МОДУЛЬ 3

РАЗДЕЛ 5. CASE-ТЕХНОЛОГИИ СТРУКТУРНОГО АНАЛИЗА И ПРОЕКТИРОВАНИЯ ПРОГРАММНЫХ СРЕДСТВ

5.1. Общие сведения о CASE-технологиях

В соответствии с положениями стандарта *СТБ ИСО/МЭК 12207–2003* процесс разработки сложных систем и ПС состоит из тринадцати работ (см. подразд. 1.2). Как показывают исследования, большинство ошибок вносится в системы и ПС при выполнении ранних работ процесса разработки (работы 2 – 6, связанные с анализом и проектированием). Существенно меньше ошибок возникает при осуществлении программирования, тестирования и последующих работ, причем устранять такие ошибки гораздо проще по сравнению с ошибками ранних работ.

Как правило, ошибки, возникающие при выполнении ранних работ процесса разработки системы или программного средства, являются следствием неполноты или некорректности функциональной спецификации или несогласованности между спецификацией и результатами проектирования. Очевидно, что основная причина этого кроется в несоответствии методов, используемых при осуществлении ранних работ процесса разработки, целям данных работ.

С учетом этого с 70-х гг. XX в. ведется разработка методов структурного анализа и проектирования, специально предназначенных для использования при выполнении ранних работ процесса разработки сложных систем широкого профиля и позволяющих

существенно сократить возможности внесения ошибок в разрабатываемую систему. Напомним, что основной *целью* методов структурного анализа и проектирования является разделение сложных систем на части с последующей иерархической организацией этих частей.

Наиболее известными и используемыми из данных методов являются:

- метод структурного анализа и проектирования SADT Росса, в дальнейшем явившийся основой методологии функционального моделирования IDEF0;
- методы, ориентированные на потоки данных (методы Йодана, ДеМарко, Гейна, Сарсона), в дальнейшем явившиеся основой методологии структурного анализа потоков данных DFD; один из таких методов – анализ сообщений – рассмотрен в п. 4.3.4;
- методы структурирования данных (методы JSP Джексона, Орра, Чена), в дальнейшем явившиеся основой методологий JSD Джексона, информационного моделирования IDEF1 и IDEF1X и др.; метод JSP Джексона подробно рассмотрен в подразд. 4.6.

Появление новых методов анализа и проектирования вызвало необходимость создания ПО, позволяющего автоматизировать их использование при разработке больших систем. С середины 80-х гг. XX в. начал формироваться рынок ПС, названных CASE-средствами.

Первоначально *термин CASE* трактовался как Computer Aided Software Engineering (компьютерная поддержка проектирования ПО). В настоящее время данному термину придается более широкий смысл и он расшифровывается как Computer Aided System Engineering (компьютерная поддержка проектирования систем). Современные CASE-средства ориентируются на моделирование предметной области, разработку спецификаций, проектирование сложных систем широкого назначения. При этом учитывается, что программное средство – это частный случай системы вообще. Считается, что разработка ПС включает в себя практически те же этапы, что и разработка систем общего назначения.

С учетом вышеизложенного введено понятие CASE-технологии.

CASE-технология – это совокупность методологий разработки и сопровождения сложных систем (в том числе ПС), поддерживаемая комплексом взаимосвязанных средств автоматизации.

Основные цели использования CASE-технологий при разработке ПС – отделить анализ и проектирование от программирования и последующих работ процесса разработки, предоставив разработчику соответствующие методологии визуального анализа и проектирования.

С середины 70-х гг. XX в. в США финансировался ряд проектов, ориентированных на разработку методов описания и моделирования сложных систем [9]. Один из них – проект ***ICAM*** (Integrated Computer-Aided Manufacturing). Его целью являлась разработка подходов, обеспечивающих повышение эффективности производства благодаря систематическому внедрению компьютерных технологий. В соответствии с проектом ICAM было разработано семейство трех методологий ***IDEF*** (ICAM DEFinition), позволяющих моделировать различные аспекты функционирования производственной среды или системы:

- ***IDEF0*** – методология функционального моделирования производственной среды или системы; отображает структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции; основана на методе SADT Росса;
- ***IDEF1*** – методология информационного моделирования производственной среды или системы; отображает структуру и содержание информационных потоков, необходимых для поддержки функций системы; основана на реляционной теории Кодда и использовании ER-диаграмм (диаграмм «Сущность–Связь») Чена;
- ***IDEF2*** – методология динамического моделирования производственной среды или системы; отображает меняющееся во времени поведение функций, информации и ресурсов системы.

В дальнейшем семейство методологий IDEF было дополнено следующими методологиями:

- ***IDEF1X*** – методология семантического моделирования данных; в стандарте названа методологией концептуального моделирования; представляет собой расширение методологии IDEF1, поэтому в литературе часто называется методологией информационного моделирования;
- ***IDEF3*** – методология моделирования сценариев процессов, происходящих в производственной среде или системе; отображает состояния объектов и потоков данных,

связи между ситуациями и событиями; используется для документирования процессов, происходящих в производственной среде или системе;

- ***IDEF4*** – методология объектно-ориентированного анализа и проектирования;
- ***IDEF5*** – методология моделирования онтологии производственной среды или системы (***онтология*** – это завершенный словарь для определенной области, имеющий набор точных определений или аксиом, которые накладывают на значения терминов в данном словаре ограничения, достаточные для непротиворечивой интерпретации данных); использует утверждения о реальных объектах, их свойствах и их взаимосвязях в производственной среде или системе.

Из вышеназванных методологий наибольшее распространение нашли методологии IDEF0, IDEF1X и IDEF3.

Методологии IDEF0 и IDEF1X являются стандартизированными. Данным методологиям посвящены международные стандарты ***ISO/IEC/IEEE 31320-1:2012 - Информационные технологии – Языки моделирования – Часть 1: Синтаксис и семантика для IDEF0*** и ***ISO/IEC/IEEE 31320-2:2012 - Информационные технологии – Языки моделирования – Часть 2: Синтаксис и семантика для IDEF1X97 (IDEFobject)***.

В Республике Беларусь действует национальный стандарт ***СТБ 2195-2011. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования***.

Резюме

В настоящее время при выполнении ранних работ процесса разработки (работ, связанных с анализом предметной области и проектированием систем и ПС) широко используются CASE-технологии. Термин CASE расшифровывается как Computer Aided System Engineering (компьютерная поддержка проектирования систем). При структурном анализе и проектировании широко применяется семейство методологий IDEF.

5.2. Методология функционального моделирования IDEF0

5.2.1. Общие сведения о методологии SADT

Основные положения методологии IDEF0 регламентированы международным стандартом *ISO/IEC/IEEE 31320-2:2012* и национальным стандартом Республики Беларусь *СТБ 2195-2011*.

Как уже отмечалось, основой для методологии функционального моделирования IDEF0 является методология структурного анализа и проектирования SADT. Методология ***SADT*** (Structured Analysis And Design Technique) [22] сформулирована в общих чертах Дугласом Т.Россом (компания SofTech) в 70-х гг. XX в. На рынке SADT появилась в 1975 г. К 1981 году SADT уже использовали более чем в 50 компаниях.

Основным назначением методологии SADT является моделирование предметной области с целью определения требований к разрабатываемой системе или программному средству и с целью их проектирования. Методология SADT может применяться при выполнении ранних работ процесса разработки системы или программного средства (работы 2 – 5, см. подразд. 1.2).

К *достоинствам методологии SADT* можно отнести:

1) универсальность – SADT может использоваться для проектирования не только ПС, но и сложных систем любого назначения (например управление и контроль, аэрокосмическое производство, телефонные сети, учет материально-технических ресурсов и др.);

2) SADT – методология, достаточно просто отражающая такие системные характеристики, как управление, обратная связь и исполнители;

3) SADT имеет развитые процедуры поддержки коллективной работы;

4) SADT предназначена для использования на ранних этапах создания систем или ПС (при выполнении работ, связанных с анализом предметной области, разработкой требований и проектированием);

5) SADT сочетается с другими структурными методами проектирования.

В моделировании SADT определены два направления: *функциональные модели* выделяют события в системе, *модели данных* выделяют объекты (данные) системы, связывающие функции между собой и с их окружением. В обоих случаях используется один и тот же графический язык блоков и дуг (но блоки и дуги меняются ролями) [15]. При наиболее полном моделировании возможно использование взаимодополняющих моделей обоих типов.

Однако наибольшее распространение и дальнейшее развитие получил функциональный вариант методологии SADT, на базе которого разработана и в дальнейшем стандартизирована методология функционального моделирования IDEF0.

Резюме

Методология структурного анализа и проектирования SADT предназначена для моделирования предметной области с целью определения требований к разрабатываемой системе и ее проектирования. Методология SADT применяется при выполнении ранних работ процесса разработки системы или программного средства. В первую очередь это такие работы, как анализ требований к системе, проектирование системной архитектуры, анализ требований к программным средствам, проектирование программной архитектуры (см. подразд. 1.2). Развитием методологии SADT является методология IDEF0.

5.2.2. Основные понятия IDEF0-модели

При IDEF0-моделировании используются следующие понятия и определения [1, 8, 22].

Под ***системой*** подразумевается совокупность взаимодействующих компонентов и взаимосвязей между ними.

Моделированием называется процесс создания точного описания системы. IDEF0-методология предназначена для создания описания систем и основана на концепциях системного моделирования.

Под ***моделью IDEF0*** подразумевается графическое и текстовое представление результатов анализа предметной области, разработанное с определенной целью и с выбранной точки зрения и идентифицирующее функции системы. IDEF0-модель дает полное и точное описание, адекватное системе и имеющее конкретное назначение.

Назначение описания называют ***целью модели.***

Формальное определение IDEF0-модели имеет следующий вид:

M есть модель системы **S**, если **M** может быть использована для получения ответов на вопросы относительно **S** с точностью **A**.

Таким образом, *целью модели* является получение ответов на некоторую совокупность вопросов. Обычно вопросы для IDEF0-модели формируются на самом раннем этапе разработки (еще нет технического задания и спецификации). Затем основная суть этих вопросов должна быть выражена в одной-двух фразах.

С определением модели тесно связан выбор точки зрения, с которой наблюдается система и создается ее модель. ***Точка зрения*** – это позиция человека или объекта, в которую надо встать, чтобы увидеть систему в действии. Методология IDEF0 требует, чтобы модель рассматривалась все время с одной и той же позиции.

Например, при разработке автоматизированной обучающей системы (АОС) точкой зрения может быть позиция неквалифицированного пользователя, квалифицированного пользователя, программиста и т.п.

Пример 5.1

Пусть необходимо разработать программное средство, предназначенное для автоматизации процесса выполнения студентами лабораторных работ. Разработку функциональной спецификации программного средства следует начать с разработки IDEF0-модели процесса выполнения лабораторных работ.

На первом этапе разработки IDEF0-модели формулируются вопросы к ней, формируется цель модели, определяются претенденты на точку зрения, выбирается точка зрения.

Например, в *перечень вопросов* к IDEF0-модели в рассматриваемом примере могут входить такие вопросы:

- Какие этапы лабораторной работы необходимо выполнить студенту?
- Какие сотрудники участвуют в процессе выполнения студентом лабораторной работы?
- Какие виды работ должен осуществлять преподаватель во время выполнения студентом лабораторной работы?

- Какие виды работ должен осуществлять лаборант во время выполнения студентом лабораторной работы?
- Какая информация является входной при выполнении студентами лабораторных работ?
- Как влияют результаты отдельных этапов на итоги выполнения лабораторной работы?
- Что необходимо для защиты лабораторной работы?

На основании перечня вопросов формулируется *цель модели*: определить основные этапы процесса выполнения лабораторной работы, их влияние друг на друга и на результаты защиты работы с целью обучения студентов методологии IDEF0.

Претенденты на точку зрения: преподаватель, студент, лаборант. С учетом цели модели предпочтение следует отдать точке зрения преподавателя, так как она наиболее полно охватывает все этапы лабораторной работы и только с этой точки зрения можно показать взаимосвязи между отдельными этапами и обязанности участников лабораторной работы.

Субъектом моделирования является сама система. Но система не существует изолированно, она связана с окружающей средой. Иногда трудно сказать, где кончается система и начинается среда. Поэтому в методологии IDEF0 подчеркивается необходимость точного определения ***границ системы***, чтобы избежать включения в модель посторонних субъектов. IDEF0-модель должна иметь *единственный субъект*.

Таким образом, субъект определяет, что включить в модель, а что исключить из нее. Точка зрения диктует автору модели выбор нужной информации о субъекте и форму ее подачи. Цель становится критерием окончания моделирования.

Конечным результатом моделирования является набор тщательно взаимоувязанных описаний, начиная с описания самого верхнего уровня всей системы и кончая подробным описанием деталей или операций системы. Каждое из таких описаний называется ***диаграммой***.

IDEF0-модель – это древовидная структура диаграмм, где верхняя диаграмма является наиболее общей, а нижние наиболее детализированы. Каждая из диаграмм

какого-либо уровня представляет собой декомпозицию некоторого компонента диаграммы предыдущего уровня.

Резюме

Методология IDEF0 создана специально для представления сложных систем путем построения моделей. IDEF0-модель – это описание системы, разработанное для единственного субъекта с определенной целью и с выбранной точки зрения. Целью служит набор вопросов, на которые должна ответить модель. Точка зрения – позиция, с которой описывается система. Цель и точка зрения – это основополагающие понятия IDEF0. Описание модели IDEF0 организовано в виде иерархии взаимосвязанных диаграмм. Вершина этой древовидной структуры представляет самое общее описание системы, а ее основание состоит из наиболее детализированных описаний.

5.2.3. Синтаксис IDEF0-диаграмм

Диаграммы являются основными рабочими элементами IDEF0-модели. Диаграммы представляют входные-выходные преобразования и указывают правила и средства этих преобразований. Каждая IDEF0-диаграмма содержит блоки (работы) и дуги (линии со стрелками). Блоки изображают функции моделируемой системы. Дуги связывают блоки вместе и отображают взаимодействия и взаимосвязи между ними [1, 8, 9, 10].

Синтаксис блоков

Функциональные блоки на диаграмме изображаются прямоугольниками (рис.5.1).

Блок представляет функцию или активную часть системы.

Каждая сторона блока имеет определенное назначение. Левая сторона предназначена для входов, верхняя – для управления, правая – для выходов, нижняя – для механизмов и вызовов.

Рис. 5.1. Основная конструкция IDEF0-модели

Назначение дуг

В IDEF0 различают ***пять типов дуг:*** вход (input), управление (control), выход (output), механизм (mechanism), вызов (call).

В основе методологии IDEF0 лежат следующие *правила*:

1) *вход* представляет собой входные данные, используемые или преобразуемые функциональным блоком для получения результата (выхода); блок может не иметь ни одной входной дуги (например блок, выполняющий генерацию случайных чисел);

2) *выход* представляет собой результат работы блока; наличие выходной дуги для каждого блока является обязательным;

3) *управление* ограничивает или определяет условия выполнения преобразований в блоке; в качестве дуг управления могут использоваться некоторые условия, правила, стратегии, стандарты, которые влияют на выполнение функционального блока; наличие управляющей дуги для каждого блока является обязательным;

4) *механизмы* показывают, кто, что и как выполняет преобразования в блоке; механизмы определяют ресурсы, непосредственно осуществляющие эти преобразования (например денежные средства, персонал, оборудование и т.п.); механизмы представляются стрелками, подключенными к нижней стороне блока и направленными вверх к блоку; наличие дуг механизмов для блока не является обязательным;

5) *вызовы* представляют собой специальный вид дуги и обозначают обращение из

данной модели или из данной части модели к блоку, входящему в состав другой модели или другой части модели, обеспечивая их связь; с помощью дуги вызова разные модели или разные части одной модели могут совместно использовать один и тот же блок; вызовы не являются компонентом собственно методологии SADT, они является расширением IDEF0-методологии и предназначены для организации коллективной работы над моделью, разделения модели на независимые модели и объединения различных моделей предметной области в одну модель; вызовы представляются стрелками, подключенными к нижней стороне блока и направленными вниз от блока; наличие дуги вызова для блока не является обязательным.

Представление блоков и дуг на диаграмме

Рассмотрим синтаксис IDEF0-диаграмм на примере IDEF0-диаграммы, содержащей основные этапы процесса выполнения лабораторной работы (см. пример 5.1 в п. 5.2.2). Данная диаграмма представлена на рис. 5.2.

В русскоязычной литературе название IDEF0-блока принято основывать на использовании отглагольного существительного, обозначающего действие (вычисление того-то, определение того-то, обработка того-то и т.д.). Блоки на рис. 5.2 имеют названия «Изучение теории», «Ответы на контрольные вопросы», «Выполнение индивидуального задания», «Написание отчета», «Защита лабораторной работы». В стандарте ISO/IEC/IEEE 31320-1:2012 рекомендуется для названий блоков использовать глаголы в неопределенной форме (в этом случае в применении к рассматриваемому примеру названия блоков будут следующими: «Изучить теорию», «Ответить на контрольные вопросы», «Выполнить индивидуальное задание», «Написать отчет», «Защитить лабораторную работу»).

Методология IDEF0 требует, чтобы в диаграмме было *не менее трех и не более шести блоков*. Это ограничение поддерживает сложность диаграмм на уровне, доступном для чтения, понимания и использования.

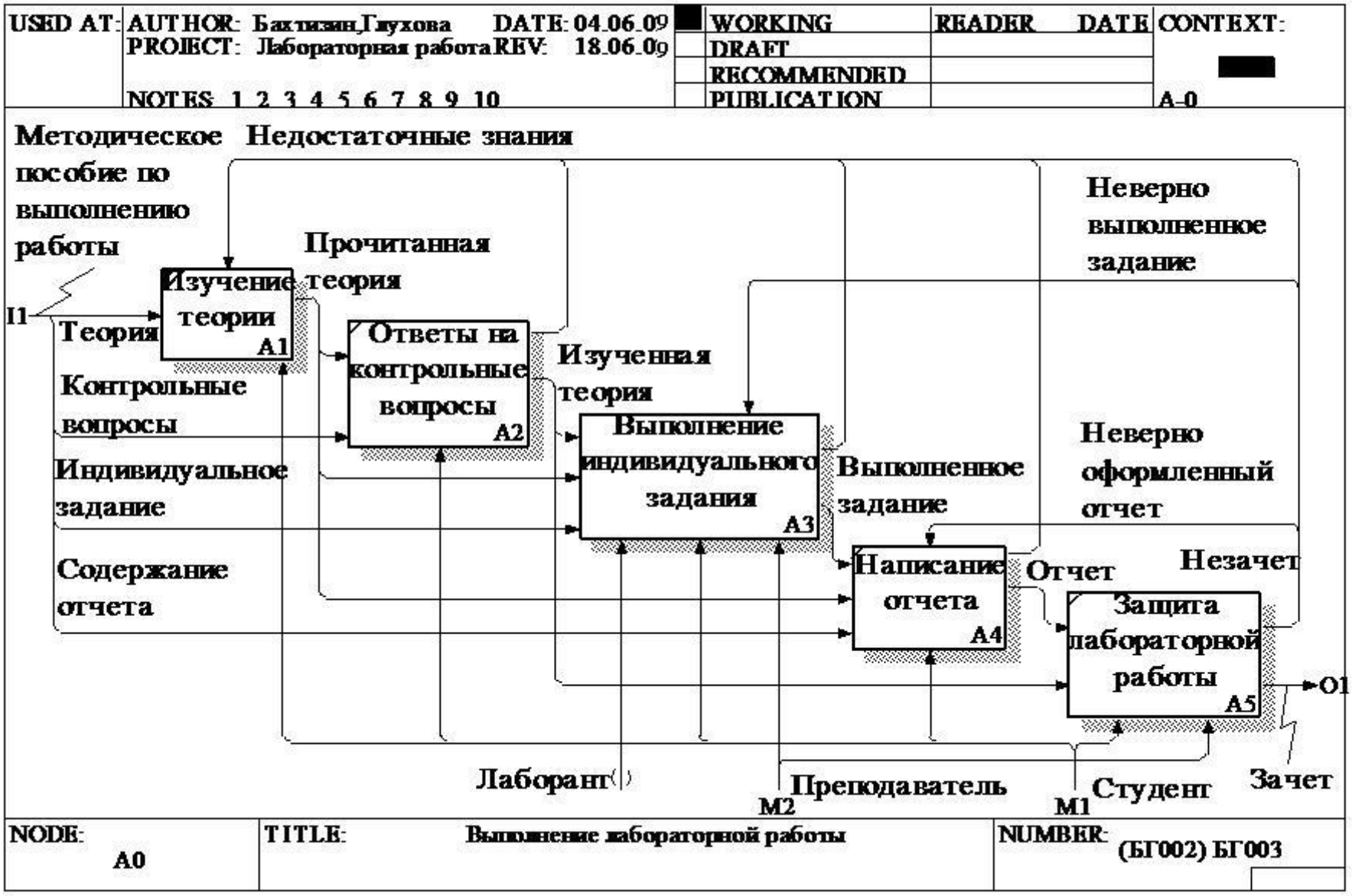


Рис. 5.2. Стандартный IDEF0-бланк и IDEF0-диаграмма, содержащая основные этапы процесса выполнения лабораторной работы

Блоки на IDEF0-диаграмме размещаются в порядке степени их важности. В IDEF0 этот относительный порядок называется ***доминированием***. Доминирование понимается как влияние, которое один блок оказывает на другие блоки диаграммы. В методологии IDEF0 принято располагать блоки по диагонали диаграммы (см. рис. 5.2). Наиболее доминирующий блок обычно размещается в левом верхнем углу диаграммы, наименее доминирующий – в правом нижнем углу.

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

Блоки на IDEF0-диаграмме должны быть пронумерованы. Нумерация блоков выполняется в соответствии с порядком их доминирования (1 – наибольшее доминирование, 2 – следующее и т.д.). Номер блока может содержать префикс **A** (Activity). Номер располагается в правом нижнем углу функционального блока.

Дуги на IDEF0-диаграмме изображаются линиями со стрелками. Для функциональных IDEF0-диаграмм дуга представляет множество объектов. Под ***объектом*** в общем случае понимаются некоторые данные (планы, машины, информация и т.п.). Основу названия дуги на IDEF0-диаграммах составляют существительные. Названия дуг называются *метками.*

Например, на диаграмме, представленной на рис. 5.2, дуги имеют названия «Индивидуальное задание», «Выполненное задание», «Отчет» и т.д.

Типы взаимосвязей между блоками

Дуги определяют, как блоки влияют друг на друга. Это влияние может выражаться:

- в передаче выходной информации к другому блоку для дальнейшего преобразования;
- в выработке управляющей информации, предписывающей, что именно должен выполнить другой блок;
- в передаче информации, определяющей средство достижения цели для другого блока.

С учетом этого в методологии IDEF0 используется ***пять типов взаимосвязей между блоками*** для описания их отношений: управление, вход, обратная связь по

управлению, обратная связь по входу, выход-механизм.

Отношение управления возникает тогда, когда выход одного блока непосредственно влияет на работу блока с меньшим доминированием (рис. 5.3).

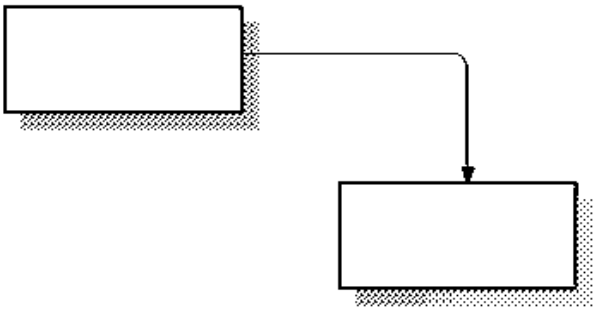


Рис. 5.3. Отношение управления

Отношение входа возникает тогда, когда выход одного блока становится входом для блока с меньшим доминированием (рис. 5.4).

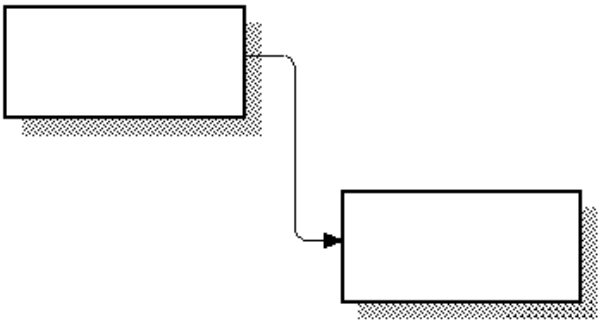


Рис. 5.4. Отношение входа

Обратные связи по управлению и по входу предназначены для представления итерации или рекурсии.

Обратная связь по управлению возникает тогда, когда выход некоторого блока влияет на работу блока с большим доминированием (рис. 5.5).

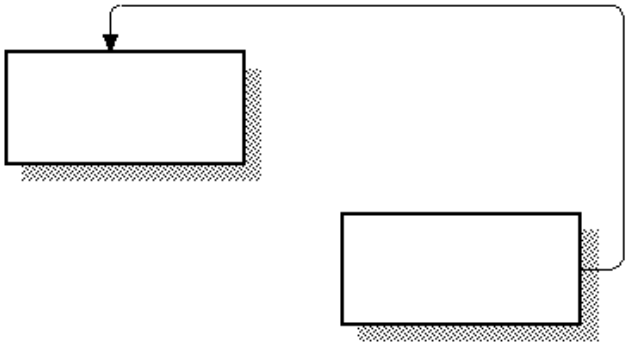


Рис. 5.5. Обратная связь по управлению

Обратная связь по входу имеет место тогда, когда выход одного блока становится входом другого блока с большим доминированием (рис. 5.6).

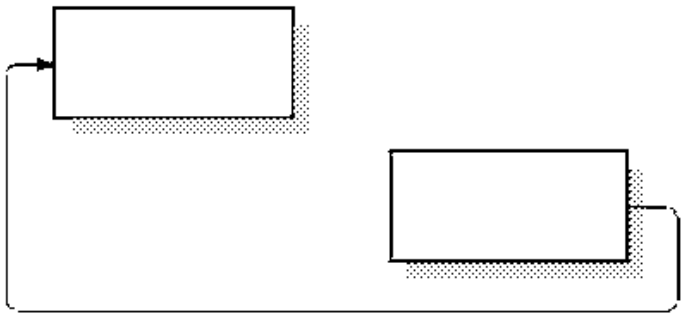


Рис. 5.6. Обратная связь по входу

Связь «выход-механизм» встречается нечасто и отражает ситуацию, при которой выход одного блока становится средством достижения цели для другого блока (рис. 5.7). Данная связь характерна при распределении источников ресурсов (например физическое пространство, оборудование, финансирование, материалы, инструменты, обученный персонал и т.п.).

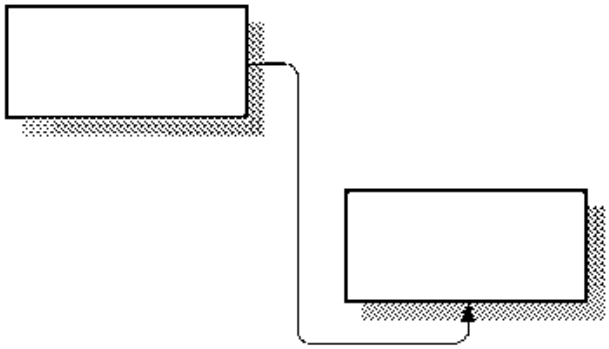


Рис. 5.7. Связь «выход-механизм»

Декомпозиция дуг

Дуга в IDEF0 редко изображает один объект. Обычно она символизирует набор объектов. Поэтому дуги могут разъединяться и соединяться.

Разветвления дуг обозначают, что все содержимое дуг или его часть может появиться в каждом ответвлении дуги. Дуга всегда помечается до разветвления, чтобы дать название всему набору (см. например входную дугу I1 «Методическое пособие по выполнению работы» диаграммы и выходную дугу «Прочитанная теория» блока «Изучение теории» на рис. 5.2). Каждая ветвь дуги помечается или не помечается в соответствии со следующими *правилами*:

- непомеченные ветви содержат все объекты, указанные в метке дуги перед разветвлением (см. ветви выходной дуги «Прочитанная теория» блока «Изучение теории» на рис. 5.2);
- каждая метка ветви указывает, какие именно объекты исходного набора содержит ветвь (см. входную дугу I1 «Методическое пособие по выполнению работы» и ее ветви «Теория», «Контрольные вопросы», «Индивидуальное задание» и «Содержание отчета» на рис. 5.2).

При ***слиянии дуг*** результирующая дуга всегда помечается для указания нового набора объектов, возникшего после объединения (см. дугу управления «Недостаточные знания» блока «Изучение теории» на рис. 5.2). Каждая ветвь перед слиянием помечается или не помечается в соответствии со следующими *правилами*:

• непомеченные ветви содержат все объекты, указанные в общей метке дуги после слияния (см. ветви дуги управления «Недостаточные знания» блока «Изучение теории» на рис. 5.2);

• метка ветви указывает, какие конкретно объекты результирующего набора содержит ветвь.

Разветвления дуг и их соединения – это синтаксис, который позволяет описывать декомпозицию (разделение на структурные части) содержимого дуг. Разветвляющиеся и соединяющиеся дуги отражают иерархию объектов, представленных этими дугами. Следует обратить внимание на то, что синтаксис дуг позволяет одной и той же ветви участвовать как в разветвлении, так и в слиянии дуг. На рис. 5.2 такой дугой, образованной в результате разветвления и слияния ветвей, является дуга управления «Недостаточные знания» блока «Изучение теории». В данном случае выходная ветвь «Незачет» блока «Защита лабораторной работы» разветвляется на три ветви. Слияние одной из этих ветвей («Недостаточные знания») вместе с тремя аналогичными выходными дугами блоков A2, A3, A4 образовало новый набор объектов с тем же названием.

Из отдельной диаграммы редко можно понять полную иерархию дуги. Для этого требуется рассмотрение некоторой части модели. Поэтому методология IDEF0 предусматривает дополнительное описание полной иерархии объектов системы посредством формирования *глоссария* для каждой диаграммы модели и объединения этих глоссариев в *Словарь данных*. Таким образом, ***Словарь данных*** – это основное хранилище полной иерархии объектов системы.

Хронологические номера диаграмм

Для систематизации информации о диаграммах и модели в целом используется *стандартный IDEF0-бланк* (см. рис. 5.2). Каждое поле бланка имеет конкретное назначение и заполняется по определенным правилам. Данные поля будут описаны ниже по ходу изложения материала.

При создании IDEF0-модели одна и та же диаграмма может перечерчиваться несколько раз, что приводит к появлению различных ее вариантов. Чтобы различать их, в методологии IDEF0 используется *схема контроля конфигурации диаграмм*, основанная

на *хронологических номерах*, или ***С-номерах***. С-номерные коды образуются из инициалов автора (авторов) и последовательных номеров. Эти коды записываются в нижнем правом углу IDEF0-бланка (БГ003, см. рис. 5.2). Если диаграмма заменяет более старый вариант, предыдущий С-номер помещается в скобках (например БГ002, см. рис. 5.2). Каждый автор проекта IDEF0 ведет реестр (список) всех созданных им диаграмм, нумеруя их последовательными целыми числами. Для этого используется специальный *бланк реестра С-номеров* IDEF0.

Резюме

Основой IDEF0-диаграмм является блок. Каждая сторона блока имеет определенное назначение (вход, управление, выход, механизм, вызов). Диаграмма в IDEF0 содержит 3 – 6 блоков, связанных дугами, и может иметь несколько версий. Чтобы различить данные версии, используются С-номера. Блоки на диаграмме представляют функции моделируемой системы, дуги – множество различных объектов системы. Блоки изображаются на диаграмме в соответствии с порядком их доминирования. Дуги могут разветвляться и объединяться.

5.2.4. Синтаксис IDEF0-моделей

Диаграммы, собранные и связанные вместе, представляют собой IDEF0-модель проектируемой или анализируемой системы. В методологии IDEF0 дополнительно к правилам синтаксиса диаграмм существуют правила синтаксиса моделей. Синтаксис IDEF0-моделей позволяет автору проекта определить границу модели, связать диаграммы в одно целое и обеспечить точное согласование между диаграммами [1, 8, 9, 10].

Декомпозиция блоков

IDEF0-модель представляет собой иерархически организованную совокупность диаграмм. Диаграмма содержит 3 – 6 блоков. Каждый из блоков потенциально может быть детализирован на другой диаграмме. Разделение блока на его структурные части (блоки и дуги) называется ***декомпозицией***.

Декомпозиция формирует границы, то есть блок и касающиеся его дуги определяют точную границу диаграммы, представляющей декомпозицию этого блока. Эта диаграмма называется ***диаграммой-потомком***. Декомпозируемый блок называется ***родительским блоком***, а содержащая его диаграмма – ***родительской диаграммой***. Название диаграммы-потомка совпадает с функцией родительского блока. Таким образом, IDEF0-диаграмма является декомпозицией некоторой ограниченной функции (субъекта). Принцип ограничения субъекта встречается на каждом уровне.

Контекстная диаграмма

Один блок и несколько дуг на самом верхнем уровне модели используются для определения границы всей системы. Этот блок описывает общую функцию, выполняемую системой. Дуги, касающиеся этого блока, описывают главные входы, выходы, управления и механизмы этой системы.

Диаграмма, определяющая границу системы и состоящая из одного блока и его дуг, называется ***контекстной диаграммой модели***. Все, что лежит внутри блока, является частью описываемой системы, а все, лежащее вне его, образует *среду системы*.

Рис. 5.8 представляет контекстную диаграмму процесса выполнения лабораторной работы.

Общая функция модели записывается на контекстной диаграмме в виде названия блока (для рассматриваемого процесса – это выполнение лабораторной работы). Блок самого верхнего уровня модели всегда нумеруется нулем.

С контекстной диаграммой связывается цель модели и точка зрения.

Декомпозицией контекстной диаграммы (ее диаграммой-потомком) является диаграмма, содержащаяся на рис. 5.2.

Название (поле TITLE IDEF0-бланка) диаграммы декомпозиции совпадает с названием декомпозируемого блока родительской диаграммы.

Для диаграмм, которые представлены на рис. 5.2 и 5.8, таким названием является «Выполнение лабораторной работы».

Название контекстной диаграммы определяется общей функцией моделируемой системы, то есть совпадает с названием блока контекстной диаграммы (см. рис. 5.8).

Рис. 5.8. Контекстная диаграмма процесса выполнения лабораторной работы

Таким образом, две диаграммы IDEF0-модели имеют одно и то же название. Это контекстная диаграмма и ее диаграмма-потомок. Названия всех остальных диаграмм модели уникальны.

Номер узла

Каждая диаграмма модели идентифицируется ***номером узла*** (NODE), расположенным на IDEF0-бланке в левом нижнем углу.

Номер узла для контекстной диаграммы имеет следующий вид: заглавная буква **A** (Activity в функциональных диаграммах), дефис и ноль (A-0, см. рис. 5.8). Номер узла диаграммы, декомпозирующей контекстную диаграмму, – тот же номер узла, но без дефиса (A0, см. рис. 5.2).

Все другие номера узлов образуются посредством добавления к номеру узла родительской диаграммы номера декомпозируемого блока. Например, номер узла родительской диаграммы – A0. Тогда номер узла диаграммы, декомпозирующей первый блок родительской диаграммы, – A01.

Первый ноль при образовании номера узла принято опускать. Таким образом, номер узла запишется в виде A1. При декомпозиции третьего блока родительской диаграммы A1 номер узла диаграммы-потомка будет соответствовать значению A13.

Для связи диаграмм при движении вверх по иерархии модели могут применяться С-номера или номера узлов. После декомпозиции родительского блока на диаграмме-потомке формируется ссылка на родительскую диаграмму. Для этого используется поле «КОНТЕКСТ» (CONTEXT), расположенное в правом верхнем углу IDEF0-бланка. В данном поле маленькими прямоугольниками изображается каждый блок родительской диаграммы (с сохранением их относительного положения), заштриховывается квадратик декомпозированного блока и размещается С-номер или номер узла родительской диаграммы.

Например, на диаграмме-потомке, декомпозирующей третий блок A3 родительской диаграммы A0, заполнение поля «КОНТЕКСТ» будет соответствовать представленному на рис. 5.9.

Связь между диаграммами посредством С-номеров или номеров узлов позволяет осуществлять тщательный контроль за введением новых диаграмм в иерархию модели.

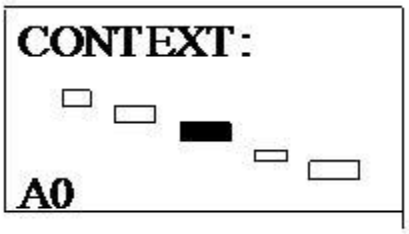


Рис. 5.9. Заполнение поля «КОНТЕКСТ» диаграммы-потомка

Организация связей по дугам между диаграммами

IDEF0-диаграммы имеют ***внешние дуги*** – это дуги, выходящие к краю диаграммы. Эти дуги являются интерфейсом между диаграммой и остальной частью модели. Диаграмма должна быть состыкована со своей родительской диаграммой, то есть внешние дуги должны быть согласованы по числу и наименованию с дугами, касающимися декомпозированного блока родительской диаграммы. Последние называются ***граничными дугами***.

В IDEF0 принята система обозначений, позволяющая авторам модели точно идентифицировать и проверять связи по дугам между диаграммами. Эта схема кодирования дуг называется ***ICOM*** (Input-Conrol-Output-Mechanism).

Правила стыковки и обозначения внешних дуг диаграммы-потомка с граничными дугами родительского блока могут быть сформулированы следующим образом:

- зрительно соединяется каждая внешняя дуга диаграммы-потомка с соответствующей граничной дугой родительского блока;
- каждой зрительной связи присваивается код (**I** – для входных дуг, **C** – для связей между дугами управления, **O** – для связей между выходными дугами, **M** – для связей между дугами механизма, см. рис. 5.2);
- после каждой буквы добавляется цифра, соответствующая положению данной дуги среди других дуг того же типа, касающихся родительского блока. Входные и

выходные дуги пересчитываются сверху вниз, а дуги управлений и механизмов – слева направо (в том порядке, как они расположены на родительской диаграмме по отношению к родительскому блоку). Например, внешние дуги М1, М2 (см. рис. 5.2) пронумерованы в соответствии с расположением граничных дуг на родительском блоке (см. рис. 5.8).

Тоннельные дуги

Особые ситуации возникают, когда дуги «входят в тоннель» между диаграммами. ***Дуга «входит в тоннель»*** в следующих случаях:

1) она является внешней дугой, которая отсутствует на родительской диаграмме (дуга имеет *скрытый источник*);

2) она касается родительского блока, но не появляется на диаграмме, которая его декомпозирует (дуга имеет *скрытый приемник*).

Тоннельные дуги от скрытого источника начинаются круглыми скобками, чтобы указать, что эти дуги идут из какой-то другой части модели, прямо извне модели или они не важны для родительской диаграммы и поэтому на ней не изображаются. Например, дуга механизма «Лаборант» является тоннельной дугой со скрытым источником (см. рис. 5.2). Данная дуга отсутствует на родительской диаграмме (см. рис. 5.8), поскольку для родительской диаграммы она является маловажной.

Тоннельные дуги со скрытым приемником заканчиваются круглыми скобками, чтобы отразить тот факт, что такая дуга идет к какой-то другой части модели, выходит из нее или не будет более в этой модели рассматриваться. Тоннельные дуги со скрытым приемником часто используются в том случае, если они должны связываться с каждым блоком диаграммы-потомка. Изображение таких дуг может привести к существенному загромождению данной диаграммы и ее потомков. Например, дуги «Предварительные знания» и «Порядок выполнения работы» (см. рис. 5.8) должны связываться с каждым блоком диаграммы декомпозиции. Их изображение на диаграмме-потомке является малоинформативным. Поэтому данные дуги реализованы в виде тоннельных дуг со скрытым приемником и на диаграмме декомпозиции (см. рис. 5.2) не показаны.

Таким образом, «вхождение в тоннель» для дуг используется чаще всего для упрощения описания системы – тогда, когда диаграммы в модели становятся слишком

сложными для чтения и понимания.

Диаграмма дерева узлов

Разработанная IDEF0-модель со всеми уровнями структурной декомпозиции может быть представлена в виде единственной диаграммы дерева узлов (рис. 5.10). На данном виде диаграмм представляется иерархия функций в модели без указания взаимосвязей (дуг) между функциями. Для изображения этого дерева нет стандартного формата. Единственное требование состоит в том, чтобы вся иерархия узлов модели была представлена наглядно [1, 8, 9, 10, 13, 21].

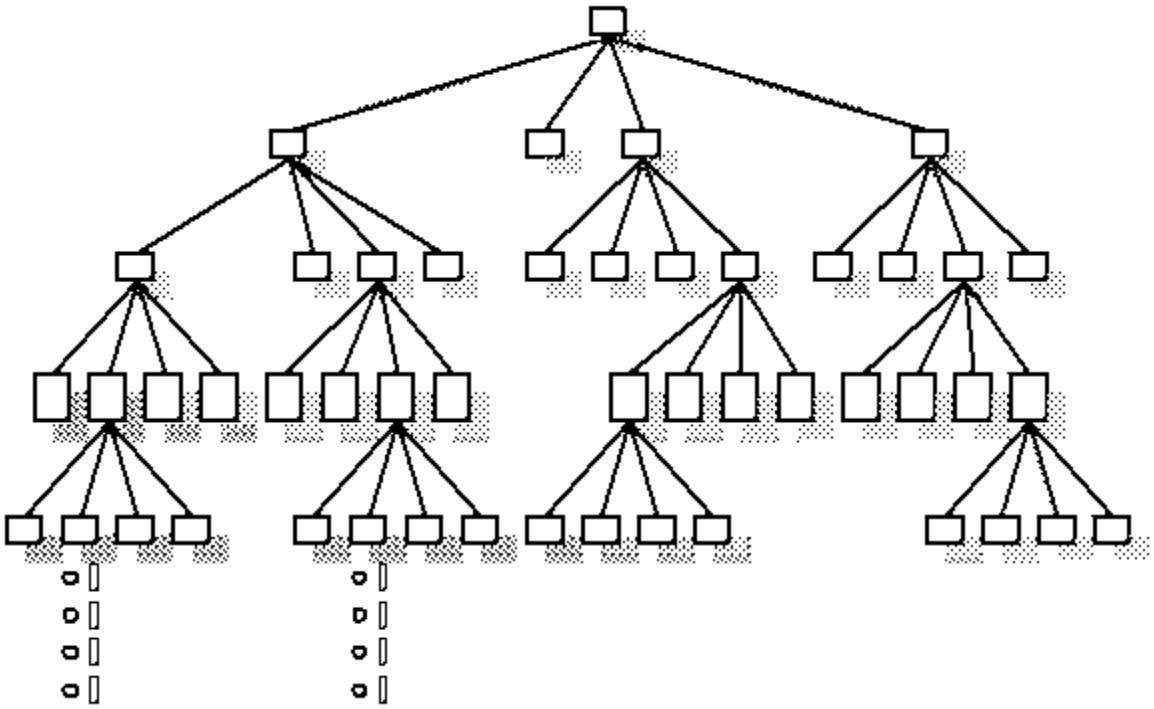


Рис. 5.10. Пример диаграммы дерева узлов

Резюме

IDEF0-диаграммы являются декомпозициями ограниченных субъектов. Субъект ограничивается блоком и касающимися его дугами. Диаграмма, содержащая границу, называется родительской диаграммой. Диаграмма, декомпозирующая блок родительской диаграммы, называется диаграммой-потомком. Для связывания родительской диаграммы и диаграммы-потомка используются С-номера, номера узлов и коды ICOM. Номер узла

идентифицирует уровень диаграммы в иерархии модели. Для упрощения описания системы используется специальный технический прием «вхождение дуг в тоннель». IDEF0-модель может быть представлена в виде единственной диаграммы дерева узлов.

5.2.5. Декомпозиция и её стратегии при IDEF0-моделировании

При создании IDEF0-модели предметной области используется *метод декомпозиции ограниченного субъекта* [9, 22].

Как уже отмечалось, декомпозиция – это процесс создания диаграммы, детализирующей определенный блок и связанные с ним дуги.

IDEF0-декомпозиция включает в себя *анализ* (начальное разделение элемента на более мелкие части) и *синтез* (последующее их соединение для смыслового описания элемента).

Следуя методу декомпозиции ограниченного субъекта, автор производит *вначале* анализ и синтез системных объектов (напомним, что под объектом в общем случае понимаются некоторые данные, изображаемые на модели дугами). Список данных начинается со всех граничных дуг и их ICOM-кодов с последующей их детализацией на составляющие. После этого некоторые составляющие могут быть объединены для смыслового выделения объектов (например объектов, которые будут выступать в качестве управляющих).

Затем автор выполняет подобный анализ и синтез функций системы, делая это в соответствии со списком данных. В процессе объединения и введения новых управляющих дуг создается список функций для дальнейшей детализации. Эти функциональные части объединяются в наборы из трех–шести блоков.

После определения функциональных частей список данных и список функций используются для чернового варианта диаграммы. Снова выполняются анализ и синтез, в результате чего формируются наборы объектов, которые представляются дугами, соединяющими блоки.

Такая последовательность выполнения декомпозиции имеет большое значение, поскольку в методологии IDEF0 анализ объектов системы оказывает важнейшее влияние на анализ функций.

В процессе создания диаграмм опытный разработчик модели постоянно следит за стратегией декомпозиции и ее влиянием на качество модели. При создании IDEF0-модели наиболее часто используются следующие ***стратегии декомпозиции***.

1. *Функциональная стратегия*. Базируется на функциональных взаимоотношениях в системе. Рекомендуется следовать этой стратегии всегда, когда это возможно.

2. *Декомпозиция в соответствии с функциями, выполняемыми людьми или организациями*. Рекомендуется использовать эту стратегию только в начале работы (так как позже взаимосвязи между исполнителями могут быть очень сложны) над моделью системы, относящейся к разряду P3 (people – люди, paper – бумага, procedures – процедуры). Это помогает собрать исходную информацию о системе. Затем следует применять более обоснованную функциональную декомпозицию в соответствии с первой стратегией.

3. *Декомпозиция в соответствии с уже известными стабильными подсистемами*. Это приводит к созданию набора моделей – по одной модели на каждую подсистему или важный компонент. Затем для описания всей системы строится составная модель, объединяющая все отдельные модели. Стратегия эффективна для систем команд и управления, когда разделение на основные части системы не меняется. Например, создается модель отдельно для торпеды, отдельно для защиты от торпед и отдельно для движения подводной лодки. Затем эти модели объединяются вместе для описания способов защиты подводной лодки от торпед.

4. *Декомпозиция, основанная на отслеживании «жизненного цикла» для ключевых входов системы*. Эффективна для моделирования систем, непрерывно преобразующих свои входы в конечный продукт (например система очистки нефти). Декомпозиция осуществляется в соответствии с этапами преобразования входа (этапами ЖЦ).

5. *Декомпозиция по физическому процессу*, основанная на выделении функциональных стадий, этапов завершения, шагов выполнения и т.п. Результатом стратегии часто является последовательное описание системы, не учитывающее ограничения, накладываемые функциями друг на друга. Поэтому эту стратегию рекомендуется использовать в случаях, если целью модели является описание физического процесса или если разработчик в начале создания IDEF0-модели не понимает, как действовать. В последнем случае по мере накопления информации о

моделируемой предметной области следует перейти на применение более обоснованной функциональной декомпозиции в соответствии с первой стратегией.

Резюме

Для построения модели используется метод декомпозиции ограниченного субъекта. Декомпозиция включает этапы анализа и синтеза. Существуют различные стратегии декомпозиции. Их выбор и получение диаграмм высокого качества часто требуют многих итераций и изменений. Декомпозиция прекращается, когда модель достаточно точна, чтобы отвечать на вопросы, составляющие ее цель.

5.2.6. Процесс моделирования в IDEF0

Процесс моделирования в IDEF0 включает сбор информации об исследуемой области, документирование полученной информации с представлением ее в виде модели и уточнение модели посредством итеративного рецензирования. На рис. 5.11 изображен процесс моделирования в IDEF0, описанный с помощью IDEF0-диаграммы [9, 22].

Процесс моделирования в IDEF0 является итерационным. Это позволяет получить точное описание системы или программного средства. Основой процесса IDEF0-моделирования является разделение функций, выполняемых участниками IDEF0-проектов (см. входы механизмов, рис. 5.11).

Разделение ролей участников проектов обеспечивает поддержку организации коллективной работы в IDEF0. Выделяются следующие роли участников проектов:

- *эксперты* – специалисты в предметной области, являющиеся источниками информации;
- *авторы* – разработчики диаграмм и моделей;
- *библиотекарь* – координатор своевременного обмена письменной информацией между участниками проекта;
- *читатели* – специалисты в предметной области, которые рецензируют модели;
- *комитет технического контроля* – группа специалистов, принимающих и утверждающих модель.

Как видно из рис. 5.11, процесс IDEF0-моделирования состоит из *пяти этапов*.

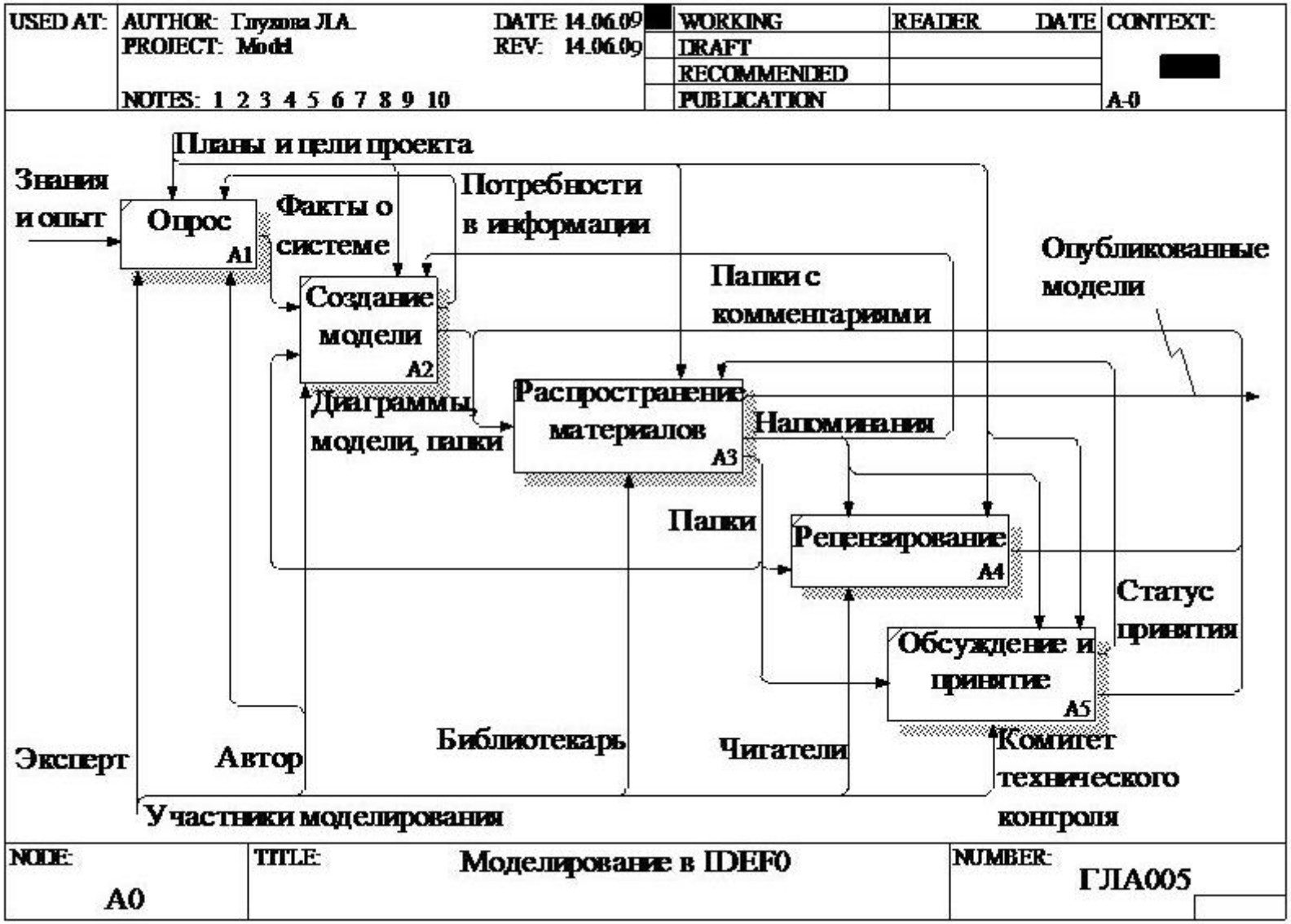


Рис. 5.11. Процесс моделирования в IDEF0

Целью *первого этапа* IDEF0-моделирования (блок А1 «Опрос» на рис. 5.11) является получение автором знаний о моделируемой системе (предметной области). Для этого могут быть использованы различные источники информации: собственные знания и опыт автора, изучение документов, опрос экспертов, наблюдение за работой системы и т. п. Результатом данного этапа являются собранные факты о моделируемой системе.

Вторым этапом моделирования является создание модели (блок А2 на рис. 5.11). На данном этапе полученные на предыдущем этапе факты о системе автор представляет в виде одной или нескольких IDEF0-диаграмм. Процесс создания модели осуществляется с помощью *метода декомпозиции ограниченного субъекта.* При его использовании автор модели *вначале* анализирует *объекты* (информацию, данные, механизмы и т.п.), входящие в систему, а *затем* использует полученные знания для анализа *функций* системы (см. п. 5.2.5). На основе этого анализа создается диаграмма, в которой объединяются сходные объекты и функции. Этот путь проведения анализа системы и документирования его результатов является уникальной особенностью методологии IDEF0.

Из рис. 5.11 видно, что этапы «Опрос» и «Создание модели» выполняются многократно (см. обратную связь по управлению «Потребности в информации»). Таким образом, создающиеся IDEF0-модели проходят через серию последовательных улучшений до тех пор, пока они в точности не будут представлять реальную предметную область. Созданные диаграммы и модели оформляются в виде *папок* – небольших пакетов с результатами работы.

Третий этап – «Распространение материалов» (блок А3) предназначен для координации передачи материалов проекта его участникам. Папки, сформированные на этапе «Создание модели» и содержащие некоторую часть работы над моделью, распространяются с целью получения отзыва от читателей.

Для эффективного моделирования важнейшее значение имеет организация своевременной обратной связи между участниками IDEF0-проекта, так как устаревшая информация способна свести на нет все усилия по разработке модели. Поэтому IDEF0-методология выделяет специальную роль *наблюдателя за процессом рецензирования*. Эту роль выполняет так называемый *библиотекарь*, который является главным координатором процесса моделирования в IDEF0. Он обеспечивает своевременное и

согласованное распространение рабочих материалов, контролирует их движение.

На *четвертом этапе* – «Рецензирование» (блок А4) выполняется критическое обсуждение специалистами в предметной области текущих результатов моделирования, представленных в папках. Результаты обсуждаются в течение определенного времени. Сделанные замечания помещаются в папку в виде нумерованных комментариев. К определенному сроку замечания поступают к библиотекарю, а от него – к автору. Автор отвечает на каждое замечание и обобщает критику, содержащуюся в замечаниях.

Этапы «Создание модели» (блок А2), «Распространение материалов» (блок А3) и «Рецензирование» (блок А4, см. рис. 5.11) составляют так называемый *цикл автор/читатель*. Данный цикл является одним из основных компонентов методологии IDEF0. В его ходе читателями выполняется многократное рецензирование достоверности текущих результатов моделирования, на основе которого авторы уточняют создаваемые модели.

Целями пятого этапа «Обсуждение и принятие» (блок А5) являются: оценка того, что создаваемая в процессе анализа модель будет точна и используема в дальнейшем; контроль качества модели; оценка соответствия выполняемой работы конечным целям всего проекта. За выполнение данного этапа отвечает *Комитет технического контроля*. Если модель признана Комитетом применимой, она публикуется. В противном случае авторам направляются замечания для необходимой доработки.

Таким образом, методология IDEF0 поддерживает как асинхронный, так и параллельный просмотры модели. Такая организация процесса является наиболее эффективным способом распределения работы в коллективе.

На практике над различными частями модели работает совместно несколько авторов, так как каждый функциональный блок модели представляет отдельный компонент, который может быть независимо проанализирован и декомпозирован.

В настоящее время существует ряд CASE-средств, поддерживающих методологию IDEF0. Среди недорогих и доступных на нашем рынке инструментальных средств следует отметить CASE-средство AllFusion Process Modeler, известное ранее под названием BPwin. Данное CASE-средство входит в линейку интегрированных CASE-средств CA ERwin Modeling Suite (AllFusion Modeling Suite) компании Computer Associates (см. подразд. 6.6).

Резюме

Процесс IDEF0-моделирования может быть разделен на несколько этапов: опрос экспертов, создание диаграмм и моделей, распространение документации, рецензирование, принятие диаграмм и моделей. Каждый из исполнителей проекта выполняет конкретные обязанности. Среди современных CASE-средств, поддерживающих технологию IDEF0-моделирования, наиболее популярно CASE-средство AllFusion Process Modeler.

5.3. Методология структурного анализа потоков данных DFD

Методология структурного анализа потоков данных ***DFD*** (Data Flow Diagrams) основана на методах, ориентированных на потоки данных (методах Йодана, ДеМарко, Гейна, Сарсона). Существуют различные графические нотации данной методологии. Наиболее известными из них являются нотация, предложенная Гейном и Сарсоном (так называемый метод Гейна–Сарсона), и нотация, предложенная Йоданом и ДеМарко (метод Йодана–ДеМарко) [19, 28]. Пример DFD-модели в нотации Йодана–ДеМарко представлен на рис. 4.2. В данном подразделе рассматривается методология DFD в нотации Гейна–Сарсона.

5.3.1. Основные понятия DFD-модели

Методология DFD является одной из методологий функционального моделирования предметной области, поэтому она имеет много общего с методологией IDEF0.

DFD-методология выделяет функции (действия, события, работы) системы. Функции соединяются между собой с помощью потоков данных (объектов). Функции на диаграммах представляются функциональными блоками, потоки данных – дугами.

Аналогично IDEF0-методологии DFD-модель должна иметь единственные цель, точку зрения, субъект и точно определенные границы (см. п. 5.2.2).

Однако если в IDEF0 дуги имеют различные типы и определяют отношения между

блоками (см. п. 5.2.3), то в DFD дуги отражают реальное перемещение объектов от одной функции к другой.

Помимо блоков, представляющих собой функции, на DFD-диаграммах используются два типа блоков – хранилища данных и внешние сущности. Данные блоки отражают взаимодействие с частями предметной области, выходящими за границы моделирования.

Резюме

Методология DFD является одной из методологий функционального моделирования предметной области. DFD-модель должна иметь единственные цель, точку зрения, субъект и точно определенные границы. DFD-модель отражает перемещение объектов, их хранение, обработку, внешние источники и потребителей данных.

5.3.2. Синтаксис DFD-диаграмм

Диаграммы являются основными рабочими элементами DFD-модели. Диаграммы отражают перемещение данных, их обработку и хранение. Каждая DFD-диаграмма содержит функциональные блоки и дуги (линии со стрелками). DFD-диаграмма может содержать хранилища данных и внешние сущности.

Функциональный блок отражает некоторую функцию моделируемой системы, преобразующую некоторые входные данные (сырье, материалы, информацию и т.п.) в выходные результаты. Функциональный блок изображается прямоугольником с закругленными углами (рис. 5.12). Все стороны функционального блока в отличие от IDEF0 равнозначны.

Функциональные блоки на диаграмме нумеруются. Номер функционального блока отмечается в его правом верхнем углу с возможным использованием префикса **A** (Activity – работа) перед ним.

Как и в IDEF0-методологии, название функционального блока основывается на использовании отглагольного существительного, обозначающего действие (вычисление того-то, определение того-то, обработка того-то и т.д.) или на использовании глагола в неопределенной форме (вычислить то-то, определить то-то, обработать то-то).

Рис. 5.12. Функциональный блок DFD

Хранилище данных отражает временное хранение промежуточных результатов обработки. Внешний вид блока, представляющего хранилище данных, иллюстрирует рис. 5.13. Название хранилища базируется на использовании существительного.

Хранилища данных на диаграмме нумеруются. Номер хранилища данных записывается слева с возможным префиксом **D** (Data store) перед ним.

Рис. 5.13. Хранилище данных

Внешние сущности являются источниками данных для входов модели и приемниками данных для ее выходов. Внешняя сущность может быть одновременно источником и приемником данных.

Внешние сущности изображаются в соответствии с рис. 5.14 и размещаются, как правило, по краям диаграмм. Название внешней сущности базируется на использовании существительного. Номер внешней сущности записывается в левом верхнем углу с возможным префиксом **E** (External) перед ним.

Одна и та же внешняя сущность (с одним и тем же номером) может быть размещена в нескольких местах диаграммы. Это позволяет в ряде случаев существенно снизить загроможденность диаграмм длинными дугами.

Дуги обозначают передвижение данных в моделируемой системе. С учетом равнозначности сторон блоков диаграммы дуги могут начинаться и заканчиваться на любой их стороне.

Рис. 5.14. Внешняя сущность

В общем случае дуга представляет множество объектов (планы, машины, информация и т.п.). Основу названия дуги на DFD-диаграммах составляют существительные. Названия дуг называются *метками.*

Дуги на DFD-диаграмме изображаются линиями со стрелками. На DFD-диаграммах могут использоваться следующие типы дуг:

- однонаправленные сплошные – отражают направление потоков объектов (данных);
- двунаправленные сплошные – обозначают обмен данными между блоками;
- однонаправленные штриховые – обозначают управляющие потоки между блоками.

Как и в IDEF0-методологии, дуги могут разветвляться и соединяться. Синтаксис и семантика разветвления и слияния дуг соответствуют описанным в п. 5.2.3 для методологии IDEF0.

На рис. 5.15 приведен пример DFD-диаграммы процесса выполнения лабораторной работы. Данный пример соответствует предметной области моделирования, подробно рассмотренной в пп. 5.2.2 – 5.2.4 при изучении методологии IDEF0.

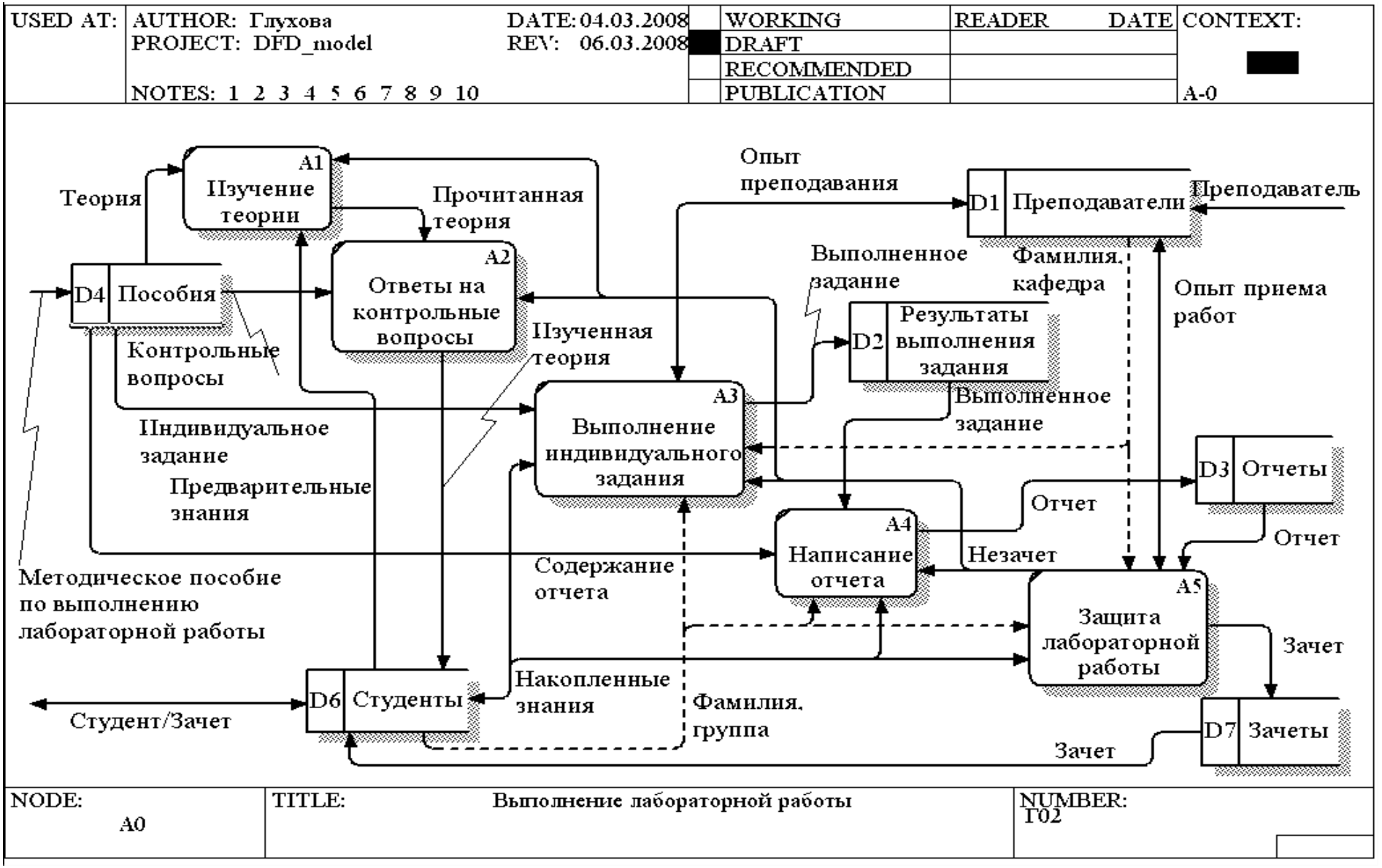


Рис. 5.15. DFD-диаграмма декомпозиции процесса выполнения лабораторной работы

Резюме

DFD-диаграммы отражают перемещение данных, их обработку и хранение. DFD-диаграмма содержит функциональные блоки и дуги, может содержать хранилища данных и внешние сущности. Функциональный блок отражает некоторую функцию моделируемой системы. Хранилище данных отражает временное хранение промежуточных результатов обработки. Внешние сущности являются источниками данных для входов модели и приемниками данных для ее выходов. Дуги могут представлять однонаправленные и двунаправленные потоки данных и управляющие потоки.

5.3.3. Синтаксис DFD-моделей

По аналогии с IDEF0-моделью DFD-модель представляет собой иерархически организованную совокупность диаграмм. Каждый из функциональных блоков диаграммы может быть декомпозирован на другой диаграмме. Функциональный блок и касающиеся его дуги определяют границу диаграммы, представляющей декомпозицию этого блока. Эта диаграмма называется ***диаграммой-потомком***. Декомпозируемый блок называется ***родительским блоком***, а содержащая его диаграмма – ***родительской диаграммой***. Название диаграммы-потомка совпадает с функцией родительского блока.

Один функциональный блок и несколько дуг на самом верхнем уровне модели используются для определения границы всей системы. Этот блок описывает общую функцию, выполняемую системой. Диаграмма, определяющая границу системы и состоящая из одного функционального блока, его дуг и внешних сущностей, называется ***контекстной диаграммой DFD-модели***. Все, что лежит внутри функционального блока, является частью описываемой системы. Внешние сущности определяют *среду системы*, то есть внешние источники и приемники объектов (данных) системы.

Рис. 5.16 представляет контекстную DFD-диаграмму процесса выполнения лабораторной работы. В отличие от контекстной IDEF0-диаграммы (см. рис. 5.8) контекстная DFD-диаграмма содержит внешние сущности «Кафедра» «Студенты», «Методические пособия» и «Правила обучения». В остальном синтаксис этих диаграмм совпадает (см. п. 5.2.4).

Общая функция DFD-модели записывается на контекстной диаграмме в виде

названия функционального блока и совпадает с именем данной диаграммы. С контекстной диаграммой связывается цель модели и точка зрения.

Декомпозицией контекстной диаграммы, представленной на рис. 5.16, является диаграмма, содержащаяся на рис. 5.15.

Как и в IDEF0-методологии, дуги в DFD-моделях могут «входить в тоннель» между диаграммами (см. дуги «Лаборант» и «Порядок выполнения работы» на рис. 5.16). Синтаксис и семантика тоннелирования дуг соответствуют описанному в п. 5.2.4 для методологии IDEF0.

Разработанная DFD-модель со всеми уровнями структурной декомпозиции может быть представлена в виде диаграммы дерева узлов (рис. 5.17). На данном виде диаграмм представляется иерархия функций в модели без указания потоков данных между ними. Формат дерева узлов может быть различным.

DFD-моделирование поддерживается рядом CASE-средств, например, CASE-средством AllFusion Process Modeler (Bpwin, см. подразд. 6.6) [21, 28].

Резюме

DFD-модель представляет собой иерархически организованную совокупность диаграмм. Диаграмма на верхнем уровне иерархии, состоящая из одного функционального блока, его дуг и внешних сущностей, называется контекстной диаграммой DFD-модели. Все, что лежит внутри функционального блока, является частью моделируемой системы. Внешние сущности определяют среду системы. Иерархию DFD-модели представляет диаграмма дерева узлов.

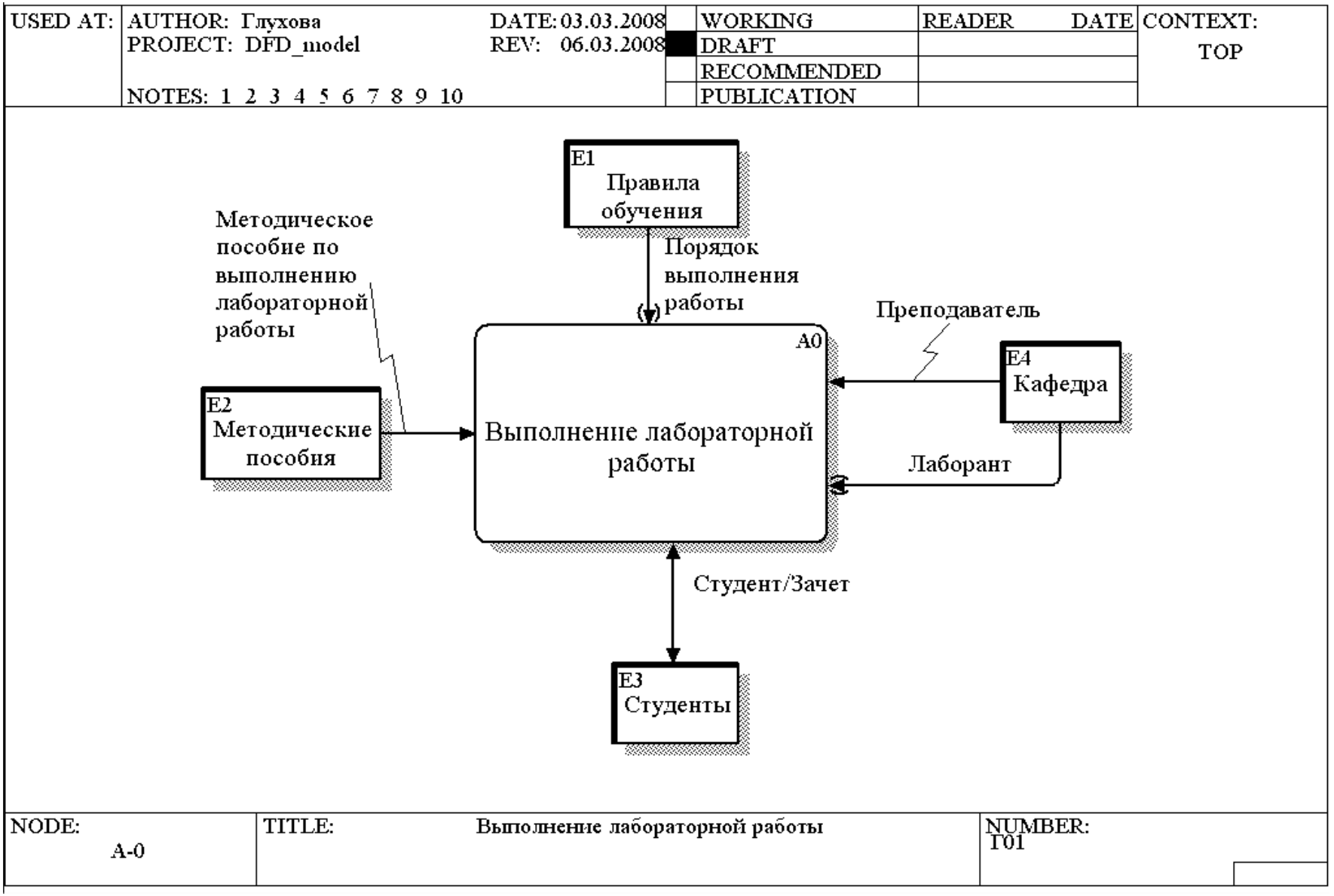


Рис. 5.16. Контекстная DFD-диаграмма процесса выполнения лабораторной работы

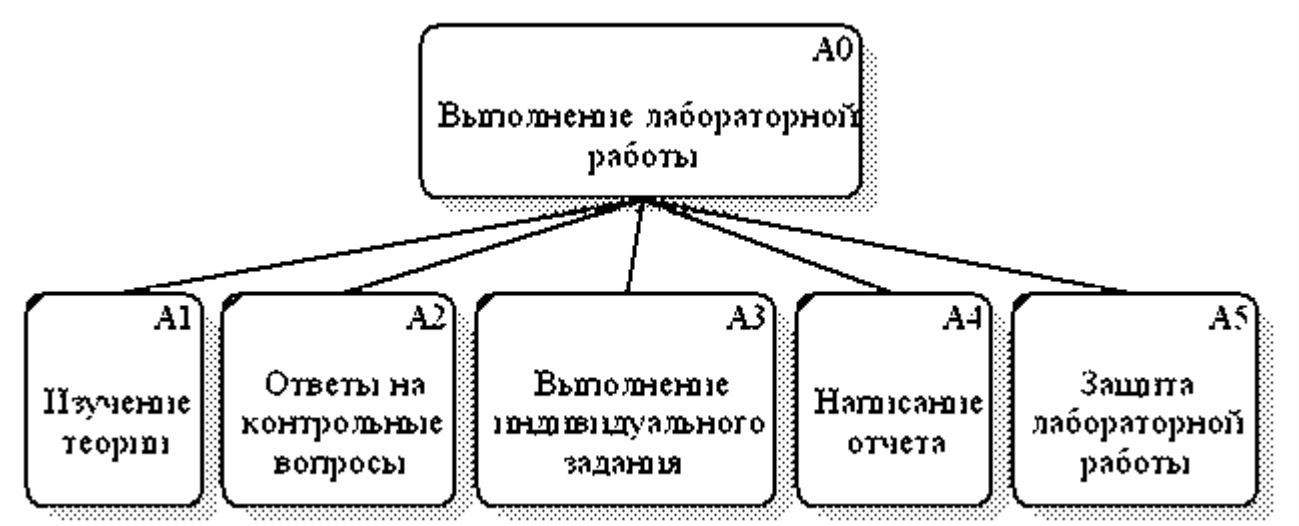


Рис. 5.17. DFD-диаграмма дерева узлов

5.4. Методология информационного моделирования IDEF1X

5.4.1. Основные понятия и определения

Методология информационного моделирования IDEF1X известна также под названиями методологии семантического моделирования данных и методологии концептуального моделирования. Основные положения данной методологии регламентированы международным стандартом *ISO/IEC/IEEE 31320-2:2012*.

Под ***информационной моделью (моделью данных)*** подразумевается графическое и текстовое представление результатов анализа предметной области, которое идентифицирует данные, используемые в организации для достижения своих целей, функций, задач, потребностей и стратегий, а также для управления организацией или ее оценки [2, 9].

Целью информационного моделирования (моделирования данных) является идентификация сущностей, составляющих предметную область, и связей между ними. *Результатом информационного моделирования* является информационная модель

предметной области, содержащая сущности, их атрибуты и отражающая взаимосвязи между сущностями.

Наиболее часто информационное моделирование используется при проектировании баз данных.

Методология IDEF1X разработана на основе диаграмм «Сущность–Связь» Чена.

Компонентами IDEF1X являются:

- сущности (Entities); подразделяются на два вида:
  - независимые сущности (Identifier-Independent Entities);
  - зависимые сущности (Identifier-Dependent Entities);
- связи (Relationships); подразделяются на четыре вида:
  - идентифицирующие соединительные связи (Identifying Connection Relationships);
  - неидентифицирующие соединительные связи (Non-Identifying Connection Relationships);
  - связи категоризации (Categorization Relationships);
  - неспецифические связи (Non-Specific Relationships);
- атрибуты/ключи (Attributes/Keys); подразделяются на четыре вида:
  - атрибуты (Attributes);
  - первичные ключи (Primary Keys);
  - альтернативные ключи (Alternate Keys);
  - внешние ключи (Foreign Keys);
- текстовые комментарии (Notes).

Резюме

Информационная модель выделяет данные предметной области и используется при проектировании баз данных. В соответствии со стандартом IDEF1X компонентами информационной модели являются сущности, их атрибуты, взаимосвязи между сущностями и текстовые комментарии.

5.4.2. Сущности

Под ***сущностью*** в информационном моделировании подразумевается представление множества реальных или абстрактных объектов предметной области

(людей, предметов, мест, идей, событий), для которого [2, 9]:

1) все элементы множества (***экземпляры***) имеют одни и те же характеристики;

2) все экземпляры подчинены одному и тому же набору правил и линий поведения и участвуют в одних и тех же связях.

Сущности подразделяются на независимые и зависимые. Сущность называется ***независимой***, если каждый экземпляр данной сущности может быть уникально идентифицирован независимо от ее связей с другими сущностями. Сущность называется ***зависимой***, если уникальная идентификация его экземпляров зависит от связи данной сущности с другими сущностями.

Каждая сущность в информационной модели должна иметь уникальное ***имя***, основанное на использовании существительного. Существительное должно быть представлено в единственном числе. Примеры имен сущностей: Человек, Дом, Студент (но не Люди, Дома, Студенты). Если имя сущности состоит из нескольких слов, они соединяются дефисом или символом подчеркивания, например, Категория-предприятий, Категория_предприятий.

Кроме того, в большой модели для организации документации сущности должны быть пронумерованы. Номер сущности записывается за именем и отделяется от последнего символом «/», например, Студент/21.

Большинство сущностей относится к следующим ***категориям*** [9]:

- реальные объекты;
- роли;
- инциденты;
- взаимодействия;
- спецификации.

Реальные объекты – это представление фактических предметов в физическом мире. Например, к сущностям данной категории относятся сущности Завод, Университет, Аэропорт, Банк.

Роли – это представление цели или назначения человека, оборудования или организации. Например, для университета сущностями-ролями являются Преподаватель, Лаборант и Студент; для магазина – Покупатель, Продавец, Кассир.

Инциденты – это представление какого-либо события. Примерами сущностей-

инцидентов могут являться сущности Землетрясение, Запуск-космического-корабля, Выборы.

Взаимодействия – сущности, получаемые из отношений между двумя сущностями. Примерами сущностей-взаимодействий являются сущности Перекресток (место пересечения улиц), Контракт (соглашение между сторонами), Соединение (место соединения некоторой детали с другой).

Спецификации – сущности, используемые для представления правил, стандартов, требований, критериев качества и т.п. Примерами сущностей-спецификаций являются сущности Рецепт (правило приготовления порции пищи), Метрика-качества (метод и шкала измерения некоторого свойства программного средства).

Каждая сущность должна сопровождаться описанием. ***Описание*** – это короткое информативное утверждение, которое позволяет установить, является ли некоторый элемент экземпляром сущности или нет.

Например, для сущности Студент описание может выглядеть следующим образом: «Человек, учащийся в ВУЗе».

Резюме

Сущность в информационном моделировании представляет множество реальных или абстрактных объектов с одинаковыми характеристиками, подчиняющихся одному набору правил и линий поведения и участвующих в одних и тех же связях. Сущности подразделяются на независимые и зависимые. Имя сущности базируется на использовании существительного. Существуют следующие категории сущностей: реальные объекты, роли, инциденты, взаимодействия, спецификации.

5.4.3. Атрибуты

Все реальные или абстрактные объекты в мире имеют некоторые характеристики (например высота, температура, возраст, координаты и т.п.).

Атрибут – это образ характеристики или свойства, которым обладают все экземпляры сущности. Каждый атрибут обеспечивается именем, уникальным в пределах сущности и основанным на использовании существительного. Существительное должно быть представлено в единственном числе [2, 9].

Примеры имен атрибутов: Адрес, Возраст, Фамилия (но не Адреса, Возрасты, Фамилии). Если имя атрибута состоит из нескольких слов, они соединяются дефисом или символом подчеркивания, например, Дата-рождения или Дата_рождения. Для обеспечения уникальности атрибута в пределах модели используется составное имя:

<Имя-сущности>**.**<Имя-атрибута>

Например, для сущности Студент обращение к его атрибуту Фамилия имеет вид

Студент**.**Фамилия

Для определенного экземпляра сущности атрибут принимает конкретное значение. Диапазон допустимых значений, которые атрибут может принимать, называется ***доменом***. Домен должен определяться для каждого атрибута.

При информационном моделировании атрибуты принято подразделять на *указывающие, описательные* и *вспомогательные*.

Указывающие атрибуты используются для присвоения имени или обозначения экземплярам сущности. Например, Счет.Номер; Студент.Фамилия.

Если значение указывающего атрибута изменяется, то это говорит о том, что изменился экземпляр сущности.

Указывающие атрибуты часто используются как идентификатор или часть идентификатора.

Идентификатор – это атрибут или совокупность нескольких атрибутов, значения которых однозначно определяют каждый экземпляр сущности. Идентификатор, состоящий из нескольких атрибутов, называется ***составным***. Идентификаторы называются также ***первичными ключами (primary keys)***.

Например, для сущности Студент атрибут Фамилия является удовлетворительным идентификатором, если в университете нет однофамильцев. В общем случае идентификатор сущности Студент будет состоять из трех атрибутов (Фамилия, Имя, Отчество), а возможно, и более (например, при наличии полных однофамильцев могут быть добавлены атрибуты Домашний-адрес, Номер-группы или Дата-рождения). Следует отметить, что обработка составных идентификаторов является достаточно сложной и без необходимости их применения нужно избегать.

Сущность может иметь несколько альтернативных идентификаторов. Например,

для сущности Аэропорт атрибут Код-аэропорта является идентификатором. Комбинация атрибутов Долгота и Широта является другим идентификатором сущности Аэропорт.

Если сущность имеет несколько альтернативных идентификаторов, один из них выбирается как ***привилегированный*** (первичный ключ). Остальные называются ***альтернативными ключами***.

Для упрощения структуры информационной модели и облегчения работы с ней рекомендуется в качестве идентификатора использовать ***идентификационный номер экземпляра сущности*** (***ID***). Это позволяет исключить необходимость обработки идентификаторов, состоящих из нескольких атрибутов. Наиболее эффективно использование идентификационных номеров целочисленного типа. Значения ID изменяются по порядку, начиная с единицы.

Описательные атрибуты представляют характеристики, внутренне присущие каждому экземпляру сущности.

Примерами описательных атрибутов являются Студент.Домашний-адрес; Собака.Вес; Книга.Название-главы.

Если значение описательного атрибута изменяется, то это говорит о том, что некоторая характеристика экземпляра изменилась, но сам экземпляр остался прежним.

В некоторых случаях описательные атрибуты могут включаться в состав составного идентификатора. Так, в рассмотренном выше примере для сущности Студент в состав идентификатора при наличии полных однофамильцев помимо указывающих атрибутов Фамилия, Имя, Отчество следует ввести некоторый описательный атрибут, например, Домашний-адрес.

Однако в общем случае описательные атрибуты идентификаторами не являются. Такие атрибуты называются ***вторичными ключами*** или ***неключевыми атрибутами***. Например, для сущности Аэропорт вторичным ключом является атрибут Тип-аэропорта, поскольку может существовать достаточно большое количество аэропортов одного типа (военных, гражданских и т.п.).

Вспомогательные атрибуты используются для связи экземпляра одной сущности с экземпляром другой. Вспомогательные атрибуты называются также ***внешними ключами (foreign keys)*** или ***мигрирующими ключами***.

Если значение вспомогательного атрибута изменяется, то это говорит о том, что другие экземпляры сущностей связываются между собой.

Например, атрибут Собака.Имя-хозяина обозначает человека, которому принадлежит собака; атрибут Счет.Идентификатор-клиента указывает идентификатор клиента, владеющего данным счетом.

Резюме

Атрибут в информационном моделировании представляет образ характеристики или свойства, которым обладают все экземпляры сущности. Имя атрибута базируется на использовании существительного. Существуют описательные, указывающие и вспомогательные атрибуты. Идентификатор – это атрибут или совокупность атрибутов, однозначно определяющих каждый экземпляр сущности. Для каждого атрибута должен быть определен диапазон допустимых значений, называемый доменом.

5.4.4. Способы представления сущностей с атрибутами

При информационном моделировании сущности с атрибутами могут быть представлены различными способами.

1. Графический способ

Графический способ используется в IDEF1X-моделировании [2, 9]. При этом независимая сущность (см. п. 5.4.9) изображается прямоугольником, а зависимая сущность – прямоугольником с закругленными углами (рис. 5.18). Имя сущности и ее номер записываются над прямоугольником.

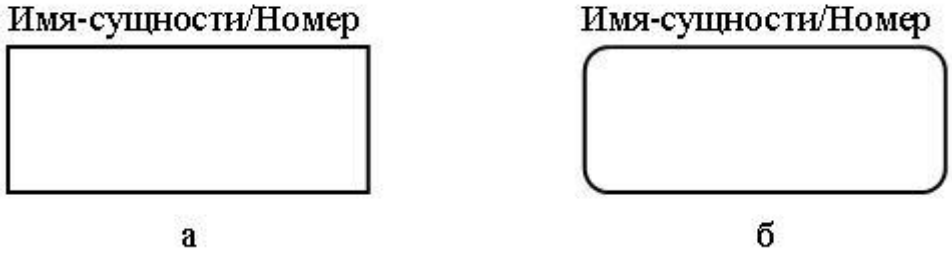


Рис. 5.18. Представление сущности: а – независимой; б – зависимой

Внутри прямоугольника записываются имена атрибутов. Атрибуты, составляющие привилегированный идентификатор сущности, записываются первыми среди атрибутов и отделяются от остальных чертой (атрибуты Фамилия, Имя, Отчество и атрибут ID-студента на рис. 5.19).

Рис. 5.19. Графическое представление сущности Студент:
а – с использованием составного идентификатора;
б – с использованием идентификационного номера

2. Текстовый способ

При текстовом способе представления сущность описывается с помощью указания ее имени, ее номера в модели (если он определен) и заключенного в круглые скобки списка атрибутов. На первом месте в списке атрибутов записываются привилегированные идентификаторы, которые некоторым образом выделяются (например подчеркиваются).

Например, сущность, представленная на рис. 5.19, а, при текстовом способе представления будет описана следующим образом:

Студент/1 (Фамилия, Имя, Отчество, Домашний-адрес, Специальность, Группа, Дата-рождения).

Текстовый способ представления сущностей удобно использовать при описании

информационной модели, например, в документации.

3. Табличный способ

При табличном способе представления сущность в информационной модели интерпретируется как таблица. Каждый экземпляр сущности представляет собой строку в таблице. Строка заполняется значениями атрибутов, соответствующими данному экземпляру.

Например, сущность Студент (см. рис. 5.19, б) при табличном способе представления интерпретируется так, как показано на рис. 5.20. На данном рисунке название таблицы представляет собой имя сущности и ее номер, первая строка таблицы содержит имена атрибутов сущности, остальные строки – значения атрибутов для конкретных экземпляров сущности.

Следует отметить, что при физическом представлении базы данных каждая сущность представляется в виде реальной таблицы.

Студент/1

ID-студента	Фамилия	Имя	Отчество	Домашний-адрес	Специальность	Группа	Дата-рождения
1	Иванов	Иван	Иванович	Ульяновская, 1-9	ПОИТ	951005	12.01.83
2	Сидоров	Петр	Власович	Скорины, 8-16	ПОИТ	051003	17.08.87
…	…	…	…	…	…	…	…

Рис. 5.20. Интерпретация сущности в виде таблицы

Резюме

Существует три основных способа представления сущностей с атрибутами: графический, текстовый, табличный. В методологии IDEF1X используется графический способ. При физическом представлении базы данных каждая сущность представляется в виде таблицы. Каждый экземпляр сущности представляет собой строку в таблице.

5.4.5. Правила атрибутов

Информационное моделирование основано на *реляционной модели данных* – представлении данных в виде отношений между ними. Поэтому сущности и их атрибуты в информационной модели должны удовлетворять требованиям к реляционной модели данных. Одним из основных требований является нормализация данных.

Нормализация – процесс уточнения и перегруппировки атрибутов в сущностях в соответствии с нормальными формами. Нормализация позволяет устранить аномалии в организации данных и сократить объем памяти для их хранения. Известны шесть нормальных форм. На практике чаще всего ограничиваются приведением модели данных к третьей нормальной форме [2, 9].

Первая нормальная форма (First Normal Form, 1NF) – сущность находится в 1NF тогда и только тогда, когда все ее атрибуты содержат только элементарные значения.

Вторая нормальная форма (Second Normal Form, 2NF) – сущность находится в 2NF тогда и только тогда, когда она находится в 1NF и каждый ее неключевой атрибут зависит от всего первичного ключа, а не от его части.

Третья нормальная форма (Third Normal Form, 3NF) – сущность находится в 3NF тогда и только тогда, когда она находится в 2NF и каждый ее неключевой атрибут не зависит от другого неключевого атрибута.

С учетом приведенных нормальных форм в информационной модели должны соблюдаться следующие *правила атрибутов* [9].

Первое правило. Один экземпляр сущности имеет одно единственное значение для каждого атрибута в любой момент времени. Данное правило вытекает из 1NF.

В табличной интерпретации сущности это означает, что должен существовать один и только один элемент данных в каждом пересечении столбца со строкой. Например, если у экземпляра сущности Служащий имеется два телефона, то нельзя одновременно присвоить их номера атрибуту Номер-телефона.

Второе правило. Атрибут не должен содержать никакой внутренней структуры. Данное правило также вытекает из 1NF.

Например, если определен атрибут Дата-рождения, то он считается одной характеристикой и его нельзя разделить на независимые атрибуты Число, Месяц, Год.

Третье правило. Если сущность имеет идентификатор, состоящий из нескольких атрибутов, то каждый атрибут, не являющийся частью идентификатора, представляет собой характеристику всей сущности, а не части его идентификатора. Данное правило вытекает из 2NF.

Например, для сущности

Перемещение-жидкости (ID-источника, ID-приемника, Объем-жидкости)

атрибут Перемещение-жидкости.Объем-жидкости обозначает объем перемещаемой жидкости, а не объем источника или приемника жидкости.

Четвертое правило. Каждый атрибут, не являющийся частью идентификатора, представляет собой характеристику экземпляра, указанного идентификатором, а не характеристику другого атрибута-неидентификатора. Данное правило вытекает из 3NF.

Например, для сущности

Порция (ID-порции, ID-рецепта, Вес, Время-приготовления)

атрибут Порция.Время-приготовления определяет фактическое время приготовления порции, а не время, определяемое рецептом.

Резюме

Информационное моделирование основано на реляционной модели данных. Поэтому сущности и их атрибуты должны удовлетворять требованию нормализация данных. Обычно ограничиваются приведением модели данных к третьей нормальной форме. В информационной модели должны соблюдаться правила атрибутов, вытекающие из нормальных форм.

5.4.6. Связи

Между различными видами существующих в предметной области объектов существуют некоторые отношения и взаимосвязи. В общем случае ***связь*** – это представление набора отношений, которые систематически возникают между различными видами реальных или абстрактных объектов (людей, предметов, мест, идей, событий, их комбинаций) в предметной области.

Применительно к IDEF1X-модели под ***связью*** понимается отношение между двумя сущностями или между экземплярами одной и той же сущности.

Соединительной связью называется связь между родительской и дочерней сущностью. Таким образом, если две сущности связаны соединительной связью, то одна из них является родительской, а вторая – дочерней.

Сущность называется ***дочерней***, если ее экземпляры могут быть связаны с нулем или одним экземпляром другой сущности (родительской). Сущность называется ***родительской***, если ее экземпляры могут быть связаны с любым количеством экземпляров другой сущности (дочерней). При этом каждый экземпляр соединительной связи связывает конкретные экземпляры родительской и дочерней сущностей. Поэтому соединительная связь полностью называется ***специфической*** (конкретной, определенной, specific) ***соединительной связью***.

Например, если две сущности Отец и Ребенок связаны соединительной связью, то сущность Отец является родительской (отец может иметь любое количество детей), а сущность Ребенок – дочерней (ребенок имеет одного отца). При этом конкретный экземпляр сущности Отец (например Сидоров Константин) связан с конкретными экземплярами сущности Ребенок (например Иван и Николай).

Очевидно, что зависимая сущность всегда является дочерней. Независимая сущность по отношению к данной соединительной связи может являться как родительской, так и дочерней.

Графически соединительная связь представляется линией от родительской к дочерней сущности с точкой со стороны дочерней сущности (рис. 5.21).

Рис. 5.21. Графическое представление соединительной связи

Каждой связи в модели присваивается уникальный номер вида R/1, R/2, …, R/i (**R** – Relationship – связь).

Каждой связи присваивается имя, образованное на основе глагола. Имя связи называется ***меткой***. В пределах модели уникальность имени связи не является обязательной. Однако для связей, существующих между двумя конкретными

сущностями, имена должны быть уникальны.

Имя связи должно быть образовано в направлении от родительской сущности к дочерней таким образом, чтобы можно было составить осмысленное предложение, в котором участвуют имя родительской сущности, имя связи и имя дочерней сущности. Например, связь между родительской сущностью Владелец-собаки и дочерней сущностью Собака может быть описана следующим образом (имя связи подчеркнуто):

Владелец-собаки владеет Собака

Метка связи может также содержать пару имен – имя связи с точки зрения родительской сущности и имя связи с точки зрения дочерней сущности. В этом случае метка образуется следующим образом: вначале записывается имя связи с точки зрения родительской сущности, затем после символа «/» – имя связи с точки зрения дочерней сущности. Связь, именуемая с точки зрения дочерней сущности, называется ***обратной (реверсной) связью***.

Для предыдущего примера связь может быть поименована следующим образом:

Владелец-собаки владеет/принадлежит Собака

Резюме

Связь определяет отношение между двумя сущностями или между экземплярами одной и той же сущности. Соединительной связью называется связь между родительской и дочерней сущностью. Сущность называется дочерней, если ее экземпляры могут быть связаны с нулем или одним экземпляром другой сущности (родительской). Сущность называется родительской, если ее экземпляры могут быть связаны с любым количеством экземпляров другой сущности (дочерней). Графически соединительная связь представляется линией от родительской к дочерней сущности с точкой со стороны дочерней сущности.

5.4.7. Безусловные и условные связи и их мощность

В теории информационного моделирования существуют ***две базовые формы связей*** – безусловные и условные.

Если в связи участвуют все экземпляры обеих сущностей, то связь называется ***безусловной***.

Существует ***три вида безусловных связей***:

1) один-к-одному (1 : 1);
2) один-ко-многим (1 : M);
3) многие-ко-многим (M : M).

Данные виды связей называются ***фундаментальными***, поскольку на их основе могут быть построены другие виды связей.

Связь один-к-одному (1 : 1) существует, когда один экземпляр родительской сущности связан с единственным экземпляром дочерней сущности и каждый экземпляр дочерней сущности связан строго с одним экземпляром родительской сущности.

Например, муж женат на одной жене, жена замужем за одним мужем.

Связь один-ко-многим (1 : M) существует, когда один экземпляр родительской сущности связан с одним или более экземпляром дочерней сущности, и каждый экземпляр дочерней сущности связан строго с одним экземпляром родительской сущности.

Например, каждый владелец собаки владеет одной или несколькими собаками, каждая собака принадлежит только одному владельцу.

Связь многие-ко-многим (M : M) существует, когда один экземпляр некоторой сущности связан с одним или более количеством экземпляров другой сущности, и каждый экземпляр второй сущности связан с одним или более количеством экземпляров первой.

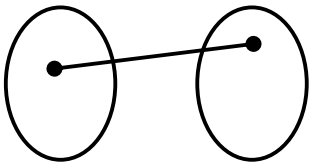
Максимальное количество экземпляров сущности, связанных с каждым экземпляром другой сущности, называется ***мощностью связи***.

В ***условной связи*** могут существовать экземпляры одной из сущностей, которые не принимают участия в связи. Связь, условная с обеих сторон, называется ***биусловной***. В этом случае могут существовать экземпляры обеих сущностей, которые не участвуют в связи.

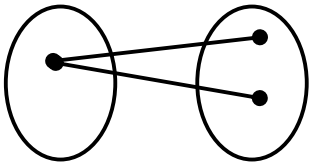
С учетом безусловных, условных и биусловных связей существует ***десять форм связей*** между сущностями (рис. 5.22) [9]. В соответствии с положениями теории информационного моделирования на данном рисунке буквой «у» на конце связи

обозначена условность данной связи.

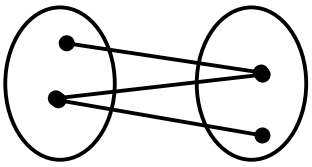
Безусловные формы



1 : 1

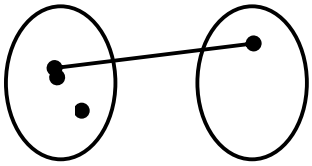


1 : M

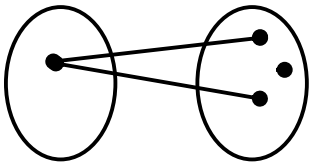


M : M

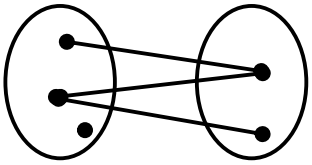
Условные формы



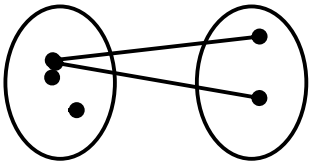
1 : 1y



1y : M

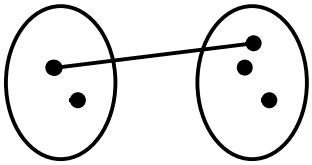


M : My

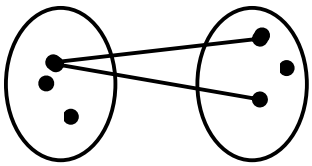


1 : My

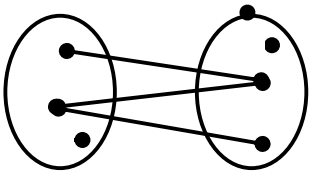
Биусловные формы



1y : 1y



1y : My



My : My

Рис. 5.22. Десять форм связи

Связь многие-ко-многим в IDEF1X-моделировании называется ***неспецифической связью***. Данный вид связи рассматривается в п. 5.4.11.

Связи один-к-одному и один-ко-многим являются соединительными связями между родительской и дочерней сущностью. Правила их графического представления в стандарте IDEF1X приведены в п. 5.4.8.

Резюме

Существует три вида безусловных связей: один-к-одному, один-ко-многим, многие-ко-многим. Мощность связи определяет максимальное количество экземпляров сущности, связанных с каждым экземпляром другой сущности. С учетом безусловных, условных и биусловных связей существует десять форм связей между сущностями.

5.4.8. Графическое представление мощности соединительных связей в IDEF1X-моделировании

На рис. 5.23 представлено графическое представление в IDEF1X мощности соединительной связи с позиции родительской сущности [2, 9]. Как видно из данного рисунка, с каждым экземпляром родительской сущности в общем случае может быть связано различное количество экземпляров дочерней сущности.

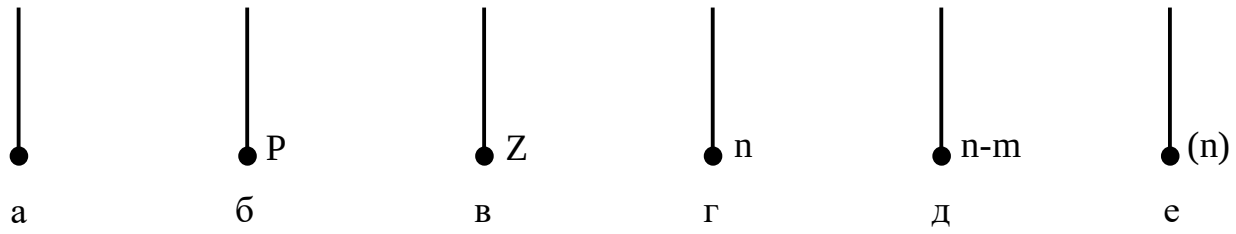


Рис. 5.23. Графическое представление
различных видов дочерней мощности соединительных связей в IDEF1X:
а – ноль-один-или-много; б – один-или-много; в – ноль-или-один;
г – точно-n; д – от n до m; е – мощность определяется условием, записанным в скобках

Максимальное количество экземпляров дочерней сущности, связанных с каждым экземпляром родительской сущности, называется ***дочерней мощностью связи***. Дочерняя мощность связи записывается рядом с точкой, находящейся на конце связи со стороны дочерней сущности.

По умолчанию значение дочерней мощности равно ноль-один-или-много. Это означает, что в каждом экземпляре связи участвует ноль, один или более экземпляров дочерней сущности. При этом дочерняя мощность связи рядом с точкой не отмечается (см. рис. 5.23, а).

Например, между сущностями Семья и Ребенок существует связь один-к-нулю-одному-или-многим, поскольку каждый экземпляр сущности Семья может не иметь ни одного ребенка, иметь одного ребенка или иметь большее количество детей (рис. 5.24).

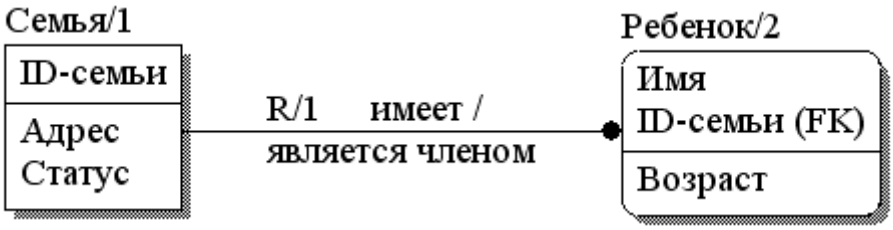


Рис. 5.24. Графическое представление в IDEF1X
связи мощностью один-к-нулю-одному-или-многим

Если рядом с точкой записан символ **P** (см. рис. 5.23, б), то это значит, что в экземпляре связи участвует один или много экземпляров дочерней сущности. Например, каждый экземпляр сущности Владелец-собаки владеет одним или несколькими экземплярами сущности Собака (рис. 5.25).

Это соответствует мощности связи один-к-одному-или-многим.

Символ **Z**, помещенный рядом с точкой, означает, что ноль или один экземпляр дочерней сущности участвует в экземпляре связи (см. рис. 5.23, в). Например, каждый экземпляр сущности Мужчина неженат (женат на нуле женщин) или женат на одной женщине (рис. 5.26).

Это соответствует мощности связи один-к-нулю-или-одному.

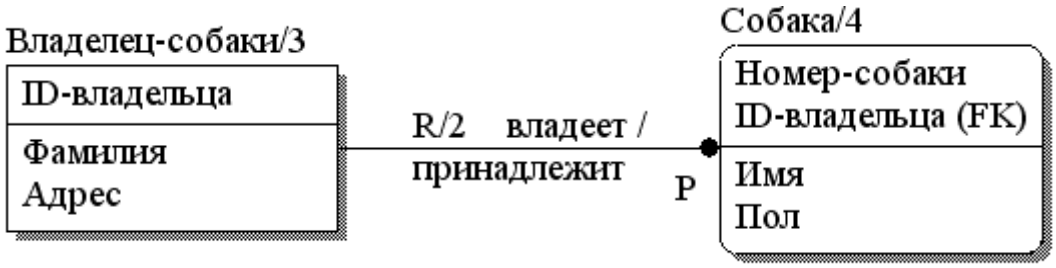


Рис. 5.25. Графическое представление в IDEF1X
связи мощностью один-к-одному-или-многим

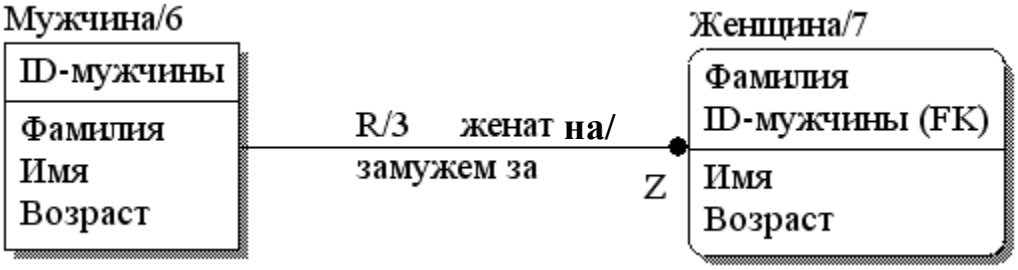


Рис. 5.26. Графическое представление в IDEF1X
связи мощностью один-к-нулю-или-одному

Конкретное число, помещенное рядом с точкой (см. рис. 5.23, г), определяет точную дочернюю мощность связи (точное количество экземпляров дочерней сущности, участвующих в связи с экземпляром родительской сущности). Например, каждый экземпляр сущности Год связан точно с двенадцатью экземплярами сущности Месяц (в году 12 месяцев, рис. 5.27).

Это соответствует мощности связи один-к-точно-12.

Если мощность дочерней связи определена в диапазоне от некоторого значения n до некоторого значения m, то данный диапазон записывается рядом с точкой в виде **n – m** (см. рис. 5.23, д). Например, каждый экземпляр сущности Месяц связан с 28 – 31 экземплярами дочерней сущности День (см. рис. 5.27). Это соответствует мощности связи один-к-диапазону-28-31.

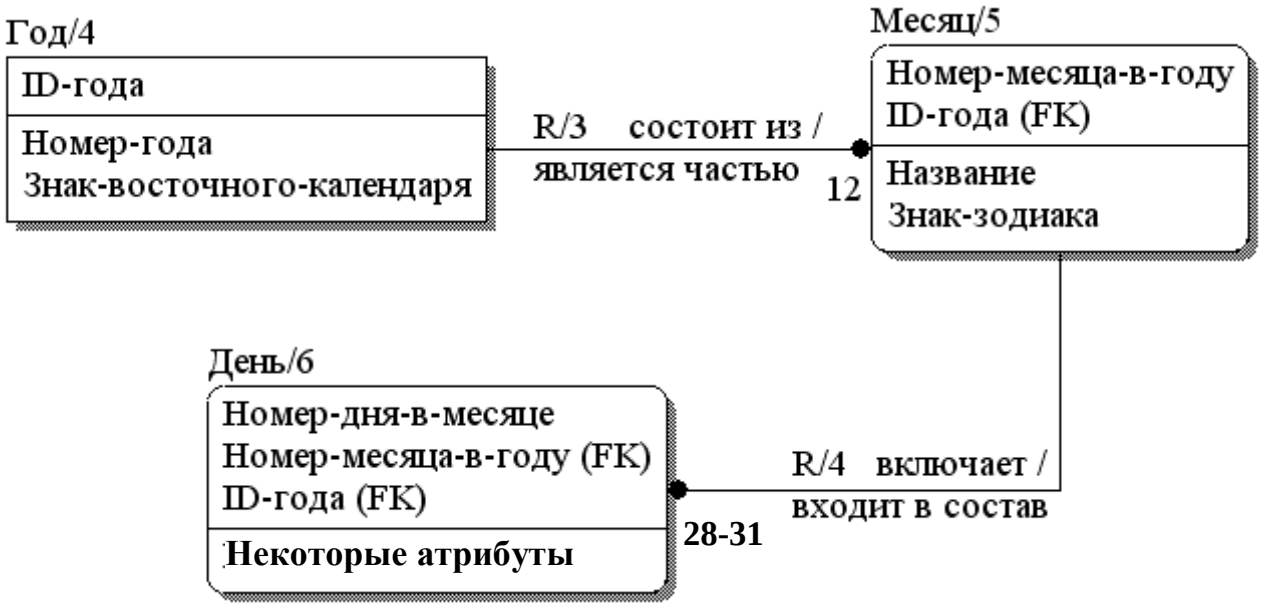


Рис. 5.27. Графическое представление в IDEF1X
связей мощностью один-к-точно-n и один-к-диапазону-n-m

Другие условия, характеризующие дочернюю мощность связи, определяются в круглых скобках (см. рис. 5.23, е). Например, книга может содержать не менее двух глав. В этом случае рядом с точкой на связи должно быть записано условие (>= 2), определяющее дочернюю мощность связи (рис. 5.28). Это соответствует мощности связи один-к-(>= 2).

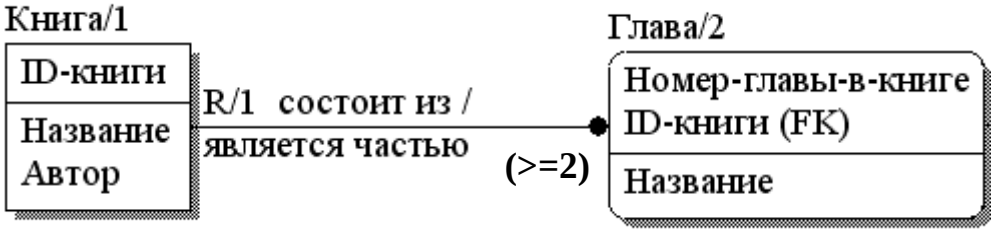


Рис. 5.28. Графическое представление в IDEF1X
связи мощностью один-к-(условие)

Следует отметить, что два последних вида мощности (один-к-диапазону-n-m и один-к-(условие)) современные CASE-средства зачастую не поддерживают, поскольку такой вид связей в предметных областях встречается достаточно редко и может быть реализован с помощью других видов мощности. Эти виды мощностей не поддерживает, например, и одно из самых распространенных в Беларуси CASE-средств для разработки баз данных CA ERwin Data Modeler, более известный под названием Erwin (см. подразд. 6.6).

С учетом дочерней мощности связь между родительской сущностью и дочерней сущностью должна описываться в прямом направлении следующим образом:

<Имя-родительской-сущности> <Имя-связи> <Дочерняя-мощность-связи> <Имя-дочерней-сущности>

и в обратном направлении как

<Дочерняя-мощность-связи> <Имя-дочерней-сущности> <Имя-связи> <Имя-родительской-сущности> .

Например, прямая и обратная связь между родительской сущностью Владелец-собаки и дочерней сущностью Собака, представленная на рис. 5.25, описывается следующим образом:

Владелец-собаки/3 R/2 владеет один-или-много Собака/4 ,

Один-или-много Собака/4 R/2 принадлежит Владелец-Собаки/3 .

Как следует из рис. 5.23, виды дочерней мощности связи, определенные в стандарте IDEF1X, представляют как условные, так и безусловные связи (см. п. 5.4.10).

Резюме

В стандарте IDEF1X определены следующие дочерние мощности соединительных связей: ноль-один-или-много; один-или-много; ноль-или-один; точно-**n**; от **n** до **m**; мощность, определяемая условием. Каждая дочерняя мощность имеет свое графическое обозначение. Не все виды дочерней мощности поддерживаются CASE-средствами.

5.4.9. Формализация соединительных связей

Целью связи является установить соотношение конкретного экземпляра одной сущности с конкретным экземпляром другой. В соединительных связях данная цель достигается размещением вспомогательных атрибутов (внешних ключей) в дочерней сущности. Связь, определенная с помощью вспомогательных атрибутов, называется ***связью, формализованной в данных***.

В качестве вспомогательных атрибутов дочерней сущности используются идентифицирующие атрибуты родительской сущности. Вспомогательные атрибуты помечаются аббревиатурой FK (Foreign Key) в скобках (см. рис. 5.24 – 5.28).

Существует *два вида связей, формализованных в данных*: идентифицирующая связь и неидентифицирующая связь.

Если конкретные экземпляры дочерней сущности определяются через связь с родительской сущностью, то такая связь называется ***идентифицирующей***. При данном виде связи каждый экземпляр дочерней сущности должен быть связан точно с одним экземпляром родительской сущности.

Графически идентифицирующие связи изображаются *сплошной линией* с точкой со стороны дочерней сущности. При идентифицирующей связи вспомогательные атрибуты включаются в состав идентификатора дочерней сущности, а сама дочерняя сущность является *зависимой* и представляется прямоугольником с закругленными углами.

Таким образом, связи, приведенные на рис. 5.24 – 5.28, являются идентифицирующими.

Например, экземпляр сущности Ребенок (см. рис. 5.24) не может быть однозначно определен по своему идентифицирующему атрибуту Имя, поскольку в данной сущности могут быть экземпляры с одинаковыми именами. Поэтому только совокупность имени ребенка и номера (идентификатора) семьи однозначно определяют конкретного ребенка.

Рис. 5.25 отражает ситуацию, при которой для каждого экземпляра сущности Владелец-собаки экземпляры сущности Собака нумеруются (Номер-собаки), начиная с единицы. Поэтому очевидно, что в сущности Собака существует много экземпляров с одинаковыми номерами и их нельзя идентифицировать без связи с конкретными экземплярами родительской сущности.

Родительская сущность в идентифицирующей связи является независимой. Однако если данная сущность является дочерней сущностью в другой идентифицирующей связи, то в рассматриваемой связи обе сущности (и родительская, и дочерняя) будут зависимыми. На рис. 5.27 родительская сущность Месяц связи R/4 одновременно является дочерней сущностью идентифицирующей связи R/3. Поэтому в связи R/4 как родительская, так и дочерняя сущности являются зависимыми. Идентификатор Родительской сущности Год (ID-года) по связи R/3 мигрирует в дочернюю сущность Месяц. Отсюда идентифицирующие атрибуты (ID-года и Номер-месяца-в-году) по связи R/4 мигрируют в дочернюю сущность День.

Если каждый экземпляр дочерней сущности может быть уникально идентифицирован без связи с родительской сущностью, то такая связь называется ***неидентифицирующей***.

Графически неидентифицирующие связи изображаются *штриховой линией* с точкой на конце дочерней сущности. При неидентифицирующей связи вспомогательные атрибуты не включаются в состав идентификатора дочерней сущности, а дочерняя сущность является *независимой*. Однако если данная сущность является дочерней сущностью другой идентифицирующей связи, то она будет зависимой.

На рис. 5.29 – 5.31 представлены примеры неидентифицирующих связей между родительской и дочерней сущностями. Данные примеры даны в привязке к примерам, приведенным на рис. 5.24 – 5.28.

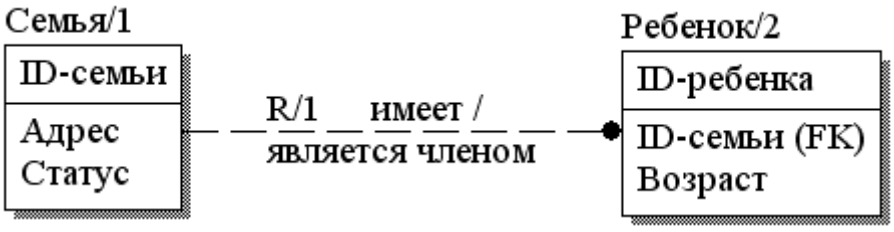


Рис. 5.29. Графическое представление в IDEF1X
неидентифицирующей связи
мощностью один-к-нулю-одному-или-многим

Организация неидентифицирующей связи между сущностями Семья и Ребенок (см. рис. 5.29) стала возможной, потому что в качестве идентификатора дочерней сущности Ребенок выбран идентификационный номер экземпляра сущности Ребенок (ID-ребенка), уникально определяющий каждый ее экземпляр (сравните с неуникальным идентификатором Имя на рис. 5.24).

Аналогично для определения экземпляров дочерних сущностей Собака и Женщина (см. рис. 5.30, 5.31) также используются идентификационные номера ID-собаки и ID-женщины, что позволило отказаться от применения в данных моделях идентифицирующих соединительных связей (сравните с рис. 5.25, 5.26).

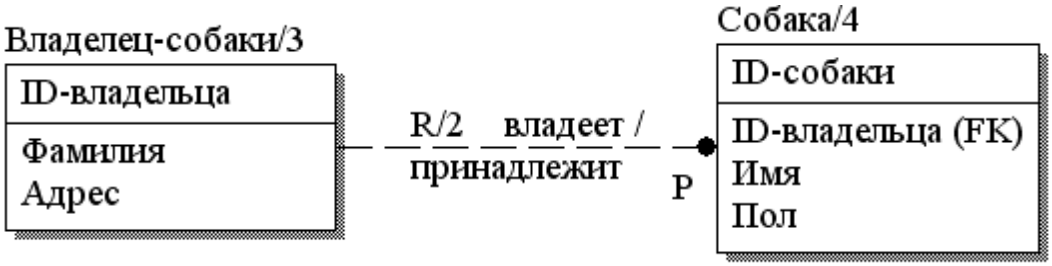


Рис. 5.30. Графическое представление в IDEF1X
неидентифицирующей связи
мощностью один-к-одному-или-многим

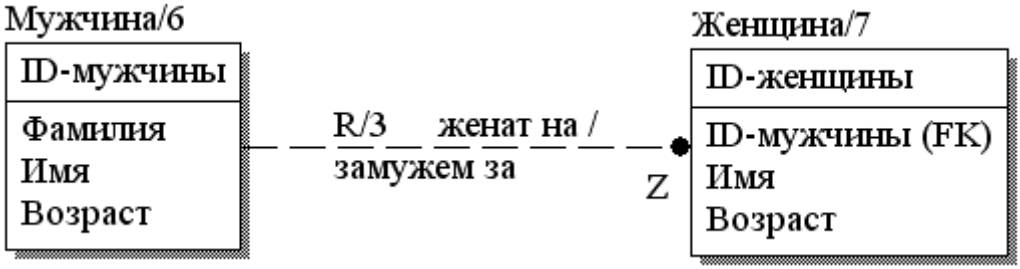


Рис. 5.31. Графическое представление в IDEF1X
неидентифицирующей связи
мощностью один-к-нулю-или-одному

Как показывают рассмотренные выше примеры (см. рис. 5.24 – 5.31), одну и ту же информационную модель в общем случае можно разработать, используя идентифицирующие или неидентифицирующие связи. Следует обратить внимание, что применение идентифицирующих связей приводит к появлению составных идентификаторов в дочерних сущностях (см. например рис. 5.27). Составные идентификаторы при работе с реальными базами данных обрабатывать сложнее, чем простые. Их использование приводит к снижению эффективности работы с базами данных. Поэтому при разработке информационных моделей предметной области предпочтение по возможности следует отдавать неидентифицирующим связям перед идентифицирующими.

Резюме

Связь, определенная с помощью вспомогательных атрибутов, называется связью, формализованной в данных. Существуют идентифицирующие и неидентифицирующие связи, формализованные в данных. При идентифицирующей связи вспомогательные атрибуты включаются в состав идентификатора дочерней сущности, при неидентифицирующей – не включаются. Предпочтение по возможности следует отдавать неидентифицирующим связям перед идентифицирующими.

5.4.10. Реализация безусловных и условных связей в IDEF1X-моделировании

Анализ рис. 5.23 показывает, что совокупность видов дочерней мощности связи, определенных в стандарте IDEF1X, определяет собой как условные, так и безусловные связи.

Виды дочерней мощности связи, не включающие в себя ноль (см. рис. 5.23) определяют, что конкретный экземпляр родительской сущности может быть соединен не менее чем с одним экземпляром дочерней сущности. К таким видам дочерней мощности относятся мощности один-или-много, точно-**n**, диапазон-**n**-**m**, мощность, определяемая условием (два последних вида, если диапазон и условие не включают ноль).

Например, сущность Владелец-собаки владеет не менее чем одной собакой (см. рис. 5.25, 5.30), сущность Год состоит из двенадцати месяцев, сущность Месяц включает 28 – 31 день (см. рис. 5.27).

Таким образом, в этом случае отдельные экземпляры родительской сущности всегда участвуют в соединительной связи. И следовательно, связи с дочерней мощностью, не включающей в себя ноль, определяют ***безусловность связи со стороны родительской сущности***.

Значения дочерней мощности связи, включающие в себя ноль (мощности ноль-один-или-много, ноль-или-один, диапазон-**n**-**m**, мощность, определяемая условием, если диапазон и условие включают ноль), определяют, что конкретный экземпляр родительской сущности может быть соединен с нулем экземпляров дочерней сущности.

Например, некоторые экземпляры сущности Семья могут не иметь ни одного ребенка (см. рис. 5.24, 5.29), некоторые экземпляры сущности Мужчина могут не иметь жены (см. рис. 5.26, 5.31).

Таким образом, отдельные экземпляры родительской сущности могут не участвовать в соединительной связи.

Следовательно, ***условность связи со стороны родительской сущности*** определяется дочерней мощностью связи, включающей в себя ноль.

Если связь является идентифицирующей, то со стороны дочерней сущности она всегда является безусловной, поскольку в данном случае выделение конкретного экземпляра дочерней сущности по определению невозможно без связи с родительской сущностью. Поэтому условность связи со стороны дочерней сущности возможна, только если связь является неидентифицирующей, причем условность или безусловность данной связи определяется ее видом.

Существует два вида неидентифицирующих связей: обязательные и необязательные.

Неидентифицирующая связь называется ***обязательной (mandatory)***, если каждый экземпляр дочерней сущности связан точно с одним экземпляром родительской сущности. Таким образом, каждый экземпляр дочерней сущности участвует в связи.

Следовательно, обязательная неидентифицирующая связь определяет ***безусловность связи со стороны дочерней сущности***.

Например, сущности Муж и Жена на рис. 5.32 связаны обязательной неидентифицирующей связью, поскольку каждая жена обязательно имеет одного мужа. Дочерняя мощность связи равна точно-1. Поэтому связь между сущностями Муж и Жена является безусловной с обеих сторон и имеет мощность один-к-одному.

Неидентифицирующая связь называется ***необязательной (optional)***, если каждый экземпляр дочерней сущности может быть связан с нулем или одним экземпляром родительской сущности. В данном случае конкретные экземпляры дочерних сущностей могут существовать без связи с конкретными экземплярами родительской сущности и существуют экземпляры дочерней сущности, не участвующие в связи.

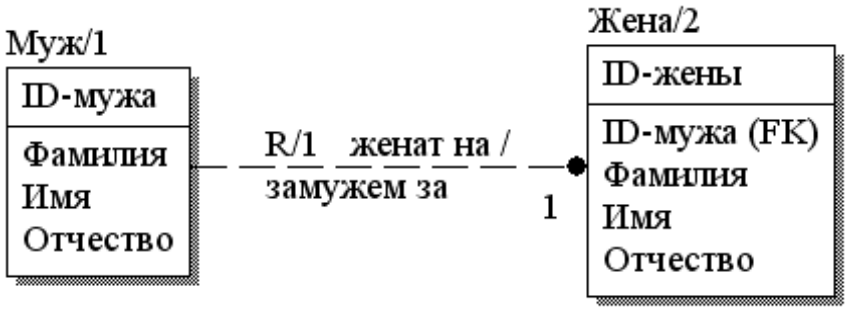


Рис. 5.32. Графическое представление в IDEF1X
обязательной неидентифицирующей связи мощностью один-к-одному

Таким образом, необязательная неидентифицирующая связь определяет ***условность связи со стороны дочерней сущности***.

Например, в рассмотренном в п. 5.4.9 примере (см. рис. 5.31) была определена условность связи со стороны родительской сущности Мужчина за счет дочерней мощности связи ноль-или-один (конкретный мужчина женат на нуле или одной женщине). При этом не был учтен факт, что конкретная женщина также может не иметь мужа. Следовательно, конкретные экземпляры сущности Женщина могут не участвовать в связи. Поэтому между сущностями мужчина и Женщина следует организовать необязательную неидентифицирующую связь.

Графически необязательная неидентифицирующая связь изображается *пунктирной линией с ромбом* со стороны родительской сущности.

На рис. 5.33 учтена условность связи со стороны дочерней сущности Женщина за счет использования необязательной неидентифицирующей связи. Результирующая мощность связи на данном рисунке равна ноль-или-один-к-нулю-или-одному. Таким образом, со стороны обеих сущностей в мощности связи присутствует ноль. Следовательно, связь между сущностями Мужчина и Женщина является *биусловной*.

Таким образом, обязательность или необязательность неидентифицирующей связи определяет ***родительскую мощность*** данной связи (количество экземпляров родительской сущности, связанных с каждым экземпляром дочерней сущности). Если связь идентифицирующая или обязательная неидентифицирующая, то родительская мощность связи равна единице. Для необязательных неидентифицирующих связей родительская мощность равна ноль-или-один.

Родительская мощность соединительной связи, равная единице, графически не отображается. Признаком родительской мощности, равной ноль-или-один, является наличие ромба у родительского конца связи.

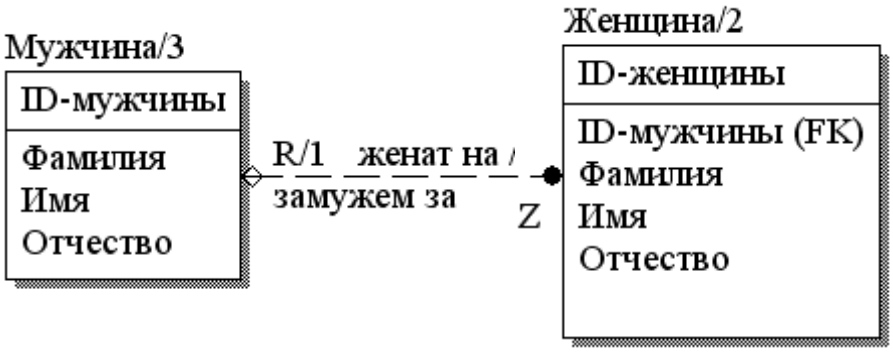


Рис. 5.33. Графическое представление в IDEF1X
необязательной неидентифицирующей связи
мощностью ноль-или-один-к-нулю-или-одному

Резюме

Безусловность связи со стороны родительской сущности определяется дочерней мощностью, не включающей в себя ноль. Условность связи со стороны родительской сущности определяется дочерней мощностью связи, включающей в себя ноль. Существуют обязательные и необязательные неидентифицирующие связи. Обязательная

неидентифицирующая связь определяет безусловность связи со стороны дочерней сущности. Необязательная неидентифицирующая связь определяет условность связи со стороны дочерней сущности.

5.4.11. Неспецифические связи

Неспецифической связью (неконкретной, неопределенной, non-specific) называется связь, при которой экземпляр каждой сущности может быть связан с некоторым количеством экземпляров другой сущности. Неспецифическая связь называется также ***связью многие-ко-многим*** (см. п. 5.4.7). Мощность неспецифической связи в общем случае равна ноль-один-или-много-к-нулю-одному-или-многим. Графически неспецифическая связь изображается сплошной линией с точками на обоих концах.

На рис. 5.34 приведен пример неспецифической связи между сущностями Владелец-квартиры и Квартира. Между данными сущностями существует связь многие-ко-многим, поскольку каждый экземпляр сущности Владелец-квартиры может владеть несколькими квартирами, и каждый экземпляр сущности Квартира может являться собственностью нескольких владельцев.

Неспецифические связи между сущностями могут быть полезны в начале работы над информационной моделью, когда сложно определить конкретные связи между экземплярами сущностей. На более поздних этапах разработки модели каждую неспецифическую связь следует заменять на пару специфических соединительных связей.

С этой целью создается третья сущность, представляющая собой общую дочернюю сущность, связанную соединительными связями с двумя исходными сущностями.

Сущности, вводимые для устранения неспецифических связей, называются ***ассоциативными.*** Ассоциативные сущности содержат идентификаторы каждого из участвующих в связи экземпляров исходных сущностей, что позволяет формализовать в данных связи многие-ко-многим, устранить избыточность информационной модели, а следовательно, и соответствующей базы данных.

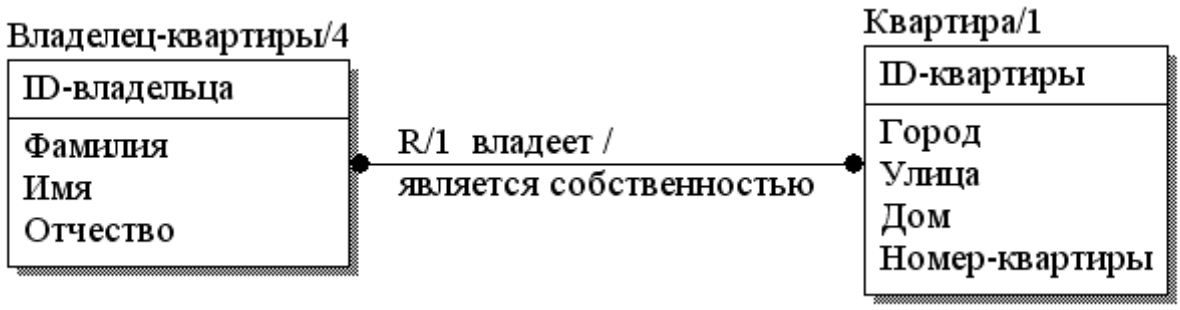


Рис. 5.34. Графическое представление в IDEF1X неспецифической связи

На рис. 5.35 создана ассоциативная сущность Владение, содержащая вспомогательные атрибуты, в качестве которых используются идентификаторы сущностей Квартира и Владелец-квартиры.

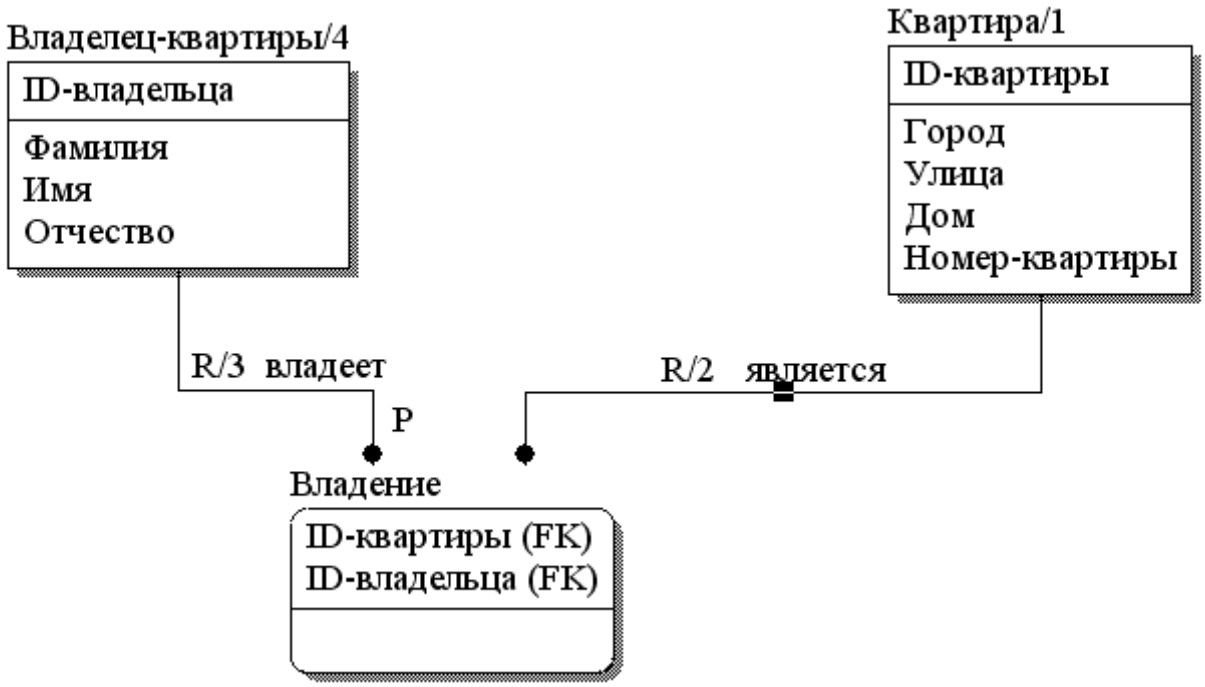


Рис. 5.35. Формализация связи многие-ко-многим
посредством ассоциативной сущности

После введения ассоциативной сущности следует определить мощность каждой из связанных с ней соединительных связей и присвоить данным связям соответствующие имена. В рассматриваемом примере каждый экземпляр сущности Владелец-квартиры владеет одним или несколькими владениями (мощность связи один-к-одному-или-многим, что отражено символом **P** на конце связи, см. рис. 5.35). Каждая квартира (с учетом того, что она может быть, например, не продана) не является владением, является одним или несколькими владениями. Это определяет мощность связи один-к-нулю-одному-или-многим.

Следует отметить, что в ряде CASE-средств (например ERwin, см. подразд. 6.6) формализация связи многие-ко-многим посредством ассоциативной сущности реализована автоматически при переходе от логического уровня представления модели к физическому [16, 21].

Резюме

При неспецифической связи (связи многие-ко-многим) экземпляр каждой сущности может быть связан с некоторым количеством экземпляров другой сущности. Каждую неспецифическую связь следует заменять на две специфических соединительных связи. Для этого вводится ассоциативная сущность, содержащая идентификаторы каждого из участвующих в экземпляре связи экземпляров исходных сущностей.

5.4.12. Организация рекурсивных связей в IDEF1X

Неидентифицирующие соединительные связи и неспецифические связи могут быть ***рекурсивными***, то есть могут связывать экземпляры сущности с другими экземплярами той же сущности.

На 5.36 приведен пример организации рекурсивной связи между экземплярами сущности с использованием *неидентифицирующей соединительной связи*. Каждый экземпляр сущности Сотрудник связан с другими экземплярами рекурсивной связью R13 (руководит / подчиняется). Каждый сотрудник имеет неидентифицирующий атрибут Должность. Если должность руководящая, то данный сотрудник руководит одним или

несколькими подчиненными. Если должность подчиненная, то данный сотрудник может не руководить другими сотрудниками, но подчиняется некоторому руководителю. Один и тот же сотрудник может руководить нулем, одним или несколькими подчиненными и в то же время подчиняется руководителю более высокого ранга. Некоторые сотрудники (например руководитель самого высокого ранга) руководителя не имеют, то есть в обратной связи (подчиняется) не участвуют. Поэтому общая мощность связи между экземплярами сущности Сотрудник равна ноль-или-один-к-нулю-одному-или-многим. Таким образом, данная связь является биусловной.

Рис. 5.36. Организация рекурсивной связи
мощностью ноль-или-один-к-нулю-одному-или-многим
(использование имени роли)

В сущности Сотрудник на рис. 5.36 появился вспомогательный атрибут (внешний ключ) ID-руководителя. Его имя отличается от идентификатора ID-сотрудника, мигрирующего по связи руководит из родительского в дочерний экземпляр сущности. В данном случае внешнему ключу присвоено имя роли. ***Имя роли*** – это имя, альтернативное базовому имени мигрирующего из родительской сущности внешнего ключа, отражающее роль данного ключа в дочерней сущности. Имя роли называется еще ***функциональным именем*** внешнего ключа.

Имя роли обычно используется, если нужно отразить назначение внешнего ключа в

дочерней сущности. Например, на рис. 5.31 вместо базового имени вспомогательного атрибута ID-мужчины в сущности Женщина можно использовать имя роли ID-мужа.

Иногда применение имен ролей является обязательным. Это касается тех случаев, когда два или несколько атрибутов одной сущности определены в одной и той области значений, но имеют разный смысл.

В рассмотренном выше примере (см. рис. 5.36) такими атрибутами являются ID-сотрудника и ID-руководителя. В данном случае применение имени роли для внешнего ключа является обязательным, поскольку по определению имена атрибутов в пределах сущности должны быть уникальны.

В общем случае в качестве имени внешнего ключа может использоваться базовое имя, имя роли, а также полное имя

<Имя-роли>.<Базовое-имя>,

включающее имя роли и базовое имя вспомогательного атрибута (рис. 5.37).

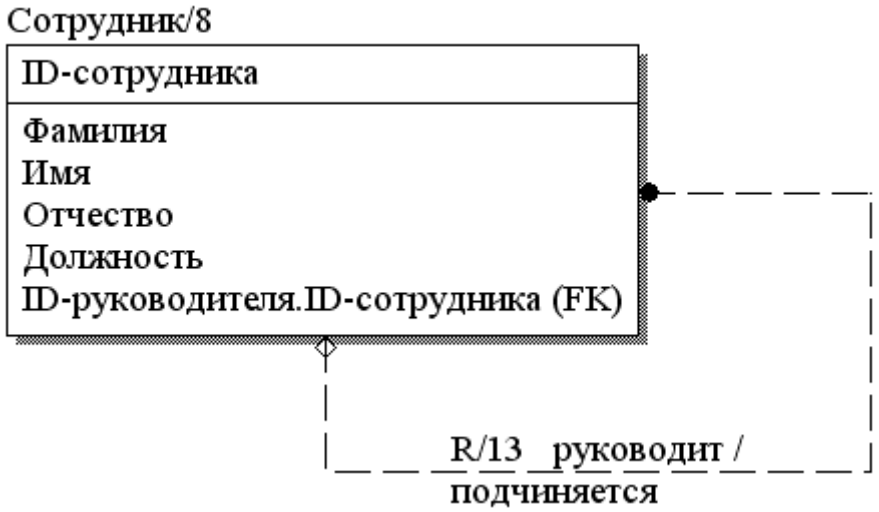


Рис. 5.37. Организация рекурсивной связи
мощностью ноль-или-один-к-нулю-одному-или-многим
(использование полного имени внешнего ключа)

Как следует из рассмотренного примера, использование неидентифицирующей соединительной связи в общем случае позволяет организовать иерархическую рекурсию между экземплярами сущности (рис. 5.38).

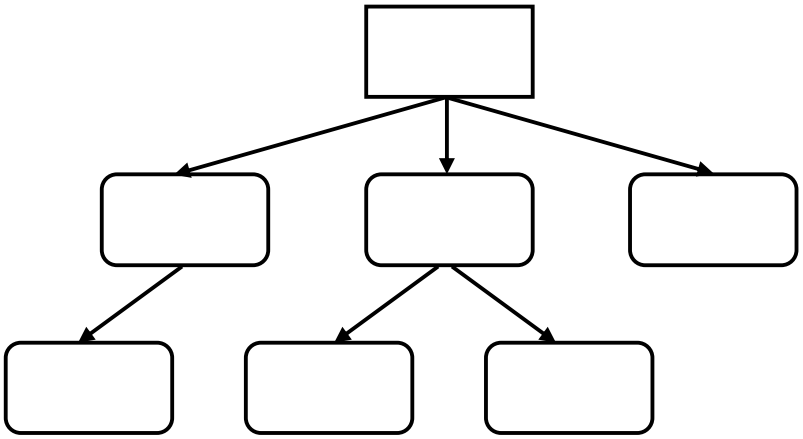


Рис. 5.38. Иерархическая рекурсия между экземплярами сущности

На рис. 5.39 приведен пример организации рекурсивной связи между экземплярами сущности с использованием *неспецифической связи*.

Рис. 5.39. Организация рекурсивной связи
мощностью многие-ко-многим

В данном примере некоторый экземпляр сущности Программный-модуль может вызывать ноль, один или много других экземпляров данной сущности и в то же время вызываться нулем, одним или многими другими экземплярами этой же сущности.

Рекурсивная связь многие-ко-многим устраняется аналогично рассмотренному в п. 5.4.11 введением ассоциативной сущности. На рис. 5.40 введена ассоциативная сущность Модуль. Связь многие-ко-многим заменена двумя соединительными связями между сущностями Программный-модуль и Модуль, каждая из которых имеет мощность один-к-нулю-одному-или-многим (программный модуль может вызывать ноль, один или много других модулей и в то же время вызываться нулем, одним или многими другими модулями).

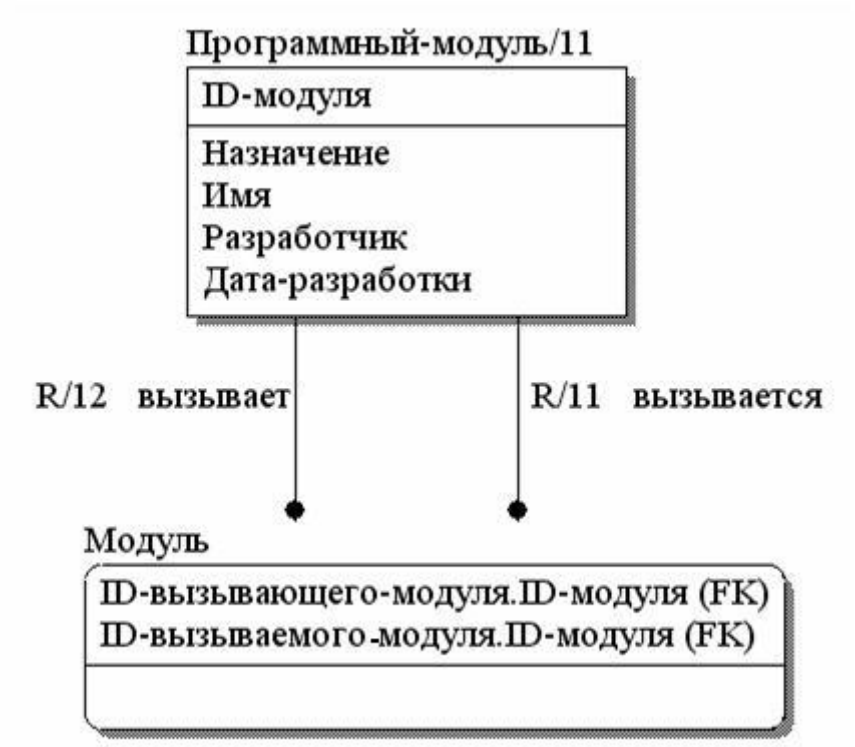


Рис. 5.40. Формализация рекурсивной связи мощностью многие-ко-многим посредством ассоциативной сущности

В ассоциативной дочерней сущности Модуль имеются два внешних ключа, определенных в одной и той области значений, но имеющих разный смысл. В этом

случае обязательно использование имен ролей. По связи R12 (вызывает) экземпляр сущности Программный модуль связан с вызываемым модулем, по связи R11 (вызывается) – с вызывающим модулем. Поэтому в качестве имен внешних ключей в сущности Модуль используются составные имена, включающие имена ролей ID-вызываемого-модуля и ID-вызывающего-модуля.

Из рассмотренного примера видно, что использование связи многие-ко-многим в общем случае позволяет организовать сетевую рекурсию между экземплярами сущности (рис. 5.41), при которой возможны связи между любыми экземплярами сущностей.

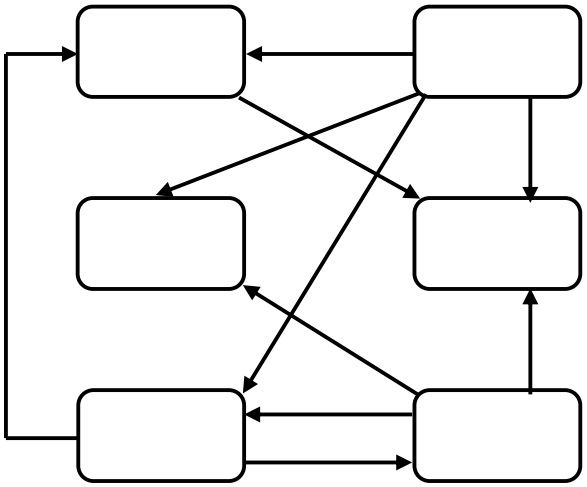


Рис. 5.41. Сетевая рекурсия между экземплярами сущности

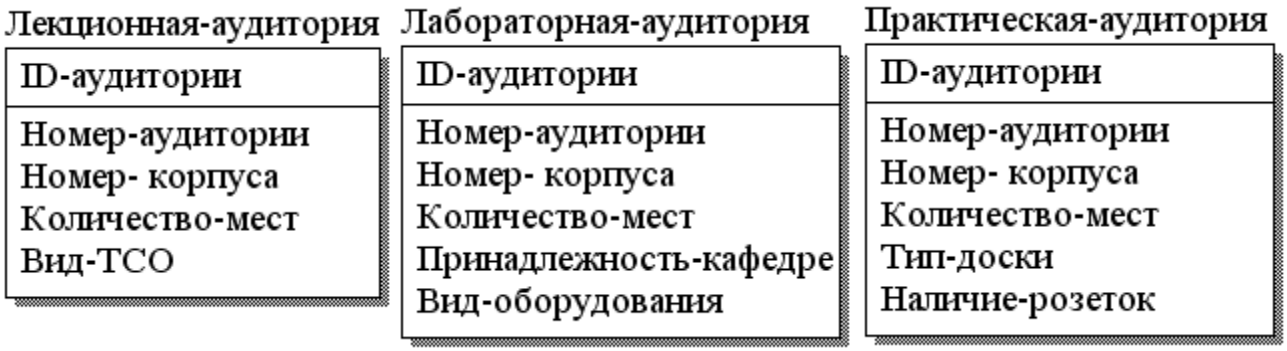
Резюме

Возможна организация рекурсивной связи между экземплярами одной и той же сущности. Для этого может использоваться неидентифицирующая соединительная связь или неспецифическая связь. Неидентифицирующая соединительная связь позволяет организовать иерархическую рекурсию. Неспецифическая связь позволяет организовать сетевую рекурсию. В качестве имени внешнего ключа может использоваться базовое имя, имя роли и составное имя. Имя роли отражает роль внешнего ключа в дочерней сущности.

5.4.13. Связи категоризации в IDEF1X

При разработке информационной модели предметной области могут возникнуть ситуации, когда несколько сущностей имеют общие атрибуты.

Рис. 5.42 иллюстрирует такую ситуацию. В аудиторном фонде университета есть аудитории различных типов (например лекционные, лабораторные, практические). Сущности, представляющие данные аудитории, имеют ряд одинаковых атрибутов (Номер-аудитории, Номер-корпуса, Количество-мест).



ТСО – технические средства обучения

Рис. 5.42. Сущности с общими атрибутами
для предметной области «Аудиторный фонд университета»

Для устранения избыточности информационной модели в таких случаях рекомендуется создать общую сущность для представления характеристик, совместно принадлежащих специализированным сущностям. Общая сущность называется ***групповой сущностью*** (родовой сущностью, generic entity), специализированная сущность – ***сущностью-категорией*** (category entity). Такие сущности связаны между собой связью категоризации (рис. 5.43).

Связь категоризации (связь супертип-подтип) – это связь между групповой сущностью и сущностью-категорией, при которой один реальный или виртуальный объект предметной области представляется комбинацией экземпляра групповой сущности и экземпляра сущности-категории.

Групповая сущность представляет собой полный набор объектов, сущности-категории – подтипы этих объектов. Сущности-категории всегда являются зависимыми.

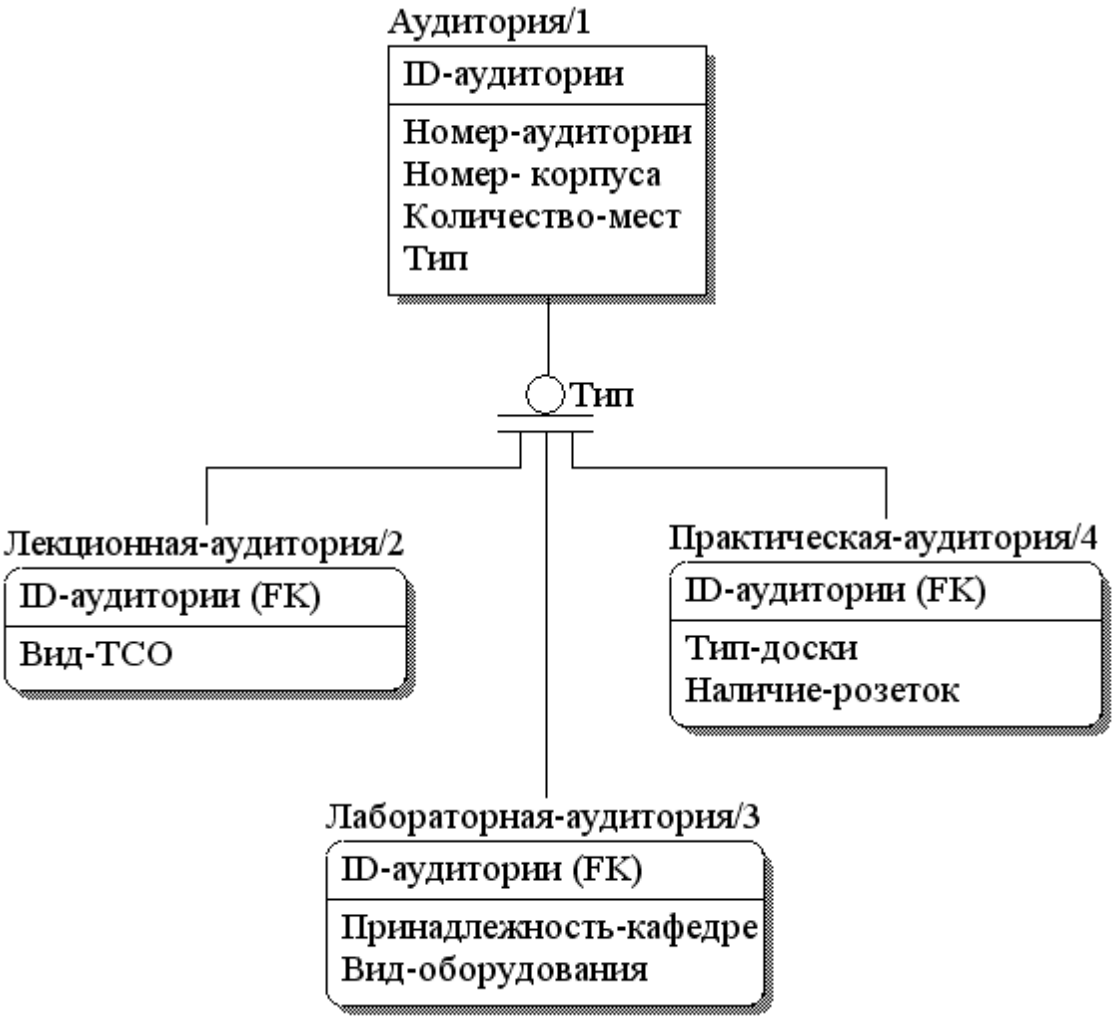


Рис. 5.43. Связь категоризации. Полная группа категорий

Таким образом, связи категоризации используются для представления структур, в которых некоторая сущность является подтипом (категорией) другой сущности. Поэтому групповая сущность называется также ***сущностью-супертипом***, а сущность-категория – ***сущностью-подтипом***.

На рис. 5.43 групповой сущностью является сущность Аудитория, сущностями-категориями – сущности Лекционная-аудитория, Практическая-аудитория, Лабораторная-аудитория.

Термин ***группа категорий*** (category cluster) определяет набор из одной или более связей категоризации и связанных с ними сущностей-категорий. Каждый экземпляр сущности-категории связан строго с одним экземпляром групповой сущности, и каждый экземпляр групповой сущности может быть связан с экземпляром только одной сущности-категории из входящих в группу. Поэтому сущности-категории в группе являются взаимно исключающими. Например, сущности Лекционная-аудитория, Практическая-аудитория, Лабораторная-аудитория входят в состав одной группы категорий, поэтому аудитория не может быть лекционной и лабораторной одновременно (см. рис. 5.43).

Однако сущность может быть групповой для нескольких групп категорий. В этом случае групповая сущность может быть связана одновременно с одной из сущностей-категорий в каждой группе. В то же время сущность-категория может входить в состав только одной группы категорий и не может быть дочерней сущностью в идентифицирующей соединительной связи с некоторой родительской сущностью.

Если в группе категорий представлены все возможные сущности-категории, то такая группа называется ***полной группой категорий***. В этом случае каждый экземпляр групповой сущности связан с экземпляром сущности-категории.

Графически полная группа категорий представляется линией, выходящей из групповой сущности, с кругом, который подчеркивается двойной линией (см. рис. 5.43). От подчеркивания отходят отдельные линии к каждой сущности-категории, входящей в данную группу. Каждая пара линий (от групповой сущности к кругу и от круга к сущности-категории) представляет собой одну из связей категоризации, входящих в группу.

Если в группе категорий представлены не все возможные сущности-категории, то такая группа называется ***неполной группой категорий***. В этом случае экземпляр групповой сущности может существовать без связи с экземпляром сущности-категории.

При графическом представлении неполной группы категорий круг, объединяющий собой группу связей категоризации, подчеркивается одинарной линией (рис. 5.44).

Мощность связей категоризации обычно графически не изображается, поскольку она всегда равна один-к-одному для полной группы категорий и один-к-нулю-или-одному для неполной группы категорий. При необходимости рядом со связью на стороне

сущности-категории может быть поставлен символ 1 или **Z**, явно отражающий данную мощность.

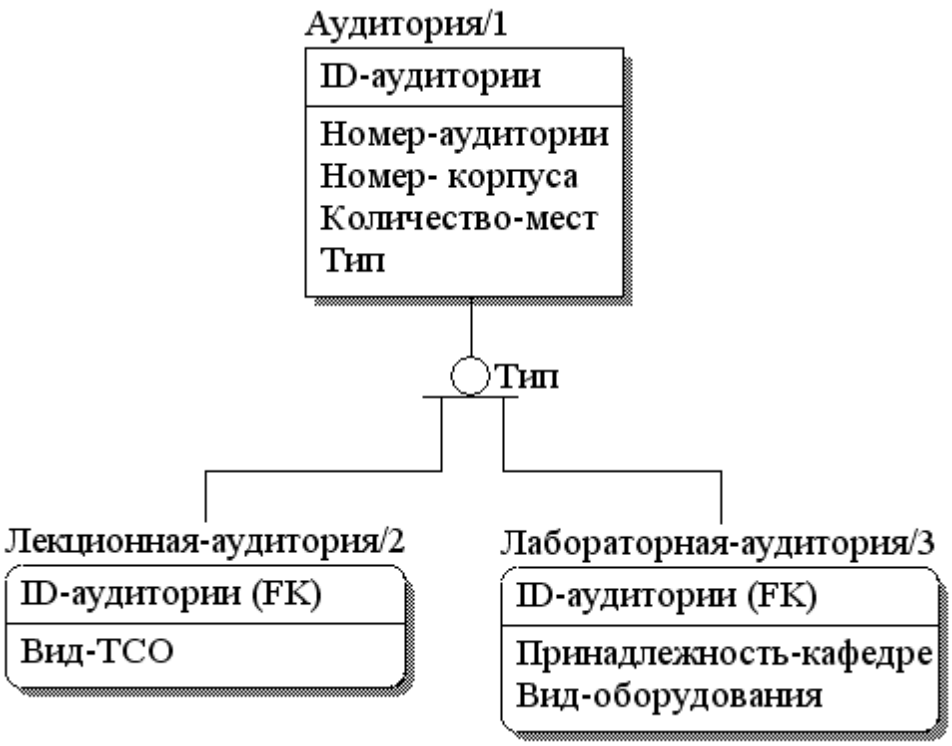


Рис. 5.44. Связь категоризации. Неполная группа категорий

Один из атрибутов групповой сущности или ее предка может быть определен как дискриминатор для группы сущностей-категорий.

Дискриминатор – это атрибут групповой сущности, значение которого определяет категорию его экземпляра. Имя дискриминатора записывается рядом с кругом, объединяющим группу связей категоризации. Для примеров, приведенных на рис. 5.43, 5.44, дискриминатором является атрибут Тип групповой сущности.

Имена дискриминаторов всех групп в модели должны быть уникальны.

Связям категоризации явные имена не присваиваются. Каждая связь групповой сущности с сущностью-категорией для неполных групп категорий может быть прочитана как «*может быть*». Например, Аудитория может быть Лекционной-аудиторией или Лабораторной-аудиторией (см. 5.44.

Каждая связь групповой сущности с сущностью-категорией для полных групп

категорий может быть прочитана как «*должен быть*». Например, Аудитория должна быть Лекционной-аудиторией, Лабораторной-аудиторией или Практической-аудиторией (см. рис. 5.43).

В обратном направлении связь читается как «*есть*». Например, Лекционная-аудитория есть Аудитория.

Групповая сущность и каждая сущность-категория должны иметь одинаковые идентификаторы (см. рис. 5.43). Идентификатор сущности-категории называется ***унаследованным идентификатором***. Он может быть передан из групповой сущности (см. рис. 5.43) или ее предка (рис. 5.45) во вложенных связях категоризации.

В примере, приведенном на рис. 5.45, введена групповая сущность Помещение. Данная сущность связана связями категоризации с сущностями-категориями Служебное-помещение и Учебная-аудитория. Последние образуют неполную группу категорий, дискриминатором в которой является атрибут Назначение сущности Помещение. В названные сущности-категории передан унаследованный из групповой сущности идентификатор ID-помещения.

Сущность Учебная-аудитория в свою очередь является групповой сущностью для полной группы категорий, в состав которой входят сущности-категории Лекционная-аудитория, Лабораторная-аудитория, Практическая-аудитория. Дискриминатором в данной группе категорий является атрибут Тип сущности Учебная-аудитория.

В данную группу сущностей-категорий передается идентификатор ID-помещения, унаследованный из сущности-предка Помещение.

Таким образом, групповые сущности, связанные с сущностями-категориями связями категоризации, образуют иерархии наследования.

В качестве имен унаследованных идентификаторов в сущностях-категориях возможно использование имен ролей. На рис. 5.46 в сущности-категории Учебная-аудитория используется идентификатор

ID-аудитории.ID-помещения,

состоящий из имени роли и базового имени унаследованного идентификатора.

Рис. 5.45. Вложенность связей категоризации.

Использование имени предка в качестве имени унаследованного идентификатора

Рис. 5.46. Вложенность связей категоризации.

Использование имени роли в качестве имени унаследованного идентификатора

Сущности-категории Лекционная-аудитория и Практическая-аудитория в качестве имени своих идентификаторов унаследовали имя роли ID-аудитории своей групповой сущности Учебная-аудитория.

В сущности-категории Лабораторная-аудитория в качестве унаследованного

идентификатора используется имя роли ID-лаборатории. При этом в качестве базового имени идентификатора используется имя групповой сущности (см. рис. 5.45) или имя ее роли, если оно есть (см. рис. 5.46).

Резюме

При использовании связи категоризации один реальный или виртуальный объект предметной области представляется комбинацией экземпляра групповой сущности и экземпляра сущности-категории. В групповую сущность включаются атрибуты, совместно принадлежащие сущностям-категориям. В группу категорий входит набор из одной или более связей категоризации и связанных с ними сущностей-категорий. Возможна организация полных и неполных групп категорий. Значение дискриминатора определяет категорию экземпляра групповой сущности. В качестве имен унаследованных идентификаторов в сущностях-категориях могут использоваться имена ролей.

5.4.14. Рабочие продукты информационного моделирования

Для информационной модели разрабатываются *три рабочих продукта*.

1. *Диаграмма информационной структуры* – графическое представление информационной модели.

В IDEF1X-моделировании существуют три концептуальных уровня представления диаграмм [2, 9]:

- диаграммы «сущность–связь», называемые *ER-диаграммами* (Entity–Relationship); на данных диаграммах отображаются сущности и связи между ними, а атрибуты не отображаются; представляют информационную модель верхнего уровня; используются в начале работы над моделью данных;
- диаграммы, основанные на ключах, называемые *KB-диаграммами* (Key–Based); на данных диаграммах отображаются сущности с первичными ключами и связи между сущностями; являются вторым уровнем детализации информационной модели;
- полные диаграммы с атрибутами, называемые *FA-диаграммами* (Fully–Attributed); на данных диаграммах отображаются сущности с первичными и вторичными

ключами, а также связи между сущностями; являются наиболее детальным отображением модели данных; представляют данные в третьей нормальной форме.

Все рассмотренные ранее в подразд. 5.4 примеры были даны в виде FA-диаграмм. Например, на рис. 5.45 содержится информационная модель, представленная в виде диаграммы FA-уровня. Та же модель, представленная ER-диаграммой, приведена на рис. 5.47. На рис. 5.48 представлен уровень KB-диаграммы для той же модели.

Рис. 5.47. Уровень ER-диаграммы

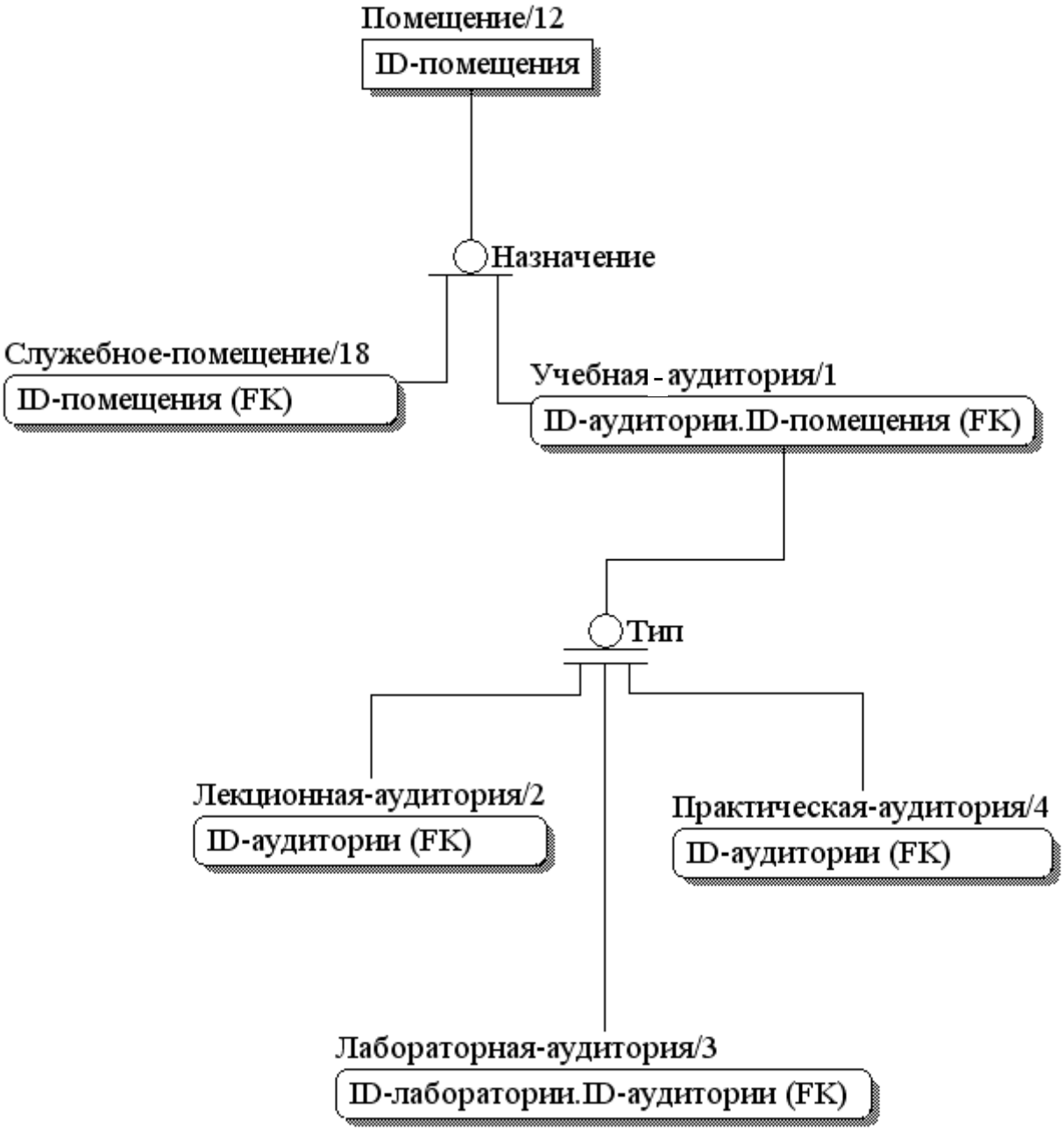


Рис. 5.48. Уровень КВ-диаграммы

2. *Описание сущностей и атрибутов* – списки всех сущностей модели, всех атрибутов (вместе с их доменами) и их описание.

3. *Описание связей* – перечни связей вместе с их описаниями.

Резюме

Для информационной модели предметной области разрабатываются диаграмма

информационной структуры, описание сущностей и атрибутов и описание связей. В IDEF1X-моделировании существует три концептуальных уровня представления диаграмм: ER-диаграммы, KB-диаграммы, FA-диаграммы.

РАЗДЕЛ 6. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

6.1.История развития CASE-средств

Очевидно, что большие размеры и высокая сложность разрабатываемых ПС при ограничениях на бюджетные и временные затраты проекта могут привести к низкому качеству конечных программных продуктов и системы в целом. В этой связи в последнее время все большее внимание уделяется современным технологиям и инструментальным средствам, обеспечивающим автоматизацию процессов ЖЦ ПС (CASE-средствам). Использование таких инструментальных средств позволяет существенно сократить длительность и стоимость разработки систем и ПС при одновременном повышении качества процесса разработки и, как следствие, качества разработанных ПС.

В истории развития CASE-средств обычно выделяется ***шесть периодов***. Данные периоды отличаются применяемой техникой и методами разработки ПС. Эти периоды используют в качестве инструментальных средств следующие средства [19].

Период 1. Ассемблеры, анализаторы.

Период 2. Компиляторы, интерпретаторы, трассировщики.

Период 3. Символические отладчики, пакеты программ.

Период 4. Системы анализа и управления исходными текстами.

Период 5. Первое поколение CASE (CASE-I). Это CASE-средства, позволяющие выполнять поддержку начальных работ процесса разработки ПС и систем (анализ требований к системе, проектирование архитектуры системы, анализ требований к программным средствам, проектирование программной архитектуры, логическое проектирование баз данных). Адресованы непосредственно системным аналитикам, проектировщикам, специалистам в предметной области. Поддерживают графические

модели, экранные редакторы, словари данных. Не предназначены для поддержки полного ЖЦ ПС.

Период 6. Второе поколение CASE (CASE-II). Представляют собой, как правило, набор (линейку) инструментальных средств, каждое их которых предназначено для поддержки отдельных этапов процесса разработки или других процессов ЖЦ ПС. В совокупности обычно поддерживают практически полный ЖЦ ПС. Используют средства моделирования предметной области, графического представления требований, поддержки автоматической кодогенерации ПС. Содержат средства контроля и управления разработкой, интеграции системной информации, оценки качества результатов разработки. Поддерживают моделирование и прототипирование системы, тестирование, верификацию, анализ сгенерированных программ, генерацию документации по проекту.

Ко второму поколению CASE-средств относятся, например, линейки Telelogic и AllFusion, обзор которых приведен в подразд. 6.5, 6.6.

CASE-технологии предлагают новый, основанный на автоматизации подход к концепции ЖЦ ПС. Современные варианты CASE-моделей ЖЦ, называемые обычно RAD-моделями, рассмотрены в подразд. 2.3.

Наибольшие изменения в ЖЦ ПС при использовании CASE-технологий касаются первых этапов ЖЦ, связанных с анализом требований и проектированием. CASE-средства позволяют использовать визуальные среды разработки, средства моделирования и быстрого прототипирования разрабатываемой системы или ПС. Это позволяет на ранних этапах разработки оценить, насколько будущая система или программное средство устраивает заказчика и насколько она дружественна будущему пользователю.

Табл. 7.1 содержит усредненные оценки трудозатрат по основным этапам разработки ПС при различных подходах к процессу разработки [19]. Номерам строк в данной таблице соответствуют: 1 – традиционная разработка с использованием классических технологий; 2 – разработка с использованием современных структурных методологий проектирования; 3 – разработка с использованием CASE–технологий.

Таблица 7.1

Сравнительная оценка трудозатрат
по этапам процесса разработки программных средств

№ подхода	Анализ, %	Проектирование, %	Кодирование, %	Тестирование, %
1	20	15	20	45
2	30	30	15	25
3	40	40	5	15

Из таблицы видно, что при традиционной разработке ПС основные усилия направлены на кодирование и тестирование, а при использовании CASE-технологий – на анализ и проектирование, поскольку CASE предполагают автоматическую кодогенерацию, автоматизированное тестирование и автоматический контроль проекта. Сопровождение кодов ПС заменяется сопровождением спецификаций проектирования. В результате данных факторов цена ошибок, вносимых в проект при разработке и сопровождении ПС и систем, существенно снижается.

Резюме

В истории развития CASE-средств обычно выделяется шесть периодов. При традиционной разработке ПС основные усилия направлены на кодирование и тестирование, при использовании CASE-технологий – на анализ и проектирование.

6.2. Базовые принципы построения Case-средств

Большинство CASE-средств основано на ***парадигме метод – нотация – средство*** [19].

Парадигма – это система изменяющихся форм некоторого понятия. В данном случае метод реализуется с помощью нотаций. Метод и нотации поддерживаются инструментальными средствами.

Метод – это систематическая процедура или техника генерации описаний компонент ПС. Примерами являются метод JSP Джексона, методология SADT (см. подразд. 4.6, п. 5.2.1).

Нотация – это система обозначений, предназначенная для описания структуры системы, элементов данных, этапов обработки; может включать графы, диаграммы, таблицы, схемы алгоритмов, формальные и естественные языки. Например, метод JSP реализуется с помощью нотации, базирующейся на применении четырех базовых конструкций данных. Современной нотацией методологии SADT является IDEF0.

Средства – это инструментарий для поддержки методов, помогающий пользователям при создании и редактировании графического проекта в интерактивном режиме, способствующий организации проекта в виде иерархии уровней абстракции, выполняющий проверки соответствия компонентов. Например, средством, поддерживающим метод JSP, являеся SmartDraw. IDEF0 поддерживается средством BPwin.

Фактически ***CASE-средство*** – это совокупность графически ориентированных инструментальных средств, поддерживающих процессы или отдельные этапы процессов ЖЦ ПС и систем.

К ***CASE-средствам*** может быть отнесено любое программное средство, обеспечивающее автоматическую помощь при разработке ПС, их сопровождении или управлении проектом, базирующееся на следующих ***основополагающих принципах***:

1. *Графическая ориентация*. В CASE-средствах используется мощная графика для описания и документирования систем или ПС и для улучшения интерфейса с пользователем.

2. *Интеграция*. CASE-средство обеспечивает легкость передачи данных между своими компонентами и другими средствами, входящими в состав линейки CASE-средств. Это позволяет поддерживать совокупность процессов ЖЦ ПС.

3. *Локализация всей проектной информации в репозитории* (компьютерном хранилище данных). Исполнителям проекта доступны соответствующие разделы репозитория в соответствии с их уровнем доступа. Это обеспечивает поддержку принципа коллективной работы. Информация из репозитория может использоваться для работ над текущим проектом, в том числе для автоматической кодогенерации ПС или

систем, разработки следующих проектов, сбора статистики по выполненным ранее проектам организации.

Помимо данных принципов в основе концептуального построения CASE-средств лежат следующие ***положения*** [19].

1. *Человеческий фактор*. Его учет позволяет привести процессы ЖЦ ПС и систем к легкой, удобной и экономичной форме.

2. *Использование базовых программных средств*, применяющихся в других приложениях (СУБД, компиляторы с различных языков программирования, отладчики, языки четвертого поколения 4GL и др.).

3. *Автоматизированная или автоматическая кодогенерация*. При автоматизированной кодогенерации выполняется частичная генерация кодов программного средства, остальные участки программируются вручную. При автоматической кодогенерации выполняется полная генерация кодов программного средства. Возможны различные виды генерации (например, генерация проектной документации, базы данных по информационной модели, кодов из разработанных спецификаций программного средства; автоматическая сборка модулей, хранящихся в репозитории).

4. *Ограничение сложности*. Такое ограничение позволяет поддерживать сложность компонентов разрабатываемого программного средства или системы на уровне, доступном для понимания, использования и модификации.

5. *Доступность для различных категорий пользователей*, в том числе заказчиков, специалистов в предметной области, системных аналитиков, проектировщиков, программистов, тестировщиков, инженеров по качеству, менеджеров проектов. CASE-средства содержат инструменты различного функционального назначения, поддерживающие различные этапы основных, вспомогательных и организационных процессов ЖЦ ПС и систем (см. подразд. 1.2, 6.5).

6. *Рентабельность,* обеспечивающая быструю окупаемость денежных средств, вложенных в приобретение CASE-средства, за счет сокращения сроков и стоимости проектов.

7. *Сопровождаемость*. CASE-средства обладают способностью адаптации к изменяющимся требованиям и целям проекта.

Резюме

CASE-средства представляют собой совокупность графически ориентированных инструментальных средств, поддерживающих ЖЦ ПС и систем. CASE-средства базируются на принципах графической ориентации, интеграции и локализации всей проектной информации в репозитории. В основе построения CASE-средств лежат человеческий фактор, использование базовых ПС, автоматизированная или автоматическая кодогенерация, ограничение сложности, доступность для разных категорий пользователей, рентабельность, сопровождаемость.

6.3. Основные функциональные возможности CASE-средств

В состав CASE-средств входят ***четыре основных компонента*** [19]:

1. *Средства централизованного хранения всей информации о проекте (репозиторий).* Предназначены для хранения информации о разрабатываемом программном средстве или системе в течение всего ЖЦ разработки.

2. *Средства ввода*. Служат для ввода данных в репозиторий, для организации взаимодействия участников проекта с CASE-средством. Должны поддерживать различные методологии анализа, проектирования, тестирования, контроля. Предназначены для использования в течение ЖЦ программного средства или системы различными категориями участников проекта (системными аналитиками, проектировщиками, программистами, тестировщиками, менеджерами, специалистами по качеству и т.д.).

3. *Средства анализа и разработки*. Предназначены для анализа различных видов графических и текстовых описаний и их преобразований в процессе разработки.

4. *Средства вывода*. Служат для кодогенерации, создания различного вида документов, управления проектом.

Все компоненты CASE-средств в совокупности обладают следующими ***функциональными возможностями*** [19]:

- поддержка графических моделей;
- контроль ошибок;

- поддержка репозитория;
- поддержка основных, вспомогательных и организационных процессов ЖЦ ПС.

Поддержка графических моделей

В CASE-средствах разрабатываемые ПС представляются схематически. На различных уровнях проектирования могут использоваться различные виды и нотации графического представления ПС. Обычно применяются диаграммы различных типов, в том числе иерархии требований, диаграммы функционального моделирования (например IDEF0, DFD, см. подразд. 5.2, 5.3), диаграммы информационного моделирования (например IDEF1X, см. подразд. 5.4), структурограммы (см. п. 4.1.3), диаграммы Джексона (см. подразд. 4.6, п. 5.5.1), диаграммы Варнье-Орра (см. п. 5.5.2), UML-диаграммы и т.п.

Разработка диаграмм осуществляется с помощью специальных графических редакторов, основными *функциями* которых являются создание и редактирование иерархически связанных диаграмм, их объектов и связей между объектами, а также автоматический контроль ошибок.

Контроль ошибок

В CASE-средствах, как правило, реализуются следующие ***типы контроля*** [19]:

1. *Контроль синтаксиса диаграмм и типов их элементов*. Например, при IDEF0-моделировании контролируется максимальное и минимальное количество функциональных блоков на диаграммах, наличие дуги управления и выходной дуги для любого функционального блока и т.п.

2. *Контроль полноты и корректности диаграмм*. При данном типе контроля выполняется проверка наличия имен у всех элементов диаграмм, проверка наличия необходимых описаний в репозитории и др.

3. *Контроль декомпозиции функций*. При данном типе контроля выполняется оценка декомпозиции на основе различных метрик. Например, может быть оценена

эффективность и корректность декомпозиции с точки зрения связности и сцепления модулей (см. подразд. 4.7).

4. *Сквозной контроль диаграмм* одного или различных типов на предмет их взаимной корректности. Например, при IDEF0-моделировании контролируется соответствие граничных дуг родительского блока с внешними дугами дочерней диаграммы. При разработке IDEF0- и IDEF1X-моделей предметной области выполняется контроль их взаимной корректности и непротиворечивости.

Поддержка репозитория

Основными функциями репозитория являются хранение, обновление, анализ, визуализация всей информации по проекту и организация доступа к ней. Репозиторий обычно хранит более 100 типов объектов (например диаграммы, определения экранов и меню, проекты отчетов, описания данных, модели данных, модели обработки, исходные коды, элементы данных) [19].

Каждый информационный объект, хранящийся в репозитории, описывается совокупностью своих свойств, например, идентификатор, тип, текстовое описание, компоненты, область значений, связи с другими объектами, времена создания и последнего обновления объекта, автор и т.п.

Репозиторий является базой для автоматической генерации документации по проекту. Основными ***типами отчетов*** являются:

- *отчеты по содержимому* – включают информацию по потокам данных и их компонентов; списки функциональных блоков диаграмм и их входных и выходных потоков; списки всех информационных объектов и их атрибутов; историю изменений объектов; описания модулей и интерфейсов между ними; планы тестирования модулей и т.п.;
- *отчеты по перекрестным ссылкам* – содержат информацию по связям всех вызывающих и вызываемых модулей; списки объектов репозитория, к которым имеет доступ конкретный исполнитель проекта; информацию по связям между диаграммами и конкретными данными; маршруты движения данных от входа к выходу;

• *отчеты по результатам анализа* – включают данные по взаимной корректности диаграмм, списки неопределенных информационных объектов, списки неполных диаграмм, данные по результатам анализа структуры проекта и т.п.;

• *отчеты по декомпозиции объектов* – включают совокупности объектов, входящих в каждый объект, а также объекты, в состав которых входит каждый объект.

Поддержка процессов жизненного цикла программных средств и систем

Основой поддержки процесса разработки являются следующие свойства современных CASE-средств [19].

1. *Покрытие всего жизненного цикла систем или программных средств.* Как отмечалось в подразд. 6.1, современные CASE-средства поддерживают практически полный ЖЦ ПС. Однако первоочередное внимание уделяется начальным работам процесса разработки – анализу требований к системе, проектированию системной архитектуры, анализу требований к программным средствам и проектированию программной архитектуры (см. подразд. 1.2) Грамотная разработка требований к системе и ПС является основой всего проекта, их полнота и корректность определяют уровень соответствия результатов разработки требованиям заказчика.

2. *Поддержка прототипирования.* Большинство моделей ЖЦ, предназначенных для разработки сложных или критичных продуктов, базируются на применении прототипирования. Это касается в первую очередь моделей, поддерживающих инкрементную и эволюционную стратегии разработки (см. пп. 2.1.3, 2.1.4, 2.3.4, 2.5.3 – 2.5.6). Прототипирование применяется на ранних этапах ЖЦ и позволяет уточнять требования к системе или программному средству, а также прогнозировать поведение разрабатываемого продукта.

3. *Поддержка современных методологий разработки систем или программных средств.* Современные линейки CASE-средств поддерживают, как правило, различные методологии, предназначенные для использования на различных этапах процесса разработки. При этом выполняется графическая поддержка построения диаграмм различных типов, контроль корректности использования шагов проектирования и подготовка документации.

4. *Автоматическая кодогенерация.* Кодогенерация позволяет построить автоматически до 90 % исходных кодов на языках высокого уровня. Различными CASE-средствами поддерживаются практически все известные языки программирования.

Средства кодогенерации можно подразделить на два вида:

- средства генерации управляющей структуры продукта; данные средства выполняют автоматическое построение логической структуры программного средства, кодов для базы данных, файлов, экранов, отчетов. Остальные фрагменты программного средства кодируются вручную;
- средства генерации полного продукта; данные средства позволяют на основе разработанных спецификаций или моделей генерировать полные коды программного средства, пользовательскую и программную документацию к нему.

Резюме

В состав CASE-средств входят средства централизованного хранения информации о проекте (репозиторий), средства ввода, средства анализа и разработки, средства вывода. Все компоненты CASE-средств в совокупности поддерживают графические модели, репозиторий, процесс разработки и ряд вспомогательных и организационных процессов, выполняют контроль ошибок.

6.4. Классификация CASE-средств

Все CASE-средства подразделяются на типы, категории и уровни [19].

6.4.1. Классификация по типам

Данная классификация отражает *функциональное назначение* CASE-средства в ЖЦ ПС и систем.

1. Анализ и проектирование

Средства этого типа используются для поддержки начальных этапов процесса разработки: анализа предметной области, разработки требований к системе, проектирования системной архитектуры, разработки требований к программным средствам, проектирования программной архитектуры, технического проектирования

программных средств (см. подразд. 1.2). Средства данного типа поддерживают известные методологии анализа и проектирования. На выходе генерируются спецификации системы, ее компонентов и интерфейсов, связывающих эти компоненты, архитектура системы, архитектура программного средства, технический проект программного средства, включая алгоритмы и определения структур данных. Таким образом, поддерживаются работы 2 – 6 процесса разработки ПС и систем.

К средствам данного типа можно отнести, например, AllFusion Process Modeler (BPwin), CASE.Аналитик, Design/IDEF, Telelogic DOORS, Telelogic Modeler, Telelogic TAU, Telelogic Rhapsody, Telelogic Statemate (см. подразд. 6.5, 6.6).

2. Проектирование баз данных и файлов

Средства этого типа обеспечивают логическое моделирование данных, автоматическое преобразование моделей данных в третью нормальную форму, автоматическую генерацию схем баз данных и описаний форматов файлов на уровне программного кода. К средствам этого типа можно отнести, например, AllFusion Data Modeler (ERwin), CA ERwin Data Model Validator (ранее ERwin Examiner), S-Designor, Silverrun, Designer2000, Telelogic TAU, Telelogic Rhapsody (см. подразд. 6.5, 6.6).

3. Программирование и тестирование

Средства этого типа поддерживают седьмую работу процесса разработки (программирование и тестирование, см. подразд. 1.2). Данные средства выполняют автоматическую кодогенерацию ПС на основе спецификаций или моделей. Содержат графические редакторы, средства поддержки работы с репозиторием, генераторы и анализаторы кодов, генераторы тестов, анализаторы покрытия тестами, отладчики.

К средствам данного типа можно отнести, например, TAU/Developer, TAU/Tester, Logiscope Audit, Logiscope RuleChecker, Logiscope TestChecker, Logiscope Reviewer, Rhapsody Developer (см. подразд. 6.5).

4. Сопровождение и реинженерия

Общей целью средств этого типа является поддержка корректировки, изменения, преобразования, реинженерия существующей системы, поддержка документации по проекту. К данным средствам относятся средства документирования, анализаторы программ, средства управления изменениями и конфигурацией ПС и систем, средства реструктурирования и реинженерии (реинженерия, реинженеринг – reverse engineering –

обратное проектирование, например, построение спецификаций или моделей по исходным текстам программ), средства обеспечения мобильности, позволяющие обеспечить перенос разработанной системы или программного средства в новое операционное или аппаратное окружение.

Средства реинженерии включают:

- *статические анализаторы* для генерирования схем программного средства из его кодов и оценки влияния модификаций;
- *динамические анализаторы*, включающие трансляторы со встроенными отладочными возможностями;
- *документаторы*, автоматически обновляющие документацию при изменении кода программного средства;
- *редакторы кодов*, автоматически изменяющие при редактировании кодов предшествующие ему структуры, в том числе и спецификации требований;
- *средства доступа к спецификациям*, позволяющие выполнять их модификацию и генерацию модифицированного кода;
- *средства реверсной инженерии*, транслирующие коды в спецификации или модели.

К средствам данного типа можно отнести, например, Telelogic DocExpress, Telelogic Synergy, Telelogic Change, средства линейки AllFusion Change Management Suite (см. подразд. 6.5, 6.6).

Следует отметить, что ряд CASE-средств других типов содержат в своем составе средства реинженерии. Это касается, например, CASE-средств AllFusion Data Modeler, Telelogic Rhapsody.

5. Окружение

К средствам данного типа относятся средства поддержки интеграции CASE-средств и данных. К данному типу можно отнести, например, Telelogic Rhapsody Gateway, Telelogic Rhapsody Interface Pack, AllFusion Data Profiler, AllFusion Model Manager, AllFusion Model Navigator (см. подразд. 6.5, 6.6).

6. Управление проектом

К средствам данного типа относятся средства поддержки процесса управления ЖЦ ПС и систем. Их функциями являются планирование, контроль, руководство,

организация взаимодействия и т.п. К средствам данного типа можно отнести, например, Telelogic Focal Point, Telelogic Dashboard, AllFusion Process Management Suite, ADvisor (см. подразд. 6.5, 6.6).

Резюме

Классификация CASE-средств по типам отражает функциональное назначение CASE-средства в ЖЦ ПС. Выделяют типы CASE-средств, ориентированные на следующие этапы процесса разработки и другие процессы ЖЦ: анализ и проектирование, проектирование баз данных и файлов, программирование и тестирование, сопровождение и реинженерия, окружение, управление проектом.

6.4.2. Классификация по категориям

Данная классификация отражает *уровень интегрированности* CASE-средств по выполняемым функциям.

1. ***Категория Tool*** (tool – рабочий инструмент)

Включает средства самого низкого уровня интегрированности. В данную категорию средств входят инструментальные средства, решающие небольшую автономную задачу при разработке программного средства или системы. Обычно средства данной категории являются компонентами CASE-средств более высокого уровня интегрированности.

2. ***Категория ToolKit*** (toolkit – набор инструментов, пакет разработчика)

Включает CASE-средства среднего уровня интегрированности. Средства данной категории используют репозиторий для всей информации о проекте и ориентированы обычно на поддержку одного этапа или одной работы процесса разработки или на поддержку одного из вспомогательных или организационных процессов ЖЦ ПС или систем. CASE-средства данной категории представляют собой интегрированную совокупность инструментальных средств, имеющих как правило общую функциональную ориентацию.

К CASE-средствам данной категории может быть отнесено, например, большинство CASE-средств из линеек Telelogic и AllFusion при их изолированном использовании (см. подразд. 6.5, 6.6).

3. Категория Workbench (workbench – рабочее место).

CASE-средства данной категории обладают самой высокой степенью интеграции. Они представляют собой интегрированную совокупность инструментальных средств, поддерживающих практически весь процесс разработки и ряд вспомогательных и организационных процессов ЖЦ ПС и систем. Используют репозиторий для хранения информации по проекту, поддерживают организацию коллективной работы над проектом.

Обычно к категории Workbench относятся линейки CASE-средств при их интегральном использовании. Примерами являются линейки Telelogic и AllFusion (см. подразд. 6.5, 6.6). Данные линейки CASE-средств поддерживает практически полностью процесс разработки ПС и систем, процессы сопровождения, документирования, управления конфигурацией, частично процессы обеспечения качества, верификации, аттестации. Таким образом, линейки Telelogic и AllFusion поддерживают практически весь ЖЦ ПС и систем.

Резюме

Классификация по категориям отражает уровень интегрированности CASE-средств по выполняемым функциям. Различают категории Tool, ToolKit, Workbench.

6.4.3. Классификация по уровням

Данная классификация связана с *областью действия* CASE-средств в ЖЦ ПС, систем и организаций.

1. Верхние (Upper) CASE-средства

CASE-средства данного уровня называют средствами компьютерного планирования. Их основной целью является помощь руководителям организаций, предприятий и конкретных проектов в определении политики организации и в создании планов проекта. CASE-средства данного уровня позволяют строить модель предметной области, проводить анализ различных сценариев (существующего, наилучших, наихудших), накапливать информацию для принятия оптимальных решений. Таким образом, применительно к ЖЦ ПС и систем данные средства поддерживают процесс заказа и первую работу процесса разработки (подготовка процесса разработки).

Графические средства данного уровня используются как формализованный язык общения между заказчиком (пользователем) и разработчиком требований (см. п. 5.2.6).

К средствам данного уровня можно отнести, например, Telelogic System Architect, Telelogic Focal Point, Telelogic Dashboard, средства линейки AllFusion Modeling Suite (см. подразд. 6.5, 6.6).

2. Средние (Middle) CASE-средства

CASE-средства данного уровня поддерживают начальные этапы процесса разработки (анализ предметной области, разработка требований к системе, проектирование системной архитектуры, разработка требований к программным средствам, проектирование программной архитектуры). Таким образом, фактически поддерживаются работы 2 – 6 процесса разработки (см. подразд. 1.2). При этом встроенные графические средства используются как формализованный язык общения между заказчиком (пользователем) и разработчиком спецификаций требований (см. п. 5.2.6).

Обычно данные средства обладают возможностями накопления и хранения информации по проекту. Это позволяет использовать накопленные данные как в текущем, так и в других проектах. Например, с помощью накопленной информации могут оцениваться продукты текущего проекта. При этом аналогичная информация предыдущих проектов используется в качестве базовой для оценки.

CASE-средства данного уровня зачастую поддерживают прототипирование и автоматическое документирование.

К CASE-средствам данного уровня можно отнести, например, линейку AllFusion Modeling Suite, средства Telelogic DOORS, Telelogic Modeler, Telelogic Tau, Telelogic Rhapsody, Telelogic Statemate, Telelogic DocExpress (см. подразд. 6.5, 6.6).

3. Нижние (Lower) CASE-средства

CASE-средства данного уровня поддерживают вторую половину работ процесса разработки ПС (как правило, начиная с шестой работы, см. п. 5.2.6). Содержат графические средства, исключающие необходимость разработки физических миниспецификаций для программных модулей. Спецификации представляются обычно в виде моделей, которые непосредственно преобразуются в программные коды разрабатываемого программного средства или системы. Автоматически генерируется до

90 % кодов. Входной информацией для кодогенераторов являются спецификации, разработанные как в CASE-средствах данного уровня, так и в CASE-средствах среднего уровня.

CASE-средства нижнего уровня, как правило, поддерживают также прототипирование, тестирование, управление конфигурацией, генерацию документации, облегчают модификацию и сопровождение ПС или систем.

К CASE-средствам данного уровня можно отнести AllFusion Data Modeler, Telelogic Rhapsody, Telelogic Tau, Telelogic Statemate, Telelogic TAU Logiscope, Telelogic Change, Telelogic Synergy, Telelogic DocExpress (см. подразд. 6.5, 6.6).

Следует отметить, что в состав CASE-средств среднего и высокого уровня интегрированности обычно входят инструментальные средства, относящиеся к нескольким уровням. Линейки CASE-средств, предназначенные для поддержки всего ЖЦ ПС и систем, включают в свой состав средства всех трех уровней.

Резюме

Классификация по уровням связана с областью действия CASE-средств в ЖЦ ПС и систем. Различают верхние, средние и нижние CASE-средства. Линейки CASE-средств включают в свой состав средства всех трех уровней.

6.5. Инструментальные средства Telelogic, предназначенные для автоматизации жизненного цикла организаций, систем и программных средств

К современным инструментальным средствам, обеспечивающим эффективную поддержку ЖЦ организаций в целом, систем и ПС, относятся интегрированные CASE-средства ***Telelogic*** [9]. В их состав входят средства поддержки оптимальных методов разработки структуры организации и увязывания ее бизнес-процессов с используемыми IT-технологиями и IT-системами; средства принятия решений, средства управления

организацией, проектами, требованиями и изменениями; средства проектирования и моделирования систем и ПС на основе языков UML (Unified Modeling Language), SysML (Systems Modeling Language), SDL (Specification and Description Language). Все средства имеют репозиторий и поддерживают автоматическое создание документации.

Использование данных средств значительно сокращает общие затраты на разработку и сопровождение ПС и систем, снижает продолжительность разработки, повышает качество процессов ЖЦ, а следовательно, и качество промежуточных и конечного продуктов разработки.

Основными среди инструментальных средств Telelogic являются следующие:

• семейство Telelogic System Architect – поддерживает построение архитектуры предприятия. Предоставляет возможность создания и взаимной увязки интегрированного набора моделей и документов для четырех ключевых областей архитектуры предприятия: бизнеса, информации, IT-систем, технологий. Обеспечивает поддержку всех областей моделирования, в том числе моделирование бизнес-процессов, компонентное и объектное моделирование с помощью UML, моделирование данных, структурный анализ и проектирование. CASE-средства данного семейства могут использоваться заказчиком в процессе заказа ЖЦ ПС и систем (см. подразд. 1.2). В состав семейства Telelogic System Architect входят:

- Telelogic System Architect – включает все инструменты, необходимые для успешного создания архитектуры предприятия;
- Telelogic System Architect для DoDAF – поддерживает архитектуру предприятий на основе стандарта DoDAF;
- Telelogic System Architect/Publisher – поддерживает создание WEB-сайта, содержащего полный набор всех моделей архитектуры предприятия, автоматически создает и публикует разносторонние отчеты о моделях;
- Telelogic System Architect для FEA – поддерживает архитектуру предприятий для правительственных учреждений;
- Telelogic System Architect/XT – представляет собой WEB-решение для архитектуры предприятия;
- Telelogic System Architect/ERP – позволяет в исследовательских целях извлекать метаданные из приложений, используемых на предприятии,

просматривать сложные внутренние взаимосвязи системы планирования ресурсов предприятия;

- Telelogic Focal Point – Web-система, поддерживающая принятие решений при управлении требованиями, проектами, рисками, разработке продуктов, планировании сценариев, выборе проектов и продуктов, выполнении любой другой деятельности, связанной со сбором и анализом информации в масштабах производства. Может использоваться в процессах заказа, поставки, разработки, управления ЖЦ ПС и систем (см. подразд. 1.2);
- Telelogic Dashboard – поддерживает контроль за разработкой и принятие решений менеджерами проектов; поддерживает оценку состояния, рисков и тенденций проекта; позволяет автоматизировать сбор метрической информации из Telelogic Change и Telelogic DOORS, проводить ее анализ и получать отчетность по важным характеристикам; предоставляет метрики оценки проектов, в том числе оценки требований, изменений, конфигурации. Может использоваться в процессах обеспечения качества и управления ЖЦ ПС и систем (см. подразд. 1.2);
- Семейство Telelogic DOORS – семейство, предназначенное для управления требованиями к ПС или системам. Поддерживает сбор требований различного уровня, автоматическую трассировку требований, управление внесением изменений в требования, анализ влияния изменений на другие требования. Может использоваться в процессах разработки и сопровождения ЖЦ ПС и систем (см. подразд. 1.2). В состав семейства Telelogic DOORS входят:
  - Telelogic DOORS – поддерживает управление требованиями при создании инженерных систем или разработке ПО; обеспечивает комплексную поддержку записи, регистрации, структурирования, управления и анализа требований и их трассировки; предоставляет возможность контроля за изменением требований;
  - Telelogic DOORS/XT – предназначено для управления требованиями в компаниях глобального уровня;
  - Telelogic DOORS/Analyst – используется для моделирования требований на основе UML в рамках процесса управления требованиями; позволяет

создавать модели, рисунки и диаграммы требований непосредственно в DOORS;

- Telelogic DOORS/Net – обеспечивает Web-доступ к управлению требованиями для удаленных пользователей;
- Telelogic DOORS Fastrak – предназначено для управления требованиями в проектах с короткими сроками реализации; поддерживает Web-доступ;

• Telelogic Modeler – средство проектирования ПО на основе стандартного графического языка (UML), позволяющего визуализировать проектирование систем и ПС. Распространяется бесплатно. Может использоваться в процессах разработки и сопровождения ЖЦ ПС и систем (см. подразд. 1.2);

• Семейство Telelogic Tau – семейство, предназначенное для моделирования требований к ПС или системам, проектирования моделей их архитектуры, создания тестов и тестирования моделей, автоматической кодогенерации на основе проверенных моделей. Поддерживает отраслевые стандарты визуального моделирования UML, SysML, SDL. Обеспечивает системное и интеграционное тестирования на основе стандартов TTCN-2 и TTCN-3. Позволяет унифицировать язык общения между системными аналитиками, проектировщиками, программистами и другими разработчиками. Может использоваться в процессах разработки и сопровождения ЖЦ ПС и систем (см. подразд. 1.2). В состав семейства Telelogic Tau входят:

- Telelogic Tau – служит для разработки систем и ПС с использованием языков моделирования UML и SysML; поддерживает этапы ЖЦ, начиная от разработки архитектуры системы или программного средства и заканчивая кодогенерацией и автоматическим созданием тестов; поддерживает промышленные стандарты визуализации разработки и тестированиия систем и ПС (UML 2.1, SysML, MDA, DoDAF, SOA, UML Testing Profile). Содержит следующие инструменты:
    - TAU/Model Author – используется авторами моделей предметной области; позволяет рисовать диаграммы моделей;

- TAU/Architect – используется системными инженерами; поддерживает возможности TAU/Model Author и верификацию диаграмм;
- TAU/Developer – используется разработчиками ПС; поддерживает возможности TAU/Architect и генерацию кода;
- TAU/Tester – используется тестировщиками ПС;

- Telelogic SDL Suite – поддерживает разработку ПО систем реального времени; предоставляет возможности создания ПО для сложных, событийно-управляемых коммуникационных систем, описываемых с использованием стандарта SDL; обеспечивает визуализацию проектирования и автоматическую кодогенерацию;
- Telelogic TTCN Suite – поддерживает автоматизацию тестирования систем реального времени и телекоммуникационных систем на основе скриптового языка сценариев тестирования TTCN-2 (Tree and Tabular Combined Notation);
- Telelogic Tau/DoDAF – поддерживает проектирование архитектуры систем, совместимых со стандартом DoDAF (Department of Defense Architecture Framework) – структурой архитектуры Министерства обороны США; поддерживает моделирование на UML и SysML, атоматическую кодогенерацию, автоматическое обнаружение ошибок;
- Telelogic Tester – поддерживает тестирование ПС на основе стандарта TTCN-3 (Testing and Test Control Notation); предоставляет возможности автоматизиции системного и интеграционного тестирования; поддерживает весь ЖЦ тестирования ПС (постановку задачи тестирования, разработку сценариев, анализ, исполнение, отладку);

- семейство Telelogic Rhapsody – представляет среду разработки на основе моделей (Model-Driven Development). Поддерживает языки моделирования UML и SysML. Предоставляет возможность создания языка предметной области DSL (Domain Specific Language), позволяющего создавать новые диаграммы и элементы диаграмм, приближенные к конкретной предметной области. Обеспечивает проектирование, кодогенерацию и тестирование встраиваемых систем и ПО реального времени. При

проектировании систем использует функционально-ориентированный подход, при проектировании ПО – объектно-ориентированный подход. Содержит средства реинженерии. Может использоваться в процессах разработки и сопровождения ЖЦ ПС и систем (см. подразд. 1.2). Семейство Telelogic Rhapsody состоит из следующего набора модулей и пакетов:

- Rhapsody System Architect – поддерживает визуализацию, разработку спецификаций и документирование систем;
- Rhapsody Systems Designer – поддерживает визуализацию, разработку спецификаций, аттестацию и документирование систем;
- Rhapsody Architect – поддерживает визуализацию, разработку спецификаций и документирование ПО;
- Rhapsody Developer – поддерживает визуализацию, разработку спецификаций, аттестацию, кодогенерацию и документирование ПО;
- Пакет AUTOSAR – поддерживает проектирование модулей систем на базе стандарта AUTOSAR;
- Пакет DoDAF – поддерживает проектирование и моделирование спецификаций систем на базе стандарта DoDAF;
- Gateway – обеспечивает интеграцию с DOORS и другими инструментами управления требованиями;
- Test Conductor – поддерживает функциональное тестирование на основе требований;
- Automatic Test Generation – поддерживает автоматическое создание тестов на основе модели;
- Interface Pack – обеспечивает интеграцию с инструментами управления конфигурацией, интеграцию с Simulink, импорт моделей Rational Rose, поддержку XMI;
- Tools & Utilities Pack – поддерживает быстрое прототипирование и настраиваемое создание документов на основе Web-интерфейса пользователя;
- Value Pack – сочетает возможности Tools & Utilities Pack и Interfaces Pack;
- Gateway Value Pack – сочетает возможности Gateway и Value Pack;

- Teamcenter System Engineering Interface – поддерживает управление жизненным циклом продукта; представляет собой результат интеграции Teamcenter и Telelogic Rhapsody;

• Telelogic Statemate – предназначено для разработки прототипов. Поддерживает моделирование, графическое представление системных требований, исполняемых спецификаций и проектирования, автоматическую генерацию кодов и тестов прототипов для быстрой разработки сложных встраиваемых систем. Использует стандартные диаграммы проектирования и языка UML. Позволяет обнаруживать и исправлять ошибки на ранних этапах разработки систем. Может использоваться в процессах разработки и сопровождения ЖЦ ПС и систем (см. подразд. 1.2);

• семейство Telelogic DocExpress – поддерживает автоматизацию документирования ПО, упрощает управление отчетностью. Предоставляет возможности сбора из различных инструментов Telelogic, форматирования и публикации проектной и технической документации в интерактивном режиме. Поддерживает исходные данные для документирования и документацию в актуальном состоянии. Упрощает сопровождение документов. Может использоваться в процессе документирования ЖЦ ПС и систем (см. подразд. 1.2). В состав семейства Telelogic DocExpress входят:

- Telelogic DocExpress Word – поддерживает представление документации в Microsoft Word; обеспечивает легкое форматирование различных типов данных и их включение в один документ; позволяет экспортировать в Word проектную информацию, в том числе структуры данных и модели;
- Telelogic DocExpress Factory – обеспечивает автоматизированное составление отчетности в автономном режиме для выбора, сбора и публикации проектной документации в различных форматах (например Adobe FrameMaker, Microsoft Word, HTML);

• Telelogic Logiscope – поддерживает оценку и обеспечение качества ПС; помогает увеличить тестовое покрытие, автоматизирует анализ кода, идентификацию модулей, которые могут содержать ошибки. Может использоваться в процессах разработки, сопровождения, обеспечения качества, верификации, аттестации ЖЦ ПС и систем (см. подразд. 1.2). Данное семейство состоит из следующего набора инструментов:

- Logiscope Audit – поддерживает оценку качества и графический анализ исходных программных кодов;
- Logiscope RuleChecker – позволяет выполнять проверку исходного кода на соответствие принятым правилам;
- Logiscope TestChecker – выполняет проверку степени покрытия исходного кода тестовыми наборами;
- Logiscope Reviewer –сочетает возможности Logiscope Audit и Logiscope RuleChecker;

• Семейство Telelogic Synergy – поддерживает управление изменениями и конфигурацией ПС. Предоставляет возможность оценки и авторизации запросов на изменения. Позволяет управлять версиями ПС. Может использоваться в процессах разработки, сопровождения, управления конфигурацией ЖЦ ПС и систем (см. подразд. 1.2). В состав семейства Telelogic Synergy входят:

- Telelogic Change – обеспечивает Web-доступ к управлению изменениями; позволяет отслеживать статус запросов на изменения на протяжении их ЖЦ; содержит централизованный репозиторий; позволяет составлять отчеты по изменениям требуемого формата;
- Telelogic Synergy – поддерживает процесс управления конфигурацией ЖЦ ПС (см. подразд. 1.2), компонентно-ориентированную разработку (см. п. 2.5.6), управление выпуском версий; содержит мощный распределенный репозиторий; поддерживает коллективную разработку продукта.

Резюме

CASE-средства Telelogic интегрируются друг с другом. Их совместное использование поддерживает практически весь ЖЦ ПС и систем, регламентированный стандартом *СТБ ИСО/МЭК 12207–2003*.

6.6. Инструментальные средства Computer Associates, предназначенные для автоматизации жизненного цикла организаций, систем и программных средств

К современным инструментальным средствам, обеспечивающим эффективную поддержку ЖЦ организаций или предприятий, систем и ПС, относится линейка AllFusion компании Computer Associates. Данная линейка представляет собой набор интегрированных средств для разработки, внедрения и управления информационными системами в организации. Все средства линейки имеют репозиторий и поддерживают автоматическое создание документации. Совокупное использование средств линейки AllFusion покрывает практически весь ЖЦ ПС и систем, регламентированный стандартом *СТБ ИСО/МЭК 12207–2003,* позволяет сократить затраты на их разработку и сопровождение, снижает продолжительность и стоимость разработки, повышает качество процессов ЖЦ, качество промежуточных и конечных продуктов разработки.

По функциональному назначению основные компоненты линейки AllFusion можно подразделить на следующие типы [26]:

- средства моделирования баз данных, бизнес-процессов и компонентов ПО;
- средства управления изменениями и конфигурациями;
- средства интегрированного управления проектами и процессами.

Средства моделирования баз данных, бизнес-процессов и компонентов ПО представлены линейкой ***CA ERwin Modeling Suite* (*AllFusion Modeling Suite*)**. Данная линейка предназначена для анализа и моделирования различных аспектов деятельности организации и проектирования информационных систем. Средства данной линейки позволяют моделировать предметные области, базы данных, компоненты ПО, деятельность и структуру организаций. CASE-средства линейки CA ERwin Modeling

Suite поддерживают структурные методы анализа и проектирования, базирующиеся на использовании методологий моделирования IDEF0, DFD, IDEF3 и IDEF1X. CASE-средства данной линейки поддерживают процессы разработки, сопровождения, верификации, аттестации и документирования ЖЦ ПС и систем.

Основными в данной линейке являются следующие средства:

- CA ERwin Process Modeler (AllFusion Process Modeler, называемый ранее BPwin) – средство функционального моделирования предметной области. Поддерживает три методологии моделирования – IDEF0 (функциональное моделирование), DFD (моделирование потоков данных) и IDEF3 (моделирование потоков работ), что позволяет комплексно описывать предметную область. Поддерживает методы функционально-стоимостного анализа хозяйственной деятельности организаций. Содержит генератор отчётов, имеет широкий набор средств документирования моделей и проектов. Методологии IDEF0 и DFD подробно описаны в подразд. 5.2, 5.3;
- CA ERwin Data Modeler (AllFusion Data Modeler, ранее ERwin) – средство проектирования, документирования и сопровождения баз данных и хранилищ данных. Поддерживает методологии информационного моделирования IDEF1X, IE, Dimensional. Содержит средства автоматической кодогенерации баз данных, средства их сопровождения и реинженерии; поддерживает различные типы СУБД. Содержит генератор отчётов, имеет широкий набор средств документирования моделей и проектов. Методология IDEF1X подробно описана в подразд. 5.4;
- CA ERwin Data Profiler (AllFusion Data Profiler) – средство для анализа и профилирования исходной информации, поступающей из множественных источников данных. Предоставляет возможности профилирования данных для ряда платформ, включая реляционные базы данных, электронные таблицы и наследуемые неструктурированные файлы. Позволяет улучшить повторное использование данных, предоставляет средства миграции и интеграции данных. Позволяет генерировать метрики и статистику качества данных;
- CA ERwin Data Model Validator (ранее ERwin Examiner) – средство для проверки структуры баз данных и качества моделей CA ERwin Data Modeler;
- CA ERwin Model Manager (ранее ModelMart) – среда для совместного моделирования в CA ERwin Data Modeler и/или CA ERwin Process Modeler;

- CA ERwin Saphir Option – средство обнаружения метаданных. Предназначено для извлечения, анализа и публикации метаданных из пакетированных корпоративных приложений. Преобразует специфические для пакета метаданные в модели данных CA ERwin Data Modeler;
- CA ERwin Model Navigator – инструмент для просмотра моделей, созданных в CA ERwin Data Modeler и CA ERwin Process Modeler. Предоставляет доступ ко всему набору отчетов из CA ERwin Data Modeler и CA ERwin Process Modeler для анализа и разработки приложений.

Средства управления изменениями и конфигурациями представлены линейкой ***AllFusion Change Management Suite*.** Данная линейка предназначена для управления изменениями на различных стадиях ЖЦ разработки приложений и выполнения интегрированных задач по настройке ПО всей организации, включая разработку многоплатформенного ПО и Web-приложений. Средства данной линейки поддерживают процессы разработки, сопровождения, управления конфигурацией и документирования ЖЦ ПС и систем.

Основными в данной линейке являются следующие средства:

- AllFusion Harvest Change Manager (CCC/Harvest) – средство для управления изменениями, конфигурациями и версиями при разработке сложных корпоративных систем на основе общего репозитория; помогает синхронизировать деятельность разработчиков на различных платформах в течение всего жизненного цикла;
- AllFusion Change Manager Enterprise Workbench – средство автоматизации процесса разработки ПО на различных платформах; поддерживает интеграцию функций управления изменениями в приложениях в масштабе организации;
- AllFusion Endevor Change Manager – средство управления изменениями, версиями, конфигурациями для мэйнфреймов; поддерживает разработку приложений в среде OS/390, Windows и UNIX;
- AllFusion CA-Librarian – средство управления библиотеками при разработке на мэйнфреймах; обеспечивает безопасность и осуществляет контроль программных ресурсов организации;

- AllFusion CA-Panvalet – средство управления кодом разрабатываемых приложений (для мэйнфреймов); предоставляет возможности централизованного хранения исходного кода приложений.

Средства интегрированного управления проектами и процессами предназначены для управления готовыми продуктами, проектами, ресурсами и передаваемыми файлами на предприятии. Средства данного типа поддерживают в первую очередь процесс управления ЖЦ ПС и систем. В состав средств данного типа входят:

- AllFusion Process Management Suite (ранее Process Continuum) – представляет собой полный набор инструментальных средств для эффективного управления процессом разработки ПО, в том числе методиками, проектами, ресурсами и конечными результатами процесса разработки;
- ADvisor – веб-среда для разработки приложений, управления информацией и работами.

Резюме

Инструментальные средства линейки AllFusion компании Computer Associates подразделяются на средства моделирования баз данных, бизнес-процессов и компонентов ПО; средства управления изменениями и конфигурациями; средства интегрированного управления проектами и процессами. Их совместное использование поддерживает практически весь ЖЦ ПС и систем, регламентированный стандартом *СТБ ИСО/МЭК 12207–2003*.

ЛИТЕРАТУРА

1. ISO/IEC/IEEE 31320-1:2012. Информационные технологии – Языки моделирования – Часть 1: Синтаксис и семантика для IDEF0. – Введ. 2012-09-15. – Женева, 2012.

2. ISO/IEC/IEEE 31320-2:2012. Информационные технологии – Языки моделирования – Часть 2: Синтаксис и семантика для IDEF1X97 (IDEFobject). –Введ. 2012-09-15. – Женева, 2012.

3. ISO/IEC 12207:1995. Информационная технология – Процессы жизненного цикла программных средств. – Введ. 1995-08-01. – Женева : ISO/IEC, 1995.

4. ISO/IEC 12207:2008. Системная и программная инженерия – Процессы жизненного цикла программных средств. – Введ. 2008-02-01. – Нью-Йорк : ISO/IEC-IEEE, 2008.

5. ГОСТ Р ИСО/МЭК ТО 12182–2002. Информационная технология. Классификация программных средств. – Введ. 2002-06-11. – М. : Изд-во стандартов, 2002.

6. ГОСТ Р ИСО/МЭК ТО 15271–2002. Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207 (Процессы жизненного цикла программных средств). – Введ. 2002-06-05. – М. : Изд-во стандартов, 2002.

7. СТБ ИСО/МЭК 12207–2003. Информационные технологии. Процессы жизненного цикла программных средств. – Введ. 2003-03-19. – Минск : Госстандарт Республики Беларусь, 2003.

8. СТБ 2195-2011. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования. – Введ. 2012-01-01. – Мн. : Гостстандарт, 2011.

9. Бахтизин, В. В. Технология разработки программного обеспечения : учеб. пособие / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2010.

10. Бахтизин, В. В. Методология функционального проектирования IDEF0 : учеб. пособие по курсу «Технология разработки программного обеспечения» для студ. спец. 40 01 01 «Программное обеспечение информационных технологий» / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2003.

11. Бахтизин, В. В. Работа с требованиями в среде DOORS : учеб.-метод. пособие / В. В. Бахтизин, Л. А. Глухова, С. Н. Неборский. – Минск : БГУИР, 2007.

12. Бахтизин, В. В. Стандартизация и сертификация программного обеспечения : учеб. пособие / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2006.

13. Бахтизин, В. В. Структурный анализ и моделирование в среде CASE-средства BPwin : учеб. пособие по курсу «Технология проектирования программ» для студ. спец. 40 01 01 «Программное обеспечение информационных технологий» / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2002.

14. Брауде, Э. Технология разработки программного обеспечения / Э. Брауде. – СПб. : Питер, 2004.

15. Глухова, Л. А. Методология структурного анализа и проектирования SADT: учеб. пособие по курсу «Технология проектирования программ» для студ. спец. Т.10.02.00 / Л. А. Глухова, В. В. Бахтизин. – Минск : БГУИР, 2001.

16. Глухова, Л. А. Информационное моделирование с помощью CASE-средства ERwin 3.0 : учеб. пособие по курсу «Технология проектирования программ» для студ. спец. Т.10.02.00 «Программное обеспечение информационных технологий» / Л. А. Глухова, В. В. Бахтизин. – Минск : БГУИР, 1999.

17. Зиглер, К. Методы проектирования программных систем / К. Зиглер. – М. : Мир, 1985.

18. Йодан, Э. Структурное проектирование и конструирование программ / Э. Йодан. – М. : Мир, 1979.

19. Калянов, Г. Н. CASE-технологии: консалтинг в автоматизации бизнес-процессов / Г. Н. Калянов. – Москва : Горячая линия-Телеком, 2002.

20. Кинг, Д. Создание эффективного программного обеспечения / Д. Кинг. – М. : Мир, 1991.

21. Маклаков, С. В. Создание информационных систем с AllFusion Modeling Suite / С. В. Маклаков. – М. : ДИАЛОГ-МИФИ, 2005.

22. Марка, Д. Методология структурного анализа и проектирования SADT / Д. Марка, К. МакГоуэн. – М. : МетаТехнология, 2003.

23. Орлов, С. А. Технологии разработки программного обеспечения : учебник для ВУЗов / С. А. Орлов, Б. Я. Цилькер. – СПб. : Питер, 2012.

24. Руководство к Своду знаний по управлению проектами (Руководство PMBOK). - Пятое издание. - Project Management Institute, 2013.

25. Рудаков, А. В. Технология разработки программных продуктов. Практикум / А. В. Рудаков, Г.Н. Федорова. – М.: Академия, 2014.

26. Средства моделирования (CASE) и поддержки всех стадий разработки ПО [Электронный ресурс]. – 2015. – Режим доступа: http://www.interface.ru/home.asp?artId=99.

27. Фатрелл, Р. Управление программными проектами: достижение оптимального качества при минимуме затрат / Р. Фатрелл, Д. Шафер, Л. Шафер. – М. : Вильямс, 2003.

28.Черемных, С. В. Моделирование и анализ систем. IDEF-технологии: практикум / С. В. Черемных, И. О. Семенов, В. С. Ручкин. – М. : Финансы и статистика, 2005.

29. Jackson, M. A. A system development method / M. A. Jackson [Электронный ресурс]. – 2015. – Режим доступа: http://mcs.open.ac.uk/mj665/Nice1981.pdf.

30.Jackson System Development [Электронный ресурс]. – 2015. – Режим доступа: http://en.wikipedia.org/wiki/Jackson_System_Development.

31.Software Development Methodology Today. Software Development Strategies and Life-Cycle Models [Электронный ресурс]. – 2015. – Режим доступа: http://www.informit.com/articles/article.aspx?p=605374&seqNum=2.