КОНСАЛТИНГ ПРИ АВТОМАТИЗАЦИИ ПРЕДПРИЯТИЙ: ПОДХОДЫ, МЕТОДЫ, СРЕДСТВА

Г.Н. Калянов,

ОГЛАВЛЕНИЕ

Предисловие

В современных условиях динамично развивается рынок комплексных интегрированных систем автоматизации предприятий и учреждений самого различного профиля (финансовых, промышленных, офисных) и самых различных размеров с разнообразными схемами иерархии, начиная от малых предприятий численностью в несколько десятков человек и завершая крупными корпорациями численностью в десятки тысяч сотрудников. Такие системы предназначены для решения задач как предприятия в целом (управление финансовыми ресурсами, управление запасами, планирование и производство, сбыт и снабжение, техническое обслуживание и ремонт оборудования, управление персоналом и т.п.), так и уровня его производственных подразделений, цехов и участков.

Фактически проблема комплексной автоматизации стала актуальной для каждого предприятия. Уже не стоит вопрос “надо или не надо автоматизировать”, предприятия столкнулись с проблемой: каким образом это осуществить. Подобная переориентация предприятий объясняется следующими основными причинами:

- повышением степени организационной и финансовой самостоятельности;
- выходом на зарубежный рынок;
- стремлением ряда западных компаний производить свои товары в России;
- завершением периода “островковой” автоматизации;
- возрастающей ориентацией предприятий на бизнес-процессы, т.е. деятельности, имеющие ценность для клиента;
- появлением на рынке как зарубежных, так и отечественных систем автоматизации, опыта их внедрения и использования и др.

Главная особенность индустрии систем автоматизации различных предприятий и учреждений, характеризующихся широкой номенклатурой входных данных с различными (и нетривиальными) маршрутами их обработки, состоит в концентрации сложности на начальных этапах анализа требований и проектирования спецификаций системы при относительно невысокой сложности и трудоемкости последующих этапов. Фактически здесь и приходит понимание того, что будет делать будущая система и каким образом она будет работать, чтобы удовлетворить предъявленным к ней требованиям. А именно нечеткость и неполнота системных требований, нерешенные вопросы и ошибки, допущенные на этапах анализа и проектирования, порождают на последующих этапах трудные, часто неразрешимые проблемы и, в конечном счете, приводят к неуспеху всей работы в целом.

С другой стороны, не существует двух одинаковых организаций. Даже в таком учреждении как Сбербанк России на уровне его отделений и филиалов выявляются различия в применяемых технологиях. А следовательно, простое тиражирование даже очень хорошей системы управления предприятием никогда не устроит заказчика полностью, поскольку не может учесть его специфики. Более того, в данном случае возникает проблема выбора именно той системы, которая наиболее подходит для конкретного предприятия. А эта проблема осложняется еще и тем, что ключевые слова, характеризующие различные системы, практически одни и те же:

- единая информационная среда предприятия;
- режим реального времени;
- независимость от законодательства;
- интеграция с другими приложениями (в том числе с уже работающими на предприятии системами);
- поэтапное внедрение и т.п.

Следует отметить, что для большинства предприятий необходим и предваряющий автоматизацию этап - наведение порядка в их деятельности, создание рациональных технологий и бизнес-

процессов. Речь даже не идет о жестком их реинжиниринге, в современных российских условиях происходит массовый бизнес-инжиниринг.

Ниже сформулированы два крайних подхода к автоматизации предприятий, полностью игнорирующие приведенные тезисы. Тем не менее читателю не составит труда привести примеры компаний, работающих по аналогичным схемам, и проектов, выполненных такими “консультантами”.

1. Короткое и “легкое” обследование предприятия и дальнейшее лоббирование одной из интегрированных систем управления предприятием под красивыми лозунгами настройки и адаптации под конкретного заказчика (кстати, стоимость такой настройки может на порядок превышать стоимость модулей системы и требовать серьезных временных затрат, совместимых с затратами на разработку новой системы). При этом, как правило, фирма-исполнитель еще до проведения обследования (да и вообще, до появления заказчика на ее горизонте) знает, какую именно систему она будет внедрять, и осуществляет соответствующую “адаптацию” результатов обследования.

2. Детальное обследование предприятия и разработка на его основе собственной системы управления, дублирующей существующие на предприятии технологии, что только усугубляет ситуацию (автоматизируя хаос и неразбериху, можно получить только “автоматизированный хаос”). А далее, с появлением нового заказчика, см. п.1.

Очевидно, что перечисленные подходы к автоматизации уже не могут устроить заказчика, желающего “увидеть” и скорректировать будущую систему до того, как она будет реализована физически, и в конечном итоге за свои немалые деньги получить реальную выгоду от ее эксплуатации.

С другой стороны, самостоятельно с задачей выбора и тем более разработки собственной системы предприятие справиться не в состоянии. И прежде всего потому, что на предприятии, как правило, отсутствует единая концепция автоматизации. Возникает необходимость в услугах независимых от производителей систем автоматизации консалтинговых фирм.

В книге обобщается опыт разработки консалтинговых проектов автоматизации различных предприятий и учреждений (банк, предприятие связи, автотранспортное предприятие, горно-обогатительный комбинат, молокозавод и др.), выполненных под руководством и при участии автора, и предлагаются основы используемого подхода для разработки таких проектов.

ВВЕДЕНИЕ

В.1. Понятие консалтинга в области информационных технологий

Термин "консалтинг" в России является каким-то аморфным и неконкретным. Практически каждая фирма, работающая на рынке информационных технологий, заявляет о предоставлении ею неких консалтинговых услуг. Что же следует понимать под консалтингом на самом деле? Какие требования предъявляются к консалтинговым структурам? Постараемся ответить на эти вопросы.

Консалтинг - это деятельность специалиста или целой фирмы, занимающихся стратегическим планированием проекта, анализом и формализацией требований к информационной системе, созданием системного проекта, иногда - проектированием приложений. Но все это до этапа собственно программирования или настройки каких-то уже имеющихся комплексных систем управления предприятием, выбор которых и осуществляется на основе системного проекта. И уж ни в коей мере сюда не входит системная интеграция. Консалтинг предваряет и регламентирует названные этапы.

Фактически консультантом выполняется два вида работ. Прежде всего это элементарное наведение порядка в организации: бизнес-анализ и реструктуризация (реинжиниринг бизнес-процессов). Это направление получило название “бизнес-консалтинг”. В конечном итоге речь, разумеется, идет об автоматизации, однако если мы будем автоматизировать существующий хаос, царящий на российских предприятиях, то в итоге получим не что иное как “автоматизированный хаос”. Любая организация - это довольно сложный организм, функционирование которого одному человеку понять просто невозможно. Руководство в общих чертах представляет себе общий ход дел, а клерк досконально изучил только свою деятельность, уяснил свою роль в сложившейся системе деловых взаимоотношений. Но как организация функционирует в целом, не знает, как правило, никто. И именно деятельность, направленная на то, чтобы разобраться в функционировании таких организмов, построить соответствующие модели и на их основе выдвинуть некоторые предложения по поводу улучшения работы некоторых звеньев, а еще лучше - бизнес-процессов (деятельностей, имеющих ценность для клиента) считается бизнес-консалтингом.

Другой вид работ - собственно системный анализ и проектирование. Выявление и согласование требований заказчика приводит к пониманию того, что же в действительности необходимо сделать. За этим следует проектирование или выбор готовой системы так, чтобы она в итоге как можно в большей степени удовлетворяла требованиям заказчика.

Кроме того, важный элемент консалтинга - формирование и обучение рабочих групп. Здесь речь идет не только о традиционной учебе, любые проекты, модели должны в итоге кем-то сопровождаться. Поэтому сотрудники предприятия с самого начала участвуют в проекте, им частично передаются внутрифирменные технологии. И по окончании работ они способны анализировать и улучшать бизнес-процессы в рамках собственной отдельно взятой организации.

Исторически консалтинговые компании появляются на рынке последними. Это связано с появлением спроса на их услуги, который возникает с переходом от “островковой” к комплексной автоматизации, когда предприятие не способно самостоятельно справится с вставшими перед ним проблемами, а следовательно рождается понимание, что необходимо платить не только за программное и аппаратное обеспечение, но и за отчеты и рекомендации. Консалтинговые структуры характеризуются следующими четырьмя позициями:

- знания и информация - главный и единственный их продукт;
- опыт персонала, приобретаемый годами и десятилетиями при работе над конкретными проектами;
- независимость;
- объективность.

Понятно, что полной независимости и объективности не бывает да и быть не может. Тем не менее, когда консалтинговая структура входит в компанию, занимающуюся собственными разработками или продвигающую западную систему автоматизации, внедрение которой стоит сотни и миллионы долларов - это нонсенс. С другой стороны, у каждого специалиста есть свои пристрастия, излюбленные продукты или подходы. Так что не стоит думать, что нанимая специалистов по консалтингу, предприятие получает истину в последней инстанции. Другое дело, что оно вправе рассчитывать на профессионализм и получение одного из лучших решений своей проблемы.

В.2. Цели и этапы разработки консалтинговых проектов

Основными целями разработки консалтинговых проектов являются:

- представление деятельности предприятия и принятых в нем технологий в виде иерархии диаграмм, обеспечивающих наглядность и полноту их отображения;
- формирование на основании анализа предложений по реорганизации организационно-управленческой структуры;
- упорядочивание информационных потоков (в том числе документооборота) внутри предприятия;
- выработка рекомендаций по построению рациональных технологий работы подразделений предприятия и его взаимодействию с внешним миром;
- анализ требований и проектирование спецификаций корпоративных информационных систем;
- рекомендации и предложения по применимости и внедрению существующих систем управления предприятиями (прежде всего классов MRP - manufacturing resource planning и ERP - enterprise resource planning).

Структура подхода к разработке консалтинговых проектов приведена на рис. В.1. Опишем перечисленные этапы более подробно.

1. Анализ первичных требований и планирование работ

Данный этап предваряет инициацию работ над проектом. Его основными задачами являются: анализ первичных бизнес-требований, предварительная экономическая оценка проекта, построение план-графика выполнения работ, создание и обучение совместной рабочей группы.

2. Проведение обследования деятельности предприятия

В рамках данного этапа осуществляется:

- предварительное выявление требований, предъявляемых к будущей системе;
- определение оргштатной и топологической структур предприятия;
- определение перечня целевых задач (функций) предприятия;
- анализ распределения функций по подразделениям и сотрудникам;
- определение перечня применяемых на предприятии средств автоматизации.

При этом выявляются функциональные деятельности каждого из подразделений предприятия и функциональные взаимодействия между ними, информационные потоки внутри подразделений и между ними, внешние по отношению к предприятию объекты и внешние информационные взаимодействия.

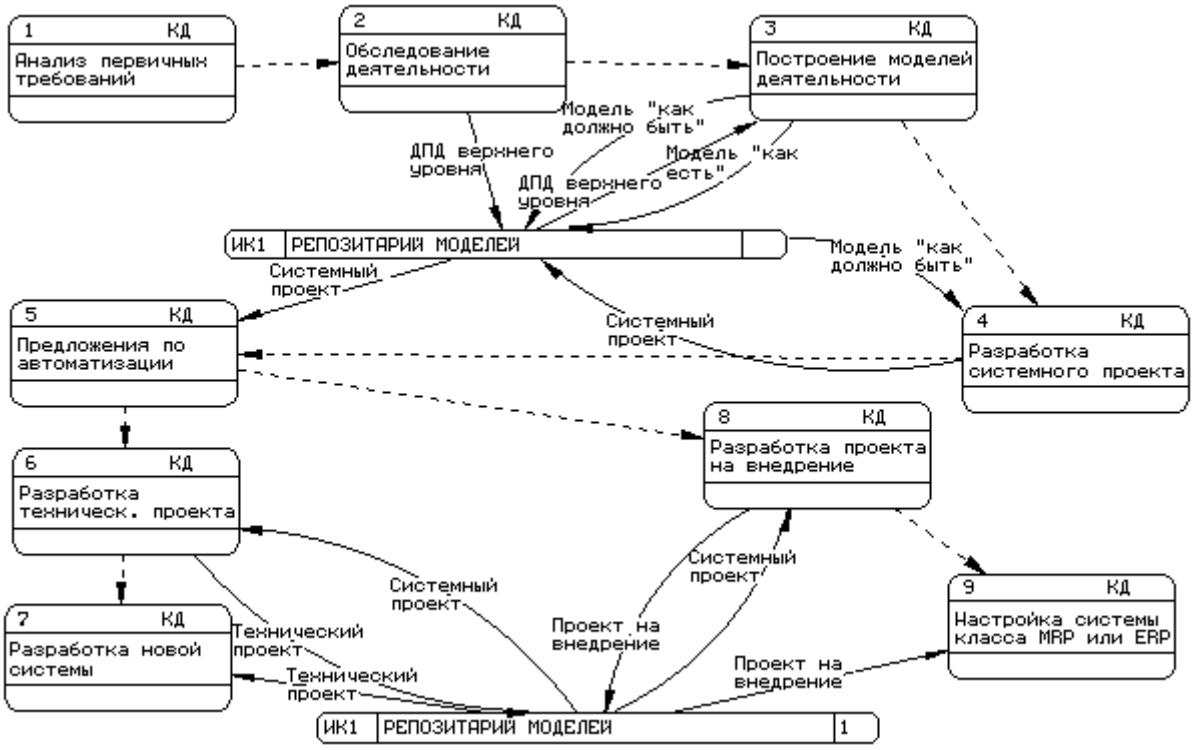


Рис. В.1. Структура подхода

В качестве исходной информации при проведении обследования и выполнении дальнейших этапов служат:

- данные по оргштатной структуре предприятия;
- информация о принятых технологиях деятельности;
- стратегические цели и перспективы развития;
- результаты интервьюирования сотрудников (от руководителей до исполнителей нижнего звена);
- предложения сотрудников по усовершенствованию бизнес-процессов предприятия;
- нормативно-справочная документация;
- опыт системных аналитиков в части наличия типовых решений.

Длительность обследования составляет 1-2 недели. По окончании обследования строится и согласуется с заказчиком предварительный вариант функциональной модели предприятия, включающей идентификацию внешних объектов и информационных взаимодействий с ними, а также детализацию до уровня основных деятельностей предприятия и информационных связей между этими деятельностями.

3. Построение моделей деятельности предприятия

На данном этапе осуществляется обработка результатов обследования и построение моделей деятельности предприятия следующих двух видов:

- ***модели "как есть"***, представляющей собой "снимок" положения дел на предприятии (оргштатная структура, взаимодействия подразделений, принятые технологии, автоматизированные и неавтоматизированные бизнес-процессы и т.д.) на момент обследования и позволяющей понять, что делает и как функционирует данное предприятие с позиций системного анализа, а также на основе автоматической верификации выявить ряд ошибок и узких мест и сформулировать ряд предложений по улучшению ситуации;
- ***модели "как должно быть"***, интегрирующей перспективные предложения руководства и сотрудников предприятия, экспертов и системных аналитиков и позволяющей сформировать видение новых рациональных технологий работы предприятия.

Каждая из моделей включает в себя полную структурную функциональную модель деятельности (например, в виде иерархии диаграмм потоков данных с разработанными для всех процессов нижнего уровня подробными их спецификациями на структурированном естественном языке или в виде иерархии SADT-диаграмм), информационную модель (как правило, с использованием нотации “сущность-связь”), а также, в случае необходимости, событийную (описывающую поведение) модель (с использованием диаграмм переходов состояний).

Переход от модели “как есть” к модели ”как должно быть” осуществляется следующими двумя способами.

1. **Совершенствование технологий на основе оценки их эффективности.** При этом критериями оценки являются стоимостные и временные затраты выполнения бизнес-процессов, дублирование и противоречивость выполнения отдельных задач бизнес-процесса, степень загруженности сотрудников (“легкий” реинжиниринг).
2. **Радикальное изменение технологий и переосмысление бизнес-процессов (“жесткий” реинжиниринг).** Например, вместо попыток улучшения бизнес-процесса проверки кредитоспособности клиента, может быть следует задуматься, а нужна ли вообще такая проверка? Возможно затраты на такие проверки каждого из клиентов во много раз превышают убытки, которые может понести компания в отдельных случаях недобросовестности (в случае, когда клиентов много, а суммы закупок незначительны)!

Построенные модели являются не просто реализацией начальных этапов разработки системы и техническим заданием на последующие этапы. Они представляют собой самостоятельный отделяемый результат, имеющий большое практическое значение, в частности:

1. Модель “как есть” включает в себя существующие неавтоматизированные технологии, работающие на предприятии. Формальный анализ этой модели позволит выявить узкие места в технологиях и предложить рекомендации по ее улучшению (независимо от того, предполагается на данном этапе автоматизация предприятия или нет).
2. Она позволяет осуществлять автоматизированное и быстрое обучение новых работников конкретному направлению деятельности предприятия (так как ее технология содержится в модели) с использованием диаграмм (известно, что "одна картинка стоит тысячи слов").
3. С ее помощью можно осуществлять предварительное моделирование нового направления деятельности с целью выявления новых потоков данных, взаимодействующих подсистем и бизнес-процессов.

4. Разработка системного проекта

Данный этап является первой фазой разработки собственно системы автоматизации (именно, фазой анализа требований к системе), на которой требования заказчика уточняются, формализуются и документируются. Фактически на этом этапе дается ответ на вопрос: "Что должна делать будущая система?". Именно здесь лежит ключ к успеху всего проекта автоматизации. В практике создания больших программных систем известно немало примеров неудачной реализации именно из-за неполноты и нечеткости определения системных требований.

На этом этапе определяются:

- архитектура системы, ее функции, внешние условия ее функционирования, распределение функций между аппаратной и программной частями;
- интерфейсы и распределение функций между человеком и системой;
- требования к программным и информационным компонентам системы, необходимые аппаратные ресурсы, требования к базе данных, физические характеристики компонент системы, их интерфейсы;
- состав людей и работ, имеющих отношение к системе;
- ограничения в процессе разработки (директивные сроки завершения отдельных этапов, имеющиеся ресурсы, организационные процедуры и мероприятия, обеспечивающие защиту информации).

Системный проект строится на основе модели “как должно быть” и включает функциональную модель будущей системы в соответствии с одним из общеупотребительных стандартов (например, IDEF0 или IDEF3), информационную модель, например, в соответствии со стандартом IDEF1X, а также техническое задание на создание автоматизированной системы (например, в соответствии с ГОСТ 34.602-89).

По завершении данного этапа (после согласования системного проекта с заказчиком) изменяется роль консультанта. Отныне он как бы становится на сторону заказчика, и одной из его основных функций на всех последующих этапах работ будет являться контроль на соответствие требованиям, зафиксированным в системном проекте.

Необходимо отметить следующее достоинство системного проекта. Для традиционной разработки характерно осуществление начальных этапов кустарными неформализованными способами. В результате заказчики и пользователи впервые могут увидеть систему после того, как она уже в большей степени реализована. Естественно, эта система отличается от того, что они ожидали увидеть. Поэтому далее следует еще несколько итераций ее разработки или модификации, что требует дополнительных (и значительных) затрат денег и времени. Ключ к решению этой проблемы и дает системный проект, позволяющий:

- описать, "увидеть" и скорректировать будущую систему до того, как она будет реализована физически;
- уменьшить затраты на разработку и внедрение системы;
- оценить разработку по времени и результатам;
- достичь взаимопонимания между всеми участниками работы (заказчиками, пользователями, разработчиками, программистами и т.д.);
- улучшить качество разрабатываемой системы, а именно: создать оптимальную структуру интегрированной базы данных, выполнить функциональную декомпозицию типовых модулей.

Системный проект полностью независим и отделяем от конкретных разработчиков, не требует сопровождения его создателями и может быть безболезненно передан другим лицам. Более того, если по каким-либо причинам предприятие не готово к реализации на основе проекта, он может быть положен "на полку" до тех пор, пока в нем не возникнет необходимость. Кроме того, его можно использовать для самостоятельной разработки или корректировки уже реализованных на его основе программных средств силами программистов отдела автоматизации предприятия.

5. Разработка предложений по автоматизации предприятия

На основании системного проекта осуществляется:

- составление перечня автоматизированных рабочих мест предприятия и способов взаимодействия между ними;
- анализ применимости существующих систем управления предприятиями для решения требуемых задач и формирование рекомендаций по выбору такой системы;
- совместное с заказчиком принятие решения о выборе конкретной системы управления предприятием или разработке собственной системы;
- разработка требований к техническим средствам;
- разработка требований к программым средствам;
- разработка предложений по этапам и срокам автоматизации.

6. Разработка технического проекта

На данном этапе на основе системного проекта и принятых решений по автоматизации осуществляется проектирование системы. Фактически здесь дается ответ на вопрос: "Как (каким образом) мы будем строить систему, чтобы она удовлетворяла предъявленным к ней требованиям?". Этот этап разделяется на два подэтапа:

- проектирование архитектуры системы, включающее разработку структуры и интерфейсов ее компонент (автоматизированных рабочих мест), согласование функций и технических требований к компонентам, определение информационных потоков между основными компонентами, связей между ними и внешними объектами;
- детальное проектирование, включающее разработку спецификаций каждой компоненты, разработку требований к тестам и плана интеграции компонент, а также построение моделей иерархии программных модулей и межмодульных взаимодействий и проектирование внутренней структуры модулей.

При этом происходит расширение системного проекта:

- за счет его уточнения;
- за счет построения моделей автоматизированных рабочих мест, включающих подсхемы информационной модели и функциональные модели, ориентированные на эти подсхемы вплоть до идентификации конкретных сущностей информационной модели;
- за счет построения моделей межмодульных и внутримодульных взаимодействий с использованием техники структурных карт.

7. Последующие этапы

В случае разработки собственной системы последующие этапы являются традиционными: по спецификациям технического проекта осуществляется программирование модулей, их тестирование и отладка и последующая комплексация в автоматизированные рабочие места и в систему в целом. При этом схема интегрированной базы данных, как правило, генерируется автоматически на основании информационной модели.

Настройка существующей системы MRP или ERP осуществляется по следующим этапам:

- наполнение системы фактическими данными;
- построение процедур их обработки;
- интеграция процедур внутри автоматизированных рабочих мест;
- интеграция автоматизированных рабочих мест в систему.

В.3. CASE-технологии - методологическая и инструментальная база консалтинга

За последнее десятилетие сформировалось новое направление в программотехнике - **CASE** (**C**omputer-**A**ided **S**oftware/**S**ystem **E**ngineering). В настоящее время не существует общепринятого определения CASE. Содержание этого понятия обычно определяется перечнем задач, решаемых с помощью CASE, а также совокупностью применяемых методов и средств. Очень грубо, CASE - технология представляет собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных систем программного обеспечения (ПО), поддержанную комплексом взаимоувязанных средств автоматизации. CASE - это инструментарий для системных аналитиков, разработчиков и программистов, заменяющий им бумагу и карандаш на компьютер для автоматизации процесса проектирования и разработки ПО.

К настоящему моменту дисциплина CASE оформилась в самостоятельное наукоемкое направление в программотехнике, повлекшее за собой образование мощной CASE-индустрии, объединившей сотни фирм и компаний различной ориентации. Среди них выделяются компании - разработчики средств анализа и проектирования ПО с широкой сетью дистрибьютерских и дилерских фирм; фирмы - разработчики специальных средств с ориентацией на узкие предметные области или на отдельные этапы жизненного цикла ПО; обучающие фирмы, которые организуют семинары и курсы подготовки специалистов; консультационные фирмы, оказывающие практическую помощь при использовании CASE-пакетов для разработки конкретных приложений; фирмы, специализирующиеся на выпуске периодических журналов и бюллетеней по CASE. Основными покупателями CASE-пакетов за рубежом являются военные организации, центры обработки данных и коммерческие фирмы по разработке ПО.

Существует мнение, что CASE является наиболее перспективным направлением в программотехнике. С этим, естественно, можно и нужно спорить, но то, что CASE - наиболее бурно и интенсивно развиваемое направление, является в настоящее время фактом. Практически ни один серьезный зарубежный программный проект не осуществляется без использования CASE-средств. Известная методология структурного системного анализа SADT (точнее ее подмножество IDEF0) принята в качестве стандарта на разработку ПО Министерством обороны США. Более того, среди менеджеров и руководителей компьютерных фирм считается чуть ли не правилом хорошего тона знать основы SADT и при обсуждении каких-либо вопросов нарисовать простейшую диаграмму, поясняющую суть дела.

CASE позволяет не только создавать "правильные" продукты, но и обеспечить "правильный" процесс их создания. Основная цель CASE состоит в том, чтобы отделить проектирование ПО от его кодирования и последующих этапов разработки, а также скрыть от разработчиков все детали среды разработки и функционирования ПО. Чем больше деятельности будет вынесено в проектирование из кодирования, тем лучше.

При использовании CASE-технологий изменяются все этапы жизненного цикла программной системы, при этом наибольшие изменения касаются этапов анализа и проектирования. В большинстве современных CASE-систем применяются методологии структурного анализа и проектирования, основанные на наглядных диаграммных техниках, при этом для описания модели проектируемой системы используются графы, диаграммы, таблицы и схемы. Такие методологии обеспечивают строгое и наглядное описание проектируемой системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структуру со все большим числом уровней.

Несмотря на то, что структурные методологии зарождались как средства анализа и проектирования ПО, сфера их применений в настоящее время выходит далеко за рамки названной предметной области. Поэтому CASE-технологии успешно применяются для моделирования практически всех предметных областей, однако устойчивое положение они занимают в следующих областях:

- бизнес-анализ (фактически, модели деятельности предприятий “как есть” и ”как должно быть” строятся с применением методов структурного системного анализа и поддерживающих их CASE-средств);
- системный анализ и проектирование (практически любая современная крупная программная система разрабатывается с применением CASE-технологий по крайней мере на этапах анализа и проектирования, что связано с большой сложностью данной проблематики и со стремлением повысить эффективность работ).

Следует отметить, что CASE - не революция в программотехнике, а результат естественного эволюционного развития всей отрасли средств, называемых ранее инструментальными или технологическими. Однако это и не ***C**onfuse **A**rray of **S**oftware that does **E**verything*, существует ряд признаков и свойств, наличие которых позволяет классифицировать некоторый продукт как CASE-средство. Одним из ключевых признаков является поддержка методологий структурного системного анализа и проектирования.

С самого начала CASE-технологии развивались с целью преодоления ограничений при использовании структурных методологий проектирования 60-70-х годов (сложности понимания, большой трудоемкости и стоимости использования, трудности внесения изменений в проектные спецификации и т.д.) за счет их автоматизации и интеграции поддерживающих средств. Таким образом, CASE-технологии, вообще говоря, не могут считаться самостоятельными методологиями, они только развивают структурные методологии и делают более эффективным их применение за счет автоматизации.

Помимо автоматизации структурных методологий и, как следствие, возможности применения современных методов системной и программной инженерии, CASE обладают следующими основными достоинствами:

- улучшают качество создаваемого ПО за счет средств автоматического контроля (прежде всего, контроля проекта);
- позволяют за короткое время создавать прототип будущей системы, что позволяет на ранних этапах оценить ожидаемый результат;
- ускоряют процесс проектирования и разработки;
- освобождают разработчика от рутинной работы, позволяя ему целиком сосредоточиться на творческой части разработки;
- поддерживают развитие и сопровождение разработки;
- поддерживают технологии повторного использования компонент разработки.

Большинство CASE-средств основано на парадигме методология/метод/нотация/ средство. Методология определяет руководящие указания для оценки и выбора проекта разрабатываемого ПО, шаги работы и их последовательность, а также правила распределения и назначения методов. Метод - это систематическая процедура или техника генерации описаний компонент ПО (например, проектирование потоков и структур данных). Нотации предназначены для описания структуры системы, элементов данных, этапов обработки и включают графы, диаграммы, таблицы, блок-схемы, формальные и естественные языки. Средства - инструментарий для поддержки и усиления методов. Эти инструменты поддерживают работу пользователей при создании и редактировании графического проекта в интерактивном режиме, они способствуют организации проекта в виде иерархии уровней абстракции, выполняют проверки соответствия компонентов.

ЧАСТЬ 1

МЕТОДЫ И СРЕДСТВА СТРУКТУРНОГО СИСТЕМНОГО АНАЛИЗА И ПРОЕКТИРОВАНИЯ

В первой части книги, состоящей из семи глав, подробно описываются базовые методы и нотации структурного системного анализа - графического языка “для передачи понимания”, используемого специалистами как бизнес-консалтинга, так и информационно-технологического консалтинга. Рассматриваемый язык основан на потоковых диаграммах, поддерживаемых большинством современных CASE-систем.

Глава 1 является введением в структурный системный анализ. В ней рассматриваются основные задачи этапов анализа требований и проектирования спецификаций системы, принципы структурного анализа, выделяются базовые средства структурного анализа и их взаимосвязи и взаимовлияния.

Глава 2 посвящена наиболее известным и часто используемым средствам функционального моделирования - диаграммам потоков данных. Приводятся основные и вспомогательные объекты диаграмм, рассматривается понятие контекстной диаграммы и детализации процесса, а также декомпозиции потока данных, даются рекомендации для построения функциональной модели в виде иерархии диаграмм потоков данных.

В **главе 3** приводится описание текстовых средств моделирования - словарей данных, предназначенных для описания структуры потоков и хранилищ данных. Вводится понятие словарной статьи, дается нотация, позволяющая формально описать расщепление и объединение (группирование) потоков.

В **главе 4** вводится понятие спецификации процесса (миниспецификации) и описываются наиболее часто применяемые методы ее задания: структурированные естественные языки, таблицы и деревья решений, визуальные языки проектирования. Дается аналитическое сравнение методов.

В **главе 5** описываются базовые средства информационного моделирования - диаграммы "сущность-связь" (при этом рассматриваются две наиболее популярные нотации - Чена и Баркера), приводятся основные этапы построения информационной модели, включая нормализацию.

В **главе 6** рассматривается метод задания спецификаций управления с использованием диаграмм переходов состояний. Вводятся основные объекты диаграмм, предлагаются правила и способы их построения.

Глава 7 посвящена средствам структурного проектирования. Рассматриваются две базовые техники структурного проектирования (Константайна и Джексона), вводятся основные символы соответствующих диаграмм, рассматриваются их достоинства и недостатки. Проводится анализ характеристик хорошего проекта, намечена схема алгоритма преобразования иерархии диаграмм потоков данных в структурные карты.

ГЛАВА 1

ПОНЯТИЕ СТРУКТУРНОГО АНАЛИЗА

1.1. Жизненный цикл программного изделия и его критичные этапы

В основе деятельности по созданию и использованию программного обеспечения (ПО) лежит понятие его жизненного цикла (ЖЦ). ЖЦ является моделью создания и использования ПО, отражающей его различные состояния, начиная с момента возникновения необходимости в данном программном изделии и заканчивая моментом его полного выхода из употребления у всех пользователей.

Традиционно выделяются следующие основные этапы ЖЦ ПО:

- *анализ требований,*
- *проектирование,*
- *кодирование (программирование),*
- *тестирование и отладка,*
- *эксплуатация и сопровождение.*

ЖЦ образуется в соответствии с принципом нисходящего проектирования и, как правило, носит итерационный характер: реализованные этапы, начиная с самых ранних, циклически повторяются в соответствии с изменениями требований и внешних условий, введением ограничений и т.п. На каждом этапе ЖЦ порождается определенный набор документов и технических решений, при этом для каждого этапа исходными являются документы и решения, полученные на предыдущем этапе. Каждый этап завершается верификацией порожденных документов и решений с целью проверки их соответствия исходным.

Существующие модели ЖЦ определяют порядок исполнения этапов в ходе разработки, а также критерии перехода от этапа к этапу. В соответствии с этим наибольшее распространение получили три следующие модели ЖЦ:

1. ***КАСКАДНАЯ МОДЕЛЬ*** (70-80г.г.) - предполагает переход на следующий этап после полного окончания работ по предыдущему этапу.
2. ***ПОЭТАПНАЯ МОДЕЛЬ С ПРОМЕЖУТОЧНЫМ КОНТРОЛЕМ*** (80-85г.г.) - итерационная модель разработки ПО с циклами обратной связи между этапами. Преимущество такой модели заключается в том, что межэтапные корректировки обеспечивают меньшую трудоемкость по сравнению с каскадной моделью; с другой стороны, время жизни каждого из этапов растягивается на весь период разработки.
3. ***СПИРАЛЬНАЯ МОДЕЛЬ*** (86-90г.г.) - делает упор на начальные этапы ЖЦ: анализ требований, проектирование спецификаций, предварительное и детальное проектирование. На этих этапах проверяется и обосновывается реализуемость технических решений путем создания прототипов. Каждый виток спирали соответствует поэтапной модели создания фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы следующего витка спирали.

Таким образом, углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

Специалистами отмечаются следующие преимущества спиральной модели:

- накопление и повторное использование программных средств, моделей и прототипов;
- ориентация на развитие и модификацию ПО в процессе его проектирования;
- анализ риска и издержек в процессе проектирования.

Главная особенность индустрии ПО состоит в концентрации сложности на начальных этапах ЖЦ (анализ, проектирование) при относительно невысокой сложности и трудоемкости последующих этапов. Более того, нерешенные вопросы и ошибки, допущенные на этапах анализа и проектирования, порождают на последующих этапах трудные, часто неразрешимые проблемы и, в конечном счете, приводят к неуспеху всего проекта. Рассмотрим эти этапы более подробно.

АНАЛИЗ ТРЕБОВАНИЙ **является первой фазой разработки ПО, на которой требования заказчика уточняются, формализуются и документируются. Фактически на этом этапе дается ответ на вопрос: "*Что должна делать будущая система?*". Именно здесь лежит ключ к успеху всего проекта. В практике создания больших систем ПО известно немало примеров неудачной реализации проекта именно из-за неполноты и нечеткости определения системных требований.**

Список требований к разрабатываемой системе должен включать:

- совокупность условий, при которых предполагается эксплуатировать будущую систему (аппаратные и программные ресурсы, предоставляемые системе; внешние условия ее функционирования; состав людей и работ, имеющих к ней отношение);
- описание выполняемых системой функций;
- ограничения в процессе разработки (директивные сроки завершения отдельных этапов, имеющиеся ресурсы, организационные процедуры и мероприятия, обеспечивающие защиту информации).

Целью анализа является преобразование общих, неясных знаний о требованиях к будущей системе в точные (по возможности) определения. На этом этапе определяются:

- архитектура системы, ее функции, внешние условия, распределение функций между аппаратурой и ПО;
- интерфейсы и распределение функций между человеком и системой;
- требования к программным и информационным компонентам ПО, необходимые аппаратные ресурсы, требования к БД, физические характеристики компонент ПО, их интерфейсы.

ЭТАП ПРОЕКТИРОВАНИЯ **дает ответ на вопрос: "*Как (каким образом) система будет удовлетворять предъявленным к ней требованиям?*". Задачей этого этапа является исследование структуры системы и логических взаимосвязей ее элементов, причем здесь не рассматриваются вопросы, связанные с реализацией на конкретной платформе. Проектирование определяется как "(итерационный) процесс получения логической модели системы вместе со строго сформулированными целями, поставленными перед нею, а также написания спецификаций физической системы, удовлетворяющей этим требованиям". Обычно этот этап разделяют на два подэтапа:**

- проектирование архитектуры ПО, включающее разработку структуры и интерфейсов компонент, согласование функций и технических требований к компонентам, методам и стандартам проектирования, производство отчетных документов;
- детальное проектирование, включающее разработку спецификаций каждой компоненты, интерфейсов между компонентами, разработку требований к тестам и плана интеграции компонент.

В результате деятельности на этапах анализа и проектирования должен быть получен проект системы, содержащий достаточно информации для реализации системы на его основе в рамках бюджета выделенных ресурсов и времени.

1.2. Идеи, лежащие в основе структурных методов

Структурные методы являются строгой дисциплиной системного анализа и проектирования, т.е. деятельностей, которые в прошлом были печально известны как сложные и перегруженные проблемами.

Методы структурного анализа и проектирования стремятся преодолеть сложность больших систем путем расчленения их на части ("черные ящики") и иерархической организации этих черных ящиков. Выгода в использовании черных ящиков заключается в том, что их пользователю не требуется знать, как они работают, необходимо знать лишь его входы и выходы, а также его назначение (т.е. функцию, которую он выполняет).

В окружающем нас мире черные ящики встречаются в большом количестве. Проиллюстрируем преимущества систем, составленных из них, на примере музыкального центра.

- Конструирование системы черных ящиков существенно упрощается. Намного легче разработать магнитофон или проигрыватель, если не беспокоиться о создании встроенного усилительного блока.
- Облегчается тестирование таких систем. Если появляется плохой звук одной из колонок, можно поменять колонки местами. Если неисправность переместилась с колонкой, то именно она подлежит ремонту; если нет, тогда проблема в усилителе, магнитофоне или местах их соединения.
- Имеется возможность простого реконфигурирования системы черных ящиков. Если колонка неисправна, то Вы можете отправить ее в ремонтную мастерскую, а сами пока продолжать слушать свои записи в моно-режиме.
- Облегчается доступность для понимания и освоения. Можно стать специалистом по магнитофонам без углубленных знаний о колонках.
- Увеличивается удобство при модификации. Вы можете приобрести колонки более высокого качества и более мощный усилитель, но это совсем не означает, что Вам необходим больших размеров проигрыватель.

Таким образом, первым шагом упрощения сложной системы является ее разбиение на черные ящики, при этом такое разбиение должно удовлетворять следующим критериям:

- каждый черный ящик должен реализовывать единственную функцию системы;
- функция каждого черного ящика должна быть легко понимаема независимо от сложности ее реализации (например, в системе управления ракетой может быть черный ящик для расчета места ее приземления: несмотря на сложность алгоритма, функция черного ящика очевидна - "расчет точки приземления");
- связь между черными ящиками должна вводиться только при наличии связи между соответствующими функциями системы (например, в бухгалтерии один черный ящик необходим для расчета общей заработной платы служащего, а другой для расчета налогов - необходима связь между этими черными ящиками: размер заработанной платы требуется для расчета налогов);
- связи между черными ящиками должны быть простыми, насколько это возможно, для обеспечения независимости между ними.

Второй важной идеей, лежащей в основе структурных методов, является идея иерархии. Для понимаемости сложной системы недостаточно разбиения ее на части, необходимо эти части организовать определенным образом, а именно в виде иерархических структур. Все сложные системы Вселенной организованы в иерархии. Да и сама она включает галактики, звездные системы, планеты, …, молекулы, атомы, элементарные частицы. Человек при создании сложных систем также подражает природе. Любая организация имеет директора, заместителей по направлениям, иерархию руководителей подразделений, рядовых служащих.

Наконец, третий момент: структурные методы широко используют графические нотации, также служащие для облегчения понимания сложных систем. Известно, что “одна картинка стоит тысячи слов”. На рис.1.1 изображен черноволосый мужчина, одетый в серое двубортное пальто. Мужчина держит в левой руке дипломат и т.д. Вообще говоря, нет необходимости комментировать это: читатель впитывает вышеизложенное описание с первого взгляда.

Однако можно добавить дополнительные подробности, которые не видны из рисунка. Например, мужчину зовут Борис Борисович, ему 45 лет. Структурные методы также позволяют дополнить “картинки” любой информацией, которая не может быть отражена при использовании соответствующей графической нотации.

О преимуществе “картинок” свидетельствует и следующий факт. Едва ли найдется человек, не читавший рассказ А.П.Чехова “Толстый и тонкий”. И тем не менее, практически никто не обращал внимание на любопытное противоречие в тексте рассказа: “Нафанаил немного подумал и снял ***шапку*** … Нафанаил шаркнул ногой и уронил ***фуражку***”. С другой стороны, имея иллюстрации этих двух сцен, легко обнаружить несоответствие.

1.3. Принципы структурного анализа

Анализ требований разрабатываемой системы является важнейшим среди всех этапов ЖЦ. Он оказывает существенное влияние на все последующие этапы, являясь в то же время наименее изученным и понятным процессом. На этом этапе, во-первых, необходимо понять, что предполагается сделать, а во-вторых, задокументировать это, т.к. если требования не зафиксированы и не сделаны доступными для участников проекта, то они вроде бы и не существуют. При этом язык, на котором формулируются требования, должен быть достаточно прост и понятен заказчику.

Во многих аспектах системный анализ является наиболее трудной частью разработки. Нижеследующие проблемы, с которыми сталкивается системный аналитик, взаимосвязаны (и это является одной из главных причин их трудноразрешимости):

- аналитику сложно получить исчерпывающую информацию для оценки требований к системе с точки зрения заказчика;
- заказчик, в свою очередь, не имеет достаточной информации о проблеме обработки данных для того, чтобы судить, что является выполнимым, а что нет;
- аналитик сталкивается с чрезмерным количеством подробных сведений как о предметной области, так и о новой системе;
- спецификация системы из-за объема и технических терминов часто непонятна для заказчика;
- в случае понятности спецификации для заказчика, она будет являться недостаточной для проектировщиков и программистов, создающих систему.

Конечно, применение известных аналитических методов снимает некоторые из перечисленных проблем анализа, однако эти проблемы могут быть существенно облегчены за счет применения современных структурных методов, среди которых центральное место занимают методологии структурного анализа.

Структурным анализом принято называть метод исследования системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структуру со все большим числом уровней. Для таких методов характерно разбиение на уровни абстракции с ограничением числа элементов на каждом из уровней (обычно от 3 до 6-7); ограниченный контекст, включающий лишь существенные на каждом уровне детали; дуальность данных и операций над ними; использование строгих формальных правил записи; последовательное приближение к конечному результату.

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, легких для понимания и решения. Второй принцип в дополнение к тому, что легче понимать проблему если она разбита на части, декларирует, что устройство этих частей также существенно для понимания. Понимание проблемы резко облегчается при организации ее частей в древовидные иерархические структуры, т.е. система может быть понята и построена по уровням, каждый из которых добавляет новые детали.

Выделение двух базовых принципов инженерии программного обеспечения вовсе не означает, что остальные принципы являются второстепенными, игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к неуспеху всего проекта). Отметим основные из таких принципов.

1. Принцип абстрагирования - заключается в выделении существенных с некоторых позиций аспектов системы и отвлечение 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 на присутствие 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 разрабатываемого ПО и используемых при этом методологий. Руководствуясь всеми принципами в комплексе, можно понять на более ранних стадиях разработки, что будет представлять из себя создаваемая система, обнаружить промахи и недоработки, что, в свою очередь, облегчит работы на последующих этапах ЖЦ и понизит стоимость разработки.

1.4. Средства структурного анализа и их взаимоотношения

Прежде чем подробно рассмотреть каждое из основных инструментальных средств структурного анализа, необходимо обсудить их в общем виде и продемонстрировать их взаимосвязи.

Для целей моделирования систем вообще, и структурного анализа в частности, используются три группы средств, иллюстрирующих:

- функции, которые система должна выполнять;
- отношения между данными;
- зависящее от времени поведение системы (аспекты реального времени).

Среди всего многообразия средств решения данных задач в методологиях структурного анализа наиболее часто и эффективно применяемыми являются следующие:

- **DFD** (**D**ata **F**low **D**iagrams) - **диаграммы потоков данных (глава 2) совместно со словарями данных (глава 3) и спецификациями процессов или миниспецификациями (глава 4);**
- **ERD** (**E**ntity-**R**elationship **D**iagrams) - **диаграммы "сущность-связь" (глава 5);**
- **STD** (**S**tate **T**ransition **D**iagrams) - **диаграммы переходов состояний (глава 6).**

Все они содержат графические и текстовые средства моделирования: первые - для удобства демонстрирования основных компонент модели, вторые - для обеспечения точного определения ее компонент и связей.

Логическая DFD показывает внешние по отношению к системе источники и стоки (адресаты) данных, идентифицирует логические функции (процессы) и группы элементов данных, связывающие одну функцию с другой (потоки), а также идентифицирует хранилища (накопители) данных, к которым осуществляется доступ. Структуры потоков данных и определения их компонент хранятся и анализируются в словаре данных. Каждая логическая функция (процесс) может быть детализирована с помощью DFD нижнего уровня; когда дальнейшая детализация перестает быть полезной, переходят к выражению логики функции при помощи спецификации процесса (миниспецификации). Содержимое каждого хранилища также сохраняют в словаре данных, модель данных хранилища раскрывается с помощью ERD. В случае наличия реального времени DFD дополняется средствами описания зависящего от времени поведения системы, раскрывающимися с помощью STD. Эти связи показаны на рис. 1.2.

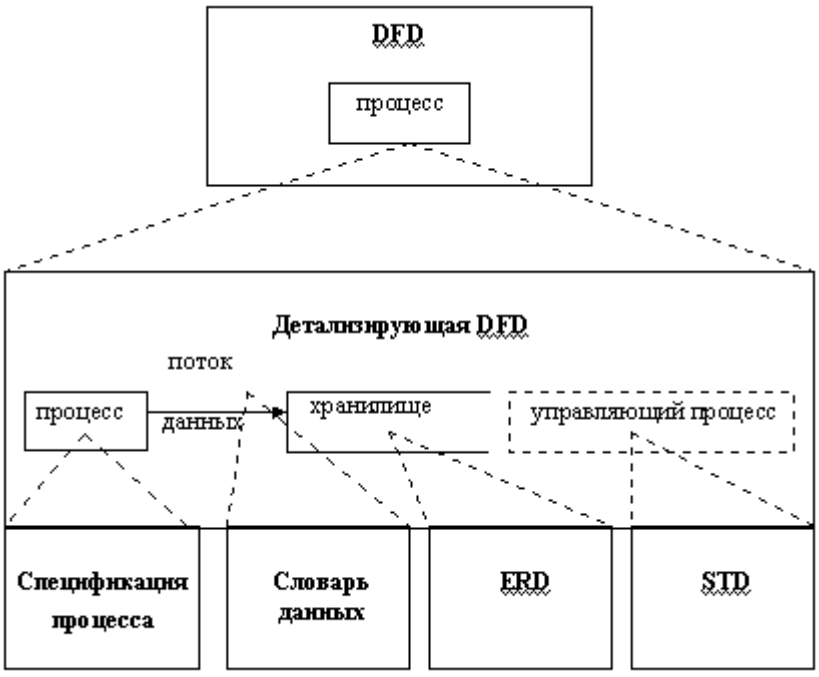


Рис. 1.2. Компоненты логической модели

Перечисленные средства дают полное описание системы независимо от того, является ли она существующей или разрабатываемой с нуля. Таким образом строится логическая функциональная спецификация - подробное описание того, что должна делать система, освобожденное насколько это возможно от рассмотрения путей реализации. Это дает проектировщику четкое представление о конечных результатах, которые следует достигать.

ГЛАВА 2

ДИАГРАММЫ ПОТОКОВ ДАННЫХ

Диаграммы потоков данных (DFD) являются основным средством моделирования функциональных требований проектируемой системы. С их помощью эти требования разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств - продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Диаграммы потоков данных известны очень давно. В фольклоре упоминается следующий пример использования DFD для реорганизации переполненного клерками офиса, относящийся к 20-м годам. Осуществлявший реорганизацию консультант обозначил кружком каждого клерка, а стрелкой - каждый документ, передаваемый между ними. Используя такую диаграмму, он предложил схему реорганизации, в соответствии с которой двое клерков, обменивающиеся множеством документов, были посажены рядом, а клерки с малым взаимодействием были посажены на большом расстоянии. Так родилась первая модель, представляющая собой потоковую диаграмму - предвестника DFD.

Для изображения DFD традиционно используются две различные нотации: Йодана (Yourdon) и Гейна-Сарсона (Gane-Sarson). Далее при построении примеров будет использоваться нотация Йодана, все исключения будут предварительно оговариваться.

2.1. Основные символы

Основные символы DFD изображены на рис.2.1. Опишем их назначение. На диаграммах функциональные требования представляются с помощью процессов и хранилищ, связанных потоками данных.

ПОТОКИ ДАННЫХ **являются механизмами, использующимися для моделирования передачи информации (или даже физических компонент) из одной части системы в другую. Важность этого объекта очевидна: он дает название целому инструменту. Потоки на диаграммах обычно изображаются именованными стрелками, ориентация которых указывает направление движения информации.**

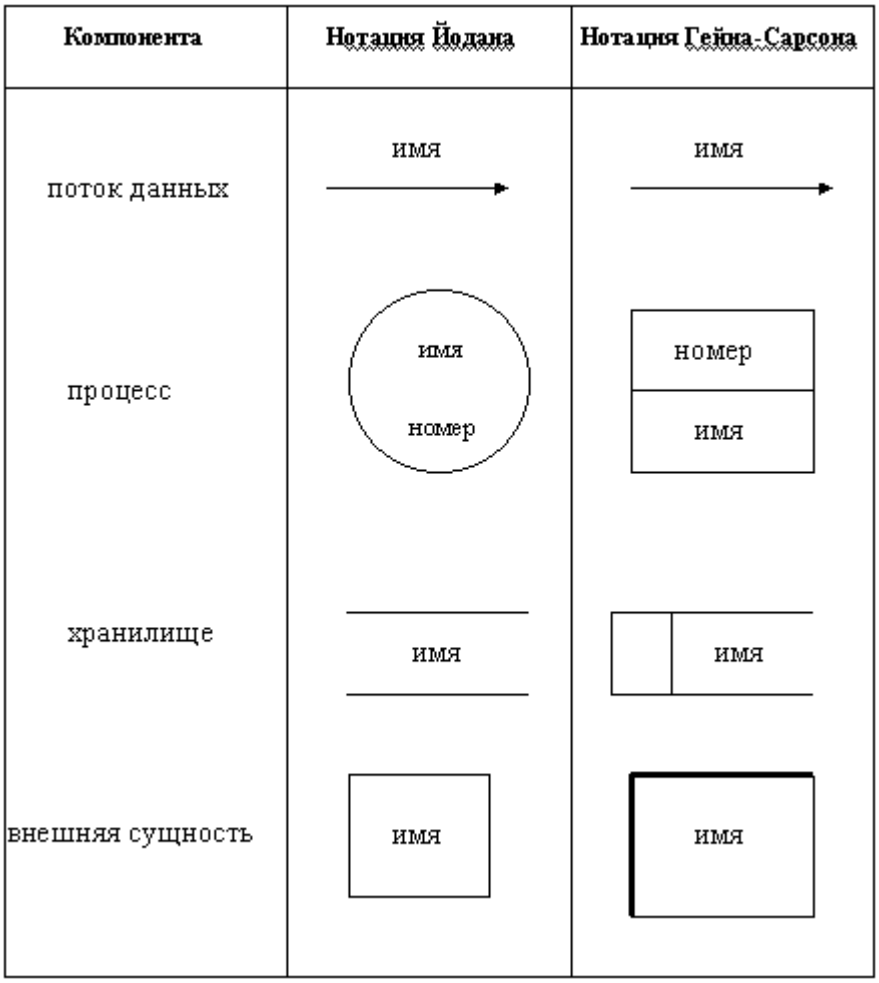


Рис. 2.1. Основные символы диаграммы потоков данных

Иногда информация может двигаться в одном направлении, обрабатываться и возвращаться назад в ее источник. Такая ситуация может моделироваться либо двумя различными потоками, либо одним - двунаправленным.

Назначение ***ПРОЦЕССА*** состоит в продуцировании выходных потоков из входных в соответствии с действием, задаваемым именем процесса. Это имя должно содержать глагол в неопределенной форме с последующим дополнением (например, *ВЫЧИСЛИТЬ МАКСИМАЛЬНУЮ ВЫСОТУ*). Кроме того, каждый процесс должен иметь уникальный номер для ссылок на него внутри диаграммы. Этот номер может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.

ХРАНИЛИЩЕ (НАКОПИТЕЛЬ) ДАННЫХ позволяет на определенных участках определять данные, которые будут сохраняться в памяти между процессами. Фактически хранилище представляет "срезы" потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее определения, при этом данные могут выбираться в любом порядке. Имя хранилища должно идентифицировать его содержимое и быть существительным. В случае, когда поток данных входит или выходит в/из хранилища, и его структура соответствует структуре хранилища, он должен иметь то же самое имя, которое нет необходимости отражать на диаграмме.

ВНЕШНЯЯ СУЩНОСТЬ **(или *ТЕРМИНАТОР*)** представляет сущность вне контекста системы, являющуюся источником или приемником системных данных. Ее имя должно содержать существительное, например, *СКЛАД ТОВАРОВ*. Предполагается, что объекты, представленные такими узлами, не должны участвовать ни в какой обработке.

2.2. Контекстная диаграмма и детализация процессов

Декомпозиция DFD осуществляется на основе процессов: каждый процесс может раскрываться с помощью DFD нижнего уровня.

Важную специфическую роль в модели играет специальный вид DFD - **контекстная диаграмма,** моделирующая систему наиболее общим образом. Контекстная диаграмма отражает интерфейс системы с внешним миром, а именно, информационные потоки между системой и внешними сущностями, с которыми она должна быть связана. Она идентифицирует эти внешние сущности, а также, как правило, единственный процесс, отражающий главную цель или природу системы насколько это возможно. И хотя контекстная диаграмма выглядит тривиальной, несомненная ее полезность заключается в том, что она устанавливает границы анализируемой системы. Каждый проект должен иметь ровно одну контекстную диаграмму, при этом нет необходимости в нумерации единственного ее процесса.

DFD первого уровня строится как декомпозиция процесса, который присутствует на контекстной диаграмме.

Построенная диаграмма первого уровня также имеет множество процессов, которые в свою очередь могут быть декомпозированы в DFD нижнего уровня. Таким образом строится иерархия DFD с контекстной диаграммой в корне дерева. Этот процесс декомпозиции продолжается до тех пор, пока процессы могут быть эффективно описаны с помощью коротких (до одной страницы) миниспецификаций обработки (спецификаций процессов).

При таком построении иерархии DFD каждый процесс более низкого уровня необходимо соотнести с процессом верхнего уровня. Обычно для этой цели используются структурированные номера процессов. Так, например, если мы детализируем процесс номер 2 на диаграмме первого уровня, раскрывая его с помощью DFD, содержащей три процесса, то их номера будут иметь следующий вид: 2.1, 2.2 и 2.3. При необходимости можно перейти на следующий уровень, т.е. для процесса 2.2 получим 2.2.1, 2.2.2. и т.д.

2.3. Декомпозиция данных и соответствующие расширения диаграмм потоков данных

Индивидуальные данные в системе часто являются независимыми. Однако иногда необходимо иметь дело с несколькими независимыми данными одновременно. Например, в системе имеются потоки *ЯБЛОКИ, АПЕЛЬСИНЫ* и *ГРУШИ*. Эти потоки могут быть сгруппированы с помощью введения нового потока *ФРУКТЫ*. Для этого необходимо определить формально поток *ФРУКТЫ* как состоящий из нескольких элементов-потомков. Такое определение задается с помощью **формы Бэкуса-Наура (БНФ)** в словаре данных (см. главу 3). В свою очередь поток *ФРУКТЫ* сам может содержаться в потоке-предке *ЕДА* вместе с потоками *ОВОЩИ, МЯСО* и др. Такие потоки, объединяющие несколько потоков, получили название **групповых.**

Обратная операция, расщепление потоков на подпотоки, осуществляется с использованием группового узла (рис. 2.2), позволяющего расщепить поток на любое число подпотоков. При расщеплении также необходимо формально определить подпотоки в словаре данных (с помощью БНФ).

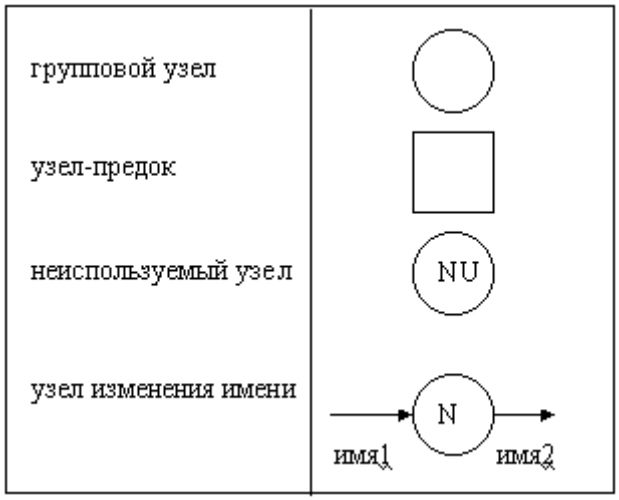


Рис 2.2. Расширения диаграммы потоков данных

Аналогичным образом осуществляется и декомпозиция потоков через границы диаграмм, позволяющая упростить детализирующую DFD. Пусть имеется поток *ФРУКТЫ*, входящий в детализируемый процесс. На детализирующей этот процесс диаграмме потока *ФРУКТЫ* может не быть вовсе, но вместо него могут быть потоки *ЯБЛОКИ* и *АПЕЛЬСИНЫ* (как будто бы они переданы из детализируемого процесса). В этом случае должно существовать БНФ-определение потока *ФРУКТЫ*, состоящего из подпотоков *ЯБЛОКИ* и *АПЕЛЬСИНЫ*, для целей балансирования.

Применение этих операций над данными позволяет обеспечить структуризацию данных, увеличивает наглядность и читабельность диаграмм.

Для обеспечения декомпозиции данных и некоторых других сервисных возможностей к DFD добавляются следующие типы объектов:

1. ***ГРУППОВОЙ УЗЕЛ.*** Предназначен для расщепления и объединения потоков. В некоторых случаях может отсутствовать (т.е. фактически вырождаться в точку слияния/расщепления потоков на диаграмме).
2. ***УЗЕЛ-ПРЕДОК.*** Позволяет увязывать входящие и выходящие потоки между детализируемым процессом и детализирующей DFD.
3. ***НЕИСПОЛЬЗУЕМЫЙ УЗЕЛ.*** Применяется в ситуации, когда декомпозиция данных производится в групповом узле, при этом требуются не все элементы входящего в узел потока.
4. ***УЗЕЛ ИЗМЕНЕНИЯ ИМЕНИ.*** Позволяет неоднозначно именовать потоки, при этом их содержимое эквивалентно. Например, если при проектировании разных частей системы один и тот же фрагмент данных получил различные имена, то эквивалентность соответствующих потоков данных обеспечивается узлом изменения имени. При этом один из потоков данных является входным для данного узла, а другой - выходным.
5. **Текст** в свободном формате в любом месте диаграммы.

Возможный способ изображения этих узлов приведен на рис. 2.2.

2.4. Построение модели

Главная цель построения иерархического множества DFD заключается в том, чтобы сделать требования ясными и понятными на каждом уровне детализации, а также разбить эти требования на части с точно определенными отношениями между ними. Для достижения этого целесообразно пользоваться следующими рекомендациями:

1. Размещать на каждой диаграмме от 3 до 6-7 процессов. Верхняя граница соответствует человеческим возможностям одновременного восприятия и понимания структуры сложной системы с множеством внутренних связей, нижняя граница выбрана по соображениям

здравого смысла: нет необходимости детализировать процесс диаграммой, содержащей всего один или два процесса.
2. Не загромождать диаграммы несущественными на данном уровне деталями.
3. Декомпозицию потоков данных осуществлять параллельно с декомпозицией процессов; эти две работы должны выполняться одновременно, а не одна после завершения другой.
4. Выбирать ясные, отражающие суть дела, имена процессов и потоков для улучшения понимаемости диаграмм, при этом стараться не использовать аббревиатуры.
5. Однократно определять функционально идентичные процессы на самом верхнем уровне, где такой процесс необходим, и ссылаться к нему на нижних уровнях.
6. Пользоваться простейшими диаграммными техниками: если что-либо возможно описать с помощью DFD, то это и необходимо делать, а не использовать для описания более сложные объекты.
7. Отделять управляющие структуры от обрабатывающих структур (т.е. процессов), локализовать управляющие структуры.

В соответствии с этими рекомендациями процесс построения модели разбивается на следующие этапы:

1. Расчленение множества требований и организация их в основные функциональные группы.
2. Идентификация внешних объектов, с которыми система должна быть связана.
3. Идентификация основных видов информации, циркулирующей между системой и внешними объектами.
4. Предварительная разработка контекстной диаграммы, на которой основные функциональные группы представляются процессами, внешние объекты - внешними сущностями, основные виды информации - потоками данных между процессами и внешними сущностями.
5. Изучение предварительной контекстной диаграммы и внесение в нее изменений по результатам ответов на возникающие при этом изучении вопросы по всем ее частям.
6. Построение контекстной диаграммы путем объединения всех процессов предварительной диаграммы в один процесс, а также группирования потоков.
7. Формирование DFD первого уровня на базе процессов предварительной контекстной диаграммы.
8. Проверка основных требований по DFD первого уровня.
9. Декомпозиция каждого процесса текущей DFD с помощью детализирующей диаграммы или спецификации процесса.
10. Проверка основных требований по DFD соответствующего уровня.
11. Добавление определений новых потоков в словарь данных при каждом их появлении на диаграммах.
12. Параллельное (с процессом декомпозиции) изучение требований (в том числе и вновь поступающих), разбиение их на элементарные и идентификация процессов или спецификаций процессов, соответствующих этим требованиям.
13. После построения двух-трех уровней проведение ревизии с целью проверки корректности и улучшения понимаемости модели.
14. Построение спецификации процесса (а не простейшей диаграммы) в случае, если некоторую функцию сложно или невозможно выразить комбинацией процессов.

ГЛАВА 3

СЛОВАРЬ ДАННЫХ

Диаграммы потоков данных обеспечивают удобное описание функционирования компонент системы, но не снабжают аналитика средствами описания деталей этих компонент, а именно, какая информация преобразуется процессами и как она преобразуется. Для решения первой из перечисленных задач предназначены текстовые средства моделирования, служащие для описания структуры преобразуемой информации и получившие название словарей данных.

Словарь данных представляет собой определенным образом организованный список всех элементов данных системы с их точными определениями, что дает возможность различным категориям пользователей (от системного аналитика до программиста) иметь общее понимание всех входных и выходных потоков и компонент хранилищ. Определения элементов данных в словаре осуществляются следующими видами описаний:

- описанием значений потоков и хранилищ, изображенных на DFD;
- описанием композиции агрегатов данных, движущихся вдоль потоков, т.е. комплексных данных, которые могут расчленяться на элементарные символы (например, *АДРЕС ПОКУПАТЕЛЯ* содержит *ПОЧТОВЫЙ ИНДЕКС, ГОРОД, УЛИЦУ* и т.д.);
- описанием композиции групповых данных в хранилище;
- специфицированием значений и областей действия элементарных фрагментов информации в потоках данных и хранилищах;
- описанием деталей отношений между хранилищами.

3.1. Содержимое словаря данных

Для каждого потока данных в словаре необходимо хранить имя потока, его тип и атрибуты. Информация по каждому потоку состоит из ряда словарных статей, каждая из которых начинается с ключевого слова - заголовка соответствующей статьи, которому предшествует символ “**@**”.

По типу потока в словаре содержится информация, идентифицирующая:

- простые (элементарные) или групповые (комплексные) потоки;
- внутренние (существующие только внутри системы) или внешние (связывающие систему с другими системами) потоки;
- потоки данных или потоки управления;
- непрерывные (принимающие любые значения в пределах определенного диапазона) или дискретные (принимающие определенные значения) потоки.

Атрибуты потока данных включают:

- имена-синонимы потока данных в соответствии с узлами изменения имени;
- БНФ-определение в случае группового потока (см. 3.2);
- единицы измерения потока;
- диапазон значений для непрерывного потока, типичное его значение и информацию по обработке экстремальных значений;
- список значений и их смысл для дискретного потока;
- список номеров диаграмм различных типов, в которых поток встречается;
- список потоков, в которые данный поток входит (как элемент БНФ-определения);
- комментарий, включающий дополнительную информацию (например о цели введения данного потока).

3.2. БНФ - нотация

БНФ-нотация позволяет формально описать расщепление/ объединение потоков. Поток может расщепляться на собственные отдельные ветви, на компоненты потока-предка или на то и другое одновременно. При расщеплении/объединении потока существенно, чтобы каждый компонент потока-предка являлся именованным. Если поток расщепляется на подпотоки, необходимо, чтобы все подпотоки являлись компонентами потока-предка. И наоборот, при объединении потоков каждый компонент потока-предка должен по крайней мере однажды встречаться среди подпотоков. Отметим, что при объединении подпотоков нет необходимости осуществлять исключение общих компонент, а при расщеплении подпотоки могут иметь такие общие (одинаковые) компоненты.

Важно понимать, что точные определения потоков содержатся в словаре данных, а не на диаграммах. Например, на диаграмме может иметься групповой узел с входным потоком *X* и

выходными подпотоками *Y* и *Z*. Однако это вовсе не означает, что соответствующее определение в словаре данных обязательно должно быть *X=Y+Z*. Это определение может быть следующим:

$$X=A+B+C;\ Y=A+B;\ Z=B+C$$

Такие определения хранятся в словаре данных в так называемой **БНФ-статье**. БНФ-статья используется для описания компонент данных в потоках данных и в хранилищах. Ее синтаксис имеет вид:

@БНФ *= <простой оператор> ! <БНФ-выражение> ,*

где *<простой оператор>* есть текстовое описание, заключенное в "/", а *<БНФ-выражение>* есть выражение в форме Бэкуса-Наура, допускающее следующие операции отношений:

- **=** - означает "композиция из",
- **+** - означает "*И*",
- **[!]** - означает "*ИЛИ*",
- **( )** - означает, что компонент в скобках не обязателен,
- **{ }** - означает итерацию компонента в скобках,
- **" "** - означает литерал.

Итерационные скобки могут иметь нижний и верхний предел, например:

- *3{болт}7 - от 3 до 7 итераций*
- *1{болт} - 1 и более итераций*
- *{шайба}3 - не более 3 итераций*

БНФ-выражение может содержать произвольные комбинации операций:

- ***@БНФ*** *= [винт ! болт + 2{гайка}2 + (прокладка) ! клей]*

Ниже приведен пример описания потока данных с помощью БНФ:

- ***@ИМЯ*** *= ВОСЬМЕРИЧНАЯ ЦИФРА*
- ***@ТИП*** *= дискретный поток*
- ***@БНФ*** *= ["0"!"1"!"2"!"3"!"4"!"5"!"6"!"7"]*

Посмотрим, как некоторые потоки, присутствующие на вышеприведенных диаграммах потоков данных, представляются в словаре данных.

- ***@ИМЯ*** *= ВВЕДЕННАЯ КРЕДИТНАЯ КАРТА*
- ***@ТИП*** *= управляющий поток*
- ***@БНФ*** *= /указывает, что кредитная карта введена/*
- ***@ИМЯ*** *= ДАННЫЕ КРЕДИТНОЙ КАРТЫ*
- ***@ТИП*** *= дискретный поток*
- ***@БНФ*** *= ПАРОЛЬ + ДЕТАЛИ КЛИЕНТА + ЛИМИТ ДЕНЕГ*
- ***@ИМЯ*** *= ДАННЫЕ ПО БАЛАНСУ*
- ***@ТИП*** *= дискретный поток*
- ***@БНФ*** *= /текущий баланс счета клиента/*
- ***@ЕДИНИЦА ИЗМЕРЕНИЯ*** *= доллар*
- ***@ДИАПАЗОН*** *= +/- 100000*
- ***@ТОЧНОСТЬ*** *= .01*
- ***@ИМЯ*** *= ДЕНЬГИ*
- ***@ТИП*** *= дискретный поток*
- ***@БНФ*** *= /деньги, выдаваемые клиенту/*
- ***@ЕДИНИЦА ИЗМЕРЕНИЯ*** *= доллар*

- ***@НОРМА*** *= 5..1000*
- ***@КОММЕНТАРИЙ*** *Сумма выдаваемых денег должна делиться на 5*
- ***@ИМЯ*** *= ПРОТОКОЛ ОБСЛУЖИВАНИЯ*
- ***@ТИП*** *= дискретный поток*
- ***@БНФ*** *= (ОБРАБОТАННАЯ ДОКУМЕНТАЦИЯ)*
- *+ (ДЕНЕЖНАЯ СУММА)*
- *+ (ДАННЫЕ ПО ИСТОРИИ ЗАПРОСА)*

ГЛАВА 4

МЕТОДЫ ЗАДАНИЯ СПЕЦИФИКАЦИЙ ПРОЦЕССОВ

Спецификация процесса (СП) используется для описания функционирования процесса в случае отсутствия необходимости детализировать его с помощью DFD (т.е. если он достаточно невелик, и его описание может занимать до одной страницы текста). Фактически СП представляют собой алгоритмы описания задач, выполняемых процессами: множество всех СП является полной спецификацией системы. СП содержат номер и/или имя процесса, списки входных и выходных

данных и тело (описание) процесса, являющееся спецификацией алгоритма или операции, трансформирующей входные потоки данных в выходные. Известно большое число разнообразных методов, позволяющих задать тело процесса, соответствующий язык может варьироваться от **структурированного естественного языка** или псевдокода до **визуальных языков проектирования** (типа **FLOW-форм** и **диаграмм Насси-Шнейдермана**) и формальных компьютерных языков.

Независимо от используемой нотации спецификация процесса должна начинаться с ключевого слова (например, ***@СПЕЦПРОЦ***). Требуемые входные и выходные данные должны быть специфицированы следующим образом:

@ВХОД = <имя символа данных>
@ВЫХОД = <имя символа данных>
@ВХОДВЫХОД = <имя символа данных>,
где <имя символа данных> - соответствующее имя из словаря данных.

Эти ключевые слова должны использоваться перед определением СП, например,

@ВХОД = СЛОВА ПАМЯТИ
@ВЫХОД = ХРАНИМЫЕ ЗНАЧЕНИЯ
@СПЕЦПРОЦ
Для всех СЛОВ ПАМЯТИ выполнить:
Распечатать ХРАНИМЫЕ ЗНАЧЕНИЯ
@

Ситуация, когда символ данных является одновременно входным и выходным, может быть описана двумя способами: либо символ описывается два раза с помощью ***@ВХОД*** и ***@ВЫХОД***, либо один раз с помощью ***@ВХОДВЫХОД***.

Иногда в СП задаются **пред-** и **пост-условия** выполнения данного процесса. В пред-условии записываются объекты, значения которых должны быть истинны перед началом выполнения процесса, что обеспечивает определенные гарантии безопасности для пользователя. Аналогично, в случае наличия пост-условия гарантируется, что значения всех входящих в него объектов будут истинны при завершении процесса.

Спецификации должны удовлетворять следующим требованиям:

- для каждого процесса нижнего уровня должна существовать одна и только одна спецификация;
- спецификация должна определять способ преобразования входных потоков в выходные;
- нет необходимости (на данном этапе) определять метод реализации этого преобразования;
- спецификация должна стремиться к ограничению избыточности - не следует переопределять то, что уже было определено на диаграмме или в словаре даннных;
- набор конструкций для построения спецификации должен быть простым и стандартным.

Ниже рассматриваются некоторые наиболее часто используемые методы задания спецификаций процессов.

4.1. Структурированный естественный язык

Структурированный естественный язык применяется для читабельного, строгого описания спецификаций процессов. Он является разумной комбинацией строгости языка программирования и читабельности естественного языка и состоит из подмножества слов, организованных в определенные логические структуры, арифметических выражений и диаграмм.

В состав языка входят следующие основные символы:

- глаголы, ориентированные на действие и применяемые к объектам;
- термины, определенные на любой стадии проекта ПО (например, задачи, процедуры, символы данных и т.п.);
- предлоги и союзы, используемые в логических отношениях;
- общеупотребительные математические, физические и технические термины;
- арифметические уравнения;
- таблицы, диаграммы, графы и т.п.;
- комментарии.

Управляющие структуры языка имеют один вход и один выход. К ним относятся:

1) последовательная конструкция:
ВЫПОЛНИТЬ *функция1*
ВЫПОЛНИТЬ *функция2*
ВЫПОЛНИТЬ *функция3*

2) конструкция выбора:
ЕСЛИ *<условие>* ***ТО***
ВЫПОЛНИТЬ *функция1*
ИНАЧЕ
ВЫПОЛНИТЬ *функция2*
КОНЕЦЕСЛИ

3) итерация:
ДЛЯ *<условие>*
ВЫПОЛНИТЬ *функция*
КОНЕЦДЛЯ
или
ПОКА *<условие>*
ВЫПОЛНИТЬ *функция*
КОНЕЦПОКА

При использовании структурированного естественного языка приняты следующие соглашения:

1. Логика процесса выражается в виде комбинации последовательных конструкций, конструкций выбора и итераций.
2. Ключевые слова ***ЕСЛИ, ВЫПОЛНИТЬ, ИНАЧЕ*** и т.д. должны быть написаны заглавными буквами.
3. Слова или фразы, определенные в словаре данных, должны быть написаны заглавными буквами.
4. Глаголы должны быть активными, недвусмысленными и ориентированными на целевое действие (заполнить, вычислить, извлечь, а не модернизировать, обработать).
5. Логика процесса должна быть выражена четко и недвусмысленно.

Ниже приведен пример спецификации процесса 1 (ПОЛУЧИТЬ ПАРОЛЬ) для диаграммы, изображенной на рис. 2.8.

@ВХОД *= ВВЕДЕННЫЙ ПАРОЛЬ*
@ВХОД *= ПАРОЛЬ*
@ВЫХОД *= СООБЩЕНИЕ*
@ВЫХОД *= КОРРЕКТНЫЙ ПАРОЛЬ*
@СПЕЦПРОЦ *1.1 ПОЛУЧИТЬ ПАРОЛЬ*
ВЫПОЛНИТЬ *выдать СООБЩЕНИЕ клиенту,*
запрашивающее ввод пароля
принять ВВЕДЕННЫЙ ПАРОЛЬ
ДОТЕХПОРПОКА *ВВЕДЕННЫЙ ПАРОЛЬ = ПАРОЛЬ*
или были сделаны три попытки ввода

КОНЕЦВЫПОЛНИТЬ
ВЫПОЛНИТЬ *установить флаг КОРРЕКТНЫЙ*
ПАРОЛЬ в случае равенства
@ *КОНЕЦ СПЕЦИФИКАЦИИ ПРОЦЕССА 1.1*

4.2. Таблицы и деревья решений

Структурированный естественный язык неприемлем для некоторых типов преобразований. Например, если действие зависит от нескольких переменных, которые в совокупности могут продуцировать большое число комбинаций, то его описание будет слишком запутанным и с большим числом уровней вложенности. Для описания подобных действий традиционно используются таблицы и деревья решений.

Проектирование спецификаций процессов с помощью **таблиц решений** (ТР) заключается в задании матрицы, отображающей множество входных **условий** в множество **действий.**

ТР состоит из двух частей. Верхняя часть таблицы используется для определения условий. Обычно условие является ***ЕСЛИ***-частью оператора ***ЕСЛИ-ТО*** и требует ответа "да-нет". Однако иногда в условии может присутствовать и ограниченное множество значений, например, *ЯВЛЯЕТСЯ ЛИ ДЛИНА СТРОКИ БОЛЬШЕЙ, МЕНЬШЕЙ ИЛИ РАВНОЙ ГРАНИЧНОМУ ЗНАЧЕНИЮ?*

Нижняя часть ТР используется для определения действий, т.е. ***ТО***-части оператора ***ЕСЛИ-ТО***. Так, в конструкции

ЕСЛИ *ИДЕТ ДОЖДЬ* ***ТО*** *РАСКРЫТЬ ЗОНТ*
ИДЕТ ДОЖДЬ является условием, а *РАСКРЫТЬ ЗОНТ* - действием.

Левая часть ТР содержит собственно описание условий и действий, а в правой части перечисляются все возможные комбинации условий и, соответственно, указывается, какие конкретно действия и в какой последовательности выполняются, когда определенная комбинация условий имеет место.

Поясним вышесказанное на примере спецификации процесса выбора символов из входного потока. При выборе символов необходимо руководствоваться следующими правилами:

1) если очередной символ является управляющим, то подать звуковой сигнал и вернуть код ошибки;
2) если буфер формируемой строки заполнен, то подать звуковой сигнал и вернуть код ошибки;
3) если очередной символ не находится в заданном диапазоне, то подать звуковой сигнал и вернуть код ошибки;
4) иначе поместить символ в буфер, увеличить значение счетчика выбранных символов и вернуть новое значение счетчика.
Таблица решений для данного примера выглядит следующим образом (таблица 4.1):

Таблица 4.1

	УСЛОВИЯ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C1	isctrl(c)	Д	Д	Д	Д	Н	Н	Н	Н
C2	I > max_lenght	Д	Д	Н	Н	Д	Д	Н	Н
C3	out_of_range(c)	Д	Н	Д	Н	Д	Н	Д	Н
	ДЕЙСТВИЯ								
D1	beep()	1	1	1	1	1	1	1	
D2	return(ERROR)	2	2	2	2	2	2	2	
D3	return(++i)								2

D4	putchar(c)								1

Заметим, что если выполняется условие *C1*, то нет необходимости в проверке условий *C2* и *C3*. Поэтому комбинации условий 1, 2, 3, 4 могут быть заменены обобщающей комбинацией (*Д,*-,-), где "-" означает любую из возможных альтернатив (в данном случае, *Д* или *Н*). Аналогично, комбинации условий 5 и 6 могут быть заменены обобщающей комбинацией (*Н,Д,*-). Редуцированная таким образом таблица решений будет иметь следующий вид (таблица 4.2):

Таблица 4.2

	УСЛОВИЯ	**1**	**2**	**3**	**4**
C1	isctrl(c)	Д	Н	Н	Н
C2	I > max_lenght	-	Д	Н	Н
C3	out_of_range(c)	-	-	Д	Н
	ДЕЙСТВИЯ				
D1	beep()	1	1	1	
D2	return(ERROR)	2	2	2	
D3	return(++i)				2
D4	putchar(c)				1

Отметим, что на основе таблицы решений легко осуществляется автоматическая кодогенерация. Для вышеприведенного примера соответствующий код может выглядеть следующим образом:

```
IF (isctrl(c)) { beep(); return(ERROR) }
  ELSE {
  IF (i>max_length) { beep(); return(ERROR) }
  ELSE {
  IF (out_of_range(c)) { beep(); return(ERROR) }
  ELSE { putchar(c); return(++i) }
  }
  }
```

Построение ТР рекомендуется осуществлять по следующим шагам:

1. Идентифицировать все условия (или переменные) в спецификации. Идентифицировать все значения, которые каждая переменная может иметь.
2. Вычислить число комбинаций условий. Если все условия являются бинарными, то существует 2**N комбинаций N переменных.
3. Идентифицировать каждое из возможных действий, которые могут вызываться в спецификации.
4. Построить пустую таблицу, включающую все возможные условия и действия, а также номера комбинаций условий.
5. Выписать и занести в таблицу все возможные комбинации условий.
6. Редуцировать комбинации условий.
7. Проверить каждую комбинацию условий и идентифицировать соответствующие выполняемые действия.
8. Выделить комбинации условий, для которых спецификация не указывает список выполняемых действий.
9. Обсудить построенную таблицу.

Вариантом таблицы решений является дерево решений (ДР), позволяющее взглянуть на процесс условного выбора с позиции схемы. Дерево решений для вышерассмотренного примера приведено на рис. 4.1

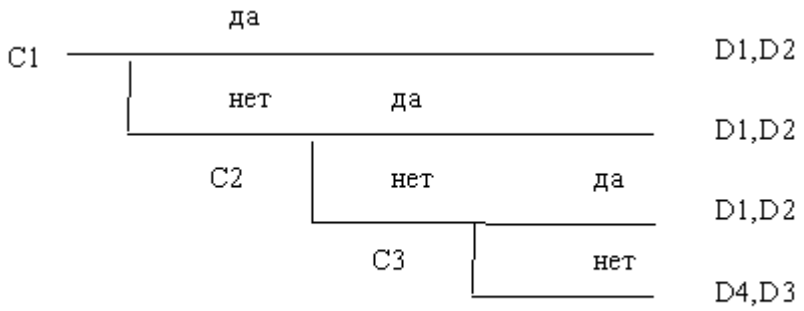


Рис. 4.1. **Дерево решений**

Обычно ДР используется при малом числе действий и когда не все комбинации условий возможны, а ТР - при большом числе действий и когда возможно большинство комбинаций условий.
Рекламная пауза!

4.3. Визуальные языки проектирования спецификаций

Визуальные языки проектирования являются относительно новой, оригинальной методикой разработки спецификаций процесса. Они базируются на основных идеях структурного программирования и позволяют определять потоки управления с помощью специальных иерархически организованных схем.

Одним из наиболее известных подходов к визуальному проектированию спецификаций является подход с использованием **FLOW-форм**. Каждый символ FLOW-формы имеет вид прямоугольника и может быть вписан в любой внутренний прямоугольник любого другого символа. Символы помечаются с помощью предложений на естественном языке или с использованием математической нотации.

Символы FLOW-форм приведены на рис. 4.2. Каждый символ является блоком обработки. Каждый прямоугольник внутри любого символа также представляет собой блок обработки.

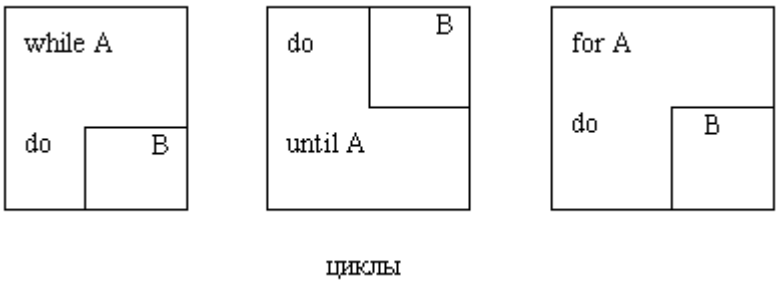


Рис. 4.2. Символы FLOW-форм.

На рис 4.3. приведен пример использования данного подхода при проектировании спецификации процесса, обеспечивающего упорядочивание определенным образом элементов массива и являющегося фрагментом алгоритма сортировки методом "поплавка".

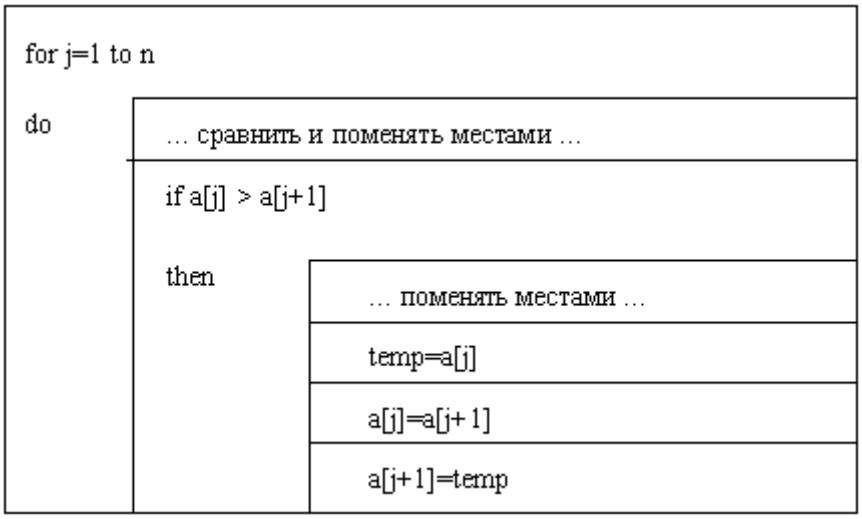


Рис. 4.3. Пример FLOW-формы

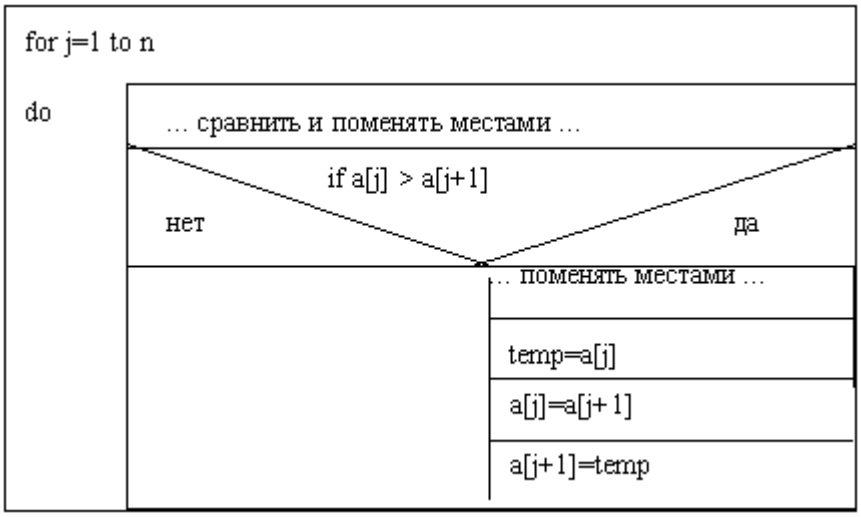


Рис. 4.4. Диаграмма Насси-Шнейдермана.

Дальнейшее развитие FLOW-формы получили в диаграммах Насси-Шнейдермана. На этих диаграммах символы последовательной обработки и цикла изображаются также, как и соответствующие символы FLOW-форм. В символах условного выбора и case-выбора собственно условие располагается в верхнем треугольнике, выбираемые варианты - на нижних сторонах треугольника, а блоки обработки - под выбираемыми вариантами. Диаграмма Насси-Шнейдермана для вышеприведенного примера изображена на рис. 4.4.

4.4. Сравнение методов

Спектр методов задания спецификаций процессов в соответствии с увеличением трудности их проектирования приведен на рис 4.5. Наиболее трудным методом задания СП являются языки программирования (C, COBOL, FORTRAN и др.). Сложность заключается в том, что языки программирования концентрируют внимание на деталях реализации, а потоки данных в DFD представляются абстрактно (их фактическая композиция определяется в словаре данных). Поэтому сложность - не в написании СП, а в их синхронизации и согласовании с DFD, поскольку при редактировании DFD, вообще говоря, должны корректироваться и спецификации процессов.

Текстовое описание	***Структурированный естественный язык***	***таблица решений***	***дерево решений***	***Визуальный язык***	***язык программирования***

Рис. 4.5. Спектр методов задания спецификаций процессов

Перечислим некоторые положительные и отрицательные стороны рассмотренных методов задания СП.

Структурированный естественный язык применяется в случаях, когда детали СП известны не полностью. Он обеспечивает быстрое проектирование СП, прост в использовании, легко понимаем проектировщиками и программистами, а также конечным пользователем. К его недостаткам относятся отсутствие процедурных возможностей и неспособность к автоматической кодогенерации из-за наличия неоднозначностей.

Таблицы и деревья решений позволяют управлять сложными комбинациями условий и действий, обеспечивают визуальное (табличное и графическое, соответственно) представление СП и легко понимаемы конечным пользователем. Кроме этого, таблицы решений позволяют легко идентифицировать несущественности и бреши в СП. Главным недостатком методов является отсутствие процедурных возможностей.

Визуальные языки проектирования поддерживаются автоматической кодогенерацией, позволяют осуществлять декомпозицию СП. Их недостаток - трудность модификации СП при изменении деталей.

4.5. Спецификации процессов для примера банковской задачи

Приведем спецификации процессов для рассмотренного в 2.5 примера банковской задачи с использованием структурированного естественного языка.

1) Спецификация процесса 1.1 приведена в 4.1.
2) Спецификация процесса 1.2.

@ВХОД *= ЛИМИТ ДЕНЕГ*
@ВХОД *= ЗАПРОС НА ОБСЛУЖИВАНИЕ*
@ВЫХОД *= ДЕНЕЖНАЯ СУММА*
@ВЫХОД *= СООБЩЕНИЕ*
@ВЫХОД *= ТРЕБУЕМОЕ ОБСЛУЖИВАНИЕ*
@СПЕЦПРОЦ *1.2 ПОЛУЧИТЬ ЗАПРОС НА ОБСЛУЖИВАНИЕ*
ВЫПОЛНИТЬ *выдать СООБЩЕНИЕ клиенту по вводу запроса на обслуживание*
принять ЗАПРОС НА ОБСЛУЖИВАНИЕ
обновить данные ТРЕБУЕМОЕ ОБСЛУЖИВАНИЕ (а именно,
ЗАПРОС ДОКУМЕНТАЦИИ, ЗАПРОС ДЕНЕГ,
ЗАПРОС БАЛАНСА, ЗАПРОС НА ОПЕРАЦИЮ)
ЕСЛИ *был сделан ЗАПРОС ДЕНЕГ*
ТО *ВЫПОЛНИТЬ запросить ДЕНЕЖНУЮ СУММУ*
выдать требуемую ДЕНЕЖНУЮ СУММУ с учетом того,
что она не должно превышать ЛИМИТ ДЕНЕГ
КОНЕЦЕСЛИ
ДОТЕХПОРПОКА *запрашивается продолжение обслуживания*
или не все обслуживание было выполнено
КОНЕЦВЫПОЛНИТЬ
@ КОНЕЦ СПЕЦИФИКАЦИИ ПРОЦЕССА *1.2*

3) Спецификация процесса 1.3.
3.1) Спецификация процесса 1.3.1.

@ВХОД *= ЗАПРОС ДОКУМЕНТАЦИИ*
@ВХОД *= ДЕТАЛИ КЛИЕНТА*
@ВЫХОД *= ОБРАБОТАННАЯ ДОКУМЕНТАЦИЯ*
@СПЕЦПРОЦ *1.3.1 ОБРАБОТАТЬ ДОКУМЕНТАЦИЮ БАНКА*
По получении ЗАПРОСА ДОКУМЕНТАЦИИ выдать ОБРАБОТАННУЮ ДОКУМЕНТАЦИЮ,

содержащую ДЕТАЛИ КЛИЕНТА, КОМПЬЮТЕРУ БАНКА
***@КОНЕЦ СПЕЦИФИКАЦИИ** ПРОЦЕССА 1.3.1*

3.2) Спецификация процесса 1.3.2.

***@ВХОД** = ДАННЫЕ ПО БАЛАНСУ*
***@ВХОД** = ЗАПРОС БАЛАНСА*
***@ВХОД** = ДЕТАЛИ КЛИЕНТА*
***@ВЫХОД** = ДАННЫЕ ПО ИСТОРИИ ЗАПРОСА*
***@ВЫХОД** = ВЫПИСКА ПО БАЛАНСУ*
***@СПЕЦПРОЦ** 1.3.2 РАСПЕЧАТАТЬ БАЛАНС КЛИЕНТА*
По получении ЗАПРОСА БАЛАНСА выдать ДАННЫЕ ПО ИСТОРИИ ЗАПРОСА
Затем выдать ВЫПИСКУ ПО БАЛАНСУ, содержащую ДАННЫЕ ПО БАЛАНСУ@ КОНЕЦ СПЕЦИФИКАЦИИ ПРОЦЕССА 1.3.2

3.3) Спецификация процесса 1.3.3.

***@ВХОД** = ДЕНЕЖНАЯ СУММА*
***@ВХОД** = ЗАПРОС ДЕНЕГ*
***@ВХОД** = ДЕТАЛИ КЛИЕНТА*
***@ВЫХОД** = ДЕНЬГИ*
***@ВЫХОД** = ВЫПИСКА О ДЕНЬГАХ*
***@ВЫХОД** = ДЕНЕЖНАЯ СУММА*
***@СПЕЦПРОЦ** 1.3.3 ПРИГОТОВИТЬ ДЕНЬГИ ДЛЯ КЛИЕНТА*
По получении ЗАПРОСА ДЕНЕГ выдать ДЕНЬГИ по значению ДЕНЕЖНОЙ СУММЫ
Выдать ВЫПИСКУ О ДЕНЬГАХ, содержащую ДЕНЕЖНУЮ СУММУ
Передать КОМПЬЮТЕРУ БАНКА информацию о ДЕНЕЖНОЙ СУММЕ
***@ КОНЕЦ СПЕЦИФИКАЦИИ** ПРОЦЕССА 1.3.3*

3.4) Спецификация процесса 1.3.4.

***@ВХОД** = ДАННЫЕ ПО СЧЕТУ*
***@ВХОД** = ЗАПРОС НА ОПЕРАЦИЮ*
***@ВХОД** = ДЕТАЛИ КЛИЕНТА*
***@ВЫХОД** = ДАННЫЕ ПО ИСТОРИИ ЗАПРОСА*
***@ВЫХОД** = ВЫПИСКА ПО ОПЕРАЦИИ*
***@СПЕЦПРОЦ** 1.3.4 РАСПЕЧАТАТЬ ОПЕРАЦИЮ КЛИЕНТА*
получении ЗАПРОСА НА ОПЕРАЦИЮ выдать ДАННЫЕ ПО ИСТОРИИ ЗАПРОСА для специфицирования ДЕТАЛЕЙ КЛИЕНТА, чтобы получить текущие ДАННЫЕ ПО СЧЕТУ
Выдать ВЫПИСКУ ПО ОПЕРАЦИИ, содержащую ДАННЫЕ ПО СЧЕТУ
***@ КОНЕЦ СПЕЦИФИКАЦИИ** ПРОЦЕССА 1.3.4*

4) Спецификация процесса 1.4.

***@ВХОД** = УДАЛЕННАЯ КРЕДИТНАЯ КАРТА*
***@ВХОДВЫХОД** = КРЕДИТНАЯ КАРТА*
***@ВЫХОД** = ДАННЫЕ КРЕДИТНОЙ КАРТЫ*
***@ВЫХОД** = ВВЕДЕННАЯ КРЕДИТНАЯ КАРТА*
***@СПЕЦПРОЦ** 1.4 ОБРАБОТАТЬ КРЕДИТНУЮ КАРТУ*
ВЫПОЛНИТЬ считать КРЕДИТНУЮ КАРТУ
записать в хранилище ДАННЫЕ КРЕДИТНОЙ КАРТЫ
выдать управляющий поток ВВЕДЕННАЯ КРЕДИТНАЯ КАРТА
по получении управляющего потока УДАЛЕННАЯ КРЕДИТНАЯ КАРТА удалить КРЕДИТНУЮ КАРТУ
***@ КОНЕЦ СПЕЦИФИКАЦИИ** ПРОЦЕССА 1.4*

ГЛАВА 5

ДИАГРАММЫ “СУЩНОСТЬ-СВЯЗЬ”

Диаграммы "сущность-связь" (ERD) предназначены для разработки моделей данных и обеспечивают стандартный способ определения данных и отношений между ними. Фактически с помощью ERD осуществляется детализация хранилищ данных проектируемой системы, а также документируются сущности системы и способы их взаимодействия, включая идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их отношений с другими объектами (связей).

Данная нотация была введена Ченом (Chen) и получила дальнейшее развитие в работах Баркера (Barker). Нотация Чена предоставляет богатый набор средств моделирования данных, включая собственно ERD, а также диаграммы атрибутов и диаграммы декомпозиции. Эти диаграммные техники используются прежде всего для проектирования реляционных баз данных (хотя также могут с успехом применяться и для моделирования как иерархических, так и сетевых баз данных).

5.1. Сущности, отношения и связи в нотации Чена

СУЩНОСТЬ представляет собой множество экземпляров реальных или абстрактных объектов (людей, событий, состояний, идей, предметов и т.п.), обладающих общими атрибутами или характеристиками. Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована. При этом имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр (например, *АЭРОПОРТ*, а не *ВНУКОВО*).

ОТНОШЕНИЕ в самом общем виде представляет собой связь между двумя и более сущностями. Именование отношения осуществляется с помощью грамматического оборота глагола (*ИМЕЕТ, ОПРЕДЕЛЯЕТ, МОЖЕТ ВЛАДЕТЬ* и т.п.).

Другими словами, сущности представляют собой базовые типы информации, хранимой в базе данных, а отношения показывают, как эти типы данных взаимоувязаны друг с другом. Введение подобных отношений преследует две основополагающие цели:

- обеспечение хранения информации в единственном месте (даже если она используется в различных комбинациях);
- использование этой информации различными приложениями.

Символы ERD, соответствующие сущностям и отношениям, приведены на рис. 5.1.

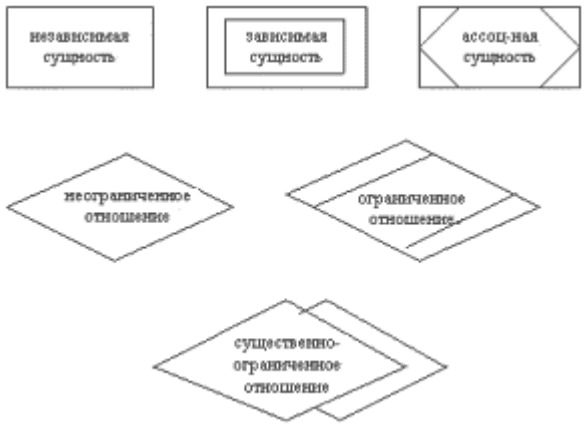


Рис.5.1. Символы ERD в нотации Чена

Независимая сущность представляет независимые данные, которые всегда присутствуют в системе. При этом отношения с другими сущностями могут как существовать, так и отсутствовать. В свою очередь **зависимая сущность** представляет данные, зависящие от других сущностей в системе. Поэтому она должна всегда иметь отношения с другими сущностями. **Ассоциированная сущность** представляет данные, которые ассоциируются с отношениями между двумя и более сущностями (см. 5.5).

Неограниченное (обязательное) отношение представляет собой безусловное отношение, т.е. отношение, которое всегда существует до тех пор, пока существуют относящиеся к делу сущности. **Ограниченное (необязательное) отношение** представляет собой условное отношение между сущностями. **Существенно-ограниченное отношение** используется, когда соответствующие сущности взаимно-зависимы в системе.

Для идентификации требований, в соответствии с которыми сущности вовлекаются в отношения, используются ***СВЯЗИ***. Каждая связь соединяет сущность и отношение и может быть направлена только от отношения к сущности.

ЗНАЧЕНИЕ связи характеризует ее тип и, как правило, выбирается из следующего множества:

{"О или 1", "0 или более", "1", "1 или более", "p:q" (диапазон)}.

Пара значений связей, принадлежащих одному и тому же отношению, определяет тип этого отношения. Практика показала, что для большинства приложений достаточно использовать следующие типы отношений:

1. **1*1 (один-к-одному)**. Отношения данного типа используются, как правило, на верхних уровнях иерархии модели данных, а на нижних уровнях встречаются сравнительно редко.
2. **1*n (один-к-многим)**. Отношения данного типа являются наиболее часто используемыми.
3. **n*m (многие-к-многим).** Отношения данного типа обычно используются на ранних этапах проектирования с целью прояснения ситуации. В дальнейшем каждое из таких отношений должно быть преобразовано в комбинацию отношений типов 1 и 2 (возможно, с добавлением вспомогательных сущностей и с введением новых отношений).

На рис.5.2 приведена диаграмма "сущность-связь", демонстрирующая отношения между объектами банковской системы (см. п.2.5). Согласно этой диаграмме каждый *БАНК ИМЕЕТ* один или более *БАНКОВСКИХ СЧЕТОВ*. Кроме того, каждый *КЛИЕНТ МОЖЕТ ВЛАДЕТЬ* (одновременно) одной или более *КРЕДИТНОЙ КАРТОЙ* и одним или более *БАНКОВСКИМ СЧЕТОМ*, каждый из которых *ОПРЕДЕЛЯЕТ* в точности одну *КРЕДИТНУЮ КАРТУ* (отметим, что у клиента может и не быть ни счета, ни кредитной карты). Каждая *КРЕДИТНАЯ КАРТА ИМЕЕТ* ровно один зависимый от нее *ПАРОЛЬ КАРТЫ*, а каждый *КЛИЕНТ ЗНАЕТ* (но может и забыть) *ПАРОЛЬ КАРТЫ.*

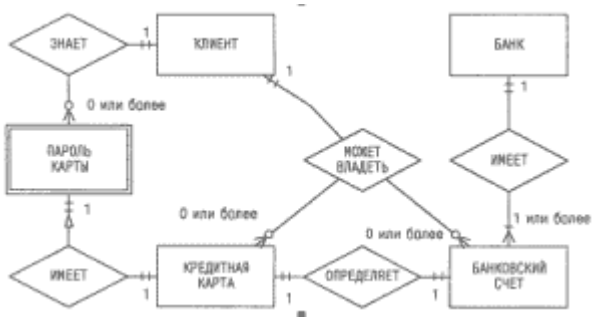


Рис 5.2. ER-диаграмма в нотации Чена.

Рис. 5.3. Диаграмма атрибутов.

5.2. Диаграммы атрибутов

Каждая сущность обладает одним или несколькими **атрибутами**, которые однозначно идентифицируют каждый экземпляр сущности. При этом любой атрибут может быть определен как ключевой.

Детализация сущности осуществляется с использованием **диаграмм атрибутов**, которые раскрывают ассоциированные с сущностью атрибуты. Диаграмма атрибутов состоит из детализируемой сущности, соответствующих атрибутов и **доменов**, описывающих области значений атрибутов. На диаграмме каждый атрибут представляется в виде связи между сущностью и соответствующим доменом, являющимся графическим представлением множества возможных значений атрибута. Все атрибутные связи имеют значения на своем окончании. Для идентификации ключевого атрибута используется подчеркивание имени атрибута.

Пример диаграммы атрибутов, детализирующей сущность *КРЕДИТНАЯ КАРТА* (см. рис. 5.2) приведен на рис. 5.3.

5.3. Категоризация сущностей

Сущность может быть разделена и представлена в виде двух или более **сущностей-категорий**, каждая из которых имеет общие атрибуты и/или отношения, которые определяются однажды на верхнем уровне и наследуются на нижнем. Сущности-категории могут иметь и свои собственные атрибуты и/или отношения, а также, в свою очередь, могут быть декомпозированы своими сущностями-категориями на следующем уровне. Расщепляемая на категории сущность получила название **общей сущности** (отметим, что на промежуточных уровнях декомпозиции одна и та же сущность может быть как общей сущностью, так и сущностью-категорией).

Для демонстрации декомпозиции сущности на категории используются **диаграммы категоризации**. Такая диаграмма содержит общую сущность, две и более сущности-категории и специальный **узел-дискриминатор**, который описывает способы декомпозиции сущностей (см. рис. 5.4).

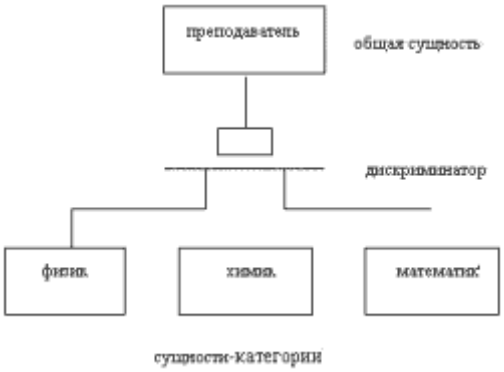


Рис. 5.4. Диаграмма категоризации

Существуют 4 возможных типа дискриминатора (рис.5.5):

1. **Полное и обязательное вхождение *E/M*** (exclusive/mandatory) - сущность должна быть одной и только одной из следуемых категорий. Для примера на рис. 5.4 это означает, что *ПРЕПОДАВАТЕЛЕМ* является *ФИЗИК,* или *ХИМИК*, или *МАТЕМАТИК*.
2. **Полное и необязательное вхождение *E/O*** (exclusive/optional) - сущность может быть одной и только одной из следуемых категорий. Это означает, что *ПРЕПОДАВАТЕЛЕМ* является *ФИЗИК,* или *ХИМИК*, или *МАТЕМАТИК,* или преподаватель какой-либо другой дисциплины (например, *ИСТОРИК*).
3. **Неполное и обязательное вхождение *I/M*** (inclusive/mandatory) - сущность должна быть по крайней мере одной из следуемых категорий. Это предполагает в дополнение к 1) задавать следующую ситуацию: *ПРЕПОДАВАТЕЛЕМ* является одновременно и *ФИЗИК* и *ХИМИК*
4. **Неполное и необязательное вхождение *I/O*** (inclusive/optional) - сущность может быть по крайней мере одной из следуемых категорий. В дополнение к 2) *ПРЕПОДАВАТЕЛЕМ* является преподаватель какой-либо другой дисциплины (например, *ИСТОРИК*).

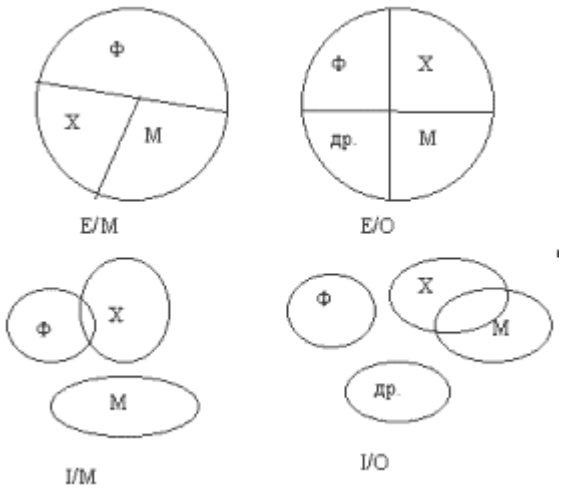


Рис 5.5. Типы дискриминаторов.

5.4. Нотация Баркера

Дальнейшее развитие ER-подход получил в работах Баркера, предложившего оригинальную нотацию, которая позволила на верхнем уровне интегрировать предложенные Ченом средства описания моделей.

В нотации Баркера используется только один тип диаграмм - ERD. Сущность на ERD представляется прямоугольником любого размера, содержащим внутри себя имя сущности, список имен атрибутов (возможно, неполный) и указатели ключевых атрибутов (знак "#" перед именем атрибута).

Все связи являются бинарными и представляются линиями с двумя концами (соединяющими сущности), для которых должно быть определено имя, степень множественности (один или много объектов участвуют в связи) и степень обязательности (т.е. обязательная или необязательная связь между сущностями). Для множественной связи линия присоединяется к прямоугольнику сущности в трех точках, а для одиночной связи - в одной точке. При обязательной связи рисуется непрерывная линия до середины связи, при необязательной - пунктирная линия. На рис. 5.6 приведен фрагмент ERD для банковской задачи в нотации Баркера.

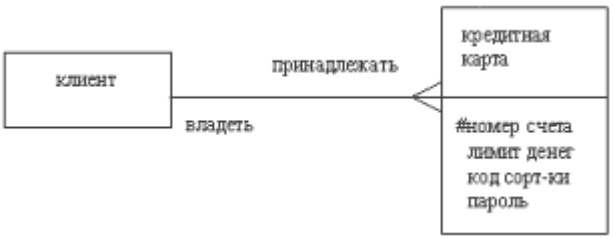


Рис. 5.6. Нотация Баркера.

Читается связь отдельно для каждого конца, показывая, как сущность *КЛИЕНТ* связывается с сущностью *КРЕДИТНАЯ КАРТА*, и наоборот. При этом необходимо учитывать степень обязательности выбранного конца связи, для этой цели используются слова "должен (быть)" или "может (быть)". Так, диаграмма, приведенная на рис. 5.6, читается следующим образом:

Каждый КЛИЕНТ может ВЛАДЕТЬ одной или более КРЕДИТНОЙ КАРТОЙ или

Каждая КРЕДИТНАЯ КАРТА должна ПРИНАДЛЕЖАТЬ ровно одному КЛИЕНТУ.

В заключение отметим, что понятия категория и общая сущность заменяются Баркером на эквивалентные понятия **подтипа** и **супертипа**, соответственно.

5.5. Построение модели

Разработка ERD включает следующие основные этапы:

1. Идентификация сущностей, их атрибутов, а также первичных и альтернативных ключей.
2. Идентификация отношений между сущностями и указание типов отношений.
3. Разрешение неспецифических отношений (отношений n*m).

<u>Этап 1</u> является определяющим при построении модели, его исходной информацией служит содержимое хранилищ данных, определяемое входящими и выходящими в/из него потоками данных. На рис. 5.7 приведен фрагмент диаграммы потоков данных, моделирующей деятельность бухгалтерии предприятия. Его единственное хранилище *ДАННЫЕ О ПЕРСОНАЛЕ* должно содержать информацию о всех сотрудниках: их имена, адреса, должности, оклады и т.п.

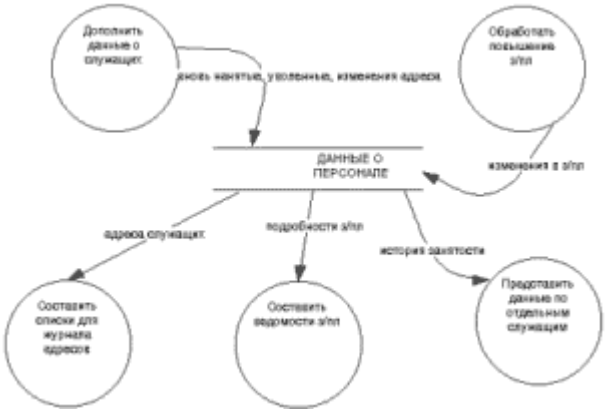


Рис. 5.7. Деятельность бухгалтерии

Первоначально осуществляется анализ хранилища, включающий сравнение содержимого входных и выходных потоков и создание на основе этого сравнения варианта схемы хранилища. Перечислим структуры данных, содержащиеся во входных и выходных потоках:

- *вновь_нанятые адрес_служащего*
- *дата_найма фамилия*
- *фамилия адрес*
- *таб_номер*
- *адрес подробности_з/пл*
- *должность фамилия*
- *начальная_з/пл таб_номер*
- *текущая_з/пл*
- *уволенные*
- *фамилия история_занятости*
- *таб_номер фамилия*
- *таб_номер*
- *изменение_адреса дата_найма*
- *фамилия история_карьеры* *
- *таб_номер должность*
- *старый_адрес дата_изменения*
- *новый_адрес история_з/пл* *
- *з/пл*
- *изменение_з/пл*
- *фамилия*
- *таб_номер*
- *старая_з/пл*
- *новая_з/пл*
- *дата_изменения*

Сравнивая входные и выходные структуры, отметим следующие моменты:

1. Поле *АДРЕС* хранит текущий адрес сотрудника, а структура *ИЗМЕНЕНИЕ_АДРЕСА* хранит и старый адрес, что не является необходимым, исходя из выходных потоков.
2. *ИСТОРИЯ_З/ПЛ*, наоборот, требует перечень всех окладов сотрудника, поэтому необходимо иметь набор, состоящий из пар (*З/ПЛ, ДАТА*), а не просто *СТАРАЯ_З/ПЛ* и *НОВАЯ_З/ПЛ* (как во входном потоке).
3. Аналогичная ситуация и с *ИСТОРИЕЙ_КАРЬЕРЫ*. Отметим, что на диаграмме вообще отсутствует поток, определяющий изменения в должности, то есть обнаружено серьезное упущение в функциональной модели!
4. Отметим, что изменение в *ДОЛЖНОСТИ* обычно (но не всегда) соответствует изменению в *З/ПЛ*.

С учетом этих моментов первый вариант схемы может выглядеть следующим образом:

- *фамилия*
- *таб_номер*
- *адрес*
- *текущая_з/пл*
- *дата_найма*
- *история_карьеры* *
- *должность*
- *дата_изменения*
- *история_з/пл* *
- *з/пл*
- *дата_изменения*

На следующем шаге осуществляется упрощение схемы за счет устранения избыточности. Действительно, *ТЕКУЩАЯ_З/ПЛ* всегда является последней записью в *ИСТОРИИ_З/ПЛ*, а *ДАТА_НАЙМА* содержится в разделах *ИСТОРИЯ_З/ПЛ* и *ИСТОРИЯ_КАРЬЕРЫ*. Кроме того, несколько дат в последних разделах одни и те же, поэтому целесообразно создать на их основе структуру *ИСТОРИЯ_З/ПЛ_КАРЬЕРЫ* и вводить в нее данные при изменении *ДОЛЖНОСТИ* и/или *З/ПЛ*.

- *фамилия*
- *таб_номер*
- *адрес*
- *история_з/пл_карьеры* *
- *з/пл*
- *должность*
- *дата_изменения*

Следующий шаг - упрощение схемы при помощи **нормализации** (удаления повторяющихся групп). Единственным способом нормализации является расщепление данной схемы на две, являющиеся более простыми. Первая схема содержит *ФАМИЛИЮ* и *АДРЕС* (которые, как правило, не меняются), вторая - каждое изменение *З/ПЛ* и *ДОЛЖНОСТИ*. Кроме того, каждая схема должна содержать *ТАБ_НОМЕР* - единственный элемент данных, уникально идентифицирующий каждого сотрудника.

Для идентификации сущностей осталось определить ключевые атрибуты. Для первой схемы ключевым атрибутом является *ТАБ_НОМЕР*, для второй - ключом является конкатенация атрибутов *ТАБ_НОМЕР* и *ДАТА_ИЗМЕНЕНИЯ* (рис.5.8), т.к. для каждого сотрудника возможно несколько записей в схеме *ИСТОРИЯ_З/ПЛ_КАРЬЕРЫ*.

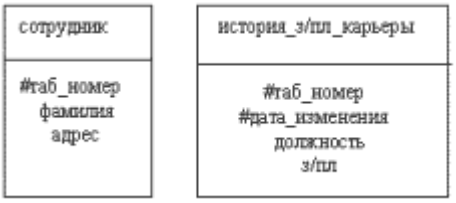


Рис. 5.8. Сущности модели

Концепции и методы нормализации были разработаны Коддом (Codd), установившим существование трех типов нормализованных схем, названных в порядке уменьшения сложности **первой, второй и третьей нормальной формой** (соответственно, **1НФ**, **2НФ** и **3НФ**). Рассмотрим, как преобразовывать схемы к наиболее простой 3НФ. При этом будем представлять схемы в общепринятом виде, например, для сущностей, приведенных на рис.5.8, имеем:

- *история_з/пл_карьеры(таб_номер,дата_изменения, должность, з/пл)*
- *сотрудник(таб_номер, фамилия, адрес)*

Для примера построения 3НФ рассмотрим следующую схему, ключ которой выбран в предположении, что заказчик не заказывает одну и ту же книгу дважды в один и тот же день:

- *заказ_на_книгу(имя_заказчика, дата_заказа, ISBN, название, автор, количество, цена, сумма_заказа)*

Согласно Кодду, любая нормализованная схема (схема без повторяющихся групп) автоматически находится в 1НФ независимо от того, насколько сложен ее ключ и какая взаимосвязь может существовать между ее элементами.

Отметим, что в последней схеме атрибуты *НАЗВАНИЕ, АВТОР, ЦЕНА* могут быть идентифицированы частью ключа (а именно, *ISBN*), тогда как атрибут *КОЛИЧЕСТВО* зависит от

всего ключа (соответственно, полная и частичная функциональная зависимость от ключа). По определению схема находится в 2НФ если все ее неключевые атрибуты полностью функционально зависят от ключа. После избавления от частичной функциональной зависимости последняя схема будет выглядеть следующим образом:

- *заказ_на_книгу (имя_заказчика, дата_заказа, ISBN, количество, сумма_заказа)*
- *книга (ISBN, автор, название, цена)*

Заметим, что возможно упростить ситуацию и дальше: атрибуты *КОЛИЧЕСТВО* и *СУММА_ЗАКАЗА* являются взаимно-зависимыми. По определению схема находится в 3НФ если она находится в 2НФ и никакой из неключевых атрибутов не является зависимым ни от какого другого неключевого атрибута. Поскольку в нашем примере атрибут *СУММА_ЗАКАЗА* фактически является избыточным, для получения 3НФ его можно просто удалить.

Иногда для построения 3НФ необходимо выразить зависимость между неключевыми атрибутами в виде отдельной схемы. Так для сотрудников, работающих по различным проектам, возможна следующая схема:

- *сотрудник (таб_номер, телефон, почасовая_оплата, N_проекта, дата_окончания)*

Очевидно, что данная схема находится в 2НФ. Однако *N_ПРОЕКТА* и *ДАТА_ОКОНЧАНИЯ* являются зависимыми атрибутами. После расщепления схемы получим 3НФ:

- *участник_проекта(таб_номер, телефон, почасовая_оплата, N_проекта)*
- *проект(N_проекта, дата_окончания)*

На практике отношения 1НФ и 2НФ имеют тенденцию возникать при попытке описать несколько реальных сущностей в одной схеме (заказ и книга, проект и сотрудник). 3НФ является наиболее простым способом представления данных, отражающим здравый смысл. Построив 3НФ, мы фактически выделяем базовые сущности предметной области.

В заключение зафиксируем алгоритм приведения ненормализованных схем в третью нормальную форму (рис. 5.9).

Этап 2 служит для выявления и определения отношений между сущностями, а также для идентификации типов отношений. На данном этапе некоторые отношения могут быть неспецифическими (n*m - многие-ко-многим). Такие отношения потребуют дальнейшей детализации на этапе 3.

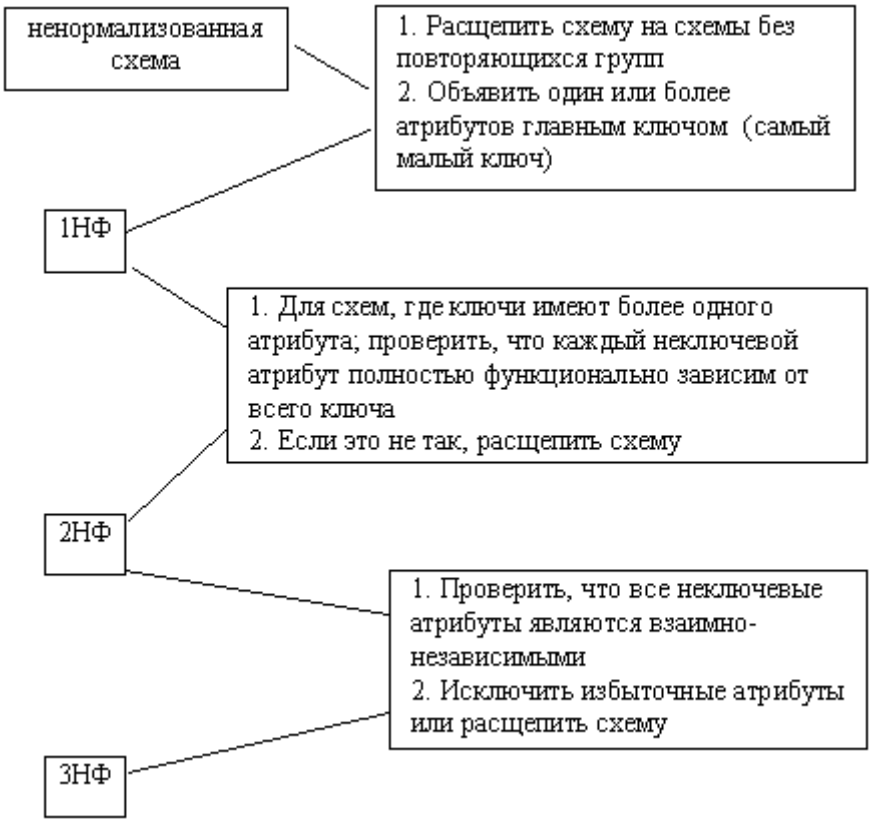


Рис. 5.9. Алгоритм приведения в 3НФ

Определение отношений включает выявление связей, для этого отношение должно быть проверено в обоих направлениях следующим образом: выбирается экземпляр одной из сущностей и определяется, сколько различных экземпляров второй сущности может быть с ним связано, и наоборот. Для примера на рис. 5.8 рассмотрим отношение между сущностями *СОТРУДНИК* и *ИСТОРИЯ_З/ПЛ_КАРЬЕРЫ*. У отдельного сотрудника должность и/или зарплата может меняться ноль, один или много раз, порождая соответствующее число экземпляров сущности *ИСТОРИЯ_З/ПЛ_КАРЬЕРЫ*. Анализируя в другом направлении, видим, что каждый экземпляр сущности *ИСТОРИЯ_З/ПЛ_КАРЬЕРЫ* соответствует ровно одному конкретному сотруднику. Поэтому между этими двумя сущностями имеется отношение типа 1*n (один ко многим) со связью "один" на конце отношения у сущности *СОТРУДНИК* и со связью "ноль, один или много" на конце у сущности *ИСТОРИЯ_З/ПЛ_КАРЬЕРЫ*.

Этап 3 предназначен для разрешения неспецифических (многие ко многим) отношений. Для этого каждое неспецифическое отношение преобразуется в два специфических отношения с введением новых (а именно, ассоциативных) сущностей. Рассмотрим пример на рис. 5.10.

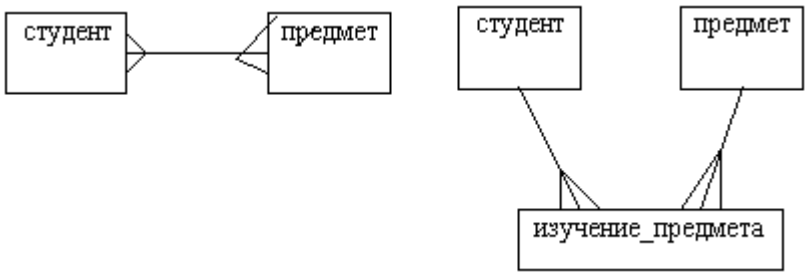


Рис. 5.10. Разрешение неспецифического отношения

Неспецифическое отношение на рис 5.10 указывает, что *СТУДЕНТ* может изучать много *ПРЕДМЕТОВ*, а *ПРЕДМЕТ* может изучаться многими *СТУДЕНТАМИ*. Однако мы не можем определить, какой *СТУДЕНТ* изучает какой *ПРЕДМЕТ*, пока не введем для разрешения этого

неспецифического отношения третью (ассоциативную) сущность *ИЗУЧЕНИЕ_ПРЕДМЕТА*. Каждый экземпляр введенной сущности связан с одним *СТУДЕНТОМ* и с одним *ПРЕДМЕТОМ.*

Таким образом, ассоциативные сущности по своей природе являются представлениями пар реальных объектов и обычно появляются на этапе 3.

ГЛАВА 6

СПЕЦИФИКАЦИИ УПРАВЛЕНИЯ

Спецификации управления предназначены для моделирования и документирования аспектов систем, зависящих от времени или реакции на событие. Они позволяют осуществлять декомпозицию управляющих процессов и описывают отношения между входными и выходными управляющими потоками на управляющем процессе-предке. Для этой цели обычно используются **диаграммы переходов состояний** (STD).

С помощью STD можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования. Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены.

STD состоит из следующих объектов:

СОСТОЯНИЕ - может рассматриваться как условие устойчивости для системы. Находясь в определенном состоянии, мы имеем достаточно информации о прошлой истории системы, чтобы определить очередное состояние в зависимости от текущих входных событий. Имя состояния должно отражать реальную ситуацию, в которой находится система, например, *НАГРЕВАНИЕ, ОХЛАЖДЕНИЕ* и т.п.

НАЧАЛЬНОЕ СОСТОЯНИЕ - узел STD, являющийся стартовой точкой для начального системного перехода. STD имеет ровно одно начальное состояние, соответствующее состоянию системы после ее инсталляции, но перед началом реальной обработки, а также любое (конечное) число завершающих состояний.

ПЕРЕХОД определяет перемещение моделируемой системы из одного состояния в другое. При этом имя перехода идентифицирует событие, являющееся причиной перехода и управляющее им. Это событие обычно состоит из управляющего потока (сигнала), возникающего как во внешнем мире, так и внутри моделируемой системы при выполнении некоторого **условия** (например, *СЧЕТЧИК=999* или *КНОПКА НАЖАТА*). Следует отметить, что, вообще говоря, не все события необходимо вызывают переходы из отдельных состояний. С другой стороны, одно и то же событие не всегда вызывает переход в то же самое состояние.

Таким образом ***УСЛОВИЕ*** представляет собой событие (или события), вызывающее переход и идентифицируемое именем перехода. Если в условии участвует входной управляющий поток управляющего процесса-предка, то имя потока должно быть заключено в кавычки, например, "*ПАРОЛЬ*"=666, где *ПАРОЛЬ* - входной поток.

Кроме условия с переходом может связываться **действие** или ряд действий, выполняющихся, когда переход имеет место. То есть ***ДЕЙСТВИЕ*** - это операция, которая может иметь место при выполнении перехода. Если действие необходимо для выбора выходного управляющего потока, то имя этого потока должно заключаться в кавычки, например:

"ВВЕДЕННАЯ КАРТА" = ***TRUE*** ,

где ВВЕДЕННАЯ КАРТА - выходной поток.

Кроме того, для спецификации A-, T-, E/D-потоков имя запускаемого или переключаемого процесса также должно заключаться в кавычки, например:

A: "ПОЛУЧИТЬ ПАРОЛЬ" - активировать процесс ПОЛУЧИТЬ ПАРОЛЬ.

Фактически условие есть некоторое внешнее или внутреннее событие, которое система способна обнаружить и на которое она должна отреагировать определенным образом, изменяя свое состояние. При изменении состояния система обычно выполняет одно или более действий (производит вывод, выдает сообщение на терминал, выполняет вычисления). Таким образом, действие представляет собой отклик, посылаемый во внешнее окружение, или вычисление, результаты которого запоминаются в системе (обычно в хранилищах данных на DFD), для того, чтобы обеспечить реакцию на некоторые из планируемых в будущем событий.

На STD состояния представляются узлами, а переходы - дугами (рис. 6.1). Условия (по-другому называемые стимулирующими событиями) идентифицируются именем перехода и возбуждают выполнение перехода. Действия или отклики на события привязываются к переходам и записываются под соответствующим условием. Начальное состояние на диаграмме должно иметь входной переход, изображаемый потоком из подразумеваемого стартового узла (иногда этот стартовый узел изображается небольшим квадратом и привязывается к входному состоянию).

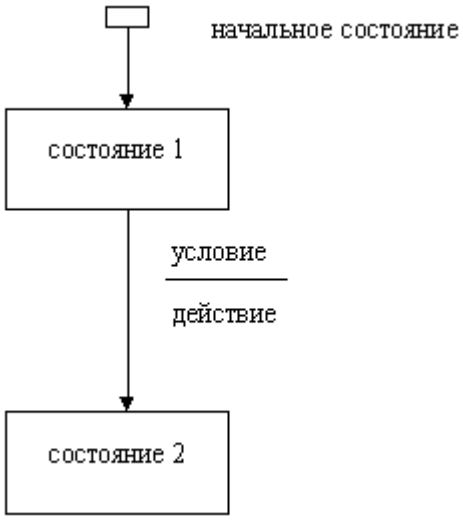


Рис. 6.1. Символы STD

Диаграмма переходов состояний для примера банковской задачи приведена на рис. 6.2. Она содержит два состояния - *ОЖИДАНИЕ* и *ОБРАБОТКА*. Переход из состояния *ОЖИДАНИЕ* в состояние *ОБРАБОТКА* осуществляется при условии ввода кредитной карты (*ВВЕДЕННАЯ КРЕДИТНАЯ КАРТА*). При этом выполняется действие по запуску процесса 1.1 на рис. 2.8 (*ПОЛУЧИТЬ ПАРОЛЬ*). Отметим, что для запуска используется А-поток, обеспечивающий непрерывность процесса 1.1, т.е. возможность повторного ввода пароля. Переход из состояния *ОБРАБОТКА* в состояние *ОЖИДАНИЕ* осуществляется двумя различными способами. При условии трехкратного ввода неверного пароля (см. спецификацию процесса 1.1) кредитная карта удаляется из системы, при этом она переходит в режим ожидания очередного клиента. При условии *КОРРЕКТНЫЙ ПАРОЛЬ* выполняются действия по обеспечению требуемого сервиса (последовательное включение процессов 1.2 и 1.3) и удалению кредитной карты, и затем переход в режим ожидания очередного клиента.

Рис.6.2. STD для банковской задачи

При построении STD рекомендуется следовать нижеперечисленным правилам:

- строить STD на как можно более высоком уровне детализации DFD;
- строить как можно более простые STD;
- по возможности детализировать STD;

- использовать те же принципы именований состояний, событий и действий, что и при именовании процессов и потоков.

Применяются два способа построения STD. Первый способ заключается в идентификации всех возможных состояний и дальнейшем исследовании всех не бессмысленных связей (переходов) между ними. По второму способу сначала строится начальное состояние, затем следующие за ним и т.д. Результат (оба способа) - предварительная STD, для которой затем осуществляется контроль состоятельности, заключающийся в ответе на следующие вопросы:

- все ли состояния определены и имеют уникальное имя?
- все ли состояния достижимы?
- все ли состояния имеют выход?
- (для каждого состояния) реагирует ли система соответствующим образом на все возможные условия (особенно на ненормальные)?
- все ли входные (выходные) потоки управляющего процесса отражены в условиях (действиях) на STD?

Таблица 6.1

ТЕКУЩЕЕ СОСТОЯНИЕ	**УСЛОВИЕ**	**ДЕЙСТВИЕ**	**СЛЕДУЮЩЕЕ СОСТОЯНИЕ**
начальное состояние	активируется каждый раз	-	ОЖИДАНИЕ
ОЖИДАНИЕ	введенная кредитная карта	получить пароль	ОБРАБОТКА
ОБРАБОТКА	некорректный пароль	удалить кредитную карту	ОЖИДАНИЕ
ОБРАБОТКА	корректный пароль	обеспечить требуемый сервис удалить кред. карту	ОЖИДАНИЕ

В ситуации, когда число состояний и/или переходов велико, для проектирования спецификаций управления могут использоваться **таблицы** и **матрицы переходов состояний**. Обе эти нотации позволяют зафиксировать ту же самую информацию, что и диаграммы переходов состояний, но в другом формате. В качестве примера таблицы переходов состояний приведена таблица 6.1, соответствующая рассмотренной выше диаграмме переходов состояний (рис. 6.2). Первая колонка таблицы содержит список всех состояний проектируемой системы, во второй колонке для каждого состояния приведены все условия, вызывающие переходы в другие состояния, а в третьей колонке - совершаемые при этих переходах действия. Четвертая колонка содержит соответствующие имена состояний, в которые осуществляется переход из рассматриваемого состояния при выполнении определенного условия.

Матрица переходов состояний содержит по вертикали перечень состояний системы, а по горизонтали список условий. Каждый ее элемент содержит список действий, а также имя состояния, в которое осуществляется переход. Используется и другой вариант данной нотации: по вертикали указываются состояния, из которых осуществляется переход, а по горизонтали - состояния, в которые осуществляется переход. При этом каждый элемент матрицы содержит соответствующие условия и действия, обеспечивающие переход из “вертикального” состояния в “горизонтальное”.

ГЛАВА 7

СРЕДСТВА СТРУКТУРНОГО ПРОЕКТИРОВАНИЯ

В предыдущих главах были рассмотрены средства структурного системного анализа, применение которых позволяет построить **модель требований** (или **логическую модель**) системы, состоящую из множества взаимоувязанных диаграмм, текстов и словаря данных. Используемые на этом этапе диаграммные техники включают DFD, ERD, STD и спецификации процессов. Модель требований описывает, что должна делать проектируемая система без ссылок на то, как это достигается.

Проектирование - фаза ЖЦ, на которой вырабатывается, как реализуются требования пользователя, которые порождены и зафиксированы на фазе анализа. На этом этапе осуществляется построение **модели реализации** (или **физической модели**), демонстрирующей, как система будет удовлетворять предъявленным к ней требованиями (без технических подробностей). Модель реализации является расширением модели требований и состоит из взаимоувязанных диаграмм (DFD, STD, ERD, структурные карты), текстов и словаря данных. Фактически структурное проектирование является мостом между структурным анализом и реализацией.

Техника **структурных карт** (схем) используется на фазе проектирования для того, чтобы продемонстрировать, каким образом системные требования будут отражаться комбинацией программных структур. При этом наиболее часто применяются две техники: структурные карты Константайна (Constantine), предназначенные для описания отношений между модулями, и структурные карты Джексона (Jackson), предназначенные для описания внутренней структуры модулей.

7.1. Структурные карты Константайна

Базовыми строительными блоками программной системы являются **модули**. Все виды модулей в любом языке программирования имеют ряд общих свойств, нижеперечисленные из которых существенны при структурном проектировании:

1. модуль состоит из множества операторов языка программирования, записанных последовательно;
2. модуль имеет имя, по которому к нему можно ссылаться как к единому фрагменту;
3. модуль может принимать и/или передавать данные как параметры в вызывающей последовательности или связывать данные через фиксированные ячейки или общие области.

Структурные карты Константайна являются моделью отношений иерархии между программными модулями. Узлы структурных карт соответствуют модулям и областям данных, потоки изображают межмодульные вызовы. При этом циклические и условные вызовы модулей моделируются специальными узлами, поэтому потоки должны быть изображены проходящими через эти специальные узлы. Межмодульные связи по данным и управлению также моделируются специальными узлами, привязанными к потокам (т.е. к вызовам модулей), стрелками указываются направления потоков и связей. Фундаментальные элементы структурных карт в соответствии со стандартами IBM, ISO и ANSI приведены на рис. 7.1.

Базовым элементом структурной карты является модуль. Возможно использовать различные типы модулей (см. рис. 7.2):

1. Собственно ***МОДУЛЬ***. Используется для представления обрабатывающего фрагмента для его локализации на диаграмме.
2. ***ПОДСИСТЕМА***. Ранее определенный модуль, детализированный посредством декомпозиции ранее определенных диаграмм. Может повторно использоваться любое число раз на любых структурных картах.
3. ***БИБЛИОТЕКА***. Отличается от подсистемы тем, что определена вне проекта данной системы.
4. ***ОБЛАСТЬ ДАННЫХ***. Используется для указания модулей, содержащих исключительно области глобальных/распределенных данных.

При построении структурных карт добавление модулей и увязывание их вместе осуществляется с использований потоков, демонстрирующих иерархию вызовов. Типы используемых при этом потоков приведены на рис. 7.3. При последовательном вызове модули вызываются в порядке их следования. При параллельном вызове модули могут вызываться в любом порядке или одновременно (параллельно).

Для моделирования условных и циклических вызовов применяются следующие узлы (рис. 7.4):

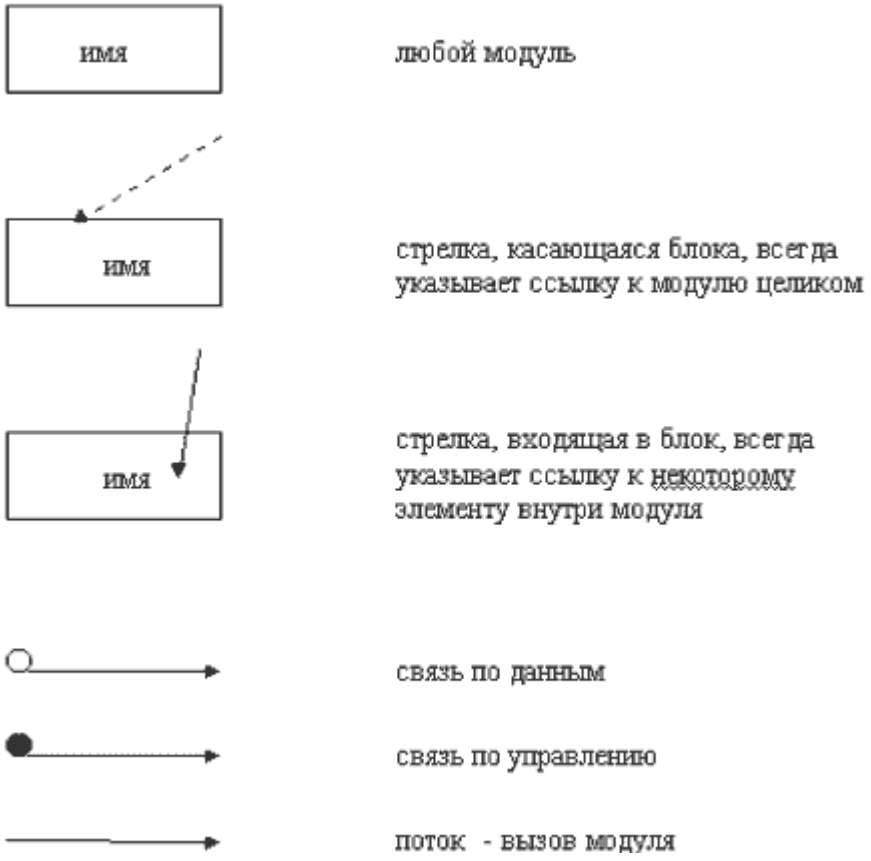


Рис. 7.1. Элементы структурных карт

Условный узел используется для условного вызова модуля-потомка. Он изображается с помощью ромба, потоки - альтернативные вызовы изображаются выходящими из него. Таким образом, если из ромба выходят два потока, это соответствует конструкции *IF-THEN-ELSE*, если один поток - конструкции *IF-THEN*.

Итерационный узел используется для того, чтобы показать, что модуль-наследник вызывается в цикле. Он изображается полуокружностью со стрелкой с выходящими из него потоками.

Если необходимо показать, что подчиненный модуль не вызывается повторно при активации главного, это осуществляется указанием цифры "1" в главном модуле напротив потока - вызова наследника.

Рис. 7.2. Типы модулей

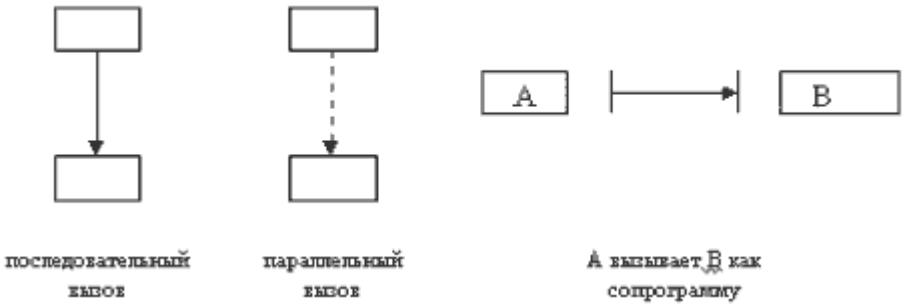


Рис.7.3. Типы вызовов модулей

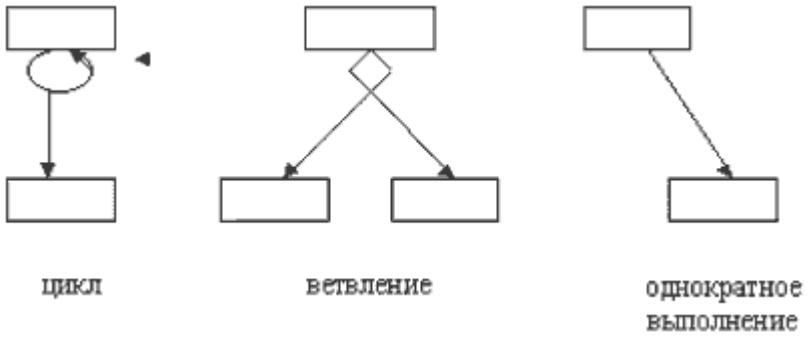


Рис. 7.4. Условные и циклические вызовы модулей

Связи по данным и управлению между модулями (передаваемые как параметры) раскрываются аннотированием потоков-вызовов (рис. 7.5). Стрелками отмечаются направления связей.

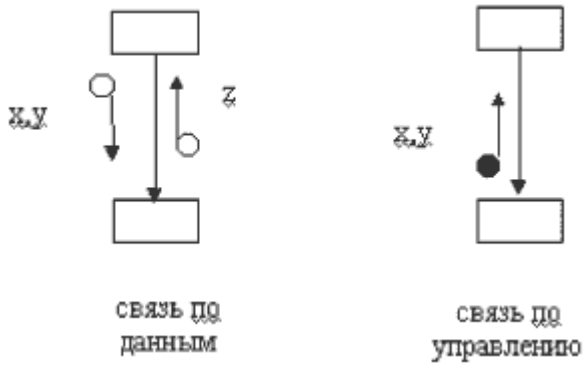


Рис. 7.5. Связи по данным и управлению

Пример структурной карты, описывающей межмодульные отношения в рассмотренном ранее фрагменте банковской системы, приведен на рис. 7.6.

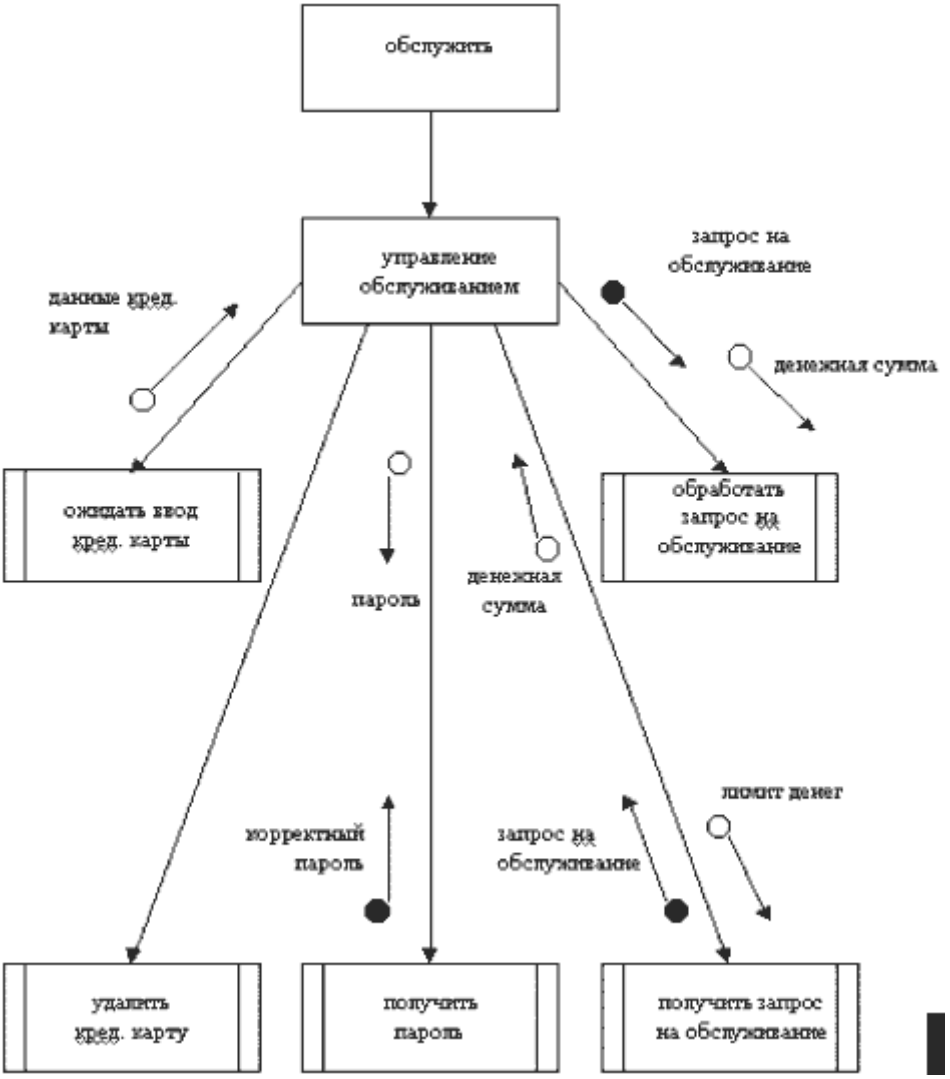


Рис. 7.6. Пример структурной карты Константайна

7.2. Структурные карты Джексона

Техника **структурных карт Джексона** основана на методологии структурного программирования Джексона и заключается в продуцировании диаграмм (структурных карт) для графического иллюстрирования внутримодульных (а иногда и межмодульных) связей и документирования проекта архитектуры системы ПО. При этом техника позволяет осуществлять проектирование нижнего уровня структуры ПО и на этом этапе является близкой к традиционным блок-схемам.

По аналогии со структурными картами Константайна диаграмма Джексона может включать объекты следующих типов:

1. ***СТРУКТУРНЫЙ*** блок (базовая компонента методологии) представляет частную функцию или блок кодов с одним входом и одним выходом.
2. ***ПРОЦЕДУРНЫЙ*** блок является специальным видом структурного блока, представляющим вызов ранее определенной процедуры.
3. ***БИБЛИОТЕЧНЫЙ*** блок аналогичен процедурному и представляет вызов библиотечного модуля.

Для взаимоувязывания блоков используются связи следующих типов:

- последовательная связь, обеспечивающая последовательное выполнение слева направо;
- параллельная связь, обеспечивающая одновременное выполнение блоков;
- условная связь, обеспечивающая выбор одной из альтернатив;

- итерационная связь, обеспечивающая выполнение блока в цикле.

Пример структурной карты Джексона приведен на рис. 7.7.

7.3. Характеристики хорошей модели реализации

Структурные карты сами по себе ничего не говорят о качестве модели (проекта) реализации, так как являются всего лишь инструментом для демонстрации структуры системы и составляющих ее модулей, а также их связей друг с другом.

Один из фундаментальных принципов структурного проектирования заключается в том, что большая система должна быть расчленена на обозримые модули. При этом существенным является то, что это расчленение должно быть выполнено таким образом, чтобы модули были как можно более независимы (критерий сцепления - **coupling**), и чтобы каждый модуль выполнял единственную (связанную с общей задачей) функцию (критерий связности - **cohesion**). Кроме этих двух взаимно дополняющих друг друга критериев в структурном проектировании существуют целый ряд других руководящих принципов, которые могут применяться для оценки и улучшения качества проекта на основании структурных карт.

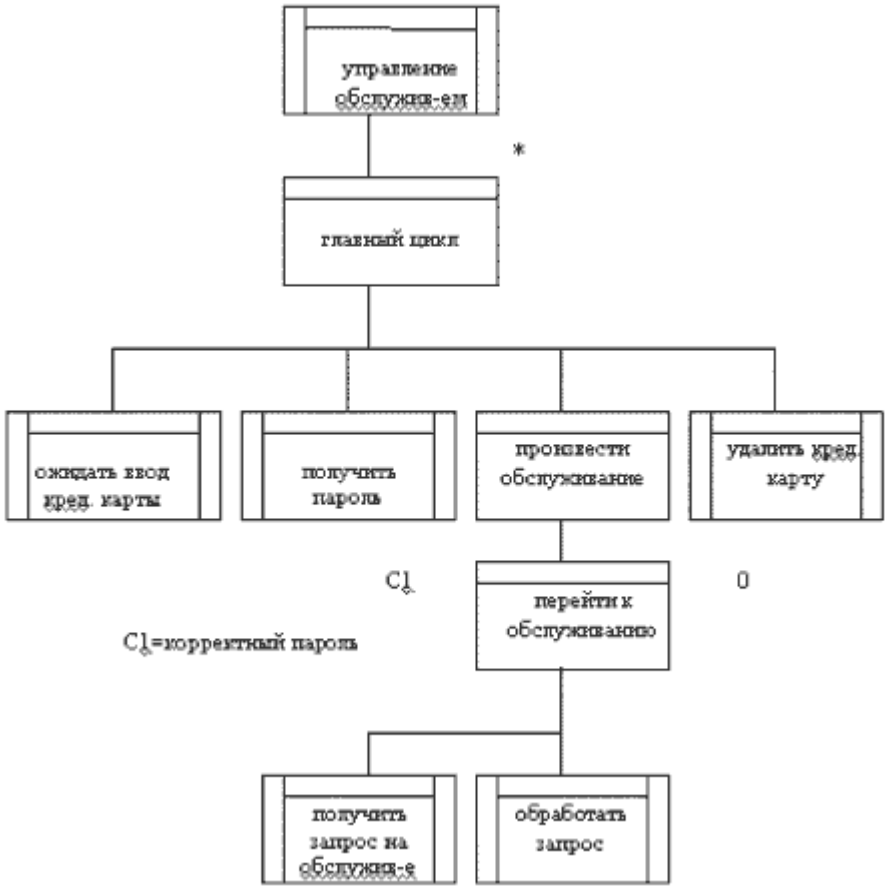


Рис 7.7. Структурная карта Джексона

7.3.1. Сцепление

Одним из способов оценки качества проекта является анализ сцепления модулей. Сцепление является мерой взаимозависимости модулей. В хорошем проекте сцепления должны быть минимизированы, т.е. модули должны быть слабозависимыми (независимыми) настолько, насколько это возможно. Слабое сцепление между модулями служит признаком хорошо спроектированной системы по следующим причинам:

- уменьшение количества соединений между двумя модулями приводит к уменьшению вероятности появления “волнового эффекта” (ошибка в одном модуле влияет на работу других модулей);

- минимизация риска появления “эффекта ряби” (внесение изменений, например, при исправлении ошибки, приводит к появлению новых ошибок), т.к. изменение влияет на минимальное количество модулей;
- при сопровождении модуля отсутствие необходимости беспокоиться о внутренних деталях других модулей;
- упрощение системы для понимания, насколько это возможно.

Слабое сцепление может быть достигнуто за счет комбинирования трех следующих способов действий:

- удаления необязательных связей;
- уменьшения количества необходимых связей;
- упрощением необходимых связей.

Специалистами предлагаются следующие практические рекомендации для ослабления сцепления модулей:

1. Создавайте минимальные по количеству параметров межмодульные связи.
2. Создавайте прямые (а не косвенные) межмодульные связи, поскольку интерфейс между двумя модулями наиболее понятен (и, следовательно, менее сложен), если человек может постигнуть его сразу без предварительной ссылки к некоторым другим информационным объектам.
3. Создавайте локализованные связи (например, значения списка параметров вычисляйте непосредственно перед вызовом модуля).
4. Создавайте явные связи. Красноречивым примером неявной связи является взаимодействие модуля А с модулем В за счет модификации области данных из В: для того, чтобы человек, сопровождающий модуль В понял, за счет чего модифицируется эта область данных, он будет должен проделать огромную работу.
5. Создавайте гибкие связи для облегчения модификаций.

На практике существуют три основных типа сцепления, используемых системными проектировщиками для связи модулей: нормальное сцепление, сцепление по общей области и сцепление по содержимому. С позиций структурного проектирования эти типы являются, соответственно, приемлемым, неприемлемым и запрещенным.

Два модуля А и В являются нормально сцепленными, если

- А вызывает В,
- В возвращает управление А,
- вся информация, передаваемая между А и В, представляется значениями параметров при вызове.

Существует три типа нормального сцепления: сцепление по данным, сцепление по образцу, сцепление по управлению. На практике наиболее часто используемым типом сцепления является сцепление по данным (data coupling). Два модуля сцеплены по данным, если они взаимодействуют через передачу параметров и при этом каждый параметр является элементарным информационным объектом. В случае небольшого количества передаваемых параметров сцепление по данным обладает наилучшими характеристиками в соответствии с п.п.1-5.

Два модуля сцеплены по образцу (stamp coupling), если один посылает другому составной информационный объект, т.е. объект, имеющий внутреннюю структуру. Примером составного объекта может быть объект *Данные о клиенте*, включающий в себе поля: *Название организации, Почтовый адрес, Номер счета* и т.п.

Два модуля сцеплены по управлению (control coupling), если один посылает другому информационный объект - флаг, предназначенный для управления его внутренней логикой. Существует два типа флагов - описательный и управляющий. Описательный флаг обычно

описывает ситуацию, которая произошла, и именуются с использованием существительных и прилагательных: *Конец файла, Введенная кредитная карта*. Управляющий флаг используется для декларирования определенных действий в вызываемом модуле и именуется с использованием глаголов: *Читать следующую запись, Установить в начало*. В общем случае управляющие флаги усиливают сцепление и, следовательно, ухудшают качество проекта.

Как уже отмечалось, вышеперечисленные три типа нормального сцепления в разной степени поддерживают суть модульности и являются приемлемыми в структурном проектировании. Ниже определяются два вида сцепления, которые выходят за пределы хорошей модульности.

Два модуля являются сцепленными по общей области (common coupling), если они ссылаются к одной и той же области глобальных данных. Сцепление по общей области является плохим по следующим причинам. Во-первых, ошибка в любом модуле, использующем глобальную область, может неожиданно проявить себя в любом другом модуле, также использующем эту глобальную область, поскольку эти глобальные данные не находятся “под защитой” модуля. Во-вторых, модули, ссылающиеся к глобальным данным, обычно используют точные имена (в отличие от модулей, которые вызываются с использованием параметров). Следовательно, гибкость модулей, сцепленных глобально, намного хуже, чем гибкость нормально сцепленных модулей. В-третьих, программы с большим количеством глобальных данных чрезвычайно трудны для понимания сопровождающим программистом, поскольку трудно определить, какие данные используются отдельным модулем.

Два модуля являются сцепленными по содержимому (content coupling), если один ссылается внутрь другого любым способом, например, если один модуль передает управление или выполняет переход в другой модуль, если один модуль ссылается (или изменяет) значения информационных объектов в другом модуле или если один модуль изменяет код другого модуля. Такое сцепление делает абсурдной концепцию модулей как черных ящиков, поскольку оно вынуждает один модуль знать о точном содержании и реализации другого модуля. Вообще говоря, только ассемблер позволяет проектировщикам применять данный вид сцепления.

Таблица 7.1

Тип сцепления	**Устойчивость к волновому эффекту**	**Модифи-цируемость**	**Понят-ность**	**Используемость в других системах**
data coupling	*	хорошая	хорошая	хорошая
stamp coupling	*	средняя	средняя	средняя
control coupling	средняя	плохая	плохая	плохая
common coupling	плохая	средняя	плохая	плохая
content coupling	плохая	плохая	плохая	плохая

* Зависит от количества параметров интерфейса.

Необходимо отметить, что любые два модуля могут быть сцеплены более чем одним способом. В этом случае тип их сцепления определяется худшим типом сцепления. Например, если два модуля сцеплены по образцу и общей области, то они характеризуются как сцепленные по общей области. Они по-прежнему сцеплены по образцу, но это сцепление выше, чем сцепление по общей области.

В таблице 7.1 приведены конкретные характеристики каждого типа сцепления.

7.3.2. Связность

Сцепление является лишь одним из критериев оценки качества разбиения системы на части: он оценивает, насколько хорошо модули отделены друг от друга. Другим критерием оценки качества расчленения системы является критерий связности, контролирующий, как действия в одном модуле связаны друг с другом. Фактически сцепление и связность являются двумя взаимозависимыми способами измерения расчленения системы на части: связность модуля часто определяет качество его сцепления с другими модулями.

Связность - это мера прочности соединения функциональных и информационных объектов внутри одного модуля. Размещение сильно связанных объектов в одном и том же модуле уменьшает межмодульные взаимосвязи и взаимовлияния. Специалисты выделяют следующие уровни связности: функциональная, последовательная, информационная, процедурная, временная, логическая и случайная. Рассмотрим эти уровни более подробно.

Функционально связный модуль содержит объекты, предназначенные для выполнения одной и только задачи, например: *Расчет заработанной платы, Считывание данных кредитной карты.* Каждый из этих модулей имеет одну четко определенную цель, при его вызове выполняется только одна задача (при этом она выполняется полностью без исполнения любого другого дополнительного действия).

Модуль имеет последовательную связность, если его объекты охватывают подзадачи, для которых выходные данные одной из подзадач служат входными данными для следующей, например: *Открыть файл - Прочитать запись - Закрыть файл.*

Информационно связный модуль содержит объекты, использующие одни и те же входные или выходные данные. Предположим, что мы хотим выяснить некоторые сведения о книге, зная ее ISBN: название книги, ее автора и цену. Эти три подзадачи являются связанными потому, что все они работают с одним и тем же входным информационным объектом - ISBN, который и делает этот модуль информационно связным.

Процедурно связный модуль является модулем, объекты которого включены в различные (и возможно несвязные) подзадачи, в которых управление переходит от каждой подзадачи к последующей (отметим, что в последовательно связном модуле данные, а не управление, переходили от одной подзадачи к последующей). В качестве примера приведем следующий перечень шагов в некотором процедурно связном модуле:

1. Сделать зарядку

2. Принять душ

3. Сварить кофе

4. Одеться

5. Отправиться на службу

Отправиться на службу - это последний шаг, внесенный в список этого "модуля". Но, например, действия *Прочитать газету* или *Сходить в магазин* могут быть в равной степени пригодными кандидатами для шага 5, поскольку шаги в этом списке связаны только тем, что они происходят в данном порядке в течение конкретного дня.

Временно связным является модуль, объекты которого включены в подзадачи, связанные временем исполнения. Представим себе картину позднего вечера:

1. Почистить зубы

2. Выключить телевизор

3. Выгнать кота в коридор

Эти действия никак не связаны друг с другом, за исключением конкретного времени их выполнения. Все они - часть установившегося режима в конце дня.

Модулем с логической связностью является модуль, объекты которого содействуют решению одной общей подзадачи, для которой эти объекты отобраны во внешнем по отношению к модулю мире. Например, собираясь в поездку, можно составить себе следующий список:

1. *Поехать автомобилем*
2. *Поехать поездом*
3. *Поплыть на корабле*
4. *Полететь на самолете*

Что связывает эти действия? Все они являются способами перемещения. Но решающий момент заключается в том, что для любой поездки человек должен выбрать конкретный способ перемещения, т.к. маловероятно, что кто-нибудь будет использовать их все для отдельной поездки.

Таким образом логически связный модуль содержит некоторое количество подзадач (действий) одного и того же общего вида. Для того, чтобы его использовать, необходимо выбрать именно ту часть (части), которые требуются. Эти различные подзадачи должны обладать одним и только одним интерфейсом с внешним миром. При этом семантика каждого параметра зависит от используемой подзадачи.

Случайно связным является модуль, объекты которого соответствуют подзадачам, незначительно связанным друг с другом:

1. *1.Ремонтировать автомобиль*
2. *Пить пиво*
3. *Смотреть фильм*

Случайно связный модуль подобен логически связному модулю, его объекты не связаны ни потоками данных, ни потоками управления. Однако, подзадачи в логически связном модуле являются по крайней мере одной категории; для случайно связного модуля даже это неверно.

В таблице 7.2 приведены конкретные характеристики каждого уровня связности.

Таблица 7.2

Уровень связности	**Сцепление**	**Модифици-руемость**	**Понятность**	**Сопровож-даемость**
функциональная	хорошее	хорошая	хорошая	хорошая
последовательная	хорошее	хорошая	близкая к хорошей	хорошая
информационная	среднее	средняя	средняя	средняя
процедурная	переменная	переменная	переменная	плохая
временная	плохое	средняя	средняя	плохая
логическая	плохое	плохая	плохая	плохая
случайная	плохое	плохая	плохая	плохая

Таким образом, связность является мерой функциональной зависимости объектов (исполняемых операторов, областей данных и т.д.) внутри одного модуля. В хорошем проекте связность каждого модуля является высокой (последовательность введенных выше определений уровней связности

соответствует направлению от лучшей связности к худшей). Вместе со сцеплением, связность является одним из лучших критериев оценки качества проекта.

7.3.3. Другие принципы проектирования

Очевидно, что для оценки качества проектируемой системы критериев сцепления и связности недостаточно. Например, если бы мы осуществляли оценку только по критерию сцепления, мы бы всегда получали системы, состоящие из одного модуля. Связность этого единственного модуля также была бы вполне приемлемой. Ниже кратко рассматриваются другие принципы проектирования, позволяющие получать качественные системы.

1) **Принцип факторизации**. Под факторизацией понимается выделение подзадачи, реализуемой некоторым модулем, в новый самостоятельный модуль. Это может быть сделано по следующим резонам:

- чтобы уменьшить размеры модуля;
- чтобы обеспечить возможность и преимущества классического проектирования сверху-вниз, позволяющего строить систему более легкой для понимания и облегчающего модификации системы;
- чтобы избегать дублирования подзадачи, выполняемой более чем одним модулем;
- чтобы отделить собственно вычисления от управления (вызовы и принятия решения);
- чтобы обеспечить более широкую пригодность модулей для их использования в различных частях системы;
- чтобы упростить реализацию.

2) **Принцип единства решения**. Процесс любого решения состоит из двух частей: распознавания того, какое действие выбрать, и исполнения этого действия. Поскольку эти две составляющие решения очень различны, информационные объекты, используемые при распознавании и исполнении также могут существенно различаться и, следовательно, могут быть недоступными в одном модуле. Такая ситуация получила название расщепления решения (decision split). Сильное расщепление решения (хотя иногда расщепления не удается избежать) обычно является симптомом плохой организации модулей. Исполнительная часть решения должна располагаться как можно ближе к распознавательной части, чтобы распознанной информации не пришлось долго “блуждать” для того, чтобы быть обработанной.

3) **Обработка ошибок**. Сообщения об ошибках целесообразно формировать и визуализировать в модуле, который ошибку обнаруживает (и, следовательно, “знает”, что это за ошибка). Тексты сообщений рекомендуется хранить вместе по следующим резонам:

- При таком походе легче сохранять согласованность формулировок и форматов сообщений. Представьте себе состояние пользователя, когда он получает различные сообщения для одной и той же ошибки, когда она встречается в разных частях системы.
- Появляется возможность хранить тексты сообщений в отдельном файле, а не внутри кода модуля.
- Легче избежать дублирования сообщений.
- Облегчается модификация сообщений (включая их перевод на другой язык).

4) **Принцип отсутствия памяти**. Когда вызванный модуль возвращает управление вызвавшему его модулю после выполнения своей функции, этот модуль “умирает”, оставляя после себя только результат. При повторном вызове он делает свою работу так, как будто бы он родился впервые. Модуль не помнит, что происходило в его предыдущих жизнях. Однако, существует тип модуля, который знает о своем прошлом благодаря так называемой памяти состояния. Память состояния (state memory) - это информационный объект внутри модуля, который продолжает существовать неизмененным между двумя вызовами модуля. Работа модуля с памятью состояния в общем случае непредсказуема, это означает, что хотя модуль вызывался с одинаковыми фактическими

параметрами, исполняться он может по-разному, и результаты его работы при разных вызовах могут быть различными. Сопровождение такого модуля резко усложняется.

5) **Инициализация и завершение**. Как правило, модули инициализации и завершения являются трудными для сопровождения из-за их плохой (временной) связности и сильного сцепления. Общая рекомендация по решению этой проблемы - инициализацию каждой функции желательно выполнять как можно позже, а действия по завершению каждой функции должны производиться как можно раньше. И, конечно, необходимо проводить инициализацию и завершение как можно ближе к тому, что инициализируется или завершается.

6) **Компромисс между ограниченностью и обобщенностью**. Ограниченный модуль обладает по крайней мере одной из следующих характеристик:

- Он выполняет излишне специфическую работу. Например, модуль, вычисляющий среднюю ежемесячную температуру для месяца продолжительностью в 30 дней, является ограниченным; на самом деле необходим модуль, который генерировал бы среднюю температуру для месяца любой продолжительности. Продолжительность месяца могла бы передаваться ему как параметр, а не быть жестко установленной внутри.
- Он имеет дело с ограниченными значениями данных, их типами и структурами (например, модуль, предполагающий, что человек не может быть собственником более одного автомобиля).
- Он включает в себя условия о месте и способе его использования.

  Противоположная крайность ограниченному модулю - сверхобобщенный модуль, обладающий по крайней мере одной из следующих характеристик:

- Он выполняет нелепо обширную работу. Примером является модуль, формирующий расписание игр чемпионата по футболу как по Грегорианскому, так и по Юлианскому календарю.
- Он имеет дело с слишком избыточными типами данных, их значениями и структурами. Например, использование числа типа REAL вместо INTEGER для того, чтобы следить за количеством болтов на складе, было бы чрезмерным обощением.
- Он принимает в качестве параметров данные, которые никогда не изменятся. Так модуль, которому передается количество дней в неделе, является определенно сверхобобщенным, а также до смешного нелепым.

7) **Принцип минимизации избыточности**. Если любой факт, условие или реализационное решение явно проявляются в более чем одном модуле, то усилия по сопровождению, состоящие из нахождения всех случаев этого факта и их изменения, увеличиваются. Также возникает опасность того, что человек, сопровождающий такую систему, забудет изменить один из дублей.

8) **Нагрузка по входу и выходу**. Под нагрузкой модуля по входу понимается количество непосредственных вызывающих его модулей. Соответственно, нагрузка модуля по выходу - это количество непосредственно подчиненных ему модулей. По уже упоминавшимся выше причинам нагрузка по выходу не должна превышать 6-7 модулей. Высокая нагрузка по входу требует от модуля хорошей связности.

7.4. Транзакционный и трансформационный анализ или как получить структурные карты из диаграмм потоков данных

В этом параграфе кратко обсуждаются три шага, регламентирующих процесс получения структурных карт из диаграмм потоков данных и, таким образом, обеспечивающих естественный переход от модели требований к модели реализации. Этими шагами являются:

- разбиение иерархии диаграмм потоков данных на относительно легко поддающиеся обработке единицы средствами транзакционного анализа;

- конвертирование каждой единицы в “хорошую” структурную карту средствами трансформационного анализа;
- обратное соединение разделенных единиц в полную модель реализации.

Определим **транзакцию** как объект, содержащий следующие пять компонент:

СОБЫТИЕ в системе или ее внешнем окружении

СИГНАЛ к системе

ДЕЙСТВИЕ системы

ОТКЛИК от системы

ВЛИЯНИЕ на систему или ее окружение

Например, в системе управления космическим кораблем транзакция может быть представлена как совокупность следующих элементов:

СОБЫТИЕ Диспетчер замечает, что корабль движется слишком быстро

СИГНАЛ Замедлить скорость корабля на 210 м/сек

ДЕЙСТВИЕ Определить какой из тормозных ракетных двигателей включить и на какое время

ОТКЛИК Сигнал тормозному двигателю ракеты для его включения на 4.8 сек

ВЛИЯНИЕ Корабль замедлил скорость на 210 м/сек

Следующий пример относится к компании, устанавливающей газовое отопление на дачных участках:

СОБЫТИЕ Владелец дачи Сергей Чеченин устал пилить и колоть дрова для камина и решил установить газовое отопление

СИГНАЛ Информация о г-не Чеченине и его даче, а также дата начала обслуживания

ДЕЙСТВИЕ Добавить данные о г-не Чеченине в базу данных клиентов

ОТКЛИК Разрешение на предоставление услуг

ВЛИЯНИЕ В освободившееся от дровяных работ время г-н Чеченин проектирует систему автоматизации газовой компании

Отметим, что подобные отдельные транзакции относятся к классам, определяемым как классы транзакционного типа. Например, три транзакции "*ДОБАВИТЬ ЧЕЧЕНИНА*", "*ДОБАВИТЬ ИВАНОВА*" и "*ДОБАВИТЬ ПЕТРОВА*" являются примерами одного транзакционного типа, который может быть назван *"ДОБАВИТЬ НОВОГО КЛИЕНТА*".

Транзакционные типы могут быть идентифицированы по модели требований системы на основании типов отдельных событий, которые получает эта система. Если модель требований включает в себя событийную модель, то транзакционные типы могут быть без труда выявлены, например из иерархии диаграмм переходов состояний.

Таким образом, транзакционный анализ является методом идентификации типов транзакций системы для их дальнейшего использования в качестве единиц проектирования.

В свою очередь, трансформационный анализ или проектирование трасформационного (преобразующего) центра является методом преобразования в структурную карту каждой из частей иерархии диаграмм потоков данных, которые мы выделили при транзакционном анализе. Трансформационный анализ включает следующие четыре шага:

- выявление центрального преобразования - фрагмента DFD, который содержит концептуальные функции системы и не зависит от особенностей реализации системных входов и выходов;
- конвертирование DFD в предварительную структурную карту;
- доработка этой структурной карты средствами структурного проектирования;
- проверка окончательной структурной карты на соответствие требованиям первоначальной DFD.

Один из способов нахождения центрального преобразования состоит в удалении центростремительных и центробежных потоков, которое осуществляется с помощью следующих трех шагов. Во-первых, необходимо проследить каждую центростремительную ветвь от края DFD к центру и пометить поток данных, который представляет системный вход в наиболее концептуальной форме. Другими словами, необходимо пометить стадию прохождения потока, на которой вычленены необходимые для обработки данные, которые еще не использовались в реальном процессе. Во-вторых, необходимо проследить каждую центробежную ветвь от края DFD к центру и пометить поток данных, который представляет выход в наиболее концептуальной форме. Другими словами, необходимо пометить стадию прохождения потока, на которой выходные данные были только что получены, но еще не отформатированы. В-третьих, необходимо соединить все отметки замкнутой кривой. Процессы внутри этой кривой и будут составлять центральное преобразование.

Напомним, что цель трансформационного анализа состоит в том, чтобы превратить DFD для одного типа транзакций в структурную карту. Основное различие между DFD и структурной картой состоит в том, что на структурной карте изображается главный модуль системы. В DFD нет главных модулей. Процессы DFD подобны довольным своей жизнью рабочим в идеалистической коммуне, каждый из которых беспокоится за свою собственную работу и передает свой продукт следующему рабочему на линии.

Для решения данной задачи поступают следующим образом. Один процесс в области центрального преобразования DFD может выделяться как потенциальный главный модуль. При этом признаком такого выделения может, например, являться то, что он выполняет незначительную обработку данных, но при этом координирует работу других процессов. Далее для того, чтобы улучшить схему, следует, по необходимости, выполнить следующее:

- Добавить модули чтения и записи для доступных источников, стоков и файлов.
- Добавить модули обработки ошибок.
- Добавить детали инициализации и завершения, если требуется.
- Проконтролировать, что все модули имеют имена в соответствии с их положением в иерархии.
- Добавить все связи по данным и управлению, которые являются необходимыми на структурной схеме, но не на DFD, например, информация: "конец потока".
- Проверить все критерии структурного проектирования и улучшить проект в соответствии с этими критериями.

Следующий шаг трансформационного анализа является критическим. Необходимо убедиться, что структурная карта, извлеченная из DFD, корректно реализует требования системы. Для этой цели можно, например, обсудить построенную структурную карту с аналитиком - автором соответствующей DFD.

Наконец, на последнем этапе из построенного набора структурных карт необходимо составить полную картину проектируемой системы. Одним из способов решения данной задачи является

связывание структурных карт путем введения общего для всех них главного модуля, координирующего работу каждой их них.

ЧАСТЬ 2

МЕТОДОЛОГИИ СТРУКТУРНОГО СИСТЕМНОГО АНАЛИЗА И ПРОЕКТИРОВАНИЯ

Во второй части книги рассматривается, как исследованные в части 1 средства и техники структурного системного анализа используются соответствующими методологиями для построения моделей различных предметных областей, дается обзор методологий структурного системного анализа и проектирования, а также кратко рассматриваются основные особенности наиболее часто используемых методологий.

Роль методологии заключается в регламентации основ разработки сложных систем. Она описывает последовательность шагов, модели и подходы, тщательное следование которым приведет к хорошо работающим системам. Хотя методологии, вообще говоря, не гарантируют качества построенных систем, тем не менее они помогают охватить и учесть все важные этапы, шаги и моменты разработки, помогают справиться с проблемами размерности, и в конечном итоге оценить продвижение вперед. Более того, методологии обеспечивают организационную поддержку, позволяющую большим коллективам разработчиков функционировать скоординированным образом.

В главе 8 приводится классификация структурных методологий по отношению к школам, по порядку построения модели и по типу целевых систем.

В главе 9 приведены основные принципы, возможности и особенности наиболее часто используемых методологий структурного системного анализа и проектирования (Йодан/ДеМарко, Гейн-Сарсон, SADT, Джексон, Варнье-Орр, Мартин), предложен сравнительный анализ наиболее часто применяемых методологий.

Глава 10 кратко описывает архитектуру современной системы и ее влияние на изменения в методологиях анализа и проектирования.

ГЛАВА 8

КЛАССИФИКАЦИЯ СТРУКТУРНЫХ МЕТОДОЛОГИЙ

Методология структурного анализа и проектирования ПО определяет руководящие указания для оценки и выбора проекта разрабатываемого ПО, шаги работы, которые должны быть выполнены, их последовательность, правила распределения и назначения операций и методов.

В настоящее время успешно используются практически все известные методологии структурного анализа и проектирования, однако наибольшее распространение получили методологии SADT (Structured Analysis and Design Technique), структурного системного анализа Гейна-Сарсона (Gane-Sarson), структурного анализа и проектирования Йодана/Де Марко (Yourdon/De Marko), развития систем Джексона (Jackson), развития структурных систем Варнье-Орра (Warnier-Orr), анализа и

проектирования систем реального времени Уорда-Меллора (Ward-Mellor) и Хатли (Hatley), информационного моделирования Мартина (Martin).

Перечисленные структурные методологии жестко регламентируют фазы анализа требований и проектирования спецификаций и отражают подход к разработке ПО с позиций рецептов "кулинарной книги". Спецификации требований включают особенности ПО и его прогнозируемые характеристики, проекты пользовательских интерфейсов (меню, экраны и формы), критерии работоспособности ПО, программное и аппаратное окружение. Полученный документ спецификаций требований в дальнейшем преобразуется в проект архитектуры, детализирующий предполагаемую реализацию ПО. Проект архитектуры идентифицирует главные модули, маршруты связи по данным и управлению между модулями, основные подпрограммы внутри каждого модуля, структуры данных, спецификации форматов входных и выходных файлов. Для ключевых процессов проектные спецификации часто включают детали алгоритмов на языке проектирования миниспецификаций. Обычно предлагается следующая последовательность шагов при проектировании спецификаций:

1. *разделение проекта на 10-50 модулей;*
2. *организация иерархии модулей;*
3. *определение маршрутов данных между модулями;*
4. *определение форматов внешних файлов;*
5. *определение способов доступа к внешним файлам;*
6. *определение структур данных;*
7. *проектирование ключевых алгоритмов;*
8. *определение подпрограмм внутри каждого модуля.*

Структурные методологии предлагают методику трансляции проектных спецификаций в модель реализации, в дальнейшем используемую при кодогенерации. Кодогенерация предполагает наличие кодовых стандартов, специфицирующих формат заголовков подпрограмм, ступенчатый вид вложенных блоков, номенклатуру для спецификации переменных и имен подпрограмм и т.п.

Несмотря на достаточно широкий спектр используемых методов и диаграммных техник, большинство методологий базируется на следующей "классической" совокупности:

- диаграммы потоков данных в нотации Йодана/Де Марко или Гейна-Сарсона, обеспечивающие анализ требований и функциональное проектирование информационных систем;
- расширения Хатли и Уорда-Меллора для проектирования систем реального времени, основанные на диаграммах переходов состояний, таблицах и деревьях решений, картах и схемах потоков управления;
- диаграммы "сущность-связь" (в нотации Чена или Баркера) или скобочные диаграммы Варнье-Орра для проектирования структур данных, схем БД, форматов файлов как части всего проекта;
- структурные карты Джексона и/или Константайна для проектирования межмодульных взаимодействий и внутренней структуры модулей, позволяющие развить модель анализа, построенную на базе вышеперечисленных средств, до модели реализации.

Современные структурные методологии анализа и проектирования классифицируются по следующим признакам:

- по отношению к школам - **Software Engineering** (SE) и **Information Engineering** (IE);
- по порядку построения модели - **процедурно-ориентированные, ориентированные на данные и информационно-ориентированные**;
- по типу целевых систем - для **систем реального времени** (СРВ) и для **информационных систем** (ИС).

SE является нисходящим поэтапным подходом к разработке ПО, начинающейся с общего взгляда на его функционирование. Затем производится декомпозиция на подфункции, и процесс

повторяется для подфункций до тех пор, пока они не станут достаточно малы для их реализации кодированием. В результате получается иерархическая, структурированная, модульная программа. SE является универсальной дисциплиной разработки ПО, успешно применяющейся как при разработке систем реального времени, так и при разработке информационных систем. IE - более новая дисциплина. С одной стороны, она имеет более широкую область применения, чем SE: IE является дисциплиной построения систем вообще, а не только систем ПО, и включает этапы более высокого уровня (например, стратегическое планирование), однако на этапе проектирования систем ПО эти дисциплины аналогичны. С другой стороны, IE - более узкая дисциплина, чем SE, т.к. IE используется только для построения информационных систем, а SE - для всех типов систем.

Разработка ПО основана на модели ***ВХОД-ОБРАБОТКА-ВЫХОД***: данные входят в систему, обрабатываются или преобразуются и выходят из системы. Такая модель используется во всех структурных методологиях. При этом важен порядок построения модели. Традиционный процедурно-ориентированный подход регламентирует первичность проектирования функциональных компонент по отношению к проектированию структур данных: требования к данным раскрываются через функциональные требования. При подходе, ориентированном на данные, вход и выход являются наиболее важными - структуры данных определяются первыми, а процедурные компоненты являются производными от данных. Информационно-ориентированный подход, как часть IE-дисциплины, отличается от подхода, ориентированного на данные, тем, что позволяет работать с неиерархическими структурами данных.

Таблица 8.1

Информационные системы	**Системы реального времени**
Управляемы данными	Управляемы событиями
Сложные структуры данных	Простые структуры данных
Большой объем входных данных	Малое количество входных данных
Интенсивный ввод/вывод	Интенсивные вычисления
Машинная независимость	Машинная зависимость

Основная особенность систем реального времени заключается в том, что они контролируют и контролируются внешними событиями; реагирование на эти события во времени - основная и первоочередная функция таких систем. Главные отличия информационных систем от систем реального времени приведены в таблице 8.1, средствами поддержки этих особенностей и различаются соответствующие структурные методологии.

Таблица 8.2

Название	**Частота использования, проценты**	**Школа**	**Порядок построения**	**Тип целевых систем**
Йодан-Де Марко	36,5	SE	Процедурно-ориентированная	ИС, СРВ
Гейн-Сарсон	20,2	SE	процедурно-ориентированная	ИС, СРВ
Константайн	10,6	SE	процедурно-ориентированная	ИС, СРВ
Джексон	7,7	SE	ориентированная на данные	ИС, СРВ
Варнье-Орр	5,8	SE	ориентированная на данные	ИС

Мартин	22,1	IE	информационно-ориентированная	ИС
SADT	3,3	IE	варианты использования: 1)проц.-ориент. 2)ор. на данные	ИС
Stradis	1,9	IE	процедурно-ориентированная	ИС

Таблица 8.2 классифицирует наиболее часто используемые методологии в соответствии с вышеперечисленными признаками (данные по частоте использования получены на основе анализа информации по 127 CASE-пакетам).

Во всех перечисленных методологиях проектирования информационных систем в различных комбинациях используются приведенные в таблице 8.3 техники структурных диаграмм.

Необходимо отметить, что для проектирования систем реального времени используются специальные типы структурных диаграмм: диаграммы потоков управления, диаграммы переходов состояний, контекстные графы, матрицы состояний/событий, таблицы решений и др.

Таблица 8.3

Название	**Процедуры**	**Данные**
1. Средства анализа		
- диаграммы потоков данных	+	
- диаграммы потоков управления	+	
- таблицы, деревья решений	+	
- матрицы	+	+
- диаграммы зависимости	+	
- диаграммы декомпозиции	+	
- SADT- диаграммы	+	+
2. Средства проектирования		
- структурные карты	+	
- диаграммы деятельности	+	
- диаграммы Варнье-Орра	+	+
- диаграммы переходов состояний	+	
- языки проектирования спец-ий	+	
- блок-схемы	+	
- схемы экранов		+
- диаграммы "сущность-связь"		+

Однако многие из них являются вариациями структурных диаграмм для проектирования информационных систем. Более того, известные методологии проектирования систем реального времени (в частности, методологии Хатли и Уорда-Меллора) базируются на перечисленных методологиях проектирования информационных систем, расширяя их соответствующими диаграммными техниками.

Ниже рассматриваются основные принципы и особенности некоторых из наиболее часто используемых методологий анализа и проектирования.

ГЛАВА 9

ПРИМЕРЫ СТРУКТУРНЫХ МЕТОДОЛОГИЙ

9.1. Методологии структурного анализа Йодана/Де Марко и Гейна-Сарсона

Как уже неоднократно отмечалось, структурный анализ - это систематический пошаговый подход к анализу требований и проектированию спецификаций системы независимо от того, является ли она существующей или создается вновь. Методологии Гейна-Сарсона и Йодана/Де Марко, основанные на идее нисходящей иерархической организации, наиболее ярко демонстрируют этот подход.

Целью рассматриваемых методологий является преобразование общих, неясных знаний о требованиях к системе в точные (насколько это возможно) определения. Обе методологии фокусируют внимание на потоках данных, их главное назначение - создание базированных на графике документов по функциональным требованиям. Методологии поддерживаются традиционными нисходящими методами проектирования спецификаций и обеспечивают один из лучших способов связи между аналитиками, разработчиками и пользователями системы за счет интеграции множества следующих средств:

1. DFD - диаграммы потоков данных. Являются графическими иерархическими спецификациями, описывающими систему с позиций потоков данных. В состав DFD могут входить четыре графических символа, представляющих потоки данных, процессы преобразования входных потоков данных в выходные, внешние источники и получатели данных, а также файлы и БД, требуемые процессами для своих операций.
2. Словари данных. Являются каталогами всех элементов данных, присутствующих в DFD, включая групповые и индивидуальные потоки данных, хранилища и процессы, а также все их атрибуты.
3. Миниспецификации обработки, описывающие DFD-процессы нижнего уровня и являющиеся базой для кодогенерации. Фактически миниспецификации представляют собой алгоритмы описания задач, выполняемых процессами: множество всех миниспецификаций является полной спецификацией системы. Миниспецификации содержат номер и/или имя процесса, списки входных и выходных данных и тело (описание) процесса, собственно и являющееся спецификацией алгоритма или операции, трансформирующей входные потоки данных в выходные. Известно большое число разнообразных методов, позволяющих задать тело процесса, соответствующий язык может варьироваться от структурированного естественного языка или псевдокода до визуальных языков проектирования (типа FLOW-форм и диаграмм Насси-Шнейдермана) и формальных компьютерных языков.

DFD-диаграммы являются ключевой частью документа спецификации требований. Каждый узел - процесс в DFD может развертываться в диаграмму нижнего уровня, что позволяет на любом уровне абстрагироваться от деталей (отметим, что структурные методологии, ориентированные на потоки управления, не обладают этим свойством). Проектные спецификации строятся по DFD и их миниспецификациям автоматически. Наиболее часто для описания проектных спецификаций используется методика структурных карт Джексона, иллюстрирующая иерархию модулей, связи между ними и некоторую информацию об их исполнении (последовательность вызовов, итерацию). Существует ряд методов автоматического преобразования DFD в структурные карты: один из таких методов, а также реализующий его алгоритм приводится Фишером в его обзорной книге по CASE-технологиям [Fisher 1988].

Отметим, что DFD моделируют функции, которые система должна выполнять, но ничего (или почти ничего) не сообщают об отношениях между данными, а также о поведении системы в зависимости от времени - для этих целей методологии использует диаграммы "сущность-связь" и диаграммы переходов состояний, соответственно.

Главной отличительной чертой методологии Гейна-Сарсона является наличие этапа моделирования данных, определяющего содержимое хранилищ данных (БД и файлов) в DFD в Третьей Нормальной Форме. Этот этап включает построение списка элементов данных, располагающихся в каждом хранилище данных; анализ отношений между данными и построение соответствующей диаграммы связей между элементами данных; представление всей информации по модели в виде связанных нормализованных таблиц. Кроме того, методологии отличаются чисто синтаксическими аспектами, так, например, различны графические символы, представляющие компоненты DFD.

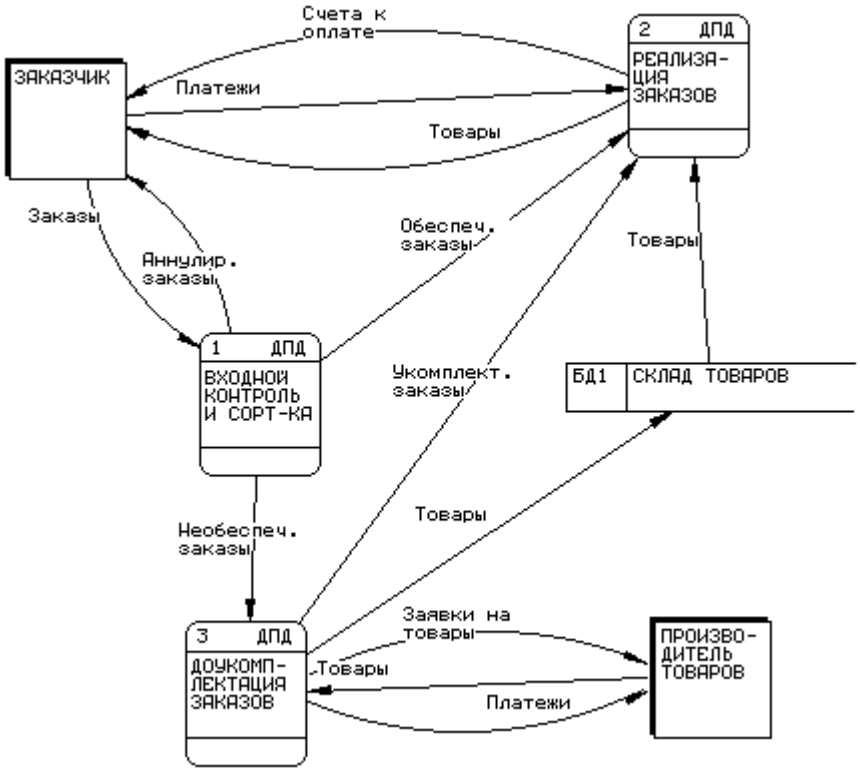


***Рис. 9.1**. Пример диаграммы Гейна-Сарсона*

Таким образом, методы в рассматриваемых подходах представляют собой "кулинарную книгу" с рецептами, помогающими от чистого листа бумаги или экрана перейти к хорошо организованной модели системы. Эти рецепты основаны на простой концепции нисходящего поэтапного разбиения функций системы на подфункции. На первом этапе формируется контекстная диаграмма верхнего уровня, идентифицирующая границы системы и определяющая интерфейсы между системой и окружением. Затем, после интервьюирования эксперта предметной области, формируется список внешних событий, на которые система должна реагировать. Для каждого из таких событий строится пустой процесс ("bubble") в предположении, что его функция обеспечивает требуемую реакцию на это событие, которая в большинстве случаев включает генерацию выходных потоков и событий (но может также включать и занесение информации в хранилище данных для ее использования другими событиями и процессами). На следующем уровне детализации аналогичная деятельность осуществляется для каждого из пустых процессов.

В качестве примера рассмотрим верхний уровень функциональной модели компании, занимающейся распределением товаров по заказам (рис. 9.1). Заказы подвергаются входному контролю и сортировке. Если заказ не отвечает номенклатуре товаров или оформлен неправильно, то он аннулируется с соответствующем уведомлением заказчика. Если заказ не аннулирован, то определяется, имеется ли на складе соответствующий товар. В случае положительного ответа выписывается счет к оплате и предъявляется заказчику, при поступлении платежа товар отправляется заказчику. Если заказ не обеспечен складскими запасами, то отправляется заявка на товар производителю. После поступления требуемого товара на склад компании заказ становится обеспеченным и повторяет вышеописанный маршрут. При построении данной модели использована нотация Гейна-Сарсона.

9.2. SADT - технология структурного анализа и проектирования

SADT (Structured Analysis and Design Technique) - одна из самых известных методологий анализа и проектирования систем, введенная в 1973 г. Россом (Ross). SADT успешно использовалась в военных, промышленных и коммерческих организациях для решения широкого спектра задач, таких как программное обеспечение телефонных сетей, системная поддержка и диагностика, долгосрочное и стратегическое планирование, автоматизированное производство и проектирование, конфигурация компьютерных систем, обучение персонала, встроенное ПО для оборонных систем, управление финансами и материально-техническим снабжением и др. Данная методология широко поддерживается Министерством обороны США, которое было инициатором разработки стандарта IDEF0 как подмножества SADT. Это, наряду с растущей автоматизированной поддержкой, сделало ее более доступной и простой в употреблении.

С точки зрения SADT модель может основываться либо на функциях системы, либо на ее предметах (планах, данных, оборудовании, информации и т.д.). Соответствующие модели принято называть активностными моделями и моделями данных. Активностная модель представляет с нужной степенью подробности систему активностей, которые в свою очередь отражают свои взаимоотношения через предметы системы. Модели данных дуальны к активностным моделям и представляют собой подробное описание предметов системы, связанных системными активностями. Полная методология SADT заключается в построении моделей обеих типов для более точного описания сложной системы. Однако, в настоящее время широкое применение нашли только активностные модели, их рассмотрению и посвящен данный раздел.

Основным рабочим элементом при моделировании является диаграмма. Модель SADT объединяет и организует диаграммы в иерархические древовидные структуры, при этом этом чем выше уровень диаграммы, тем она менее детализирована. В состав диаграммы входят блоки, изображающие активности моделируемой системы, и дуги, связывающие блоки вместе и изображающие взаимодействия и взаимосвязи между блоками. SADT требует, чтобы в диаграмме было 3-6 блоков: в этих пределах диаграммы и модели удобны для чтения, понимания и использования. Вместо одной громоздкой модели используются несколько небольших взаимосвязанных моделей, значения которых взаимодополняют друг друга, делая понятной структуризацию сложного объекта. Однако такое жесткое требование на число блоков на диаграмме ограничивает применение SADT для ряда предметных областей. Например, в банковских структурах имеется 15-20 равноправных деятельностей, которые целесообразно отразить на одной диаграмме. Искусственное их растаскивание по разным уровням SADT-модели явно не улучшает ее понимаемость.

Блоки на диаграммах изображаются прямоугольниками и сопровождаются текстами на естественном языке, описывающими активности. В отличие от других методов структурного анализа в SADT каждая сторона блока имеет вполне определенное особое назначение: левая сторона блока предназначена для ***Входов***, верхняя - для ***Управления***, правая - для ***Выходов,*** нижняя - для ***Исполнителей***. Такое обозначение отражает определенные принципы активности: *Входы* преобразуются в *Выходы, Управления* ограничивают или предписывают условия выполнения, *Исполнители* описывают, за счет чего выполняются преобразования.

Дуги в SADT представляют наборы предметов и маркируются текстами на естественном языке. Предметы могут состоять с активностями в четырех возможных отношениях: *Вход, Выход, Управление, Исполнитель*. Каждое из этих отношений изображается дугой, связанной с определенной стороной блока - таким образом, стороны блока чисто графически сортируют предметы, изображаемые дугами. Входные дуги изображают предметы, используемые и преобразуемые активностями. Управляющие дуги обычно изображают информацию, управляющую действиями активностей. Выходные дуги изображают предметы, в которые преобразуются входы. Исполнительские дуги отражают (по крайней мере частично) реализацию активностей (рис. 9.2).

Блоки на диаграмме размещаются по "ступенчатой" схеме в соответствии с их доминированием, которое понимается как влияние, оказываемое одним блоком на другие. Кроме того, блоки должны быть пронумерованы, например, в соответствии с их доминированием. Номера блоков

служат однозначными идентификаторами для активностей и автоматически организуют эти активности в иерархию модели.

Взаимовлияние блоков может выражаться либо в пересылке *Выхода* к другой активности для дальнейшего преобразования, либо в выработке управляющей информации, предписывающей, что именно должна делать другая активность. Таким образом, диаграммы SADT являются предписывающими диаграммами, описывающими как преобразования между *Входом* и *Выходом*, так и предписывающие правила этих преобразований.

Рис. 9.2. *Пример SADT-блока*

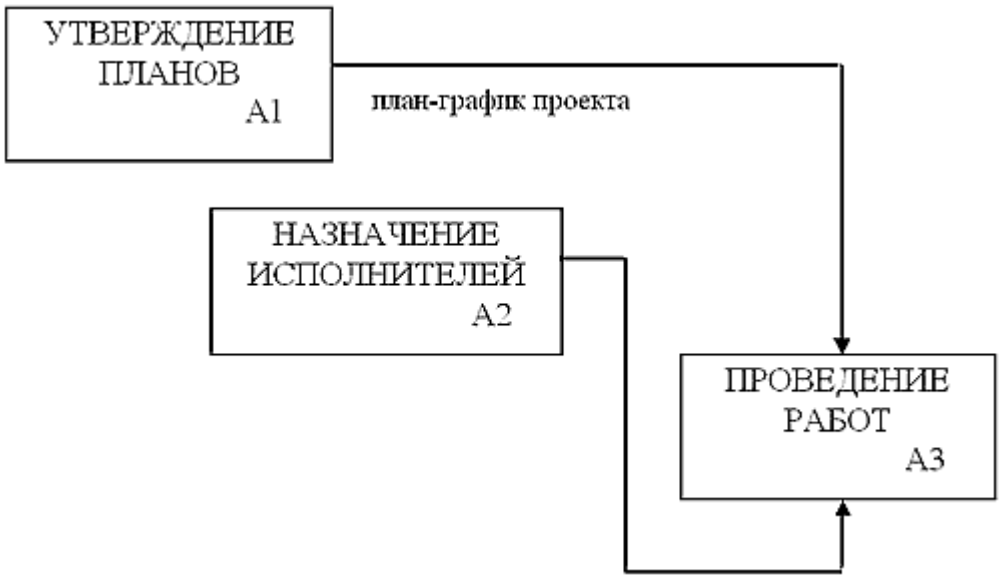


Рис. 9.3. *Пример отношения Выход-Исполнитель*

В SADT требуются только пять типов взаимосвязей между блоками для описания их отношений: *Управление, Вход, Управленческая Обратная Связь, Входная Обратная Связь, Выход - Исполнитель*. Отношения *Управления* и *Входа* являются простейшими, поскольку они отражают интуитивно очевидные прямые воздействияы. Отношение *Управления* возникает тогда, когда *Выход* одного блока непосредственно влияет на блок с меньшим доминированием. Отношение *Входа* возникает, когда *Выход* одного блока становится *Входом* для блока с меньшим доминированием. Обратные связи более сложны, поскольку они отражают итерацию или рекурсию - *Выходы* из одной активности влияют на будущее выполнение других функций, что впоследствии влияет на исходную активность. *Управленческая Обратная Связь* возникает, когда *Выход* некоторого блока влияет на блок с большим доминированием, а отношение *Входной Обратной Связи* имеет место, когда *Выход* одного блока становится *Входом* другого блока с большим доминированием. Отношения *Выход - Исполнитель* встречаются нечасто и представляют особый интерес. Они отражают ситуацию, при которой *Выход* одной активности становится средством достижения цели другой активностью (рис. 9.3).

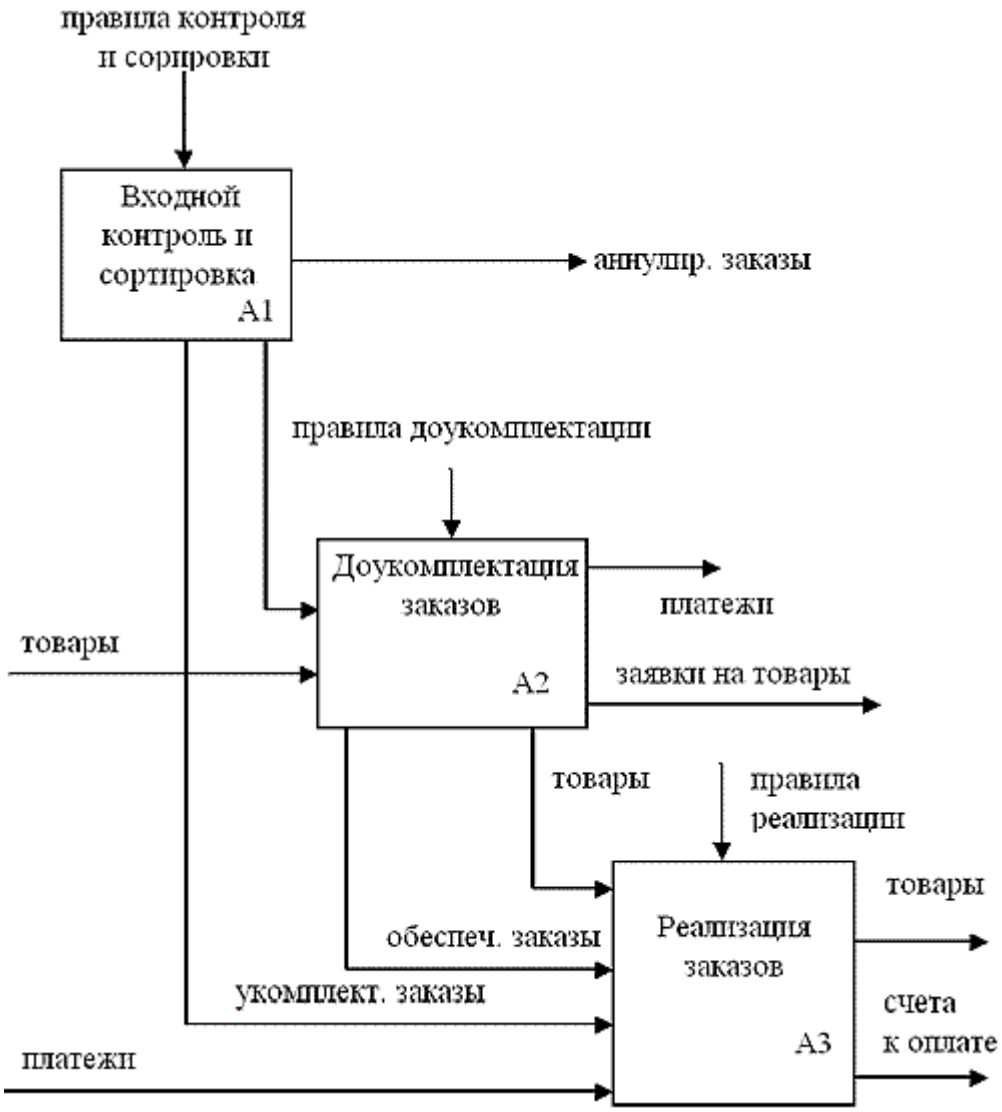


***Рис. 9.4**. Пример SADT-диаграммы*

Дуги SADT, как правило, изображают наборы предметов, поэтому они могут разветвляться и соединяться вместе различным образом. Разветвления дуги означают, что часть ее содержимого (или весь набор предметов) может появиться в каждом ответвлении дуги. Дуга всегда помечается до разветвления, чтобы дать название всему набору. Кроме того, каждая ветвь дуги может быть помечена в соответствии со следующими правилами: считается, что непомеченная ветвь содержит все предметы, указанные в метке перед разветвлением; каждая метка ветви уточняет, что именно содержит эта ветвь. Слияние дуг указывает, что содержимое каждой ветви участвует в формировании после слияния объединенной дуги. После слияния дуга всегда помечается для указания нового набора, кроме того, каждая ветвь перед слиянием может помечаться в соответствии со следующими правилами: считается, что непомеченные ветви содержат все предметы, указанные в общей метке после слияния; каждая метка ветви уточняет, что именно содержит эта ветвь.

При создании модели одна и та же диаграмма чертится несколько раз, что создает различные ее варианты. Чтобы различать различные версии одной и той же диаграммы, в SADT используется схема контроля конфигурации диаграмм, основанная на их номерах. Если диаграмма замещает более старый вариант, то предыдущий номер помещается в скобках для указания связи с предыдущей работой.

Пример SADT-диаграммы, моделирующей деятельность компании, занимающейся распределением товаров по заказам (рис. 9.1), приведен на рис. 9.4.

SADT, как и другие методологии проектирования, целесообразно использовать на ранних этапах ЖЦ: для понимания системы до ее воплощения. SADT позволяет сократить дорогостоящие ошибки на ранних этапах создания системы, улучшить контакт между пользователями и

разработчиками, сгладить переход от анализа к проектированию. Несомненное достоинство SADT заключается в том, что она легко отражает такие характеристики как управление, обратная связь и исполнители.

9.3. Сравнительный анализ SADT-моделей и потоковых моделей

Как уже отмечалось, практически во всех методах структурного анализа используются три группы средств моделирования:

- диаграммы, иллюстрирующие функции, которые система должна выполнять, и связи между этими функциями - для этой цели чаще всего используются DFD или SADT (IDEF0);
- диаграммы, моделирующие данные и их взаимосвязи (ERD);
- диаграммы, моделирующие поведение системы (STD).

Таким образом, наиболее существенное различие между разновидностями структурного анализа заключается в методах и средствах функционального моделирования. С этой точки зрения все разновидности структурного системного анализа могут быть разбиты на две группы - применяющие методы и технологию DFD (в различных нотациях) и использующие SADT-методологию. Соотношение применения этих двух разновидностей структурного анализа в существующих CASE-средствах составляет по материалам CASE Consulting Group 90% для DFD и 10% для SADT. По данным автора, основанным на анализе 127 существующих CASE-пакетов, это соотношение выглядит как 94% к 3%, соответственно. Оставшиеся 3% CASE-средств используют методологии, не относящиеся ни к одной из перечисленных разновидностей. Представляется очевидным, что соотношение такого же порядка справедливо и для цифр распространенности рассматриваемых методологий на практике.

Сравнительный анализ этих двух разновидностей методологий проводится по следующим параметрам:

- адекватность средств рассматриваемой проблеме;
- согласованность с другими средствами структурного анализа;
- интеграция с последующими этапами разработки (и прежде всего с этапом проектирования).

1) Адекватность. Выбор той или иной структурной методологии напрямую зависит от предметной области, для которой создается модель. Предметом бизнес-консалтинга являются организационные системы (точнее, функционирование или деятельность таких систем). Для моделирования таких систем традиционно используется методология SADT (точнее ее подмножество IDEF0). Однако статическая SADT-модель не обеспечивает полного решения задач бизнес-консалтинга, необходимо иметь возможность исследования динамических характеристик бизнес-процессов. Одним из решений является использование методологии и средств динамического моделирования, основанной, например, на цветных (раскрашенных) сетях Петри CPN (Color Petri Nets). Фактически SADT и CPN служат компонентами интегрированной методологии бизнес-консалтинга: SADT-диаграммы автоматически преобразуются в прообраз CPN-модели, которая затем дорабатывается и исполняется в различных режимах, чтобы получить соответствующие оценки.

Следует отметить, что не существует принципиальных ограничений в использовании DFD в качестве средства построения статических моделей деятельностей. Более того, в настоящий момент доступен ряд методологий и продуктов динамического моделирования (INCOME Mobile, CPN-AMI и др.), базирующихся на сетях Петри различного вида и интегрируемых с DFD-моделью, которые позволяют успешно решать задачи бизнес-консалтинга.

Методология SADT успешно работает только для реорганизации хорошо специфицированных и стандартизованных западных бизнес-процессов, поэтому она и принята на Западе в качестве типовой. Например, в Министерстве Обороны США десятки лет существуют четкие должностные инструкции и методики, которые жестко регламентируют деятельность, делают ее

высокотехнологичной и ориентированной на бизнес-процесс. В российской действительности с ее слабой типизацией бизнес-процессов, их стихийным появлением и развитием, разумнее ориентироваться на методологию организации и/или реорганизации потоков информации и отношений: для таких задач методологии, основанные на потоковых диаграммах, не просто допустимы, а являются единственно возможными.

Если же речь идет об информационно-технологическом консалтинге, где методологии применяются к системам обработки информации, а не к системам вообще, как это предполагается в SADT, то здесь DFD вне конкуренции. Практически любой класс систем успешно моделируется при помощи DFD-ориентированных методов: в этом случае вместо реальных объектов рассматриваются отношения, описывающие свойства этих объектов и правила их поведения. Примерами таких систем служат системы документооборота, управления и другие системы, богатые разнообразными отношениями.

SADT-диаграммы значительно менее выразительны и удобны для моделирования систем обработки информации (сравните рис. 9.1 и 9.4). Так, дуги в SADT жестко типизированы (вход, выход, управление, механизм). В то же время применительно к системам обработки информации стирается смысловое различие между входами-выходами, с одной стороны, и управлениями и механизмами, с другой: входы, выходы и управления являются потоками данных и/или управления и правилами их трансформации. Анализ системы при помощи потоков данных и процессов, их преобразующих, является более прозрачным и недвусмысленным.

Более того, в SADT вообще отсутствуют выразительные средства для моделирования особенностей систем обработки информации. DFD с самого начала создавались как средство проектирования информационных систем (тогда как SADT - как средство проектирования систем вообще) и имеют более богатый набор элементов, адекватно отражающих специфику таких систем (например, хранилища данных являются прообразами файлов или баз данных, внешние сущности отражают взаимодействие моделируемой системы с внешним миром).

Наличие миниспецификаций DFD-процессов нижнего уровня позволяет преодолеть логическую незавершенность SADT (а именно, обрыв модели на некотором достаточно низком уровне, когда дальнейшая ее детализация становится бессмысленной) и построить полную функциональную спецификацию разрабатываемой системы. Это позволит расширить возможности применения созданной модели (например, ее можно будет использовать для автоматизированного и быстрого обучения новых работников конкретному направлению деятельности).

Ограничения SADT, запрещающие использовать более 5-7 блоков на диаграмме, вынуждают искусственно детализировать систему, что затрудняет понимание модели заказчиком, резко увеличивает ее объем и, как следствие, ведет к неадекватности модели реальной картине.

2) Согласованность. Главным достоинством любых моделей является возможность их интеграции с моделями других типов. В данном случае речь идет о согласованности функциональных моделей со средствами информационного и событийного (временного) моделирования. Согласование SADT-модели с ERD и/или STD практически невозможно или носит тривиальный характер. В свою очередь, DFD, ERD и STD взаимно дополняют друг друга и по сути являются согласованными представлениями различных аспектов одной и той же модели (см. рис. 1.2).

Таблица 8.4 отражает возможность такой интеграции для DFD и SADT-моделей.

Таблица 8.4

Название	ERD	STD	Структурные карты
DFD	+	+	+
SADT	+	-	-

Отметим, что интеграция DFD-STD осуществляется за счет расширения классической DFD специальными средствами проектирования систем реального времени (управляющими процессами, потоками, хранилищами данных), и STD является детализацией управляющего процесса, согласованной по управляющим потокам и хранилищам. Интеграция DFD-ERD осуществляется с использованием отсутствующего в SADT объекта - хранилища данных, структура которого описывается с помощью ERD и согласуется по соответствующим потокам и другим хранилищам на DFD.

3) Интеграция с последующими этапами. Важная характеристика методологии - ее совместимость с последующими этапами применения результатов анализа (и прежде всего с этапом проектирования, непосредственно следующим за анализом и опирающимся на его результаты).

DFD могут быть легко преобразованы в модели проектирования (структурные карты) - это близкие модели. Более того, известен ряд алгоритмов автоматического преобразования иерархии DFD в структурные карты различных видов, что обеспечивает логичный и безболезненный переход от этапа анализа требований к проектированию системы. С другой стороны, автору неизвестны формальные методы преобразования SADT-диаграмм в проектные решения системы обработки информации.

В заключение необходимо отметить, что рассмотренные разновидности структурного анализа по сути - два приблизительно одинаковых по мощности языка для передачи понимания. И одним из основных критериев выбора является следующий: насколько хорошо каждым из этих языков владеет консультант или аналитик, насколько грамотно он может на этом языке выражать свои мысли. Автору неоднократно приходилось видеть проекты, выполненные с использованием как DFD, так и SADT, в которых просто невозможно разобраться.

9.4. Методология SSADM

Примером еще одной методологии, ориентированной на диаграммы потоков данных, является методология SSADM (Structured Systems Analysis and Design Method), созданная в начале 80-х годов и принятая в 1993 году в качестве национального стандарта Великобритании для разработки информационных систем. Ее несомненным достоинством является наличие взаимосогласованных методик, регламентирующих начальные этапы разработки системы, центральным из которых является этап итеративного определения требований. В то же время SSADM не распространяется на этапы, связанные с реализацией, внедрением и сопровождением системы, отсылая разработчика к другим общедоступным методологиям, рекомендуемым британским государственным агенством по информатике и вычислительной технике.

В SSADM применяется нисходящий подход к построению интегрированных функциональных, информационных и событийных моделей. При моделировании функций используются классические DFD (включающие только базовые объекты: процесс, поток данных, хранилище данных, внешнюю сущность) с миниспецификациями на структурированном естественном языке. Моделирование данных осуществляется с использованием нотации LDS (Logical Data Structure), являющейся диалектом ER-модели. Для событийного моделирования используются диаграммы истории жизни сущностей ELN (Entity Life History), поддерживающие индикаторы состояний, события с привязанными к ним действиями, возможность задавать последовательные, параллельные и итеративные конструкции, а также конструкции выбора.

Согласно SSADM определение системных требований включает следующие шесть основных этапов этапов:

1. *Оценка реализуемости.* Данный этап предваряет инициациацию работ по созданию системы, его основными процессами являются следующие:
    - анализ первичных бизнес-требований (включая определение целевого назначения будущей системы, ее основных пользователей и т.п.)
    - предварительная экономическая оценка проекта

- построение план-графика выполнения основных работ
- подготовка документов по оценке возможности создания системы.

2. *Предпроектное обследование и моделирование требований*. Результатом данного этапа должна являться функционально полная модель требований заказчика к будущей системе, а также оценки важности этих требований для будущего пользователя и оценки необходимых для реализации каждого требования ресурсов. SSADM рекомендует следующие шаги для достижения результата этапа:
   - определение границ будущей системы
   - выявление основных требований
   - выявление процессов обработки информации
   - выявление обрабатываемых данных
   - построение информационно-логической модели требований
   - обобщение результатов и подготовка отчетов
3. *Выбор варианта автоматизации.* На основании результатов (оценок) предшествующего этапа предлагаются несколько (от 3 до 6) вариантов автоматизации, из которых на основании выявленных ограничений совместно с заказчиком выбирается окончательный вариант.
4. *Разработка логического проекта.* На данном этапе осуществляется пересмотр выявленных требований с учетом выбранного варианта автоматизации с фиксацией неотраженных данным вариантом требований. При этом требования детализируются и уточняются, выявляются противоречия между ними, создается и оценивается прототип будущей системы. После завершения данного этапа SSADM запрещает добавление новых функциональных требований, допускается лишь их корректировка и уточнение.
5. *Выбор варианта реализации.* Этап включает проработку нескольких вариантов реализации, касающихся технической и программной среды, их оценку и совместный с заказчиком выбор приемлемого варианта.
6. *Физическое проектирование.*
   - разработка физической информационной модели
   - разработка спецификаций к программным компонентам
   - оптимизация информационной модели
   - уточнение спецификаций к программным компонентам
   - оформление документации.

Отличительной чертой SSADM является четкое выделение и поддержка соответствующими методиками так называемых “нефункциональных требований”. Нефункциональные требования специфицируют, ***с какам уровнем качества*** система должна выполнять свои функции. Примерами таких требований являются:

- среднее время наработки на отказ
- время отклика
- ограничения доступа
- требования безопасности и т.п.

9.5. Методологии, ориентированные на данные

С позиций ориентированных на данные методологий вход и выход модели являются наиболее важными, структуры данных (а не потоки данных) определяются первыми, а процедурные компоненты строятся как производные от структур данных. Фактически процесс проектирования заключается в определении структур данных, слиянии их в некий прообраз иерархической структуры программы и наполнении этой структуры детальной логикой обработки данных. Для поддержки такого подхода традиционно используются сетевые диаграммы для определения потоков, источников и приемников данных, древовидные структурные диаграммы для представления иерархии как структур данных, так и программных структур, а также диаграммы детализации логики процедур (обычно на базе структурированного естественного языка).

Классическим примером рассматриваемого подхода является структурное проектирование Джексона. Его базовая процедура проектирования предназначена для "простых" программ

("сложная" программа разбивается на "простые" традиционными методами) и включает следующие 4 этапа:

1. *Этап проектирования данных.*
   - Построение системной сетевой диаграммы, демонстрирующей все хранилища, источники и стоки данных в программе.
   - Представление каждой входной и выходной структуры данных древовидной структурной диаграммой.
2. *Этап проектирования программ.*
   - Формирование структуры программы комбинированием структур данных.
   - Идентификация всех связей между компонентами структур данных.
   - Верификация полученной структуры программы.
3. *Этап проектирования операций.*
   - Построение списка операций, необходимых для продуцирования выходных структур данных из входных.
   - Назначение операций компонентам структуры программы.
4. *Этап проектирования текстов.*
   - Трансляция построенной модели программы в текстовый вид с добавлением ряда логических условий для управления выполнением циклов и выбором данных.

Другим примером рассматриваемого подхода является DSSD (Data-Structured Systems Development), предложенная Варнье-Орром и ориентированная на разработку систем со структурными данными методология, использующая теорию множеств для описания проекта ПО. Также как и в математике, множество определяется перечислением его элементов. Так множество отделов компании может быть описано следующим образом:

компания = {бухгалтерия, маркетинг, производство, распространение}

DSSD использует аналогичную нотацию, а именно множественную скобку (рис. 9.5).

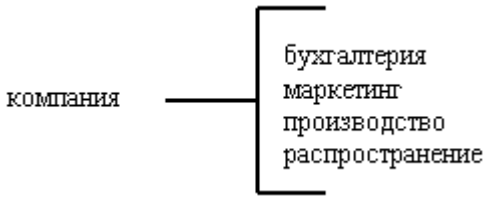


***Рис. 9.5**. Множественная скобка*

Подобно методологии Джексона, в рассматриваемом подходе структура программы строится на базе структур данных, а главным отличием является то, что в методологии Джексона необходимо сливать все входные и выходные структуры данных для продуцирования структуры программы, а в DSSD входные данные и структура программы продуцируются из выходных структур. Таким образом, главная аксиома DSSD утверждает, что выходные структуры данных полно и точно определяют входные структуры, которые, в свою очередь, определяют и логику их обработки.

При построении модели в DSSD используются диаграммы сущностей (диалект DFD) для определения системного контекста и диаграммы Варнье-Орра (assembly-line diagrams) в качестве основного средства моделирования системы. Базовым элементом диаграммы Варнье-Орра является множественная скобка. Детализация элементов данных производится слева-направо, предполагаемая последовательность действий осуществляется слева-направо и сверху-вниз. Такая нотация удобна для представления композиции структур, определения структур данных, спецификации форматов файлов, и может быть использована для иллюстрирования структуры программы и иерархии модулей (заменой структур данных на модули или файлы, а на нижних уровнях - на подпрограммы, DO-циклы, условные и другие операторы), являясь в этом случае неким аналогом визуального языка проектирования типа FLOW-форм. Основные этапы методологии изображены на рис. 9.6 с помощью диаграммы Варнье-Орра.

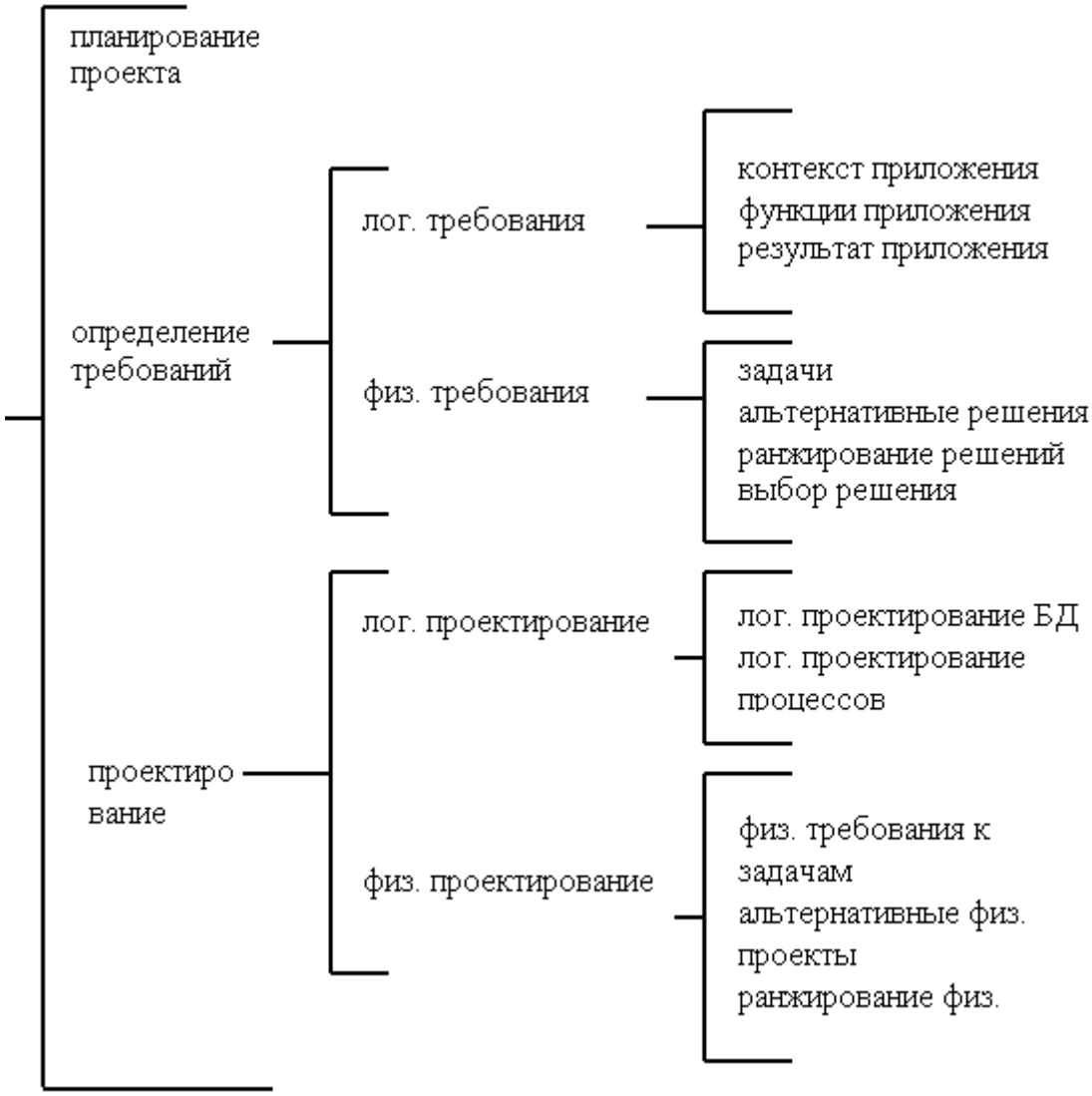


***Рис. 9.6**. Диаграмма Варнье-Орра*

9.6. Основные этапы подхода Мартина

IE-методология Мартина предоставляет общую стратегию разработки информационных систем, фокусирующую внимание на стратегическом планировании и бизнес-процессах. В то же время она является и инженерным подходом к разработке ПО, т.к. обеспечивает нисходящую пошаговую процедуру построения информационной системы (позволяя при этом работать с неиерархическими структурами данных). Подход Мартина базируется на двух концепциях:

- послойного целостного подхода к разработке интегрированных приложений, базирующегося на стратегическом плане развития информационных систем;
- первоначальной направленности на моделирование данных, а затем на функциональное моделирование

Основные этапы подхода Мартина приведены на рис. 9.7.

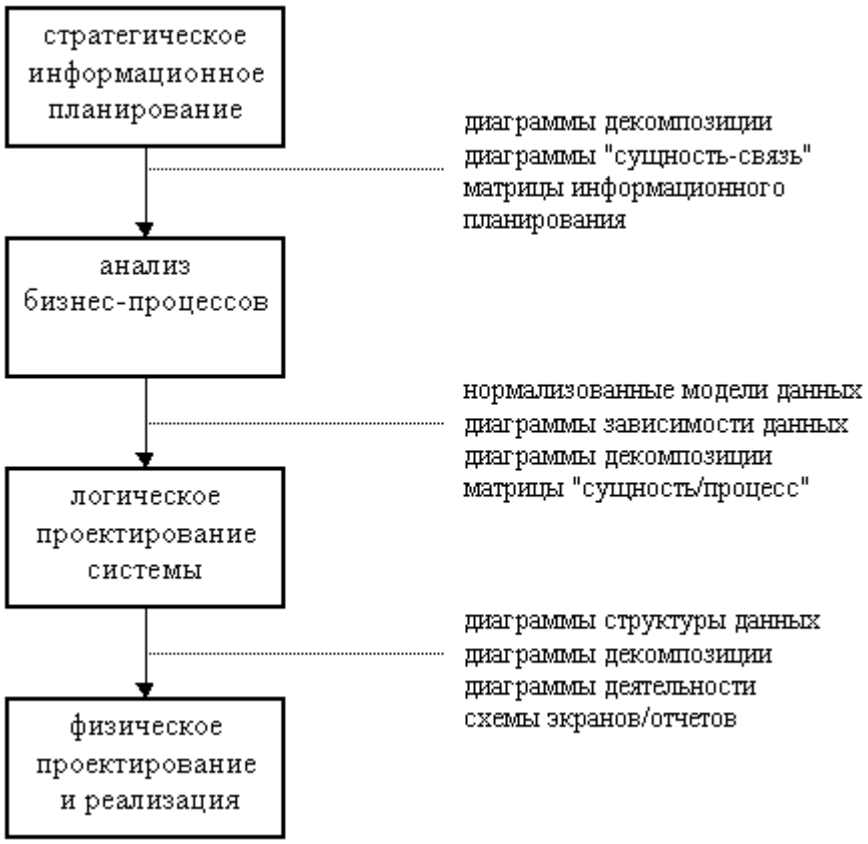


***Рис. 9.7**. Основные этапы подхода Мартина*

1. Этап *стратегического информационного планирования* начинается с построения стратегического плана для бизнес-системы, включающего цели и стратегии их достижения. Далее строится модель предметной области, отражающая существующую специфику и определяющая основные бизнес-процессы и организационную структуру бизнес-системы, а также определяется порядок разработки информационной системы. При моделировании используются диаграммы декомпозиции (иерархические древовидные структурные диаграммы) и диаграммы "сущность-связь" для представления основных бизнес-процессов и структур данных, соответственно.
2. На этапе *анализа* основные бизнес-процессы, разработанные на этапе 1), используются для разбиения общей задачи на частные, при этом основное внимание уделяется определению информационной и функциональной моделей для частных задач. При этом диаграммы "сущность-связь" трансформируются в нормализованную модель данных, а диаграммы декомпозиции распределяются по подзадачам. Для представления процессов служат DFD, диаграммы зависимости данных (диалект DFD) и диаграммы декомпозиции, а для соотнесения данных и процессов, в которых эти данные используются, применяются матрицы "сущность/процесс".
3. На этапе *логического проектирования* IE становится аналогична SE для разработки ПО. Базой для проектирования являются процессы, разработанные на этапе анализа. Используя методики нисходящей функциональной декомпозиции, проектируются спецификации обработки в процессах и их логические структуры данных. При этом используются диаграммы структуры данных (диалект ERD), определяющие типы сущностей, их атрибуты и связи, диаграммы декомпозиции и диаграммы деятельности (вид миниспецификации), детализирующие логику процессов. Для согласования требований пользователя создаются прототипы пользовательских интерфейсов с помощью схем экранов/отчетов.
4. На этапе *физического проектирования* и реализации производится преобразование логической модели ИС в физическую и ее реализация.

ГЛАВА 10

АРХИТЕКТУРА СОВРЕМЕННЫХ СИСТЕМ И МЕТОДОЛОГИИ

В центре любой методологии находится некоторая системная архитектура, и лишь затем совокупность стратегий и методов анализа и проектирования. Архитектура современных систем является трехслойной (рис.10.1) и имеет следующие характеристики:

- четко определенные слои
- формальные и явные интерфейсы между слоями
- скрытые и защищенные детали внутри каждого слоя.

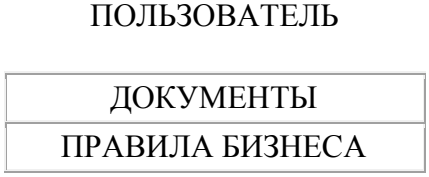


Рис. 10.1. Архитектура современной системы

Три слоя (база данных, правила бизнеса, документы) отражают возрастание уровня абстракции в рассматриваемой системной архитектуре. Наиболее детальным слоем является база данных, более высокий уровень абстракции - слой правил бизнеса, наивысший уровень абстракции - слой документов. В данной архитектуре слой правил бизнеса является относительно новой концепцией, соответствующей функциям руководителей среднего звена. Процессы данного слоя отражают:

- выполнение требуемых задач
- принятие решений в соответствующей компетенции
- запуск других задач в слое правил бизнеса и других слоях.

Независимость слоев трехслойной системной архитектуры обеспечивает следующие основные преимущества:

- улучшение базы данных - отделение базы данных от изменений в технологиях, а следовательно, поддержка согласованности и осмысленности данных в течении длительного периода времени;
- гибкость интерфейсов пользователя - изменение интерфейсов без влияния на бизнес-процессы и наоборот;
- разделение усилий коллектива разработчиков.

Трехслойная архитектура (а именно, выделение слоя бизнес-правил) требует модификации существующих методологий, в первую очередь, информационно-ориентированных методологий и методологий, ориентированных на данные. Такие методологии имеют следующие две характеристики, нуждающиеся в изменении:

- информационная модель (и база данных) рассматриваются как центральные понятия при анализе и проектировании;
- функциональная модель (а следовательно, и правила бизнеса) является некоторым дополнением к информационной модели.

Согласно такому подходу, информационная модель является первичной, занимает центральное место и регламентирует весь процесс анализа и проектирования, что приводит к следующим ограничениям:

- построенная на ее основе функциональная модель либо является слабо связанной с информационной моделью, либо неадекватно отражает существующие бизнес-процессы и правила;
- сама по себе информационная модель является недостаточной (хотя и важной) для решения задач консалтинга;
- информационная модель плохо понимаема неспециалистами, поэтому попытки вовлечь руководство в разработку обречены на неудачу.

С другой стороны, руководство прекрасно ориентируется в технологиях и бизнес-процессах предприятия. Более того, функциональные модели (например, на базе диаграмм потоков данных) интуитивно понимаемы неспециалистами.

Таким образом, в центре современного проекта лежат две вещи - база данных и бизнес-процесс. При этом основным центром является бизнес-процесс, база данных - менее важный из двух центров, т.е. процесс становится первичным и во многом определяет весь проект. Модель процесса является ценным средством для размышлений и совместной работы над перспективами развития предприятия и системной разработкой. Тем не менее информационная модель

продолжает оставаться важной и соответствующим образом влиять на разрабатываемую функциональную модель.

В таблице 10.1 представлена трехслойная системная архитектура в разрезе регламентируемых методологией этапов разработки (анализ требований, проектирование, реализация).

Таблица 10.1

Слои	Анализ	Проектирование	Реализация
Документы	Поток работ	Поток форм	Формы
Правила бизнеса	Поток процессов	Модель компонентов	Программы
База данных	Модель данных	Схема базы данных	Таблицы и т.п.

Анализ требований. В слое документа рассматриваются обобщенные потоки между подразделениями и конкретными сотрудниками предприятия без подробного описания каких-либо учетных форм и интерфейсов. На уровне правил бизнеса рассматриваются детальные модели требований. На уровне базы данных строится концептуальная модель, увязанная с функциональной моделью требований на уровне укрупненных подсхем будущей информационной модели.

Проектирование. На уровне документа макетируются последовательности форм. На уровне бизнес-правил осуществляется детальное проектирование будущих рабочих мест с привязкой к конкретным сущностям информационной модели. На уровне базы данных концептуальная модель преобразуется в диаграмму “сущность-связь”.

Реализация. На данном этапе проект преобразуется в систему.

В следующей главе рассматривается методология выполнения консалтинговых проектов, адаптированная для трехзвенной архитектуры прежде всего за счет ее ориентации на первичность правил бизнеса.

ЧАСТЬ 3

ЭТАПЫ РАЗРАБОТКИ КОНСАЛТИНГОВЫХ ПРОЕКТОВ

В третьей части книги рассматривается, как исследованные в главах 1-10 методы и средства структурного системного анализа могут быть использованы на ранних этапах разработки систем, т.е. будет намечена **методология разработки консалтинговых проектов** - общий подход к консалтингу, основанный на структурном анализе и проектировании.

В главе 11 обсуждаются основные этапы методологии и подробно описываются задачи и виды работ, выполняемых консультантом на этапе обследования предприятия.

Глава 12 посвящена обработке результатов обследования - построению моделей. Приводятся основные характеристики моделей деятельности и системного проекта (модели требований), описываются принципиальные различия моделей.

В главе 13 рассматриваются шаги и этапы выработки предложений по автоматизации предприятия, разработки технического проекта. Дается пример фрагмента технического проекта автоматизации ремонтной службы автобазы.

ГЛАВА 11

ПРОВЕДЕНИЕ ОБСЛЕДОВАНИЯ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ

11.1. Цели и основные этапы консалтинга

Основными целями разработки консалтинговых проектов являются:

- представление деятельности предприятия и принятых в нем технологий в виде иерархии диаграмм, обеспечивающих наглядность и полноту их отображения;
- формирование на основании анализа предложений по реорганизации организационно-управленческой структуры;
- упорядочивание информационных потоков (в том числе документооборота) внутри предприятия;
- выработка рекомендаций по построению рациональных технологий работы подразделений предприятия и его взаимодействию с внешним миром;
- анализ требований и проектирование спецификаций корпоративных информационныхсистем;
- рекомендации и предложения по применимости и внедрению существующих систем управления предприятиями, прежде всего классов MRP (manufacturing resource planning) и ERP (enterprise resource planning).

Структура подхода к разработке консалтинговых проектов приведена на рис. 11.1.

Этап 1 (анализ первичных требований и планирование работ) предваряет инициацию работ над проектом. Его основными задачами являются: предварительное изучение задачи, анализ первичных бизнес-требований, предварительная экономическая оценка проекта, построение план-графика выполнения работ, создание и обучение совместной рабочей группы. Важнейшими на данном этапе являются и организационные мероприятия: должны быть изданы соответствующие приказы по проведению работ, назначены ответственные по направлениям - без подобной поддержки со стороны руководства предприятия бессмысленно вообще затевать консалтинговый проект.

Рис. 11.1. Структура подхода

Первым шагом собственно разработки является **предварительное изучение** задачи. Предварительное изучение должно ответить на ряд вопросов:

- В чем недостатки существующей ситуации?
- Какие улучшения возможны?
- На кого окажет влияние новая система?

На данном этапе целесообразно построить обзорную диаграмму потоков данных для оценки существующей ситуации с целью ее использования для подгонки всех фрагментов друг к другу и выявления недостатков.

Предварительное изучение может потребовать от двух дней до четырех недель. К его окончанию аналитик должен разумно оценить преимущества внедрения новой системы, а также обосновать временные затраты и стоимость следующего шага разработки - детального изучения. Результаты предварительного изучения рассматриваются руководством соответствующего уровня, на их основе может быть санкционирована возможность детального изучения.

Детальное изучение, включающее этапы 2-4, строится на фактах, выявленных во время предварительного изучения и проведения обследования деятельности предприятия, и предполагает более детальное и точное документирование ограничений существующей системы, а также уточнение функций этой системы до уровня, необходимого для написания спецификаций новой (модернизированной) системы.

В рамках этапа 2 (проведение обследования деятельности предприятия) осуществляется:

- предварительное выявление требований, предъявляемых к будущей системе;
- определение оргштатной и топологической структур предприятия;
- определение перечня целевых задач (функций) предприятия;
- анализ распределения функций по подразделениям и сотрудникам;
- определение перечня применяемых на предприятии средств автоматизации.

При этом выявляются функциональные деятельности каждого из подразделений предприятия и функциональные взаимодействия между ними, информационные потоки внутри подразделений и между ними, внешние по отношению к предприятию объекты и внешние информационные взаимодействия.

Длительность обследования составляет 1-2 недели. По окончании обследования строится и согласуется с заказчиком предварительный вариант функциональной модели предприятия, включающей идентификацию внешних объектов и информационных взаимодействий с ними, а также детализацию до уровня основных деятельностей предприятия и информационных связей между этими деятельностями.

На этапе 3 (построение моделей деятельности предприятия) осуществляется обработка результатов обследования и построение моделей деятельности предприятия следующих двух видов:

- ***модели “как есть”***, представляющей собой "снимок" положения дел на предприятии (оргштатная структура, взаимодействия подразделений, принятые технологии, автоматизированные и неавтоматизированные бизнес-процессы и т.д.) на момент обследования и позволяющей понять, что делает и как функционирует данное предприятие с позиций системного анализа, а также на основе автоматической верификации выявить ряд ошибок и узких мест и сформулировать ряд предложений по улучшению ситуации;
- ***модели “как должно быть”***, интегрирующей перспективные предложения руководства и сотрудников предприятия, экспертов и системных аналитиков и позволяющей сформировать видение новых рациональных технологий работы предприятия.

Главным результатом детального изучения является построение **системного проекта (модели требований)**, являющегося первой фазой разработки собственно системы автоматизации (именно, фазой анализа требований к системе), на которой требования заказчика уточняются, формализуются и документируются. Системный проект строится на основе модели “как должно быть” и результатов обследования предприятия в части выявления требований к будущей системе.

При презентации системного проекта аналитик должен быть готов услышать больше критических замечаний, чем при использовании традиционных подходов, т.к. диаграммы легче понять и

обнаружить какие-либо несоответствия и ошибки. В результате презентации принимается решение о продолжении разработки или ее прекращении, а также устанавливается сумма бюджета проекта. Поэтому аналитик должен создать несколько альтернативных моделей систем, имеющих разный набор преимуществ и предполагающих различные капиталовложения.

По завершении данного этапа (после согласования системного проекта с заказчиком) изменяется роль консультанта. Отныне он как бы становится на сторону заказчика, и одной из его основных функций на всех последующих этапах работ будет являться контроль на соответствие требованиям, зафиксированным в системном проекте.

Отметим, что для построения каждой из требуемых моделей необходима интенсивная работа 6-7 квалифицированных системных аналитиков в течении 2-4 месяцев.

После выбора системного проекта на основе выявленных и согласованных требований осуществляется разработка предложений по автоматизации (этап 5), включающих:

- составление перечня автоматизированных рабочих мест предприятия и способов взаимодействия между ними;
- анализ применимости существующих систем управления предприятиями (прежде всего классов MRP и ERP)для решения требуемых задач и формирование рекомендаций по выбору такой системы;
- совместное с заказчиком принятие решения о выборе конкретной системы управления предприятием или разработке собственной системы;
- разработка требований к техническим средствам;
- разработка требований к программым средствам;
- разработка предложений по этапам и срокам автоматизации.

На этапе 6 на основании принятых решений по автоматизации осуществляется преобразование системного проекта в технический проект (модель реализации), включающее следующие действия:

- уточнение логической модели (разработка подробной логики каждого процесса с использованием диаграмм потоков данных и спецификаций процессов);
- проектирование физической базы данных;
- построение иерархии функций модулей, подлежащих программированию;
- оценка затрат на реализацию.

Перечисленные работы должны выполняться консультантами совместно с проектировщиками системы - именно здесь и находится граница, разделяющая консалтинг и разработку. Тем не менее желательно, чтобы на этапе реализации системы консультант также действовал в интересах заказчика - а именно, контролировал соответствие создаваемой программной системы системному и техническому проектам, а также участвовал в работах по ее расширению и модификации, т.к. планирование расширений должно осуществляться на основе модели требований.

11.2. Проведение обследования

Обследование является важнейшим и определяющим этапом выполнения консалтинговых проектов, на его основе осуществляется вся последующая деятельность. Длительность обследования обычно составляет 1-2 недели. По окончании обследования строится и согласуется с заказчиком предварительный вариант функциональной модели предприятия, включающей идентификацию внешних объектов и информационных взаимодействий с ними, а также детализацию до уровня основных деятельностей предприятия и информационных связей между этими деятельностями. В дальнейшем на основании согласованных моделей верхнего уровня и осуществляется построение детальных моделей.

Необходимо отметить, что каждый из участвующих в проекте системных аналитиков должен обследовать не более 2-3 деятельностей предприятия (таких как, например, учет кадров, бухгалтерия, маркетинг, ремонт оборудования, перевозки и т.п.) для того, чтобы тщательно в них разобраться. Современное предприятие является сложной системой, состоящей из крупных взаимоувязанных подсистем (деятельностей), а возможности человека в одновременном охвате большого количества таких подсистем ограничены, поэтому здесь в полной мере должен использоваться принцип “разделяй и властвуй”. И в этой связи вызывают недоумение заявления некоторых компаний о готовности провести обследование предприятия (обычно культивирующего 15-25 деятельностей) за 1-2 дня силами в 2-3 человека.

В качестве исходной информации при проведении обследования и выполнении дальнейших этапов служат:

- данные по оргштатной структуре предприятия;
- информация о принятых технологиях деятельности;
- стратегические цели и перспективы развития;
- результаты интервьюирования сотрудников (от руководителей до исполнителей нижнего звена);
- предложения сотрудников по усовершенствованию бизнес-процессов предприятия;
- нормативно-справочная документация;
- данные по имеющимся на предприятии средствам и системам автоматизации;
- опыт системных аналитиков в части наличия типовых решений.

При проведении обследования целесообразно применять следующие методы:

- анкетирование
- сбор документов
- интервьюирование.

Анкетирование является начальным этапом обследования и предваряет выезд группы системных аналитиков на предприятие. Анкеты позволяют составить грубое представление о деятельностях предприятия, что позволит спланировать первоначальное распределение работ группы аналитиков. Анкеты должны рассылаться руководителям структурных подразделений и содержать графы для идентификации фамилии и должности анкетируемого, отдельно излагается просьба приложить шаблоны документов, с которыми работают сотрудники соответствующего подразделения. Список вопросов должен быть ограничен (не более 15-20) с тем, чтобы вся анкета не занимала более двух листов. Автору приходилось видеть анкеты размером в 50 страниц, содержащие до 500 тщательно продуманных вопросов, но не встречался ни один человек, добровольно (а следовательно, также тщательно и с пользой для дела) на них ответивший. Примерный вариант анкеты приведен ниже:

- *ФИО руководителя подразделения, телефон*
- *Координаты контактного лица (к кому в отсутствие или при занятости руководителя можно обращаться)*
- *Каковы (с позиций Вашего подразделения) должны быть цели создания интегрированной системы управления предприятием*
- *Основные функции подразделения*
- *Какая информация поступает из других подразделений (заявки, запросы, отчеты и т.п.)*
- *Какая информация передается в другие подразделения*
- *Какая информация формируется (“рождается”) в подразделении*
- *С какими внешними предприятиями (банк, заказчик, поставщик и т.п.) взаимодействует подразделение и какой информацией обменивается*
- *Физическое представление информационных потоков и хранилищ (документ, дискета, сеть, журнал, картотека и т.п.)*
- *Время хранения информации*
- *Документы от и для руководства*
- *Штатная структура и квалификация кадров*

- *Техническое оснащение подразделения (компьютеры, сеть, модем и т.п.)*
- *Используемые программные продукты*
- *Подпись*

Просьба приложить:

1) Положение о подразделении

2) Набор документальных форм без внутреннего наполнения, т.е. используемые формы, бланки и др. (например, карточка складского учета, отчет по форме N, наряд-задание, товарно-транспортная накладная)

Сбор документов должен осуществляться на всех этапах проведения обследования, соответствующие формы, бланки и т.п. в дальнейшем сослужат неоценимую службу при разработке информационной модели предприятия (выявлении сущностей информационной модели и наполнении их атрибутикой). В дальнейшем целесообразно подготовить альбом форм с разбивкой их по деятельностям предприятия. Такой альбом будет являться хорошим вспомогательным результатом консалтинга для предприятия - своими силами подобная работа обычно не проводится (за исключением уровня отдельных исполнителей).

Интервьюирование является важнейшим и необходимым методом обследования, только с его помощью возможно разобраться во всех тонкостях применяемых на предприятии технологий. Современное предприятие является сложнейшей системой, как оно функционирует не знает ни один человек. Конечно, руководство представляет ситуацию в целом, с другой стороны, клерк досконально знает свою деятельность, но полной картины не имеет никто. И только интервьюирование представителей всех звеньев оргштатной структуры позволит выявить и, в дальнейшем, формализовать эту картину.

- С другой стороны, интервьюирование является и наиболее сложной задачей: необходимо найти контакт с сотрудником и направить беседу в необходимое для аналитика русло. Ниже предлагается несколько общих рекомендаций, касающихся линии поведения аналитика при интервьюировании.
- Тезис в начале беседы - я ничего (или почти ничего) не знаю о Вашей работе, расскажите как можно подробнее, чем Вы занимаетесь?
- Правило 1 - если Вам начали подробно рассказывать технологию работы, ни в коем случае не перебивайте, необходимые уточнения можно сделать и в конце беседы.
- Правило 2 - если в беседе участвуют несколько аналитиков, вести беседу и задавать уточняющие вопросы должен один из них, неясные для других вопросы проясняются в конце беседы.
- Правило 3 - даже если Вы прекрасно знаете предметную область, не говорите много сами и не учите интервьюируемого: в любом случае выявляются тонкости и детали, специфичные для данного предприятия и, естественно, Вам неизвестные.

В принципе этих и подобных им правил достаточно для выявления в ходе беседы необходимой аналитику информации приблизительно у 90% интервьюируемых, а этого более, чем достаточно в соответствии с законом “20 на 80” (сравните: 20% людей выпивают 80% пива). Тем не менее постараемся составить основанную на опыте типизацию остальных 10% и предложить возможные действия по выходу из тупика.

1) “Отказник” - как правило, квалифицированный специалист, осознающий свою незаменимость. Обычно руководству известен его характер, поэтому необходимы жесткие меры: либо данная деятельность не будет включена в модель, либо она будет промоделирована на основании опыта и соображений здравого смысла.

2) “Говорун” - как правило, руководитель среднего звена, понимающий, что по-старому работать нельзя и хватающийся за любую возможность улучшить ситуацию. Очень полезный для поддержки проекта человек, тем не менее в беседе готов бесконечно обсуждать свои трудности и

проблемы, получить от него необходимую для построения модели информацию практически невозможно. Единственный способ работы с ним - обсуждение уже построенной (пусть примитивной и во многом ошибочной) модели с целью ее доводки.

3) “Балласт” - человек, давно работающий на предприятии и непонятно чем занимающийся. На вопросы типа “Какие функции Вы выполняете?”, ”С какими документами Вы работаете?” агрессивно повторяет как попугай ”Я делаю все”, ”Со всеми документами”, ”Все документы ко мне приходят и все уходят”. Какой-либо информации получить не удается по причине ее отсутствия. Естественно никакого отражения подобной “деятельности” в модели не производится.

4) Человек, занимающий экзотическую и малопонятную должность типа “главный обогатитель”. Представляет собой модификацию варианта 3) с той лишь разницей, что реально деятельность по обогащению руды существует и, следовательно, должна быть отражена в модели.

5) “Мелкая сошка” - человек, не привыкший к проявлению интереса к себе и своей работе и занимающий низшую должность. При должном терпении реально получение того небольшого куска информации, которым он владеет. При обследовании диспетчерской службы одного из северных предприятий на одной из удаленных точек автор имел неосторожность во время кратковременной беседы включить диктофон. Беседа была тут же прервана, и автору пришлось ждать минут сорок, пока интервьюируемая приводила себя в порядок - накладывала косметику и делала прическу!

Какую же информацию нужно выявлять прежде всего во время интервьюирования? Во-первых, необходимо ограничить контекст системы - с этой целью должны быть выявлены все внешние объекты, с которыми моделируемое предприятие взаимодействуют, технологии взаимодействия со стороны предприятия, а также информационные (и, возможно, материальные) потоки, обеспечивающие эти взаимодействия. Во-вторых, должны быть детально выявлены реальные технологии работы предприятия - нормативно-справочная документация (если она имеется) описывает их неполно. В-третьих, должны быть определены реальные функции подразделений и их взаимосвязи и взаимозависимости, поскольку положения о подразделениях такую информацию не содержат. В-четвертых, должны быть выявлены и специфицированы все информационные хранилища (в том числе и бумажные: картотеки, архивы и т.п.). В-пятых, должна быть оценена аппаратно-техническая база предприятия, а также исследовано работающее на ней программное обеспечение. Наконец, в-шестых, должны быть собраны статистические данные по бизнес-процессам предприятия. Остановимся на последнем более подробно.

Статистические данные при проведении обследования необходимо собирать по каждому объекту будущей модели: потоку данных, элементу данных, процессу, хранилищу данных, внешней сущности и т.п. - все они со временем сослужат хорошую службу. Так на этапе анализа моделей наличие подробной статистики обеспечит их адекватную верификацию на полноту и непротиворечивость и позволит на начальных этапах выявить ошибки и узкие места в построенных моделях. В следующих пунктах будет показано, как эти статистические данные работают на дальнейших этапах, начиная с этапа выработки предложений по автоматизации и заканчивая собственно разработкой или внедрением выбранной системы.

1) Составные данные. Для составных данных статистика собирается, как правило, лишь для итеративных (повторяющихся) компонентов - необходимо точно знать количество итераций для каждого из них. Например, заказ на книги включает в себя перечень заказанных книг с их атрибутами. Поэтому для формирования требований к функции распечатки соответствующего бланка необходимо знать: сколько книг обычно заказывается? как часто производится нетипичный заказ и каковы его размеры? сколько авторов обычно бывает у книги? …

Статистика по итеративным компонентам внутри составных данных в дальнейшем будет использоваться для проектирования экранов, отчетов и, естественно, при проектировании базы данных.

2) Элементы данных. О каждом элементе данных необходимо знать формат данных и допустимые значения этого элемента. Формат (включая тип) и физическая длина очень полезны при проектировании экранных форм и определении размеров баз данных.

3) Потоки данных. Такие характеристики потока как скорость и интенсивность являются необходимыми при определении требований к аппаратным (техническим) средствам. Кроме того, для любого составного потока данных полезно знать распределение компонентов внутри этого потока данных. Например, если в фирме “Рога и Копыта” заказ определяется, как *заказ={заказ на рога!заказ на копыта},* и выясняется, что 12% всех заказов составляют заказы на рога, 84% - на копыта, а 4% заказов - на заполнение стержней для шариковых ручек, то данная статистика может использоваться для определения пиковых нагрузок на соответствующие обрабатывающие процессы (а также, возможно, для принятия решения об оказании дополнительного вида услуги - upgrade стержней).

4) Процессы. Важнейшими характеристиками процессов являются частота и время выполнения. Именно здесь и лежит ключ к улучшению их функционирования. Кроме того, такие сведения являются необходимыми при определении требований к аппаратным средствам.

5) Хранилища данных. По хранилищам данных обычно собирается следующая информация: среднее количество записей в каждом хранилище данных, количество чтений, добавлений, изменений и удалений записей по каждому из процессов, включающих перечисленные действия. Проектировщик баз данных может использовать эту статистику для нескольких целей - например, решить вопрос, какой ключ считать первичным, сортировать ли хранилище и по какому ключу, решить, нужно ли завести дополнительную таблицу с целью обеспечения скорости доступа и т.д. Более того, к этой информации потребуется обратиться и при выборе подходящей СУБД, которая сможет обеспечить необходимую частоту и/или гибкость доступа к данным.

Ценной информацией является и хронология доступа. Так запись о конкретном заказе, как правило, однажды создается и однажды удаляется. Но обычно доступ к этой записи осуществляется очень часто в начале ее существования (запросы о покупателе, счета, платежи, накладные) и крайне редко в дальнейшем (месячные и квартальные отчеты), что позволяет своевременно осуществлять ее архивацию.

6) Внешние объекты. Наконец, необходимо собрать определенную статистику об окружении, в котором система должна работать ("ограничения окружения"). Наиболее важным здесь является количество пользователей, их способы использования системы и географическое распределение. По этой статистике можно будет сделать заключения о стоимости периферии, о типе системы телекоммуникаций и даже о том, как данные должны быть физически распределены для обеспечения удаленного доступа. Другие данные об окружении могут включать температуру, уровень шума, существующую отделку помещения, уровень радиации и т.п.

Следует отметить, что часто возникает необходимость в проведении дополнительного обследования: какие-то моменты были не до конца выяснены, где-то возникли нестыковки, что-то было просто упущено. Обычно дополнительное обследование занимает 2-3 дня, и при его проведении очень полезно обсудить с интервьюированными уже наработанные модели.

ГЛАВА 12
ПОСТРОЕНИЕ МОДЕЛЕЙ

12.1. Построение и анализ моделей деятельности предприятия

Построение и анализ моделей деятельности предприятия относится к области бизнес-консалтинга, включающего в себя построение моделей текущего и целевого состояния предприятия, выработку предложений по совершенствованию его деятельности, формирование целевой программы развития предприятия и плана перехода из текущего состояния в целевое. На данном этапе осуществляется обработка результатов обследования и построение функциональных, информационных и, по необходимости, событийных моделей технологий работы предприятия следующих двух видов:

- модели “как есть”, представляющей собой "снимок" положения дел на предприятии (оргштатная структура, взаимодействия подразделений, принятые технологии, автоматизированные и неавтоматизированные бизнес-процессы и т.д.) на момент обследования и позволяющей понять, что делает и как функционирует данное предприятие с позиций системного анализа, а также на основе автоматической верификации выявить ряд ошибок и узких мест и сформулировать ряд предложений по улучшению ситуации;
- модели “как должно быть”, интегрирующей перспективные предложения руководства и сотрудников предприятия, экспертов и системных аналитиков по совершенствованию деятельности предприятия.

При этом переход от модели “как есть” к модели ”как должно быть” обычно осуществляется следующими двумя способами:

- Совершенствованием технологий на основе оценки их эффективности. При этом критериями оценки являются стоимостные и временные затраты выполнения бизнес-процессов, дублирование и противоречивость выполнения отдельных задач бизнес-процесса, степень загруженности сотрудников (“легкий” реинжиниринг).
- Радикальным изменением технологий и переосмыслением бизнес-процессов (“жесткий” реинжиниринг). Например, вместо попыток улучшения бизнес-процесса проверки кредитоспособности клиента, может быть следует задуматься, а нужна ли вообще такая проверка? Возможно затратына такие проверки каждого из клиентов во много раз превышают убытки, которые может понести банк в отдельных случаях недобросовестности (в случае, когда клиентов много, а суммы сделок незначительны)!

Необходимость подобного перехода, собственно, и повлекла за собой создание подходов к реорганизации деятельности предприятий (бизнес-реинжинирингу). Наиболее популярные из таких подходов будут обсуждены в пятой части настоящей книги. В данной главе рассматривается собственно методика построения моделей деятельности.

В рамках создания моделей деятельности должен быть осуществлен:

- анализ функциональной деятельности структурных подразделений предприятия;
- анализ функционального взаимодействия структурных подразделений;
- анализ внутреннего документооборота структурных подразделений;
- анализ информационных потоков и информационного взаимодействия структурных подразделений;
- анализ применяемых в настоящее время средств автоматизации как в структурных подразделениях, так и на предприятии в целом.

По результатам анализа и моделирования осуществляется оценка эффективности деятельности структурных подразделений предприятия, на основе которой формируются предложения по совершенствованию его структуры, технологий работы структурных подразделений и предприятия в целом. Критериями такой оценки должны являться:

- количество потребителей продукции предприятия;
- стоимость издержек производства продукции;
- длительность типовых операций производства продукции;
- дублирование и противоречивость функций, информационных потоков и документооборота;

- стоимость и длительность выполнения отдельных шагов технологии или отдельных технологических цепочек шагов;
- дублирование и противоречивость выполнения отдельных шагов технологии или отдельных технологических цепочек шагов;
- степень загруженности структурных подразделений и должностных лиц;
- степень загруженности оборудования, используемого при реализации отдельных шагов технологии или технологических участков;
- степень применения средств автоматизации при поддержке выполнения отдельных шагов технологии или отдельных технологических цепочек шагов.

Результатом проведения анализа и оценки являются предложения по совершенствованию деятельности предприятия, а именно:

- по изменению технологий целевой и обеспечивающей деятельности предприятия, операций учета, планирования, управления и контроля;
- по построению рациональных технологий работы структурных подразделений предприятия с учетом существующих автоматизированных систем;
- по созданию перспективной оргштатной структуры предприятия, осуществляющей реализацию рациональных технологий работы;
- по изменению информационных потоков и документооборота, обеспечивающих реализацию рациональных технологий работы;
- по разработке проектов схем внутреннего и внешнего документооборота, проекта положения о документообороте, проекта альбома форм входных и выходных документов.

На основе разработанных и согласованных предложений формируется целевая программа развития предприятия и план мероприятий по переходу из текущего состояния в целевое. Целевая программа развития предприятия должна включатьдолгосрочные решения, цели, задачи и основные параметры развития. План мероприятий перехода из текущего состояния в целевое должен содержать:

- последовательность, формы, способы и время выполнения задач, поставленных структурным подразделениям предприятия;
- распределение сотрудников структурных подразделений и материальных средств по решаемым задачам;
- порядок информационного и других видов взаимодействия структурных подразделений и органов управления.

В связи с вышесказанным каждая из моделей деятельности должна включать:

- полную функциональную модельс глубиной проработки до уровня конкретного действия должностного лица структурного подразделения предприятия;
- информационную модель, интегрированную с функциональной моделью;
- динамические, стоимостные, событийные и т.п. модели для осуществления соответствующих оценок.

Ниже перечислены основные виды и последовательность работ, рекомендуемые при построении моделей деятельности.

1) Разработка структурной функциональной модели деятельности предприятия:

- определение информационных потоков между основными процессами деятельности, связей между процессами и внешними объектами;
- оценка объемов и интенсивности информационных потоков;
- разработка иерархии диаграмм, образующих структурную функциональную модель деятельности предприятия;
- анализ и оптимизация структурной функциональной модели.

2) Разработка информационной модели предприятия:

- определение сущностей модели и их атрибутов;
- проведение атрибутного анализа и оптимизация сущностей;
- идентификация отношений между сущностями и определение типов отношений;
- разрешение неспецифических отношений;
- анализ и оптимизация информационной модели.

3) Разработка событийной модели предприятия:

- идентификация перечня состояний модели и определение возможностей переходов между состояниями;
- определение условий, активирующих переходы, и действий, влияющих на дальнейшее поведение;
- анализ и оптимизация событийной модели.

Следует отметить, что построенные модели деятельностей являются не просто промежуточным результатом, используемым консультантом для выработки каких-либо рекомендаций и заключений. Они представляют собой самостоятельный результат, имеющий большое практическое значение, в частности:

1) Модели позволяют осуществлять автоматизированное и быстрое обучение новых работников конкретному направлению деятельности предприятия (так как ее технология содержится в модели) с использованием диаграмм (известно, что "одна картинка стоит тысячи слов").

2) С их помощью можно осуществлять предварительное моделирование нового направления деятельности с целью выявления новых потоков данных, взаимодействующих подсистем и бизнес-процессов.

Ниже приводятся некоторые основополагающие рекомендации по структурированию моделей деятельности.

1) Основной принцип заключается в том, что структурирование должно осуществляться в соответствии с деятельностями и бизнес-процессами предприятия, а не в соответствии с его оргштатной структурой. Именно бизнес-процессы представляют ценность для клиента, и именно их улучшением предстоит в дальнейшем заниматься консультанту. Модель, основанная на оргштатной структуре, может предемонстрировать лишь хаос, царящий в организации (о котором, в принципе, руководству и так известно, иначе оно не воспользовалось бы услугами консультантов), на ее основе возможно внести предложения только об изменении этой структуры. С другой стороны, модель, основанная на бизнес-процессах, содержит в себе (не всегда в явном виде) и оргштатную структуру предприятия.

2) Верхний уровень модели должен отражать только контекст системы - взаимодействие моделируемого единственным контекстным процессом предприятия с внешним миром и ничего более. В случае построения модели структуры, включающей в себя несколько разнотипных предприятий, на контекстном уровне необходимо отразить каждое из них и их соответствующие взаимосвязи. Например, контекстная диаграмма горно-обогатительного комбината может содержать процессы *Автобаза, Карьер, Фабрика и Управление ГОК*, контекстная диаграмма регионального банка Сбербанка РФ содержит процессы *Территориальное Управление, Типовое Отделение, Типовой Филиал*.

3) На втором уровне модели должны быть отражены основные деятельности предприятия и их взаимосвязи. Например, для автотранспортного предприятия одним из решений может быть выделение следующих деятельностей:*Эксплуатация автотранспорта, Ремонт и техническое обслуживание, Контроль безопасности, Управление производством, Обеспечивающая деятельность*. В случае большого количества деятельностей некоторые из них можно вынести на третий уровень модели. Так *Обеспечивающая деятельность* может включать в себя *Учет кадров,*

Бухгалтерский учет, Экономическое планирование, Материально-техническое снабжение, Складской учет и т.п. Но в любом случае под деятельности необходимо отводить не более двух уровней модели.

4) Каждая из деятельностей, в свою очередь, должна быть детализирована на бизнес-процессы (желательно, единственного уровня). Например, деятельность по учету кадров включает в себя бизнес-процессы *Прием на работу, Увольнение* и т.п.

5) Дальнейшая детализация бизнес-процессов осуществляется посредством бизнес-функций. Так процесс *Прием на работу* содержит в себе функции *Прием заявления, Оформление приказа, Регистрация* и др. Обычно для моделирования бизнес-функции достаточно 2-3 уровней детализации, которая завершается описанием элементарного алгоритма с помощью миниспецификации.

6) Таким образом, общее число уровней в модели не должно превышать 6-7. Практика показывает, что этого вполне достаточно для построения полной модели деятельности современного предприятия любой отрасли.

В заключение приведем три примера реальных ситуаций, для которых на основании построенных моделей деятельности удалось убедить соотвествующее руководство в необходимости коренного изменения существовавших технологий.

Первый пример касается автобазы, входящей в состав горнообогатительного комбината и занимающейся перевозкой породы от нескольких территориально разделенных предприятий по добыче руды (карьеров) на аналогичные предприятия по ее обогащению (фабрики). Парк автобазы содержит около 200 самосвалов “БелАЗ” грузоподъемностью 120 тонн, работы по перевозкам осуществляются в 3 смены. На каждую смену водителю выписывается путевой лист, содержащий 52 графы для однократного заполнения (хотя реально не все заполняются), при этом 5 граф заполняются многократно в соответствии с количеством погрузок/разгрузок. Кроме этого, на каждом путевом листе должно быть проставлено 17 подписей самых различных лиц, прежде чем он попадает в бухгалтерию автобазы и на его основе производится расчет соответствующих выплат. Даже если на получение каждой подписи и заполнение графы затратить в среднем по одной минуте, то оформление одного путевого листа (не включая его обработку в бухгалтерии) занимает более часа, а в день таких листов в принципе может быть шестьсот! Конечно, руководство автобазы прекрасно понимало проблему и ставило задачу сократить количество подписей хотя бы до 9-10. После проведения обследования и построения и анализа моделей выяснилось, что **ВСЯ ИНФОРМАЦИЯ**, за исключением контроля состояния водителя и, частично, самосвала (техническая исправность, медицинский контроль), **ДУБЛИРУЕТСЯ** в различных первичных документах (прежде всего, в диспетчерской сводке, ведомостях на выдачу талонов и различных накладных на отпуск горючего, масел и т.п.), то есть по своей сути путевой лист является производным документом! После предоставления таких результатов с резюме об уничтожении путевых листов как класса у руководства оставался единственный аргумент - требования ГАИ. Но, позвольте, для таких большегрузных самосвалов требуются специальные дороги, да и ездят они по четко определенным маршрутам карьер-фабрика. Более того, у них отсутствует государственный номер, весь учет ведется в соответствии с гаражным номером (от первого до двухсотого), не говоря уже о том, что за все время длительных командировок автору не встретился ни один инспектор.

Второй пример относится к распределенной диспетчерской службе того же самого комбината. Фактически имеется 8 диспетчерских пунктов (2 автобазы, 3 карьера, 2 фабрики, контора), на которых собирается и сводится одна и та же информация по перевозкам породы: карьер собирает данные по вывозу, фабрика - по разгрузке, автобаза - по перевозке, контора - всю эту информацию по каждому из предприятий, то есть одни и те же данные фиксируются 4 раза! Более того, все эти данные не совпадают, это связано со спецификой производства: например, в сырую погоду при разгрузке на кузов самосвала может налипнуть до 5 тонн породы! А поскольку объемы перевозок оказывают существенное влияние на начисляемую зарплату, все эти 8 диспетчеров ежедневно тратят уйму времени, сил и нервов на согласование этих данных (и в конце концов находят

компромиссное решение). А дальше начинается самое интересное: с определенной периодичностью (неделя, месяц) специалист-маркшейдер делает замеры, сравнивает их результаты с предыдущими и выдает информацию по вывезенной породе за соответствующий период. И именно эта информация служит основой для начисления зарплаты и формирования отчетов по деятельности!

Третий пример относится к деятельности одного из молокозаводов, осуществляющего разлив и упаковку молокопродуктов. Вывоз молокопродуктов осуществляется водителями молочных магазинов. При этом с них берется залог - стоимость тары (контейнеров, ящиков и т.п.). В один прекрасный день умные головы в руководстве этого отдельно взятого молокозавода решили практически вдвое повысить стоимость тары (размер залога). Буквально на следующий день все склады молокозавода были заполнены порожней тарой: водители со всего города моментально сориентировались и вернули тару (в том числе и принадлежащую другим молокозаводом города). А еще через день руководством нашего молокозавода был подписан контракт на построение моделей деятельности, до этого успешно пролежавший в кабинетах несколько месяцев. Воистину, пока гром не грянет, мужик не перекрестится!

12.2. Разработка системного проекта

Создание системного проекта (по другому, модели требований к будущей системе) является первой фазой разработки собственно системы автоматизации (именно, фазой анализа требований к системе), на которой требования заказчика уточняются, формализуются и документируются, так как если требования нигде не зафиксированы, то их вроде-бы и не существует. Системный проект строится на основе модели “как должно быть” и результатов обследования предприятия в части выявления требований к будущей системе.

Фактически на этом этапе дается ответ на вопрос: "Что должна делать будущая система?". Именно здесь лежит ключ к успеху всего проекта автоматизации. В практике создания больших программных систем известно немало примеров неудачной реализации именно из-за неполноты и нечеткости определения системных требований.

На этом этапе определяются:

- архитектура системы, ее функции, внешние условия ее функционирования, распределение функций между аппаратной и программной частями;
- интерфейсы и распределение функций между человеком и системой;
- требования к программным и информационным компонентам системы, необходимые аппаратные ресурсы, требования к базе данных, физические характеристики компонент системы, их интерфейсы;
- состав людей и работ, имеющих отношение к системе;
- ограничения в процессе разработки (директивные сроки завершения отдельных этапов, имеющиеся ресурсы, организационные процедуры и мероприятия, обеспечивающие защиту информации).

В рамках системного проектирования должно быть осуществлено:

- определение состава, структуры и характеристик функциональных задач в рамках деятельности структурных подразделений;
- определение состава и структуры программных средств автоматизации технологии решения задач с учетом существующих средств в структурных подразделениях;
- определение структуры и характеристик информационного обеспечения технологии решения задач;
- разработка технических решений по построению информационного обеспечения (логических структур баз данных, структур классификаторов);
- разработка состава автоматизируемых процедур документооборота.

Системный проект должен включать:

- полную функциональную модель требований к будущей системе;
- комментарии к функциональной модели (спецификации процессов нижнего уровня в текстовом виде);
- пакет отчетов и документов по функциональной модели, включающий характеристику объекта моделирования, перечень подсистем, требования к способам и средствам связи для информационного обмена между компонентами, требования к характеристикам взаимосвязей системы со смежными системами, требования к функциям системы;
- концептуальную модель интегрированной базы данных (пакет диаграмм);
- архитектуру системы с привязкой к концептуальной модели;
- предложения по оргштатной структуре для поддержки системы.

Таким образом, системный проект содержит функциональную, информационную и, возможно, событийную модели требований к будущей системе. Виды и последовательность работ при построении этих моделей требований аналогичны соответствующим работам по построению моделей деятельности. Дополнительно системный проект включает в себя техническое задание на создание автоматизированной системы.

Необходимо отметить следующее достоинство системного проекта. Для традиционной разработки характерно осуществление начальных этапов кустарными неформализованными способами. В результате заказчики и пользователи впервые могут увидеть систему после того, как она уже в большей степени реализована. Естественно, эта система отличается от того, что они ожидали увидеть. Поэтому далее следует еще несколько итераций ее разработки или модификации, что требует дополнительных (и значительных) затрат денег и времени. Ключ к решению этой проблемы и дает системный проект, позволяющий:

- описать, "увидеть" и скорректировать будущую систему до того, как она будет реализована физически;
- уменьшить затраты на разработку и внедрение системы;
- оценить разработку по времени и результатам;
- достичь взаимопонимания между всеми участниками работы (заказчиками, пользователями, разработчиками, программистами и т.д.);
- улучшить качество разрабатываемой системы, а именно: создать оптимальную структуру интегрированной базы данных, выполнить функциональную декомпозицию типовых модулей.

Системный проект полностью независим и отделяем от конкретных разработчиков, не требует сопровождения его создателями и может быть безболезненно передан другим лицам. Более того, если по каким-либо причинам предприятие не готово к реализации системы на основе проекта, он может быть положен "на полку" до тех пор, пока в нем не возникнет необходимость. Кроме того, его можно использовать для самостоятельной разработки или корректировки уже реализованных на его основе программных средств силами программистов отдела автоматизации предприятия.

Системное проектирование по сравнению с построением моделей деятельностей имеет важную особенность в технике структурирования модели. Особую роль здесь играют хранилища (накопители) данных: практически все процессы модели связываются не напрямую, а с использованием этих объектов (что реально соответствует чтению/записи информации из/в базу данных). При этом операции записи должны удовлетворять основному критерию проектирования: данные должны заносится в накопитель один раз в том месте, где они впервые появляются.

Основное правило введения накопителей данных заключается в следующем: если данные из некоторого накопителя используются по крайней мере двумя процессами, то этот накопитель должен присутствовать на содержащей эти процессы диаграмме. Поэтому на втором уровне модели (детализации контекстной диаграммы) должны быть введены базовые накопители, к которым осуществляют доступ основные подсистемы будущей системы. Базовым накопителям должны соответствовать основные подсхемы информационной модели. К выявлению базовых накопителей следует подходить чрезвычайно тщательно, поскольку именно с ними будут работать бизнес-процессы и бизнес-функции на всех без исключения уровнях детализации модели.

В качестве примера введения накопителей рассмотрим фрагмент модели требований к системе автоматизации упоминавшейся выше автобазы, входящей в состав горнообогатительного комбината и занимающейся перевозками породы. На рис. 12.1. приведена диаграмма потоков данных, детализирующая рассматриваемую систему на основные подсистемы:

1. *Подсистема управления производством* - включает в себя требования по автоматизации деятельности начальника автобазы, главного инженера, главного механика, главного энергетика, организации документооборота, деятельности центра управления производством - ЦУП (включая контроль неснижаемого запаса на оборотном складе, планирование ремонтов дизелей по периодам, планирование ремонтов и технического обслуживания (ТО) автосамосвалов по периодам, расчет резерва времени по шинам и фильтрам, расчет средней наработки и анализ отказов узлов автосамосвала и дизеля, формирование заказов на изготовление деталей, заявок на запчасти, наряд-заданий на ремонт и ТО) и технического отдела (включая учет транспортных средств, анализ надежности парка, узлов и агрегатов, анализ расхода запчастей и материалов, трудоемкости ТО и ремонтов, расчет коэффициента технической готовности, планирование, контроль и формирование отчетности).
2. *Подсистема ремонта и технического обслуживания* - включает в себя требования по автоматизации деятельности по диагностике (дефектоскопия, технический контроль состояния гидросистемы, силового агрегата и электрической части автосамосвала, химический анализ масел, топлива и охлаждающей жидкости), ремонту (уточнение наряд-задания, определение ремонтного участка, оформление заявки на запчасти, сдача деталей на оборотный склад, учет выполненного ремонта по каждому из ремонтных участков), техническому обслуживанию всех видов (ТО-250, ТО-500, ТО-1500), а также учет на оборотном складе.
3. *Подсистема эксплуатации* - включает в себя требования по автоматизации оперативного учета перевозок руды и вскрыши, прием заявок на перевозки, формирование графика выхода автосамосвалов на линию, оформление путевых листов, выпускной контроль, формирование диспетчерских отчетов и т.п.
4. *Подсистема контроля безопасности* - включает в себя требования по автоматизации учета мероприятий по контролю безопасности движения и учета дорожно-транспортных происшествий, контроля пожарной безопасности, контроля технической безопасности (включая безопасность работ на высоте и шиномонтажном участке).
5. *Подсистема обеспечивающей деятельности* - включает в себя требования к следующим подсистемам: материально-техническое снабжение, бухгалтерский учет, складской учет (запчастей, инструмента и расходных материалов), учет кадров и экономическое планирование (включая планирование труда и зарплаты и планово-экономическую деятельность).

На данном уровне введены накопители данных, используемые в нескольких подсистемах и являющиеся прообразами подсхем интегрированной базы данных автобазы:

1. *Сотрудники* - предназначен для хранения данных о сотрудниках автобазы. Используется при учете кадров (при приеме и увольнении, подготовке пенсионных дел, награждении), учете ремонтов и ТО (для фиксации, кем выполнен ремонт), бухгалтерии (при проведении начислений и удержаний, учете материальных ценностей) и др.
2. *Технологический транспорт* - используется для хранения данных по автосамосвалам: учетной карточки, данных по проведенным ТО, истории автосамосвала.
3. *Перевозки* - используется для хранения данных по перевозкам на основе диспетчерской сводки.
4. *Ремонты* - используется для хранения данных о любом ремонте, включая перечень замененных узлов и агрегатов.
5. *Запасные части и материалы* - используется для хранения данных о имеющихся в наличии запчастях и материалах, включая данные по складу запчастей, складу материалов, инструментальному складу и оборотному складу.

Обмен диспетчерскими данными моделируется с использованием информационного канала *Оперативные диспетчерские данные*.

Все перечисленные накопители детализируются на нижних уровнях в тех процессах, где такая детализация необходима. Например, в процессе *Химический анализ масел и жидкостей* введен накопитель *Масла и охлаждающие жидкости*, являющейся частью накопителя *Запасные части и материалы*, и по сути моделирующий единственную таблицу из базы данных, в которой хранятся данные об имеющихся в наличии на автобазе маслах и охлаждающих жидкостях (тип, место хранения, объем, результаты спектрального анализа и т.п.).

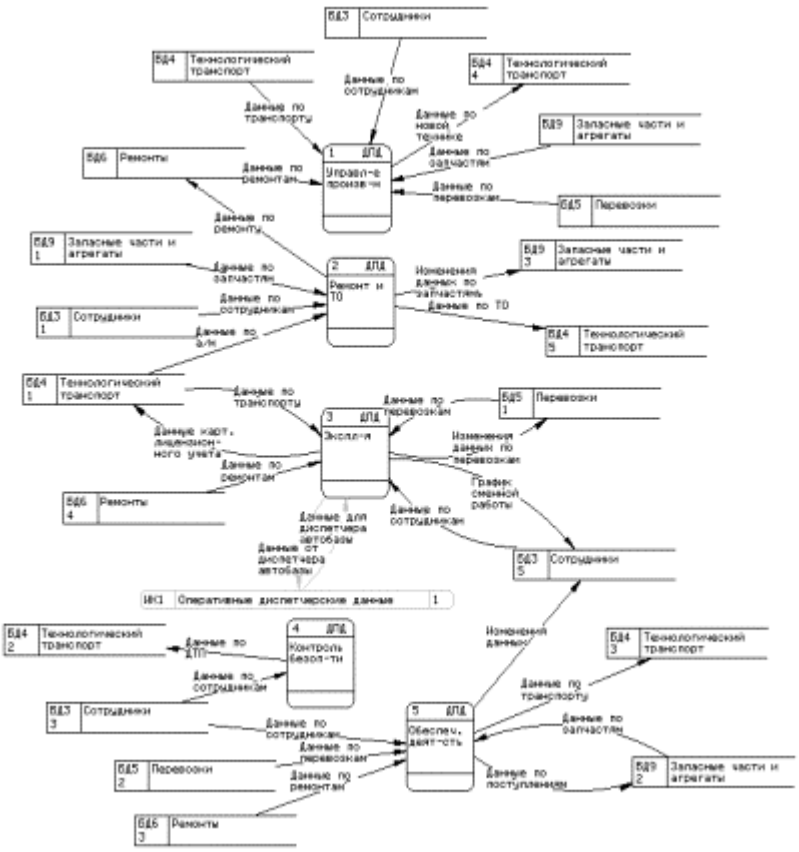


Рис. 12.1. Фрагмент системного проекта

ГЛАВА 13
РАЗРАБОТКА ПРЕДЛОЖЕНИЙ ПО АВТОМАТИЗАЦИИ И ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

13.1. Предложения по автоматизации

После построения системного проекта, содержащего требования к будущей системе, на его основе осуществляется разработка предложений по автоматизации предприятия, включающая в себя:

- составление перечня автоматизированных рабочих мест предприятия, их состава и структуры, а также способов и схем информационного взаимодействия между ними;
- разработку требований к техническим средствам;
- разработку требований к программным средствам;
- разработку топологии, состава и структуры локальной вычислительной сети;
- анализ имеющихся на рынке систем управления предприятием с учетом их соответствия системному проекту и формирование рекомендаций по выбору такой системы;
- совместное с заказчиком принятие решения о выборе конкретной системы управления предприятием (или отдельных ее элементов) или о разработке собственной системы;
- разработку предложений по этапам и срокам автоматизации.

Далее рассматриваются общие соображения по выбору программного и технического (аппаратного) обеспечения, который необходимо сделать, прежде чем приступить к детальному проектированию.

1) *Обозначение границ реализации*. Практически любая система может быть разбита на части, отражающие четыре основных типа реализации систем: ручную, пакетную, диалоговую, реального времени. Из этих четырех типов первый реализуется людьми, остальные три являются автоматическими реализациями системы. Рассмотрим критерии назначения частям системного проекта наиболее приемлемых для них типов реализации.

Ручная реализация имеет три основных преимущества перед автоматической. Во-первых, процессы не требуется заранее точно определять. По крайней мере, они могут определяться не так тщательно, как при автоматической реализации: люди хорошо знают, как заполнить пробелы в спецификации. Во-вторых, ручная система может откликаться на неожидаемые запросы, а не только на заранее планируемые. Например, ручная система бронирования авиабилетов может ответить на запрос о возможности парковки автомобиля около аэропорта. В-третьих, система может быть реализована в окружении, где автоматизация невозможна по целому ряду причин, например, психологических: хотя и возможно полностью автоматизировать процесс предоставления ссуды, люди не могут примириться с тем, что их прошения беспристрастно отклонены машиной.

Безусловно, ручные системы имеют массу недостатков. Например, люди устают, болеют, увольняются, требуют повышения зарплаты. Однако наиболее важно, что размер и сложность ручной системы будут возрастать с увеличением числа запросов, поскольку человек может обрабатывать лишь небольшое количество данных.

После определения границ ручной реализации необходимо решить, какая часть системы должна быть пакетной, а какая диалоговой? Для большинства современных приложений вся автоматизированная система должна быть диалоговой, если только не доказано противное. Соответствующее заключение может быть сделано на основе собранных статистических данных, например скорости поступления запросов и частоты изменения данных. В качестве примеров причин для пакетной реализации можно привести следующие:

- некоторые запросы требуют длительной работы со срезом базы данных за определенный период (годовой отчет, пересылка накопленной информации и т.п.);
- некоторые отклики (например, отчеты о продажах) содержат большое количество статичных данных, актуальность которых не изменяется в течение дней или даже недель.

Следующий шаг - выделение частей, реализуемых как подсистемы реального времени. Существует два принципиальных отличия системы реального времени от просто диалоговой системы. Первое из них связано с концептуальным уровнем: в системе реального времени время поступления события в систему само по себе несет определенную информацию, которая не может быть закодирована. Второе связано с уровнем реализации: время отклика системы реального времени является критичным и сопоставимым со скоростью выполнения технологических операций. В целом, рекомендуется реализовать как подсистемы реального времени те части системы, из которых должен быть исключен человек, то есть те части, в которых приоритетны следующие факторы: скорость (например, противоракетная оборона), опасность (например, контроль радиоактивности), утомляемость (работа авиадиспетчера).

2) Выбор подходящих технических средств. Разработав системный проект и определив границы реализации, можно начинать выбор аппаратной платформы, на которой будет функционировать система (или, по крайней мере, сужать область для такого выбора).

3) Анализ и выбор существующей системы. Зная типы подсистем и потенциальную аппаратную платформу, можно приступать к поиску коммерческих пакетов, которые удовлетворяют требованиям, выявленным и зафиксированным на этапе системного проектирования, и могут справиться с размерами и мощностью, определяемыми собранной статистикой. Следует отметить,

что к такому выбору необходимо подходить сверхосторожно: стоимость интегрированной системы (включая ее внедрение на предприятии), в комплексе решающей стоящие перед предприятием задачи, может составлять сотни и миллионы долларов, а ключевые слова, характеризующие различные системы, практически одни и те же:

- единая информационная среда предприятия;
- режим реального времени;
- независимость от законодательства;
- интеграция с другими приложениями (в том числе с уже работающими на предприятии системами);
- поэтапное внедрение и т.п.

И здесь неоценимую помощь оказывает системный проект, позволяющий выбрать систему, наиболее полно подходящую конкретному предприятию, либо отвергнуть данный путь и приступить к разработке и реализации собственной системы.

Ниже перечислены некоторые из критериев выбора готовой системы:

- поддержка большинства функций, выявленных при анализе требований;
- поддержка концептуальной модели данных;
- наличие высокоуровневых механизмов разработки для компенсации отсутствующих данных и функций;
- функционирование на различных аппаратных платформах;
- достаточные размеры внутренних таблиц;
- локализация.

Помимо чисто технических критериев выбора, важную роль играют также деловые критерии, например, опыт внедрения и надежность продавца.

4) Разработка собственной системы. Отметим недостатки такого подхода по сравнению с покупкой готовой системы:

- трудозатраты на создание собственной интегрированной системы огромны и составляют сотни и тысячи человеко-лет, стоимость разработки соизмерима со стоимостью готовой системы(а часто значительно превышает ее): такие продукты должны реализовываться большими коллективами программистов;
- использование готовой системы менее рискованно, чем разработка собственной;
- готовая система внедряется поэтапно и поэтому частично может быть доступна в рабочем режиме гораздо быстрее, чем собственная.

13.2. Техническое проектирование

На данном этапе на основе системного проекта и принятых решений по автоматизации осуществляется проектирование системы. Фактически здесь дается ответ на вопрос: "Как (каким образом) мы будем строить систему, чтобы она удовлетворяла предъявленным к ней требованиям?". Этот этап разделяется на два подэтапа:

- проектирование архитектуры системы, включающее разработку структуры и интерфейсов ее компонент (автоматизированных рабочих мест), согласование функций и технических требований к компонентам, определение информационных потоков между основными компонентами, связей между ними и внешними объектами;
- детальное проектирование, включающее разработку спецификаций каждой компоненты, разработку требований к тестам и плана интеграции компонент, а также построение моделей иерархии программных модулей и межмодульных взаимодействий и проектирование внутренней структуры модулей.

При этом происходит расширение системного проекта:

- за счет его уточнения;
- за счет построения моделей автоматизированных рабочих мест, включающих подсхемы информационной модели и функциональные модели, ориентированные на эти подсхемы вплоть до идентификации конкретных сущностей информационной модели;
- за счет построения моделей межмодульных и внутримодульных взаимодействий с использованием техники структурных карт.

Центральное место среди перечисленных видов работ занимает построение моделей автоматизированных рабочих мест. Ниже приводится фрагмент технического проекта на создание системы автоматизации ремонтной службы на автобазе технологического транспорта.

13.3. Фрагмент технического проекта ремонтной службы

Основной деятельностью ремонтной службы является техническое обслуживание и ремонт технологического транспорта, поддержание коэффициента технической готовности парка не менее 0.75.

В состав ремонтной службы входят:

- центр управления ремонтным производством (ЦУП)
- ремонтные участки (участки технического обслуживания (ТО) и текущего ремонта (ТР), шиномонтажный и шиноремонтный участок, участок диагностики и оперативного аварийного ремонта, участок ремонта дизелей и топливной аппаратуры, участки механосварочного производства и ремонта агрегатов)
- оборотный склад
- инструментальная кладовая
- мойка автосамосвалов

1) Состав, структура и характеристики функциональных задач в рамках деятельности ремонтной службы

1.1) Ремонтные участки

АРМ каждого из ремонтных участков должен фиксировать информацию по решению следующих функциональных задач:

- уточнение наряд-задания
- выявление необходимых деталей и их бронирование
- оформление заявок на запчасти
- сдача деталей на оборотный склад
- учет выполненного ремонта

1.2) ЦУП

Осуществляет оперативное управление ремонтными подразделениями и проводит анализ текущей и перспективной обстановки.

АРМ ЦУП должен обеспечить решение следующих задач:

- контроль неснижаемого запаса на оборотном складе
- планирование ремонтов дизелей по периодам
- планирование ремонтов автосамосвалов по периодам (включая ППР, текущие ремонты и ТО на двое суток и на месяц вперед)
- расчет резерва времени по шинам
- расчет резерва времени по фильтрам
- расчет средней наработки и анализ отказов узлов автосамосвала
- расчет средней наработки и анализ отказов узлов дизеля

- формирование заказов на изготовление деталей
- формирование заявок на запчасти
- формирование наряд-заданий
- анализ и учет расхода запасных частей, оборотных агрегатов, шин, материалов ГСМ и др.
- учет движения крупных агрегатов после ремонта по самосвалам (дизель, генератор, РМК, электродвигатели), сопровождение происходивших по ним неисправностей для облегчения диагностирования самосвалов в дальнейшем
- проведение анализа технического состояния парка по определенным критериям (пробег, моточасы, проведенные ремонты в хронологическом порядке, результаты диагностики по спектральному анализу масел и др.)
- учет времени простоев по определенным критериям (по вине персонала, по причине отсутствия запчастей, ГСМ, инструментов, по причине ожидания ремонта из-за отсутствия места или ремонтников)
- сбор статистики и анализ ходимости узлов и агрегатов с целью определения своевременного срока замены при предельном износе
- контроль за аварийным запасом з/ч и своевременное оповещение служб снабжения и техотдела;
- на основании графика проведения ТО, выданного техотделом, и с учетом реального времени наработки осуществление перерасчета постановки на техобслуживание автосамосвалов.
- выбор места установки а/с в ремонтной зоне или отстое в ожидании ремонта по определенным критериям (подъезд манипулятора (автопогрузчика) к а/с, потребность в г/п механизмах, длительность ремонта, занятость необходимых специалистов на других самосвалах, наличие необходимых з/ч для проведения ремонта, потребность в сварочных работах).

1.3) Оборотный склад

- учет номенклатуры и количества поступивших на оборотный склад отремонтированных агрегатов и узлов с участков
- учет расхода агрегатов и узлов с оборотного склада на конкретные самосвалы

1.4) Инструментальная кладовая

- учет номенклатуры и количества поступивших на склад инструментов и расходных материалов к ним
- учет расхода инструментов и материалов

1.5) Мойка автосамосвалов

- учет операций по мойке автосамосвалов

Ниже рассматриваются фрагменты технического проектирования двух автоматизированных рабочих мест: АРМ ДИАГНОСТИКА и АРМ ХИМИЧЕСКИЙ АНАЛИЗ.

2) Логическая модель базы данных ремонтной службы и ее привязка к функциональной модели автобазы

2.1) Схема базы данных ремонтной службы

Фрагменты схем базы данных ремонтной службы для АРМ ДИАГНОСТИКА и ХИМИЧЕСКИЙ АНАЛИЗ приведены на рис. 13.1 и 13.2, соответственно. Подробное описание атрибутов, выделенных в этих фрагментах приводится ниже.

1) ДЕФЕКТОСКОПИЯ - результаты дефектоскопии автосамосвала

ДАТА ИСПЫТАНИЙ - дата проведения дефектоскопии
ТИП ДИАГНОСТИКИ - тип проводимых испытаний
НОМЕР ШАССИ - номер шасси проверяемого автосамосвала

2) ДИАГНОСТИКА - результаты диагностики автосамосвала

ДАТА ИСПЫТАНИЙ - дата проведения диагностики
ТИП ДИАГНОСТИКИ - тип проводимых испытаний
НОМЕР ШАССИ - номер шасси проверяемого автосамосвала
ТАБЕЛЬНЫЙ НОМЕР СОТР - табельный номер сотрудника автобазы, проводящего испытания

3) ДИАГНОСТИКА ГИДРАВЛИКИ - результаты диагностики гидравлики

ДАТА ИСПЫТАНИЙ- дата проведения диагностики
ТИП ДИАГНОСТИКИ - тип проводимых испытаний

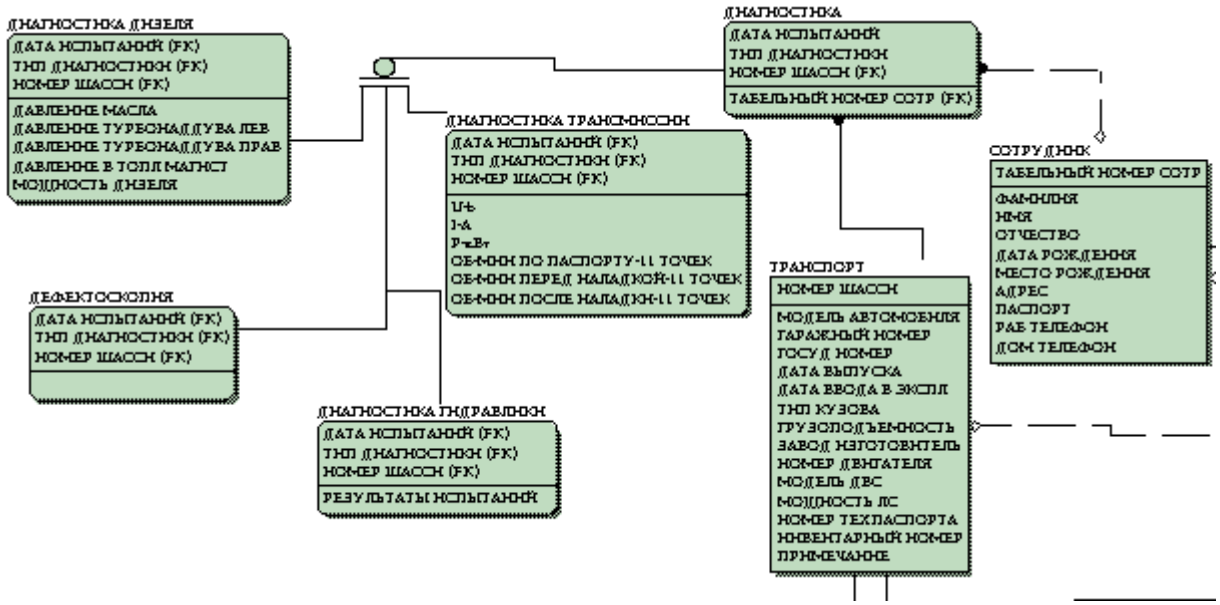


Рис. 13.1

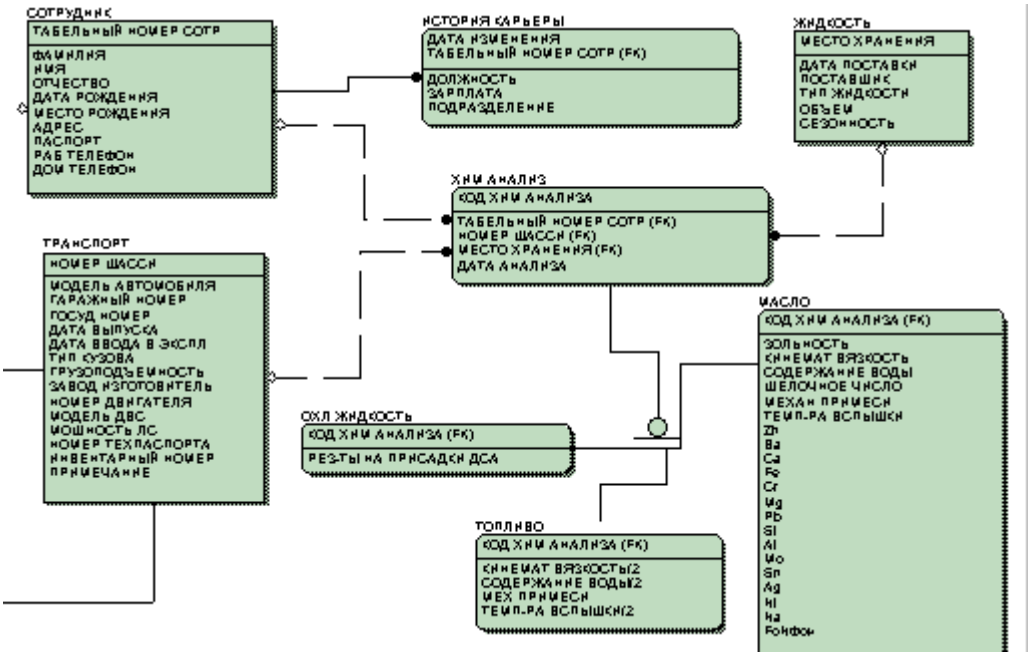


Рис. 13.2

НОМЕР ШАССИ - номер шасси проверяемого автосамосвала
РЕЗУЛЬТАТЫ ИСПЫТАНИЙ - результаты диагностики гидравлики

4) ДИАГНОСТИКА ДИЗЕЛЯ - результаты диагностики дизеля

ДАТА ИСПЫТАНИЙ- дата проведения диагностики
ТИП ДИАГНОСТИКИ - тип проводимых испытаний
НОМЕР ШАССИ - номер шасси проверяемого автосамосвала
ДАВЛЕНИЕ МАСЛА - давление масла в системе
ДАВЛЕНИЕ ТУРБОНАДДУВА ЛЕВ - давление турбонаддува левого
ДАВЛЕНИЕ ТУРБОНАДДУВА ПРАВ - давление турбонаддува правого
ДАВЛЕНИЕ В ТОПЛ МАГИСТ - давление в топливной магистрали
МОЩНОСТЬ ДИЗЕЛЯ - мощность двигателя в л/с

5) ДИАГНОСТИКА ТРАНСМИССИИ - результаты диагностики трансмиссии

ДАТА ИСПЫТАНИЙ - дата проведения диагностики
НОМЕР ШАССИ - номер шасси проверяемого автосамосвала
ТИП ДИАГНОСТИКИ - тип проводимых испытаний
I-A - ток в амперах
P-кВт - мощность в киловаттах
U-b - напряжение в вольтах
ОБ-МИН ПО ПАСПОРТУ - количество оборотов в минуту по паспорту (по 11 точкам)
ОБ-МИН ПЕРЕД НАЛАДКОЙ - количество оборотов в минуту перед наладкой (по 11 точкам)
ОБ-МИН ПОСЛЕ НАЛАДКИ- количество оборотов в минуту после наладки (по 11 точкам)

6) ЖИДКОСТЬ - топлива, масла и охлаждающие жидкости, имеющиеся в запасе на автобазе

МЕСТО ХРАНЕНИЯ - код тары, в которой хранится жидкость
ДАТА ПОСТАВКИ - дата поставки жидкости
ПОСТАВЩИК - завод-изготовитель
ТИП ЖИДКОСТИ - топливо, масло или охлаждающая жидкость
ОБЪЕМ - объем жидкости
СЕЗОННОСТЬ - идентификатор времени применения

7) ИСТОРИЯ КАРЬЕРЫ - движения сотрудников автобазы в рамках ее оргштатной структуры

ДАТА ИЗМЕНЕНИЯ - дата изменения должности, зарплаты или подразделения автобазы, в котором работает сотрудник
ТАБЕЛЬНЫЙ НОМЕР СОТР - табельный номер сотрудника
ДОЛЖНОСТЬ - занимаемая сотрудником автобазы должность
ЗАРПЛАТА - зарплата сотрудника
ПОДРАЗДЕЛЕНИЕ - код подразделения, в котором работает сотрудник

8) МАСЛО - результаты химического анализа масел

КОД ХИМ АНАЛИЗА - уникальный внутренний для автобазы номер проводимого химического анализа
ЗОЛЬНОСТЬ
КИНЕМАТ ВЯЗКОСТЬ
СОДЕРЖАНИЕ ВОДЫ
ЩЕЛОЧНОЕ ЧИСЛО

МЕХАН ПРИМЕСИ
ТЕМП-РА ВСПЫШКИ
Результаты анализа (Ag, Al, Ba, Ca, Cr, Fe, Co, Фон, Mg, Mo, Na, Ni, Pb, Si, Sn, Zn)

9) ОХЛ ЖИДКОСТЬ - результаты химического анализа охлаждающих жидкостей

КОД ХИМ АНАЛИЗА - уникальный внутренний для автобазы номер проводимого химического анализа
РЕЗУЛЬТАТЫ НА ПРИСАДКИ ДСА - результаты анализа

10) СОТРУДНИК - данные о сотрудниках автобазы

ТАБЕЛЬНЫЙ НОМЕР СОТР - табельный номер сотрудника
ФАМИЛИЯ - фамилия сотрудника
ИМЯ - имя сотрудника
ОТЧЕСТВО - отчество сотрудника
ДАТА РОЖДЕНИЯ - дата рождения сотрудника
МЕСТО РОЖДЕНИЯ - место рождения сотрудника АДРЕС - домашний адрес сотрудника
ПАСПОРТ - паспортные данные сотрудника
РАБ ТЕЛЕФОН - рабочий телефон сотрудника
ДОМ ТЕЛЕФОН - домашний телефон сотрудника

11) ТОПЛИВО - результаты химического анализа топлива

КОД ХИМ АНАЛИЗА - уникальный внутренний для автобазы номер проводимого химического анализа
КИНЕМАТ ВЯЗКОСТЬ
СОДЕРЖАНИЕ ВОДЫ
МЕХ ПРИМЕСИ
ТЕМПЕРАТУРА ВСПЫШКИ

12) ТРАНСПОРТ - данные по автопарку

НОМЕР ШАССИ - номер шасси автомобиля
МОДЕЛЬ АВТОМОБИЛЯ
ГАРАЖНЫЙ НОМЕР - номер, присвоенный автомобилю на автобазе (только для технологического транспорта)
ГОС. НОМЕР - государственный номер автомобиля (только для хозяйственного транспорта)
ДАТА ВЫПУСКА - дата выпуска автомобиля заводом-изготовителем
ДАТА ВВОДА В ЭКСПЛ - дата ввода автомобиля в эксплуатацию на автобазе
ТИП КУЗОВА - тип кузова автомобиля (самосвал, тягач, автоцистерна, легковой, бортовой, сед. тягач, полуприцеп, прицеп бортовой, микроавтобус и др.)
ГРУЗОПОДЪЕМНОСТЬ - отсутствует для легков., сед. тягача, микроавтобуса
ЗАВОД ИЗГОТОВИТЕЛЬ - название завода изготовителя
НОМЕР ДВИГАТЕЛЯ - номер двигателя автомобиля
МОДЕЛЬ ДВС - модель двигателя
МОЩНОСТЬ ЛС - мощность двигателя в лошадиных силах
НОМЕР ТЕХПАСПОРТА - только для хозяйственного транспорта
ИНВЕНТАРНЫЙ НОМЕР - номер инвентаризации
ПРИМЕЧАНИЕ - цвет и т.п.

13) ХИМ АНАЛИЗ - результаты химического анализа топлива, масел и охлаждающих жидкостей

КОД ХИМ АНАЛИЗА - уникальный внутренний для автобазы номер проводимого химического анализа

НОМЕР ШАССИ - номер шасси автомобиля
МЕСТО ХРАНЕНИЯ - код тары, в которой хранится жидкость
ДАТА АНАЛИЗА - дата проведенного анализа
ТАБЕЛЬНЫЙ НОМЕР СОТР - табельный номер сотрудника, проводившего анализ

2.2) Взаимосвязи информационной и функциональной моделей

Соответствие сущностей информационной модели и накопителей данных функциональной модели приведено в таблице 13.1.

3) Состав и структура автоматизированных рабочих мест ремонтной службы

3.1) АРМ ДИАГНОСТИКА

Функции АРМ ДИАГНОСТИКА (рис. 13.3)

- учет выполненной диагностики по электрической трансмиссии
- учет выполненной диагностики по дизелю
- учет выполненной диагностики по гидравлической системе

Таблица 13.1

Сущность	Накопитель
дефектоскопия	технологический транспорт
диагностика	технологический транспорт
диагностика гидравлики	технологический транспорт
диагностика дизеля	технологический транспорт
диагностика трансмиссии	технологический транспорт
жидкость	масла, топливо и охл. жидкость
история карьеры	сотрудники
масло	масла, топливо и охл. жидкость
охл жидкость	масла, топливо и охл. жидкость
сотрудник	сотрудники
топливо	масла, топливо и охл. жидкость
транспорт	технологический транспорт
хим анализ	масла, топливо и охл. жидкость

3.1.1) Учет выполненной диагностики по электрической трансмиссии

1) Занесение в таблицу **ДИАГНОСТИКА** следующей информации:

- дата испытаний
- номер шасси
- тип диагностики (электрическая трансмиссия)
- табельный номер сотрудника

2) Занесение в таблицу **ДИАГНОСТИКА ТРАНСМИССИИ** следующей информации:

- U(b)
- I(A)
- P(кВт)
- n(об/мин) по паспорту, перед наладкой и после наладки по 11 точкам измерений

3.1.2) Учет выполненной диагностики по дизелю

1) Занесение в таблицу **ДИАГНОСТИКА** следующей информации:

- дата испытаний
- номер шасси
- тип диагностики (дизель)
- табельный номер сотрудника

2) Занесение в таблицу **ДИАГНОСТИКА ДИЗЕЛЯ** следующей информации:

- давление масла в магистрали смазки
- давление турбонаддува (левого и правого)
- давление в топливной магистрали между топливным насосом и форсунками
- мощность дизеля

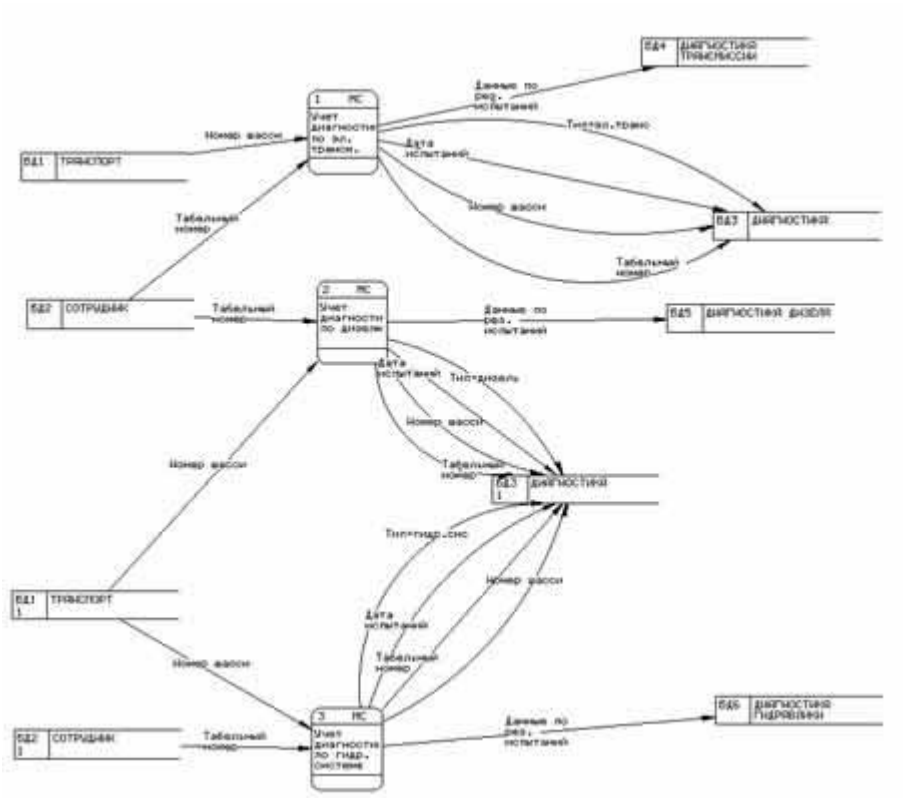


Рис. 13.3. АРМ ДИАГНОСТИКА

3.1.3) Учет выполненной диагностики по гидравлической системе

1) Занесение в таблицу **ДИАГНОСТИКА** следующей информации:

- дата испытаний
- номер шасси
- тип диагностики (гидравлическая система)
- табельный номер сотрудника

2) Занесение в таблицу **ДИАГНОСТИКА ГИДРАВЛИКИ** следующей информации:

- данные по результатам испытаний

<3.2) АРМ ХИМИЧЕСКИЙ АНАЛИЗ

Функции АРМ ХИМИЧЕСКИЙ АНАЛИЗ (рис. 13.4)

- учет результатов химического анализа масел
- учет результатов химического анализа топлива
- учет результатов химического анализа охлаждающих жидкостей

<3.2.1) Учет результатов химического анализа масел

1) Занесение в таблицу **ХИМ АНАЛИЗ** следующей информации:

- дата испытаний
- номер шасси или место хранения
- тип химического анализа (масла)
- табельный номер сотрудника

2) Занесение в таблицу **МАСЛО** следующей информации:

- зольность масла
- кинематическая вязкость
- содержание воды
- щелочное число
- механические примеси
- температура вспышки
- параметры по спектральному анализу (Zh, Ba, Ca, Fe, Cr, Mg, Pb, Si, Al, Mo, Sn, Ag,Ni, FoN Фон, Na)

3.2.2) Учет результатов химического анализа топлива

1) Занесение в таблицу **ХИМ АНАЛИЗ** следующей информации:

- дата испытаний
- номер шасси или место хранения
- тип химического анализа (топлива)
- табельный номер сотрудника

2) Занесение в таблицу **ТОПЛИВО** следующей информации:

- кинематическая вязкость
- содержание воды
- механические примеси, определенные методом фильтрации
- температура вспышки
- сезонность

3.2.3) Учет результатов химического анализа охлаждающих жидкостей

1) Занесение в таблицу **ХИМ АНАЛИЗ** следующей информации:

- дата испытаний
- номер шасси или место хранения
- тип химического анализа (охлаждающие жидкости)
- табельный номер сотрудника

2) Занесение в таблицу **ОХЛ ЖИДКОСТЬ** следующей информации:

<>- результат анализа на присадки ДСА

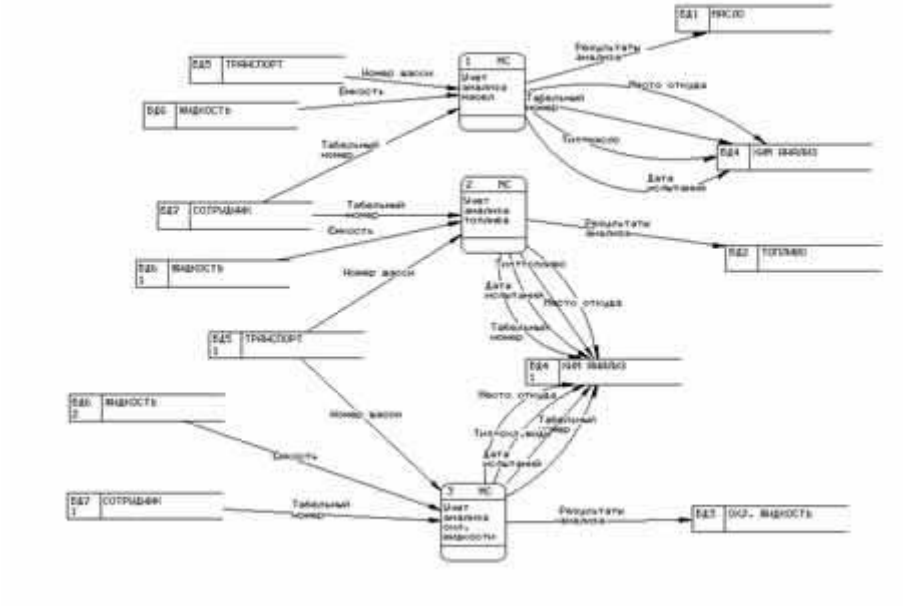


Рис. 13.3. АРМ ХИМИЧЕСКИЙ АНАЛИЗ

ЧАСТЬ 4
CASE-СРЕДСТВА АВТОМАТИЗАЦИИ МЕТОДОЛОГИЙ СТРУКТУРНОГО СИСТЕМНОГО АНАЛИЗА И ПРОЕКТИРОВАНИЯ

Четвертая часть книги посвящена описанию CASE-средств автоматизации методологий структурного системного анализа и проектирования - инструмента современного консультанта.

В главе 14 рассматриваются концептуальные основы CASE-технологий, прослеживается эволюция CASE как самостоятельной дисциплины в программотехнике, приводится CASE-модель жизненного цикла программного продукта и рассматриваются ее отличия от традиционной модели. Анализируется состав, структура и функциональные особенности современных CASE-средств.

В главе 15 приводится классификация CASE-средств по типам, категориям и уровням.

В главе 16 рассматривается отечественное CASE-средство первого поколения - пакет CASE.Аналитик. Приведено описание основных функций пакета, а также основные особенности используемых средств структурного системного анализа.

Глава 17 посвящена аналитическому обзору российского рынка CASE-средств.

ГЛАВА 14
КОНЦЕПТУАЛЬНЫЕ ОСНОВЫ CASE - ТЕХНОЛОГИЙ

14.1. Эволюция CASE - средств

С самого начала CASE-технологии развивались с целью преодоления ограничений ручных применений методологий структурного анализа и проектирования 60-70-х годов (сложности понимания, большой трудоемкости и стоимости использования, трудности внесения изменений в проектные спецификации и т.д.) за счет их автоматизации и интеграции поддерживающих средств. Таким образом CASE-технологии не могут считаться самостоятельными методологиями, они только делают более эффективными пути их применения. CASE - не революция в программотехнике: современные CASE-средства являются естественным продолжением эволюции всей отрасли средств разработки ПО. Традиционно выделяют шесть периодов, качественно отличающихся применяемой техникой и методами разработки ПО, которые характеризуются использованием в качестве инструментальных следующих средств:

- *ассемблеров, дампов памяти, анализаторов;*
- *компиляторов, интерпретаторов, трассировщиков;*
- *символических отладчиков, пакетов программ;*
- *систем анализа и управления исходными текстами;*
- *CASE-средств анализа требований, проектирования спецификаций иструктуры, редактирования интерфейсов (**первая генерация CASE-I);***
- *CASE-средств генерации исходных текстов и реализации интегрированного окружения поддержки полного жизненного цикла (ЖЦ) разработки ПО(**вторая генерация CASE-II).***

CASE-I является первой технологией, адресованной непосредственносистемным аналитикам и проектировщикам,и включающей средства для поддержки графических моделей, проектирования спецификаций, экранных редакторов и словарей данных. Она не предназначена для поддержки полного ЖЦ и концентрирует внимание на функциональных спецификациях и начальных шагах проекта - системном анализе, определении требований, системном проектировании, логическом проектировании БД.

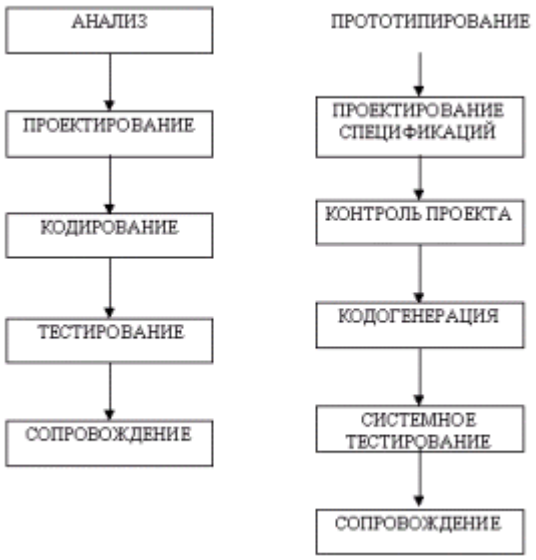
CASE-II отличается значительно более развитыми возможностями,улучшенными характеристиками и исчерпывающим подходом к полному ЖЦ.В ней в первую очередь используются средства поддержки автоматическойкодогенерации, а также обеспечивается полная функциональная поддержкадля порождения графических системных требований и спецификаций проектирования; контроля,анализа и связывания системной информации,а такжеинформации по управлению проектированием; построения прототипов и моделей системы; тестирования, верификации и анализа сгенерированныхпрограмм; генерации документов по проекту; контроля на соответствиестандартам по всем этапам ЖЦ. CASE-II может включать свыше 100 функциональных компонент, поддерживающих все этапы ЖЦ, при этом пользователям предоставляется возможность выбора необходимых средств и их интеграции в нужном составе.

14.2. CASE-модель жизненного цикла ПО

CASE-технологии предлагают новый, основанный на автоматизацииподход к концепции ЖЦ ПО. При использовании CASE изменяются все фазыЖЦ, при этом наибольшие изменения касаются фаз анализа и проектирования. На рис.14.1 приводится простейшая модель ЖЦ (рис.14.1а) и соответствующая CASE-модель (рис.14style='font-size:12.0pt;'>.1б), в которой фаза прототипирования заменяеттрадиционную фазу системного анализа. Необходимо отметить, что наиболее автоматизируемыми фазами являются фазы контроля проекта и кодогенерации (хотя все остальные фазы также поддерживаются CASE-средствами).

В таблице 14.1 приведены оценки трудозатрат по фазам ЖЦ. Перваястрока таблицы соответствует традиционной разработке, вторая - разработке с использованием структурных методологий проектирования, третья - разработке с использованием CASE-технологий.В таблицу 14.2 сведены основные изменения в ЖЦ при использованииCASE-технологий по сравнению с традиционной разработкой.

На рис. 14.2 представлены результаты сравнения традиционной разработки программных проектов и разработки с применением CASE-технологий.



А) В)

Рис. 14.1. Модели жизненного цикла

Таблица 14.1

Способ разработки	**Анализ**	**Проектирование**	**Кодирование**	**Тестирование**
Традиционная разработка	20%	15%	20%	45%
Использование структурных методологий проектирования	30%	30%	15%	25%
Использование CASE-технологий	40%	40%	5%	15%

Таблица 14.2

	Традиционная разработка	**CASE**
1	Основные усилия – на кодирование и тестирование	Основные усилия - на анализ и проектирование
2	“Бумажные” спецификации	Быстрое итеративное прототипирование
3	Ручное кодирование	Автоматическая кодогенерация
4	Ручное документирование	Автоматическая генерация документации
5	Тестирование кодов	Автоматический контроль проекта
6	Сопровождение кодов	Сопровождение спецификаций проектирования

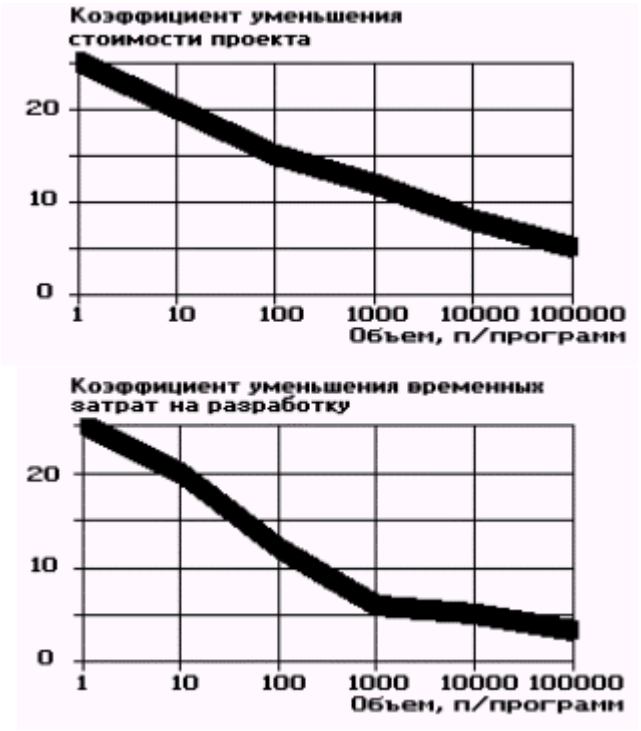


Рис. 14.2. Уменьшение затрат на проектирование программных систем за счет CASE-технологий

14.3. Состав, структура и функциональные особенности CASE-средств

CASE-средства служат инструментарием для поддержки и усиления методов структурного анализа и проектирования. Эти инструменты поддерживают работу пользователей при создании и редактировании графического проекта в интерактивном режиме, онespecобствуют организации проекта в виде иерархии уровней абстракции,выполняют проверки соответствия компонентов.Фактически CASE-средства представляют собой новый тип графически-ориентированных инструментов, восходящих к системе поддержки ЖЦ ПО.Обычно к ним относят любое программное средство, обеспечивающее автоматическую помощь при разработке ПО,его сопровождении или деятельности по управлению проектом,и проявляющее следующие дополнительные черты:

- мощная графика для описания и документирования систем ПО, а такжедля улучшения интерфейса с пользователем, развивающая творческие возможности специалистов и не отвлекающая их от процесса проектированияна решение второстепенных вопросов;
- интеграция,обеспечивающая легкость передачи данных между средствами и позволяющая управлять всем процессом проектирования и разработки ПО непосредственно через процесс планирования проекта;
- использование компьютерного хранилища (репозитария) для всей информации о проекте, которая может разделяться между разработчиками и исполнителями как основа для автоматического продуцирования ПО и повторного его использования в будущих системах.

Помимо перечисленных основополагающих принципов графической ориентации, интеграции и локализации всей проектной информации в репозитарии в основе концептуального построения CASE-средств лежат следующие положения:

1. Человеческий фактор, определяющий разработку ПО как легкий,удобный и экономичный процесс.
2. Широкое использование базовых программных средств, получившихмассовое распространение в других приложениях (БД и СУБД, компиляторыс различных языков программирования, отладчики, документаторы, издательские системы, оболочки экспертных систем и базы знаний, языкичетвертого поколения и др.).

3. Автоматизированная или автоматическая кодогенерация,выполняющаянесколько видов генерации кодов: преобразования для получения документации, формирования БД,ввода/модификации данных,получения выполняемыхмашинных кодов из спецификаций ПО, автоматической сборки модулей изсловарей и моделей данных и повторно используемых программ, автоматической конверсии ранее используемых файлов в форматы новых требований.
4. Ограничение сложности, позволяющее получать компоненты, поддающиеся управлению, обозримые и доступные для понимания, а также обладающие простой и ясной структурой.
5. Доступность для разных категорий пользователей.
6. Рентабельность.
7. Сопровождаемость,обеспечивающая способность адаптации при изменении требований и целей проекта.

Интегрированный CASE-пакет содержит четыре основные компоненты:

1) Средства централизованного хранения всей информации о проектируемом ПО в течении всего ЖЦ (**репозитарий**) являются основой CASE-пакета. Соответствущая БД должна иметь возможность поддерживать большую систему описаний и характеристик и предусматривать надежные меры позащите от ошибок и потерь информации. Репозитарий должен обеспечивать:

- инкрементный режим при вводе описаний объектов;
- распространение действия нового или скорректированного описания на информационное пространство всего проекта;
- синхронизацию поступления информации от различных пользователей;
- хранение версий проекта и его отдельных компонент;
- сборку любой запрошенной версии;
- контроль информации на корректность, полноту и состоятельность.

2) Средства **ввода** предназначены для ввода данных в репозитарий, а также для организации взаимодействия с CASE-пакетом. Этисредства должны поддерживать различные методологии и использоватьсяна всем ЖЦ разными категориями разработчиков: аналитиками,проектировщиками, инженерами, администраторами и т. д.

3) Средства **анализа, проектирования и разработки** предназначены длятого, чтобы обеспечить планирование и анализ различных описаний, а также их преобразования в процессе разработки.

4) Средства **вывода** служат для документирования, управления проектами кодовой генерации.

Все перечисленные компоненты в совокупности должны:

- поддерживать графические модели;
- контролировать ошибки;
- организовывать и поддерживать репозитарий;
- поддерживать процесс проектирования и разработки.

14.4. Поддержка графических моделей

Графическая ориентация CASE заключается в том, что программы являются схематическими проектами и формами, которые много проще в использовании, чем многостраничные описания. Для представления программ применяются структурные диаграммы различных типов, дополнительное достоинство которых заключается в их использованиии в качестве наглядной “двумерной” документации по проекту.

Для CASE существенны 4 типа диаграмм: диаграммы функционального проектирования (для этих целей наиболее часто употребляются DFD - диаграммы потоков данных), диаграммы моделирования данных (как правило, ERD - диаграммы “сущность-связь”), диаграммы моделирования поведения (как правило, STD - диаграммы переходов состояний) и структурные

диаграммы (карты), применяющиеся на этапе проектирования и описывающие отношения между модулями и внутримодульную структуру. Создание и модификация подобных диаграмм осуществляется с помощью специальных графических редакторов (диаграммеров), являющихся сервисными средствами на этапах анализа требований и проектирования спецификаций. Современные диаграммеры обеспечивают:

- создание иерархически связанных диаграмм, в которых комбинируются графические и текстовые объекты;
- создание и редактирование объектов в любом месте диаграммы;
- создание, перемещение и выравнивание групп объектов, изменение их размеров, масштабирование;
- сохранение связей между объектами при их перемещении и изменении размеров;
- автоматический контроль ошибок и др.

Реализация подобных возможностей позволяет пользователю целиком сосредоточится на собственно проектировании, не отвлекаясь на решение второстепенных вопросов, связанных с размещением элементов диаграмм, их компоновкой и т.п.

Полученные диаграммы дают ясное понимание и решение проблемы, позволяют проанализировать функционирование создаваемого ПО, фиксируют связи между разработчиками, пользователями и руководителями, обеспечивают стандартизацию представления структуры программы и данных.

14.5. Контроль ошибок

Важность контроля ошибок на этапах анализа требований и проектирования спецификаций обуславливается возможностью их автоматического обнаружения на ранних этапах ЖЦ. CASE обеспечивает автоматическую верификацию и контроль проекта на полноту и состоятельность на ранних этапах ЖЦ, что влияет на успех разработки в целом. В подтверждение этого можно привести следующие статистические данные, основанные на отчетах фирмы TRW по анализу 5 крупных проектов:

- при традиционной организации работ ошибки проектирования и кодирования составляют, соответственно, 64% и 32% от общего числа ошибок;
- ошибки проектирования в 100 раз труднее обнаружить на этапе сопровождения ПО, чем на этапах анализа требований и проектирования спецификаций.

В CASE диаграммеры и верификаторы способны осуществлять следующие типы контроля:

1) *Контроль синтаксиса диаграмм и типов их элементов*. Обычно такой контроль осуществляется при вводе и редактировании элементов диаграмм. Примеры контролируемых ситуаций:

- *по синтаксису:* любой функциональный элемент диаграммы должен иметь по крайней мере один входной и один выходной поток; два элемента данных не могут быть непосредственно связаны;
- *по типам:* функциональный элемент должен всегда использоваться для представления процедурной компоненты; поток данных всегда должен быть представлен компонентой данных.

2) *Контроль полноты и состоятельности диаграмм:* все элементы диаграмм должны быть идентифицированы и отражены в репозитарии. Например, для DFD контролируются неименованные или несвязанные потоки данных, процессы и хранилища данных; источники и стоки данных (внешние сущности) вне контекстной диаграммы; хранилища данных на контекстной диаграмме и т.п. При анализе словаря данных необходимо выявлять циклические определения, эквивалентные определения, неопределенные объекты.

3) *Контроль декомпозиции функций* включает оценку качества на основе различных метрик ПО и частичный семантический контроль.

4) Сквозной контроль диаграмм одного или различных типов на предмет их состоятельности по уровням - *вертикальное и горизонтальное балансирование диаграмм*. При вертикальном балансировании (диаграммы одного типа) выявляются несбалансированные потоки данных между детализируемой и детализирующей диаграммами. Горизонтальное балансирование определяет некорректности между DFD, ERD, STD, словарями данных и миниспецификациями процессов. Так при балансировании DFD-ERD контролируется соответствие каждого хранилища данных на DFD сущности или отношению на ERD. Контроль DFD-STD осуществляется по следующим правилам: каждый управляющий процесс на DFD детализируется спецификацией управления STD, и наоборот, каждой STD должен соответствовать управляющий процесс; каждое условие (действие) в STD должно соответствовать входному (выходному) управляющему потоку на DFD, и наоборот, каждому управляющему потоку в зависимости от его направленности должно соответствовать условие/действие на STD. При балансировании DFD-словарь данных-миниспецификация доложны проверяться следующие правила:

- каждый поток и хранилище данных должны быть определены в словаре данных (контроль неопределенных значений), и наоборот, каждое определение в словаре должно быть отражено на диаграмме, в миниспецификации или другом определении (контроль неиспользуемых значений);
- каждый процесс на DFD должен детализироваться с помощью DFD или миниспецификации (но не тем и другим одновременно), и наоборот, каждая миниспецификация должна соответствовать единственному процессу;
- ссылки к данным в миниспецификациях должны соответствовать объектам на диаграммах и в словаре данных;
- по возможности должна контролироваться семантика миниспецификации: например, если входные и/или выходные потоки связаны с хранилищем данных, то это должно быть отражено в миниспецификации (операторами READ, GET, WRITE, PUT и т.п.).

14.6. Организация и поддержуа репозитария

Основные функции средств организации и поддержки репозитария - хранение, доступ, обновление, анализ и визуализация всей информации по проекту ПО. Содержимое репозитария включает не только информационные объекты различных типов, но и отношения между их компонентами, а также правила использования или обработки этих компонент (рис. 14.3). Репозитарий может хранить свыше 100 типов объектов, примерами которых являются структурные диаграммы, определения экранов и меню, проекты отчетов, описания данных, логика обработки, модели данных, модели организации, модели обработки, исходные коды, элементы данных и т.п.

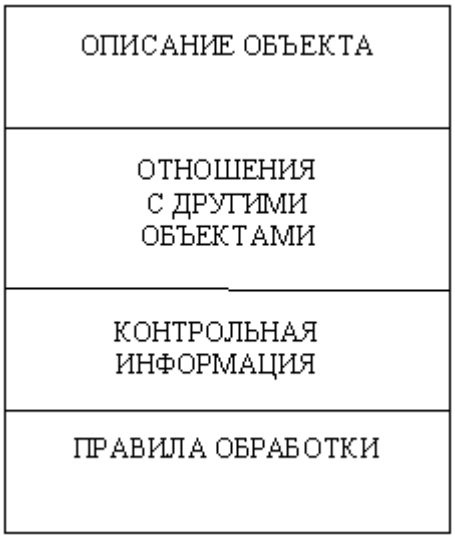


Рис. 14.3. Содержимое репозитария

Каждый информационный объект в репозитарии описывается перечислением его свойств: идентификатор, имена-синонимы, тип, текстовое описание, компоненты, файл-хранилище, область значений. Кроме этого, хранятся все отношения с другими объектами (например, все объекты, в которых данный объект используется; все перекрестные ссылки), правила формирования и редактирования объекта, а также контрольная информация о времени порождения объекта, времени его последнего обновления, кем и в каком проекте он был порожден, номере версии, возможности обновления и т.п.

На основе репозитария осуществляется интеграция CASE-средств и разделение системной информации между разработчиками. При этом возможности репозитария обеспечивают несколько уровней интеграции: общий пользовательский интерфейс по всем средствам, передачу данных между средствами, интеграцию этапов разработки через единую систему представлений фаз ЖЦ, передачу данных и средств между аппаратурными платформами.

Репозитарий является базой для стандартизации документации по проекту и контроля состоятельности проектных спецификаций. Все отчеты строятся автоматически по репозитарию, ниже перечислены основные их типы:

1. *Отчеты по содержимому* включают сводки потоков данных и их компонент, сводки всех пар интерфейсов в описывающих межмодульные отношения структурных диаграммах, списки входных и выходных потоков для каждого функционального блока диаграмм, списки измененных за определенный период объектов, истории всех изменений объектов, описания модулей, планы тестирования модулей и подпрограмм, списки всех данных и их атрибутов, а также отношений между их компонентами и правил их обработки.
2. *Отчеты по перекрестным ссылкам* включают списки всех вызывающих и вызываемых модулей; списки объектов репозитария, к которым имеет доступ конкретный разработчик; сводки диаграмм, использующих конкретные данные; маршруты движения данных от входа к выходу.
3. *Отчеты по результатам анализа* включают сводки балансирования диаграмм по уровням, списки неопределенных информационных объектов, списки неполных диаграмм, сводки результатов анализа структуры проекта, списки несогласованных в диаграммах и репозитарии объектов, списки неиспользуемых объектов, списки удаленных объектов.
4. *Отчеты по декомпозиции объектов* включают таблицы иерархии всех объектов модели.

Пример отчета по функциональным блокам SADT-модели управления банком, автоматически создаваемого пакетом Design/IDEF, приведен ниже.

Activity Report

[A0] Банк
Inputs: Платежные документы
Outputs: Деньги
Controls: Законы, Время, Баланс
Mechanisms: Техника, Сотрудники
Sub-Activities: [A1] Операционные залы,
[A2] Управление банком,
[A3] Центральный банк

[A1] Операционные залы
Inputs: Платежные документы
Outputs: Принятые платежные документы
Controls: Законы, Продолжит. раб. дня, Остатки счетов клиентов
Mechanisms: Сотрудники, Терминал БД

[A2] Управление банком
Inputs: Принятые платежные документы
Outputs: (Unlabled), Деньги, (Unlabled)
Controls: Спец. законы, Расчет баланса, Срок обработки
Mechanisms: Управленческий персонал, Компьютеры

[A3] Центральный банк
Inputs: (Unlabled)
Outputs: Деньги, (Unlabled)
Controls: Срок отправки
Mechanisms: Экспедиторы, Автомашины

Важные функции управления и контроля проекта также реализуются на основе репозитария. В частности через репозитарий может осуществляться контроль безопасности (ограничения доступа, привелегии доступа), контроль версий, контроль изменений и др.

14.7. Поддержка процесса проектирования и разработки

При поддержке процесса проектирования и разработки основную роль играют следующие возможности CASE-пакетов: покрытие ЖЦ, поддержка прототипирования, поддержка структурных методологий, автоматическая кодогенерация.

При *покрытии ЖЦ* наибольшее внимание уделяется его наиболее критичным этапам - анализу требований и проектированию спецификаций. Последние являются основой всего проекта, поэтому их полнота и корректность влияют на успех разработки в целом.

Важную роль при автоматизации ранних этапов ЖЦ играют возможности *поддержки прототипирования.* Соответствующие средства используются для определения системных требований и ответа на вопросы об ожидаемом поведении системы. Такие средства как генераторы меню, экранов и отчетов позволяют быстро построить прототипы пользовательских интерфейсов и снабдить моделью функционирования системы с позиций конечного пользователя. Использование языков четвертого поколения (4GL) позволяет строить более сложные модели, при этом прототип позволяет промоделировать основные функции системы, но не способен контролировать ее ожидаемое поведение. Исполняемые языки спецификаций преобразуют процесс разработки в следующий итеративный процесс: спецификации определяются и выполняются, затем производится переопределение или корректировка. Созданные таким образом прототипы позволяют определять, является ли проектируемая система полной и корректной.

Поддержка структурных методологий осуществляется за счет средств их автоматизации на следующих двух уровнях:

- подготовка документации, графическая поддержка построения структурных диаграмм различных типов, продуцирование спецификаций для детализации функциональных блоков в диаграммах и структур данных на нижних уровнях (для таких спецификаций введен специальный термин - “миниспецификация”);
- корректное использование шагов обработки в методологиях.

Кодогенерация осуществляется на основе репозитария и позволяет автоматически построить до 80-90% объектных кодов или текстов программ на языках высокого уровня. При этом различными CASE-пакетами поддерживаются практически все известные языки программирования, однако наиболее часто в качестве целевых языков выступают COBOL, C и ADA. Средства кодогенерации по отношению к полноте целевого продукта разделяются на средства генерации каркаса ПО и средства генерации полного продукта. В первом случае автоматически строится откомментированная логика (потоки управления) ПО, а также коды для БД, файлов, экранов, отчетов и т.п., остальные фрагменты ПО кодируются вручную. Во втором случае из проектных спецификаций генерируется полная документированная программа, включая выполняемый код, пользовательскую и программную документацию, наборы тестов и т.д. Все эти компоненты

полной программы связываются в единый объект, хранящийся в репозитарии для облегчения доступа и сопровождения.

Идея автоматической кодогенерации на основе модели заключается в следующем. Любая программа схематически может быть представлена в виде тройки: обрабатываемые данные, логический каркас обработки, линейные участки обработки. Схема базы данных может быть легко сгенерована на основании модели “сущность-связь”, и современные средства информационного моделирования (например, ERWin, Designer/2000 и др.) обеспечивают такую генерацию. Логика обработки генерируется на основе диаграмм потоков данных: известны алгоритмы автоматического преобразования иерархии DFD в структурные карты, а с задачей получения из структурных карт соответствующих кодов легко справляется теория компиляции. Наконец, линейным участкам соответствуют миниспецификации модели. И именно здесь лежит ключ к высокому проценту автоматически сгенерированного кода, существенно зависящему от метода задания миниспецификаций.

ГЛАВА 15
КЛАССИФИКАЦИЯ CASE - СРЕДСТВ

Все CASE-средства делятся на типы, категории и уровни. Классификация ***по типам*** отражает функциональную ориентацию CASE-средств в технологическом процессе.

1) ***АНАЛИЗ И ПРОЕКТИРОВАНИЕ***. Средства данной группы используются для создания спецификаций системы и ее проектирования; они поддерживают широко известные методологии проектирования (см. часть 2). К таким средствам относятся: *CASE.Аналитик (Эйтэкс), The Developer (ASYST Technologies), POSE (Computer Systems Advisers), ProKit*Workbench (McDonnell Douglas), Excelerator (Index Technology), Design-Aid (Nastec), Design Machine (Optima), MicroStep (Meta Systems), vsDesigner (Visual Software), Analist/Designer (Yourdon), Design/IDEF (Meta Software), BPWin (Logic Works), SELECT (Select Software Tools), System Architect (Popkin Software & Systems), Westmount I-CASE Yourdon (Westmount Technology B.V. & CADRE Technologies), CASE/4/0 (microTOOL GmbH).* Их целью является определение системных требований и свойств, которыми система должна обладать, а также создание проекта системы, удовлетворяющей этим требованиям и обладающей соответствующими свойствами. На выходе продуцируются спецификации компонент системы и интерфейсов, связывающих эти компоненты, а также “калька” архитектуры системы и детальная “калька” проекта, включающая алгоритмы и определения структур данных.

2) ***ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ И ФАЙЛОВ***. Средства данной группы обеспечивают логическое моделирование данных, автоматическое преобразование моделей данных в Третью Нормальную Форму, автоматическую генерацию схем БД и описаний форматов файлов на уровне программного кода: *ERWin (Logic Works), Chen Toolkit (Chen & Asssociates), S-Designor (SDP), Designer2000 (Oracle), Silverrun (Computer Systems Advisers).*

3) ***ПРОГРАММИРОВАНИЕ***. Средства этой группы поддерживают этапы программирования и тестирования, а также автоматическую кодогенерацию из спецификаций, получая полностью документированную выполняемую программу: *COBOL 2/Workbench (Mikro Focus), DECASE (DEC), NETRON/CAP (Netron), APS (Sage Software).* Помимо диаграммеров различного назначения и средств поддержки работы с репозитарием, в эту группу средств включены и традиционные генераторы кодов, анализаторы кодов (как в статике, так и в динамике), генераторы наборов тестов, анализаторы покрытия тестами, отладчики.

4) ***СОПРОВОЖДЕНИЕ И РЕИНЖИНИРИНГ***. К таким средствам относятся документаторы, анализаторы программ, средства реструктурирования и реинжениринга: *Adpac CASE Tools (Adpac), Scan/COBOL u SuperStructure (Computer Data Systems), Inspector/Recoder (Language Technology).* Их целью является корректировка, изменение, анализ, преобразование и реинжениринг существующей системы. Средства позволяют осуществлять поддержку всей системной документации, включая коды, спецификации, наборы тестов; контролировать покрытие тестами для оценки полноты тестируемости; управлять функционированием системы и т.п. Особый интерес представляют средства обеспечения мобильности (в CASE они получили

название средств миграции) и реинжиниринга. К средствам миграции относятся трансляторы, конверторы, макрогенераторы и др., позволяющие обеспечить перенос существующей системы в новое операционное или аппаратурное окружение. Средства реинжиниринга включают:

- статические анализаторы для продуцирования схем системы ПО из ее кодов, оценки влияния модификаций (например,"эффекта ряби" - внесение изменений с целью исправления ошибок порождает новые ошибки);
- динамические анализаторы (обычно, компиляторы и интерпретаторы с встроенными отладочными возможностями);
- документаторы, позволяющие автоматически получать обновленную документацию при изменении кода;
- редакторы кодов, автоматически изменяющие при редактировании и все предшествующие коду структуры (например, спецификации);
- средства доступа к спецификациям, их модификации и генерации нового (модифицированного) кода;
- средства реверсного инжиниринга, транслирующие коды в спецификации.

5) ***ОКРУЖЕНИЕ***. Средства поддержки платформ для интеграции, создания и придания товарного вида CASE-средствам: *Multi/Cam (AGS Management Systems), Design/OA (Meta Software).*

6) ***УПРАВЛЕНИЕ ПРОЕКТОМ***. Средства, поддерживающие планирование, контроль, руководство, взаимодействие, т.е. функции, необходимые в процессе разработки и сопровождения проектов: *Project Workbench (Applied Business Technology).*

Классификация ***по категориям*** определяет уровень интегрированности по выполняемым функциям и включает вспомогательные программы (tools), пакеты разработчика (toolkit) и инструментальные средства (workbench). Категория **tools** обозначает вспомогательный пакет, решающий небольшую автономную задачу, принадлежащую проблеме более широкого масштаба. Категория **toolkit** представляет совокупность интегрированных программных средств, обеспечивающих помощь для одного из классов программных задач; использует репозитарий для всей технической и управляющей информации о проекте, концентрируясь при этом на поддержке, как правило, одной фазы или одного этапа разработки ПО. Категория **workbench** представляет собой интеграцию программных средств, которые поддерживают системный анализ, проектирование и разработку ПО; используют репозитарий, содержащий всю техническую и управляющую информацию о проекте; обеспечивают автоматическую передачу системной информации между разработчиками и этапами разработки; организуют поддержку практически полного ЖЦ (от анализа требований и проектирования ПО до получения документированной выполняемой программы). Workbench, по сравнению с toolkit, обладает более высокой степенью интеграции выполняемых функций, большей самостоятельностью и автономностью использования, а также наличием тесной связи с системными и техническими средствами аппаратно-вычислительной среды, на которой workbench функционирует. По существу, workbench может рассматриваться как автоматизированная рабочая станция, используемая как инструментарий для автоматизации всех или отдельных совокупностей работ по созданию ПО.

Классификация ***по уровням*** связана с областью действия CASE в пределах жизненного цикла ПО. Однако четкие критерии определения границ между уровнями не установлены, поэтому данная классификация имеет, вообще говоря, качественный характер.

Верхние (Upper) CASE часто называют средствами компьютерного планирования. Они призваны повышать эффективность деятельности руководителей фирмы и проекта путем сокращения затрат на определение политики фирмы и на создание общего плана проекта. Этот план включает цели и стратегии их достижения, основные действия в свете целей и задач фирмы, установление стандартов на различные виды взаимосвязей и т.д. Использование верхних CASE позволяет построить модель предметной области, отражающую всю существующую специфику. Она направлена на понимание общего и частного механизмов функционирования, имеющихся возможностей, ресурсов, целей проекта в соответствии с назначением фирмы. Эти средства

позволяют проводить анализ различных сценариев (в том числе наилучших и наихудших), накапливая информацию для принятия оптимальных решений.

Средние (Middle) CASE считаются средствами поддержки этапов анализа требований и проектирования спецификаций и структуры ПО. Их использование существенно сокращает цикл разработки проекта; при этом важную роль играет возможность накопления и хранения знаний, обычно имеющихся только в голове разработчика-аналитика, что позволит использовать накопленные решения при создании других проектов. Основная выгода от использования среднего CASE состоит в значительном облегчении проектирования систем, проектирование превращается в итеративный процесс, включающий следующие действия:

- пользователь обсуждает с аналитиком требования к проектируемой системе;
- аналитик документирует эти требования, используя диаграммы и словари входных данных;
- пользователь проверяет эти диаграммы и словари, при необходимости модифицируя их;
- аналитик отвечает на эти модификации, изменяя соответствующие спецификации.

Кроме того, средние CASE обеспечивают возможности быстрого документирования требований и быстрого прототипирования.

Нижние (Lower) CASE являются средствами разработки ПО (при этом может использоваться до 30% спецификаций, созданных средствами среднего CASE). Они содержат системные словари и графические средства, исключающие необходимость разработки физических спецификаций. Имеются системные спецификации, которые непосредственно переводятся в программные коды разрабатываемой системы (при этом автоматически генерируется до 80-90% кодов). На эти средства возложены также функции тестирования, управления конфигурацией, формирования документации. Главными преимуществами нижних CASE являются: значительное уменьшение времени на разработку, облегчение модификаций, поддержка возможностей прототипирования (совместно со средними CASE).

ГЛАВА 16
ПРИМЕР РЕАЛИЗАЦИИ – ПАКЕТ CASE.АНАЛИТИК

Пакет CASE.Аналитик является единственной отечественной, доведенной до рынка разработкой, которая без всяких натяжек может быть отнесена к CASE первой генерации. В основе пакета лежит методология структурного системного анализа Гейна-Сарсона, применимая к широкому классу систем обработки информации: информационно-вычислительных, АСУ, АСУТП, систем автоматизации делопроизводства, бухгалтерских систем, баз данных, систем автоматизации эксперимента, организационных систем и т.п. Пакет обеспечивает:

- помощь для ясного понимания потребностей пользователя;
- точное взаимодействие между членами бригады разработчиков;
- "принудительный" хороший стиль;
- возможность для пользователя уже с первых шагов разработки "увидеть" и проверить создаваемое программное обеспечение.
- предоставление разработчику машинно-реализованных средств формального описания системы, и, прежде всего, в графической нотации, понятной также заказчику и пользователю создаваемой системы, вовлекаемых, таким образом, в разработку системы на ее ранней стадии, когда еще можно что-либо менять без особых затрат;
- предоставление доступа к любой части проекта;
- контроль полноты и непротиворечивости каждой части системных требований при помощи встроенных средств контроля.

Результат работы в среде пакета - информационно-логическая модель анализируемой системы. Эта модель представляется в виде иерархии диаграмм потоков данных и структурограмм данных. Диаграммы верхних уровней иерархии определяют основные функции/подсистемы системы с внешними входами и выходами и используемыми файлами. Далее эти основные функции/подсистемы детализируются при помощи диаграмм нижнего уровня. Такая функциональная декомпозиция продолжается, создавая таким образом многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором функциональный процесс становится элементарным, невозможным для дальнейшей детализации. Когда дальнейшая детализация логических функций перестает быть полезной, то переходят к выражению внутренней логики процессов при помощи миниспецификаций - алгоритмов преобразования входных потоков в выходные. На рис. 16.1 показаны эти связи.

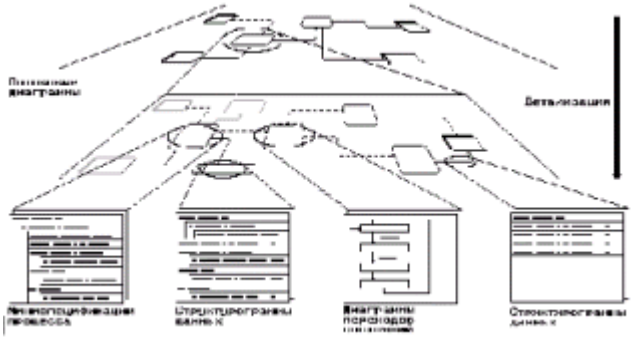


Рис. 16.1. Информационно-логическая модель системы

Информационно-логическая модель образует исчерпывающее описание системы независимо от того, является ли она существующей или же "новой, которую предстоит построить. После подготовки информационно-логической модели имеется логическая функциональная спецификация - подробное описание функций системы, освобожденное от деталей реализации. Такая информационно-логическая модель дает проектировщику четкое представление о конечных результатах.

В состав пакета входят:

1) **База данных проекта**. В базе данных проекта CASE.Аналитик хранит всю информацию о модели системы - как о топологии и иерархии диаграмм, так и о структурных компонентах. При этом пользователю предоставляется графический интерфейс с базой данных и возможность получения разнообразных отчетов по проекту. В CASE.Аналитик используется база данных в формате СУБД Paradox. Для работы с CASE.Аналитик и базой данных проекта, порождаемой CASE.Аналитик, не требуется каких-либо дополнительных программ. В то же время база данных проекта доступна для программ, работающих с форматом Paradox. В этом смысле база данных проекта является открытой.

База данных проекта включает контекстные диаграммы, диаграммы потоков данных, диаграммы управляющих потоков, структуры данных, описания логики процессов, спецификации элементов данных, сигналов и структурных объектов, а также исходные данные о системе, проекте, причастных лицах и организациях, разработчиках и т.п.

2) **Графические редакторы потоковых диаграмм и структурограмм данных.** Все действия над диаграммами при редактировании отображаются на экране в графическом виде. При вводе элементов диаграмм и их редактировании осуществляется контроль корректности вводимой информации и ее совместимости с остальными частями проекта. Введенная (измененная) информация запоминается в базе данных проекта автоматически и по запросу пользователя.

3) **Средства вывода экранных и печатных форм** для контроля и анализа проекта и его презентации. Предусмотрены следующие экранные и печатные формы: контекстная диаграмма, диаграмма потоков данных, диаграмма потоков управления, структурограмма данных, перечни

объектов словаря данных, отсортированных и выбранных различными способами, содержание элементов словаря данных, миниспецификация логики процесса, протоколы верификации проекта, отчеты проекта.

4) **Документатор**. Состав и содержание документов проекта системы регламентируются комплексами стандартов и руководящих документов. CASE.Аналитик поддерживает следующие стандарты и руководящие документы:

- *Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. М.: Госстандарт СССР, 1991г., - ГОСТы 34.ХХХ;*
- *Единая система программной документации - ГОСТы 19.ХХХ.*

Кроме того, оформление диаграмм при печати может быть выполнено в соответствии с требованиями ЕСКД: автоматически генерируется рамка и надписи.

5) **Верификатор**. Принципиальные решения по верификации проекта делает пользователь-аналитик. Эти решения аналитик принимает по результатам простых, но очень трудоемких процедур контроля и верификации, которые CASE.Аналитик выполняет автоматически и по запросу. CASE.Аналитик предоставляет следующие средства верификации:

- автоматический контроль выполнения формальных правил построения модели при вводе и редактировании;
- автоматическая поддержка согласованности при детализации подсистем, процессов и данных, т.е. при переходе с уровня на уровень;
- верификация (по запросу) согласованности модели;
- вывод на дисплей и печать разнообразных отчетов, которые могут быть использованы для верификации.

16.1. Особенности потоковых диаграмм информационно-логический модели

Формально **потоковая диаграмма** есть направленный граф, нагруженный по дугам и узлам. Потоковая диаграмма описывает асинхронный процесс преобразования информации от ее ввода в систему до выдачи потребителю. Внешние сущности - источники информации порождают информационные потоки. Информационные потоки переносят информацию к подсистемам или процессам. Подсистемы или процессы в свою очередь преобразуют информацию и порождают новые информационные потоки, которые переносят информацию к другим процессам или подсистемам, накопителям информации или внешним сущностям - потребителям информации. Использование ограниченного числа символов позволяет нам построить изображение системы, не связывая себя решениями о ее возможной реализации. На рис. 16.2 приведен пример диаграммы потоков данных, построенной с помощью пакета CASE.Аналитик.

Внешние сущности. Внешними сущностями системы обычно являются логические классы предметов или физических лиц, представляющие собой источник или приемник информации, например, *заказчики, персонал, поставщики, налогоплательщики, клиенты*. Это могут быть специфические источники или приемники, такие, как *Бухгалтерия, Информационно-поисковая система, Склад*. Если система, которую мы рассматриваем, принимает данные от другой системы или передает данные в другую систему, то эта другая система является элементом внешнего окружения. Внешняя сущность обозначается квадратом, как бы расположенным "над" диаграммой и бросающим на нее тень, для того, чтобы можно было выделить этот символ среди других обозначений на диаграмме (рис. 16.3).

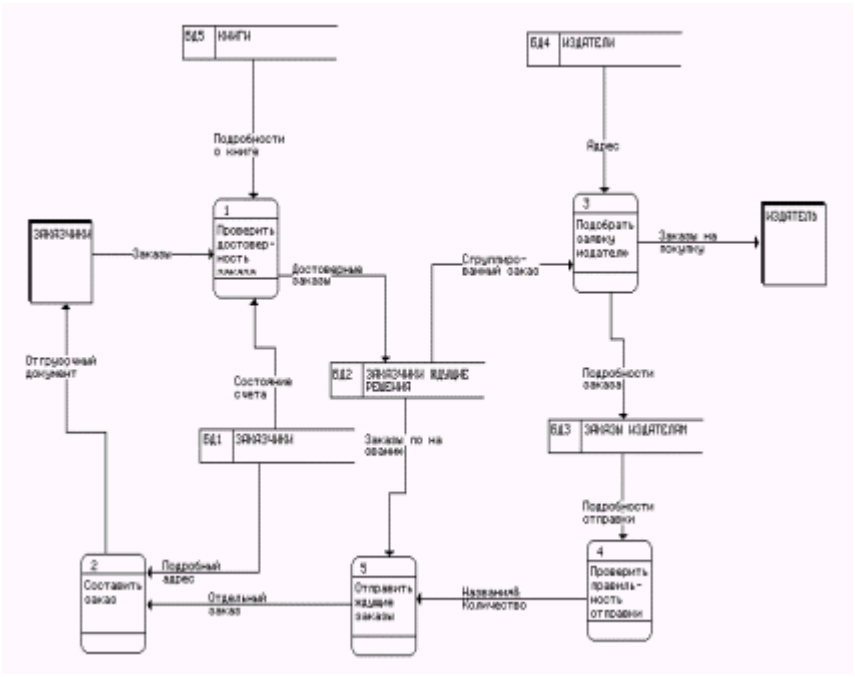


Рис. 16.2. Пример диаграммы потоков данных

Рис. 16.3. Изображение внешней сущности на диаграммах

Проектируя некоторое устройство или некоторую систему как внешнюю сущность, мы точно указываем, что она находится за пределами границ рассматриваемой системы. Когда анализ проделан и изучены требования пользователей, мы можем перенести некоторые внешние сущности внутрь диаграммы нашей системы или, наоборот, вынести какую-то часть функций нашей системы и рассматривать всю эту часть как внешнюю сущность с исходящим и входящим потоками данных.

Система/подсистема. При построении информационно-логической модели сложной (и, как правило, распределенной) системы ее структуризация на отдельные взаимодействующие подсистемы может быть уже заданной или с очевидностью следующей из внешних условий, налагаемых на систему. Структуризация системы на подсистемы показывается при помощи контекстных диаграмм. Подсистема на контекстной диаграмме изображается, как показано на рис. 16.4.

Рис. 16.4. Условное обозначение подсистемы

Процесс. Логически процесс является неким устройством, принимающим входные потоки и преобразующим их в соответствии со своей внутренней логикой в выходные потоки. В действительности процесс может быть реализован самыми разными способами: подразделение организации (например, отдел), выполняющие нужную обработку входных документов и выпуск соответствующих отчетов, программа ЭВМ, аппаратно реализованное логическое устройство и т.д. Процессы обозначаются прямоугольниками с закругленными углами, разделенными на три поля (см. рис. 16.5). Необходимо дать каждому процессу имя, отражающее его функцию и, по

возможности, привязать его к физической реализации. Для идентификации процессы автоматически нумеруются.

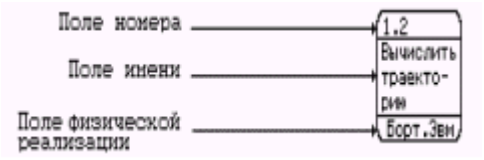


Рис. 16.5. Условное обозначение процесса

Управляющий процесс. Логически управляющий процесс есть некий командный пункт, который, реагируя на изменение внешних условий, передаваемых ему управляющим потоком (или потоком событий), выдает в соответствии со своей внутренней логикой команды, выполняемые другими процессами. Эти команды переносятся также управляющими потоками, а их исполнение процессами приводит к изменению состояния системы. Управляющий процесс может быть реализован, например, в виде командного пункта, на котором командир принимает данные (сигналы) об обстановке, и в соответствии с уставом, заданием и знаниями (внутренняя логика) выдает команды подчиненным; или в виде административного центра; или в виде процессора, управляемого многозадачной ОС. Управляющий процесс обозначается в виде прямоугольника с закругленными углами, изображенного пунктирными линиями.

Накопители данных. Логически накопители данных есть некие устройства для хранения информации, куда ее можно поместить и через некоторое время изъять. При этом на этапе анализа мы не уточняем способ помещения и извлечения данных в накопитель, нас не интересует, происходит ли извлечение данных в смысле чтения (копирования) или в смысле изъятия и другие подобные вопросы. Накопителем данных в реальности может быть микрофиша, ящик для хранения карточек, таблица в памяти, файл на ленте или диске. Накопитель данных обозначается двумя горизонтальными параллельными линиями, замкнутыми с одного края - рис. 16.6. Каждый накопитель данных идентифицируется для ссылки буквами "БД" и произвольным числом в квадрате с левой стороны, определяемым автоматически.

Рис. 16.6. Условное обозначение накопителя данных

Информационный канал. При детализации подсистем/процессов часто (за исключением простейших систем) возникает необходимость в детализации (структуризации) информационных потоков. Например, на диаграмме верхнего уровня может появиться поток *Годовой отчет*, который получает руководство организации. При детализации на следующих уровнях иерархии может выясниться, что годовой отчет готовят различные подразделения, и для этого порождаются такие потоки данных, как: *Финансовый отчет, Анализ сбыта, Номенклатура изделий.* Т.е. поток *Годовой отчет* есть результат слияния трех потоков. Такое слияние осуществляется через логический информационный канал. Логически информационный канал есть среда передачи информации. Информационный канал может реализоваться в виде, например, пневмопочты, курьерской службы, почты, магистрали или шины данных и т.д. Условное обозначение канала содержит идентифицирующую ссылку (буквы “ИК” и автоматически определяемый номер), поле имени и поле номера копии (рис. 16.7).

Рис. 16.7. Условное обозначение информационного канала

Информационный поток. Логически информационный поток есть некоторое соединение, по которому информация от источника передается приемнику. В реальности информационный поток может быть, например, информацией, передаваемой по кабелю между двумя устройствами, письмом, пересылаемым между респондентами, магнитной лентой или дискетой, переносимой между ЭВМ. Информационный поток может физически содержаться в телефонном звонке, при переходе от программы к программе через спутниковую информационную связь - при любом варианте прохождения данных от одного объекта или процесса к другому.

Рис. 16.8. Условные обозначения информационных потоков

16.2. Структурограммы данных

Детализация содержания информационных потоков и накопителей данных описывается при помощи **структурограмм описания данных**, в состав которых могут входить элементы данных различных типов, структуры данных, а также информация об альтернативном, условном или итеративном вхождении в структурограмму элемента или структуры данных. Пример структурограммы приведен на рис.16.9.

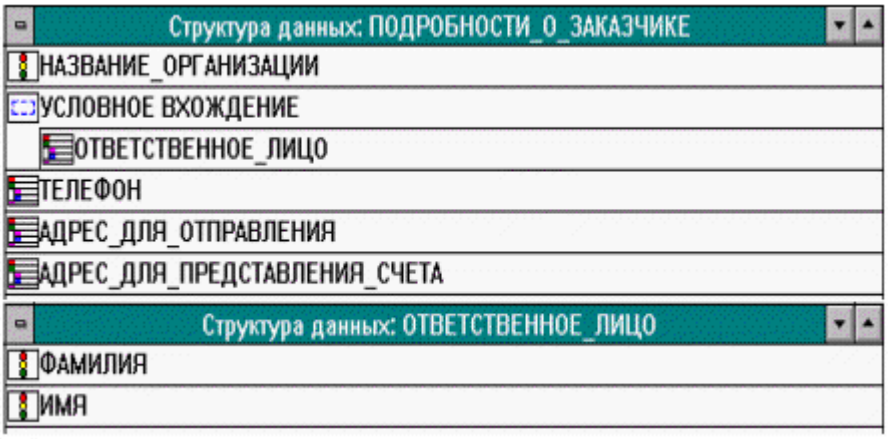


Рис. 16.9. Пример структурограммы

Структура данных *ПОДРОБНОСТИ О ЗАКАЗЧИКЕ* содержит элемент данных *НАЗВАНИЕ ОРГАНИЗАЦИИ* и вложенные структуры данных *ОТВЕТСТВЕННОЕ ЛИЦО* (условно входит), *ТЕЛЕФОН, АДРЕС ДЛЯ ОТПРАВЛЕНИЯ* и *АДРЕС ДЛЯ ПРЕДСТАВЛЕНИЯ СЧЕТА*. В нижней части рисунка приведен состав элементов одной из входящих в структурограмму структур данных (*ОТВЕТСТВЕННОЕ ЛИЦО*).

16.3. Описание структурных элементов

Несмотря на то, что в своей основе информационно-логическая модель является графической, структурные элементы диаграмм и структурограмм имеют дополнительно текстовые описания, хранящиеся в базе данных.

Со всеми структурными элементами информационно-логической модели - процессами, внешними сущностями и т.д. - связаны аннотации, куда аналитик может записать произвольный текст, каким-то образом характеризующий или поясняющий введенный элемент, а также номер ссылки, где можно указать источник возникновения элемента. Для каждого элемента модели можно указать синонимы, возникающие из-за того, что пользователи в разных отделах одному и тому же понятию дают разные имена, например, то, что служащие на складе называют *НОМЕРОМ ТРЕБОВАНИЯ*, занимающиеся закупочной деятельностью могут называть *НОМЕРОМ ЗАКАЗА*.

Ниже приводится состав описательной информации для каждого типа элементов данных, которая вносится аналитиком в соответствующие формы. Отметим, что едва ли не большая часть информации о том или ином конкретном структурном элементе генерируется CASE.Аналитиком автоматически.

Непрерывные данные - данные, которые на практике могут принимать любое значение в пределах диапазона, например, сумма в рублях может быть от нуля до 999999,99 с точностью до копейки или температура - от 0^o до 300^o. Для непрерывных данных указывается единица измерения, диапазон значений, типичное значение, точность и кодировка. **Дискретные данные** - данные, которые могут принимать только определенные значения, например, *номер отдела*, который может быть равен 36,08,29 или 71. Для дискретных данных заполняется таблица значений в соответствующем формате.

Аналоговые сигналы - это, как правило, измерительные данные от различного рода датчиков. Для них указывается единица измерения, диапазон, уровень сигнала и тип датчика. **Дискретные сигналы** - это, как правило, сигналы от датчиков положения (состояния) тех или иных устройств. Для них указывается уровень сигнала и тип датчика.

16.4. Сводка основных функций пакета

1) **Построение и редактирование потоковых диаграмм**. Соответствующие средства позволяют передвигать объекты диаграммы с автоматическим перерисовыванием связанных с этими объектами потоков, копировать накопители данных, информационные каналы и внешние сущности, удалять объекты с автоматической поддержкой целостности модели, вводить и редактировать спецификации объектов, погружаться вглубь любого объекта диаграммы с автоматическим сохранением произведенных на верхнем уровне модификаций, а также просматривать всю диаграмму в целом в уменьшенном виде и выполнять все вышеперечисленные операции над ее объектами.

2) **Навигация по диаграммам**. Средства позволяют осуществлять навигацию по горизонтали с использованием специального окна навигации, навигацию по вертикали (вглубь, наверх), а также выбор и загрузку любой диаграммы с использованием дерева диаграмм проекта.

3) **Редактирование структурограмм**. Имеются возможности передвижения элементов структурограмм, их удаления с автоматической поддержкой целостности, ввода и редактирования спецификаций данных, а также погружения вглубь структуры.

4) **Навигация по данным**. Включает передвижение по структурограмме наверх/вглубь, а также выбор и загрузку любой структурограммы по дереву структур данных.

5) **Описание логики процессов**. Позволяет вводить и редактировать миниспецификации процессов с использованием структурированного естественного языка.

6) **Навигация по базе данных проекта**. Позволяет осуществлять доступ к спецификации любого объекта модели, используя списки и перечни объектов, поиск по имени, а также доступ из диаграмм и структурограмм.

7) **Верификация проекта** на полноту исходных данных, полноту диаграмм, полноту данных, согласованность накопителей и информационных каналов, а также анализ нагрузки информационных каналов и анализ объема накопителей данных.

8) **Печать диаграмм** - имеется возможность использования следующих режимов: качественная печать, быстрая печать, пропорциональная печать, печать для презентации.

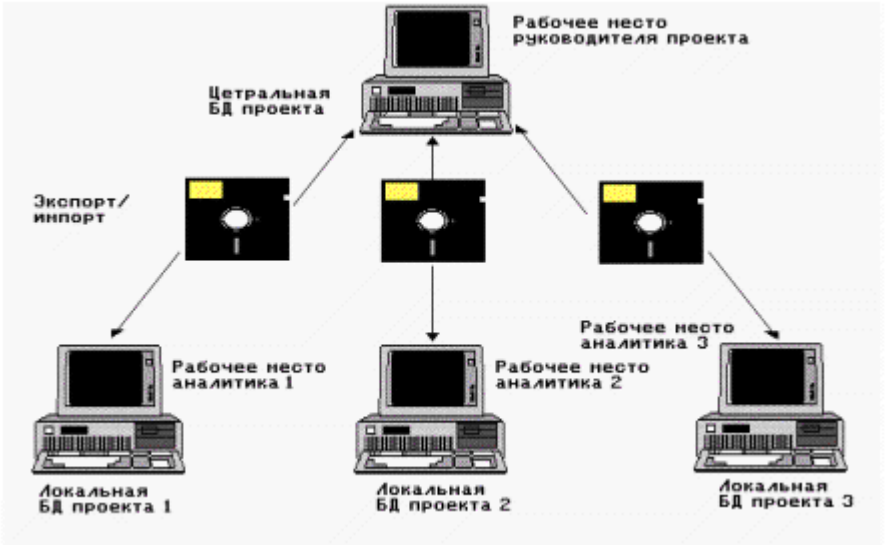


Рис. 16.10. Экспорт/импорт частей проекта

9) **Генерация отчетов и документов**. Средства позволяют генерировать 12 отчетов по проекту в соответствии с вышеперечисленными стандартами, а также отчеты по спецификациям объектов (13 типов объектов), перечням объектов (11 отчетов) и верификации (6 отчетов).

10) **Экспорт/Импорт**. Благодаря этой функции возможно взаимодействие аналитиков, работающих на автономных рабочих местах. Руководитель проекта ведет центральную базу данных проекта и экспортирует для проработки другим аналитикам части (поддеревья) иерархической информационно-логической модели системы (рис. 16.10).

При этом, для обеспечения целостности проекта, руководителю закрывается доступ для редактирования как экспортируемого дерева, так и его контекста. Аналитик может редактировать только детализацию корневого процесса (подсистемы). После проработки поддеревьев, они импортируются в центральную базу данных проекта руководителя проекта.

1) **Связь с пакетом ERWin.** CASE.Аналитик позволяет строить только концептуальные модели данных с использованием структурограмм. При необходимости построения логической модели данных (в виде диаграмм “сущность-связь”) имеется возможность взаимодействия с одним из наиболее популярных пакетов построения информационных моделей - пакетом ERWin фирмы Logic Works. Для этой цели разработан отдельный программный продукт Catherine, предназначенный для обмена данными между CASE.Аналитик 1.1+ и >ERWin/ERX 2.5. Эта программа работает с базой данных CASE.Аналитика. Данные для экспорта могут формироваться как в автоматическом (все структурограммы), так и в интерактивном (отдельные фрагменты концептуальной модели) режиме. По окончании выбора экспортируемых данных они записываются в словарь данных ERWin, при этом автоматически контролируется непротиворечивость построенной ER-модели.

Из проекта, выполненного в CASE.Аналитике экспортируются:

- структурограммы данных вместе с их внутренней структурой
- накопители данных и их структура
- аннотации для накопителей и данных.

При переносе объектов используются следующие правила:

- детализированная структура (или накопитель данных) определяется как сущность;
- данные типов “аналоговое/дискретное данное/сигнал”, используемые при детализации структурограммы (накопителя), определяются как атрибуты соответствующей сущности;
- недетализированная структура второго и более уровня определяется как атрибут соответствующей сущности.

Таким образом, в ERWin могут быть экспортированы сущности и их атрибуты. Предполагается, что дальнейшее построение логической модели данных будет выполняться непосредственно в этом пакете.

В заключение отметим, что для функционирования CASE.Аналитика требуется минимальный объем аппаратных ресурсов:

- процессор - Intel 386 и выше
- память - 4 Mb RAM, 5 Mb HDD
- операционная система - MS Windows 3.* или Windows 95.

Глава 17
ОБЗОР РОССИЙСКОГО РЫНКА CASE-СРЕДСТВ

17.1. Краткое описание основных возможностей пакетов Продукты фирмы CA (BPWin, ERWin)

Пакет BPWin основан на методологии IDEF0 и предназначен для функционального моделирования и анализа деятельности предприятия. Модель в BPWin представляет собой совокупность SADT-диаграмм, каждая из которых описывает отдельный процесс в виде разбиения его на шаги и подпроцессы. С помощью соединяющих дуг описываются объекты, данные и ресурсы, необходимые для выполнения функций. Имеется возможность для любого процесса указать стоимость, время и частоту его выполнения. Эти характеристики в дальнейшем могут быть просуммированы с целью вычисления общей стоимости затрат - таким образом выявляются узкие места технологических цепочек, определяются затратные центры. BPWin может импортировать фрагменты информационной модели из описываемого ниже средства проектирования баз данных ERWin (при этом сущности и атрибуты информационной модели ставятся в соответствие дугам SADT-диаграммы). Генерация отчетов по модели может осуществляться в формате MS Word и MS Excel. Требования к ресурсам:

- процессор - Intel 386 и выше
- память - 8 Mb RAM
- операционная система - MS Windows 3.1 или Windows 95.

Семейство продуктов ERWin предназначено для моделирования и создания баз данных произвольной сложности на основе диаграмм “сущность-связь”. В настоящее время ERWin является наиболее популярным пакетом моделирования данных благодаря поддержке широкого спектра СУБД самых различных классов: SQL-серверов (Oracle, Informix, Sybase SQL Server, MS SQL Server, Progress, DB2, SQLBase, Ingress, Rdb и др.) и “настольных” СУБД типа XBase (Clipper, dBASE, FoxPro, MS Access, Paradox и др.).

Информационная модель представляется в виде диаграмм “сущность-связь”, отражающих основные объекты предметной области и связи между ними. Дополнительно определяются атрибуты сущностей, характеристики связей, индексы и бизнес-правила, описывающие ограничения и закономерности предметной области. После создания ER-диаграммы пакет автоматически генерирует SQL-код для создания таблиц, индексов и других объектов базы данных. По заданным бизнес-правилам формируются стандартные триггеры БД для поддержки целостности данных, для сложных бизнес-правил можно создавать собственные триггеры, используя библиотеку шаблонов.

Пакет может осуществлять реинжиниринг существующих БД: по SQL-текстам автоматически генерируются ER-диаграммы. Таким образом, пакет полностью поддерживает технологию FRE (forward and reverse engineering), последовательность этапов которой приведена ниже:

- импорт с сервера существующей БД
- автоматическая генерация модели БД
- модификация модели

- автоматическая генерация новой схемы и построение физической БД на том же самом или любом другом сервере.

Для разработки клиентской части приложения имеются специальные версии пакета, обеспечивающие интеграцию с такими инструментами как SQLWindows, PowerBuilder, Visual Basic, Delphi. Предлагаются и усеченные версии продукта:

- ERWin/SQL, обеспечивающая лишь прямое проектирование для любых СУБД
- ERWin/Desktop, поддерживающая технологию FRE только для“настольных” СУБД.

Требования к ресурсам, необходимым для функционирования пакета, совпадают с приведенными выше соответствующими требованиями для пакета BPWin.

Для коллективной разработки модели БД предназначен специальный продукт ModelMart, позволяющий контролировать версии модели, гибко распределять права доступа между членами группы, строить библиотеки моделей, осуществлять объединение моделей и т.п. Продукт построен в архитектуре “клиент-сервер”, репозитарий использует одну из трех СУБД - Oracle, Sybase, MS SQL Server и требует 32 Mb RAM и 50 Mb HDD. ERWin-клиент для своего функционирования требует процессор Intel 486 или Pentium, 16 Mb RAM и 10 Mb HDD.

Пакет CASE/4/0 (microTOOL GmbH)

Пакет CASE/4/0 включает в себя структурные средства системного анализа, проектирования и программирования и обеспечивает поддержку всего жизненного цикла разработки вплоть до сопровождения, основанную на сетевом репозитарии, контролирующем целостность проекта и поддерживающем согласованную работу всех участников проекта (системных аналитиков, проектировщиков, программистов).

Анализ базируется на классической структурной методологии Уорда-Меллора, являющейся расширением подхода Йодана/Де Марко с целью его ориентации на разработку систем реального времени, проектирование основано на подходе Джексона. Для целей анализа и проектирования используются следующие типы диаграмм:

- древовидные диаграммы функциональной декомпозиции
- диаграммы потоков данных
- диаграммы переходов состояний
- диаграммы “сущность-связь”
- структурные карты Джексона.

Помимо графических редакторов перечисленных диаграмм и репозитария, основными компонентами пакета являются:

- дизайнер диалогов для моделирования интерфейса пользователя
- средства разработки на Cobol, C/C++, Visual Basic
- синтаксически-ориентированные редакторы кодов
- средства генерации документов.

Пакет состоит из двух компонентов: клиентской части, устанавливаемой на рабочих местах разработчиков (MS Windows 3.11, Windows-NT, Windows 95), и интегрированного сетевого репозитария, устанавливаемого на сервере (Novell, MS Windows, HP Unix, Sinix, IBM OS/2, IBM AIX).

Пакет Design/IDEF (Meta Software)

CASE-пакет Design/IDEF автоматизирует все этапы проектирования сложных систем различного назначения: формулировку требований и целей проектирования, разработку спецификаций, определение компонентов и взаимодействий между ними, документирование проекта, проверку

его полноты и непротиворечивости. Наиболее успешно пакет применяется для описания и анализа деятельности предприятия; он позволяет оценить такую структуру, как единый организм, сочетающий управленческие, производственные и информационные процессы. В основе пакета лежит методология структурного проектирования и анализа сложных систем IDEF0/SADT. Design/IDEF строит иерархические модели сложных систем посредством декомпозиции ее компонентов, поддерживает коллективную разработку IDEF-модели, позволяя в любой момент объединять различные подмодели в единую модель системы, создает словарь данных для хранения всей информации о функциях и структурах данных проекта; формирует 5 типов отчетов, поддерживающих процесс разработки и анализа моделей.

Кроме IDEF0, пакетом поддерживаются методологии моделирования данных IDEF1, IDEF1X (основанные на диаграммах "сущность-связь"), а также методология моделирования динамики систем IDEF/CPN, основанная на "цветных" или "раскрашенных" сетях Петри. Последнюю методологию реализует система динамического моделирования Design/CPN. Фактически Design/IDEF и Design/CPN являются комонентами интегрированной методологии разработки систем: диаграммы, постоенные Design/IDEF, автоматически импортируются в Disign/CPN и дорабатываются вручную для динамического моделирования и фактической оценки. Design/CPN позволяет “отлаживать” модель с целью оценки ее динамики: такая оценка позволяет эффективно распределять ресурсы и оптимизировать систему, а также верифицировать ее поведение в различных режимах.

Пакеты Design/IDEF и Design/CPN базируются на открытой архитектуре Design/OA (инструментальной среде для создания CASE-средств), позволяющей дополнять их модулями, ориентированными на конкретную задачу пользователя, включая генерацию кода на произвольном целевом языке.

Design/IDEF реализован на платформах MS Windows, Macintosh Plus и выше, Sun Solaris (X Window System), HP9000 модели 700 и 800 (X Window System). Для функционирования Design/CPN требуется: Sun (SPARC), HP9000 модели 700 и 800, X Window System (X11R5), 24 Mb RAM, 32 Mb HDD.

Design/IDEF также интегрирован с пакетом динамического анализа сложных систем WorkFlow Analyzer и пакетом функционально- стоимостного анализа EasyABC.

Пакет Designer/2000 (Oracle)

Designer/2000 - инструмент, работающий в среде MS Windows и развивающий подход фирмы Oracle к созданию и сопровождению сложных информационных систем. В основе подхода лежит собственная методология фирмы CASE*Method, базирующаяся на структурном анализе и проектировании системы, четком разбиении ее жизненного цикла на этапы, автоматизации перехода между этапами.

Центральной частью пакета является репозитарий, содержащий спецификации проекта на всех его этапах и обеспечивающий согласованную работу всех его участников. Для доступа к репозитарию и управления им используется специальное средство (навигатор по объектам репозитария), позволяющее просматривать и модифицировать объекты, хранящиеся в репозитарии, а также осуществлять административные функции: удаление, управление доступом, экспорт/импорт и т.п.

Первым этапом методологии является моделирование и анализ процессов, т.е. построение моделей деятельности предприятия, выявление их недостатков и возможных источников усовершенствования. Поддерживающие средства позволяют строить наглядные представления процессов и их взаимосвязей, а также анализировать их с использованием средств мультимедиа.

Второй этап (системное моделирование) предполагает разработку детальных концептуальных моделей предметной области и фактически является этапом выявления, анализа и формализации требований к будущей системе. Для описания используются диаграммы “сущность-связь”, диаграммы иерархии функций и диаграммы потоков данных.

Третий этап (системное проектирование) на основании концептуальных моделей вырабатывает технические спецификации будущей системы, при этом первоначальный вариант спецификаций может быть получен автоматически. На этом этапе применяются диаграммы схем БД (расширения ER-диаграмм), диаграммы взаимодействий модулей (аналог структурных карт Джексона) и схемы модулей, описывающие структуру модулей с позиций используемых в них данных.

Наконец, на четвертом этапе (генерация приложений) создаются программы, отвечающие требованиям проектных спецификаций. Так генератор серверной части по спецификации БД автоматически генерирует SQL-тексты, а генераторы приложений строят экранные формы и отчеты. При необходимости сгенерированные тексты могут быть доработаны с помощью дополнительного пакета Developer/2000.

Имеется облегченная версия пакета (Database Designer), основанная на диаграммах “сущность-связь” и предназначенная для создания информационных моделей.

Пакет EasyCASE (Evergreen CASE Tools)

Пакет предназначен для моделирования как информационных систем, так и систем реального времени, им поддерживаются традиционные модели:

- диаграммы потоков данных
- диаграммы “сущность-связь”
- диаграммы переходов состояний
- структурные карты в нотации Константайна.

Отличительной чертой пакета является возможность настройки на различные нотации диаграмм потоков данных, используемые в методологиях Гейна-Сарсона, Йодана/ДеМарко, Уорда-Меллора, SSADM и др. Другой особенностью пакета является его взаимодействие с СУБД не напрямую, а через ODBC-интерфейс с использованием внешних файлов для описания специфики конкретной СУБД.

Пакет обеспечивает генерацию схем БД для Oracle, Informix, Sybase, Progress, SQLBase, Ingress, а также MS Access, Paradox и др. и поддерживает технологию FRE. Имеется интерфейс с системой Delphi.

Пакет VantageTeam Builer (CAYENNE)

Пакет предназначен для проектирования как информационных систем, так и систем реального времени. Им реализуется методология Йодана и поддерживаются следующие типы диаграмм:

- диаграммы потоков данных в нотации Йодана
- диаграммы переходов состояний
- диаграммы “сущность-связь” в нотации Чена
- структурные карты Константайна.

Важный методологический момент заключается в возможности как нисходящего, так и восходящего построения иерархии диаграмм. Рекомендуется использовать пакет на рабочих местах аналитика, системного архитектора, проектировщика и программиста - такая совокупность рабочих мест содержит полный набор инструментов поддержки жизненного цикла разрабатываемой системы. При переходе с одного рабочего места на другое осуществляется верификация модели на полноту и состоятельность.

В отличие от других пакетов, генерирующих лишь схему базы данных, кодогенерация в VantageTeam Builer включает:

- генерацию SQL-текстов на основе диаграмм “сущность-связь”

- генерацию экранных форм на основе диаграмм последовательности и содержания экранных форм
- генерацию текстов модулей на 4GL на основе структурных карт и предопределенных модулей.

Внимания заслуживают и возможности настройки пакета, включающие:

- настройку графических редакторов для расширения нотации
- создание собственного интерфейса (модификация окон, расширение меню)
- введение дополнительных атрибутов любого объекта модели
- создание собственных шаблонов кодогенерации.

Пакет работает на всех основных UNIX-платформах и OpenVMS. В качестве рабочего места пользователя помимо UNIX-рабочей станции может использоваться X-терминал или ПК с программой X-эмуляции. Для работы пакета необходимы средства разработки приложений (Informix 4GL, Ingress, Uniface, C) и СУБД для репозитария (Informix, Ingress, Oracle, Sybase), все вместе требует 100-200Mb HDD в зависимости от платформы и СУБД и не менее 24 Mb RAM.

Пакет ProKit*WORKBENCH
(McDonnell Douglas Information Systems)

Средства автоматизации проектирования информационных систем фирмы McDonnell Douglas Information Systems базируются на методологии проектирования информационных систем STRADIS. Она определяет порядок создания информационной системы, требования к необходимым ресурсам и распределению работ между разработчиками на этапах ЖЦ системы, требования к составу и содержанию разрабатываемых на промежуточных этапах проектных материалов, методику выполнения проектных работ, программирования, проверки и управления разработкой. В STRADIS определены следующие стадии ЖЦ информационной системы: стратегическое планирование, анализ, проектирование, разработка, сопровождение. Первые три стадии поддерживаются CASE-пакетом ProKit*WORKBENCH, последующие стадии - технологической средой программирования на языке четвертого поколения PRO-IV.

ProKit*WORKBENCH обеспечивает:

- функциональное моделирование с использованием диаграмм потоков данных в нотации Гейна-Сарсона;
- информационное моделирование с использованием диаграмм “сущность-связь”, при этом сущности модели синхронизируются с накопителями данных соответствующих DFD;
- функциональное прототипирование будущей системы на основании средств описания экранов и выходных документов, режимов работы и сценариев диалога;
- проектирование модулей, основанное на технике структурных карт Константайна;
- интегрированное хранение всей проектной информации в репозитарии;
- экспорт/импорт фрагментов проекта;
- формирование отчетов по проекту;
- передачу результатов анализа и проектирования в среду PRO-IV для последующей разработки.

Несомненным достоинством пакета является возможность одновременной поддержки различных версий проекта (до 8 версий). Пакет функционирует в MS Windows 3.1, Windows NT, Windows 95 (существует версия для MS DOS), требует 16 Mb RAM и 30 Mb HDD.

Пакет S-Designor (Sybase/Powersoft)

Пакет S-Designor предназначен для проектирования баз данных и по своим функциональным возможностям близок к пакету ERWin. Основное отличие заключается в том, что ERWin взаимодействует с поддерживаемыми СУБД напрямую, а в S-Designor работа с СУБД ведется

через ODBC-интерфейс с использованием внешних файлов для описания специфики конкретной СУБД.

Пакетом поддерживаются традиционные диаграммы “сущность-связь” и реализуется стандартная двухуровневая методология информационного моделирования, включающая поэтапное создание концептуальной (логической) и физической моделей данных. На основе физической модели генерируются SQL-тексты для широкого набора СУБД, включая Oracle, Informix, Sybase SQL Server, MS SQL Server, Progress, DB2, SQLBase, Rdb, MS Access, Paradox. В качестве средств разработки клиентской части поддерживаются PowerBuilder, TeamWindows, Progress, Uniface. Требования к ресурсам:

- процессор - Intel 386 и выше
- память - 8 Mb RAM, 7 Mb HDD
- операционная система - MS Windows 3.1, Windows NT, Windows 95.

Пакет SILVERRUN (Computer Systems Advisers)

В основе пакета лежит собственная методология DATARUN, предназначенная для создания информационных систем и регламентирующая все этапы жизненного цикла от стадии первоначальной экономической оценки затрат на проект до получения реального приложения. Основными этапами методологии являются следующие:

- построение бизнес-модели предметной области
- построение архитектуры информационной системы
- проектирование
- создание подсистем
- интеграция подсистем.

Пакет обеспечивает гибкую настройку на различные нотации диаграммных техник (Йодан, Гейн-Сарсон, Уорд-Меллор и др.), более того, имеется возможность вводить собственные нотации пользователя.

В состав входят три основные подсистемы: модуль построения диаграмм потоков данных BPM, модуль построения концептуальных информационных моделей (диаграмм “сущность-связь”) ERX и модуль построения реляционных моделей (также диаграмм “сущность-связь”) RDM. Каждый из перечисленных модулей является самостоятельным продуктом и поставляется отдельно. Для интеграции модулей в единое целое служит менеджер репозитария WRM.

Полезной особенностью модуля ERX является встроенная экспертная система, помогающая построить концептуальную модель в виде, допускающем реализацию в реляционной СУБД (используя ответы на содержательные вопросы о взаимосвязи данных).

Генерация схемы базы данных осуществляется в модуле RDM (для 16 СУБД), однако для полного использования специфики каждой конкретной СУБД применяются отдельно поставляемые мосты, поддерживающие технологию FRE. Пакет имеет мосты к следующим СУБД: Oracle, Informix, Sybase SQL Server, MS SQL Server, Progress, DB2, SQLBase, Ingress. Для обмена данными с языками разработки приложений также используются мосты: PowerBuilder, Progress, SQLWindows, Uniface.

Для функционирования пакета требуется:

- память - 16 Mb RAM, 20 Mb HDD
- операционная система - MS Windows 3.1, Windows NT, Windows 95.

Также пакет работает на платформах OS/2, Macintosh, Sun Solaris.

Пакет Visible Analyst Workbench (Visible Systems)

Visible Analyst Workbench представляет собой сетевое многопользовательское средство проектирования информационных систем, базирующееся на репозитарии, хранимом на сервере SQLBase, Oracle или Informix. Пакет основан на методологии Мартина и поддерживает следующие диаграммные техники:

- диаграммы функциональной декомпозиции
- диаграммы потоков данных в нотациях Йодана и Гейна-Сарсона
- диаграммы “сущность-связь”
- структурные карты в нотации Константайна.

Пакет обеспечивает генерацию схем БД для вышеперечисленных СУБД и поддерживает технологию FRE. Имеется возможность экспорта проектов в системы SQLWindows, PowerBuilder и Uniface.

К достоинствам пакета может быть отнесено наличие развитых средств верификации проекта, и прежде всего возможностей вертикального и горизонтального балансирования диаграмм. Так функциональная и информационная модели сильно коррелированы, что позволяет избавиться от лишних объектов моделей.

17.2. Номенклатура пакетов и виды проектной деятельности

В настоящее время CASE-системы прочно вошли в практику программной индустрии. При этом они используются не только (и не столько) как комплексные технологические конвейеры для производства программных систем, но и как мощный инструмент решения исследовательских и проектных задач, связанных с начальными этапами разработки, таких, как анализ предметной области, разработка проектных спецификаций, выпуск проектной документации, планирование и контроль разработок, моделирование деловых приложений с целью решения задач оперативного и стратегического планирования и управления ресурсами и т.п. К настоящему моменту наиболее интенсивное развитие получили два главных направления применения CASE-средств:

1. **BPR (business process reengineering)** - перепроектирование бизнес-процессов. Под перепроектированием понимается “фундаментальное переосмысление и радикальное перепланирование критических бизнес-процессов, имеющее целью резко улучшить их выполнение по отношению к затратам, качеству обслуживания и скорости”. При этом бизнес-процесс представляет собой некоторую деятельность, получающую входные данные одного или нескольких типов и выдающую результат, имеющий ценность для клиента. Например, процесс выполнения заказа на входе получает заказ и выдает в качестве результата заказанные товары. Другими словами, доставка заказанных товаров клиенту и есть та ценность, которую создает процесс.
2. **системный анализ и проектирование**, включающий функциональное, информационное и событийное моделирование как вновь создаваемой, так и существующей системы.

Необходимо отметить, что такое разбиение является весьма условным, поскольку при анализе предприятия и разработке проекта его автоматизации используются элементы BPR (более того, теоретически BPR должно быть первым этапом разработки), в то же время необходимым этапом перепроектирования является по крайней мере создание функциональной модели бизнес-процесса.

В таблице 17.1 приведен перечень доступных на российским рынке CASE-средств и поддерживаемые ими виды проектной деятельности.

Таблица 17.1

Название	Фирма	BPR	Функции	Данные	События
BPWin	Logic Works	+	+	-	-

CASE.Аналитик	Эйтэкс	-	+	+	+
CASE/4/0	MicroTOOL	-	+	+	+
Database Designer	Oracle	-	-	+	-
Design/IDEF	Meta Software	+	+	+	-
Designer/2000	Oracle	+	+	+	-
EasyCASE	Evergreen CASE Tools	-	+	+	+
ERWin	Logic Works	-	-	+	-
I-CASE Yourdon	CAYENNE	-	+	+	+
Prokit*WORKBENCH	MDIS	-	+	+	-
S-Designor	Sybase/Powersoft	-	+	+	-
SILVERRUN	CSA	-	+	+	+
Visible Analyst Workbench	Visible Systems	-	+	+	+

Средства BPR

Для моделирования бизнес-процессов обычно используется методология SADT (точнее ее подмножество IDEF0), поддерживаемая пакетами BPWin и Design/IDEF. Однако статическая SADT-модель не обеспечивает полного решения задач перепроектирования, необходимо иметь возможность исследования динамических характеристик бизнес-процессов. Одним из решений является использование системы динамического моделирования Design/CPN, основанной на цветных (раскрашенных) сетях Петри. Фактически Design/IDEF и Design/CPN являются компонентами интегрированной методологии перепроектирования: статические SADT-диаграммы автоматически преобразуются в прообраз динамической модели, которая дорабатывается вручную и затем исполняется в различных режимах с целью получения соответствующих оценок.

Другой возможный подход реализуется пакетом Designer/2000: моделирование бизнес-процессов является первым этапом разработки системы, а соответствующая модель является основой для разработки концептуальных моделей и проектирования системы. Нотация для моделирования бизнес-процессов включает следующие элементы: базовый процесс, шаг процесса, хранилище, поток, событие и организационная единица. Для каждого элемента можно задать разнообразные количественные параметры (временные затраты, ресурсы и т.п.), а затем с помощью специальной процедуры анимации проследить поведение модели в динамике с учетом введенных параметров. Использование средств мультимедиа, включая визуализацию, видеоизображение, звуковое сопровождение и т.п. позволяет существенно повысить выразительность построенной бизнес-модели.

Следует отметить, что не существует принципиальных ограничений в использовании в качестве средства построения статических моделей бизнес-процессов и традиционных DFD - диаграмм потоков данных. Более того, в настоящий момент за рубежом доступен ряд продуктов динамического моделирования (INCOME Mobile, CPN-AMI и др.), базирующихся на сетях Петри различного вида и интегрируемых с DFD-моделью, которые позволяют успешно решать задачи перепроектирования.

Средства функционального моделирования

Для решения задачи функционального моделирования на базе структурного анализа традиционно применяются два типа моделей: SADT-диаграммы и диаграммы потоков данных. В случае наличия в моделируемой системе программной/программируемой части (т.е. практически всегда) предпочтение, как правило, отдается DFD по следующим соображениям:

1. DFD с самого начала создавались как средство проектирования программных систем (тогда как SADT - как средство проектирования систем вообще) и имеют более богатый набор элементов, адекватно отражающих их специфику (например, хранилища данных являются прообразами файлов или баз данных).
2. Наличие миниспецификаций DFD-процессов нижнего уровня позволяет преодолеть логическую незавершенность SADT (а именно, обрыв модели на некотором достаточно низком уровне, когда дальнейшая ее детализация становится бессмысленной) и построить полную функциональную спецификацию разрабатываемой системы.
3. Существуют (и поддерживаются рядом CASE-пакетов) алгоритмы автоматического преобразования иерархии DFD в структурные карты, демонстрирующие межмодульные и внутримодульные связи, а также иерархию модулей, что в совокупности с миниспецификациями является завершенным заданием для программиста.

Наконец, в части автоматизированной поддержки моделей приблизительно 85-90% существующих CASE-пакетов поддерживают DFD и лишь 2-3% - SADT.

Средства событийного моделирования

Традиционный подход к моделированию аспектов поведения системы основывается на расширении диаграмм потоков данных за счет введения управляющих потоков (сигналов) и управляющих процессов, фактически являющихся интерфейсом между DFD и спецификациями управления, собственно моделирующими поведение. Наиболее часто спецификации управления формализуются с помощью диаграмм переходов состояний STD, позволяющих задавать состояния различных объектов системы (например, лицевой счет может иметь состояния ОТКРЫТ, ЗАКРЫТ, ЗАБЛОКИРОВАН и т.п.), условия переходов из одного состояния в другое (как внешние по отношению к системе, так и внутренние, возникающие в самой системе), а также совершаемые при переходах действия.

В таблице 17.2 приведен перечень пакетов, поддерживающих DFD, и основные составляющие функциональных моделей.

Таблица 17.2

Название	CASE.Аналитик	Мини-спец	Поведение	Структ. карты
CASE.Аналитик	Гейн-Сарсон	структ. язык	упр.потоки и процессы	-
Название	Нотация DFD	Мини-спец	Поведение	Структ. карты
CASE.Аналитик	Гейн-Сарсон	структ. язык	упр.потоки и процессы	-
CASE/4/0	Гейн-Сарсон,Йодан	структ. язык	Уорд-Меллор(с STD)	Константайн
I-CASE Yourdon	Йодан	3GL	STD	Константайн
Prokit*WORKBENCH	Гейн-Сарсон	-	-	Константайн
S-Designor	Гейн-Сарсон, Йодан	-	-	-
SILVERRUN	произвольная	-	упр.потоки и процессы	-
Visible Analyst Workbench	Гейн-Сарсон,Йодан	-	-	Константайн

Средства информационного моделирования

Для целей информационного моделирования на сегодняшний день не существует альтернативы диаграммам "сущность-связь" ERD. Практически все из приведенных в таблице 17.1 пакетов поддерживают ту или иную нотацию ERD. При этом разработка информационной модели в рассматриваемых средах включает в себя не только проектирование логической модели, но и преобразование ее в физическую модель с последующей генерацией схемы БД с учетом специфики конкретной СУБД.

ЧАСТЬ 5

РЕОРГАНИЗАЦИЯ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЙ

В пятой части книги кратко рассматриваются существующие подходы к реорганизации деятельности предприятий, неформально описывающие принципы, правила, методы, этапы и т.д. перехода от модели “как есть” к модели ”как должно быть”.

Необходимо отметить, что реорганизация деятельности часто путается с другими процессами, происходящими на предприятии. На самом деле реорганизация существенно отличается даже от тех процессов, с которыми она имеет некоторые общие исходные понятия.

Во-первых, реорганизация - это не то же самое, что автоматизация. Автоматизировать существующие процессы - “это все равно, что асфальтировать дорожки, по которым коровы ходят на пастбище”. Автоматизация - это просто возможность более эффективно делать неправильные вещи.

Также не следует путать реорганизацию с так называемым информационным перепроектированием, означающим перестройку устаревших информационных систем с использованием более современной технологии. В результате информационного перепроектирования часто возникают только сложные компьютеризованные системы, автоматизирующие устаревшие процессы.

Реорганизация - это не сокращение размеров предприятия, означающее снижение выпуска продукции для того, чтобы удовлетворить сегодняшние пониженные требования рынка. Такое сокращение позволяет достигнуть меньшей производительности с меньшими затратами, тогда как реорганизация служит, наоборот, достижению большей производительности с меньшими затратами.

Также реорганизация не означает изменение оргштатной структуры предприятия, хотя этот процесс действительно может отразиться на этой структуре. Основные проблемы, с которыми сталкиваются предприятия, являются результатом неверной структуры бизнес-процесса, а не предприятия. Пытаться совместить новую оргштатную структуру со старым процессом - это все равно, что “переливать прокисшее вино в новые бутылки”.

Предприятия, которые начинают борьбу с бюрократией, также находятся на неверном пути. Если Вас не устраивает бюрократия, попробуйте обойтись без нее, но в результате Вы получите хаос. Бюрократия - это тот цемент, который скрепляет все компоненты традиционной корпорации. Глубинная проблема, которую решает бюрократия - это проблема фрагментированных процессов. Для того, чтобы избавиться от бюрократии и перестроить структуру предприятия, необходимо перепроектировать процессы так, чтобы они не были фрагментированы. После этого предприятие может обойтись и без бюрократического аппарата.

Спектр существующих подходов к реорганизации предприятия варьируется от мягких постепенных методов улучшения его деятельности, основанных в значительной степени на соображениях здравого смысла, до жестких, регламентирующих его коренную ломку и декларирующих принцип “отбрось все старое и начни заново”. Ниже рассматриваются наиболее известные из таких подходов.

В главе 18 рассматриваются подходы к постепенной реорганизации предприятий. Анализируется классический подход фирмы IBM, разработанный Мартином (Martin) в середине 70-х годов - методика BSP (Business Systems Planning). Современный вариант этой методики, адаптированный автором с целью ее ориентации на современные структурные методы, фактически описан в главах 10-12. Рассматривается подход к непрерывному улучшению процессов Деминга и его японский аналог управления качеством, а также подход CMM как применение подхода Деминга для улучшения процессов разработки и сопровождения программного обеспечения.

Глава 19 посвящена введению в “жесткий” реинжиниринг Хаммера (Hammer) и Чампи (Champy). Приводится и расшифровывается определение BPR, анализируются общие свойства перепроектированных бизнес-процессов, перчисляются причины неудач при проведении BPR.

В главе 20 рассматриваются два популярных метода оценки деятельности предприятия, помогающей осуществить переход от модели “как есть” к модели ”как должно быть”: динамическое моделирование на базе сетей Петри и функционально-стоимостное моделирование ABC.

ГЛАВА 18

ПОДХОДЫ К УЛУЧШЕНИЮ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ

18.1. Реорганизация деятельности по методике BSP

Методика BSP определяется как “подход, помогающий предприятию определить план создания информационных систем, удовлетворяющих его ближайшие и перспективные информационные потребности”. Информация является одним из основных ресурсов и должна планироваться в масштабах всего предприятия, информационная система должна проектироваться независимо от текущего состояния и структуры предприятия.

BSP основывается на нисходящем анализе информационных объектов и регламентирует 13 этапов выполнения работ. Особенностью подхода является выделение трех организационных этапов, обеспечивающих так называемый “запуск” проекта, а именно:

- получение поддержки руководства предприятия
- подготовка к анализу
- проведение стартового совещания.

На этапе 4 формируется перечень основных деятельностей предприятия и содержащихся в них бизнес-процессов и дается их краткое описание. Фрагмент такого перечня для автобазы приведен в таблице 18.1 (см. соответствие с рис. 12.1).

Таблица 18.1

Деятельность	Процесс
Ремонт и техническое обслуживание	Диагностика Ремонт
	Техническое обслуживание
	Учет на оборотном складе
Эксплуатация	Оперативный диспетчерский учет
	Организация перевозок
Контроль безопасности	Контроль безопасности движения

Контроль пожарной безопасности
Контроль технической безопасности

На этапе 5 выявляются основные классы данных (логически связанные категории данных). Для нашего примера такими классами являются: Сотрудники, Ремонты, Технологический транспорт и т.д. В итоге выполнения этапов 4 и 5 формируется матрица связей, пример которой приведен в таблице 18.2. Для упрощения по вертикали приведены деятельности, реально необходимо связывать бизнес-процессы. На пересечении строк и столбцов указывается тип доступа к классам данных (чтение, запись).

Таблица 18.2

	Сотрудники	Запчасти	Ремонты	Транспорт	Перевозки
Ремонт	Чт	Чт/Зп	Зп	Чт/Зп	-
Эксплуатация	Чт/Зп	-	-	Чт/Зп	Чт/Зп
Контроль безопасности	Чт	-	-	Зп	-

На следующем (шестом) этапе осуществляется анализ существующих на предприятии деловых и системных взаимодействий. По аналогии с этапом 5 строятся следующие четыре матрицы, демонстрирующие использование существующих и планируемых информационных подсистем:

- матрица “руководители - процессы”, демонстрирующая основные обязанности руководителей, степень их вовлеченности в основные бизнес-процессы предприятия;
- матрица “информационные системы - руководители”, показывающая какими системами (существующими или планируемыми) пользуются руководители;
- матрица “информационные системы - процессы”, демонстрирующая как системы соотносятся с бизнес-процессами предприятия;
- матрица “информационные системы - файлы данных”, показывающая, какие файлы данных и какими системами используются.

На седьмом этапе осуществляется интервьюирование руководителей с использованием построенных матриц. При этом необходимо решить следующие задачи:

- уточнение матриц
- определение и оценка необходимой руководству информации
- определение приоритетов потребностей
- определение текущих задач
- привлечение на свою сторону руководства.

На этапе 8 производится обработка интервью. Прежде всего каждая выявленная проблема или точка зрения фиксируется с указанием ее возможного решения, оценкой результатов предложенного решения, потребности в информационной системе, затрагиваемого и причинного процессов. Далее все проблемы разделяются на три вида:

- проблемы, не относящиеся к автоматизации и не затрагивающие информационные системы;
- проблемы, связанные с существующими информационными системами;
- проблемы, связанные с будущими системами.

Проблемы первого вида передаются руководству предприятия для принятия соответствующих решений. Оставшиеся проблемы сортируются по бизнес-процессам.

На следующем (девятом) этапе традиционными методами осуществляется проектирование архитектуры информационной системы. Десятый этап определяет приоритеты в реализации и намечает последовательность ее этапов. Этап 11 определяет планирование модификаций информационной системы в связи с постоянным процессом появления новых требований к такой системе. Наконец, этапы 12 и 13 заключаются в выработке рекомендаций и планов и формировании отчетности по проведенным работам.

Анализ и реорганизация деятельности предприятия производится на основе построенных матриц и выявленных проблем (естественно, эти матрицы детализируются до уровня бизнес-функций), основные изменения осуществляются с целью ориентации предприятия на спроектированную информационную систему.

18.2. Подход CPI/TQM

Подход CPI (Continuous Process Improvement) и его японский аналог TQM (Total Quality Management) успешно применялись при реорганизации предприятий еще в середине века. Самый впечатляющий результат его применения - подъем японской послевоенной промышленности и доведение качества японских товаров до современного опережающего многие страны уровня. Этот подход продолжает активно использоваться и в настоящее время, о чем свидетельствует, например, возрастающий объем применения стандартов серии ISO 9000, фактически поддерживающих CPI.

В основе подхода лежит очевидная концепция управления качеством выпускаемой продукции. Качество должно быть направлено на удовлетворение текущих и будущих потребностей потребителя как самого важного звена производственной линии. Достижение соответствующего уровня качества требует постоянного совершенствования производственных процессов. Для решения этой задачи Демингом было предложено 14 принципов, в совокупности составляющих теорию управления и применимых для предприятий произвольных типов и различных масштабов. Безусловно, этих принципов недостаточно для полного решения стоящих перед современными предприятиями проблем, тем не менее они являются основой трансформации промышленности Японии и США.

Принцип 1. Постоянное совершенствование товара или услуги, что предполагает:

- долгосрочное планирование
- введение новых услуг/товаров, использование новых материалов, обновление способов производства и производственного оборудования
- затраты на исследования и образование
- приближение к потребителю за счет непрерывного совершенствования конструкции товара и формы услуги.

Принцип 2. Следование новой философии производства (рожденной в Японии и распространяемой по всему миру). Производственный брак, ошибки, “получающие” (а не зарабатывающие) деньги сотрудники, неэффективный контроль, некомпетентное руководство, неквалифицированный персонал, подсиживания и доносы, грязь на рабочих местах, вандализм по отношению к средствам производства - все это ведет к недовольству своей работой и, как следствие, к недобросовестному исполнению своих обязанностей.

Принцип 3. Отказ от массового контроля. Контроль качества готового товара является запоздалой и дорогостоящей мерой, неявно подразумевающей планирование брака. Необходимо оптимизировать производство, а не контроль, качество готового товара невозможно улучшить.

Принцип 4. Установление долгосрочных партнерских отношений. Следует прекратить практику установления взамодействий между покупателем и поставщиком только на основании цены. Сама по себе безотносительно к качеству цена не имеет смысла. Из дешевых некачественных комплектующих невозможно получить качественный товар. Более того, замена качественных

комплектующих одного поставщика на качественные комплектующие другого приводит к потерям времени, перспективе срыва планов, а следовательно, к нервозности и повышению вероятности появления брака. Постоянное повышение качества гораздо важнее цены, но оно достигается только на основе долгосрочных доверительных отношений между партнерами.

Принцип 5. Постоянное совершенствование системы производства и обслуживания. Качество должно закладываться в товар при его производстве. Шанс добиться успеха заключается в следовании правилу, что любой товар является для потребителя единственным в своем роде. Каждый следующий заказ должен выполняться лучше предыдущего, необходимо постоянно совершенствовать материалы, умение и навыки сотрудников и т.п.

Принцип 6. Обучение руководства. Руководство нуждается в обучении, чтобы знать все процессы предприятия от исходных материалов до потребителей, так как одна из основных задач руководства - оценка отклонений. Японский управляющий начинает свою работу в компании с низших звеньев. За 5-10 лет он проработает в разных подразделениях, и ему будут знакомы все проблемы производства.

Принцип 7. Функция руководителя - руководство, а не надзор. Руководитель должен, прежде всего, знать работу, которую он контролирует, быть лидером и стремиться к повышению качества товаров и услуг. В его прямые обязанности должно входить устранение препятствий, мешающих сотрудникам гордиться результатами своего труда. Он должен обладать реальной властью и быть ориентированным на информирование высшего руководства предприятия о проблемах и условиях, нуждающихся в изменении.

Принцип 8. Устранение страха. Если человек не чувствует себя защищенным, боится высказывать идеи и задавать вопросы, в конечном счете, боится потерять работу, он никогда не достигнет наилучших показателей.

Принцип 9. Разрушение барьеров между подразделениям. Сотрудники одного подразделения должны знать о проблемах, возникающих в смежных подразделениях. Фактически у каждого сотрудника есть свой потребитель в смежном подразделении: один, например, осуществляет закупку материалов, другой конструирует из них изделия, источником брака может быть деятельность любого из них.

Принцип 10. Отмена лозунгов. Необходимо исключить лозунги, плакаты, наставления и т.п. с призывами о повышении производительности и улучшении качества. Они адресованы не тем людям. Как может рабочий не допускать брак, если поставляемые детали (а выбор поставщика от него не зависит) являются некачественными?

Принцип 11. Отказ от количественных показателей. Нормативы, как правило, устанавливаются из расчета на среднего рабочего. Способные делать больше ограничиваются нормой, те, кто не может ее выполнить, производят брак, приносят убытки и текучесть кадров. Еще хуже сдельная работа, откровенно стимулирующая брак: рабочий понимает, что чем больше он выпустит деталей (не важно, бракованных или нет), тем больше ему заплатят. В Японии нет ни одного завода, на котором работают сдельно. Кроме этого, работников, устанавливающих нормы и считающих производительность, часто больше, чем непосредственно занятых на производстве.

Принцип 12. Поддержка профессиональной гордости. Необходимо устранить препятствия, лишающие людей их профессиональной гордости. Нельзя заставлять рабочих гнать план любой ценой, заставлять производить товары из некачественных материалов, относиться к людям как к товару и т.п. Как можно гордиться своей работой при требованиях выполнения плана невзирая на качество, плохих инструментах, постоянных проявлениях недоверия в виде ежегодных аттестаций, учетах заслуг и просчетов. Работа является вторым домом человека, мы всегда говорим “на нашем предприятии”, “наш заказ”, “моя работа” - и это чувство нуждается в поддержке.

Принцип 13. Поощрение образования и совершенствования. “Хороший человек” - это не профессия. Предприятию нужны квалифицированные специалисты, постоянно совершенствующиеся с ориентацией на новейшие технологии в своей отрасли. Более того, в любой отрасли существует дефицит людей с высоким уровнем знаний.

Принцип 14. Необходимые действия для осуществления изменений:

- Инициатива изменений исходит, как правило, от руководителей среднего звена. Именно здесь лежит слой бизнес-правил предприятия и происходит понимание необходимости внесения в них изменений.
- Убеждение высшего руководства и получение его согласия и поддержки в проведении изменений.
- Объяснение (обладающими реальной властью руководителями) как можно большему числу сотрудников предприятия необходимости перемен и необходимости их непосредственного участия в процессе.
- Создание группы, перед которой ставится задача совершенствования качества.

Перечисленные действия необходимы для успешного старта проекта по улучшению качества.

Таким образом, данный подход характеризуется ориентацией на требования рынка и потребителя и применим в условиях, когда существует достаточная стабильность производства и желание сохранения кадров.

18.3. Требования CMM для совершенствования разработки программного обеспечения

Требования CMM (Capability Maturity Model) разработаны институтом SEI (Software Engineering Institute) для предприятий, стремящихся к осуществлению качественного процесса разработки и сопровождения программного обеспечения, и являются примером применения подхода CPI для конкретной отрасли промышленности.

CMM описывает характеристики совершенства (качества) процессов разработки и сопровождения ПО (ПО-процессов), а также критерии перехода от “плохих” к хорошо управляемым ПО-процессам в терминах уровней совершенства модели. CMM применяется для:

- улучшения ПО-процессов, когда предприятие планирует, разрабатывает и реализует их изменения;
- оценки ПО-процессов, когда определяется состояние текущих ПО-процессов предприятия и приоритетные процессы, а также осуществляется организационная поддержка их улучшения;
- оценки возможностей ПО при квалификации партнеров, осуществляющих заказную разработку ПО или управляющих состоянием существующих ПО-процессов.

Фактически CMM является комплексом требований к ключевым элементам эффективного ПО-процесса и способам его эволюционного улучшения. CMM поддерживает этапы планирования, инжиниринга, управления разработкой и сопровождением ПО, что улучшает возможности предприятия в достижении целей по стоимости, функциональности и качеству производимого ПО.

CMM декларирует 5 уровней совершенства ПО-процесса, определяющих его возможности (т.е. описывающих вырабатываемые им результаты). Каждый из уровней (за исключением первого) включает несколько ключевых областей процесса, содержащих цели эффективной реализации проекта. Фактически набор целей и определяет рассматриваемый уровень совершенства ПО-процесса. В свою очередь, каждая из ключевых областей организована в виде 5 разделов (взятие обязательств, осуществимость, выполнение, оценка и анализ, верифицируемая реализация), названных общими характеристиками и регламентирующих эффективность, повторяемость и продолжительность действий по достижению целей из ключевой области процесса. Наконец, каждая из общих характеристик специфицирует собственные ключевые применения, содержащие

действия, совокупное выполнение которых и позволяет достигнуть целей ключевых областей процесса (рис. 18.1).

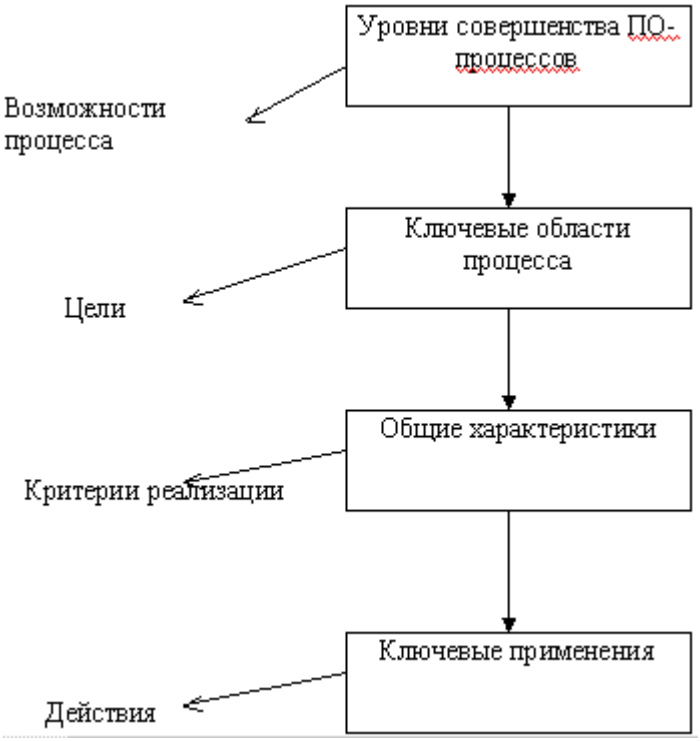


Рис. 18.1. Уровни совершенства CMM

Ниже кратко описывается каждый из 5 уровней совершества ПО-процесса.

Уровень 1 - Инициализация. Возможности ПО-процесса на этом уровне непредсказуемы, поскольку процесс постоянно модифицируется по мере работы. Его характеристики зависят от исполнителей, различий в их подходах, знаниях и мотивациях.

Уровень 2 - Повторение. На данном уровне решаются задачи управления проектом ПО и устанавливаются процедуры решения задач управления. Планирование и управление новыми проектами основывается на опыте аналогичных проектов. Цели уровня заключаются в установлении эффективных процессов управления ПО-проектами, позволяющих предприятию использовать успешный опыт других проектов (при этом отдельные процессы могут и отличаться от ранее выполненных). При этом эффективным процессом считается практичный, документированный, измеряемый, способный к улучшению и хорошо осваиваемый процесс. Ключевыми областями процесса являются:

- *Управление требованиями.* Целью является установление “взаимопонимания” между пользователями и проектными спецификациями, основанными на их требованиях. Это является основой планирования и управления ПО-проектами.
- *Планирование ПО-проекта.* Целью является формирование разумных планов для проектирования ПО и управления ПО-проектом, без таких планов проект не может быть выполнен эффективно.
- *Ведение проекта.* Целью является отслеживание текущего состояния проекта и эффективных воздействий на него в случае отклонений от планов.
- *Управление подпроектами.* Цель заключается в выборе квалифицированных субподрядчиков и эффективных способов управления ими.
- *Гарантия качества.* Целью является обеспечение управления наблюдаемостью и возможностью исследовать ПО-проект и создаваемый программный продукт.
- *Управление конфигурацией ПО.* Целью является установление и поддержка состава и конфигурации ПО в проекте на протяжении всего жизненного цикла проекта.

Уровень 3 - Определение. Включает стандартные организационные ПО-процессы, регламентирующие эффективную разработку ПО и управление проектами и содержащие следующие ключевые области:

- *Представление организационного процесса.* Целью является установление организационных соответствий деятельностям ПО-процессов, улучшающим предприятие за счет возможностей ПО-процессов.
- *Определение организационного процесса.* Цель - развитие и сопровождение полезной совокупности ПО-процессов для их использования на предприятии.
- *Программа обучения.* Целью является повышение квалификации персонала для эффективного выполнения своих обязанностей.
- *Интегрированное управление ПО.* Цель - интеграция деятельностей по проектированию ПО и управлению в ПО-процесс, “прошитый” стандартами предприятия и включающий бизнес-окружение и технические требования к проекту.
- *Разработка ПО.* Цель - выработка четко определенного процесса разработки, интегрирующего все деятельности по созданию корректного, состоятельного и эффективного ПО.
- *Координация рабочих групп.* Целью является распределение обязанностей между субподрядчиками для эффективного проектирования, а также отслеживание межгрупповых взаимодействий.
- *Просмотр.* Цель - улучшение понимания ПО и его корректировка на ранних стадиях разработки.

Уровень 4 - Управление. Включает установление количественных и качественных оценок как для ПО-процессов, так и для используемых в них инструментальных средств и содержит следующие ключевые области:

Количественное управление процессами. Целью является установление количественных характеристик ПО-процессов (в том числе и промежуточных) на всех этапах жизненного цикла.

Качественное управление процессами. Целью является развитие количественных оценок в сторону критериев качества ПО и их достижение.

Уровень 5 - Оптимизация. Декларирует совершенствование предприятия как непрервное улучшение ПО-процессов и содержит следующие ключевые области, охватывающие как само предприятие и выполняемые текущие проекты, так и собственно ПО-процессы:

- *Предупреждение ошибок.* Целью является определение причин ошибок и принятие мер к предотвращению их повторений за счет изменения соответствующих ПО-процессов.
- *Управление технологическими изменениями.* Целью является анализ применимости новых технологий и их ориентация на конкретное предприятие.
- *Управление изменениями ПО-процессов.* Целью является улучшение ПО-процессов на предприятии (повышение качества, увеличение продуктивности, снижение времени на разработку).

18.4. ISO 9000 - стандарт на качество проектирования, разработки, изготовления и послепродажного обслуживания

ISO 9000 определяет базовый набор мероприятий по контролю качества и представляет собой схему функционирования бизнес-процессов предприятия, обеспечивающую высокое качество его работы. В то же время ISO 9000 не является стандартом качества собственно для производимых предприятием товаров/услуг. Схема покрывает все этапы выпуска товаров/услуг, включая закупку сырья и материалов, проектирование, создание и доставку товаров, обслуживание клиентов, обучение персонала и т.п.

ISO 9000 (на самом деле представляющий собой серию стандартов 9000, 9001, 9002, 9003, 9004) регламентирует два ключевых момента:

- наличие и документирование соответствующего бизнес-процесса
- измеряемость его качества.

Сертификация предприятия по стандарту ISO 9000 включает следующие три этапа:

- применение стандартов на предприятии, заключающееся в разработке и вводе в действие ряда мер (процессов), предписываемых стандартами;
- проведение собственно сертификации аккредитованными ISO органами;
- периодические (2 раза в год) проверки предприятия на предмет следования стандартам.

Следует отметить, что сертификация по ISO 9000 является добровольным делом каждого предприятия. Основной побудительной причиной сертификации является то, что многие зарубежные компании требуют наличие сертификата от своих поставщиков (например, для поставщиков NASA и Министерство обороны США - это является обязательным условием). Более того, наличие сертификата может оказаться обязательным условием участия предприятия в международных тендерах, госзаказах, а также получения льготных кредитов и страховок.

ГЛАВА 19

BPR - РЕИНЖИНИРИНГ ПО ХАММЕРУ И ЧАМПИ

Данная глава основывается на книге Хаммера (Hammer) и Чампи (Champy) “Reengineering the Corporation” и представляет собой краткий конспект основных ее положений.

Хаммер и Чампи определяют реинжиниринг (BPR - business process reingineering) как фундаментальное переосмысление и радикальное перепланирование бизнес-процессов компаний, имеющее целью резкое улучшение показателей их деятельности, таких как затраты, качество, сервис и скорость. Это определение содержит четыре ключевых слова.

Первое ключевое слово - “фундаментальное”. При BPR организаторы производства должны задать себе основные вопросы, касающиеся работы компаний: *Почему мы делаем то, что мы делаем? И почему мы делаем это так, а не иначе?* Эти фундаментальные вопросы заставляют людей задуматься над негласными правилами и предположениями, определяющими управление бизнесом. Эти правила часто оказываются устаревшими, ошибочными или просто неподходящими для конкретной ситуации, тем не менее они изначально заложены в большинство процессов. Поэтому BPR начинается с того, что отбрасываются все предположения и все данности. Например, вопрос: “Как наиболее эффективно проверить кредитоспособность клиента?” предполагает, что такая проверка необходима. Во многих случаях, однако, затраты на проверку кредитоспособности могут превысить потери, связанные с неуплатой долгов, которые эта проверка помогает избежать. То есть при перепроектировании сначала определяется, что должна делать компания, а затем, как она должна это делать. BPR не принимает ничего как данность. Он игнорирует то, *что* есть, и концентрируется на том, что *должно быть*.

Второе ключевое слово в определении - “радикальное”. Радикальное перепланирование означает проникновение в корень вещей - не поверхностные изменения, а отбрасывание старого и изобретение абсолютно новых путей выполнения работы. BPR - это почти то же, что изобретение бизнеса заново, а не просто улучшение, усовершенствование, модификация бизнеса.

Третье ключевое слово - “резкие”. При BPR не просто вносятся незначительные изменения, а резко (в разы и порядки) увеличиваются ее показатели. Если, например, требуется на 10% повысить производительность и улучшить качество обслуживания клиентов, то эта компания *не нуждается* в BPR. Из ямы глубиной 10% ее могут вытащить традиционные методы - начиная от произнесения зажигательных речей перед сотрудниками и заканчивая проведением программ повышения качества. Незначительные улучшения достигаются путем настройки; для того, чтобы добиться резких улучшений, необходимо взорвать все старое и заменить новым.

Четвертое ключевое слово - “процессы”. Это - наиболее важное слово в определении, и в то же время именно оно является наиболее затруднительным для понимания, так как большинство людей ориентированы не на процессы, а на задачи, рабочие места, персонал. Под бизнес-процессом понимается совокупность действий, получающая на входе данные различных типов и продуцирующая результат, имеющий ценность для потребителя. Например, процесс выполнения заказа на входе получает заказ и выдает в качестве результата заказанные товары, т.е. доставка заказанных товаров потребителю и есть та ценность, которую создает процесс. Находясь под влиянием идей Адама Смита относительно разбиения работы на простейшие задачи и поручения каждой из них низкоквалифицированному специалисту, современные компании сосредотачиваются на отдельных задачах, составляющих этот процесс - оформление заказа, получение товаров на складе и т.п., и имеют тенденцию терять из виду главную цель - доставку товаров в руки заказчика. Отдельные задачи, составляющие данный процесс, безусловно важны, но для заказчика ни одна из них не будет иметь значения, если весь процесс в целом не работает - то есть не производит доставки товаров.

Авторы выделяют три типа компаний, которые приходят к пониманию необходимости BPR. Во-первых, компании, оказавшиеся в тяжелом положении. У них нет выбора: затраты на порядок выше, чем у конкурента, обслуживание клиентов поставлено так, что вызывает их открытую ругань и т.п. Во-вторых, компании, которые еще не находятся в тяжелом положении, однако их руководство обладает достаточным даром предвидения, чтобы понять, что беда надвигается: появляются новые конкуренты, у клиентов возникают новые требования, изменяется правовая или экономическая среда. Эти компании достаточно дальновидны, чтобы начать реинжиниринг перед лицом крупных неприятностей. Третий тип компаний - это те, кто находится на вершине успеха. У них нет заметных трудностей (как сейчас, так и в перспективе), но их руководство ведет честолюбивую и агрессивную политику и стремятся увеличить свой отрыв от конкурентов, сделать их жизнь еще тяжелее. Признаком действительного успеха работы компании является ее готовность отказаться от того, что приносило ей успех на протяжении долгого времени. Истинно великая компания никогда не удовлетворена своим текущим состоянием и качеством работы и охотно отказывается от устоявшейся практики в надежде и расчете на нечто лучшее.

Различия между этими тремя типами компаний ярко характеризуется следующими образами: входящие в первую категорию безнадежны - они врезались в стену и лежат в обломках. Те, кто входит во вторую категорию, идут вперед на крейсерской скорости, однако в свете своих бортовых огней видят: что-то приближается. Может быть, это стена? Входящие в третью категорию совершают автомобильную прогулку. День ясный, и не видно никаких препятствий. По их мнению, это самый подходящий момент, чтобы остановиться и построить стену для кого-нибудь другого.

19.1. Переосмысление бизнес-процессов

Понятно, что бизнес-процесс, возникший в результате BPR, должен сильно отличаться от традиционного, принимая при этом самые различные формы. Тем не менее различные перепроектированные процессы обладают многими общими харктеристиками и свойствами. Эти общие свойства не зависят от отрасли производства и даже от индивидуальных особенностей отдельного процесса. Многое из того, что справедливо для автомобильной компании, перепроектировавшей свои процессы, справедливо также для страхового общества или магазина.

Не следует удивляться, что на разных предприятиях при проведении BPR повторяется одно и то же. Дело в том, что облик прошедшего BPR предприятия, так же как и облик традиционного предприятия, основан на нескольких изначальных предпосылках. Индустриальная модель строится на предпосылке, что работники обладают невысокой квалификацией, и им недостает времени и способностей для обучения. Поэтому задачи, предлагаемые этим работникам, должны быть очень простыми. Более того, Адам Смит доказывал, что люди работают наиболее эффективно тогда, когда им предлагается для выполнения всего одна хорошо понятная им работа. Однако для того, чтобы связать все простые задачи вместе, требуются сложные процессы. В результате на протяжении двухсот лет предприятия принимали как должное неудобство, неэффективность и дороговизну сложных процессов, необходимых для того, чтобы использовать

преимущества, обеспечиваемые простыми задачами. При BPR эта модель переворачивается с ног на голову: утверждается, что для получения высокого качества, уровня обслуживания, гибкости, низких затрат и т.п. процессы должны быть простыми.

Ниже приводятся общие свойства и характеристики перепроектированных бизнес-процессов.

1) Несколько работ объединяются в одну. Основная особенность перепроектированных процессов - отсутствие сборочного конвейера: многие работы и задачи, которые раньше выполнялись по отдельности, теперь объединяются в одну, выполняемую одним специалистом (или, по крайней мере, специалистами одного подразделения, на которых возложена полная ответственность за выполнение работы). Безусловно, в таком подразделении могут возникнуть проблемы, связанные с распределением заданий и приводящие к задержкам и ошибкам, но они будут незначительными по сравнению с проблемами, возникавшими раньше, когда задания распределялись между разными подразделениями. Критичным является то, что теперь каждый сотрудник знает, кто отвечает за быстрое и точное выполнение работы.

Выигрыш от введения интегрированных процессов и отвечающих за них сотрудников может быть огромным. Предприятие избавляется от ошибок, задержек и дополнительной работы в связи с проблемами, которые возникали при распределении заданий. Обычно процесс, ориентированный на такого сотрудника, выполняется в десять раз быстрее, чем его конвейерная версия. При этом, поскольку новый процесс порождает меньшее число ошибок и недоразумений, предприятию не нужны дополнительные работники для их исправления.

Кроме этого, интегрированный процесс требует меньше работы по его администрированию. Поскольку сотрудники, задействованные в процессе, отвечают за то, чтобы работа исполнялась вовремя и без ошибок, контроля за ними требуется меньше. Вместо этого предприятие стимулирует этих работников, наделенных новыми полномочиями, к постоянному поиску новых, творческих путей уменьшения длительности рабочего цикла и затрат при производстве качественного продукта или услуги. Еще одно преимущество интегрированных процессов - улучшение качества управления: поскольку в этих процессах занято меньшее число людей, становится легче давать им работу и следить за ее выполнением.

2) Исполнители принимают решения. Предприятия, осуществляющие BPR, уплотняют процессы не только горизонтально (возлагая на сотрудников множество последовательных заданий), но также и вертикально. Вертикальное уплотнение означает, что в тех местах процесса, где сотрудники обычно обращались за ответом к руководству, теперь принимаются самостоятельные решения. В отличие от той ситуации, когда принятие решений изолируется от самой работы, в перепроектированном процессе принятие решений становится частью работы, и теперь сами исполнители выполняют ту часть работы, которую раньше выполняло руководство.

Идеология массового производства подразумевает, что у людей, фактически выполняющих работу, нет ни времени, ни желания контролировать ее выполнение, а также, что им недостает широты и глубины знаний для того, чтобы они могли принимать решения относительно своей работы. Из этого предположения и вытекает практика построения иерархических управленчесих структур, принятая на производстве. Бухгалтеры, ревизоры и контролеры фиксируют выполнение работы, проверяют и контролируют исполнителя, руководитель контролирует подчиненного, рассматривая исключительные случаи и принимая по ним решения. Это предположение необходимо отбросить. Выгоды от вертикального уплотнения рабочего процесса включают в себя уменьшение числа задержек, снижение затрат на управление, повышение уровня работы с клиентами и расширение полномочий сотрудников.

3) Этапы процесса выполняются в естественном порядке. Реинжиниринг процессов освобождает их от предопределенной линейной последовательности выполнения этапов, при которой работы очередного этапа начинаются по завершении предыдущего; в перепроектированном процессе этапы организованы в такой последовательности, в которой это необходимо. Отказ от линейности ускоряет работу процессов по двум причинам. Во-первых, многие этапы выполняются одновременно. Во-вторых, уменьшается время между началом и

окончанием выполнения процесса и тем самым снижается вероятность переделывания уже выполненной работы из-за устаревания информации или противоречий с ранее выполненными работами.

4) Существуют различные версии процесса. Традиционные (стандартизированные) процессы были ориентированы на производство массовой продукции для массового рынка. Все входные данные обрабатывались одинаково, и поэтому предприятия производили одинаковую продукцию. В настоящее время такая технология устарела: для того, чтобы удовлетворить современным требованиям, необходимо иметь несколько версий одного и того же процесса, каждая из которых была бы настроена на требования различных заказчиков, ситуаций и входных данных.

Авторы приводят пример с человеком, который для проведения мелких перестроек своего дома должен был в течение шести месяцев ожидать, когда его вопрос будет заслушан городскими властями, которые дали свое согласие в течение двадцати секунд. Его заявление, к которому был приложен набросок, сделанный от руки, прошло те же инстанции, что и проекты офисов, состоящие из многих томов чертежей, планов и списков требуемых материалов (последующее строительство которых обошлось в миллионы долларов). Если бы городские власти перепроектировали свою систему рассмотрения заявлений на постройку, им, вероятно, потребовалось бы иметь три версии процесса: для мелких, крупных и средних проектов. Простой сортировки, основанной на заранее заданных признаках, оказалось бы достаточно, чтобы быстро и эффективно направлять заявление каждого клиента по нужному каналу и эффективно обрабатывать его.

Традиционные процессы обычно сложны, поскольку включают в себя специальные средства для обработки массы ситуаций, в том числе исключительных. Наоборот, процесс, разложенный на несколько разных версий, прост, поскольку каждая версия обрабатывает только “свои” случаи, особые случаи и исключения отсутствуют.

5) Работа выполняется там, где ее целесообразно делать (выход работы за границы организационных структур). На традиционных предприятиях работа выполняется специалистами, сгруппированными в тематические подразделения: бухгалтер умеет считать, а работник отдела снабжения умеет заказывать товары. Когда бухгалтерии нужны новые карандаши, приобретает их отдел снабжения. Снабженцы находят поставщиков, оговаривают цены, составляют заказ, проверяют поставленный товар и оплачивают счет. Это дорогостоящий процесс, поскольку в его выполнении участвует несколько разных отделов, а также из-за дополнительной деятельности, связанной с обработкой всех нужных бумаг и связыванием воедино всех частей процесса.

Одна компания в результате эксперимента выяснила, что ее внутренние расходы на покупку батареек стоимостью $3 составили $100, а также, что 35% всех ее заказов на снабжение были на товар стоимостью менее $500. Компанию не обрадовала перспектива и в будущем тратить $100 для того, чтобы закупить товара на $500 или меньше. Поэтому она решила возложить ответственность за закупку товара на потребителей этого процесса. Другими словами, теперь бухгалтеры сами покупают свои карандаши, и то же делают все остальные. Они знают, к кому обратиться и сколько заплатить, поскольку отдел снабжения оговорил с поставщиками все цены и выдал бухгалтерии список одобренных поставщиков. В результате заказчики получают свои заказы быстрее и проще, а компания теперь тратит гораздо меньше $100 на обработку заказа.

Другими словами, после BPR связи между процессами могут сильно отличаться от первоначальных, при этом работы могут выноситься за рамки организационных структур для более эффективного выполнения процесса. Поскольку значительная часть выполняемой предприятиями работы состоит в интегрировании ее взаимосвязанных компонентов, выполняемых независимыми подразделениями, то вынос работы за пределы организационных структур вообще исключает необходимость в подобном интегрировании.

6) Снижение доли работ по проверке и контролю. Традиционные процессы изобилуют операциями проверки и контроля, которые являются разновидностью бесполезной работы,

поскольку не создают никаких ценностей, но тем не менее нужны для того, чтобы гарантировать отсутствие нарушений. Например, при выполнении типичного процесса покупки отдел снабжения проверяет подпись заказавшего товар человека (чтобы убедиться, что он имеет на это право), а также имеется ли достаточно средств на счету данного подразделения. Все эти проверки имеют одну цель - убедиться, что подразделения не приобретают того, что они приобретать не должны. Возможно, эта цель достойна одобрения, но многие предприятия не замечают расходов, которые влечет организация строгого контроля. Все эти проверки требуют времени и сил - в некоторых случаях даже больших, чем сам процесс приобретения товара (более того, расходы на проверку могут оказаться выше стоимости приобретаемых товаров).

Перепроектированный процесс использует более сбалансированный подход. Вместо того, чтобы немедленно проверять каждую операцию, в перепроектированном процессе применяется групповой или отложенный контроль. Такие системы контроля терпимо относятся к случаям незначительных нарушений, перенося на более поздний срок момент их обнаружения или анализируя совокупность действий, а не каждое из них в отдельности. Тем не менее перепроектированные системы контроля оказываются более выгодными, несмотря на возможное увеличение числа нарушений, благодаря снижению расходов и прочих неудобств, связанных с самим контролем.

7) Минимизация согласований. Согласования являются другой разновидностью бесполезной работы, которую перепроектированный процесс сводит к минимуму. Это достигается уменьшением числа точек внешнего контакта, имеющихся в процессе, что приводит к снижению вероятности получения противоречивых данных, по которым, собственно, и требуется согласование.

Так процесс оплаты счетов в компании Ford включал три точки контакта с поставщиками: в отделе снабжения - в виде заказа, в отделе получения товаров - в виде документа на получение товара, и при оплате счетов - в виде счета. Эти три точки контактов являлись источниками возникновения противоречивой информации: заказ мог содержать расхождения с документом на получение или счетом, два последних документа также могли содержать противоречивую информацию. Исключив один из документов (счет), компания Ford сократила число точек внешнего контакта до двух, соответственно, вероятность возникновения противоречивых данных уменьшилась на 1/3. В результате отпала необходимость в работах по проверке и согласованию, которые ранее выполнялись при оплате счетов.

8) Ответственный менеджер является единственной точкой контакта. Ответственный менеджер бывает полезен в ситуациях, когда различные этапы процесса настолько сложны или до такой степени разбросаны в пространстве, что одному человеку (и даже небольшой группе) не под силу справиться с их интеграцией. Выступая в качестве буфера между сложным процессом и заказчиком, он ведет себя как отвечающий за выполнение всего процесса, хотя на самом деле это и не так. Для того, чтобы выполнить эту функцию (то есть иметь возможность ответить на вопросы заказчика и разрешить возникшие у него проблемы) ответственный менеджер должен иметь доступ ко всем информационным системам, используемым сотрудниками, реально выполняющими процесс, а также возможность контактировать с этими людьми, при необходимости получая от них помощь и ответы на свои вопросы.

9) Сочетание централизованных и децентрализованных операций. Прошедшие BPR предприятия имеют возможность сочетать выгоды, получаемые от централизации и децентрализации, в одном и том же процессе. Информационная технология позволяет подразделениям действовать, как полностью автономным единицам, в то же время пользуясь преимуществами централизации. Например, снабжение торговых представителей на местах компьютерами класса notebook, соединенными с помощью модемов с головным отделением, дает им возможность получения немедленного доступа к хранящейся там информации. В то же время средства контроля, заложенные в программное обеспечение, используемое этими торговыми представителями при заключении контрактов, не позволяют им устанавливать неразумные цены либо задавать условия доставки (или какие-то иные условия), которые предприятие не может выполнить. Используя эту технологию, предприятия могут перепроектировать процесс продаж

таким образом, чтобы избавиться от бюрократического аппарата в местных отделениях, повысить независимость и уровень полномочий торговых представителей и одновременно усовершенствовать контроль за расценками и условиями продаж.

Приведенные примеры и характеристики, общие практически для всех перепроектированных процессов, вовсе не доказывают, что все перепроектированные процессы выглядят одинаково или что перепроектирование процесса - это нечто простое и стандартное. Этого и не может быть, хотя бы потому что некоторые характеристики конфликтуют друг с другом. На самом деле создание нового плана выполнения процесса требует глубокого знания самого процесса, творческого подхода и дальновидности.

19.2. BPR и индуктивное мышление

Для того, чтобы оценить мощности, заложенные в современных информационных технологиях и представить себе все возможности их применения, необходимо пользоваться особым образом мышления, которым люди обычно не владеют и который приводит их в замешательство. Большинство менеджеров и директоров умеют мыслить дедуктивно. Это означает, что они могут определить проблему или проблемы, а затем найти решения и дать им оценку. Но применение BPR требует индуктивного мышления, то есть способности сначала найти мощное решение, а затем найти подходящие проблемы, которые оно поможет преодолеть, причем руководство предприятия может даже не осознавать, что эти проблемы у него есть. Так руководство Ford изначально считало, что их проблема - найти способ обработки счетов поставщика быстрее и с участием меньшего количества людей. Вместо этого они нашли такое решение, которое позволило им вообще избавиться от счетов.

Основная ошибка, которую допускают предприятия, рассматривая технологии - это то, что они рассматривают их с позиций существующих бизнес-процессов. Они задают себе вопрос: “Как можно использовать возможности, предоставляемые новыми технологиями, для того, чтобы усовершенствовать (рационализировать, улучшить) то, что мы уже делаем?” Вместо этого они должны спросить себя: “Как можно использовать новые технологии для того, чтобы делать то, чего мы еще не делаем?” BPR отличается от автоматизации тем, что оно связано с введением нового, с использованием последних возможностей технологий для достижения абсолютно новых целей. Один из самых трудных моментов при реинжиниринге - выявление новых, неизвестных возможностей технологии, которыми можно заменить то, что уже известно.

Даже основатель IBM Томас Дж. Ватсон-мл. подпал под влияние этой недальновидной психологии, когда он заявил, что мировая потребность в компьютерах ограничивается цифрой, меньшей 50. Двадцать лет спустя изготовители больших компьютеров типа mainframe отвергли идею миникомпьютера, назвав его игрушкой. Через десять лет такую же оценку получил персональный компьютер: “Мы уже удовлетворяем свои потребности при помощи больших машин,” - таков был традиционный ход мысли, - “для чего же нам маленькие?” Ответ заключался в том, что преимущества миникомпьютеров и последовавших за ними персональных компьютеров лежали не в области применения больших машин, а в том, что они породили принципиально новые классы приложений.

Дедуктивный образ мысли применительно к технологии приводит не только к тому, что люди выпускают из вида ее важнейшие особенности, но и к тому, что слишком много внимания уделяется тривиальным и незначительным технологиям и приложениям. Недавно, например, кому-то пришла в голову идея объединения персонального компьютера с телефоном. В результате такого объединения экономится место на рабочем столе, кроме того, данное удовольствие обойдется дешевле приобретения по отдельности телефона и компьютера. Возможно, это и так, но объединение двух устройств в одно не содержит в себе никаких прорывов в смысле новых возможностей. Оно не позволяет людям делать нечто важное, что они не могли делать раньше: таким образом, это - незначительное улучшение.

Недостаток индуктивного мышления применительно к технологии - проблема не новая и характерная не только для непрофессионалов. Томас Эдисон однажды сказал, что ценность

изобретенного им фонографа заключалась в его способности “записывать последние желания умирающих джентльменов”. Один из изобретателей радио рассматривал его как беспроволочный телеграф, передающий сообщения из одного пункта в другой; он не заметил возможностей радио как средства широковещания.

В свое время реальные возможности ксерокопирования ускользнули даже от такой компании, как IBM. В конце 50-х годов, когда фирма Xerox проводила исследования по своей первой копировальной машине, эта компания испытывала проблемы и хотела продать свой проект. Он был предложен IBM, которая наняла фирму-консультанта для проведения исследований рынка. Консультант пришел к выводу, что даже если новое устройство на 100% вытеснит с рынка все существовавшие в то время средства копирования документов (копировальную бумагу, гектограф и др.), то вложения в нее все равно не окупятся. И естественно, IBM приняла решение отказаться от покупки и от копировального бизнеса. Теперь мы знаем (и кажется очевидным), что возможности копировальной машины Xerox заключаются не в том, чтобы заменить копировальную бумагу и другие копировальные технологии, а в ее способности выполнять действия, лежащие за пределами возможностей этих технологий. До изобретения ксерокопирования люди не знали, что у них есть потребность в тридцати копиях документа, которые можно раздать всем сотрудникам. Поскольку тридцать копий нельзя было сделать быстро и недорого, никто просто не воспринимал это как потребность.

Французский экономист начала XIX века Жан Баптист Сэй (Say) открыл закон, гласящий, что во многих случаях предложение рождает свой собственный спрос. Люди не знают, что какой-то продукт им нужен, пока они не могут его иметь. После этого оказывается, что они не могут без него жить. И случаи, когда технология создавала собственные области применения, о которых никто раньше не задумывался, как раз и являются примерами сработывания этого закона. Никому был не нужен Xerox до тех пор, пока он не появился. После этого скрытая, невыраженная потребность вдруг стала осязаемой и всеобъемлющей.

Следовательно, не имеет смысла просто спрашивать у людей, как они хотели бы использовать новую технологию в своем бизнесе. Они непременно ответят, как она могла бы усовершенствовать работу, которую они уже делают. Можно извлечь пользу из опроса о предпочтении людьми молока в стеклянных бутылках или в картонной упаковке. Все знают, что такое молоко и что это за два типа упаковки, поэтому они могут дать полезную информацию о своих предпочтениях и о причинах таких предпочтений. Однако, если бы исследователь рынка спросил у людей до того, как появилось ксерокопирование, об их отношении к копировальным машинам (что и было сделано), люди ответили бы, что эти машины не стоят того, чтобы использовать их вместо копировальной бумаги.

Другой пример. Если задать вопрос человеку, которому по работе приходится много путешествовать, что могло бы облегчить ему жизнь, то он, скорее всего, ответит, что ему нужен более удобный путь до аэропорта или выразит желание приобрести личный самолет. Но он не скажет, что ему нужно средство телепортирования, которое встречается в фантастических книгах. Он не скажет этого потому, что подобное средство находится за рамками его представлений. Когда он слышит о деловых поездках, то представляет себе известный ему процесс: автомобильные пробки по дороге в аэропорт, очередь, неудобное сиденье, ужасная еда. Вот проблемы, которые ему известны и решение которых он будет искать. Истинная сила новой технологии в том, чтобы предложить ему решения, о которых он не знает - например, как вообще избежать путешествия в самолете.

Корпорация Sony добилась своего успеха в значительной мере за счет того, что следовала этому основному правилу: бессмысленно изучать спрос на продукт, которого еще не существует. Когда руководство Sony впервые рассматривало идею создания плейера, то пришло к выводу, что не стоит проводить изучение спроса на этот продукт среди потенциальных покупателей. Было принято решение начать выпуск плейеров, исходя из представлений разработчиков о потребностях людей и из возможностей современной технологии. Плейер явился не ответом на потребности людей слушать музыку определенным образом, а, скорее, трансформировал эти потребности.

Главное заключается в том, что человеческие потребности, равно как и желания, формируются в рамках представлений о том, что возможно, а что нет. Революционная технология делает возможными такие вещи и действия, о которых никто раньше и не мечтал. И большинство предприятий допускают именно эту ошибку - они не могут увидеть всех возможностей, скрытых в новой технологии. Реальная сила новой технологии - не в том, чтобы заставить старые процессы работать лучше, а в том, чтобы дать предприятиям возможность изменить старые правила и создать новые способы выполнения работы.

Фактически, чтобы научиться индуктивно мыслить во время BPR, необходимо отказываться от существующих правил. Именно разрушительная сила технологии, ее возможность опрокидывать те или иные правила и ограничения, принятые в работе, и делает ее привлекательной для предприятий, находящихся в поиске новых возможностей. Ниже приведены примеры правил организации работы, которые могут быть отменены с появлением различных информационных технологий.

1) Старое правило: Информация может появляться одновременно только в одно время в одном месте

Новая технология: Совместно используемые базы данных

Новое правило: Информация может появляться одновременно во многих местах, если это требуется

2) Старое правило: Только опытные специалисты (эксперты) могут выполнять сложные работы

Новая технология: Экспертные системы

Новое правило: Менеджер может выполнять ту же работу, что и эксперт

3) Старое правило: Приходится выбирать между централизацией и децентрализацией

Новая технология: Телекоммуникационные сети

Новое правило: Можно одновременно использовать выгоды, получаемые от централизации и децентрализации

4) Старое правило: Руководитель принимает все решения

Новая технология: Средства поддержки принятия решений

Новое правило: Принятие решений входит в обязанности каждого

5) Старое правило: Персонал нуждается в офисах для получения, хранения, коррекции и передачи информации

Новая технология: Беспроволочные средства коммуникации и портативные компьютеры

Новое правило: Работники на местах могут получать и передавать информацию из любого места, где бы они ни находились

6) Старое правило: Лучший способ контакта с потенциальным покупателем - личный контакт

Новая технология: Интерактивный оптический диск

Новое правило: Лучший способ контакта с потенциальным покупателем - эффективный контакт

7) *Старое правило:* Чтобы найти объект, необходимо знать, где он находится

Новая технология: Автоматическое индексирование и отслеживание

Новое правило: Объекты сами сообщают, где они находятся

8) *Старое правило:* Планы периодически пересматриваются

Новая технология: Высокопроизводительные компьютеры

Новое правило: Планы могут пересматриваться мгновенно и постоянно

Из приведенных примеров очевидно, что по мере дальнейшего развития технологий будет происходить отказ от все большего количества правил, по которым организован бизнес. Правила, которые представляются непогрешимыми сегодня, могут устареть менее чем за год. Из этого вытекает, что использование возможностей, заложенных в новых технологиях, для изменений бизнес-процесса и резких его усовершенствований - это постоянная (а не одноразовая) деятельность. Следование новейшим технологиям и нахождение способов их применения на предприятии должно происходить непрерывно, так же, как исследования, разработки, маркетинг. Необходимо иметь практическое видение и творческую мысль для того, чтобы распознать возможности, скрытые в технологии, которая на первый взгляд кажется вообще не имеющей ничего общего с работой предприятия. Более того, предприятия должны сделать применение новых технологий одним из своих основных занятий, если они хотят быть впереди в наше время динамичных технологических изменений. Те, кто лучше сможет разглядеть и оценить возможности, скрытые в новой технологии, получат постоянное, растущее преимущество над конкурентами. Рекордсмена НХЛ по забитым шайбам и результативным передачам Уэйна Гретцки однажды спросили, что сделало его великим хоккеистом. Он ответил: “Я стараюсь быть там, где будет шайба, а не там, где она есть”. То же правило работает и применительно к новой технологии. Если Вы строите свою стратегию на основе того, что сейчас может приобрести каждый, то Ваше предприятие будет всегда в погоне за конкурентами, у которых это уже производится. А Ваши конкуренты, которые знают, что они будут делать с технологией до того, как она стала доступна, будут готовы к ее применению, когда она действительно станет доступной.

19.3. Причины неудач при BPR

Следует отметить, что многие предприятия, осуществляющие BPR, не добиваются в нем успеха. По оценкам Хаммера и Чампи, от 50 до 70% всех осуществляющих BPR предприятий не достигают тех результатов, на которые они рассчитывали. Однако, BPR нельзя назвать и рискованным предприятием. Сравните, например, риск при игре в рулетку и в шахматы. Рулетка - игра с высоким уровнем риска, а шахматы - нет; однако человек может проигрывать в шахматы не реже, чем в рулетку. Исход игры в рулетку решает случай; в шахматах, наоборот, случай не влияет на исход игры. Тот, кто играет лучше, может рассчитывать на победу, а проигрыш является результатом недостаточного умения и недостаточно хорошо продуманной стратегии. То же справедливо и по отношению к BPR. Ключ к успеху - не в везении, а в знании и способностях. Тем не менее при BPR постоянно допускаются одни и те же ошибки.

1) Попытка зафиксировать существующий процесс. Самая грубая ошибка при BPR - это когда никакого BPR вообще не происходит, а под этим названием понимаются незначительные изменения в процессе. В последнее время термин “business process reengineering” используется по отношению к самым разным программам, которые ничего общего не имеют с радикальным перепланированием процесса. Если повесить на корову плакат с надписью “Я лошадь”, то от этого корова не станет лошадью.

Предприятия часто идут на огромные жертвы, чтобы избежать радикальной перепланировки, связанной с BPR. Они могут провести реорганизацию (изменяется не процесс выполнения работы, а всего лишь административные рамки, в которых находятся сотрудники), сокращение (использование меньшего числа сотрудников для выполнения той же самой работы, иногда в меньших размерах), другие мероприятия, которые различными способами побуждают людей

работать более интенсивно. Существующие процессы, даже в том случае, если они являются источником проблем для предприятия, все же хорошо знакомы и привычны; инфраструктура, нужная для организации этих процессов, уже имеется. Кажется гораздо проще и разумнее улучшить эти процессы, а не отбросить их и начать с нуля. Путь наименьшего сопротивления, по которому идут большинство предприятий - внесение небольших изменений. Это также самый верный путь к провалу при проведении BPR.

2) Внимание не фокусируется на бизнес-процессах. Одна из причин неудач BPR заключается в неправильно поставленной задаче. Такие цели как расширение полномочий, повышение эффективности работы в группе, улучшение обслуживания клиентов и т.д. являются абстрактными понятиями, это желательные для предприятия характеристики или атрибуты, при этом не существует какого-то четкого пути к их достижению. Они являются последствиями планирования процессов и достигнуты могут быть только в таком контексте. Как можно расширить чьи-то полномочия, если не в рамках организации рабочих процессов?

3) Игнорируется все, кроме перепланирования процесса. BPR приводит к изменениям разного свойства: организация рабочего места, организационные структуры, управление - все, связанное с процессом, необходимо заново отшлифовать для того, чтобы получить искомый алмаз.

Когда корпорация Ford перепроектировала процесс расплаты с поставщиками, это затронуло даже складских работников, которые стали людьми, ответственными за принятие решений. Теперь вместо того, чтобы ставить на бумагу штамп с датой и временем, им приходилось пользоваться компьютером для выяснения соответствия ли полученного груза и заказа. Если груз не соответствовал заказу, их обязанностью было отказаться от него и отправить его обратно. Люди, на которых раньше не лежало практически никакой ответственности ни за что, теперь должны были думать и принимать решения.

Даже те руководители, которые стремятся к радикальному перепланированию процесса, часто бывают напуганы размахом изменений, вызванных этим перепланированием. Обычно события развиваются по такому сценарию: руководитель нанимает группу проведения BPR для того, чтобы резко улучшить плохо работающий процесс. Некоторое время спустя группа приходит к нему со своей идеей прорыва и показывает, каким образом при ее внедрении можно сократить рабочий цикл на 90%, затраты на 95%, а число ошибок - на 99%. Руководитель вне себя от радости. После этого группа объясняет, что перепроектированный процесс потребует новой системы оплаты труда, слияния ряда подразделений, снижения роли руководства и нового стиля трудовых отношений. Менеджер снова вне себя, но уже не от радости: “Я просил вас снизить затраты и число ошибок, а не переделывать предприятие”. Затем группа, как правило, распускается, и больше никто никогда не слышит о ее идее прорыва. Но BPR и заключается именно в том, чтобы переделать предприятие.

4) Не принимаются во внимание ценности и убеждения людей. Недостаточно просто запустить новые процессы, людям нужна какая-то причина, по которой они будут хорошо работать в их рамках, поскольку перемены, связанные с изменением отношений, принимаются нелегко. Руководство должно создать у сотрудников мотивы, достаточные для участия в этих процессах, оказывая поддержку новым ценностям и убеждениям, которых требуют эти процессы. Другими словами, руководство должно интересоваться не только тем, что происходит у людей на столах, но в равной степени и тем, что происходит у них в головах.

5) Предпочтительность незначительных результатов. Искушение выбрать легкий путь и остановиться на небольших улучшениях велико. В перспективе, однако, оказывается, что небольшие улучшения приносят только вред, они усложняют существующий процесс, вследствие чего становится труднее выяснить, как же он функционирует на самом деле. Более того, дополнительные вложения времени и денег в существующий процесс усиливают консерватизм руководства по отношению к позиции “отбросить старое и начать заново”.

6) Жесткие ограничения при постановке задачи. Попытка проведения BPR обречена на неудачу, когда руководство еще до начала самого процесса жестко ограничивает круг решаемых

проблем или масштаб проведения BPR. Определение проблемы и масштаба - это шаги, которые выполняются в рамках BPR, начинающегося с задания целей, которые должны быть достигнуты, а не путей достижения этих целей.

7) Попытки начать BPR снизу. BPR никогда не начинается снизу. Существует две причины, по которым работники низшего звена и руководители среднего звена не могут инициировать и реализовать BPR. Первая причина заключается в том, что работники низшего звена не могут видеть полной картины того, что требуется при BPR. Их опыт ограничивается их должностными обязанностями и теми подразделениями, в которых они работают. Во-вторых, любой бизнес-процесс выходит за границы предприятия, поэтому ни один руководитель среднего звена не обладает достаточными полномочиями, чтобы настаивать на изменении такого процесса. Более того, некоторые из руководителей такого ранга опасаются, что радикальные изменения существующего процесса приведут к снижению их собственной власти, влияния, авторитета. В случае, если радикальные изменения начнутся снизу, они могут оказать сопротивление и задушить их. Только сильное руководство сверху приведет к тому, что эти люди примут перемены, связанные с BPR.

8) Недостаток ресурсов на проведение BPR. Предприятие не сможет добиться успеха при проведении BPR, если не будет вкладывать средства в этот процесс. При этом наиболее важная составляющая этих вложений - время и внимание лучших сотрудников, включая личное и непосредственное участие руководства верхнего звена. От руководства не требуется самим проводить все работы по BPR, но ответственность за весь процесс нельзя перекладывать ни на кого. Урезание ресурсов на BPR означает также и то, что руководство не считает эту работу такой уж важной, и тем самым толкает сотрудников на уклонение от этой работы или сопротивление ей.

9) Попытка провести BPR, чтобы никого не обидеть. Для BPR очень верна поговорка: “Чтобы сделать яичницу, нужно разбить яйца”. От результатов BPR выигрывают не все. Одни потеряют работу, другие будут недовольны той работой, которую они будут выполнять после BPR. Желание угодить всем приведет к тому, что BPR сведется к незначительным изменениям или его реализация будет отложена на потом. Поэтому сопротивление является неизбежной реакцией на серьезные изменения, к которым приводит BPR. Руководство должно ожидать этого сопротивления и не позволить завалить все дело.

Заметим, что все рассмотренные ошибки (а безусловно, существуют и другие) имеют одну общую характеристику. Они связаны с ролью, которую играет руководство верхнего звена. Если попытка проведения BPR провалилась, то глубинная причина провала в том, что руководство недостаточно поддерживало процесс.

ГЛАВА 20

МЕТОДЫ ОЦЕНКИ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ

Среди большого числа методов оценки деятельности предприятий наибольшее распространение (по крайней мере в отечественных консалтинговых проектах) получили следующие два:

- метод динамического функционального анализа на основе сетей Петри различного вида;
- метод функционально-стоимостного анализа ABC.

Каждый из этих методов (и соответствующих поддерживающих инструментальных средств) регламентирует следующие основные этапы выполнения оценок:

- построение статической функциональной модели (с использованием SADT или DFD-нотации);
- расширение статической модели соответственно поведенческими или стоимостными характеристиками ее объектов;
- сбор и ввод в модель необходимой фактической информации;
- “исполнение” модели и получение соответствующих оценок.

20.1. Динамическое моделирование с использованием сетей Петри

Сеть Петри представляет собой ориентированный граф с вершинами двух типов (позициями и переходами), в котором дугами могут соединяться только вершины различных типов. В позиции сети помещаются специальные маркеры (“фишки”), перемещение которых и отображает динамику моделируемой системы. Изменение маркировки (движение маркеров) происходит в результате выполнения (срабатывания) перехода на основе соответствующего внешнего события. Точнее, переход срабатывает, если во всех его входных позициях имеются маркеры и происходит соответствующее переходу событие. При этом из каждой входной позиции срабатываемого перехода маркер удаляется, а в каждую выходную позицию - заносится.

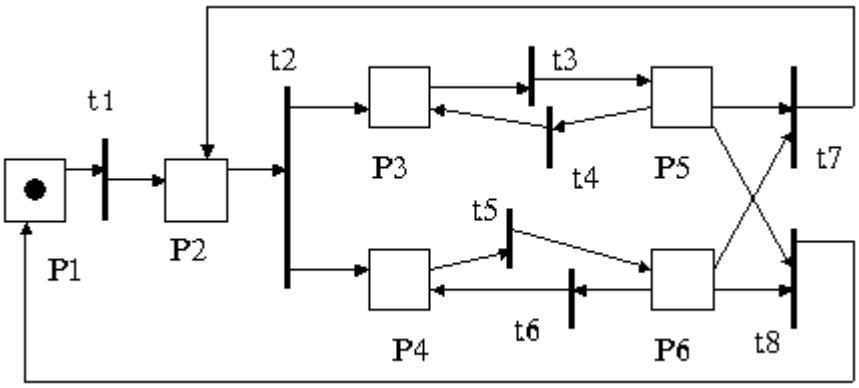


Рис. 20.1. Пример сети Петри

На рис.20.1 приведен пример сети Петри с позициями P1-P6 и переходами t1-t8. Единственный маркер находится в позиции P1, все остальные позиции пусты. При срабатывании перехода t1 маркер переносится из позиции P1 в позицию P2, при срабатывании перехода t2 маркер переносится из позиции P2 в позиции P3 и P4 и т.д.

Фактически сеть Петри декомпозирует систему на активные (переходы) и пассивные (позиции - хранилища маркеров) элементы. Следует отметить, что рассмотренные ранее диаграммы переходов состояний являются вырожденными сетями Петри, а именно, сетями с одним типом вершин (переходами).

На практике обычно применяются более сложные и развитые сети Петри. Модификации, как правило, касаются следующих трех моментов:

- введение иерархии (иерархические сети Петри);
- определение различий в маркерах, каждый из которых имеет свои уникальные характеристики (цветные/раскрашенные сети Петри);
- введение многоместных (содержащих несколько маркеров) позиций, как последовательных, так и параллельных (сети Петри с многоместными позициями).

Последнее вносит в работу сети специфику, характеризуемую правилами срабатывания переходов. Последовательная позиция соответствует дисциплине FIFO (first in - first out): входящий маркер ставится в конец очереди, выходящий берется из ее начала. Поэтому срабатывание перехода обуславливается характеристиками начального маркера - если эти характеристики являются неблагоприятными, то переход блокируется и функционирование сети прекращается. Из параллельной позиции может выйти любой из находящихся в ней маркеров, удовлетворяющий условию срабатывания перехода (при этом для избежания конфликтов маркерам присваиваются приоритеты).

В консалтинговых проектах динамическое моделирование с использованием сетей Петри осуществляется на основании статической функциональной и частично информационной моделей. Соответствующие инструментальные средства (например, Design/CPN для SADT и CPN-AMI,

INCOME для DFD) осуществляют автоматическое преобразование функциональных моделей в прообразы сетей Петри, которые затем дорабатываются вручную. Такое преобразование базируется на том, что маркер моделирует порцию потока данных, а позиция - накопление и хранение таких порций. Каждая из диаграмм функциональной модели трансформируется в соответствующую компоненту (подсеть) иерархической сети Петри. При этом процессы и потоки DFD-диаграммы (активности и потоки SADT-диаграммы) отображаются, соответственно, переходами и позициями. Хранилища данных и внешние сущности также преобразуются в позиции для каждого входящего/исходящего потока (при этом для внешних сущностей маркируются позиции, соответствующие исходящим из них потокам). На основе информационной модели определяются правила срабатывания переходов в зависимости от значений, которые принимают атрибуты используемых сущностей.

- С использованием динамической модели подобного типа можно описать и проанализировать:
- механизмы взаимодействия процессов (последовательность, параллелизм, альтернатива)
- временные отношения между выполнениями процессов (одновременность, наложение, поглощение, одинаковое время запуска/завершения и т.п.);
- абсолютные времена (длительность процесса, время запуска, зависимости от времени выполнения процесса и др.);
- управление исключительными ситуациями, определяемое нарушениями.

Построенные динамические модели позволяют осуществлять следующие операции:

- статический анализ системы (компоненты сети, иерархия сети, соответствие типов);
- динамический анализ системы для конкретного маркирования сети;
- имитационное моделирование системы с построением графиков движения маркеров относительно позиций сети в системном времени, определяемом моментами срабатывания переходов, и в реальном времени путем задания для переходов задержек времени, отображающих продолжительность реальных операций.

20.2. ABC - метод функционально-стоимостного анализа

ABC (Activity Based Costing) - метод определения стоимости и других характеристик товаров и услуг на базе функций и ресурсов, задействованных во всех деятельностях предприятия (производстве, маркетинге, обслуживании клиентов, оказании услуг, технической поддержке и т.п.). Он был разработан как “операционно-ориентированная” альтернатива традиционным подходам, основанным на использовании прямых затрат труда и материалов как основы для вычисления накладных расходов. ABC-метод рассматривает деятельность предприятия как множество последовательно выполняемых процессов/функций (в том числе и косвенных, вносящих большой вклад в формирование стоимости), распределяя при этом накладные расходы в соответствии с детальными расчетами использования ресурсов, подробными моделями процессов и их влиянием на себестоимость.

Определение стоимости производится в два этапа:

- определение затрат на выполнение функций на основе необходимых для этого ресурсов, включающих прямые затраты материалов и труда, косвенные затраты труда и накладные расходы
- определение затрат на стоимостные объекты (товары, услуги, обслуживание клиентов) на основе используемых ими функций.

Фактически ABC-модель содержит три взаимоувязанных модуля:

- модуль ресурсов, моделирующий все необходимые для деятельности предприятия ресурсы в денежном выражении - затраты на аренду помещений, оборудование, оплату труда, сырье и материалы и т.п.

- модуль функций, составляющих в совокупности деятельность предприятия (представляющий собой иерархическую функциональную модель предприятия, обеспечивающую как представление обобщенной картины его деятельности, так и доступ к детализированным процессам нижних уровней)
- модуль стоимостных объектов, моделирующий результаты деятельности предприятия, на которые в конечном счете и расходуются средства.

Разработка АВС-модели включает следующие этапы:

- выявление требуемых ресурсов
- выявление стоимостных объектов
- определение функций
- определение факторов ресурсов - показателей, применяемых для установления взаимосвязей между модулями ресурсов и функций
- определение стоимости функций
- отбор функциональных факторов - показателей, применяемых для установления взаимосвязей между модулями функций и стоимостных объектов.

Задача определения функций заключается в построении функциональной модели деятельности предприятия и решается с использованием методов структурного системного анализа, поддерживающих иерархии SADT или DFD-диаграмм.

На следующем этапе осуществляется связывание модулей ресурсов и функций за счет присваивания каждой функции факторов ресурсов, характеризующих потребление ресурсов функцией. Например, потребление функцией Ремонт ресурса *Затраты на аренду помещения* может определяться на основе фактора ресурсов *Занимаемая площадь*, значение которого представляет собой размер площади, занимаемой под ремонтные мастерские.

Вычисление итоговой стоимости функций целесообразно осуществлять путем восходящего суммирования: сначала необходимо определить стоимость выполнения элементарных функций на нижнем уровне иерархии, а затем последовательно суммировать стоимость выполнения функций снизу вверх по всем уровням модели.

Целью следующего этапа является выбор функциональных факторов, определяющих стоимость товаров и услуг. При этом значение каждого функционального фактора должно определть долю стоимости данной функции в каждом стоимостном объекте. Например, стоимость функции *Тиражирование рекламных материалов* распределяется по стоимостным объектам (рекламируемым товарам) пропорционально количеству страниц в этих материалах.

После построения АВС-модели необходимо ввести конкретные числовые значения, характеризующие величины выбранных параметров (значения затрат, ресурсов и факторов), после этого ее можно использовать для анализа и принятия решений.

Следует отметить, что АВС-модель лишь обеспечивает получение важной для бизнес-процесса информации, содержащей стоимостную картину деятельности и характеризующей ее эффективность и прибыльность товаров и услуг. Для дальнейшего ее анализа и основанного на нем управления предприятием применяется методика **ABM (Activity Based Management)**, регламентирующая средства и способы управления с целью совершенствования бизнес-процессов и повышения прибыльности. Фактически АВМ представляет собой комплекс методов анализа АВС-модели для реорганизации бизнес-процессов с целью повышения производительности, снижения стоимости и улучшения качества:

- стратегический анализ, облегчающий выбор наилучшей стратегии и определение наиболее прибыльного пути достижения стратегических целей (включая ценообразование, определение ассортимента товаров и услуг, анализ прибыльности клиентов, изучение конкурентов, определение компромисса между производством деталей и получением их от поставщика);

- стоимостной анализ, облегчающий поиск возможностей снижения стоимости, а также обеспечивающий прогнозирование результатов модификаций и моделирование последствий конкретного решения;
- определение целевой стоимости, помогающее планировать выпуск товаров и оказание услуг с заданной стоимостью;
- исчисление стоимости исходя из жизненного цикла, определяющее совокупные затраты на выпуск товара для облегчения оценки его стоимости и прибыльности (при планировании на период такая оценки не может быть сделана).

ПРИЛОЖЕНИЕ 1

Внедрение структурного подхода и выбор CASE-средств

Внедрение структурного подхода, как и любое нововведение, вызывает естественные проблемы, связанные с такими человеческими качествами, как боязнь за свою карьеру в случае неудачи, инертность, нежелание перестраиваться и отказываться от старых привычек, с одной стороны, и шапкозакидательство, с другой. Поэтому критичным является первый опыт использования структурного анализа и проектирования, в успехе которого решающую роль играет выбор хорошего опытного проекта. Ниже приведен ряд рекомендаций для подбора такого проекта:

- Выбирайте небольшой, достаточно простой проект. Идеальным кандидатом для опытного проекта является проект продолжительностью до полугода (включая обучение), требующий для реализации около четырех человеко-лет. Такой проект имеет достаточную продолжительность, чтобы быть заслуживающим внимания руководства, но достаточно коротким для обеспечения приемлемой по времени обратной связи.
- Выбирайте проект, который не является критичным для Вашей организации. Прессинг руководства и ответственность вероятнее всего заставит аналитиков и проектировщиков вернуться к старым методам и привычкам, особенно если дела пойдут неудачно.
- Не жалейте денег на обучение, но и не направляйте Ваших специалистов на многочисленные одновременные курсы: аналитик предметной области, системный аналитик и проектировщик - это разные профессии, освоить их одновременно невозможно.
- Выбирайте обычных специалистов для опытного проекта. Если Вы отберете группу, включающую только наиболее квалифицированных специалистов, руководство сочтет, что структурные методы по силам только суперштату, и, следовательно, доверие к таким методам будет незначительным.
- Выбирайте ответственных людей, не опускающих рук при первом же препятствии. Обязательность и дисциплина являются двумя жизненно важными факторами успеха для любых методов программной инженерии.
- Пригласите опытного консультанта для контроля двух ключевых моментов в проекте: на этапе начала его создания и на этапе завершения перед его проверкой. Одна человеко-неделя консультирования может сохранить несколько человеко-месяцев усилий по лечению.
- Тщательно выбирайте инструментальные средства.

На выбор CASE-пакетов для выполнения консалтинговых проектов влияет огромное количество факторов. Ниже приводится ряд рекомендаций, помогающих обойти лишь некоторые из подводных камней, неизбежно возникающих при переходе к новым технологиям.

1. Поддержка методологий структурного (а не объектно-ориентированного) анализа и проектирования на начальных этапах проекта. Если Вы при общении с руководством или экспертом предметной области (например, с бухгалтером) будете употреблять слова “наследование”, ”инкапсуляция”, ”полиморфизм” и т.п., то в лучшем случае столкнетесь с непониманием.
2. Поддержка классических методов структурного анализа и проектирования. Это позволит Вам в случае неудовлетворенности пакетом относительно легко подобрать новый, не переделывая, а лишь перерисовывая (в худшем случае) наработанные модели.

3. Выбор в качестве первого опыта недорогих продуктов с учетом информации о реальных проектах, выполненных с их использованием. Например, известен ряд фирм и банков, использующих на начальных этапах проектирования автоматизированных банковских систем пакеты CASE.Аналитик для построения функциональной, а ERwin для построения информационной моделей.
4. Наличие средств экспорта/импорта фрагментов проекта, что при коллективной работе поможет избежать множества проблем, связанных с мультипользовательским доступом.
5. Обязательная поддержка автоматической верификации на полноту и состоятельность проекта и генерации отчетов по верификации.
6. Автоматическая генерация проектной документации в соответствии с общепринятыми стандартами (отечественных заказчиков вполне удовлетворяют ГОСТы, зарубежных - DOD STD-2167A).
7. Для функционального моделирования - наличие миниспецификаций процессов нижнего уровня (задаваемых общепринятыми методами), а не возможности задавать аналогичную информацию в качестве комментария при определении процессов. Это позволит полностью охватить технологии, применяемые заказчиком, и расширит возможности созданного проекта (например, его можно будет использовать для автоматизированного и быстрого обучения новых работников конкретному направлению деятельности).
8. Для информационного моделирования - наличие средств генерации схем БД для широкого спектра СУБД, а также поддержки обратного проектирования (reverse engineering), т.е. создания информационных моделей из существующих БД.

ПРИЛОЖЕНИЕ 2

Системы класса MRP

В конце 60-х годов Американским обществом управления производством и запасами (APICS), были сформулированы принципы управления предприятием, которые легли в основу концепции MRP (Material Requirement Planning), в настоящее время ставшей стандартом де-факто:

- производственная деятельность описывается как поток взаимосвязанных заказов;
- при выполнении заказов учитываются ограничения ресурсов;
- обеспечивается минимизация производственных циклов и запасов;
- заказы снабжения и производства формируются на основе заказов реализации и производственных графиков;
- движение заказов увязывается с экономическими показателями;
- выполнение заказа завершается к тому моменту, когда он необходим.

В последующие годы обновление концепции осуществлялось по пути расширения функциональных возможностей, соответствующих растущим потребностям предприятий по обслуживанию клиентов, и включало следующие этапы:

- планирование потребностей в материалах на основе данных о составе изделий и складских запасов (MRP);
- формирование производственной программы в масштабах всего предприятия и контроль ее выполнения на уровне подразделений (Closed Loop MRP);
- прогнозирование, планирование и контроль производства по всему циклу, начиная от закупки сырья и заканчивая отгрузкой товара потребителю (MRP II - Manufacturing Resource Planning);
- планирование потребностей в распределении и ресурсах при наличии у предприятия территориально-распределенной структуры и получение окончательного итога процесса моделирования сбытового и производственного планов в денежном выражении (MRP III - Money Resource Planning / ERP - Enterprise Resource Planning).

Современные автоматизированные системы управления предприятием являются программными реализациями перечисленных концепций и предназначены, прежде всего, для обеспечения

руководства предприятия средствами управления производством. Ниже перечислены основные функции таких систем.

1. Планирование, включающее прогнозирование, производственное и объемно-календарное планирование, планирование потребностей в материалах и производственных мощностях.
2. Производство, включая ведение основных производственных данных по маршрутизации и составу изделия, оперативное управление производством, учет рабочего времени, расчет и контроль себестоимости продукции.
3. Управление запасами, и том числе управление данными по запасам, статистическое управление запасами, управление размещением запасов и планирование их распределения, управление партиями.
4. Управление сбытом, включая предложения и контракты на продажу, управление заказами на продажу, статистику по реализации, маркетинговую информацию.
5. Управление снабжением, включающее ведение предложений и контрактов, управление заказами и статистику.
6. Управление финансами, в том числе расчеты с заказчиками и поставщиками, главная книга, управление затратными центрами, субучет, финансовая отчетность и сводный баланс, учет основных средств.
7. Техническое обслуживание, включающее организацию техобслуживания и ремонта, обработку заявок, управление возвращенной продукцией.
8. Управление проектами, в том числе финансовая подготовка проекта (сметы), планирование проекта, конфигурация нестандартных изделий.
9. Проектно-конструкторские работы, включая управление чертежами, классификацию продукции, конструкторские спецификации и связь с САПР.

Примерами западных систем рассматриваемого класса являются R3 (продукт германской фирмы SAP AG), CA-MK/X (пакет американской фирмы Computer Associates), BAAN IV (продукт голландской фирмы Baan Int), IFS (пакет шведской фирмы IFS AB). Отечественные разработки значительно уступают перечисленным продуктам прежде всего по функциональности и степени интеграции модулей в единое информационное пространство. Кроме того, в отечественных продуктах отсутствуют встроенные средства наглядного описания бизнес-процессов (фактически, функционального и информационного моделирования).

С другой стороны, отечественные системы ориентированы на структуры российских предприятий и гораздо более адаптивны к непрерывным изменениям в налоговой и правовой сфере.

ПРИЛОЖЕНИЕ 3

Выдержки из 34 комплекса государственных стандартов на автоматизированные системы

П3.1. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания

Настоящий стандарт распространяется на автоматизированные системы (АС), используемые в различных видах деятельности (исследование, проектирование, управление и т.п.), включая их сочетания, создаваемые в организациях, объединениях и на предприятиях (далее организациях). Стандарт устанавливает стадии и этапы создания АС, а также содержание работ на каждом этапе.

Процесс создания АС представляет собой совокупность упорядоченных во времени, взаимосвязанных, объединенных в стадии и этапы работ, выполнение которых необходимо и достаточно для создания АС, соответствующей заданным требованиям. Стадии и этапы создания АС в общем случае приведены ниже.

1. Формирование требований к АС
1.1. Обследование объекта и обоснование необходимости создания АС
1.2. Формирование требований пользователя к АС
1.3. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания)

2. Разработка концепции АС
2.1. Изучение объекта
2.2. Проведение необходимых научно-исследовательских работ
2.3. Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователя
2.4. Оформление отчета о выполненной работе
3. Техническое задание
3.1. Разработка и утверждение технического задания на создание АС
4. Эскизный проект
4.1. Разработка предварительных проектных решений по системе и ее частям
4.2. Разработка документации на АС и ее части
5. Технический проект
5.1. Разработка проектных решений по системе и ее частям
5.2. Разработка документации на АС и ее части
5.3. Разработка и оформление документации на поставку изделий для комплектования АС и/или технических требований (технических заданий) на их разработку
5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации
6. Рабочая документация
6.1. Разработка рабочей документации на систему и ее части
6.2. Разработка или адаптация программ
7. Ввод в действие
7.1. Подготовка объекта автоматизации к вводу АС в действие
7.2. Подготовка персонала
7.3. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями)
7.4. Строительно-монтажные работы
7.5. Пусконаладочные работы
7.6. Проведение предварительных испытаний
7.7. Проведение опытной эксплуатации
7.8. Проведение приемочных испытаний
8. Сопровождение АС
8.1. Выполнение работ в соответствии с гарантийными обязательствами
8.2. Послегарантийное обслуживание

Допускается исключать стадию "Эскизный проект" и отдельные этапы работ на всех стадиях, объединять стадии "Технический проект" и "Рабочая документация" в одну стадию "Технорабочий проект". В зависимости от специфики создаваемых АС и условий их создания допускается выполнять отдельные этапы работ до завершения предшествующих стадий, параллельное во времени выполнение этапов работ, включение новых этапов работ.

На этапе 1.1 в общем случае проводят сбор данных об объекте автоматизации и осуществляемых видах деятельности; оценку качества функционирования объекта и осуществляемых видов деятельности, выявление проблем, решение которых возможно средствами автоматизации; оценку (технико-экономической, социальной и т. п.) целесообразности создания АС.

На этапе 1.2 проводят подготовку исходных данных для формирования требований к АС (характеристика объекта автоматизации, описание требований к системе, ограничения допустимых затрат на разработку, ввод в действие и эксплуатацию, эффект, ожидаемый от системы, условия создания и функционирования системы); формулировку и оформление требований пользователя к АС.

На этапе 1.3 проводят оформление отчета о выполненных работах на данной стадии и оформление заявки на разработку АС (тактико-технического задания) или другого замещающего ее документа с аналогичным содержанием.

На этапах 2.1 и 2.2 организация-разработчик проводит детальное изучение объекта автоматизации и необходимые научно-исследовательские работы (НИР), связанные с поиском путей и оценкой возможности реализации требований пользователя, оформляют и утверждают отчеты о НИР.

На этапе 2.3 в общем случае проводят разработку альтернативных вариантов концепции создаваемой АС и планов их реализации; оценку необходимых ресурсов на их реализацию и обеспечение функционирования; оценку преимуществ и недостатков каждого варианта; сопоставление требований пользователя и характеристик предлагаемой системы и выбор оптимального варианта; определение порядка оценки качества и условий приемки системы; оценку эффектов, получаемых от системы.

На этапе 2.4 подготавливают и оформляют отчет, содержащий описание выполненных работ на стадии, описание и обоснование предлагаемого варианта концепции системы.

На этапе 3.1 проводят разработку, оформление, согласование и утверждение технического задания на АС и, при необходимости, технических заданий на части АС.

На этапе 4.1 определяются функции АС; функции подсистем, их цели и эффекты; состав комплексов задач и отдельных задач; концепции информационной базы, ее укрупненная структура; функции системы управления базой данных; состав вычислительной системы; функции и параметры основных программных средств.

На этапе 5.1 обеспечивают разработку общих решений по системе и ее частям, функционально-алгоритмической структуре системы, по функциям персонала и организационной структуре, по структуре технических средств, по алгоритмам решений задач и применяемым языкам, по организации и ведению информационной базы, системе классификации и кодирования информации, по программному обеспечению.

На этапах 4.2 и 5.2 проводят разработку, оформление, согласование и утверждение документации в объеме, необходимом для описания полной совокупности принятых проектных решений и достаточном для дальнейшего выполнения работ по созданию АС.

На этапе 6.1 осуществляют разработку рабочей документации, содержащей все необходимые и достаточные сведения для обеспечения выполнения работ по вводу АС в действие и ее эксплуатации, а также для поддержания уровня эксплуатационных характеристик (качества) системы в соответствии с принятыми проектными решениями, ее оформление, согласование и утверждение.

На этапе 6.2 проводят разработку программ и программных средств системы, выбор, адаптацию и/или привязку приобретаемых программных средств, разработку программной документации.

На этапе 7.1 проводят работы по организационной подготовке объекта автоматизации к вводу АС в действие, в том числе: реализацию проектных решений по организационной структуре АС; обеспечение подразделений объекта управления инструктивно-методическими материалами; внедрение классификаторов информации.

На этапе 7.2 проводят обучение персонала и проверку его способности обеспечить функционирование АС.

На этапе 7.6 осуществляют испытания АС на работоспособность и соответствие техническому заданию в соответствии с программой и методикой предварительных испытаний; устранение неисправностей и внесение изменений в документацию на АС, в том числе эксплуатационную в соответствии с протоколом испытаний; оформление акта о приемке АС в опытную эксплуатацию.

На этапе 7.7 проводят опытную эксплуатацию АС; анализ результатов опытной эксплуатации АС; доработку (при необходимости) программного обеспечения АС; дополнительную наладку (при необходимости) технических средств АС; оформление акта о завершении опытной эксплуатации.

На этапе 7.8 проводят испытание на соответствие техническому заданию в соответствии с программой и методикой приемочных испытаний; анализ результатов испытаний АС и устранение недостатков, выявленных при испытаниях; оформление акта о приемке АС в постоянную эксплуатацию.

На этапе 8.1 осуществляют работы по устранению недостатков, выявленных при эксплуатации АС в течение установленных гарантийных сроков, внесению необходимых изменений в документацию на АС.

На этапе 8.2 осуществляют работы по анализу функционирования системы; выявлению отклонений фактических эксплуатационных характеристик АС от проектных значений; установлению причин этих отклонений; устранению выявленных недостатков и обеспечению стабильности эксплуатационных характеристик АС; внесению необходимых изменений в документацию на АС.

П3.2. РД 50-34.698-90. Автоматизированные системы. Требования к содержанию документов

Настоящие методические указания распространяются на автоматизированные системы (АС), используемые в различных сферах деятельности (управление, исследование, проектирование и т.п.), включая их сочетание, и устанавливают требования к содержанию документов, разрабатываемых при создании АС.

Содержание документов является общим для всех видов АС и, при необходимости, может дополняться разработчиком документов в зависимости от особенностей создаваемой АС. Допускается включать в документы дополнительные разделы и сведения, объединять и исключать разделы. Содержание каждого документа определяет разработчик в зависимости от объекта проектирования (системы, подсистема и т.д.).

1. **Пояснительные записки** к эскизному, техническому проектам содержат разделы: общие положения; описание процесса деятельности; основные технические решения; мероприятия по подготовке объекта автоматизации к вводу системы в действие.

В разделе "Общие положения" приводят:

- наименование проектируемой АС и наименования документов, их номера и дату утверждения, на основании которых ведут проектирование АС;
- перечень организаций, участвующих в разработке системы, сроки выполнения стадий;
- цели, назначение и области использования АС;
- подтверждение соответствия проектных решений действующим нормам и правилам техники безопасности, пожаро- и взрывобезопасности и т.п.;
- сведения об использованных при проектировании нормативно-технических документах;
- сведения о НИР, передовом опыте, изобретениях, использованных при разработке проекта;
- очередность создания системы и объем каждой очереди.

В разделе "Описание процесса деятельности"

- отражают состав процедур (операций) с учетом обеспечения взаимосвязи и совместимости процессов автоматизированной и неавтоматизированной деятельности;
- формируют требования к организации работ в условиях функционирования АС.

В разделе "Основные технические решения" приводят:

- решения по структуре системы, подсистем, средствам и способам связи для информационного обмена между компонентами системы, подсистем;
- решения по взаимосвязям АС со смежными системами, обеспечению ее совместимости;
- решения по режимам функционирования, диагностированию работы системы;
- решения по численности, квалификации и функциям персонала АС, режимам его работы, порядку взаимодействия;
- сведения об обеспечении заданных в техническом задании (ТЗ) потребительских характеристик системы (подсистем), определяющих ее качество;
- состав функций, комплексов задач, реализуемых системой (подсистемой);
- решения по комплексу технических средств, его размещению на объекте;
- решения по составу информации, объему, способам ее организации, видам машинных носителей, входным и выходным документам и сообщениям, последовательности обработки информации и другим компонентам;
- решения по составу программных средств, языкам деятельности, алгоритмам процедур и операций и методам их реализации.

В разделе "Мероприятия по подготовке объекта автоматизации к вводу системы в действие" приводят:

- мероприятия по приведению информации к виду, пригодному для обработки на ЭВМ;
- мероприятия по обучению и проверке квалификации персонала;
- мероприятия по созданию необходимых подразделений и рабочих мест;
- мероприятия по изменению объекта автоматизации;
- другие мероприятия, исходящие из специфических особенностей создаваемых АС.

2. **Схема функциональной** структуры содержит:

- элементы функциональной структуры АС (подсистемы АС); автоматизированные функции и/или задачи (комплексы задач); совокупности действий (операций), выполняемых при реализации автоматизированных функций только техническими средствами (автоматически) или только человеком;
- информационные связи между элементами и с внешней средой с кратким указанием содержания сообщений и/или сигналов, передаваемых по связям, и при необходимости, связи других типов (входимости, подчинения и т. д.);
- детализированные схемы частей функциональной структуры (при необходимости).

3. **Описание автоматизируемых функций** содержит разделы: исходные данные; цели АС и автоматизированные функции; характеристика функциональной структуры; типовые решения (при наличии).

В разделе "Исходные данные" приводят:

- перечень исходных материалов и документов, использованных при разработке функциональной части проекта АС;
- особенности объекта управления, влияющие на проектные решения по автоматизированным функциям;
- данные о системах управления, взаимосвязанных с разрабатываемой АС, и сведения об информации, которой она должна обмениваться с абонентами и другими системами;
- описание информационной модели объекта вместе с его системой управления.

В разделе "Цели АС и автоматизированные функции" приводят описание автоматизированных функций, направленных на достижение установленных целей.

Раздел "Характеристика функциональной структуры" содержит:

- перечень подсистем АС с указанием функций и/или задач, реализуемых в каждой подсистеме;

- описание процесса выполнения функций (при необходимости);
- необходимые пояснения к разделению автоматизированных функций на действия (операции), выполняемые техническими средствами и человеком;
- требования к временному регламенту и характеристикам процесса реализации автоматизированных функций (точности, надежности и т.п.) и решения задач.

В разделе "Типовые решения" приводят перечень типовых решений с указанием функций, задач, комплексов задач, для выполнения которых они применены.

4. **Описание постановки задачи** (комплекса задач) содержит разделы: характеристики комплекса задач; выходная информация; входная информация.

В разделе "Характеристики комплекса задач" приводят:

- назначение комплекса задач;
- перечень объектов (технологических объектов управления, подразделений предприятия и т. п.), при управлении которыми решают комплекс задач;
- периодичность и продолжительность решения;
- условия, при которых прекращается решение комплекса задач автоматизированным способом (при необходимости);
- связи данного комплекса задач с другими комплексами (задачами) АС;
- должности лиц и/или наименования подразделений, определяющих условия и временные характеристики конкретного решения задачи (если они не определены общим алгоритмом функционирования системы);
- распределение действий между персоналом и техническими средствами при различных ситуациях решения комплекса задач.

Раздел "Выходная информация" содержит:

- перечень и описание выходных сообщений;
- перечень и описание имеющих самостоятельное смысловое значение структурных единиц информации выходных сообщений (показателей, реквизитов и их совокупностей, сигналов управления) или ссылку на документы, содержащие эти данные.

В описании по каждому выходному сообщению следует указывать:

- идентификатор;
- форму представления сообщения (документ, видеокадр, сигнал управления) и требования к ней;
- периодичность выдачи;
- сроки выдачи и допустимое время задержки решения;
- получателей и назначение выходной информации.

В описании по каждой структурной единице информации следует указывать:

- наименование;
- идентификатор выходного сообщения, содержащего структурную единицу информации;
- требования к точности и надежности вычисления (при необходимости).

Раздел "Входная информация" должен содержать:

- перечень и описание входных сообщений (идентификатор, форму представления, сроки и частоту поступления);
- перечень и описание структурных единиц информации входных сообщений или ссылку на документы, содержащие эти данные.

В описании по каждой структурной единице информации входных сообщений следует указывать:

- наименование;
- требуемую точность ее числового значения (при необходимости);
- источник информации (документ, видеокадр, устройство, кодограмма, информационная база на машинных носителях и т. д.);
- идентификатор источника информации.

5. **Общее описание системы** содержит разделы: назначение системы; описание системы; описание взаимосвязей АС с другими системами; описание подсистем (при необходимости).

В разделе "Назначение системы" указывают:

- вид деятельности, для автоматизации которой предназначена система;
- перечень объектов автоматизации, на которых используется система;
- перечень функций, реализуемых системой.

В разделе "Описание системы" указывают:

- структуру системы и назначение ее частей;
- сведения об АС в целом и ее частях, необходимые для обеспечения эксплуатации системы;
- описание функционирования системы и ее частей.

В разделе "Описание взаимосвязей АС с другими системами" указывают:

- перечень систем, с которыми связана данная АС;
- описание связей между системами;
- описание регламента связей;
- описание взаимосвязей АС с подразделениями объекта автоматизации.

В разделе "Описание подсистем" указывают:

- структуру подсистем и назначение ее частей;
- сведения об подсистемах и их частях, необходимые для обеспечения их функционирования;
- описание функционирования подсистем и их частей.

6. **Программа и методика испытаний** должна содержать перечни конкретных проверок (решаемых задач), которые следует осуществлять при испытаниях для подтверждения выполнения требований ТЗ, со ссылками на соответствующие методики (разделы методик) испытаний. Перечень проверок, подлежащих включению в программу испытаний, включает:

- соответствие системы ТЗ;
- комплектность системы;
- комплектность и качество документации;
- комплектность, достаточность состава и качество программных средств и программной документации;
- количество и квалификация обслуживающего персонала;
- степень выполнения требований функционального назначения системы;
- контролепригодность системы;
- выполнение требований техники безопасности, противопожарной безопасности, промышленной санитарии, эргономики;

функционирование системы с применением программных средств.

Программа испытаний содержит разделы: объект испытаний; цель испытаний; общие положения; объем испытаний; условия и порядок проведения испытаний; материально-техническое обеспечение испытаний; метрологическое обеспечение испытаний; отчетность.

В разделе "Объект испытаний" указывают: полное наименование системы, обозначение; комплектность испытательной системы.

В разделе "Цель испытаний" указывают конкретные цели и задачи, которые должны быть достигнуты и решены в процессе испытаний.

В разделе "Общие положения" указывают:

- перечень руководящих документов, на основании которых проводят испытания;
- место и продолжительность испытаний;
- организации, участвующие в испытаниях;
- перечень ранее проведенных испытаний;
- перечень предъявляемых на испытания документов, откорректированных по результатам ранее проведенных испытаний.

В разделе "Объем испытаний" указывают:

- перечень этапов испытаний и проверок, а также количественные и качественные характеристики, подлежащие оценке;
- последовательность проведения и режимы испытаний;
- требования по испытаниям программных средств;
- перечень работ, проводимых после завершения испытаний, требования к ним, объем и порядок проведения.

В разделе "Условия и порядок проведения испытаний" указывают:

- условия проведения испытаний;
- условия начала и завершения отдельных этапов испытаний;
- имеющиеся ограничения в условиях проведения испытаний;
- требования к техническому обслуживанию системы;
- меры, обеспечивающие безопасность и безаварийность проведения испытаний;
- порядок взаимодействия организаций, участвующих в испытаниях;
- порядок привлечения экспертов для исследования возможных повреждений в процессе проведения испытаний;
- требования к персоналу, проводящему испытания, и порядок его допуска к испытаниям.

7. **Схема организационной структуры** содержит:

- состав подразделений (должностных лиц) организации, обеспечивающих функционирование АС либо использующих при принятии решения информацию, полученную от АС;
- основные функции и связи между подразделениями и отдельными должностными лицами, указанными на схеме, и их подчиненность.

8. **Описание организационной структуры** содержит разделы: изменения в организационной структуре управления объектом; организация подразделений; реорганизация существующих подразделений управления. В разделе "Изменения в организационной структуре управления объектом" указывают: проектные решения по изменению организационной структуры управления объектом и их обоснование; описание изменений во взаимосвязях между подразделениями. В разделе "Организация подразделений" приводят: описание организационной структуры и функций подразделений, создаваемых с целью обеспечения функционирования АС; описание регламента работ; перечень категорий работников и число штатных единиц. В разделе "Реорганизация существующих подразделений управления" указывают описание изменений, обусловленных созданием АС, которые необходимо осуществить в каждом из действующих подразделений управления объектом в организационной структуре, функциях подразделений, регламенте работы, составе персонала подразделений.

9. **Методика автоматизированного проектирования** содержит разделы: общие положения; постановка задачи; методика проектирования; исходные данные; проектные процедуры; оценка результатов. В разделе "Общие положения" указывают класс объектов, на которые распространена методика, состав специалистов-пользователей, требования и ограничения на условия применения методики. В разделе "Постановка задачи" указывают основные пути и направления решения задачи, требования и ограничения на решение, критерии оценки результатов. В разделе "Методика проектирования" описывают выбранные математические методы, используемые при проектировании, указывают состав и назначение проектных процедур, порядок взаимодействия проектных процедур в процессе выполнения. В разделе "Исходные данные" определяют состав, порядок выбора, представления и формирования массивов используемой информации, перечень обозначений элементов, описывающих предметную область, с указанием их наименований, единиц измерений, диапазона изменения значений, критерии оценки исходных данных, выбирают методы и модели решения. В разделе "Проектные процедуры" указывают по каждой проектной процедуре состав нормативно-справочных входных данных, правила доступа к ним, порядок выполнения процедуры, состав и форму выходных сообщений. В разделе "Оценка результатов" приводят анализ полученного проектного решения на соответствие заданным критериям.

10. **Перечень входных сигналов и данных** содержит разделы: перечень входных сигналов; перечень входных данных.

В разделе "Перечень входных сигналов" указывают:

- для аналогового сигнала - наименование измеряемой величины, единицы измерения, диапазон изменения, требования точности и периодичности измерения, тип сигнала;
- для дискретного сигнала - наименование, разрядность и периодичность, тип сигнала;
- для сигнала "да-нет" - источник формирования и смысловое значение сигнала.

В разделе "Перечень входных данных" указывают:

- наименование, кодовое обозначение и значность реквизитов входных данных;
- наименования и кодовые обозначения документов или сообщений, содержащих эти данные.

11. **Перечень выходных сигналов** (документов) содержит разделы: перечень выходных сигналов; перечень выходных документов. Раздел "Перечень выходных сигналов" содержит перечень выходных сигналов с указанием их наименований, назначения, единиц измерения и диапазонов изменения, способа представления, пользователей информации. Раздел "Перечень выходных документов" содержит перечень выходных документов с указанием их наименований, кодовых обозначений, перечня и значности реквизитов, пользователей информации.

12. **Описание информационного обеспечения системы** содержит разделы: состав информационного обеспечения; организация информационного обеспечения; организация сбора и передачи информации; построение системы классификации и кодирования; организация внутримашинной информационной базы; организация внемашинной информационной базы.

В разделе "Состав информационного обеспечения" указывают наименование и назначение всех баз данных и наборов данных.

В разделе "Организация информационного обеспечения" приводят:

- принципы организации информационного обеспечения системы;
- обоснование выбора носителей данных и принципы распределения информации по типам носителей;
- описание принятых видов и методов контроля в маршрутах обработки данных при создании и функционировании внемашинной и внутримашинной информационных баз с указанием требований, на соответствие которым проводят контроль;

- описание решений, обеспечивающих информационную совместимость АС с другими системами управления по источникам, потребителям информации, по сопряжению применяемых классификаторов (при необходимости), по использованию в АС унифицированных систем документации.

В разделе "Организация сбора и передачи информации" приводят:

- перечень источников и носителей информации с указанием оценки интенсивности и объема потоков информации;
- описание общих требований к организации сбора, передачи, контроля и корректировки информации.

В разделе "Построение системы классификации и кодирования" приводят:

- описание принятых для применения в АС классификаций объектов во вновь разработанных классификаторах и в тех действующих классификаторах, из которых используется часть кода;
- методы кодирования объектов классификации во вновь разработанных классификаторах.

В разделе "Организация внутримашинной информационной базы" приводят:

- описание принципов построения внутримашинной информационной базы, характеристики ее состава и объема;
- описание структуры внутримашинной информационной базы на уровне баз данных с описанием характера взаимосвязей баз данных и указанием функций АС, при реализации которых используют каждую базу данных, характеристики данных, содержащихся в каждой базе данных.

В разделе "Организация внемашинной информационной базы" приводят характеристики состава и объема внемашинной информационной базы, принципы ее построения, в том числе основные положения по организации и обслуживанию фонда нормативно-справочной информации во взаимосвязи с автоматизированными функциями.

13. **Описание организации информационной базы** содержит описание логической и физической структуры базы данных и состоит из двух частей: описание внутримашинной информационной базы; описание внемашинной информационной базы. Части документа содержат следующие разделы: логическая структура; физическая структура (для внутримашинной информационной базы); организация ведения информационной базы. В разделе "Логическая структура" приводят описание состава данных, их форматов и взаимосвязей между данными. В разделе "Физическая структура" приводят описание избранного варианта расположения данных на конкретных машинных носителях. При описании структуры внутримашинной информационной базы должны быть приведены перечни баз данных и массивов и логические связи между ними. Для массива информации указывают логическую структуру внутри массива или дают ссылку на документ "Описание массива информации". При описании структуры внемашинной информационной базы приводят перечень документов и других информационных сообщений, использование которых предусмотрено в системе, с указанием автоматизируемых функций, при реализации которых формируют или используют данный документ. В разделе "Организация ведения информационной базы" при описании внутримашинной базы приводят последовательность процедур при создании и обслуживании базы с указанием, при необходимости, регламента выполнения процедур и средств защиты базы от разрушения и несанкционированного доступа, а также с указанием связей между массивами баз данных и массивами входной информации. При описании внемашинной информационной базы должна быть приведена последовательность процедур по маршруту движения групп документов до передачи их на ВЦ, а также описан маршрут движения выходных документов.

14. **Описание систем классификации и кодирования** содержит перечень применяемых в АС зарегистрированных классификаторов всех категорий по каждому классифицируемому объекту,

описание метода кодирования, структуры и длины кода, указания о системе классификации и другие сведения по усмотрению разработчика.

15. **Описание массива информации** содержит:

- наименование массива;
- обозначение массива;
- наименование носителей информации;
- перечень реквизитов в порядке их следования в записях массива с указанием по каждому реквизиту: обозначения алфавита, длины в знаках и диапазона изменения (при необходимости), логических и семантических связей с другими реквизитами данной записи и другими записями массива;
- оценку объема массива;
- другие характеристики массива (при необходимости).

16. **Отчет по ГОСТ 7.32 на стадии "Формирование требований к АС"** содержит разделы:

- характеристика объекта и результатов его функционирования;
- описание существующей информационной системы;
- описание недостатков существующей информационной системы;
- обоснование необходимости совершенствования информационной системы объекта;
- цели, критерии и ограничения создания АС;
- функции и задачи создаваемой АС;
- выводы и предложения.

В разделе "Характеристика объекта и результатов его функционирования" описывают тенденции развития, требования к объему, номенклатуре и качеству результатов функционирования, а также характер взаимодействия объекта с внешней средой. Раздел "Описание существующей информационной системы" содержит описание функциональной и информационной структуры системы, качественных и количественных характеристик, раскрывающих взаимодействие ее компонентов в процессе функционирования. В разделе "Описание недостатков существующей информационной системы" приводят результаты диагностического анализа, при котором оценивают качество функционирования и организационно-технологический уровень системы, выявляют недостатки в организации и технологии функционирования информационных процессов и определяют степень их влияния на качество функционирования системы. В разделе "Обоснование необходимости совершенствования информационной системы объекта" при анализе соответствия показателей функционирования объекта предъявляемым требованиям оценивают степень соответствия прогнозируемых показателей требуемым, а также выявляют необходимость совершенствования информационной системы путем создания АС. Раздел "Цели, критерии и ограничения создания АС" содержит: формулировку производственно-хозяйственных, научно-технических и экономических целей и критериев создания АС; характеристику ограничений по созданию АС.

Раздел "Функции и задачи создаваемой АС" содержит:

- обоснование выбора перечня автоматизированных функций и комплексов задач с указанием очередности внедрения;
- требования к характеристикам реализации функций и задач в соответствии с действующими нормативно-техническими документами, определяющими общие технические требования к АС конкретного вида;
- дополнительные требования к АС в целом и ее частям, учитывающие специфику создаваемой АС.

Раздел "Ожидаемые технико-экономические результаты создания АС" содержит:

перечень основных источников экономической эффективности получаемых в результате создания АС (в том числе - экономия производственных ресурсов, улучшение качества продукции,

повышение производительности труда и т. д.) и оценку ожидаемых изменений основных технико-экономических и социальных показателей производственно-хозяйственной деятельности объекта (например, показателей по номенклатуре и объемам производства, себестоимости продукции, рентабельности, отчислениям в фонд экономического стимулирования, уровню социального развития);

оценку ожидаемых затрат на создание и эксплуатацию АС с распределением их по очередям создания АС и по годам;

- ожидаемые обобщающие показатели экономической эффективности АС.
  Раздел "Выводы и предложения" рекомендуется разделять на подразделы: выводы о производственно-хозяйственной необходимости и технико-экономической целесообразности создания АС; предложения по совершенствованию организации и технологии процесса деятельности; рекомендации по созданию АС. Подраздел "Выводы о производственно-хозяйственной необходимости и технико-экономической целесообразности создания АС" содержит: сопоставление ожидаемых результатов создания АС с заданными целями и критериями создания АС (по целевым показателям и нормативным требованиям); принципиальное решение вопроса о создании АС (положительное или отрицательное). Подраздел "Предложения по совершенствованию организации и технологии процесса деятельности" содержит предложения по совершенствованию: производственно-хозяйственной деятельности; организационной и функциональной структур системы, методов деятельности, видов обеспечения АС.
  Подраздел "Рекомендации по созданию АС" содержит рекомендации:
- по виду создаваемой АС, ее совместимости с другими АС и неавтоматизируемой частью соответствующей системы;
- по организационной и функциональной структуре создаваемой АС;
- по составу и характеристикам подсистем и видов обеспечения АС;
- по организации использования имеющихся и приобретению дополнительных средств вычислительной техники;
- по рациональной организации разработки и внедрения АС;
- по определению основных и дополнительных, внешних и внутренних источников и видов объемов финансирования и материального обеспечения разработок АС;
- по обеспечению производственных условий создания АС;
- другие рекомендации по созданию АС.

17. **Отчет по ГОСТ 7.32 на стадии "Разработка концепции АС"** содержит:

- описание результатов изучения объекта автоматизации;
- описание и оценку преимуществ и недостатков разработанных альтернативных вариантов концепции создания АС;
- сопоставительный анализ требований пользователя к АС и вариантов концепции АС на предмет удовлетворения требованиям пользователя;
- обоснование выбора оптимального варианта концепции и описание предлагаемой АС;
- ожидаемые результаты и эффективность реализации выбранного варианта концепции АС;
- ориентировочный план реализации выбранного варианта концепции АС;
- необходимые затраты ресурсов на разработку, ввод в действие и обеспечение функционирования;
- требования, гарантирующие качество АС;
- условия приемки системы.

П3.3. ГОСТ 234.003-90. Автоматизированные системы. Термины и определения

Настоящий стандарт устанавливает термины и определения основных понятий в области автоматизированных систем (АС) и распространяется на АС, используемые в различных сферах деятельности (управление, исследования, проектирование и т.п., включая их сочетание), содержанием которых является переработка информации.

Автоматизированная система. Система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций.

Интегрированная АС. Совокупность двух или более взаимоувязанных АС, в которой функционирование одной из них зависит от результатов функционирования другой (других) так, что эту совокупность можно рассматривать как единую АС.

Функция АС. Совокупность действий АС, направленная на достижение определенной цели.

Алгоритм функционирования АС. Алгоритм, задающий условия и последовательность действий компонентов автоматизированной системы при выполнении ею своих функций.

Пользователь АС. Лицо, участвующее в функционировании АС или использующее результаты ее функционирования.

Организационное обеспечение АС. Совокупность документов, устанавливающих организационную структуру, права и обязанности пользователей и эксплуатационного персонала АС в условиях функционирования, проверки и обеспечения работоспособности АС.

Методическое обеспечение АС. Совокупность документов, описывающих технологию функционирования АС, методы выбора и применения пользователями технологических приемов для получения конкретных результатов при функционировании АС.

Техническое обеспечение АС. Совокупность всех технических средств, используемых при функционировании АС.

Математическое обеспечение АС. Совокупность математических методов, моделей и алгоритмов, примененных в АС.

Программное обеспечение АС. Совокупность программ на носителях данных и программных документов, предназначенная для отладки, функционирования и проверки работоспособности АС.

Информационное обеспечение АС. Совокупность форм документов, классификаторов, нормативной базы и реализованных решений по объемам, размещению и формам существования информации, применяемой в АС при ее функционировании.

Лингвистическое обеспечение АС. Совокупность средств и правил для формализации естественного языка, используемых при общениии пользователей и эксплуатационного персонала АС с комплексом средств автоматизации при функционировании АС.

Правовое обеспечение АС. Совокупность правовых норм, регламентирующих правовые отношения при функционировании АС и юридический статус результатов ее функционирования.

Эргономическое обеспечение АС. Совокупность реализованных решений в АС по согласованию психологических, психофизиологических, антропометрических, физиологических характеристик и возможностей пользователей АС с техническими характеристиками комплекса средств автоматизации АС и параметрами рабочей среды на рабочих местах персонала АС.

Компонент АС. Часть АС, выделенная по определенному признаку или совокупности признаков и рассматриваемая как единое целое.

Информационная база АС. Совокупность упорядоченной информации, используемой при функционировании АС.

Внемашинная информационная база АС. Часть информационной базы АС, представляющая собой совокупность документов, предназначенных для непосредственного восприятия человеком без применения средств вычислительной техники.

Машинная информационная база АС. Часть информационной базы АС, представляющая собой совокупность используемой в АС информации на носителях данных.

Автоматизированное рабочее место (АРМ). Программно-технический комплекс АС, предназначенный для автоматизации деятельности определенного вида.

Эффективность АС. Свойство АС, характеризуемое степенью достижения целей, поставленных при ее создании.

Совместимость АС. Комплексное свойство двух или более АС, характеризуемое их способностью взаимодействовать при функционировании (включающее техническую, программную, информационную, организационную и лингвистическую совместимость).

Адаптивность АС. Способность АС изменяться для сохранения своих эксплуатационных показателей в заданных пределах при изменениях внешней среды.

Надежность АС. Комплексное свойство АС сохранять во времени в установленных пределах значения всех параметров, характеризующих способность АС выполнять свои функции в заданных режимах и условиях эксплуатации.

Живучесть АС. Свойство АС, характеризуемое способностью выполнять установленный объем функций в условиях воздействий внешней среды и отказов компонентов системы в заданных пределах.

Жизненный цикл АС. Совокупность взаимосвязанных процессов создания и последовательного изменения состояния АС от формирования исходных требований к ней до окончания эксплуатации и утилизации комплекса средств автоматизации АС.

Сопровождение АС. Деятельность по оказанию услуг, необходимых для обеспечения устойчивого функционирования или развития АС.

Диалоговый режим выполнения функции АС. Режим выполнения функции АС, при котором человек управляет решением задачи, изменяя ее условия и/или порядок функционирования АС на основе оценки информации, представляемой ему техническими средствами АС.

Техническое задание на АС. Документ, оформленный в установленном порядке и определяющий цели создания АС, требования к АС и основные исходные данные, необходимые для ее разработки, а также план-график создания АС.

Технический проект АС. Комплект проектных документов на АС, разрабатываемый на стадии "Технический проект", утвержденный в установленном порядке, содержащий основные проектные решения по системе в целом, ее функциям и всем видам обеспечения АС и достаточный для разработки рабочей документации на АС.

Рабочая документация на АС. Комплект проектных документов на АС, разрабатываемый на стадии "Рабочая документация", содержащий взаимоувязанные решения по системе в целом, ее функциям, всем видам обеспечения АС, достаточные для комплектации, монтажа, наладки и функционирования АС, ее проверки и обеспечения работоспособности.

Эксплуатационная документация на АС. Часть рабочей документации на АС, предназначенная для использования при эксплуатации системы, определяющая правила действия персонала и пользователей системы при ее функционировании, проверке и обеспечении ее работоспособности.

Технорабочий проект АС. Комплект проектных документов АС, утвержденный в установленном порядке и содержащий решения в объеме технического проекта и рабочей документации на АС.

Входная информация АС. Информация, поступающая в АС в виде документов, сообщений, данных, сигналов, необходимая для выполнения функций АС.

Выходная информация АС. Информация, получаемая в результате выполнения функций АС и выдаваемая на объект ее деятельности, пользователю или в другие системы.

Нормативно-справочная информация АС. Информация, заимствованная из нормативных документов и справочников и используемая при функционировании АС.