается аналитиком исходя из следующих критериев:

* наличие у процесса относительно небольшого количества входных и выходных потоков данных (2-3 потока);
* возможности описания преобразования данных процессом в виде последовательного алгоритма;
* выполнение процессом единственной логической функции преобразования входной информации в выходную; возможность описания логики процесса при помощи миниспецификации небольшого объема (не более 20-30 строк).

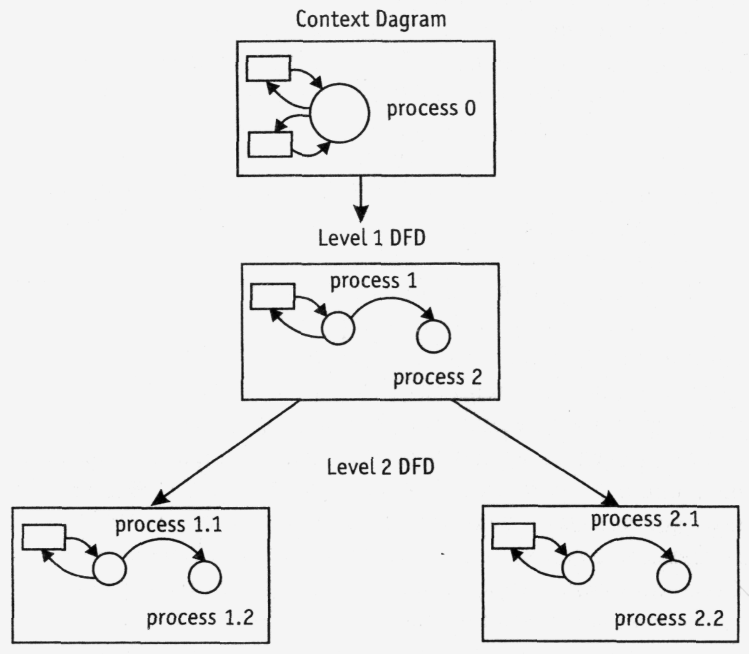
При построении иерархии ДПД переходить к детализации процессов следует только после определения содержания всех потоков и накопителей данных, которое описывается при помощи структур данных. Структуры данных конструируются из элементов данных и могут содержать альтернативы, условные вхождения и итерации. Условное вхождение означает, что данный компонент может отсутствовать в структуре. Альтернатива означает, что в структуру может входить один из перечисленных элементов. Итерация означает вхождение любого числа элементов в указанном диапазоне. Для каждого элемента данных может указываться его тип (непрерывные или дискретные данные); для непрерывных данных – единица измерения (кг, см и т.п.), диапазон значений, точность представления и форма физического кодирования; для дискретных данных – таблица допустимых значений.

После построения законченной модели системы ее необходимо верифицировать (проверить на полноту и согласованность). В полной модели все ее объекты (подсистемы, процессы, потоки данных) должны быть подробно описаны и детализированы. Выявленные недетализированные объекты следует детализировать, вернувшись на предыдущие шаги разработки. В согласованной модели для всех потоков данных и накопителей данных должно выполняться правило сохранения информации: все поступающие куда-либо данные должны быть считаны, а все считываемые данные должны быть записаны.

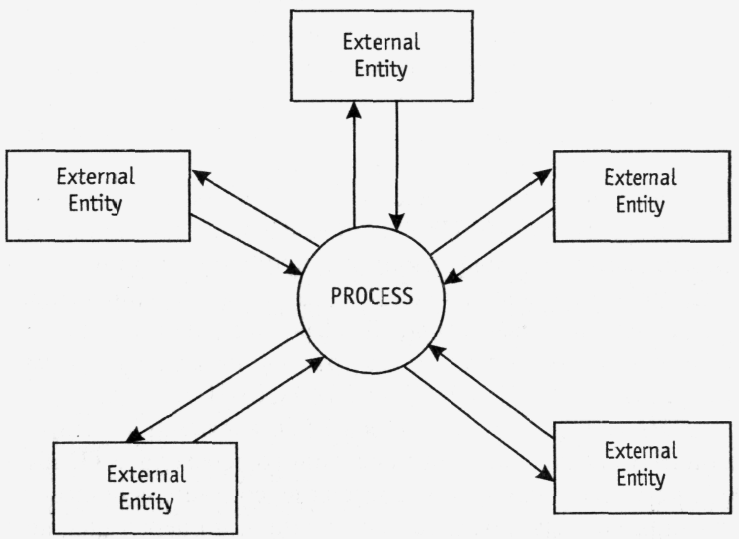
Естественно, это весьма грубое описание предметной области. Уточнение модели производится путем детализации необходимых функций на DFD-диаграмме следующего уровня. Так, мы можем разбить функцию «Определение потребностей и обеспечение материалами» на подфункции «Определение потребностей», «Поиск поставщиков», «Заключение и анализ договоров на поставку», «Контроль платежей», «Контроль поставок», связанные собственными потоками данных, которые будут представлены на отдельной диаграмме. Детализация модели должна производиться до тех пор, пока она не будет содержать всю информацию, необходимую для построения информационной системы.

Как отмечалось ранее, диаграммы потоков данных строятся по иерархическому принципу. Структура иерархии DFD-диаграмм показана на рис. 9.

Контекстная диаграмма верхнего уровня определяет границы модели. Как правило, она имеет звездообразную топологию, в центре которой находится главный процесс, соединенный с приемниками и источниками информации, являющимися внешним окружением моделируемой информационной системы (рис. 10).



*Рис. 9. Структура иерархии DFD-диаграмм*



*Рис. 10. Контекстная диаграмма потоков данных*

Для каждого процесса диаграммы первого уровня может быть произведена декомпозиция, которая, в свою очередь, также может быть раскрыта более подробно. Декомпозиция процессов заканчивается, когда достигнута требуемая степень детализации или отображаемые на очередном уровне диаграмм процессы являются элементарными и не могут быть разбиты на более мелкие.

**Лекция №20**

**Содержание лекции**

[Методология описания и моделирования процессов 2](#_Toc253841536)

[Метод описания процессов IDEF3 2](#_Toc253841537)

[Описание IDEF3 2](#_Toc253841538)

[Основные элементы диаграмм описания последовательности процессов 3](#_Toc253841539)

[Функциональный элемент (UOB) 3](#_Toc253841540)

[Элемент связи 4](#_Toc253841541)

[Перекресток 6](#_Toc253841542)

[Элемент «референт» 13](#_Toc253841543)

[Декомпозиция процесса 18](#_Toc253841544)

# Методология описания и моделирования процессов

## Метод описания процессов IDEF3

Стандарт IDEF3 – это методология описания процессов, рассматривающая последовательность их выполнения и причинно-следственные связи между ситуациями и событиями для структурного представления знаний о системе, а также описания изменения состояний объектов, являющихся составной частью описываемых процессов. Моделирование в стандарте IDEF3 производится с использованием графического представления процесса, материальных и информационных потоков в этом процессе, взаимоотношений между операциями и объектами в процессе. При помощи IDEF3 описывают логику выполнения работ, очередность их запуска и завершения, т.е. IDEF3 предоставляет инструмент моделирования сценариев действий сотрудников организации, отделов, цехов и т.п., например порядок обработки заказа или события, на которые необходимо реагировать за конечное время, выполнение действий по производству товара и т.д. Метод IDEF3 использует категорию сценариев для упрощения структуры описаний сложного многоэтапного процесса. Сценарии определяют граничные условия описания. Здесь под сценарием (Scenario) понимают повторяющуюся последовательность ситуаций или действий, которые описывают типичный класс проблем, присутствующих в организации или системе, описание последовательности изменений свойств объекта в рамках рассматриваемого процесса (например, описание последовательности обработки менеджером заявки). IDEF3 предоставляет инструментарий для наглядного исследования и моделирования сценариев выполнения процессов. Метод позволяет проводить описание с необходимой степенью подробности посредством использования декомпозиции. IDEF3 как инструмент моделирования фиксирует следующую информацию о процессе:

* объекты, которые участвуют при выполнении сценария;
* роли, которые выполняют эти объекты (например, агент, транспорт и т.д.);
* отношения между работами в ходе выполнения сценария процесса;
* состояния и изменения, которым подвергаются объекты;
* время выполнения и контрольные точки синхронизации работ;
* ресурсы, которые необходимы для выполнения работ.

## Описание IDEF3

IDEF3 – уникальная методология, способная фиксировать и структурировать описание работы системы. При этом сбор сведений может производиться из многих источников, что позволяет зафиксировать информацию от экспертов о поведении системы, а не наоборот – строить модель, чтобы приблизить поведение системы. Эта особенность IDEF3 как инструмента моделирования выделяется среди основных характеристик, отличающих IDEF3 от альтернативных предложений. В стандарте IDEF3 существуют два типа диаграмм, представляющих описание одного и того же сценария технологического процесса в разных ракурсах. Диаграммы, относящиеся к первому типу, называются диаграммами описания последовательности выполнения процесса (Process Flow Description Diagrams, PFDD). Второй тип диаграмм описывает состояния объекта и трансформаций в процессе и называется сетью изменений объектных состояний (Object State Transition Network, OSTN). Если диаграммы PFDD описывают процесс, то другой класс диаграмм IDEF3 OSTN используется для иллюстрации трансформаций объекта, которые происходят на каждой стадии выполнения соответствующих работ («с точки зрения объекта»).

Объектные сети изменений состояния (OSTN) фиксируют центрированные объектом представления процессов, суммируя допустимые изменения состояния, которым объекты могут подвергнуться повсюду во время специфического технологического маршрута. Ключевыми понятиями диаграммы описания последовательности действий являются понятия, процесс и логика процесса. Состояния объекта и изменение состояния являются ключевыми понятиями OSTN-диаграммы.

## Основные элементы диаграмм описания последовательности процессов

Диаграммы процесса (наиболее распространенные) обеспечивают механизм визуализации для центрированных процессов описаний сценария. Графические элементы, которые включают диаграммы процесса, включают модуль полей (UOB) поведения, связей старшинства, перекрестков, объектов ссылки и примечаний. Объекты ссылки и примечания – конструкции, которые являются общими для диаграмм процесса и объектов. Каждый из графических элементов, используемых для разработки диаграмм процесса, представлен ниже. Первым рассматривается наиболее фундаментальный из этих стандартных блоков – UOB-функциональный элемент.

### Функциональный элемент (UOB)

Описание процесса представляет всевозможные ситуации (процессы, функции, действия, акты, события, сценарии, процедуры, операции или решения), которые могут происходить в моделируемой системе в логических и временных отношениях. Каждый процесс представлен полем, отображающим название процесса (рис. 1). Номер идентификатора процесса назначается последовательно. В правом нижнем углу UOB-элемента располагается ссылка (IDEF0/USER или другие), которая используется для указания ссылок либо на элементы из функциональной модели IDEF0, либо на отделы или конкретных исполнителей, которые будут выполнять указанную работу.

Основная цель этапа: преобразование требований в детальные спецификации ИС.

Имя

Номер

Ссылка

Функция

Процесс

Действие

Акт

Событие

Сценарий

Процедура

Операция

Решение

Рис. 1. Синтаксис UOB-элемента

### Элемент связи

IDEF3-элемент диаграммы описания процесса связи необходим для связи элементов диаграммы и описания динамики происходящих процессов. Связи используются, прежде всего, для обозначения отношений между функциональными элементами UOB. Для отображения временной последовательности выполнения сценариев в диаграммах описания процесса используются два основных типа связей: связи старшинства и относительные связи. Для описания специфических отношений между элементами предназначены четыре дополнительных типа связей – сдерживаемых связей старшинства. Использование в IDEF3-диаграммах описания процесса различных типов связей дает возможность пользователям метода фиксировать дополнительную информацию о специфике отношений между элементами диаграммы. Символы, которые представляют каждый тип связей, изображены на рис. 2.

Простая связь старшинства

Сдерживаемые связи старшинства

Относительная связь

Рис. 2. Типы связей в диаграммах описания процесса

Подобно процессам, нумерация стрелок производится последовательно, согласно порядку, в котором они добавлены. Номера стрелок содержат префикс «L»   
и назначенного, последовательного номера.

#### *Связи старшинства*

Связи старшинства выражают временные отношения старшинства между элементами диаграммы. При этом первый элемент должен завершиться прежде, чем начнет выполняться следующий. Графически связь предшествования (старшинства) отображается сплошной линией с одиночной стрелкой (рис. 3).

А

1

B

2

Рис. 3. Семантика использования связи старшинства

#### *Сдерживаемые связи старшинства*

Данные виды связей не представлены ни в одном из CASE-продуктов, поддерживающих методологию IDEF3. Сдерживаемые связи старшинства указывают (в дополнение к семантике запуска связей простого старшинства):

* элементу А должен предшествовать элемент В;
* элементу В должен предшествовать элемент А;
* нижняя диаграмма указывает, что любой элемент должен сопровождаться элементом В и что элементу В должен предшествовать элемент А. Эти связи добавляют дополнительные условия к системе (рис. 4). Эти дополнительные условия не только выражают то, как система работает, но и устанавливают требования к тому, как система должна себя вести.

Существует также обобщенное представление сдерживаемых связей предшествования, когда в процессе разработки модели неясно, какая именно связь предшествования должна использоваться, но понятно, что должна использоваться именно сдерживаемая связь предшествования.

А

1

B

2

Рис. 4. Обобщенное представление сдерживаемых связей предшествования

#### *Относительные связи*

Использование относительной связи указывает на тот факт, что между взаимодействующими элементами диаграммы описания процесса существуют отношения неопределенного типа. Относительные связи графически показываются как пунктирные линии.

#### *Связь «поток объектов»*

Тип связи «поток объектов» предложен разработчиками CASE-средств, поддерживающих моделирование в стандарте IDEF3. Графически эта связь показывается как сплошная линия с двойной стрелкой (рис. 5). Этот тип связи выражает перенос одного или нескольких объектов от одного функционального элемента к другому, а также наследует все свойства простой связи старшинства. Таким образом, значение связи «поток объектов» таково: между UOB-элементами происходит передача объекта(ов), причем первый элемент UOB должен завершиться прежде, чем начнет выполняться следующий.

А

1

B

2

Рис. 5. Представление связи «поток объектов»

### Перекресток

Перекрестки используются для отображения логики отношений между множеством событий и временной синхронизации активизации элементов диаграмм IDEF3. Различают перекрестки для слияния (Fan-in Junction) и разветвления (Fan-out Junction) стрелок (рис. 6).

Перекресток не может использоваться одновременно для слияния и для разветвления. При внесении перекрестка в диаграмму необходимо указать тип перекрестка. Тип перекрестка определяет логику и временные параметры отношений между элементами диаграммы. Все перекрестки в PFDD-диаграмме нумеруются, каждый номер имеет префикс «J».

A

B

C

A

B

C

A

B

C

A

B

C

Рис. 6. Перекрестки разветвления и слияния

#### *Типы перекрестков*

Тип перекрестка обозначается на элементе следующим образом:

* & – логический И;
* О – логический ИЛИ;
* X – логический перекресток НЕЭКВИВАЛЕНТНОСТИ.

Стандарт IDEF3 предусматривает разделение перекрест ков типа & и О на синхронные и асинхронные (рис. 7). Это разделение позволяет учитывать в диаграммах описания процессов синхронизацию времени активизации.

ТИП

Асинхронный

ТИП

Синхронный

Рис. 7. Пример обозначения синхронности и асинхронности перекрестков

Для последующего изложения материала необходимо ввести понятие график запуска. График запуска – это визуальное отображение временной последовательности выполнения UOB-элементов. Возможный график запуска для ситуации представлен на рис. 8, семантика использования связи старшинства была приведена на рис. 3.

Время

A

B

Рис. 8. Пример графика запуска

Визуальное отображение на графике запуска временной последовательности выполнения UOB-элементов поможет правильно понять, как перекрестки описывают логику отношений между элементами диаграммы описания процессов и каким образом перекрестки позволяют синхронизировать по времени выполнение UOB-элементов.

Методология IDEF3 использует пять логических типов для моделирования возможных последовательностей действий процесса в сценарии (табл. 1).

Использование комбинаций перекрестков (рис. 9, 11, 13, 14, 16, 18)   
и соответственно графиков запуска (рис. 10, 12, 15, 17) представлено на схемах ниже.

A

1

&

B

2

C

4

E

3

F

6

D

5

&

J1

J2

Рис. 9. Использование перекрестка «асинхронный AND»

Время

A

B

C

D

E

F

Рис. 10. Возможный график запуска для рис. 9

Таблица 1

**Логические типы**

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Смысл в случае слияния стрелок**  **(Fan-in Junction)** | **Смысл в случае разветвления**  **стрелок (Fan-out Junction)** |
| Asynchronous AND | Все предшествующие процессы должны быть завершены | Все следующие процессы должны быть запущены |
| Synchronous AND | Все предшествующие процессы завершены одновременно | Все следующие процессы запускаются одновременно |
| Asynchronous OR | Один или несколько предшествующих процессов должны быть завершены | Один или несколько следующих процессов должны быть запущены |
| Synchronous OR | Один или несколько предшествующих процессов завершаются одновременно | Один или несколько следующих процессов запускаются одновременно |
| XOR (Exclusive OR) | Только один предшествующий процесс завершен | Только один следующий процесс запускается |

A

1

&

B

2

C

4

E

3

F

6

&

D

5

J1

J2

Рис. 11. Использование перекрестка «синхронный AND»

Время

A

B

C

D

E

F

Рис. 12. Возможный график запуска для рис. 11

A

1

O

B

2

C

4

E

3

F

6

D

5

O

J1

J2

Рис. 13. Использование перекрестка «асинхронный OR»

A

1

O

B

2

C

4

E

3

F

6

O

D

5

J1

J2

Рис. 14. Использование перекрестка «синхронный OR»

A

B

C

D

E

F

A

B

C

D

E

F

Рис. 15. Возможный график запуска для рис. 13 и рис. 14

A

1

&

D

4

E

5

B

2

C

3

O

J1

J2

Рис. 16. Использование «асинхронного AND» перекрестка разветвления   
и «асинхронного OR» перекрестка слияния

A

B

C

D

E

A

B

C

D

E

A

B

C

D

E

Рис. 17. Возможные графики запуска для рис. 16

A

1

O

D

4

E

5

B

2

C

3

&

J1

J2

Рис. 18. Невозможное совместное использование перекрестков

### Элемент «референт»

Элемент «референт» — это элемент ссылки. Референты расширяют границы понимания диаграммы и упрощают конструкцию описания (тем самым исключают неоднозначность). Референты используются как в IDEF3-диаграммах описания процесса, так и в объектных диаграммах OSTN (табл. 2). Референты предназначены для:

* обращения к предварительно определенному функциональному элементу UOB без дублирования его определения;
* передачи управления или организации возвратных циклов;
* организации связи между IDEF3-диаграммами описания процесса и OSTN-объектными диаграммами. Каждый тип референта может использоваться как в IDEF3-диаграмме описания процесса, так и в объектной диаграмме OSTN. Однако наиболее продуктивно референты используются в IDEF3-диаграммах описания процесса.

#### *Виды референтов*

Помимо деления на виды, методология IDEF3 определяет два вида референтов по способу запуска (рис. 19).

Тип референта / Имя референта

Locator

Тип референта / Имя референта

Locator

РЕФЕРЕНТ

«Запустить и Продолжить»

РЕФЕРЕНТ

«Запустить и Ждать»

Рис. 19. Синтаксис референта

Таблица 2

**Использование референтов в диаграмме**

|  |  |  |
| --- | --- | --- |
| **Тип референта** | **Обозначение референта** | **Locator** |
| UOB | Имя функционального элемента UOB | Номер UOB |
| SCENARIO | Название сценария | Номер Scenario |
| TS (Transition Schematic) | Название диаграммы перехода состояний | Номер диаграммы перехода |
| GO-TO используется только в IDEF3-диаграммах описания процесса | Имя функционального элемента UOB | Номер сценария или декомпозиции, в котором находится номер UOB |

Разделение на референты «запустить и продолжить» и «запустить и ждать» позволяет описать временные границы выполнения референта. Так, использование референта «запустить и продолжить» указывает, что упомянутый элемент «референт» должен лишь инициализироваться (активизироваться) раньше, чем выполнение элемента IDEF3, вызывающего элемент «референт», будет завершено. Для такой ситуации возможное развитие событий представлено на рис. 20.

UOB /  
Выполнение UOB B

15.1.27 / 15.1

A

6.1.12

C

6.1.13

Время

Реф

UOB

A

C

Рис. 20. Использование референта «запустить и продолжить»  
и возможный график запуска

#### *Использование референта «запустить и ждать»*

Использование референта «запустить и ждать» указывает, что упомянутый референт должен активизироваться и завершиться прежде, чем элемент, его вызвавший, завершит свое выполнение, как это показано на рис. 21.

Если тип референта UOB или SCENARIO, то такой элемент может являться источником для связи старшинства. Другой особенностью референта «запустить и ждать» является невозможность использования GO-TO-референта.

UOB /  
Выполнение UOB D

4.1.10 / 4.1

C

6.1.17

Время

Реф

UOB

C

Рис. 21. Использование референта «запустить и ждать» и возможный график запуска

#### *Использование референта «запустить и продолжить»*

Если используется референт «запустить и продолжить», который имеет тип UOB, SCENARIO или GO-TO, то на выходе такого элемента не может использоваться стрелка старшинства. Это утверждение становится очевидным, если посмотреть график запуска на рис. 22.

При использовании после референта «запустить и продолжить» стрелки старшинства получаем неопределенную, непоследовательную и противоречивую ситуацию. Существует противоречие в совместном использовании референта «запустить и продолжить» и стрелки старшинства.

A

6.1.12

Время

Реф

UOB

A

C

UOB /  
Выполнение UOB B

15.1.27 / 15.1

С

6.1.13

**?**

Рис. 22. Невозможное использование референта «запустить и продолжить»

#### *UOB-референт*

Если тип референта UOB, то наименование этого референта должно быть идентично наименованию элемента UOB, который предварительно определен. Если референт UOB используется в диаграмме описания процесса и приложен к элементу диаграммы, то во время выполнения вызывающего UOB-элемента осуществляется активизация соответствующего UOB-элемента (причем временных ограничений по завершению вызываемого UOB-элемента нет). Если UOB-референт приложен к стрелке, которая связывает элементы состояния объекта в диаграмме объекта, то выполнение упомянутого в референте UOB должно начаться прежде, чем начнет изменяться состояние объекта. Если UOB-референт приложен к элементу состояния объекта в объектной диаграмме OSTN, то это указывает, что упомянутый UOB содержит объект в соответствующем состоянии.

#### *SCENARIO-референт*

Если используется референт типа Scenario, то его название соответственно должно совпадать с названием сценария, на который ссылается вышеуказанный референт. При использовании Scenario-референта в диаграмме описания процесса во время выполнения вызывающего UOB-элемента осуществляется активизация вызванного сценария (причем временных ограничений по завершению вызываемого сценария нет). Вызываемый сценарий выполняется на всю «глубину» декомпозиции. Если Scenario-референт приложен к стрелке, которая связывает элементы состояния объекта в диаграмме объекта, то выполнение упомянутого в референте Scenario должно начаться прежде, чем начнет изменяться состояние объекта.

#### *Элемент «примечание»*

Элемент «примечание» может использоваться как в диаграммах описания процесса, так и в объектных диаграммах OSTN. Этот элемент может быть приложен к функциональному элементу UOB, перекрестку, связи, объекту или референту. Элемент «примечание» предназначен для:

* идентификации и подчеркивания участия специфических объектов или отношений, связанных с функциональным элементом UOB, связью или переходом;
* присоединения примеров, объектов и т.п. (например, экранных форм);
* отображения специальных условий, уточнений соединения или ограничений, связанных с элементами диаграмм.

Примечания могут использоваться для обеспечения дополнительной информации   
в процессе моделирования, для присоединения к диаграммам иллюстраций, текста, экранных форм, комментариев и т.д. Примечания предоставляют возможность выразить идеи или концепции вместо использования относительных связей.

На рис. 23 показана семантика использования элемента «примечание».

Описание предмета примечания

№ Вызывающего Элемента / № Примечания

Рис. 23. Семантика элемента «примечание»

Поле примечания разделено на две части. Верхняя часть элемента используется для идентификации примечания и содержит идентификатор примечания, составленный из номера элемента, для которого делается примечание, и номера примечания с префиксом N (например, J1/N1). Нижняя часть примечания называется полем примечания и предназначена непосредственно для текста, рисунка и т.п. Стандартом IDEF3 не определены какие-либо ограничения на форму и состав содержания поля примечания, хотя группой разработчиков модели могут быть оговорены некоторые соглашения для решения каких-либо целей.

## Декомпозиция процесса

Методология IDEF3 дает возможность представлять процесс в виде иерархически организованной совокупности диаграмм. Диаграммы состоят из нескольких элементов описания процесса IDEF3, причем каждый функциональный элемент UOB потенциально может быть детализирован на другой диаграмме. Такое разделение сложных комплексных процессов на их структурные части называется декомпозицией. Декомпозиция формирует границы для описания процесса, и каждый UOB-элемент рассматривается как формальная граница некоторой части целой системы, которая описывает весь процесс. Декомпозированная диаграмма, называемая диаграммой-потомком, более детально описывает процесс. Декомпозируемый UOB-элемент называется родительским, а содержащая его диаграмма, соответственно, родительской диаграммой. Итак, декомпозиция – это процесс создания диаграммы, детализирующей определенный UOB-элемент. Результатом ее является описание, которое представляет собой дробление родительского UOB-элемента на меньшие и более частные операции или функции. Кроме того, само слово «анализ» означает разложение на составляющие. Но декомпозиция – это больше, чем разложение на части. Она включает также синтез. Подлинная декомпозиция заключается в начальном разделении объекта на более мелкие части и последующем соединении их в более детальное описание (рис. 24).

Применяя принцип декомпозиции неоднократно, возможно структурировать описание процесса до любого уровня подробности. Декомпозиция обеспечивает средства организации более детального описания UOB-элементов. Каждый UOB-элемент может иметь любое число различных декомпозиций на том же самом уровне детализации в целях представления различных точек зрения или обеспечения большей подробности при описании исходного процесса.

1

2

3

4

3.1.5

3.1.6

3.1.6

3.2.8

3.2.9

3.2.10

**×**

6.1.11

6.2.12

6.2.13

Рис. 24. Пример нумерации UOB-элементов

Нумерация элементов диаграммы описания процесса приведена на рис. 24.