ЛЕКЦИЯ № 4.1. МЕТОДИКИ ПРОЕКТИРОВАНИЯ АВТОМАТИЗИРОВАННЫХ СИСТЕМ.

1. CASE-системы.

2. Спецификации проектов программных систем.

3. Технологии реинжиниринга и параллельного проектирования.

4. Методики IDEF.

5. Унифицированный язык моделирования UML.

6. Программное обеспечение CASE-систем для концептуального проектирования.

7. Метамоделиистандарты CDIF (CASE Data Interchange Format).

1. CASE-системы.

В современных информационных технологиях важное место отводится инструментальным средствам и средам разработки АС, в частности, системам разработки и сопровождения их ПО. Эти технологии и среды образуют системы, называемые *CASE-системами.*

Используется двоякое толкование аббревиатуры CASE, соответствующее двум направлениям использования CASE-систем. Первое из них — ComputerAidedSystemEngineering — подчеркивает направленность на поддержку концептуального проектирования сложных систем, преимущественно слабоструктурированных. Далее CASE-системы этого направления будем называть *системами CASE для концептуального проектирования.* Второе направление было рассмотрено выше, его название

ComputerAidedSoftwareEngineering переводится, как автоматизированное проектирование программного обеспечения, соответствующие CASE-системы называют *инструментальными CASE* или *инструментальными средами* разработки ПО (одно из близких к этому названий — RAD — RapidApplicationDevelopment).

Среди систем CASE для концептуального проектирования различают системы функционального, информационного или поведенческого проектирования. Наиболее известной методикой *функционального проектирования* сложных систем является методика SADT (StructuredAnalysisandDesignTechnique), предложенная в 1973 г. Р.Россом и впоследствии ставшая основой международного стандарта IDEF0 (IntegratedDEFinition 0).

Системы ***информационного проектирования*** реализуют методики инфологического проектирования БД. Широко используются язык и методика создания информационных моделей приложений, закрепленные в международном стандарте IDEF1X. Кроме того, развитые коммерческие СУБД, как правило, имеют в своем составе совокупность CASE-средств проектирования приложений.

Основные положения стандартов IDEF0 и IDEF1X использованы также при создании комплекса стандартов ISO 10303, лежащих в основе технологии STEP для представления в компьютерных средах информации, относящейся к проектированию и производству в промышленности.

Поведенческое моделирование сложных систем используют для определения динамики функционирования сложных систем. В его основе лежат модели и методы имитационного моделирования систем массового обслуживания, сети Петри, возможно применение конечно-автоматных моделей, описывающих поведение системы, как последовательности смены состояний.

Применение инструментальных CASE-систем ведет к сокращению затрат на разработку ПО за счет уменьшения числа итераций и числа ошибок, к улучшению качества продукта за счет лучшего взаимопонимания разработчика и заказчика, к облегчению сопровождения готового ПО.

Среди инструментальных CASE-систем различают интегрированные комплексы инструментальных средств для автоматизации всех этапов жизненного цикла ПО (такие системы называют Workbench) и специализированные инструментальные средства для выполнения отдельных функций (Tools). Средства CASE по своему функциональному назначению принадлежат к одной из следующих групп: 1) средства программирования; 2) средства управления программным проектом; 3) средства верификации (анализа) программ; 4) средства документирования.

К первой группе относятся компиляторы с алгоритмических языков; построители диаграмм потоков данных; планировщики для построения высокоуровневых спецификаций и планов ПО (возможно на основе баз знаний, реализованных в экспертных системах); интерпретаторы *языков спецификаций* и *языков четвертого поколения*; прототайпер для разработки внешних интерфейсов — экранов, форм выходных документов, сценариев диалога; генераторы программ определенных классов (например, конверторы заданных языков, драйверы устройств программного управления, постпроцессоры); кросс-средства; отладчики программ. При этом под *языками спецификаций* понимают средства укрупненного описания разрабатываемых алгоритмов и программ, к языкам 4GL относят языки для компиляции программ из набора готовых модулей, реализующих типовые функции достаточно общих приложений (чаще всего это функции технико-экономических систем).

Управление программным проектом называют также *управлением конфигурациями* ПО (SCM — softwareconfigurationmanagement). Этому понятию соответствуют корректное внесение изменений а программную систему при ее проектировании и сопровождении, контроль целостности проектных данных, управление версиями проекта, организация параллельной работы членов коллектива разработчиков. Использование средств управления конфигурациями позволяет создавать программные системы из сотен и тысяч модулей, значительно сокращать сроки разработки, успешно модернизировать уже поставленные заказчикам системы.

Основой средств управления программным проектом является репозиторий — БД проекта. Именно в репозитории отражена история развития программного проекта, содержатся все созданные версии (исходный программный код, исполняемые программы, библиотеки, сопроводительная документация и т.п.) с помощью репозитория осуществляется контроль и отслеживание вносимых изменений.

Средства верификации служат для оценки эффективности исполнения разрабатываемых программ и определения наличия в них ошибок и противоречий. Различают статические и динамические анализаторы. В статических анализаторах ПО исследуется на наличие неопределенных данных, бесконечных циклов, недопустимых передач управления и т.п. Динамический анализатор функционирует в процессе исполнения проверяемой программы; при этом исследуются трассы, измеряются частоты обращений к модулям и т. п. Используемый математический аппарат — сети Петри, теория массового обслуживания.

В последнюю из перечисленных групп входят документаторы для оформления программной документации, например, отчетов по данным репозитория; различные редакторы для объединения, разделения, замены, поиска фрагментов программ и других операций редактирования.

Проектирование ПОс помощью CASE-систем включает в себя несколько этапов. Начальный этап

— предварительное изучение проблемы. Результат представляют в виде исходной диаграммы потоков данных и согласуют с заказчиком. На следующем этапе выполняют детализацию ограничений и функций программной системы, и полученную логическую модель вновь согласуют с заказчиком. Далее разрабатывают физическую модель, т. е. определяют модульную структуру программы, выполняют ин-фологическое проектирование БД, детализируют граф-схемы программной системы и ее модулей.

2. Спецификации проектов программных систем.

Важное значение в процессе разработки ПО имеют средства

спецификации проектов ПО. Средства спецификации в значительной мере определяют суть методов CASE.

Способы и средства спецификации классифицируют по базовой методологии, используемой для декомпозиции ПО, как сложной системы, и по аспектам моделирования ПО.

Различают два подхода к декомпозиции ПО Первый способ называют*функциональным* или *структурным.* Он основан на выделении функций и потоков данных. Второй способ - *объектный,* выражает идеи объектно-ориентированного проектирования и программирования. Проектирование ПОиз готовых компонентов, рассмотренное в предыдущей главе, есть выражение объектного подхода.

Аспектами моделирования приложений являются функциональное, поведенческое и информационное описания.

Практически все способы *функциональных* спецификаций имеют следующие общие черты:

— модель имеет иерархическую структуру, представляемую в виде диаграмм нескольких уровней;

— элементарной частью диаграммы каждого уровня является конструкция вход-функция-выход;

— необходимая дополнительная информация содержится в файлах поясняющего текста.

В большинстве случаев функциональные диаграммы являются диаграммами потоков данных (DFD — DataFlowDiagram). В DFD блоки (прямоугольники) соответствуют функциям, дуги — входным и выходным потокам данных. Поясняющий текст представлен в виде “словарей данных”, в которых указаны компонентный состав потоков данных, число повторений циклов и т.п. Для описания структуры информационных потоков можно использовать нотацию Бэкуса-Наура.

Одна из нотаций для DFD предложена Е.Йорданом. В ней описывают процессы (функции), потоки данных, хранилища и внешние сущности, их условные обозначения показаны на рис. 6. 1.

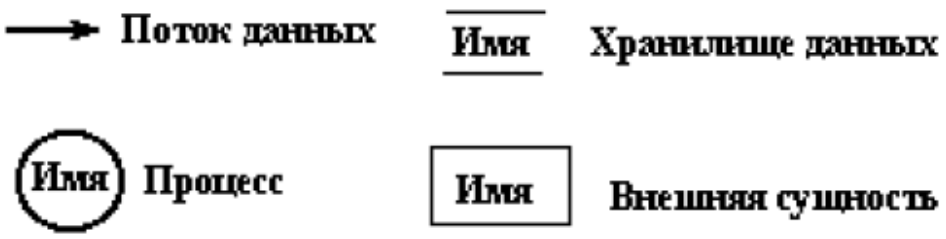


Рис. 6.1. Изображения элементов в нотации Йордана

Разработка DFD начинается спостроения диаграммы верхнего уровня, отражающей связи программной системы, представленной в виде единогопроцесса, с внешней средой. Декомпозиция процессапроводится до уровня, на котором фигурируют элементарные процессы, которые могут

быть представлены одностраничными описаниями алгоритмов (миниспецификациями) на терминальном языке программирования.

Для описания *информационных* моделей наибольшее распространение получили диаграммы сущность-связь (ERD — Entity-RelationDiagrams), в которых предусмотрены средства для описания сущностей, атрибутов и отношений. Спецификации хранилищ данных в CASE, как правило, даются с помощью диаграмм сущность-связь Стандартной методикой построения таких диаграмм является IDEF1X.

Поведенческие модели описывают процессы обработки информации. В инструментальных CASE-системах их представляют в виде граф-схем, диаграмм перехода состояний, таблиц решений, псевдокодов (языков спецификаций), процедурных языков программирования, в том числе языков четвертого поколения.

В граф-схемах блоки, как и в DFD, используют для задания процессов обработки, но дуги имеют иной смысл — они описывают последовательность передач управления (вместе с специальными блоками управления).

В диаграммах перехода состояний узлы соответствуют состояниям моделируемой системы, дуги — переходам изсостояние в состояние, атрибуты дуг — условиям перехода и инициируемым при их выполнении действиям. Очевидно, что как и в других конечно-автоматных моделях, кроме графической формы представления диаграмм перехода состояний, можно использовать также табличные формы. Так, при изоморфном представлении с помощью таблиц перехода состояний каждому переходу соответствует строка таблицы, в которой указываются исходное состояние, условие перехода, инициируемое при этом действие и новое состояние после перехода.

Близкий по своему характеру способ описания процессов основан на таблицах (или деревьях) решений. Каждый столбец таблицы решений соответствует определенному сочетанию условий, при выполнении которых осуществляются действия, указанные в нижерасположенных клетках столбца.

Таблицы решений удобны при описании процессов с многократными ветвлениями. В этих случаях помогают также визуальные языки программирования, в которых для описания процессов используют графические элементы, подобные приведенным на рис. 6.2.

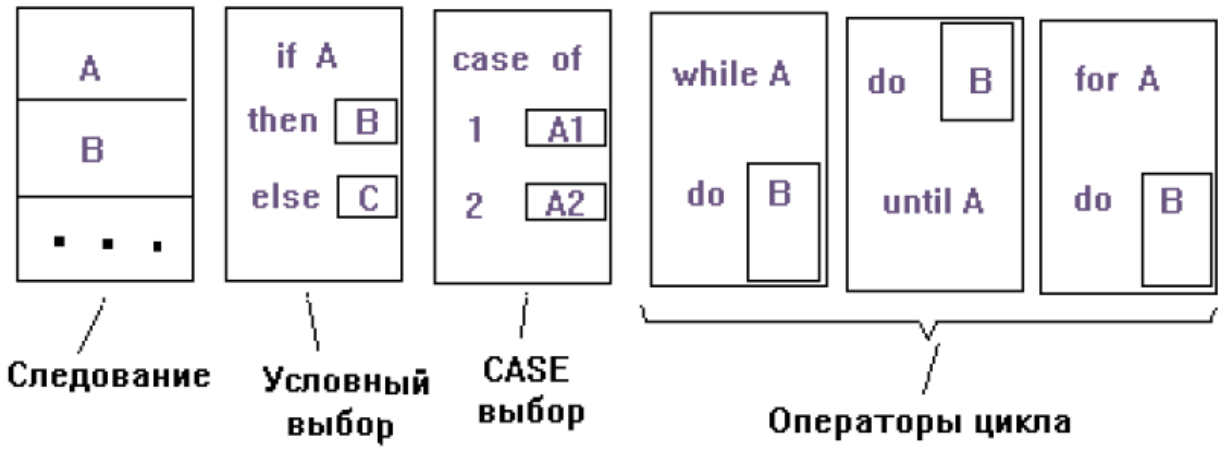


Рис.6.2. Примеры описания операторов в визуальных языках программирования

В псевдокодах алгоритмы записываются с помощью как средств некоторого языка программирования (преимущественно для управляющих операторов), так и естественного языка (для выражения содержания вычислительных блоков). Используются конструкции (операторы) следования, условные, цикла. Служебные слова из базового языка программирования или из DFD записываются заглавными буквами, фразы естественного языка — строчными.

Языки четвертого поколения предназначены для описания программ как совокупностей заранее разработанных программных модулей. Поэтому одна команда языка четвертого поколения может соответствовать значительному фрагменту программы на языке 3GL. Примерами языков 4GL могут служить Informix-4GL, JAM, NewEra, XAL.

Миниспецификации процессов могут быть выражены с помощью псевдокодов (языков спецификаций), визуальных языков проектирования или языков программирования,

Объектный подход представлен компонентно-ориентированнными технологиями разработки ПО. При объектном подходе ПО формируется из компонентов, объединяющих в себе алгоритмы и данные и взаимодействующих путем обмена сообщениями. Для поддержки объектного подхода разработан стандартный язык моделирования приложений UML.

3. Технологии реинжиниринга и параллельного проектирования.

Взаимосвязанная совокупность методик IDEF для концептуального проектирования разработана по программе IntegratedComputerAidedManufacturing в США. В этой совокупности имеются методики функционального, информационного и поведенческого моделирования и проектирования, в ее состав в настоящее время входят IDEF-методики, представленные в приложении (табл. П. 1), часть из которых имеет статус международного стандарта.

Методики IDEF задают единообразный подход к моделированию приложений, но не затрагивают проблем единообразного представления данных в процессах информационного обмена между разными компьютерными системами и приложениями. Необходимость решения этих проблем в интегрированных АС привела к появлению ряда унифицированных форматов представления данных в меж- компьютерных обменах, среди которых наиболее известными являются форматы IGES, DXF (в машиностроительных приложениях), EDIF (в электронике) и некоторые другие. Однако ограниченные возможности этих форматов обусловили продолжение работ в направлении создания более совершенных методик и представляющих их стандартов. На эту роль в настоящее время претендует совокупность стандартов STEP.

4. МетодикиIDEF.

4.1. Методика IDEF0.

Как отмечено выше, наиболее известной методикой *функционального моделирования* сложных систем является методика SADT (StructuredAnalysisandDesignTechnique), положенная в основу стандарта IDEF0.

IDEF0 — это более четко очерченное представление методики SADT.SADT — методика, рекомендуемая для начальных стадий проектирования сложных искусственных систем управления, производства, бизнеса, включающих людей, оборудование, ПО. Начиная с момента создания первой версии, методика успешно применялась для проектирования телефонных сетей, систем управления воздушными перевозками, производственных предприятий и др.

Разработку SADT-модели начинают с формулировки вопросов, на которые модель должна давать ответы, т.е. формулируют цель моделирования. Далее строят иерархическую совокупность диаграмм с лаконичным описанием функций.

Недостатки SADT-моделей — их слабаяформализованность для автоматического выполнения проектных процедур на их основе. Однако наличие графического языка диаграмм, удобного для восприятия человеком, обусловливает полезность и применимость методики SADT.

Описание объектов и процессов в SADT (IDEF0) выполняется в виде совокупности взаимосвязанных блоков (рис. 6.3).

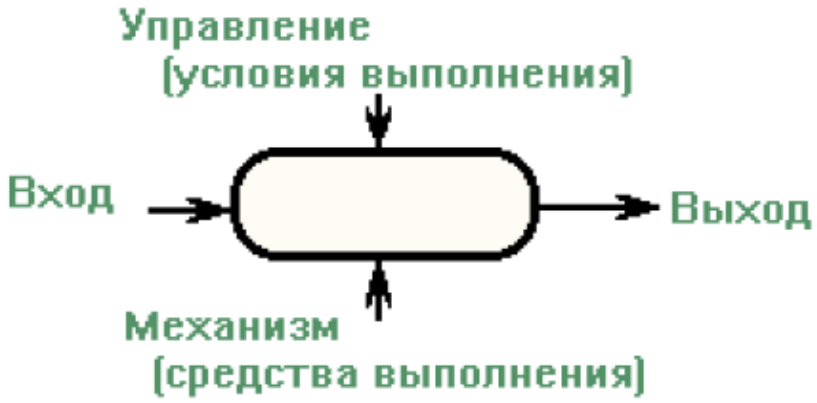


Рис. 6.3. Блок ICOM в IDEF0-диаграммах

Блоки выражают функции (работы), поэтому их названиями обычно являются глаголы или отглагольные существительные. Типичные примеры функций: планировать, разработать, классифицировать, измерить, изготовить, отредактировать, рассчитать, продать (или планирование, разработка, классификация, измерение, изготовление, редактирование, расчет, продажа). Число блоков на одном уровне иерархии — не более 6, иначе восприятие диаграмм будет затруднено.

Число уровней иерархии не ограничено, но обычно их не более 5. Блоки нумеруются (номер записывается в правом нижнем углу). Дуги (стрелки) отображают множества объектов (данных), их имена — существительные. Управление определяет условия выполнения, примеры управления: требования, чертеж, стандарт, указания, план. Механизм выражает используемые средства, например: компьютер, оснастка, заказчик, фирма. Входы и выходы могут быть любыми объектами.

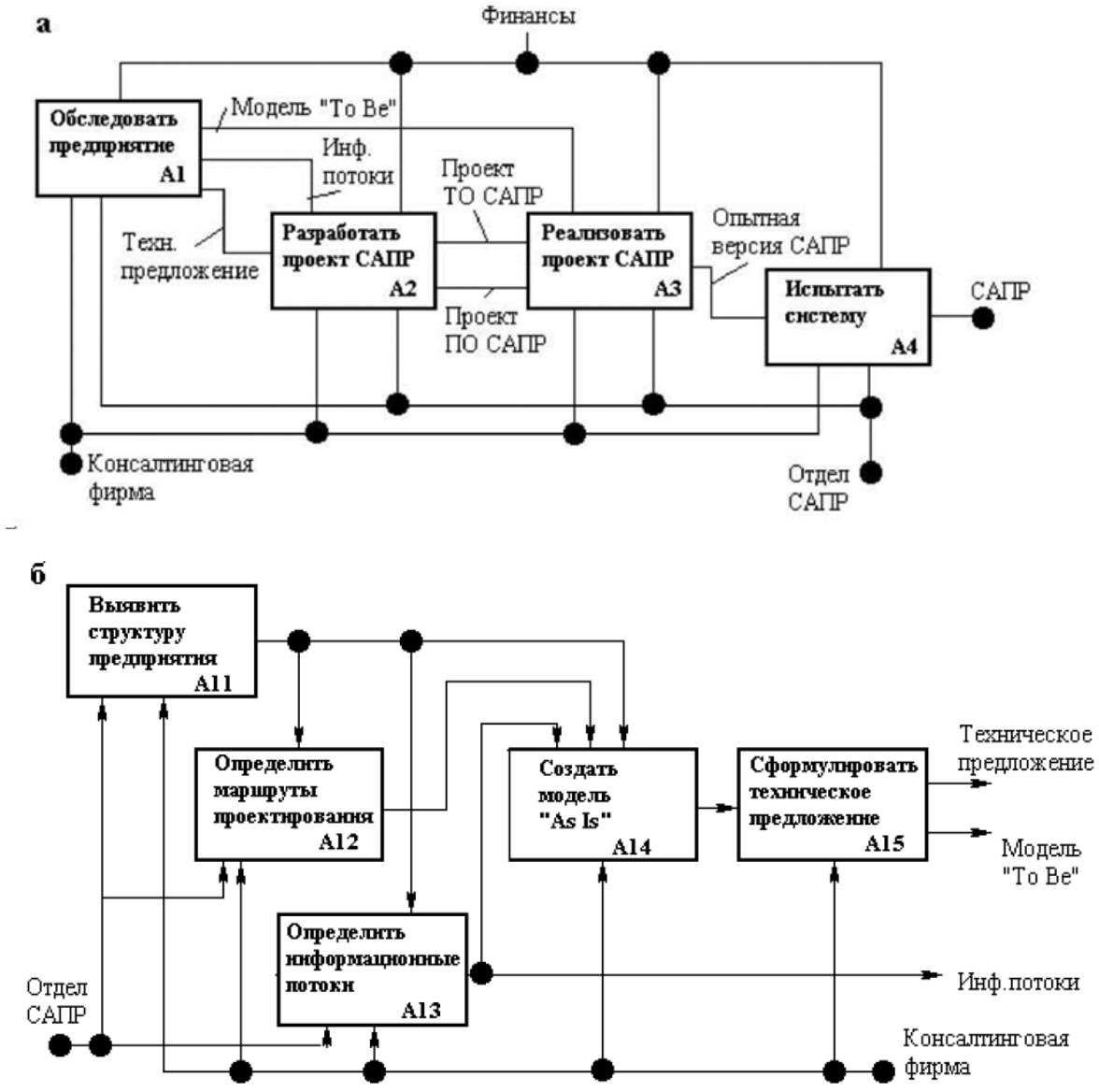
Блоки рис. 6.3 в англоязычной литературе называют блоками ICOM (Input — Control — Output - Mechanism).

Рассмотрим пример функциональной модели для процесса создания САПР на предприятии, на котором ранее автоматизация проектирования была развита слабо.

Диаграмма верхнего (нулевого) уровня А0 включает единственный блок ICOM “Разработать САПР”. В качестве исполнителей фигурируют специализированная организация, занимающаяся проектированием автоматизированных систем и называемая консалтинговой фирмой, а также представители организации-заказчика, объединенные в создаваемый на предприятии отдел САПР.

Диаграмма первого уровня, показанная на рис. 6.4,а, включает блоки А1 — обследования предприятия, А2 — проектирования САПР, А3 — реализации САПР и А4 - испытаний системы. Диаграммы следующего второго уровня, раскрывающие первые блоки А1, А2 и А3, представлены на рис. 6.4,б, в и г соответственно (на этих рисунках не отмечены данные,

соответствующие внутренним стрелкам диаграмм). При обследовании предприятия специалисты консалтинговой фирмы вместе с работниками отдела САПР, изучают структуру предприятия, типичные маршруты проектирования, информационные потоки и на этой базе разрабатывают модель “AsIs”. Далее создается новая модель “ToBe” с учетом не только требований автоматизации проектирования, но и будущих информационных потребностей процессов управления и делопроизводства. Модель “ToBe” составляет основу технического предложения на создание САПР.



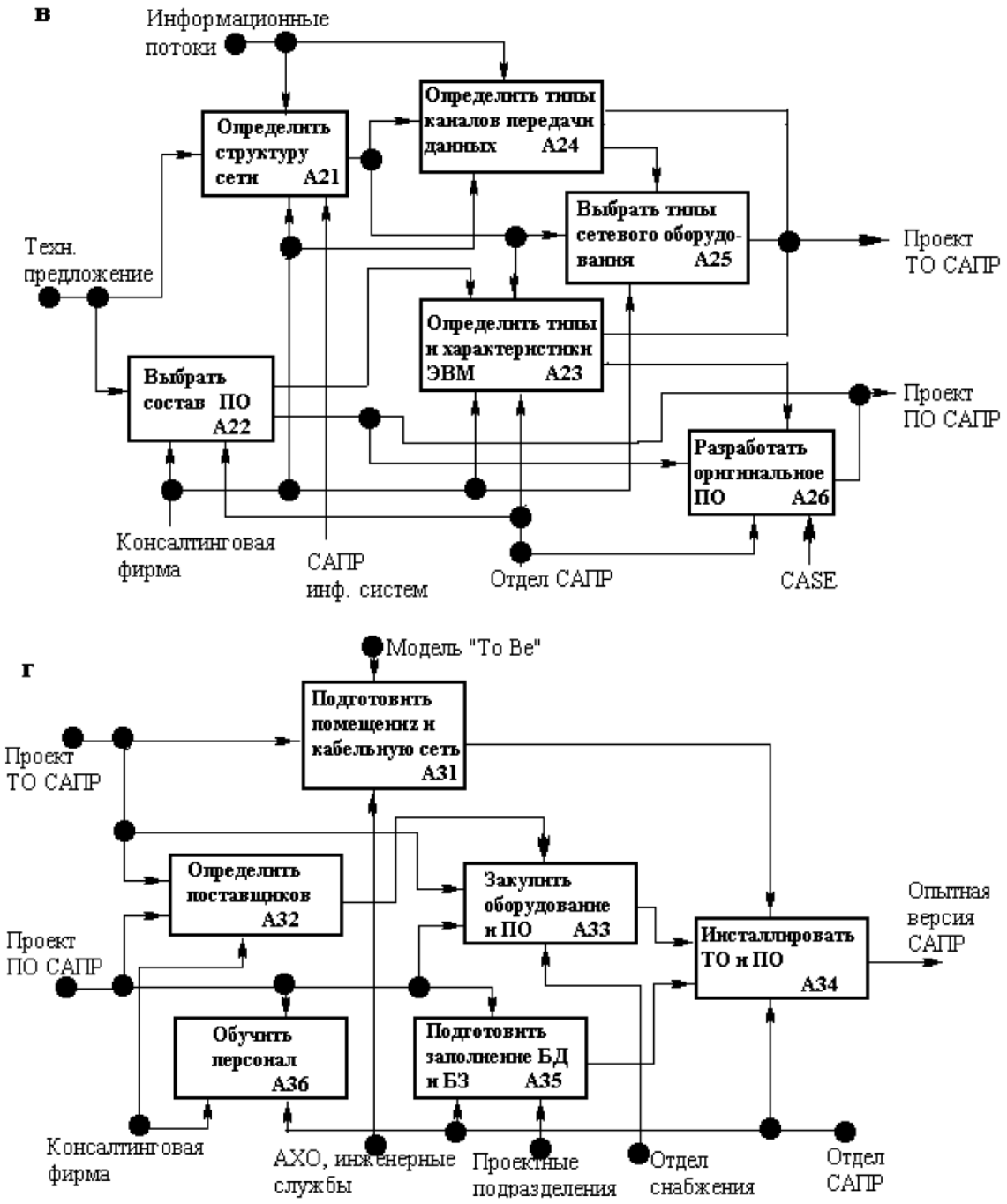


Рис. 6.4. Функциональная модель процесса создания САПР: а) IDEFO-диаграмма первого уровня; б) IDEFO-диаграмма обследования предприятия; в) IDEFO-диаграмма проектирования САПР; г) IDEFO-диаграмма реализации проекта САПР

При проектировании САПР выбирают аппаратно-программную платформу, базовое ПО проектирующих и обслуживающих подсистем, разрабатывают структуру корпоративной сети, определяют типы сетевого оборудования, серверов и рабочих станций, выявляют необходимость разработки оригинальных программных компонентов.

Реализация проекта САПР включает подготовку помещений, монтаж кабельной сети, обучение будущих пользователей САПР, закупку и инсталляцию ТО и ПО.

Разработка SADT-моделей состоит из ряда этапов.

1. Сбор информации. Источниками информации могут быть документы, наблюдение, анкетирование и т.п. Существуют специальные методики выбора экспертов и анкетирования.

2. Создание модели. Используется нисходящий стиль: сначала разрабатываются верхние уровни, затем нижние.

3. Рецензирование модели. Реализуется в итерационной процедуре рассылки модели на отзыв и ее доработки по замечаниям рецензентов, в завершение собирается согласительное совещание.

Связи функциональной модели, отражающей функции, со структурной моделью, отражающей средства выполнения функций, выражаются с помощью специальных словарей, дающих однозначное толкование вводимым именам ресурсов.

Дальнейшее использование IDEFO-модели — конкретизация задач выбора ресурсов, разработка планов реализации, переход к имитационным моделям и т.п.

4.2. Методика IDEF3.

Поведенческое моделирование сложных систем используют для исследования динамики их функционирования. В основе поведенческого моделирования лежат модели и методы имитационного моделирования систем массового обслуживания, сети Петри, возможно применение конечно-автоматных моделей, описывающих поведение системы, как последовательности смены состояний.

Поведенческие аспекты приложений отражает методика IDEF3. Если методика IDEFO связана с функциональными аспектами и позволяет отвечать на вопросы “Что делает система?”, то в IDEF3 детализируются и конкретизируются IDEFO-функции, IDEF3-модель отвечает на вопросы “Как система это делает?” Язык IDEF3 — язык диаграмм, помогающий разработчику моделей наглядно представить моделируемые процессы. В IDEF3 входят два типа описаний: 1) процесс-ориентированные в виде последовательности операций; 2) объект-ориентированные, выражаемые диаграммами перехода состояний, характерными для конечно-автоматных моделей.

На рис. 6.5 представлен пример процесс-ориентированнойIDEF3-диаграммы. Здесь функции (операции) показаны прямоугольниками с горизонтальной чертой, отделяющей верхнюю секцию с названием функции

от нижней секции, содержащей номер функции. Связи, отражающие последовательность выполнения функций, изображаются сплошными линиями-стрелками. Для указания разветвлений и слияний связей (их принято называть перекрестками) используют квадраты, у которых одна или обе вертикальные стороны представлены двойными линиями, а внутри квадрата записан один из символов &, О или Х. При разветвлении эти символы означают реакцию всех, некоторых или только одной из последующих функций на входное воздействие соответственно. Аналогичный смысл имеют символы &, О или Х при слиянии - последующая функция начинает выполняться после окончания всех, некоторых или только одной из входных операций.

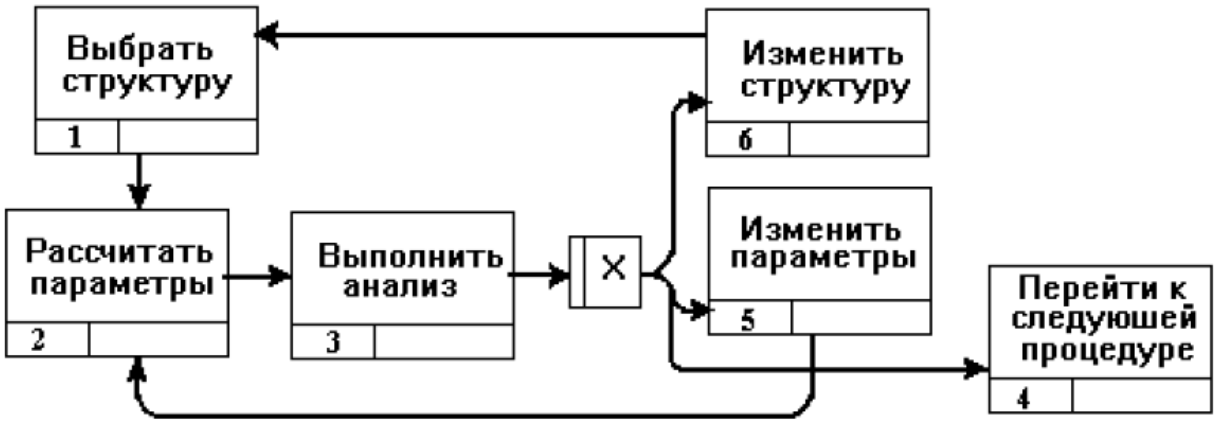


Рис. 6.5.IDEFS-диаграмма последовательсноти операций

На рис. 6.6 представлены пример объект-ориентированной IDEF3-диаграммы. В таких диаграммах имеются средства для изображения состояний системы, активностей, переходов из состояния в состояние и условий перехода.

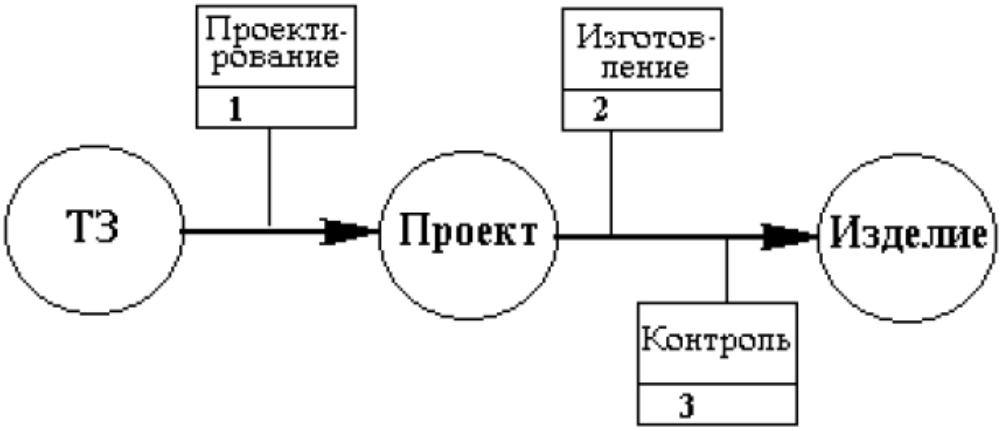


Рис. 6.6.IDEF3- диаграмма перехода состояний

Диаграммы IDEF0 или IDEF3 могут быть преобразованы в имитационные модели, если задать дополнительные свойства функций, характеризующие затраты ресурсов. Чаще всего имитационные модели представляют в виде сетей Петри. Преобразование связано с введением времени в функциональнуюIDEF0 или в поведенческую IDEF3-модель, с заменой функций переходами, а объектов, отождествляемых со стрелками блоков ICOM, с метками в сетях Петри.

4.3. Методика IDEF1X.

IDEF1 — методика *информационного (инфологического)* проектирования приложений, в настоящее время применяется ее усовершенствованный вариант IDEF1X. В IDEF1X имеется ясный графический язык для описания объектов и отношений в приложениях. Это язык диаграмм сущность-связь.

Основные компоненты описаний в IDEF1X: сущности (блоки), отношения (связи), атрибуты.

Сущность — множество объектов, обладающих общими свойствами (в языках программирования понятие сущности совпадает с понятием типа). Конкретные элементы этого множества называют *экземплярами* сущности. Атрибуты характеризуют свойства сущностей, их значения однозначно идентифицируют экземпляры сущностей. Если сущность A может быть определена только с помощью ссылки на свойства некоторой другой сущности B, то A называется зависимой (дочерней) сущностью, а B выступает в роли родительской сущности.

Сущности в IDEF1X-диаграммах изображаются в виде прямоугольников, при этом у зависимых сущностей углы прямоугольников должны быть скругленными.

Отношения (связи) между сущностями в IDEF1X являются бинарными отношениями. Выделяют *идентифицирующие* отношения — связи типа родитель-потомок, в которых потомок (зависимая сущность) однозначно определяется своей связью с родителем, и неидентифицирующие отношения, означающие, что у связанного этим отношением экземпляра одной сущности может быть, а может и не быть соответствующего экземпляра второй сущности (пример идентифицирующего отношения изготовитель-товар, неидентифицирующего отношения — рабочая станция — дигитайзер). Идентифицирующее отношение изображают на диаграмме сплошной линией между прямоугольниками связанных сущностей, неидентифицирующее отношение показывают пунктирной линией. На дочернем конце линии должно быть утолщение (жирная точка). Мощность k связи - число экземпляров зависимой сущности, соответствующее одному экземпляру родительской сущности. Известное значение мощности может быть указано около утолщенного конца линии связи. При этом символ p означает $k \geq 1$, а

символу z соответствует $k = 0$ или 1. Отсутствие символа интерпретируется $k \leq 0$.

Различают также специфические и неспецифические отношения. *Неспецифические* отношения — это связи типа “многие ко многим” и обозначаются сплошной линией с утолщениями на обоих концах.

В отношениях родитель-потомок возможно наличие у потомка единственного родителя (характеристическая связь) или нескольких родителей (ассоциативная связь). Выделяют также отношения категоризации (наследования), отражающие связи между некоторой общей сущностью и вариантами ее реализации (категориями). Примером категориальной связи является отношение тип прибора — альтернативные варианты этого прибора.

Среди атрибутов различают ключевые и неключевые. Значение *ключевого атрибута (ключа)* однозначно идентифицирует экземпляр сущности. *Внешний ключ* - это атрибут (или атрибуты), входящий в ключ родителя и наследуемый потомком. На IDEF1X-диаграммах ключи записывают в верхней части прямоугольника сущности, причем внешние ключи помечают меткой FK (ForeignKey), неключевые атрибуты помещают в нижнюю часть прямоугольников. В идентифицирующих отношениях все ключи родителя входят и в ключи потомка, в неидентифицирующих ключи родителя относятся к неключевым атрибутам потомка.

Нормальные формы отношений позволяют выявить атрибуты, которые целесообразно (с целью устранения избыточности) считать сущностями. Известно несколько нормальных форм, обычно используют первые три из них.

Первая нормальная форма требует, чтобы шапка таблицы (отношения) была одноэтажная (т.е. все атрибуты характеризуются атомарными значениями), строки-дубли должны быть устранены.

Вторая нормальная форма устанавливается для сущностей, удовлетворяющих условиям первой нормальной формы и имеющих составные ключи. Она определяется отсутствием атрибутов, зависящих только от части составного ключа. Подобные атрибуты должны быть выделены в отдельные сущности.

Третья нормальная форма дополнительно характеризуется отсутствием транзитивных связей (взаимозависимости) атрибутов.

Разработка информационной модели по IDEF1X выполняется за несколько стадий.

Стадия 0. Выяснение цели проекта, составление плана сбора информации. Обычно отправным пунктом для разработки информационной модели является IDEF0-модель.

Стадия 1. Выявление и определение сущностей. Это неформальная процедура.

Стадия 2. Выявление и определение основных отношений. Результат представляется или графически в виде ER-диаграмм или в виде матрицы отношений, элемент которой $A_{ij}=1$, если имеется связь между сущностями *i* и j, иначе $A_{ij}=0$. Транзитивные связи не указываются.

Стадия 3. Детализация неспецифических отношений, определение ключевых атрибутов, установление внешних ключей. Детализация неспецифических отношений заключается в замене связей "многие ко многим" (M ↔M) на связи "M↔1" и "1↔*M*" введением сущности-посредника. Например, отношение "преподаватель — студенческая группа" может быть заменено на отношения этих сущностей с сущностью-посредником "расписание".

Стадия 4. Определение атрибутов и их принадлежности сущностям.

Основные элементы графического языка IDEF1X представлены на рис. 6.7.

Между IDEFO и IDEF1X-моделями одного и того же приложения существуют определенные связи. Так, стрелкам на IDEFO-диаграммах соответствуют атрибуты некоторых сущностей в IDEF1X-моделях, что нужно учитывать при построении информационных моделей.

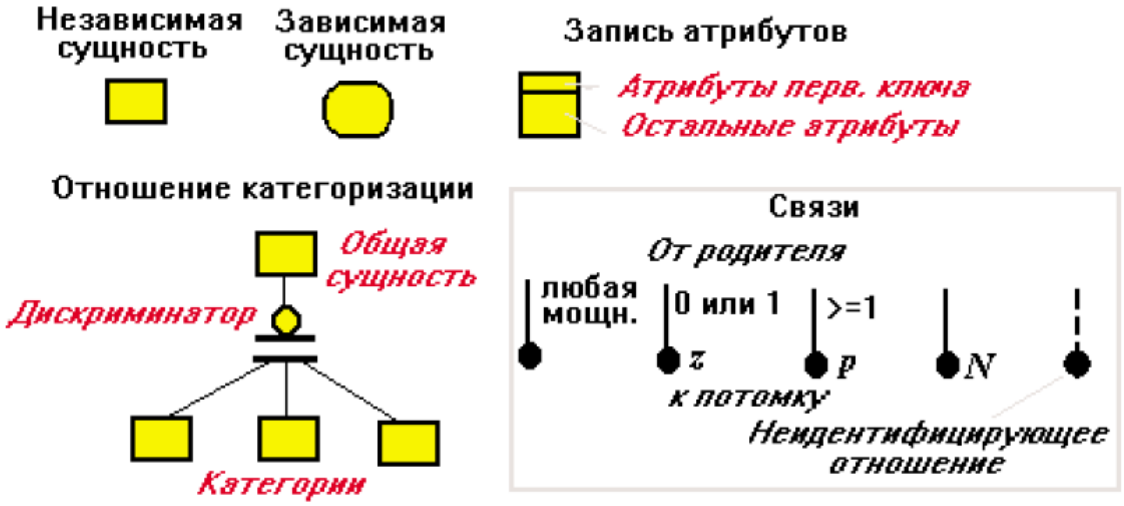


Рис. 6.7. Элементы языка IDEF1X

4.4. Обзор других методик IDEF.

Методика IDEF4 реализует *объектно-ориентированное проектирование* больших систем. При процедурном программировании кодированию предшествует удобное для пользователя изображение программы на графическом языке граф-схем или диаграмм потоков данных.Целесообразно

иметь аналогичные средства, учитывающие специфику объектно-ориентированного программирования.

В частности, такие средства предоставляет IDEF4. Другим вариантом графического языка поддержки объектноориентированного проектирования ПО является язык UML (UnifiedModelingLanguage), рассматриваемый консорциумом OMG на предмет стандартизации.

Методика IDEF4 содержит графический язык для изображения взаимосвязей классов, атрибутов, методов в виде ряда диаграмм: типов, наследования, протоколов, клиентов, таксономии методов. Примеры диаграмм приведены на рисунках. В этих диаграммах прямоугольники с поперечными линиями соответствуют классам, имена которых указаны ниже поперечных линий, а сверху линий записаны идентификаторы атрибутов. Процедуры (методы) в IDEF4 изображены прямоугольниками без поперечных линий. Передаваемые параметры записаны в овальных фигурах.

Примеры диаграмм типов данных и наследования приведены на рис. 6.8 и 6.9 соответственно. В примере рис. 6.9 объекты класса “Деталь” наследуют часть атрибутов из классов “Геометрия” и “Материал”.

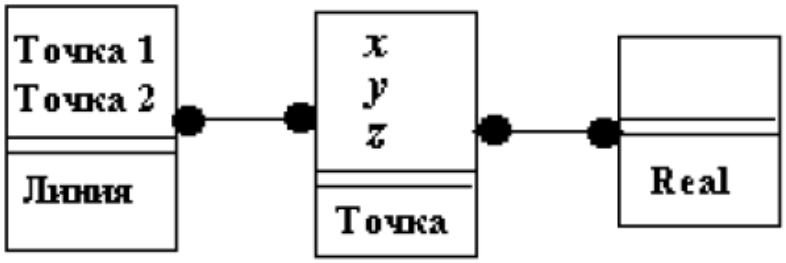


Рис. 6.8. IDEF4-диаграмма типов

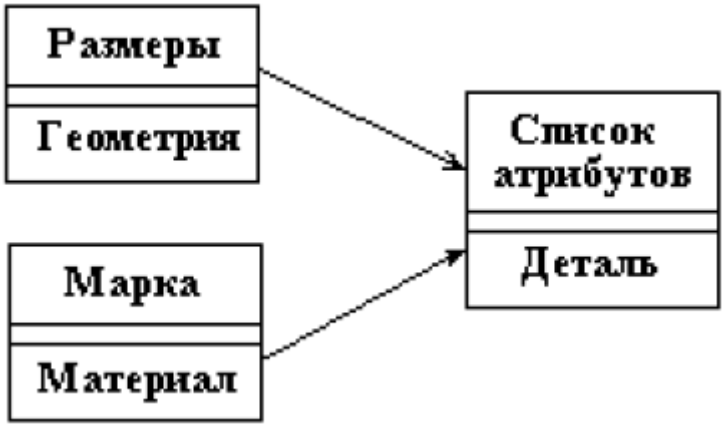


Рис. 6.9. IDEF4-диаграмма наследования

Из рис. 6.10 ясно, что для процедуры моделирования некоторой схемы входными параметрами являются атрибуты источников сигналов и

параметрыкомпонентов схемы, а результатом — значения выходных параметров.

На рис. 6.11 показан пример классификации методов, согласно которой методы решения перечисленных частных задач относятся к методам дискретной оптимизации.

Связи вызывающих и вызываемой процедур представлены на рис 6.12.

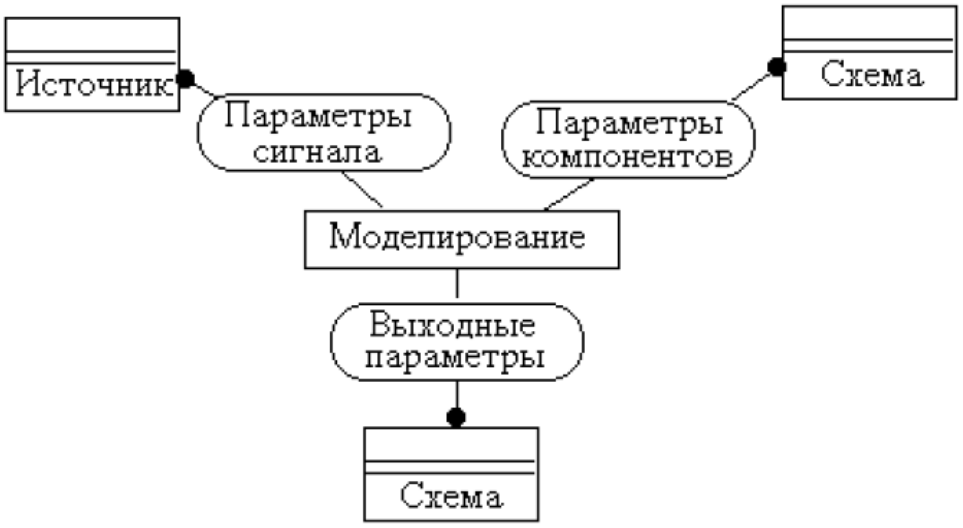


Рис. 6.10. IDEF4-диаграмма протоколов

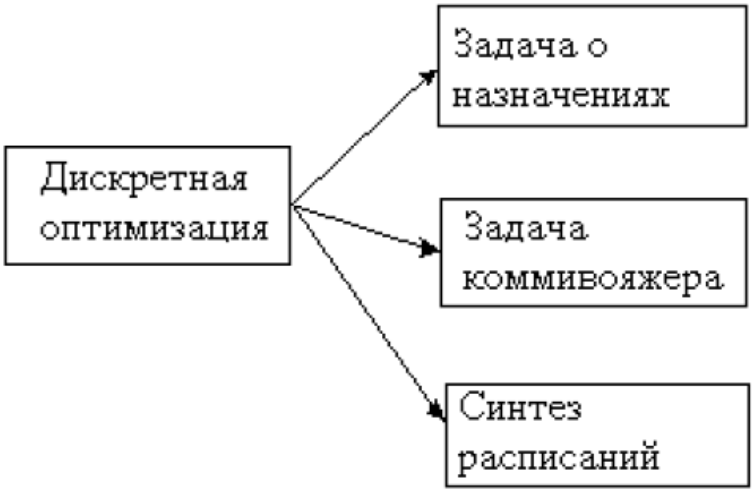


Рис. 6.11. IDEF4-диаграмма таксономии методов

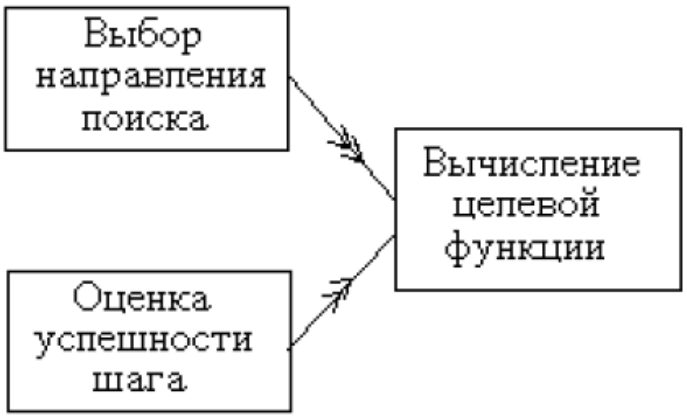


Рис. 6.12. IDEF4-диаграмма клиентов

Методика IDEF5 направлена на представление *онтологической информациприложения* в удобном для пользователя виде. Онтология связана с определениями ипонятиями, используемыми для характеристики объектов и процессов вместе с их взаимосвязями. Для этого приметают символические обозначения (дескрипторы) объектов, их ассоциаций, ситуаций и схемный язык описания отношений (классификации, часть-целое, перехода и т.п.), составляют словарь дескрипторов. В методике имеются правила связывания объектов (термов) в правильные предложения, языковые механизмы для установления соответствия между объектами реального мира и их идентификаторами (дескрипторами).

В IDEF5 имеются две части:

1) схемный язык;

2) язык разработки (elaboration).

Основные символы схемного языка представлены на рис. 6.13, пример классификационной схемы — на рис. 6.14 и пример диаграммы перехода состояний с символикой IDEF5 — на рис. 6.15.

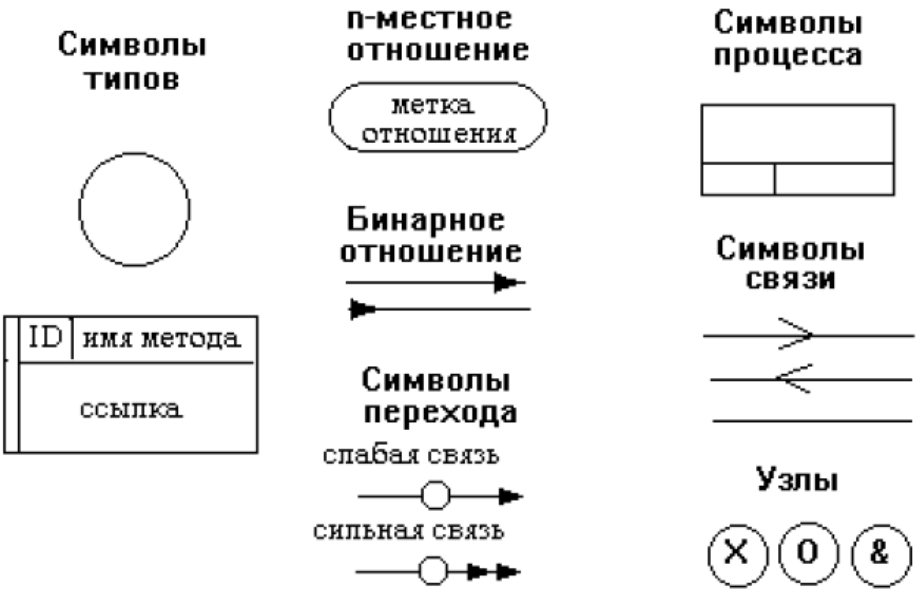


Рис. 6.13. Символы графического языка IDEF5

Рис. 6.14. Диаграмма классификации

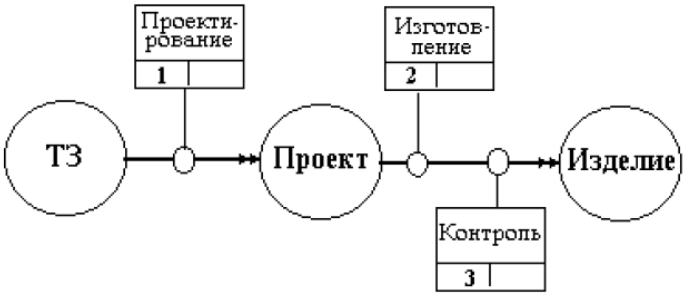


Рис. 6.15. Диаграмма перехода состояний в IDEF5

Развитиеметодикреинжиниринга (BPR Business Process Reenginiring) продолжаетсявСШАпопрограмме IICE (Information Integration for Concurrent Engineering). Разработаны, но пока (1998 г.) не получили официального статуса от органов стандартизации методики, имеющие индексы IDEF6, 8, 9, 14, разрабатываются методики IDEF7, 10, 12.

IDEF6 (DesignRationaleCapture) направлена на получение и представление решений по выбору стратегии проектирования и обоснованию предпринятых шагов. В отличие от других методик IDEF, в которых фиксируются результаты проектирования, в IDEF6 главный упор сделан на пути получения этих результатов и обоснование промежуточных решений. Такой подход особенно важен при разработке сложных систем в недостаточно определенных ситуациях. Фиксация шагов и обоснований помогает при дальнейших модернизациях систем, сохранению и использованию рационального опыта проектирования. Методика упорядочивает обнаружение и устранение неопределенностей, ошибок, неудовлетворенных ограничений. Язык методики включает предложения, связывающие компоненты проекта с пунктами обоснования. Под компо-

нентами проекта обычно подразумевают компоненты, отражаемые на диаграммах IDEF0-5, например, стрелки ICOM из IDEF0, сущности, атрибуты, отношения из IDEF1X, объекты, сообщения, события из IDEF4 и т.п. В качестве пунктов обоснования могут фигурировать стандарты, экспериментальные данные, ограничения и т.п.

IDEF8 (Human-SystemInteractionDesign) предназначена для проектирования взаимодействия человека с технической системой. Эта методика не является методикой создания графического пользовательского интерфейса и потому обычно дополняется некоторой системой GUI (GraphicUserInterface). Здесь определяется содержание (на абстрактном уровне) той части работы, которую выполняет человек. Создаваемые сценарии должны удовлетворять ряду оговоренных в методике принципов таких, как уменьшение нагрузки на человека, идентичность средств диалога в разных системах, наличие обратной связи для исправления ошибок, хранение истории диалога, помощь советами по выполнению действий и т.д.

IDEF9 (BusinessConstraintDiscovery) нацелена на выявление разнообразных ограничений (технических, физических, юридических, политических, организационных), которые должны быть учтены при разработке системы, и для анализа их влияния на принимаемые решения в процессе реинжиниринга. Обычно в качестве систем фигурируют сложные информационные системы с ориентацией на экономические и управленческие приложения. Ограничение — это отношение,

которое должно соблюдаться. Ограничения делятся на контексты (группы родственных ограничений). Применение IDEF9 заключается в выполнении нескольких шагов: 1) сбор свидетельств (фактов, указывающих на наличие ограничения); 2) классификация — определение контекстов, объектов, отношений; 3) прогнозирование — выявление ограничений на основе свидетельств; 4) отбор значимых ограничений; 5) определение экспертов для тестирования результатов; 6) детализация и фильтрация ограничений. В методике даны рекомендации по выполнению этих шагов. Предлагается графический язык, элементами которого являются система, блоки ограничений, контексты, линии связи, логические связки OR, AND, XOR (исключающее ИЛИ).

IDEF14 (NetworkDesign) предназначена для проектирования корпоративных вычислительных сетей, их представления на графическом языке с описанием конфигураций, очередей, сетевых компонентов, требований к надежности и т.п. Чаще всего методика применяется для модернизации уже существующих сетей. Поэтому в ней предусматривается разработка моделей как “ASIS”, так и “TOBE”. Проектирование включает в себя определение топологии сети или схемы коммуникаций, реализацию нужного качества обслуживания, анализ функционирования (трафик, дисциплины обслуживания в узлах, протоколы доступа). Модель топологии

дополняется моделями очередей, надежности, материальных затрат. Важную роль играет библиотека методов построения и компонентов сетей. Методика основана на выполнении ряда шагов: установление целей модернизации, исследование существующей сети, определение типов компонентов в ней, построение модели “ASIS”, ее верификация, анализ результатов, корректировка с переходом к “TOBE”. В графическом языке IDEF14 сети и подсети изображаются в виде облаков, топологические связи представляются линиями, для узлов используются специальные иконки, возможны поясняющие надписи, список характеристик размещается в прямоугольниках.

5. Унифицированный язык моделирования UML.

Язык UML положен в основу RationalUnifiedProcess (RUP) — известной методологии проектирования информационных систем, развиваемой фирмой RationalSoftware. В UML также используется ряд диаграмм.

К основным следует отнести прежде всего диаграммы классов. Они имеют следующие отличия от аналогичных диаграмм в IDEF4.

Во-первых, в прямоугольнике класса имеются три секции, в верхней секции записывается имя класса, в средней секции — атрибуты, в нижней части — процедуры класса. При записи атрибутов указываются символ доступности (+ — public, # — protected, - - private), идентификатор атрибута, тип атрибута. Запись процедуры аналогична подобным записям в языках программирования: указываются имя процедуры и в скобках — список параметров.

Во-вторых, в диаграммах классов UML отображение отношений часть-целое (отношений агрегации) выполняется с помощью линий с ромбовидной стрелкой, направленной от класса-части к классу-целому, и отношений наследования (суперкласс-подкласс) — с помощью линий с обычной стрелкой, направленной от подкласса к суперклассу.

Поведенческий аспект моделирования отражен в диаграммах процессов, имеющихся в UML. Они бывают двух типов — диаграммы сценариев (ДС) и диаграммы взаимодействия объектов (ДВО).

Сценарий — это последовательность событий, заключающихся в воздействиях (посылках сообщений) одного объекта на некоторый другой объект. В ДС объекты изображаются прямоугольниками и располагаются в горизонтальном ряду объектов. Ось времени направлена от этого ряда вертикально вниз. От каждого объекта параллельно оси времени идут такназываемые их линии жизни (lifelines). Каждое событие изображается горизонтальной линией со стрелкой от линии жизни объекта, посылающего сообщение, к линии жизни объекта, принимающего сообщение. Над этими

линиями возможен поясняющий текст. Линии располагаются одна над другой в порядке, в котором события совершаются (пример ДС на рис. 6.16).

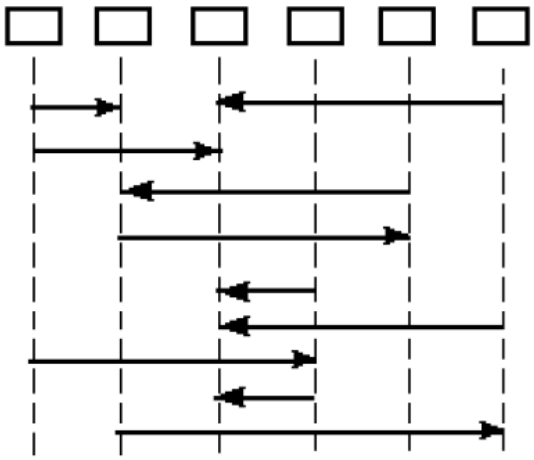


Рис. 6.16. Вид диаграммы сценариев

Диаграмма ДВО представляет собой граф, в котором вершины соответствуют объектам, а ребра — воздействиям. Около ребер возможны поясняющие записи, в частности, последовательные номера, указывающие порядок совершения событий.

К числу других диаграмм относятся диаграммы использования, цель которых — отобразить взаимодействие системы с пользователем. В этих диаграммах отображены в виде овалов те функции, которые непосредственно должен (или может) выполнять пользователь. При этом пользователи различаются ролями, выполняемыми ими при эксплуатации системы.

Проектирование информационной системы в RUP начинается с построения диаграмм использования. При этом определяется и согласовывается внешняя функциональность системы и в итоге формируется техническое задание на разработку ПО. Далее разрабатываются диаграммы взаимодействия “пользователь-система”, при этом выявляются необходимые объекты, строятся диаграммы классов, формируется компонентная структура ПО.

6. Программное обеспечение CASE-систем для концептуального проектирования.

На рынке программных продуктов имеется много CASE-систем для концептуального проектирования АС.

Чаще всего в них поддерживается методология IDEF. В России широко известны программы BPwin, ERwin, OOwin фирмы PlatinumTechnology, Design/IDEF фирмы MetaSoftware, CASE-Аналитик фирмы Эйтэкс, Silverrun фирмы CSA и др.

BPwin (BusinessProcessing) предназначена для разработки функциональных моделей по методике IDEF0.

ERwin предназначена для разработки информационных моделей по методике IDEF1X. Имеются средства, обеспечивающие интерфейс с серверами БД (от пользователя скрыто общение на SQL-языке), перевод графических изображений ER-диаграмм в SQL-формы или в форматы других популярных СУБД. Предусмотрены интерактивные процедуры для связывания дуг IDEF0 с сущностями и атрибутами IDEF1X, т.е. для установления связей между BPwin и ERwin.. В систему включены также типичные для CASE средства разработки экранных форм.

OOwin служит для поддержки объектно-ориентированных технологий проектирования информационных систем. Один из способов использования OOwin — детализация объектно-ориентированной модели на базе созданной ER-модели. При преобразовании ER в OO-представление сущности и атрибуты становятся классами (множествами подобных объектов). Классы могут быть дополнены описанием услуг класса, т.е. выполняемых операций, передаваемых и возвращаемых параметров, событий. Другой способ использования OOwin — реинжиниринг, так как модернизация проводится на уровне существующей модели.

Система Design/IDEF (фирма MetaSoftware) предназначена для концептуального проектирования сложных систем. С ее помощью разрабатываются спецификации, IDEF0 и IDEF ^-диаграммы, словари данных, проводится документирование и проверяется непротиворечивость проектов. Имеется дополнительная система Design/CPN, позволяющая проводить имитационное моделирование на основе моделей, преобразованных в цветные сети Петри.

Другой известной инструментальной средой моделирования приложений является Designer/2000 фирмы Oracle. Модель приложения может быть сгенерирована по ответам пользователя на вопросы системы. Используются собственные методики Oracle, позволяющие строить диаграммы потоков данных, сущность-отношение, иерархические деревья данных с возможностью их представления в SQL формах и, следовательно, поддерживается связь с любыми СУБД, работающими в ODBC.

Система Silverrun (фирма ComputerSystemsAdvisors) предназначена для анализа и проектирования информационных систем. Реализовано раздельное функциональное и информационное моделирование. Включает в себя четыре основные подсистемы: моделирование бизнес-процессов, построение моделей сущность-отношение, инфологическое проектирование реляционных баз данных, управление групповой работой. Имеет интерфейс к Oracle, Informix, Sybase и ряду других СУБД.

Среди отечественных систем выделяется CASE Аналитик, в которой выполняется построение диаграмм потоков данных, получение отчетов, генерация макетов документов и др. Имеется интерфейс к ERwin.

Методология объектно-ориентированного анализа и проектирования ПОпометодике Г.Буча с использованием языка UML реализована в системах RationalRose (фирма RationalSoftwareCorporation) и PlatinumParadigmPlus (фирма PlatinumTechnology). В RationalRose поддерживается генерация кода по построенным диаграммам классов, обратное моделирование (т.е. построение UML-модели по программному коду на таких языках, как C++, Java, VisualBasic, IDLCORBA), визуальное программирование. Язык UML применяют и в ряде других систем, например, в инструментальной среде объектно-ориентированного проектирования ПО objectiF (фирма microTOOL), в которой автоматически генерируется программный код по графическомуUML-описанию.

Ряд программных продуктов, реализующих IDEF-модели, разработаны фирмой KBSI, в частности, ProSim реализует IDEF3, SmartER — IDEF1 и IDEF1X, SmartClass — IDEF4.

Поведенческое моделирование предприятий предусмотрено также в некоторых системах реинжиниринга, например, в системе BAAN IV

Для преобразования функциональных или поведенческих моделей в имитационные применяют специальные программы. Так, вместе с программой BPWin для получения имитационных моделей используют программу BPSimulator. Преобразование IDEF0-модель □ сеть Петри реализовано в таких программах, как CPN/Design (фирма MetaSoftware) со специальным языком программирования ML, ProTem(SoftwareConsultantsInternationalLimited) с вариацией типов меток, PACE (Grossenbachersoftware) с программированием на языке Smalltalk.

7. Метамодели и стандарты CDIF (CASE Data Interchange Format).

Метамодель — средство, являющееся инвариантным к частным представлениям индивидуальных пользователей, служащее промежуточным звеном в процедурах взаимодействия приложений, характеризуемых своими локальными моделями.

Место метамодели в информационных процессах взаимодействия иллюстрирует рис. 6.17. Из рисунка ясно, что вместо непосредственного обращения одного приложения к другому, при котором каждое приложение должно иметь конверторы всех других локальных моделей, используется трансляция передаваемой информации на промежуточный язык метамодели, а принимающее приложение переводит метамодельное представление в свой собственный формат. Метамодельный подход имеет ряд преимуществ, например, каждое приложение становится открытым и может развиваться

независимо от других, система не имеет ограничений на включение новых приложений.

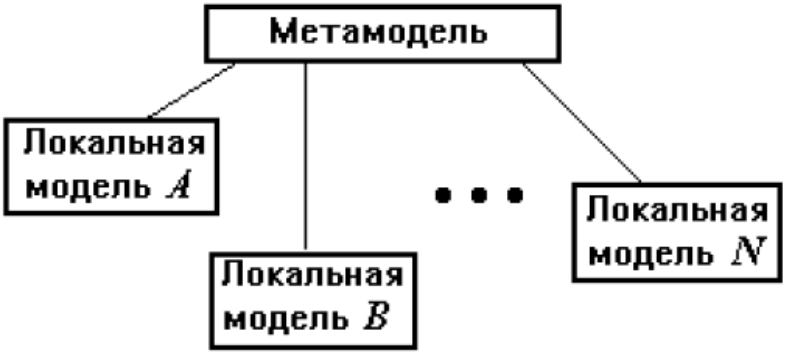


Рис. 6.17. Место метамодели в процессах информационного обмена.

Примерами метамоделей могут служить технология ODBC взаимодействия различных СУБД, основанная на языке SQL, графические системы типа GKS, концепция байт-кодов в языке Java и т.п.

В технологиях проектирования АС и реинжиниринга предприятий важное место отводится разработке метамоделей, направленных на взаимную трансформацию функциональных, информационных и структурных моделей. Для этого, в частности, требуется систематизация понятий, фигурирующих в приложениях, и построение словарей соответствия моделей этих типов.

Другое важное назначение метамоделей — интеграция CASE-средств разных производителей. Такая интеграция требуется, например, при недостаточных возможностях каждого из доступных CASE пакетов в отдельности, для доступа в условиях изменения программного и лингвистического обеспечений к информации, разработанной с помощью разных версий CASE-систем и накапливающейся длительное время в архивах.

Целям интеграции CASE-средств разных производителей служат стандарты серии CDIF, разрабатываемые организацией EIA (ElectronicsIndustriesAssociation) и признаваемые Международной организацией стандартизации ISO (InternationalStandardOrganization).

Метамодель в CDIF определяется, как средство, с помощью которого осуществляется правильная интерпретация данных при их передаче из одной CASE-среды в другую. Такая интерпретация требуется при взаимодействии сред, использующих различные формы представления однородной в смысловом отношении информации. Другими словами, метамодель применяют для передачи и правильной интерпретации данных с одинаковой семантикой, но с разным представлением в частных CASE системах. Например, данные, близкие в семантическом отношении, но различающиеся по представлению, фигурируют в методиках информационного моделирования (datamodeling), моделирования потоков данных

(dataflowmodeling), событийного моделирования переходов состояний (stateeventmodeling), объектно-ориентированного анализа и проектирования (objectorientedanalysisanddesign). CDIF-метамодель осуществляет интерфейс между ними.

Программное обеспечение, поддерживающее CDIF, позволяет представлять данные в желаемой форме (в соответствии с предметной областью). Например, конечно-автоматная модель может быть представлена в форме графа или матрицы перехода состояний, объектно-ориентированная модель — с использованием прямоугольников или произвольно очерченных фигур и т.п. Клиент, поддерживающий CDIF, транслирует форму источника информации в форму, доступную клиенту с сохранением семантики данных.

Очевидно, что для каждой предметной области, характеризуемой своим множеством семантически близких понятий можно построить свою метамодель. Такие предметные области в стандартах CDIF называют SubjectAreas, для многих предметных областей разработаны свои CDIF-стандарты (метамодели). Очевидно также, что потребности в метамоделях могут возникать для новых предметных областей, поэтому в CDIF отдельная методика посвящена включению в стандарты новых метамоделей. Имеются также общие для различных предметных областей компоненты метамоделей. Обычно интегрированная метамодель строится на основе парадигмы сущность-отношение.

Обменный файл в CDIF состоит из трех частей: заголовка (имя, дата, источник, способ кодирования и другие общие атрибуты), метамодели (указывается тип используемой метамодели) и собственно передаваемых данных.

Список стандартов CDIF приведен в приложении. Стандарты подразделены на три группы. Первая группа содержит обзор стандартов CDIF и общие правила их расширения.

Вторая группа определяет форматы представления данных, т.е. синтаксис и способы кодирования передаваемых данных.

Третья группа содержит стандарты, ориентированные на представление семантики передаваемых данных. Каждый из стандартов относится к определенной предметной области. Например, есть стандарты или проекты стандартов для таких областей, как объектно-ориентированный анализ и проектирование, моделирование бизнес-процессов, проектирование автоматизированных систем управления, описание потоков данных, данных в реляционных базах данных и др. Кроме того, введены иерархическая структура метамодели и возможности наследования, благодаря выделению наиболее общих частей, справедливых для многих предметных областей, и их представлению в отдельных стандартах.

Таким образом, в метамодели CDIF имеет место отделение семантики от способа представления данных. Правильная передача семантики сочетается с варьированием форм представления данных.