ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М. А. БОНЧ-БРУЕВИЧА»

М. В. Котлова, Е. В. Давыдова

МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Учебное пособие

СПб ГУТ)))

Санкт-Петербург
2015

УДК 004.75
ББК 32.973.202
К73

Рецензенты:
кандидат технических наук, заведующий кафедрой робототехники и автоматизации производственных систем Санкт-Петербургского государственного электротехнического университета «ЛЭТИ» им. В. И. Ульянова (Ленина)
М. П. Белов;
кандидат технических наук, доцент кафедры конструирования и производства радиоэлектронных средств Санкт-Петербургского государственного университета телекоммуникаций им. проф. М. А. Бонч-Бруевича
Т. В. Матюхина

Утверждено редакционно-издательским советом СПбГУТ в качестве учебного пособия

Котлова, М. В.

К73 Методы и средства проектирования информационных систем и технологий : учебное пособие / М. В. Котлова Е. В. Давыдова ; СПбГУТ. – СПб., 2015. – 64 с.

ISBN 978-5-89160-100-0

Предназначено для изучения дисциплины «Методы и средства проектирования информационных систем и технологий», а также может быть полезно при освоении дисциплины «Инструментальные средства информационных систем»

Представлена общая характеристика информационных систем, приведены основные понятия и определения процесса проектирования автоматизированных информационных систем на основе анализа предметной области, рассмотрены методология проектирования информационных систем, основные средства информационных систем, этапы анализа предметной области, понятия и структура унифицированного языка моделирования UML.

Предназначено для подготовки бакалавров по направлению 09.03.02 «Информационные системы и технологии», а так же для студентов других направлений СПбГУТ, изучающих методы и средства проектирования информационных систем и технологий.

УДК 004.75
ББК 32.973.202

ISBN 978-5-89160-100-0

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Содержание пособия отражает разделы учебной дисциплины «Методы и средства проектирования информационных систем и технологий». В учебном пособии приведены основные понятия и определения процесса проектирования автоматизированных информационных систем на основе анализа предметной области, даны методология проектирования информационных систем, основные средства информационных систем, этапы анализа предметной области, понятия и структура унифицированного языка моделирования UML.

В разд. 1 дано понятие информационной системы, описана ее архитектура и классификация. Приведены определения, компоненты и особенности каждого вида обеспечения автоматизированных информационных систем. Представлено одно из базовых понятий методологии проектирования таких систем – понятие и модели жизненного цикла.

Разд. 2 посвящен описанию методологий проектирования информационных систем, а именно, даны: базовые понятия процессов проектирования, рассмотрены каноническое и типовое проектирование.

В разд. 3 представлены этапы анализа предметной области и реинжиниринг бизнес-процессов.

Разд. 4 включает понятия и структуру унифицированного языка моделирования UML. Дает представление о структурных и поведенческих диаграммах UML.

Предназначено для подготовки бакалавров направления 09.03.02 «Информационные системы и технологии», а так же для студентов других направлений обучения СПбГУТ, желающих изучить методы и средства проектирования информационных систем и технологий.

1. ОБЩАЯ ХАРАКТЕРИСТИКА ИНФОРМАЦИОННЫХ СИСТЕМ

1.1. Определение информационных систем

Сегодня в научно-технической литературе, в связи с динамично протекающими процессами накопления знаний в области информационных технологий, пока отсутствует устоявшееся, однозначное универсальное определение понятия «информационная система».

Информационная система (ИС) – взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Информационная система – это автоматизированная система, предназначенная для организации, хранения, пополнения, поддержки и представления пользователям информации в соответствии с их запросами.

В состав основных *объектов информационной системы* могут быть включены:

– средства вычислительной техники, информационно-вычислительные системы, комплексы и сети;

– программные средства – операционные системы, системы управления базами данных, прикладные программы, программные средства телекоммуникации и др.;

– автоматизированные системы управления, автоматизированного проектирования, обработки данных и т. п.;

– системы и сети связи и телекоммуникации (для передачи, распределения и приема информации), средства и обеспечения;

– лингвистические средства – словари, тезарусы, классификаторы и др.

К основным обязательным *признакам* современной информационной системы можно отнести: выполнение одной или нескольких функций в отношении информации:

– единство системы (общая файловая система, единые стандарты и протоколы, единое управление (администрирование) и т. п.);

– возможность композиции и декомпозиции объектов системы при выполнении заданной функции.

В связи с вышеизложенным, будем использовать следующее определение:

информационная система – совокупность информационных, экономико-математических методов и моделей, технических программных, технологических средств и специалистов, предназначенная для сбора, хранения, обработки и выдачи информации и принятия управленческих решений;

автоматизированная информационная система (АИС) – это система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию установленных функций.

1.2. Архитектура информационных систем

Термин «архитектура» применительно к вычислительным системам появился задолго до создания первых АИС, тем не менее, он является одним из основополагающих и в сфере информационных технологий. Существуют различные подходы к определению архитектуры АИС, различные точки зрения и различная степень детализации рассмотрения; приведем некоторые из них.

Архитектура – это организационная структура автоматизированной системы.

Архитектура – это концептуальное описание структуры системы, включающее описание элементов системы, их взаимодействия и внешних свойств.

Выделяют два уровня архитектуры АИС:

– бизнес-архитектуру (бизнес-уровень);

– уровень информационных технологий (технический уровень).

Бизнес-архитектура является предметной областью для анализа и проведения автоматизации. На бизнес-уровне определяется набор задач, требований, характеристик, осуществляемых с помощью АИС.

Уровень информационных технологий (технический уровень) представляет собой интегрированный комплекс технических средств, используемых в АИС для реализации задач предприятия, и включает в себя как логические, так и технические (программные и аппаратные компоненты). В состав данного уровня включены следующие компоненты:

– архитектура программных систем;

– информационная архитектура;

– технологическая (инфраструктурная архитектура).

Под архитектурой программных систем понимают совокупность следующих технических решений:

– общий архитектурный стиль и общую организацию программной части АИС;

– деление программного комплекса на функциональные подсистемы и модули;

– свойства модулей, методы их взаимодействия и объединения, используемые интерфейсы.

Архитектура программной системы охватывает не только структурные и поведенческие аспекты, но и правила ее использования и интеграции с другими системами, функциональность, производительность, гибкость, надежность, эргономичность, технологические ограничения.

Информационная архитектура представляет собой логическую организацию данных, с которыми работает АИС, т. е. практически структуры баз данных и баз знаний, а также принципы их взаимодействия.

Технологическая архитектура описывает инфраструктуру, используемую для передачи данных. На этом уровне решаются вопросы сетевой структуры, применяемых каналов связи и т. д.

По мере развития программных систем все большее значение приобретает их комплексная интеграция для построения единого информационного пространства предприятия. Обеспечение такой интеграции является важнейшим элементом архитектуры, в противном случае АИС окажется неэффективной.

В современных стандартах четко определены процессы создания архитектуры, способной к удовлетворению не только сформулированных, но и потенциальных потребностей пользователей. К числу самых известных и авторитетных разработчиков стандартов в области АИС относятся следующие международные организации:

– SEI (Software Engineering Institute);
– WWW (консорциум World Wide Web);
– OMG (Object Management Group);
– организация разработчиков Java — JCP (Java Community Process);
– IEEE (Institute of Electrical and Electronics Engineers).

1.3. Классификация информационных систем

Информационные системы (ИС) можно классифицировать по различным признакам. В основу нижеприведенной классификации положен ряд существенных признаков, определяющих функциональные возможности и особенности построения современных систем; также принимались во внимание объем решаемых задач, используемых технических средств, организации функционирования и др.

По типу хранимых данных:

– под *фактографическим* типом данных принято понимать данные, представляющие собой описание некоторых фактов предметной области. Факт в ИС предстает в виде набора некоторых свойств (атрибутов), количественное значение которых, как правило, выражается простым типом данных. Пример: «1С: Бухгалтерия»;

– *документальные системы.* Под *документом* будем понимать хранящийся в информационной базе объект произвольной структуры, содержащий информацию произвольного характера, доступ к которому можно получить по его реквизитам. Под *реквизитами* документа будем понимать совокупность свойств этого документа, позволяющих однозначно его идентифицировать. Например, название документа, его номер, дата создания, электронная подпись и т. п. в качестве примеров документов можно привести статьи, тексты приказов, карты местности, звуковые записи и т. д. Примерами документальных информационных систем служат справочные юридические системы типа «Гарант», «Консультант» и др.

По характеру обработки данных:

– *информационно-поисковые системы* – производят ввод, систематизацию, хранение, выдачу информации по запросу пользователя без сложных преобразований данных (например, ИС продажи билетов, ИС библиотечного обслуживания и т. п.);

– *информационно-решающие системы* – осуществляют, кроме того, операции переработки информации по определенному алгоритму.

По характеру использования выходной информации:

– *управляющие АИС.* Для этих систем свойственны задачи расчетного характера и обработка больших объемов данных, например, АИС планирования производства или заказов, бухгалтерского учета;

– *советующие АИС*. Эти системы имитируют интеллектуальные процессы обработки знаний, а не данных (например, экспертные системы).

По областям применения:

– *системы организационного управления* – предназначены для автоматизации функций управленческого персонала, как промышленных предприятий, так и непромышленных объектов (гостиниц, банков, магазинов и др.). Основными функциями данных систем являются: оперативный контроль и регулирование, оперативный учет и анализ, перспективное и оперативное планирование, бухгалтерский учет, управление сбытом, снабжением и другие экономические и организационные задачи;

– *системы управления технологическими процессами* – служат для автоматизации функций производственного персонала по контролю и управлению производственными операциями. В таких системах обычно предусматривается наличие развитых средств измерения параметров технологических процессов (температуры, давления, химического состава и т. п.), процедур контроля допустимости значений параметров и регулирования технологических процессов;

– *системы автоматизированного проектирования (САПР)* – предназначены для автоматизации функций инженеров проектировщиков, конструкторов, архитекторов, дизайнеров при создании новой техники, сооружений или технологий. Основными функциями подобных систем являются: инженерные расчеты, создание графической и проектной документации, моделирование проектируемых объектов;

– *интегрированные (корпоративные) АИС* – используются для автоматизации всех функций фирмы (корпорации) и охватывают весь цикл работ – от планирования деятельности до сбыта продукции. Они включают в себя ряд модулей (подсистем), работающих в едином информационном пространстве и выполняющих функции поддержки соответствующих направлений деятельности.

В интегрированных АИС выделяют *функциональные* и *обеспечивающие* подсистемы.

Функциональные подсистемы информационно обслуживают определенные виды деятельности, характерные для структурных подразделений предприятия или функций управления. Интеграция функциональных подсистем в единую систему достигается за счет создания и функционирования обеспечивающих подсистем.

Функциональная подсистема представляет собой комплекс задач с высокой степенью информационных обменов (связей) между задачами. При этом под задачей понимается некоторый процесс обработки информации с четко определенным множеством входной и выходной информации. Состав функциональных подсистем определяется характером и особенностями автоматизируемой деятельности, отраслевой принадлеж-

ностью, формой собственности, размером предприятия. Деление АИС на функциональные подсистемы может строиться по различным принципам:

– предметному;
– функциональному;
– проблемному;
– смешанному (предметно-функциональному).

Состав *обеспечивающих подсистем* не зависит от конкретных функцио-нальных подсистем и предметной области.

1.4. Обеспечение информационных систем

Различают девять обеспечивающих подсистем или так называемое обеспечение АИС. Ниже приведены определения каждого вида обеспечения, его компоненты и особенности.

Информационное обеспечение – совокупность форм документов, классификаторов, нормативной базы и реализованных решений по объемам, размещению и формам существования информации, применяемой в АИС при ее функционировании.

Информационное обеспечение включает:

– описание технологических процессов;
– описание организации информационной базы;
– описание входных потоков;
– описание выходных сообщений;
– описание систем классификации и кодирования;
– формы документов;
– описание структуры массивов.

Системы классификации позволяют группировать объекты, выделяя определенные классы, которые характеризуются рядом общих свойств. *Классификаторы* представляют собой систематизированные своды, перечни классифицируемых объектов и имеют определенное (обычно числовое) обозначение. Применяются государственные, отраслевые, региональные классификаторы.

Назначение классификаторов:

– систематизация наименований кодируемых объектов;
– однозначная интерпретация одних и тех же объектов в различных задачах;
– возможность обобщения информации по заданной совокупности признаков;
– возможность сопоставления одних и тех же показателей, содержащихся в формах статистической отчетности;
– возможность поиска и обмена информацией между подсистемами и внешними АИС;

– оптимизация использования ресурсов вычислительной техники при работе с кодируемой информацией.

Используются три метода классификации объектов:

– иерархический;

– фасетный;

– дескрипторный.

В *иерархической системе* классификации каждый объект на любом уровне должен быть отнесен к одному классу, характеризуемому конкретным значением выбранного классификационного признака. Количество уровней классификации, соответствующее числу признаков, выбранных в качестве основания деления, характеризует глубину классификации.

Достоинства иерархической системы классификации: простота построения и использование независимых классификационных признаков в различных ветвях иерархической структуры.

Недостатками этой системы являются жесткая структура, осложняющая внесение изменений, так как это приводит к перераспределению классификатора, и невозможность группировать объекты по заранее не предусмотренным сочетанием признаков.

При использовании *фасетного метода* классификации допустимо выбирать признаки классификации независимо как друг от друга, так и от семантического содержания классифицируемого объекта. Признаки классификации называются фасетами (facet – рамка). Каждый фасет содержит совокупность однородных значений данного классификационного признака, причем значения в фасете могут располагаться в произвольном порядке. Схема построения фасетной системы классификации представляется в виде таблицы. Названия столбцов соответствуют выделенным классификационным признакам (фасетам). В каждой клетке таблицы хранится конкретное значение фасета. Процедура классификации состоит в присвоении каждому объекту соответствующих значений из фасетов.

Достоинства фасетной системы классификации: возможность создания большой емкости классификации, т. е. использование большого числа признаков классификации и их значений для создания группировок; возможность простой модификации всей системы классификации без изменения структуры существующих группировок.

Недостатком системы является сложность ее построения, так как необходимо использовать все многообразие классификационных признаков.

Для организации поиска информации, для ведения тезаурусов (словарей) эффективно используется *дескрипторная* (описательная) система классификации, язык которой приближается к естественному языку описания информационных объектов. Особенно широко она используется в библиотечной системе поиска.

Системы классификации принципиально отличаются от систем кодирования в соответствии с определением.

Система кодирования – совокупность правил кодового обозначения объектов. Код строится на базе алфавита, состоящего из букв, цифр и других символов. Код характеризуется: длиной – число позиций в коде, и структурой – порядок расположения в коде символов, используемых для обозначения классификационного признака.

Кодирование применяется для замены названия объекта на условное обозначение (код) в целях обеспечения удобной и более эффективной обработки информации.

Лингвистическое обеспечение – совокупность средств и правил для формализации естественного языка, используемых при общении пользователей и эксплуатационного персонала АИС с комплексом средств автоматизации при функционировании АИС.

Языковые средства лингвистического обеспечения делятся на две группы: традиционные языки (естественные, математические, алгоритмические, моделирования) и языки, предназначенные для диалога с ЭВМ.

Математическое обеспечение – совокупность математических методов, моделей и алгоритмов, применяемых в АИС.

В состав математического обеспечения входят:

– средства математического обеспечения (средства моделирования типовых задач управления, методы многокритериальной оптимизации, математической статистики, теории массового обслуживания и др.);

– техническая документация (описание задач, алгоритмы решения задач, экономико-математические модели);

– методы выбора математического обеспечения (методы определения типов задач, методы оценки вычислительной сложности алгоритмов, методы оценки достоверности результатов).

Методическое обеспечение – совокупность документов, описывающих технологию функционирования АИС, методы выбора и применения пользователями технологических приемов для получения конкретных результатов при функционировании АИС.

Организационное обеспечение – совокупность документов, устанавливающих организационную структуру, права и обязанности пользователей и эксплуатационного персонала АИС в условиях функционирования, проверки и обеспечения работоспособности АИС.

Организационное обеспечение реализует следующие функции:

– анализ существующей системы управления предприятием (организа-цией), где используется АИС, выявление задач, подлежащих автоматизации;

– подготовка задач к автоматизации, включая подготовку технических заданий и технико-экономических обоснований эффективности;

– разработку управленческих решений по изменению структуры организации и методологий решения задач, направленных на повышение эффективности системы управления.

Организационное обеспечение включает:

– методические материалы, регламентирующие процесс создания и функционирования АИС;

– совокупность средств для эффективного проектирования и функционирования АИС;

– техническую документацию, получаемую в процессе обследования предприятия, проектирования, внедрения и сопровождения системы;

– персонал (организационно-штатные структуры предприятия), проектирующий, внедряющий, сопровождающий и использующий ИС.

Правовое обеспечение – совокупность правовых норм, регламентирующих правовые отношения при функционировании АИС и юридический статус результатов ее функционирования. Правовое обеспечение реализуется в организационном обеспечении АИС.

Программное обеспечение – совокупность программ на носителях данных и программных документов, предназначенных для отладки, функционирования и проверки работоспособности АИС.

Техническое обеспечение – совокупность всех технических средств, используемых при функционировании АИС.

Выбор технических средств, организация их эксплуатации, технологический процесс обработки данных, технологическое оснащение документально оформляются.

Эргономическое обеспечение – совокупность реализованных решений в АИС по согласованию психологических, психофизиологических, антропометрических, физиологических характеристик и возможностей пользователей АИС с техническими характеристиками комплекса средств автоматизации АИС и параметрами рабочей среды на рабочих местах пересонала АИС.

1.5. Жизненный цикл ИС

1.5.1. Основные понятия

Одним из базовых понятий методологии проектирования АИС является понятие жизненного цикла ее программного обеспечения (ЖЦ ПО). ЖЦ ПО – это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации.

В качестве определяющего документа для создания и испытания АИС целесообразно рассматривать международный стандарт ISO/IEC 12207 (ISO, International Organizational of Standartizational – Международная

организация по стандартизации; IEC – International Electrotechnical Commission – Международная комиссия по электротехнике). Данный стандарт в структуре жизненного цикла определяет процессы, которые выполняются при создании ПО АИС. Эти процессы подразделяют на три группы.

Разработка. Данный процесс включает в себя все основные работы по созданию информационной системы. Сюда входят: предварительный анализ, разработка архитектуры ИС, программирование, проектирование баз данных, интеграция отдельных модулей в единую систему, подготовка эксплуатационной документации, разработка тестов, разработка методических материалов, необходимых для обучения персонала, и другие виды работ.

Эксплуатация. Процесс предполагает обучение персонала, эксплуатационное тестирование, непосредственно эксплуатацию, локализацию и устранение проблем, модификацию программного обеспечения, подготовку предложений по модернизации системы, развитие системы, поддержку пользователей (оказание помощи).

Сопровождение. Процесс предполагает следующие виды работ: определение задач технического обслуживания и выделение задач, решаемых специализированными сервисными центрами, подготовка плана технического обслуживания, проведение анализа ресурсов, необходимых для организации обслуживания. Процесс активизируется при изменениях в программном продукте и соответствующей документации.

К вспомогательным относятся управление конфигурацией и документирование:

Управление конфигурацией. Процесс позволяет организовывать, систематически учитывать и контролировать внесение изменений в различные компоненты АИС на всех стадиях ее жизненного цикла (ЖЦ), для больших информационных систем, состоящих из множества модулей, которые могут разрабатываться независимо, и имеющих несколько версий или вариантов реализации, управление конфигурацией является важнейшим процессом.

Документирование. Процесс предусматривает формализованное описание информации, созданной в течении жизненного цикла информационной системы.

К организационным относится процесс управление проектами.

Данный процесс связан с вопросами планирования и организации работ, создания коллективов разработчиков, контроля сроков и качества выполнения работ. Техническое и организационное обеспечение проекта включает:

– выбор методов и инструментальных средств реализации проекта;

– определение методов описания состояния процесса разработки;

– разработку методов и средств испытаний созданного программного обеспечения;

– обучение персонала.

Обеспечение качества проекта связано с проблемами верификации и проверки компонентов АИС.

Верификация – процесс определения соответствия текущего состояния разработки, достигнутого на данном этапе, требованиям этого этапа.

Проверка – процесс определения соответствия параметров разработки исходным требованиям.

В 2002 г. был опубликован стандарт на процессы ЖЦ автоматизированных систем (ISO/IEC 15288 System life cycle processes). Согласно данному стандарту, в структуру ЖЦ включены следующие группы процессов:

договорные процессы:

– приобретение (внутренние решения или решения внешнего поставщика);
– поставка (внутренние решения или решения внешнего поставщика);

процессы предприятия:

– управление окружающей средой предприятия;
– инвестиционное управление;
– управление ЖЦ ИС;
– управление ресурсами;
– управление качеством;

проектные процессы:

– планирование проекта;
– оценка проекта;
– контроль проекта;
– управление рисками;
– управление конфигурацией;
– управление информационными потоками;
– принятие решений;

технические процессы:

– определение требований;
– анализ требований;
– разработка архитектуры;
– внедрение;
– интеграция;
– верификация;
– переход;
– аттестация;
– эксплуатация;
– сопровождение;
– утилизация;

специальные процессы:

– определение и установка взаимосвязей исходя из задач и целей.

1.5.2. Модели жизненного цикла информационных систем

Рассмотренные выше процессы характеризуются определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе результатами. Результатами анализа, в частности, являются функциональные модели, информационные модели и соответствующие им диаграммы. При этом ЖЦ носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах. Известные модели ЖЦ ПО (каскадная, итерационная, спиральная) определяют порядок исполнения этапов в ходе разработки, а также критерии перехода от этапу к этапу.

По аналогии с известным определением модели ЖЦ ПО и в соответствии с устоявшейся среди специалистов терминологий приведем определение модели ЖЦ АИС.

Модель жизненного цикла АИС – это структура, описывающая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения в течение всего жизненного цикла системы.

Модель ЖЦ АИС отражает состояние системы с момента осознания необходимости создания данной АИС до полной ее утилизации. Выбор модели жизненного цикла зависит от специфики, масштаба, сложности проекта и набора условий, в которых АИС создается и функционирует. Модель ЖЦ АИС включает:

– стадии;

– результаты выполнения работ на каждой стадии;

– ключевые события или точки завершения работ и принятия решений.

В соответствии с известными моделями ЖЦ ПО определяют следующие модели ЖЦ АИС: каскадную, итерационную спиральную.

Каскадная модель предусматривает последовательную организацию работ, причем основной особенностью модели является разбиение всей работы на этапы. Выделяют пять устойчивых этапов разработки (рис. 1.1).

На первом этапе проводится исследование проблемной области, формулируются требования заказчика. Результатом данного этапа является техническое задание (ТЗ), согласованное со всеми заинтересованными сторонами.

В ходе второго этапа, согласно требованиям ТЗ, разрабатываются те или иные проектные решения. В результате появляется комплект проектной документации.

Третий этап – реализация проекта. Методы реализации при этом принципиального значения не имеют. Результатом выполнения этапа является готовый программный продукт.

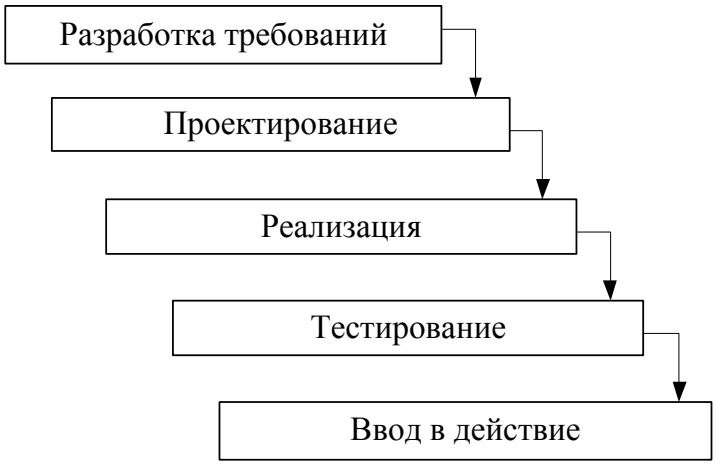

Рис. 1.1. Каскадная модель ЖЦ АИС

На четвертом этапе проводится проверка полученного программного обеспечения на предмет соответствия требованиям, заявленным в ТЗ.

Последний этап – сдача готового проекта.

Достоинства каскадной модели:

– на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности. На заключительных этапах разрабатывается пользовательская документация, охватывающая все предусмотренные стандартами виды обеспечения АИС;

– последовательное выполнение этапов работ позволяет планировать сроки завершения и соответствующие затраты.

Недостатки каскадной модели:

– существенная задержка в получении результатов;

– ошибки и недоработки на любом из этапов проявляются, как правило, на последующих этапах работ, что приводит к необходимости возврата;

– сложность параллельного ведения работ по проекту;

– чрезмерная информационная перенасыщенность каждого из этапов;

– сложность управления проектом;

– высокий уровень риска и ненадежность инвестиций.

Построение ***итерационной модели*** заключается в серии коротких циклов (шагов) по планированию, реализации, изучению, действию.

Создание сложных АИС предполагает проведение согласований проектных решений, полученных при реализации отдельных задач. Подход к проектированию «снизу-вверх» обуславливает необходимость таких итераций возвратов, когда проектные решения по отдельным задачам объединяются в общие системные решения. При этом возникает потребность в пересмотре ранее сформировавшихся требований.

Достоинством итерационной модели являются межэтапные корректировки, которые обеспечивают меньшую трудоемкость разработки по сравнению с каскадной моделью.

Недостатками итерационной модели являются:

– время жизни каждого этапа растягивается на весь период разработки;

– вследствие большого числа итераций возникают рассогласования выполнения проектных решений и документации;

– запутанность архитектуры;

– трудности использования проектной документации на стадиях внедрения и эксплуатации вызывают необходимость перепроектирования всей системы.

Спиральная модель в отличие от каскадной, но аналогично предыдущей предполагает итерационный процесс разработки АИС. При этом возрастает значение начальных этапов, таких как анализ и проектирование, на которых проверяется и обосновывается реализуемость технических решений путем создания прототипов.

Прототип – модель информационной системы, призванная продемонстрировать пользователю некоторые возможности будущего продукта, и позволяющая на ранних стадиях проектирования подключить заказчика к разработке.

Каждая итерация представляет собой законченный цикл разработки, приводящий к выпуску внутренней и внешней версии изделия (или подмножества конечного продукта), которое совершенствуется от итерации к итерации, чтобы стать законченной системой (рис. 1.2).

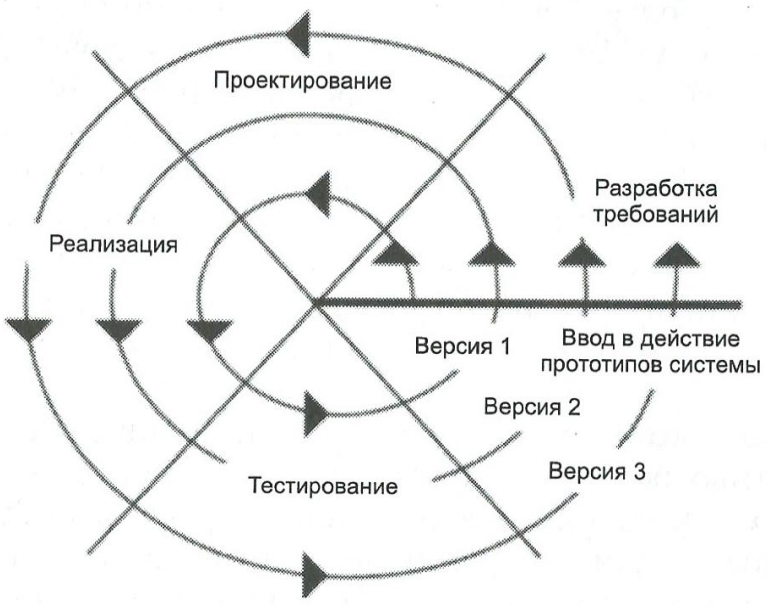

Рис. 1.2. Спиральная модель ЖЦ АИС

Каждый виток спирали соответствует созданию фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы на следующем витке спирали. Каждая итерация служит для углубления и последовательной конкретизации деталей проекта, в результате этого выбирается обоснованный вариант окончательной реализации. Главная задача каждой итерации – как можно быстрее создать работоспособный продукт для демонстрации пользователям. Таким образом, существенно упрощается процесс внесения уточнений и дополнений в проект.

Основная проблема спирального цикла – трудность определения момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла.

2. МЕТОДОЛОГИЯ И ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

2.1. Основные понятия

Технология проектирования АИС – это совокупность методов и средств проектирования АИС, а также методов и средств организации проектирования (управление процессом создания и модернизации проекта АИС). В основе технологий проектирования лежит технологический процесс (ТП), который определяет действия, их последовательность, состав, исполнителей, средства и ресурсы, требуемые для выполнения этих действий.

Технология проектирования АИС представляет собой совокупность последовательно-параллельных, связанных и соподчиненных цепочек действий, каждое из которых может иметь свой предмет. Действия, которые выполняются при проектировании АИС, могут быть определены как неделимые технологические операции или как подпроцессы технологических операций. Все действия могут быть собственно проектировочными, которые формируют или модифицируют результаты проектирования, и оценочными, которые вырабатывают по установленным критериям оценки результатов проектирования.

Таким образом, технология проектирования задается регламентированной последовательностью технологических операций, выполняемых в процессе создания проекта на основе того или иного метода.

Предметом выбираемой технологии проектирования должно служить отражение взаимосвязанных процессов проектирования на всех стадиях жизненного цикла АИС.

Требования, предъявляемые к технологии проектирования:

– созданный с помощью этой технологии проект должен отвечать требованиям заказчика;

– технология должна максимально отражать все этапы цикла жизни проекта;

– технология должна обеспечивать минимальные трудовые и стоимостные затраты на проектирование и сопровождение проекта;

– технология должна способствовать росту производительности труда проектировщиков;

– технология должна обеспечивать надежность процесса проектирования и эксплуатации проекта;

– технология должна способствовать простому ведению проектной документации.

Технология проектирования АИС реализует определенную методологию проектирования. В свою очередь, методология проектирования предполагает наличие некоторой концепции, принципов проектирования и реализуется набором методов и средств.

2.2. Классификация методов проектирования информационных систем

Методы проектирования АИС можно классифицировать по степени использования средств автоматизации, типовых проектных решений, адаптивности к предполагаемым изменениям.

По степени автоматизации:

– ручное проектирование, при котором проектирование компонентов АИС осуществляется без использования специальных инструментальных программных средств; программирование производится на алгоритмических языках;

– компьютерное проектирование, при котором генерация или конфигурация (настройка) проектных решений производится с использованием специальных инструментальных программных средств.

По степени использования типовых проектных решений:

– оригинальное (индивидуальное) проектирование, когда проектные решения разрабатываются с нуля в соответствии с требованиями к АИС; предполагает максимальный учет особенностей автоматизированного объекта;

– типовое проектирование, предполагающее конфигурацию АИС из готовых типовых проектных решений (программных модулей); выполняется на основе готовых решений и является обобщением опыта, полученного ранее при создании родственных проектов.

По степени адаптивности проектных решений:

– реконструкция – адаптация проектных решений выполняется путем переработки соответствующих компонентов (перепрограммирования программных модулей);

– параметризация – проектные решения настраиваются в соответствии с заданными и изменяемыми параметрами;

– реструктуризация модели – изменяется модель предметной области, что приводит к автоматическому переформированию проектных решений.

Сочетание различных признаков классификации методов проектирования обуславливает характер используемой технологии проектирования АИС.

2.3. Технологии проектирования информационных систем

2.3.1. Классификация технологии проектирования информационных систем

Выделяются два основных класса технологии проектирования: каноническая и индустриальная (табл. 2.1). Индустриальная технология проектирования разбивается на два подкласса: автоматизированное (использование CASE-технологий) и типовое (параметрически-ориентированное) или модельно-ориентированное проектирование. Использование индустриальных технологий проектирования не исключает использования в отдельных случаях канонической технологии.

Таблица 2.1

Характеристики классов технологий проектирования

Класс технологии проектирования	**Степень автоматизации**	**Степень типизации**	**Степень адаптивности**
Каноническое проектирование	Ручное проектирование	Оригинальное проектирование	Реконструкция
Индустриальное автоматизированное проектирование	Компьютерное проектирование	То же	Реструктуризация модели
Индустриальное типовое проектирование	То же	Типовое сборочное проектирование	Параметризация и реструктуризация модели

2.3.2. Каноническое проектирование информационных систем

Каноническое проектирование ИС отражает особенности ручной технологии индивидуального (оригинального) проектирования, осуществляемого на уровне исполнителей без использования каких-либо инструментальных средств, позволяющих интегрировать выполнение элементарных операций. Как правило, каноническое проектирование применяется для небольших локальных ИС. В основе канонического проектирования лежит каскадная модель жизненного цикла ИС. Стадии и этапы такого проектирования описаны в ГОСТ 34.601–90.

В зависимости от сложности объекта автоматизации и набора задач, требующих решения при создании конкретной АИС, стадии и этапы работ

могут иметь различную трудоемкость. Допускается объединять последовательные этапы и исключать некоторые из них на любой стадии проекта. Допускается также начинать выполнение работ следующей стадии до окончания предыдущей.

Стадии и этапы создания АИС, выполняемые организациями-участниками, прописываются в договорах и технических заданиях на выполнение работ.

Стадии создания АИС:

1) формирование требований к АИС;
2) разработка концепции АИС;
3) техническое задание;
4) эскизный проект (не является обязательным);
5) технический проект;
6) рабочая документация;
7) ввод в действие;
8) сопровождение АИС.

Рассмотрим специфику составляющих некоторых стадий подробнее.

Обследование – это изучение и анализ организационной структуры предприятия, его деятельности и существующей системы обработки информации.

На этапе обследования целесообразно выделить две составляющие: *определение стратегии внедрения* АИС и *детальный анализ деятельности организации*.

Результатом этапа *определения стратегии* является документ (технико-экономическое обоснование – ТЭО – проекта).

На этапе детального анализа деятельности организации изучается деятельность, обеспечивающая реализацию функций управления, организационная структура, штаты и содержание работ по управлению предприятием, а также характер подчиненности вышестоящим органам управления. Здесь следует наметить инструктивно-методические и директивные материалы, на основании которых определяются состав подсистем и перечень задач, а также возможности применения новых методов решения задач.

Аналитики собирают и фиксируют информацию в двух взаимосвязанных формах:

– функции – информация о событиях и процессах, которые происходят в автоматизированной организации;

– сущности – информация о классах объектов, имеющих значение для организации и о которых собираются данные.

Еще одной задачей данного этапа является *описание документооборота организации*.

При обследовании документооборота составляется схема маршрута движения документа, которая должна отразить: количество документов,

место формирования показателей документа, взаимосвязь документов при их формировании, внутренние и внешние информационные связи, место использования и хранения документа, объем документа в знаках.

По результатам обследования устанавливают перечень задач управления, подлежащих автоматизации, и очередность их разработки. Также по завершении стадии обследования появляется возможность определить вероятные технические подходы к созданию системы и оценить затраты на ее реализацию (на аппаратное обеспечение, на закупаемое программное обеспечение, и на разработку нового программного обеспечения).

Результаты обследования представляют объективную основу для формирования технического задания на АИС.

Техническое задание (ТЗ) – это документ, определяющий цели, требования и основные исходные данные, необходимые для разработки автоматизированной системы управления.

Эскизный проект предусматривает разработку предварительных проектных решений по системе и ее частям. В соответствии с ГОСТ 19.102–77 стадия эскизного проектирования содержит два этапа: разработку эскизного проекта и утверждение эскизного проекта.

На основании ТЗ и эскизного проекта разрабатывается технический проект АИС.

Технический проект системы – это техническая документация, содержащая общесистемные проектные решения, алгоритмы решения задач, а также оценку экономической эффективности АИС и перечень мероприятий по подготовке объекта к внедрению.

На этом этапе осуществляется комплекс научно-исследовательских и экспериментальных работ для выбора основных проектных решений и расчет экономической эффективности системы.

На стадии «Рабочая документация» осуществляется создание программного продукта и разработка всей сопровождающей документации.

На стадии «Ввод в действие» для АИС устанавливают следующие основные виды испытаний: *предварительные испытания, опытная эксплуатация* и *приемочные испытания.*

В зависимости от взаимосвязей компонентов АИС и объекта автоматизации испытания могут быть *автономные* и *комплексные***.**

В *автономных* испытаниях участвуют компоненты системы. Их проводят по мере готовности частей системы к сдаче в опытную эксплуатацию. *Комплексные* испытания проводят для групп взаимосвязанных компонентов (подсистем) или для системы в целом.

Предварительные испытания проводят для определения работоспособности системы и решения вопроса о возможности ее приемки в опытную эксплуатацию.

Опытную эксплуатацию системы проводят с целью определения фактических значений количественных и качественных характеристик системы и готовности персонала к работе в условиях ее функционирования, а также определения фактической эффективности и корректировки, при необходимости, документации.

Приемочные испытания проводят для определения соответствия системы техническому заданию, оценки качества опытной эксплуатации и решения вопроса о возможности приемки системы в постоянную эксплуатацию.

2.3.3. Типовое проектирование информационных систем

Типовое проектирование АИС предполагает создание системы из готовых типовых элементов. При этом основным требованием метода является возможность декомпозиции проектируемой АИС на множество составляющих компонентов (подсистем, комплексов задач, программных модулей и т. д.). Для реализации выделенных компонентов выбираются имеющиеся на рынке типовые проектные решения, которые настраиваются на особенности конкретного предприятия.

Типовое проектное решение (ТПР) – это тиражируемое (пригодное к многократному использованию) проектное решение.

ТПР классифицируются по уровням декомпозиции системы.

Приняты следующие классы:

– элементные ТПР. Типовое решение задачи или отдельного вида обеспечения задачи (информационного, программного, технического, технологического, математического, организационного);

– подсистемные ТПР. Решение является отдельной функционально полной подсистемой;

– объектные ТПР. Типовой проект, включающий полный набор функциональных и обеспечивающих подсистем АИС (для вида деятельности, отрасли и т. д.).

ТПР должно содержать не только функциональные элементы (программные или аппаратные), но и документацию с детальным описанием состава компонентов и процедуры настройки в соответствии с задачами проекта.

Для реализации типового проектирования могут использоваться два подхода: *параметрически-ориентированное* и *модельно-ориентированное проектирование*.

Параметрически-ориентированное проектирование включает следующие этапы:

– определение критериев оценки пригодности пакетов прикладных программ (ППП) для решения поставленных задач;

– анализ и оценка доступных ППП по сформулированным критериям;

– выбор и закупка наиболее подходящих пакетов;
– настройка параметров (доработка закупленных) ППП.

Критерии оценки ППП делятся на следующие группы:

– назначение и возможности пакета;
– характеристики и свойства пакета;
– требования к аппаратным и программным средствам;
– документация пакета;
– финансовые факторы;
– особенности установки и настройки пакета;
– особенности эксплуатации пакета;
– обязательства поставщика по внедрению и сопровождению пакета;
– оценка качества пакета и опыт его использования;
– перспективы развития пакета.

Внутри каждой группы критериев выделяется некоторое подмножество частных показателей, детализирующих каждый из приведенных аспектов анализа выбираемых ППП. Числовые значения показателей для конкретных ППП устанавливаются экспертами по выбранной шкале оценок. На их основе формируются групповые оценки и комплексная оценка пакета (путем вычисления средневзвешенных значений). Нормированные взвешивающие коэффициенты также получаются экспертным путем.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой АИС и автоматизируемого объекта (предприятия).

На рис. 2.1 представлена конфигурация АИС, проектируемая на основе модельно-ориентированной технологии.

Репозиторий – специальная база метаданных – содержит модель объекта автоматизации, на основе которой осуществляется конфигурирование программного обеспечения. Репозиторий содержит базовую (ссылочную) модель АИС, типовые (референтные) модели определенных классов АИС, модели конкрентных АИС предприятий.

Базовая модель АИС содержит описание бизнес-функций, бизнес-процессов, бизнес-правил, организационной структуры, которые поддерживаются программными модулями типовой АИС.

Типовые модели описывают конфигурации АИС для определенных отраслей или типов производства.

Модель конкретного предприятия строится либо путем выбора фрагментов основной или типовой модели в соответствии со специфическими особенностями предприятия, либо путем автоматизированной адаптации этих моделей в результате экспертного опроса.

Построенная модель предприятия в виде метаописания хранится в репозитории и при необходимости может быть откорректирована. На основе

этой модели автоматически осуществляются конфигурирование и настройка АИС.

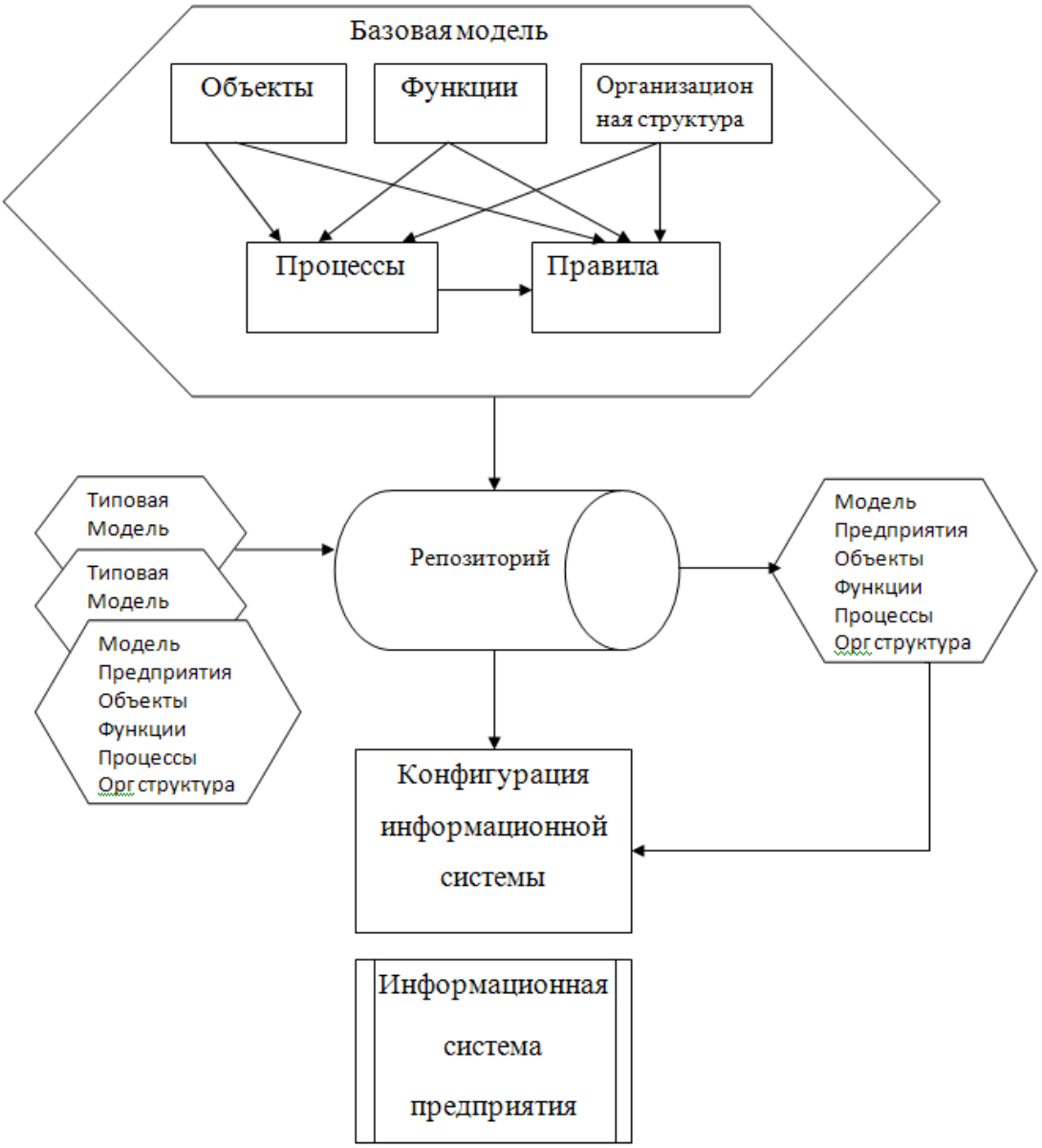

Рис. 2.1. Конфигурация АИС на основе модельно-ориентированной технологии

Бизнес-правила определяют условия корректности совместного применения различных компонентов АИС и используются для поддержания целостности создаваемой системы.

Модель бизнес-функций представляет собой иерархическую декомпозицию функциональной деятельности предприятия.

Модель бизнес-процессов отражает выполнение работ для функций самого нижнего уровня модели бизнес-функций. Модель бизнес-процессов позволяет выполнить настройку программных модулей – приложений АИС в соответствии с характерными особенностями конкретного предприятия.

Модель бизнес-объектов используется для интеграции приложений, поддерживающих исполнение различных бизнес-процессов.

Модель организационной структуры предприятия представляет собой традиционную иерархическую структуру подчинения подразделений и персонала.

Внедрение типовой АИС начинается с анализа требований, которые выявляются на основе результатов предпроектного обследования объекта автоматизации. Для оценки соответствия этим требованиям программных продуктов может использоваться описанная выше методика оценки ППП. После выбора программного продукта на базе имеющихся в нем референтных моделей строится предварительная модель АИС, в которой отражаются все особенности реализации АИС для конкретного предприятия. Предварительная модель является основой для выбора типовой модели системы и определения перечня компонентов, которые будут реализованы с использованием других программных средств или потребуют разработки с помощью имеющихся в составе типовой АИС инструментальных средств.

Реализация типового проекта предусматривает выполнение следующих операций:

- установку глобальных параметров системы;
- задание структуры объекта автоматизации;
- определение структуры основных данных;
- задание перечня реализуемых функций и процессов;
- описание интерфейсов;
- описание отчетов;
- настройку автоматизации доступа;
- настройку системы архивирования.

2.4. Средства проектирования информационных систем

Для конкретных видов технологий проектирования свойственно применение определенных средств разработки ИС, которые поддерживают выполнение как отдельных проектных работ, этапов, так и их совокупностей. Поэтому перед разработчиками ИС, как правило, стоит задача выбора средств проектирования, которые по своим характеристикам в наибольшей степени соответствуют требованиям конкретного предприятия.

Средства проектирования должны быть:

- инвариантны к объекту проектирования (в своем классе);
- способны охватить в совокупности все этапы жизненного цикла ИС;
- технически, программно и информационно совместимыми;
- просты в освоении и применении;
- экономически целесообразны.

Средства проектирования ИС можно разделить на два класса: *без использования ЭВМ* и *с использованием ЭВМ*.

Средства проектирования *без использования ЭВМ* применяются на всех стадиях и этапах. Как правило, это средства организационно-методического обеспечения операций и в первую очередь различные стандарты, регламентирующие процесс проектирования систем. Сюда же относятся единая система классификации и кодирования информации, унифицированная система документации, модели описания и анализа потоков информации и т. п.

Средства проектирования с использованием ЭВМ могут применяться как на отдельных, так и на всех стадиях и этапах процесса проектирования ИС и соответственно поддерживают разработку элементов, разделов, проекта системы в целом. Все множество средств проектирования с использованием ЭВМ делят на четыре подкласса.

1. Операционные средства, которые поддерживают проектирование операций обработки информации. К данному подклассу средств относятся алгоритмические языки, библиотеки стандартных подпрограмм и классов объектов, макрогенераторы, генераторы программ типовых операций обработки данных и т. п., а также средства расширения функций операционных систем (утилиты). В данный класс включаются также такие простейшие инструментальные средства проектирования, как средства для тестирования и отладки программ, поддержки процесса документирования проекта и т. п. Особенность последних программ заключается в том, что с их помощью повышается производительность труда проектировщиков, но не разрабатывается законченное проектное решение. Таким образом, средства данного подкласса поддерживают отдельные операции проектирования ИС и могут применяться независимо друг от друга.

2. Средства, поддерживающие проектирование отдельных компонентов. К данному подклассу относятся средства общесистемного назначения:

– системы управления базами данных (СУБД);

– методо-ориентированные пакеты прикладных программ (решение задач дискретного программирования, математической статистики и т. п.);

– табличные процессоры;

– статистические ППП;

– оболочки экспертных систем;

– графические редакторы;

– текстовые редакторы;

– интегрированные ППП (интерактивная среда с встроенными диалоговыми возможностями, позволяющая интегрировать вышеперечисленные программные средства).

Для перечисленных средств характерно их использование для разработки технологических подсистем ИС: ввода информации, организации

хранения и доступа к данным, вычислений, анализа и отображения данных, принятия решений.

4. Средства, поддерживающие проектирование разделов проекта. В этом подклассе выделяют функциональные средства проектирования. Функциональные средства направлены на разработку автоматизированных систем, реализующих функции, комплексы задач и задачи управления. К функциональным средствам проектирования систем обработки информации относятся типовые проектные решения, функциональные пакеты прикладных программ, типовые проекты.

5. Средства, поддерживающие разработку на стадиях и этапах процесса проектирования. К данному классу относятся средства автоматизации проектирования (CASE-средства). Современные CASE-средства, в свою очередь, классифицируются по двум признакам:

– по охватываемым этапам процесса разработки ИС;
– по степени интегрированности:
  - отдельные локальные средства (tools);
  - набор неинтегрированных средств, охватывающих большинство этапов разработки ИС (toolkit);
  - полностью интегрированные средства, связанные общей базой проектных данных – репозиторием (workbench).

3. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ ИНФОРМАЦИОННЫХ СИСТЕМ

3.1. Этапы анализа предметной области

Проект разработки АИС для любой предметной области и любого предприятия следует рассматривать как крупные инвестиции, которые должны окупиться за счет повышения эффективности, поэтому сначала необходимо определить, какие именно функциональные области и какие типы производства нужно охватить, т. е. провести анализ предметной области АИС. Для эффективного анализа предметной области необходимо следующее.

1. Разработать стратегию комплексной автоматизации.

Понятие стратегии автоматизации основывается на базовых принципах автоматизации предприятия, которая включает в себя следующие компоненты:

– цели – области деятельности предприятия и последовательность в которой они будут автоматизированы;

– способ автоматизации – по участкам, направлениям, комплексная автоматизация;

– долгосрочная техническая политика – комплекс внутренних стандартов, поддерживаемых на предприятии;

– ограничения;

– процедура управления изменениями плана.

Во главе стратегии автоматизации должна лежать стратегия бизнеса предприятия: миссия предприятия, направления и модель бизнеса.

К основным ***ограничениям***, которые необходимо учитывать при выборе стратегии автоматизации, относятся:

– *финансовые ограничения* – определяются величиной инвестиций, которые предприятие способно вложить в развитие автоматизации;

– *временные ограничения* – связаны со сменой технологий основного производства, рыночной стратегией предприятия, государственным регулированием экономики;

– *технические ограничения* – определяются степенью материального и морального износа технического парка предприятия (организации);

– ограничения, связанные с влиянием *человеческого фактора* – корпоративная культура или отношение персонала к автоматизации и особенности рынка труда в части трудового законодательства.

2. Провести анализ деятельности предприятия.

Под анализом деятельности предприятия понимается сбор и представление информации о деятельности предприятия в формализованном виде, пригодном для принятия решения о разработке определенного класса АИС.

3. Рассмотреть вопросы реорганизации деятельности.

Реорганизация деятельности преследует, как правило, цель повышения эффективности деятельности предприятия в целом и может предусматривать применение следующих методологий: BSP, TQM или BPR.

Методология BSP определяется как «подход, помогающий предприятию определить план создания информационных систем, удовлетворяющих его ближайшим и перспективным информационным потребностям».

Информация является одним из основных ресурсов и должна планироваться в масштабах всего предприятия, АИС должна проектироваться независимо от текущего состояния и структуры предприятия.

Подход TQM (Total Quality Management)

В основе подхода лежит очевидная концепция управления качеством выпускаемой продукции. Качество должно быть направлено на удовлетворение текущих и будущих потребностей потребителя как самого важного звена производственной линии.

Достижение соответствующего уровня качества требует постоянного совершенствования производственных процессов. Для решения этой задачи Э. Демингом было предложено 14 принципов, в совокупности составляющих теорию управления качеством и применимых для предприятий произвольных типов и различных масштабов. Безусловно, этих принципов недо-

статочно для полного решения стоящих перед современными предприятиями проблем.

BPR (Business Process Reengineering) – реинжиниринг бизнес-процессов – определяется как фундаментальное переосмысление и радикальное перепланирование бизнес-процессов компаний.

3.2. Реинжиниринг бизнес-процессов

Бизнес-процесс – совокупность взаимосвязанных операций (работ) по изготовлению готовой продукции или выполнению услуг на основе потребления ресурсов.

Целью реинжиниринга бизнес-процессов является системная реорганизация материальных, финансовых и информационных потоков, направленная на упрощение организационной структуры, перераспределение и минимизацию использования различных ресурсов, сокращение сроков реализации потребностей клиентов, повышение качества их обслуживания.

При этом используются следующие положения:

– несколько работ объединяются в одну;

– исполнителям делегируется право по принятию решений;

– этапы процесса выполняются в естественном порядке;

– реализуются различные версии процесса;

– работа выполняется там, где ее целесообразно делать (выход работы за границы организационных структур);

– снижаются доли работ по проверке и контролю;

– минимизируется количество согласований;

– ответственный менеджер является единственной точкой контакта с клиентом процесса;

– используются и централизованные, и децентрализованные операции.

Реинжиниринг бизнес-процессов решает следующие задачи.

1. Определение оптимальной последовательности выполняемых функций, которое приводит к сокращению длительности цикла изготовления и продажи товаров и услуг, обслуживание клиентов, следствием чего служат повышение оборачиваемости капитала и рост всех экономических показателей фирмы.

2. Оптимизация использования ресурсов в различных бизнес-процессах, в результате которой минимизируются издержки и обеспечивается оптимальное сочетание различных видов деятельности.

3. Построение адаптивных бизнес-процессов, нацеленных на быструю адаптацию к изменениям потребностей конечных потребителей продукции, производственных технологий, поведение конкурентов на рынке и, следовательно, повышение качества обслуживания клиентов в условиях динамичности внешней среды.

4. Определение рациональных схем взаимодействия с партнерами и клиентами и, как следствие, рост прибыли, оптимизация финансовых потоков.

Реинжиниринг бизнес-процессов предполагает изменение архитектуры корпоративной экономической информационной системы, которая призвана:

– *на оперативном уровне* обеспечить ускорение информационных потоков, связывающих участников деловых процессов, и улучшить синхронизацию одновременно выполняемых деловых процессов;

– *на тактическом уровне* способствовать повышению качества принимаемых управленческих решений, позволяющих адаптировать деловые процессы к изменению внешней среды;

– *на стратегическом уровне* обеспечивать процесс принятия решений относительно проектирования новых и перепроектирование существующих бизнес-процессов.

Организационный анализ компании при таком подходе проводится по определенной схеме с помощью *полной бизнес-модели* компании.

На рис. 3.1 представлена обобщенная схема организационной бизнес-модели.

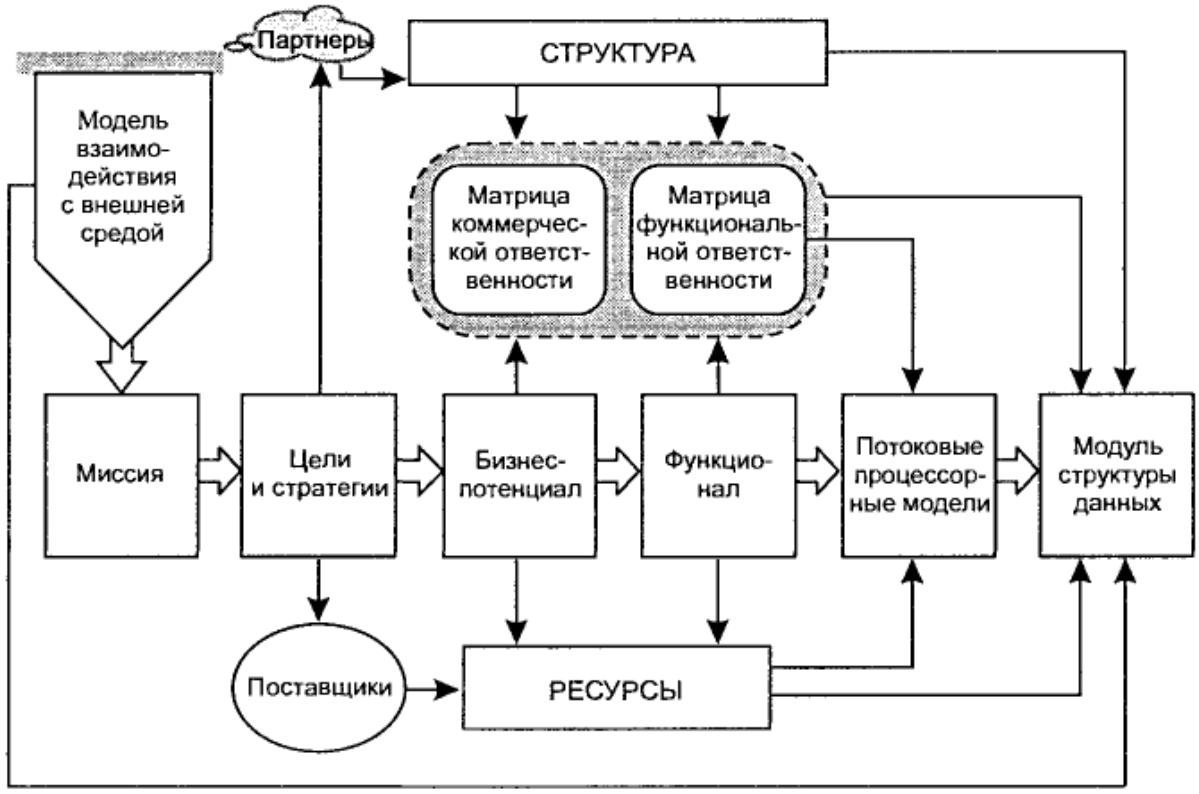

Рис. 3.1. Обобщенная схема организационного бизнес-моделирования

Построение бизнес-модели компании начинается с моделирования ее взаимодействия с внешней средой, т. е. с определения миссии компании.

Миссия – это:

1) деятельность, осуществляемая предприятием для того, чтобы выполнить функцию, для которой оно было учреждено, предоставление заказчикам продукта или услуги;

2) механизм, с помощью которого предприятие реализует свои цели и задачи.

Миссия компании по удовлетворению социально значимых потребностей рынка определяется как компромисс интересов рынка и компании.

При этом миссия как атрибут открытой системы разрабатывается, с одной стороны, исходя из рыночной конъюнктуры и позиционирования компании относительно других участников внешней среды, а с другой – исходя из объективных возможностей компании и ее субъективных ценностей, ожиданий и принципов.

Определение миссии позволяет сформировать *дерево целей* компании – иерархические списки уточнения и детализации миссии.

Дерево целей формирует *дерево стратегий* – иерархические списки уточнения и детализации целей. При этом на корпоративном уровне разрабатываются стратегии роста, интеграции и инвестиции бизнесов.

Блок *бизнес-стратегий* определяет продуктовые и конкурентные стратегии, а также стратегии сегментации и продвижения.

Ресурсные стратегии определяют стратегии привлечения материальных, финансовых, человеческих и информационных ресурсов.

Функциональные стратегии определяют стратегии в организации компонентов управления и этапов жизненного цикла продукции (это позволяет обеспечить заказчикам необходимый продукт требуемого качества, в нужном количестве, в нужном месте, в нужное время и по приемлемой цене).

Бизнес-потенциал определяет функционал компании – перечень бизнес-функций, функций менеджмента и функций обеспечения, требуемых для поддержания на регулярной основе указанных видов коммерческой деятельности.

Кроме того, уточняются необходимые для этого ресурсы (материальные, человеческие, информационные) и структура компании.

Построение бизнес-потенциала и функционала компании позволяет с помощью матрицы проекций определить зоны ответственности менеджмента.

Матрица проекций – модель, представленная в виде матрицы, задающей систему отношений между классификаторами в любой их комбинации.

Матрица коммерческой ответственности закрепляет ответственность структурных подразделений за получение дохода в компании от реализации коммерческой деятельности.

Ее дальнейшая детализация (путем выделения центров финансовой ответственности) обеспечивает построение финансовой модели компании, что позволяет внедрить систему бюджетного управления.

Матрица функциональной ответственности закрепляет ответственность структурных звеньев (и отдельных специалистов) за выполнение бизнес-функций при реализации процессов коммерческой деятельности (закупка, производство, сбыт и проч.), а также функций менеджмента, связанных с управлением этими процессами (планирование, учет, контроль в области маркетинга, финансов, управления персоналом и проч.).

Дальнейшая детализация матрицы (до уровня ответственности отдельных сотрудников) позволит получить функциональные обязанности персонала, что в совокупности с описанием прав, обязанностей, полномочий обеспечит разработку пакета должностных инструкций.

Описание бизнес-потенциала, функционала и соответствующих матриц ответственности представляет собой *статическое описание компании.* При этом процессы, протекающие в компании, идентифицируются, классифицируются и закрепляются за исполнителями (будущими хозяевами этих процессов). На этом этапе бизнес-моделирования формируется общепризнанный набор основополагающих внутрифирменных регламентов:

– базовое положение об организационно-функциональной структуре компании;

– пакет положений об отдельных видах деятельности (финансовой, маркетинговой и т. д.);

– пакет положений о структурных подразделениях (цехах, отделах, секторах, группах и т. п.);

– должностные инструкции.

Дальнейшее развитие (детализация) бизнес-модели происходит на этапе *динамического описания компании* на уровне *процессных потоковых моделей.*

Процессная потоковая модель – это модель, описывающая процесс последовательного во времени преобразования материальных и информационных потоков компании в ходе реализации какой-либо бизнес-функции или функции менеджмента (рис. 3.2).

Сначала (на верхнем уровне) описывается логика взаимодействия участников процесса, а затем (на нижнем уровне) – технология работы отдельных специалистов на своих рабочих местах.

Организационное бизнес-моделирование завершается разработкой ***модели структур данных,*** которая определяет перечень и форматы документов, сопровождающих процессы в компании, а также задает форматы описания объектов внешней среды, компонентов и регламентов самой компании. При этом создается система справочников, на основании которых получают пакеты необходимых документов и отчетов.

Такой подход позволяет описать деятельность компании с помощью универсального множества управленческих регистров (цели, стратегии, продукты, функции, организационные звенья и т. д.).

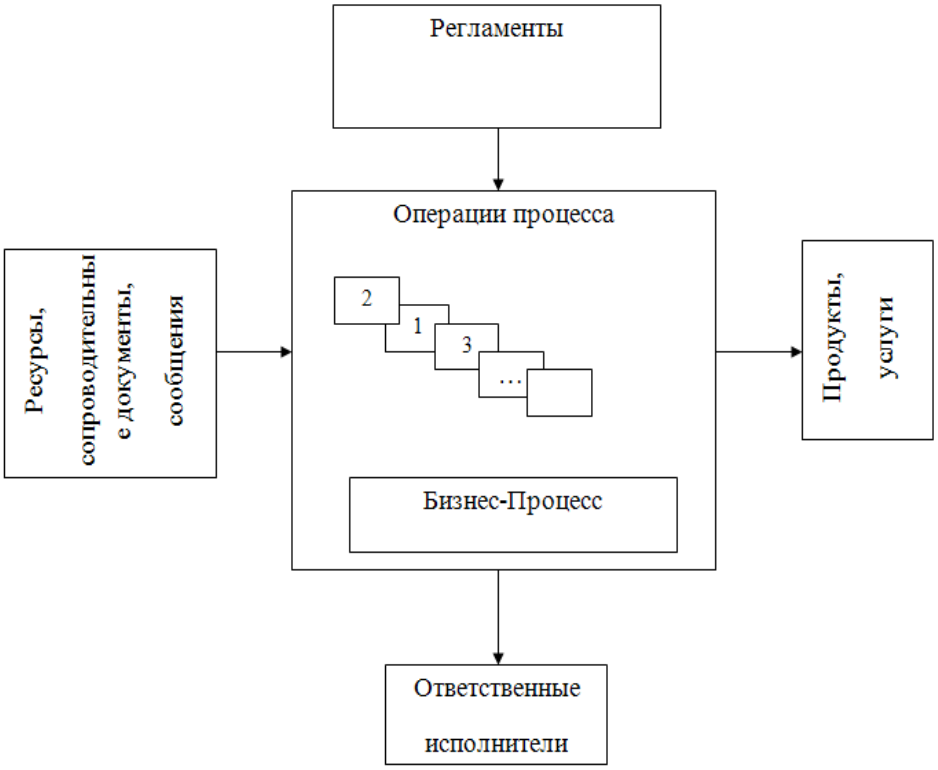

Рис. 3.2. Потоковая процессная модель

Полная бизнес-модель компании – это совокупность функционально ориентированных информационных моделей, обеспечивающая взаимосвязанные ответы на перечисленные выше вопросы (рис. 3.3).

Таким образом, организационный анализ предполагает построение комплекса взаимосвязанных информационных моделей компании, который включает:

– стратегическую модель целеполагания (отвечает на вопросы: зачем компания занимается именно этим бизнесом, почему предполагает быть конкурентоспособной, какие цели и стратегии для этого необходимо реализовать?);

– организационно-функциональную модель (отвечает на вопрос: кто что делает в компании и кто за что отвечает?);

– функционально-технологическую модель (отвечает на вопрос: что как реализуется в компании?);

– процессно-ролевую модель (отвечает на вопрос: кто – что – как – кому?);

– количественную модель (отвечает на вопрос: сколько необходимо ресурсов?);

– модель структуры данных (отвечает на вопрос: в каком виде описываются регламенты компании и объекты внешнего окружения?).

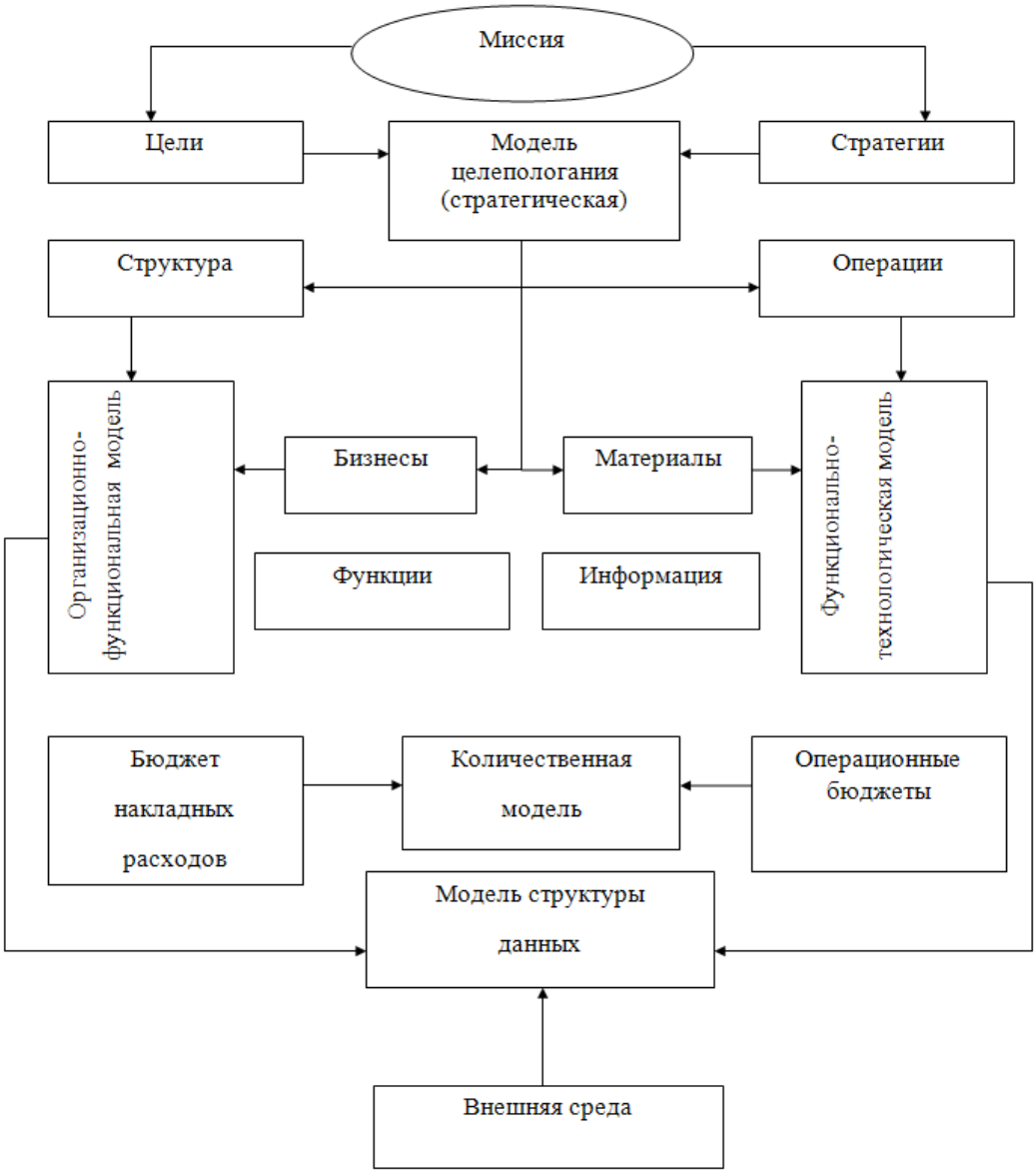

Рис. 3.3. Полная бизнес-модель компании

Представленная совокупность моделей обеспечивает необходимую полноту и точность описания компании и позволяет вырабатывать понятные требования к проектируемой информационной системе.

3.3. Методы сборов материалов исследования

Для построения рассмотренных выше моделей необходимо получить и проанализировать соответствующую информацию.

Рассмотрим разнообразные методы сбора данных (материалов) обследования.

Метод бесед и консультаций с руководителями чаще всего проводится в форме обычной беседы с руководителями предприятий и подразделе-

ний или в форме деловой консультации со специалистами по вопросам, носящим глобальный характер и относящимся к определению проблем и стратегий развития и управления предприятием.

Метод опроса исполнителей на рабочих местах используется в процессе сбора сведений непосредственно у специалистов путем бесед, которые требуют тщательной подготовки. Заранее составляют список сотрудников, с которыми намереваются беседовать, разрабатывают перечень вопросов о роли и назначении работ в деятельности объекта автоматизации, порядке их выполнения.

Метод анализа операций заключается в расчленении рассматриваемого делового процесса и работы на составные части, задачи, расчеты, операции и даже элементы.

Метод анализа представленного материала применим, в основном, при выяснении таких вопросов, на которые нельзя получить ответ от исполнителей.

Метод фотографии рабочего дня исполнителя работ предполагает непосредственное участие проектировщиков и применение рассчитанного для регистрации данных наблюдения специального листа фотографии рабочего дня и распределения его между работами.

Метод выборочного хронометража отдельных работ требует предварительной подготовки, известных навыков и наличия специального секундомера. Данные хронометража позволяют установить нормативы выполнения отдельных операций и собрать подробный материал о технике осуществления некоторых работ.

Метод личного наблюдения применим, если изучаемый вопрос понятен по существу и необходимо лишь уточнение деталей без существенного отрыва исполнителей от работы.

Метод документальной инвентаризации управленческих работ заключается в том, что на каждую работу в отдельности открывается специальная карта обследования, в которой приводятся все основные данные о регистрируемой работе или составляемых документах.

Метод ведения индивидуальных тетрадей – дневников. Записи в дневнике производятся исполнителем в течение месяца ежедневно, сразу же после выполнения очередной работы.

Метод самофотографии рабочего дня заключается в том, что наблюдение носит более детальный характер и происходит в короткий срок. Этот метод дает сведения о наиболее трудоемких или типичных отдельных работах, которые используются для определения общей трудоемкости выполнения всех работ.

Расчетный метод применяется для определения трудоемкости и стоимости работ, подлежащих переводу на выполнение с помощью ЭВМ, а также для установления объемов работ по отдельным операциям.

Метод аналогии основан на отказе от детального обследования какого-либо подразделения или какой-либо работы. Использование метода требует наличия тождественности и не исключает общего обследования и выяснения таких аспектов, на которые аналогия не распространяется.

4. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. ЯЗЫК УНИФИЦИРОВАННОГО МОДЕЛИРОВАНИЯ UML

4.1. Введение в UML

Объектно-ориентированное моделирование и проектирование – это подход к решению задач с использованием моделей, основанных на понятиях реального мира. Фундаментальным элементом является объект, объединяющий структуру данных с поведением. Объектно-ориентированные модели полезны для понимания задач, взаимодействия с экспертами по работе с приложениями, моделирования предприятий, подготовки документации и проектирования программ и баз данных.

UML (UnifiedModelingLanguage – унифицированный язык моделирования) – искусственный язык, который имеет некоторые черты естественного языка, и формальный язык, который имеет черты неформального.

В 1980-х гг. существовало множество различных методологий моделирования. Каждая из них имела свои достоинства и недостатки, а также свою нотацию. Это время получило название «войны методов». Проблема заключалась в том, что разные люди использовали разные нотации, и для того чтобы понять, что описывает та или иная *диаграмма*, зачастую требовался «переводчик». Один и тот же символ мог означать в разных нотациях абсолютно разные вещи. На рис. 4.1 представлены методы, которые существовали в то время и в какой-то мере повлияли на UML.

Объектно-ориентированный подход внес достаточно радикальные изменения в сами принципы создания и функционирования программ, но, в то же время, позволил существенно повысить производительность труда программистов, по-иному взглянуть на проблемы и методы их решения, сделать программы более компактными и легко расширяемыми. Как результат, языки, первоначально ориентированные на традиционный подход к программированию, получили ряд объектно-ориентированных расширений.

4.2. Структура языка UML

Язык UML имеет сложную иерархическую структуру (рис. 4.2).

Первый иерархический уровень языка UML составляют: сущности, отношения между сущностями и наглядные диаграммы.

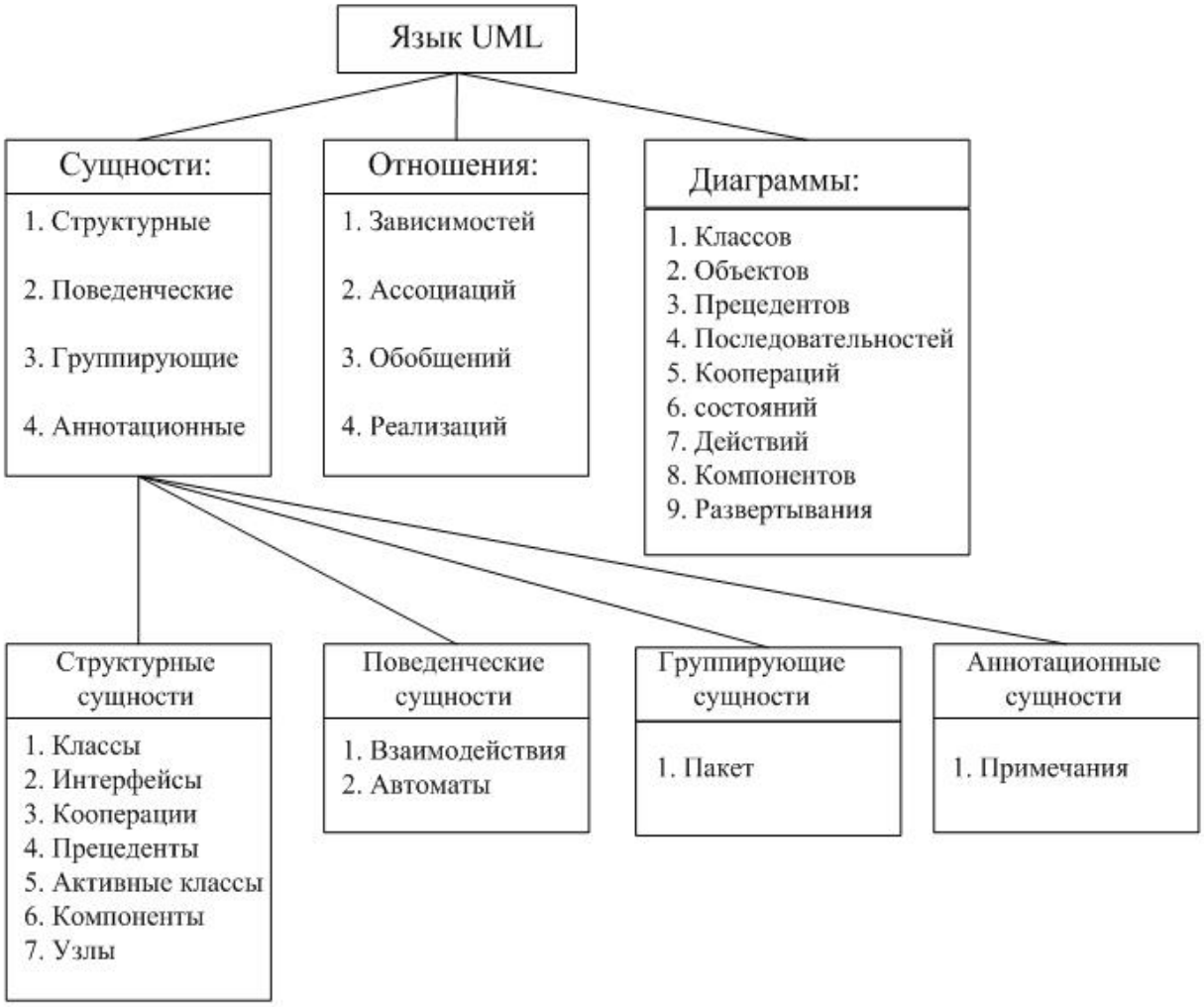

Рис. 4.1. Влияние методологий моделирования

Рассмотрим последовательно эти три основные понятия языка UML.

Язык UML имеет четыре вида сущностей: структурные, поведенческие, группирующие и аннотационные сущности. Понятие «структурные сущности» включает в себя: классы, интерфейсы, кооперации, прецеденты, активные классы, компоненты и узлы.

Класс – это совокупность (множество) объектов с общими атрибутами и операциями, а также с общими отношениями и семантикой. Класс изображается прямоугольником, разделенным на три поля. В первом поле помещается имя класса, однозначно определяющее данный класс среди множества других классов. Во втором поле помещаются атрибуты (общие свойства) класса. В третьем поле располагаются типовые операции, выполняемые объектами, принадлежащими данному классу.

На рис. 4.3 представлены примеры классов «Window» и «Frame». Класс «Window» показывает основные свойства (характеристики) присущие объектам этого класса. Класс «Frame» показывает подробные характеристики объектов. Детальная нотация класса дает возможность програм-

мистам и аналитикам визуализировать, специфицировать, конструировать и документировать класс на любом желаемом уровне детализации свойств класса, достаточном для поддержки прямого и обратного проектирования моделей и кода.

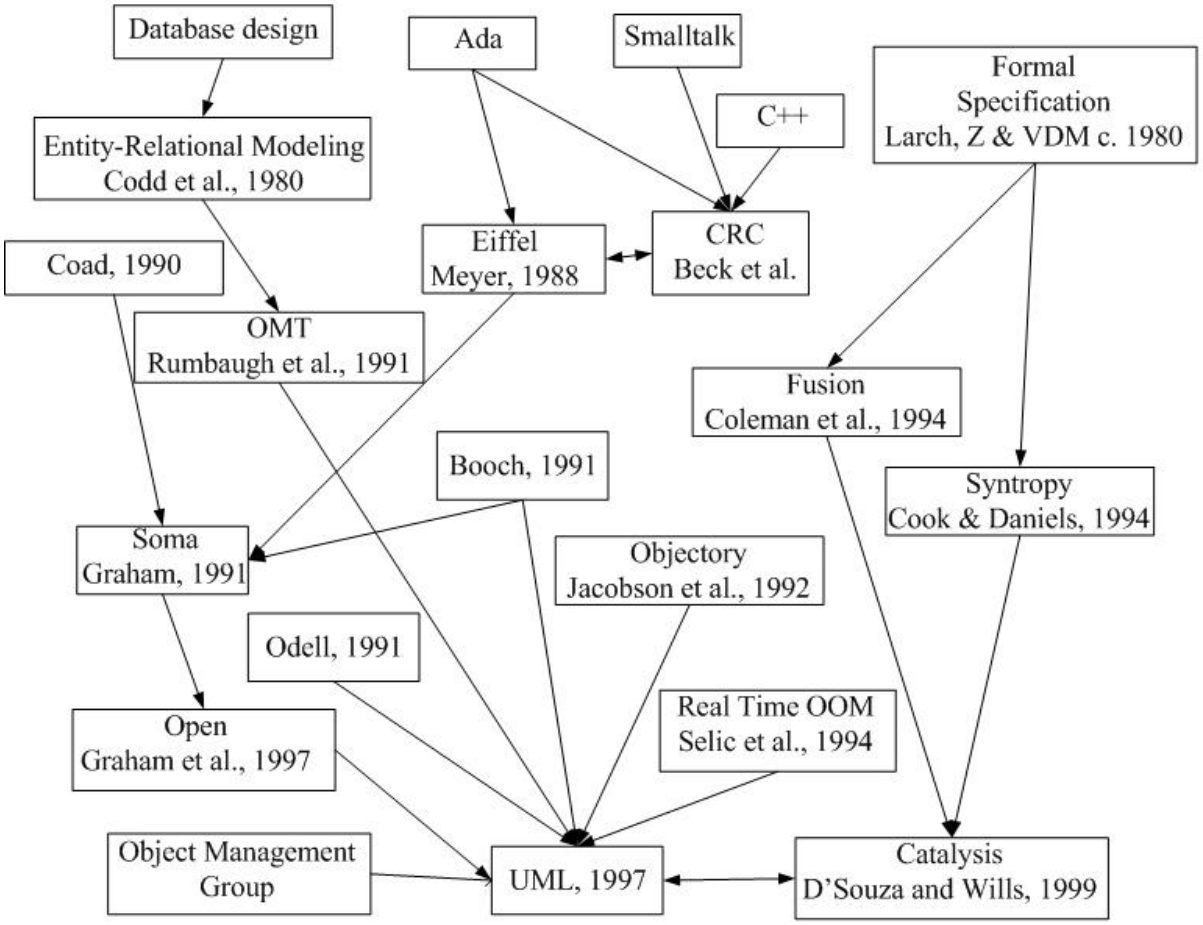

Рис. 4.2. Структура языка UML

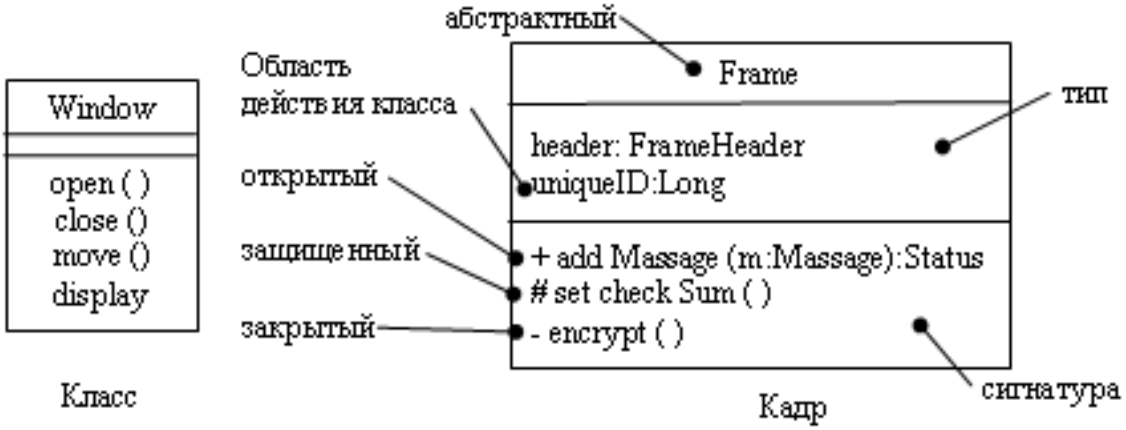

Рис. 4.3. Примеры классов

Интерфейс – это совокупность операций, предоставляемых классом или компонентом. Следовательно, интерфейс описывает поведение класса или компонента, видимое извне. Интерфейс определяет только описание (спецификации) операций класса или компонента, но он никогда не определяет физические реализации операций. Графически интерфейс изображается небольшим кружочком под которым пишется его имя.

Прецедент это описание множества последовательных событий, выполняемых компьютерной системой, которые приводят к наблюдаемому актером результату. Графически прецедент изображается в виде ограниченного непрерывной линией эллипса, обычно содержащего только имя прецедента.

Актер – это кто-то (или что-то) внешний по отношению к компьютерной системе, кто взаимодействует с ней. Графически актер изображается в виде пиктограммы, представляющей человека, поскольку актер это человек или группа людей, использующих данные, предоставляемые компьютерной системой. На UML-диаграммах пиктограммы прецедента и актера (рис. 4.4) обычно располагаются рядом. В совокупности они могут описывать внешнюю границу компьютерной системы.

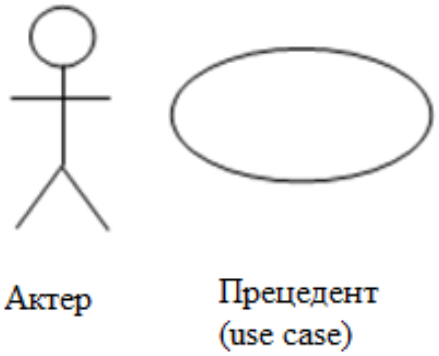

Рис. 4.4. Пример изображения актеров и прецедентов

«Пакет» отображает единственную в языке UML первичную группирующую сущность (рис. 4.5). В пакет можно поместить структурные и поведенческие сущности, а так же другие пакеты. Изображается пакет в виде папки с закладкой. Существуют также вариации пакетов, например, каркасы, модели и подсистемы.

Примечание изображается в виде прямоугольника с загнутым краем (рис. 4.5). На UML-диаграмме примечание присоединяется к одному или нескольким элементам диаграммы. Внутри прямоугольника-примечания помещаются комментарии или ограничения, относящиеся к элементу (или нескольким элементам) диаграммы. Комментарий может быть текстовым или графическим.

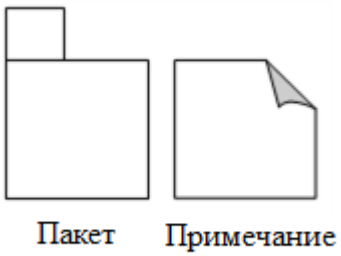

Рис. 4.5. Отображение сущностей пакета и примечания

Кооперация определяет взаимодействие, например классов. Участвуя в кооперации, классы совместно производят некоторый кооперативный результат. Один и тот же класс может принимать участие в нескольких кооперациях. Графически кооперация изображается в виде эллипса, ограниченного пунктирной линией.

Компонент – это физическая часть компьютерной или иной системы (рис. 4.6). Компонент соответствует некоторому набору интерфейсов и обеспечивает физическую реализацию этого набора. Компоненты могут быть разных видов. Например, одним из видов компонентов, используемых для моделирования программного обеспечения компьютерной системы, могут быть файлы исходного кода. Компонент, как правило, представляет собой физическую упаковку логических элементов, таких как классы, интерфейсы и кооперации. Графически компонент изображается в виде прямоугольника с вкладками.

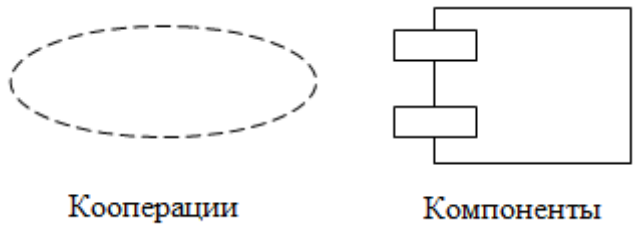

Рис. 4.6. Отображение сущностей кооперации и компонентов

Рассмотрим отношения, используемые в UML-диаграммах (рис. 4.7). Отношения составляют вторую ветвь структурного дерева языка UML. Однонаправленные отношения представляются на UML-диаграммах стрелками различных видов, а двунаправленное отношение представляется линией.

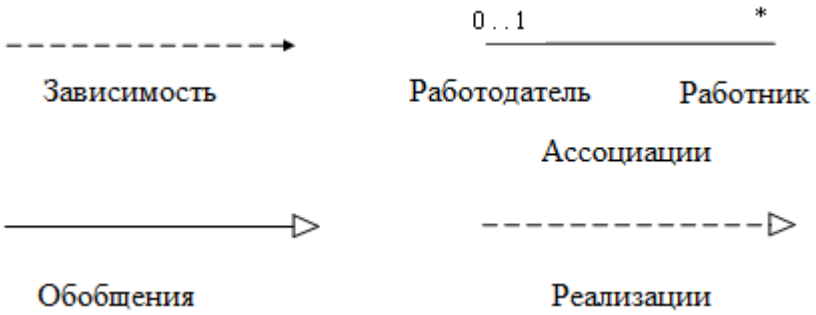

Рис. 4.7. Графическое отображение отношений

Однонаправленное отношение «зависимость» – это семантическое отношение между двумя сущностями, такое при котором изменение одной (первичной) сущности вызывает изменение семантики другой, зависимой сущности.

Ассоциация – это структурное двунаправленное отношение, описывающее совокупность взаимоотношений между объектами.

Частным случаем ассоциации является отношение типа «часть/целое». Отношение такого типа называется агрегированием (рис. 4.8). Агрегирование изображается в виде ассоциации с ромбом со стороны целого.

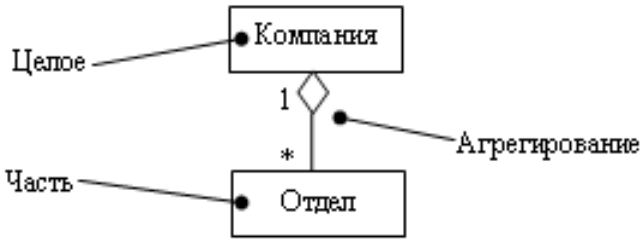

Рис. 4.8. Агрегирование

Обобщение – это однонаправленное отношение, называемое «потомок/прародитель», в котором объект «потомок» может быть подставлен вместо объекта прародителя (родителя или предка). Потомок наследует структуру и поведение своего родителя. Стрелка всегда указывает на родителя.

Поведенческие сущности делятся на два вида диаграмм. Диаграммы первого вида называются взаимодействиями (структурные), второго вида – автоматами (поведенческие диаграммы).

Структурные диаграммы используют для отражения физической организации сущностей в системе, т. е. отношений между объектами.

К структурным диаграммам относятся:

– *диаграммы классов* – используют классы и интерфейсы для отражения подробной информации о сущностях, образующих систему, и статических отношений между ними;

– *диаграммы компонентов* – отражают логическую организацию и зависимости, задействованные в реализации системы;

– *диаграммы составной структуры* – на концептуальном уровне диаграммы составной структуры связывают диаграммы классов с диаграммами компонентов;

– *диаграммы пакетов* составляют специфическую разновидность диаграмм классов. Основное внимание уделяется группировке классов и интерфейсов;

– *диаграммы объектов* показывают взаимосвязи между фактическими экземплярами классов в некоторый момент времени. Используются для представления «снимков» отношений в системе на стадии выполнения.

Поведенческие диаграммы сконцентрированы на поведении элементов системы.

К поведенческим диаграммам относятся:

– *диаграммы деятельности* – отражают переходы от одного поведения (или деятельности) к другому;

– *диаграммы коммуникаций* – разновидность диаграмм взаимодействия, в которых центральное место занимают элементы, участвующие в некотором поведении, и сообщения, которыми они обмениваются;

– *диаграммы обзора взаимодействия* – представляют собой упрощенные версии диаграмм деятельности. Они показывают, кому принадлежит «фокус управления» в ходе работы системы;

– *диаграммы последовательности* – разновидность диаграмм взаимодействия, в которой центральное место занимает тип и порядок сообщений, передаваемых между элементами в процессе выполнения;

– *диаграммы состояний* отражают внутренние переходы элемента (от отдельного класса до целой системы) между разными состояниями;

– *диаграммы синхронизации* – разновидность диаграмм взаимодействий, в которой центральное место занимает подробное указание хронометражной последовательности сообщений;

– *диаграммы вариантов использований* – отражают функциональные требования к системе. Дают представление (независящее от реализации) о том, что должна делать система, и позволяют разработчику модели сосредоточится на потребностях пользователя (вместо подробностей реализации).

4.3. Диаграммы классов

Диаграммы классов являются фундаментом почти всех объектно-ориентированных методов. Данная диаграмма использует классы и интерфейсы для отражения подробной информации о сущностях, образующих систему, и статических отношений между ними (рис. 4.9).

Имеется два основных вида статических отношений:

– ассоциации (например, клиент может взять напрокат ряд видеокассет);

– подтипы (медсестра является разновидностью личности).

На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между объектами.

Рассмотрим каждый фрагмент этой диаграммы.

Ассоциации представляют собой отношения между экземплярами классов (сотрудник работает в компании, компания имеет несколько офисов).

С концептуальной точки зрения ассоциации представляют концептуальные отношения между классами. На диаграмме показано, что Заказ может поступить только от одного Клиента, а Клиент в течение некоторого времени может сделать несколько Заказов. Каждый из этих Заказов может содержать несколько Строк заказа, причем каждая Строка заказа должна соответствовать единственному Товару.

Каждая из ассоциаций имеет два конца ассоциации; при этом каждый из концов ассоциации присоединяется к одному из классов этой ассоциации. Конец ассоциации может быть явно помечен некоторой меткой. Такая метка называется именем роли (концы ассоциации часто называют ролями).

Так, например, на рис. 4.9 конец ассоциации, направленной от класса Заказ к классу Строка заказа, имеет название *Позиции заказа.* Если такая метка отсутствует, концу ассоциации присваивается имя класса-цели.

Например: конец ассоциации от класса Заказ к классу Клиент может быть назван *клиент.*

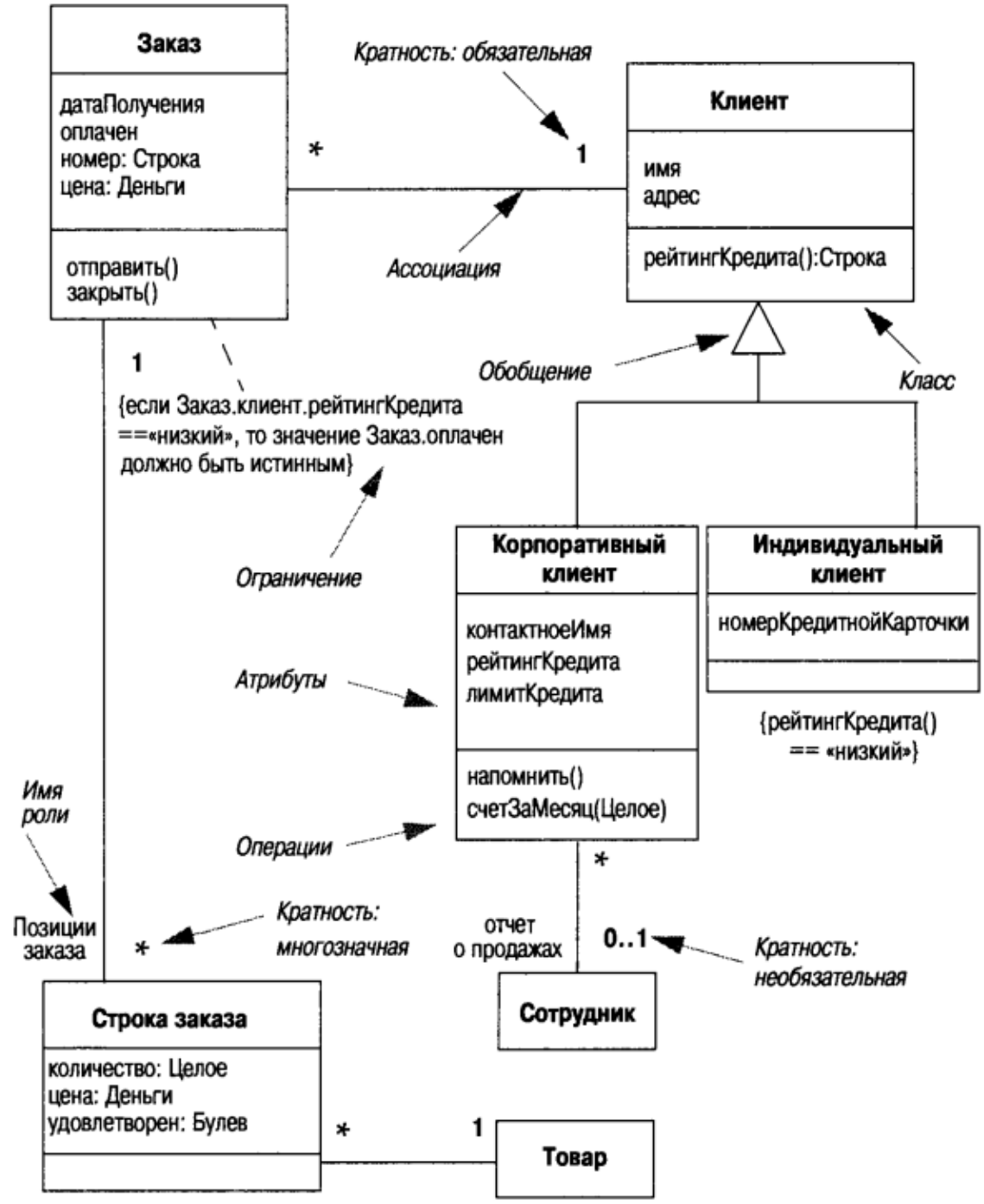

Рис. 4.9. Диаграмма классов

Конец ассоциации также обладает кратностью, которая показывает, сколько объектов может участвовать в данном отношении. На рис. 4.9 символ «*» возле класса Заказ для ассоциации между классами Заказ и Клиент показывает, что с одним клиентом может быть связано много заказов; напротив, символ «1» показывает, что каждый из заказов может поступить только от одного клиента.

В общем случае кратность указывает нижнюю и верхнюю границы количества объектов, которые могут участвовать в отношении. При этом символ «*» означает диапазон 0...∞: клиент может не сделать ни одного заказа, но верхний предел количества заказов, сделанных одним клиентом, никак не ограничен. 1 означает диапазон 1...1, т. е. заказ должен быть сделан одним и только одним клиентом.

На практике наиболее распространенными вариантами кратности являются «1», «*» и «0...1» (либо ноль, либо единица). В более общем случае для кратности может использоваться единственное число (например, 11 для количества игроков футбольной команды), диапазон (например, 2...4 для количества игроков карточной игры канаста) или дискретная комбинация из чисел или диапазонов (например, 2, 4 для количества дверей в автомобиле).

С точки зрения спецификации ассоциации представляют собой ответственности классов.

На рис. 4.9 предполагается, что существует один или более методов, связанных с классом Клиент, с помощью которых можно узнать, какие заказы сделал конкретный клиент. Аналогично в классе Заказ существуют методы, с помощью которых можно узнать, какой Клиент сделал конкретный Заказ и какие Строки заказа входят в этот Заказ.

Если установить стандартные соглашения по наименованию методов запросов, то, вероятно, из диаграммы можно было бы вывести названия этих методов.

Например: можно принять соглашение, в соответствии с которым взаимно однозначные отношения реализуются посредством метода, который возвращает связанный объект, а многозначные отношения реализуются посредством итератора, указывающего на совокупность связанных объектов.

На рис. 4.10 изображена та же диаграмма классов, но с добавлением стрелок к ассоциациям. Эти стрелки показывают направление **навигации.**

В модели спецификации таким способом можно показать, что Заказ обязан ответить на вопрос, к какому Клиенту он относится, а Клиенту нет необходимости отвечать, к какому Заказу он имеет отношение. Вместо симметричных ответственностей мы указываем теперь только односторонние. На диаграмме реализации это будет обозначать, что класс Заказ содержит указатель на класс Клиент, но класс Клиент не имеет указателей на класс Заказ.

Как можно увидеть, навигация имеет существенное значение на диаграммах спецификации и реализации.

На концептуальных диаграммах, которые строятся в самом начале разработки, направления навигации чаще всего отсутствуют. Они появляются при построении диаграмм классов уровня спецификации и реализации. Направления навигации, вероятно, могут отличаться с точки зрения спецификации и реализации.

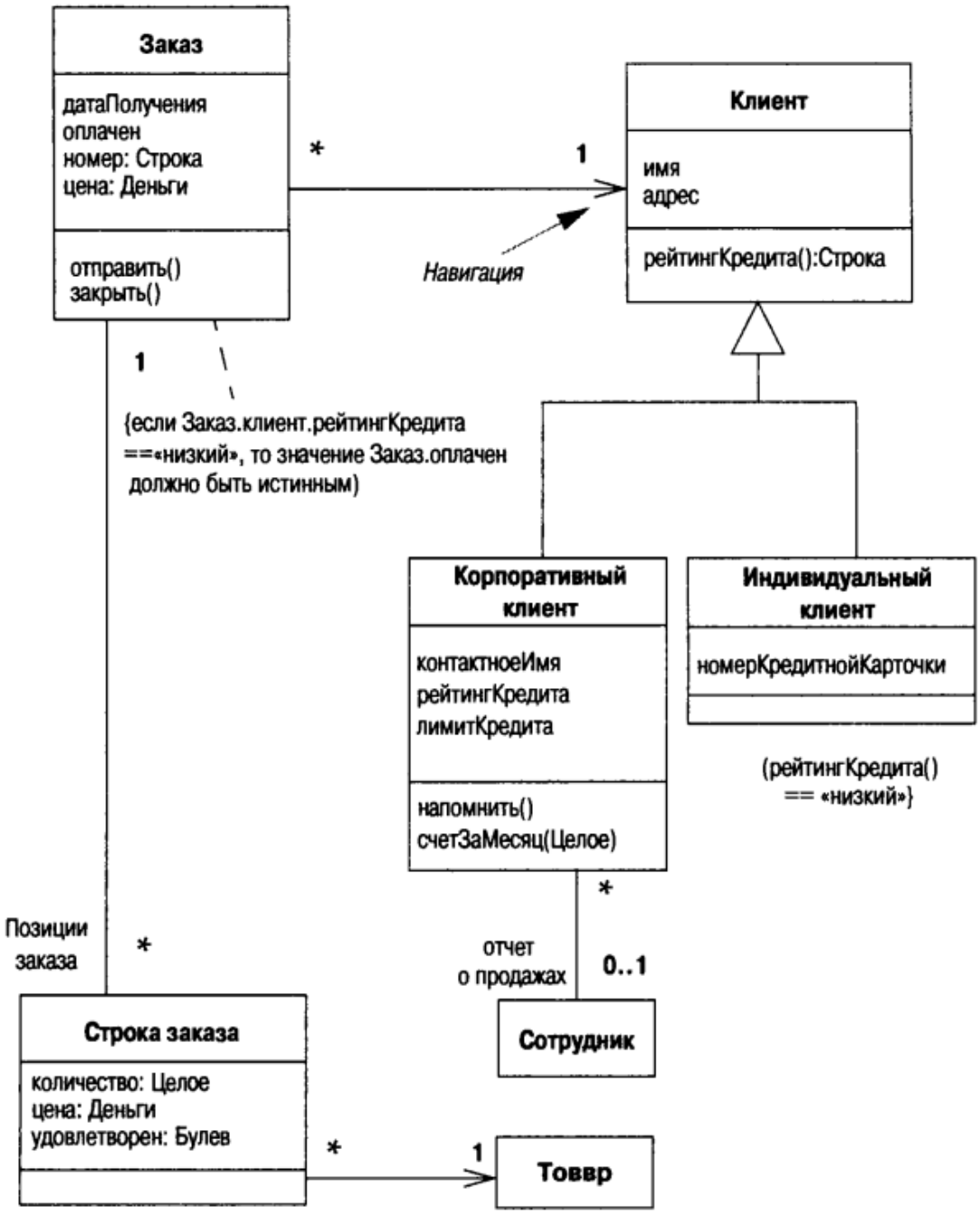

Рис. 4.10. Диаграмма классов с навигацией

Если навигация указана только в одном направлении, то такая ассоциация называется однонаправленной ассоциацией. У двунаправленной ассоциации навигация указывается в обоих направлениях. Если ассоциация не имеет стрелок навигации, то язык UML трактует это следующим образом: направление навигации неизвестно или ассоциация является двунаправленной.

Двунаправленные ассоциации содержат дополнительное ограничение, которое заключается в том, что эти две навигации являются инверсными

(обратными) по отношению друг к другу. Это аналогично обозначению обратных функций в математике.

Атрибуты очень похожи на ассоциации. На концептуальном уровне атрибут «имя Клиента» указывает на то, что клиенты обладают именами. На уровне спецификации этот атрибут указывает на то, что объект Клиент может сообщить вам свое имя и обладает некоторым способом для установления имени. На уровне реализации объект Клиент содержит некоторое поле для своего имени (называемое также переменной экземпляра или элементом данных).

В зависимости от степени детализации диаграммы обозначение атрибута может включать имя атрибута, тип и присваиваемое по умолчанию значение. В синтаксисе языка UML это выглядит следующим образом: *<видимость><имя>*: *<тип>-<значение по умолчанию>*.

Операции представляют собой процессы, реализуемые некоторым классом. Существует очевидное соответствие между операциями и методами класса. На уровне спецификации операции соответствуют общедоступным методам над определенным типом объекта. Обычно можно не показывать такие операции, которые просто манипулируют атрибутами, поскольку они и так подразумеваются. Однако иногда возникает необходимость показать, что данный атрибут предназначен только для чтения (read-only) или является неизменным (frozen), т. е. его значение никогда не изменяется. В модели реализации можно также указать защищенные и закрытые операции.

Полный синтаксис операций в языке UML выглядит следующим образом:

<видимость><имя> (<список-параметров>): <выражение-возвращающее-значение-типа> {<строка-свойств>},

где

– *видимость* может принимать одно из трех значений:

«+» общедоступная (public),

«#» защищенная (protected),

«-» закрытая (private);

– *имя* представляет собой строку символов.

– *список-параметров* содержит разделенные запятой параметры, синтаксис которых аналогичен синтаксису атрибутов: *<направление><имя>: <тип> = <значение по умолчанию».* При этом дополнительным элементом является *направление,* которое применяется, чтобы показать характер использования параметра – для входа (*in*), выхода (*out*) или в обоих направлениях *(inout).* Если значение *направления* отсутствует, оно предполагается входным (*in*);

– *выражение-возвращающее-значение-типа* содержит список разделенных запятой значений типов;

– *строка-свойств* указывает значения свойств, которые применяются к данной операции.

4.4. Диаграммы компонентов

Диаграммы компонентов отражают логическую организацию и зависимости, задействованные в реализации системы.

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На ней изображаются компоненты программного обеспечения системы и связи между ними. При этом выделяют два типа компонентов: исполняемые компоненты и библиотеки кода.

На рис. 4.11 изображена диаграмма компонентов для клиента ATM. На этой диаграмме показаны компоненты клиента системы ATM. В данном случае система разрабатывается с помощью языка C++. У каждого класса имеется свой собственный заголовочный файл и файл с расширением .CPP, так что каждый класс преобразуется в свои собственные компоненты на диаграмме. Например, класс ATM Screen преобразуется в два компонента ATM Screen диаграммы. Вместе эти компоненты представляют тело и заголовок класса ATM Screen. Выделенный темным компонент называется спецификацией пакета и соответствует файлу тела класса ATM Screen на языке C++. Невыделенный компонент также называется спецификацией пакета, но соответствует заголовочному файлу класса языка C++ (файл с расширением .H). Компонент ATM.exe является спецификацией задачи и представляет поток обработки информации. В данном случае поток обработки – это исполняемая программа.

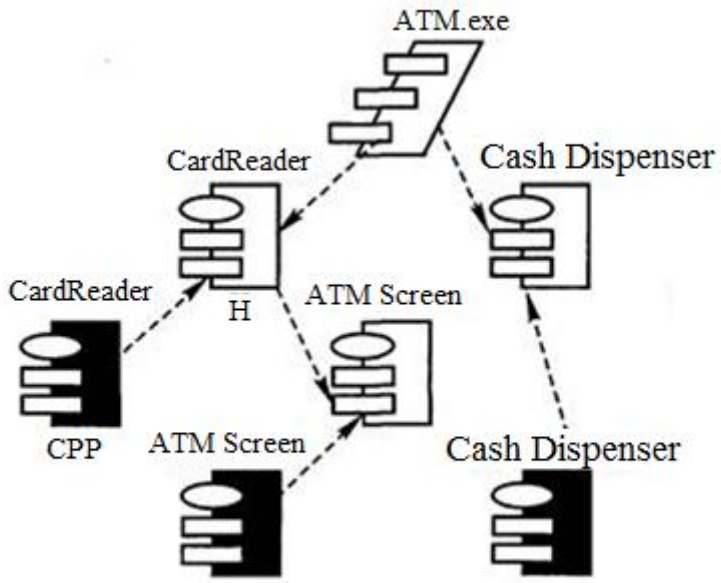

Рис. 4.11. Диаграмма компонентов

Компоненты соединены штриховой линией, отображающей зависимости между ними. Например, класс CardReader зависит от класса ATM Screen. Следовательно, для того чтобы класс CardReader мог быть скомпи-

лирован, класс ATM Screen должен уже существовать. После компиляции всех классов может быть создан исполняемый файл ATMClient.exe.

Пример ATM содержит два потока обработки, и, таким образом, получаются два исполняемых файла. Один из них – это клиент ATM, он содержит компоненты CashDispenser, CardReader и ATM Screen. Второй файл – сервер ATM, включающий в себя компонент Account.

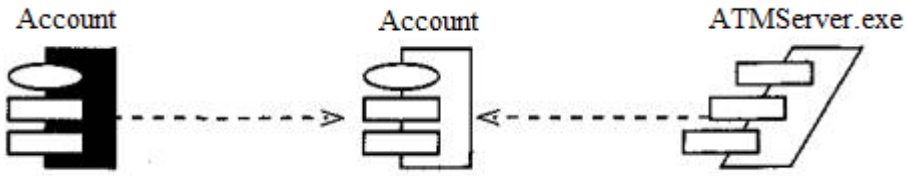

Рис. 4.12. Диаграмма Компонентов для сервера ATM

У системы может быть несколько диаграмм компонентов в зависимости от числа подсистем или исполняемых файлов. Каждая подсистема является пакетом компонентов. В общем случае, пакеты – это совокупности компонентов. В примере ATM используются два пакета: клиент ATM и сервер ATM. Диаграммы компонентов применяются теми участниками проекта, кто отвечает за компиляцию системы. Диаграмма Компонентов дает представление о том, в каком порядке надо компилировать компоненты, а также какие исполняемые компоненты будут созданы системой. Диаграмма показывает соответствие классов реализованным компонентам.

Разработка диаграммы компонентов предполагает использование информации как о логическом представлении модели системы, так и об особенностях ее физической реализации. До начала разработки необходимо принять решения о выборе вычислительных платформ и операционных систем, на которых предполагается реализовывать систему, а также о выборе конкретных баз данных и языков программирования.

После этого можно приступать к общей структуризации диаграммы компонентов. В первую очередь, необходимо решить, из каких физических частей (файлов) будет состоять программная система. На этом этапе следует обратить внимание на такую реализацию системы, которая обеспечивала бы не только возможность повторного использования кода за счет рациональной декомпозиции компонентов, но и создание объектов только при их необходимости.

Общая производительность программной системы существенно зависит от рационального использования ею вычислительных ресурсов. Для этой цели необходимо большую часть описаний классов, их операций и методов вынести в динамические библиотеки, оставив в исполняемых компонентах только самые необходимые для инициализации программы фрагменты программного кода.

Завершающий этап построения диаграммы компонентов связан с установлением и нанесением на диаграмму взаимосвязей между компонентами, а также отношений реализации. Эти отношения должны иллюстрировать все важнейшие аспекты физической реализации системы, начиная с особенностей компиляции исходных текстов программ и заканчивая исполнением отдельных частей программы на этапе ее выполнения.

Если же проект содержит некоторые физические элементы, описание которых отсутствует в языке UML, то следует воспользоваться механизмом расширения. В частности, использовать дополнительные стереотипы для отдельных нетиповых компонентов или помеченные значения для уточнения их отдельных характеристик.

4.5. Диаграммы вариантов использования

Диаграммы вариантов использования дают представление (независящее от реализации) о том, что должна делать система, и позволяют разработчику модели сосредоточится на потребностях пользователя (вместо подробностей реализации).

Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать.

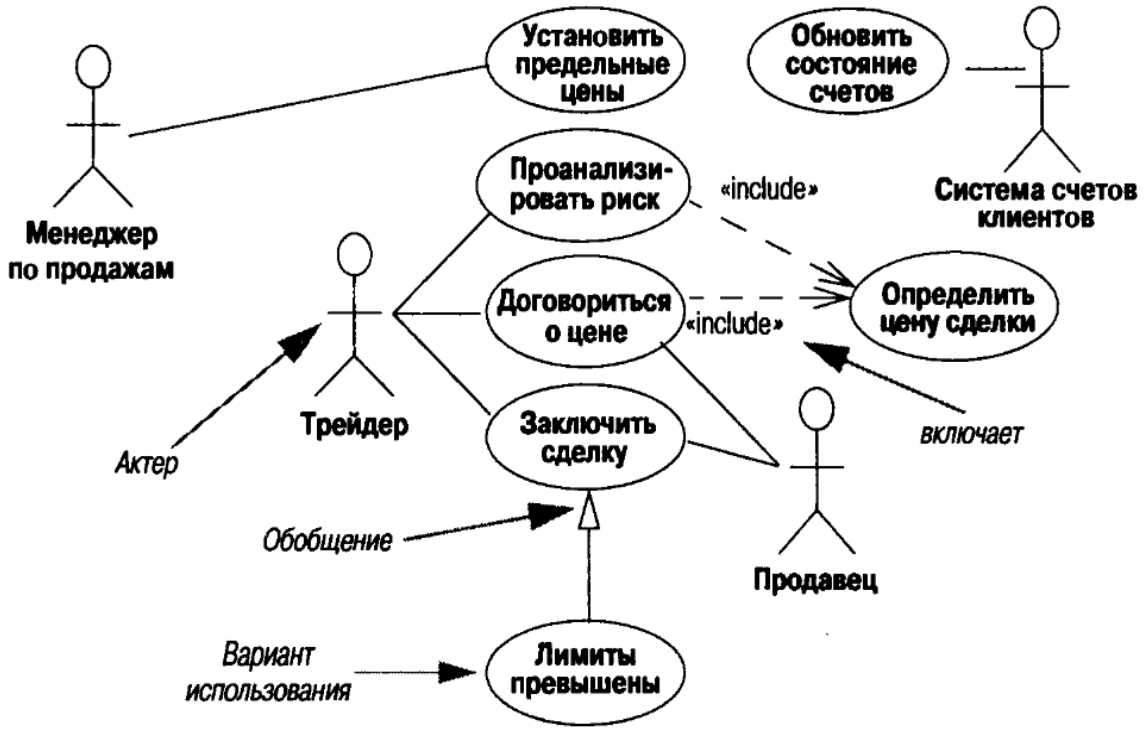

Рис. 4.13. Диаграмма вариантов использования
для финансовой торговой системы

Актер представляет собой некоторую роль, которую играет пользователь по отношению к системе.

В качестве актеров могут выступать:

- пользователи;
- системы, взаимодействующие с данной;
- время, если от него зависит запуск каких-либо событий в системе.

На рис. 4.13 представлены 4 актера: менеджер по продажам, трейдер (оптовый торговец), продавец и система счетов клиентов. Отдельный пользователь может играть и более одной роли.

Актеры связаны с вариантами использования. Один актер может выполнять несколько вариантов использования; в свою очередь, у варианта использования может быть несколько актеров, которые его выполняют.

В языке UML на диаграммах вариантов использования поддерживается несколько типов связей между элементами диаграммы. Это связи коммуникации (communication), включения (include), расширения (extend) и обобщения (generalization).

Связь коммуникации – это вязь между вариантом использования и действующим лицом. На языке UML связи коммуникации показывают с помощью однонаправленной ассоциации (сплошной линии).

Связь включения применяется в тех ситуациях, когда имеется какой-либо фрагмент поведения системы, который повторяется более чем в одном варианте использования. С помощью таких связей обычно моделируют многократно используемую функциональность.

Связь расширения применяется при описании изменений в нормальном поведении системы. Она позволяет варианту использования только при необходимости использовать функциональные возможности другого (рис. 4.14).

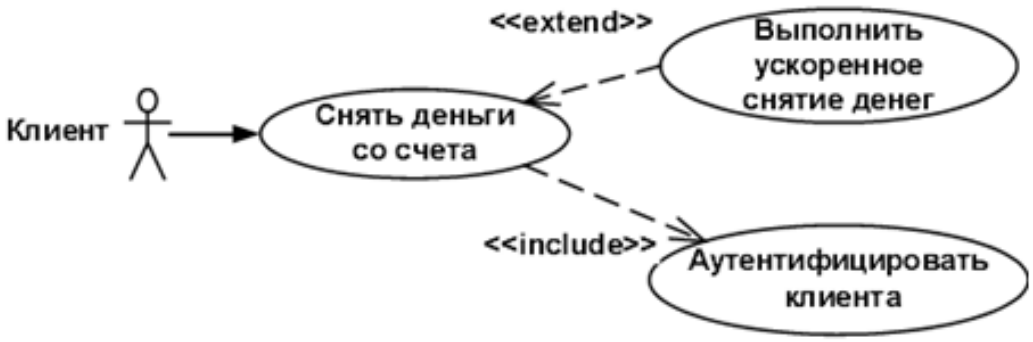

Рис. 4.14. Пример связи включения и расширения

С помощью связи обобщения показывают, что у нескольких действующих лиц имеются общие черты (рис. 4.15).

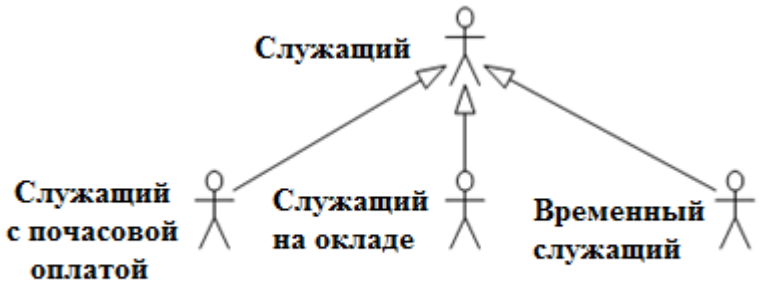

Рис. 4.15. Пример связи обобщения

4.6. Диаграмма состояний

Диаграммы состояний являются хорошо известным методом описания поведения систем. Они изображают все возможные состояния, в которых может находиться конкретный объект, а также изменения состояния объекта, которые происходят в результате влияния некоторых событий на этот объект. В большинстве объектно-ориентированных методов диаграммы состояний строятся для единственного класса, чтобы показать динамику поведения единственного объекта.

Существует несколько разновидностей представления диаграмм состояний, незначительно отличающихся друг от друга семантикой.

На рис. 4.16 изображена диаграмма состояний, описывающая поведение заказа в системе обработки заказов. На диаграмме представлены различные состояния, в которых может находиться заказ.

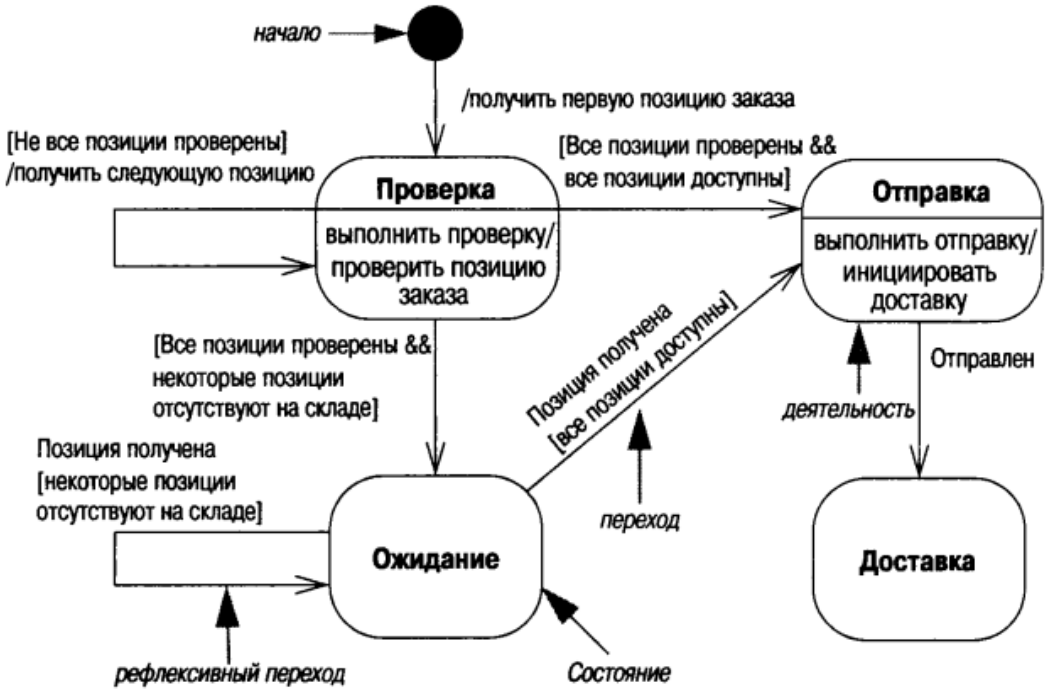

Рис. 4.16. Диаграмма состояний

Синтаксис метки перехода состоит из трех частей, каждая из которых является необязательной: *Событие [Сторожевое условие] / Действие.* В данном случае метка состоит только из действия «получить первую позицию заказа». После выполнения этого действия мы попадаем в состояние «проверка». С этим состоянием ассоциируется некоторая деятельность, которая обозначается меткой со следующим синтаксисом: *выполнить/деятельность.* В данном случае деятельность называется «проверить позицию заказа».

Действия ассоциируются с переходами и рассматриваются как мгновенные и непрерываемые. Деятельности ассоциируются с состояниями и могут продолжаться достаточно долго. Деятельность может быть прервана некоторым событием.

Если метка перехода не содержит никакого события, это означает, что переход произойдет, как только завершится какая-либо деятельность, ассоциированная с данным состоянием; в данном случае – как только будет выполнена «проверка». Из состояния «проверка» выходят три перехода. Метка каждого из них включает только «Сторожевое условие». Сторожевое условие – это логическое условие, которое может принимать одно из двух значений: «истина» или «ложь». Переход со сторожевым условием выполняется только в том случае, если данное сторожевое условие принимает значение «истина».

Из конкретного состояния в данный момент времени может быть осуществлен только один переход, таким образом, сторожевые условия должны быть взаимно исключающими для любого события. На рис. 4.16 изображены три условия:

1) если проверены не все позиции, входящие в заказ, мы получаем следующую позицию и возвращаемся в состояние «проверка»;

2) если проверены все позиции и все они имеются на складе, то мы переходим в состояние «отправка»;

3) если проверены все позиции, но не все из них имеются на складе, то мы переходим в состояние «ожидание».

Сначала рассмотрим состояние «ожидание». В этом состоянии не существует деятельностей, поэтому данный заказ находится в состоянии ожидания, пока не наступит некоторое событие. Оба перехода из состояния Ожидание помечены событием «Позиция получена». Это означает, что соответствующий заказ находится в состоянии «ожидание» до тех пор, пока он не обнаружит наступление данного события. В этот момент оцениваются сторожевые условия данных переходов, и выполняется соответствующий переход либо в состояние «отправка», либо обратно в состояние «ожидание».

В состоянии «отправка» имеется деятельность, которая инициирует доставку. Из этого состояния имеется единственный безусловный переход,

который происходит в результате наступления события «отправлен». Это означает, что рассматриваемый переход обязательно произойдет, если наступит данное событие. При этом следует заметить, что этот переход не произойдет, даже если завершится деятельность; наоборот, когда деятельность «инициировать доставку» завершится, данный заказ останется в состоянии «отправка», пока не наступит событие «Отправлен».

Рассмотрим переход с именем «отмена». Необходимо располагать возможностью отменить заказ в любой момент, пока заказ не доставлен клиенту. Это можно сделать, изобразив отдельные переходы из каждого состояния: «проверка», «ожидание» и «отправка». Альтернативный вариант – определить некоторое суперсостояние для трех перечисленных состояний, после чего нарисовать единственный выходящий из него переход. В этом случае подсостояния просто наследуют любые переходы суперсостояния. Оба подхода изображены на рис. 4.17 и 4.18. Они описывают одно и то же поведение системы.

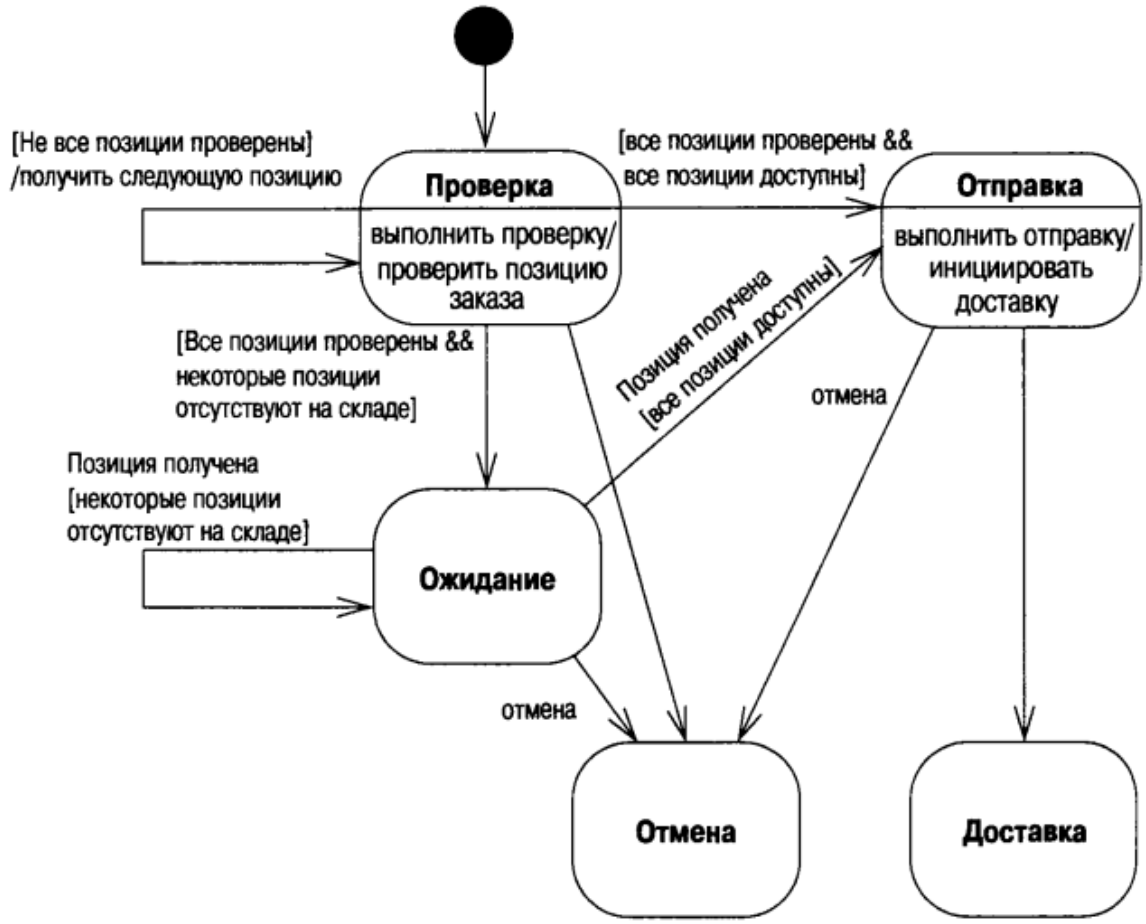

Рис. 4.17. Диаграмма состояний без суперсостояний

Диаграмма на рис. 4.18 выглядит перегруженной, хотя на ней изображено всего три дублирующих перехода.

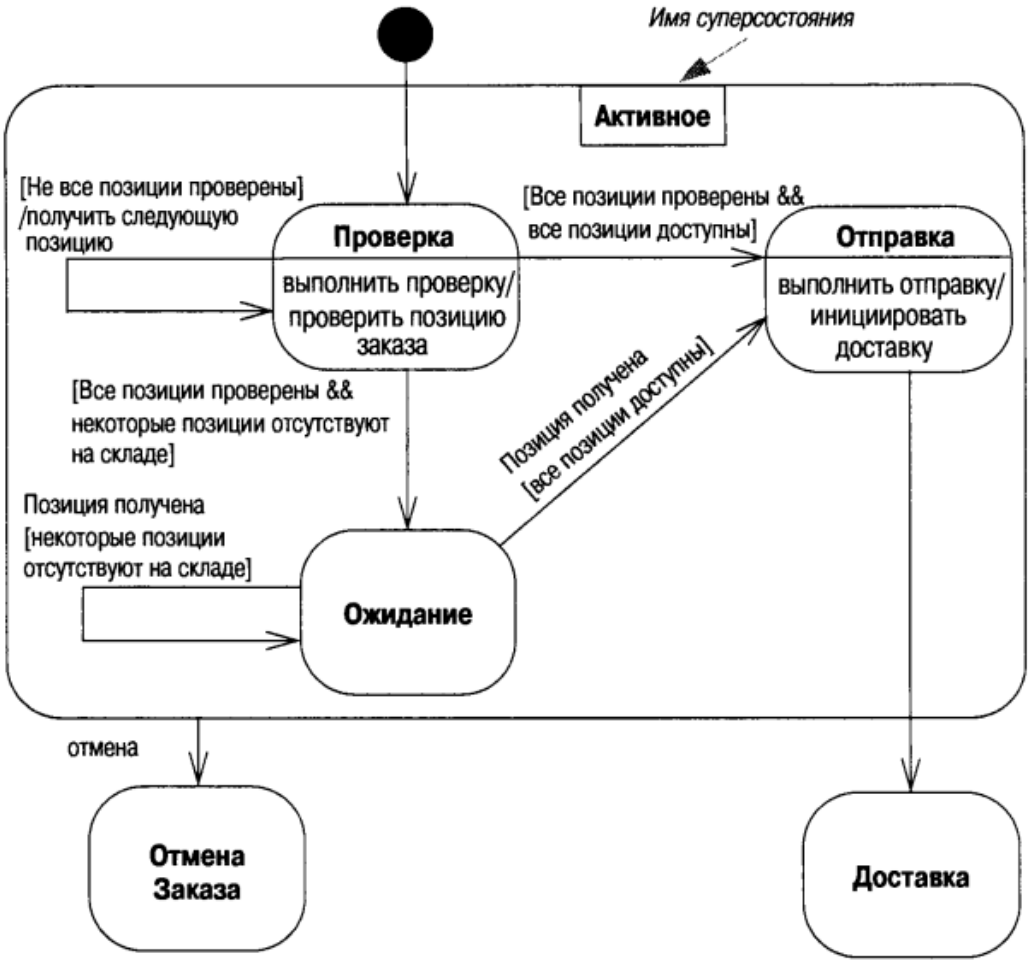

Рис. 4.18. Диаграмма состояний с супериcториями

Данная диаграмма выглядит более детальной, и если в последствии потребуется внести какие-либо изменения, то будет значительно труднее упустить из вида событие «Отмена».

Существуют также два особых события: «Вход» и «Выход». Любое действие, связанное с событием «Входа», выполняется в момент перехода объекта в данное состояние. Действие, ассоциированное с событием «Выхода», выполняется в том случае, когда объект покидает данное состояние в результате осуществления некоторого перехода. Если имеется так называемый рефлексивный переход, возвращающий объект обратно в то же самое состояние и связанный с каким-либо действием, то сначала должно выполниться действие «Выхода», затем действие данного перехода и, наконец, действие входа. Если с данным состоянием ассоциирована некоторая деятельность, то она начнет выполняться сразу после действия «Входа».

Помимо состояний заказа, связанных с наличием позиций заказа, существуют также состояния, связанные с подтверждением оплаты заказа. Эти состояния могут быть представлены диаграммой состояний, подобной той, которая изображена на рис. 4.19.

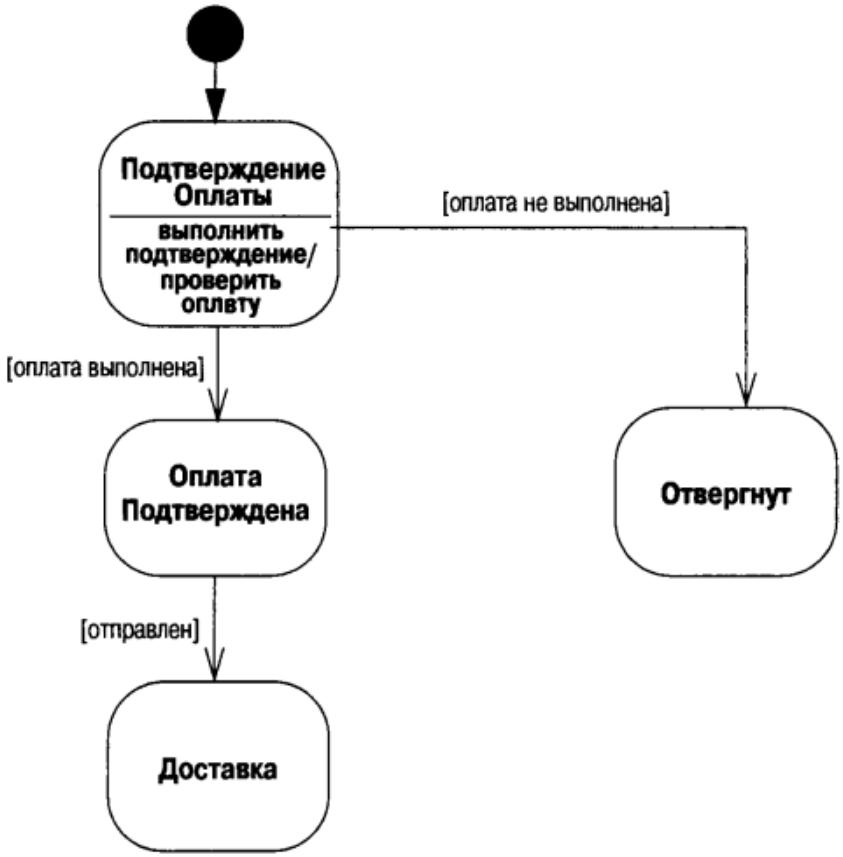

Рис. 4.19. Диаграмма состояний для подтверждения оплаты заказа

В рассматриваемом случае все начинается с проверки подтверждения оплаты. Деятельность «проверить оплату» завершается сообщением о результате выполнения данной проверки. Если оплата заказа выполнена, то данный заказ ожидает в состоянии «оплата подтверждена» до тех пор, пока не наступит событие «Отправлен». В противном случае заказ переходит в состояние «отвергнут».

Таким образом, поведение объекта Заказ определяется как комбинация поведений. Все эти состояния и рассмотренное ранее состояние «отмена» можно объединить в одну диаграмму параллельных состояний (рис. 4.20).

Обратите внимание, что на рис. 4.20 детали внутренних состояний не изображены.

Суть параллельных секций диаграммы состояний заключается в том, что в любой момент времени данный заказ находится одновременно в двух различных состояниях, каждое из которых относится к своей исходной диаграмме. Когда заказ покидает параллельные состояния, он оказывается только в одном состоянии. Из этой диаграммы можно увидеть, что в начальный момент заказ оказывается одновременно в двух состояниях: «проверка позиции заказа» и «подтверждение оплаты». Если деятельность «проверить оплату» в состоянии «подтверждение оплаты успешно» завер-

шится первой, то заказ окажется в двух состояниях: «проверка позиции заказа» и «оплата подтверждена». Если же наступит событие «отменена», то заказ окажется только в состоянии «отмена».

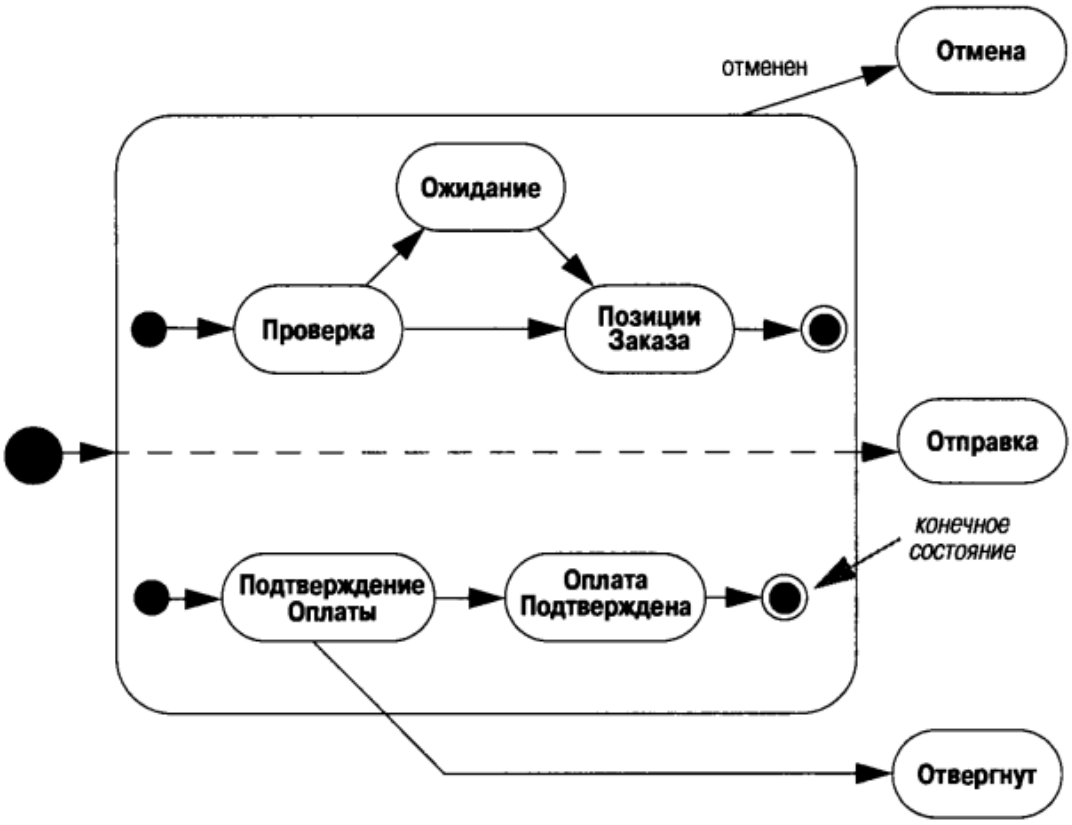

Рис. 4.20. Диаграмма параллельных состояний

Диаграммы параллельных состояний полезны в тех ситуациях, когда некоторый объект обладает множеством независимых поведений. Отметим, однако, что не следует создавать слишком большое количество параллельных состояний, описывающих поведение одного объекта. Если для некоторого объекта имеется несколько достаточно сложных диаграмм параллельных состояний, то следует рассмотреть возможность разделения этого объекта на отдельные объекты.

4.7. Диаграмма последовательности

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций.

На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия

(рис. 4.21). Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.

Каждое сообщение представляется в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице – сверху вниз. Каждое сообщение помечается как минимум именем сообщения. При желании можно добавить также аргументы и некоторую управляющую информацию. Можно показать самоделегирование (self-delegation) – сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни.

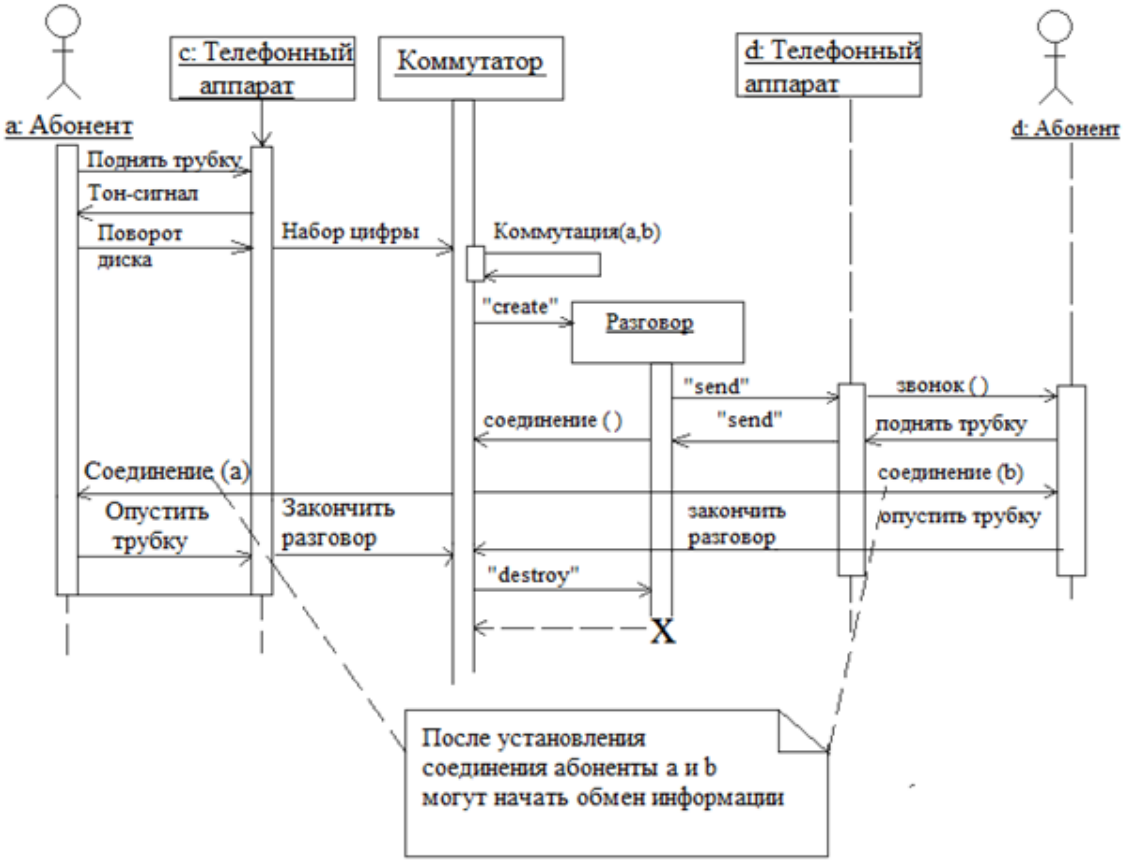

Рис. 4.21. Диаграмма последовательности

4.8. Кооперативная диаграмма

Диаграмма кооперации (диаграмма коммуникации) – передает ту же информацию, что и диаграмма последовательности.

Основными символами в диаграммах кооперации являются прямоугольник, называемый классификатором роли, и линия, обозначающая сообщение и называемая связью.

В качестве примера рассмотрим построение диаграммы кооперации для моделирования процесса телефонного разговора с использованием обычной телефонной сети. Напомним, что объектами в этом примере являются два абонента *a* и *b*, два телефонных аппарата *c* и *d*, коммутатор и сам разговор как объект моделирования.

На начальном этапе изобразим все объекты и связи между ними на диаграмме кооперации при помощи соответствующих обозначений (рис. 4.22).

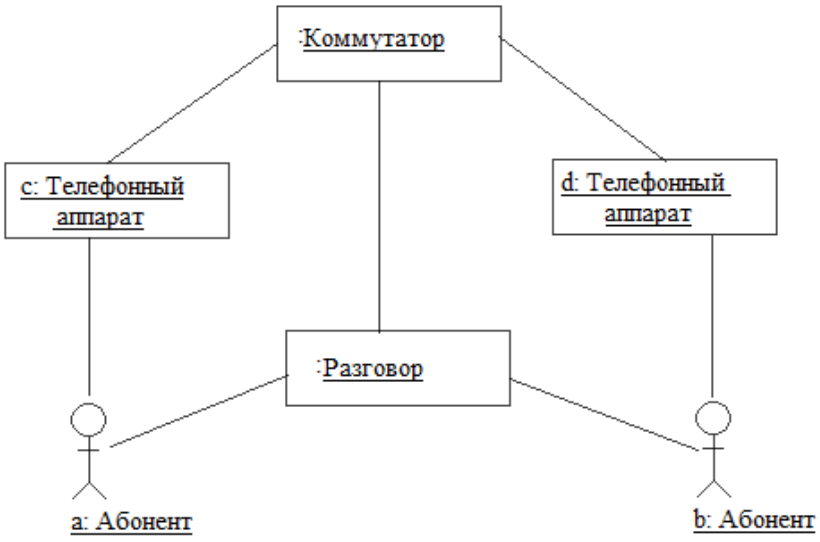

Рис. 4.22. Начальный фрагмент диаграммы кооперации
для примера моделирования обычного телефонного разговора

В последующем необходимо специфицировать все связи на этой диаграмме, указав на их концах необходимую информацию в форме ролей связей. Дополненный таким образом вариант диаграммы кооперации изображен ниже (рис. 4.23). Заметим, что для объекта «Разговор» указано помеченное значение {transient}, которое означает, что этот объект создается в процессе выполнения объемлющего процесса и уничтожается до его завершения. Напомним, что помеченные значения (taggedvalues) являются стандартными элементами языка UML.

На диаграмму кооперации необходимо нанести все сообщения, указав их порядок и семантические особенности. Окончательный фрагмент диаграммы кооперации изображен на рис. 4.24 и содержит модель кооперации только для начала разговора. Эта диаграмма может быть дополнена сообщениями, необходимыми для окончания разговора.

Диаграмма кооперации для примера с телефонным разговором не содержит ни временных особенностей передачи сообщений, ни особенностей жизненного цикла участвующих в данной кооперации объектов. Поэтому может быть принято решение о том, что она является избыточной при наличии построенной диаграммы последовательности. Этот факт не

вызывает сомнений в тех случаях, когда структура взаимодействующих объектов является достаточно тривиальной.

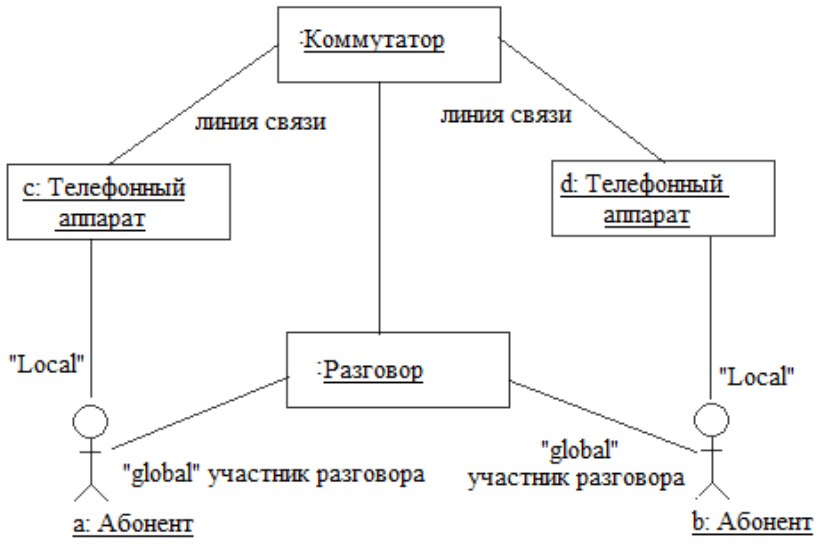

Рис. 4.23. Фрагмент диаграммы кооперации, дополненный стереотипами ролей связей, именами ассоциаций и помеченным значением объекта

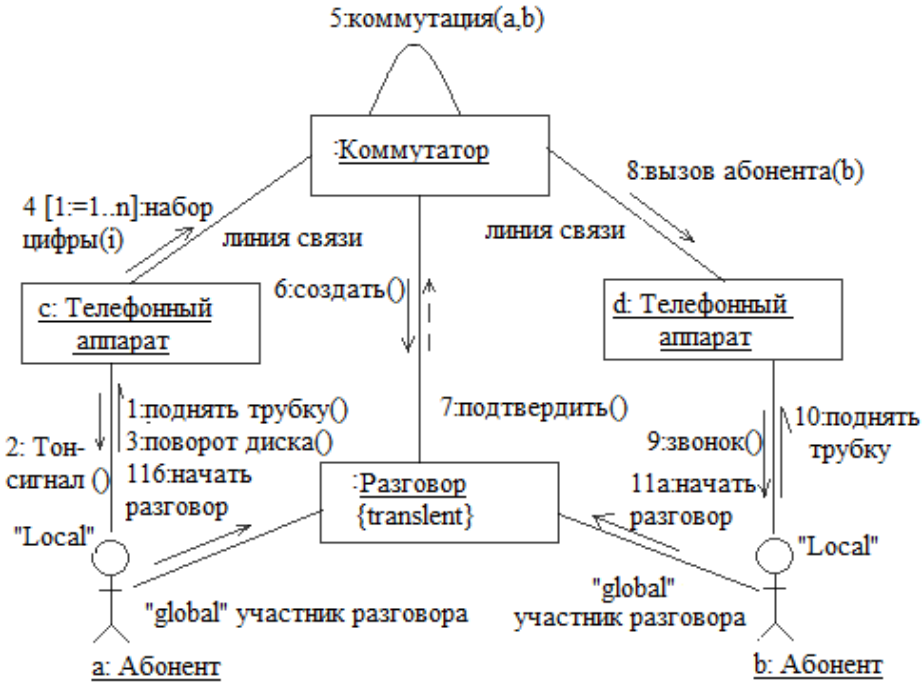

Рис. 4.24. Окончательный вариант диаграммы кооперации для моделирования телефонного разговора

Если же взаимодействующие объекты образуют между собой различные типы отношений-ассоциаций (композиция, агрегация), то диаграмма кооперации оказывается необходимым представлением модели на всех ее уровнях.

СПИСОК ЛИТЕРАТУРЫ

1. Анализ требований к автоматизированным информационным системам [Электронный ресурс] : учебное пособие / Ю. А. Маглинец. – М. : Интернет-Университет Информационных Технологий (ИНТУИТ), БИНОМ. Лаборатория знаний, 2013. – 200 с.

2. Антонов А. В. Системный анализ : учебник для вузов / А. В. Антонов. – 2-е изд., стер. – М. : Высшая школа, 2006. – 454 с.: ил.

3. Авдеев О. Н. Моделирование систем : учебное пособие / О. Н. Авдеев, Л. В. Мотайленко. – СПб. : Изд-во СПбГТУ, 2001.

4. Буч, Г. Язык UML. Руководства пользователя / Г. Буч, Д. Рамбо, И. Якобсон. – М. : ДМК Пресс, 2007. – 496 с.

5. Васильев, А. Java. Объектно-ориентированное программирование : учебное пособие. Стандарт третьего поколения / А. Васильев. – СПб. : Питер, 2011. – 400 с.

6. Веников, В. А. Теория подобия и моделирования / В. А. Веников, Г. В. Веников. – М. : Высшая школа, 1984.

7. Гагарина, Л. Г. Разработка и эксплуатация автоматизированных информационных систем : учебное пособие / Л. Г. Гагарина. – М. : ИД «Форум»: Инфра-М, 2013. – 384 с.

8. Ипатова, Э. Р. Методологии и технологии системного проектирования информационных систем / Э. Р. Ипатова, Ю. В. Ипатов. – М. : Флинта, 2008. – 255 с.

9) Мишин, А. В. Информационные технологии в профессиональной деятельности [Электронный ресурс] : учебное пособие / А. В. Мишин. – М. : Рос. акад. правосудия, 2011. – 311 с.

10. Моделирование систем : учебник для вузов / Б. Я. Советов, С. А. Яковлев. – 4 изд. стер. – М. : Высшая школа, 2005. – 343 с.: ил.

11. Пирогов, В. Информационные системы и базы данных: организация и проектирование / В. Пирогов. – СПб. : БХВ – Петербург, 2010. – 528 с.

12. Советов, Б. Я. Информационные технологии : учебник для вузов / Б. Я. Советов, В. В. Цехановский. – М. : Высшая школа, 2003. – 262 с.

13. Советов, Б. Я. Информационная технология / Б. Я. Советов. – М. : Высшая школа, 1994.

14. Советов, Б. Я. Моделирование систем / Б. Я. Советов, С. Я. Яковлев. – М. : Высшая школа, 1985.

15. Татарникова, Т. М. Управление данными : учебное пособие / Т. М. Татарникова // Федер. агентство связи, ГОУ ВПО СПбГУТ им. проф. М. А. Бонч-Бруевича. – СПб. : СПбГУТ, 2006. – 84 с.

16. Шелухин, О. И. Моделирование информационных систем : учебное пособие для вузов / О. И. Шелухин. – М. : Горячая линия – Телеком, 2012. – 516 с.

17. Яковлев, С. А. Эволюционные имитационные модели процессов и систем как методологическая основа интеллектуальных технологий обучения / С. А. Яковлев // Тез. докл. Междунар. конф. «Современные технологии обучения». – СПб., 1996.

Давыдова Екатерина Викторовна
Котлова Мария Владимировна

МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Учебное пособие

Редактор ***Л. К. Паршина***
Компьютерная верстка ***Е. А. Головинской***

План 2015 г., п. 66

Подписано к печати 25.11.2014
Объем 4,0 усл.-печ. л. Тираж 30 экз. Заказ 529

Редакционно-издательский центр СПбГУТ
191186 СПб., наб. р. Мойки, 61
Отпечатано в СПбГУТ

М. В. Котлова, Е. В. Давыдова

МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Учебное пособие

Санкт-Петербург
2015