Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Северный (Арктический) федеральный университет
имени М.В. Ломоносова»

А.И. Попов

Свободные инструменты проектирования информационных систем

Учебно-методическое пособие

Архангельск
ИПЦ САФУ
2012

УДК 004.415.2
ББК 638.72
П58

Рекомендовано к изданию редакционно-издательским советом Северного (Арктического) федерального университета имени М.В. Ломоносова

Рецензенты:
доктор технических наук, профессор **В.И. Воробьев**,
кандидат технических наук **А.А. Докучаев**

Попов, А.И.
П58 Свободные инструменты проектирования информационных систем: учеб.-метод. пособие / А.И. Попов; Сев. (Арктич.) федер. ун-т им. М.В. Ломоносова. – Архангельск: ИПЦ САФУ, 2012. – 78 с.: ил.
ISBN 978-5-261-00747-0

В пособии рассматриваются основные графические нотации, применяемые при проектировании программных систем. Языкам графического моделирования сопоставляются функциональные возможности ряда свободных CASE-инструментов. Проводится обзор и анализ таких инструментов, даются рекомендации по их установке и использованию. Также предлагается материал для выполнения лабораторных работ, связанных с разработкой системы требований к программным продуктам.

Издание предназначено для студентов, обучающихся по математическим и информационным направлениям и специальностям.

УДК 004.415.2
ББК 638.72

ISBN 978-5-261-00747-0

Оглавление

ВВЕДЕНИЕ

Проведение лабораторных занятий по ряду дисциплин, связанных с разработкой программных систем, предполагает применение CASE-средств. В качестве примеров таких дисциплин можно назвать «Проектирование информационных систем», «Технологии разработки программного обеспечения», «Верификация моделей и программ», «Тестирование программного обеспечения», «Разработка и стандартизация программных средств и информационных систем», «CASE-технологии и язык UML» и др.

В данном пособии рассматриваются графические языки, широко используемые в ИТ-проектах, и программное обеспечение, позволяющее работать с такими языками. Данные, процессы, взаимосвязь данных и процессов, взаимосвязь сущностей друг с другом, взаимодействие объектов, взаимодействие проектируемой системы с внешней средой, взаимосвязь работ, выполняемых в ходе реализации проектов, – все это может быть описано при помощи многочисленных диаграмм, выполняемых в различных графических нотациях. Соответственно, можно назвать разные сферы применения таких нотаций: разработка и анализ программного обеспечения, работа с бизнес-процессами, проектирование информационных систем, управление проектами.

Технологии разработки программного обеспечения предполагают построение ряда графических моделей на стадии формирования системы требований к программному продукту – одной из наиболее ответственных задач проекта, решаемой в условиях трудной формализуемости [50]. Соответствующий процесс содержит следующие этапы:

1) формирование первичных требований;
2) анализ и систематизация требований;
3) углубленный анализ и детализация требований;
4) управление требованиями в ходе реализации проекта.

Результатом первого этапа является массив неструктурированных и несогласованных первичных требований, выраженных на языке заказчика. При этом требования предоставляются различными источниками, среди которых – руководство заказчика, пользователи, предметная область, рынок, общество, технические регламенты и стандарты, ИТ-отрасль и др. При анализе первичных требований этот массив приводится в порядок с учетом позитивных и негативных взаимосвязей характеристик качества. Так, например, обычно считается, что функциональность и надежность продукта отрицательно сказываются друг на друге, а мобильность и сопровождаемость – положительно. Данный этап заканчивается фиксацией требований в документации, например, в техническом задании.

Наконец, при углубленном анализе требований моделируется все, что является существенным в предметной области. Здесь применяется множество методов моделирования и соответствующих графических нотаций. Например, применение диаграмм состояний-переходов – формализма из теории автоматов, раскрывающего динамику системы, – относится к математическим методам моделирования. При алгоритмических методах широко используются схемы алгоритмов и программ, IDEF0-диаграммы, диаграммы потоков данных. К инструментам объектных методов относятся диаграммы «сущность – связь» и язык UML. Результатом углубленного анализа требований являются спецификации.

Стремление к повышению эффективности бизнес-процессов организаций привело к появлению ряда относительно новых графических нотаций. Среди них EPC и BPMN. Стоит, однако, заметить, что они основаны на давно известных схемах алгоритмов и программ.

Эффективное управление проектами также требует наличия средств компактного и наглядного представления работ проекта с расположением работ на временной оси, отображением связей работ с необходимыми ресурсами и работ между собой, демонстрацией проблемных мест в графике проекта и т.д. Такими средствами являются диаграммы Гантта, сетевые графики, диаграммы использования ресурсов и др.

Далее в работе делается обзор некоторых наиболее доступных CASE-инструментов. В основном это свободное программное обеспечение. Однако в настоящее время существует тенденция, в соответствии с которой крупные производители профессиональ-

ных CASE-систем создают бесплатные образовательные версии своих коммерческих продуктов. В пособии рассмотрены две такие системы.

Данное пособие – не учебник по проектированию или сборник конкретных заданий и рекомендаций для проведения лабораторных работ. Это скорее путеводитель по наиболее распространенным графическим нотациям и наиболее доступным CASE-системам. Источников информации по рассматриваемой тематике существует огромное количество. Дадим здесь их небольшой обзор в виде списка.

1. Стандартизация в области разработки программного обеспечения и систем:

– организации по стандартизации [15; 65];

– жизненный цикл программного обеспечения и систем [17; 43; 46; 47];

– документирование программного обеспечения [41], в частности разработка технического задания [44];

– качество программных продуктов [16; 19; 18].

2. Учебники:

– технологии разработки программного обеспечения [52; 58];

– проектирование информационных систем [49; 63], структурный [31] и объектно-ориентированный [48] подходы;

– инструментальные средства поддержки жизненного цикла, CASE-системы [36];

– стандартизация и разработка программного обеспечения информационных систем [50];

– анализ программного обеспечения: оценка сложности, тестирование, верификация [53; 58];

– управление проектами [54; 56; 60];

– бизнес-процессы, экономические информационные системы [37; 64].

3. Интернет:

– бесплатные курсы (в том числе видеолекции) по проектированию информационных систем в интернет-университете INTUIT.ru [13; 14];

– обучающее видео на английском языке [30];

– CASE-системы [1; 2; 3; 4; 7; 8; 10; 11; 22; 24; 28; 32; 59; 62];

– графические нотации [5; 20; 29; 35];

– Википедия – свободная энциклопедия [38].

Учебное пособие построено следующим образом. Сначала дано краткое описание некоторых графических нотаций, приведены примеры диаграмм. Далее рассмотрено программное обеспечение. Для каждого программного продукта представлены общие сведения, функциональные возможности, особенности установки, платформы, информационное обеспечение, способы расширения возможностей. Приведены примеры экранных форм. В завершение даны краткие текстовые описания ряда предметных областей, которые можно использовать для организации учебных проектов, и глоссарий.

Глава 1
ОБЗОР МЕТОДОЛОГИЙ ГРАФИЧЕСКОГО АНАЛИЗА

1.1. Схемы алгоритмов и программ

При процедурном подходе для визуального представления алгоритма выполнения программы применяются схемы алгоритмов и программ. Их применение соответствует алгоритмическому походу к углубленному анализу требований к программному продукту [50]. Для описания структурных алгоритмов наряду со схемами алгоритмов и программ используются псевдокоды, Flow-формы и диаграммы Насси – Шнейдермана [52].

Терминология, система графических обозначений, правила выполнения схем алгоритмов зафиксированы в ряде государственных стандартов в 1980–1990-х годов [39; 40; 42].

При построении схем алгоритмов и программ используются следующие группы символов: символы данных (рис. 1), символы процесса (рис. 2), линии (рис. 3) и специальные символы (рис. 4). Некоторые символы являются устаревшими, например символы, описывающие хранение данных на перфолентах и перфокартах.

Пример схемы программы приведен на рисунке 5. В примере использованы символы начала и завершения программы, символы процессов, ветвление «ИЛИ», ветвление и слияние «И», ввод/вывод данных, комментарий. Диаграмма описывает следующий алгоритм. Сначала выполняется «Процесс 1», затем проверяется «условие». Если «условие» выполнено, запускаются параллельные процессы «Процесс 2» и «Процесс 3», если нет – выполняется «Процесс 4». Комментарий в данном случае указывает на параллельные процессы. Вывод данных происходит либо после завершения «Процесса 4», либо после завершения «Процесса 2» и «Процесса 3».

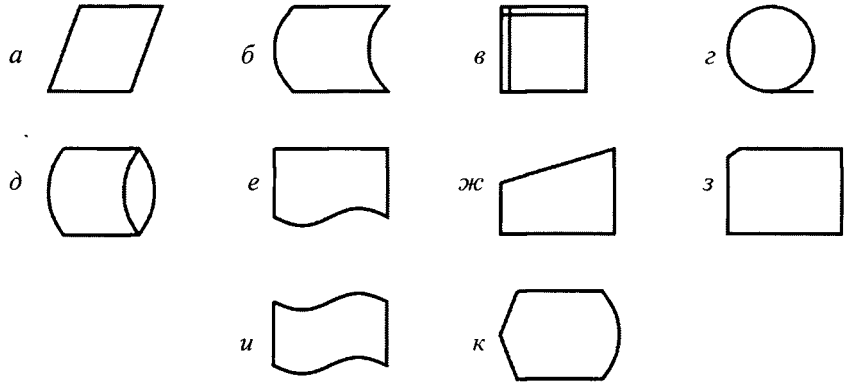

Рис. 1. Символы данных: *а* – данные, *б* – запоминаемые данные, *в* – оперативное запоминающее устройство, *г* – запоминающее устройство с последовательным доступом, *д* – запоминающее устройство с прямым доступом, *е* – документ, *ж* – ручной ввод, *з* – карта, *и* – бумажная лента, *к* – дисплей

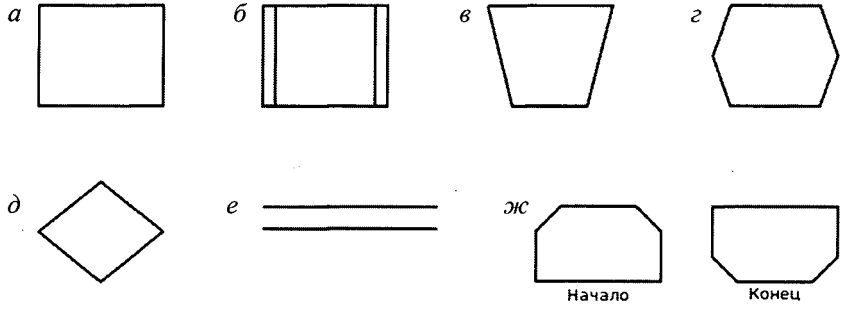

Рис. 2. Символы процесса: *а* – процесс, *б* – предопределенный процесс, *в* – ручная операция, *г* – подготовка, *д* – решение, *е* – параллельные действия, *ж* – границы цикла

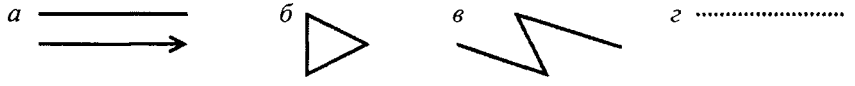

Рис. 3. Символы линий: *а* – линия, *б* – передача управления, *в* – канал связи, *г* – пунктирная линия

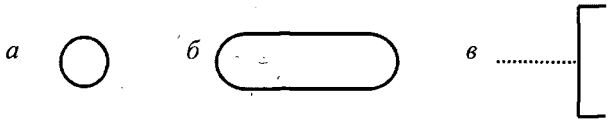

Рис. 4. Специальные символы: *а* – соединитель, *б* – терминатор, *в* – комментарий

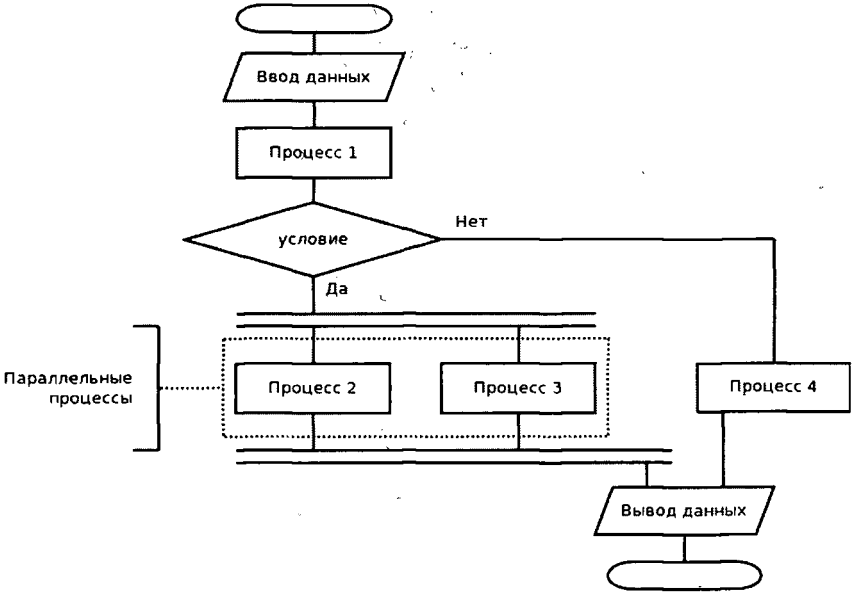

Рис. 5. Схема программы

В зарубежной литературе широко распространен термин Activity Diagram (AD), который обычно переводится на русский язык как «диаграмма активностей» или «диаграмма деятельности». AD определяется как графическое представление рабочего потока (workflow). AD содержат символы для представления отдельных действий, ветвлений «И» и «ИЛИ», потока управления. По существу, речь идет о тех же схемах алгоритмов. В русском языке также существует понятие «блок-схема». Это понятие довольно часто употребляется, например в системе школьного образования. Однако использование такого термина часто не приветствуется, так как в государственных стандартах закреплено понятие «схема алгоритмов и программ».

Рассмотрение схемы алгоритма как графа с целью применения аппарата теории графов к исследованию модели программы в свое время привели к появлению терминов «граф-схемы алгоритмов» (ГСА), «параллельные граф-схемы алгоритмов» (ПарГСА, ПГСА). Такой подход к пониманию схем алгоритмов дает широкие возможности для автоматизированного анализа программ, включая оценку их сложности, тестирование, верификацию. В частности, наличие граф-схемы параллельного алгоритма обеспечивает возможность его верификации на логическом уровне [51].

1.2. Диаграммы состояний-переходов

Использование диаграмм состояний-переходов (State Transition Diagram, STD) составляет сущность одного из популярных методов математического моделирования, используемых при проектировании программных систем. STD позволяет описать динамику системы с конечным числом состояний, то есть характеризует поведение системы во времени под управлением извне в форме управляющих воздействий. STD является графической формой конечного автомата, который, в свою очередь, используется для моделирования поведения технических объектов или объектов реального мира [50]. Диаграммы состояний-переходов нашли важное применение в формальной верификации, то есть в строгом доказательстве правильности программ, а именно в методе проверки на модели Model Checking [53].

В классическом варианте STD представляет собой ориентированный граф, вершины которого соответствуют состояниям, а дуги – переходам. Около дуг описываются условия переходов, а также внешнее проявление переходов (действия [52]). Для изображения STD применяются различные системы обозначений.

STD-диаграмма Харела, являющаяся частью языка UML, предоставляет больший набор выразительных средств, в частности, в ней возможна декомпозиция состояний, рассмотрение активности внутри состояний и др.

Что касается систем с параллелизмом, то STD не может их описать по определению. Формализм, который позволяет это сделать – сеть Петри. В ней допускается переход из нескольких состояний в несколько состояний, то есть система одновременно может находиться в нескольких состояниях.

На рисунке 6 приведен пример STD, упрощенно описывающей некоторые состояния таксиста во время рабочего дня. Пусть сначала таксист находится в состоянии «Ожидание заявки». Если таксист принимает некоторый заказ (срабатывает условие перехода «Принят заказ»), то происходит переход в состояние «Поездка к месту назначения». Выполнение условия «Пассажир сел в машину» приводит к переходу в состояние «Поездка к месту назначения». После завершения поездки пассажир оплачивает поездку (состояние «Расчет») и покидает машину. Таксист снова переходит в состояние «Ожидание заявки».

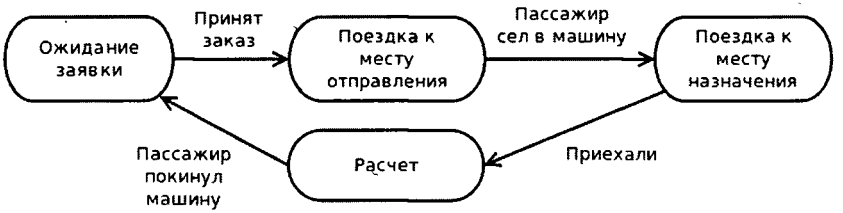

Рис. 6. Пример STD

На рисунке 7 приведена STD в другой нотации. Данный пример описывает состояния программы, реализующей некоторую вычислительную процедуру и активно не взаимодействующую с пользователем [52]. Здесь в отличие от предыдущего случая используются терминальные состояния, то есть исходное состояние и состояние завершения. Программа переходит из исходного состояния в состояние «Ввод» вне зависимости от каких-либо условий, соответствующее действие называется «Инициализация». Затем проводятся некоторые вычисления, выводится результат, и программа завершает свою работу.

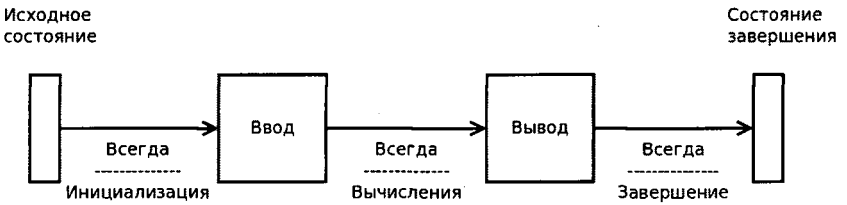

Рис. 7. Пример STD

1.3. Методология функционального моделирования IDEF0

Методология IDEF0 основана на технологии структурного анализа и проектирования SADT (Structured Analysis and Design Technique), разработанной Дугласом Т. Россом в 1969–1973 годах. Стандарт IDEF0 разработан в 1981 году департаментом военно-воздушных сил США в рамках программы автоматизации промышленных предприятий (Integrated Computer Aided Manufacturing, ICAM). Аббревиатура IDEF расшифровывается как ICAM DEFinition и представляет собой набор методологий: IDEF0, IDEF1, IDEF1x, IDEF2, IDEF3 и т.д. В 2001 году в России были приняты рекомендации по стандартизации [61], в которых методология IDEF0 названа методологией функционального моделирования.

Основу IDEF0 составляет графический язык моделирования систем. Диаграмма в IDEF0 состоит из именованных блоков, соединенных стрелками (рис. 8).

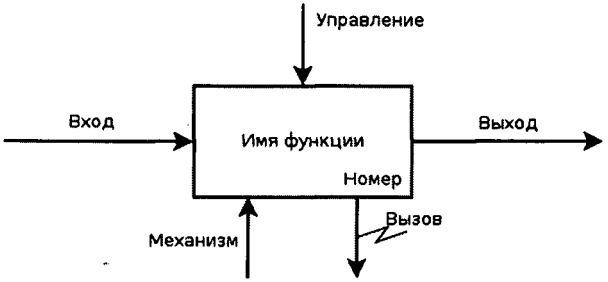

Рис. 8. Блок IDEF0-диаграммы в общем виде

Блоки соответствуют функциям[1]. Стрелки трактуются в зависимости от того, с какой стороны они присоединены к блоку. Стрелки слева соответствуют входам (то, что преобразуется или расходуется функцией), справа – выходам (продукт выполнения функции), сверху – управлениям (условия, необходимые функции для формирования правильного выхода), снизу – механизмам (то, с помощью чего выполняется функция)[2].

[1] Термин «функция» принят в Р 50.1.028–2001 [61]. На самом деле блок может соответствовать целому комплексу процессов, отдельному процессу, действию и т.д. В англоязычной литературе все это обозначается общим словом activity (активность).

[2] Стрелки, присоединенные к блоку снизу и направленные вниз, используются для того, чтобы обозначить обращение из данной модели

IDEF0-модель содержит не одну диаграмму, а иерархию диаграмм, возникающую в результате декомпозиции функций. Каждый блок имеет номер, определяющий его место в иерархии. Единственный блок диаграммы самого верхнего уровня (контекстная диаграмма) имеет номер А0 и детализируется на диаграмме следующего уровня блоками А1, А2, А3, ... Блок А1 детализируется блоками А11, А12, А13, ... ; А2 – блоками А21, А22, А23, ... и т.д. На одной диаграмме не должно быть очень мало (кроме контекстной диаграммы) и слишком много блоков – требуется соблюдение правила «3–7 ± 2» в соответствии с законом Миллера [12]. Как следствие, элементы номера представляют собой однозначные числа, и разделители между ними не нужны.

Кроме диаграмм в состав IDEF0-модели входят текст, используемый для объяснений и уточнений характеристик, потоков, соединений и т.п., и глоссарий, в котором определяются аббревиатуры, ключевые слова и фразы, используемые в диаграммах.

Для принципиального понимания методологии IDEF0 достаточно весьма компактного ее описания, что часто и применяется в соответствующих учебных материалах. Однако на практических занятиях, при решении задач, более или менее приближенных к реальности, возникают трудности вследствие причин самых разных уровней: от незнания нюансов методологии до непонимания самой организации процесса функционального моделирования. Поэтому перед выполнением конкретных заданий полезно внимательно ознакомиться с рекомендациями по стандартизации [61]. В данном документе содержатся подробные сведения о методологии IDEF0, графическом языке описания моделей, а также практические указания по методике разработки таких моделей. Рассматриваются синтаксис и семантика графического языка IDEF0; свойства диаграмм, связанные, например, с параллелизмом в предметной области, ветвлением и слиянием потоков, отношениями между блоками диаграмм, отношениями между диаграммами и окружающей средой; правила построения диаграмм и нумерации блоков; методические приемы, облегчающие практическое применение методологии; способы применения типовых моделей и отдельных диаграмм; организация процесса функционального моделирования и управления проектом и др. В приложениях представлены диаграммы

или из данной части модели к блоку, входящему в состав другой модели или другой части модели [там же]. Иногда стрелкам снизу приписывают ресурсы [52]. Это позволяет, в частности, применять методологию IDEF0 в управлении проектами.

типовой функциональной модели промышленного предприятия в форме IDEF0. Документ хорошо структурирован, снабжен большим числом показательных примеров.

На рисунке 9 показана IDEF0-диаграмма, отображающая на общем уровне состав процессов в типовой информационной системе, предназначенной для ввода, хранения, обработки и вывода некоторых данных (например, данных, получаемых в результате измерений). Имеется управляющий процесс, а также следующие процессы, точнее целые комплексы процессов: ввод, хранение, обработка, вывод. Можно считать, что названные группы процессов выполняются отдельными подсистемами: подсистема управления, подсистема ввода и т.д. Инструкции пользователя преобразуются в управляющие воздействия на подсистемы. Входные данные представляются в некотором внутреннем формате (подсистема ввода осуществляет соответствующее преобразование). Данные для обработки извлекаются из хранилища, результаты обработки могут выводиться, а могут помещаться обратно в хранилище.

1.4. Диаграммы потоков данных

Применение диаграмм потоков данных (Data Flow Diagram, DFD) также относится к структурному анализу и имеет место при углубленном анализе функциональных требований к проектируемой системе.

Модели потоков данных были независимо предложены сначала в 1975 году Э. Йорданом, а в 1979 году – Ч. Гейном и Т. Сарсоном.

DFD-модель представляет собой набор диаграмм, представляющих систему на разных уровнях декомпозиции. Вершины DFD соответствуют внешним сущностям, функциям и хранилищам данных; вершины соединяются стрелками, которые моделируют потоки данных. Поток данных моделирует передачу данных из одной части системы в другую. Функция продуцирует выходные потоки из входных в соответствии с действием, задаваемым именем функции. Хранилище определяет данные, сохраняемые между процессами. Внешняя сущность определяется как источник или приемник данных, находящийся вне системы. Хранилища данных не преобразуют данные, а преобразования потоков данных во внешних сущностях игнорируются. Все компоненты диаграммы имеют имена, к которым предъявляются определенные требования. Компоненты DFD в нотациях Йордана – Демарко и Гейна –

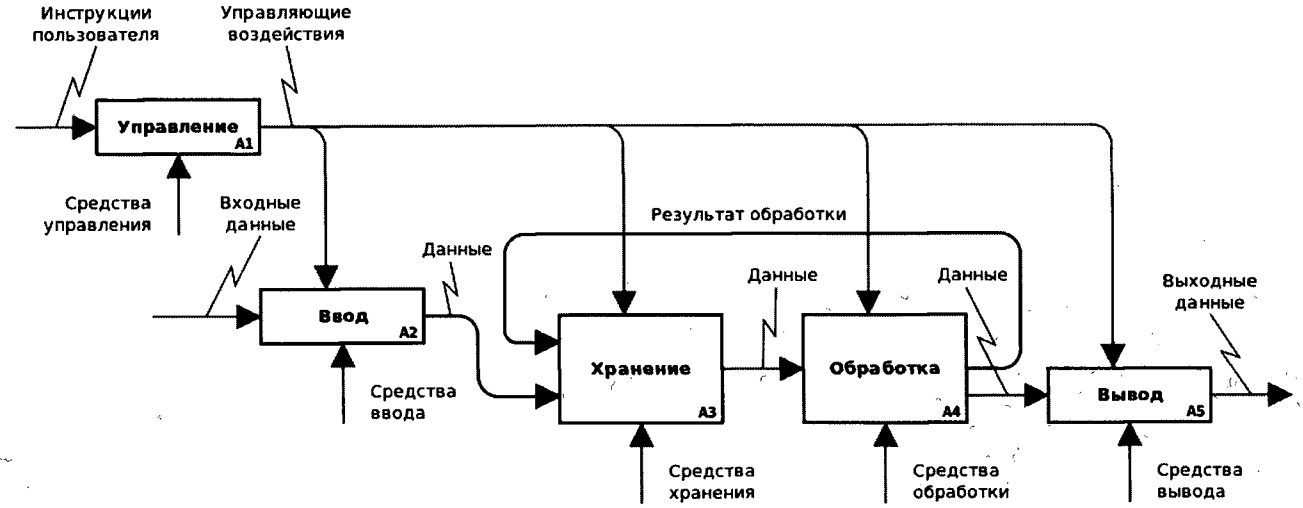

Рис. 9. Пример диаграммы IDEF0

Сарсона показаны на рисунке 10. Из рисунка видно, что диаграмму в нотации Йордана – Демарко удобней рисовать вручную.

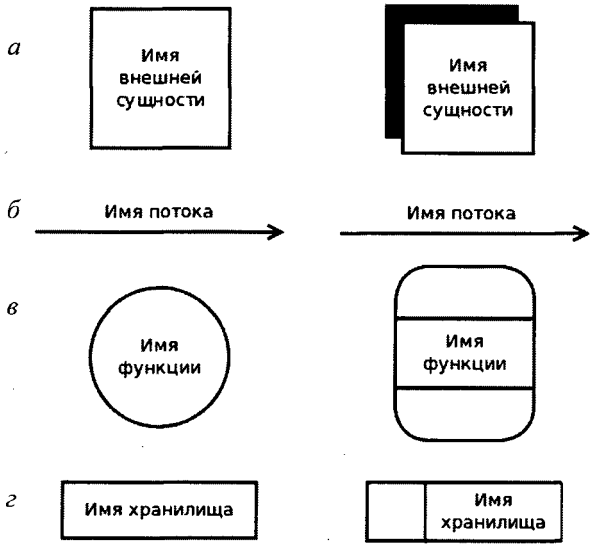

Рис. 10. Компоненты диаграммы потоков данных в нотациях Йордана – Демарко (слева) и Гейна – Сарсона (справа): *а* – внешняя сущность, *б* – поток данных или управляющий поток, *в* – функция, *г* – хранилище данных

Так же как и в случае с IDEF0 применяется декомпозиция функций. Нумерация функций производится с той же целью и по тем же принципам, что в IDEF0, то есть применяется иерархическая нумерация. Дуги могут разветвляться или сливаться, в этом выражается декомпозиция потоков. Процесс моделирования, как и в случае с IDEF0, начинается с построения контекстной диаграммы. Единственный процесс контекстной диаграммы подвергается декомпозиции на диаграммах следующих уровней. Действует правило «3–7 ± 2». Декомпозиция прекращается при выполнении критериев остановки процесса декомпозиции: функция не делится более чем на две части; функция описывается миниспецификацией.

Иногда на диаграмме изображаются также управляющие потоки, для этого используются пунктирные линии. Пунктиром могут изображаться и функции, если необходимо подчеркнуть, что их назначением является управление. Применение управляющих

потоков в DFD подробно обсуждается в [58], там же есть содержательные примеры таких диаграмм.

В качестве примера рассмотрим диаграммы потоков данных, полученные в результате довольно беглого анализа процессов, связанных с вызовом городского такси по телефону. На контекстной диаграмме (рис. 11) показаны сущности внешней среды, имеющие дело с этими процессами (внешние сущности), и связанные с ними потоки данных. В примере внешние сущности – клиент и водитель. Они связаны друг с другом комплексом процессов «Обслуживание клиентов». Клиент сообщает системе адреса пунктов отправления и назначения (маршрут). Системе, кроме того, становится доступен номер его телефона. Клиент в зависимости от ситуации может получить приветствие, отказ, стоимость, номер автомобиля, которого следует дожидаться. Таксисты получают по рации данные о заявках (маршрут). Водители сообщают системе (диспетчеру) номер машины и примерное время ожидания клиента (в случае приема заказа).

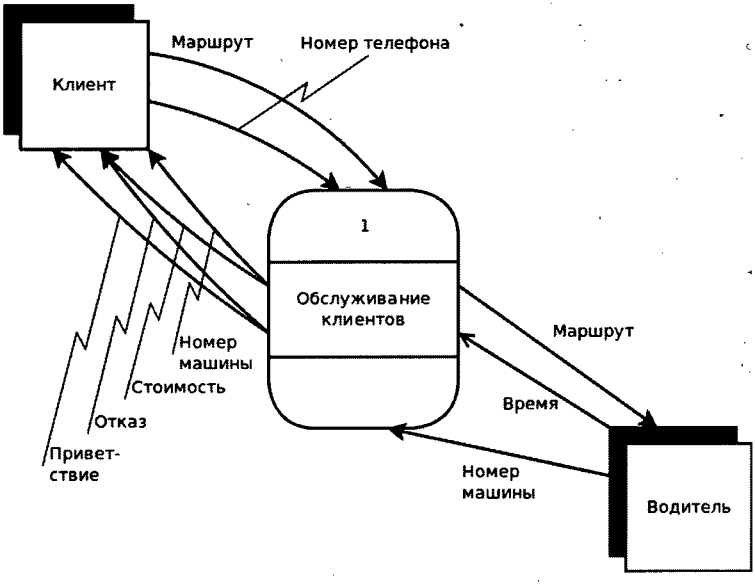

Рис. 11. Пример контекстной DFD

Диаграмма следующего уровня (рис. 12) является детализацией комплекса процессов «Обслуживание клиентов». Здесь нет внешних сущностей, зато появляются хранилища и новые потоки

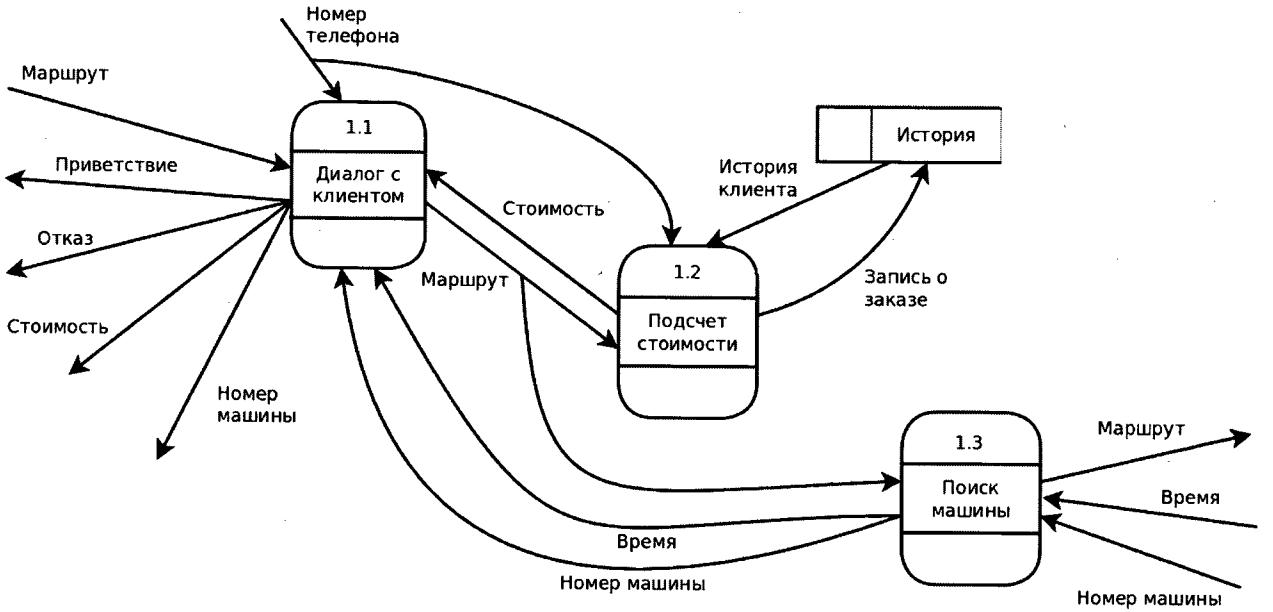

Рис. 12. Пример детализирующей DFD

данных. Потоки согласованы с контекстной диаграммой. Хранилище данных используется для хранения истории клиентов, с тем чтобы, например, обеспечить предоставление скидок постоянным клиентам или наоборот занести некоторых клиентов в «черный список».

1.5. Event-driven Process Chains (EPC)

Event-driven Process Chains (EPC) можно перевести как «цепочки процессов, управляемые событиями». Диаграммы EPC предназначены для представления рабочих потоков бизнес-процессов. Они впервые появились в системе планирования ресурсов организации (ERP-системе) SAP R/3 [27]. Сейчас EPC широко распространены, например, используются в ARIS Toolset [3; 4].

На рисунке 13 показаны основные компоненты EPC-диаграмм. Нетрудно заметить, что многие символы взяты из рассмотренных выше схем алгоритмов и программ.

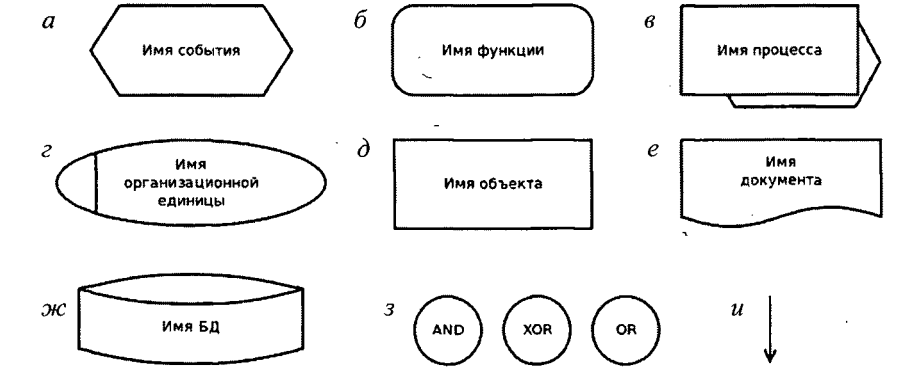

Рис. 13. Основные компоненты EPC-диаграмм: *а* – событие, *б* – функция, *в* – путеводитель по процессам, *г* – организационная единица, *д* – информационный, материальный или ресурсный объект, *е* – документ, *ж* – база данных, *з* – логические коннекторы, *и* – потоки

События – пассивные элементы в EPC. Функция происходит после выполнения некоторого события и завершается некоторым событием. EPC-диаграмма всегда должна начинаться событием

и всегда заканчивается событием. Функция – активный элемент. Функции моделируют задачи, решаемые внутри организации. Функция описывает преобразование входного состояния в выходной. Для того чтобы из одной диаграммы сделать ссылку на другую, используются так называемые иерархические функции. Элемент «Имя организационной единицы» позволяет указать лицо или структурное подразделение, ответственное за выполнение функции. Информационные, материальные и ресурсные объекты моделируют соответствующие объекты реального мира. Для описания логики процессов используются логические соединители (logical connectors). Поддерживаются три вида логических отношений: «И», «ИЛИ», исключающее «ИЛИ». Управляющий поток соединяет события с функциями, события с логическими соединителями и т.д. Информационные потоки соединяют функции с входными и выходными данными функций. Связь с организационной единицей (organization unit assignment) соединяет функцию и имя организационной единицы. Процессный путь соединяет две диаграммы, то есть два бизнес-процесса.

Абстрактный пример EPC-диаграммы приведен на рисунке 14. Бизнес-процесс начинается с некоторого события, обозначенного как «начальное событие». Выполняется «Функция 1», на входе которой – «входящий документ», на выходе – «исходящий документ». «Функцию 1» выполняет «исполнитель 1». Имеется также некий информационный объект, названный в примере «Информация». Это хранилище, куда помещаются данные о выполненной операции; «xor» означает ветвление «ИЛИ» (исключающее). После этого символа – два события. Им соответствуют некоторые условия. Эти условия должны быть ортогональны, альтернативны. Если произойдет «Событие 1», то будет выполняться «Функция 2», если произойдет «Событие 2», то будет выполняться «Функция 3». Перед слиянием параллельных процессов тоже должны произойти некоторые события: «Событие 3» и «Событие 4».

1.6. Business Process Modeling Notation (BPMN)

Нотация для описания бизнес-процессов BPMN (Business Process Modeling Notation) разработана Business Process Management Initiative (BPMI) и с 2005 года поддерживается Object Management Group (OMG) [23]. Основная идея, поддерживаемая в

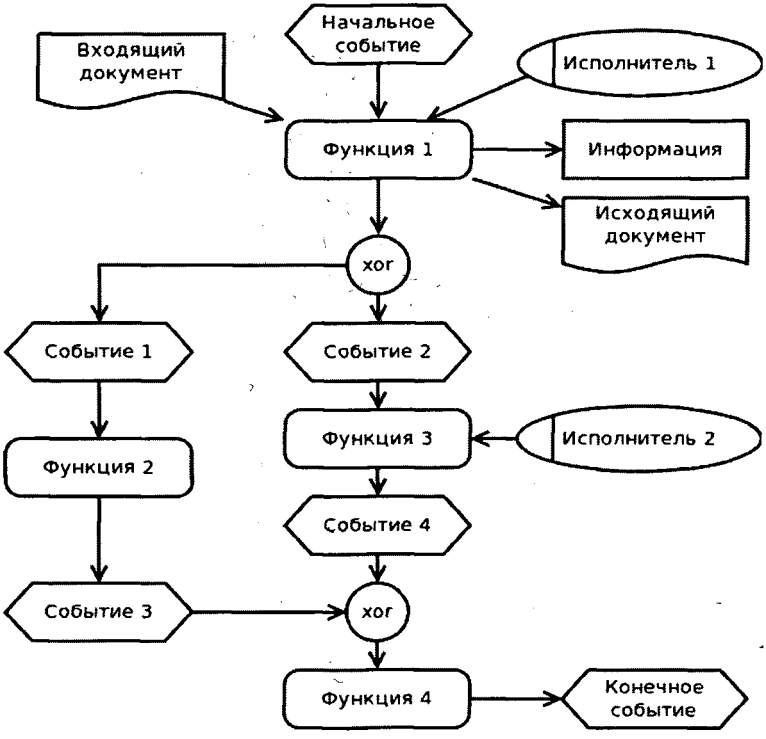

Рис. 14. Пример EPC-диаграммы

этой нотации, – понятность как для технических специалистов, так и для участников бизнес-процессов. Спецификация BPMN определяет не только графический язык описания бизнес-процессов, но и то, как диаграммы могут быть трансформированы в исполняемые модели на языке BPEL [5]. BPEL (Business Process Execution Language) – язык на основе XML, предназначенный для формального описания бизнес-процессов и протоколов их взаимодействия между собой.

Учебного материала по BPMN на русском языке очень мало, однако спецификация BPMN читается легко и содержит исчерпывающую информацию [5]. Кроме того, для изучения BPMN можно рекомендовать англоязычные слайды по BPMN от IBM [29], доступную в Интернете. Слайды содержат введение в нотацию и два упражнения на построение диаграмм.

Диаграммы, используемые в рамках BPMN (Business Process Diagram, BPD), строятся из элементов следующих категорий: объекты потока управления (события, действия и логические операторы); соединяющие объекты (поток управления, поток сообщений и ассоциации); роли (пулы и дорожки); артефакты (данные, группы и текстовые аннотации). Дорожки и пулы дорожек применяются для указания распределения ролей. На дорожку (swimlane) помещаются только те действия, за которые ответствен определенный исполнитель. Несколько дорожек можно объединить в пул. Свернутые пулы используются для компактности и отвлечения от деталей. На рисунке 15 приведены базовые элементы (core elements), используемые при построении BPD, в соответствии со спецификацией BPMN.

На рисунке 16 показаны некоторые элементы расширенного множества BPD: другие события (отмена, компенсация, условие, сигнал, составное событие, ссылка, останов); События делятся также на события обработки (изображение в кружке белое) и генерации (изображение в кружке закрашено). Кроме того, для начального события окружность изображается одинарной тонкой линией, промежуточное – двойной, а завершающее – одинарной жирной линией. Действия в расширенном множестве также возникают в результате комбинирования обозначений. Так появляются циклические, многократные действия, подпроцессы и др.

На рисунке 17 представлен пример BPMN-диаграммы, описывающей процесс взаимодействия клиента, вызывающего такси, и такси. Пример упрощен в результате стремления к наглядности. Видно, что BPD представляет собой несколько схем алгоритмов, расположенных рядом и соответствующих параллельно выполняющимся процессам.

Клиент представлен свернутым пулом. Пул «Такси» содержит дорожки «Диспетчер» и «Водители». Начальное сообщение соответствует звонку клиента. Клиент при этом сообщает диспетчеру адрес пункта отправления и адрес пункта назначения. В службе такси в это время имеет место процесс «Получить данные». Затем следует группа процессов «Поиск машины», в которой участвуют и диспетчер, и водители. Имеет место цикл: диспетчер предлагает заказ водителям (по рации), передавая при этом полученные от клиента адреса в эфир. Водители принимают решение, взяться ли за выполнение этого заказа. Если в течение пять секунд ответа от водителей не последовало, то диспетчер повторяет адреса. К про-

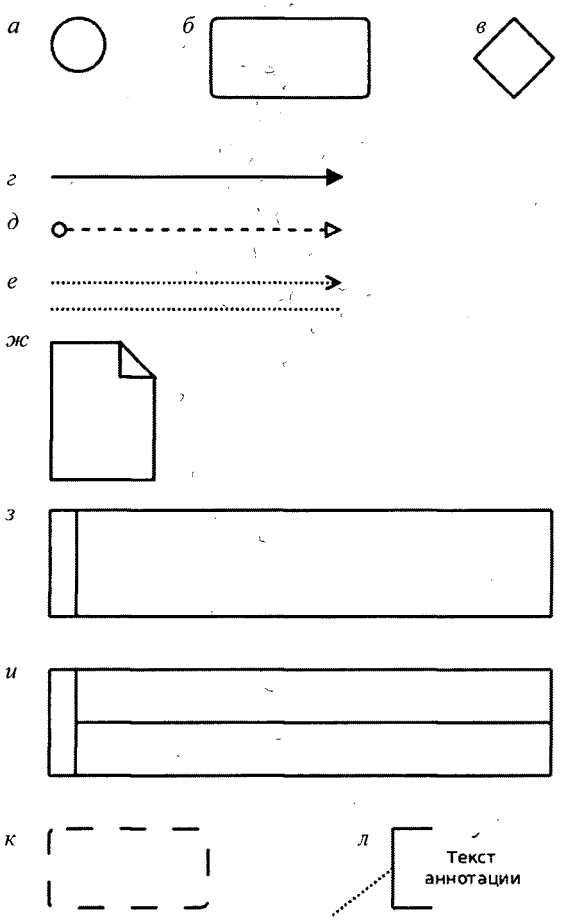

Рис. 15. Базовые элементы BPD: *а* – событие, *б* – активность, *в* – шлюз, *г* – поток управления, *д* – поток сообщений, *е* – ассоциации, *ж* – объект данных, *з* – пул, *и* – дорожки, *к* – группа, *л* – текстовая аннотация

цессу «Думать» прикреплен символ сообщения. Движение от этого символа происходит, если процесс завершился не успешно. Такая трактовка прикрепленного символа сообщения предусмотрена спецификацией BPMN. Если после нескольких итераций ни один водитель так и не взялся за выполнение заказа, то выполняется процесс «Отказать», и бизнес-процесс завершается. При этом кли-

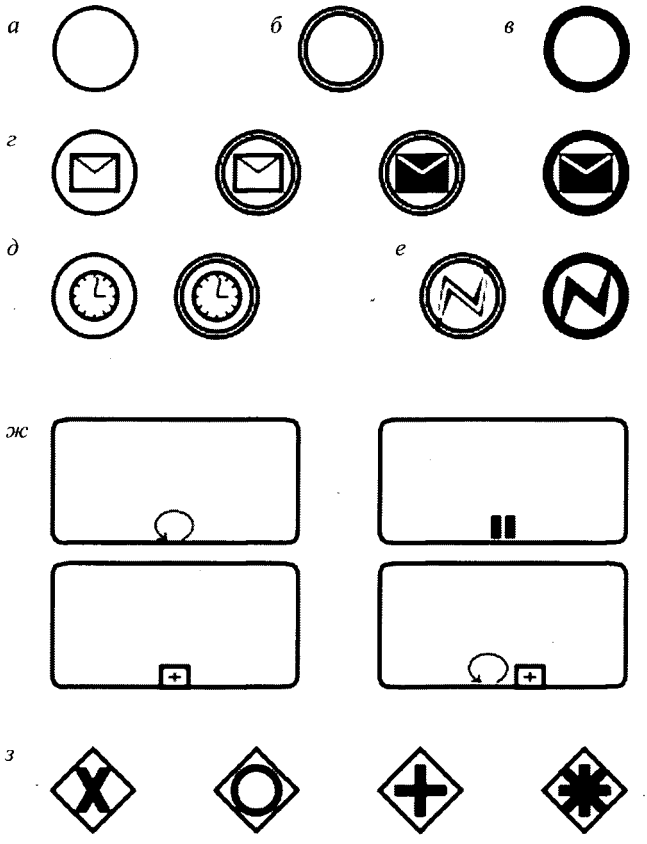

Рис. 16. Некоторые объекты потока управления из расширенного множества элементов BPD: *а* – начальное событие, *б* – промежуточное событие, *в* – завершающее событие, *г* – события-сообщения, *д* – события-таймеры, *е* – события-ошибки, *ж* – действия (циклическое, многократное, свернутый подпроцесс, свернутый циклический подпроцесс), *з* – логические операторы (исключающее ИЛИ, включающее ИЛИ, И, сложный оператор)

ент получает примерно такое сообщение: «Извините, свободных машин нет». Если некоторый водитель принял заказ, то выполняются действия «Оформить заказ», «Выполнить заказ», «Взять деньги», и бизнес-процесс также завершается.

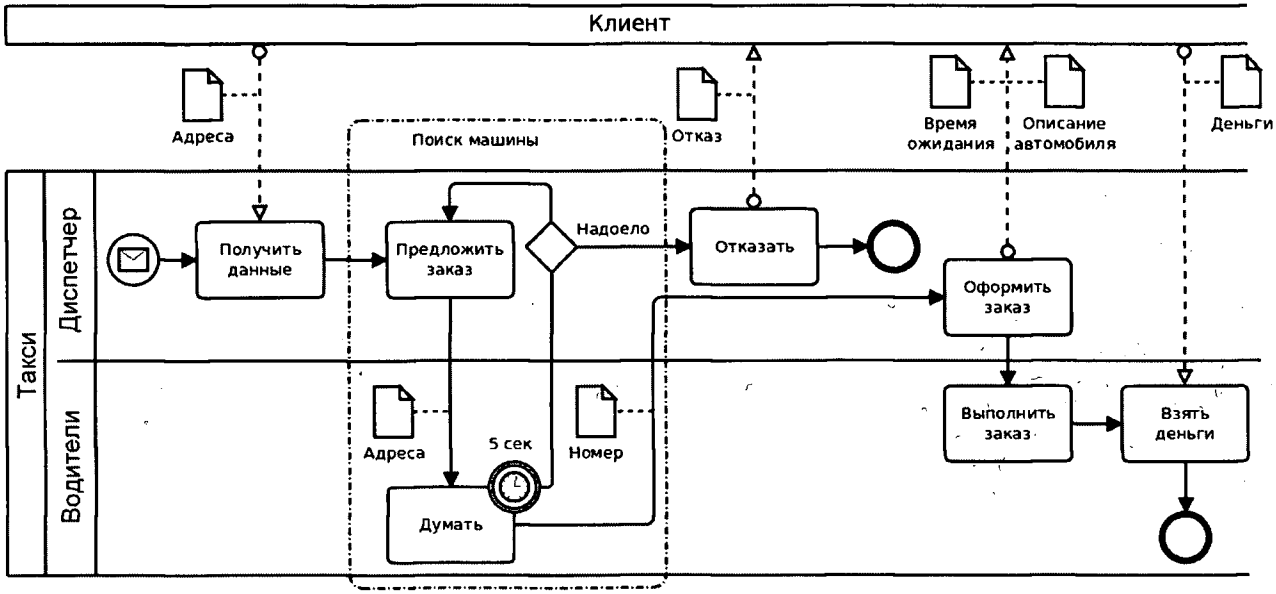

Рис. 17. Пример BPD

1.7. Иерархические модели данных

Структурой данных называется совокупность правил и ограничений, отражающих связи между отдельными компонентами данных [52]. Приведем некоторые классификации структур данных в соответствии с этим определением. Различают абстрактные и конкретные структуры данных. Абстрактные структуры, в свою очередь, делятся на структуры несвязанные (множество, кортеж) с неявными связями (таблицы) и структуры с явными связями (всевозможные графы). Конкретные структуры относятся к представлению данных в компьютерной системе. В качестве примеров конкретных структур можно привести массив, строку, линейный список и т.д. Можно сказать, что применение абстрактных структур относится скорее к концептуальному моделированию, а применение конкретных находится ближе к физическому.

Другая классификация делит структуры данных на иерархические, которые описывают отношение вхождения, и сетевые, основанные на графах и позволяющие описывать связность элементов данных независимо от вида отношения. Остановимся пока на способах графического моделирования иерархических структур данных.

При построении иерархической модели данных опираются на утверждение о том, что любая структура данных может быть представлена при помощи всего трех конструкций: последовательности, ветвления и цикла. Аналогичный тезис известен для последовательных алгоритмов. Иерархические модели данных представляют из себя деревья. Дерево возникает в результате детализации структуры данных. Терминальные вершины соответствуют элементам данных.

Примерами графических нотаций для представления иерархических моделей данных являются диаграммы Джексона (1975) и скобочные диаграммы Орра (1974). На рисунке 18 представлены основные конструкции, используемые в диаграммах Джексона. На верхнем уровне каждой такой конструкции располагается так называемый блок конструкции, нижний уровень возникает в результате детализации этого блока. Представленные на рисунке 18 фрагменты следует читать так: «Компонент А состоит из компонентов a1, a2 и a3» (например: «Курсовая работа состоит из введения, основной части и заключения»), «В – это или b1, или

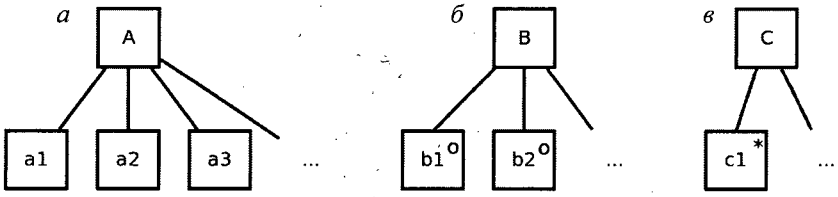

Рис. 18. Базовые конструкции на языке диаграмм Джексона:
а – последовательность, *б* – ветвление, в цикл

b2» (например: «Отметка о сдаче зачета – это "зачтено" или "не зачтено"»), «С представляет собой повторяющиеся компоненты c1» (например: «Слово – это несколько букв»). На месте многоточия может располагаться еще что-то.

На рисунке 19 приведен фрагмент диаграммы Джексона, описывающей структуру курсовой работы студента. Курсовая работа состоит из титульного листа, оглавления, основной части, заключения, списка литературы и др. Основная часть включает в себя несколько глав. В списке литературы содержатся несколько источников, которые пронумерованы и среди которых могут встречаться ссылки на страницы в Интернете, книги, статьи и др.

Скобочные диаграммы Орра основаны на тех же принципах[3], отличаются только обозначения. На рисунке 20 приведен пример диаграммы Орра для той же структуры данных. Диаграмма Орра обычно получается более компактной, чем диаграмма Джексона.

Полезность рассмотренных моделей особенно проявляется при использовании их совместно моделями, разрабатываемыми в других аспектах, например DFD. Так, описать потоки данных и хранилища, указываемые на DFD, можно при помощи рассмотренных нотаций и тем самым получить исходный материал для организации ряда последующих этапов разработки. Речь идет, в частности, об оценке сложности системы, например, с использованием функционально-ориентированных метрик [50; 58], формировании тестов с применением функционального и структурного подходов [58] и др.

[3] На самом деле диаграммы Джексона и Орра являются разными вариантами графического представления одной и той же модели данных – модели Джексона – Орра.

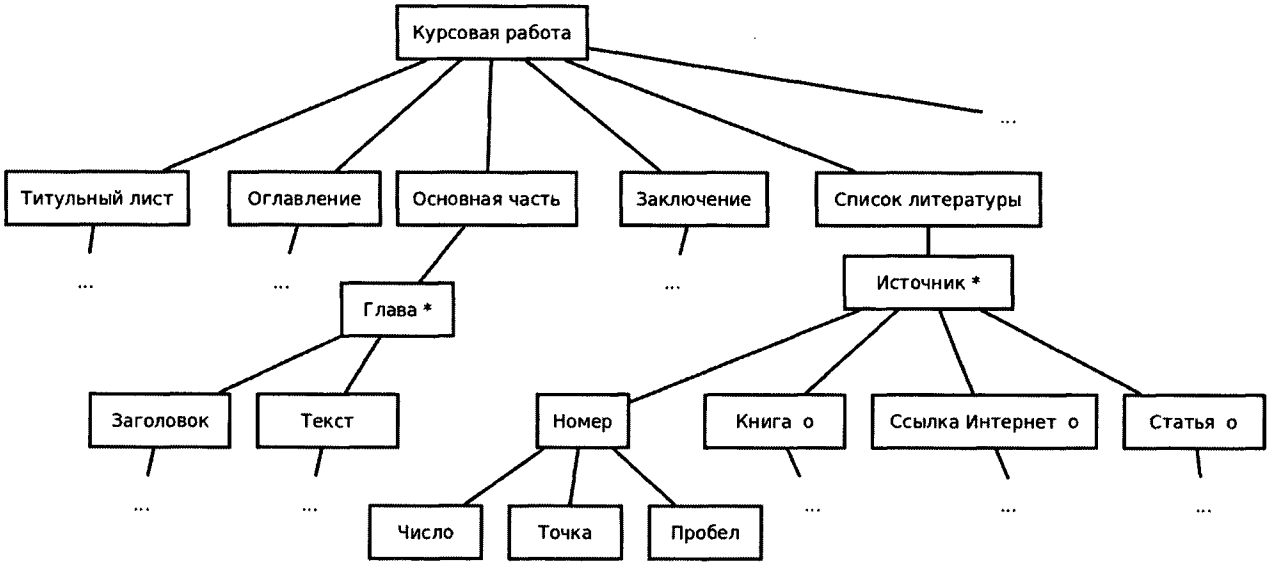

Рис. 19. Пример использования нотации Джексона

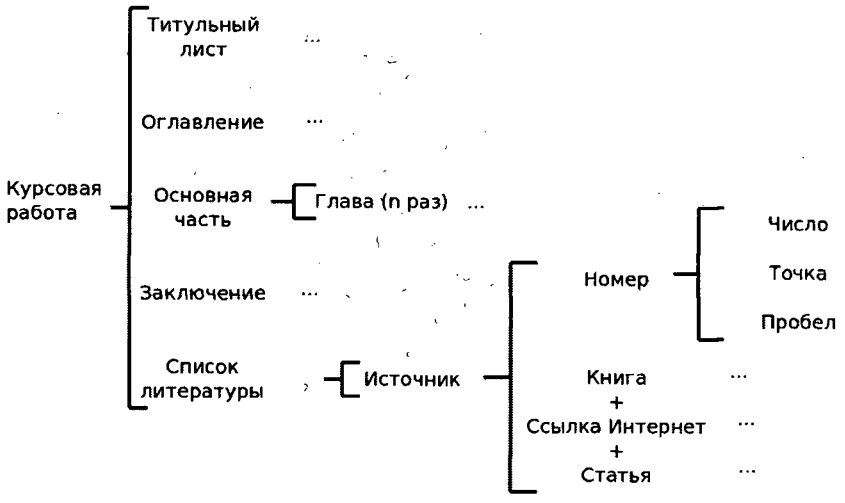

Рис. 20. Пример использования нотации Орра

1.8. Диаграммы «Сущность – связь»

Диаграмма «Сущность – связь» (Entity-Relationship Diagram, ERD) является концептуальной моделью данных. Концептуальная модель свободна от инструментов реализации компонентов информационной системы. ERD может быть преобразована в другую модель данных, например реляционную [20]. В этом смысле ERD называют языком метамоделирования.

Применение ERD при разработке программной системы относится к объектному подходу к углубленному анализу требований [50].

Основными понятиями ERD являются сущность, атрибут и связь. Поясним значения этих терминов. Сущность – объект, имеющий значение для предметной области. Каждая сущность имеет уникальное имя и обладает атрибутами, которые принадлежат сущности либо наследуются через связь. Сущность представляет собой множество экземпляров объектов. Имя сущности отражает класс объекта. Атрибут – любая характеристика сущности, значимая для предметной области. Экземпляр атрибута – определенная характеристика конкретного экземпляра сущности. Связь – именованная ассоциация между сущностями, значимая для рассматриваемой предметной области [52].

Из этих определений следует, что сущности, атрибуты и связи выявляются проектировщиком при помощи абстрагирования, отвлечения от несущественных деталей.

Существуют ключевые и описательные атрибуты. Первичный ключ – атрибут или совокупность атрибутов и/или связей, предназначенные для идентификации экземпляров сущностей. Внешний ключ – атрибут (или группа атрибутов), который является первичным ключом в другой сущности или наследуется через связь. Описательные атрибуты могут быть обязательными или опциональными.

Если сущность ассоциирована некоторым образом сама с собой, то говорят о рекурсивной связи. Например, сотрудник организации может быть напарником другого сотрудника. Независимая сущность может быть связана или не связана с другими сущностями. Например, некоторый товар может не присутствовать ни в одном заказе. Зависимая сущность (слабая сущность) представляет данные, зависящие от других сущностей, она обязательно должна быть связана с другими сущностями. Например, заказ всегда связан с каким-нибудь товаром.

Тип (мощность, кардинальность) связей определяет количественный характер участия сущностей в связи. Возможны следующие типы: «один-к-одному», «один-ко-многим», «многие-ко-многим».

Если некоторым образом ассоциированы две слабые сущности (например, сотрудник организации и структурное подразделение в предположении, что каждый сотрудник должен быть приписан к некоторому подразделению, а в каждом подразделении обязательно есть сотрудники), то будем для краткости говорить, что связь является «обязательной с обеих сторон». Если в слабой связи является только одна сущность, что будем говорить, что связь является «обязательной только с одной стороны» (пример – студент и преподаватель, когда студент пишет курсовую работу, а преподаватель является научным руководителем). Подобным образом можно ввести понятие связи, «необязательной с обеих сторон». Существует ряд недопустимых связей, например, рекурсивная связь, обязательная только с одной стороны. Наличие такой связи предполагало бы, что сущность одновременно является зависимой и независимой. Такая ситуация является противоречивой.

Существует множество нотаций для преставления ER-модели: Питера Пин-Шен Чена (первая нотация, 1976 г.) (рис. 21), Джеймса

Мартина, IDEF1X, Ричарда Баркера (рис. 22) и др. Нотация Баркера используется в CASE-инструментах от Oracle. Последние три нотации похожи между собой. Различные программные средства, реализующие одну и ту же нотацию, могут отличаться своими возможностями.

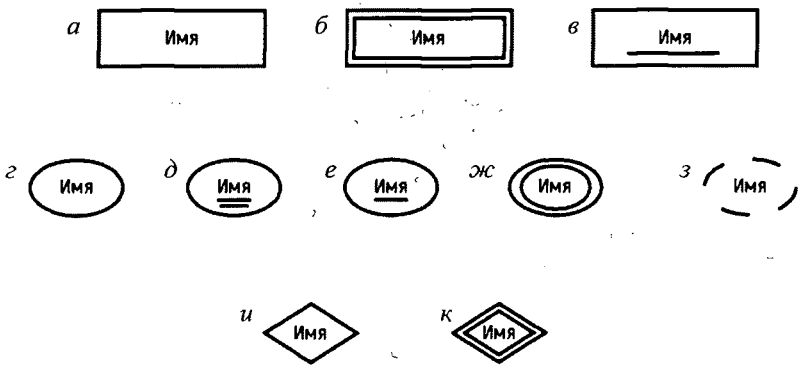

Рис. 21. Компоненты ERD в нотации Чена: *а* – независимая сущность, *б* – зависимая сущность, *в* – родительская сущность, *г* – атрибут, *д* – первичный ключ, *е* – внешний ключ, *ж* – многозначный атрибут, *з* – наследуемый атрибут, *и* – связь, *к* – идентифицирующая связь

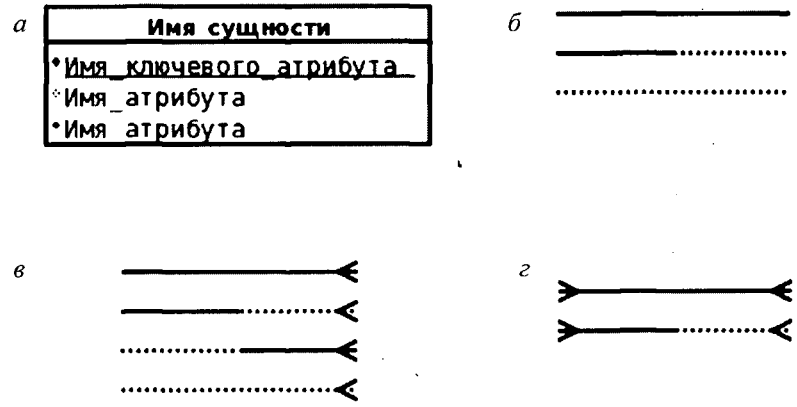

Рис. 22. Компоненты ERD в нотации Баркера: *а* – сущность, *б* – связь «один-к-одному», *в* – связь «один-ко-многим», *г* – связь «многие-ко-многим»

Кардинальность связей наглядно описывается при помощи нотации Crown's Foot[4] (рис. 23). Используется, например, в ER Мартина.

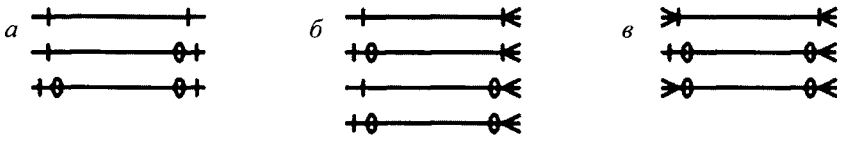

Рис. 23. Связи в нотации Crown's Foot: *а* – один-к-одному, *б* – один-ко-многим, *в* – многие-ко-многим

На рисунке 24 приведены самые простые примеры ER-диаграмм в нотациях Чена и Баркера. В примере сущности «Сотрудник» и «Подразделение» (имеется в виду структурное подразделение некоторой организации) сотрудник идентифицируется номером (первичный ключ) и работает в одном из структурных подразделений. Атрибуты сотрудника: «номер сотрудника», «имя», «фамилия» и др. Атрибуты структурного подразделения: «код подразделения» (первичный ключ), «название».

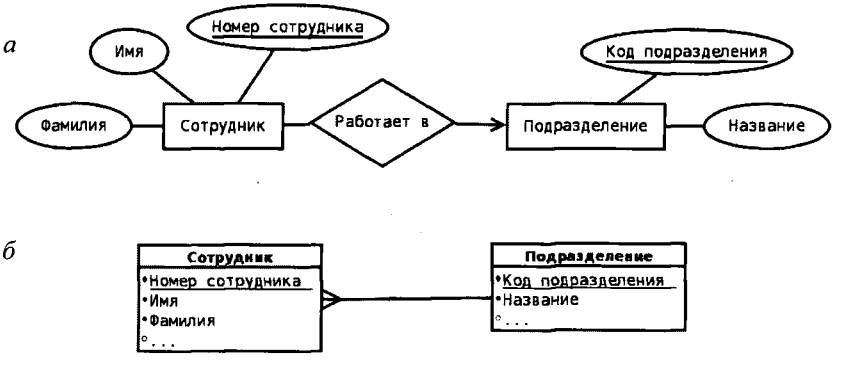

Рис. 24. Примеры ER-диаграмм: *а* – в нотации Чена, *б* – в нотации Баркера

ER-модель и реляционная модель данных – не одно и то же! Вследствие привычки к реляционным базам данных при построении ER-модели исходят из того, что сущности впоследствии будут преобразованы в таблицы реляционной СУБД, то есть, работая с

[4] Crown's Foot – воронья лапа (англ.).

сущностью, проектировщик представляет таблицу. Это вредно: происходит смешивание ER-модели и реляционной модели, концептуальной и физической моделей данных.

1.9. Объектно-ориентированное проектирование. Unified Modeling Language (UML)

Если сущности характеризуются не только атрибутами, но и поведением, представляемым в виде набора отдельных функций, то такие сущности называются классами, экземпляры классов – объектами, функций – методами классов. Соответствующий подход к анализу, проектированию и программированию носит название объектно-ориентированного (ООП). Это может означать и «объектно-ориентированное проектирование». Существует также понятие «объектно-ориентированный анализ». ООП, как и ER-моделирование относится к объектному подходу к углубленному анализу требований к программному продукту [50]. ER-моделирование можно считать прообразом ООП.

В CASE-системах, реализующих такой подход, используется ряд диаграмм, объединенных в унифицированном языке моделирования (Unified Modeling Language, UML).

Начальные версии UML разработаны Г. Бучем, Д. Рамбо и А. Якобсоном (Rational Software, Object Management Group) в 1994–1996 годах. Версии 0.9 и 0.91 были опубликованы в 1996 году. В то же время для работы над языком был создан консорциум UML Partners, в который вошли такие компании, как Digital Equipment Corporation, Hewlett-Packard, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, ObjecTime, Oracle Corporation, Platinum Technology, Pretch, Rational Software, Reich Technologies, Softeam, Taskon, Texas Instruments и Unisys. Опубликованные версии UML: 1.0 (январь 1997), 1.1 (ноябрь 1997), 1.3 (июнь 1999), 1.4 (сентябрь 2001), 1.5 (март 2003), 2.0 (август 2005). Последняя версия: UML 2.3 (май 2010). UML 1.4.2 зафиксирована в международном стандарте ISO/IEC 19501:2005.

ООП и UML – темы отдельного большого разговора. Существует много работ по данной тематике. В качестве одной из основных можно рекомендовать [48]. Для краткого введения в UML можно пройти бесплатный курс в интернет-университете INTUIT.ru [35].

Здесь ограничимся лишь перечислением и кратким описанием некоторых диаграмм UML, а также приведем пример.

Ниже перечислены основные диаграммы UML (на самом деле диаграмм в UML больше):

– диаграмма прецедентов (Use-Case Diagram) представляет поведенческие аспекты системы, связывает экторов с прецедентами. Под эктором (от англ. actor) понимается множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями, например человек, другая система, подсистема, класс. Прецедент – это описание событий, приводящих к наблюдаемому эктором результату;

– диаграмма классов показывает, какие выделены классы, как они устроены, и представляет ассоциации между классами;

– диаграммы объектов показывают множество объектов и отношений между ними в определенный момент времени;

– диаграмма последовательностей и диаграмма взаимодействия отображают взаимодействие объектов в динамике. Имеется в виду информационное взаимодействие на основе сообщений. Диаграммы альтернативны друг другу, выглядят по-разному. Во второй используется нумерация, для того чтобы показать порядок действий. В первой порядок действий задается расположением символов;

– диаграмма состояний показывает состояния системы, возможные переходы между состояниями, условия переходов;

– диаграмма развертывания представляет инфраструктуру, на которой будет развернута программная система;

– диаграмма активности раскрывает алгоритмы, по которым протекают процессы системы.

На рисунке 25 приведен пример диаграммы классов для предметной области, которую можно назвать «Измерения». От класса «Средство измерений» наследуются классы «Измерительный преобразователь», «Измерительная установка» и «Измерительный прибор». Далее от класса «Измерительный преобразователь» наследуются классы «Аналого-цифровой преобразователь» (АЦП) и «Датчик». Здесь используется общепринятая в метрологии классификация средств измерений. АЦП обладает методами «старт» и «стоп», что позволяет начать и остановить сеанс динамических измерений. Другие классы также обладают методами, но они опущены для компактности рисунка. В измерительной установке используются и измерительный прибор, и измерительный преобразователь, что также показано на диаграмме.

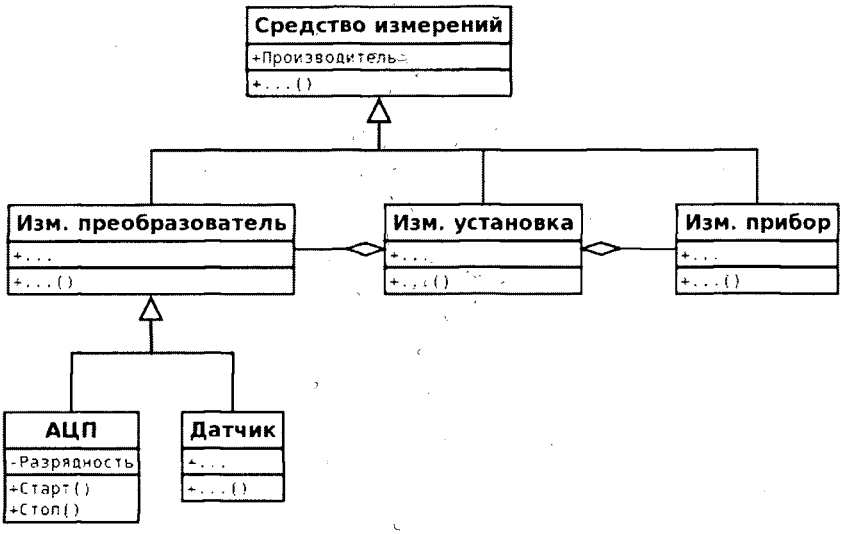

Рис. 25. UML. Диаграмма классов

Глава 2

ПРОГРАММНЫЕ ИНСТРУМЕНТЫ ПРОЕКТИРОВАНИЯ И УПРАВЛЕНИЯ ПРОЕКТАМИ

2.1. DBDesigner4

Назначение. Визуальное проектирование и обслуживание баз данных.

Разработчик. Майкл Дж. Зиннер (Michael G. Zinner) (Австрия).

Версии. Последние версии: 4.0.5.6 (Windows), 4.0.5.4 (Linux). В настоящее время проект остановлен.

Платформы. MS Windows, GNU/Linux.

Установка. Исходный код программы и бинарные файлы можно получить на сайте проекта [10]. В MS Windows права администратора для установки не требуются. Лицензия GPL.

Функциональные возможности:

– построение ER-диаграмм;

– администрирование баз данных (подключение к базе данных, управление ею);

– синхронизация модели и базы данных;

– реверс-инжиниринг (по базе данных строится графическая модель).

MySQL поддерживает различные базы данных: MySQL, Oracle, ODBC, SQLite, MSSQL.

Формат файлов. Модели сохраняются в .xml-файлах. Этим обеспечивается удобство разработки инструментов, дополняющих возможности DBDesigner, например инструментов анализа моделей данных.

Информация. Полезна любая информация о ER-моделировании. На англоязычном сайте проекта [10] имеется краткое описание системы, снимки экранных форм, FAQ, документация, обучающее видео, форум и др.

На рисунке 26 представлены главное окно системы и окно настройки свойств базы данных во время выбора типа базы данных (в данном случае SQLite). Программа запущена в GNU/Linux Ubuntu 10.4 при помощи wine.

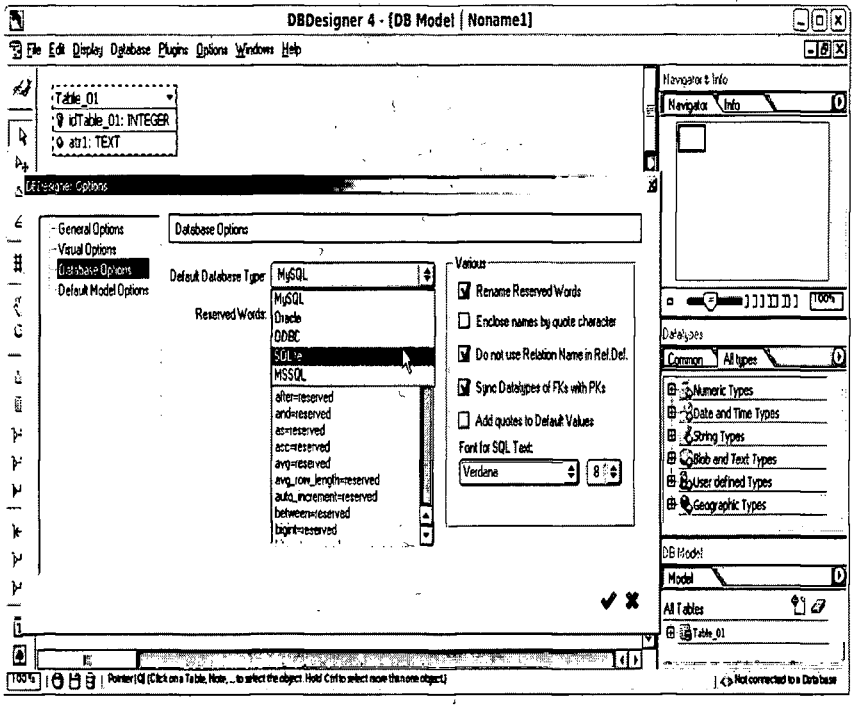

Рис. 26. DBDesigner4. Графический интерфейс

2.2. MySQL Workbench

Назначение. Визуальное проектирование и обслуживание баз данных MySQL.

Разработчики. MySQL Developer Tools Team (Alfredo Kojima, Alexander Musienko, Sergei Tkachenko, Johannes Taxacher, Mike Lischke, Maksym Yehorov).

Версии. Проект является преемником DBDesigner4. Первый релиз представлен в сентябре 2005 года. Свежая версия – 5.2.31 (13 октября 2010 года).

MySQL распространяется в двух редакциях: Community Edition (лицензия GNU GPL), Standard Edition (ежегодная оплачиваемая подписка; имеются дополнительные функции).

Платформы. MS Windows, GNU/Linux.

Установка. Файлы для установки можно получить на официальном сайте проекта [22].

Функциональные возможности:

– базовый набор функций для визуального проектирования баз данных (выбор нотации, определение сущностей и атрибутов, построение ER-диаграмм, определение представлений и др.);

– трансляция диаграммы в SQL-скрипты, выполнение которых на сервере приводит к созданию базы данных (прямой инжиниринг);

– восстановление визуальной модели из SQL-скрипта или базы данных, с которой установлено соединение (реверс-инжиниринг). Синхронизация модели и существующей базы данных (внесение изменений в диаграмме сопровождается соответствующими изменениями в базе данных, с которой установлено соединение);

– администрирование MySQL-сервера (создание экземпляров MySQL-серверов, управление ими, конфигурирование и администрирование существующих экземпляров сервера).

Дополнительный функционал в коммерческой версии обеспечивается подключаемыми модулями. К нему относятся, например, документирование схемы базы данных с помощью технологии DBDoc [6], валидация схемы (общая валидация – поиск типовых ошибок, допускаемых при ER-моделировании, и специфическая для MySQL). Существует возможность создания новых подключаемых модулей.

Формат файлов. Файлы имеют расширение .wmb, формат хранения основан на xml.

Информация. На официальном сайте проекта [22] содержатся общие сведения о проекте и команде разработчиков, новости, FAQ, весьма подробное руководство, вики, средства поддержки дискуссий – чаты, форумы и др.

MySQL Workbench имеет графический интерфейс со вкладками, что обеспечивает компактное размещение элементов управления при широких возможностях системы. Некоторые моменты MySQL Workbench приведены на рисунках 27, 28, 29, 30.

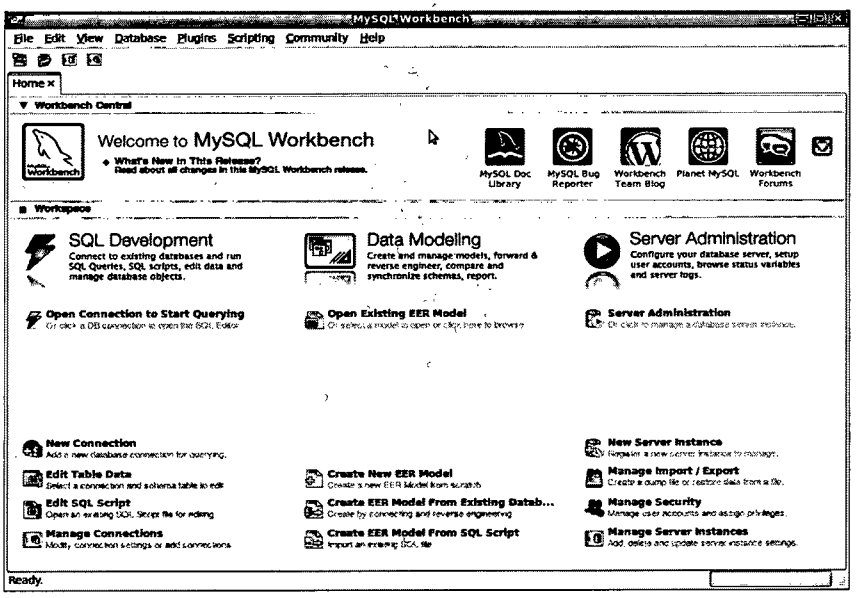

Рис. 27. MySQL Workbench. Вкладка Home

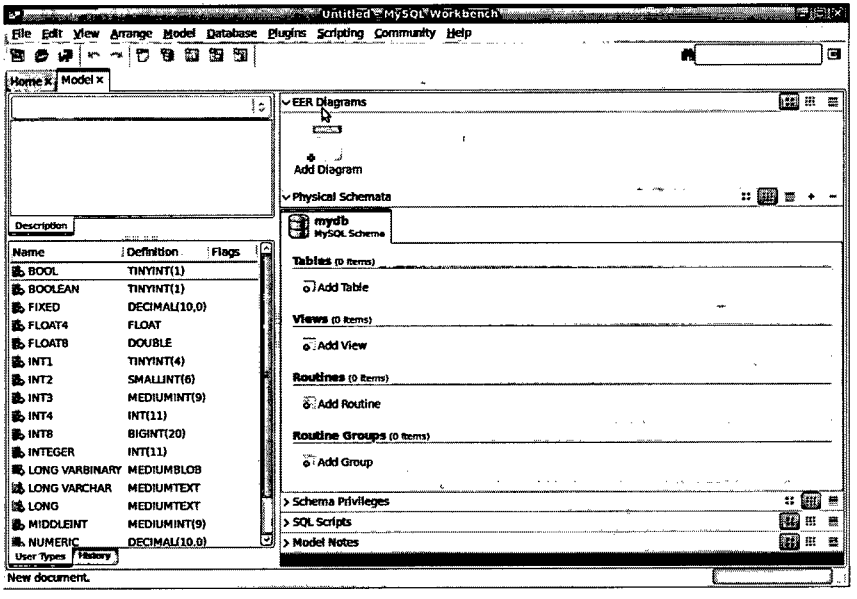

Рис. 28. MySQL Workbench. Создание новой модели

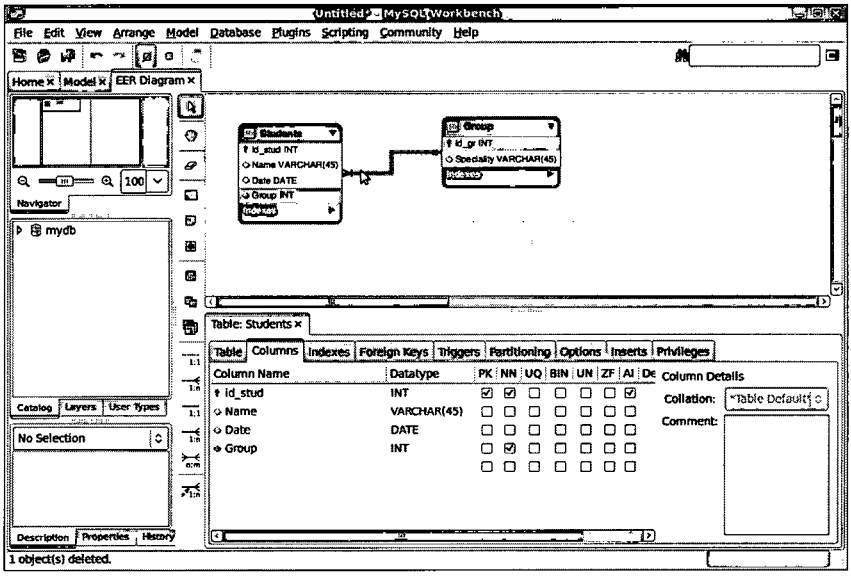

Рис. 29. MySQL Workbench. Редактирование таблицы

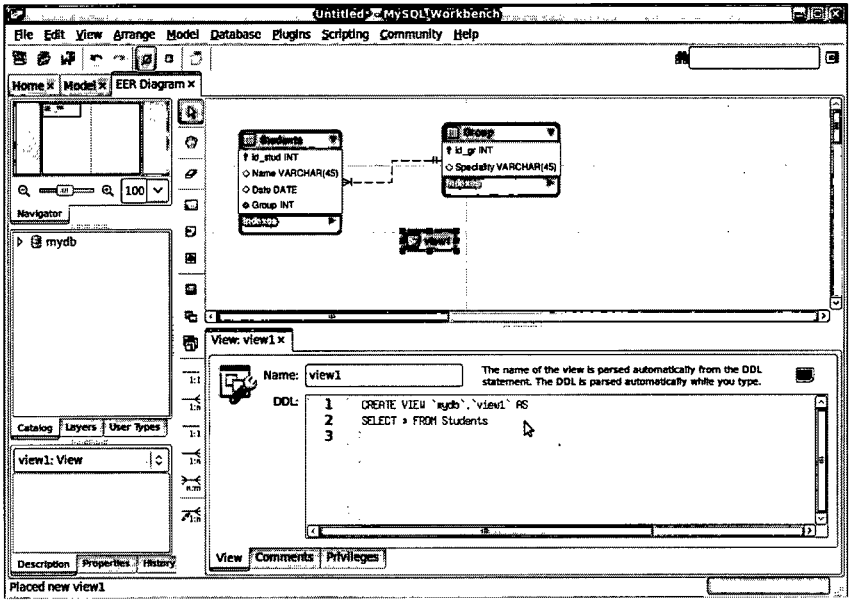

Рис. 30. MySQL Workbench. Определение представления

2.3. Dia

Назначение. Построение диаграмм в различных нотациях. Рассматривается создателями в качестве альтернативы коммерческой MS Office Visio. Приложение является частью GNOME Office.

Разработчики. Первоначальный автор: Alexander Larsson (Швеция). В настоящее время разработчиками являются Cyrille Chepelov и Lars Clausen. Hans Breuer и Steffen Macke занимаются портированием Dia под win32.

Версии. Первая новость на официальном сайте проекта [7] «Updated to V0.11, fixes stupid segfault bug. Works on Solaris» датируется 15 августа 1998. Текущая версия – 0.97.1 bug-fix release (4 января 2010 года).

Платформы. GNU/Linux, win32.

Установка. Установка Dia в GNU/Linux, как правило, может быть выполнена при помощи менеджера пакетов. Также можно получить файлы Dia, включая исходный код, с официального сайта проекта [7]. В MS Windows установка заключается в обычной распаковке архива. Запуск в MS Windows осуществляется следующим образом: dia/bin/dia.exe, однако возможны проблемы с запуском, если в пути к dia.exe встречаются русские буквы. Лицензия GPL.

Функциональные возможности:

– построение диаграмм в различных нотациях;

– экспорт визуальных моделей в различные растровые и векторные форматы, в макросы на TeX и др.;

– генерация исходного кода на основе UML-диаграмм;

– работа на нескольких слоях.

Графические элементы сгруппированы в библиотеки: схемы алгоритмов и программ, ER в различных нотациях, DFD в нотации Гейна и Сарсона, SADT/IDEF0, BPMN, UML, Cisco и многие другие. Существует возможность создания библиотек путем комбинирования имеющихся или вновь разрабатываемых элементов. Определение каждого элемента содержится в XML-файле на языке, являющемся подмножеством SVG. Для создания нового элемента можно вручную редактировать такой файл, либо нарисовать объект и сохранить его таким образом, чтобы затем включить в библиотеку компонентов. Dia предоставляет такую возможность.

На рисунке 31 продемонстрировано добавление недостающего элемента «Узел объекта» в библиотеку UML. Узел объекта в UML

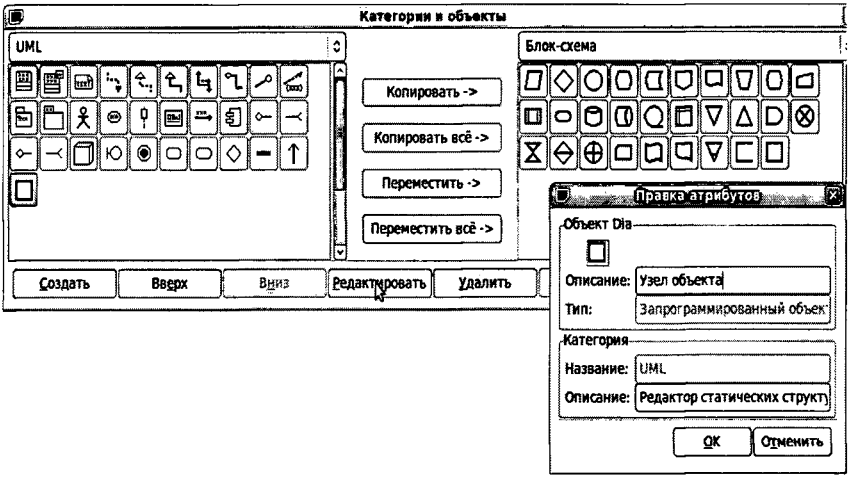

Рис. 31. Dia. Добавление элемента в библиотеку

представляет собой прямоугольник с текстом и возможностью присоединения стрелки. Похожий элемент берется из библиотеки «Блок-схема».

Существует много программ, совместное использование которых с Dia обеспечивает решение задач прямого и обратного инжиниринга. Их список приведен на сайте [8]. Некоторые примеры таких программ: cpp2dia (построение UML-диаграммы по исходному коду на C++), OraSchemaDoc (документирование баз данных Oracle), Dia2Code (генерация исходного кода на основе UML-диаграммы), Dia2SQL (генерация SQL-скриптов на базе ER-диаграмм).

Функциональность Dia может быть расширена написанием скриптов на языке Phyton.

Формат файлов. Данные хранятся в собственном бинарном формате программы и имеют расширение .dia.

Информация. На официальном сайте проекта [7] содержатся исходные коды программы, бинарные файлы, новости проекта, данные о разработчиках, различная документация (FAQ, краткое введение в Dia, подробное руководство, введение в язык UML); примеры диаграмм; скрипты на языке Python (например, для генерации исходного кода по диаграмме).

На рисунке 32 показан экран компьютера во время работы программы. Выбрана библиотека элементов SADT/IDEF0.

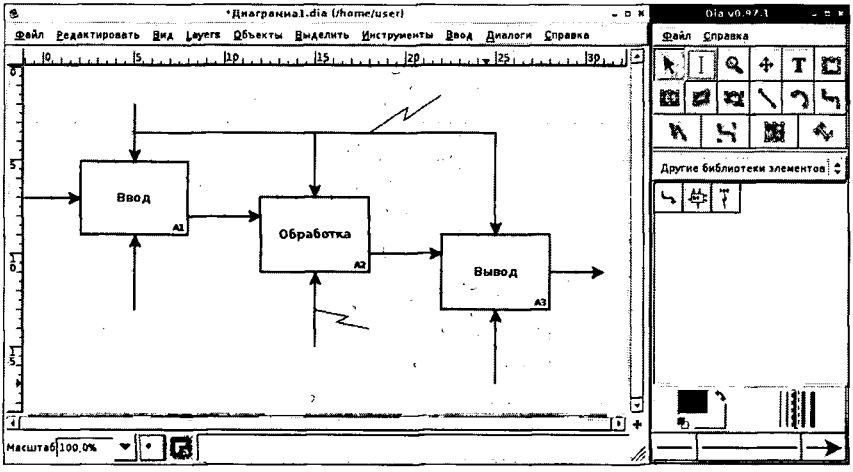

Рис. 32. Dia. Основные окна

2.4. Star UML

Назначение. UML-моделирование.

Разработчик. Plastic Software.

Версии. Изначально система имела называние Plastic. Последняя стабильная версия StarUML – 5.0 выпущена 30 декабря 2005 года, однако последняя новость на официальном сайте проекта [28], связанная с добавлением нового модуля в систему, датируется 8 января 2008 года.

Платформа. MS Windows.

Установка. Требуется инсталляция системы. Файлы для установки можно скачать с сайта [28]. Лицензия GPL.

Функциональные возможности:

– построение диаграмм на языке UML 2.0;

– поддержка концепции модельно-ориентированного подхода к разработке программного обеспечения MDA [21];

– поддержка стандарта для обмена метаданными XMI;

– генерация исходного кода на языках C++, Java и C#;

– генерация документации в форматах от Microsoft: doc, xls, ppt.

Функциональность расширяется при помощи подключаемых модулей, которые можно получить на сайте [28]. Примеры модулей: Software Process Engineering Metamodel (SPEM, моделирова-

ние процесса разработки программного обеспечения с определением ролей, задач, продуктов в соответствии со спецификацией OMG [23] SPEM); Agent Modeling Language Profile (AML, моделирование многоагентных систем на языке AML); the Softgoal Profile (анализ нефункциональных требований) и др.

Формат файлов. Данные сохраняются в xml-формате, файл имеет расширение .uml. Система StarUML совместима с популярным средством UML-моделирования Rational Rose.

Информация. На сайте проекта представлены общие сведения о проекте, снимки экранных форм, шаблоны, техническая документация (руководства для пользователя и разработчика, в том числе на русском языке), статьи, подробная информация о дополнительных модулях, форум [28].

2.5. Umbrello

Назначение. Объектно-ориентированное проектирование на языке UML.

Разработчики. The Umbrello Team (© 2001 Пауль Хенсген (Paul Hensgen), © 2002–2005 Авторы Umbrello).

Версии. Текущая версия – 2.51. Обновления выпускаются регулярно. Версии 2.x включены в состав рабочего стола KDE4. В KDE3 программа не входила.

Платформа. GNU/Linux, MS Windows.

Установка. Установка в GNU/Linux выполняется из репозитория. Кроме того, файлы для установки и исходный код можно получить на официальном сайте проекта [32]. Лицензия GPL.

Функциональные возможности:

– базовые функции для построения UML-диаграмм;

– генерация на основе построенной модели исходного кода более чем на 15 языках, среди которых C++, C#, PHP, Java и др. Для этого используются так названные генераторы кода;

– экспорт ER-диаграммы в SQL-скрипт, запуск которого на сервере приводит к созданию базы данных;

– реверс-инижиниринг, состоящий в генерации диаграмм по исходному коду. Соответствующая функция в Umbrello носит название «Импортировать классы»;

– автоматическое создание технической документации в формате DocBook [9], содержащей полную информацию о модели. Существует множество утилит, преобразующих DocBook-

документы в такие форматы, как pdf, html, OpenDocument, man-страницы и др.

Формат файлов. Данные проекта можно сохранять в собственном формате программы или в форматах DocBook и XHTML.

Информация. Полезна любая информация, касающаяся объектно-ориентированного анализа и языка UML. Описание функциональных возможностей Umbrello, снимки экранных форм, техническую документацию, данные о разработчиках и другую информацию можно получить на официальном сайте проекта [32]. Однако в связи с включением Umbrello в KDE4 сайт обновляется редко.

Система Umbrello обладает интуитивно-понятным интерфейсом, что видно из рисунке 33. Впрочем, дизайн систем аналогичного назначения очень похож.

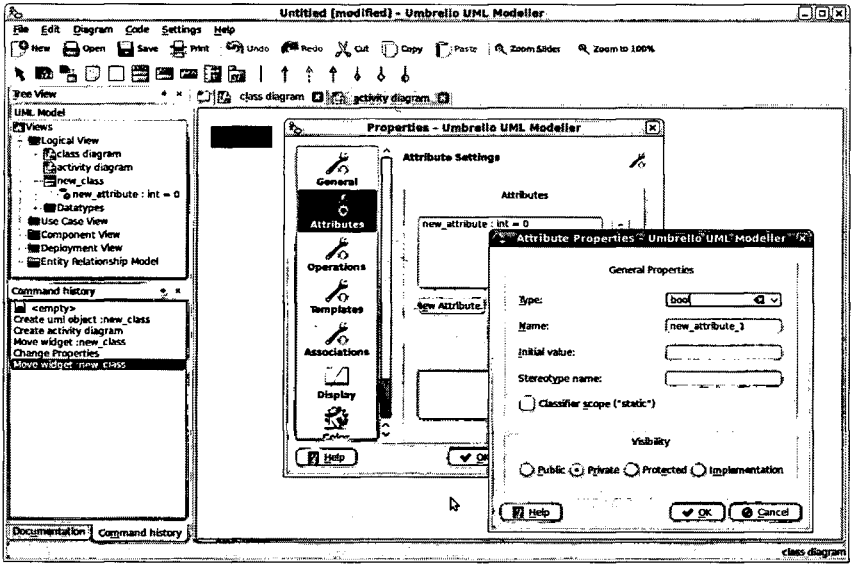

Рис. 33. Графический интерфейс Umbrello

2.6. ArgoUML

Назначение. UML-моделирование.

Разработчик. Коммерческая организация CollabNet, которая, однако, оказывает поддержку развитию программного обеспече-

ния с открытым исходным кодом и занимается разработкой инструментов управления жизненным циклом программного обеспечения для распределенных команд разработчиков.

Версии. Последняя версия: 0.32 (30 января 2011 года).

Платформа. Программное обеспечение разработано на Java, следовательно, является кроссплатформенным.

Установка. Скачать можно с сайта проекта [2]. ArgoUML является открытым программных обеспечением, защищенным лицензией EPL.

Функциональные возможности:

– поддержка UML (версии до 1.4);

– поддержка стандарта обмена метаданными XMI;

– генерация исходного кода на PHP 4.x, PHP 5.x, C#, C++, Java;

– реверс-инжиниринг (в том числе на основе байт-кода Java);

– верификация модели во время проектирования, в фоновом режиме [2];

– экспорт диаграмм в различные графические форматы.

Формат файлов. Модели сохраняются в xml-формате. Файл имеет расширение .uml или .zargo. В последнем случае xml-файл сжат архиватором.

Информация. Следует отметить богатое информационное наполнение официального сайта проекта. На сайте содержатся новости проекта, вики, руководство для быстрого старта, подробное руководство пользователя, документация, FAQ, форум и др. В разделе «User Manual» имеется подробное описание и самого языка UML.

На рисунке 34 показан экран с запущенной системой ArgoUML во время редактирования диаграммы прецедентов (Use Case) для службы такси.

2.7. MasterTZ

Назначение. Программа для создания технических заданий на программное обеспечение, сайт.

Разработчик. MySoftware (ООО «Хранители файлов») [57].

Платформа. Написана для MS Windows. Запускается в GNU/Linux при помощи wine.

Установка. Установка не требуется, программа представляет из себя единственный исполняемый файл.

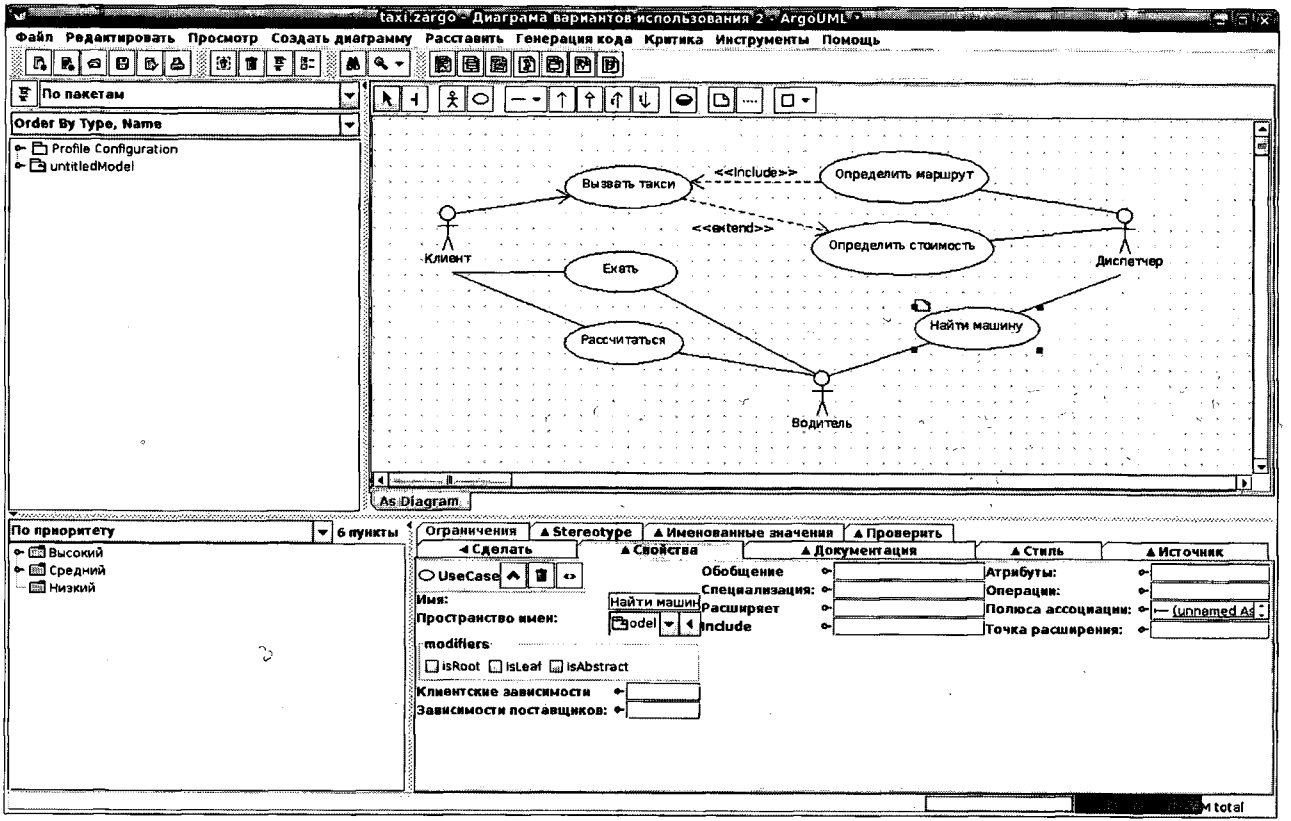

Рис. 34. ArgoUML. Диаграмма Use Case

Функциональные возможности. По существу программа предоставляет интерфейс для редактирования примера технического задания, вписанного в некий шаблон. Точнее, шаблона два: один для технического задания на программу, другой – на сайт. Структура документа может быть изменена – можно добалять новые разделы, редактировать и удалять существующие.

Формат файлов. На выходе – файл .doc или .rtf. Однако текст оказывается отформатированным при помощи пробелов и табуляций, что следует отнести к недостаткам программы. Существует также собственный формат .tz для хранения данных проекта.

Информация. Состав, содержание, правила оформления технического задания на создание автоматизированной системы регламентируется государственным стандартом [44]. Что касается изучения самой программы MasterTZ, то она обладает интуитивно-понятным интерфейсом (рис. 35), дополнительная информация не требуется.

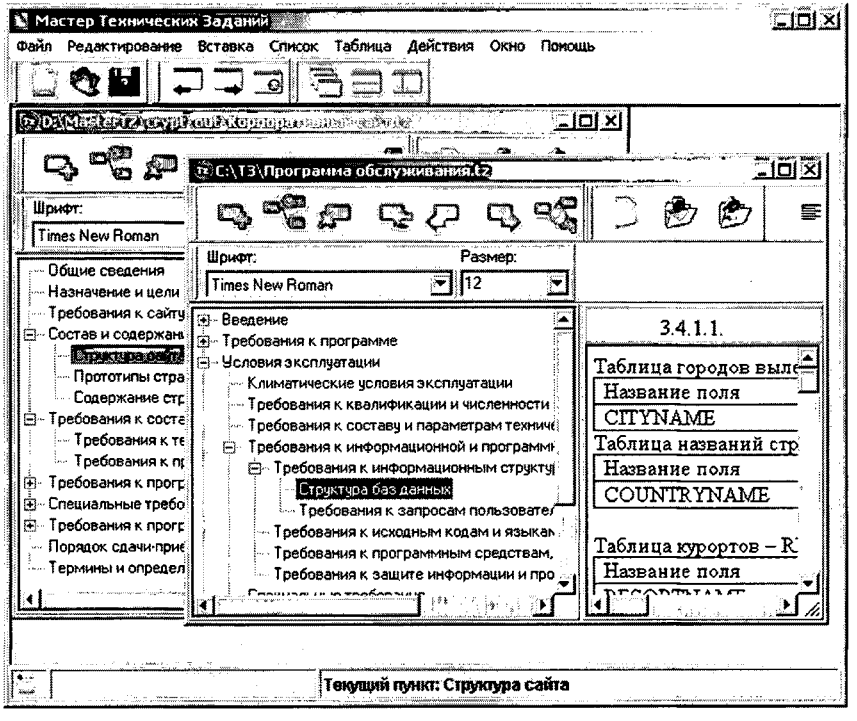

Рис. 35. Графический интерфейс программы MasterTZ

2.8. Planner

Назначение. Программа управления проектами для среды GNOME.

Разработчик. Шведская компания Imendio (разработка и поддержка программ для GTK+), которая в 2009 году разделилась на Lanedo и GmbH. Официальный сайт проекта – [22].

Версии. Текущая версия: v0.14.4 Release (15 апреля 2009 года).

Платформы. GNU/Linux, MS Windows.

Установка. Установка в GNU/Linux осуществляется из репозитория при помощи менеджера пакетов. Файлы установки для MS Windows можно скачать с сайта [25]. Существует также portable-версия Planner (то есть программу можно запускать, например, с USB-накопителя), найти ее можно на сайте [26].

Функциональные возможности:

– определение ресурсов и групп ресурсов;

– определение и детализация задач проекта. Для каждой задачи задаются: название, продолжительность, тип и значения других атрибутов. Могут быть введены дополнительные характеристики задач, например «продукт»;

– связывание задач и ресурсов. В этом и есть основная идея, реализованная в таких системах – привязать ресурсы к задачам;

– связывание задач между собой. Для каждой задачи указываются предшествующие задачи, в результате получается график выполнения работ;

– построение диаграммы Гантта[5]. На диаграмме задачи выделяются цветом. Например, красным (на нашем рисунке – темно-серым) показаны задачи, сокращение времени выполнения которых приведет к уменьшению продолжительности проекта в целом (критический путь);

---

[5] Диаграмма Гантта является популярной формой представления графика работ проекта. Инструмент для построения диаграммы Гантта есть практически во всех системах управления проектами. Диаграмма представляет собой графическое изображение работ в виде линий на временной шкале с указанием дат начала, окончания, степени их завершенности на текущий момент. Иногда кроме перечисленного на временной шкале отображаются и ключевые события. Диаграмма предложена в 1910 году американским инженером Генри Ганттом (1861–1919), изучавшим менеджмент на примере постройки кораблей во время Первой мировой войны.

– построение диаграммы использования ресурсов. Для каждого ресурса показано время, в течение которого он занят, а также насколько занят ресурс. Диаграмма использования ресурсов позволяет увидеть проблемы в графике выполнения работ, для этого фрагменты диаграммы окрашиваются в разные цвета. Например, красный означает, что ресурс задействован одновременно в двух задачах, так что в сумме он загружен более чем на 100%, серый – ресурс задействован не полностью (не на 100%), зеленый – ресурс вообще не задействован в течение какого-то времени. Синий цвет означает, что ресурс задействован на 100% (наиболее желательная ситуация);

– экспорт данных проекта в веб-страницу.

Формат файлов. Используется собственный формат хранения данных проекта, основанный на xml. Файлы имеют расширение .planner. Формат несовместим с другими системами управления проектами, однако является очень простым.

Информация. Разумеется, полезны книги об управлении проектами, например [54] и [56]. Имеется подробная справочная система на английском языке внутри самой программы.

Рассмотрим пример использования Planner. Для демонстрации некоторых возможностей системы мы специально допустим ошибку в определенном месте.

Пусть необходимо разработать небольшую программу, обеспечивающую взаимодействие компьютера и некоторого внешнего оборудования, например аналого-цифрового преобразователя (АЦП). Программа состоит из двух модулей (например, первый занимается вводом сигнала, второй – обработкой в режиме реального времени), разработку которых можно поручить разным людям. Таким образом, для создания программы привлекаются два исполнителя.

Сначала можно определить группы ресурсов и сами ресурсы (рис. 36). В данном случае ресурсами из группы «Исполнители» являются А.И. Попов и А.В. Родионов. Тип исполнителей – «Работа». Другой тип – «Материал», такой тип имеет ресурс «АЦП».

Далее определяются задачи, проводится их декомпозиция (рис. 37). Задачи в примере: разработка требований (разработка первичных требований, анализ первичных требований, углубленный анализ требований, разработка технического задания), выступление на семинаре, разработка (разработка модуля 1, разработка модуля 2, интеграция модулей). Выступление на семинаре выпол-

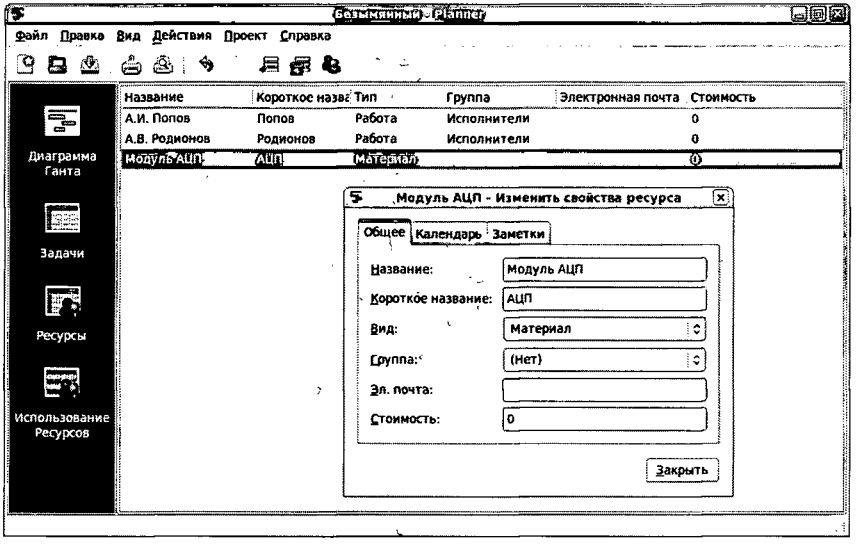

Рис. 36. Planner. Ресурсы и группы ресурсов

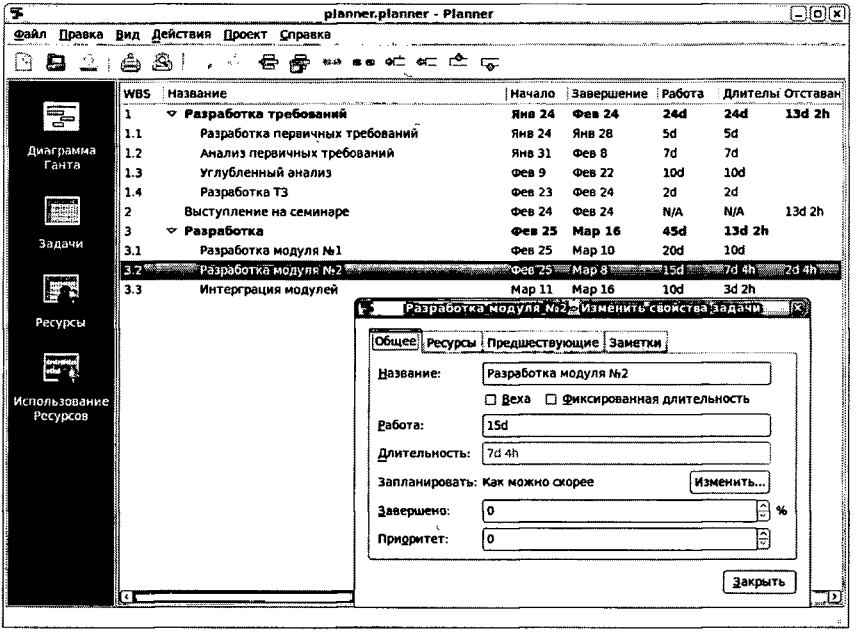

Рис. 37. Planner. Редактирование задачи

няется с целью обсуждения полученных требований. Будем иметь в виду, что разработка модулей включает и их тестирование, для которого необходим АЦП (для каждого модуля). Начало разработки каждого модуля запланировано на одно и то же время, но разработка модуля 2 занимает на 5 дней меньше, чем разработка модуля 1.

Рассмотрим характеристики задач:

– «Название»;

– «Работа» – величина усилия, необходимого для решения задачи;

– «Длительность» – объем календарного времени, необходимого для решения задачи. Вычисляется автоматически на основе данных в поле «Работа», данных о календаре и данных о ресурсах, привязанных к задаче. Задача может быть вехой, не имеет длительности, является событием, некоторой точкой в проекте. Задача может иметь фиксированную длительность, которая не зависит от ресурсов. Начало решения задачи можно запланировать тремя способами: как можно скорее, не позднее указанного числа, точно в указанное число;

– «Приоритет»;

– «Процент завершения».

К каждой задаче привязываются ресурсы (рис. 38). Можно указать загруженность ресурсов в процентах. В рассматриваемом примере разработкой требований занимаются оба исполнителя. АЦП в это время не задействован. На семинаре выступает один из исполнителей (Родионов). В разработке модуля 1 задействованы исполнитель Родионов и АЦП, в разработке модуля 2 – исполнитель Попов и АЦП. Здесь специально допущена ошибка: могут возникнуть затруднения при одновременном использовании оборудования[6]. Проблема проявляется, например, если программисты находятся в разных местах.

Диаграмма Гантта приведена на рисунке 39. Для задач указываются предшествующие задачи. Это можно сделать, находясь в рабочих пространствах «Задачи» и «Диаграмма Гантта». Во втором случае задачи связываются при помощи мыши. Красным цветом (на нашем черно-белом рисунке это темный цвет) выделены задачи, сокращение времени выполнения которых уменьшает про-

[6] Можно избежать проблемы, указав 50%-ную загруженность АЦП во время разработки модулей. Но не вполне понятно, что конкретно это означало бы.

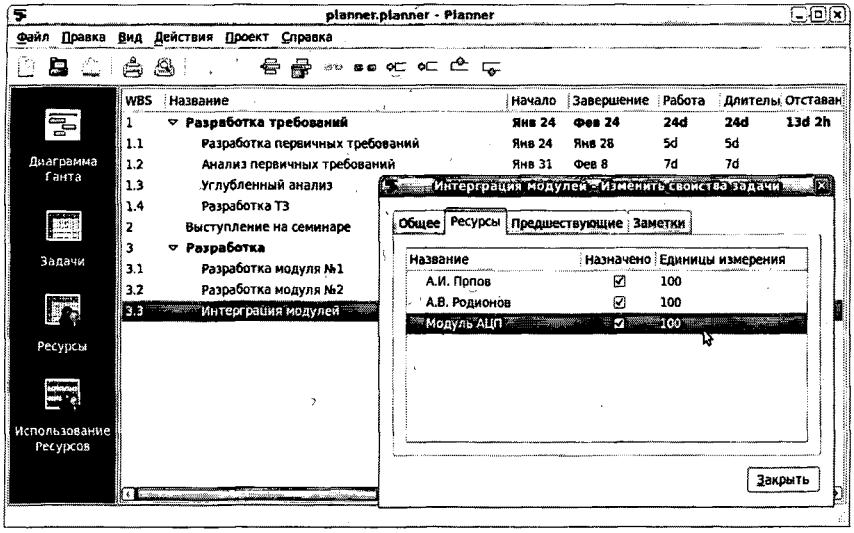

Рис. 38. Planner. Привязка ресурсов к задачам

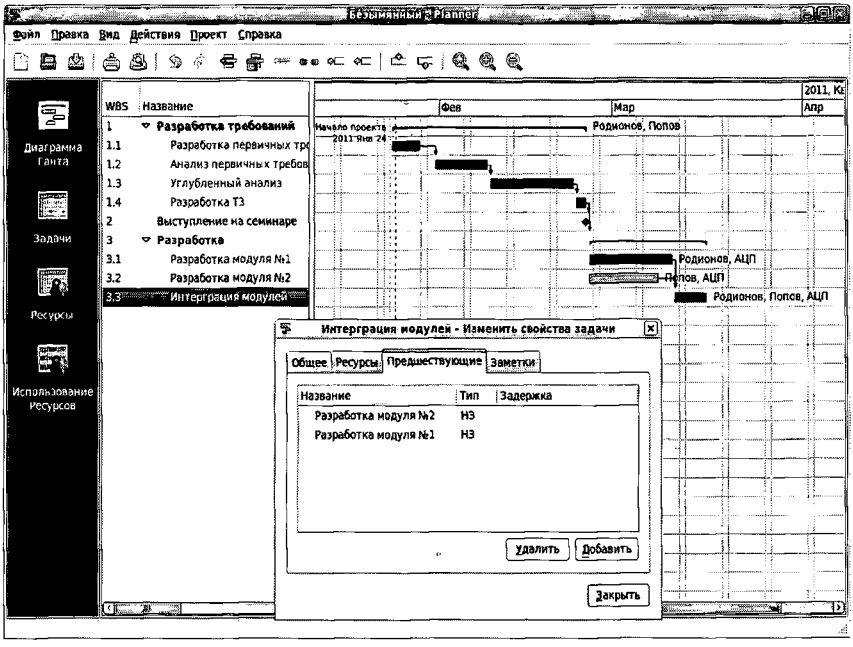

Рис. 39. Planner. Диаграмма Гантта

должительность проекта в целом. Другими словами, показан критический путь. Также показаны группы задач, которые для компактности можно сворачивать. Веха (важное событие в проекте) обозначается ромбом. Вертикальными серыми линиями выделены выходные дни и нерабочие часы. Возможно изменение масштаба диаграммы – от часов до кварталов.

На рисунке 40 приведена диаграмма использования ресурсов. На ней показано, в течение какого времени и в решении каких задач задействован тот или иной ресурс. Разными цветами (здесь – оттенки серого) показаны ситуации:

1) ресурс используется на 100% (желательно);

2) ресурс используется менее чем на 100%. Недогрузка;

3) ресурс не используется. В примере: АЦП не задействован во время решении задачи «Разработка требований». В это время его можно занять в другом проекте. Исполнитель Попов ничем не занят, пока исполнитель Родионов завершает разработку модуля 2;

4) ресурс используется в нескольких работах одновременно и суммарная загрузка превышает 100%. Выделяется красным. В примере АЦП используется при разработке модулей 1 и 2.

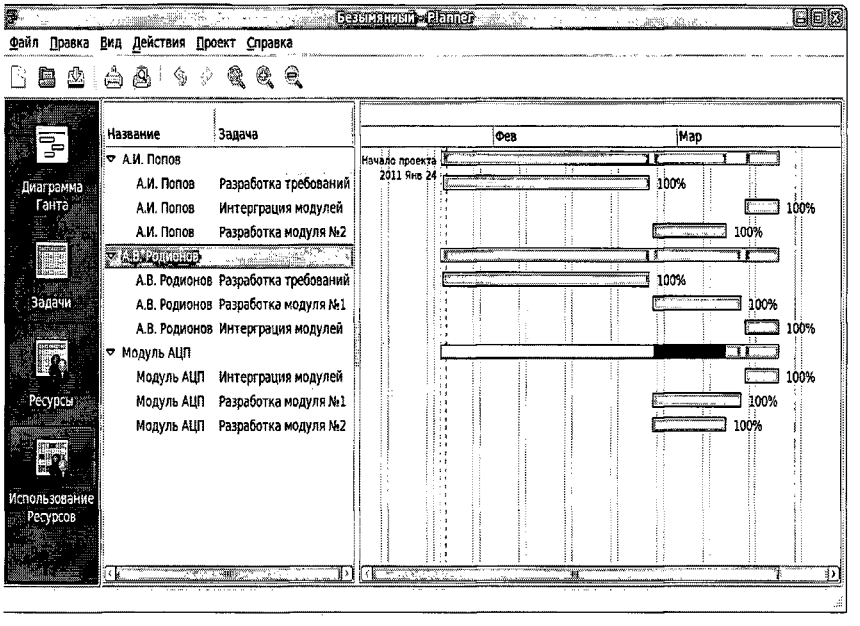

Рис. 40. Planner. Диаграмма использования ресурсов

На рисунке 41 показана веб-страница, полученная при экспорте данных проекта. Динамичность страницы достигается использованием JavaScript. Страница может использоваться как краткое руководство для всех участников проекта, она содержит общие сведения о проекте, диаграмму Гантта, подробные данные о задачах и данные о ресурсах.

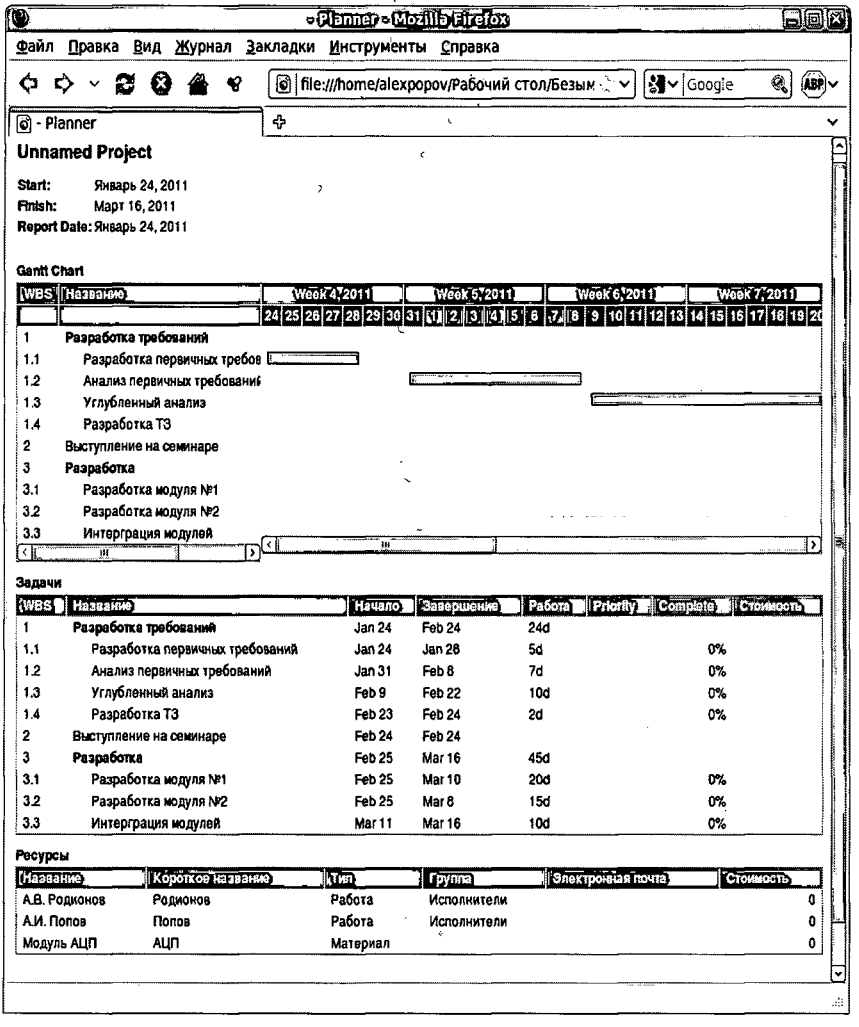

Рис. 41. Planner. Экспорт в html

2.9. GanttProject

Назначение. Система управления проектами.

Разработчики. Авторами проекта являются Dmitry Barashev и Alexandre Thomas. Полный список разработчиков и переводчиков можно найти на официальном сайте проекта [11].

Версии. Текущая версия – 2.0.10 (14 сентября 2009 года).

Платформы. Программное обеспечение системы написано на языке Java, чем обеспечивается его кроссплатформенность.

Установка. Файлы для установки находятся на сайте проекта. Система защищена лицензией GNU GPL.

Функциональные возможности:

– определение ресурсов;

– определение, детализация работ, определение фаз и вех проекта;

– привязка ресурсов к работам;

– построение диаграммы Гантта;

– генерация сетевой диаграммы PERT[7] на основе диаграммы Гантта;

– экспорт диаграмм в изображения в формате png;

– формирование отчетов в форматах pdf и html;

– импорт/экспорт документов Microsoft Projec;

– сохранение данных в формате csv;

– совместная работа над проектом по протоколу WebDAV. Web-based Distributed Authoring and Versioning (WebDAV) – защищенный сетевой протокол высокого уровня, работающий поверх HTTP для доступа к объектам и коллекциям.

Формат файлов. Данные проекта сохраняются в xml-файле с расширением .gan или .xml.

Информация. Как и в предыдущем случае, полезна литература по управлению проектами. На сайте проекта информации обучающего характера не слишком много [11].

На рисунке 42 приведен экран системы GanttProject во время построения диаграммы Гантта и сетевого графика (PERT диаграмма). В примере проект содержит два этапа, первый этап включает задачи 1.1, 1.2, 1.3, второй – 2.1 и 2.2. Построение сетевого графика

[7] Program (Project) Evaluation and Review Technique (PERT) – технология оценки и анализа программ (проектов) [54; 56]. GanttProject строит по диаграмме Гантта диаграмму PERT, представляющую собой граф, вершины которого соответствуют работам.

происходит автоматически при выборе соответствующего пункта меню.

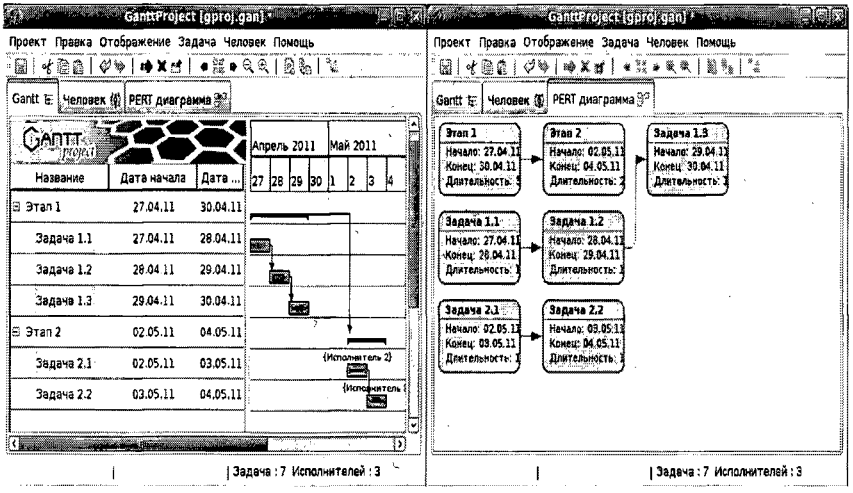

Рис. 42. GanttProject. Диаграмма Гантта и сетевой график

2.10. Ramus Educational

Назначение. Программное обеспечение Ramus предназначено для описания бизнес-процессов предприятия на языках IDEF0 и DFD с созданием систем классификации и кодирования. Ramus рассматривается разработчиками как инструмент бизнес-аналитиков в проектах по построению или реорганизации систем управления предприятием. Ramus предназначен для использования в проектах следующих классов: реинжиниринг бизнес-процессов, внедрение процессного управления, построение систем менеджмента качества, построение систем управления знаниями и др.

Разработчик. Ramus, официальный русскоязычный сайт [59].

Версии. Текущая версия – 1.2.7 (1 октября 2010 года).

Для свободного скачивания с официального сайта доступен Ramus Educational (текущая версия – 1.1.1). Ramus Educational ориентирован прежде всего на использование студентами вузов. Он не имеет количественных ограничений, но имеет функциональные, в частности, доступен только в локальном варианте, ограничен перечень атрибутов классификаторов и др.

На официальном сайте также можно найти демонстрационную версию продукта, которая не ограничена по функциональности, но имеет некоторые количественные ограничения.

Внутри Ramus работает также проект Open Ramus, в рамках которого предоставляются базовые библиотеки Ramus (Ramus Core), написанные на Java и открытые под лицензией GPL. К Ramus Core относятся библиотеки для работы с данными, сохранения данных в файл, добавления стандартных типов данных, работы с СУБД, создания объектов, необходимых для работы с данными, добавления данных, необходимых для работы редакторов IDEF0 и DFD. Также доступны примеры программ, использующие библиотеки Ramus Core.

Платформы. Программное обеспечение Ramus написано на Java, поэтому является кроссплатформенным.

Установка. Все необходимые для установки файлы можно найти на сайте проекта [59]. Для установки в MS Windows права администратора не требуются.

Функциональные возможности. В образовательной версии Ramus представлены следующие группы функций:

– моделирование процессов с использованием методологий IDEF0 и DFD;

– разработка систем классификации и кодирования предприятия с внутренними связями и связями с моделями процессов;

– импорт/экспорт в формат IDL.

Профессиональная версия обладает значительно более широкими возможностями, включающими, например, формирование отчетности, генерацию сайта для доступа к данным проекта и др.

Формат файлов. Используется собственный формат хранения данных. Обеспечивается частичная совместимость с известным инструментом для моделирования, анализа, документирования и оптимизации бизнес-процессов AllFusion Process Modeler, ранее имевшим название BPWin [1].

Информация. В качестве основного источника информации можно указать официальный сайт проекта. На нем можно найти снимки экранных форм системы, презентационные материалы, публикации о современных подходах к управлению бизнес-процессами, форум и др.

На рисунке 43 представлено главное окно Ramus Educational во время редактирования диаграммы потоков данных.

Рабочее пространство «Классификаторы» продемонстрировано на рисунке 44.

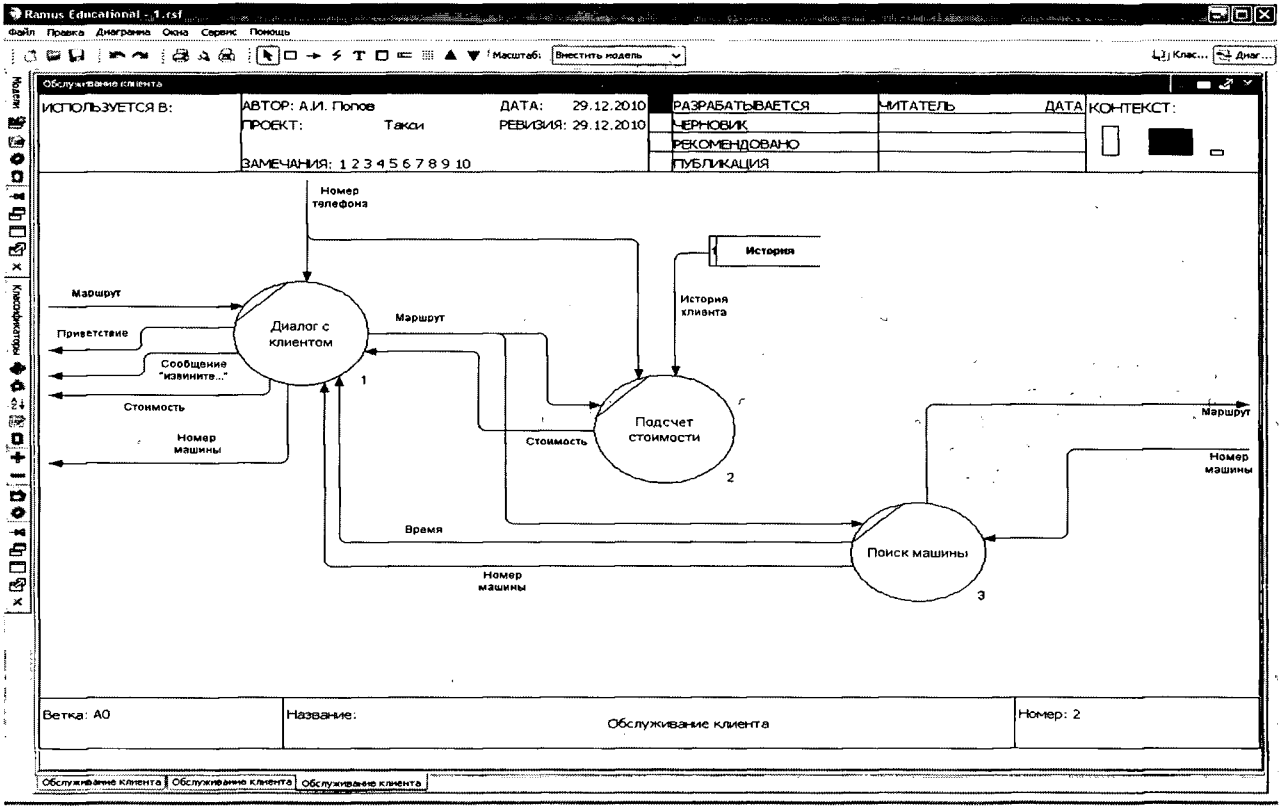

Рис. 43. Ramus Educational. Редактирование диаграммы потоков данных

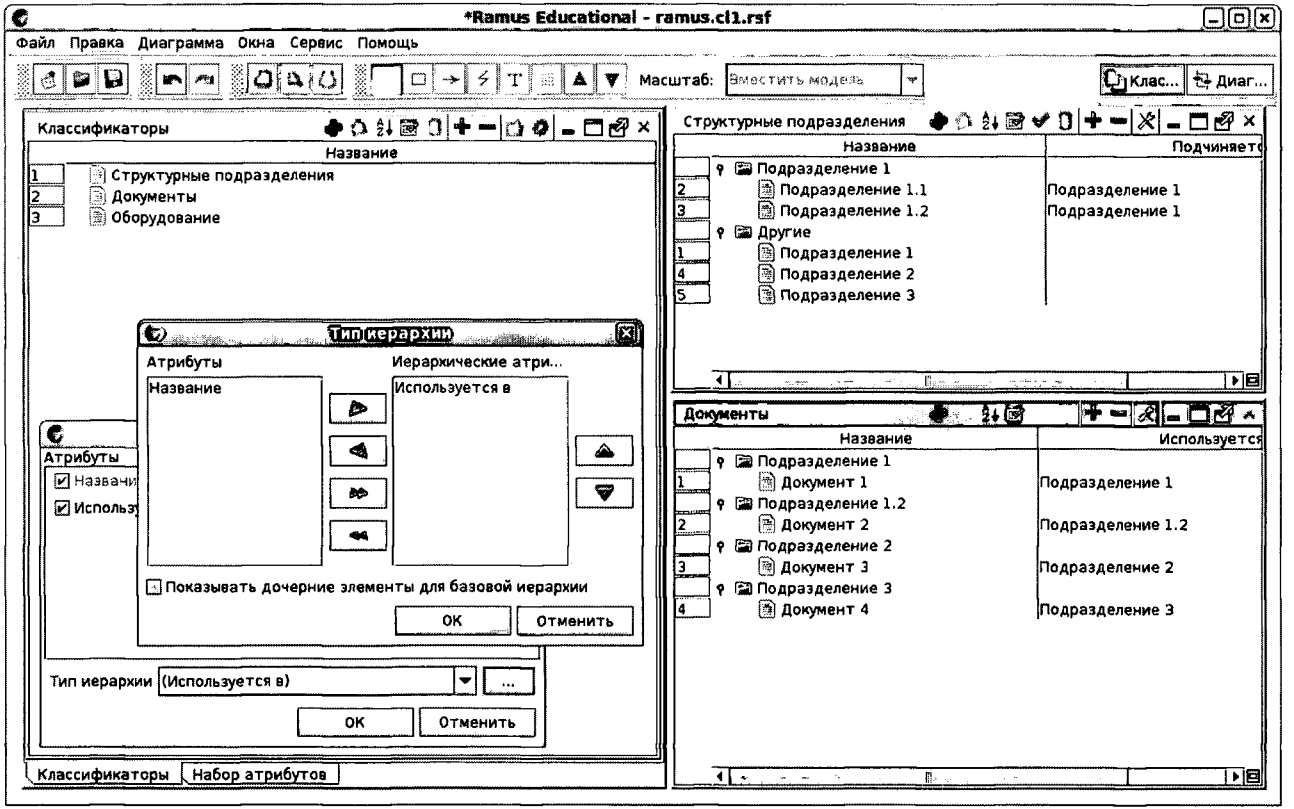

Рис. 44. Ramus Educational. Работа с классификаторами

В примере рассматриваются классификаторы, используемые в некоторой организации: «Структурные подразделения», «Документы», «Оборудование». Структурное подразделение характеризуется следующими атрибутами: «Название» и «Подчиняется» (показывает, какому структурному подразделению подчиняется данное подразделение). Атрибуты документа и единицы оборудования: «Название», «Используется в» (в каком подразделении используется документ или оборудование). Присвоив конкретные значения атрибутам, можно получить, например, иерархии для структурных подразделений и документов.

2.11. ARIS Express

Назначение. ARIS – методология и программный продукт для моделирования бизнес-процессов компании [4].

Разработчик. Права принадлежат Software AG.

Версии. ARIS Express – бесплатная редакция ARIS. ARIS Express рассматривается как инструмент для старта в деле управления бизнес-процессами. Текущая версия ARIS Express: 2.3 – 7.1.0.531253 (июнь 2010).

Платформы. Программное обеспечение написано на Java, поэтому является кроссплатформенным.

Установка. Для установки необходима Java версии не меньше чем 1.6.0. Необходимые файлы можно получить на официальном сайте проекта [4]. В GNU/Linux нужна Java от Sun/Oracle. Для установки требуется вступление в сообщество ARIS. Установщик требует введения ID и пароля, которые используются пользователем для входа на сайт сообщества [4]. Поэтому сначала надо зарегистрироваться на указанном сайте.

Функциональные возможности:

– стандартные функции моделирования бизнес-процессов, организационной структуры, данных, ИТ-инфраструктуры, ландшафта систем и др. (всего 9 диаграмм);

– экспорт в .pdf и .emf (формат хранения векторных графических файлов);

– модульность, реализуемая за счет использования так называемых фрагментов модели, определяемых пользователем;

– совместимость с профессиональной версией.

Система ориентирована на обучение моделированию процессов, на университеты и студентов. Интерфейс является интуитив-

но понятным. Во время моделирования предлагаются подсказки. Кроме того, поддерживается технология, называемая Smart design и предназначенная для новичков в моделировании бизнес-процессов. На основе вводимых пользователем текстовых данных система автоматически строит диаграммы, которые затем можно редактировать. Данная технология позволяет пользователям концентрироваться на содержании, а не на стандартах моделирования или правильности размещения объектов диаграмм.

Главное отличие профессиональной версии – в количестве поддерживаемых диаграмм, в ней их более 150. На сайте приведена таблица, в которой более детально сравниваются возможности ARIS Express и ARIS Professional [4]. В профессиональной версии, кроме большего количества диаграмм, доступны многопользовательская поддержка, центральное хранилище моделей, поддержка систем управления и документации и многое другое.

Формат файлов. Используется собственный формат файлов. Файлы имеют расширение .adf.

Информация. Система широко распространена, поэтому в Интернете много информации о ней. Это прежде всего официальный сайт проекта [4], содержащий описание продукта, инструкции по инсталляции, FAQ, системные требования, обучающее видео, примеры моделей, форумы, руководства, статьи, материалы по нотации BPMN. Существует русскоязычный ARIS-Портал [3].

На рисунке 45 показано окно программы ARIS Express во время построения EPC-диаграммы. Справа вверху – доступные символы, справа внизу – типовые фрагменты моделей.

2.12. Сравнение возможностей инструментов

Данные о рассмотренных CASE-средствах сведены в приведенной ниже таблице. Строки таблицы соответствуют функциональным возможностям инструментов (поддерживаемые графические нотации; прямой и обратный инжиниринг; автоматизированное формирование документации, в том числе стандартной; управление проектами) и другим важным характеристикам программного обеспечения (платформы, на которых работает приложение; является ли программное обеспечение свободным).

Знак «+» означает, что некоторая возможность предоставлена или некоторая особенность характерна для соответствующего

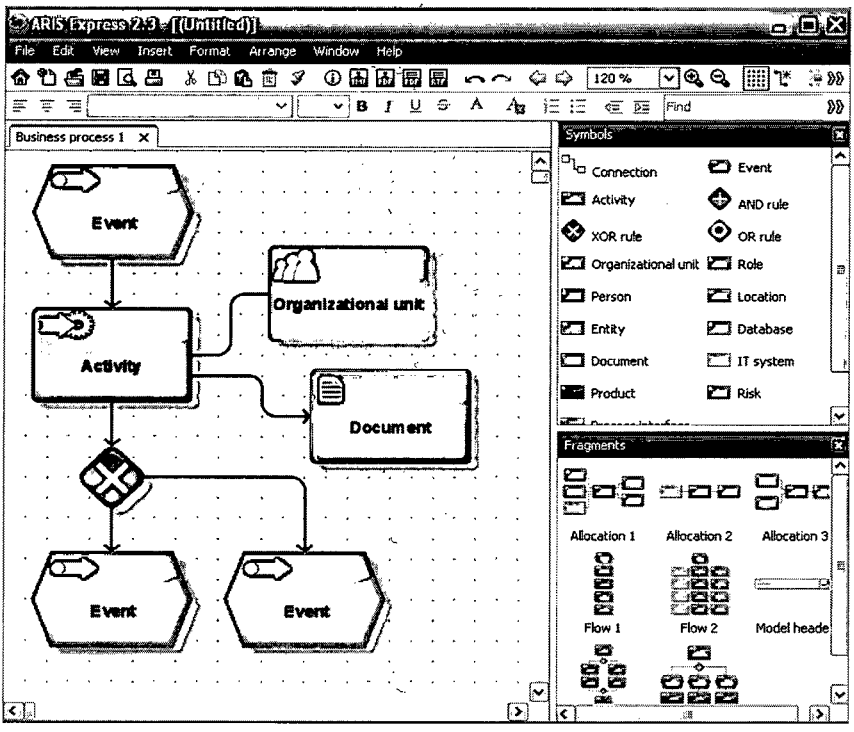

Рис. 45. Построение EPC-диаграммы в ARIS Express

инструмента. Отсутствие какого-либо знака в клетке следует понимать так, что возможность или характеристика вообще не соответствует назначению инструмента. Например, Planner не предназначен для UML-моделирования, а Dia, разработаный как редактор векторной графики, не поддерживает возможностей управления проектами. В некоторых клетках встречается знак «–». Его присутствие соответствует одной из следующих ситуаций: функция вписывается в область применения системы, но не реализована в конкретном продукте; программное обеспечение является несвободным; система не запускается на указанной платформе.

Все программы, представленные здесь, бесплатны, однако не все из них являются свободными. Так, ARIS Express и Ramus Educational Edition – бесплатные версии собственнического программного обеспечения.

Две из рассмотренных программ: «Мастер технических заданий» и StarUML не предназначены для запуска в GNU/Linux, однако они хорошо работают в этих системах при помощи wine [33].

Возможности программных инструментов

CASE-средства	Dia	MySQL Workbench OSE	DBDesigner	Mastertz	Planner	GanttProject	Umbrello	ArgoUML	StarUML	Ramus Educational Edition	ARIS Express
IDEF0	+									+	–
DFD	+									+	–
ERD	+	+	+				+	+	+		+
EPC	–										+
BPMN	+										+
UML	+						+	+	+		–
Прямой инжиниринг	+	+	+				+	+	+	–	–
Реверс-инжиниринг	+	+					+	+	–	–	–
Управление проектами					+	+					–
Документирование		+	–	+	+	+	+	–	+	–	–
MS Windows	+	+	+	+	+	+	+	+	+	+	+
GNU/Linux	+	+	+	–	+	+	+	+	–	+	+
GNU/Linux + Wine*				+					+		
Свободное ПО	+	+	+	+	+	+	+	+	+	–	–

* Приложение работает в системе GNU/Linux при помощи wine.

УЧЕБНЫЕ ПРОЕКТЫ

Лабораторные работы по дисциплинам, сходным с теми, названия которых перечислены во введении, часто выполняются в форме учебных проектов. Для начала проекта требуется краткое словесное описание предметной области. Такие описания содержатся во многих учебниках, например в [63]. Здесь мы также предложим некоторые предметные области, связанные с проектами.

Имея словесное описание предметной области, необходимую теоретическую подготовку и представленное в пособии программное обеспечение, можно предложить студентам выполнить следующие конкретные задания.

1. Формирование массива первичных требований [50]. В учебном проекте в качестве источников требований выступают студенты, преподаватель, предметная область, подобные системы и др. Много информации можно получить из Интернета. Полезным оказывается коллективное обсуждение первичных требований.

2. Углубленный анализ требований. Выполнение этого задания связано с выбором подхода к моделированию и построением ряда диаграмм. Например, можно выбрать алгоритмический или объектно-ориентированный подход.

3. Оценка сложности программного обеспечения при помощи функционально-ориентированного подхода [58].

4. Подготовка тестовых вариантов для проведения тестирования программного обеспечения [58].

5. Управление проектом, включая составление структурной схемы работ, сетевого графика работ, диаграммы Гантта и др.

Учебный проект № 1
Служба доставки

Руководитель организации, которая осуществляет доставку по городу некоторого товара (например пиццы), задумался о созда-

нии интернет-представительства своей фирмы, чтобы сделать ее более привлекательной для клиентов.

Сайт должен прежде всего обеспечивать возможность заказа товаров в режиме онлайн. Для этого пользователю предоставляется страница, содержащая краткое, но исчерпывающее описание каждого предлагаемого товара. В случае системы доставки пиццы речь идет о наличии меню. Информация о товарах должна быть актуальной. Например, если на кухне пиццерии закончился перец, то на сайте службы доставки пиццы скорее всего не должен предлагаться товар с названием «Чили». Кроме того, на сайте полезно представить информацию об альтернативных способах формирования заявки (например обычный заказ по телефону), общие сведения об организации, информацию о проводимых в настоящее время акциях и др. Работа с сайтом должна быть максимально простой для клиента.

Клиенту можно предложить зарегистрироваться в системе, чтобы в дальнейшем упростить оформление заявки (например, чтобы каждый раз не вводить адрес), получать новости в виде рассылки и т.д. Однако регистрация не является обязательной, то есть заказ может сделать и незарегистрированный пользователь.

По другую сторону сайта можно предусмотреть автоматическую доставку информации через соответствующий интерфейс к исполнителям заказа. В целом разрабатываемая система не должна быть изолированной. Необходимо продумать подходы к интеграции системы в существующую инфраструктуру предприятия. В частности, если в организации уже налажена работа некоторой системы бухгалтерского учета, то следует организовать связь с ней. Проектируемую систему также можно расширить средствами хранения «истории клиентов», что позволит, например, предоставлять скидки постоянным клиентам.

Учебный проект № 2
Телемедицина

Задача состоит в создании аппаратно-программного комплекса для одновременной регистрации основных физиологических показателей состояния организма человека: температуры, пульса, гастроэнтерологических показателей и др. Измерительная часть комплекса исполняется в виде портативных устройств для сбора, аккумуляции, первичной обработки и беспроводной пере-

дачи биомедицинских сигналов с использованием встраиваемых систем. Аппаратура строится на базе компонентов, выпускаемых современной промышленностью: датчиков, цифровых сигнальных процессоров, микроконтроллеров и др. Измерительные данные используются для диагностики, длительного мониторинга и др. Поддерживается необходимый уровень защиты данных. Комплекс строится с опорой на соответствующую нормативно-правовую базу, а также метрологические стандарты и стандарты представления медицинских данных в вычислительной системе.

Учебный проект № 3

Научный семинар

Кафедра информационных технологий некоторого университета проводит научный семинар, на который для докладов и дискуссий приглашаются студенты, магистранты, аспиранты, преподаватели, а также представители различных организаций. Работа семинара происходит в форме занятий, которые проводятся с некоторой периодичностью, например два раза в месяц. На одном занятии может быть представлено более одного доклада. Темы докладов относятся к сфере прикладных информационных технологий. У семинара имеется руководитель из числа преподавателей кафедры. Возможно, имеется также секретарь, в обязанности которого входит вывешивание объявлений о предстоящих занятиях, помощь в подготовке кабинета к занятию, ведение архива семинара и др.

Требуется произвести усовершенствование информационной инфраструктуры семинара с использованием современных информационных и, в частности, интернет-технологий. С одной стороны, это означает организацию хранения материалов, с другой – создание веб-сайта семинара.

Сайт создается прежде всего с целью информирования участников о работе семинара и содержит общие сведения о семинаре, объявления о предстоящих занятиях. Пользователям предоставляется возможность подписаться на рассылку объявлений о занятиях. Сайт предоставляет интерфейс для подачи заявки на выступление, которая содержит название доклада, аннотацию, ключевые слова, данные о докладчике и др.

Сайт обеспечивает доступ к архиву прошедших занятий с возможностями получения краткой информации о докладах, скачи-

вания прикрепленных файлов (презентаций, текстов докладов, видео), поиск материалов (по ключевым словам, с использованием классификаторов УДК, ГРНТИ и др.).

Управление контентом осуществляется секретарем, администрирование сайта – системным администратором. Их работа контролируется руководителем семинара, который также может совмещать в себе все названные роли, особенно если семинар не слишком масштабный.

Учебный проект № 4

Свободное программное обеспечение

В качестве еще одной предметной области можно предложить уже готовую и работающую программную систему и поставить задачу усовершенствования такой системы. Решение этой задачи возможно в полной мере, если исследуемое программное обеспечение защищено одной из свободных лицензий. Например, можно попытаться добавить в рассмотренный выше инструмент Dia новую функциональную возможность или разработать (доработать существующую) библиотеку элементов Dia. При решении подобных задач можно приобрести весьма полезный опыт изучения и описания архитектуры системы, анализа исходного кода программного обеспечения, реверс-инжиниринга с использованием специальных инструментов и др.

ГЛОССАРИЙ

Бизнес-процесс – совокупность взаимосвязанных мероприятий или задач, направленных на создание определенного продукта или услуги для потребителей [38].

Валидация – подтверждение на основе представления объективных свидетельств, что требования, предназначенные для конкретного использования или применения, выполнены, декларируемые свойства и характеристики подтверждаются, а поставленная цель (предназначение системы, комплекса, устройства) достигнута [45].

Жизненный цикл программного обеспечения – непрерывный процесс, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации [36].

Информационная система – комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, а также системный персонал и обеспечивающий поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей [55].

Классификатор – систематизированный перечень наименованных объектов, каждому из которых в соответствие поставлен уникальный код. Классификация объектов производится согласно правилам распределения множества объектов на подмножества в соответствии с установленными признаками их различия или сходства [38].

Концептуальная модель (инфологическая модель) – модель предметной области, состоящей из перечня взаимосвязанных понятий, используемых для описания этой области, вместе со свойствами и характеристиками, классификацией этих понятий по типам, ситуациям, признакам в данной области и законов протекания процессов в ней [38].

Нотация – система условных обозначений, принятая в какой-либо области знаний или деятельности [38].

Программная система – комплекс связанных между собой программ (программный комплекс), снабженный необходимой документацией, ориентированный на сбор, хранение, поиск, обработку, передачу и отображение информации в некоторой предметной области [50].

Реверс-инжиниринг (обратный инжиниринг) – исследование программного продукта, а также документации на него с целью понимания принципа его работы и, возможно, создания продукта с аналогичными функциями, но без копирования как такового [38].

Реинжиниринг бизнес-процесса – фундаментальное переосмысление и радикальное перепроектирование бизнес-процессов предприятий для достижения резких, скачкообразных улучшений в основных актуальных показателях их деятельности: стоимость, качество, услуги и темпы [38].

Свободное программное обеспечение – программные решения, в которых права пользователя на неограниченные установку, запуск, а также свободное использование, изучение, распространение и изменение (совершенствование) программ защищены юридически авторскими правами при помощи свободных лицензий [38].

Тестирование программного обеспечения – пошаговый процесс, при котором выполняется многократное исполнение программы с целью обнаружения ошибок. Шаги процесса задаются тестами [58].

Техническое задание (на автоматизированную систему) – основной документ, определяющий требования и порядок создания, развития или модернизации автоматизированной системы, в соответствии с которым проводится разработка автоматизированной системы и ее приемка при вводе в действие [44].

Управление проектом – процесс руководства всеми работами по проекту от начала до завершения [60].

Формальная верификация программы – доказательство с помощью формальных методов правильности или неправильности программы в соответствии с формальным описанием свойств программы [38].

CASE-средство (CASE-инструмент, CASE-средство) – любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла программного обеспечения и обладающее следующими основными характерными особен-

ностями: мощные графические средства для описания и документирования информационной системы, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности; интеграция отдельных компонент CASE-средств, обеспечивающая управляемость процессом разработки информационной системы; использование специальным образом организованного хранилища проектных метаданных (репозитория) [36].

DocBook – язык разметки для написания технической документации. Изначально был ориентирован на написание технической документации на компьютерное оборудование и программное обеспечение, однако сейчас допускает более широкое применение [9].

FAQ (Frequently Asked Question) – собрание часто задаваемых вопросов по какой-либо теме и ответов на них [38].

GNU GPL (GNU General Public License, универсальная общественная лицензия GNU, универсальная общедоступная лицензия GNU, открытое лицензионное соглашение GNU) – лицензия на свободное программное обеспечение, созданная в рамках проекта GNU в 1988 году. Предоставляет пользователю права копировать, модифицировать и распространять (в том числе на коммерческой основе) программы, а также гарантировать, что и пользователи всех производных программ получат эти права [38].

IDL (Interface Description Language или Interface Definition Language) – язык спецификаций для описания интерфейсов, синтаксически похожий на описание классов в языке C++ [38].

MDA (OMG's Model Driven Architecture) – модельно-ориентированный подход к созданию сложного программного обеспечения, разрабатываемый OMG [23] и базирующийся на стандартах OMG. В соответствии с данным подходом платформно-независимая модель приложения или интегрированной системы, построенная с использованием UML или других поддерживаемых OMG стандартов моделирования, трансформируется в продукт для конкретной платформы, открытой или проприетарной, включая web-сервисы, .NET, CORBA, J2EE и др. [21].

XMI (XML Metadata Interchange) – стандарт OMG для обмена метаданными с помощью языка XML. XMI определяет способ представления объектов в терминах XML-объектов и атрибутов; включает стандартный механизм для организации ссылок внутри файла или из одного файла в другой. Кроме того, к сфере действия XMI относится валидация XMI-документа с использованием языка XML Schema [34].

XML (eXtensible Markup Language) – язык разметки, представляющий собой свод общих синтаксических правил; текстовый формат, предназначенный для хранения структурированных данных, для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки (например XHTML, SVG) [38].

WINE (WINE is not emulator) – программное обеспечение, которое позволяет пользователям GNU/Linux, Mac, FreeBSD и Solaris запускать Windows-приложения при отсутствии установленной копии MS Windows [33].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. AllFusion Process Modeler 7 (BPwin). Interface.ru: [сайт]. URL: http://www.interface.ru/home.asp?artId=102 (дата обращения: 29.10.2012).

2. URL: http://argouml.tigris.org/ (дата обращения: 29.10.2012).

3. ARIS-PORTAL: все о методологии и программном обеспечении ARIS. URL: http://www.aris-portal.ru/ (дата обращения: 29.10.2012).

4. Business process management discussions, news and articles | ARIS BPM Community. URL: http://www.ariscommunity.com/ (дата обращения: 29.10.2012).

5. Business Process Model and Notation (BPMN) 1.2. URL: http://www.omg.org/spec/BPMN/1.2/ (дата обращения: 29.10.2012).

6. Dbdocs: Database documentation generator. URL: http://code.google.com/p/dbdocs/ (дата обращения: 29.10.2012).

7. Dia a drawing program. URL: http://projects.gnome.org/dia/ (дата обращения: 29.10.2012).

8. Dia Links. URL: http://projects.gnome.org/dia/links.html (дата обращения: 29.10.2012).

9. DocBook.org. URL: http://docbook.org/ (дата обращения: 29.10.2012).

10. fabFORCE.net. URL: http://www.fabforce.net/ (дата обращения: 29.10.2012).

11. GanttProject. URL: http://www.ganttproject.biz/ (дата обращения: 29.10.2012).

12. George A. Miller. The Magical Number Seven, Plus or Minus Two // The Psychological Review. 1956. Vol. 63. P. 81–97.

13. Интернет-университет информационных технологий: [сайт]. Видеокурс. URL: http://www.intuit.ru/department/itmngt/designis/ (дата обращения: 29.10.2012).

14. Интернет-университет информационных технологий: [сайт]. Учебный курс. URL: http://www.intuit.ru/department/se/devis/ (дата обращения: 29.10.2012).

15. International Organization for Standartization. ISO. URL: http://www.iso.org/ (дата обращения: 29.10.2012).

16. ISO 9001:2000. Quality management systems – Requirements. URL: http://www.iso.org/iso/catalogue_detail?csnumber=21823 (дата обращения: 29.10.2012).

17. ISO/IEC 12207:1995 Information Technology – Software life cycle processes. URL: http://www.iso.org/iso/iso.catalogue/catalogue_tc/catalogue_detail.htm?csnumber=21208 (дата обращения: 29.10.2012).

18. ISO/IEC 15504. Information technology – Software process assessment. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=54537 (дата обращения: 29.10.2012).

19. ISO/IEC 9000-3-2002. Системная и программная инженерия (руководство по применению стандарта ISO 9001-2000 к программному обеспечению). URL: http://www.inteltec.ru/inteltec-forum/index.php?topic=42.0 (дата обращения: 29.10.2012).

20. Mapping ER Models into Relations. URL: http://db.grussell.org/section006.html (дата обращения: 29.10.2012).

21. MDA. URL: http://www.omg.org/mda/ (дата обращения: 29.10.2012).

22. MySQL Workbench 5.2.31. URL: http://wb.mysql.com/ (дата обращения: 29.10.2012).

23. Object Management Group. URL: http://www.omg.org/ (дата обращения: 29.10.2012).

24. Planner – GNOME Live! URL: http://live.gnome.org/Planner (дата обращения: 29.10.2012).

25. Planner on Windows. URL: http://winplanner.sourceforge.net/ (дата обращения: 29.10.2012).

26. PortableApps.com. Portable software for USB, portable and cloud drives. URL: http://portableapps.com/ (дата обращения: 29.10.2012).

27. SAP Deutschland. Geschäftsanwendungen, Unternehmenssoftware, Services und Support. URL: http://www.sap.com/germany/index.epx (дата обращения: 29.10.2012).

28. StarUML. The Open Source UML/MDA Platform. URL: http://staruml.sourceforge.net/en/ (дата обращения: 29.10.2012).

29. *Stephen A. White.* BPM Architect, IBM. Introduction to BPMN. October 16. 2006.

30. TeacherTube Videos. URL: http://wwwl.teachertube.com (дата обращения: 29.10.2012).

31. The McGraw-Hill Companies. Structured system analysis and design. New Dehli, 2007. 406 p.

32. Umbrello UML Modeller. URL: http://uml.sourceforge.net/ (дата обращения: 29.10.2012).

33. WineHQ. Run Windows applications on Linux, BSD, Solaris and Mac OS X. URL: http://www.winehq.org/ (дата обращения: 29.10.2012).

34. XMI. URL: http://www.omg.org/spec/XMI/ (дата обращения: 29.10.2012).

35. *Бабич А.В.* Введение в UML. URL: http://www.intuit.ru/department/se/intuml/ (дата обращения: 29.10.2012).

36. *Вендров А.М.* CASE-технологии: современные методы и средства проектирования информационных систем. URL: http://citforum.ru/database/case/index.shtml (дата обращения: 29.10.2012).

37. *Вендров А.М.* Проектирование программного обеспечения экономических информационных систем: учебник М.: Финансы и статистика, 2002. 352 с.

38. Википедия. URL: http://ru.wikipedia.org/wiki/ (дата обращения: 29.10.2012).

39. Схемы алгоритмов и программ. Правила выполнения: ГОСТ 19.002-80. Введ. 1981-07-01. URL: http://fmi.asf.ru/library/book/Gost/19-002-80-82.html (дата обращения: 29.10.2012).

40. Схемы алгоритмов и программ. Обозначения условные графические: ГОСТ 19.003-80. Введ. 1981-07-01. URL: http://fmi.asf.ru/Library/Book/Gost/19003-80-82.html (дата обращения: 29.10.2012).

41. Единая система программной документации. Стадии разработки: ГОСТ 19.102-77. Введ. 1980-01-01. URL: http://protect.gost.ru/document.aspx?control=7&id=158091 (дата обращения: 29.10.2012).

42. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения: ГОСТ 19.701-90. Введ. 1992-01-01. URL: http://progost.ru/gost/001.001.080.050/gost-19.701-90/ (дата обращения: 29.10.2012).

43. Автоматизированные системы. Стадии создания: ГОСТ 34.601-90. Введ. 1992-01-01. URL: http://progost.ru/gost/001.035.080/gost-34.601-90/ (дата обращения: 29.10.2012).

44. Техническое задание на создание автоматизированной системы: ГОСТ 34.602-89. Введ. 1990-01-01. URL: http://progost.ru/gost/001.035.080/gost-34.602-89/ (дата обращения: 29.10.2012).

45. Системы менеджмента качества. Основные положения и словарь: ГОСТ Р ИСО 9000-2001. Введ. 2001-08-31. URL: http://progost.ru/gost/001.003.120.010/gost-r-iso-9000-2001/ (дата обращения: 29.10.2012).

46. Информационная технология. Процессы жизненного цикла программных средств: ГОСТ Р ИСО/МЭК 12207-99. Введ. 2000-07-01. URL: http://progost.ru/gost/001.035.080/gost-r-isodmjek-12207-99/ (дата обращения: 29.10.2012).

47. Системная инженерия – процессы жизненного цикла программных средств: ГОСТ Р ИСО/МЭК 15288-2005. Введ. 2007-01-01. URL: http://progost.ru/gost/001.035.080/gost-r-isodmjek-15288-2005/ (дата обращения: 29.10.2012).

48. *Гради Буч.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++. М.: Бином; СПб.: Невский Диалект, 1998. 560 с.

49. *Грекул В.И., Денищенко Г.Н., Коровкина Н.Л.* Проектирование информационных систем. М.: Бином, 2008. 300 с.

50. *Гусятников В.Н., Безруков А.И.* Стандартизация и разработка программных систем. М.: Финансы и статистика, 2010. 288 с.

51. *Закревский А.Д.* Параллельные алгоритмы логического управления. Минск, 1999. 202 с.

52. *Иванова Г.С.* Технология программирования. М.: МГТУ, 2006. 336 с.

53. *Кларк Э.М., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking: пер. с англ. / под ред. Р. Смелянского. М.: МЦНМО, 2002. 416 с.

54. *Клиффорд Ф. Грей, Эрик У. Ларсон.* Управление проектами: практическое руководство. М.: Дело и сервис, 2003. 528 с.

55. *Когаловский М.Р.* Перспективные технологии информационных систем. М.: ДМК Пресс: Компания АйТи, 2003. 288 с.

56. *Мартин Т., Тейт К.* Управление проектами. СПб.: Питер. 2006. 224 с.

57. Мастер технических заданий. My Software: [сайт]. Скачать программу. URL: http://www.mysoftware.ru/download/mastertz/ (дата обращения: 29.10.2012).

58. *Орлов С.А.* Технологии разработки программного обеспечения. СПб.: Питер, 2002. 321 с.

59. Официальный русскоязычный сайт проекта Ramus. URL: http://ramussoft.co.cc/ (дата обращения: 29.10.2012).

60. *Портни Стенли И.* Управление проектами «для чайников». М.: Вильямс, 2005. 352 с.

61. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования: Р 50.1.028-2001. Введ. 2002-07-01. URL: http://www.i-mash.ru/normatdok/r_pr_rd/2867-r_5010282001.html (дата обращения: 29.10.2012).

62. Разработка программного обеспечения. Создание программ. Программное обеспечение на заказ. URL: http://www.mysoftware.ru/ (дата обращения: 29.10.2012).

63. *Рыбанов А.А.* Инструментальные средства автоматизированного проектирования баз данных. Волгоград: Изд-во ВолгГТУ, 2007. 96 с.

64. *Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф.* Проектирование экономических информационных систем. М.: Финансы и статистика, 2003. 512 с.

65. Федеральное агентство по техническому регулированию и метрологии. URL: http://www.gost.ru/ (дата обращения: 29.10.2012).

Учебное издание

Попов Александр Игоревич

Свободные инструменты проектирования информационных систем

Учебно-методическое пособие

Фото на обложке www.egsc.usgs.gov

Редактор *Т.Ю. Ирмияева*
Оригинал-макет и дизайн обложки *О.Е. Болдыревой*

Подписано в печать 28.12.2012. Формат 60×84/16
Усл. печ. л. 4,5. Тираж 100 экз. Заказ № 638

Издательско-полиграфический центр имени В.Н. Булатова
ФГАОУ ВПО САФУ
163060 г. Архангельск, ул. Урицкого, д. 56