ЛЕКЦИЯ 9 МЕТОДИКИ ПРОЕКТИРОВАНИЯ АВТОМАТИЗИРОВАННЫХ СИСТЕМ

1. CASE-системы.

2. Спецификации проектов программных систем.

3. Методика IDEF0.

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 — Computer Aided System Engineering — 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

Computer Aided Software Engineering переводится, как автоматизированное проектирование программного обеспечения, соответствующие CASE-системы называют *инструментальными CASE* или *инструментальными средами* разработки ПО (одно из близких к этому названий — RAD — Rapid Application Development).

Среди систем CASE для концептуального проектирования различают системы функционального, информационного или поведенческого проектирования. Наиболее известной методикой *функционального проектирования* сложных систем является методика SADT (Structured Analysisand Design Technique), предложенная в 1973 г. Р.Россом и впоследствии ставшая основой международного стандарта IDEF0 (Integrated DEFinition 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 — software configuration management). 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 ПО с 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 инфологическое 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 ПО из 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 — Data Flow Diagram). 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. 1.

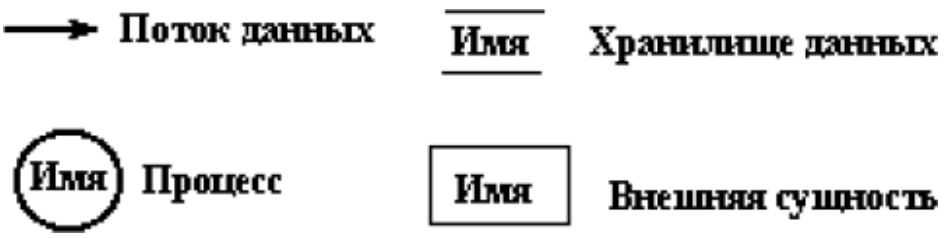


Рис. 1 Изображения элементов в нотации Йордана

Разработка DFD начинается с построения диаграммы верхнего уровня, отражающей связи программной системы, представленной в виде единого процесса, с внешней средой. Декомпозиция процесса проводится 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 из состояние 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. 2.

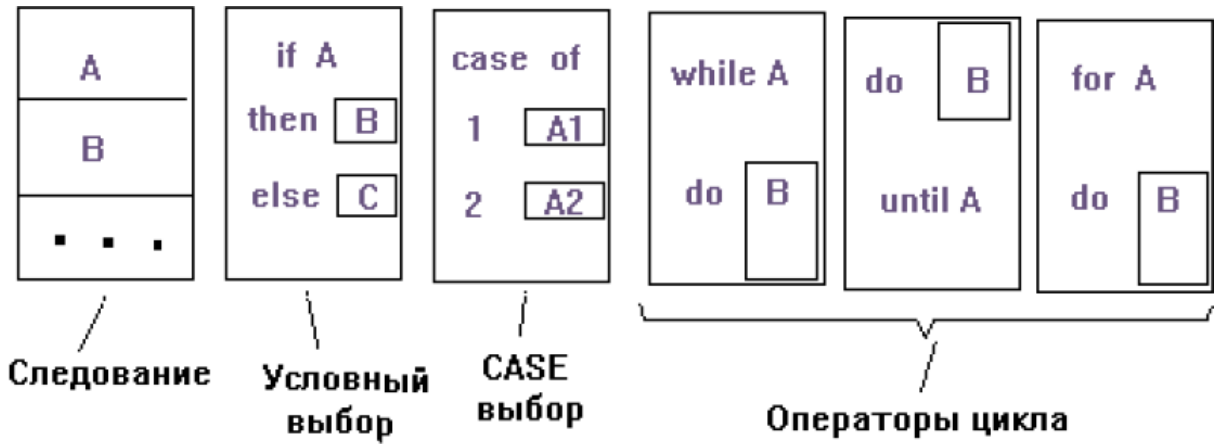


Рис. 2. Примеры описания операторов в визуальных языках программирования

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-ориентированными 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 Методика IDEF0.

Взаимосвязанная совокупность методик IDEF для концептуального проектирования разработана по программе Integrated Computer Aided Manufacturing 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 межкомпьютерных 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.

Как отмечено выше, наиболее известной методикой *функционального моделирования* сложных систем является методика SADT (Structured Analysisand Design Technique), 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 слабая формализованность 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. 3).

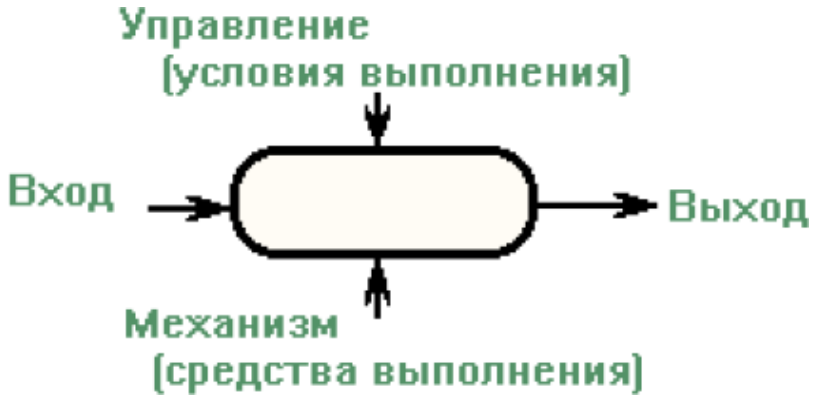


Рис. 3. Блок ICOM в IDEF0-диаграммах

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.

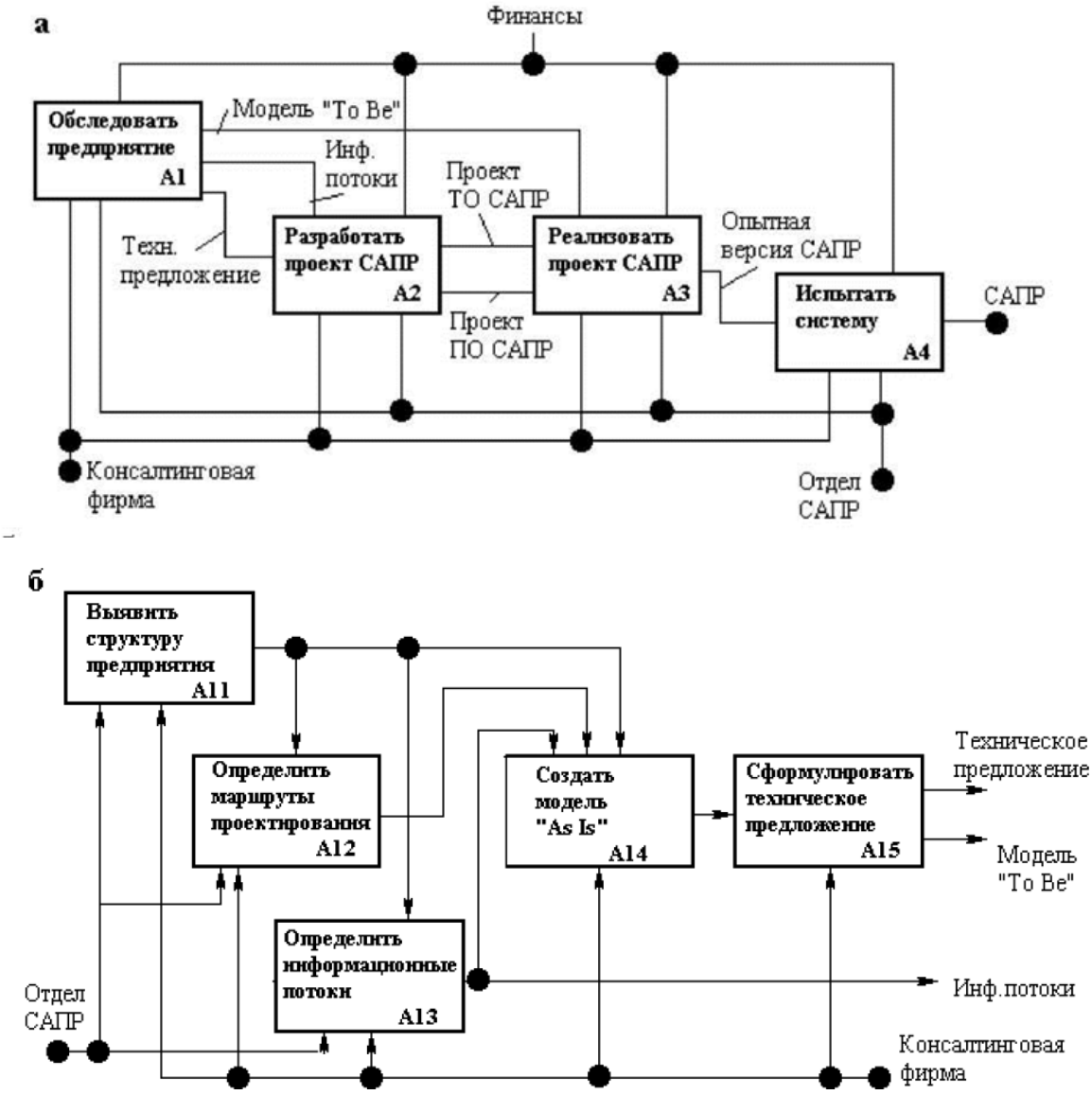
Блоки рис. 3 в англоязычной литературе называют блоками ICOM (Input — Control — Output - Mechanism).

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. 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. 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.



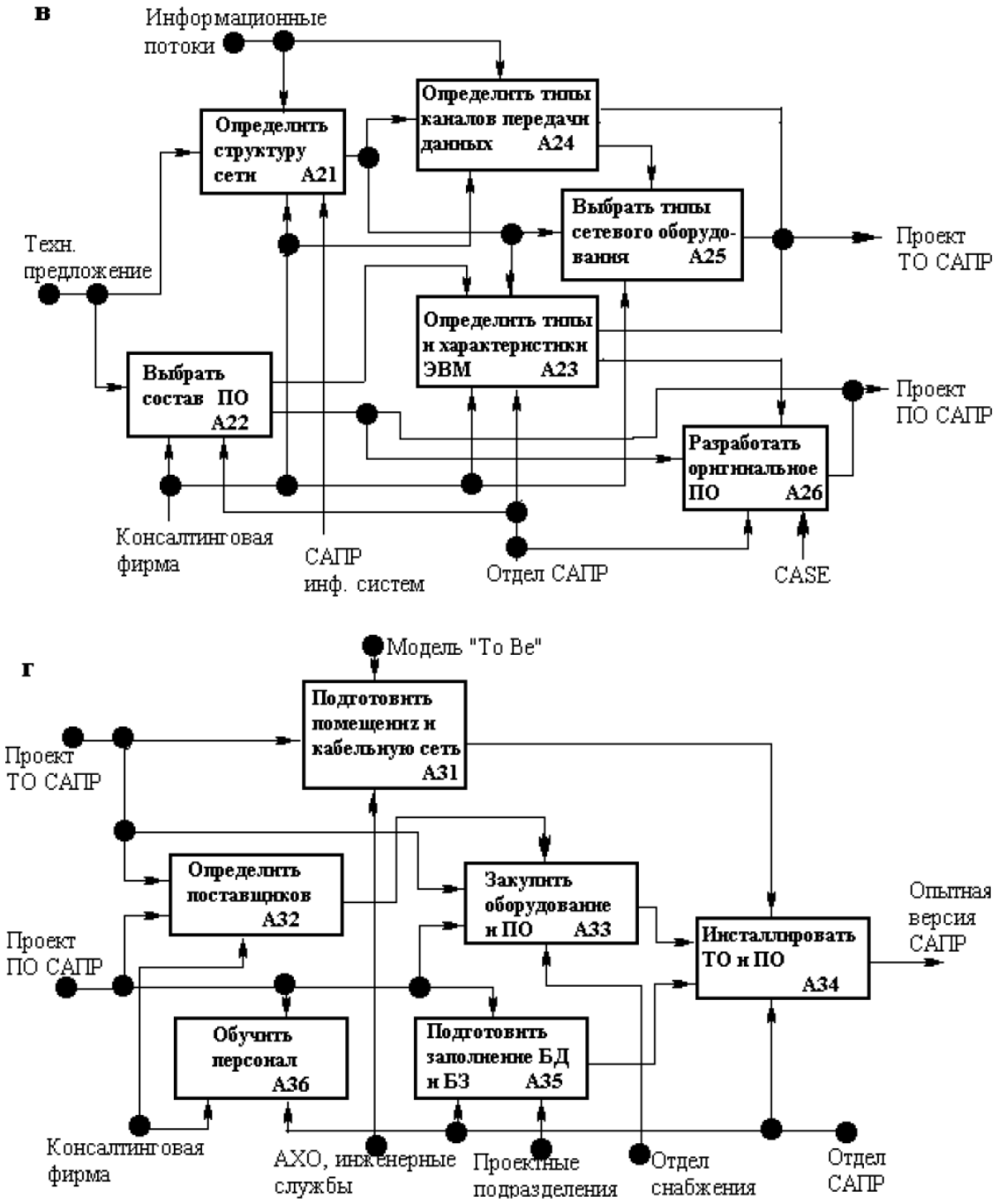


Рис. 4. Функциональная модель процесса создания САПР: а) IDEFO-диаграмма первого уровня; б) IDEFO-диаграмма обследования предприятия; в) IDEFO-диаграмма проектирования САПР; г) IDEFO-диаграмма реализации проекта САПР

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.