Федеральное государственное автономное образовательное учреждение
высшего профессионального образования

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(НИУ «БелГУ»)

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

МАТОРИН С.И.
ЗИМОВЕЦ О.А.

ТЕОРИЯ СИСТЕМ И СИСТЕМНЫЙ АНАЛИЗ

Учебное пособие

Белгород 2012 г.

УДК
ББК

Печатается по решению
редакционно-издательского совета
НИУ «БелГУ»

Рецензенты:

Профессор кафедры информатики и информационных технологий Белгородской государственной сельскохозяйственной академии им. В.Я. Горина, доктор технических наук Ломазов В.А.

Профессор кафедры математического и программного обеспечения информационных систем НИУ БелГУ, доктор технических наук Корсунов Н.И.

Маторин С.И.
Теория систем и системный анализ: Учебное пособие / С.И. Маторин., О.А. Зимовец. – Белгород: Изд-во НИУ «БелГУ», 2012. – 288 с.

Учебно-практическое пособие по курсу «Теория систем и системный анализ» для студентов очной и заочной форм обучения представляет собой подборку теоретическогог материала и практикум по соответствующему курсу. Учебно-практическое пособие составлено в соответствии с требованиями по обязательному минимуму содержания и уровню подготовки бакалавров с высшим образованием Государственных образовательных стандартов высшего профессионального образования по направлениям 230700 «Прикладная информатика», 010300 «Математика. Компьютерные науки», «Фундаментальная информатика и информационные технологии» и 010500 «Математическое обеспечение и администрирование информационных систем».

УДК
ББК

Предисловие

Тотальная информатизация всех видов экономической деятельности приводит к необходимости тесного взаимодействия, с одной стороны, информационных процессов и технологий, с другой стороны, процессов, которые должны быть поддержаны средствами информатизации. Это, в свою очередь, приводит к возникновению «узкого места», для преодоления которого требуется взаимопонимание, с одной стороны, специалистов по компьютеризации и информационным технологиям, с другой стороны, специалистов, использующих эти средства для решения своих практических задач.

Для взаимопонимания и эффективного взаимодействия этих разных категорий специалистов необходимы:

– умение специалистов по информационным технологиям и программистов решать практические задачи путем создания программного обеспечения;

– умение заказчиков и потребителей программного обеспечения использовать его для повышения эффективности своей деятельности.

К сожалению, в настоящее время, только небольшая часть специалистов (с обеих сторон) может похвастаться наличием таких умений. Более того, многие отечественные специалисты, до сих пор, не осознают такой необходимости.

Данная проблема, на самом деле, уже имеет продолжительную и не только отечественную историю. Известно, например, что один из основоположников искусственного интеллекта американец Дж. Вейценбаум еще в 70-х годах прошлого столетия сетовал на то, что «к сожалению, многие университеты предлагают студентам учебные программы, которые позволяют учащимся выбирать путь, ориентированный только на освоение языков и процедур программирования без должного внимания к анализу и проектированию, и даже поощряют их в этом. Когда такие студенты завершают курс обучения, они оказываются в положении людей, научившихся прекрасно говорить на иностранном языке, но которые, попытавшись написать что-нибудь на этом языке, обнаруживают, что им самим сказать буквально нечего».

Не удивительно, поэтому, что 85% программных средств, разрабатываемых, например, для информационной поддержки бизнеса, по данным университета Карнеги Меллон оказываются неэффективными и просто выбрасываются в урну ввиду несоблюдения методологии и технологии разработки.

При этом профессионалы в области программирования (например, М. Фаулер, Г. Буч и т.д.) подчеркивают, что основной (чаще всего возникающий) риск при разработке программного обеспечения состоит не в создании плохой программы, а в создании программы, не соответствующей требованиям, т.е. бесполезной.

Современные специалисты по компьютерным технологиям (и, в частности, выпускники таких бакалавриатов как «прикладная информатика» и «математи-

ческое обеспечение и администрирование информационных систем» и т.д.) должны быть профессионалами в области информационно-аналитической деятельности. Это связано с тем, что на них обычно возлагаются:

– исследование, создание (структурирование, систематизация) и обеспечение использования информационных ресурсов (данных и знаний) в сфере деятельности организации;

– моделирование структуры, состава и функционирования организации;

– формулирование миссии организации, разработка стратегических планов;

– проектирование бизнес-процессов, разработка схем материальных и информационных потоков;

– выработка рекомендаций по обеспечению эффективности функциональных процессов, технологий работы функциональных структур и их взаимодействия, административных процессов, организационно-управленческой структуры;

– организация выполнения консалтинговых проектов и проектов по реинжинирингу.

Для подготовки выпускников указанных выше специальностей к решению данных задачи и читается, в частности, дисциплина «теория систем и системный анализ».

Содержание

Тема 1. Системные исследования

Цели и задачи изучения темы

Целью изучения данной темы является ознакомление с системными исследованиями как самостоятельным научным направлением.

При этом ставятся следующие задачи:

- ознакомление со структурой самостоятельного научного направления;
- структурирование системных исследований как самостоятельного научного направления;
- изучение эволюции системного подхода и его принципов.

1.1. Структура самостоятельного научного направления

Рассмотрим основные компоненты самостоятельного научного направления и их взаимодействие, обеспечивающее его функциональную целостность.

Схема взаимодействия компонент самостоятельного научного направления представлена на рисунке 1.1. Описание схемы заимствовано из работ Г.П. Мельникова.

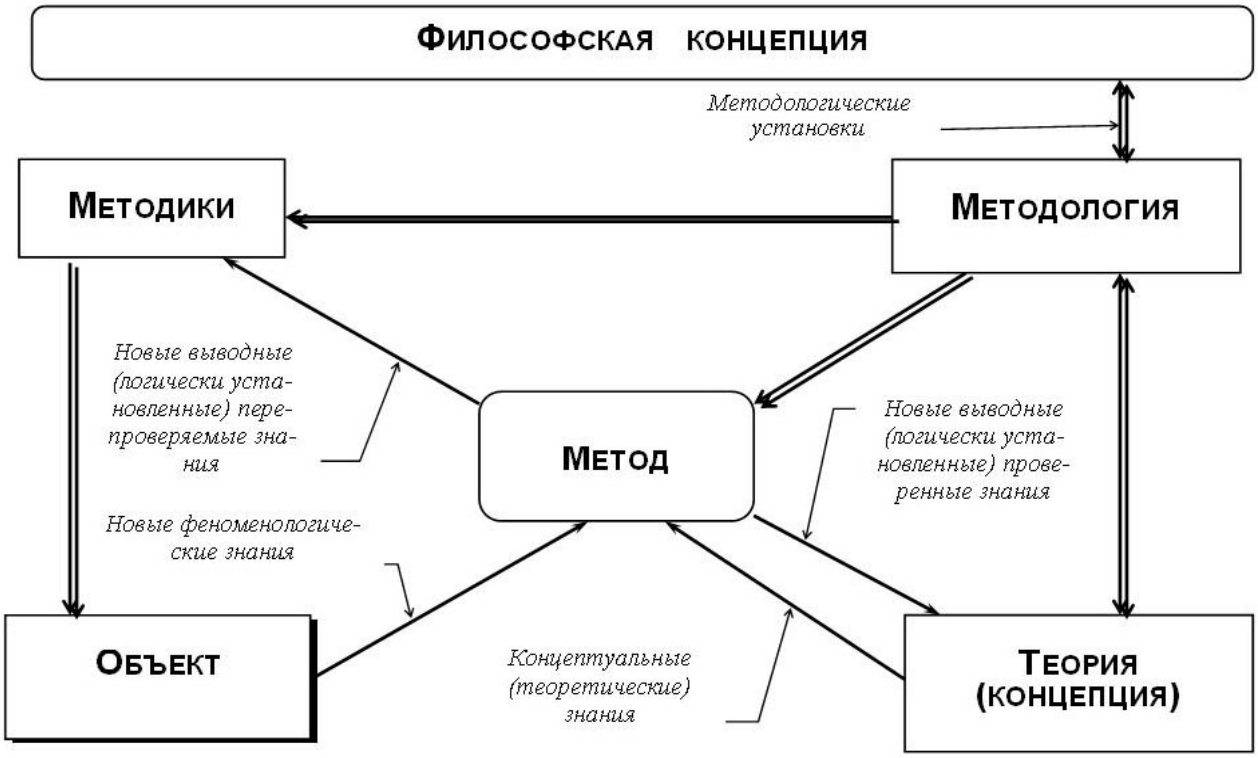


Рис. 1.1 - Схема взаимодействия компонент самостоятельного научного направления.

На схеме, кроме блоков (компонент научного направления или научной дисциплины), представлено еще два типа связей между этими компонентами. Связи типа «что» или «по данным», по которым осуществляется информационное взаимодействие компонент (обмен знаниями), изображены одинарными стрелками. Связи типа «как» или «по управлению», по которым осуществляется методологическое или методическое управление компонентами, изображены двойными стрелками.

Цель исследовательской деятельности – познание наблюдаемых и потенциальных свойств ***объекта*** исследований, являющегося первым необходимым компонентом научного направления. Результат исследовательской деятельности – знания его (объекта) текущего состояния, предыстории становления и умения на достаточно серьезных основаниях прогнозировать его будущее и делать выводы о его поведении, в том числе и в необычных для него условиях.

При рассмотрении научной дисциплины как системы в ее структуре функционально выделяется два вида знаний, обеспечивающих взаимодействия ее компонентов. Во-первых, это ***феноменологические*** знания, основанные на учете того, что дано непосредственно в наблюдении за объектом исследования как явлением, феноменом. Во-вторых, это ***логические*** знания, вырабатываемые на основе феноменологических, с учетом знания законов, отражающих причинно-следственные связи явлений. При равной степени полноты и достоверности знаний об изучаемом объекте предпочтительнее тот путь исследования, который позволяет обойтись меньшей долей феноменологически получаемых знаний и, соответственно, большую долю знаний об объекте получать логически.

Отработанные приемы наблюдения и эксперимента для получения новых феноменологических, эмпирических знаний об объекте представляют собой ***методики*** исследований, также являющиеся необходимым компонентом научной дисциплины.

Получение на основании феноменологических знаний новых логически устанавливаемых (выводных) может быть обеспечено только с помощью опорных, априорных знаний более универсального характера. Накопление, хранение и передачу таких знаний об объекте обеспечивает еще один необходимый компонент научной дисциплины – ***концепция*** или ***теория***, обслуживающая данное научное направление.

Выведение на основе новых феноменологических знаний и ранее установленных концептуальных знаний новых логически устанавливаемых знаний о данном объекте обеспечивается следующим компонентом научной дисциплины – ***методом*** исследования. Метод представляет собой способ теоретического освоения наблюдаемого и выявленного в эксперименте, ориентированный на определенную феноменологию и концепцию.

Любая самостоятельная научная дисциплина (научное направление), представляя собой определенную целостность конкретно-научных знаний, не существует, тем не менее, сама по себе. Она функционирует в среде общенаучных знаний, которые периодически привлекаются данной дисциплиной, что обеспечивает формирование навыков и приемов философского осмысления и анализа специальных концепций, теорий, методов и методик данной науки. Эти специфические навыки, возникающие из потребностей философского осмысления проблем конкретной науки и конкретно-научного осмысления философских категорий и законов, формируют особый компонент данной науки – ***методологию***. Методология обеспечивает эффективность всей научной деятельности за счет согласования и усовершенствования всех ее компонентов, особенно в «нештатных» ситуациях.

О существовании самостоятельной научной дисциплины (научного на-

правления) можно говорить в тех случаях, когда достаточно четко сформировались границы ее ***объекта***, сложились ***методики*** получения новых феноменологических знаний, существует ***концепция*** или ***теория*** данной дисциплины. Для существования самостоятельной научной дисциплины, кроме того, должен существовать ***метод*** (или набор методов) данной научной дисциплины и должна сформироваться определенная ***методология***. Лишь в этом случае возможна эффективная исследовательская деятельность, имеющая в своем составе все необходимые компоненты: объект, методики, метод, теорию или, хотя бы, концепцию и методологию.

Самостоятельная научная дисциплина (научное направление) как система навыков изучения определенного объекта и накопленных данных о нем отражает в себе объект несколькими проекциями в совокупности своей составляющими ***предмет*** (***концепт***) данной науки: объект как носитель некоторой сущности представлен как отражение этой сущности в концептуальном ядре теории; объект как феномен, как явление представлен с помощью средств индикации и фиксации в массиве феноменологических данных актуально и в феноменологической части теории.

При анализе природы научной дисциплины представляется полезным учитывать мнения основоположников системологии о соотношении формального и содержательного.

«С позиций системологии получение выводного знания, то есть главная функция метода, может основываться лишь в частном случае на формально-логических и математических процедурах. В общем случае – это должен быть прежде всего содержательный вывод, учитывающий законы развития, адаптации и эволюции».

«Стремление к сущностному познанию – важнейшая часть системной методологии любой научной дисциплины. Подмена сущностного рассмотрения конструктами и логически оправданными моделями есть отступление от системной методологии».

1.2. Структура системных исследований

Любая концепция системного подхода функционирует и развивается в рамках самостоятельного научного направления, носящего название ***системные исследования***. Рассмотрим специфику этого направления с целью уточнения места и особенностей основного предмета нашего рассмотрения, то есть ***системологии***.

Системные исследования – одно из самых современных и молодых научных направлений. Они представлены, в настоящее время, ***системным подходом, общей*** или ***абстрактной теорией систем*** и ***системным анализом.*** Данные дисциплины составляют определенные аспекты или стороны современных системных исследований. По своим задачам названные компоненты системных исследований, выходят за рамки существующего сегодня дисциплинарного членения науки и техники. Получаемые этими компонентами результаты применимы к целым комплексам научных и технических дисциплин [1].

Главным из этих компонентов является системный подход, имеющий методологическую природу и общенаучный междисциплинарный характер [2]. Ведущая, ключевая роль системного подхода обусловлена его положением в структуре взаимодействия компонентов системных исследований.

В рамках системных исследований используются средства концептуального и методологического анализа в соединении с формальным аппаратом современной логики и некоторых разделов математики. Специалистами в этой области разрабатываются принципы, на основе которых разрабатываются системные модели объектов произвольной природы и системные методы их исследования. Результатом этих исследований и разработок, с учетом специфики подхода, являются, во-первых, методы анализа особого класса природных и социальных объектов, наиболее адекватным названием которых можно считать термин «сложные системы», а, во-вторых, средства систематизации, классификации и упорядочения научных (и не только научных) знаний.

Системные исследования, в целом, и системный подход, в частности, представляют собой определенный этап в развитии методов познания, методов исследовательской и конструкторской деятельности, способов описания природы естественных или искусственных объектов, то есть определенный этап в развитии науки.

Схема на рисунке 1.2 иллюстрирует функции каждого из компонентов современных системных исследований по аналогии с функциями ***объекта***, ***теории (концепции)***, метода, ***методики*** и ***методологии*** самостоятельной научной дисциплины (научного направления) [3, 4].

Главной целью научно-исследовательской деятельности является познание свойств и сущности объекта, который в рамках системных исследований

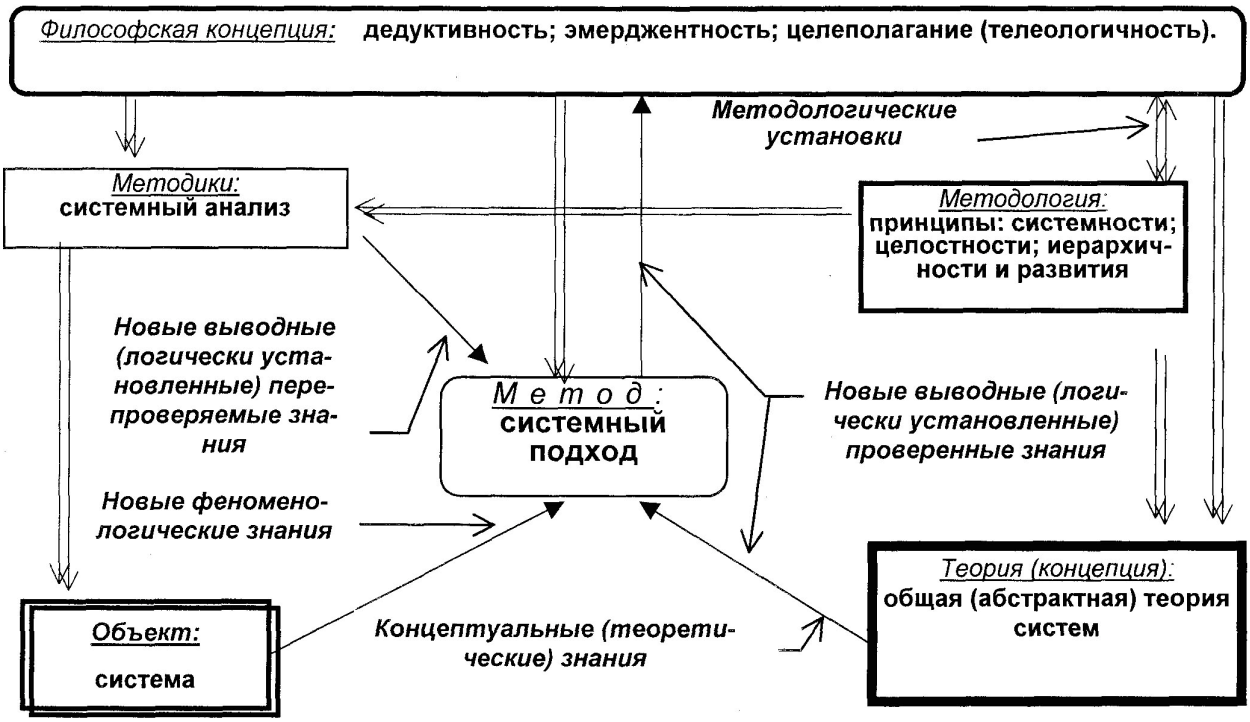


Рис. 1.2. - Схема взаимодействия компонентов системных исследований.

представляет собой ***систему,*** как предельную абстракцию объекта исследований произвольной природы.

Накопление, хранение и передачу знаний об объекте (системе) обеспечивает концепция или теория, обслуживающая данное научное направление, которая в рамках системных исследований представляет собой ***общую*** или ***абстрактную теорию систем***, как теорию (концепцию) предельной степени абстракции. Из множества существующих определений этой теории можно упомянуть следующее.

Общая теория систем — это научная и методологическая концепция исследования объектов, представляющих собой системы. Она тесно связана с системным подходом и является конкретизацией его принципов и методов. Первый вариант общей теории систем был выдвинут Людвигом фон Берталанфи, основная идея которого состоит в признании изоморфизма законов, управляющих функционированием системных объектов.

Получение новых феноменологических, эмпирических знаний об объекте (системе) обеспечивают методики исследований, которые в рамках системных исследований должны представлять собой ***системный анализ***, как единственно возможный способ получения данных о предельно абстрактном объекте, то есть системе. Из множества существующих определений этого анализа можно упомянуть следующие.

Системный анализ — это совокупность приемов научного познания, представляющая собой последовательность действий по установлению структурных связей между переменными или элементами исследуемой системы. Опирается на комплекс общенаучных, экспериментальных, естественнонаучных методов. ***Системный анализ*** возник в эпоху разработки компьютерной техники. Успех его применения при решении сложных задач во многом определяется современными возможностями информационных технологий. Н.Н. Моисеев приводит, по его выражению, довольно узкое определение системного анализа: «Системный анализ — это совокупность методов, основанных на использовании ЭВМ и ориентированных на исследование сложных систем — технических, экономических, экологических и т.д. Результатом системных исследований является, как правило, выбор вполне определенной альтернативы: плана развития региона, параметров конструкции и т.д. Поэтому истоки системного анализа, его методические концепции лежат в тех дисциплинах, которые занимаются проблемами принятия решений: исследование операций и общая теория управления».

Выведение на основе новых феноменологических и ранее установленных концептуальных знаний новых знаний, логически устанавливаемых, о данном объекте обеспечивается методом исследования. Метод, как средство выведения новых знаний о предельно абстрактных объектах (системах), в рамках системных исследований является ***системным подходом***. Из множества существующих определений этого подхода можно упомянуть следующее.

Системный подход — это методическое направление исследования, в основе которого лежит рассмотрение объекта как целостного множества элементов в совокупности отношений и связей между ними, то есть рассмотрение

объекта как системы. Говоря о ***системном подходе***, можно говорить о некотором способе организации наших действий, таком, который охватывает любой род деятельности, выявляя закономерности и взаимосвязи с целью их более эффективного использования. При этом ***системный подход*** является не столько методом решения задач, сколько методом постановки задач. Как говорится, «Правильно заданный вопрос — половина ответа». Это качественно более высокий, нежели просто предметный, способ познания.

Эффективность всей научной деятельности за счет согласования и усовершенствования всех ее компонентов, особенно в «нештатных» ситуациях, обеспечивается методологией, которая в рамках системных исследований представляет собой диалектический ***принцип системности***.

Рассмотренный способ представления структуры взаимодействия компонентов системных исследований позволяет предположить, что с увеличением сложности, общности, абстрактности объекта исследований методы и методики исследований должны все в большей степени соответствовать системному подходу и системному анализу. Системная концепция или теория конкретного научного направления при этом должны все более соответствовать абстрактной теории систем, а методология – принципам диалектики.

Анализ специфики и структуры системных исследований показывает, таким образом, что системный подход, в общем, и системология, в частности, представляют собой универсальный метод изучения и исследования сложных объектов произвольной природы. Это и делает знание основных положений системологического подхода в настоящее время чрезвычайно актуальным, а их использование в научной практике чрезвычайно эффективным и продуктивным.

1.3. Эволюция системного подхода

Понимание места и роли системологии в рамках науки и конкретного научного направления позволяет рассмотреть ее специфические особенности более подробно. Рассмотрим эти особенности путем сравнения современной системологической концепции с предшествующей системной.

Широкое внедрение системного подхода во всех сферах научной, конструкторской и управленческой деятельности, как и любая существенная перестройка, является процессом длительным и противоречивым, обусловленным, в первую очередь, продолжением формирования и развития самого системного подхода как самостоятельного научно-методологического направления. В ходе своего развития и совершенствования системный подход не сразу обретает признаки и черты, окончательно свободные от влияния аналитического этапа развития науки и полностью соответствующие современному ноосферному этапу. Среди существующих концепций системного подхода такими чертами обладает только концепция, которая называется ***системологической*** или просто ***системологией***. [2, 3].

Данная концепция обладает существенными преимуществами, выгодно отличающими ее от известных ранее концепций и уже привычных. Эти пре-

имущества могут быть проиллюстрированы в ходе сравнительного анализа фундаментальных принципов, лежащих в основе любой концепции системного подхода. Это связано с тем, что системный подход, как существенный аспект ноосферного этапа развития науки, в целом, и системных исследований, в частности, принципиально ориентирован на применение для выполнения своих задач ***диалектических принципов: системности, целостности, иерархичности и развития*** [1, 5, 6]. Сравнительный анализ применения основополагающих диалектических принципов традиционным системным подходом и системологическим (системологией), результаты которого приведены в таблице 1.1, показывает, что первый в значительной степени продолжает использовать методологические установки прошедшего аналитического этапа развития науки и, следовательно, малоэффективен и бесперспективен в современных условиях.

Таблица 1.1. – Сравнение системных подходов.

Системный подход	***Системология***
Принцип системности: Понятие системы многозначно (более 40 определений). Понятия «мера» или «степень системности» отсутствуют. Объекты рассматриваются как системы только при определённых условиях [7].	***Принцип системности***: Понятие системы однозначно (предельно абстрактно). Введены понятия «мера» или «степень системности» для рассмотрения любого объекта как системы, имеющей определённую меру [5].
Принцип целостности и многоаспектности: Учитывается либо только структурная целостность, когда природа связываемых элементов считается несущественной, либо только субстанциальная целостность, когда не учитываются связи между частями целого [8, 9].	***Принцип целостности и многоаспектности***: Учитывается комплексный характер целостности системы, что обеспечивает взаимосогласование структуры и субстанции системы при её взаимодействии со средой. Основной аспект целостности – функциональный [10].
Принцип иерархичности: Учитывается иерархичность только внутренней структуры системы, что обеспечивает исследование объектов методом, так называемого, «серого (светлого) ящика» [11].	***Принцип иерархичности***: Учитывается, в том числе иерархичность структуры внешней для системы среды, что обеспечивает исследование объектов методом «всё более и более светлого ящика» [5, 11].
Принцип развития: Рассматриваются только статические параметры системы. Не рассматриваются причины возникновения системы и этапы её становления. Нет понятия «адаптация системы» [7, 10].	***Принцип развития***: Рассматриваются, в том числе, динамические характеристики системы, что обеспечивает понимание причин её возникновения и этапов становления. Введено понятие адаптации системы [3, 8].

Системология же, последовательно применяя принципы диалектики, методологически однозначно соответствует новому ноосферному этапу и, следовательно, обладает более высокой эффективностью и перспективностью при решении современных научных и практических задач.

Основным результатом последовательного применения принципов диалектики системологией является ее более выраженная и более универсальная методологическая направленность. Общепринятый системный подход может функционировать в качестве методологического средства преодоления барьеров конкретного научного познания при возникновении «нештатных ситуаций» или «методологических порочных кругов», но ограничено [12]. Системология же представляет собой, по сути дела, ориентированое на методологическое использование изложение понятий и принципов диалектики. ***Интерпретированная в терминах конкретной науки системология, может выполнять в этой науке методологические функции неограниченно***, «в том числе при выборе таких оптимальных методов исследования частных задач, которые ... откроют возможность рассмотрения объекта исследования с разных позиций без утраты целостного и сущностного о нем представления» [4, с. 42].

Одновременно, в связи с тем, что в настоящее время «проблема организации и улучшения ноосферы приобретает особое значение», системология может дополнительно использоваться как средство «диалектической обработки» создаваемого конкретной наукой знания, то есть как средство «организации» ноосферы [12, с. 191].

Названные преимущества системологии позволяют широко применять ее при исследовании слабоструктурированных и слабоформализуемых проблемных областей, представляющих собой основное поле деятельности современной науки.

Вопросы для повторения

1. Нарисуйте схему самостоятельного научного направления.
2. Опишите причины возникновения системных исследований.
3. Назовите компоненты системных исследований.
4. Дайте определение системного подхода.
5. Что такое системный анализ?
6. Для решения каких проблем применяется системный анализ?
7. Дайте определения принципам системного подхода.
8. В чем отличия традиционного системного подхода от системологии?

Резюме по теме

В данном разделе рассмотрена структура самостоятельного научного направления, с одной стороны, и системных исследований, с другой. Показано, что системные исследования могут в настоящее время рассматриваться как самостоятельное научное направление. Кроме того, прослежена эволюция системного подхода и его принципов; выявлены различия традиционного системного подхода и системного подхода ноосферного этапа развития науки, т.е. системологии.

Тема 2. Моделирование и анализ систем. Основные подходы

Цели и задачи изучения темы

Целью изучения данной темы является ознакомление с основными подходами к моделированию и анализу сложных систем: с традиционным системным (системно-структурным) подходом, с объектно-ориентированным подходом и современным системным подходом, соответствующим ноосферному этапу развития науки (системологией).

При этом ставятся следующие задачи:

- ознакомление с проблемами традиционного системного подхода и системного анализа;
- изучение причин существования проблем традиционного системного анализа;
- освоение особенностей объектно-ориентированного подхода;
- обоснование необходимости интеграции системно-структурного и объектно-ориентированного подходов;
- изучение основных понятий системологии;
- сопоставление системологии и основных научно-практических дисциплин, направленных на рационализацию организационных систем (теории организации, логистики и инжиниринга бизнеса), а также системологии и объектно-ориентированного подхода.

2.1. Традиционный системный подход

2.1.1. Особенности и проблемы традиционного системного подхода и системного анализа

Как известно, современным информационным системам (ИС) и информационным технологиям (ИТ) присуща сложность. Она обусловлена, с одной стороны, «неудовлетворительными способами описания поведения больших дискретных систем», а, с другой стороны, слабой структурированностью и «сложностью реальных предметных областей», из которых, в настоящее время, в основном и исходит заказ на разработку [13, с. 22]. При этом считается, что ***системный анализ*** сложных объектов является тем средством, которое обеспечивает возможность решения различных научных, деловых, управленческих и производственных слабоструктурированных и слабоформализуемых проблем [14, 15].

Некоторые методы системного анализа, оказавшиеся весьма эффективными в самых различных областях человеческой деятельности, возведены в ранг государственных стандартов по анализу и проектированию сложных объектов и процессов. Например: система стандартов Icam Definition в США или SSADM – методология структурного системного анализа и проектирования, стандартизованная в Великобритании.

В систему стандартов Icam Definition [16, 17, 18], в частности, входит стандарт IDEF0 (FIPS183), предназначенный для создания функциональной мо-

дели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции. Данный документ представляет собой оформление (по инициативе Министерства обороны США) в виде стандарта технологии анализа сложных систем SADT (Structured Analysis and Design Technique), разработанной в период с 1969 по 1973 годы группой аналитиков во главе с Дугласом Т. Россом.

Эти методы успешно применяются для решения различных задач, например, при управлении финансами, планировании производства, организации материально-технического обеспечения, построении систем диагностики, автоматизации производства и проектирования, организации системы образования, управлении космическими средствами связи и т.д. [19].

Однако, специалисты по консалтингу [20, 21], в обязанности которых входит применение этих методов на практике, и специалисты по разработке программного обеспечения (ПО) [22] оценивают их следующим образом:

- ***DFD-диаграммы*** (составная часть технологии 3VM) разработаны для проектирования программного обеспечения и ориентированы на системных аналитиков и программистов и не учитывают особенности восприятия менеджерами своей предметной области. Для описания бизнес-процессов менеджерам требуются более мощные выразительные средства, чем описание входов и выходов на диаграммах потоков данных.
- ***SADT-технология*** (стандарт IDEF0) разработана как средство моделирования и анализа любых систем. Однако в настоящее время оказалось, что её выразительных средств недостаточно для моделирования, например ИС. В результате данная технология практически используется относительно редко (менее чем в 10% существующих CASE-средств). Для создания динамических моделей требуется использование дополнительных специальных расширений или других средств, с которыми SADT плохо согласуется.
- ***SSADM-методология*** является специализированным средством систематизации и стандартизации процессов (этапов, задач и документов) создания ИС и не предназначена для анализа и моделирования бизнес-систем и бизнес-процессов. Для обеспечения понимания событий, которые происходят в реальном мире и которыми проектируемая система должна управлять, строится модель пользователя, но не бизнес-системы, так как для этого нет необходимых средств.
- Кроме того, стандарты ***SADT (IDEF0)*** и ***SSADM*** приспособлены только для хорошо специфицированных и стандартизованных «западных» бизнес-процессов, что делает их практически непригодными для отечественного рынка.

Методы системного анализа, основанные на потоковых диаграммах (DFD: предназначенные в первую очередь для моделирования информационных процессов), оцениваются еще и следующим образом: «Диаграммы потоков данных обеспечивают удобное описание функционирования компонент системы, но не снабжают аналитика средствами описания деталей этих компонент, а именно, какая информация преобразуется процессами и как она преобразуется»

[20, с. 53]. «Их практическое применение высокоэффективно, как правило, только для отражения информационных структур бизнес-процесса, оптимизация которого проведена уже другими средствами» [23].

Одной из существенных проблем при использовании методологий моделирования IDEF, по мнению, например, специалистов проектировщиков, является дальнейшее использование результатов в практической работе, например при внедрении или создании корпоративной ИС: «Часто оказывается, что применить результаты длительной работы достаточно сложно» [23, 24]. При этом профессиональные аналитики заявляют о несоответствии SADT требованиям «кибернетического стандарта» на моделирование бизнес-процессов [25].

Методология «моделирования в терминах системы», созданная усилиями специалистов BAAN, ориентирована исключительно на разработку программных продуктов и не удобна для описания самих бизнес-процессов [23, 24].

Кроме того, в литературе отмечаются следующие проблемы традиционных методов системного структурного анализа [15, 26 - 28]:

- методы и результаты решения существуют только для относительно простых классов систем, а для более сложных развиваются методы упрощения;
- несмотря на существующие попытки, не достаточно разработаны методы описания процессов адаптации и эволюции систем в связи со слабой изученностью развивающихся и самоорганизующихся систем;
- при наличии возможности описать (до определенного уровня сложности), что представляет собой процесс или объект, не существует способов показать мотивацию, смысл и взаимосвязь между действиями и сущностями, т.е. почему он именно такой;
- «строгие формальные основания этих систем ведут к крайней скудости выразительных возможностей и ограниченности сферы применения» [28, с. 89].

В [29, 30] приведен обзор существующих методов анализа и моделирования сложных систем, в котором относительно самых конструктивных методов, основанных на формализованном представлении системы, отмечается следующее:

- ***Аналитические методы***. Для сложных систем получить требуемые аналитические зависимости очень трудно. Если это даже удается сделать, то практически невозможно доказать правомерность применения аналитических выражений, т.е. адекватность модели рассматриваемой задаче. Популярные же (в связи с разработанностью теории и знакомством с ними большинства исследователей) схемы дифференциальных уравнений, отражая динамику системы, не отражают возможности восприятия системами входных сигналов и выдачи выходных, что является одной из основных особенностей сложных систем.
- ***Статистические методы***. Далеко не всегда можно получить статистические закономерности или доказать правомерность их применения. Определение или получение репрезентативной выборки часто требует недопустимо больших затрат времени.

- ***Теоретико-множественные представления***. Трудно вводить правила или закономерности, формально используя которые, можно получить новые результаты, адекватные реальным моделируемым объектам и процессам.
- ***Логические методы***. Смысловыражающие возможности ограничены базисом и функциями алгебры логики и не всегда позволяют адекватно отобразить реальную проблемную ситуацию.
- ***Лингвистические, семиотические представления***. Трудно гарантировать правильность получаемых результатов из-за возникающих проблем алгоритмической разрешимости.
- ***Графические представления***. Работают только в сочетании с другими подходами.

В работе [31] отмечается, что анализ и моделирование систем с помощью средств моделирования случайных процессов или теории массового обслуживания «…зачастую носят кустарный характер и уже не соответствуют современным запросам практики. Они приводят, по мере возрастания сложности задач, к значительному увеличению трудоемкости подготовки моделей…. Эти обстоятельства снижают эффективность имитационного моделирования и препятствуют его широкому распространению как инструмента повседневного использования…» [31, с. 202].

Как известно, при использовании традиционных методов системного анализа создается обычно три или более вида моделей (*3-View Modeling – 3VM*: DFD – функциональная диаграмма потоков данных, ERD – информационная диаграмма «сущность-связь», STD – диаграмма переходов состояний [32]) одного и того же объекта, каждая из которых имеет свою понятийную, терминологическую и графическую базу, и при этом применяются разные инструментальные средства [19, 20, 33]. Это обстоятельство приводит к необходимости проведения специального сквозного контроля диаграмм одного или разных типов, т.е. соответственно вертикального и горизонтального ***балансирования диаграмм***, для выявления весьма вероятных ошибок [20]. Данная ситуация обусловлена несовершенством традиционного системного подхода, в рамках которого отсутствуют концептуальные средства, связывающие между собой функцию объекта, его субстанцию (состав элементов), его структуру и динамику процессов.

Рекордисткой в этом отношении является известная инструментальная система анализа и моделирования ARIS Toolset, которая обеспечивает четыре «взгляда» на модель (Процессы, Функции, Данные, Организация) и для каждого взгляда поддерживает три уровня анализа (требования, спецификация, внедрение), каждый из которых состоит из своего комплекта моделей различных типов. Всего инструмент может построить 85 типов моделей, из которых 57 типов предназначено для моделирования процессов. Примечательно, что в работе [34], в которой автор, помимо обзора методов и средств моделирования, явно рекламирует эту систему, подчеркивается необходимость «интеграции результатов моделирования в рамках общего проекта или общей модели».

Существующие методы системного структурного анализа являются либо процедурно-ориентированными, либо ориентированными на данные [19, 20, 22, 33]. Следовательно, результаты, полученные с их помощью, не могут быть эффективно использованы при разработке ПО средствами технологии объектно-ориентированного программирования (object-oriented programming – OOP), которая в настоящее время является основной и самой перспективной.

Более того, установлено (см. например [13]), что ***все методы системного структурного анализа полностью ортогональны принципам объектно-ориентированного проектирования***. Данное обстоятельство обусловлено, например, тем, что традиционные методы системного анализа не обеспечивают выявления иерархии классов предметной области (не используют в рамках своих процедур концептуальных классификационных моделей), а также не поддерживают объектно-ориентированную концепцию инкапсуляции.

Очень показательны для проводимого в данном пункте анализа попытки применения при выполнении объектно-ориентированного проектирования (object-oriented design – OOD) упомянутых выше методов структурного системного анализа (например, 3VM). Относительно конкретных диаграмм, например диаграмм ERD, в [32, с. 24-25] сделаны следующие выводы: «Применение ERD очень выгодно, однако при работе с этими средствами возникают определенные трудности. Во-первых, идентифицированные сущности не всегда соответствуют понятиям области приложения, особенно когда аналитик пытается создать сущности в третьей нормальной форме. Во-вторых, ERD не подходят для идентификации объектов, не хранящих данные; в эту категорию часто попадают, например, объекты, распознающие происхождение событий или осуществляющие функции контроля». В общем, авторы приходят к следующему выводу: «Методы 3VM, весьма полезные для идентификации компонентов или объектов, не позволяют, тем не менее, идентифицировать ***подходящее*** множество объектов для разрабатываемой системы. Процесс обнаружения объектов все ещё управляется мнениями, интуицией и инсайтом» [32, с. 26].

Следствием ортогональности системного анализа и OOD является, таким образом, проблема использования результатов анализа при проектировании объектно-ориентированного ПО. Специалисты отмечают, что на этапе анализа лучше использовать методы 3VM, IDEF и т.д., так как «аналитик имеет дело с бизнес-процессами, по сути, являющимися функциями». Однако, с точки зрения проектирования ПО, конечно, необходимо использовать объектные методы (на основе UML) особенно в случае разработки сложных программных приложений. При этом, естественно, «описание бизнес-процессов должно делаться в том же средстве, в котором на последующих этапах работ будет проектироваться ИС» [35 - 37]. Это и создает противоречие.

В настоящее время, существуют проблемы принципиального характера и в рамках самого объектно-ориентированного подхода, а именно при проведении объектно-ориентированного анализа (object-oriented analysis – OOA).

В соответствии с требованиями этой методологии любую систему на этапе анализа необходимо представить в ***канонической форме***. Эта форма представления системы включает в себя две ортогональных иерархии: ***иерархию***

объектов и ***иерархию классов***. Предполагается, что такая объектно-ориентированная декомпозиция системы, позволяет вскрыть ее полную архитектуру, т.е. структуру объектов и структуру классов. При этом свойства объектов (их поведение и т.д.) в, так называемой, объектной модели, определяет соответствующий класс в иерархии классов, описывающей моделируемую предметную область.

«Объектно-ориентированная методика позволяет значительно повысить качество и продуктивность разработки программного обеспечения. Однако извлечь из нее выгоду можно только тогда, когда правильно определено множество объектов. Подходящее множество объектов для конкретной области приложения обеспечивает повторное применение системы и возможности ее расширения, а также гарантируют качество и продуктивность потенциальных улучшений, присущих объектно-ориентированной парадигме. Без формальных методов определения объектов разработчики программного обеспечения рискуют остаться просто хакерами на объектном уровне» [32, с. 23].

Однако в литературе по объектно-ориентированному подходу отмечается, что «к сожалению, пока не разработаны строгие методы классификации и нет правила, позволяющего выделять классы и объекты. … Как и во многих технических дисциплинах, выбор классов является компромиссным решением» [13, с. 147]. Неудивительно, поэтому, что основной проблемой OOA и OOD считается отсутствие обоснованного метода «выбора правильного набора абстракций для описания заданной предметной области» [13, с. 56], т.е. метода построения иерархии классов или концептуальной классификационной модели предметной области (или ***модели онтологии*** [38]).

Существующие методики OOA на сегодняшний день предлагают эвристические правила идентификации классов и объектов, основанные на опыте классификации в других науках [13], а специалисты по системному анализу дополняют построение диаграмм информационным анализом, основанным на лингвистике (LIA: Linguistic-based Information Analysis) [32].

При этом специалисты по моделированию поведения сложных систем утверждают, что объектно-ориентированный стиль неестественен и не удобен для описания и имитации поведения бизнес-систем. «Для поведенческого моделирования бизнес-систем, особенно на ранних его стадиях, более удобен традиционный процессо-ориентированный структурный стиль мышления с использованием наглядных графо-аналитических выразительных средств» [39, с.8]. По мнению данного автора «эффективное моделирование поведения бизнес-системы может быть осуществлено на основе интеграции процессо- и объектно-ориентированного подходов» [39, с.9].

Специалисты по консалтингу и CASE-технологиям считают, что «диаграммные техники, отражающие специфику объектного подхода, гораздо менее наглядны и плохо понимаемы непрофессионалами» в сравнении с системно-структурными техниками [40, с.44].

Следует отметить, что оценка перспектив развития методов и средств системного анализа и моделирования в работе [41] и по ныне не утратила своего значения. До сих пор разработка эффективных «проблемно-инвариантных»

методов, «допускающих непосредственный перенос в другие прикладные области» вызывает большие трудности. Эти трудности по-прежнему связаны с «усложнением исследуемых систем, их «глобальностью», существенным использованием в них человеческого фактора и, главное, переходом к уникальным проектам», где применение количественных методов анализа в их обычной аналитической форме с применением традиционной логики становится затруднительным. По-видимому, делают вывод авторы названной работы (и это справедливо и в настоящее время), речь идет «не только о способах редукции сложных задач к доступному уровню за счет их жесточайшей специализации, но и о переходе к иным средствам, существенно учитывающим возможности человека-исследователя при описании систем». Решение данной задачи, по мнению авторов, состоит в создании «адаптивных, управляемых проблемно-инвариантных процедур» системного анализа, адекватных характеру изучаемых объектов [41, с. 145-146].

Таким образом, проблемы системного анализа в связи с особенностями объектно-ориентированного подхода могут быть сформулированы следующим образом:

- невозможность моделирования систем высокого уровня сложности, требуемого в настоящее время;
- недостаточный учет динамических аспектов и процессов развития реальных систем;
- ограниченность выразительных возможностей и сфер применения (отсутствие содержательной системной теории целостного объекта);
- невозможность адекватного описания причинно-следственных связей и собственно системных отношений;
- несоответствие требованиям OOA и OOD;
- отсутствие рационального подхода к выбору классов и объектов в ходе системного анализа при разработке объектных приложений.

Это свидетельствует об актуальности создания новых системных методов анализа и моделирования, в том числе для решения проблем объектно-ориентированной методологии, на основе интеграции принципов системного и объектного подходов, а также концептуального классификационного моделирования.

2.1.2. Причины существования проблем традиционного системного подхода и системного анализа

Рассмотрим причины существования недостатков методов традиционного системного анализа и причины несоответствия этих методов требованиям OOD [26, 27, 42].

В первую очередь отмеченные выше недостатки и проблемы, очевидно, обусловлены тем, что сложность таких систем как социальные структуры, бизнес-процессы, человеко-машинные системы, экологические системы, чрезвычайные ситуации и т.п. предъявляет особые требования к методам их описания. Поэтому все большее значение приобретает разработка методов и инструмен-

тальных средств, обеспечивающих существенное повышение эффективности описания, анализа, синтеза и прогнозирования поведения сложных систем. Отсутствие таких средств тормозит решение разнообразных слабоформализованных задач, для решения которых и предназначались изначально известные системные методы.

Недостатки существующих методов системного анализа обусловлены тем, что в нормативных системах, описывающих эти методы, отсутствуют [26]:

- строгая и формализуемая, но адекватная действительности концепция системы, реализующая все принципы системного подхода;
- средства описания специфических содержательных системных свойств и отношений и, в частности, «системного эффекта»;
- системная теория и формальный аппарат, учитывающие принципиальное отличие понятия «система» от понятия «множество», отмечаемое, например, еще в [6].

Последнее обстоятельство, собственно, и не позволяет обеспечить инкапсуляцию выявляемых объектов (необходимую при OOD), так как система (в традиционном системном анализе и системном подходе) рассматривается как множество элементов, свойств, состояний и т.д. [например: 41, 43, 44], внутреннее содержание которого не может быть скрыто, в отличие от действительно системного подхода, акцентирующего внимание на целостности и функционировании системы [5, 6, 45].

Рассмотрим это обстоятельство более подробно.

Сравнительный анализ системного и теоретико-множественного подходов однозначно свидетельствует о том, что эти подходы принципиально противоположны и ни один из них не сводим к другому, что обусловлено, в частности, следующим [6]:

- Во-первых, в концепции множества изначально заложена первичность элемента (части) по отношению к множеству (целому). Множество существует тогда и только тогда, когда тем или другим образом заданы его элементы. В системной же концепции первичным является понятие системы (целого), которая уже потом может быть (а может и не быть) представлена в виде совокупности взаимодействующих частей.
- Во-вторых, теоретико-множественный подход характеризуется абсолютной неразборчивостью, т.е. позволяет рассматривать как одно множество любую совокупность любых явлений (с учетом известных парадоксов). Системный же подход претендует на рассмотрение действительности (предметной области) в виде естественно взаимодействующих системных образований, что накладывает определенные ограничения на представление совокупности явлений в виде одной системы.
- В-третьих, теоретико-множественный подход (как следует из выше сказанного) характеризуется гносеологичностью, так как реальные объекты не имеют теоретико-множественной природы. Системный же подход, по своему замыслу, ориентирован на описание целостной природы реальных объектов и, таким образом характеризуется онтологичностью, так как реальные объекты имеют системную природу.

Применение, следовательно, для формального описания системных отношений (методов и процедур анализа систем) аппарата теории множеств или другого, сводимого к теоретико-множественному, фактически сводит на нет специфические особенности и преимущества системного подхода. Тем не менее, например, в стандарте IDEF0 (FIPS183), система определяется как совокупность взаимосвязанных и взаимодействующих частей, выполняющих некоторую полезную работу, т.е. в теоретико-множественном смысле [46 - 48].

Не удивительно, что сложные явления и процессы (т.е. то, что действительно имеет системную природу) оказываются «не по зубам» традиционному системному анализу. Все что формализовано, на сегодняшний день, под вывеской системного подхода и системного анализа могло быть сделано в принципе под другой (какой-либо теоретико-множественной), так как фактически сделано обычными традиционными формальными средствами, не описывающими специфические системные отношения.

По поводу роли формального аппарата и математических методов в системных исследованиях отмечается, например в [14, 15, 19], что «основное содержание системного анализа заключено не в формальном математическом аппарате, описывающем «системы» и «решение проблем» и не в специальных математических методах, … а в его концептуальном, т.е. понятийном аппарате, в его идеях, подходе и установках». При этом в работе [49, с. 73-74] подчёркивается, что «реальные системы не полностью поддаются описанию с помощью математических моделей» и что аналитические (т.е. формальные) методы непригодны для изучения живых и, следовательно, социальных (организационных) систем. Кроме того, «использование математики переносит акцент с содержания на структуру явления» [49, с. 86]. При этом теоретические системные построения, основанные на результатах физико-математических наук, автор упомянутой работы называет ***теориями жёстких систем***, применение которых к экономическим и организационным системам позволяет создать количественные модели чрезвычайно бедные, однако, по своему содержанию [49, с. 103]. Более того, там же подчёркивается, что если теория связана только с понятиями структуры и цели и не связана с понятиями субстанции и содержания, то бесполезно ожидать появления конкретных полезных приложений такой системной теории [49, с. 103].

При применении методов математической статистики зачастую предполагается независимость, одинаковая распределённость и нормальность используемых совокупностей случайных величин [50, 51]. Однако, «данные предположения, как правило, не выполняются, и это обстоятельство может приводить к потере точности и достоверности результатов моделирования …. Хотя в моделировании существуют приемы сведения данных к виду, пригодному для использования традиционных методов статистики, эти методы часто носят эвристический характер либо приспособлены для изучения частной модели» [50, с. 19].

«Трудности практического использования моделей математического программирования связаны, прежде всего, с обеспечением полноты, точности и достоверности исходных данных, необходимых для модели, с учетом динамики

функционирования системы и с многокритериальным характером выбора варианта структуры [52, с. 16].

При этом весьма показательным является не принятие стандарта математического моделирования – IDEF2, которое «было вполне естественным, так как при реализации математической модели приходилось либо жертвовать ее точностью, либо ... самой возможностью что-то моделировать, ввиду чего никогда нельзя было с полной уверенностью говорить о соответствии математической модели функциональной» [23].

Таким образом, именно использование чисто формальных математических средств является одной из причин определенной ограниченности существующих методов системного анализа организационных систем и их несоответствия содержательному по своей природе объектно-ориентированному подходу. С этой точки зрения весьма существенным для данного исследования является введение еще В.М. Глушковым понятия ***обобщённых динамических систем*** (организмы, предприятия, государства и т.д.), которые принципиально не могут быть описаны классической математикой [53, 54].

Кроме того, объекты системного анализа, как правило, являются не строго и не точно определенными. Это позволяет оставаться актуальным известному предупреждению Н. Винера о том, что применение точных формул к вольно определяемым понятиям есть не что иное, как обман и пустая трата времени [55], а также резкому суждению А. Эйнштейна о том, что математика может доказать что угодно и использоваться как отличнейшее средство водит за нос даже самого себя, а главное состоит в содержании, а не в математике [56].

Почему же до сих пор не существует теории систем, как такой же фундаментальной и самостоятельной теории, какой является теория множеств? Почему же описывая и моделируя системы, зачастую, используют аппарат именно теории множеств?

Среди, очевидно, целого ряда известных причин, обуславливающих данную ситуацию, обратим внимание на следующее [27]. Любая научная теория, особенно обслуживаемая формальной системой, для выполнения своих многообразных функций (информативной, систематизирующей, прогностической, объяснительной) должна обладать средствами, позволяющими осуществлять процедуры ***анализа*** и ***синтеза*** описываемых данной теорией объектов. Совершенно очевидно, что теоретико-множественный подход в лице теории множеств такими средствами обладает, что и подвигает, так сказать «пользователей», на применение этой теории для решения конкретных задач, в частности задач системного анализа.

А как обстоят дела в этом смысле у «традиционного» системного подхода?

Существующие варианты системных подходов (их анализ, например, в работе [57]) используют такое понятие системы (в настоящее время более 40 определений; например уже в работе [58] собрано 35), которое, не обуславливает (не задает) определенной возможности для проведения операций анализа или синтеза ***объектов как систем***, т.е. с учетом специфических системных отношений. Данные операции либо вообще остаются без определения, либо опреде-

ляются с помощью теоретико-множественных средств с потерей возможности целостного представления системы. А, как известно, «системные представления, построенные при игнорировании признака целостности, оказываются неэффективными или даже дают заведомо ошибочные результаты» [57, с. 73].

Например, в одном из самых первых вариантов системного подхода система определялась как средство решения проблем, но конструктивных подходов к анализу и синтезу таких средств не предлагалось [59]. В более поздних вариантах система рассматривается как упорядоченное определенным образом множество взаимосвязанных между собой компонент той или иной природы, характеризующееся единством (целостностью), которое выражается в интегральных свойствах и функциях множества [60, 61]. При этом заданная возможность теоретико-множественного анализа и синтеза таких систем (как множеств) исключает конструктивное определение специфической целостности системы и причины появления её интегральных, т.е. собственно системных свойств. В теории организации, например, система определяется как целое, созданное из частей и элементов, для целенаправленной деятельности. При этом системе приписывается стремление к сохранению структуры, потребность в управлении и сложная зависимость свойств от входящих в нее элементов [62]. Все эти характеристики, однако, имеют в рамках данной теории исключительно лингвистическое описание.

Совершенно очевидно, что в названных концепциях способы анализа и синтеза систем с учетом их специфических системных отношений не заданы, не определены. Такая же ситуация существует и в рамках формализованных системных построений. Практически все формальные определения системы или непосредственно, или после своего раскрытия имеют теоретико-множественный характер, в рамках которого либо затруднено проведение анализа и синтеза систем, либо фактически не учитывается целостность этих систем. Довольно большая коллекция таких определений представлена, например в работах [29 и 41].

В классическом труде [8], описывающим так называемый феноменологический системный подход, исходное целостное представление системы в виде декартова произведения множеств входных (X) и выходных (Y) объектов: $S \subset X \times Y$, является представлением «черного ящика», процедура анализа которого как системы, естественно, не определена. Для обеспечения анализа и синтеза систем в названной работе вводится функциональное представление системы: $S: X \rightarrow Y$. Уточнение последнего представления осуществляется путем введения понятия глобальной реакции системы: $R: (C \times X) \rightarrow Y$, которое, в свою очередь, зависит от множества внутренних состояний (C) системы. Введение в рассмотрение множества внутренних состояний обеспечивает возможность анализа и синтеза функциональных систем в представлении [8] теоретико-множественными средствами, однако, ценой отказа от целостного системного представления. Последнее обстоятельство обусловлено тем, что для определения системы (при таком ее понимании) исследователь, по сути дела, обязан рассматривать внутренности данной системы и не ее целостность.

Таким образом, включение в формальное определение системы кроме (или вместо) множества ее элементов множества свойств или состояний, а также различных вариантов отношений между ними ничего не меняет по существу данной проблемы. Например, в работе [31] система моделируется с помощью, так называемых, ***кусочно-линейных агрегатов***, образовываемых с помощью трех множеств: входов, выходов и состояний. Решая, безусловно, с помощью такого подхода ряд практических задач для некоторых классов систем, тем не менее, невозможно обеспечить целостное рассмотрение системы и «инкапсуляцию» ее свойств, необходимую в ходе OOD.

В не менее классической работе [9], в которой система вообще рассматривается не как реальная вещь, а как абстрагирование или отображение некоторых свойств реального объекта, формальное представление системы осуществляется путем задания множества свойств объекта и их проявлений, а также так называемой «базы» и множества ее элементов. При этом база может состоять из элементов исследуемого объекта (системы) со всеми вытекающими отсюда и отмеченными выше достоинствами и недостатками теоретико-множественного подхода. В других случаях база может представлять собой некоторые контекстные ситуационные характеристики и, тогда, процедуры анализа и синтеза таких систем не определяются.

В работе же [63] в понятие системы включается только совокупность свойств объекта, но не сам объект. Таким образом, системность отрывается от субстанции, что делает невозможным моделирование и анализ ее реализации, без которой реальное существование системы становится проблематичным.

Даже если применяются, чисто логические методы исследования систем или мощный алгебраический аппарат, сама система все равно определяется через понятие множества каких-либо входящих в систему сущностей [например: 64, 65]. Включение же в определение системы понятия среды [например: 66 - 69] фактически еще более усложняет процедуры анализа и синтеза объектов как систем, так как затрудняет определение границ системы.

При этом необходимо иметь в виду, что представление системы в виде множества ее взаимосвязанных элементов (свойств, состояний и т.п.) рано или поздно приводит, в конце концов, к паре множеств: множеству элементов (свойств, состояний и т.п.) и множеству связей (отношений), заданному на первом множестве. Это, в свою очередь, приводит к пониманию системы как модели, т.е. как некоторому средству описания реально существующих объектов. В настоящее же время общепризнанно, что системность является неотъемлемым атрибутом действительности, т.е. имеет не гносеологический, а онтологический статус [например: 6 и 70].

Таким образом, предполагаемая в традиционных подходах к системе её структурированность является не более чем декларацией, так как путей конкретной реализации структурирования объекта как системы (а не как множества) не указывается. При этом в теории множеств возможность анализа задана и однозначно определена. Она задана в самой концепции множества, согласно которой множество задается через его элементы, т.е. между множеством и его элементами существуют, хотя и примитивные, но однозначно оговоренные от-

ношения. Таким образом, пути и способы теоретико-множественного анализа оказываются четко определенными, что находит свое воплощение в формальном аппарате, в первую очередь, в операции «∈». Отношение же целое – часть (система – подсистема (элемент)) в упомянутых системных подходах не определено. Утверждается только факт его существования. Это приводит к тому, что способ анализа системы определяется, по сути дела, прихотью аналитика, обусловленной, конечно, решаемой задачей, но не приобретающей от этого определенности и объективности [30].

Например, во множестве «Города России» любой исследователь в ходе анализа будет выявлять именно населенные пункты, имеющие статус города, а не что-либо ещё; во множестве «Животные заповедников» – животных (не их части, не их сообщества и т.п.), про которых известно, что они живут в заповедниках. В ходе же системного анализа, например, системы «Город» или системы «Биоценоз» разные аналитики выявят разные части (подсистемы), в зависимости не только от целей анализа, но и от своих субъективных предпочтений [например, 71]. Причем, с точки зрения именно традиционного системного подхода, все они будут иметь для своих действий одинаковые основания, т.е. полное их отсутствие, так как путь (способ) анализа системы в рамках упомянутых и наиболее распространенных системных концепций априорно не определен. В публикациях аналитиков об этом так прямо и говорится: «Это значит, что мы можем увидеть, выделить и изучить в городе столько систем, сколько захотим или сколько требует реальная практика управления» [72, с. 93].

По отношению к процедуре синтеза в традиционных системных подходах (в сравнении с теоретико-множественным) существует аналогичное положение. В рамках теоретико-множественного подхода отношение части к целому строго задано заранее оговоренными правилами «сборки» частей в целое в виде соответствующих операций над множествами. В рамках же обсуждаемых системных подходов процедура синтеза системы в нечто еще более целое опять становится зависящей от субъекта исследования, его понимания текущей ситуации, так как простое декларирование функционирования системы в среде или подчиненности системы (выходов системы) некоторой цели, если не оговариваются отношения со средой или что это за цели, чьи они, откуда берутся, остается не более чем лозунгом, ровным счетом ничего не дающим для обоснованного проведения операции синтеза. Данная ситуация в системных исследованиях даже получила свое особое наименование: ***системный эффект*** или ***эмерджентность*** свойств целого. При этом признается, что в ходе синтеза целого (системы) из его частей (подсистем) возникают принципиально новые свойства, но механизм синтеза или, как его еще называют, «***алгоритм сборки***» остается до сих пор тайной за семью печатями [например 73]. И в этом нет ничего удивительно, так как в наиболее распространенных системных подходах процедура синтеза (путь соединения объектов как систем, а не как множеств) не задана в принципе.

Например, множество «Города России» можно однозначно описать (синтезировать) путем объединения множеств «Города области…»; множество «Животные заповедников» – путем объединения множеств «Животные запо-

ведника №…». Синтезирование же системы «Город» из подсистем населенного пункта такого вида или системы «Биоценоз» из частей, представляющих собой различные виды организмов, проживающих на данной территории, в значительной степени является эвристической процедурой, которую разные исследователи могут и будут выполнять по разному.

Имеющиеся попытки алгоритмизировать процедуры анализа и синтеза в системных исследованиях сводятся к подмене термина «анализ» более красивым термином «***декомпозиция***», а термина «синтеза» – термином «***агрегирование***». При этом о самих процедурах декомпозиции и агрегирования ничего не говорится [7].

В заключении следует отметить, что ни один метод традиционного системного анализа не использует понятия класса при осуществлении своих процедур и построении моделей, т.е. в принципе не использует концептуальных классификационных моделей, применение которых обязательно при осуществлении ООА [например 13, 74]. Поэтому при использовании традиционных системных методов для проведения OOD все равно приходится выходить за их рамки и использовать дополнительные средства [32, 75].

В общем сложившаяся ситуация хорошо охарактеризована известным специалистом по CASE-технологиям, консалтингу и реинжинирингу Г.Н. Коляновым: «Однако разработка программных средств поддержки реорганизации бизнес-процессов вызывает значительные затруднения по причинам отсутствия единого теоретического аппарата и достаточно полных методических основ системного анализа бизнес-процессов, общих математических моделей бизнес-процессов и формальных методов их создания и исследования, а также программных средств их реализации» [40, с.11].

Таким образом, основными причинами существования проблем и недостатков методов традиционного системного анализа и их несоответствия объектной парадигме является теоретико-множественный и вообще чисто формальный подход к понятию системы, а также отсутствие в их арсенале инструментальных средств классификационного моделирования. Следовательно, справедливыми, пока, остаются высказанные в работе [76] слова о непреодолимых трудностях, с которыми сталкиваются исследователи при создании комплексных социально-экономических моделей, т.е. моделей организационных систем, при использовании традиционных формальных средств. По справедливому мнению авторов: «Для этого необходимы не нынешние способы агрегирования, а некое целостно-укрупненное отображение внутренне сложных полиструктурных элементов посредством иных методов» [76, с. 262].

2.2. Объектно-ориентированный подход

2.2.1. Особенности объектно-ориентированного подхода

Одним из самых распространенных видов моделирования в настоящее время является моделирование с применением различных информационных технологий, т.е. построение и использование компьютерных моделей в виде

прикладных программ различного назначения (программных приложений). Создание этих приложений может осуществляться традиционными методами, основанными на алгоритмах, процедурах и данных, а также более современными – объектно-ориентированными, основанными на концептуальном моделировании предметной области. При этом общепризнанно, что «если код приложения генерируется не на основе описания предметной области, то невозможно построить эффективное приложение со сложной бизнес-логикой» [77, с. 215].

Рассмотрим в соответствии с [13, 78 – 87] основные понятия ***объектно-ориентированной методологии*** создания программного обеспечения (ПО), а также средства стандартного ***унифицированного языка объектного моделирования (UML – Unified Modelling Language)***.

Особенностью данной методологии является программное описание не процедур, необходимых для решения задачи, а тех сущностей, которые участвуют в этих процедурах и их обеспечивают. Дело в том, что процедуры осуществляются не сами по себе, не каким-то абстрактным образом, а путем взаимодействия некоторых вполне конкретных объектов, в соответствии с имеющимися у них свойствами. И если эти объекты и их свойства описаны в программе, то остаётся только дать этим объектам возможность взаимодействовать. Процедуры, которые необходимо выполнить для решения задачи, будут осуществляться как результат этого взаимодействия.

Объектно-ориентированное моделирование и разработка ПО осуществляются путем последовательного выполнения [13, 78]:

- ***Объектно-ориентированного анализа*** **(*object-oriented analysis – OOA*),** при котором моделируемая и разрабатываемая системы анализируются с точки зрения ***классов*** и ***объектов*,** выявленных в предметной области.
- ***Объектно-ориентированного проектирования*** **(*object-oriented design – OOD*)**, при котором путем ***объектной декомпозиции*** создается ***объектная модель*** разрабатываемой системы.
- ***Объектно-ориентированного программирования*** **(*object-oriented programming – OOP*)**, при котором программа представляется в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют ***иерархию наследования*.**

В соответствии с требованиями OOA и OOD для построения объектной модели сложной системы её необходимо представить в ***канонической форме***. Эта форма представления системы включает в себя две ортогональных иерархии: ***иерархию классов*** и ***иерархию объектов*** [13, 78]. Предполагается, что такая объектно-ориентированная декомпозиция системы, позволяет вскрыть ее полную архитектуру, т.е. структуру классов и структуру объектов.

При этом если объект, как экземпляр класса, соответствует конкретному предмету или явлению, определенному во времени и в пространстве, то класс – абстракции существенного в объекте. Таким образом, в рамках объектно-ориентированного подхода ***класс*** рассматривается как множество объектов (экземпляров), имеющих общую структуру и общее поведение. ***Объект*** же «представляет собой конкретный опознаваемый предмет, единицу или сущность (реальную или абстрактную), имеющую четко определенное функциональное на-

значение в данной предметной области» [13, с. 92].

При этом свойства (***состояние*** и ***поведение***) объектов, как экземпляров классов, в объектной модели, определяет соответствующий класс в иерархии классов, описывающей моделируемую предметную область. Главной же задачей OOA и OOD считается «выбор правильного набора абстракций для описания заданной предметной области» [13, с. 56], что подчеркивает важность концептуального классификационного моделирования в процессе объектно-ориентированной разработки.

Концептуальная база методологии объектно-ориентированного моделирования включает в себя четыре главных взаимосвязанных понятия [13, 78 – 82]: ***абстрагирование***, ***иерархия***, ***инкапсуляция*** и ***модульность***.

Абстрагирование – способ выделения существенных характеристик некоторого объекта (абстракций), отличающих его от всех других видов объектов и, таким образом, четко определяющих его концептуальные границы.

Иерархия – способ упорядочения абстракций (классов) по уровням.

Инкапсуляция – способ отделения элементов объекта (класса), определяющих его устройство, от элементов, определяющих его поведение (т.е. отделение ***реализации*** от ***интерфейса***).

Модульность – способ разложения системы на связанные, но относительно самостоятельные части (модули).

Объектно-ориентированное моделирование, проектирование и программирование в обязательном порядке основано на использовании названных понятий при описании классов и объектов.

Описание класса включает ***атрибуты***, с помощью которых характеризуется состояние объектов данного класса, и ***операции***, с помощью которых характеризуется функционирование (поведение) объектов данного класса. При этом разграничивается внешний облик класса (спецификация или интерфейс) и его внутреннее устройство (реализация). Главное в интерфейсе – объявление операций, поддерживаемых экземплярами класса. В него также могут входить объявления других классов, переменных, исключительных ситуаций, уточняющих абстракцию, которую класс должен выражать. Реализация, как правило, состоит в определении операций, объявленных в интерфейсе класса.

Между классами могут существовать следующие отношения:

- ***Наследование*** – такое отношение между классами, при котором класс повторяет описание состояния и поведения суперкласса (одного – одиночное наследование; нескольких – множественное наследование). Данное отношение является основным при выявлении иерархии классов предметной области. Узкие, специализированные классы в иерархии, от которых создаются экземпляры (объекты), называются ***конкретными классами***. Общие классы, от которых экземпляры не производятся, называются ***абстрактными классами***. Самый общий класс в иерархии классов называется ***базовым*** или корневым.
- ***Ассоциация*** – отношение семантической зависимости, показывающее какие роли классы играют друг для друга. Ассоциации различаются по мощности: «один-к-одному», «один-ко-многим», «многое-ко-многим».

– ***Агрегация*** – отношение, соответствующее отношению «часть-целое» между объектами данных классов.
– ***Использование*** – может рассматриваться как разновидность отношения ассоциации, при котором одна из сторон (***клиент***) пользуется услугами или ресурсами другой стороны (***сервера***). Кроме того, использование может рассматриваться как один из аспектов отношения наследования, так как подкласс, наследуя состояние и поведение класса, выступает в роли его клиента. Так же клиентом класса является его экземпляр (объект), который использует атрибуты и операции данного класса.

При этом операция рассматривается как услуга, которую класс может предоставить своим клиентам. Как правило, эти операции бывают следующих видов:

– ***Модификатор*** – операция изменения состояния объекта.
– ***Селектор*** – операция считывания состояния объекта (без изменения состояния).
– ***Итератор*** – операция, организовывающая доступ ко всем частям объекта в строго определенной последовательности.
– ***Конструктор*** – операция создания и/или инициализации объекта.
– ***Деструктор*** – операция, освобождающая состояние объекта и/или разрушающая сам объект.

Описание объекта включает в себя описание его состояния, которое характеризуется перечнем атрибутов, соответствующих классу, экземпляром которого является данный объект, и их текущими значениями. Кроме того, описание объекта включает также описание его поведения (функции), которое характеризуется ***методами***, реализующими операции класса, экземпляром которого является данный объект.

Объединяя понятия состояния и поведения объекта, можно ввести понятие ***роли*** или ***ответственности*** объекта. Ответственность объекта имеет две стороны – знания, которые объект поддерживает, и действия, которые объект может исполнить. Она выражает смысл его предназначения и место в системе. Ответственность понимается как совокупность всех услуг и всех контрактных обязательств объекта. Следовательно, состояние и поведение объекта определяют его роль в системе, а она, в свою очередь, необходима соответствующему классу для выполнения его ответственности.

При выявлении структуры объектов моделируемой системы между объектами устанавливаются различные ***связи*** (посредством передачи сообщений), представляющие собой «физическое или концептуальное соединение между объектами». Связи обозначают равноправные или «клиент-серверные» отношения между объектами. «Наиболее желательной является функциональная связность, при которой все элементы…тесно взаимодействуют в достижении определенной цели» [13, с. 139]. Участвуя в связях, объект может выполнять одну из трех ролей:

– ***актора*** – при которой объект может воздействовать на другие объекты, но сам никогда не подвергается их воздействию (активный объект);

– ***сервера*** – при которой объект может только подвергаться воздействию со стороны других объектов, но никогда не выступает в роли воздействующего объекта (пассивный объект);

– ***агента*** – при которой объект может быть и активным, и пассивным.

Кроме того, между объектами могут быть выявлены иерархические отношения, а именно отношение «часть-целое», т.е. агрегация. При этом, идя от целого (агрегата), можно придти к его частям (атрибутам). Агрегация означает физическое или концептуальное вхождение одного объекта в другой.

Классы и объекты выявляются (обнаруживаются, открываются) на этапе OOA и в совокупности составляют ***словарь предметной области***. На этапе OOD могут понадобиться новые классы, объекты и механизмы их взаимодействия для обеспечения поведения, требуемого моделью, которые приходится уже изобретать.

При проведении OOA и OOD необходимо осуществлять декомпозицию области приложения и разрабатываемой информационной системы. Для обеспечения такой декомпозиции в соответствии с объектно-ориентированным подходом (ООП) используются абстрагирование и иерархия [13, 78]. Однако в настоящее время задача объектно-ориентированной декомпозиции и выделения классов и объектов не имеет законченного формального решения. «Объектно-ориентированная методика позволяет значительно повысить качество и продуктивность разработки программного обеспечения. Однако извлечь из нее выгоду можно только тогда, когда правильно определено множество объектов. Подходящее множество объектов для конкретной области приложения обеспечивает повторное применение системы и возможности ее расширения, а также гарантирует качество и продуктивность потенциальных улучшений, присущих объектно-ориентированной парадигме. Без формальных методов определения объектов разработчики программного обеспечения рискуют остаться просто хакерами на объектном уровне» [32, с. 23].

При этом в литературе по объектно-ориентированному подходу отмечается, что «к сожалению, пока не разработаны строгие методы классификации и нет правила, позволяющего выделять классы и объекты. … Как и во многих технических дисциплинах, выбор классов является компромиссным решением» [13, с. 147]. Существующие методики OOA на сегодняшний день предлагают эвристические правила идентификации классов и объектов, основанные на опыте классификации в других науках. При этом разные авторы предлагают различные наборы базовых классов для объектного моделирования и проектирования. Например, в работе [88] предлагается набор: Осязаемые предметы, Роли, События, Взаимодействие. В работе [89] предлагается следующий набор: Люди, Места, Предметы, Организации, Концепции, События. В работе [90] предлагается такой набор: Структуры, Другие системы, Устройства, События, Разыгрываемые роли, Места, Организационные единицы.

Наличие разных вариантов и отсутствие каких-либо обоснований этих вариантов свидетельствует о том, что, на сегодняшний день, системные методы классифицирования, позволяющие обоснованно выделять классы и объекты, в объектно-ориентированной технологии пока еще не применяются.

2.2.2. Необходимость интеграции объектного и системного подходов

В настоящее время компонентная методология разработки ИС является базовой методологией создания программного инструментария информационно-аналитического сопровождения деятельности организационных систем. При этом, являясь развитием объектно-ориентированного подхода, данная методология использует все его понятия и принципы. Однако, объектно-ориентированный анализ и проектирование (OOAD) не располагают какими-либо собственными методами анализа и моделирования предметной области, для которой разрабатывается программное приложение, так как изначально предназначены для моделирования ПО. Отсутствуют даже методы простого выявления необходимых для моделирования классов и объектов. Использование же широко рекламируемого языка UML не изменяет ситуации, так как, по справедливому замечанию специалистов (например, М. Фаулера), это только язык, а не метод [81].

Одним из оснований для подобных высказываний является определение понятия «метод», предложенное, например, в работе [91]. Кратко оно может быть представлено в следующем виде: «Метод – есть система правил и приемов достижения определенных результатов, ***исходящая из знания закономерностей исследуемого предмета***, помогающая исследователю выбрать существенное, наметить путь восхождения от известного к неизвестному». Данное определение позволяет утверждать, например, что карточки CRC (Класс – Ответственность – Кооперация) не могут рассматриваться как метод выявления классов предметной области. Подобные карточки являются способом представления информации, который призван облегчить ее последующий анализ. Никакой связи с закономерностями предметной области или каких-либо подходов к выявлению в предметной области существенного они не имеют. Это то же самое, что библиотечная картотека. Сама по себе она есть только удобный способ хранения информации. Для анализа и классификации этой информации применяются совершенно другие методы.

По той же самой причине Рациональный Унифицированный Процесс разработки ПО, как модель жизненного цикла ПО, может рассматриваться только как метод его разработки и управления ей, но не как метод анализа и моделирования той предметной области, для решения проблем в которой и проводится разработка. Доказательством такого понимания может служить, например, работа [92], в которой описание технологического процесса моделирования предметной области, названное «моделированием производства», содержит подробные объяснения необходимости этого моделирования, описание исполнителей и артефактов, общее описание технологического процесса, показывающее место модели предметной области, общие рекомендации по переходу к модели программной системы и т.д. Вопрос моделирования предметной области как таковой не рассматривается и везде остается «одной строчкой». Как моделировать производство так и не объясняется. При этом предложение использовать для него средства моделирования ПО не обосновывается.

Кажущееся естественным применение методов системного анализа, изна-

чально предназначенных для моделирования производства, в рамках OOAD оказывается малоэффективным. Данное обстоятельство обусловлено тем, что методы традиционного системного анализа (именуемого в литературе также структурным или системно-структурным) «полностью ортогональны принципам объектно-ориентированного проектирования» [13, с. 161]. Специалисты по системному анализу приходят к однозначному выводу, что при использовании традиционных системных методов для проведения OOAD все равно приходится выходить за их рамки и использовать дополнительные средства, так как современные методы такого анализа «не позволяют, тем не менее, идентифицировать подходящее множество объектов для разрабатываемой системы. Процесс обнаружения объектов все еще управляется мнениями, интуицией и инсайтом» [32, с. 26].

Отмеченная выше ортогональность обусловлена тем, что традиционные методы системного (или системно-структурного) анализа не связаны с выявлением иерархии классов предметной области, т.е. не решают задачи концептуального классификационного моделирования (ККМ). Таким образом, ***традиционный системный анализ не поддерживает такие обязательные принципы объектного подхода, как абстрагирование и иерархию***.

К сожалению, в литературе по OOAD также, кроме сетований на то, что «рецептов для поиска классов не существует», не предлагается путей решения задачи ККМ, хотя и утверждается, что это основная задача OOAD [например, 13 и 78]. Например, вся литература от Rational Software (под редакцией «трех друзей») ограничивается только рекомендациями общего характера. При этом подчеркивается, что они не знают ни методик построения классификаций, ни даже самого понятия «хорошая классификация». Попытки строить классификации имеются, но они осуществляются на уровне интуиции и экспертных знаний специалистов конкретной предметной области без привлечения какого-либо методического обеспечения, представленного, например, в работах [91 или 93].

Кроме того, ***традиционные методы системного (или системно-структурного) анализа не поддерживают объектно-ориентированную концепцию инкапсуляции*** (отделение интерфейса объекта от его реализации). Последнее обстоятельство, в свою очередь, обусловлено тем, что система в традиционном системном анализе и системном подходе (как уже было отмечено выше) всегда рассматривается как множество или элементов, или свойств, или состояний и т.д. В концепции же множества изначально заложена первичность элемента (части) по отношению к множеству (целому). Множество существует тогда и только тогда, когда тем или другим образом заданы его элементы. Теоретико-множественное понимание системы, собственно, и не позволяет обеспечить в ходе традиционного системного анализа инкапсуляцию выявляемых объектов (необходимую при OOAD), так как множество есть явление, внутреннее содержание которого не может быть скрыто. Сущностью же действительно системной концепции («pure system»), на самом деле, является выделение в первую очередь целого, которое уже потом будет (а может и не будет) представлено в виде совокупности взаимодействующих частей (подсистем). На это давно уже обращали внимание ведущие методологи системных исследований

[например, 6], однако до сих пор это не нашло практического воплощения в технологиях и инструментах системного анализа (см. далее).

Необходимо подчеркнуть также, что методы традиционного системного анализа и объектно-ориентированный анализ направлены на выявление различных структур исследуемой системы. В системном (системно-структурном) анализе речь идет о функциональной структуре (вход, процесс, выход, связующие потоки) без внимания к реализующим эти функции объектам (субстанции). В OOAD речь идет о структуре объектов (и классов) системы. Функциональность (поведение) этих объектов рассматривается с точки зрения ответственности компонент ПО, так как объектный подход есть результат развития технологии программирования, а не анализа и моделирования действительности. Потеря при этом способности адекватно отображать соотношения вход-выход и потоковые процессы является его (объектного подхода) большим недостатком, резко снижающим эффективность моделирования реальных (а не программных) явлений [37]. Дело в том, что реальная (не виртуальная) действительность целиком и полностью состоит из проточных объектов, постоянно преобразующих входные потоки в выходные. Например, реальный сервер для своего существования должен поддерживаться не только потоками входящей энергии и запасных частей, но и потоками данных, которые кто-то как-то должен периодически туда добавлять, и потоками исполняемого кода, необходимого для обеспечения его функционирования в постоянно меняющихся условиях. Только тогда в ответ на запрос он сможет удовлетворить клиента. Иначе, его просто выключат и выбросят.

Таким образом, сложившаяся в настоящее время (и, безусловно, перспективная) практика разработки сложного ПО средствами объектно-ориентированного (или компонентного) подхода, а также отсутствие методов анализа и моделирования предметной области в рамках OOAD привели к возникновению узкого места, требующего создания таких методов. При этом недостатки объектно-ориентированной методологии (с точки зрения моделирования предметной области) и ее ортогональность с существующими методами системно-структурного анализа (способными в принципе компенсировать эти недостатки) позволяют говорить о существовании проблемы синтеза системного и объектно-ориентированного подходов.

2.3. Системология – системный подход ноосферного этапа развития науки

2.3.1. Основные понятия

Рассмотрим кратко основные понятия функциональной системологии, описанные в работах [2, 5, 27, 45, 94 - 96].

Система – функциональный объект, функция которого обусловлена функцией объекта более высокого яруса (функцией ***надсистемы***).

Функция системы – роль, предназначение системы в надсистеме, которая проявляется в наличии ***функциональных связей*** рассматриваемой системы.

Функциональные связи системы – связи с другими системами в конкретной надсистеме.

Поддерживающие связи системы – функциональные связи ее подсистем. Они создают структуру системы. При этом подсистемы находятся в узлах этой структуры и поддерживают функционирование (***функциональные связи***) системы.

Подсистема – функциональный объект нижнего яруса иерархии системы, составляющий вместе с другими объектами ее ***субстанцию***.

Таким образом, функции подсистем обусловлены функцией системы, функция системы – функцией надсистемы и т.д.

Внешняя детерминанта системы (***функциональный запрос надсистемы***) – явление обуславливания функции системы функцией надсистемы.

Внутренняя детерминанта системы – в действительности проявляемая общая функция системы (ее функционирование). Эта детерминанта определяет функции подсистем (частные функции системы) и их взаимосвязи, т.е. субстанциальные и структурные характеристики системы.

Адаптация системы к запросу надсистемы – приближение внутренней детерминанты системы к ее внешней детерминанте. Критерием адаптированности является отношение между возможностями системы и функциональными требованиями надсистемы. Совершенной или оптимально адаптированной является система, у которой внутренняя детерминанта равна внешней.

Свойство системы (или ***валентность***) определяется как способность поддерживать (при определенных условиях) связи одних видов и препятствовать осуществлению связей других видов. Любая же ***связь*** рассматривается как проявление между системами процесса обмена (т.е. потока) элементами, представляющими собой субстанции определенных глубинных ярусов связанных систем. При этом принято говорить об ***экстенциальных*** валентностях, если связи в действительности существуют, и об ***интенциальных*** (или ***потенциальных***) валентностях, если связей в действительности нет. В первом случае способности системы проявлены как реально существующие свойства-связи, во втором случае свойства системы не проявлены, но они существуют как способности (возможности).

Таким образом, ***функциональное свойство*** системы – есть свойство, которым обязательно должна обладать система для выполнения своих функций, т.е. способность поддерживать связи (потоки), на основе которых протекают важные для надсистемы взаимодействия системы с окрестностными системами. ***Поддерживающее свойство*** системы – свойство, необходимое для поддержания и обеспечения устойчивости функциональных свойств, т.е. способность поддерживать связи (потоки), служащие средством внутреннего поддержания, стабилизации функциональных свойств (связей).

В зависимости от пути проявления целостности, как основного признака системности, рассматривается два вида систем: ***внутренние*** и ***внешние***.

Внутренняя система (***система-явление***) – это целостное образование, к которому можно применить процедуры членения, представляя эту систему в виде некоторой структуры составляющих частей [6].

Внешняя система (***система-класс***) – это класс объектов общей природы, объединенных некоторой целостной сущностью. Элементы такой системы «могут не обладать ни пространственной, ни временной общностью, ни даже генетической связью… Важна лишь общность природы образующих систему объектов» [6, с. 69].

Как видно из этого описания, концепция системы, предлагаемая системологией, четко и однозначно определяет отношение «часть-целое» или «целое-часть». Это отношение является специфическим системным отношением, не сводимым к отношению между множествами и не описываемым теоретико-множественными средствами [6]. Оно называется ***отношением поддержания функциональной способности целого*** [5]. Совершенно очевидно, таким образом, что рассмотренная системная концепция задаёт вполне определённые, конкретные возможности для проведения операций анализа или синтеза ***систем как функциональных объектов*** в упомянутом выше смысле, т.е. является конструктивной.

Рассмотрим подробно процесс формирования (становления) системы, так как при его анализе понятийный аппарат системной концепции проявляется достаточно полно.

Предпосылкой начала процесса формирования системы, согласно данной концепции, является возникновение противоречия между функционированием надсистемы и поддерживающими надсистему функциями ее систем, т.е. противоречия между функциональными и поддерживающими потоками надсистемы. Это противоречие представляет собой нарушение баланса потоков связей (экстенциальных валентностей) в соответствующем узле надсистемы, когда возникают свободные интенциальные валентности окрестностных систем, а ***узел*** оказывается ***вакантным***. С точки зрения системы, которая начнет формироваться из-за возникновения данного противоречия, оно представляет собой функциональный запрос надсистемы (т.е. внешнюю детерминанту системы). Этот запрос, представленный в виде вакантного узла надсистемы, определяет потребность надсистемы, т.е. родившуюся в ней необходимость в системе с данной функцией. Таким образом, запрос (внешняя детерминанта) задает ***область требуемых функциональных состояний*** для формирующейся системы через интенциальные валентности (связи) окрестностных систем.

В соответствии с внешней детерминантой системы, задающей область требуемых функциональных состояний, из резерва (набора систем) выбирается некоторая система как ***исходный материал***. Эта система обладает ***областью возможных состояний***, характеризующей ее предрасположенность (интенцию/потенцию) к выполнению определенных (в данном случае требуемых надсистемой) функций. В результате фактического попадания исходного материала в вакантный узел надсистемы, необходимость превращается в возможность, потоки замыкаются, интенции превращаются в экстенции. Таким образом, система начинает функционировать в соответствии с запросом. При этом исходный материал превращается в субстанцию надсистемы, что и определяет процесс формирования системы с данными функциональными свойствами для поддержания функционирования надсистемы.

Фактическое функционирование системы в ранее вакантном узле надсистемы (новая внутренняя детерминанта системы) становится причиной возникновения противоречия между функциональными и поддерживающими потоками уже системы. Это противоречие становится причиной формирования подсистем с определенными поддерживающими систему функциями и т.д.

В результате описанного процесса система адаптируется к функциональному узлу надсистемы. Процесс ***адаптации*** системы к запросу надсистемы, таким образом, как заключительная фаза становления системы, начинается с того момента, когда данная система в качестве исходного материала помещается в соответствующий вакантный функциональный узел надсистемы. До начала адаптации, когда данная система еще является исходным материалом, внутренние поддерживающие свойства (потоки) данной системы имеют потенции (и скорее даже интенции) к поддержанию требуемых функциональных свойств (потоков), что и способствует выбору именно данной системы в качестве исходного материала. Но, в то же самое время, внутренние свойства (потоки) данной системы, как явления, потенциально и даже экстенциально могут и поддерживают множество других, в данном случае не требуемых, функциональных свойств. Это и обеспечивает ширину области возможных состояний системы (исходного материала), достаточную для включения области требуемых функциональных состояний вакантного узла, т.е. – определенную избыточность свойств до начала адаптации.

В ходе адаптации данной системы к конкретному функциональному запросу под воздействием ее внутренней детерминанты внутренние свойства (потоки) системы, поддерживающие требуемую функцию, будут превращаться из интенций в экстенции, а поддерживающие, в данном случае не нужные, функциональные свойства, наоборот, – из экстенций в интенции и далее в потенции. Таким образом, в результате адаптации уменьшается избыточность свойств системы, все сильнее проявляются ее существенные для данной надсистемы свойства. Система из исходного материала, потенциально пригодного для выполнения заданной функции, превращается во все более совершенную субстанцию данной надсистемы, все более соответствующую запросу.

Чем глубже адаптирована система к функциональному запросу надсистемы, тем ярче проявляются ее существенные для надсистемы свойства, тем выше степень сформированности её сущности, тем меньше избыточность её свойств. Система, у которой область возможных состояний в результате адаптации к запросу надсистемы, не просто покрывает, а максимально близка к области требуемых функциональных состояний, называется ***оптимально адаптированной*** или ***совершенной*** [5]. Такие системы представляют собой четко сформированные, ярко проявляющиеся, вполне устойчивые явления определенной сущности. При этом процесс адаптации системы к функциональному запросу, который периодически сам изменяется, рассматривается в системологии как ***эволюция*** системы.

Одним их примеров системы – функционального адаптивного объекта – может служить сотрудник (специалист) как конкретное «должностное лицо» исполняющий свои функциональные обязанности по занимаемой должности, в

конкретном отделе (надсистеме) научно-исследовательской организации (наднадсистеме). В этом случае можно проследить все, отмеченные выше, этапы формирования системы и ее взаимодействия с надсистемой.

В частности, постановка новой, не предусматривавшейся ранее, задачи отделу со стороны организации приводит к рассогласованию между объемом и видом выполняемых отделом работ, с одной стороны, а также штатным расписанием (структурой) отдела и функциональными обязанностями сотрудников, рассчитанными на прежний круг и объём задач, с другой. Следствием этого рассогласования может быть появление вакантного узла («вакансии») в структуре отдела, прежде всего, содержательно, а затем и формально путем введения в штатное расписание новой должности с требуемыми функциональными обязанностями.

Совершенно очевидно, что на эту должность будут подбираться сотрудники (из некоторого множества кандидатов как резерва исходного материала, например – выпускников вузов) потенциально пригодные для выполнения требуемых функциональных обязанностей, т.е. такие, у которых область их возможных состояний покрывает область требуемых функциональных. Совершенно очевидно также, что после того как кандидат приступает к исполнению служебных обязанностей по данной должности, т.е. становится сотрудником отдела (субстанцией), его потенциальные возможности с течением времени становятся всё более и более соответствующими предъявляемым требованиям. Таким образом, происходит формирование вполне устойчивого в данных условиях явления определенной профессиональной сущности, т.е. специалиста.

2.3.2. Системология – язык теории организации, логистики и инжиниринга бизнеса

Существенной полезной особенностью функциональной системологии является ее адекватность теории организации, логистике и инжинирингу бизнеса, о чем свидетельствуют результаты сравнительного анализа понятийного аппарата системологии и упомянутых методологий теоретического освоения и практического использования организационных систем.Для сравнения выберем конкретные понятия теории организации, логистики и инжиниринга бизнеса.

Результаты сопоставления понятий, законов и принципов теории организации с понятиями функциональной системологии представлены в таблице 2.1. Из таблицы хорошо видно сходство, например, понятий «общий потенциал организации» и «частный потенциал организации» с понятиями «функциональные свойства» и «поддерживающие свойства». При этом, очевидно, что явление или закон синергии представляет собой, по сути дела, ни что иное, как конкретное проявление системологического принципа системности. Очевидно, можно даже утверждать, что организация, в которой не проявляется явление синергии, грубо говоря, вообще не является системой (либо является системой с очень маленькой ***мерой системности*** [5]). Очень близким, например, к понятию теории организации «миссия организации» является системологическое понятие «внешняя детерминанта (запрос надсистемы)».

Таблица 2.1. Теория организации и системология.

Теория организации	***Функциональная системология***
Организационная система.	Функциональный объект, функция которого обусловлена функцией объекта более высокого порядка.
Миссия организации.	Запрос надсистемы (внешняя детерминанта системы) в виде вакантного узла надсистемы.
Потенциал организации.	Область возможных функциональных состояний системы (потенциальная валентность).
Общий потенциал организации.	Общая функция системы (внутренняя детерминанта системы); функциональные связи (свойства).
Частный потенциал организации.	Частная функция системы (функция подсистемы); поддерживающая связь (свойство).
Закон синергии.	Принцип системности (системный эффект).
Закон приоритета целого над частью.	Отношение поддержания функциональной способности целого.
Принцип приоритета цели над функцией.	Выбор внутренней детерминанты в соответствии с внешней детерминантой.
Принцип приоритета функции над элементами и структурой.	Формирование внутренних поддерживающих свойств (субстанции и структуры) в соответствии с функциональными свойствами (внутренней детерминантой).
Принцип соответствия между поставленными целями и выделенными ресурсами.	Соответствие области возможных функциональных состояний исходного материала области требуемых состояний вакантного узла надсистемы.

Результаты сопоставления понятий и принципов логистики с понятиями функциональной системологии представлены в таблице 2.2.

Таблица 2.2. Логистика и системология.

Логистика	***Функциональная системология***
Логистическая система.	Функциональный объект, функция которого обусловлена функцией объекта более высокого порядка.
Поток.	Связь как проявление процесса обмена элементами глубинных ярусов связанных систем.
Запас.	Исходный материал.
Логистическая цепь.	Структура связей системы (поддерживающих).
Логистический процесс.	Функциональные (связи) свойства логистической системы.
Принцип системности.	Отношение поддержания функциональной спо-

Логистика	*Функциональная системология*
	собности целого.
Принцип интегральной оптимальности.	Выбор внутренней детерминанты в соответствии с внешней детерминантой.
Способность логистической системы к целенаправленному приспосабливающемуся поведению.	Формирование внутренних поддерживающих свойств (субстанции и структуры) в соответствии с областью требуемых состояний вакантного узла надсистемы, адаптация.

Из таблицы видно сходство, например, логистического понятия «поток» с понятием функциональной системологии «связь», а также то, что логистический принцип системности представляет собой, по сути дела, проявление в логистической системе системологического отношения поддержания функциональной способности целого.

Результаты сопоставления понятий инжиниринга бизнеса с понятиями функциональной системологии представлены в таблице 2.3.

Таблица 2.3. Инжиниринг бизнеса и системология.

Инжиниринг бизнеса	*Функциональная системология*
Внешняя модель (П-модель) бизнеса.	Модель функциональных связей.
Внутренняя модель (О-модель) бизнеса.	Модель поддерживающих связей.
Образ будущей компании (спецификация целей).	Внешняя детерминанта (предельная внутренняя детерминанта) системы.
Бизнес-процесс.	Формирование из потенциально пригодного исходного материала на глубинном ярусе системы определенной субстанции для обмена по функциональной связи с окрестностной системой.
Почему компания делает то, что она делает?	Функциональный запрос надсистемы (детерминантный анализ).
Почему компания делает то, что она делает таким образом?	Текущая внутренняя детерминанта системы (партитивное классифицирование).
Усовершенствование бизнеса.	Адаптация системы.
Реинжиниринг бизнеса.	Эволюция системы.

Из таблицы видно сходство, например, мероприятий по определению «почему компания делает то, что она делает» с процедурой определения функционального запроса надсистемы.

Представленные таблицы свидетельствуют о концептуальном сходстве

современной теории организации, логистики и инжиниринга бизнеса, а также о том, что эти дисциплины представляют собой, по сути дела, проекцию (конкретизацию) системологии на различные аспекты бизнеса. Это свидетельствует о возможности и целесообразности применения единого системологического подхода для решения проблем этих дисциплин.

2.3.3. Системологический и объектно-ориентированный подход

Традиционные методологии разработки программных систем ориентированы на описание данных и процедур, но не на описание сущностей реального мира и их поведение. Поскольку инжиниринг бизнеса ориентирован на реальные бизнес-структуры и бизнес-процессы, а не на формальные данные и процедуры, традиционные подходы оказались неадекватными. Объектно-ориентированный подход является единственным подходом, позволяющим описывать как сущности, так и их реальное поведение. Кроме того, он обеспечивает создание прозрачных, легко модифицируемых моделей бизнеса и ИС, допускающих повторное использование отдельных компонент. Именно поэтому, в настоящее время, объектно-ориентированная методология анализа, моделирования и разработки ИС является базовой методологией создания программного инструментария для реинжиниринга бизнес-процессов [97].

Сравним основные понятия системологии и объектно-ориентированной методологи.

Главным предметом исследования системологии является «***система***», объектно-ориентированная методология опирается на понятие «***объект***». Концептуальное сходство системологического и объектно-ориентированного подходов (ООП) основывается, в первую очередь, на сходном понимании природы ***системы*** и ***объекта***, соответственно.

С точки зрения системологии, как уже было отмечено, ***система*** – есть функциональный объект, функция которого обусловлена функцией объекта более высокого яруса, т.е. надсистемой [5]; а ***объект***, с точки зрения ООП, представляет собой предмет или сущность, имеющую определенное функциональное назначение в данной предметной области [13, 78].

При этом, вроде бы за небольшим внешним текстуальным сходством, определений системы и объекта скрывается их глубокое концептуальное единство, состоящее в том, что оба понятия не определяются в терминах теории множеств и не рассматриваются как разновидности понятия «множество». Рассмотрение же системы как множества, свойственное традиционному системному подходу, приводит к невозможности поддержать средствами соответствующего системного анализа инкапсуляцию, являющуюся неотъемлемой составной частью ООП и обеспечивающую разделение внешней и внутренней стороны объекта.

Объектно-ориентированная методология в значительной степени опирается также на следующие основные понятия [13, 79]:

- Дихотомию «***класс/объект***», обеспечивающую представление разрабатываемой системы в «канонической форме», т.е. в виде двух ортогональ-

ных иерархий – иерархии классов и иерархии объектов (экземпляров классов).

- Дихотомию «***интерфейс/реализация***», обеспечивающую раздельное рассмотрение поведения и реализующего (поддерживающего) это поведение устройства экземпляра или абстракции.
- Отношение «***клиент – сервер***», определяющее функциональные роли элементов и компонент предметной области и разрабатываемой системы.
- Понятие «***ответственности***», связанное с контрактным проектированием и выражающее предназначение и место объекта или класса в системе.

Анализ этих понятий и результатов их сравнения с понятиями системологии позволяет выявить глубокую аналогию основных положений ООП и системологического подходов.

С точки зрения дихотомии «***класс/объект***» в системологии рассматривается два вида систем: ***системы-классы*** и ***системы-явления*** (называемые в [6] внешними и внутренними системами соответственно).

При этом в настоящее время нам удалось обеспечить единство содержательного и формального рассмотрения обоих видов систем как функциональных объектов. Рассмотрение систем-явлений предметной области позволяет оценить её с точки зрения целостности, устойчивости функционирования, глубины и оптимальности адаптации. Рассмотрение систем-классов предметной области позволяет оценить её с точки зрения естественности (онтологичности) и функционального соответствия объективным запросам систем более высокого порядка [45, 94].

С точки зрения дихотомии «***интерфейс/реализация***» в системологии рассматривается два вида свойств: ***функциональные и поддерживающие***.

С точки зрения отношения «***клиент – сервер***» (в данном случае удобнее сказать «сервер – клиент»), при котором сервер своими ресурсами и услугами (своим функционированием) поддерживает функционирование клиента, в системологии рассматривается ***отношение поддержания функциональной способности целого***. Данное отношение представляет собой отношение система – надсистема, обеспечивающее приобретение надсистемой функциональной способности, поддерживаемой системами и несводимой к способностям систем. Противоположным ему (соответствующим именно отношению «клиент – сервер») является ***отношение детерминирования свойств частей свойствами целого***.

С точки зрения контрактного проектирования и, в частности, понятия «***ответственности***», которое выражает предназначение объекта в системе через совокупность его контрактных обязательств, в системологии рассматриваются понятия ***внутренней детерминанты*** и ***внешней детерминанты***, которая и определяет выбор внутренней.

Сходство представленных понятий имеет большое значение, как для системологии – с точки зрения выявления еще одной важной области её применения: объектно-ориентированного проектирования современного ПО; так и для фундаментального теоретического обоснования и развития самой объектно-ориентированной методологии.

Кроме того, требование исследования иерархии объектов и иерархии классов в ходе объектно-ориентированного анализа системы находится в полном согласии и соответствии с требованиями системологии, в которой предполагается необходимость ***детерминантного анализа*** внутренних систем предметной области (систем-явлений или экземпляров) и внешних систем (систем-классов). Следовательно, системологический подход позволяет проанализировать полную и целостную архитектуру рассматриваемой системы в соответствии с требованиями методологии OOAD. При этом рассмотрение внутреннего и внешнего аспекта системности характерно именно для системологии и ни в какой другой концепции системного подхода не используется.

Связь между системологией и объектно-ориентированным подходом представлена в таблице 2.4.

Таблица 2.4. Объектный подход и системология.

Объектный подход	***Системология***
Объект.	Система-явление (внутренняя система).
Класс.	Система-класс (внешняя система).
Объектно-ориентированная декомпозиция.	Системная декомпозиция, на основе отношения поддержания функциональной способности целого.
Структура (иерархия) объектов.	Партитивная (цело-частная) классификация.
Структура (иерархия) классов.	Таксономическая (родо-видовая) классификация.
Контрактное программирование (клиент-сервер).	Внешняя детерминанта (запрос надсистемы) – Внутренняя детерминанта (функция системы, соответствующая запросу).
Варианты использования (прецеденты).	Внутренняя детерминанта.
Главная задача объектно-ориентированного проектирования: выбор правильного набора абстракций (классов).	Метод построения классификаций, отражающих существенные свойства предметной области: системологический классификационный анализ.

Объектно-ориентированное мировоззрение, при котором «требования к системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области» [13, с. 54], и системологическое мировоззрение, при котором предметная область рассматривается как иерархия функциональных объектов (систем-явлений), находящихся в отношении поддержания функциональной способности целого [5], а также как иерархия систем-классов, находящихся в том же отношении [94], таким образом, совершенно сходятся. Можно даже утверждать, что такие составные части объектно-ориентированной методологии

как OOA и OOD есть, по сути своей, ни что иное, как изложение системологии в терминах программной инжинирии.

Вопросы для повторения

1. Назовите основные проблемы традиционного системного подхода.
2. В чем состоит основное отличие понятия «система» от понятия «множество»?
3. Что такое объектно-ориентированный анализ?
4. Что такое объектно-ориентированное проектирование?
5. Что такое объектно-ориентированное программирование?
6. Назовите основные понятия объектно-ориентированного подхода.
7. Что такое «UML»?
8. Зачем необходима интеграция системно-структурного и объектно-ориентированного подходов?
9. Дайте определение системы как функционального объекта?
10. Что такое связь между системами с точки зрения системологии?
11. Что такое внешняя детерминанта?
12. Что такое внутренняя детерминанта?
13. Что такое адаптация системы?
14. Что такое эволюция системы?
15. Какие понятия теории организации соответствуют каким понятиям системологии?
16. Какие понятия логистики соответствуют каким понятиям системологии?
17. Какие понятия инжиниринга бизнеса соответствуют каким понятиям системологии?
18. Какие понятия объектно-ориентированного подхода соответствуют каким понятиям системологии?

Резюме по теме

В данном разделе рассмотрены основные подходы к моделированию и анализу сложных слабоформализованных систем. При этом изучены проблемы традиционного системного подхода и системного анализа, а также причины их существования. Кроме того, показаны особенности объектно-ориентированного подхода и обоснована необходимость интеграции системно-структурного и объектно-ориентированного подходов. Приведены основные понятия системологии, которые сопоставлены с основными понятиями дисциплин, направленных на рационализацию организационных систем (теории организации, логистики и инжиниринга бизнеса), а также с основными понятиями объектно-ориентированного подхода.

Тема 3. Технологии системного моделирования

Цели и задачи изучения темы

Целью изучения данной темы является теоретическое и практическое освоение технологий системно-структурного моделирования и анализа сложных систем.

При этом ставятся следующие задачи:

- изучение технологии моделирования «3-View Modeling» (технологий построения DFD, ERD и STD-диаграмм);
- изучение стандартных методов системно-структурного анализа, описанных серией федеральных стандартов США «Icam Definition»;
- изучение CASE-инструментария системно-структурного моделирования и анализа (AllFusion Process Modeler).

3.1. Технология системно-структурного моделирования и анализа «3-View Modeling»

3.1.1. Диаграммы потоков данных: нормативная система; построение модели; словарь данных; спецификация процесса

По мнению специалистов в области системного анализа [19, 20, 22, 32, 33, 47, 48, 98] для решения задач анализа и проектирования некоторого объекта необходимо моделировать:

- функции этого объекта, например, с помощью ***диаграмм потоков данных – DFD (Data Flow Diagrams)***;
- отношения между данными, которые в нем используются, например, с помощью ***диаграмм «сущность – связь» – ERD (Entity-Relationship Diagrams)***;
- поведение объекта (события), например, с помощью ***диаграмм переходов состояний – STD (State Transition Diagrams)***.

Рассмотрим графические средства построения этих диаграмм, а также вспомогательные текстовые средства (***спецификацию процесса*** и ***словарь данных***), обеспечивающие точное определение их компонент (см. рис. 3.1. и [20]).

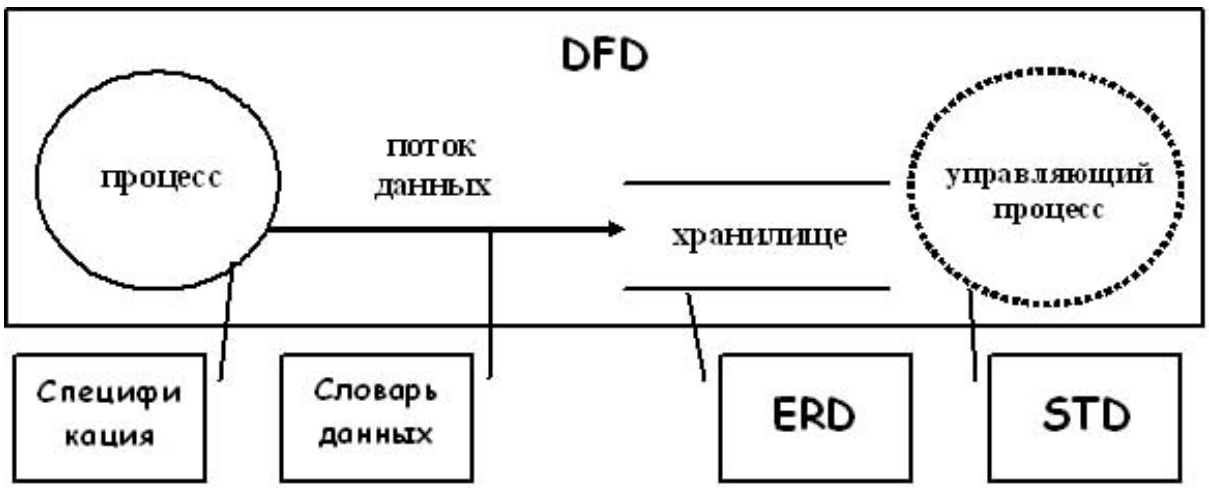


Рис. 3.1. - Схема технологии моделирования 3VM.

Таким образом, с помощью DFD-диаграмм изображаются функции (процессы), накопители (хранилища) данных и связывающие их потоки. При этом идентифицируются внешние по отношению к системе источники и адресаты данных. Каждый процесс может быть детализирован с помощью DFD-диаграммы нижнего уровня или, в конце концов, описан в виде спецификации процесса. Структура потоков данных и определения их компонент, а также содержимое хранилищ описывается в словаре данных. Модель данных хранилища раскрывается с помощью ERD-диаграммы. Если моделируется система реального времени, то дополнительно поведение системы описывается с помощью STD-диаграммы. Данная технология системно-структурного анализа используется при построении моделей в рамках британской стандартной методологии анализа и проектирования систем SSADM.

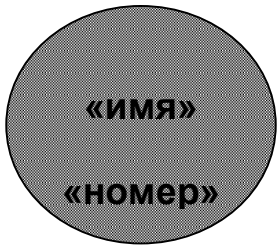
Рассмотрим технологию 3VM в первую очередь в соответствии с работой [20], а также работами [22, 32, 98] подробнее.

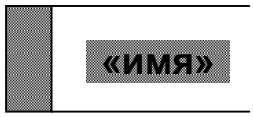
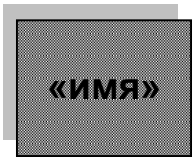
Диаграммы потоков данных (***DFD***) являются средством моделирования функциональных требований к проектируемой системе. С их помощью эти требования разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств – продемонстрировать,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

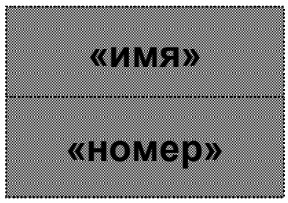
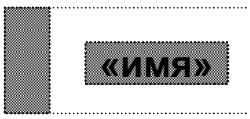
Нормативная система

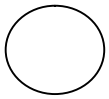
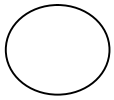
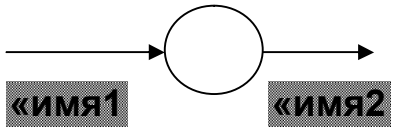
Для изображения DFD-диаграмм традиционно используются две различные нотации: Йордана и Гейна-Сарсона. Алфавит, используемый для построения DFD-диаграмм, представлен в таблице 3.1.

Таблица 3.1. Нормативная система (нотация) DFD.

Элемент алфавита	***Нотация Йордана***	***Нотация Гейна-Сарсона***
ПОТОК ДАННЫХ Используется для моделирования передачи информации (или даже физических компонент) из одной части системы в другую. На диаграммах изображаются именованными стрелками, ориентация которых указывает направление потока.		
ПРОЦЕСС Используется для моделирования процесса преобразования входного потока в выходной. Его имя должно содержать глагол в неопределенной форме с последующим дополнением (например, «Вычислить высоту»). Кроме того, каждый процесс должен иметь уникальный номер для ссылок на него внутри диаграммы, который совместно с номером диаграммы уникален во всей модели.		

Элемент алфавита	*Нотация Йордана*	*Нотация Гейна-Сарсона*
ХРАНИЛИЩЕ (НАКОПИТЕЛЬ) ДАННЫХ Используется для моделирования данных (или даже физических компонент), которые будут сохраняться между процессами. Информация, которую оно содержит, может использоваться в любое время после ее определения, при этом данные могут выбираться в любом порядке. Имя хранилища должно идентифицировать его содержимое и быть существительным. В случае, когда поток данных входит или выходит в/из хранилища, и его структура соответствует структуре хранилища, он должен иметь то же самое имя, которое нет необходимости отражать на диаграмме.		
ВНЕШНЯЯ СУЩНОСТЬ (ТЕРМИНАТОР) Используется для моделирования сущностей вне системы (контекстных сущностей), являющихся источником или приемником системных данных. Ее имя должно содержать существительное, например, «Склад». Предполагается, что объекты, представленные такими сущностями, не должны участвовать ни в какой обработке.		
УПРАВЛЯЮЩИЙ ПОТОК Используется для моделирования связи, через которую передается управляющая информация. Его имя не должно содержать глаголов, а только существительные и прилагательные. Обычно управляющий поток имеет дискретное, а не непрерывное значение. Это может быть, например, сигнал, представляющий состояние или вид операции. Имеются следующие типы управляющих потоков: а) **Т-поток** (trigger flow) – поток управления, который может вызывать выполнение процесса. При этом процесс как бы включается одной короткой операцией. Это аналог выключателя света, единственным нажатием которого «запускается» процесс горения лампы. б) **А-поток** (activator flow) – поток управления, который может изменять выполнение процесса. Используется для обеспечения непрерывности выполнения процесса до тех пор, пока поток «вклю-		

Элемент алфавита	*Нотация Йордана*	*Нотация Гейна-Сарсона*
чен» (т.е. течет непрерывно); с «выключением» потока выполнение процесса завершается. Это – аналог переключателя лампы, которая может быть как включена, так и выключена. в) **E/D-поток** (enable/disable flow) – поток управления, который может переключать выполнение процесса. Течение по E-линии вызывает выполнение процесса, которое продолжается до тех пор, пока не возбуждается течение по D-линии. Это аналог выключателя с двумя кнопками: одной для включения света, другой для его выключения. Можно использовать 3 типа таких потоков: E-поток, D-поток, E/D-поток.		
УПРАВЛЯЮЩИЙ ПРОЦЕСС Используется для моделирования преобразователя входных управляющих потоков в выходные управляющие потоки; при этом точное описание этого преобразования должно задаваться в спецификации управления. Его имя указывает на тип управляющей деятельности, описанной в спецификации. Логически управляющий процесс есть командный пункт, реагирующий на изменения внешних условий, которые сообщаются ему с помощью управляющих потоков, и продуцирующий в соответствии со своей внутренней логикой команды для других процессов системы.		
УПРАВЛЯЮЩЕЕ ХРАНИЛИЩЕ Используется для моделирования управляющей информации, которая будут сохраняться между процессами. Содержащаяся в нем управляющая информация может использоваться в любое время после ее занесения в хранилище, при этом соответствующие данные могут быть использованы в произвольном порядке. Имя управляющего хранилища должно идентифицировать его содержимое и быть существительным. Управляющее хранилище отличается от традиционного тем, что может содержать только управляющие потоки; все другие их характеристики идентичны.		

Элемент алфавита	*Нотация Йордана*	*Нотация Гейна-Сарсона*
ГРУППОВОЙ УЗЕЛ Используется для моделирования расщепления и объединения потоков.		
УЗЕЛ-ПРЕДОК Используется для связывания входящих и выходящих потоков между детализируемым процессом и детализирующей DFD-диаграммой.		
НЕИСПОЛЬЗУЕМЫЙ УЗЕЛ Используется для моделирования ситуации, когда декомпозиция данных производится в групповом узле, при этом требуются не все элементы входящего в узел.		
УЗЕЛ ИЗМЕНЕНИЯ ИМЕНИ Используется для обеспечения возможности неоднозначного именования потоков, содержимое которых эквивалентно. Например, если при проектировании разных частей системы один и тот же фрагмент данных получил различные имена, то эквивалентность соответствующих потоков данных обеспечивается узлом изменения имени. При этом один из потоков данных является входным для данного узла а другой – выходным.		
ТЕКСТ В свободном формате в любом месте диаграммы.		

Построение модели

Главная цель построения модели в виде иерархического множества DFD-диаграмм заключается в том, чтобы сделать требования ясными и понятными на каждом уровне детализации, а также разбить эти требования на части с точно определенными отношениями между ними. Для достижения этого рекомендуется пользоваться следующими правилами [20]:

1. Размещать на каждой диаграмме от 3 до 6-7 процессов. Верхняя граница соответствует человеческим возможностям одновременного восприятия и понимания структуры сложной системы с множеством внутренних связей, нижняя граница выбрана по соображениям здравого смысла: нет необходимости детализировать процесс диаграммой, содержащей всего один или два процесса.
2. Не загромождать диаграммы несущественными на данном уровне детализации сущностями.
3. Декомпозицию потоков данных осуществлять параллельно с декомпозицией процессов; эти две работы должны выполняться одновременно, а не одна после завершения другой.

4. Выбирать ясные, отражающие суть дела, имена процессов и потоков для улучшения восприятия диаграмм, при этом не рекомендуется использовать аббревиатуры.
5. Однократно определять функционально идентичные процессы на самом верхнем уровне иерархии, где такой процесс необходим, и ссылаться на него на нижних уровнях иерархии.
6. Пользоваться простейшими диаграммными техниками: если что-либо возможно описать с помощью DFD-диаграмм, то это и необходимо делать, а не использовать для описания более сложные объекты.
7. Отделять управляющие структуры от обрабатывающих структур (т.е. процессов), локализовать управляющие структуры.

В соответствии с этими рекомендациями процесс построения модели разбивается на следующие этапы [20]:

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

14. Построение спецификации процесса (а не простейшей диаграммы) в случае, если некоторую функцию сложно или невозможно выразить комбинацией процессов.

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

Декомпозиция DFD-диаграммы осуществляется на основе процессов: каждый процесс может раскрываться с помощью DFD-диаграммы нижнего уровня. DFD-диаграмма первого уровня строится как декомпозиция процесса, который присутствует на контекстной диаграмме. Построенная диаграмма первого уровня также имеет множество процессов, которые в свою очередь могут быть декомпозированы. Таким образом строится иерархия DFD-диаграмм с контекстной диаграммой в корне дерева. Этот процесс декомпозиции продолжается до тех пор, пока процессы могут быть эффективно описаны с помощью коротких (до одной страницы) спецификаций процессов.

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

Пример иерархии DFD-диаграмм, описывающих ***систему управления лифтом***, аналогичную представленной в работе [32] (Elevator Control System – ECS), подготовленный с применением инструментального пакета BPwin, изображен на рисунках 3.2 – 3.7. Дополнительные возможности нотации позволяют, в частности, с помощью двойных стрелок отличить материальные потоки от информационных. Данный пример используется далее для описания различных аспектов технологии моделирования 3VM.

Диаграммы (3.2 – 3.7) построены с учетом следующих требований к системе управления лифтом [32, стр. 15-17]:

– лифт используется для перевозки людей в 40-а этажном офисном здания обычным способом (при вызове лифта с этажа он должен останавливаться при попутном движении; лифт не должен менять направления, пока едущие в нем пассажиры не достигнут назначенных этажей; пустой лифт должен стоять там, где он остановился последний раз);

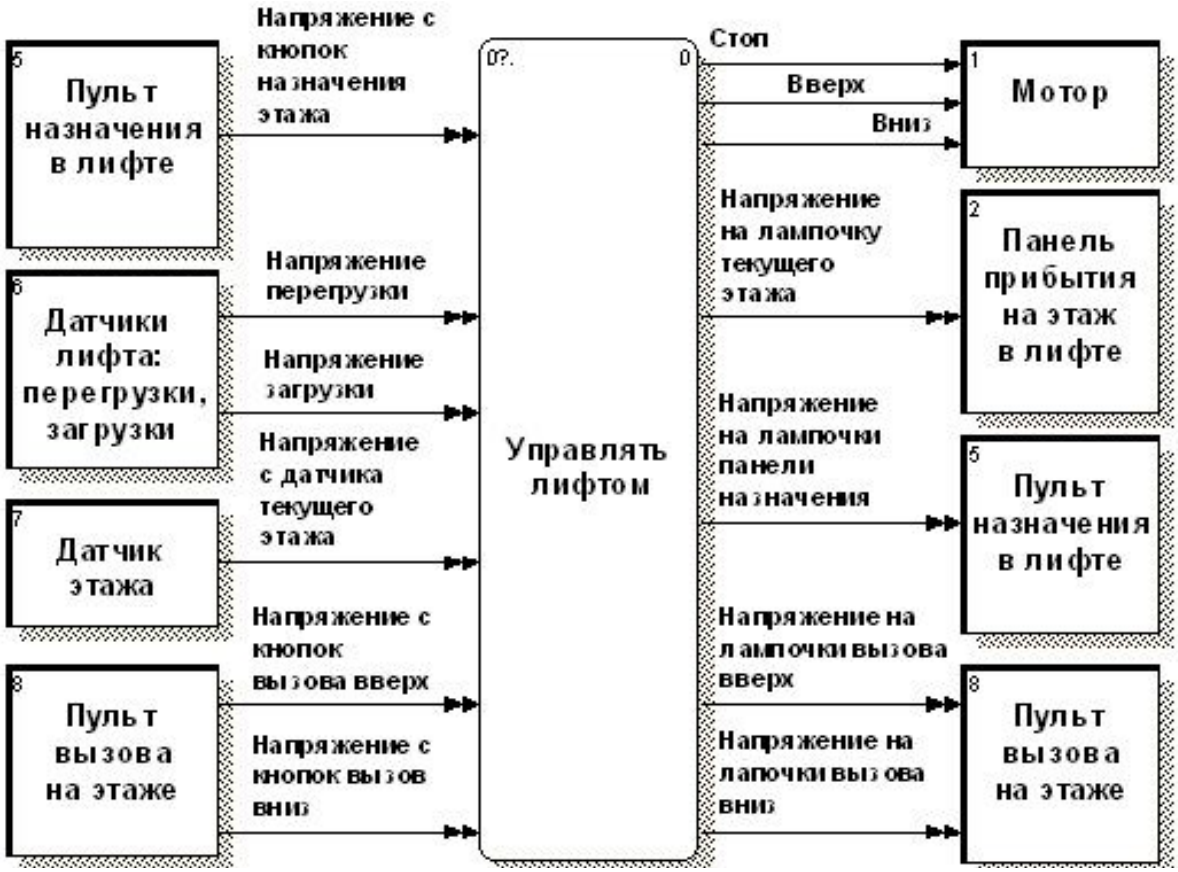


Рис. 3.2. - Контекстная DFD-диаграмма ECS.

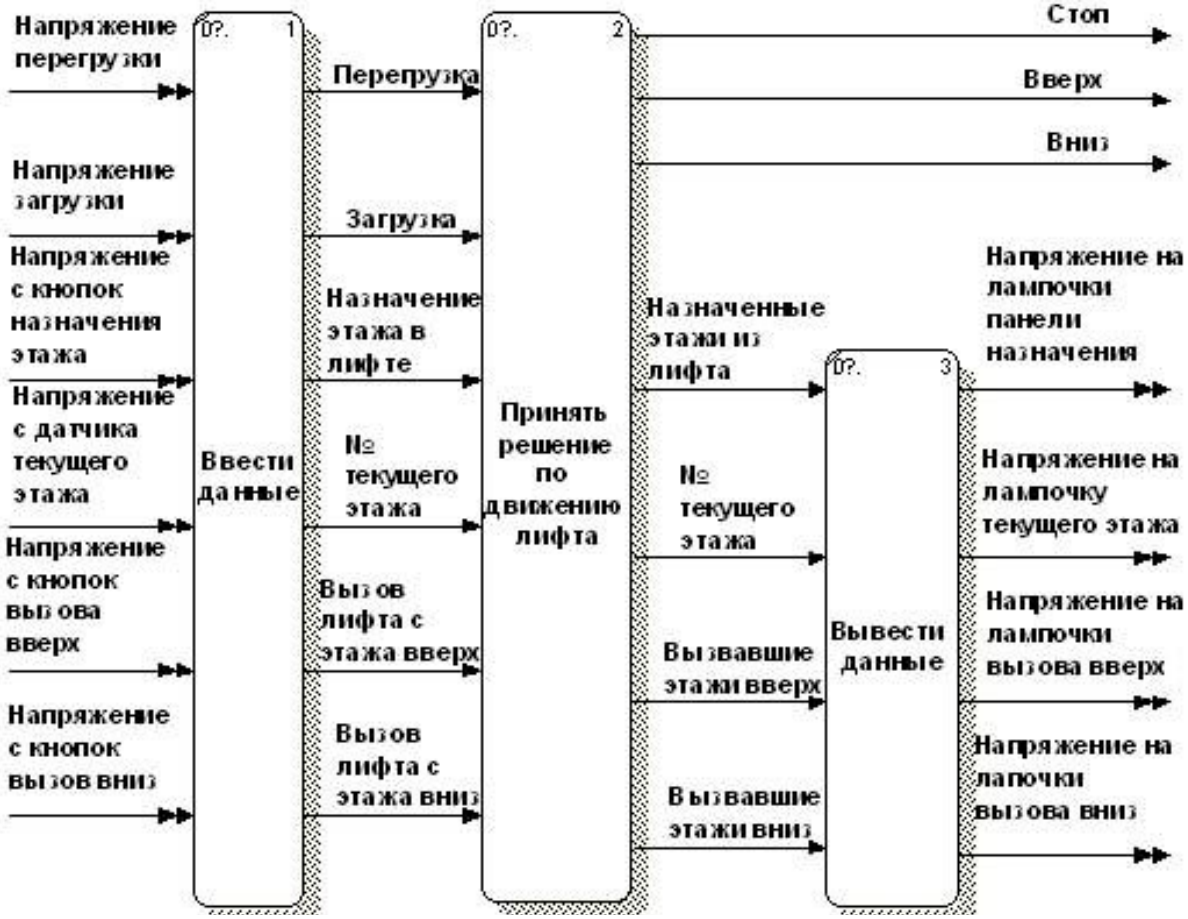


Рис. 3.3. - DFD-диаграмма ECS первого уровня.

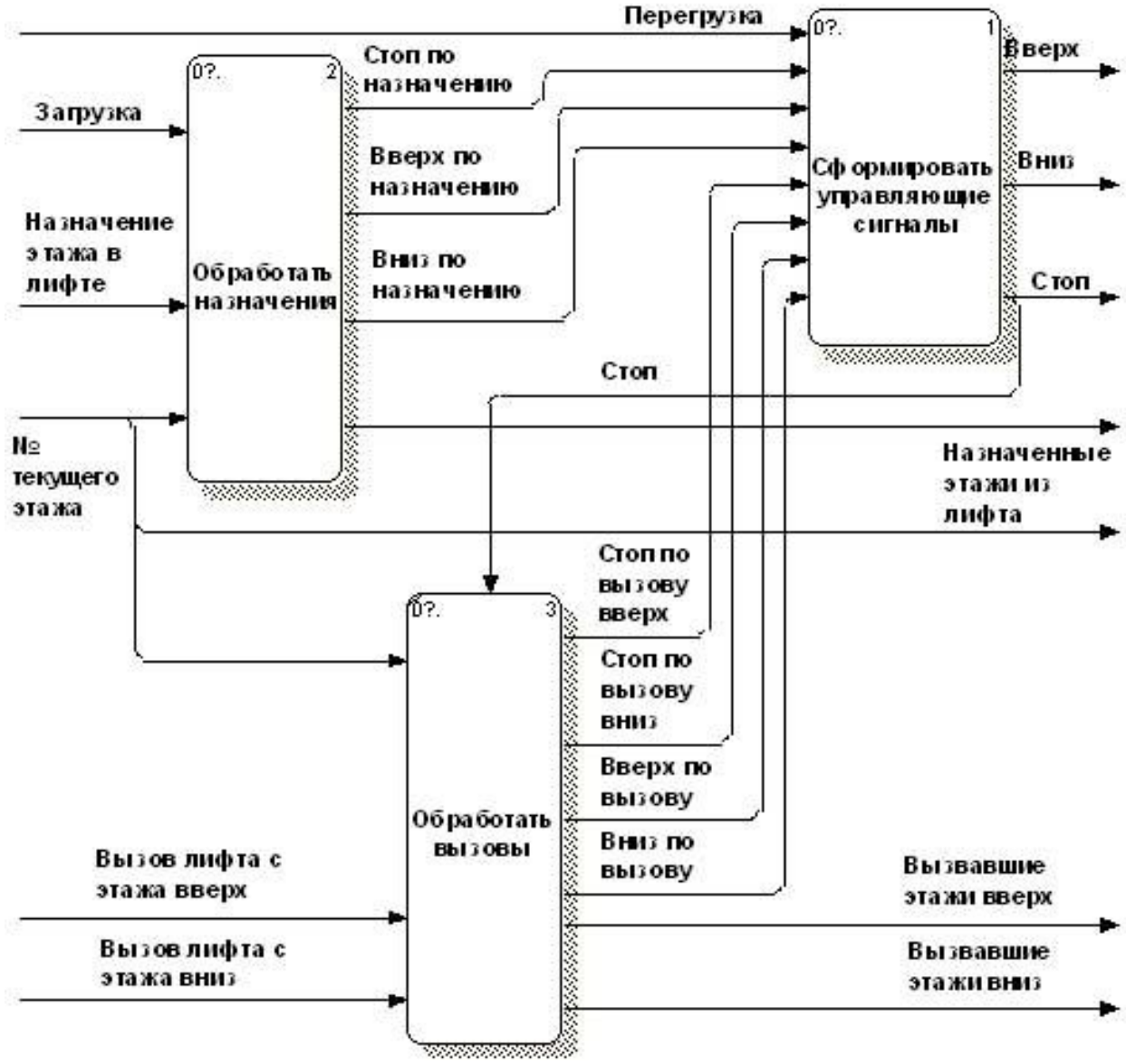


Рис. 3.4. - DFD-диаграмма второго уровня (процесса принятия решений).

Рис. 3.5. - DFD-диаграмма третьего уровня (обработка назначений).

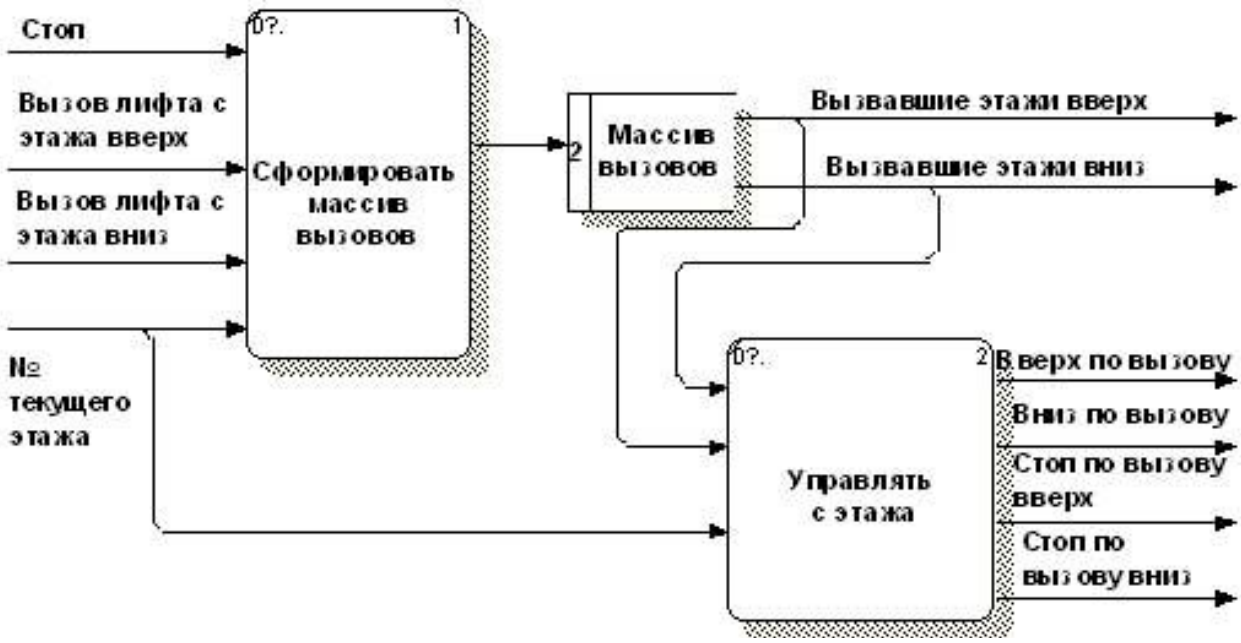


Рис. 3.6. - DFD-диаграмма третьего уровня (обработка вызовов).

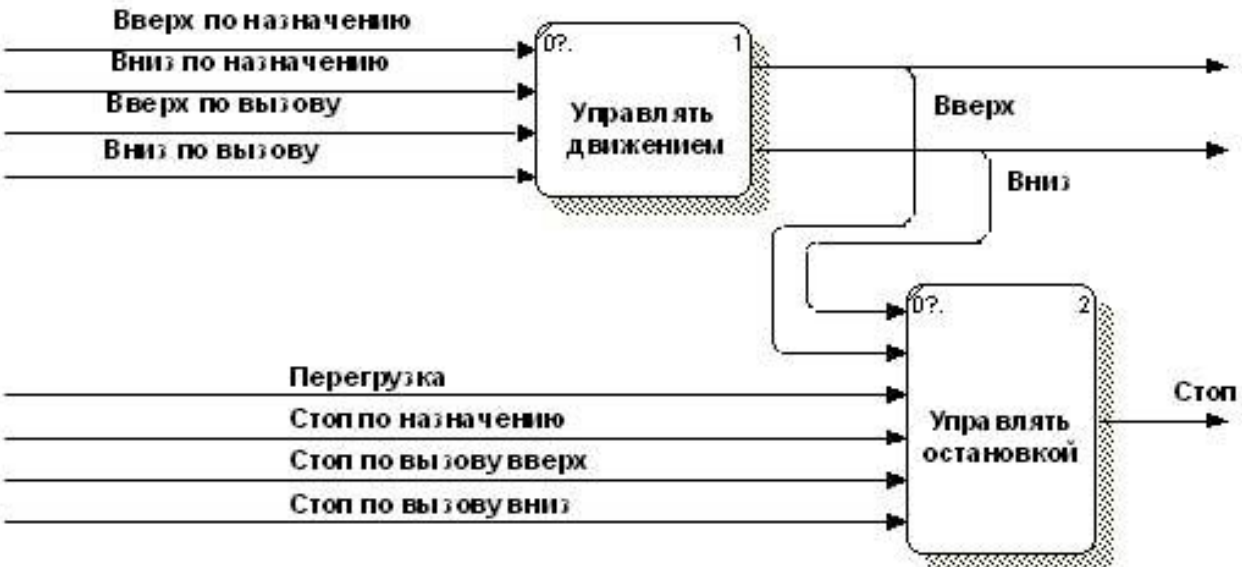


Рис. 3.7. - DFD-диаграмма третьего уровня (формирование управляющих сигналов).

- ECS должна реагировать на электромагнитные сигналы от датчиков (каждого этажа и состояния лифта), а также кнопок (панелей вызова лифта с каждого этажа и панели назначения этажа в лифте);
- должна обеспечиваться индикация этажа прибытия лифта и принятия ECS вызовов и назначений;
- ECS должна обеспечивать «связанное с этажом» составление расписания движения лифта, путем выработки управляющих сигналов «стоп», «вверх» «вниз» на мотор в зависимости от текущего этажа, нажатых кнопок и состояния лифта.

Словарь данных

Диаграммы потоков данных обеспечивают описание взаимодействий функциональных компонент системы (ее структуры), но не имеют, сами по себе, средств для описания связей между компонентами и их функций, а именно, какая информация преобразуется процессами и как она преобразуется. Для решения первой из перечисленных проблем предназначено текстовое средство моделирования, служащее для описания содержания преобразуемой информации. Оно получило название «словарь данных».

Словарь данных представляет собой определенным образом организованный список всех элементов, составляющих потоки данных системы с их точными определениями. Это дает возможность различным категориям пользователей (от системного аналитика до программиста) иметь общее понимание всех входных и выходных потоков и компонент хранилищ.

Определение элементов данных в словаре осуществляется с помощью описаний следующих видов [20]:

- описанием характеристик потоков и хранилищ, изображенных на DFD-диаграммах;
- описанием композиций данных, движущихся вдоль потоков, т.е. комплексных данных, которые могут расчленяться на элементарные (например, «Адрес» содержит «Индекс», «Город», «Улица» и т.д.);
- описанием композиций данных в хранилище.

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

Типы потоков определяются в словаре как:

- простые (элементарные) или групповые (комплексные) потоки;
- внутренние (существующие только внутри системы) или внешние (связывающие систему с другими системами) потоки;
- потоки данных или потоки управления;
- непрерывные (принимающие любые значения в пределах диапазона) или дискретные (принимающие определенные значения) потоки.
- Атрибуты потока данных включают:
- имена-синонимы потока данных в соответствии с узлами изменения имени;
- определение с помощью БНФ-нотации (см. далее) в случае группового потока;
- единицы измерения потока;
- диапазон значений для непрерывного потока, типичное его значение и информацию по обработке экстремальных значений;
- список значений и их смысл для дискретного потока;
- список номеров диаграмм, в которых поток встречается;
- список потоков, в которые данный поток входит;
- комментарий, включающий дополнительную информацию (например о

цели введения данного потока).

БНФ-нотация позволяет формально описать расщепление/объединение потоков. Поток может расщепляться на собственные отдельные ветви, на компоненты потока-предка или на то и другое одновременно.

При расщеплении/объединении потока существенно, чтобы каждый компонент потока-предка являлся именованным. При объединении подпотоков нет необходимости осуществлять исключение общих компонент, а при расщеплении подпотоки могут иметь такие общие (одинаковые) компоненты.

Точные определения потоков содержатся в словаре данных, а не на диаграммах. Например, на диаграмме может иметься групповой узел с входным потоком X и выходными подпотоками Y и Z. Однако, это вовсе не означает, что соответствующее определение в словаре данных обязательно должно быть X=Y+Z. Это определение может быть следующим: X=A+B+C; Y=A+B; Z=B+C. Такие определения хранятся в словаре данных в так называемой БНФ-статье.

БНФ-статья используется для описания компонент данных в потоках данных и в хранилищах. Ее синтаксис имеет следующий вид:

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

где <***простой оператор***> есть текстовое описание, заключенное в «/», а <***БНФ-выражение***> есть выражение в ***форме Бэкуса-Наура***, заключенное в «***[ ]***» и допускающее следующие операции и отношения: **=** – означает «***композиция из***»; **+** – означает «***И***»; ***!*** – означает «***ИЛИ***»; ***()*** – означает, что компонент в скобках не обязателен; ***{}*** – означает итерацию компонента в скобках; ***« »*** – означает литерал.

Итерационные скобки могут иметь нижний и верхний предел, например:

3 {болт} 7 – от 3 до 7 итераций

1 {болт} – 1 и более итераций

{шайба} 3 – не более 3 итераций

БНФ-выражение может содержать произвольные комбинации операций:

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

Ниже приведен пример описания потока данных подсистемы принятия решений ECS «№ текущего этажа» (рис. 3.3 – 3.6) с помощью БНФ-статьи:

@ИМЯ = № ТЕКУЩЕГО ЭТАЖА

@ТИП = ПРОСТОЙ, ВНУТРЕННИЙ, ДИСКРЕТНЫЙ ПОТОК ДАННЫХ

@БНФ = [«1» ! «2» ! … ! «39» ! «40»]

Далее приведен пример описания входных потоков данных блока формирования управляющих сигналов на примере потока «Перегрузка», а также выходных потоков этого блока на примере потока «Стоп» (рис. 3.7):

@ИМЯ = ПЕРЕГРУЗКА

@ТИП = ПРОСТОЙ, ВНУТРЕННИЙ, ДИСКРЕТНЫЙ ПОТОК ДАННЫХ

@БНФ = [«0» ! «1»]

@ИМЯ = СТОП

@ТИП = ПРОСТОЙ, ВНЕШНИЙ, ДИСКРЕТНЫЙ ПОТОК УПРАВЛЕНИЯ

@БНФ = [«0» ! «1»]

Спецификация процесса

Спецификация процесса (СП) используется для описания процесса, когда нет необходимости детализировать его с помощью DFD-диаграммы (т.е. если он невелик и его описание может занимать не более одной страницы текста). СП представляет собой описание алгоритма, соответствующего данному процессу и трансформирующего входные потоки данных в выходные. Множество всех СП является полной спецификацией системы. СП содержит номер и/или имя процесса, списки входных и выходных данных и тело (описание) процесса.

Известно большое число разнообразных методов, позволяющих задать тело процесса: от **структурированного естественного языка** или псевдокода до **визуальных языков проектирования** (типа **FLOW-форм)**, а также формальных языков программирования.

Независимо от используемого метода СП должна начинаться со следующих ключевых слов:

@ВХОД = <имя символа данных >
@ВЫХОД = <имя символа данных>
@СПЕЦПРОЦ

где ***<имя символа данных>*** – соответствующее имя из словаря данных.

Например, для процессов формирования массивов (рис. 3.5 и 3.6),

@ВХОД = Назначение этажа в лифте; № текущего этажа; Стоп по назначению
@ВЫХОД = Массив назначений
@СПЕЦПРОЦ
Сформировать очередь №№ этажей, назначенных в лифте
@

@ВХОД = Вызов лифта с этажа вверх; Вызов лифта с этажа вниз; № текущего этажа; Стоп
@ВЫХОД = Массив вызовов
@СПЕЦПРОЦ
Сформировать очередь №№ вызвавших этажей
@

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

СП должна удовлетворять следующим требованиям [20]:

- для каждого процесса нижнего уровня должна существовать одна и только одна спецификация;
- спецификация должна определять способ, но не метод преобразования входных потоков в выходные;
- спецификация должна стремиться к ограничению избыточности: не сле-

дует переопределять то, что уже было определено на диаграмме или в словаре данных;

Ниже рассматриваются некоторые наиболее часто используемые методы задания СП.

Структурированный естественный язык является разумной комбинацией строгости языка программирования и читабельности естественного языка. Он состоит из подмножества слов, организованных в определенные логические структуры, арифметических выражений и диаграмм. В состав структурированного языка входят следующие основные символы:

- глаголы, ориентированные на действия и применяемые к объектам данной предметной области;
- существительные, определенные на любой стадии анализа, моделирования и проектирования (например, при разработке информационной системы: задачи, процедуры, символы данных и т.п.);
- предлоги и союзы, используемые в логических отношениях;
- общеупотребительные математические, логические, физические и технические термины;
- арифметические выражения;
- таблицы, диаграммы, графики и т.п.;
- комментарии.

Управляющие конструкции структурированного языка имеют вид:

Последовательная конструкция:
ВЫПОЛНИТЬ функция1
ВЫПОЛНИТЬ функция2
ВЫПОЛНИТЬ функция3

Конструкция выбора:
ЕСЛИ<условие>ТО
ВЫПОЛНИТЬ функция1
ИНАЧЕ
ВЫПОЛНИТЬ функция2
КОНЕЦЕСЛИ

Итерация:
ДЛЯ <условие>
ВЫПОЛНИТЬ функция
КОНЕЦДЛЯ

Или
ПОКА <условие>
ВЫПОЛНИТЬ функция
КОНЕЦПОКА

При использовании структурированного естественного языка приняты следующие соглашения [20]:

- логика процесса выражается в виде комбинации последовательных конструкций, конструкций выбора и итераций;
- ключевые слова ЕСЛИ, ВЫПОЛНИТЬ, ИНАЧЕ и т.д. должны быть написаны заглавными буквами;
- слова или фразы, определенные в словаре данных, должны быть написаны заглавными буквами;
- глаголы должны быть активными, недвусмысленными и ориентированными на целевое действие (заполнить, вычислить, извлечь, а не модернизировать, обработать);
- логика процесса должна быть выражена четко и недвусмысленно.

Таблицы и деревья решений применяются в тех случаях, когда структурированный естественный язык оказывается неприемлем. Например, если действие зависит от нескольких переменных, которые в совокупности могут продуцировать большое число комбинаций, то его описание будет слишком запутанным и с большим числом уровней вложенности. Для описания подобных действий традиционно используются таблицы и деревья решений.

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

ТР состоит из двух частей. Верхняя часть таблицы используется для определения условий. Обычно условие является **ЕСЛИ**-частью оператора **ЕСЛИ-ТО** и требует ответа «да-нет». Нижняя часть ТР используется для определения действий, т.е. **ТО**-части оператора **ЕСЛИ-ТО.**

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

Построение ТР рекомендуется осуществлять путем выполнения следующих шагов [20]:

1. Идентифицировать все условия (или переменные), участвующие в описываемом процессе. Идентифицировать все значения, которые каждая переменная может иметь.
2. Вычислить число комбинаций условий.
3. Идентифицировать каждое из возможных действий, которые могут вызываться в описываемом процессе.
4. Построить пустую таблицу, включающую все возможные условия и действия, а также номера комбинаций условий.
5. Выписать и занести в таблицу все возможные комбинации условий.
6. Редуцировать комбинации условий (если это возможно).
7. Проверить каждую комбинацию условий и идентифицировать соответствующие выполняемые действия.
8. Выделить комбинации условий, для которых в данном процессе нет выполняемых действий.
9. Обсудить построенную таблицу.

Поясним вышесказанное на примере СП управления движением лифта (рис. 3.7, процесс №1) и СП управления остановкой лифта (рис. 3.7, процесс №2) (см. таблицы 3.2 и 3.3 соответственно).

При составлении таблиц использованы следующие обозначения входных потоков процессов управления движением и остановкой лифта, как условий: перегрузка – **П**, стоп по назначению – **Сн**, вверх по назначению – **Вн**, вниз по назначению – **Нн**, стоп по вызову вверх – **Свв**, стоп по вызову вниз – **Снв**, вверх по вызову – **Вв**, вниз по вызову – **Нв**; и выходных потоков этих процессов, как действий: стоп – **С**, вверх – **В**, вниз – **Н**.

Таблица 3.2. Таблица решений процесса управления движением лифта

Условия	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Вн	0	0	0	0	0	0	1	1	1
Нн	0	0	0	1	1	1	0	0	0
Вв	0	0	1	0	0	1	0	0	1
Нв	0	1	0	0	1	0	0	1	0
Действия									
В	0	0	1	0	0	0	1	1	1
Н	0	1	0	1	1	1	0	0	0

Таблица 3.3. Таблица решений процесса управления остановкой лифта

Условия	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
П	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Сн	0	0	0	0	0	0	0	0	0	0	0	0	1	*
Свв	0	0	0	0	0	0	1	1	1	1	1	1	*	*
Снв	0	0	0	1	1	1	0	0	0	1	1	1	*	*
В	0	0	1	0	0	1	0	0	1	0	0	1	*	*
Н	0	1	0	0	1	0	0	1	0	0	1	0	*	*
Действия														
С	0	0	0	1	1	0	1	0	1	1	1	1	1	1

Обсудим построенные таблицы. Данные таблицы визуализируют обычную логику работы лифта в офисном здании. Согласно табл. 3.2, если из лифта было назначено движение вниз или вверх, то независимо от вызовов лифт будет выполнять эти назначения. При этом назначений вверх и вниз одновременно быть не может, что обеспечивается формированием очереди №№ этажей, назначенных в лифте (см. рис. 3.5). При отсутствии назначений будут выполняться вызова с этажей, которые по той же причине (см. рис. 3.6) не могут быть одновременно и вверх, и вниз. Согласно табл. 3.3 наличие перегрузки или требования остановится из лифта вызывают безусловную остановку. Требования остановится с этажа выполняются только при попутном движении лифта. Управляющий сигнал «**С**» при наличии требования на остановку с этажа (вызова с этажа) в случае, когда лифт стоит (**В** = 0; **Н** = 0), используется для открывания двери на вызвавшем этаже механической системой лифта.

Вариантом таблицы решений является **дерево решений** (ДР), позволяющее взглянуть на процесс условного выбора с позиции схемы. Обычно ДР используется при малом числе действий и когда не все комбинации условий возможны, а ТР – при большом числе действий и когда возможно большое число комбинаций условий. На основе ТР легко осуществляется автоматическая кодогенерация.

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

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

Символы FLOW-форм приведены на рисунке 3.8. Каждый символ является блоком обработки. Каждый прямоугольник внутри любого символа также представляет собой блок обработки.

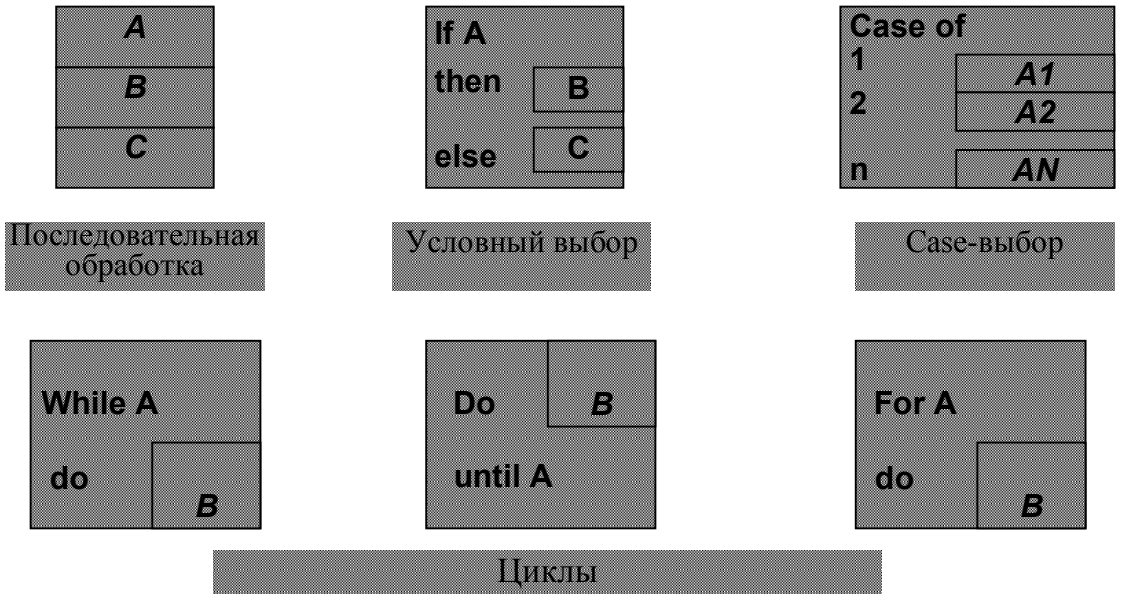


Рис. 3.8. - Символы FLOW-форм.

На рисунке 3.9 приведен пример использования данного подхода при проектировании СП, обеспечивающего упорядочивание определенным образом элементов массива и являющегося фрагментом алгоритма сортировки методом «поплавка» [20].

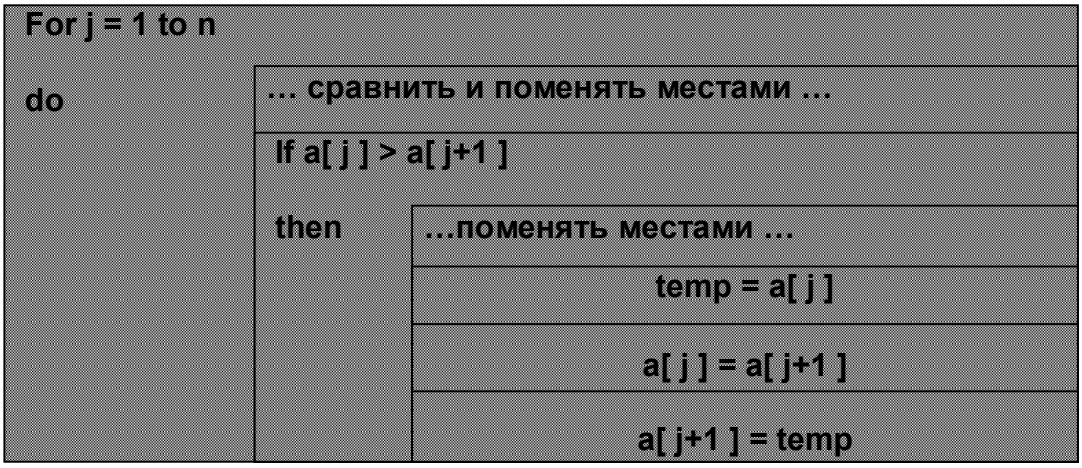


Рис. 3.9. - Пример FLOW-формы.

Дальнейшее развитие FLOW-формы получили в **диаграммах Насси-Шнейдермана.** На этих диаграммах символы последовательной обработки и цикла изображаются так же, как и соответствующие символы FLOW-форм. В символах условного выбора и case-выбора собственно условие располагается в верхнем треугольнике, выбираемые варианты – на нижних сторонах треугольника, а блоки обработки – под выбираемыми вариантами. Диаграмма Насси-Шнейдермана для вышеприведенного примера изображена на рисунке 3.10 [20].

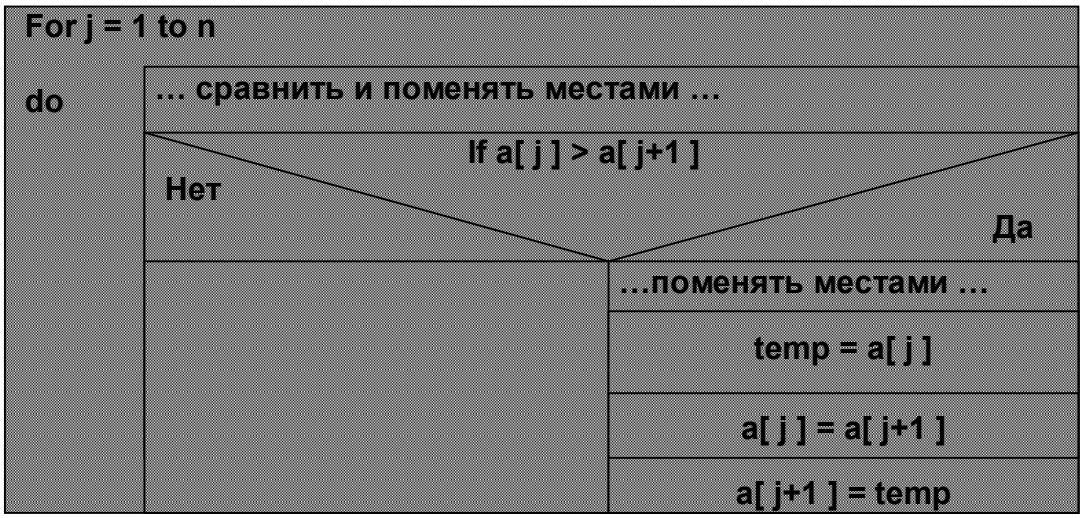


Рис. 3.10. - Пример диаграммы Насси-Шнейдермана.

3.1.2. Диаграммы «сущность-связь»: нотация Чена; нотация Баркера; построение модели

Диаграммы «сущность-связь» (***ERD***) предназначены для разработки моделей данных и обеспечивают стандартный способ определения данных и отношений между ними. С помощью ERD-диаграмм осуществляется детализация хранилищ данных моделируемой и проектируемой системы путем идентификации и документирования объектов (сущностей), важных для предметной области, свойств этих объектов (атрибутов) и их отношений с другими объектами (связей).

Нотация Чена

Нотация ERD-диаграмм была предложена Ченом. Нотация Чена предоставляет богатый набор средств моделирования данных, включая собственно ***ERD-диаграммы***, а также ***диаграммы атрибутов*** и ***диаграммы категоризации***. Эти диаграммные техники используются для моделирования и проектирования как реляционных, так и иерархических и сетевых баз данных.

Сущность представляет собой множество экземпляров реальных или абстрактных объектов (людей, событий, предметов, состояний, идей и т.п.), обладающих общими характеристиками (атрибутами). Любой объект системы мо-

жет быть представлен только одной сущностью, которая должна быть уникально идентифицирована. При этом имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр (например, ***Город,*** а не ***Москва***).

Отношение представляет собой связь между двумя и более сущностями. Именование отношения осуществляется с помощью грамматического оборота глагола ***(Имеет***, ***Определяет***, ***Может владеть*** и т.п.).

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

- обеспечение хранения информации в единственном месте (даже если она используется в различных комбинациях);
- использование этой информации различными приложениями. Символы ERD-диаграмм, соответствующие сущностям и отношениям, приведены на рис. 3.11.

Рис. 3.11. - Символы ERD-диаграмм в нотации Чена.

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

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

Для идентификации отношений, в которые вовлекаются сущности, используются ***связи.*** Каждая связь соединяет сущность и отношение и может быть направлена только от отношения к сущности. ***Значение связи*** характеризует ее тип и, как правило, выбирается из следующего множества:

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

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

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

На рисунке 3.12 приведен пример диаграммы «сущность–связь», моделирующей отношения между сущностями, содержащимися в хранилищах системы управления лифтом, в нотации Чена.

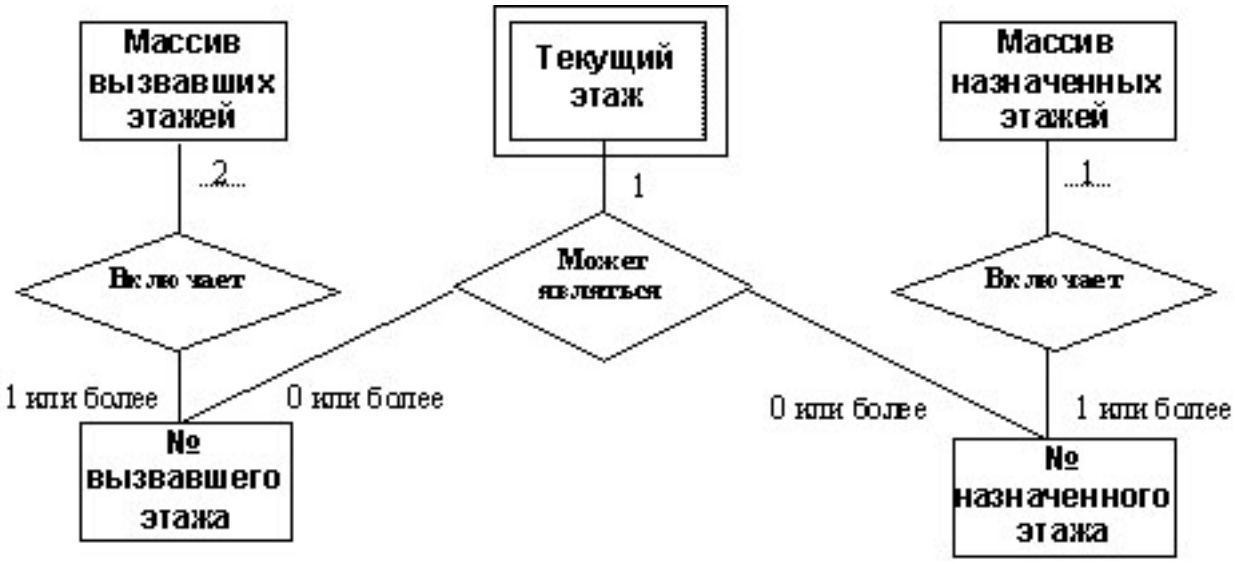


Рис. 3.12. - ERD-диаграмма в нотации Чена.

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

Детализация сущности осуществляется с использованием ***диаграммы атрибутов*,** которая раскрывает характеристики (атрибуты) сущности. Диаграмма атрибутов состоит из детализируемой сущности, соответствующих атрибутов и ***доменов*,** описывающих области значений атрибутов.

Пример диаграммы атрибутов, приведен на рис. 3.13. На диаграмме каждый атрибут представляется в виде связи между сущностью и соответствующим доменом, являющимся графическим представлением множества возможных значений атрибута. Все атрибутные связи имеют значения на своем окон-

чании. Для идентификации ключевого атрибута используется подчеркивание имени атрибута.

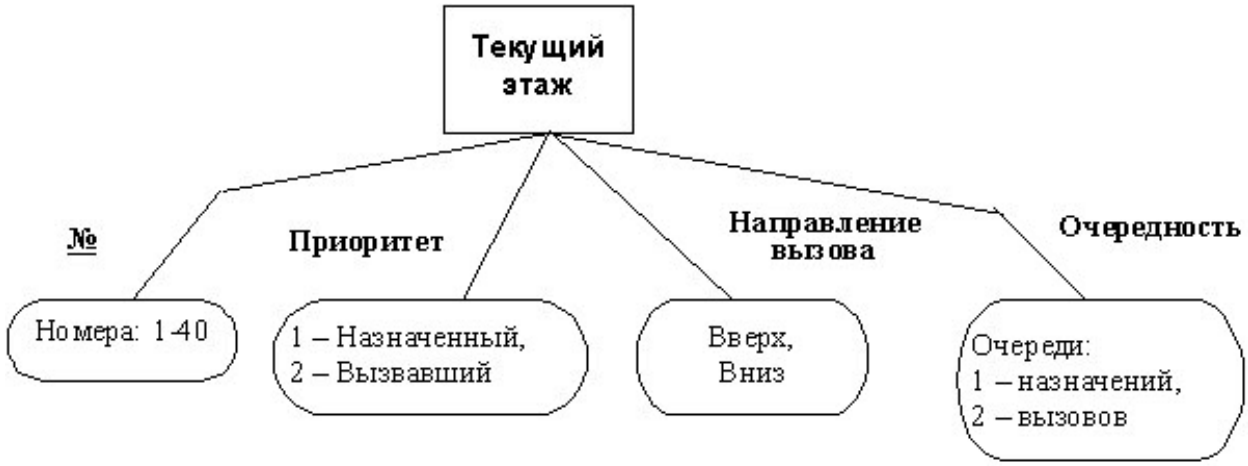


Рис. 3.13. - Диаграмма атрибутов.

Сущность может быть разделена и представлена в виде двух или более ***сущностей-категорий*****,** имеющих общие атрибуты и/или отношения. Эти атрибуты определяются однажды на верхнем уровне для разделяемой сущности (***общей сущности***) и наследуются на нижнем. Сущности-категории должны иметь и свои собственные атрибуты и/или отношения, а также, в свою очередь, могут быть декомпозированы своими сущностями-категориями на следующем уровне. На промежуточных уровнях декомпозиции одна и та же сущность может быть как общей сущностью, так и сущностью-категорией.

Для демонстрации декомпозиции сущности на категории используется ***диаграмма категоризации*****.** Такая диаграмма содержит общую сущность, две и более сущности-категории и специальный ***узел-дискриминатор*****,** который описывает способы декомпозиции сущностей (см. рис. 3.14).

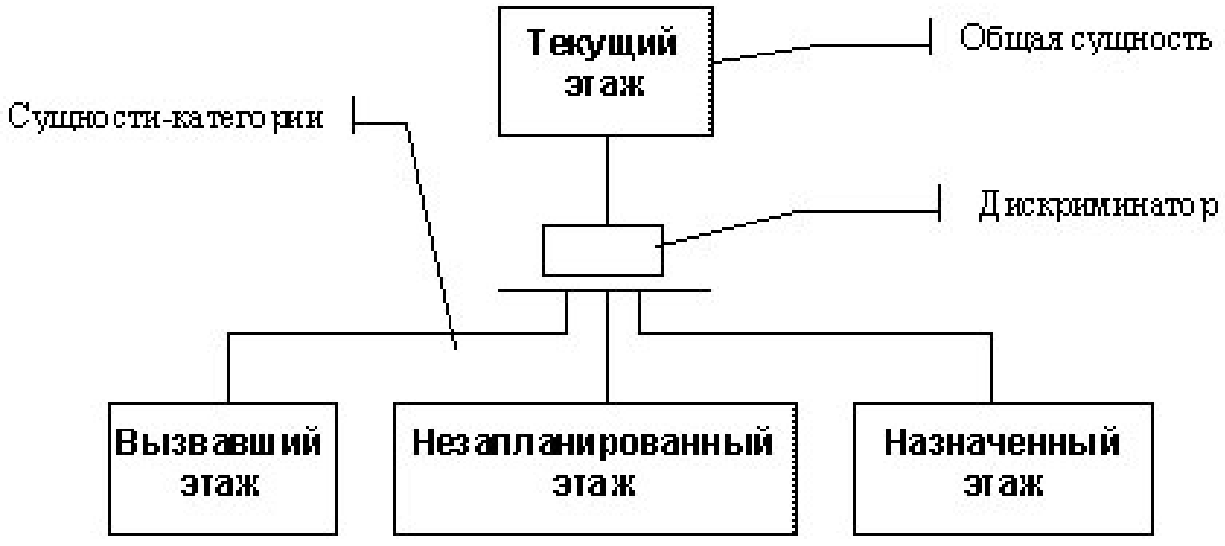


Рис. 3.14. - Диаграмма категоризации.

Существуют 4 возможных типа дискриминатора:

- ***E/M*** (exclusive/mandatory) – ***полное и обязательное вхождение*** – сущность должна быть одной и только одной из категорий декомпозиции.
- ***E/0*** (exclusive/optional) – ***полное и необязательное вхождение*** – сущность может быть одной и только одной из категорий декомпозиции.
- ***VM*** (inclusive/mandatory) – ***неполное и обязательное вхождение*** – сущность должна быть по крайней мере одной из категорий декомпозиции.
- ***I/O*** (inclusive/optional) – ***неполное и необязательное вхождение*** – сущность может быть, по крайней мере, одной из категорий декомпозиции.

Нотация Баркера

Дальнейшее развитие диаграммы «сущность-связь» получили в работах Баркера, предложившего оригинальную нотацию (см. рис. 3.15), которая позволила интегрировать средства описания моделей Чена.

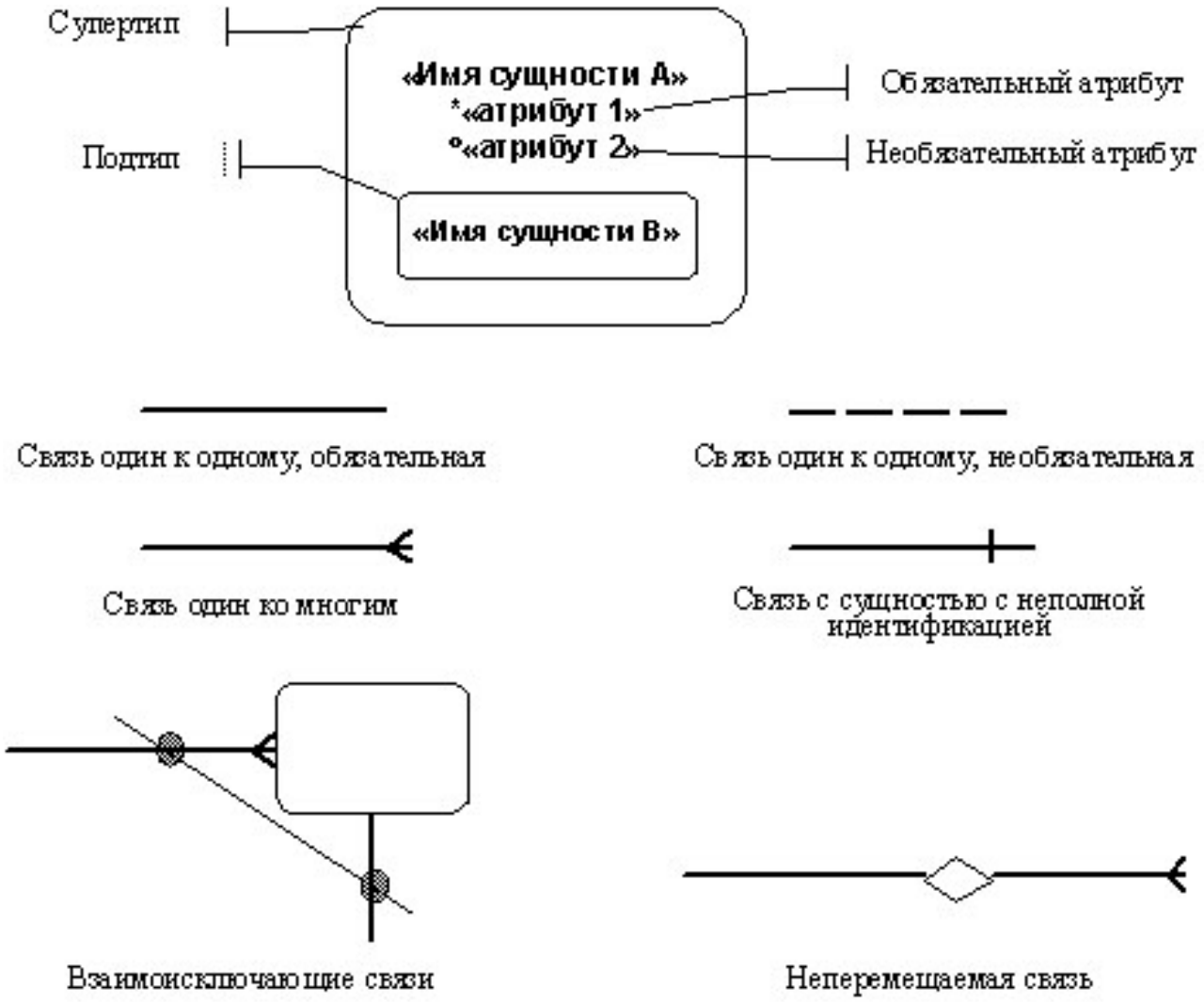


Рис. 3.15. - Символы ERD-диаграмм в нотации Баркера.

В нотации Баркера используется только один тип диаграмм: собственно ***ERD-диаграммы***. Сущность на ERD-диаграмме представляется прямоугольником любого размера, содержащим внутри себя имя сущности, список имен атрибутов (возможно, неполный) и указатели ключевых атрибутов (знак «**#**» перед именем атрибута). Понятия категории и общей сущности Чена заменяются Баркером на эквивалентные понятия ***подтипа*** и ***супертипа*** соответственно. Атрибуты сущностей могут быть обязательными и необязательными. Кроме того, атрибуты могут быть описательными (дескрипторами сущности) или входить в состав первичного ключа (уникального идентификатора сущности). Сущность, как правило, может быть полностью идентифицирована своим первичным ключом. Однако, могут быть случаи, когда в идентификации данной сущности участвуют также атрибуты другой сущности-родителя. Кроме первичных ключей могут использоваться альтернативные (возможные) ключи.

Все связи являются бинарными и представляются линиями, соединяющими родительскую сущность с одной или несколькими сущностями-потомками. Для связей должно быть определено имя (выражаемое грамматическим оборотом глагола), степень множественности (один или много объектов участвуют в связи) и степень обязательности (т.е. обязательная или необязательная связь между сущностями). Для множественной связи линия присоединяется к прямоугольнику сущности в трех точках, а для одиночной связи – в одной точке. При обязательной связи рисуется непрерывная линия до середины связи, при необязательной – пунктирная линия. Могут иметь место ситуации, при которых связи данной сущности рассматриваются как взаимоисключающие. Кроме того, связь может быть рекурсивной, а также неперемещаемой, если экземпляр сущности не может быть перенесен из одной связи в другую.

На рис. 3.16 приведен пример ERD-диаграммы, моделирующей отношения между сущностями, содержащимися в хранилищах системы управления лифтом (см. рис. 3.12), в нотации Баркера.

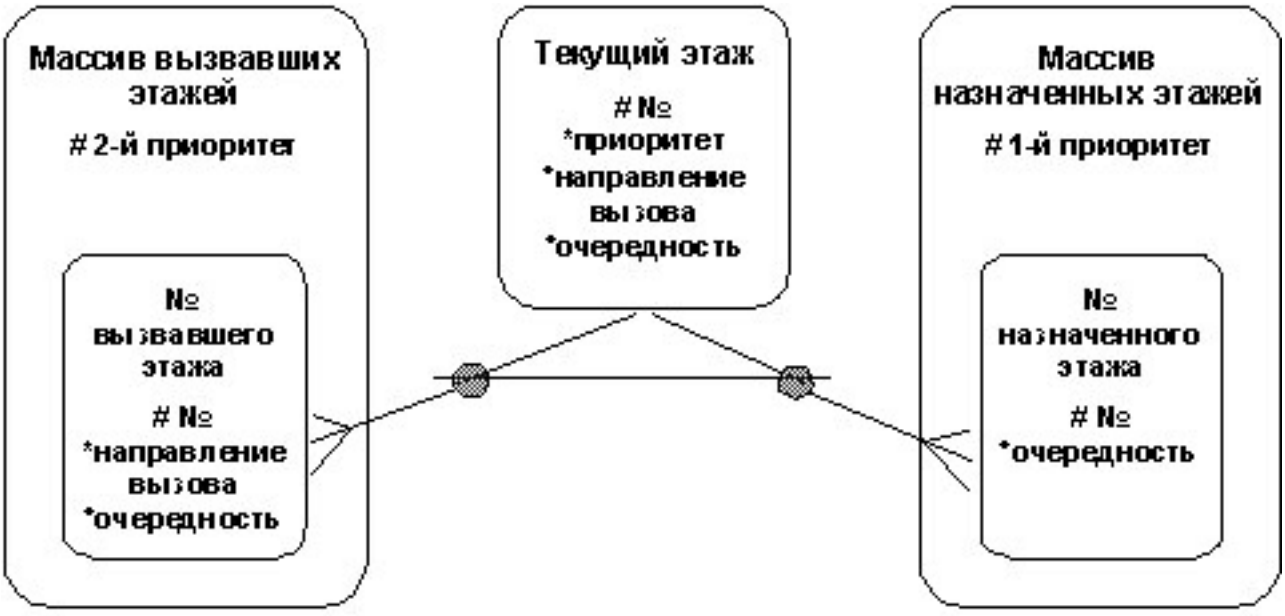


Рис. 3.16. - ERD-диаграмма в нотации Баркера.

Построение модели

Разработка ERD-диаграммы включает следующие основные этапы [20]:

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

Рассмотрим эти этапы в соответствии с работой [20] более подробно.

1-й этап построения ERD-диаграммы (или модели данных) является определяющим. Исходной информацией для данного этапа служит содержимое хранилищ данных, зависящее от входящих и выходящих потоков данных, представленных на DFD-диаграмме.

Таким образом, для построения ERD-диаграммы необходимо иметь предварительно построенную DFD-диаграмму. На рисунке 3.17 приведен фрагмент DFD-диаграммы, моделирующей деятельность бухгалтерии предприятия по ведению данных о персонале и начислению зарплаты.

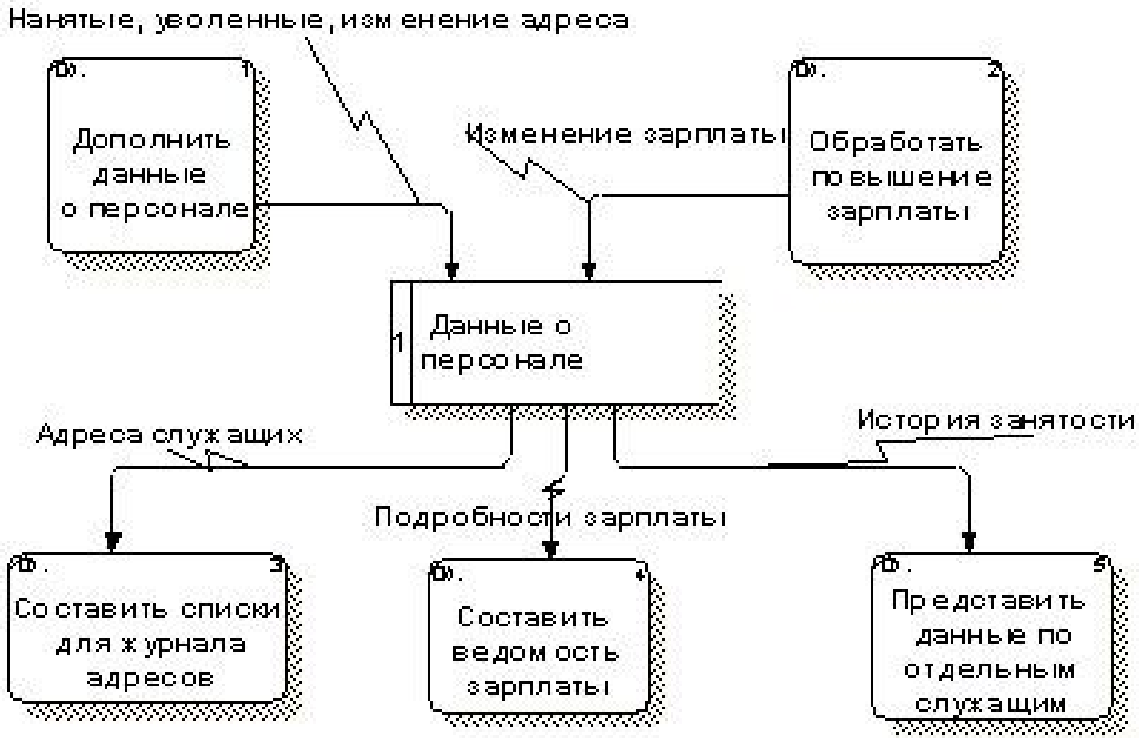


Рис. 3.17. - DFD-диаграмма деятельности бухгалтерии.

ERD-диаграмма является результатом анализа DFD-диаграммы. При этом первоначально осуществляется анализ хранилища, включающий сравнение содержимого входных и выходных потоков и создание на основе этого сравнения варианта схемы хранилища. Из диаграммы на рис. 3.17 следует, что хранилище должно содержать информацию о всех сотрудниках: имена, адреса, должности, оклады и т.д. Рассмотрим данные и их структуры, содержащиеся во входных и выходных потоках:

Таблица 3.4. Структуры данных для работы бухгалтерии.

ВХОДНЫЕ СТРУКТУРЫ ДАННЫХ	**ВЫХОДНЫЕ СТРУКТУРЫ ДАННЫХ**
НАНЯТЫЕ Дата_найма Фамилия таб_номер адрес должность начальная_зарплата **УВОЛЕННЫЕ** Фамилия таб_номер **ИЗМЕНЕНИЕ_АДРЕСА** Фамилия таб_номер старый_адрес новый_адрес **ИЗМЕНЕНИЕ_ЗАРПЛАТЫ** Фамилия таб_номер старая_зарплата новая_зарплата дата_изменения	**АДРЕС_СЛУЖАЩЕГО** фамилия адрес **ПОДРОБНОСТИ_З/ПЛ** фамилия таб_номер текущая_зарплата **ИСТОРИЯ_ЗАНЯТОСТИ** фамилия таб_номер дата_наима **ИСТОРИЯ_КАРЬЕРЫ** * должность дата_изменения **ИСТОРИЯ_ЗАРПЛАТЫ** * зарплата

Сравнивая входные и выходные структуры данных, отметим следующее:

- Поле «адрес» (элемент структуры входного потока «Нанятые») хранит текущий адрес сотрудника, а структура входного потока «Изменение адреса» хранит еще и старый адрес, что не является необходимым, с точки зрения выходных потоков.
- Для формирования структуры «История зарплаты» (составной части выходного потока «История занятости»), наоборот, требуется перечислить все оклады сотрудника, поэтому необходимо иметь набор, состоящий из пар («зарплата, дата»), а не просто «старая_зарплата» и «новая_зарплата» (как в структуре входного потока «Изменение зарплаты»).
- Аналогичная ситуация и со структурой «Историей карьеры» (другой составной части структуры выходного потока «История занятости»). Отметим, что на диаграмме вообще отсутствует входной поток со структурой данных, определяющей изменения в должности, что является серьезным упущением функциональной модели.
- Отметим, что изменение в должности обычно (но не всегда) соответствует изменению в зарплате.

С учётом этих моментов первый вариант схемы хранилища (базы данных) может выглядеть следующим образом:

фамилия
таб_номер
адрес
текущая_зарплата
дата_найма
история_карьеры *
должность
дата_изменения
история_зарплаты *
зарплата
дата_изменения

На следующем шаге осуществляется упрощение схемы за счет устранения избыточности.

Действительно, «***текущая_зарплата***» всегда является последней записью в «***история_зарплата***», а «***дата_найма***» по сути дела всегда содержится в разделах «***история_зарплаты***» и «***история_карьеры***». Кроме того, несколько дат в последних разделах одни и те же, поэтому целесообразно создать на их основе единую структуру «***история_зарплаты_карьеры***» и вводить в нее данные при изменении должности и/или зарплаты. Результатом приведенного рассуждения может быть схема:

фамилия
таб_номер
адрес
история_зарплаты_карьеры *
зарплата
должность
дата_изменения

Следующий шаг – упрощение схемы при помощи ***нормализации*** (удаления повторяющихся элементов). Основным способом нормализации является расщепление данной схемы на две схемы, являющиеся более простыми. Первая схема будет содержать фамилию и адрес (которые, как правило, не меняются), вторая – каждое изменение зарплаты и должности. Кроме того, каждая схема должна содержать «***таб_номер***» – единственный элемент данных, уникально идентифицирующий каждого сотрудника. Данные схемы и представляют те сущности, которые должны быть представлены в хранилище данных о персонале с учетом их связей и отношений.

Для идентификации сущностей определим ключевые атрибуты. Для первой схемы ключевым атрибутом является «***таб_номер***», для второй – ключом является совокупность атрибутов «***таб_номер***» и «***дата_изменения***», т.к. для каждого сотрудника возможно несколько записей в разделе «***история_зарплаты_карьеры***». Таким образом, хранилище данных о персонале содержит две сущности (рис. 3.18), которые в общепринятом виде могут быть представлены следующим образом:

сотрудник (таб_номер, фамилия, адрес)

история_зарплаты_карьеры (таб_номер, дата_изменения, должность, зарплата)

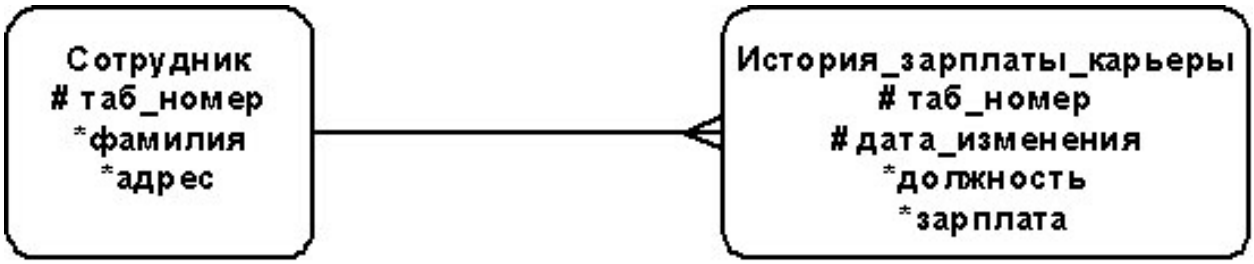


Рис. 3.18. – Сущности модели данных бухгалтерии.

Концепции и методы нормализации были разработаны Коддом (Codd), установившим существование трех типов нормализованных схем, названных в порядке уменьшения сложности ***первой, второй и третьей нормальной формой*** (соответственно, ***1НФ***, ***2НФ*** и ***3НФ***). Рассмотрим, как преобразовывать схемы к наиболее простой 3НФ.

Для примера построения 3НФ предварительно рассмотрим схему, ключ которой выбран в предположении, что заказчик не заказывает одну и ту же книгу дважды в один и тот же день [20]:

заказ_на книгу (имя_заказчика, дата_заказа, ISBN, название, автор, количество, цена, сумма_заказа)

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

В последней схеме атрибуты «название», «автор», «цена» зависят от части ключа (а именно, от «ISBN»), тогда как атрибут «количество» зависит от всего ключа. Это называется, соответственно, частичной и полной функциональной зависимостью от ключа. По определению схема находится в 2НФ, если она находится в 1НФ и все ее неключевые атрибуты полностью функционально зависят от ключа. Таким образом, для приведения схемы в 2НФ, необходимо избавится от частичной функциональной зависимости. После избавления от нее последняя схема будет выглядеть следующим образом:

заказ_на_книгу (имя заказчика, дата заказа, ISBN, количество, сумма_заказа)

книга (ISBN, автор, название, цена)

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

Иногда для построения 3НФ необходимо выразить зависимость между неключевыми атрибутами в виде отдельной схемы. Так для сотрудников, рабо-

тающих по различным проектам, возможна следующая схема [20]:

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

Очевидно, что данная схема находится в 2НФ. Однако «N_проекта» и «дата_окончания» являются зависимыми атрибутами. После расщепления схемы получим 3НФ:

участник проекта (таб_номер, телефон, почасовая_оплата, N_проекта)

проект (N_проекта, дата_окончания)

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

В завершении данного этапа зафиксируем алгоритм приведения ненормализованных схем в третью нормальную форму [20] (рис. 3.19).

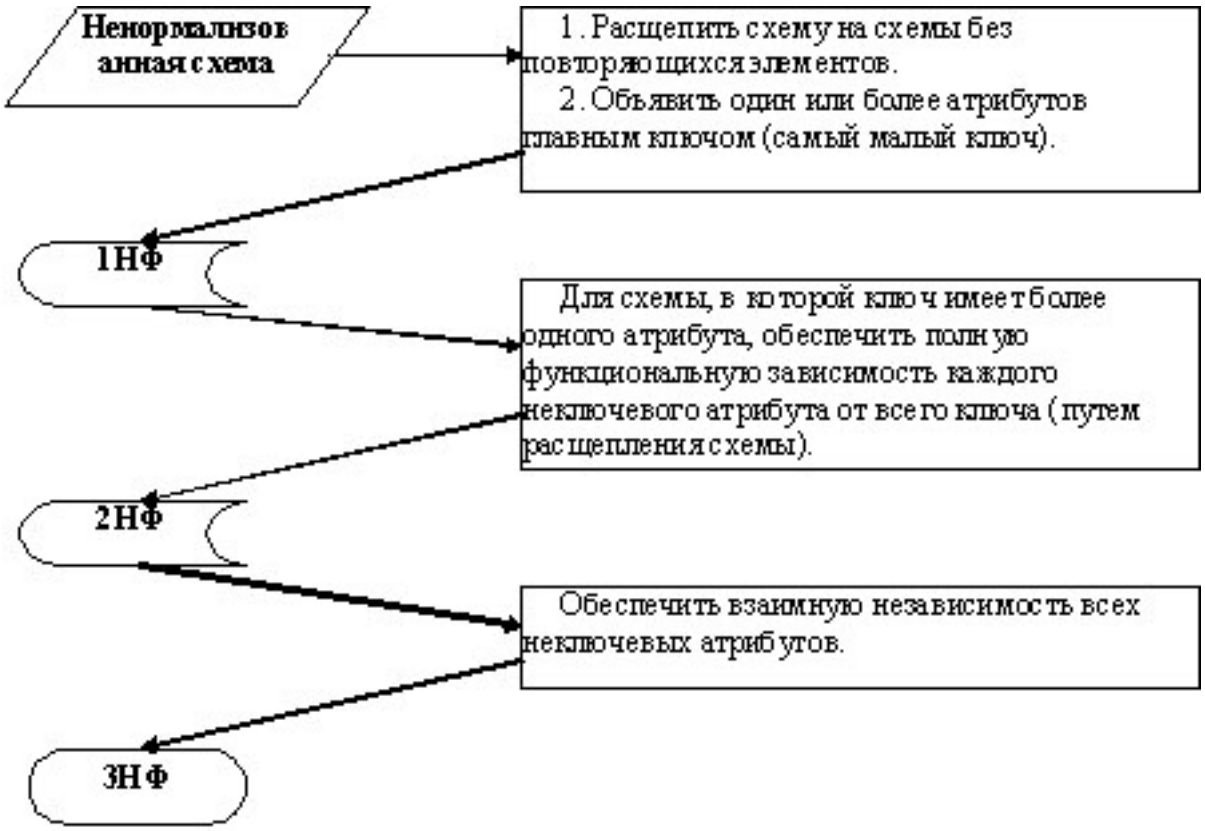


Рис. 3.19. – Алгоритм приведения к 3НФ.

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

Определение отношений включает выявление связей. Для этого отношение должно быть проверено в обоих направлениях следующим образом: выбирается экземпляр одной из сущностей и определяется, сколько различных экземпляров второй сущности может быть с ним связано, и наоборот. Для примера рассмотрим отношение между сущностями ***Сотрудник и История_зарплаты_карьеры***. У отдельного сотрудника должность и/или зарплата может меняться ноль, один или много раз, порождая соответствующее число экземпляров сущности ***История_зарплаты_карьеры***. Анализируя в другом направлении, видим, что каждый экземпляр сущности ***История_зарплаты_карьеры*** соответствует ровно одному конкретному сотруднику. Поэтому между этими двумя сущностями имеется отношение типа 1*n (один ко многим) со связью «один» на конце отношения у сущности ***Сотрудник*** и со связью «ноль, один или много» на конце у сущности ***История_зарплаты_карьеры***, что и показано на рис. 3.18.

3-й этап построения ERD-диаграммы (модели данных) предназначен для разрешения неспецифических (многие ко многим) отношений. Для этого каждое неспецифическое отношение преобразуется в два специфических отношения с введением новых (а именно, ассоциативных) сущностей. Рассмотрим пример из работы [20], представленный на рисунке 3.20.

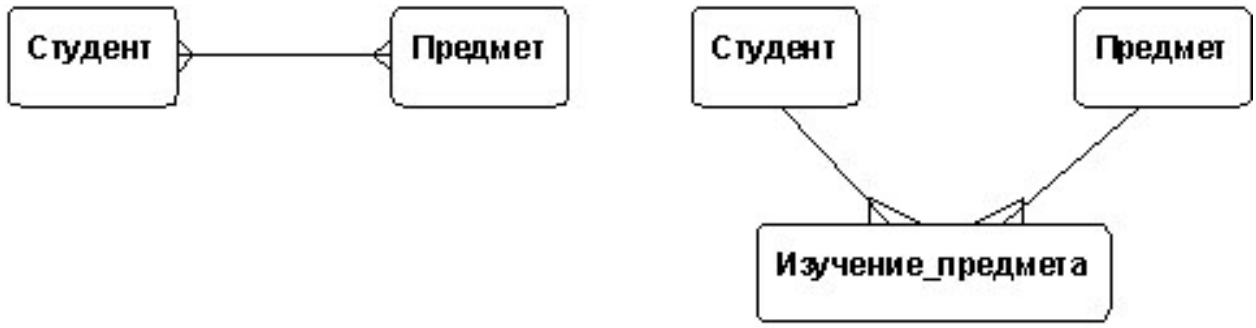


Рис. 3.20. – Разрешение неспецифического отношения.

Неспецифическое отношение на рис 3.20 указывает, что ***Студент*** может изучать много ***Предметов***, а ***Предмет*** может изучаться многими ***Студентами***. Однако мы не можем определить, какой студент изучает какой предмет, пока не введем для разрешения этого неспецифического отношения третью (ассоциативную) сущность ***Изучение_предмета***. Каждый экземпляр введенной сущности связан с одним ***Студентом*** и с одним ***Предметом***.

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

В заключении отметим, что нотация Баркера являются самой распространенной при построении ERD-диаграмм и используется в CASE-средстве ***Oracle Designer***.

3.1.3. Диаграммы переходов состояний

Диаграммы переходов состояний (***STD***) предназначены для моделирования и документирования аспектов систем, зависящих от времени или реакции на событие. Они позволяют осуществлять декомпозицию управляющих процессов и описывают отношения между входными и выходными управляющими потоками для управляющего процесса-предка.

С помощью STD-диаграмм можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования. Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены. STD-диаграмма состоит из следующих объектов [20]:

Состояние – набор характеристик, описывающих условия устойчивости системы. Находясь в определенном состоянии и имея информацию о прошлой истории системы, можно определить очередное состояние в зависимости от текущих входных событий (потоков). Имя состояния должно отражать реальную ситуацию, в которой находится система, например, ***Нагревание***, ***Охлаждение*** и т.п.

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

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

Таким образом, ***условие*** представляет собой событие (или события), вызывающее переход и идентифицируемое именем перехода. Если в условии участвует входной управляющий поток управляющего процесса-предка, то имя потока должно быть заключено в кавычки, например, «пароль»=999, где Пароль - входной поток управляющего процесса-предка.

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

«***Введенная карта***» = ***true*****,** где ***Введенная карта – выходной поток управляющего процесса-предка***.

Кроме того, для спецификации A-, T-, E/D-потоков (см. табл. 3.1) имя запускаемого или переключаемого процесса также должно заключаться в кавычки, например:

A: «***Получить пароль***» – активировать процесс Получить пароль.

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

На STD-диаграмме состояния представляются узлами, а переходы – дугами (рис. 3.21). Условия (по-другому называемые стимулирующими событиями) идентифицируются именем перехода и возбуждают его выполнение. Действия или отклики на события привязываются к переходам и записываются под соответствующим условием. Начальное состояние на диаграмме должно иметь входной переход, изображаемый потоком из подразумеваемого стартового узла (иногда этот стартовый узел изображается небольшим квадратом и привязывается к входному состоянию).

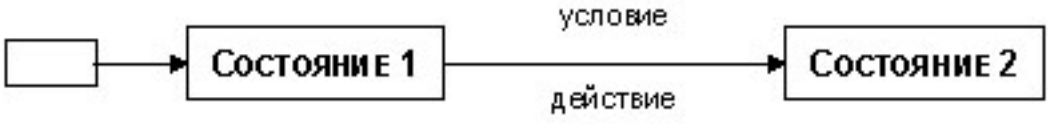


Рис. 3.21. - STD-диаграмма.

При построении STD-диаграммы рекомендуется [20] следовать перечисленным ниже правилам:

- строить STD-диаграмму на как можно более высоком уровне детализации DFD-диаграммы;
- строить как можно более простые STD-диаграммы;
- по возможности детализировать STD-диаграммы;
- использовать те же принципы именований состояний, событий и действий, что и при именовании процессов и потоков.

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

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

В качестве примера рассмотрим диаграмму переходов состояний для системы управления лифтом (рис. 3.22). Для этого используем DFD-диаграммы этой системы (рис. 3.2 – 3.7).

В ситуации, когда число состояний и/или переходов велико, для проектирования STD-диаграммы могут использоваться ***таблицы или матрицы переходов состояний***. Обе эти нотации позволяют зафиксировать ту же самую информацию, что и диаграммы переходов состояний, но в другом формате.

Матрица переходов состояний содержит по вертикали перечень состояний системы, из которых осуществляется переход, а по горизонтали – состояния, в которые осуществляется переход. При этом каждый элемент матрицы содержит соответствующие условия и действия, обеспечивающие переход из «вертикального» состояния в «горизонтальное». В качестве примера данного варианта матрицы переходов состояний приведена таблица 8, соответствующая диаграмме переходов состояний (рис. 21).

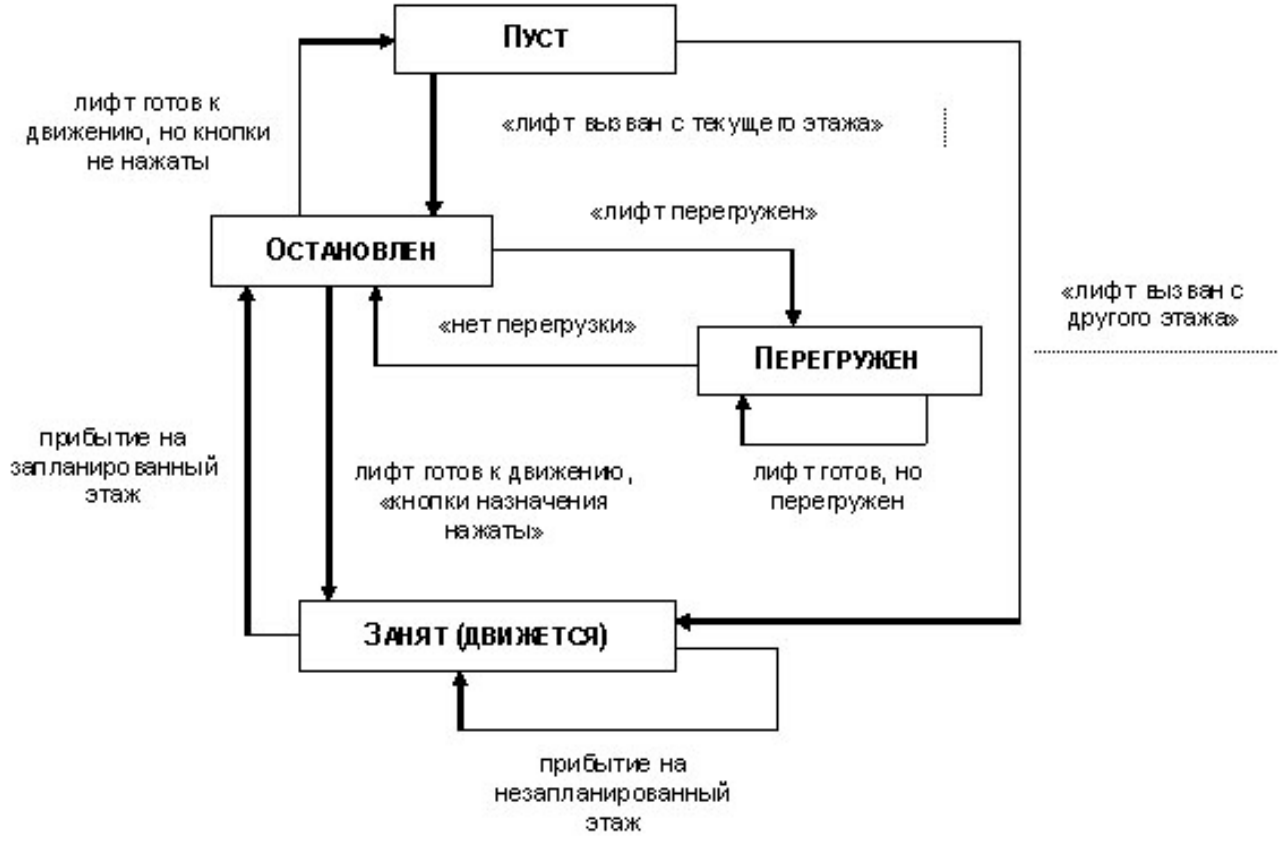


Рис. 3.22. - STD-диаграмма ECS.

В завершении рассмотрения технологии 3VM следует отметить, что описанные три вида диаграмм и набор дополнительных средств позволяют комплексно визуализировать, формализовать и документировать процессы и потоки в исследуемой системе. Данная технология является весьма полезным средством, однако, только в том случае, если в дальнейшем планируется автоматизация и разработка программных средств методами и языками процедурного программирования.

Таблица 3.5. Матрица переходов состояний ECS.

	Занят (движется)	**Остановлен**	**Пуст (стоит)**	**Перегружен (стоит)**
Занят (движется)	прибытие на незапланированный этаж	прибытие на запланированный этаж		
Остановлен	лифт готов к движению, «кнопки назначения нажаты»		лифт готов к движению, но кнопки не нажаты	«лифт перегружен»
Пуст (стоит)	«лифт вызван с другого этажа»	«лифт вызван с текущего этажа»		
Перегружен (стоит)		«нет перегрузки»		Лифт готов, но перегружен

Отсутствие средств представления характеристик объектов, реализующих процессы и потоки, делает невозможным применение данной технологии при объектно-ориентированной разработке информационной системы.

Выводы

1. Для решения сложных проблем используются методы системного анализа, предоставляющие в распоряжение аналитика визуальные графоаналитические средства для построения моделей бизнес-систем и бизнес-процессов.
2. Технология системно-структурного анализа 3VM позволяет представить модель системы в виде трех взаимосвязанных диаграмм: функциональной (потоков данных – DFD), информационной (сущность-связь – ERD), динамической (переходов состояний – STD).
3. Данная технология является весьма полезным средством в случае автоматизации и разработки программных средств методами и языками процедурного программирования. Применение данной технологии при объектно-ориентированной разработке информационной системы нецелесообразно.

3.2. Стандарты системного моделирования и анализа серии «Icam DEFinition»

3.2.1. Стандарт функционального моделирования IDEF0

Далее рассмотрим стандартные методы системного структурного анализа, описанные серией федеральных стандартов США «***Icam Definition***», в соответствии с [19, 20, 46 - 48, 98, 99]. Подробную информацию по всем стандартам этой серии можно найти на сайте http://www.idef.com.

Стандарт ***IDEF0*** (FIPS183) предназначен для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции. Данный документ представляет собой оформление (по инициативе Министерства обороны США) в виде стандарта технологии анализа сложных систем ***SADT*** (Structured Analysis and Design Technique), разработанной группой американских аналитиков во главе с Дугласом Россом в 1973 году.

Метод, предлагаемый стандартом IDEF0, предназначен для функционального моделирования, то есть моделирования выполнения функций объекта, путем создания описательной графической модели, показывающей что, как и кем делается в рамках функционирования предприятия. Функциональная модель представляет собой структурированное изображение функций производственной системы или среды, информации и объектов, связывающих эти функции. Модель строится методом декомпозиции: от крупных составных структур к более мелким, простым. Элементы каждого уровня декомпозиции представляют собой действия по переработке информационных или материальных ресурсов при определенных условиях с использованием заданных механизмов. Каждое действие раскладывается на более мелкие операции по переработке определенной части информационных или материальных ресурсов при определенных условиях с использованием части заданных механизмов. Аналогично раскладываются операции. Последний шаг декомпозиции должен приводить к получению модели, степень детализации которой удовлетворяет требованиям, заданным в самом начале процесса создания модели.

Методология IDEF0 основана на следующих положениях:

1. ***Система и модель***. Модель – искусственный объект, представляющий собой отображение (образ) системы и ее компонентов. ***M*** моделирует ***A***, если ***M*** отвечает на вопросы относительно ***A***. Здесь ***M*** – модель, ***A*** – моделируемый объект (оригинал). Модель разрабатывают для понимания, анализа и принятия решений о реконструкции (реинжиниринге) или замене существующей, либо проектировании новой системы. Система представляет собой совокупность взаимосвязанных и взаимодействующих частей, выполняющих некоторую полезную работу. Частями (элементами) системы могут быть любые комбинации разнообразных сущностей, включающие людей, информацию, программное обеспечение, оборудование, изделия, сырье или энергию. Модель описывает, что происходит в системе, как ею управляют, какие сущности она преобразует, какие средства использует для выполнения своих функций и что производит.

2. ***Блочное моделирование и его графическое представление***. Основной концептуальный принцип методологии IDEF – представление любой изучаемой системы в виде набора взаимодействующих и взаимосвязанных блоков, отображающих процессы, операции, действия, происходящие в изучаемой системе. В IDEF0 все, что происходит в системе и ее элементах, принято называть функциями. Каждой функции ставится в соответствие блок. На IDEF0-диаграмме (чаще говорят SADT-диаграмме), основном документе при анализе и проектировании систем, блок представляет собой прямоугольник. Интерфейсы, посредством которых блок взаимодействует с другими блоками или с внешней по отношению к моделируемой системе средой, представляются стрелками, входящими в блок или выходящими из него. Входящие стрелки показывают, какие условия должны быть одновременно выполнены, чтобы функция, описываемая блоком, осуществилась.
3. ***Строгость и формализм***. Разработка моделей IDEF0 требует соблюдения ряда строгих формальных правил, обеспечивающих преимущества методологии в отношении однозначности, точности и целостности сложных многоуровневых моделей. Эти правила описываются ниже с точки зрения технологии SADT. Здесь отмечается только основное из них: все стадии и этапы разработки и корректировки модели должны строго, формально документироваться с тем, чтобы при ее эксплуатации не возникало вопросов, связанных с неполнотой или некорректностью документации.
4. ***Итеративное моделирование***. Разработка модели в IDEF0 представляет собой пошаговую, итеративную процедуру. На каждом шаге итерации разработчик предлагает вариант модели, который подвергают обсуждению, рецензированию и последующему редактированию, после чего цикл повторяется. Такая организация работы способствует оптимальному использованию знаний системного аналитика, владеющего методологией и техникой IDEF0, и знаний специалистов – экспертов в предметной области, к которой относится объект моделирования.
5. ***Отделение «организации» от «функций»***. При разработке моделей следует избегать изначальной «привязки» функций исследуемой системы к существующей организационной структуре моделируемого объекта (предприятия, фирмы). Это помогает избежать субъективной точки зрения, навязанной организацией и ее руководством. Организационная структура должна явиться результатом использования (применения) модели. Сравнение результата с существующей структурой позволяет, во-первых, оценить адекватность модели, а во-вторых – предложить решения, направленные на совершенствование этой структуры.

Примеры задач, решаемых на основе методологии моделирования IDEF0:

- Анализ и реинжиниринг бизнес-процессов.
- Разработка информационной системы управления данными о качестве.
- Разработка регламентов и процедур обеспечения качества продукции и создания систем обработки данных о качестве. Функциональная модель

позволяет заменить традиционные руководства по качеству в виде описательных текстовых бумажных документов – стандартизованными электронными моделями, целостность и непротиворечивость которых поддерживается автоматически. При необходимости из них всегда можно получить бумажный отчет в привычном виде.

- Проектирование информационной инфраструктуры, процедур и регламентов информационного взаимодействия.
- Задачи анализа рисков в плане информационной безопасности.

Рассмотрим в соответствии с работой [47] принципы построения диаграмм по технологии SADT (IDEF0).

Графический язык SADT прост и гармоничен. В основе методологии лежат четыре основных понятия.

Первым из них является понятие «***функциональный блок***». Функциональный блок графически изображается в виде прямоугольника (см. рис. 3.23) и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, «***ПРОИЗВОДИТЬ УСЛУГИ***», а не «***производство услуг***»). Каждая из четырех сторон функционального блока имеет своё определенное значение (роль), при этом: верхняя сторона имеет значение «***управление***» (cont***r***ol); левая сторона имеет значение «***вход***» (***input***); правая сторона имеет значение «***выход***» (***output***); нижняя сторона имеет значение «***механизм***» (***mechanism***). Каждый функциональный блок в рамках единой рассматриваемой системы должен иметь свой уникальный идентификационный номер.

Рис. 3.23. - Функциональный блок на SADT-диаграмме.

Вторым «китом» методологии SADT является понятие «***интерфейсная дуга***». Также интерфейсные дуги часто называют потоками или стрелками. Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную данным функциональным блоком. Графическим отображением интерфейсной дуги является однонаправленная стрелка. Каждая интерфейсная дуга должна иметь свое уникальное наименование. По требованию стандарта, наименование должно быть оборотом существительного.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.).

В зависимости от того, к какой из сторон подходит данная интерфейсная дуга, она носит название «входящей», «исходящей» или «управляющей». Кроме того, «источником» (началом) и «приемником» (концом) каждой функциональной дуги могут быть только функциональные блоки, при этом «источником» может быть только выходная сторона блока, а «приемником» любая из трех оставшихся.

Необходимо отметить, что любой функциональный блок по требованиям стандарта должен иметь по крайней мере одну управляющую интерфейсную дугу и одну исходящую. Это и понятно – каждый процесс должен происходить по каким-то правилам (отображаемым управляющей дугой) и должен выдавать некоторый результат (выходящая дуга), иначе его рассмотрение не имеет никакого смысла.

При построении SADT-диаграмм важно правильно отделять входящие интерфейсные дуги от управляющих, что часто бывает непросто, так как имеется сходство природы входящих и управляющих интерфейсных дуг. Однако, для систем одного класса всегда есть определенные разграничения. Например, в случае рассмотрения предприятий и организаций существуют пять основных видов объектов: материальные потоки (детали, товары, сырье и т.д.), финансовые потоки (наличные и безналичные, инвестиции и т.д.), потоки документов (коммерческие, финансовые и организационные документы), потоки информации (информация, данные о намерениях, устные распоряжения и т.д.) и ресурсы (сотрудники, станки, машины и т.д.). При этом в различных случаях входящими и исходящими интерфейсными дугами могут отображаться все виды объектов, управляющими только относящиеся к потокам документов и информации, а дугами-механизмами только ресурсы.

Третьим основным понятием технологии SADT является понятие «***декомпозиция***». Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели. Декомпозиция позволяет постепенно и структурированно представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Модель SADT всегда начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой, и обозначается идентификатором «A-0».

В пояснительном тексте к контекстной диаграмме должна быть указана ***цель*** построения диаграммы в виде краткого описания и зафиксирована ***точка зрения***. Определение и формализация цели разработки SADT-модели является крайне важным моментом. Фактически цель определяет соответствующие об-

ласти в исследуемой системе, на которых необходимо фокусироваться в первую очередь. Например, если мы моделируем деятельность предприятия с целью построения в дальнейшем на базе этой модели информационной системы, то эта модель будет существенно отличаться от той, которую бы мы разрабатывали для того же самого предприятия, но уже с целью оптимизации логистических цепочек. Точка зрения определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему. Например, функциональные модели одного и того же предприятия с точек зрения главного технолога и финансового директора будут существенно различаться по направленности их детализации. Это связано с тем, что в конечном итоге, финансового директора не интересуют аспекты обработки сырья на производственных станках, а главному технологу ни к чему прорисованные схемы финансовых потоков. Правильный выбор точки зрения существенно сокращает временные затраты на построение конечной модели.

В процессе декомпозиции, функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы и называется дочерней по отношению к ней (каждый из функциональных блоков, принадлежащих дочерней диаграмме соответственно называется дочерним блоком). В свою очередь, функциональный блок-предок называется родительским блоком по отношению к дочерним блокам, а диаграмма, к которой он принадлежит – родительской диаграммой. Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. В каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок, или исходящие из него фиксируются на дочерней диаграмме. Этим достигается структурная целостность SADT-модели.

Часто бывают случаи, когда отдельные интерфейсные дуги не имеет смысла продолжать рассматривать в дочерних диаграммах ниже какого-то определенного уровня в иерархии, или наоборот – отдельные дуги не имеют практического смысла выше какого-то уровня. Например, интерфейсную дугу, изображающую «деталь» на входе в функциональный блок «Обработать на токарном станке» не имеет смысла отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия. С другой стороны, случается необходимость избавиться от отдельных «концептуальных» интерфейсных дуг и не детализировать их глубже некоторого уровня. Для решения подобных задач в технологии SADT предусмотрено понятие «**туннелирование**». Обозначение «туннеля» в виде двух круглых скобок вокруг начала интерфейсной дуги обозначает, что эта дуга не была унаследована от функционального родительского блока и появилась (из «туннеля») только на этой диаграмме. В свою очередь, такое же обозначение вокруг конца (стрелки) интерфейсной дуги в непосредственной близи от блока-приёмника

означает тот факт, что в дочерней по отношению к этому блоку диаграмме эта дуга отображаться и рассматриваться не будет. Чаще всего бывает, что отдельные объекты и соответствующие им интерфейсные дуги не рассматриваются на некоторых промежуточных уровнях иерархии – в таком случае, они сначала «погружаются в туннель», а затем, при необходимости «возвращаются из туннеля».

Последним из специфических понятий SADT является понятие «***глоссарий***». Для каждого из элементов SADT-диаграмм: функциональных блоков, интерфейсных дуг – существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элементом. Набор, называемый ***глоссарием***, является описанием сущности данного элемента. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией. Если технологию SADT рассматривать как аналог технологии DFD, то глоссарий будет выполняет роль словаря данных, используемого при построении DFD-диаграммы.

Обычно SADT-модели несут в себе сложную и концентрированную информацию, и для того, чтобы ограничить их перегруженность и сделать удобочитаемыми, в соответствующем стандарте приняты соответствующие ограничения сложности:

- ограничение количества функциональных блоков на диаграмме тремя-шестью. Верхний предел (шесть) заставляет разработчика использовать иерархии при описании сложных предметов, а нижний предел (три) гарантирует, что на соответствующей диаграмме достаточно деталей, чтобы оправдать ее создание;
- ограничение количества подходящих к одному функциональному блоку интерфейсных дуг четырьмя.

Разумеется, строго следовать этим ограничениям вовсе необязательно, однако, они являются весьма практичными в реальной работе.

Технология SADT содержит набор процедур, позволяющих разрабатывать и согласовывать модель большой группой людей, принадлежащих к разным областям деятельности моделируемой системы. Обычно процесс разработки является итеративным и состоит из следующих условных этапов [47]:

- Создание черновика модели группой специалистов, относящихся к различным сферам деятельности предприятия. Эта группа в терминах SADT (IDEF0) называется авторами. Построение первоначальной модели является динамическим процессом, в течение которого авторы опрашивают компетентных лиц о структуре различных процессов. На основе имеющихся положений, документов и результатов опросов создается черновик модели.
- Распространение черновика для рассмотрения, согласований и комментариев. На этой стадии происходит обсуждение черновика модели с широким спектром компетентных лиц (в терминах SADT (IDEF0) – читателей) на предприятии. При этом каждая из диаграмм черновой модели пись-

менно критикуется и комментируется, а затем передается автору. Автор, в свою очередь, также письменно соглашается с критикой или отвергает её с изложением логики принятия решения и вновь возвращает откорректированный черновик для дальнейшего рассмотрения. Этот цикл продолжается до тех пор, пока авторы и читатели не придут к единому мнению.

– Официальное утверждение модели. Утверждение согласованной модели происходит руководителем рабочей группы в том случае, если у авторов модели и читателей отсутствуют разногласия по поводу ее адекватности. Окончательная модель представляет собой согласованное представление о предприятии (системе) с заданной точки зрения и для заданной цели.

Примеры контекстных SADT-диаграмм и их декомпозиции, подготовленный с применением инструментального пакета BPwin, представлен на рисунках 3.24 – 3. 28.

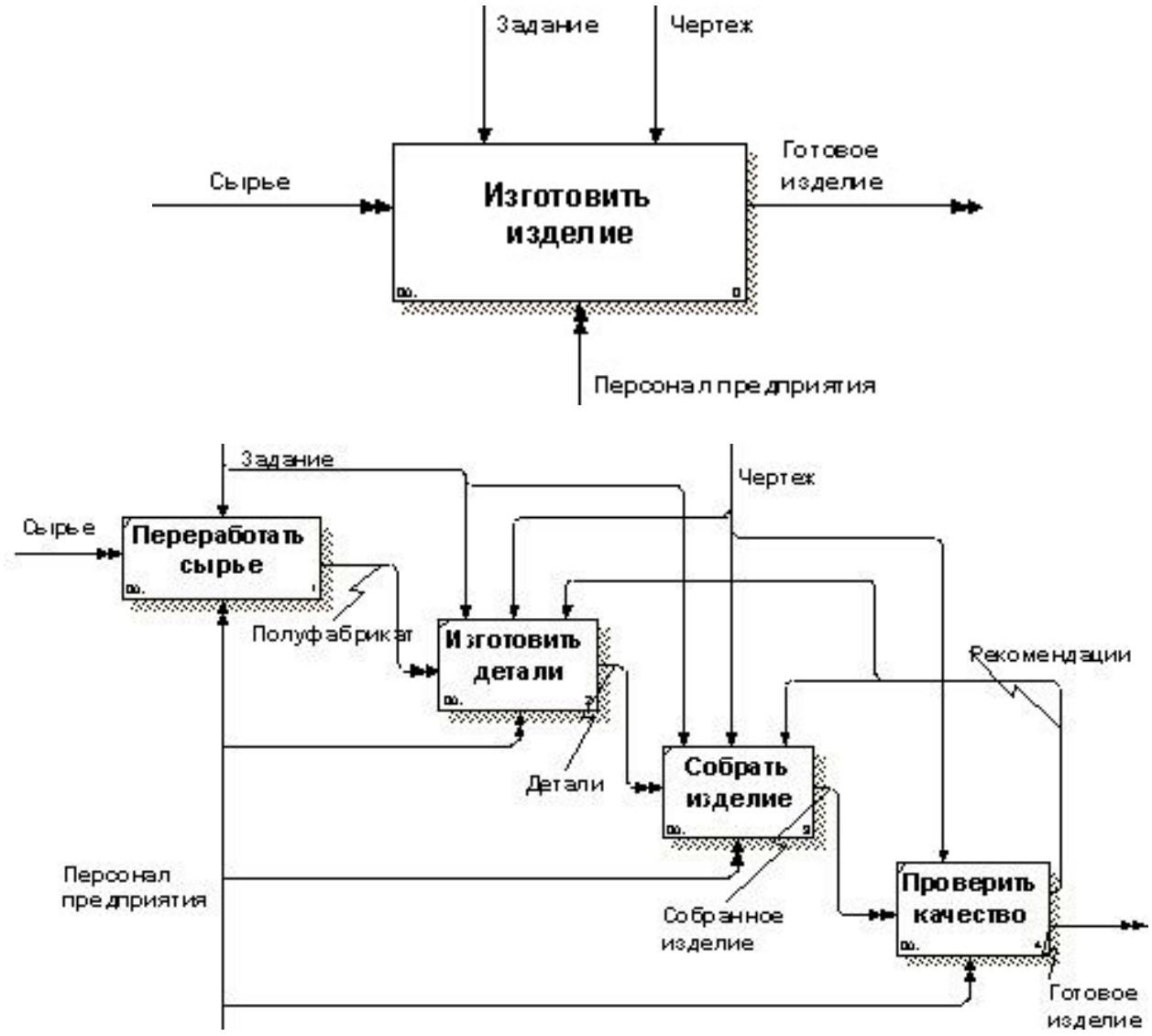


Рис. 3.24. - Контекстная SADT-диаграмма производства и ее декомпозиция.

Рис. 3.25. - Контекстная SADT-диаграмма ресторана.

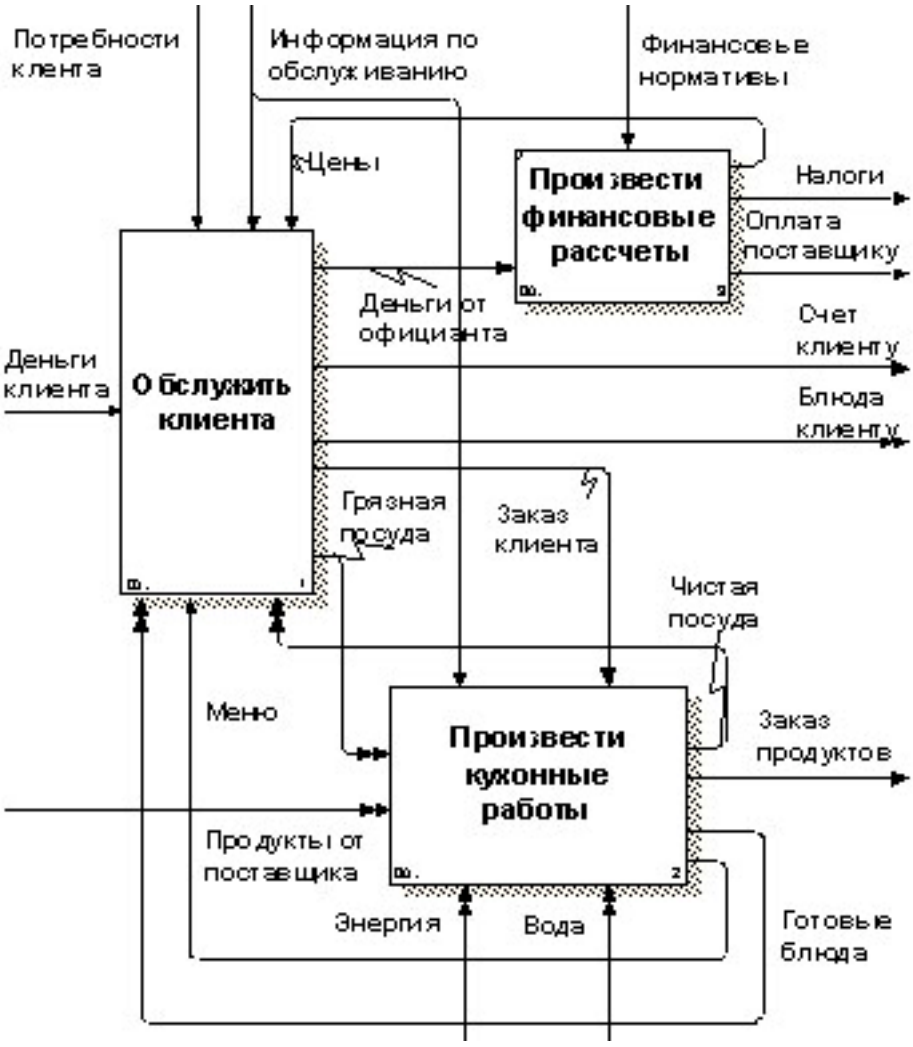


Рис. 3.26. - SADT-диаграмма ресторана первого уровня.

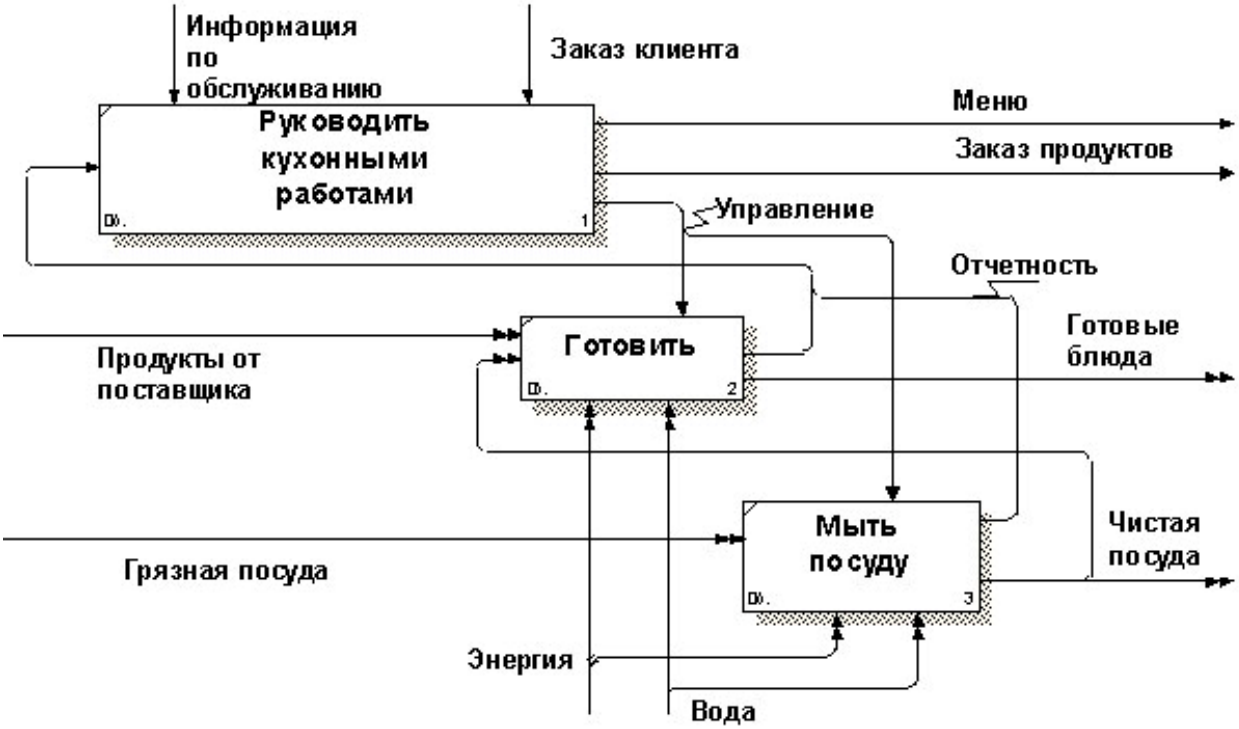


Рис. 3.27. - SADT-диаграмма ресторана второго уровня (кухонные работы).

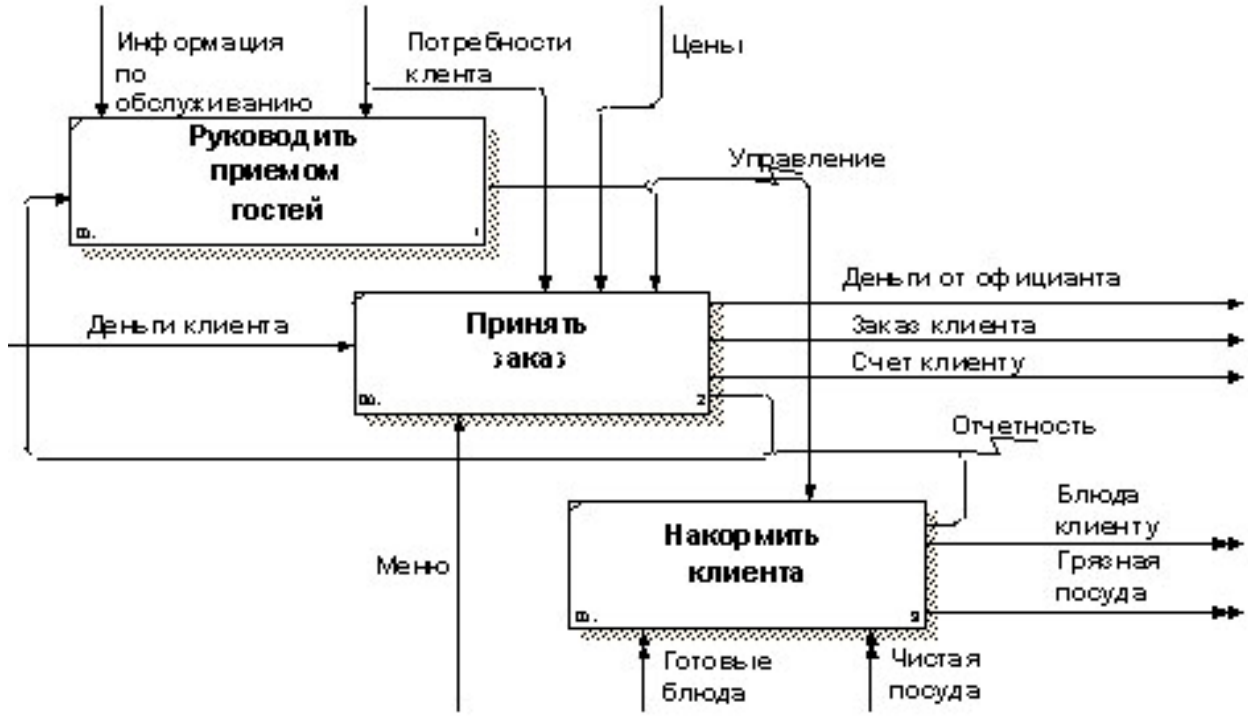


Рис. 3.28. - SADT-диаграмма ресторана второго уровня (обслуживание клиентов).

Наглядность графического языка SADT делает модель вполне читаемой и для лиц, которые не принимали участия в проекте ее создания, а также эффективной для проведения показов и презентаций. В дальнейшем, на базе построенной модели могут быть организованы новые проекты, нацеленные на производство изменений на предприятии (в системе).

3.2.2. Стандарт информационного моделирования IDEF1

Стандарт ***IDEF1*** (FIPS 184) предназначен для построения информационной модели, отображающей структуру и содержание информационных потоков, необходимых для поддержки функций системы.

Рассмотрим особенности данного стандарта, используя работу [48].

Деятельность любого предприятия можно представить как непрерывное изменение состояния физических и интеллектуальных объектов, имеющих отношение к предприятию, таких как сотрудники, средства производства, производимые продукты, идеи, финансы и т.д. Для эффективного менеджмента этим процессом, каждое изменение того или иного объекта должно иметь свое документальное отображение. Этими отображениями служат личные дела сотрудников, отчеты, рекламная продукция, служебные записки и т.д. Их совокупность называется ***информационной областью предприятия***. Движение информации (например, документооборот) и изменение ее называется ***информационными потоками***. Очевидно, что любому бизнес-процессу, а также любому изменению физических объектов должен соответствовать определенный информационный поток. Более того, руководство, при построении планов развития и управлении деятельностью предприятия, (издавая приказы, распоряжения и т.д.), фактически руководствуется информационными потоками и вносит в них изменения, осуществляя ***информационный менеджмент***.

Стандарт IDEF1 был разработан как инструмент анализа взаимосвязей между информационными потоками в рамках коммерческой деятельности предприятия. Целью подобного исследования является структуризация и, при необходимости, дополнение существующей информации, а также обеспечение качественного управления информационными потоками. Необходимость в подобной реорганизации информационной области, как правило, возникает на начальном этапе построения корпоративной информационной системы и методология IDEF1 позволяет наглядно обнаружить слабые места в существующей структуре информационных потоков. Применение IDEF1, как инструмента построения наглядной модели информационной структуры предприятия по принципу «Как должно быть», позволяет решить следующие задачи [48]:

- Выяснить структуру и содержание существующих потоков информации на предприятии.
- Определить какие проблемы, выявленные в результате функционального анализа и анализа потребностей, вызваны недостатком управления соответствующей информацией.
- Выявить информационные потоки, требующие дополнительного управления для эффективной реализации модели.

С помощью IDEF1 происходит изучение существующей информации о различных объектах в области деятельности предприятия. Характерно то, что IDEF1-модель включает в рассмотрение не только автоматизированные компоненты, базы данных и соответствующую им информацию, но также и реальные объекты, такие как сами сотрудники, кабинеты, телефоны и т.д. Назначение методологии IDEF1 состоит в том, чтобы выявить и четко постулировать потреб-

ности в информационном менеджменте в рамках деятельности предприятия. В отличие от методов разработки структур баз данных (например, IDEF1X или ERD-диаграмм), IDEF1 является аналитическим методом и используется преимущественно для выполнения следующих действий [48]:

- Определения самой информации и структуры ее потоков, имеющей отношение к деятельности предприятия.
- Определения существующих правил и законов, по которым происходит движение информационных потоков, и принципов управления ими.
- Выяснения взаимосвязей между существующими информационными потоками в рамках предприятия.
- Выявления проблем, возникающих вследствие недостатка качественного информационного менеджмента.

Результаты анализа информационных потоков могут быть использованы для стратегического и тактического планирования деятельности предприятия и улучшения информационного менеджмента. Однако основной целью использования методологии IDEF1 все же остается исследование движения потоков информации и принципов управления ими на начальном этапе процесса проектирования корпоративной информационно-аналитической системы, которая будет способствовать более эффективному использованию информационного пространства. Наглядные модели IDEF1 обеспечивают базис для построения мощной и гибкой информационной системы.

Методология IDEF1 позволяет на основе простых графических изображений моделировать информационные взаимосвязи между:

- реальными объектами;
- физическими и абстрактными зависимостями, существующими среди реальных объектов;
- информацией, относящейся к реальным объектам;
- структурой данных, используемой для приобретения, накопления, применения и управления информацией.

Одним из основных преимуществ методологии IDEF1 является обеспечение последовательного и строго структурированного процесса анализа информационных потоков. Другим отличительным свойством IDEF1 является широко развитая модульность, позволяющая эффективно выявлять и корректировать неполноту и неточности существующей структуры информации, на всем протяжении этапа моделирования.

При построении информационной модели проектировщик всегда оперирует с двумя основными глобальными областями, каждой из которой соответствует множество характерных объектов. Первой из этих областей является реальный мир, т.е. совокупность физических и интеллектуальных объектов таких, как люди, места, вещи, идеи и т.д., а также все свойства этих объектов и зависимости между ними. Второй же является информационная область. Она включает в себя существующие информационные отображения объектов первой области и их свойств. Информационное отображение, по существу, не является объектом реального мира, однако изменение его, как правило, является следст-

вием некоторого изменения соответствующего ему объекта реального мира. Методология IDEF1 разработана как инструмент для исследования статического соответствия вышеуказанных областей и установления строгих правил и механизмов изменения объектов информационной области при изменении соответствующих им объектов реального мира.

Центральным понятием методологии IDEF1 является понятие ***сущности***. При этом IDEF1 разделяет элементы структуры информационной области, их свойства и взаимосвязи на ***классы***. Класс сущностей представляет собой совокупность информации, накопленной и хранящейся в рамках предприятия и соответствующей определенному объекту или группе объектов реального мира. Основными концептуальными свойствами сущностей в IDEF1 являются:

- ***Уникальность.*** Любая сущность может быть однозначно идентифицирована из другой сущности.
- ***Устойчивость***. Информация, имеющая отношение к той или иной сущности постоянно накапливается.

Каждая сущность имеет своё ***имя*** и ***атрибуты***. Атрибуты представляют собой характерные свойства и признаки объектов реального мира, относящихся к определенной сущности. **Класс атрибутов** представляет собой набор пар, состоящих из имени атрибута и его значения для определенной сущности. Атрибуты, по которым можно однозначно отличить одну сущность от другой называются ***ключевыми атрибутами***. Каждая сущность может характеризоваться несколькими ключевыми атрибутами. ***Класс взаимосвязей*** в IDEF1 представляет собой совокупность взаимосвязей между сущностями. ***Взаимосвязь*** между двумя отдельными сущностями считается существующей в том случае, если класс атрибутов одной сущности содержит ключевые атрибуты другой сущности. Имя взаимосвязи всегда выражается в глагольной форме. Если же между двумя или несколькими объектами реального мира не существует установленной зависимости, то, с точки зрения IDEF1, между соответствующими им сущностями взаимосвязь также отсутствует. Каждый из вышеописанных классов имеет свое условное графическое отображение.

На рис. 3.29 приведен пример IDEF1-диаграммы. На ней представлены две сущности с именами «Подразделение ресторана» и «Сотрудник» и взаимосвязь между ними с именем «Работает в».

На рис. 3.30 показан пример анализа взаимосвязи информации при принятии заказа клиента официантом в ресторане.

3.2.3. Стандарт моделирования баз данных IDEF1X

Стандарт IDEF1 является методом изучения и анализа информации, в отличие от очень сходного по терминологии и семантике стандарта ***IDEF1X*** (в рамках которого используются ERD-диаграммы), предназначенного для разработки реляционных баз данных. IDEF1X создан на основе совершенствования стандарта IDEF1 с учетом таких требований, как простота для изучения и возможность автоматизации. IDEF1X-диаграммы используются в ряде распространенных CASE-средств (в частности, ERwin, Design/IDEF). В стандарте

IDEF1X, кроме того, уточнены основные понятия ERD-диаграмм.

Рис. 3.29. - Пример диаграммы IDEF1.

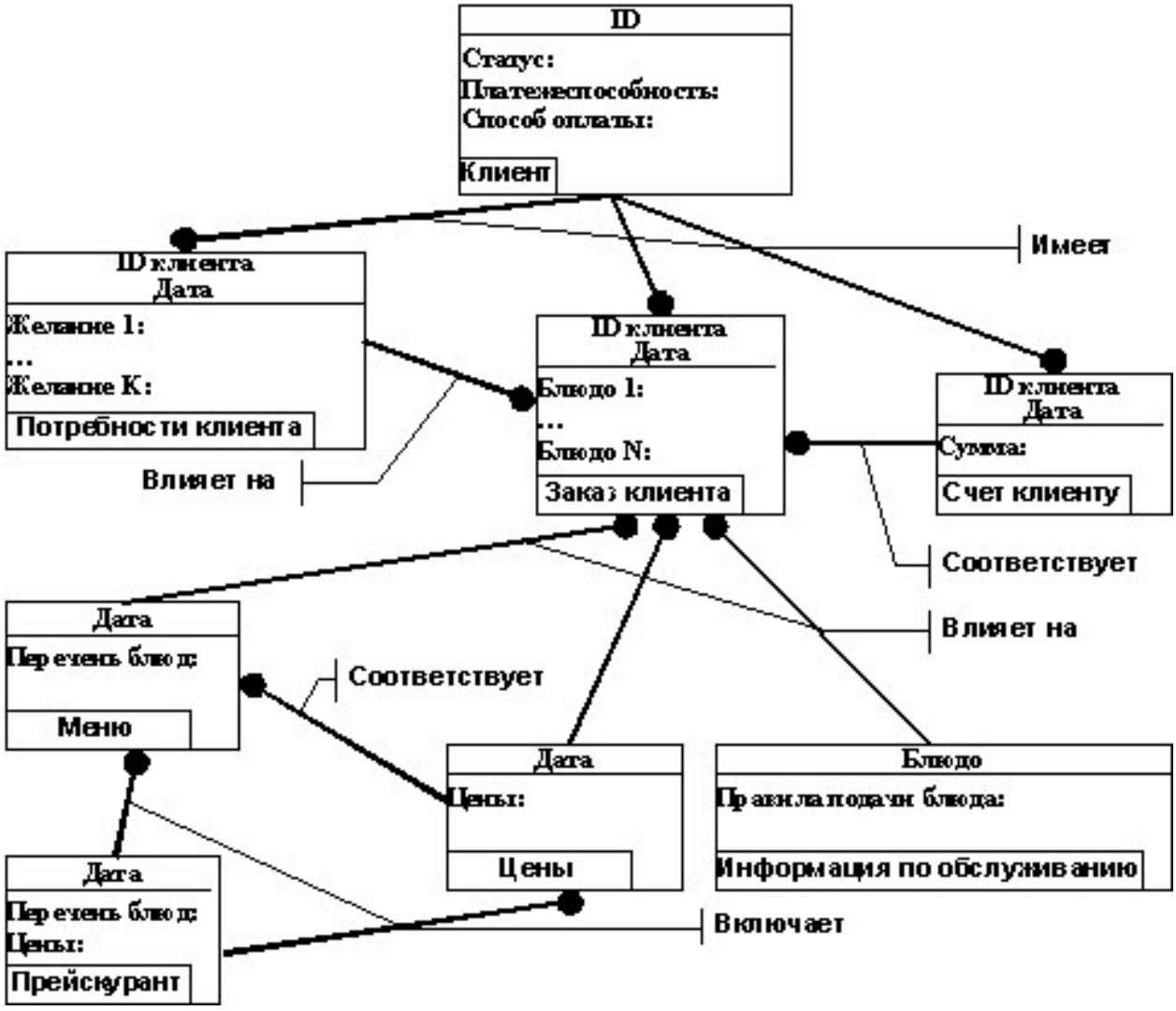


Рис. 3.30. - Диаграмма IDEF1, показывающая взаимосвязь информации при формировании заказа клиента официантом ресторана (см. рис. 3.28).

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

Каждая сущность должна обладать уникальным ***идентификатором***. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности. Каждая сущность должна обладать некоторыми свойствами:

- иметь уникальное ***имя***; к одному и тому же имени должна всегда применяться одна и та же интерпретация; одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- иметь один или несколько атрибутов, которые либо принадлежат сущности, либо наследуются через ***связь***;
- иметь один или несколько атрибутов, которые однозначно идентифицируют каждый экземпляр сущности.

Каждая сущность может обладать любым количеством связей с другими сущностями модели.

Связь (***Relationship***) – поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь – это ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, и наоборот.

Атрибут (***Attribute***) – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или свойств, ассоциированных с множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, предметов и т.д.). Экземпляр атрибута – это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым ***значением атрибута***. На диаграмме «сущность-связь» атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Сущность является ***независимой*** от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется ***зависимой*** от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности (рис. 3.31, 3.32). Каждой сущности присваиваются уникальные имя и номер, разделяемые косой чертой «/» и помещаемые над блоком.

Связь может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может по-

рождать каждый экземпляр сущности-родителя). В IDEF1X могут быть выражены следующие ***мощности связей***:

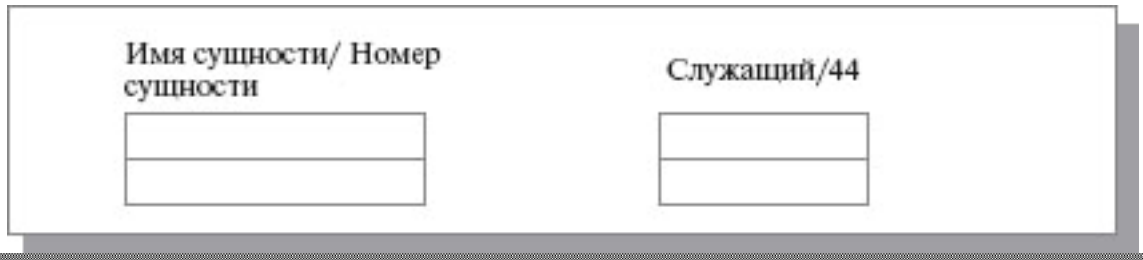


Рис. 3.31. - Независимые от идентификации сущности.

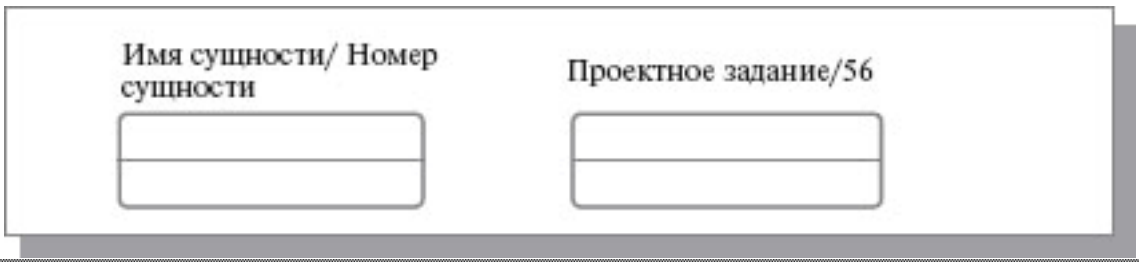


Рис. 3.32. - Зависимые от идентификации сущности.

- каждый экземпляр сущности-родителя может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется ***идентифицирующей***, в противном случае – ***неидентифицирующей***. Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка (рис. 3.33). Мощность связей может принимать следующие значения: ***N*** – ноль, один или более, ***Z*** – ноль или один, ***P*** – один или более. По умолчанию мощность связей принимается равной ***N***.

Идентифицирующая связь между сущностью-родителем и сущностью-потомком изображается сплошной линией. Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее связями с другими сущностями).

Пунктирная линия изображает неидентифицирующую связь (рис. 3.34). Сущность-потомок в неидентифицирующей связи будет не зависимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

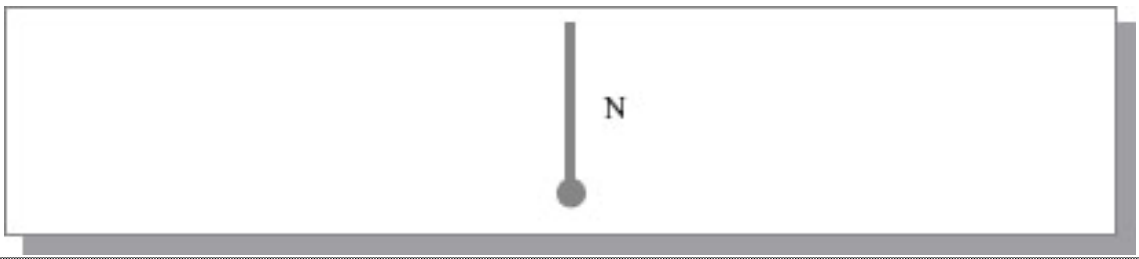


Рис. 3.33. - Графическое изображение мощности связи.

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие ***первичный ключ***, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой (рис. 3.34). Сущности могут иметь также ***внешние ключи*** (***Foreign Key***), которые могут использоваться в качестве части или целого первичного ключа или ***неключевого атрибута***. Для обозначения внешнего ключа внутрь блока сущности помещают имена атрибутов, после которых следуют буквы FK в скобках (рис. 3.34).

Рис. 3.34. - Неидентифицирующая связь.

3.2.4. Стандарт моделирования сценариев IDEF3.

Стандарт ***IDEF3*** предназначен для документирования технологических процессов, происходящих на предприятии, и предоставляет инструментарий для наглядного исследования и моделирования их ***сценариев***.

Рассмотрим особенности данного стандарта, используя работу [48].

Сценарием называется описание последовательности изменений свойств объекта, в рамках рассматриваемого процесса (например, описание последовательности этапов обработки детали в цеху и изменение её свойств после прохождения каждого этапа). Исполнение каждого сценария сопровождается соответствующим документооборотом, который состоит из двух основных потоков: документов, определяющих структуру и последовательность процесса (технологических указаний, описаний стандартов и т.д.), и документов, отображающих ход его выполнения (результатов тестов и экспертиз, отчетов о браке, и

т.д.). Для эффективного управления любым процессом, необходимо иметь детальное представление о его сценарии и структуре сопутствующего документооборота. Средства документирования и моделирования IDEF3 позволяют выполнять следующие задачи:

- Документировать имеющиеся данные о технологии процесса, выявленные, скажем, в процессе опроса компетентных сотрудников, ответственных за организацию рассматриваемого процесса.
- Определять и анализировать точки влияния потоков сопутствующего документооборота на сценарий технологических процессов.
- Определять ситуации, в которых требуется принятие решения, влияющего на жизненный цикл процесса, например изменение конструктивных, технологических или эксплуатационных свойств конечного продукта.
- Содействовать принятию оптимальных решений при реорганизации технологических процессов.
- Разрабатывать имитационные модели технологических процессов, по принципу «Как будет если...».

Существуют два типа диаграмм в стандарте IDEF3, представляющих описание одного и того же сценария технологического процесса в разных ракурсах. Диаграммы относящиеся к первому типу называются ***диаграммами Описания Последовательности Этапов Процесса*** (***Process Flow Description Diagrams – PFDD***), а ко второму – ***диаграммами Состояния Объекта в Процессе его Трансформации*** (***Object State Transition Network – OSTN***).

Предположим, требуется описать процесс окраски детали в производственном цеху на предприятии. С помощью диаграмм PFDD (см. рис. 3.35 и [48]) документируется последовательность и описание стадий обработки детали в рамках исследуемого технологического процесса. Диаграммы OSTN (см. рис. 3.36 и [48]) используются для иллюстрации трансформаций детали, которые происходят на каждой стадии обработки.

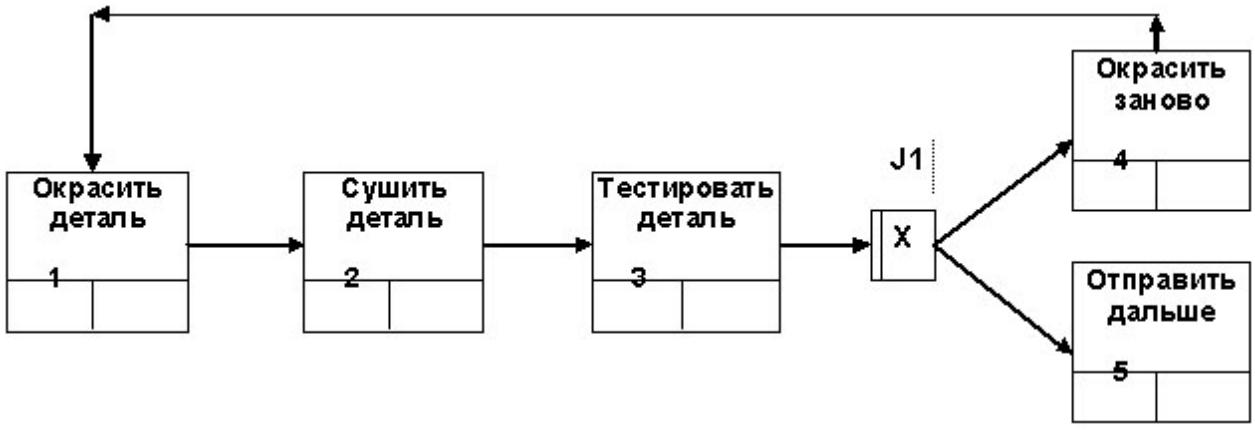


Рис. 3.35. - Пример диаграммы PFDD.

На рис. 3.35 изображена диаграмма PFDD, являющаяся графическим отображение сценария обработки детали. В целом, этот процесс состоит непосредственно из самой окраски, производимой на специальном оборудовании и этапа контроля ее качества, который определяет, нужно ли деталь окрасить заново (в

случае несоответствия стандартам и выявления брака) или отправить ее на дальнейшую обработку. Прямоугольники на диаграмме PFDD называются функциональными элементами, единицами работы или элементами поведения (***Unit of Behavior – UOB***) и обозначают событие, стадию процесса или принятие решения. Каждый UOB имеет свое имя, отображаемое в глагольном наклонении и уникальный номер. Стрелки являются отображением перемещения детали между UOB-блоками в ходе процесса. Они бывают следующих видов:

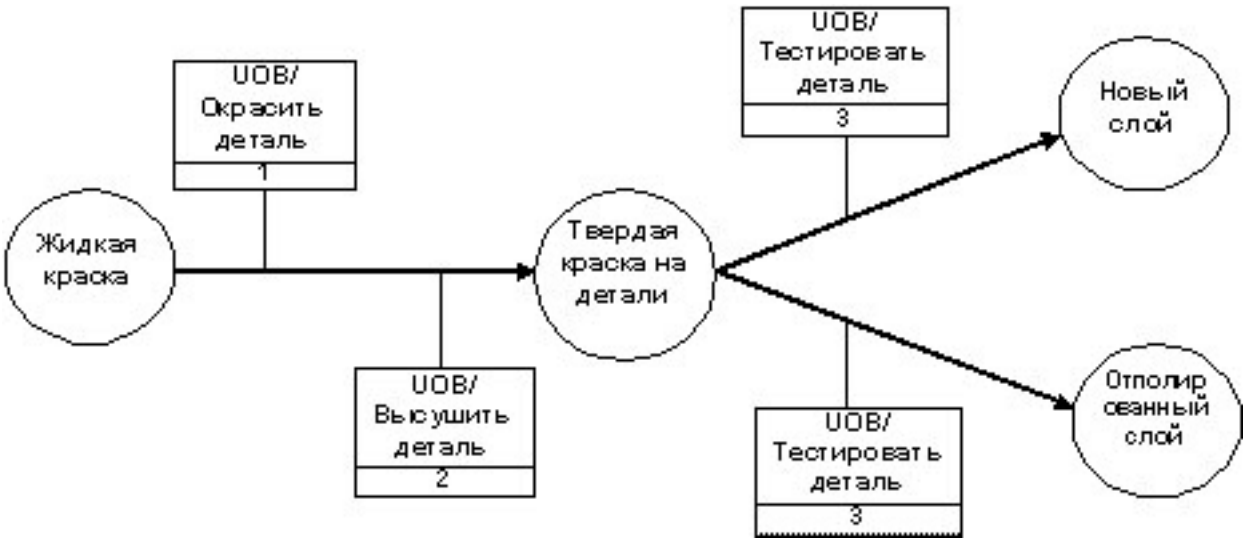


Рис. 3.36. - Пример диаграммы OSTN.

- Предшествование – сплошная одинарная стрелка, связывающая UOB в соответствии с последовательностью выполнения работ. Рисуется слева направо или сверху вниз.
- Отношение – пунктирная одинарная стрелка, использующаяся для изображения различного вида связей между UOB.
- Поток объектов – сплошная стрелка с двумя наконечниками, используемая для описания того факта, что объект, порождаемый одной работой, используется для выполнения другой.

Объект, обозначенный ***J1*** – называется ***перекрестком*** (***junction***). Перекрестки используются для отображения логики взаимодействия стрелок (потоков) при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки для слияния (fan-in junction) и разветвления (fan-out junction) стрелок. Все перекрестки в PFDD диаграмме нумеруются, каждый номер имеет префикс «J». Классификация возможных типов перекрестков приведена в таблице 3.6.

Каждый функциональный блок UOB может иметь последовательность декомпозиций, и, следовательно, может быть детализирован с любой необходимой точностью. Под декомпозицией понимается представление каждого UOB с помощью отдельной IDEF3 диаграммы.

Таблица 3.6. Возможные перекрестки в стандарте IDEF3.

Обозначение	***Наименование***	***Слияние стрелок (Fan-in Junction)***	***Разветвление стрелок (Fan-out Junction)***
	Asynchronous AND	Все предшествующие процессы должны быть завершены.	Все следующие процессы должны быть запущены.
	Synchronous AND	Все предшествующие процессы завершены одновременно.	Все следующие процессы запускаются одновременно.
	Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены.	Один или несколько следующих процессов должны быть запущены.
	Synchronous OR	Один или несколько предшествующих процессов завершаются одновременно.	Один или несколько следующих процессов запускаются одновременно.
	XOR Exclusive OR)	Только один предшествующий процесс завершен.	Только один следующий процесс запускается.

Например, можно декомпозировать UOB «Окрасить деталь», представив его отдельным процессом и построив для него свою PFDD диаграмму. При этом эта диаграмма будет называться дочерней, по отношению к изображенной на рисунке, а та, соответственно, родительской. Номера UOB дочерних диаграмм имеют сквозную нумерацию, т.е., если родительский UOB имеет номер «1», то блоки UOB на его декомпозиции будут соответственно иметь номера «1.1», «1.2» и т.д.

Если диаграммы PFDD представляет технологический процесс «с точки зрения наблюдателя», то другой класс диаграмм IDEF3 – OSTN позволяет рассматривать тот же самый процесс «с точки зрения объекта». Состояния объекта (в данном случае детали, см. рис. 3.36) и Изменение состояния являются ключевыми понятиями OSTN-диаграммы. Состояния объекта отображаются окружностями, а их изменения направленными одинарными стрелками. Каждая стрелка имеет ссылку на соответствующий функциональный блок UOB, в результате которого произошло отображаемое ею изменение состояния объекта. Таким образом, OSTN-диаграмма представляет собой аналог STD-диаграммы технологии 3VM.

С точки зрения основного примера, рассматриваемого в данном разделе (ресторана), стандарт IDEF3 является удобным средством анализа и документирования процессов (сценариев) приема гостей в различных ситуациях (например: ресторан «fast food» или вечерний ресторан; обычный обед/ужин, заказанное мероприятие, праздничный ужин, завтрак), а также технологических процессов приготовления различных блюд. На рисунках 3.37 и 3.38 представлен пример сценария приема гостей в вечернем ресторане при обслуживании обычного обеда/ужина (подготовлен с помощью BPwin (IDEF3)).

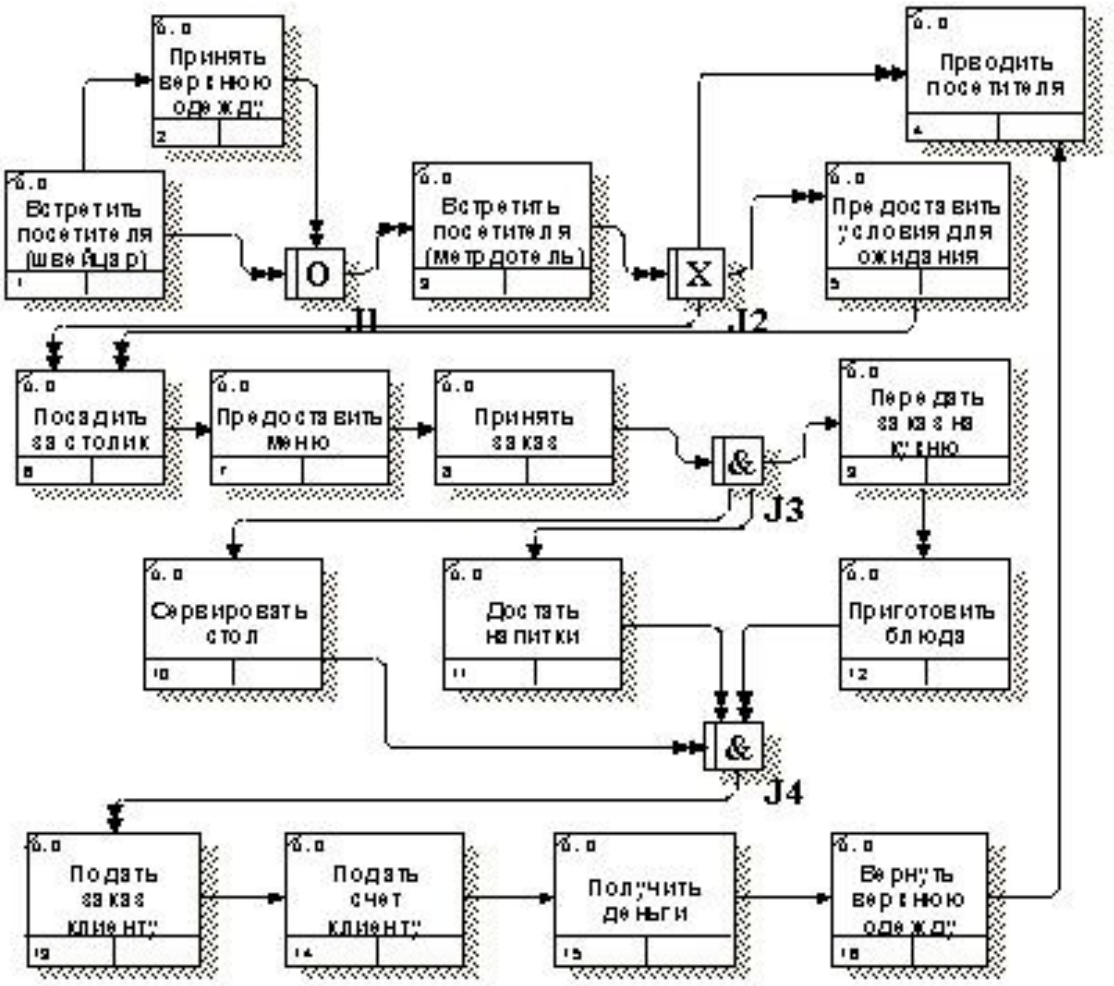


Рис. 3.37. - Пример PFDD-диаграммы сценария приема гостей в вечернем ресторане.

3.2.5. Стандарт моделирования онтологий IDEF5

Исторически, понятие онтологии появилось в одной из ветвей философии, называемой метафизикой, которая изучает устройство реального мира. Основной характерной чертой ***онтологического анализа*** является, в частности, разделение реального мира на составляющие его классы объектов и определение их онтологий, или же совокупности фундаментальных свойств, которые определяют их изменения и поведение. Таким образом, естественная наука представляет собой типичный пример онтологического исследования. Например, атомная физика классифицирует и изучает свойства наиболее фундаментальных объектов реального мира, таких как элементарные частицы, а биология, в свою очередь, описывает характерные свойства живых организмов, населяющих планету.

Однако фундаментальные и естественные науки не обладают достаточным инструментарием для того, чтобы полностью охватить область, представляющую интерес для онтологического исследования. Например, существует большое количество сложных формаций или систем, созданных и поддерживаемых человеком, таких как производственные фабрики, военные базы, коммерческие предприятия и т.д. Эти формации представляют собой совокупность взаимосвязанных между собой объектов и процессов, в которых эти объекты

тем или иным образом участвуют. Онтологическое исследование подобных сложных систем позволяет накопить ценную информацию об их работе, результаты анализа которой будут иметь решающее значение при проведении процесса реорганизации существующих и построении новых систем.

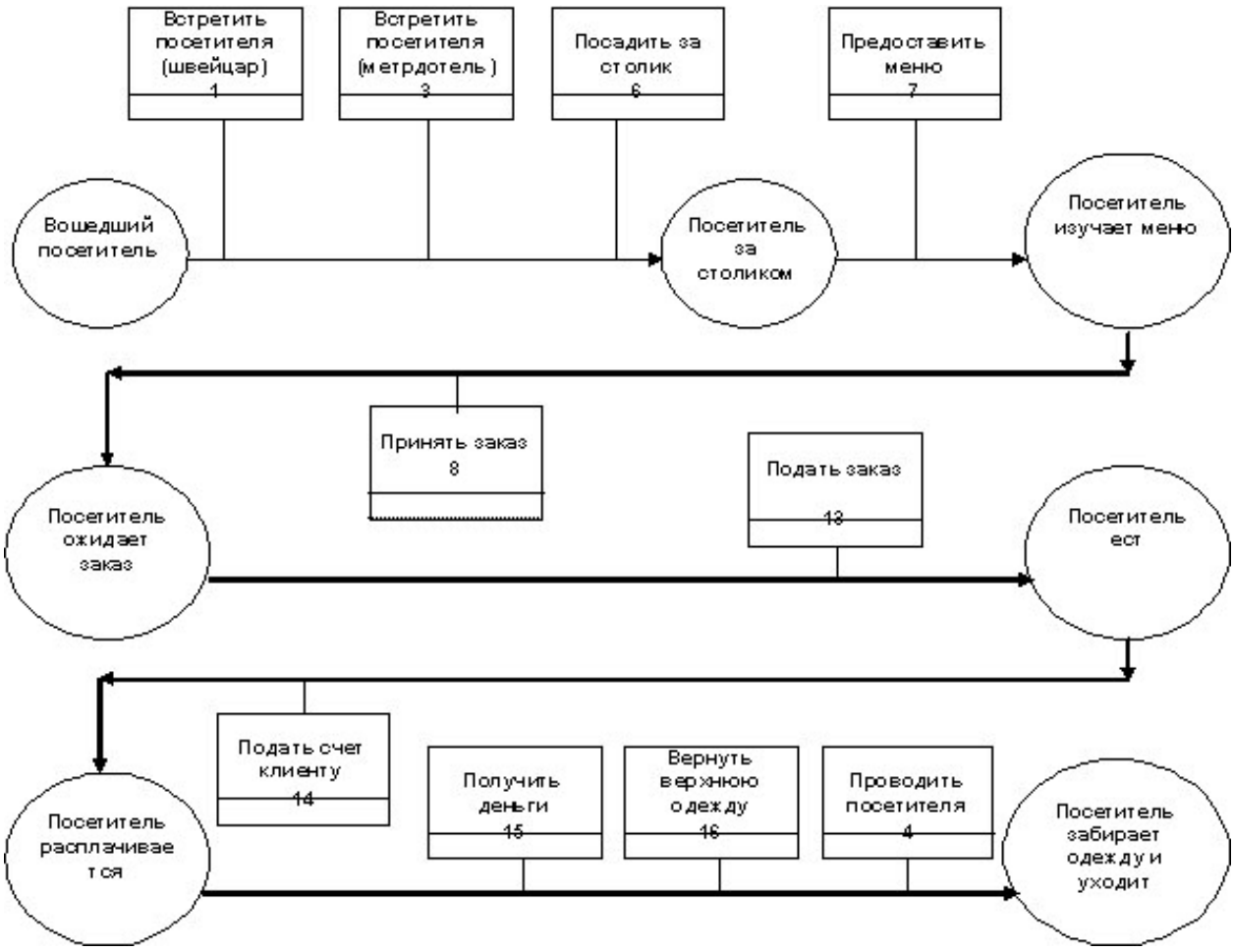


Рис. 3.38. - Пример OSTN-диаграммы сценария приема гостей в вечернем ресторане.

Методология ***IDEF5*** обеспечивает наглядное представление данных, полученных в результате обработки онтологических знаний, в простой естественной графической форме.

Рассмотрим особенности данного стандарта, используя работу [48].

Онтологический анализ обычно начинается с составления словаря терминов, который используется при обсуждении и исследовании характеристик объектов и процессов, составляющих рассматриваемую систему, а также создания системы точных определений понятий, соответствующих этим терминам. Кроме того, документируются основные логические взаимосвязи между понятиями. Результатом этого анализа является онтология системы или же совокупность терминов, точных определений их понятий и взаимосвязей между ними.

Таким образом, онтология включает в себя термины и правила, согласно которым эти термины могут быть скомбинированы для построения достоверных утверждений о состоянии рассматриваемой системы в некоторый момент времени. Кроме того, на основе этих утверждений, могут быть сделаны соот-

ветствующие выводы, позволяющие вносить изменения в систему, для повышения эффективности ее функционирования.

В любой системе существует две основные категории предметов: сами объекты, составляющие систему (физические и интеллектуальные) и взаимосвязи между этими объектами. В терминах онтологии, понятие взаимосвязи однозначно описывает или, другими словами, является точным дескриптором зависимости между объектами системы в реальном мире, а термины – являются, соответственно, точными дескрипторами самих реальных объектов.

При построении онтологии, в первую очередь происходит создание списка или базы данных дескрипторов и с помощью них, если их набор достаточен, создается модель системы. При этом существует огромное количество утверждений, достоверно отображающих состояние системы в различных аспектах, а построенная онтологическим способом модель должна выбирать из них наиболее полезные для эффективного рассмотрения в том или ином контексте. Кроме того, модель помогает описывать поведение объектов и соответствующие изменения взаимосвязей между ними или, другими словами, поведение системы. Таким образом, онтология представляет собой словарь данных, включающий в себя и терминологию, и модель поведения системы.

Процесс построения онтологии, согласно методологии IDEF5 состоит из пяти основных действий:

1. ***Изучение и систематизирование начальных условий***. В рамках этого действия устанавливаются основные цели и контексты проекта по разработке онтологии, а также распределяются роли между членами проекта.
2. ***Сбор и накапливание данных.*** На этом этапе происходит сбор и накапливание необходимых начальных данных для построения онтологии.
3. ***Анализ данных.*** Эта стадия заключается в анализе и группировке собранных данных и предназначена для облегчения построения терминологии.
4. ***Начальное развитие онтологии.*** На этом этапе формируется предварительная онтология, на основе отобранных данных. Создается и документируется словарь терминов. Описываются правила и ограничения, согласно которым на базе введенной терминологии могут быть сформированы достоверные утверждения, описывающие состояние системы.
5. ***Уточнение и утверждение онтологии.*** Заключительная стадия процесса. Построение модели, которая на основе существующих утверждений, позволяет формировать необходимые новые утверждения.

Для поддержания процесса построения онтологий в IDEF5 существуют специальные онтологические языки: ***схематический язык*** (Schematic Language – SL) и ***язык доработок и уточнений*** (Elaboration Language – EL). SL является наглядным графическим языком, специально предназначенным для изложения компетентными специалистами в рассматриваемой области системы основных данных в форме онтологической информации (рис. 3.39).

Этот несложный язык позволяет естественным образом представлять основную информацию в процессе развития онтологии и дополнять существующие онтологии новыми данными. Язык SL позволяет строить разнообразные типы диаграмм и схем в IDEF5. Основная цель всех этих диаграмм – наглядно и

визуально представлять основную онтологическую информацию. EL представляет собой структурированный текстовой язык, который позволяет детально характеризовать элементы онтологии.

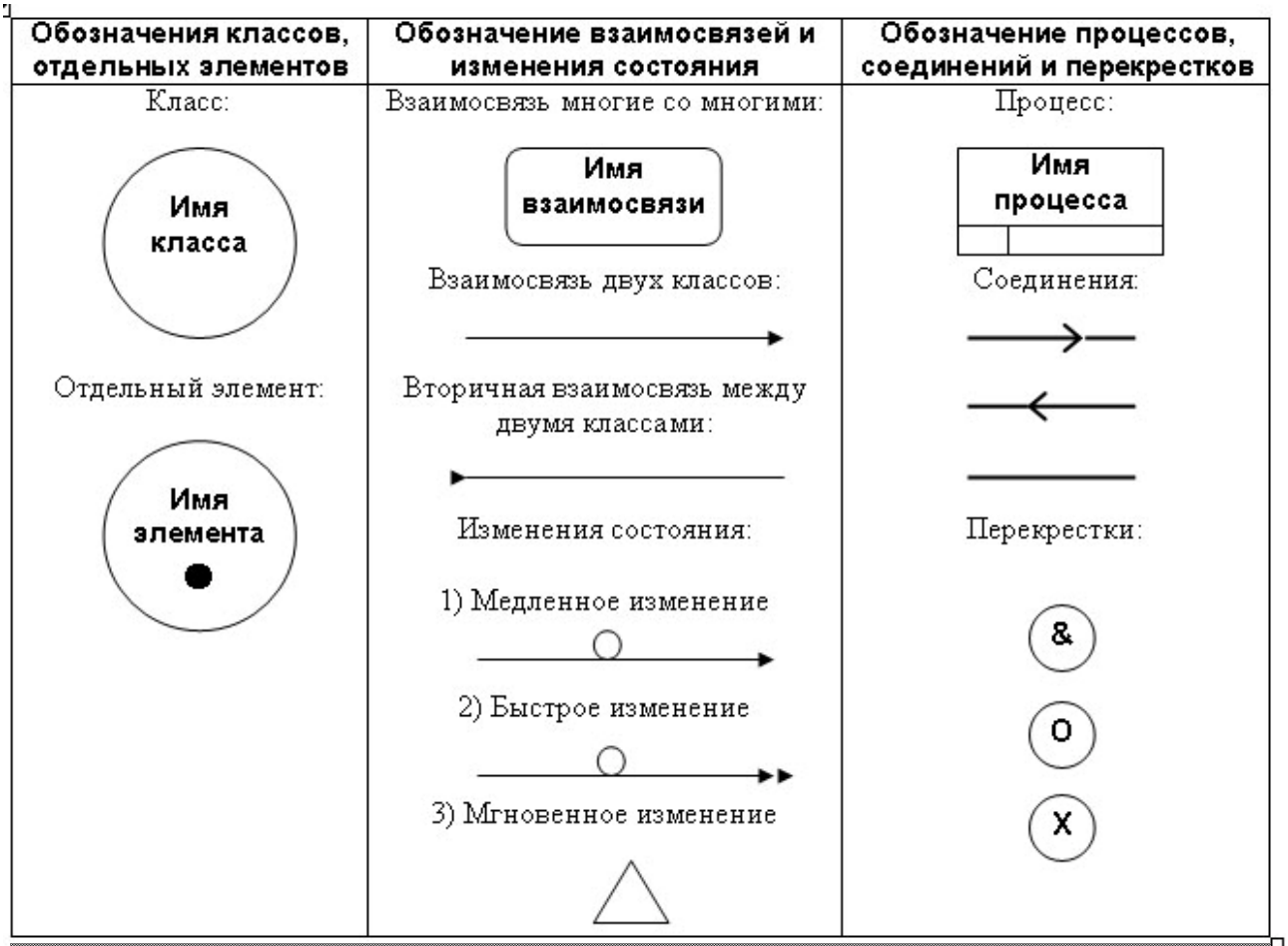


Рис. 3.39. - Графические обозначения IDEF5.

Несмотря на кажущееся сходство, семантика и обозначения схематического языка SL существенно отличается от семантики и обозначений других графических языков. Дело в том, что часть элементов графической схемы SL может быть изменена или вовсе не приниматься во внимание языком EL. Причина этого состоит в том, что основной целью применения SL является создание лишь вспомогательной структурированной конструкции онтологии и графические элементы SL не несут достаточной информации для полного представления и анализа системы, тем самым они не предназначены для сохранения на конечном этапе проекта. Тщательный анализ, обеспечение полноты представления структуры данных, полученных в результате онтологического исследования, являются задачей применения языка EL.

Как правило, наиболее важные и заметные зависимости между объектами всегда являются преобладающими, когда конкретные люди высказывают свои знания и мнения, касающиеся той или иной системы. Подобные взаимосвязи явным образом описываются языками IDEF5. Всего существует четыре основных вида схем, которые наглядно используются для накопления информации об онтологии в достаточно прозрачной графической форме.

1. ***Диаграмма классификации.*** Диаграмма классификации обеспечивает механизм для логической систематизации знаний, накопленных при изучении системы. Существует два типа таких диаграмм: ***диаграмма строгой классификации*** (Description Subsumption – DS) и ***диаграмма естественной или видовой классификации*** (Natural Kind Classification – NKC). Основное отличие диаграммы DS заключается в том, что определяющие свойства классов высшего и всех последующих уровней являются необходимым и достаточным признаком принадлежности объекта к тому или иному классу. На рисунке 3.40 приведен пример такой диаграммы, построенной на основе тривиальной возможности классификации многоугольников по количеству углов. Из геометрии известно точное математическое определение многоугольника, суть определяющих свойств родительского класса. Определяющим свойством каждого дочернего класса дополнительно является количество углов в многоугольнике. Очевидно, зная это определяющее свойство для любого многоугольника, можно однозначно отнести его к тому или иному дочернему классу. С помощью диаграмм DS, как правило, классифицируются логические объекты. Диаграммы естественной классификации или же диаграммы NKC, наоборот, не предполагают того, что свойства класса являются необходимым и достаточным признаком для принадлежности к ним тех или иных объектов. В этом виде диаграмм определение свойств класса является более общим. Пример такой диаграммы также приведен на рис. 3.40.

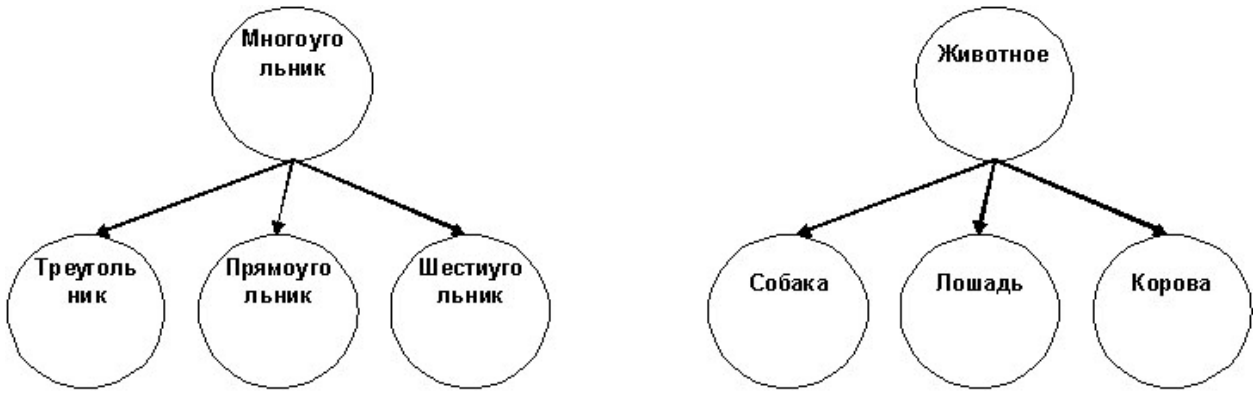


Рис. 3.40. - Пример диаграмм IDEF5: строгой классификации (слева), естественной классификации (справа).

2. ***Композиционная схема***. Композиционные схемы (Composition Schematic) являются механизмом графического представления состава объектов онтологии и фактически представляют собой инструменты онтологического исследования по принципу «Что из чего состоит». В частности, композиционные схемы позволяют наглядно отображать состав объектов, относящихся к тому или иному классу. На рисунке 3.41 изображена композиционная схема шариковой ручки, относящейся к классу шариковых автоматических ручек. В данном случае шариковая ручка является системой, к которой мы применяем методы онтологического ис-

следования. С помощью композиционной схемы мы наглядно документируем, что авторучка состоит из нижней и верхней трубки, нижняя трубка в свою очередь включает в себя кнопку и фиксирующий механизм, а верхняя трубка включает в себя стержень и пружину.

3. ***Схема взаимосвязей***. Схемы взаимосвязей (Relation Schematic) позволяют разработчикам визуализировать и изучать взаимосвязи между различными классами объектов в системе. В некоторых случаях схемы взаимосвязей используются для отображения зависимостей между самими же классовыми взаимосвязями. Мотивацией для развития подобной возможности послужило то тривиальное правило, что все вновь разработанные концепции всегда базируются на уже существующих и изученных. Это тесно согласуется с теорией, суть которой состоит в том, что изучение любой системы часто происходит от частного к общему, то есть, происходит изыскание и исследование новой частной информации, влияющее на конечные характеристики более общей концепции, к которой эта информация имела прямое отношение. Исходя из этой гипотезы, естественным способом изучения новой или плохо понимаемой взаимосвязи является соотнесение ее с достаточно изученной взаимосвязью.
4. ***Диаграмма состояния объекта***. Диаграмма состояния объекта (Object State Schematic) позволяет документировать тот или иной процесс с точки зрения изменения состояния объекта. В происходящих процессах могут произойти два типа изменения объекта: объект может поменять свое состояние или класс. Между этими двумя видами изменений по сути не существует принципиальной разницы: объекты, относящиеся к определенному классу в начальном состоянии в течение процесса могут перейти к дочернему или просто родственному классу. Например, полученная в процессе нагревания теплая вода, уже относится не к классу ВОДА, а к его дочернему классу ТЕПЛАЯ ВОДА. Однако при формальном описании процесса, во избежание путаницы, целесообразно разделять эти виды изменений и для такого разделения используется обозначение следующего вида: «**КЛАСС: СОСТОЯНИЕ**». Например, теплая вода будет описываться следующим образом: «вода: теплая», холодная – «вода: холодная» и так далее. Таким образом, диаграммы состояния в IDEF5 наглядно представляют изменения состояния или класса объекта в течение всего процесса. Пример такой диаграммы приведен на рис. 3.42.

Суммируя вышеизложенное, отметим, что строение и свойства любой системы могут быть эффективно исследованы и задокументированы при помощи следующих средств: словаря терминов, используемых при описании характеристик объектов и процессов, имеющих отношение к рассматриваемой системе, точных и однозначных определений всех терминов этого словаря и классификации логических взаимосвязей между этими терминами. Набор этих средств, по сути, и является онтологией системы, а стандарт IDEF5 предоставляет структурированную графоаналитическую методологию, с помощью кото-

рой можно наглядно и эффективно разрабатывать, поддерживать и изучать эту онтологию.

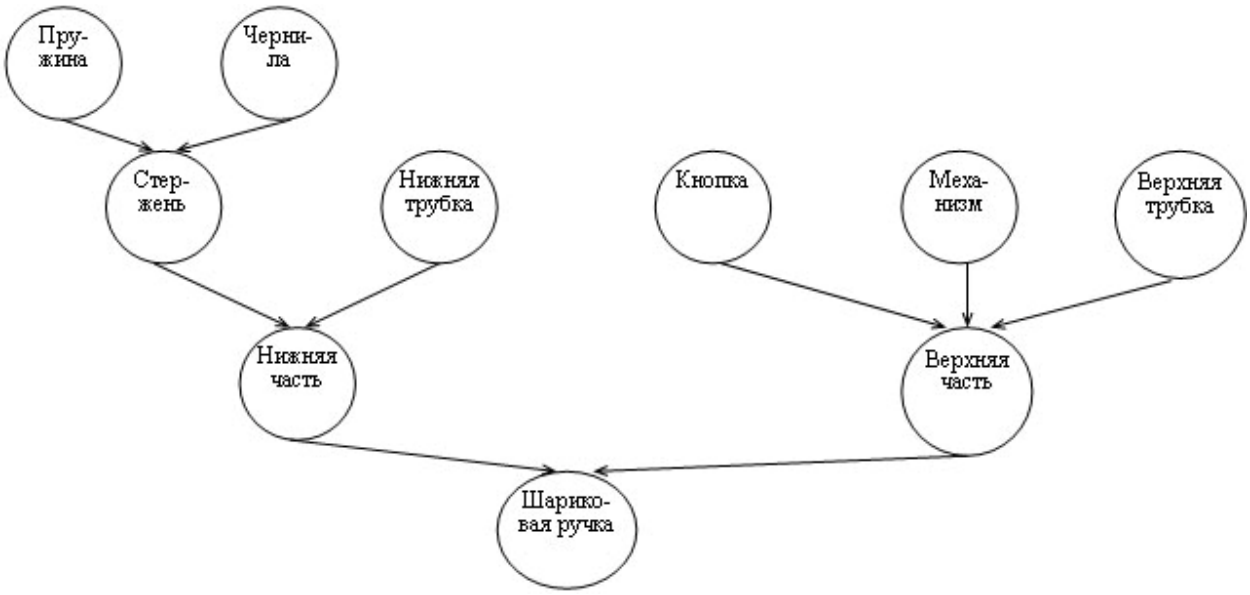


Рис. 3.41. - Пример композиционной схемы.

Рис. 3.42. - Пример диаграммы состояний.

Онтология ресторанного бизнеса может быть представлена с помощью средств, предлагаемых стандартом IDEF5, например, следующим образом. На рисунке 3.43 представлена композиционная схема ресторана. На рисунке 3.44 представлена диаграмма классификации (NKC) видов информации, необходимой для функционирования ресторана.

Схема взаимосвязей может быть составлена как для классов представленных в композиционной схеме, так и для классов представленных на диаграмме классификации. Схема взаимосвязей классов информации (см. рис. 3.44) будет подобна диаграмме IDEF1, однако, будет учитывать большее число ее видов.

Диаграмма состояний объекта с точностью до обозначений будет подобна диаграмме OSTN стандарта IDEF3, если в качестве объекта будет выбран посетитель ресторана.

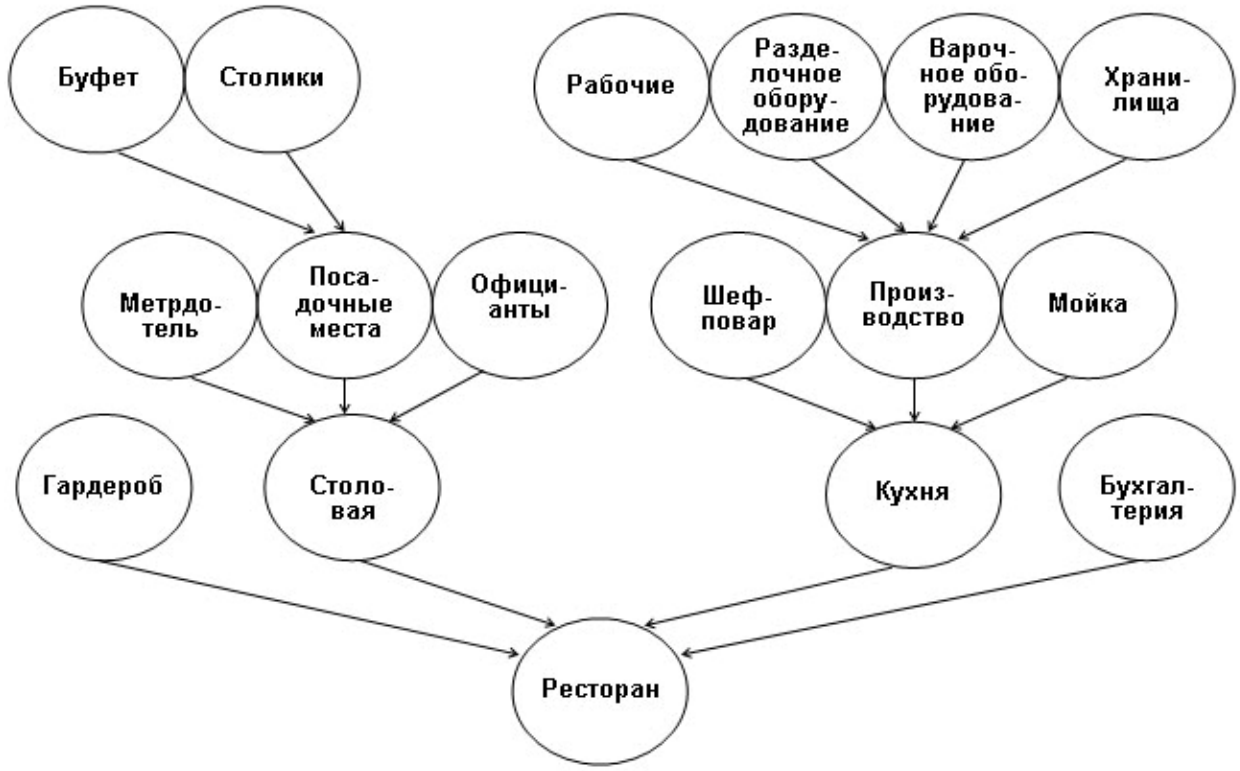


Рис. 3.43. - Пример композиционной схемы ресторана.

В заключении рассмотрения серии стандартов Icam Definition следует отметить, что упомянутые стандарты представляют собой хорошо структурированную и продуманную аналитическую технологию. Данная серия стандартов аналогична технологии 3VM, однако, является, с одной стороны, более строгой и формализованной, а, с другой стороны, обладающей более широкими возможностями. При этом функционально технология DFD в более формализованном виде и с дополнительными возможностями представлена в стандарте IDEF0 (SADT), технология ERD – в стандарте IDEF1X, технология STD – в стандарте IDEF3 (OSTN-диаграммы). Стандарт же IDEF5 обеспечивает все эти и другие возможности в менее формализованном виде, расширяя, таким образом, сферу применения данной серии стандартов.

Опыт применения технологии Icam Definition показывает, что комплексное использование рассмотренных стандартов позволяет проанализировать систему (предметную область) достаточно целостно. Стандарты данной технологии являются конструктивно дополняющими друг драга частями. Это позволяет при их совместном использовании в рамках итерационного процесса обнаруживать и устранять неточности, ошибки или пропуски информации. Например, при разработке функциональной SADT-модели ресторана, в первоначальном варианте этой модели были пропущены информационные потоки «меню» и «цены», необходимые для приема заказа от посетителя. Информационный анализ процесса приема заказа с помощью IDEF1-диаграммы позволил выявить и устранить этот пропуск, так как потребовал в явном виде определить все необходимые данные.

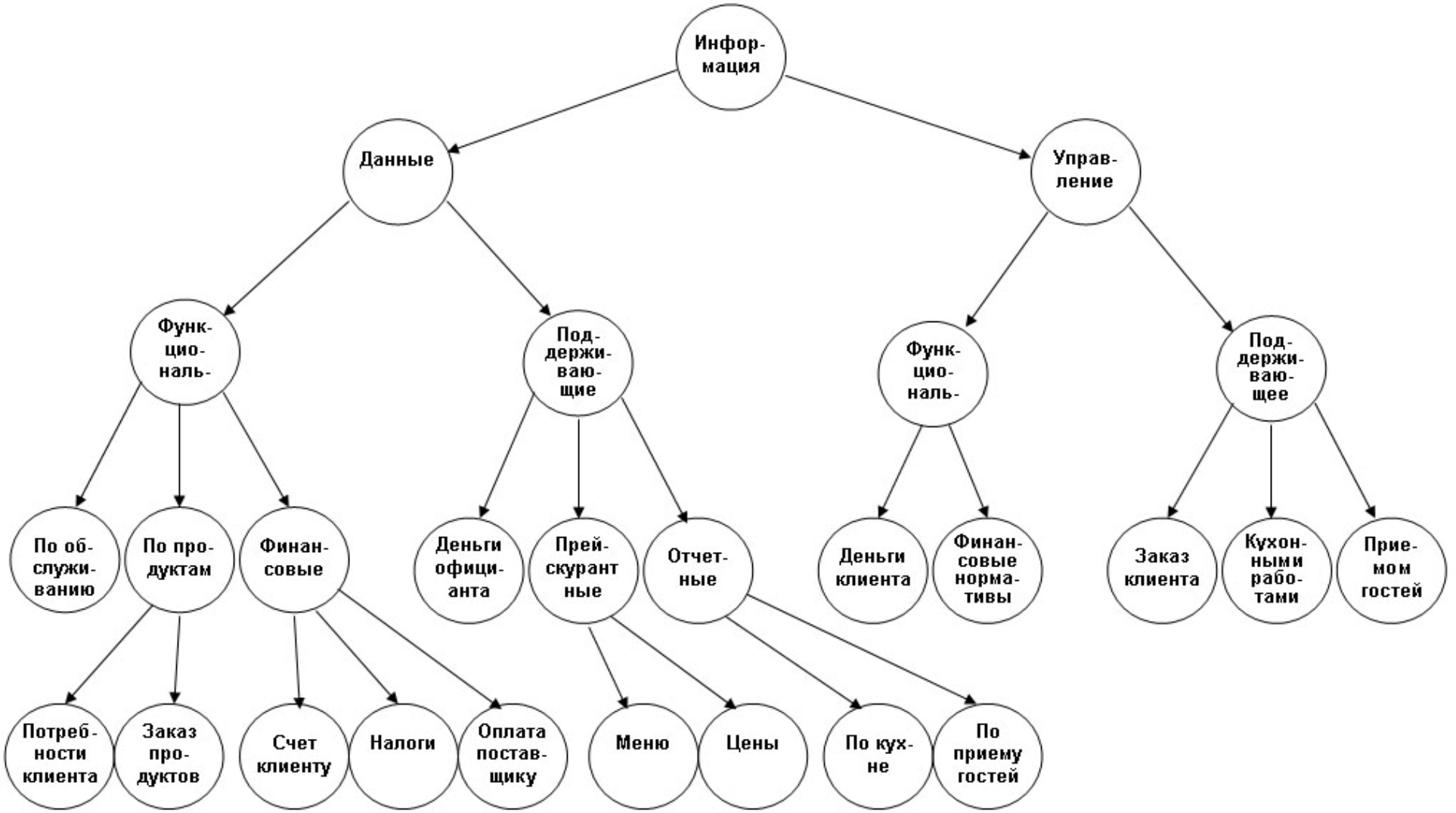


Рис. 3.44. - Пример диаграммы классификации (NKC) видов информации, необходимой для работы ресторана.

Несмотря на перечисленные достоинства технологии Icam Definition, она обладает всеми недостатками, приведенными при завершении рассмотрения технологии 3VM.

Выводы

1. Для решения сложных проблем используются методы анализа, предоставляющие в распоряжение аналитика визуальные графоаналитические средства для построения моделей бизнес-систем и бизнес-процессов.
2. Технология системно-структурного анализа, представленная несколькими стандартами серии Icam Definition, аналогична технологии 3VM. При этом DFD-даграммы в более формализованном виде и с дополнительными возможностями соответствуют стандарту функционального моделирования IDEF0 (SADT), ERD-диаграммы – стандарту информационного моделирования IDEF1, STD-диаграммы – стандарту моделирования сценариев IDEF3 (OSTN-диаграммы).
3. Стандарт же онтологического моделирования IDEF5 обеспечивает все эти и другие возможности в менее формализованном виде. Комплексное использование рассмотренных стандартов позволяет проанализировать систему (предметную область) достаточно целостно. Недостатки данной технологии аналогичны технологии 3VM.

3.3. CASE-инструментарий системного моделирования и анализа

3.3.1. Назначение и возможности «AllFusion Process Modeler/BPwin»

Реорганизация или реинжиниринг бизнес-процессов уже довольно давно является отдельной дисциплиной; в этой области рынка работает множество компаний, услуги которых весьма ценятся. С точки зрения профессионалов, ключевым моментом успеха проекта по реорганизации является коммуникация между всеми группами лиц, заинтересованных в выполнении задачи. Это взаимодействие достигается посредством составления различных моделей, которые отображают бизнес-процессы и понятны всем участникам проекта. Одновременно модель служит для формализации и документирования существующего состояния дел и изучения возможностей улучшения работы [100].

На рынке существует несколько технологий, которые предназначены для моделирования бизнес-процессов и позволяют облегчить обмен информацией. Инструменты для разработки, моделирования и анализа получили название CASE-средств. Понятие CASE-средства охватывает самые различные инструменты, которые служат для компьютерного анализа и моделирования, и инструменты для анализа бизнес-процессов представляют собой лишь небольшую часть всего семейства. Однако именно изучение бизнес-процессов является ключевым моментом при разработке любого приложения и позволяет четко и однозначно определить задачи, которые стоят перед разработчиками. Таким образом, инструменты анализа бизнес-процессов, кроме решения основной за-

дачи, являются также неотъемлемой частью начального этапа разработки жизнеспособной информационной системы [100].

В качестве примера инструмента моделирования и анализа бизнеса рассмотрим (используя работы [100 – 103], а также сайт компании «Interface Ltd.») широко распространенный пакет ***BPwin*** или, как теперь он называется, ***AllFusion Process Modeler*** (производитель компания «Computer Associates»).

BPwin – мощный инструмент моделирования, который используется для анализа, документирования и реорганизации сложных бизнес-процессов. Модель, созданная средствами BPwin, позволяет четко документировать различные аспекты деятельности: действия, которые необходимо предпринять, способы их осуществления, требующиеся для этого ресурсы и др. Таким образом, формируется целостная картина деятельности предприятия – от моделей организации работы в маленьких отделах до сложных иерархических структур. При разработке или закупке программного обеспечения модели бизнес-процессов служат прекрасным средством документирования потребностей, помогая обеспечить высокую эффективность инвестиций в сферу информационных технологий. В руках же системных аналитиков и разработчиков BPwin – еще и мощное средство моделирования процессов при создании корпоративных информационных систем.

Модели BPwin дают основу для осмысления бизнес-процессов и оценки влияния тех или иных событий, а также описывают взаимодействие процессов и потоков информации в организации. Неэффективная, высоко затратная или избыточная деятельность может быть легко выявлена и, следовательно, усовершенствована, изменена или устранена в соответствии с общими целями организации.

Внешние обстоятельства зачастую вынуждают вносить изменения в деятельность организации. Последствия этих изменений должны быть тщательно изучены и осмыслены перед тем, как система будет переделана с их учетом. BPwin может помочь пользователю на протяжении всего цикла, предоставив возможность оптимизировать бизнес-процесс, которого коснутся эти изменения.

С помощью BPwin пользователь может сделать свою работу более продуктивной. Действия и другие объекты создаются буквально несколькими щелчками мыши, а затем легко отбуксированы в нужное место. Интерфейс BPwin, выполненный в стиле «проводника» облегчает навигацию и редактирование сложных процессов с иерархической структурой. Развитые возможности изменения масштаба представления позволяют быстро найти и сосредоточиться на необходимой для работы части модели процесса.

BPwin обладает интуитивно-понятным графическим интерфейсом, быстро и легко осваивается, что позволяет сосредоточиться на анализе самой предметной области, не отвлекаясь на изучение инструментальных средств. BPwin помогает быстро создавать и анализировать модели с целью оптимизации деловых и производственных процессов. Применение графического языка бизнес-моделирования обеспечивает логическую целостность и полноту описания, необходимую для достижения точных и непротиворечивых результатов.

BPwin имеет широкие возможности по представлению диаграмм. Графическое представление модели может быть изображено при помощи различных цветов, шрифтов и прочих параметров представления, которые выделяют важные или, наоборот, тушируют незначительные аспекты модели. Эта незначительная на первый взгляд возможность является ключевой во время представления и обсуждения модели с заказчиком или экспертами предметной области, т.к. правильно подобранное графическое представление позволяет им быстрее сориентироваться в модели [100].

BPwin автоматизирует решение многих вспомогательных задач, которые обычно связаны с построением модели процесса, и обеспечивает логическую строгость, необходимую для достижения корректных и согласованных результатов. BPwin отслеживает связи в диаграммах, сохраняя их целостность при внесении изменений в модель. Динамическая «подсветка» объектов служит подсказкой при построении модели и предостерегает от повторения распространенных ошибок в моделировании. Кроме этого, BPwin поддерживает заданные пользователем свойства, что позволяет вносить соответствующую вашим потребностям информацию.

BPwin позволяет:

- Обеспечить эффективность операций, рассматривая текущие бизнес-операции через мощные инструменты моделирования.
- Совершенствовать бизнес-процессы, формулируя и определяя альтернативные реакции на воздействия рынка.
- Быстро исключать непродуктивные операции, легко и интуитивно сопоставляя операционные изменения. Неэффективные, неэкономичные или избыточные операции могут быть легко выявлены и, следовательно, улучшены, изменены или вовсе исключены – в соответствии с целями компании.

Различные варианты оформления с гибким использованием шрифтов, цвета и других средств форматирования придают документам большую наглядность. Пользователь может просматривать и распечатывать общее представление своей модели в виде древовидных диаграмм. С помощью средства создания FEO диаграмм (For Exposition Only) вариации модели или проблемной области можно проанализировать, не внося изменений в основную модель. Возможности настройки пользовательских палитр цветов позволяют легко адаптировать вид документов в соответствии с особенностями принтера или демонстрационного проектора без внесения изменений в саму модель.

Функциональность BPwin заключается не только в рисовании диаграмм, но и в проверке целостности и согласованности модели. BPwin обеспечивает логическую четкость в определении и описании элементов диаграмм, а также проверку целостности связей между диаграммами. Инструмент обеспечивает коррекцию наиболее часто встречающихся ошибок при моделировании, таких, как «зависание» связей при переходе от диаграммы к диаграмме, нарушение ассоциации связей в различных диаграммах модели и т.п. Кроме того, интерактивное выделение объектов обеспечивает постоянную визуальную обратную связь при построении модели [100].

BPwin позволяет адаптироваться к постоянно меняющимся реалиям современного рынка. Конкуренция предполагает мгновенную реакцию на новые возможности, угрозы и потребности покупателей. Сегодня постоянные изменения стали нормой. Поскольку бизнес-процессы становятся все более сложными, требуются решения, представляющие интегрированный взгляд на функционирование компании. Таким решением и является новый BPwin (или AllFusion Process Modeler).

Модели бизнес-процессов в масштабах всего предприятия могут оказаться очень сложными. BPwin предоставляет возможности, призванные облегчить инкрементальную разработку моделей и разграничение процессов. Средства объединения дают возможность нескольким проектным группам проводить анализ различных фрагментов деятельности, а затем создать глобальное представление. Иногда бывает необходимо более детально изучить определенную часть общей модели. BPwin позволяет разбить модель на фрагменты, поработать с ними, а затем вновь объединить их в одно целое.

Анализ сложных бизнес-процессов зачастую может потребовать привлечения целого коллектива специалистов. Создание комфортной среды для их работы может быть осуществлено путем масштабирования BPwin для поддержки корпоративных бизнес-моделей за счет интеграции с системой управления моделированием ***ModelMart***. ModelMart поддерживает мощный набор инструментальных программных средств, обеспечивающих совместное (групповое) проектирование и разработку программных систем, включая механизмы объединения моделей и анализа изменений, контроль версий, возможность создания «компонент» модели и т.д. Для организации хранилища моделей в ModelMart используются СУБД на различных платформах [100].

BPwin совмещает в одном инструменте средства моделирования функций (IDEF0), потоков данных (DFD) и потоков работ (IDEF3), координируя эти три основных аспекта бизнеса для соответствия потребностям бизнес-аналитиков и системных аналитиков. BPwin позволяет повторно использовать ключевую информацию моделирования с точки зрения базовых аспектов, чтобы определить точки конфликтов и, в конечном счете, достичь их согласования.

С помощью функционального моделирования (нотация IDEF0), можно провести систематический анализ бизнеса, сосредоточившись на регулярно решаемых задачах (функциях), свидетельствующих об их правильном выполнении показателях, необходимых для этого ресурсах, результатах и исходных материалах (сырье).

Моделирование потоков данных, часто используемое при разработке программного обеспечения, сосредоточено вокруг потоков данных, передающихся между различными операциями, включая их хранение, для достижения максимальной доступности и минимального времени ответа. Такое моделирование позволяет рассмотреть конкретный процесс, проанализировать операции, из которых он состоит, а также точки принятия решений, влияющих на его ход.

Моделирование потоков работ или логики выполнения процессов позволяет рассмотреть конкретный процесс, проанализировать операции, из которых он состоит, а также точки принятия решений, влияющих на его ход.

BPwin предоставляет средства для изучения операций и управления операциями на различных уровнях детализации. Например, иногда бывает важно сосредоточиться на определенной части бизнеса организации. BPwin позволяет вам разделить сложный процесс на множество управляемых частей, обеспечивая группам разработчиков модели возможность сосредоточиться на интересующих их аспектах. В итоге, эти различные аспекты могут быть согласованы и объединены, чтобы составить единый, целостный взгляд на ваше предприятие. BPwin позволяет вам объединить раздельные модели в единую согласованную модель и достигнуть согласования проекта. BPwin помогает вам понять общее влияние изменений на существующие бизнес-процессы, обеспечивая быструю и эффективную адаптацию.

Встроенный механизм вычисления стоимости позволяет оценивать и анализировать затраты на осуществление различных видов деловой активности Механизм вычисления расходов на основе выполняемых действий (***Activity-Based Costing***, ABC) – это технология, применяемая для оценки затрат и используемых ресурсов. Она помогает распознать и выделить наиболее дорогостоящие операции для дальнейшего анализа.

Одним из важнейших средств BPwin является генератор отчетов. На деле, генератор отчетов ***RPTwin*** представляет собой автономный продукт и позволяет генерировать подробные и многогранные отчеты по модели. Вместе с BPwin устанавливается набор стандартных отчетов, которые позволяют осветить модель с различных сторон. Отчеты обычно сопровождают окончательный вариант модели бизнес-процессов, созданной при помощи BPwin, и содержат информацию, размещение которой на модели сделало бы ее трудной для восприятия. Например, отчет может содержать подробное описание каждого элемента диаграммы, что помогает представить себе назначение данного элемента без дополнительных разъяснений со стороны системного аналитика, создававшего диаграмму. Кроме того, существуют отчеты, которые предназначены для самого системного аналитика, например, отчет по целостности модели [100 - 103].

BPwin может генерировать отчеты непосредственно в формате MS Excel и Word для последующей обработки и использования в других приложениях. Связь с ***ERwin*** (моделирование данных в стандарте IDEF1X) позволяет сократить время проектирования и разработки сложных информационных систем. Для системных аналитиков тесная интеграция BPwin с инструментом проектирования баз данных открывает уникальные возможности по созданию действительно комплексных систем, в которых ERwin служит для описания информационных объектов системы, в то время как BPwin отражает функциональные особенности предметной области. Связывая сущности и атрибуты модели данных с информацией о выполняемых действиях, можно продолжить анализ процессов на новом уровне с одновременной перекрестной проверкой моделей процессов и данных.

Таким образом, пакет BPwin может повысить конкурентоспособность, оптимизировать процессы управления. Результатом использования BPwin является исключение излишних и бесполезных действий, снижение затрат, повышение гибкости и эффективности всего бизнеса.

3.3.2. Особенности «BPwin»

- Интуитивно-понятный графический интерфейс, который быстро и легко осваивается, что позволяет сосредоточиться на анализе самой предметной области, не отвлекаясь на изучение инструментальных средств. Интерактивное выделение объектов обеспечивает постоянную визуальную обратную связь при построении модели. BPwin поддерживает ссылочную целостность, не допуская определения некорректных связей и гарантируя непротиворечивость отношений между объектами.
- BPwin автоматизирует многие задачи, обычно связанные с построением моделей процессов, обеспечивая семантическую точность, необходимую для гарантии правильных и согласованных результатов. Подсветка объектов упрощает построение модели, исключая часто встречающиеся ошибки моделирования.
- BPwin может быть настроен для сбора информации, существенной для кокретного бизнеса. Эта информация становится сразу же доступной через генератор отчетов BPwin и может быть экспортирована в другие программы, например, Microsoft Word и Excel.
- BPwin поддерживает диаграммы ***Swim Lane***, предоставляя эффективный механизм для визуализации и оптимизации сложных бизнес-процессов. Диаграммы Swim Lane координируют сложные процессы и функциональные ограничения и позволяют вам видеть процессы, роли и обязанности во всем их многообразии.
- Новая структура словаря модели делает ввод и управление информацией быстрым и простым. Этот настраиваемый интерфейс электронных таблиц прост в применении и предоставляет отличный механизм для распространения моделей, независимо от того, вводите вы данные вручную или импортируете их.
- Контекстные диаграммы для описания границ системы, области действия, назначения объектов. Иерархическая структура диаграмм, облегчающая последовательное уточнение элементов модели. Декомпозиционные диаграммы для описания особенностей взаимодействия различных процессов. BPwin также поддерживает автоматическую настройку размеров диаграмм и возможность изменения масштабов изображения моделей.
- Организационные структуры оказывают огромное влияние на определение и выполнение бизнес-процессов. BPwin поддерживает явное определение ролей, а это определяет и категоризирует задачи или работы, составляющие бизнес-процессы. Основываясь на ролях, определенных пользователем, BPwin формирует организационные диаграммы.
- BPwin обеспечивает совместное и повторное использование технологий моделирования бизнес-процессов (IDEF0), потоков работ (IDEF3) и потоков данных (DFD).
- BPwin полностью поддерживает методы расчета себестоимости по объему хозяйственной деятельности (ABC) и оптимизирована для анализа

процессов. Развитые средства подготовки отчетов и двунаправленный интерфейс со специализированным инструментарием ABC облегчают реализацию корпоративной стратегии на основе управления хозяйственной деятельностью.

- Собственный генератор отчетов. ***Report Template Builder*** (RTB) – это новый генератор отчетов, общий для ERwin и BPwin, создающий разнообразные отчеты и Web-страницы. Вы можете определять шаблоны отчетов, применяя их затем к любым своим моделям. Подход «определить однажды – применять повторно и повсюду» позволяет организации быстро создавать и продвигать стандарты отчетности. RTB поддерживает множество форматов, включая RTF, HTML, XLS (Excel) и текст.
- Интерфейс к средствам имитационного моделирования. Для моделирования сложных условий деятельности BPwin предлагает интерфейс к имитационному программному обеспечению (например, ***Arena***). Это позволяет использовать готовые модели для изучения изменяющегося во времени (динамического) взаимодействия бизнес-процессов. Распределение ресурсов и потоки могут быть оптимизированы для достижения эффективной загрузки. Имитационное моделирование позволяет в динамике проанализировать воздействие изменений. Прежде чем эти изменения будут произведены, можно проверить различные сценарии и обеспечить тем самым принятие оптимального решения.
- Поддерживаемые операционные системы Windows 95 – XP.

Пакет BPwin интегрирован с другими программными продуктами, дополняющими его функциональные возможности.

ERwin – средство моделирования баз данных №1 в мире. Уникальная особенность BPwin – возможность удостовериться в том, что информационная модель оптимально согласуется с потребностями бизнес-процесса. BPwin обеспечивает двунаправленную синхронизацию с ERwin. Использование BPwin позволяет проверить качество и согласованность моделей данных ERwin, получить важную информацию о том, как и где используются данные, и обеспечить ее доступность в нужный момент и в нужном месте. Такая интеграция гарантирует, что новые распределенные системы и хранилища данных в действительности будут соответствовать потребностям вашего бизнеса.

Для системных аналитиков тесная интеграция BPwin с ERwin открывает возможности по созданию действительно комплексных систем, в которых ERwin служит для описания информационных объектов системы, в то время как BPwin отражает функциональные особенности предметной области. Связывая сущности и атрибуты модели данных с информацией о выполняемых действиях, Вы можете продолжить анализ процессов на новом уровне с одновременной перекрестной проверкой моделей процессов и данных.

ModelMart – среда для работы группы проектировщиков над одним проектом. Возможности обмена информацией о моделях между BPwin и ERwin еще более расширяются при использовании продукта ModelMart, при этом данные об объектах и атрибутах могут синхронизироваться динамически в процессе моделирования. Информация, переданная в процессе такой синхронизации,

может быть обработана развитыми средствами ModelMart для управления изменениями, анализа последствий и формирования отчетов.

AllFusion – линейка продуктов для поддержки всех стадий разработки программного обеспечения. BPwin входит в семейство продуктов AllFusion для поддержки всех стадий жизненного цикла разработки программного обеспечения (аналог Rational Suite). Туда, в частности, входит линейка CASE-средств AllFusion Modeling Suite (ERwin, BPwin, ModelMart, Paradigm Plus, ERwin, Examiner) и средства управления проектами. Совместное применение этих продуктов обеспечивает прочный фундамент для построения, развертывания и управления приложениями. При этом не накладываются ограничения на выбор базовых технологий, методов и платформ разработки. AllFusion предлагает моделирование и управление процессами, проектами, изменениями, конфигурациями.

Paradigm Plus – средство проектирования компонентов программного обеспечения и кодогенерации. BPwin интегрирован с Paradigm Plus – средством моделирования компонентов программного обеспечения и генерации объектного кода на основе созданных моделей. Двусторонняя связь между BPwin и Paradigm Plus позволяет импортировать смоделированные бизнес-процессы в Paradigm Plus как «юз-кейсы» (use cases) и экспортировать «юз-кейсы» в BPwin.

Model Navigator – продукт для просмотра моделей ERwin и BPwin с возможностью генерации отчетов. Зачастую необязательно приобретать много лицензий BPwin, когда кому-то из пользователей требуется лишь просмотр моделей, а не их редактирование. Model Navigator, своеобразный «viewer», обеспечивает динамический просмотр моделей, а также содержит полный спектр средств для формирования отчетов.

Arena – система имитационного моделирования. BPwin имеет интерфейс к средствам имитационного моделирования, ведущим из которых является Arena. Это позволяет использовать готовые модели для изучения изменяющегося во времени (динамического) взаимодействия бизнес-процессов. Распределение ресурсов и потоки могут быть оптимизированы для достижения эффективной загрузки. Имитационное моделирование – создание компьютерной модели системы (физической, технологической, финансовой и т. п.) и проведение на ней экспериментов с целью наблюдения/предсказания. (Подробнее об имитационном моделировании см. на сайте www/interface.ru).

3.3.3. Недостатки инструментария системного моделирования

Анализируя пакет BPwin (и, таким образом, SADT-технологию (стандарт IDEF0), стандарт IDEF3, а также DFD-технологию), следует сказать, что первые стандарты разрабатывались как средства моделирования и анализа любых систем, последняя технология – именно информационных систем. Однако, в настоящее время оказалось, что выразительных средств SADT и IDEF3 недостаточно для моделирования систем информационных, и, кроме того, SADT, IDEF3 и DFD не поддерживают объектно-ориентированного проектирования. В результате данные технологии практически используется относительно редко

(менее чем в 10% существующих CASE-средств) [20].

Будучи основанным на системных структурных методах пакет BPwin предусматривает построение двух или трех моделей одного и того же объекта: функциональной (активностной), информационной (данных), а также динамической. Это обстоятельство приводит к необходимости проведения специального сквозного контроля диаграмм одного или разных типов, т.е. соответственно вертикального и горизонтального ***балансирования диаграмм***, для выявления весьма вероятных ошибок [20]. При этом для создания динамических моделей требуется использование дополнительных специальных расширений или других средств, с которыми SADT плохо согласуется [20].

Жесткие искусственные ограничения на число блоков в диаграмме, а также принципиально ограниченное количество типов связей и типов отношений (взаимодействий) между блоками не позволяют гарантировать во всех случаях адекватность модели объекту и затрудняют понимание диаграмм.

Кроме того, изображение функциональных связей каждого элемента на SADT-диаграммах в виде входа, управления, механизма (или ресурса) и выхода не обеспечивается никаким методом распределения связей в конкретных случаях по данным категориям. Результатом этого является представление, например, производственного подразделения как элемента, ресурсом которого изображаются люди, которые в нем работают, т.е. которые составляют, на самом деле, его части, а не входы [98].

«Диаграммы потоков данных обеспечивают удобное описание функционирования компонент системы, но не снабжают аналитика средствами описания деталей этих компонент, а именно, какая информация преобразуется процессами и как она преобразуется» [20, с. 53]. Их практическое применение высокоэффективно, как правило, только для отражения информационных структур бизнес-процесса, оптимизация которого проведена уже другими средствами. DFD ориентированы на системных аналитиков и программистов и не учитывают особенности восприятия менеджерами предметной области.

Кроме того, в статье [25] отмечается, что наименьший вред организации принесет инструментарий моделирования, «лишающий разработчика той части «творческих» возможностей, которые ведут к разнообразию представления организационных моделей». Данное требование непосредственным образом связано с тем, что инструментарий моделирования должен быть средством поддержки принятия решений, а не художественного творчества. При этом степень соответствия этому требованию инструментария, использующего нотацию SADT (IDEF0), оценивается как крайне низкая.

Рассмотренные технологии и инструментарий приспособлены для хорошо специфицированных и стандартизованных «западных» бизнес-процессов. При моделировании больших, сложных, иерархических систем создаваемые диаграммы становятся слишком сложными для понимания и корректировки.

Ну и основная, очевидно, проблема состоит в том, что существует процедурно-ориентированный вариант технологий (3VM, Icam Definition) и вариант, ориентированный на данные, однако, не существует объектно-ориентированного варианта.

Вопросы для повторения

1. Что такое методология системного анализа 3VM?
2. Опишите процесс построения модели информационной системы с помощью DFD-диаграмм.
3. Что такое ERD-диаграммы? Для чего они используются?
4. Что такое STD-диаграммы?
5. Опишите процесс построения модели производственной системы с помощью IDEF0-диаграмм.
6. Для чего применяются стандарты моделирования IDEF1 и IDEF1X?
7. В чем специфика моделирования процессов в соответствии со стандартом IDEF3?
8. Что такое стандарт моделирования онтологий IDEF5?
9. Опишите назначение и возможности CASE-инструментария системно-структурного моделирования и анализа (AllFusion Process Modeler).
10. Что такое ФСА? Какие процедуры осуществляются в связи с этим с помощью AllFusion Process Modeler?

Резюме по теме

В данном разделе рассмотрены технологии системно-структурного моделирования и анализа сложных систем: технология моделирования «3-View Modeling»; стандарты системно-структурного анализа серии «Icam Definition»; а также CASE-инструментарий системно-структурного моделирования и анализа (AllFusion Process Modeler).

Тема 4. Технология объектного моделирования и анализа

Цели и задачи изучения темы

Целью изучения данной темы является теоретическое и практическое освоение технологий объектного моделирования и анализа сложных систем.

При этом ставятся следующие задачи:

- изучение сущностей, отношений и диаграмм универсального языка моделирования, а также процедуры объектного моделирования;
- ознакомление с требованиями к объектному моделированию организационных систем и их информационного обеспечения;
- изучение CASE-инструментария объектного моделирования и анализа (IBM Rational Software Architect).

4.1. UML – язык объектного моделирования

Объектно-ориентированное моделирование, как и методы системного анализа, предполагает использование некоторой нормативной системы, т.е. языка, состоящего из набора символов, имеющих определенное значение (семантику), и правил манипулирования ими (синтаксиса).

В настоящее время благодаря усилиям концерна ***Object Management Group*** (***OMG***) создан единый стандарт языка объектного моделирования – Unified Modelling Language (UML).

Язык UML – это, в первую очередь, стандартное средство для составления «чертежей» программного обеспечения (ПО). Однако, сфера его применения не ограничивается моделированием программ. Он предназначен для визуализации, специфицирования, конструирования и документирования различных аспектов анализируемых и проектируемых систем произвольной природы. При этом, в дальнейшем, обеспечивается возможность компьютерного моделирования этих систем с помощью объектно-ориентированных языков программирования.

Нормативная система языка UML включает два вида символов: ***сущности*** и ***отношения***, а также правила создания комбинаций из этих символов, т.е. ***диаграммы***. Кроме того, существуют ***механизмы расширения*** языка для уточнения семантики символов сущностей и отношений при моделировании конкретной предметной области.

Рассмотрим основные элементы нормативной системы языка UML, используя работы [13, 79, 81 - 86, 104 - 106].

4.1.1. Сущности: структурные; поведенческие; группирующие; аннотационные

Сущности – это символы, являющиеся основными элементами объектной модели. Принято использовать четыре вида сущностей при построении моделей: ***структурные***, ***поведенческие***, ***группирующие***, ***аннотационные***.

Структурные сущности

Структурные сущности (Structural things) представляют собой символы статических частей объектной модели, соответствующие концептуальным или физическим элементам системы.

Класс (***Class***) – это описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой (рис. 4.1).

При этом атрибут – это именованное свойство всех экземпляров данного класса. Он имеет тип, обычно указываемый в классе, и значение, которое указывается в экземпляре класса. Операция – это описание поведения всех экземпляров данного класса, или функция, которую могут выполнять его экземпляры, или услуга, которую могут запросить у этих экземпляров.

Экземпляр (***Instance***) – конкретная реализация абстракции (класса, прецедента, компонента и т.д.) или объект, графически представляемый как класс с подчеркнутым именем (рис. 4.1), после которого может следовать двоеточие (:) и имя класса, от которого производится данный объект. Методы объекта представляют собой реализации соответствующих операций класса и, как правило, не указываются.

Активный класс (***Active class***) – класс, объекты которого могут инициировать управляющее воздействие, и вовлечены в один или несколько процессов или потоков. Изображается также как класс, но толстыми линиями.

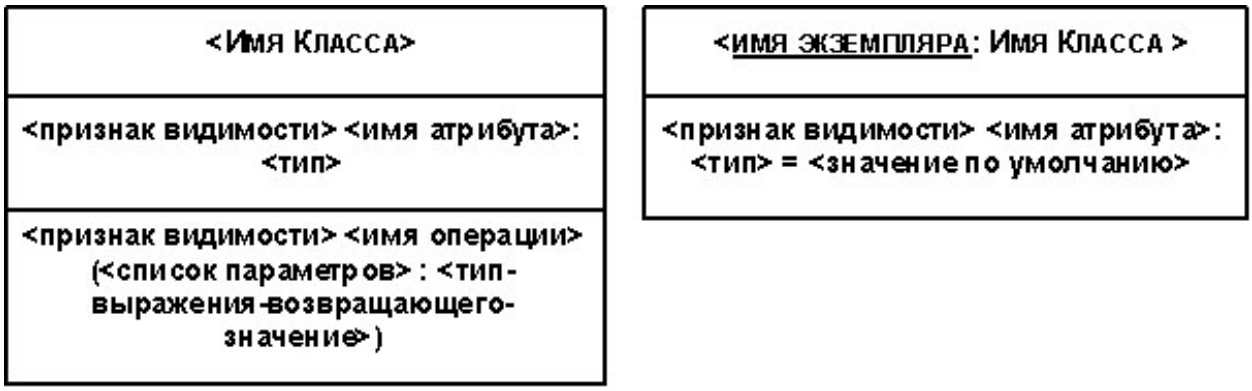


Рис. 4.1. - Графическое изображение класса и экземпляра.

Признак видимости может принимать одно из трех значений:

- + – открытый (public) – может использовать любой класс или объект;
- # – защищенный (protected) – может использовать любой потомок класса;
- - – закрытый (private) – может использовать только данный класс.

Интерфейс (***Interface***) – совокупность операций, которые определяют набор услуг, предоставляемых классом или компонентом, т.е. описание видимого извне поведения элемента (рис. 4.2). Определяет спецификации операций (сигнатуры), но не их реализацию и присоединяется к реализующему его классу или компоненту.

Кооперация (***Collaboration***) – представляет взаимодействия элементов, которые, работая совместно, производят некоторый кооперативный эффект, не сводящийся к простой сумме слагаемых (рис. 4.2).

Прецедент / Вариант использования (***Use case***) – описание последовательности выполняемых системой действий, в результате которых образуется наблюдаемый результат, значимый для какого-нибудь определенного актора (рис. 4.2).

Рис. 4.2. - Графическое изображение интерфейса, кооперации и прецедента.

Компонент (***Component***) – физически заменяемая часть системы, соответствующая некоторому набору интерфейсов и обеспечивающая их реализацию (рис. 4.3). Как правило, это физическая упаковка логических элементов (файлы), таких как классы, интерфейсы и кооперации.

Узел (***Node***) – элемент реальной (физической) системы, существующей во время функционирования программного комплекса и представляющий собой вычислительные ресурсы (объем памяти, скорость обработки) (рис. 4.3).

Рис. 4.3. - Графическое изображение компонента и узла.

Поведенческие сущности

Поведенческие сущности (***Behavioral things***) представляют собой символы динамических составляющих объектной модели.

Взаимодействие (***Interaction***) – обмен сообщениями между объектами для достижения определенной цели (рис. 4.4).

Состояние (***Statechart***) – алгоритм поведения, определяющий последовательность состояний, через которые объект или взаимодействие проходят на протяжении своего жизненного цикла в ответ на различные события (рис. 4.4).

Активность (***Activity***) – алгоритм поведения, определяющий последовательность процессов, осуществляемых объектом или происходящих с объектом на протяжении его жизненного цикла (рис. 4.4).

Рис. 4.4. - Графическое изображение сообщения, состояния и активности.

Группирующие сущности

Группирующие сущности (***Grouping things***) представляют собой симво-

лы, организующие различные компоненты объектной модели. Это блоки, на которые можно разложить модель.

Пакет (***Package***) – механизм организации элементов (структурных, поведенческих и группирующих сущностей) модели в группы (рис. 4.5). В отличие от компонентов носят чисто концептуальный характер.

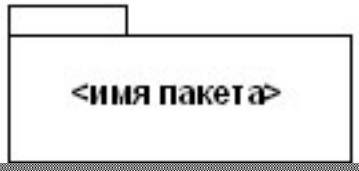


Рис. 4.5. - Графическое изображение пакета.

Аннотационные сущности

Аннотационные сущности (***Summary things***) представляют собой пояснительные символы объектной модели.

Примечание (***Note***) – символ для изображения комментариев или ограничений, присоединенных к элементу или группе элементов (рис. 4.6).

Рис. 4.6. - Графическое изображение примечания.

4.1.2. Отношения

Отношения – это символы, связывающие различные сущности. В языке определено четыре вида отношений: ***обобщение***, ***ассоциация***, ***зависимость*** и ***реализация***.

Обобщение (***Generalization***) – отношение (потомок-родитель), при котором специализированный элемент (потомок) может быть подставлен вместо обобщенного элемента (родителя) (рис. 4.7). Потомок наследует структуру и поведение своего родителя.

Ассоциация (***Association***) – отношение связи (соединения) между объектами (рис. 4.7). На графическом изображении ассоциации могут присутствовать дополнительные обозначения кратности, ролей или меток.

Агрегирование – разновидность ассоциации, т.е. ассоциация между целым и его частями (рис. 4.7).

Композиция - такое отношение агрегирования, при котором часть является неотъемлемой составляющей целого (рис. 4.7).

Рис. 4.7. - Графическое изображение обобщения, ассоциации, агрегирования и композиции.

Зависимость (***Dependency)*** – отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой (см. рис. 4.8).

Реализация (***Realization***) – отношение между элементами, при котором один элемент определяет «контракт», а другой гарантирует его выполнение (см. рис. 4.8). Используется между интерфейсами и реализующими их классами или компонентами, а также между прецедентами и реализующими их кооперациями.

Рис. 4.8. - Графическое изображение зависимости и реализации.

4.1.3. Диаграммы

Диаграммы – это совокупности правил соединения различных символов языка моделирования. Диаграмма представляет собой связанный граф, вершинами которого являются сущности, а ребрами – отношения. Любая диаграмма представляет собой модель некоторого аспекта разрабатываемой системы. Предусмотрены следующие виды диаграмм (правил соединения символов):

Диаграмма вариантов использования/прецедентов (***Use case diagram***) – диаграмма поведения, показывающая функции моделируемой системы и ее связи с другими системами, т.е., в случае разработки программного обеспечения, требования пользователей. Диаграмма состоит из множества прецедентов, классов (акторов: пользователей или любых других внешних систем) и отношений между ними (рис. 4.9).

Проектирование системы осуществляется путем реализации на каждой итерации одного или нескольких вариантов использования. Данная диаграмма является основой для моделирования поведения и тестирования системы.

Рис. 4.9. - Диаграмма вариантов использования.

На данной диаграмме показаны следующие виды отношения между прецедентами: ***расширение*** и ***использование***.

Связь типа «***расширение***» применяется в тех случаях, когда один вариант использования подобен другому, но несет несколько другую нагрузку и, в связи с этим, необходимо описать изменение в обычном поведении системы.

Связь типа «***использование***» применяется в тех случаях, когда имеется какой-либо фрагмент поведения системы, который повторяется более чем в одном варианте использования и нет смысла копировать его описание.

Диаграмма классов (***Class diagram***) – структурная диаграмма, показывающая структуру классов предметной области. Диаграмма состоит из классов, интерфейсов, кооперaций и отношений между ними (рис. 4.10). Абстрактные классы, от которых не производятся экземпляры, помечаются курсивом.

Классы на диаграмме отражают понятия и представления пользователей о соответствующей предметной области, т.е. названная диаграмма является ее концептуальной моделью.

Диаграмма объектов (***Object diagram***) – структурная диаграмма, показывающая объектную модель системы, состоящую из множества экземпляров классов (объектов) и отношений между ними.

Рис. 4.10. - Диаграмма классов.

Диаграмма взаимодействия – диаграмма поведения, показывающая взаимодействие нескольких объектов, например, в рамках одного варианта использования. Взаимодействия изображаются с помощью ***диаграммы последовательностей*** (***Sequence diagram***), подчеркивающей временную последовательность событий (рис. 4.11), или ***диаграммы кооперации*** (***Collaboration diagram***), подчеркивающей структурную организацию объектов, посылающих и принимающих сообщения (рис. 4.12).

Диаграмма состояний (***Statechart diagram)*** – диаграмма поведения, показывает поведение одного объекта в нескольких вариантах использования. Диаграмма представляет автомат, включающий в себя состояния, переходы, события и виды действий. Данная диаграмма акцентирует внимание на поведении объекта, зависящем от последовательности событий, и особенно важна при описании поведения классов, интерфейсов и кооперaций.

Диаграмма деятельности (***Activity diagram***) – диаграмма поведения, показывающая поведение разрабатываемой системы вместе с управляющей структурой. Диаграмма представляет автомат и подчеркивает переходы потока управления от одной деятельности к другой (рис. 4.13). Может отображать несколько объектов в нескольких вариантах использования или реализацию метода. Позволяет определять параллельные процессы.

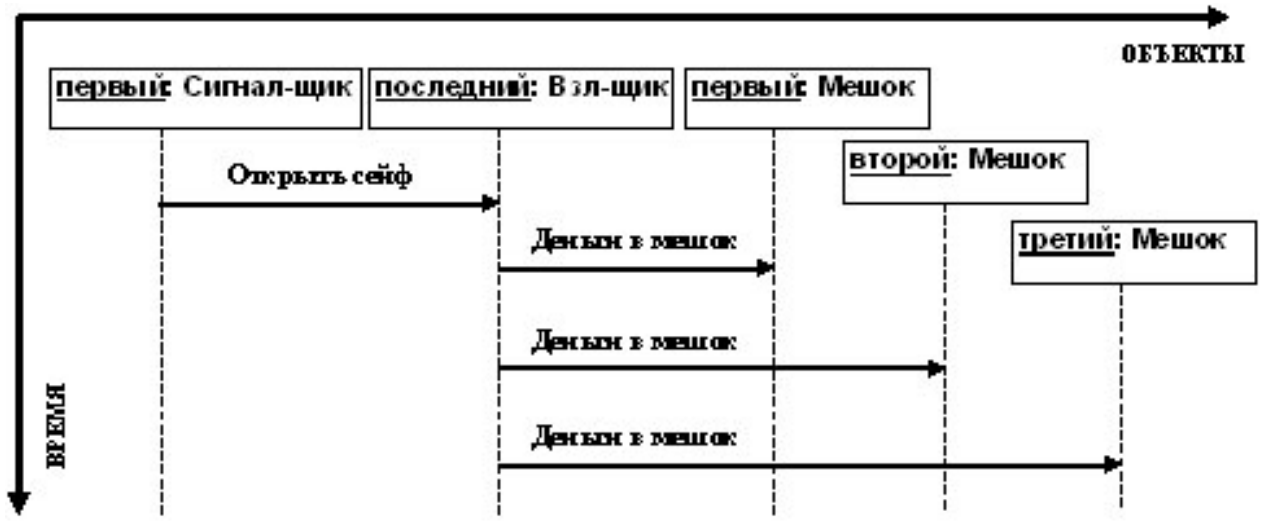


Рис. 4.11. - Диаграмма последовательности.

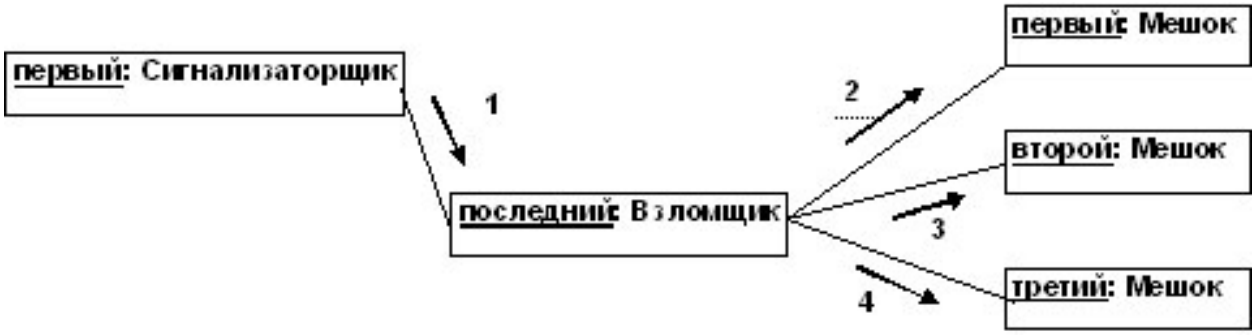


Рис. 4.12. - Диаграмма кооперации.

Сообщения: 1 – открыть сейф; 2 – деньги в первый мешок; 3 – деньги во второй мешок; 4 – деньги в третий мешок.

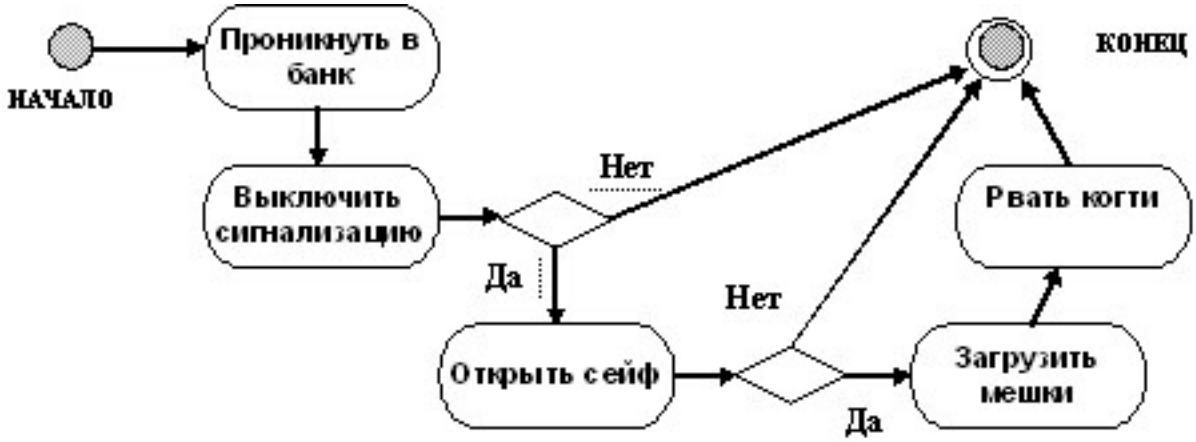


Рис. 4.13. - Диаграмма деятельности (активности).

Диаграмма компонентов (***Component diagram***) – диаграмма, показывающая организацию совокупности компонент (файлов) и их зависимости.

Диаграмма развертывания (***Deployment diagram***) – диаграмма, показывающая узлы и отношения между ними (физическое размещение компонент в аппаратных средствах).

Применение языка UML существенно облегчается за счет следующих механизмов:

- ***спецификаций*** (***specifications***) – текстовых описаний синтаксиса и семантики соответствующих элементов модели (диаграммы);
- ***принятых делений*** (***common divisions***) – в соответствии с дихотомией «класс/объект» и дихотомией «интерфейс/реализация»;
- ***механизмов расширения*** (***extensibility mechanisms***) – обеспечивающих расширение синтаксиса и семантики языка.

Предусмотрены следующие механизмы расширения языка:

- ***стереотипы*** (***stereotypes***), позволяющие расширять словарь языка и представлять на основе существующих элементов новые специфичные для конкретных проблем элементы модели;
- ***помеченные значения*** (***tagged values***), позволяющие расширять свойства элементов языка и представлять новые атрибуты и новую информацию в спецификацию элемента;
- ***ограничения*** (***constraint***), позволяющие расширять семантику элементов языка и определить новые или изменить старые правила.

UML не зависит от процесса его использования или метода моделирования. Однако, лучше всего он поддерживает ***Рациональный Унифицированный Процесс*** (***Rational Unified Process – RUP***) изготовления программных продуктов.

В общих чертах процесс анализа и моделирования системы в терминах UML представляет собой последовательность следующих шагов:

- Анализ внешних по отношению к данной систем и формулирование требований к моделируемой системе с их точки зрения. Результаты этого шага представляются в виде диаграммы прецедентов.
- Уточнение потоков событий в прецедентах, например, в виде диаграммы деятельности или текстового документа.
- Определение типов объектов (классов) и их свойств, с помощью которых можно будет реализовать деятельность в прецедентах. Результаты этого шага представляются в виде диаграммы классов.
- Декомпозиция моделируемой системы и представление ее модели в виде диаграммы кооперации (взаимодействия объектов).

Консорциум OMG начал разрабатывать UML 2.0, чтобы преодолеть существенные недостатки ранних версий. В число проблем, которые пытаются решить архитекторы стандарта UML 2.0, входит проблема разбухания ранних версий и недостатки в определении семантики. В процессе работы над новым стандартом возникла необходимость включить в него поддержку ***разработки на базе моделей*** (***Model-Driven Development – MDD***) и, в частности, подхода к такой разработке с позиций ***архитектуры на базе моделей*** (***Model-Driven Ar-***

chitecture – MDA). В результате стандартизации языка моделирования общественными усилиями был разработан очень громоздкий и сложный вариант стандарта. По мнению известных экспертов [106, с. 34] «Кажется, конечный результат полностью противоречит благим намерениям, породившим радикальный пересмотр стандарта. Многочисленные концепции моделирования, плохо определенная семантика и облегченные механизмы расширения UML 2.0 затрудняют его изучение и применение в MDD» ... «Язык UML 2.0 в некоторых отношениях превосходит предыдущие версии, но его громоздкость и сложность могут создать проблемы для пользователей, разработчиков инструментальных средств и рабочих групп OMG, развивающих этот стандарт».

4.1.4. Процесс объектно-ориентированного моделирования/проектирования: начальная фаза; исследование; построение; внедрение; дополнительные средства

В данном пункте представлено описание стандартного процесса объектно-ориентированной разработки информационного (программного) обеспечения бизнеса – ***Рационального Унифицированного Процесса – РУП*** (***Rational Unified Process***). Для его составления использованы работы [79 и 81].

Основным направляющим фактором разработки ПО является его ***системная архитектура***. Архитектура программной системы представляет собой совокупность существенных решений, охватывающих ее структурные и поведенческие аспекты, использование (функциональность), производительность, гибкость, возможность повторного использования компонент, полноту, ограничения и компромиссы, а также эстетические вопросы. Выбор архитектуры должен учитывать аспекты рассмотрения создаваемого программного продукта представленные на рисунке 4.14 и в таблице 4.1, из которых точка зрения вариантов использования (прецедентов) является основной.

Рис. 4.14. - Аспекты представления системной архитектуры.

Основанный на архитектуре и управляемый прецедентами процесс разработки ПО является итеративным и инкрементным. Таким образом, он, во-первых, подразумевает создание версий программной системы и, во-вторых – развитие системной архитектуры при выпуске новых версий. ***Этот пошаговый процесс, предполагает постепенное проникновение в суть проблемы с по-***

мощью последовательных уточнений, в котором ПО разрабатывается и реализуется по частям, путем получения все более емкого решения.

Таблица 4.1. Аспекты представления системной архитектуры.

Вид	***Аспект представления***	***Диаграммы представления UML***	
		Статики	***Динамики***
Прецедентов.	Варианты использования системы конечным пользователем, ее поведение.	Прецедентов.	Деятельности, состояний и взаимодействия.
Проектирования.	Поддержка функциональных требований. Словарь предметной области (классы, интерфейсы, кооперации).	Классов и объектов.	Последовательности и кооперации.
Процессов.	Процессы и потоки, с точки зрения производительности, пропускной способности, масштабируемости.	– « –	Деятельности, состояний и взаимодействия.
Реализации.	Компоненты и файлы с точки зрения сборки системы и управления конфигурацией версий.	Компонентов.	– « –
Развертывания.	Узлы физической системы с точки зрения распределения, поставки и установки.	Развертывания.	– « –

Данный эффект достигается вследствие того, что РУП состоит из ***фаз***, в которых достигаются конкретные хорошо определенные цели и принимается решение о переходе к следующей фазе: **НАЧАЛО, ИССЛЕДОВАНИЕ, ПОСТРОЕНИЕ, ВНЕДРЕНИЕ**. При этом каждая фаза состоит из итераций, содержащих ***рабочие процессы***, соответствующие этапам жизненного цикла ПО. В каждой фазе и на каждой итерации основные усилия сосредотачиваются на различных аспектах процесса разработки. В таблице 4.2 показан общий вид этого процесса.

Рабочие процессы выполняются в каждой фазе и на каждом шаге итерации. При этом в разных фазах основной упор делается на разных работах. Внутри каждого рабочего процесса сосредоточены связанные между собой артефакты (artifact) – документы, программы и т.д. и деятельности (activity) – анализ, выполнение, а также способы и рекомендации по решению различных задач.

Одно прохождение четырех фаз процесса называется ***циклом разработки***. Если после этого работа над проектом не заканчивается, то полученный продукт продолжает развиваться и снова проходит цикл из четырех фаз, каждая из которых состоит из рабочих процессов, представленных в таблице 4.3.

Таблица 4.2. Фазы и рабочие процессы РУП.

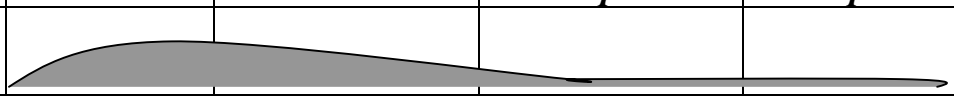
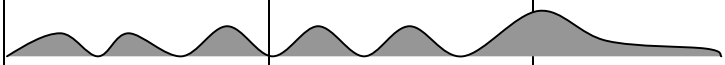
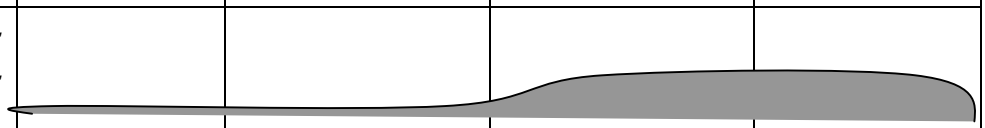
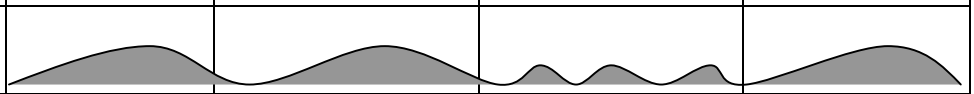
Рабочие процессы:	***Начало***	***Исследование***	***Построение***	***Внедрение***
Моделирование бизнес-процессов				
Требования				
Анализ и проектирование				
Реализация				
Тестирование				
Развертывание				
Управление конфигурацией и изменениями				
Управление проектом				
Рабочая среда				
Вспомогательные процессы				
Номер итерации →	1 … n	n+1 … m	m+1 … k	k+1 … s

Таблица 4.3. Цикл разработки.

Наименование процесса:	***Содержание процесса:***
Моделирование бизнес-процессов	Описывается структура и динамика организации.
Требования	Описываются требования пользователей методом, основанным на прецедентах.
Анализ и проектирование	Описывается предметная область и различные виды архитектуры разрабатываемой системы.
Реализация	Разрабатывается программные модули, проводится автономное тестирование и их интеграция.
Тестирование	Описываются тестовые сценарии, процедуры и метрики для измерения числа ошибок.
Развертывание	Осуществляется конфигурирование поставляемой системы.
Управление конфигурацией и изменениями	Управление изменениями и поддержание целостности проекта.
Управление проектом	Выработка и описание различных стратегий работы с итеративным процессом.
Рабочая среда	Рассмотрение вопросов инфраструктуры, необходимой для разработки системы.

Начальная фаза проекта (Inception)

В начальной фазе пишется экономическое обоснование и определяются границы и бизнес-цели проекта. В результате итеративно выполняемых рабочих процессов на данной фазе вырабатываются критерии успешности, оцениваются риски, определяются необходимые ресурсы и составляются планы, нередко создается исполняемый прототип, демонстрирующий реалистичность концепции.

Начальная фаза должна занимать несколько дней работы, в течение которых следует решить, стоит ли проводить более глубокие исследования, которые могут продолжаться несколько месяцев. Именно в этой фазе спонсор проекта принимает на себя определенные обязательства и дает согласие не более чем на серьезное рассмотрение проекта.

Важнейшие цели:

- Определить функциональные возможности создаваемого ПО и его эксплуатационные свойства, а также критерии, по которым будет определяться его готовность.
- Определить наиболее трудоемкие и критичные варианты использования ПО и их сценарии.
- Проработать архитектуру ПО для таких сценариев.
- Оценить стоимость и сроки выполнения проекта.
- Дать обобщенную оценку потенциальных рисков.
- Определить инфраструктуру (рабочую среду) проекта.

Важнейшие действия:

- Анализ бизнес-процессов, для которых разрабатывается ПО.
- Описание контекста и наиболее важных требований к ПО.
- Оценка альтернатив по управлению рисками, по укомплектованности персоналом и другим ресурсам, по планированию процесса разработки.
- Оценка соотношения стоимости и доходности ПО.
- Выбор инструментов.
- Создание основного документа «Vision» (см. ниже в конце данного пункта).

Исследование (Elaboration)

С этой фазы начинается собственно проектирование. На этой фазе необходимо проанализировать предметную область, сформулировать большую часть требований к разрабатываемой системе, определить ее структуру и принять архитектурные решения, а также устранить наиболее опасные риски. Для подтверждения правильности принятых решений и целесообразности построения системы создается прототип, демонстрирующий выбранные принципы в действии и реализующий некоторые наиболее важные прецеденты.

В этой фазе следует дать ответы на следующие вопросы [81]:

- Что вы на самом деле собираетесь создать?
- Как вы собираетесь это сделать?
- Какую технологию вы собираетесь использовать?

Решая эти вопросы необходимо учитывать проблемы, которые могут быть присущи данному проекту. По мнению, например М. Фаулера, существуют четыре категорий проблем, которые он называет «***рисками проекта***»:

- ***Риски, связанные с требованиями***. Самая большая опасность заключается в том, что построенная система не будет удовлетворять требованиям пользователей. Во время фазы исследования необходимо как следует разобраться с требованиями и их относительными приоритетами.
- ***Технологические риски***. Необходимо разобраться имеется ли достаточный опыт для применения требуемой или выбранной технологии проектирования. Насколько хорошо эта технология работает? Можно ли с помощью данной технологии действительно реализовать те функции, которых требуют пользователи.
- ***Риски, связанные с квалификацией персонала***. Имеются ли сотрудники с необходимым опытом и квалификацией?
- ***Политические риски***. Существуют ли силы, которые могут оказаться на вашем пути и серьезно повлиять на проект?

Одной из важнейших средств уточнения требований пользователей (***устранения рисков, связанных с требованиями***) является построение ***диаграммы вариантов использования*** **(*прецедентов*)**, которая является «движущей силой» всего процесса разработки. Вариант использования отражает типичное взаимодействие пользователя с системой для достижения некоторых целей и является основой для общения и взаимопонимания между спонсорами (заказчиками) и разработчиками при планировании проекта. Необходимо определить все возможные варианты использования разрабатываемой системы. Для этого в фазе исследования планируется встреча с пользователем (заказчиком) с целью определения таких вариантов.

Варианты использования не рекомендуется [81] чересчур детализировать, вполне достаточно текстового описания размером от одного до трех абзацев. Текст должен быть достаточно содержательным и для пользователей, чтобы они поняли основную идею, и для разработчиков, чтобы они хорошо представляли, что скрывается за этим описанием.

Другой важнейшей задачей является разработка ***концептуальной модели предметной области*** (***модели анализа***). Концептуальная модель предметной области закладывает основы для моделирования объектов, которые будут позднее использоваться в процессе разработки объектной модели системы. Модель предметной области это любая модель, описывающая ту часть реального мира, для которой создается компьютерная система, независимо от стадии, на которой находится процесс разработки.

Модель предметной области строится, как правило, до определения любого варианта использования; ее назначение – исследование лексики предметной области в терминах, имеющих значение и смысл для экспертов (заказчиков). Модели предметной области и варианты использования охватывают функциональные требования к системе.

Команда, занимающаяся моделированием предметной области, должна представлять собой небольшую группу (от двух до четырех человек), в состав

которой входят разработчики и эксперты предметной области. Минимально жизнеспособная команда – это один разработчик и один эксперт. Эксперт предметной области (и, желательно, разработчик тоже) должен пройти обучение использованию соответствующих диаграмм UML для концептуального моделирования [81].

Моделирование предметной области является весьма важным дополнением вариантов использования. Для выявления и описания вариантов использования модель предметной области следует показывать эксперту и определять по его реакции правильность сложившегося представления о бизнес-процессах. Для построения моделей предметной области наиболее важными являются следующие два инструмента UML [81]:

- ***Диаграммы классов***. Построенные с концептуальной точки зрения они очень хорошо подходят для выражения понятий предметной области. С помощью этих диаграмм можно выразить те понятия, которые эксперты предметной области используют в своей деятельности, и определить способы, с помощью которых эксперты связывают отдельно взятые понятия в стройную систему.
- ***Диаграммы деятельности***. Они дополняют диаграммы классов за счет описания потоков работ, т.е. пошаговых процессов выполнения работ отдельными людьми. Ключевым аспектом диаграмм деятельности является то, что они стимулируют поиск параллельных процессов, важный с точки зрения исключения избыточных последовательностей в бизнес-процессах.

Не рекомендуется слишком заботиться о строгости и пытаться зафиксировать каждую деталь. При этом может быть построено множество несвязанных диаграмм. Тем не менее, такой процесс достаточно быстро приводит к пониманию проблемы. Благодаря этому пониманию можно легко идентифицировать варианты использования для различных пользователей. После этого с помощью одного или двух экспертов следует объединить различные диаграммы в единую согласованную модель предметной области, которая будет удовлетворять всем требованиям, зафиксированным в более ранних разрозненных моделях. Эта модель может затем использоваться в качестве отправной точки для построения классов и более тщательного проектирования классов в фазе построения. Если эта модель велика, то, используя пакеты (packages), ее можно разделить на составные части.

Моделирование предметной области существенно продвигается по мере определения вариантов использования. Как только появляются очередные варианты использования, команда разработчиков должна оценить, насколько сильно они влияют на модель предметной области. Если сильно, то их нужно исследовать повнимательнее, если нет – отложить на некоторое время.

Команда должна интенсивно работать в течение всей фазы исследования, пока не завершит модель. В течение этого времени менеджер проекта должен следить, чтобы команда, с одной стороны, не завязла в трясине мелких деталей, а с другой стороны, не витала в облаках, а стояла ногами на земле. Когда они разберутся в сути того, что делают, погружение в мелкие детали станет самой

большой опасностью. В этой ситуации, чтобы сконцентрировать их внимание в нужном направлении, рекомендуется установить сжатый срок завершения работы.

Чтобы лучше понять требования к системе, рекомендуется построить прототип любых более или менее сложных вариантов использования. Прототипирование – это хороший способ для достижения наилучшего понимания динамики функционирования системы. Для выделения частей системы, которые нуждаются в прототипировании, используют модель предметной области.

После того как построены модели предметной области и вариантов использования, разрабатывается ***модель проектирования*****,** которая, с одной стороны, представляет информацию о системе, отражающуюся в объектах предметной области, и, с другой стороны, поведение системы, отражающееся в вариантах использования. Модель проектирования добавляет классы, выполняющие некоторую реальную работу в системе и образующие повторно используемую архитектурную основу для ее последующих расширений. В больших проектах можно построить промежуточную модель анализа для исследования влияния внешних требований на принятие проектных решений.

РУП не требует, чтобы фазы исследования или построения системы выполнялись «подобно водопаду». На самом деле важно правильно определить основные классы предметной области и варианты использования, затем построить повторно используемую архитектуру системы, обеспечивающую последующие расширения. Новые варианты использования могут постепенно добавляться и включаться в модель проектирования как часть итеративного процесса разработки. Не следует пытаться строить всю систему «одним махом».

Самые большие технологические риски возникают при сборке разнородных компонент в единый проект, а не в каждом из компонентов в отдельности. Поэтому самый хороший способ ***справиться с технологическими рисками*** – это строить прототипы с помощью той технологии, которая будет применяться в фазе построения системы.

Можно достаточно хорошо разбираться в C++ и в реляционных базах данных, однако применение одновременно того и другого может оказаться не таким уж простым делом. Именно поэтому рекомендуется на ранней стадии процесса создания системы совместно испытать все средства, которые будут использоваться в дальнейшем.

На этой стадии рекомендуется также отработать все архитектурные решения. При этом обычно имеются в виду состав основных компонент и способ их построения. Это особенно важно, если создается распределенная система. Особенное внимание необходимо уделить тем решениям, которые будет трудно изменить впоследствии. Проектирование необходимо выполнять таким образом, который позволил бы относительно легко вносить изменения в проект. Необходимо дать ответы на следующие вопросы:

- Что случится, если какие-то технологические элементы не будут работать?
- Что, если не удастся соединить между собой два фрагмента мозаики?
- Какова вероятность возникновения незапланированных трудностей? Если

они все-таки возникнут, каким образом мы могли бы с ними справиться?

Как и модель предметной области, необходимо внимательно анализировать варианты использования по мере их появления, чтобы оценить, нет ли в них чего-либо, что может привести в негодность разрабатываемый проект.

Простейшим способом, позволяющим ***устранить риски, связанные с квалификацией персонала***, является обучение. Тем не менее, самой распространенной ошибкой при выполнении объектно-ориентированной разработки по мнению, М. Фаулера, является недостаточное внимание, уделяемое обучению.

Обучение – это способ избежать ошибок, поскольку учителя их уже сделали. Ошибки отнимают время, а время стоит денег. Таким образом, не обучившись, вы понесете те же расходы, только проектирование при этом продлится дольше, чем нужно. При этом необходимо иметь в виду, что небольшой формальный курс обучения может быть полезным, но это всего лишь начало. Он даст общее представление о том, что необходимо знать, но не может дать квалификации, необходимой для выполнения серьезных проектов [81].

Если планируется пройти небольшой курс обучения, рекомендуется уделить серьезное внимание выбору преподавателя. Рекомендуется потратить больше денег на преподавателя, который способен доходчиво донести свои знания до слушателей и, таким образом, обеспечить хорошее понимание процесса. Кроме того, курсы следует проводить именно по тем проблемам, которыми вы в настоящее время занимаетесь. Если вы не примените свои знания на практике сразу после окончания курса, то скоро все забудете.

Для приобретения практических навыков применения объектно-ориентированного подхода рекомендуется воспользоваться помощью опытного наставника, под руководством которого команда будете работать над проектом в течение достаточного периода времени. Такой наставник покажет, как делаются те или иные вещи, будет наблюдать за тем, что делается, и давать по ходу дела полезные советы и рекомендации. Наставник вникает в специфику проекта и знает, какие навыки и когда лучше использовать. На ранних стадиях проекта он является одним из участников команды разработчиков, помогая принимать правильные решения. Со временем команда приобретает необходимый опыт, и наставник станет скорее наблюдать, чем делать что-либо. Рекомендуется найти такого наставника, который не только сам обладает знаниями, но и может передать их другим. Выбор хорошего наставника может оказаться самым важным фактором успеха проекта; не рекомендуется экономить на качестве.

Если нет возможности привлечь постоянного наставника, рекомендуется устраивать разбор проекта хотя бы через каждую пару месяцев или около этого. При этом для анализа различных аспектов проекта следует на несколько дней приглашать опытного наставника. В течение этого времени он может проанализировать любые части проекта, заслуживающие повышенного внимания, выдвинуть дополнительные идеи и наметить какие-либо полезные методы, которыми команда не владеет. Хотя от постоянного наставника гораздо больше пользы, такой способ тоже может сыграть важную роль в успехе проекта.

Разумеется, можно также повысить свою квалификацию путем чтения нужной литературы. Рекомендуется, по крайней мере, каждый месяц читать серьезную техническую литературу. Еще лучше заниматься ее изучением совместно с другими разработчиками. Рекомендуется найти пару других людей, желающих прочесть ту же самую книгу и договорится с ними читать по несколько глав в неделю и в течение часа или двух обсуждать прочитанное. Если поступить таким образом, то можно лучше понять книгу, чем при чтении ее в одиночку. Менеджерам рекомендуется поощрять сотрудников к такому подходу. Необходимо обеспечить помещение и время для такого группового чтения и обсуждения; выделить средства на приобретение технической литературы. В среде разработчиков такая групповая работа над книгами считается весьма полезной.

Одним из самых важных результатов фазы исследования является ***базовая архитектура*** разрабатываемой системы. Эта архитектура включает:

- перечень вариантов использования, определяющих требования к системе;
- модель предметной области, которая отражает понимание бизнеса разработчиком и служит отправным пунктом для формирования основных классов предметной области;
- технологическую платформу, определяющую основные элементы технологии реализации системы и их взаимодействие.

Эта архитектура является основой всей разработки, она служит своего рода проектом для последующих фаз. В дальнейшем неизбежны незначительные изменения в деталях архитектуры, однако серьезные изменения маловероятны.

Кроме того, результатом данной фазы является ***план***, определяющий последовательность итераций построения системы и варианты использования, реализуемые на каждой итерации. Для каждого варианта использования должна быть определена своя итерация и назначена дата начала каждой итерации. На данном этапе более детальное планирование не рекомендуется. Для решения задачи планирования рекомендуется выполняются следующие начальные шаги [81]:

- Пользователи должны указать уровень приоритетности для каждого варианта использования. Три уровня: «Эта функция абсолютно необходима для любой реальной системы»; «Какое-то небольшое время я смогу прожить без этой функции»; «Эта функция важна, но пока я могу обойтись и без нее».
- Разработчики должны принимать во внимание «архитектурный» риск, связанный с каждым вариантом использования, который заключается в следующем: если реализацию данного варианта использования отложить на слишком долгое время, то вся выполненная до этого работа может потребовать значительной переделки. Три категории оценки: а) высокая степень риска, б) возможно, но маловероятно и в) маленький риск.
- Разработчики должны оценить степень своей уверенности относительно объема работы, требуемого для реализации каждого варианта использо-

вания. Это называется риском планирования. Три уровня: «Я абсолютно уверен в том, сколько времени потребуется на реализацию»; «Я могу оценить время только с точностью до человеко-месяца»; «У меня нет никаких соображений по этому поводу».

После выполнения этих шагов рекомендуется с точностью до человеко-месяца оценить время, требуемое для реализации каждого варианта использования. Данная оценка исходит из предположения, что придется выполнить анализ, проектирование, кодирование, автономное тестирование, интеграцию и документирование. Предполагается также, что разработчики будут полностью, без каких-либо отвлечений, заняты в данном проекте.

Оценку должны выполнять разработчики, а не менеджеры. При этом, разумеется, нужно быть уверенным, что разработчик, оценивающий данный вариант использования, разбирается в этом наилучшим образом [81].

После выполнения таких оценок рекомендуется выделить те варианты использования, которые обладают высоким риском планирования. Если с этими вариантами связана большая часть проектного времени или высокая степень «архитектурного» риска, то необходимо выполнить дальнейшее исследование.

Следующий шаг заключается в определении длительности итерации. Рекомендуется, чтобы длина итерации была фиксированной на протяжении всего проекта, тогда можно будет получать результаты с заданной регулярностью. Итерация должна быть достаточно длительной для того, чтобы успеть реализовать некоторое количество вариантов использования. В частности, при применении C++ – от шести до восьми недель.

Теперь рекомендуется рассмотреть трудоемкость каждой итерации. Например, эффективность работы разработчиков составляет в среднем 50%, т.е. на реализацию вариантов использования они расходуют половину своего рабочего времени. Следует умножить длительность итерации на количество разработчиков и на 0,5. В результате получится трудоемкость каждой итерации. Например, если имеется восемь разработчиков и продолжительность итерации составляет три недели, то на каждую итерацию потребуется 12 человеко-недель (8x3x0,5).

Далее рекомендуется просуммировать время на реализацию всех вариантов использования, разделить на трудоемкость одной итерации и добавить единицу. В результате получится начальная оценка количества итераций, которое потребуется для вашего проекта.

Следующий шаг заключается в распределении вариантов использования по итерациям. Варианты использования, обладающие высоким приоритетом, «архитектурным» риском и/или риском планирования, следует реализовывать в первую очередь. Рекомендуется разделить слишком большие варианты использования на несколько менее крупных и, возможно, пересмотреть некоторые начальные оценки вариантов использования в соответствии с порядком их реализации.

На внедрение (тонкую настройку и компоновку конечного продукта) рекомендуется отвести от 10% до 35% времени построения. (Если нет опыта выполнения этих операций в данной среде, то времени отводится еще больше.)

Затем рекомендуется добавить фактор случайности – от 10% до 20% времени построения, в зависимости от степени риска. Воздействие этого фактора обычно ощущается в конце фазы внедрения. Для разработчиков выход конечного продукта планируется без учета этой случайности, однако внешний план для пользователей должен ее учитывать.

После выполнения всех перечисленных рекомендаций будет создан план, в котором для каждой итерации указаны реализуемые варианты использования. Этот план отражает согласованное мнение разработчиков и пользователей, его вполне можно назвать согласованным планом. Такой план не является чем-то застывшим – разумеется, вполне возможно, что по ходу проекта он будет изменяться. Однако поскольку этот план согласованный, постольку изменения в него должны вноситься совместно разработчиками и пользователями.

Итак, как можно видеть из предыдущего обсуждения, варианты использования служат основой для планирования проекта, и именно поэтому в UML им уделяется такое серьезное внимание.

Основными признаками завершения фазы уточнения являются два события:

- Разработчики в состоянии оценить с достаточно высокой точностью, сколько времени потребуется на реализацию каждого варианта использования.
- Идентифицированы все наиболее серьезные риски, и самые важные из них осознаны настолько, что известно, как с ними справиться.

Важнейшие цели:

- Гарантировать устойчивость (стабильность) требований, архитектуры и планов.
- Смягчить риски.
- Создать прототип.
- Обосновать соответствие архитектуры требованиям и трудоемкость (стоимость, сроки) разработки.
- Обеспечить инфраструктуру (рабочую среду) проекта.

Важнейшие действия:

- Определение и согласование архитектуры ПО и планов на фазу построения.
- Доработка документа «Vision» (см. в конце данного пункта).
- Создание инфраструктуры (рабочей среды) проекта.
- Отбор готовых компонентов для повторного использования.

Построение (Construction)

На стадии построения разработка системы выполняется путем серии итераций. Каждая итерация является своего рода мини-проектом. На каждой итерации для конкретных вариантов использования выполняются анализ, проектирование, кодирование, тестирование и интеграция. Итерация завершается демонстрацией результатов пользователям и выполнением системных тестов с целью контроля корректности реализации вариантов использования.

Причиной такого построения процесса разработки является необходимость снижение степени риска. Причиной появления риска является откладывание решения сложных проблем на самый конец проекта. При итеративной разработке на каждой итерации выполняется весь процесс, что позволяет оперативно справляться со всеми возникающими проблемами.

Главная идея итеративной разработки – поставить весь процесс разработки на регулярную основу с тем, чтобы команда разработчиков смогла получить конечный продукт. Однако есть некоторые вещи, которые не следует выполнять слишком рано, например оптимизация.

Оптимизация снижает прозрачность и расширяемость системы, однако повышает ее производительность. В этой ситуации необходимо принять компромиссное решение – в конце концов, система должна быть достаточно производительной, чтобы удовлетворять пользовательским требованиям. Слишком ранняя оптимизация затруднит последующую разработку, поэтому ее следует выполнять в последнюю очередь.

Рекомендуется очень серьезно относиться к тестированию. Объем написанного разработчиком кода тестов должен быть по меньшей мере равен объему кода самого программного продукта. Тестирование должно быть непрерывным процессом. Не рекомендуется писать программный код до тех пор, пока не известно, как его тестировать. Как только написан код, сразу же пишите для него тесты. Пока все тесты не отработают, нельзя утверждать, что написание кода завершено [81].

Однажды написанный тестовый код должен использоваться постоянно. Тестовый код должен быть организован таким образом, чтобы можно было запускать каждый тест с помощью простой командной строки или нажатия кнопки на графическом экране. Результатом выполнения теста должно быть «ОК» или список ошибок. Кроме того, все тесты должны проверять свои собственные результаты. Ничто не приводит к столь неоправданной трате времени, как получение на выходе теста числового значения, с интерпретацией которого нужно разбираться отдельно.

Рекомендуется разделить все тесты на автономные и системные. Автономные тесты рекомендуется писать самим разработчикам. Они должны быть организованы в пакеты и тестировать интерфейсы всех классов. Системные тесты рекомендуется разрабатывать отдельной небольшой командой, которая занимается исключительно тестированием. Такая команда должна рассматривать всю систему как черный ящик и находить особое удовольствие в поиске ошибок.

Итерации на стадии конструирования являются одновременно инкрементными (наращиваемыми) и повторяющимися.

- Итерации являются инкрементными в соответствии с той функцией, которую они выполняют. Каждая итерация добавляет очередные конструкции к вариантам использования, реализованным во время предыдущих итераций.
- Они являются повторяющимися по отношению к разрабатываемому коду. На каждой итерации некоторая часть существующего кода заново пере-

писывается с целью сделать его более гибким. В этом процессе очень часто используется метод реорганизации (см. далее). Рекомендуется внимательно наблюдать за тем, какой объем кода оказывается ненужным после каждой итерации. Если каждый раз выбрасывается менее 10% предыдущего кода, то это должно вызывать подозрение.

Процесс интеграции должен быть непрерывным. В конце каждой итерации должна выполняться полная интеграция. Тем не менее, интеграция может и должна выполняться еще чаще. Разработчику рекомендуется интегрировать приложения после выполнения любой сколько-нибудь значительной части работы. Во время каждой интеграции должен выполняться полный набор автономных тестов, чтобы обеспечить таким образом полное регрессионное тестирование.

В рамках каждой итерации рекомендуется заниматься более детальным планированием. Самой важной частью любого плана являются меры, которые нужно предпринимать, если что-то происходит не так, как было запланировано. Главная особенность итеративной разработки заключается в том, что она жестко ограничена временными рамками, и сдвигать сроки недопустимо. Исключением может быть перенос реализации каких-либо вариантов использования на более позднюю итерацию по соглашению с заказчиком. Смысл таких ограничений – поддерживать строгую дисциплину разработки и не допускать переноса сроков. Если, например, слишком много вариантов использования перенесено на более поздний срок, то пора корректировать план, пересмотрев при этом оценку трудоемкости реализации вариантов использования. На этой стадии разработчики должны иметь более глубокое представление о такой оценке.

На стадии построения полезными являются все методы UML.

Концептуальная диаграмма классов полезна для приблизительного описания некоторых понятий варианта использования и определения того, как эти понятия согласуются с уже разработанным ПО, а также более детального отображения классов. Если вариант использования содержит значительные элементы потоков работ, то рекомендуется обратиться к их представлению на диаграмме деятельности.

Преимущество этих методов на данной стадии заключается в том, что они могут быть использованы в сотрудничестве с экспертом предметной области. Анализ становится осмысленным только при непосредственном участии эксперта предметной области.

Диаграммы взаимодействия полезны для представления взаимодействия классов, в котором выражается реализация варианта использования. Если поведение класса в течение его жизненного цикла достаточно сложно описать, рекомендуется использовать диаграмму состояния.

Рекомендуется ограничить документацию теми случаями, когда она действительно помогает. Если обнаруживается, что от документации нет никакого толку, это верный признак, что что-то идет не так. Документация в нормальном случае должна содержать три основных элемента:

– одну или две страницы с описанием нескольких классов, присутствующих на диаграмме классов;

– несколько диаграмм взаимодействия, показывающих, как классы взаимодействуют между собой;
– небольшое текстовое описание, связывающее диаграммы воедино.

Важнейшие цели:

– Завершить анализ, проектирование, кодирование и тестирование всех вариантов использования (функциональных возможностей).
– Создать готовый к использованию продукт.
– Добиться надлежащего качества и полезности версий.
– Оценить готовность к развертыванию.
– Уменьшить затраты на разработку, оптимизируя продукт и используемые ресурсы.

Важнейшие действия:

– Управление ресурсами.
– Контроль качества и оптимизация.
– Разработка компонент, автономное тестирование.
– Интеграция и системное тестирование.
– Определение готовности ПО с учетом критериев его приемки.

Внедрение (Transition)

На стадии внедрения продукт не дополняется никакой новой функциональностью (кроме, разве что, самой минимальной и абсолютно необходимой). Здесь только исправляют ошибки, настраивают систему и передают ее пользователю. Хорошим примером для фазы внедрения может служить период времени между выпуском бета-версии и появлением окончательной версии продукта.

Важнейшие цели:

– Отрекламировать и сбыть продукт.
– Завершить бета-тестирование.
– Согласовать разработанный продукт с унаследованным ПО.
– Адаптировать базы данных и знаний.
– Обучить пользователей.
– Повысить работоспособность.
– Поддержать пользователей.

Важнейшие действия:

– Выполнение плана развертывания.
– Тестирование поставляемого продукта.
– Настройка поставляемого продукта.
– Сопровождение продукта у пользователя.
– Обеспечение обратной связи с пользователем.
– Обеспечение доступности продукта для пользователей.

Дополнительные средства

Реорганизация.

При дополнении программы новыми функциями можно либо перепроек-

тировать эту программу, чтобы внести изменения наилучшим образом, либо просто поправить уже существующий код. Как правило, дописывается новый код поверх существующего, не обращая при этом особого внимания на прежнюю программу, хотя теоретически программу лучше перепроектировать.

Перепроектирование, в свою очередь, выливается в большую дополнительную работу, поскольку переписывание существующей программы порождает новые ошибки и проблемы. Однако, если не перепроектировать программу, дополнения могут оказаться более сложными, чем могли бы быть в противном случае. Со временем за такую «сверхсложность» придется платить все более высокую цену. Вследствие этого, как правило, приходится идти на компромисс: перепроектирование влечет за собой серьезные трудности в краткосрочном плане, зато приносит выгоду в перспективе. Находясь под прессом жестких плановых сроков, большинство разработчиков предпочитает отодвигать эти трудности на будущее [81].

Термин «реорганизация» используется для описания методов, которые позволяют уменьшить краткосрочные трудности, связанные с перепроектированием. Реорганизация не изменяет функциональность программы; однако при этом изменяется ее внутреннюю структуру для того, чтобы сделать ее более понятной и модифицируемой. Реорганизационные изменения обычно довольно невелики: переименование метода, перемещение атрибута из одного класса в другой, консолидация двух подобных методов в суперкласс. Каждое отдельное изменение очень мало, однако даже пара часов, потраченных на внесение таких небольших изменений, могут существенно улучшить программу.

Реорганизацию можно облегчить, если следовать следующим принципам [81]:

- Не следует одновременно заниматься реорганизацией программы и расширением ее функциональности. Необходимо проводить четкую границу между этими действиями. В процессе работы можно переключаться от одного действия к другому – например, в течение получаса заниматься реорганизацией, затем потратить час на добавление новой функции и полчаса на реорганизацию только что добавленного кода.
- Перед началом реорганизации убедитесь, что у вас имеются хорошие тесты. Запускайте их как можно чаще. Таким образом, вы быстро узнаете, не нарушили ли чего-нибудь ваши изменения. Вносите изменения минимальными и тщательно обдуманными шагами. Переместите атрибут из одного класса в другой. Объедините два подобных метода в один суперкласс. Реорганизация часто включает выполнение множества небольших локальных изменений, которые, в конечном счете, выливаются в крупномасштабное изменение. Если каждый шаг будет небольшим и будет завершаться обязательным тестированием, то вы сможете избежать в будущем длительной отладки.

Реорганизацию следует выполнять в следующих случаях [81]:

- Вы расширяете функциональность своей программы и обнаружили, что с существующим кодом возникают некоторые проблемы. Тогда процесс расширения следует приостановить до тех пор, пока не будет реорганизо-

ван старый код.

– Код становится трудным для понимания. В этом случае реорганизация служит хорошим способом облегчить понимание кода и сохранить это понимание на будущее.

Потребность в реорганизации возникает, когда приходится иметь дело с кодом, написанным каким-либо другим разработчиком. Если вы занимаетесь такой реорганизацией, то делайте это вместе с автором кода. Не так просто написать код, который другие могли бы легко понимать. Наилучший способ реорганизации – это работать бок о бок с тем, кто разбирается в данном коде.

Образцы.

Образцы – примеры моделей, представляющих результаты процесса объектно-ориентированной разработки. Образцы предназначены для описания наиболее общих решений и накапливаются, обычно, у тех разработчиков, которые умеют находить в своих проектах повторяющиеся решения. Эти разработчики описывают каждое решение таким образом, чтобы другие люди могли воспринять этот образец и понять, как его применить.

Примером образца может служить ***объект-посредник*** (***proxy***), заменяющий реальный (удаленный) объект (обладающий тем же самым интерфейсом) и отвечающий за передачу ему любых сообщений. Это избавляет объекты локального процесса от проблем поиска других объектов в сети или выполнения вызовов удаленных процедур. Посредник – это проектный образец, поскольку он определяет метод проектирования.

Образцы могут использоваться и в других областях. Например, можно использовать образец ***сценария***, который является образцом анализа, поскольку он определяет фрагмент модели предметной области. Образцы анализа имеют важное значение, поскольку их хорошо использовать в самом начале работы с новой предметной областью.

«У образцов анализа есть одна интересная особенность – они могут пригодиться в самый неожиданный момент. Когда я начал работать над проектом корпоративной системы финансового анализа, мне здорово помогли образцы, накопленные мною ранее в проекте из области здравоохранения» [81].

Образец – это нечто гораздо большее, чем просто модель. Образец должен также включать обоснование, почему он именно такой, какой есть. Часто говорят, что образец представляет собой решение проблемы. Образец должен придать проблеме необходимую ясность, объяснить, почему он является решением данной проблемы, а также в каких ситуациях он работает или не работает.

Образцы очень важны, поскольку они являются следующим шагом после понимания основ языка или метода моделирования. Образцы предлагают набор готовых решений, а также показывают, что представляет собой хорошая модель и как ее построить. Они обучают на примерах.

Образцы следует использовать постоянно. Чем бы вы ни занимались – анализом, проектированием, кодированием или управлением проектами – всегда стоит заняться поиском любых доступных образцов, которые могли бы вам помочь.

Содержание документа «Vision».

1. Введение.
   1.1. Цели документа.
   1.2. Область действия документа.
   1.3. Список определений и сокращений.
   1.4. Перечень ссылок.
   1.5. Краткий обзор документа.
2. Позиции (основные характеристики) проекта.
   2.1. Возможности (деловые), получаемые в результате выполнения проекта.
   2.2. Постановка задачи (описание проблемы).
   2.3. Описание назначения (позиций, характеристик) проектируемого ПО.
3. Описание заинтересованных лиц и потребителей.
   3.1. Рынок.
   3.2. Список и описание заинтересованных лиц.
   3.3. Список и описание пользователей.
   3.4. Условия работы пользователей.
   3.5. Характеристика заинтересованных лиц.
      3.5.1. Stakeholder Name.
   3.6. Характеристика пользователей.
      3.6.1. User Name.
   3.7. Основные потребности (проблемы) заинтересованных лиц и пользователей.
   3.8. Альтернативы и конкурирующие продукты.
      3.8.1. Основные конкуренты.
      3.8.2. Прочие конкуренты.
4. Общее описание проектируемого ПО.
   4.1. Перспективы использования.
   4.2. Перечень возможностей.
   4.3. Предположения и допущения.
   4.4. Калькуляция.
   4.5. Лицензирование и ввод в эксплуатацию.
5. Свойства (особенности) разрабатываемого ПО.
   5.1. Основные свойства (особенности).
   5.2. Прочие свойства.
6. Ограничивающие условия.
7. Диапазон качества.
8. Приоритеты.
9. Другие требования к разрабатываемому ПО.
   9.1. Применяемые стандарты.
   9.2. Системные требования.
   9.3. Требования к функционированию (эксплуатационные показатели).
   9.4. Требования (экологические) к рабочей среде.
10. Требования к документации.
   10.1. User Manual (Руководство пользователя).
   10.2. Online Help (Интерактивная справка).

10.3. Installation Guides, Configuration, and Read Me File (Руководство по установке и конфигурированию).

10.4. Маркировка и упаковка.

Appendix 1 – Особые свойства / атрибуты особенностей: А.1. Статус. А.2. Польза (выгода). А.3. Объем работ. А.4. Риск. А.5. Устойчивость. А.6. Целевой выпуск (целевая версия). А.7. Назначение. А.8. Причина (основание).

Выводы

1. Объектно-ориентированная методология, использующая стандартный язык UML, предоставляет полезные средства анализа и моделирования как информационных, так и организационных систем.
2. В отличие от функциональной декомпозиции системных структурных методов при данном подходе используется объектная декомпозиция, учитывающая в определенной степени и поведение объектов.
3. Однако, в основном, она ориентирована на поведение элементов программных систем, что ограничивает возможности применения этой методологии для моделирования бизнес-процессов. Кроме того, как и в методах системного структурного анализа, в рамках объектно-ориентированного анализа отсутствуют реальные возможности имитации функционирования моделируемой системы.
4. UML лучше всего поддерживает Рациональный Унифицированный Процесс (Rational Unified Process – RUP) создания ПО. Этот пошаговый процесс, предполагает постепенное проникновение в суть проблемы с помощью последовательных уточнений, ПО, при этом, разрабатывается и реализуется по частям, путем получения все более емкого решения.

4.2. Требования к объектному моделированию бизнес-систем

Рассмотрим требования к моделированию бизнес-систем и бизнес-процессов, описанные, например, в [97].

Главной задачей, решаемой в рамках моделирования бизнеса, является анализ окружающей предприятие среды и того, как предприятие взаимодействует с этой средой. Под окружающей средой, в данном случае, понимается все, с чем предприятие взаимодействует в ходе выполнения своих бизнес-процессов, т.е. клиентов, партнеров, субподрядчиков, конкурентов и т.д. Полная модель бизнеса показывает работникам всех уровней и подразделений предприятия, что должно быть сделано, когда и как именно.

Наиболее известная модель бизнеса – организационная (иерархическая) структура предприятия. Эта модель совершенно недостаточна для того, чтобы проанализировать, спроектировать и (или) изменить предприятие (оптимизировать деловую активность, решить проблемы). Вместо этого нужны модели, показывающие предприятие в контексте его клиентов, поставщиков, партнеров и т. д., т.е. модели, которые представляют бизнес-процессы предприятия и то, как оно производит товары и услуги для внешнего мира (рынка).

Методы и средства, используемые для моделирования бизнеса, должны

обеспечивать [97]:

- Визуализацию образа существующего и будущего предприятия и окружающего его мира (рынка).
- Возможность работы с альтернативными архитектурами бизнеса и моделирования их воздействие на деятельность предприятия.
- Описание продукции предприятия в контексте того, как, когда и в ходе какого процесса она обрабатывается.
- Адаптацию выбранного архитектурного решения к существующему предприятию для его последующего внедрения.
- Описание реализации конечного проекта с учетом как человеческих, так и технических ресурсов.
- Представление реконструированного предприятия таким образом, чтобы каждый участник понял новую организацию работ, свои новые задачи и способы их выполнения, т.е. модель должна быть понятна персоналу без длительного обучения и серьезного вмешательства в его работу.

Моделирование бизнеса принято осуществлять в двух направлениях. Во-первых, в направлении обратного инжиниринга, что позволяет понять, как в настоящее время работает предприятие. Обычно это делается для того, чтобы получить прочную основу для кардинального улучшения различных аспектов предприятия в будущем или тогда, когда требуется понять и объяснить, как функционирует предприятие или некоторый его процесс. Во-вторых, в направлении прямого инжиниринга, что обеспечивает описание нового предприятия. Эта работа начинается с формулирования целей и образа (vision) будущего предприятия. После этого набрасываются различные сценарии. Для каждого сценария создается общее описание процесса, включающее заказчиков, поставщиков и т.д., а также сам процесс. Далее проводится имитационное моделирование различных процессов при помощи деловой игры или компьютерной модели. Наконец, выбранная альтернатива реализуется.

В ***бизнес-модели*** необходимо уметь выразить [97]:

1. ***Процессы***, т.е. структурированный, измеряемый набор действий, осуществляемый для того, чтобы произвести определенный выход для конкретного клиента на рынке. При моделировании бизнеса модели процессов должны показывать, как клиент может использовать бизнес, и включать в себя полный поток событий в системе, описывающий, как клиент начинает, ведет и завершает бизнес-процесс. Язык моделирования должен описывать различные типы задач или внутренних процессов, из которых состоит почти каждый бизнес-процесс, а также способ взаимодействия этих внутренних процессов при обслуживании клиента.
2. ***Ресурсы***, т.е. как внутренний процесс реализуется при помощи человеческих или технических ресурсов и откуда эти ресурсы будут взяты. Особенно важно уметь показать, как процесс может поддерживаться информационной системой. Должна существовать согласованность между внутренним процессом в организации бизнеса и требованиями, предъявляемыми к поддерживающей бизнес информационной системе.

3. ***Продукцию***. Понятия «товар» и «услуга» или их обобщение – понятие «продукция». Продукция должна иметь характеристики, которые описывают, что с ней можно делать.
4. ***Потоки событий***. Необходимо описывать входные данные из внешнего мира; действия (операции) процесса, которые он производит над исходными данными; потребляемые ресурсы; решения, которые будут приняты, и выход (результат), который процесс отправит во внешний мир. Так как «вход» и «выход» – это способы общения процесса с клиентом, они должны быть ориентированы на клиента.
5. Наконец, существуют ***общие требования к моделированию бизнеса*** в целом. При моделировании предприятия обязательно необходимо иметь в виду, что бизнес будет подвержен изменениям. Следовательно, важно, чтобы язык был гибким. Должна существовать возможность менять процессы, но при этом производить продукцию для клиентов бизнеса. Необходимо также иметь возможность адаптировать процессы к новым ситуациям. Например, если предприятие открывает филиалы, должна быть предусмотрена возможность специализировать процесс продаж так, чтобы он работал и в филиале.

При разработке модели бизнеса как в ходе прямого, так и в ходе обратного инжиниринга, согласно рассматриваемой методологии, рекомендуется создавать две модели организации: ***внешнюю*** и ***внутреннюю***.

4.2.1. Внешняя модель бизнес-системы

Внешняя модель – прецедент-модель (***П-модель***). Она описывает бизнес и его окружение, предприятие в целом и его внешний мир (сегмент рынка), а также процессы, которые удовлетворяют интересы клиентов и интересы вне предприятия. Процессы моделируются при помощи прецедентов (вариантов использования), а окружение моделируется при помощи так называемых действующих лиц или ***субъектов***.

П-модель показывает, как внешнее окружение взаимодействует с бизнесом, т.е. как отдельные субъекты общаются с бизнесом посредством прецедентов. П-модель описывает бизнес так, как он виден извне, т.е. так, как он воспринимается теми, кто хочет его использовать. Следовательно, структуры внутри бизнеса, которые невидимы для субъектов, не следует описывать в П-модели. П-модель представляет собой, по сути дела, диаграмму прецедентов UML. Чтобы создать корректную модель бизнеса, важно очертить его окружение, в котором присутствуют клиенты, и именно они накладывают на бизнес наиболее существенные требования.

Субъект обозначает роль, которую кто-то или что-то может играть по отношению к бизнесу. В рассматриваемых моделях окружение представлено субъектами. Субъекты обозначают все то в окружении, что взаимодействует с бизнесом, и подлежит моделированию. Например, это клиенты, поставщики или партнеры. Субъектов можно разделить на две группы: человеческие и технические. Например, компьютерная система другой компании может быть

представлена как технический субъект. С другой стороны, работников моделируемого предприятия или его технику не следует рассматривать как субъекты, так как они являются неотъемлемой частью самого бизнеса, т.е. они представляют собой ресурсы, которые используются для выполнения задач системой.

Важно понимать, что субъект представляет собой абстракцию кого-либо или чего-либо, использующего бизнес. Субъект может представлять различные виды явлений в окружении предприятия. Не следует путать реальных людей с субъектами. Реальная личность, в отличие от субъекта, может играть несколько ролей в бизнес-системе: конкретный человек может быть и посетителем, и одним из поставщиков ресторана.

Прецедент, в данном случае – это последовательность ***транзакций*** в системе, выполняемых для получения ***измеримой потребительской ценности*** для некоторого ***индивидуального субъекта*** бизнес-системы.

Индивидуальный субъект. Это понятие упрощает нахождение правильных прецедентов, т.е. позволяет избегать слишком сложных прецедентов. Рассмотрение прецедента следует начинать с индивидуальных субъектов – с экземпляров действующих лиц. Например, при моделировании предприятия, специализирующегося на продаже некоторой продукции, необходимо осознавать, что субъект, названный «клиент», на самом деле представляет трех различных клиентов. Во-первых, простого потребителя продукции (каких много); во-вторых, покупателя продукции, т.е. кого-то, кто разбирается в закупках, но не обязательно знает, для чего используется продукция; и, в-третьих, того, кто может компетентно судить о продукции в сравнении с конкурирующими продуктами. Каждый из этих случаев требует отдельного прецедента, так как субъекты играют в них различные роли по отношению к системе.

Измеримая потребительская ценность. Эти слова служат ключом для выбора не слишком детального уровня прецедента. Должна быть предусмотрена возможность оценить эффективность прецедента в терминах цены или трудоемкости. Например, обращение в банк за кредитом имеет ценность для клиента банка.

Транзакция – это неделимое множество действий, которые или выполняются все целиком, или не выполняются вообще. Она запускается при получении системой стимула от субъекта или при наступлении определенного момента времени. Транзакция состоит из набора действий, решений и передачи стимулов некоторому субъекту (субъектам).

Следует различать ***класс прецедентов*** и ***экземпляр прецедента***. Класс прецедентов определяется его описанием, а экземпляр прецедента – действиями, которые выполняются, когда поток событий, соответствующий описанию прецедента, проходит через систему. Класс прецедентов может содержать несколько альтернативных путей через систему, но экземпляр следует только одному из них.

При описании модели нужно иметь возможность говорить как об общих характеристиках некоторого типа субъектов, так и об уникальных характеристиках конкретного субъекта. Другими словами, надо иметь возможность различать класс субъектов и экземпляр субъекта. В модели конкретные люди яв-

ляются не классами, а экземплярами некоторого класса. Субъект (класс) представляет собой роль, которую реальные люди могут играть. Один человек может играть несколько ролей, т.е. реализовывать несколько классов. Субъекты взаимодействуют с системой, посылая стимулы. Чтобы лучше понять субъект, необходимо знать, в каком прецеденте он участвует. В П-модели это указывается отношениями между субъектом и прецедентом.

Чтобы придать П-модели большую глубину, составляются детальные описания каждого прецедента. Следует не просто описывать поток событий в прецеденте, но и то, как он взаимодействуют с окружением, т.е. с субъектами. Таким образом, прецедент это такая «машина состояний-событий», в которой состояние экземпляра прецедента представляет, какие стимулы могут быть получены следующими прецедентами и какие стимулы могут вызвать переход в другое состояние, после завершения транзакции. Экземпляр прецедента, будучи инициированным, может пройти через некоторое количество состояний до того, как он завершится.

Цель П-модели – внешнее представление системы. Это означает, что модель предназначена в первую очередь для заказчиков и пользователей системы, а не для тех, кто ее реализует. Учитывая вышеизложенное, необходимо помнить, что надо и чего не надо показывать в П-модели [97]:

- В модели не показывается коммуникация между потоками событий. Таким образом, экземпляры прецедента не могут общаться друг с другом, иначе пришлось бы описывать интерфейс между ними.
- Экземпляры одного или различных классов прецедентов, очевидно, будут влиять в бизнесе друг на друга. Конечно, такие отношения очень важны, но они показывают детали внутренней работы бизнеса, не интересные людям, для которых предназначена П-модель. Следовательно, эти влияния будут показаны во внутренней модели.
- В П-модели не показывается параллельность хода событий. Предполагается, что прецедент интерпретируется как один экземпляр и одна транзакция в каждый промежуток времени. Следовательно, транзакции прецедентов описываются так, как если бы они были неделимыми и последовательными.
- В соответствии со всем сказанным выше, в П-модели показываются только отношения классов.

П-модель – хорошее средство как для описания требований к бизнесу, так и для представления наглядной картины того, что бизнес выполняет. Тем не менее, П-модель не дает ясной картины о внутреннем устройстве бизнеса. Она ничего не говорит о том, как различные виды деятельности реализуют бизнес-процессы, как эти виды деятельности связываются вместе в цепочки процессов, какой вид ресурсов должен использоваться для реализации того или иного вида деятельности и т. д.

П-модель иллюстрирует функции бизнеса, его окружение и бизнес-процессы, которые он предлагает внешнему миру. Для более полного понимания бизнеса требуется описание его более полное чем то, которое дается П-моделью.

4.2.2. Внутренняя модель бизнес-системы

Внутренняя модель – объект-модель (***О-модель***). Она описывает, как строится каждый бизнес-процесс предприятия из различных рабочих задач (внутренних процессов) и какие ресурсы он использует. Внутренняя модель использует объекты, соответствующие рабочим задачам, и объекты, соответствующие предметам бизнеса, например продукцию. Если прецеденты выражают, что бизнес должен делать, то внутренняя модель описывает, как бизнес работает. Таким образом, П-модель – есть «что модель», а О-модель – «как модель», представляющая собой UML-диаграммы классов и взаимодействия.

Используются два вида внутренних моделей: идеальные и реальные. ***Идеальная О-модель*** – это модель, не учитывающая, как бизнес должен реализоваться на практике. Такая модель не учитывает, например, что предприятие (фирма) географически распределена на несколько филиалов. ***Реальная О-модель*** учитывает эти факторы. Она учитывает, что в настоящее время предприятие не располагает персоналом, имеющим уровень компетенции, предполагаемый идеальной моделью. Во многих случаях достаточно построить идеальную модель и предоставить предприятию возможность решать, как следует работать, чтобы приблизиться к реальной модели.

При описании ***О-модели*** объекты представляют участников процессов и различного рода сущности (продукция, предметы, задачи), используемые в ходе их выполнения. Следует различать класс объектов и экземпляр класса (объект). Имя, которое дается классу объектов, должно как можно более ясно выражать суть свойственную экземплярам этого класса, т.е. объектам. Классам часто дают имена, состоящие из нескольких слов (как правило, существительных), так как ясность важнее, чем краткость.

Объект может соответствовать задачам, продукции или сущностям в бизнесе. Задачи могут быть подразделены на два типа: те, что обеспечивают взаимодействие субъектов с бизнесом, и те, которые являются чисто внутренними. Удобно определить различные типы объектов, для того чтобы сделать более ясными задачи, которые они выполняют в модели. Например, А. Якобсон [97] предлагает различать следующие типы объектов: ***объекты-сущности***, ***управляющие объекты*** и ***интерфейсные объекты***. При этом интерфейсные и управляющие объекты представляют задачи, а не типы ресурсов. С другой стороны, эти задачи выполняются людьми, относящимися к определенной категории ресурсов. Часто, однако, несколько задач помещаются вместе в один объект, который реализуется одним конкретным экземпляром ресурса.

Интерфейсные объекты представляют в бизнесе операции, каждая из которых должна выполняться одним и тем же ресурсом. Эта задача включает взаимодействие с окружением бизнеса. Следовательно, к коммуникабельности людей, выполняющих эту задачу, предъявляются определенные требования.

Интерфейсный объект может участвовать в нескольких прецедентах. Часто объекты этого типа несут ответственность за координацию процесса в той его части, которая касается непосредственного контакта с клиентами. Каждый конкретный клиент должен иметь как можно меньше контактов с предприятием. Это, впрочем, не противоречит теориям, которые рекомендуют работникам

предприятия быть ближе к клиенту, а просто означает, что в интересах клиента его контакты с предприятием должны быть как можно проще и реже. Можно сказать, что часть бизнеса, имеющая непосредственный контакт с внешним миром, видима как интерфейсные объекты, в то время как объекты-сущности и управляющие объекты более независимы от окружения.

Управляющие объекты, как и интерфейсные объекты, представляют операции в бизнесе. Различие заключается в том, что эти операции не имеют непосредственного контакта с окружением бизнеса. Название «управляющий объект» используется потому, что эти объекты активны, они управляют или принимают участие в потоке управления при обработке продукции. Следовательно, экземпляр управляющего объекта часто имеет то же время жизни, что и экземпляр прецедента, в котором он участвует. Типичные примеры управляющих объектов на предприятии – это ***Разработчик*** и ***Менеджер Проекта***.

Объекты-сущности представляют такие сущности, как продукция и предметы, которые обрабатываются бизнесом. Объект-сущность участвует не только в одном прецеденте. Один и тот же экземпляр может принимать участие во многих событиях в бизнесе. В отличие от интерфейсных и управляющих объектов, объекты-сущности представляют сущности, не являющиеся человеческими или техническими ресурсами. Типичные примеры объектов-сущностей: ***Продукция***, ***Извещение*** (***Invoice***) и ***Заказ***.

О-модель с помощью устанавливаемых между объектами отношений показывает, каким образом реализуются прецеденты, описанные в П-модели. Данные отношение связывают два объекта (экземпляра или класса) и изображаются стрелками (дугами). Изучая прецеденты, в которых участвует объект, можно получить представление об обязательствах объекта по отношению к его окружению. Поведение, описываемое в классе объекта, должно подтверждать эти обязательства. Если необходимо более подробно показать, как объекты взаимодействуют при выполнении прецедентов, то для этого привлекают диаграммы взаимодействий.

П-модель и О-модель – это разные типы описания бизнеса. О-модель, состоящая из сотен объектов и более, неудобна для работы и малопонятна, так как слишком сложна. Она содержит детали того, как объекты связаны друг с другом, как они общаются, какой интерфейс используют, что они знают друг о друге и т.д. Информация такого рода делает любое описание слишком детальным, и есть риск, что больше энергии будет тратиться на объяснение внутреннего взаимодействия, чем на понимание бизнес-процессов. Конечно, информация такого рода очень важна при создании подробных моделей бизнес-систем, однако она не обязательна для понимания прецедентов.

Как указывают многие специалисты, выявлять объекты для модели не сложно, сложно находить правильные объекты. Рекомендуется для каждого прецедента определить объекты, необходимые для его выполнения. Если в объектную модель включать объекты, принимающие участие в различных прецедентах, то будет больше шансов определить действительно нужные объекты. Рассмотрев все прецеденты, в которых некоторый объект играет какие-либо роли, можно понять назначение этого объекта в системе.

4.2.3. Пример UML-модели бизнес-системы

В качестве примера анализируемой и моделируемой бизнес-системы рассмотрим систему «***Ресторан***», аналогично примеру, использованному в работе [97]. В целях конкретизации задачи предполагается, что анализ и моделирование осуществляются с целью проектирования информационной системы поддержки реинжиниринга подобного вида организаций, которая должна использовать модель существующего бизнеса. Диаграммы выполнены с помощью инструментального пакета Rational Rose.

На рисунке 4.15 представлена внешняя модель (П-модель) бизнес-системы «Ресторан», описывающая функции ресторана по отношению к его клиентам и поставщикам. Данная модель является UML-диаграммой прецедентов или вариантов использования, выполненной с учетом отношений расширения и использования между прецедентами.

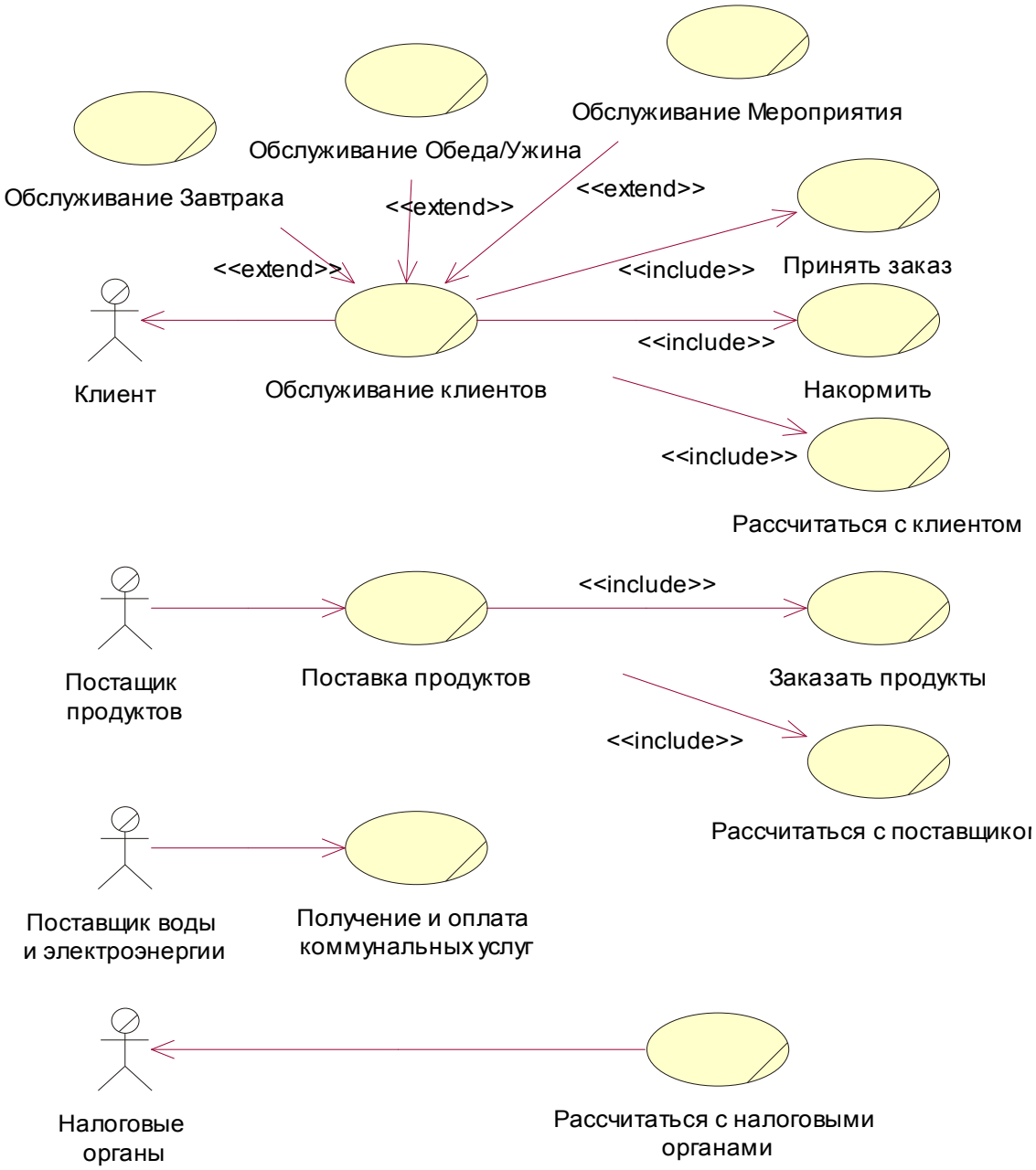


Рис. 4.15. - П-модель (внешняя модель) бизнес-системы «Ресторан».

На рисунках 4.16 и 4.17 представлена внутренняя модель (О-модель) бизнес-системы «Ресторан», описывающая типы объектов (классы) ресторана и взаимодействия между ними. Модель на рисунке 4.16 является UML-диаграммой классов для одного из вариантов использования бизнес-системы «Ресторан» (для прецедента «Обслуживание обеда/ужина»). Модель на рисунке 4.17 является UML-диаграммой кооперации между экземплярами (объектами) классов, представленных на диаграмме 4.16, т.е. для того же прецедента «Обслуживание обеда/ужина».

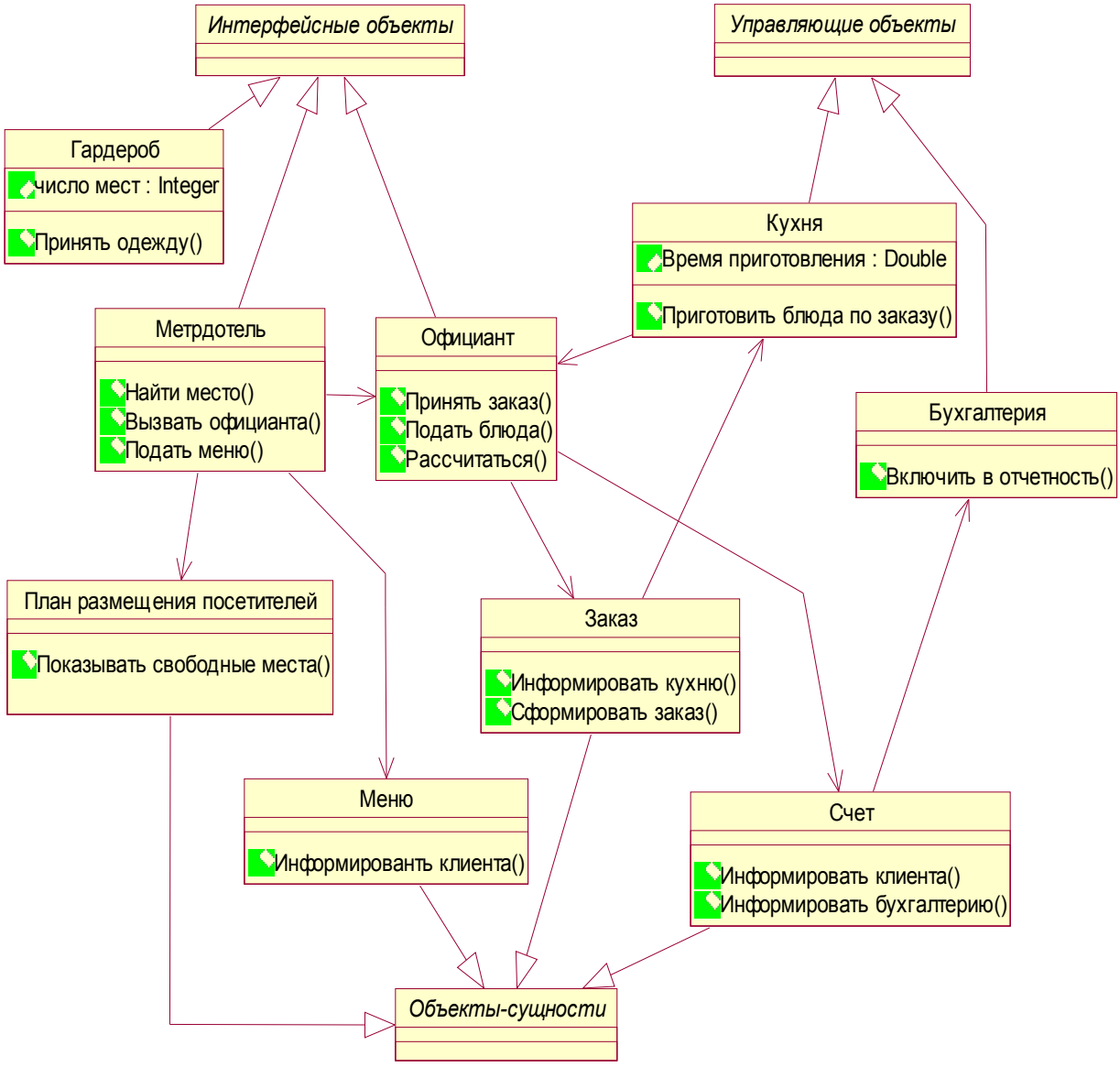


Рис. 4.16. - О-модель (внутренняя модель) бизнес-системы «Ресторан». Диаграмма классов для прецедента «Обслуживание обеда/ужина».

4.2.4. Пример модели информационного обеспечения бизнеса

Для обеспечения более глубокого понимания процесса объектного моделирования рассмотрим модель компьютеризированной системы организации товарооборота и обработки платежей, используемой в современных магазинах. Данная система, именуемая обычно терминальной системой розничной торгов-

ли, представляет собой устройство для считывания штрих-кода, подключенное к компьютеру, на котором функционирует программное обеспечение решающие задачи оформления продажи, денежных расчетов, связи с базой данных по товарам и т.д. Рассматриваемый пример с некоторыми исправлениями и уточнениями соответствует примеру, приведенному в работе [84]. Диаграммы выполнены с помощью инструментального пакета Rational Rose.

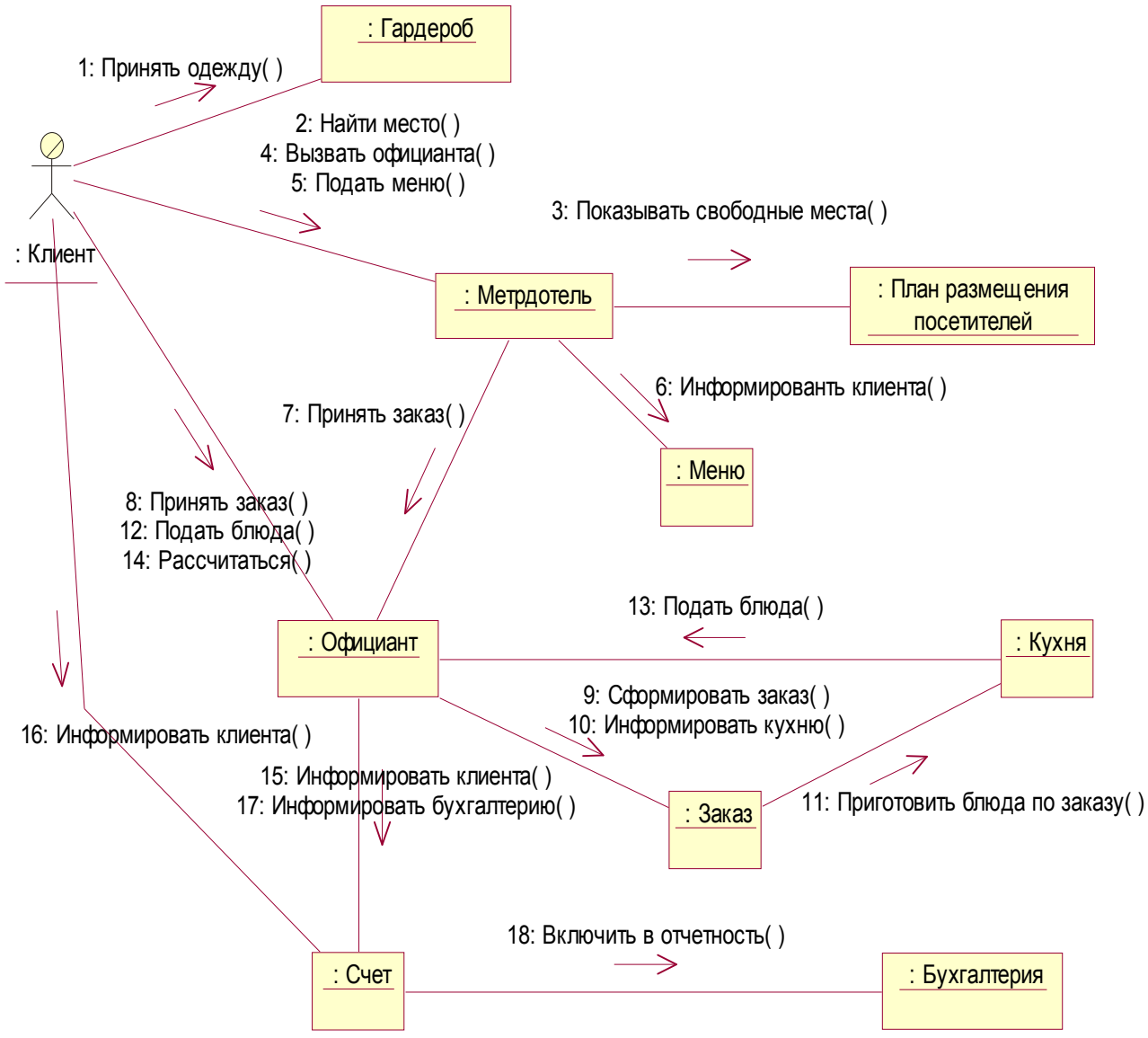


Рис. 4.17. - О-модель (внутренняя модель) бизнес-системы «Ресторан». Диаграмма кооперации объектов для прецедента «Обслуживание обеда/ужина».

На рисунке 4.18 представлена диаграмма прецедентов (вариантов использования) терминальной системы розничной торговли. Она показывает функции системы и взаимодействующих с ней людей (акторов), потребности (требования) которых выражены с помощью указанных прецедентов.

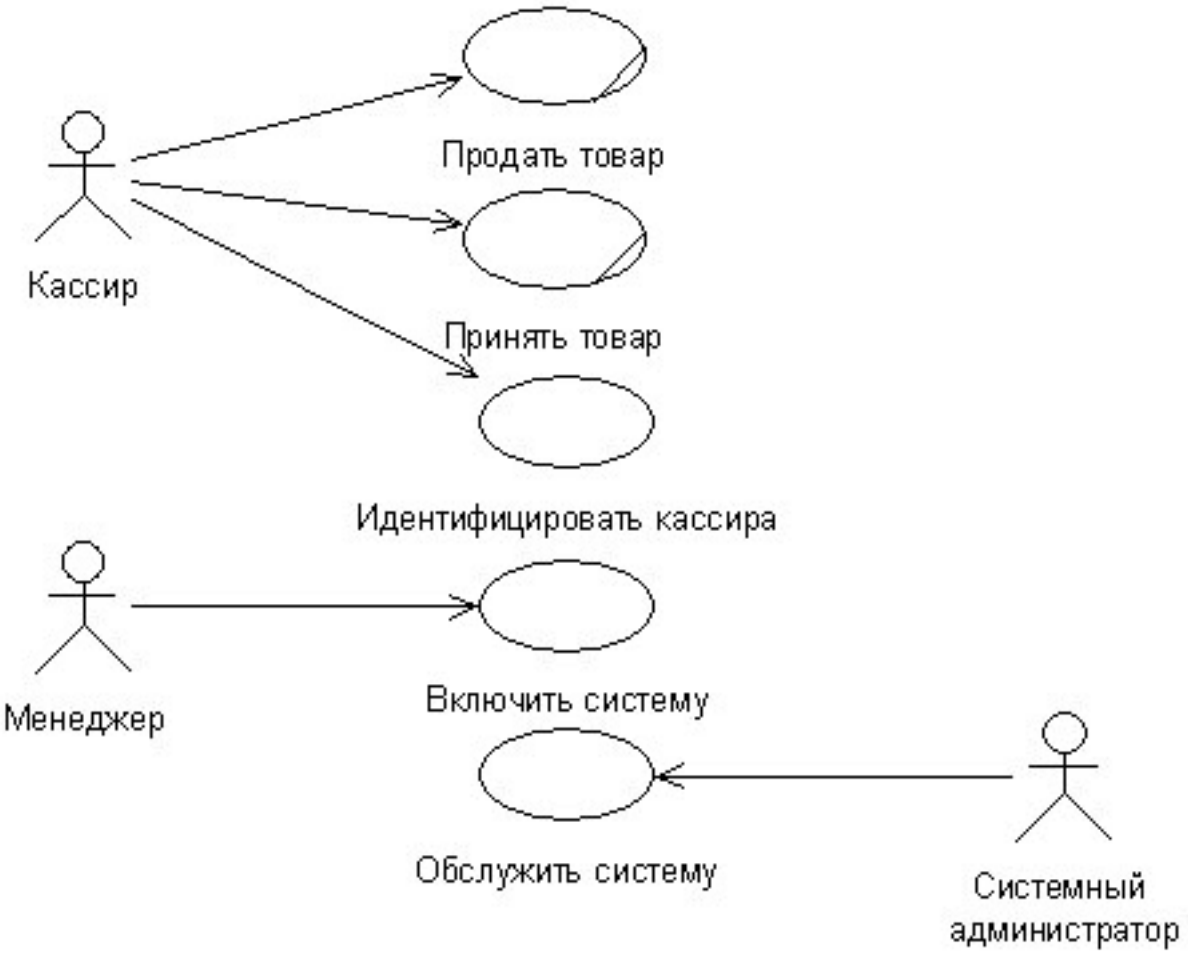


Рис. 4.18. - Диаграмма прецедентов терминальной системы розничной торговли.

В отличии от примера в работе [84], покупатели не рассматриваются в качестве акторов для моделируемой системы по той причине, что они фактически не взаимодействуют с данной компьютерной системой.

Ошибочного назначения акторов легко избежать, если построение диаграммы прецедентов начинать с выявления реально взаимодействующих с моделируемой системой людей или других систем (контекстных сущностей). При этом под взаимодействием следует понимать взаимодействие, которое может быть представлено в виде потока материальных или информационных элементов. Данное требование обусловлено тем, что только такое взаимодействие предъявляет к моделируемой системе реальные конкретные требования, в реализации которых и состоит процесс проектирования системы.

На рисунках 4.19 и 4.20 представлены диаграммы деятельности (активности) для прецедентов «Продать товар» и «Принять товар» соответственно. Данные диаграммы уточняют поток событий конкретного прецедента. Это уточнение необходимо для выявления классов и построения объектной модели.

В общем случае описание потока событий в прецеденте (в диаграмме деятельности) должно показывать начало и завершение прецедента, обрабатываемые в рамках прецедента сущности, а также исключительные ситуации, которые помечаются ссылками на другие прецеденты, которые расширяют данный

прецедент или используются в нем. При этом разработка диаграммы прецедентов, уточняемой диаграммами деятельности (или текстовыми документами), должно осуществляться параллельно с разработкой диаграммы классов.

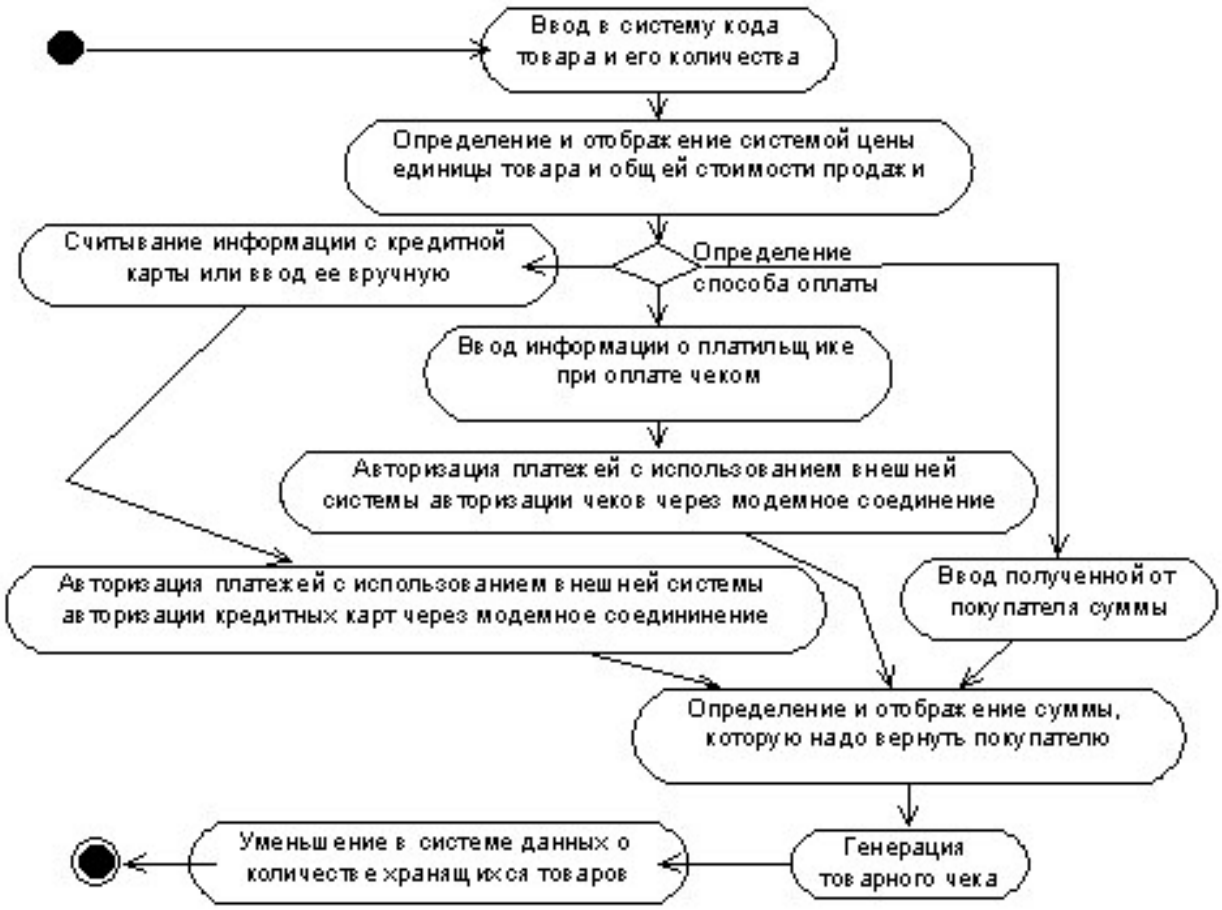


Рис. 4.19. - Диаграмма деятельности для прецедента «Продать товар».

Рис. 4.20. - Диаграмма деятельности для прецедента «Принять товар».

На рисунке 4.21 представлена диаграмма классов бизнес-процесса розничной торговли с использованием компьютерной терминальной системы. Эта диаграмма, в отличие от предыдущих, нуждается в некотором комментарии.

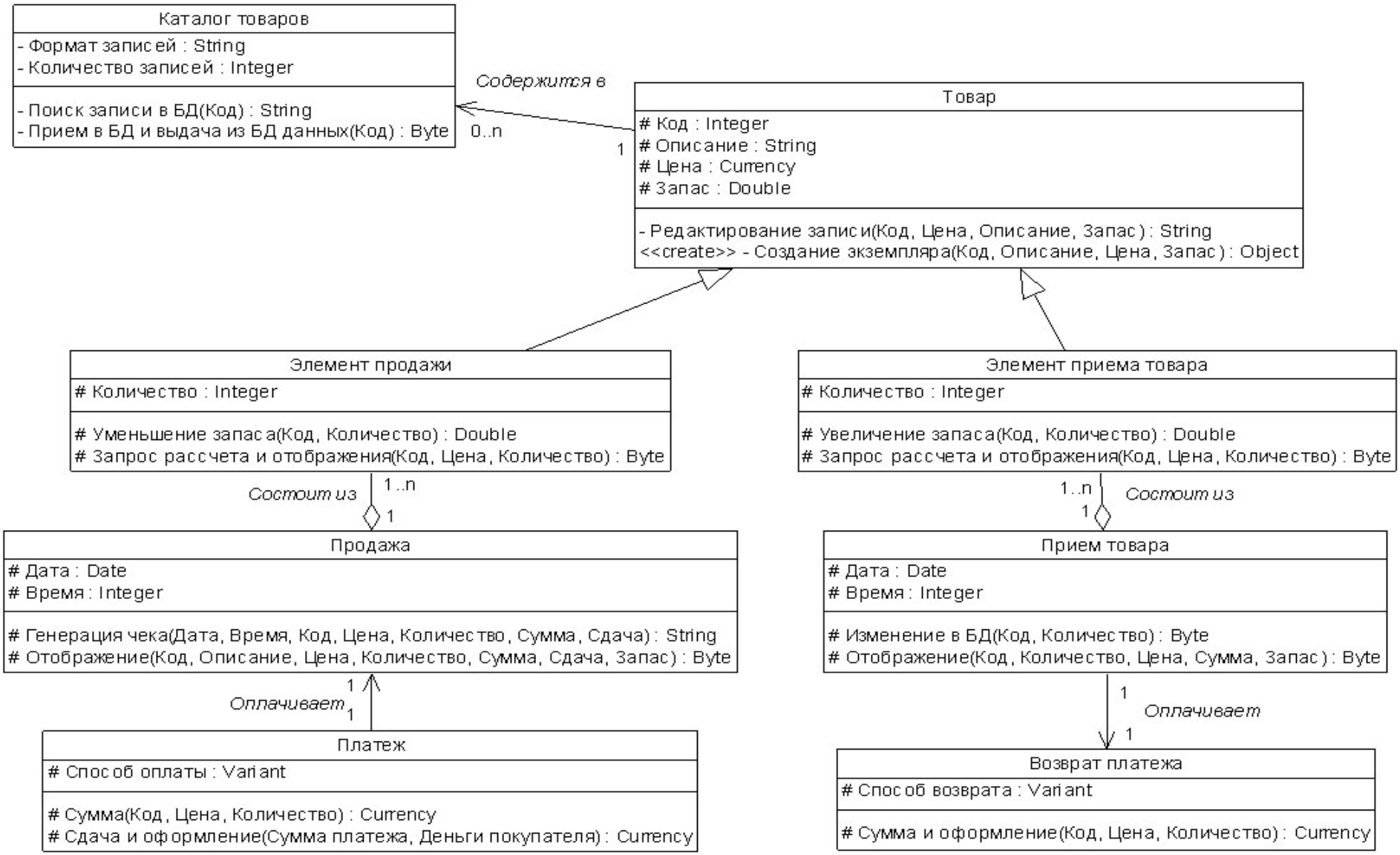


Рис. 4.21. - Диаграмма классов для бизнес-процесса розничной торговли с применение компьютерного терминала.

Целью построения данной диаграммы является выявление и документирование типов «сущностей», с помощью которых может быть описана моделируемая система. Если есть необходимость (для обеспечения, например, лучшего взаимопонимания с заказчиком), то номенклатура этих сущностей может быть расширена до описания фрагмента соответствующей предметной области. Поэтому диаграмма классов представляет собой концептуальную модель предметной области и не является моделью собственно проектируемой системы. В связи с этим в виде диаграммы классов, как правило, в первую очередь, описываются сущности (абстракции), имеющие непосредственное отношение к предметной области (в данном случае розничной торговли), т.е. к бизнес-логике.

Отсутствие в арсенале ООА каких-либо конструктивных методов выявления абстракций, необходимых для решения задачи, вынуждает аналитиков и проектировщиков использовать опыт, интуицию и субъективные мнения о предметной области. Это и зафиксировано в представленной на рис. 64 диаграмме. Однако, по мнению автора, в ней представлены все сущности, функциональные свойства которых могут обеспечить реализацию прецедентов «Продать товар» и «Принять товар». Названные функциональные свойства представлены в виде операций классов. Они позволяют экземплярам этих классов (объектам) выполнять функции в соответствии с диаграммой деятельности самостоятельно, а также путем передачи друг другу сообщений (параметров) для вызова (инициализации) необходимых функций.

Данная диаграмма является воплощением специфики объектного подхода к моделированию, так как именно она представляет в распоряжение аналитиков и разработчиков естественное средство описания свойств составных частей будущей информационной (или организационной) системы – концептуальную классификационную модель сущностей предметной области и самой системы. Именно на основе информации, заложенной в данной диаграмме, осуществляются анализ и моделирование проектируемой (разрабатываемой) системы.

На рисунке 4.22мичы с представлена диаграмма кооперации, реализующая бизнес-процесс (прецедент) «Продать товар». Данная диаграмма уже является собственно моделью проектируемой системы, так как описывает функциональное взаимодействие объектов (экземпляров классов), обеспечивающее получение результатов, требуемых в рамках данного прецедента. Как видно из представленной диаграммы объекты, взаимодействуя между собой, обеспечивают реализацию потока событий в прецеденте «Продать товар». Это достигается путем сообщения (посылки) экземпляром Кассира необходимых параметров (Код товара, Количество товара, Деньги покупателя) объектам системы. Эти сообщения вызывают у объектов ***проводник***: ***Элемент продажи***, ***счетчик*** функционирование соответствующих методов. Все объекты, кроме того, умеют вызывать методы других объектов, в соответствии со свойствами, описанными в диаграмме классов. Эти вызовы в соответствии с проставленными номерами обеспечивают выполнение необходимых функций.

При этом необходимо иметь в виду, что нас, в первую очередь, интересует модель бизнес-логики этого прецедента, а не реализация программной сис-

темы. Это позволяет опустить некоторые дополнительные подробности языка моделирования (например: роли ассоциаций, идентификаторы объектов), используемые при детальном проектировании собственно программных систем.

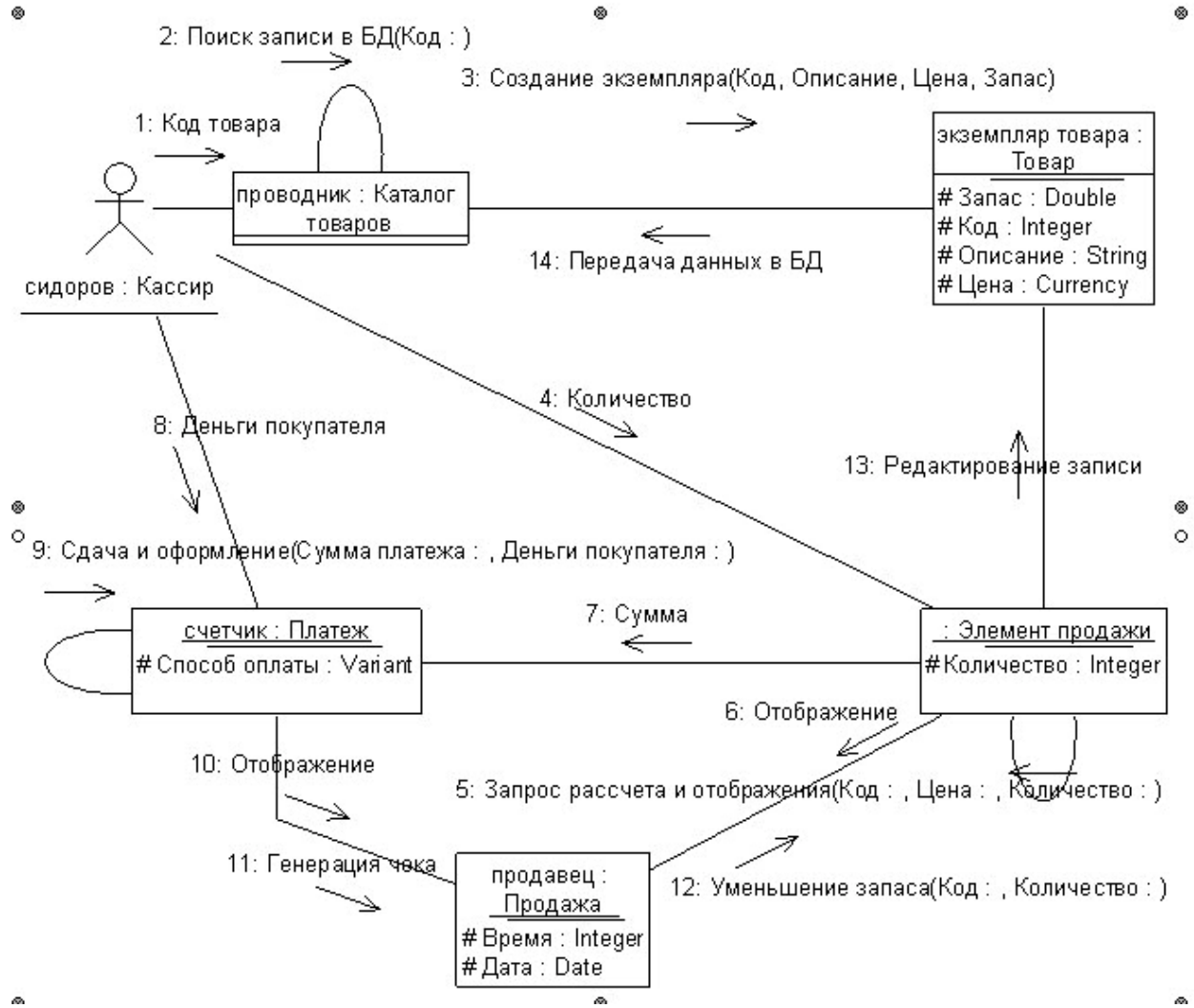


Рис. 4.22. - Диаграмма кооперации для прецедента «Продать товар».

Диаграмма кооперации (рис. 4.22) может быть автоматически преобразована в Rational Rose в диаграмму последовательности. Результата такого преобразования представлен на рисунках 4.23 – 4.24.

Таким образом, объектно-ориентированная методология, использующая стандартный язык UML, предоставляет полезные средства анализа и моделирования как информационных, так и организационных систем. При этом используется три вида моделей, аналогичных по своему предназначению трем моделям технологии 3VM или стандартам IDEF [84, 105]:

– ***Модель состояния***, которая представляет статический взгляд на структуру и состав элементов системы. Основными средствами визуализации этой модели являются диаграммы классов и объектов.

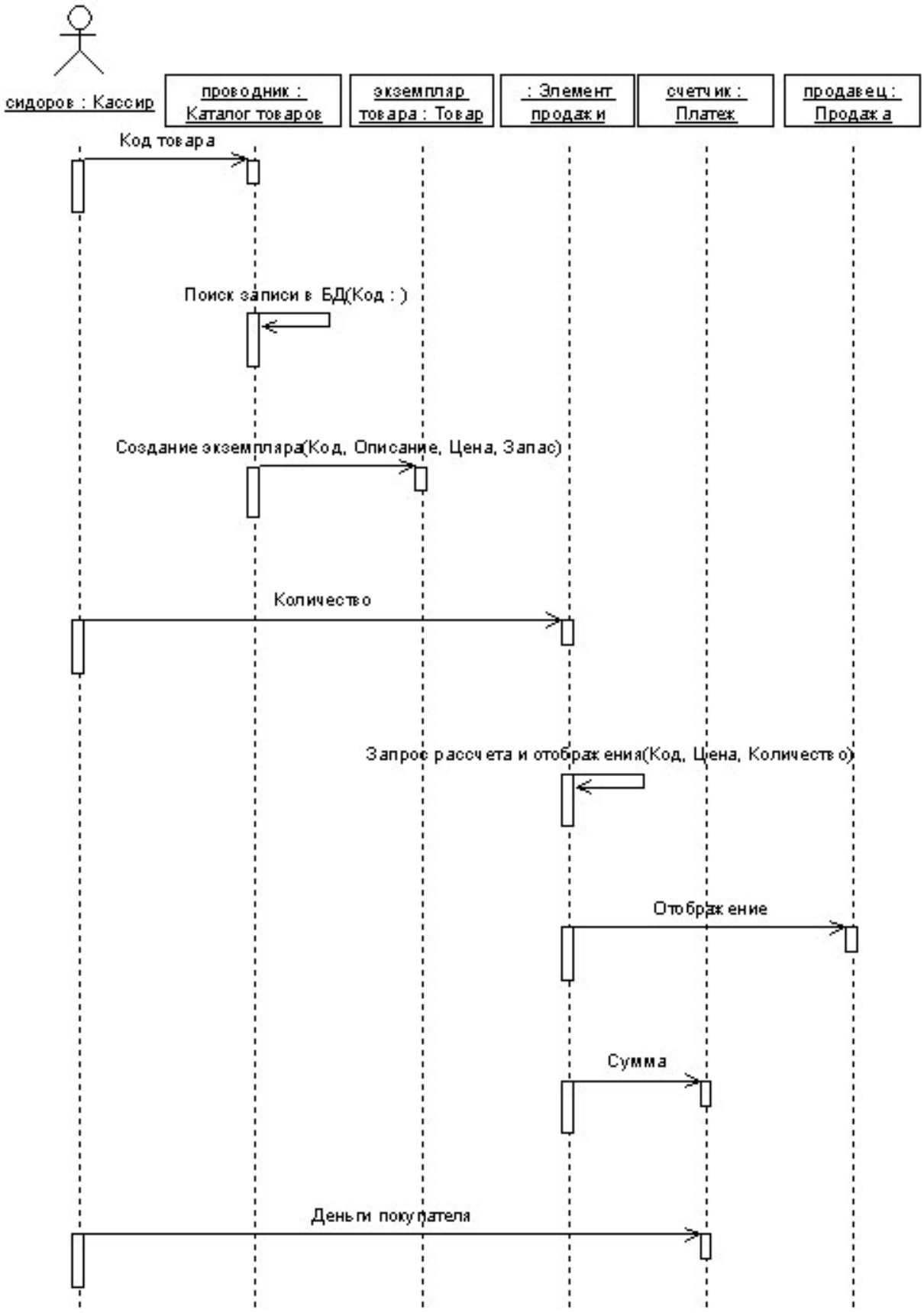


Рис. 4.23. - Диаграмма последовательности «Продать товар» (начало).

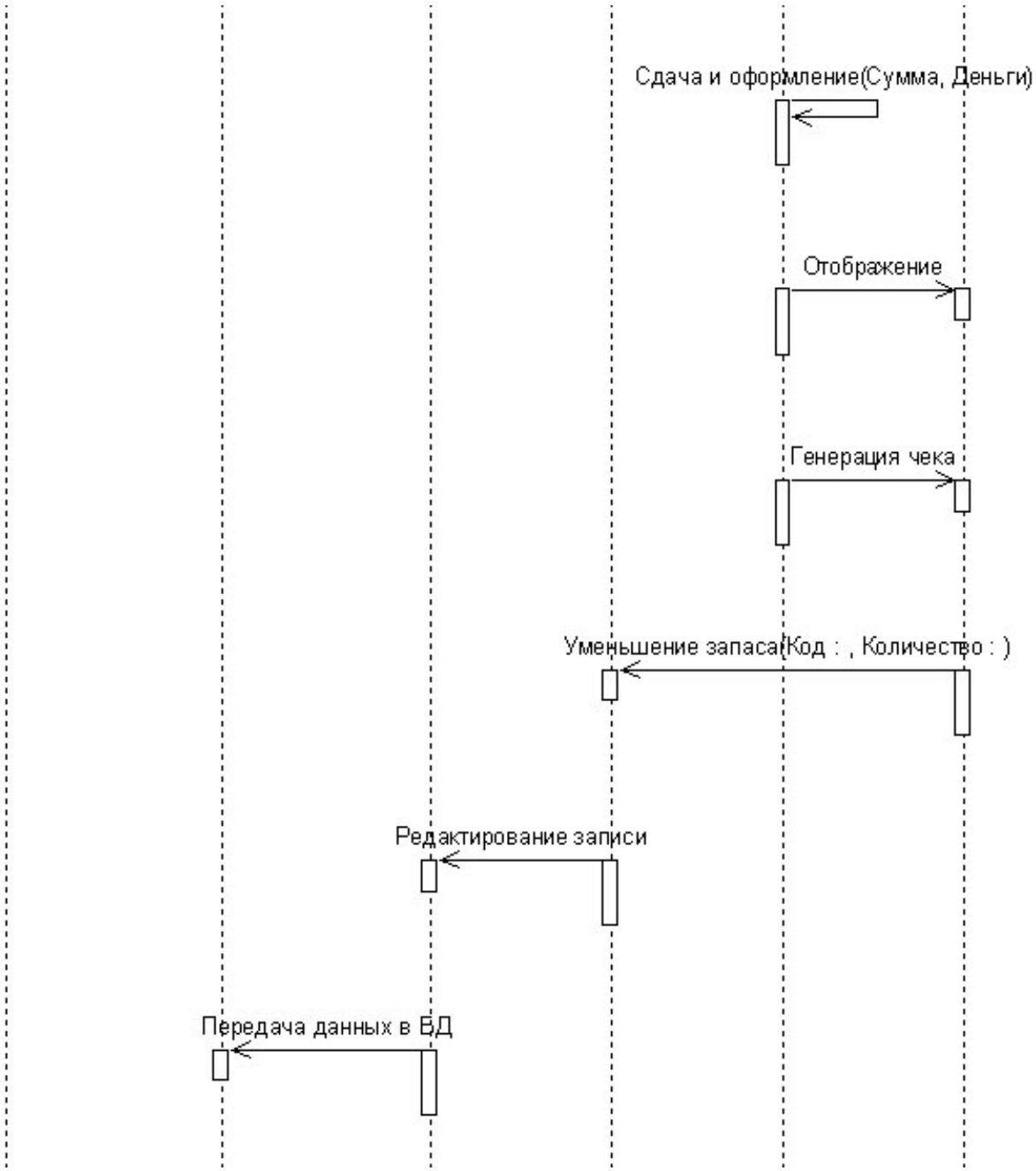


Рис. 4.24. - Диаграмма последовательности «Продать товар» (окончание).

- ***Модель поведения***, которая представляет операционный взгляд на систему и показывает действия осуществляемые ее элементами. Основными средствами визуализации этой модели являются диаграммы прецедентов, деятельности, взаимодействия.
- ***Модель изменения состояния***, которая представляет динамический взгляд на систему и показывает эволюцию элементов во времени. Основными средствами визуализации этой модели являются диаграмма состояний.

В отличие от функциональной декомпозиции системных структурных методов при данном подходе используется объектная декомпозиция, учитываю-

щая в определенной степени и поведение объектов. Однако, в основном, оно ориентировано на поведение элементов программных систем, что ограничивает возможности применения этой методологии для моделирования бизнес-процессов. Кроме того, как и в методах системного структурного анализа, в рамках объектно-ориентированного анализа отсутствуют реальные возможности имитации функционирования моделируемой системы.

Выводы

1. При разработке модели бизнеса как в ходе прямого, так и в ходе обратного инжиниринга, согласно рассматриваемой методологии, рекомендуется создавать две модели организации: внешнюю (П-модель) и внутреннюю (О-модель).
2. П-модель – средство как для описания требований к бизнесу, так и для представления наглядной картины того, что бизнес выполняет. Тем не менее, П-модель не дает ясной картины о внутреннем устройстве бизнеса. Она ничего не говорит о том, как различные виды деятельности реализуют бизнес-процессы.
3. О-модель описывает, как строится каждый бизнес-процесс предприятия из различных рабочих задач (внутренних процессов) и какие ресурсы он использует.
4. В бизнес-модели необходимо отображать: процессы, т.е. структурированный, измеряемый набор действий, осуществляемый для того, чтобы произвести определенный выход для конкретного клиента на рынке; ресурсы, т.е. как внутренний процесс реализуется при помощи человеческих или технических ресурсов и откуда эти ресурсы будут взяты; продукцию, которая должна иметь характеристики, описывающие, что с ней можно делать; потоки событий (входные данные из внешнего мира; действия (операции) процесса, которые он производит над исходными данными; потребляемые ресурсы; решения, которые будут приняты, и выход (результат), который процесс отправит во внешний мир.

4.3. CASE-инструментарий объектного моделирования и анализа

Существует несколько CASE-средств, поддерживающих язык UML. Наиболее известным является ***Rational Rose***, которое мы рассмотрим, используя, в первую очередь, работу [82], а также [22, 86, 87].

4.3.1. Назначение и возможности «IBM Rational Software Architect»

Rational Rose – CASE-средство фирмы Rational Software Corporation (США) – предназначено для автоматизации этапов анализа и проектирования программного обеспечения (ПО), а также для генерации кодов на различных языках и выпуска проектной документации. Rational Rose использует методологию объектно-ориентированного анализа и проектирования. Конкретный ва-

риант Rational Rose определяется языком, на котором генерируются коды программ (C++, Smalltalk, PowerBuilder, Ada, SQLWindows, ObjectPro). Кроме того, Rational Rose содержит средства реинжиниринга программ, обеспечивающие повторное использование программных компонент в новых проектах.

В составе Rational Rose можно выделить 6 основных структурных компонент: ***репозитарий***, ***графический интерфейс пользователя***, ***средства просмотра проекта*** (browser), ***средства контроля проекта, средства сбора статистики*** и ***генератор документов***. К ним добавляются ***генератор кодов*** (индивидуальный для каждого языка) и ***анализатор*** для C++, обеспечивающий реинжиниринг – восстановление модели проекта по исходным текстам программ.

Репозитарий представляет собой объектно-ориентированную базу данных. ***Средства просмотра*** обеспечивают «навигацию» по проекту, в том числе, перемещение по иерархиям классов и подсистем, переключение от одного вида диаграмм к другому и т.д. ***Средства контроля*** и ***сбора статистики*** дают возможность находить и устранять ошибки по мере развития проекта, а не после завершения его описания. ***Генератор отчетов*** формирует тексты выходных документов на основе содержащейся в репозитарии информации.

Средства автоматической генерации кодов программ на языке C++, используя информацию, содержащуюся в логической и физической моделях проекта, формируют файлы заголовков и файлы описаний классов и объектов. Создаваемый таким образом скелет программы может быть уточнен путем прямого программирования на языке C++. ***Анализатор кодов*** C++ реализован в виде отдельного программного модуля. Его назначение состоит в том, чтобы создавать модули проектов в формате Rational Rose на основе информации, содержащейся в определяемых пользователем исходных текстах на C++. В процессе работы анализатор осуществляет контроль правильности исходных текстов и диагностику ошибок. Модель, полученная в результате его работы, может целиком или фрагментарно использоваться в различных проектах. Анализатор обладает широкими возможностями настройки по входу и выходу. Например, можно определить типы исходных файлов, базовый компилятор, задать, какая информация должна быть включена в формируемую модель и какие элементы выходной модели следует выводить на экран. Таким образом, Rational Rose/C++ обеспечивает возможность повторного использования программных компонент.

В основе работы Rational Rose лежит построение различного рода диаграмм и спецификаций, определяющих логическую и физическую структуры модели, ее статические и динамические аспекты.

Диаграммы классов показывают классы и их иерархии, отображая логику построения прикладной системы. Для больших систем обычно строится не одна, а несколько подобных диаграмм. Диаграмма классов определяет только статические аспекты, относящиеся к классам. Динамика их поведения представляется на диаграммах состояний. Диаграммы кооперации показывают существующие объекты и их взаимодействие в логическом проекте прикладной сис-

темы. Модульные диаграммы (диаграммы компонентов) определяют распределение классов и объектов в модулях, физически реализующих проект. Одна модульная диаграмма представляет всю или часть модульной архитектуры системы. Для решения задачи распределения аппаратных ресурсов в Rational Rose используются диаграммы размещения.

Не вся необходимая информация о проекте может быть наглядно отражена в диаграммах, поэтому Rational Rose содержит средства ввода спецификаций, которые дополняют набор диаграмм. Спецификации создаются для прецедентов, пакетов, классов, атрибутов, операций, подсистем, объектов, компонент, диаграмм и т.д.

Основные свойства Rational Rose, обеспечивающие ее высокую конкурентоспособность на мировом рынке программных средств, следующие:

- охват всех этапов жизненного цикла работы над проектом с единой методикой и нотацией;
- возможность повторного использования программных разработок пользователей за счет средств реинжиниринга;
- наличие средств автоматического контроля, позволяющих вести отладку проекта по мере его разработки;
- возможность описания проекта на разных уровнях для различных категорий пользователей;
- удобный для пользователя графический интерфейс;
- автоматическая генерация кодов на языках C++, Ada, Smalltalk, PowerBuilder, SQLWindows, VisualBasic;
- наличие средств групповой разработки;
- широкий спектр применения системы – базы данных, банковские системы, телекоммуникация, системы реального времени и т. д.
- возможность использования различных платформ: IBM PC (в среде Windows), Sun SPARC Stations (Solaris, SunOS), Hewlett-Packard (HP UX), IBM RS/6000 (AIX).

4.3.2. Интерфейс «IBM Rational Software Architect»

Интерфейса Rational Rose имеет пять основных элементов [82]:

- Браузер (browser), который используется для навигации по модели.
- Окно документации (documentation window), которое применяется для работы с документацией элементов модели.
- Панели инструментов (toolbars), которые применяются для быстрого доступа к наиболее распространенным командам.
- Окно диаграммы (diagram window), которое используется для просмотра и редактирования одной или нескольких диаграмм UML.
- Журнал (log), который применяется для просмотра ошибок и отчетов о результатах выполнения различных команд.

Браузер – это иерархическая структура, позволяющая легко осуществлять навигацию по модели. С помощью браузера можно [82]:

- добавлять к модели элементы (действующие лица, варианты использования, классы, компоненты, диаграммы и т.д.);
- просматривать существующие элементы модели;
- просматривать существующие отношения между элементами модели;
- перемещать элементы модели;
- переименовывать эти элементы;
- добавлять элементы модели к диаграмме;
- связывать элемент с файлом или адресом Интернет;
- группировать элементы в пакеты;
- работать с детализированной спецификацией элемента;
- открывать диаграмму.

Браузер поддерживает четыре представления (view): ***представление вариантов использования***, ***логическое представление***, ***представление компонентов***, ***представление размещения*** (см. ниже).

Браузер организован в древовидном стиле. Каждый элемент модели может содержать другие элементы, находящиеся ниже его в иерархии. Знак «–» около элемента означает, что его ветвь полностью раскрыта. Знак «+» – что его ветвь свернута. По умолчанию браузер появляется в левой верхней части экрана. Затем его можно перетащить в любое другое место, закрепить там или оставить плавать свободно, а также, вообще, скрыть его.

С помощью окна документации можно документировать элементы модели. Например, можно сделать короткое описание каждого действующего лица. При документировании класса все, что будет написано в окне документации, появится затем, как комментарий в сгенерированном коде, что избавляет от необходимости впоследствии вносить эти комментарии вручную. Документация будет выводиться также в отчетах, создаваемых в среде Rational Rose.

Если в браузере или на диаграмме выбрать другой элемент, окно документации автоматически обновится, показав то, что соответствует этому элементу. Как и браузер, окно документации можно закрепить или оставить свободно перемещаться. По умолчанию оно появляется в нижнем левом углу окна Rational Rose, но может быть передвинуто оттуда или скрыто.

Панели инструментов Rational Rose обеспечивают быстрый доступ к наиболее распространенным командам. В этой среде существует два типа панелей инструментов: стандартная панель и панель диаграммы. Стандартная панель видна всегда, ее кнопки соответствуют командам, которые могут использоваться для работы с любой диаграммой. Панель диаграммы своя для каждого типа диаграмм UML. Все панели могут быть настроены пользователем.

В окне диаграммы видно, как выглядит одна или несколько UML-диаграмм модели. При внесении в элементы диаграммы изменений Rational Rose автоматически обновит браузер. Аналогично, при внесении изменений в элемент с помощью браузера Rational Rose обновит соответствующие диаграм-

мы. Это помогает поддерживать модель в непротиворечивом состоянии.

По мере работы над моделью определенная информация будет направляться в окно журнала. Например, туда помещаются сообщения об ошибках, возникающих при генерации кода. Не существует способа закрыть журнал совсем, но его окно может быть минимизировано.

4.3.3. Представление модели в «IBM Rational Software Architect»: представление вариантов использования; логическое представление; представление компонент; представление размещения

В модели Rational Rose, как уже было отмечено, поддерживается четыре представления (views) – представление вариантов использования (прецедентов), логическое представление, представление компонентов и представление размещения. Каждое из них предназначено для своих целей и для соответствующей аудитории.

Представление вариантов использования

Представление вариантов использования (прецедентов) содержит [82]:

- Действующих лиц.
- Варианты использования.
- Диаграммы вариантов использования. Обычно у системы бывает несколько таких диаграмм, каждая из которых показывает подмножество действующих лиц и/или вариантов использования.
- Документацию по вариантам использования, детализирующую происходящие в них процессы (потоки событий), включая обработку ошибок.
- Диаграммы деятельности, а также диаграммы взаимодействия, отображающие объекты или классы, принимающие участие в одном потоке событий варианта использования. Для каждого варианта использования можно создать множество диаграмм взаимодействия. Это делается либо в представлении вариантов использования, либо в логическом представлении системы. Как правило, не зависящие от языка программирования и реализации, диаграммы взаимодействия создают в представлении вариантов использования. Обычно такие диаграммы показывают взаимодействия объектов, а не классов. Диаграммы взаимодействия, зависящие от языка, обычно находятся в логическом представлении системы. Они, как правило, отображают классы, а не объекты.
- Пакеты, являющиеся группами вариантов использования и/или действующих лиц. Пакеты представляют собой механизм языка UML, позволяющий группировать вместе сходные элементы. Как правило, в системе существует сравнительно мало вариантов использования и действующих лиц, так что образовывать из них пакеты не требуется. Тем не менее, этот инструмент всегда может помочь в организации представления вариантов использования.

В начале работы над проектом представление вариантов использования нужно для заказчиков, аналитиков и менеджеров проекта. Работая с вариантами использования, их диаграммами и документацией по ним, они смогут прийти к соглашению о том, как должна выглядеть система на высоком уровне. При этом данное представление рассматривает только то, что именно будет делать система. Обсуждение деталей ее реализации надо оставить на будущее. В процессе работы над проектом все члены команды могут ознакомиться с этим представлением, чтобы достичь понимания системы на высоком уровне.

Документация варианта использования описывает соответствующий поток событий. С помощью этой информации специалисты по контролю качества смогут начать сразу писать тестовые программы для системы, а технические писатели – документацию для пользователей. Аналитики и заказчики будут уверены, что учтены все требования. Разработчики поймут, какие высокоуровневые элементы системы предстоит создать, и как будет работать ее логика.

Согласовав варианты использования и действующих лиц, заказчики должны будут принять решение и по поводу области применения системы. После этого разработчики смогут перейти к ее логическому представлению, уделяющему больше внимания тому, как система будет реализовывать поведение, определяемое вариантами использования.

Логическое представление

Логическое представление, концентрируется на том, как система будет реализовывать поведение, описанное в вариантах использования (прецедентах). Оно дает подробную картину составных частей системы и описывает взаимодействие этих частей. С помощью логического представления разработчики смогут сконструировать детальный проект создаваемой системы.

Логическое представление содержит [82]:

- Классы.
- Диаграммы классов. Как правило, для описания системы используется несколько диаграмм классов, каждая из которых отображает некоторое подмножество всех классов системы.
- Диаграммы взаимодействия, применяемые для отображения классов, участвующих в одном потоке событий варианта использования. Как упоминалось ранее, диаграммы взаимодействия создаются в представлении вариантов использования или в логическом представлении. При этом в первом случае, как правило, на этих диаграммах изображают объекты, а во втором – классы системы.
- Диаграммы состояний.
- Пакеты, являющиеся группами взаимосвязанных классов. Объединять классы в пакеты не обязательно, но настоятельно рекомендуется. Типичная система может содержать сотню или более классов, и объединение их в пакеты помогает уменьшить сложность вашей модели. Для понимания общей картины системы достаточно просто взглянуть на ее пакеты. Чтобы изучить систему на более детальном уровне, приходится углубляться

в пакеты и исследовать классы, находящиеся внутри.

Классы, которые могут появляться на диаграммах взаимодействия в представлении вариантов использования, представляют собой ***классы анализа***. После определения всех классов анализа каждый из них может быть преобразован в ***класс проекта***. Класс проекта уже содержит специфические для данного языка детали. Например, можно представить себе класс анализа, отвечающий за обмен информацией с другой системой. На этапе анализа не важно, на каком языке он будет написан, внимание уделяется только его данным и поведению. Преобразуя его в класс проекта, однако, придется коснуться специфических для языка деталей. Допустим, например, что это будет класс Java. В таком случае для реализации того, что заложено в классе анализа, может понадобиться два класса Java. Это значит, что отсутствует строгое соответствие между классами того и другого типа. Классы проекта изображают на диаграммах взаимодействия логического представления системы.

В этом представлении основное внимание уделяется логической структуре системы. В нем изучаются данные и поведение системы, определяются ее составные части и исследуется взаимодействие между ними. Одной из ключевых особенностей здесь является возможность повторного использования. Тщательно соотнося данные и поведение классов, группируя классы вместе и исследуя взаимодействия между классами и пакетами, можно определить, какие из них допускают повторное использование. Завершая новые и новые проекты, можно постепенно увеличивать библиотеку повторно используемых классов и пакетов. В результате выполнение будущих проектов будет все более и более становиться процессом сборки целого из имеющихся составных частей, а не разработки этих частей «с чистого листа».

Почти все участники команды, создающей систему, получают пользу от изучения логического представления системы. Взглянув на классы и их диаграммы, аналитики смогут убедиться, что все бизнес-требования будут реализованы в коде. Изучая классы, пакеты и диаграммы классов, специалисты по контролю качества поймут, из каких элементов состоит система и какие нуждаются в тестировании. Из диаграмм состояний они поймут также, как должны вести себя конкретные классы. Менеджер проекта из тех же элементов представления сможет уяснить, хорошо ли структурирована система, а также получить оценку степени ее сложности.

Однако больше всего с этим представлением работают разработчики и архитекторы. Первые смогут понять, какие классы надо создавать, и какие данные и поведение должен иметь каждый класс. Для вторых важнее всего структура системы в целом. Их задача – добиться того, чтобы архитектура системы была стабильна, но в то же время гибка настолько, чтобы быть адаптируемой к возможным изменениям в требованиях, а также предусмотреть возможность повторного использования.

Определив классы системы на диаграммах, можно приступить к работе с представлением компонентов, имеющим дело с ее физической структурой.

Представление компонент

Представление компонент содержит информацию о библиотеках кода, исполняемых файлах, динамических библиотеках и других компонентах моделей. Компонент – это физический модуль кода. Представление компонентов системы позволяет увидеть связи между этими модулями. Представление компонент содержит [82]:

- Компоненты, являющиеся физическими модулями кода.
- Диаграммы компонентов.
- Пакеты, являющиеся группами связанных компонентов. Как и в случае классов, повторное использование является одним из мотивов объединения компонентов в пакеты. Группу связанных компонентов системы легко можно использовать в других приложениях при условии, что связи между этой группой и другими тщательно отслеживаются.

Представление компонент более всего используется теми участниками проекта, кто отвечает за управление кодированием, компиляцию и размещение приложения. Часть компонентов – это библиотеки кода. Остальные – это динамические компоненты, такие, как исполняемые файлы или файлы динамических библиотек (.DLL). Разработчики с помощью этого представления смогут понять, какие библиотеки кода были созданы, и какие классы содержатся в каждой из них.

Представление размещения

Последнее представление Rational Rose – это представление размещения. Оно соответствует физическому размещению системы, которое может отличаться от ее логической архитектуры. Например, система может иметь логическую трехуровневую архитектуру. Иными словами, интерфейс может быть логически отделен от бизнес-логики, а она, в свою очередь, от базы данных. Однако размещение системы может быть и двухуровневым. Интерфейс может находиться на одном компьютере, а остальные две части – на другом.

Представление размещения отражает также и такие проблемы, как отказоустойчивость системы, ширина полосы пропускания по сети, восстановление после сбоев и время отклика. В представление размещения входят [82]:

- Процессы, являющиеся потоками (threads), исполняемыми в отведенной для них области памяти.
- Процессоры, включающие компьютеры, способные обрабатывать данные. Любой процесс выполняется на одном или нескольких процессорах.
- Устройства, то есть любая аппаратура, не способная обрабатывать данные. К числу таких устройств относятся, например, терминалы ввода-вывода и принтеры.
- Диаграмма размещения, на которой обозначены процессоры и устройства сети, а также физические соединения между ними. Кроме того, на диаграмме размещения изображают еще и процессы, и обозначают, какие процессы выполняются на каких компьютерах.

Как и в предыдущем случае, из представления размещения может извлечь пользу вся работающая над проектом команда, так как оно позволяет понять, как будет физически размещена система. Основными ее пользователями, однако, будут участники работы, отвечающие за распределение приложения.

4.3.4. Недостатки инструментария объектного моделирования

Рассмотрим недостатки пакета Rational Rose в соответствии в данными, приведенными в работе [37].

Средство моделирования Rational Rose в последнее время получает все большее распространение для целей анализа деятельности предприятий. Подтверждение этому факту легко найти в Internet, проанализировав требования, которые формулируют различные компании к кандидатам на разработку информационных систем. В большинстве случаев в состав требований обязательно включается знание Rational Rose и языка моделирования UML.

Кроме того, часто приходится слышать и читать, что UML и Rational Rose являются универсальными средствами, которые вполне подходят и для моделирования бизнес-процессов. Так, на сайте компании «Интерфейс Ltd» (партнера фирмы Rational Software) приводятся следующие слова вице-президента Rational Роджера Оберга: «Rational Rose стала стандартом при разработке приложений и бизнес-моделировании». Там же, в свое время, было опубликовано следующее сообщение: «Корпорация Rational Software объявила о выходе Rational Rose 2000e – новой версии CASE-средства визуального проектирования информационных систем, позволяющего моделировать как компоненты программного обеспечения, так и бизнес-процессы». Там же опубликована статья А. Новичкова «Эффективная разработка программного обеспечения с использованием технологий и инструментов компании Rational», в конце которой приводится рекомендация: «Есть смысл приобретать AnalystStudio для проведения бизнес-моделирования. Для данных целей набор содержит все необходимое». На сайте другого партнера фирмы Rational Software, компании «АйТи», утверждается: «Rational Rose 2000 предназначено для создания сложных коммерческих приложений и корпоративных информационных систем и ориентировано на аналитиков, разработчиков архитектуры и программистов».

При этом AnalystStudio – это набор продуктов фирмы Rational Software, рекомендованный аналитикам и включающий в себя Rational Rose, как основной продукт, и Rational Unified Process, Rational Requisite PRO, Rational ClearQuest и Rational SoDA, как дополнительные.

По мнению автора работы [37], «предложение использовать Rational Rose в такой неоправданно широкой области – серьезное заблуждение. Во всяком случае, на рынке CASE-средств давно присутствуют и успешно используются инструменты, существенно лучше реализующие потребности аналитика при описании и анализе деятельности предприятия». При этом в качестве более пригодного для моделирования бизнеса инструмента называется пакет BPwin, а по поводу Rational Rose указываются следующие недостатки.

Rational Rose не поддерживает ни одну из известных методологий моделирования и анализа бизнес-процессов. Методика построения бизнес-моделей, содержащаяся в дополнительном наборе рекомендаций или методике RUP, которая сопровождает пакет Rational Rose, предлагает диаграммы вариантов использования или прецедентов (Use Case) и диаграммы деятельности (Activity) для описания бизнес-процессов. Однако эти диаграммы позволяют описать лишь малую часть сведений, которые нужны для моделирования бизнес-процессов и которые представляются, например, средствами IDEF0. Кроме того, дуги Use Case и Activity диаграмм не имеют тех смысловых типов, которые были указаны для дуг SADT.

По мнению автора работы [37], некие синтаксические соглашения, диктуемые системой при разработке Use Case и Activity-диаграмм, не объединены в законченную и понятную систему. Этим диаграммам (что, наверное, главное) не дается никакой интерпретации, объясняющей, как их применять при моделировании. Действительно, что означает, что два процесса соединены стрелкой – просто последовательность их исполнения или, например, то, что второй процесс обрабатывает некоторые (какие?) результаты деятельности первого. А может быть, наоборот, для работы первого процесса необходима некая (какая?) информация, которую подготавливает второй? Точно так же непонятно, как интерпретировать связи «процесс-состояние», «состояние-состояние» и др. Поэтому Rational Rose допускает построение синтаксически корректных Activity-диаграмм, не просто не имеющих смысла с точки зрения моделируемого объекта, но вообще не поддающихся объяснению с позиции здравого смысла [37].

По этим причинам пользователям Rational Rose при разработке Use Case и Activity-диаграмм приходится придумывать свои оригинальные синтаксические соглашения и давать свою интерпретацию имеющимся, чтобы отразить всю существенную для анализируемого процесса информацию. Например, чтобы имитировать три вида характерных для SADT входящих в процесс стрелок – вход, механизм, управление – можно каждую из них подкрашивать своим цветом, а для того, чтобы отличить входящие документы от исходящих, можно использовать пунктирные и сплошные стрелки. Другими словами, пользователь Rational Rose вынужден разрабатывать свои формализмы для получения методики построения моделей и анализа бизнес-процессов. При этом, возможно, придется не только разрабатывать свою методику, но и отклоняться от стандартов UML. Зачем это делать, если существует апробированная и признанная во всем мире IDEF0 (а также другие стандартные, средства и языки, например IDEF3), а преимущества стандартного подхода совершенно очевидны.

Даже если удастся придумать, как реализовать в Rational Rose соглашения IDEF0, или разработать свою методологию анализа бизнес-процессов, не уступающую IDEF0 и органично реализуемую в Rational Rose, то система все равно не научится «читать» разработанные модели. Дело в том, что это в ней не заложено изначально, а, следовательно, обработка и анализ моделей будут целиком на плечах аналитика. Или же придется разрабатывать свои процедуры выдачи отчетов, которые будут ориентированы на отсутствующие в стандарт-

ном UML (но имеющиеся в IDEF0) синтаксические соглашения. Но и это еще не все, поскольку не будет решена задача поддержки и контроля синтаксиса для разработанной пользователем методологии, и, следовательно, не будет никакой гарантии корректности разработанной модели.

Поэтому при необходимости проведения работ, где анализ бизнес-процессов играет важнейшую роль, выбор в качестве средства моделирования продуктов фирмы Rational Software является серьезной ошибкой.

Вопросы для повторения

1. В чем суть основных шагов по моделированию и анализу системы в терминах UML?
2. Что такое «прецедент»?
3. Какие существуют отношения между прецедентами?
4. Что такое «кооперация»? Что такое «взаимодействие»?
5. Какие существуют отношения между классами?
6. Что такое «сообщение»?
7. Как выглядит и для чего применяется диаграмма активности?
8. Что такое рациональный Унифицированный Процесс создания ПО? Объясните, почему РУП является итеративным и инкрементным процессом.
9. Что такое фазы РУП? Что такое рабочие процессы РУП?
10. Каковы основные цели в фазе «Исследование»?
11. Какие основные диаграммы используются в фазе «Исследование»?
12. Что такое модель анализа? Что такое модель проектирования?
13. Чем отличается объектно-ориентированный анализ от объектно-ориентированного проектирования?
14. В чем состоит суть дихотомии «интерфейс/реализация»?
15. Что такое каноническая форма представления системы?
16. Каким образом описываются структурные свойства и поведение объектов?
17. Каковы особенности объектной декомпозиции, отличающие ее от алгоритмической?
18. Чем отличаются друг от друга П-модель и О-модель организации?
19. Что такое субъект бизнес-системы и чем он отличается от актера?
20. Какие существуют CASE-средства, поддерживающие UML?

Резюме по теме

В данном разделе рассмотрена технология объектного моделирования и анализа сложных систем: сущности, отношения и диаграммы UML, а также процедура объектного моделирования; требования к объектному моделированию организационных систем и их информационного обеспечения; а также CASE-инструментарий объектного моделирования и анализа (IBM Rational Software Architect).

Тема 5. Технология системно-объектного моделирования и анализа

Цели и задачи изучения темы

Целью изучения данной темы является теоретическое и практическое освоение оригинальной технологий системно-объектного моделирования и анализа сложных систем.

При этом ставятся следующие задачи:

- изучение методологии системно-объектного моделирования, т.е. системологического подхода, с помощью которого система представляется в виде трехсторонней конструкции «Узел-Функция-Объект»;
- ознакомление с процедурой системно-объектного моделирования сложных систем, т.е. с алгоритмом построения УФО-моделей и средствами формализации системно-объектного УФО-анализа;
- изучение CASE-инструментария системно-объектного моделирования и анализа (UFO-toolkit).

5.1. Методология системно-объектного моделирования и анализа

5.1.1. Системологический подход «Узел-Функция-Объект»

Решение проблемы синтеза системного и объектно-ориентированного подходов позволяет создать метод системно-объектного анализа и моделирования. Этот метод обеспечивает интеграцию функциональной (процедурной) и объектной (субстанциальной) декомпозиции систем (например, организационной или информационной) с учетом ее взаимодействия со средой.

Рассмотрим, каким образом может быть решена данная проблема средствами ноосферного системного подхода – системологии [2, 5, 6, 27].

Для обеспечения единства системного и объектного подходов необходимо, в первую очередь, преодолеть различия этих подходов с точки зрения направленности процесса декомпозиции. Как было отмечено выше, системному подходу (анализу) свойственна так называемая ***процедурная*** (функциональная) декомпозиция системы, а объектному подходу – ***объектная***. При этом они рассматриваются как ортогональные всеми специалистами, как по системному анализу, так и по объектному подходу.

Рассмотрим вариант интеграции функциональной и объектной декомпозиции системы средствами представленного концептуального аппарата теории системологического анализа и моделирования (см. п. 2.3 настоящего пособия).

Решение поставленного вопроса следует начать с обсуждения понятия «структурная декомпозиция», которое часто используется специалистами аналитиками, вероятно, ввиду того, что некоторые методы и подходы системного анализа именуются «структурными технологиями» (например, SADT). Дело в том, что любой метод анализа и проектирования всегда будет структурным. Всегда в процессе анализа и моделирования выявляется и моделируется некоторая структура. Однако, структура (сама по себе) есть чистая абстракция, что

делает ее, конечно, удобным предметом формального исследования, особенно средствами классической математики, которая ни для чего более, собственно, и не предназначена. Но выявления структуры «самой по себе» недостаточно при проведении содержательного (читай системного) анализа сложных, например организационных, систем. Вот почему формализация процедур системного анализа средствами классической математики резко снижает выразительные возможности системных методов. Например, до сих пор так никто и не знает, как могут быть применены для практических нужд системного анализа известные и очень математические рассуждения по поводу систем (в теоретико-множественном их понимании) М. Месаровича, Д. Михайло и Я. Такахара, в книге которых и сами авторы не делают даже малейшего намека на такую возможность [8].

В действительности, структура всегда является структурой чего-либо и вариантов существует не так уж много: это или функции, или объекты. Таким образом, вопрос состоит только в том, о какой структуре идет речь. В традиционном системном анализе речь идет о функциональной структуре (структуре процессов) без внимания к реализующим эти функции объектам, т.е. субстанции системы. В ООА речь идет о структуре объектов (субстанции) системы, функциональность которых рассматривается только с точки зрения ответственности компонент ПО.

Настоящие системные (т.е. системологические) представления о реальной действительности позволяют объединить процессы выявления и функциональной, и объектной структуры. Согласно данным представлениям, как было отмечено выше (см. п. 2.3), во-первых, система всегда есть ***функциональный объект***, функционирование которого, с одной стороны, поддерживает надсистему, а, с другой стороны, само поддерживается функционированием подсистем. Во-вторых, свойства системы есть внутренние способности этой системы поддерживать связи некоторого вида и/или препятствовать осуществлению связей какого-либо вида, т.е. ***характеризуются связями с другими системами***. Любая же связь между системами есть ***процесс обмена*** между ними элементами определенных глубинных ярусов связанных систем. Таким образом, свойства системы представляют собой проявления ее активности включаться в связи, в ***обменные потоки*** с другими системами в структуре надсистемы [2, 5].

Следовательно, анализ средствами функциональной системологии свойств системы как целостного функционального объекта «основывается прежде всего на обнаружении тех потоков, в которые он включен как элемент надобъекта, т.е. как «проточный» элемент в сети замкнутых обменных потоков надобъекта. Естественно, что обнаружение этих качеств будет одновременно и достаточно полной характеристикой функций этого объекта, и выразителем его целостности, ибо в качественных характеристиках не может не проявиться в этом случае балансность втекающих и вытекающих потоков» [5, с. 43].

Описанные представления позволяют говорить о том, что структура, в действительности (при системном, конечно, ее рассмотрении) не может быть или только функциональной, или только объектной (субстанциальной). Это на

самом деле одна и та же структура, а их разделение есть результат произвола мышления аналитика, ограниченного рамками привычной методологии.

Во-первых, каждая система характеризуется определенными видами связей с другими системами. Если связи отсутствуют, то данную систему рассматривать вообще не имеет никакого смысла. При этом, с точки зрения других систем, любая конкретная система представляется перекрестком, т.е. ***узлом***, связей, по которым что-либо поступает к ней («втекает») от других и что-либо поступает от нее («вытекает») к другим. Таким образом, необходимо учитывать, что любая система обязательно является и потребителем каких-то видов ресурсов (материальных и информационных) других систем, и поставщиком каких-то видов ресурсов для других систем. Качественная узловая характеристика системы является основной и характеризует ее целостно как элемент (подсистему) системы более высокого яруса, т.е. надсистемы. Можно сказать, что узел входящих и выходящих связей характеризует миссию данной системы в структуре надсистемы, так как он определяет, что и от кого поступало к системе и что и кому поступало от нее, т.е. характеризует предназначение системы.

Во-вторых, с точки зрения втекающих и вытекающих потоков/связей, каждая система характеризуется функциональными способностями (процессами, ***функциями***), обеспечивающими преобразование «втекающих» по связям ресурсов в «вытекающие» ресурсы. Эти функциональные способности (процессы) обеспечивают баланс «притока» и «оттока» по функциональным связям узла, занимаемого данной системой. При этом баланс одного и того же узла может быть обеспечен, в принципе, разными наборами функциональных способностей (наборами процессов), т.е. разными функциональными зависимостями выхода от входа. Формальная функциональная характеристика системы является второстепенной и характеризует теоретическую способность (потенциальную возможность) системы сбалансировать определенный узел.

В-третьих, с точки зрения функциональных способностей балансировать определенный узел, каждая система характеризуется как материальный ***объект***, реализующий эти функциональные способности (функциональные зависимости), т.е. физически осуществляющий эти процессы. При этом один и тот же набор функциональных способностей может быть реализован, в принципе, различными по своей природе и конструкции объектами. Необходимо только, чтобы производительности этих объектов по входу и выходу соответствовали количественным характеристикам втекающих и вытекающих потоков объектов, связанных с данной системой. Количественная объектная характеристика системы является третьестепенной и характеризует практическую действительную ее способность сбалансировать определенный узел.

Данные рассуждения приводят к необходимости единовременного представления любой системы с трех точек зрения (см. рис. 5.1):

- как ***структурного элемента*** надсистемы в виде перекрестка связей с другими системами – ***узла*****,** который будем обозначать $\mathbf{(L^{in})L^{out}}$, где **L** – связь/поток;

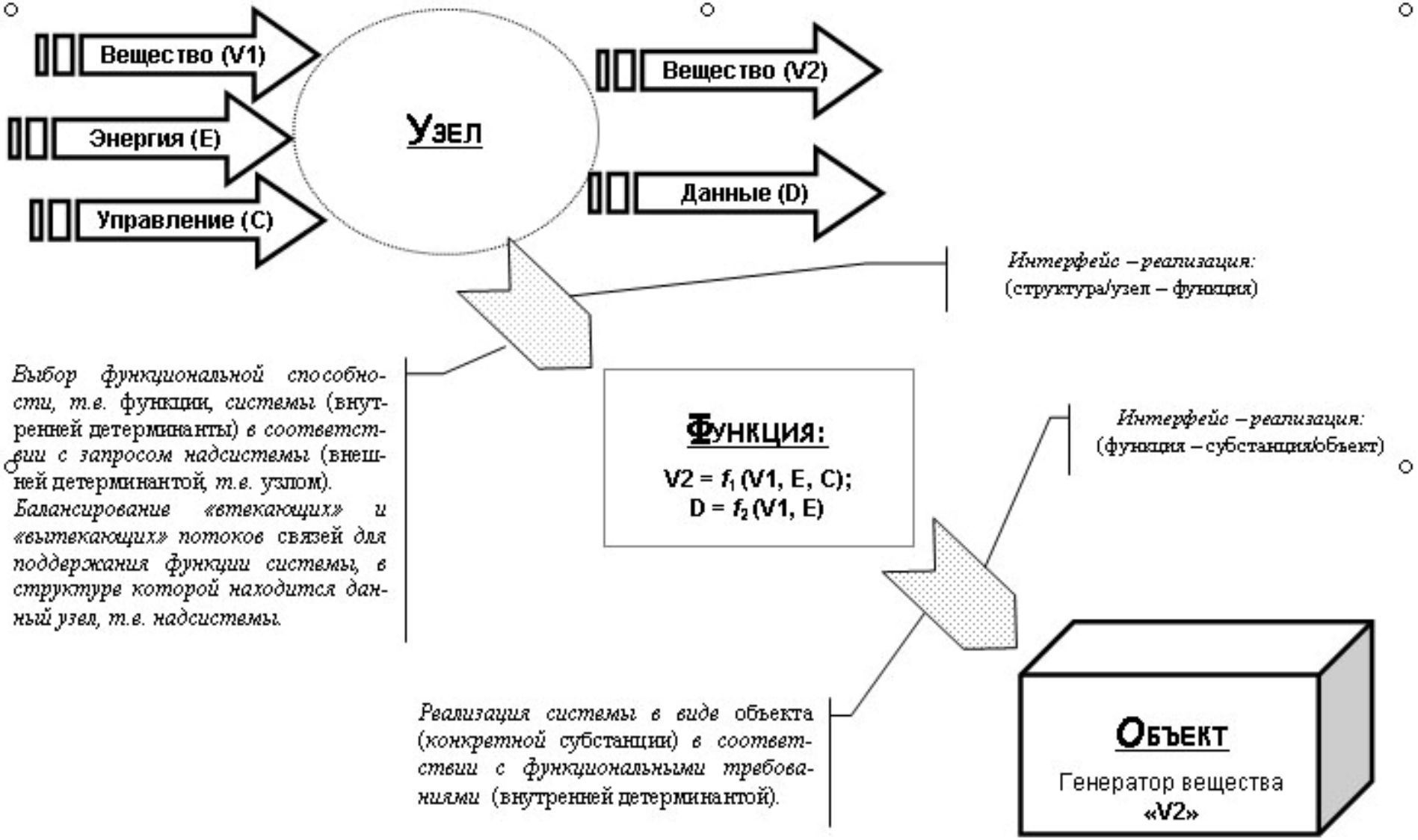


Рис. 5.1. - Принцип подхода «Узел – Функция – Объект».

– как ***функционального элемента***, выполняющего определенную роль с точки зрения поддержания надсистемы путем балансирования данного узла – ***функции***, которую будем обозначать $\mathbf{L^{out}(L^{in})}$;
– как ***субстанциального элемента***, реализующего данную функцию в виде некоторого материального образования, обладающего конструктивными, эксплуатационными и т.д. характеристиками – ***объекта,*** который будем обозначать $\mathbf{L^{out}L^{in}}$.

Следовательно, разбиение системы на подсистемы, представляющие собой трехэлементные конструкции «***Узел – Функция – Объект***» (***УФО-элементы***), обеспечит единство функциональной (т.е. той, которая раньше и считалась системной) и объектной декомпозиций, так как является наиболее адекватным реальной действительности способом представления структуры, состава и функциональности системы, с учетом ее взаимодействия со средой.

Подход «Узел – Функция – Объект» (***УФО-подход***) позволяет рассматривать любую систему или предметную область как совокупность взаимодействующих УФО-элементов (как ***УФО-конфигурацию***), так как любое явление действительности (см., например, таблицу 5.1) представляет собой структурную часть еще более целого (взаимодействует с другими явлениями); функционирует определенным образом и при этом является каким-то материальным образованием. Описанный подход позволяет обеспечить функциональное и объектное моделирование одновременно, т.е. в одной модели, в ходе системно-объектного ***УФО-анализа***.

Таблица 5.1. Аспекты системного подхода Узел-Функция-Объект.

Узел - $(L^{in})L^{out}$	*Интенция*	*Причина*	*Потребность*	*Мотив*	*Требования (Задание)*
Функция- $L^{out}(L^{in})$	*Потенция*	*Условие*	*Возможность*	*Алиби*	*Проектирование*
Объект- $L^{out}L^{in}$	*Экстенция*	*Следствие*	*Деятельность*	*Вещественные доказательства*	*Реализация*

5.1.2. Адаптивная нормативная система УФО-анализа

Основу любого метода традиционного системного (системно-структурного, структурно-функционального) или объектного анализа составляет некоторая нормативная система. Неотъемлемой частью нормативной системы (как и любой формальной системы) являются алфавит, включающий в себя знаки (символы), используемые для записи по определенным правилам выражений (в данном случае построения диаграмм), а также правила манипулирования этими знаками (в данном случае правила построения диаграмм).

В случае системно-объектного подхода «Узел-Функция-Объект» в качестве алфавита рассматривается набор **у**злов (перекрестков связей/потоков), набор **ф**ункций, балансирующих эти узлы, и набор **о**бъектов, реализующих эти функции. При этом для формирования набора узлов используется классификация, определяемая ***базовой таксономической*** (***родовидовой***) ***классификацией связей*** (рис. 5.2) в зависимости от «протекающих» по ним ресурсов.

Классификация связей обеспечивает ***параметричность*** классификации узлов и ***конструктивное определение семантики*** символов этих узлов. Пример классификации УФО-элементов по их узлам (для бинарного случая) на уровне базовой классификации связей представлен на рисунке 5.3.

Естественно, классификации связей **L**, узлов **(L)L** (а также изоморфные последней классификации функций **L(L)** и объектов **L**L, т.е. УФО-элементов) могут быть специализированы с любой степенью точности для любой конкретной предметной области.

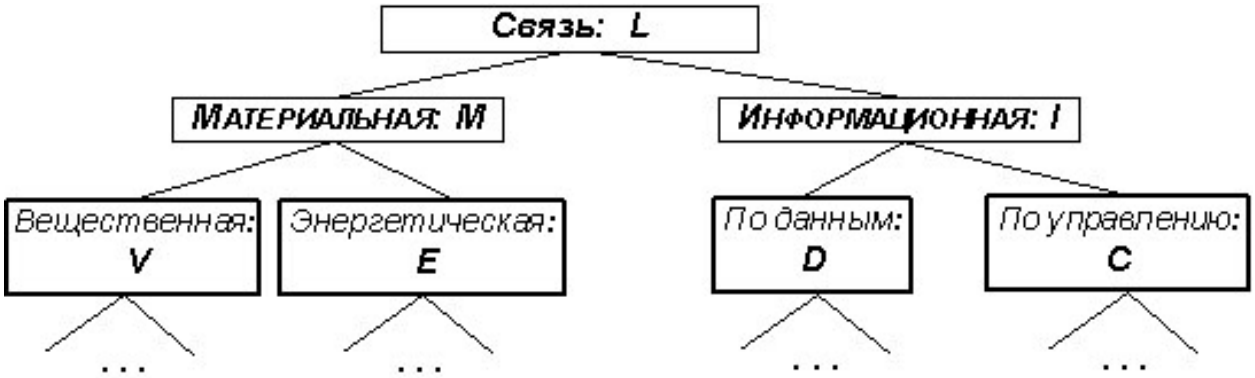


Рис. 5.2. - Базовая классификация связей.

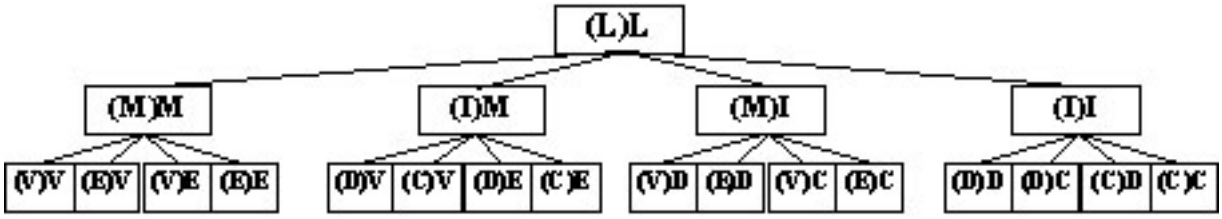


Рис. 5.3. - Классификация алфавитных УФО-элементов по их узлам.

В качестве алфавитных элементов могут рассматриваться любые УФО-элементы, получаемые путем комбинирования связей из базовой классификации либо из классификации, представляющей собой результат ее специализации. При этом естественным образом возникают следующие варианты.

1. Система (УФО-элемент) представляет собой объект $\mathbf{L^{out}L^{in}}$, занимающий узел $\mathbf{(L^{in})L^{out}}$ с одним входом и одним выходом и реализующий функцию $\mathbf{L^{out}(L^{in})}$ преобразования одного переменного. Данный УФО-элемент называется «элементарным» или «алфавитным». Его элементарность не означает невозможности его дальнейшей декомпозиции.

2. Система (УФО-элемент) представляет собой объект $\mathbf{L^{out}L^{i}}$ (i = 1, …, n), занимающий узел $\mathbf{(L^{i}, …, L^{n})L^{out}}$ с несколькими входами и одним выходом и реализующий функцию $\mathbf{L^{out}(L^{i}, …, L^{n})}$ преобразования нескольких переменных. Данный УФО-элемент является «композицией» нескольких алфавитных элементов, объединенных в одну целостную (эмерджентную) субстанцию в связи с тем, что они обеспечивают одну общую функциональность.
3. Система (УФО-элемент) представляет собой объект $\mathbf{L^{i}L^{in}}$ (i = 1, …, n), занимающий узел $\mathbf{(L^{in})L^{i}, …, L^{n}}$ с одним входом и несколькими выходами, обслуживаемыми одним входом, и реализующий функции $\mathbf{L^{i}(L^{in}), …, L^{n}(L^{in})}$. Данный УФО-элемент является «суперпозицией» разных алфавитных элементов, объединенных в одну субстанцию в связи с одинаковостью входных потоков. Скорее всего, разъединение алфавитных элементов не представляется возможным или целесообразным.
4. Система (УФО-элемент) представляет собой объект $\mathbf{L^{j}L^{i}}$ (i = 1, …, n; j = 1, …, m), занимающий узел $\mathbf{(L^{i}, …, L^{n})L^{j}, …, L^{m}}$ с несколькими входами и несколькими выходами, реализующий сложную функцию $\mathbf{L^{j}, …, L^{m}(L^{i}, …, L^{n})}$. Данный УФО-элемент является «агрегацией», состоящей из нескольких функционально независимых элементов, каждый из которых будет экземпляром определенного класса 1-го, и/или 2-го, и/или 3-го типа, описанных выше. В принципе данные функции могут быть выполнены разными элементами.

Классификационный способ задания алфавита формальной (в данном случае – нормативной) системы играет роль алгоритма для задания семантики знаков этой системы, что превращает ее (нормативную систему) в алгоритмически построенную или конструктивную систему. Следовательно, данный способ обеспечивает получение алфавита нормативной системы, обладающего не только совершенно абстрактной или сугубо математической семантикой, но и предметно (проблемно) -ориентированной, что позволяет рассматривать данный алфавит как ***формально-семантический***, а таким же образом саму нормативную систему. При этом реализуются актуальные предложения специалистов по информатике, которые считают, что уже давно «возникла практическая потребность перехода от формально-математического к содержательному анализу информационных феноменов и их «движущей силы» в самоорганизующихся системах социальной природы» [107].

Использование классификации для порождения семантики алфавитных символов обеспечивает еще одну принципиальную отличительную особенность такого алфавита. Как при конечном, так и при бесконечном формальном алфавите количество исходных понятий, соответствующих знакам алфавита обычной формальной системы, является конечным и ограниченным. Это обстоятельство приводит к тому, что совершенно различные предметные области моделируются с помощью одного и того же набора алфавитных символов нормативной системы какого-либо метода традиционного системного анализа. Задание же алфавита с помощью классификации позволяет изменять количественный и

качественный состав исходных понятий и соответствующих алфавитных символов, т.е. адаптировать алфавит (и нормативную систему) в зависимости от исследуемой и моделируемой предметной области.

Таким образом, предложенный способ формирования алфавита (**узлов**, **функций** и **объектов**: **УФО-элементов**) позволяет использовать при решении каждой конкретной задачи свой конкретный набор средств моделирования, т.е. алфавитных символов. Например, для моделирования информационного бизнеса (организационных систем – средств массовой информации) можно уточнить до соответствующих конкретных классов классы, связанные с информационными связями, а классы, связанные с веществом и энергией оставить в виде абстрактных классов; для моделирования электроэнергетических предприятий необходимо конкретизировать классы, связанные с энергетическими связями; для моделирования транспортных компаний – классы с вещественными связями; для моделирования производства – классы, моделирующие получение вещества соответствующего вида. Незыблемым остается лишь принцип, в соответствии с которым свойства объектов (функций и узлов), используемых для создания модели, определяются их параметрической таксономической классификацией, т.е. базовой классификации классов, учитывающей эти свойства.

Придание алфавиту нормативной системы предметно-ориентированной семантики и свойства адаптивности улучшает ее интерпретационные характеристики и упрощает ее использование для моделирования в различных конкретных ситуациях. Однако, это не снижает степени формальности такой нормативной системы, так как при условии алгоритмического задания семантики и синтаксиса она будет удовлетворять всем требованиям, предъявляемым к формальной системе с точки зрения явного и строгого описание средствами самой формальной системы всех свойств и отношений всех используемых символов, а также с точки зрения распознавания всех символов только по их форме.

При этом обеспечивается возможность объективной (обоснованной) декомпозиции организационной системы (бизнес-системы), и даже возможность передачи, например, компьютерной информационной системе поддержки бизнеса (реинжиниринга бизнеса) формально-логических редакторских и контрольных функций при создании моделей и управлении ими.

Декомпозиция системы, осуществляемая, например, в ходе ООА, помимо выявления структуры классов, направлена также на выявление структуры объектов моделируемой системы и построение объектной модели. Следовательно, необходимо задать определенные правила манипулирования символами алфавита для построения объектных моделей, что, собственно, и обеспечит превращение получаемого с помощью предложенной иерархии классов алфавита в нормативную систему.

Так как символы предложенного алфавита по определению представляют собой различные системные компоненты (узлы, функции, объекты), то, следовательно, правила оперирования этими символами должны быть основаны на системных отношениях, рассматриваемых в рамках выбранного системного подхода. Как известно, функциональная системология рассматривает в качестве

основного системного отношения ***отношение поддержания функциональной способности целого*** [5; см. также п. 2.3]. Это позволяет сформулировать в качестве основного правила оперирования символами алфавита системных компонент ***закон системной декомпозиции***: «Элементы на i-ом ярусе системы должны находиться в отношении поддержания функциональной способности i+1-го яруса системы (системы должны поддерживать надсистему, подсистемы – систему и т.д.)».

Соблюдение данного закона обеспечивается путем выполнения следующих ***правил системной декомпозиции***, естественным образом вытекающих из положений функциональной системологии:

1. ***Правило присоединения***: элементы должны присоединяться друг к другу в соответствии с качественными и количественными характеристиками присущих им связей;
2. ***Правило баланса***: при присоединении элементов друг к другу (в соответствии с правилом 1) должен обеспечиваться качественный и количественный баланс «притока» и «оттока» по входящим и выходящим функциональным связям;
3. ***Правило реализации***: при присоединении элементов друг к другу (в соответствии с правилами 1 и 2) должно быть обеспечено соответствие интерфейсов и объектных характеристик функциональным;
4. ***Правило замкнутости***: внутренние (поддерживающие) связи/потоки элементов в системе должны быть замкнутыми.

Данные правила позволяют собирать из УФО-элементов модели различной сложности, которые называются УФО-конфигурациями. При этом инструментарий УФО-подхода (см. далее) позволяет определять более правильные конфигурации и рационализировать, например, процедуру организационного проектирования бизнес-системы.

Таким образом, предложен алфавит, необходимый для построения моделей организационных систем, который вместе со сформулированными правилами системной декомпозиции представляет собой нормативную систему, особенности которой заключаются в том, что она:

- объективизирует процесс декомпозиции объектов и классов организационной системы;
- предоставляет для каждого вида организационной системы (бизнеса) свою номенклатуру функциональных объектов (алфавитных символов), формируемую по единому принципу;
- обеспечивает имитацию понимания компьютером свойств классов и экземпляров объектной модели.

Построенная таким образом нормативная система, описываемая адаптивным (гибким, динамическим) алфавитом, обладающим содержательной (но однозначной) семантикой, позволяет реализовать новый метод анализа и моделирования организационных систем – УФО-анализ.

5.1.3. Классификация бизнес-систем

Классы УФО-элементов, используемые в качестве алфавитных при проведении УФО-анализа, могут оказаться довольно сложными (4-й случай). Это затрудняет использование таксономических (иерархических) классификационных структур для хранения заготовленных элементов и построения из них концептуальных моделей. С целью преодоления этой проблемы для хранения УФО-элементов и концептуального моделирования используются фасетные классификации в виде таблиц (***УФО-библиотеки***), строки которых соответствуют входным связям, а столбцы – выходным.

При этом, с практической точки зрения, нецелесообразно использование «информационных» связей без указания конкретного вида материального носителя данного вида информации, «энергетических» связей без указания конкретного вида вещественного носителя данного вида энергии, связей «по управлению» без указания конкретного вида данных, являющихся носителями этого вида управления. Таким образом, для построения библиотечных элементов целесообразно использовать связи, являющиеся комбинациями следующих потоков: **V** – поток вещества, **VE** – поток энергии (с учетом вещественного носителя), **VD** – поток данных (с учетом вещественного носителя, например бумажный документ), **VED** – поток данных (с учетом энергетического носителя, например электронный документ), **VDC** – поток управления (с учетом вещественного носителя потока данных), **VEDC** – поток управления (с учетом энергетического носителя потока данных).

Кроме того, эффективность анализа и моделирования любой системы, как известно, в значительной степени зависит от правильности формулирования требований к ней, что выражается в необходимости построения контекстной модели системы перед осуществлением ее декомпозиции. Для управления процессом контекстного моделирования организационных или бизнес-систем в рамках УФО-анализа предлагается узел, занимаемый системой, представлять в виде «образа», изображенного на рисунке 5.4.

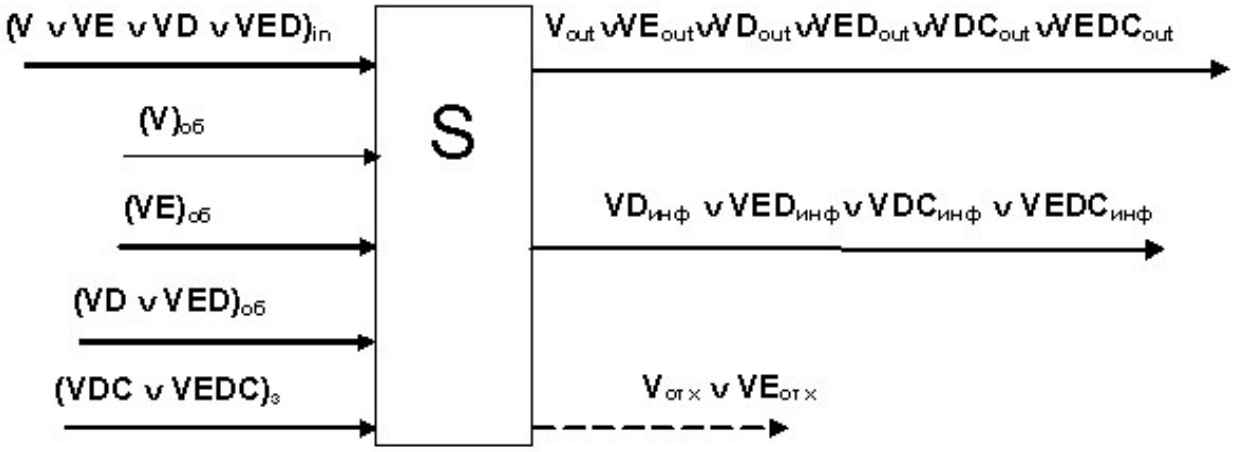


Рис. 5.4. - «Образ» бизнес-системы.

На данном рисунке показано, что любая система ***S*** для производства своей выходной продукции должна получать на вход «предметы труда» (то из чего будет делаться продукция):

- или в виде вещества $\mathbf{V_{in}}$;
- или в виде энергии $\mathbf{VE_{in}}$ на некотором носителе;
- или в виде данных $\mathbf{VD_{in}}$ или $\mathbf{VED_{in}}$ опять же на некотором (вещественном или энергетическом) носителе.

Эта связь соответствует связи «input» стандарта моделирования IDEF0.

Для своего нормального функционирования система должна также получать:

- материально-техническое обеспечение $\mathbf{V_{об}}$, например, оборудование;
- энергетическое обеспечение $\mathbf{VE_{об}}$, например, электроэнергию;
- информационное обеспечение $\mathbf{VD_{об}}$ или $\mathbf{VED_{об}}$, например, описания технологических процессов.

Эти связи соответствуют связи «mechanism» в стандарте IDEF0.

Кроме того, система должна получать управляющие воздействия $\mathbf{VDC_з}$ или $\mathbf{VEDC_з}$, которые, в первую очередь, являются запросами (потребностями) тех систем, для которых данная система вырабатывает свои товары или услуги.

Эта связь соответствует связи «control» в стандарте IDEF0.

При этом допускается отсутствие у системы отдельных входов при наличии у нее достаточных собственных внутренних ресурсов данного вида.

На выходе системы в зависимости от отрасли деятельности могут быть:

- или $\mathbf{V_{out}}$ – вещество;
- или $\mathbf{VE_{out}}$ – энергия на некотором носителе;
- или $\mathbf{VD_{out}}$ ($\mathbf{VED_{out}}$) – данные на некотором носителе;
- или $\mathbf{VDC_{out}}$ ($\mathbf{VEDC_{out}}$) – управляющая информация на носителе.

Эти связи соответствуют связи «output» в стандарте IDEF0.

Кроме того, на выходе системы может иметь место информация (данные – $\mathbf{VD_{инф}}$ и/или $\mathbf{VED_{инф}}$, либо управляющая информация – $\mathbf{VDC_{инф}}$ и/или $\mathbf{VEDC_{инф}}$) о ее функционировании. Это могут быть, например, заявки другим системам на материалы и комплектующие или отчеты в налоговые органы, а также вещество или энергия, представляющие собой отходы производства ($\mathbf{V_{отх}}$ и/или $\mathbf{VE_{отх}}$), например, макулатура.

Определены следующие классы «образов» систем в зависимости от поступающих на вход «предметов труда» и выходной продукции (см. табл. 5.2). В таблице использованы сокращенные обозначения для данных на вещественных (VD = D) и энергетических (VED = G) носителях, а также для управляющей информации на вещественных (VDC = C) и энергетических (VEDC = Q) носителях. Это обусловлено тем, что, в настоящее время, широкое распространение имеют только бумажные (D и C) и электронные (G и Q) носители информации.

Если вход и выход системы принадлежат к потокам разного вида, т.е. выход есть результат некоторого преобразования входа, то получается образ организации ПРОИЗВОДСТВЕННОГО типа. При этом если на входе один вид вещества

V_{in}, а на выходе другой вид вещества V^*_{out}, то имеет место система класса **SPV**, осуществляющая вещественное производство.

Таблица 5.2. Аспекты системного подхода Узел-Функция-Объект.

		Входы:					**Выходы:**		
		Производственный	*Обеспечивающие*			Управляющий	Продуктовый	Информационный	Отходы
			Вещественный	Энергетический	Информационный				
Производство	Вещества – **SPV**	V_{in}	$V_{об}$	$VE_{об}$	$D(G)_{об}$	$C(Q)_{об}$	V^*_{out}	$D(G)_{out}$ $C(Q)_{out}$	$V_{отх}$ $VE_{отх}$
	Энергии – **SPE**	V_{in} VE_{in}	–‘’–	–‘’–	–‘’–	–‘’–	VE^*_{out}	–‘’–	–‘’–
	Информации - **SPI**	$D(G)_{in}$	–‘’–	–‘’–	–‘’–	–‘’–	$D^*(G)^*_{out}$ $C(Q)_{out}$	–‘’–	$V_{отх}$
Транспорт	Вещества – **STV**	V*	–‘’–	–‘’–	–‘’–	–‘’–	V*	–‘’–	–‘’–
	Энергии – **STE**	VE*	–‘’–	–‘’–	–‘’–	–‘’–	VE*	–‘’–	$V_{отх}$ $VE_{отх}$
	Информации - **STI**	D*(G*) C*(Q*)	–‘’–	–‘’–	–‘’–	–‘’–	D*(G*) C*(Q*)	–‘’–	$V_{отх}$
Распределение	Вещества – **SLV**	V	–‘’–	–‘’–	–‘’–	–‘’–	V”	–‘’–	–‘’–
	Энергии – **SLE**	VE	–‘’–	–‘’–	–‘’–	–‘’–	VE”	–‘’–	$V_{отх}$ $VE_{отх}$
	Информации - **SLI**	D(G)	–‘’–	–‘’–	–‘’–	–‘’–	D”(G”)	–‘’–	–‘’–

Если на входе один вид энергии VE_{in} и/или какой-то вид вещества V_{in}, а на выходе другой вид энергии VE^*_{out}, то имеет место система класса **SPE**, осуществляющая энергетическое производство. Если на входе один вид данных на бумаге D_{in} и/или в электронном виде G_{in}, а на выходе другой вид данных D^*_{out} и/или G^*_{out}, либо управляющая информация на бумаге C_{out} и/или в электронном виде Q_{out}, то имеет место система класса **SPI**, осуществляющая информационное производство. Это могут быть административные организации или организации

аналитического типа. При этом их легко отличить. На выходе административных организаций, естественно, должны быть C_{out} и/или Q_{out}, а на выходе аналитических – только D_{out} и/или G_{out}.

Если вход и выход системы принадлежат к потокам одного вида (в идеале, эквивалентны), то получается образ системы ТРАНСПОРТНОГО класса. При этом если на входе некоторый вид вещества V^*_{in} и на выходе этот же вид вещества V^*_{out}, а также выполняется равенство $V^*_{in} = V^*_{out}$, то имеет место система класса **STV**, осуществляющая транспортировку вещества. Если на входе некоторый вид энергии VE^*_{in} и на выходе этот же вид энергии VE^*_{out}, а также выполняется равенство $VE^*_{in} = VE^*_{out}$, то имеет место система класса **STE**, осуществляющая транспортировку энергии. Если на входе некоторый вид данных на бумаге D^*_{in} и/или в электронном виде G^*_{in}, или управляющей информации на бумаге C^*_{in} и/или в электронном виде Q^*_{in}, а на выходе этот же вид данных D^*_{out} и/или G^*_{out}, или управляющей информации C^*_{out} и/или Q^*_{out}, то имеет место система класса **STI**, осуществляющая транспортировку информации.

Ситуация, когда на входе и на выходе один и тот же вид вещества, энергии или информации характерна и для систем ЗАГОТОВИТЕЛЬНО-РАСПРЕДЕЛИТЕЛЬНОГО класса **SL**. Сюда же относятся контролирующие организации. Однако «образы» этих классов отличаются от «образов» транспортных классов тем, что их производственные входы и их продуктовые выходы могут иметь разные количественные характеристики.

В заключении классифицирования видов «образов» организационных систем необходимо отметить, что выбор того либо иного образа для моделирования системы будет определяющим с точки зрения результатов анализа или проектирования. Так для анализа деятельности, например, кафедры высшего учебного заведения или органов управления теоретически можно использовать как образ информационно-производственной системы, так и заготовительно-распределительной. Совершенно ясно, что результаты моделирования будут принципиально отличаться. Следовательно, выбор образа должен быть обусловлен, в первую очередь, требованиями тех систем, которые реально связаны или будут связаны с выходами рассматриваемой (анализируемой) системы.

Таким образом, в рамках УФО-анализа предполагается использование девяти типов библиотек, в соответствии с классами «образов» бизнес-систем. Каждая библиотека в настоящее время представляет собой таблицу, строки которой соответствуют видам входных связей узлов, а столбцы – видам выходных связей. Каждая ячейка таблицы хранит сведения о функциональном процессе (или наборе процессов), обеспечивающем преобразование входов данного узла в выходы. При этом могут храниться сведения о нескольких вариантах таких процессов или вариантах наборов. Кроме того, относительно каждого варианта процесса (или варианта набора) хранятся сведения об объекте (или наборе объектов), способном реализовать данный процесс (или данный набор процессов). При этом могут храниться сведения о нескольких объектах или нескольких наборах объектов для каждого процессов или каждого набора процессов. Пример функциональных характеристик УФО-элементов высокоуровневого алфавита

для описания организационных систем представлен в таблице 5.3, объектных – в таблице 5.4.

Таким образом, средства УФО-анализа (в первую очередь его формально-семантическая нормативная система) позволяют конкретизировать и наполнить предметным содержанием абстрактные подходы и концепции традиционных методов системного структурного анализа (например, представление о связях «вход», «выход», «управление», «механизм» технологии моделирования SADT/IDEF0), повышая выразительные возможности системного анализа организационных систем.

Применение данных результатов для построения модели производственной системы одного из предприятий показало, что все подразделения такой системы могут быть представлены как изображения перечисленных образов, а сама производственная система – как конфигурация этих изображений (см. далее). При этом такое формально-семантическое представление модели производства позволяет выработать ряд конструктивных рекомендаций по совершенствованию организационной структуры.

Рассмотренная классификация образов организационных систем (бизнес-систем) позволяет значительно упростить решение первой (и самой важной) задачи моделирования и анализа, а именно, задачи построения контекстной модели. Важность данного результата обусловлена тем, что, до настоящего времени, выбор контекстного представления системы (в том числе организационной) являлся исключительно эвристической процедурой. Однако, именно контекстная модель является исходной точкой всего анализа, задающей его направление и, по сути дела, определяющей в значительной степени результат.

5.2. Процедура системно-объектного моделирования и анализа

5.2.1 Алгоритм УФО-анализа.

Рассмотрим этапы УФО-анализа [95, 108, 109].

Суть этапов данного метода анализа может быть представлена следующими основными шагами:

- выявление узлов связей в структуре моделируемой системы на основании функциональных связей системы в целом;
- выявление функциональности, поддерживающей (обеспечивающей) обнаруженные узлы;
- определение объектов, соответствующих выявленной функциональности.

Первый шаг УФО-анализа может быть отождествлен с этапом собственно анализа проблемы и моделируемой системы, второй – с этапом ее проектирования, а третий – с ее реализацией. Эти шаги предваряются обязательным подготовительным этапом, задачей которого является настройка (адаптация) соответствующей библиотеки УФО-элементов к конкретной проблеме и сфере деятельности бизнес-системы.

Таблица 5.3. Пример описания функциональных характеристик бизнес-систем как УФО-элементов.

Выходы: / Входы:		Вещество	Энергия	Данные на вещественных носителях	Управление с механическим приводом	Данные на электроносителях	Управление с электроприводом
		V	VE	VD	VDC	VED	VEDC
Вещество	V	Транспортировка, хранение, преобразование	Генерация, накопление	Измерение, обнаружение	Регулирование	Измерение, обнаружение	Регулирование
Энергия	VE	Расходование (окисление)	Передача, преобразование	Измерение, обнаружение	Регулирование	Измерение, обнаружение	Регулирование
Выходы: / Входы:				Бумажные информационные документы	Бумажные нормативные документы	Электронные информационные документы	Электронные нормативные документы
		V	VE (E)	D (VD)	C (VDC)	G (VED)	Q (VEDC)
Бумажные информационные документы	D (VD)	Утилизация документов		Пересылка, передача, хранение, размножение; создание новых	Создание новых; подписание, утверждение	Сканирование и набор; хранение в БД; создание новых с использованием ЭВМ	Нет
Бумажные нормативные документы	C (VDC)			Создание новых документов	Пересылка, передача, хранение, размножение; создание новых; подписание, утверждение	Создание новых с использованием ЭВМ	Сканирование, хранение в БД
Электронные информационные документы	G (VED)			Распечатывание; создание новых с использованием ЭВМ	Создание новых с использованием ЭВМ, подписание, утверждение	Пересылка по e-mail, хранение в БД или эл. библиотеке; создание новых с использованием ЭВМ	Нет
Электронные нормативные документы	Q (VEDC)			Создание новых документов с использованием ЭВМ	Создание новых с использованием ЭВМ, подписание, утверждение	Создание новых с использованием ЭВМ	Пересылка по e-mail, хранение в БД или эл. библиотеке

Таблица 5.4. Пример описания объектных характеристик бизнес-систем как УФО-элементов

Входы: \ Выходы:		Вещество	Энергия	Данные на вещественных носителях	Управление с механическим приводом	Данные на электроносителях	Управление с электроприводом
		V	VE	VD	VDC	VED	VEDC
Вещество	V	Продуктопроводы; хранилища; хим. генераторы	Химические источники питания (аккумуляторы); плотины; кормление животных.	Механические датчики и счетчики (уровня, расхода)	Механические регуляторы (уровня: запорные устройства)	Электронные датчики и счетчики (наличия, уровня, расхода) с встроенным питанием	Электронные регуляторы (уровня, расхода) с встроенным питанием
Энергия	VE	Камеры сгорания (печи и камины)	Зубчатые и рычажные передачи; гидроприводы, электрогенераторы; вторичные источники питания; электросети; теплотрассы	Механические счетчики (электричества); стрелочные амперметры, вольтметры, барометры, часы; термометры;	Механические регуляторы (температуры биметаллический регулятор)	Электронные счетчики (электричества); осциллографы; эл. часы, термометры	Электронные регуляторы (уровня, расхода); авторепе регуляторы; взрыватели (детонаторы)
Входы: \ Выходы:				Бумажные информационные документы	Бумажные нормативные документы	Электронные информационные документы	Электронные нормативные документы
		V	VE (E)	D (VD)	C (VDC)	G (VED)	Q (VEDC)
Бумажные информационные документы	D (VD)	Мусоросборник		Канцелярия, секретариат; бухгалтерия; библиотека; архив отделы: НТИ, плановый, экономический, технологический, производственный; разработчики	Администрация	Бухгалтерия, секретариат, отделы: НТИ, плановый, экономический, маркетинга, производственный; разработчики	Нет
Бумажные нормативные документы	C (VDC)			Бухгалтерия; отделы: НТИ, плановый, экономический, технологический, маркетинга, производственный; разработчики	Администрация; канцелярия; секретариат; отдел стандартизации; библиотека; архив	Бухгалтерия, отделы НТИ, плановый, экономический, технологический, маркетинга, производственный; разработчики	Администрация; секретариат; эл. библиотека
Электронные информационные документы	G (VED)			Бухгалтерия; отделы: НТИ, плановый, экономический, технологический, маркетинга, производственный; разработчики	Администрация	Бухгалтерия, отделы НТИ, плановый, экономический, технологический, маркетинга, производственный; разработчики	Нет
Электронные нормативные документы	Q (VEDC)			Бухгалтерия; отделы: НТИ, плановый, экономический, технологический, маркетинга, производственный; разработчики	Администрация	Бухгалтерия, отделы НТИ, плановый, экономический, технологический, маркетинга, производственный; разработчики	Администрация; секретариат; отдел стандартизации; эл. библиотека

УФО-подход и основанная на нем методика УФО-анализа по сути дела предоставляют аналитику конструктор (типа конструктора «Лего») для сборки моделей бизнес-систем. Существенным в данном случае отличием является предоставляемая аналитику возможность самому создавать детали, из которых он будет потом конструировать нужные модели. Эти детали описываются в виде библиотечных УФО-элементов и хранятся в соответствующей УФО-библиотеке. Данная возможность позволяет достичь необходимой степени точности и адекватности создаваемых моделей.

Кроме того, наличие библиотеки, т.е. алфавита, УФО-элементов и формальных правил комбинирования ими позволяет автоматизировать процесс сборки конфигурации из этих элементов. Для этого необходимо, во-первых, доработать классификацию связей с учетом особенностей анализируемой бизнес-системы и, в первую очередь, ее миссии. Во-вторых, необходимо адаптировать наиболее подходящую для данного случая библиотеку УФО-элементов таким образом, чтобы она включала как можно больше частей, потенциально пригодных для моделирования (сборки) системы. В-третьих, необходимо с максимальной степенью точности и подробности описать анализируемую бизнес-систему в виде узла, т.е. перекрестка входных и выходных связей из доработанной классификации. В-четвертых, при моделировании бизнес-системы необходимо использовать только такие конфигурации, которые могут быть названы «***логистическими конфигурациями***». Данные конфигурации отличаются тем, что любой выход каждого элемента такой конфигурации или повторяет его вход, или является выходом такого типа, которого еще не было во всей этой конфигурации, начиная с входа первого элемента. Это соответствует реальной действительности, так как, если из какого-то материала или сырья сделана некоторая деталь, то никогда не происходит процесса превращения этой детали обратно в этот же материал. При выполнении названных условий построение модели бизнес-системы из ее частей может рассматриваться как сборка УФО-конфигурации из библиотечных УФО-элементов, которая выполняется по формальным правилам, т.е. может быть автоматизирована.

Общая схема УФО-анализа представлена в соответствием с требованиями ЕСПД на рис. 5.5. Из схемы алгоритма видно, что процесс УФО-анализа начинается с анализа требований заказчика на разработку системы «сквозь призму» исходной классификации связей (см. рис. 5.2). Эта таксономическая классификация, а также библиотека классов (фасетная классификация) «Узлы – Функции – Объекты» должны быть предварительно адаптированы для проведения анализа и моделирования конкретной системы. Это значит, что они должны включать в себя конкретные связи, узлы, функции и объекты, используемые в данной предметной области. Чем больше видов конкретных классов будет предварительно заготовлено в данных классификациях, тем эффективнее и проще будут происходить анализ и проектирование с помощью данного алгоритма.

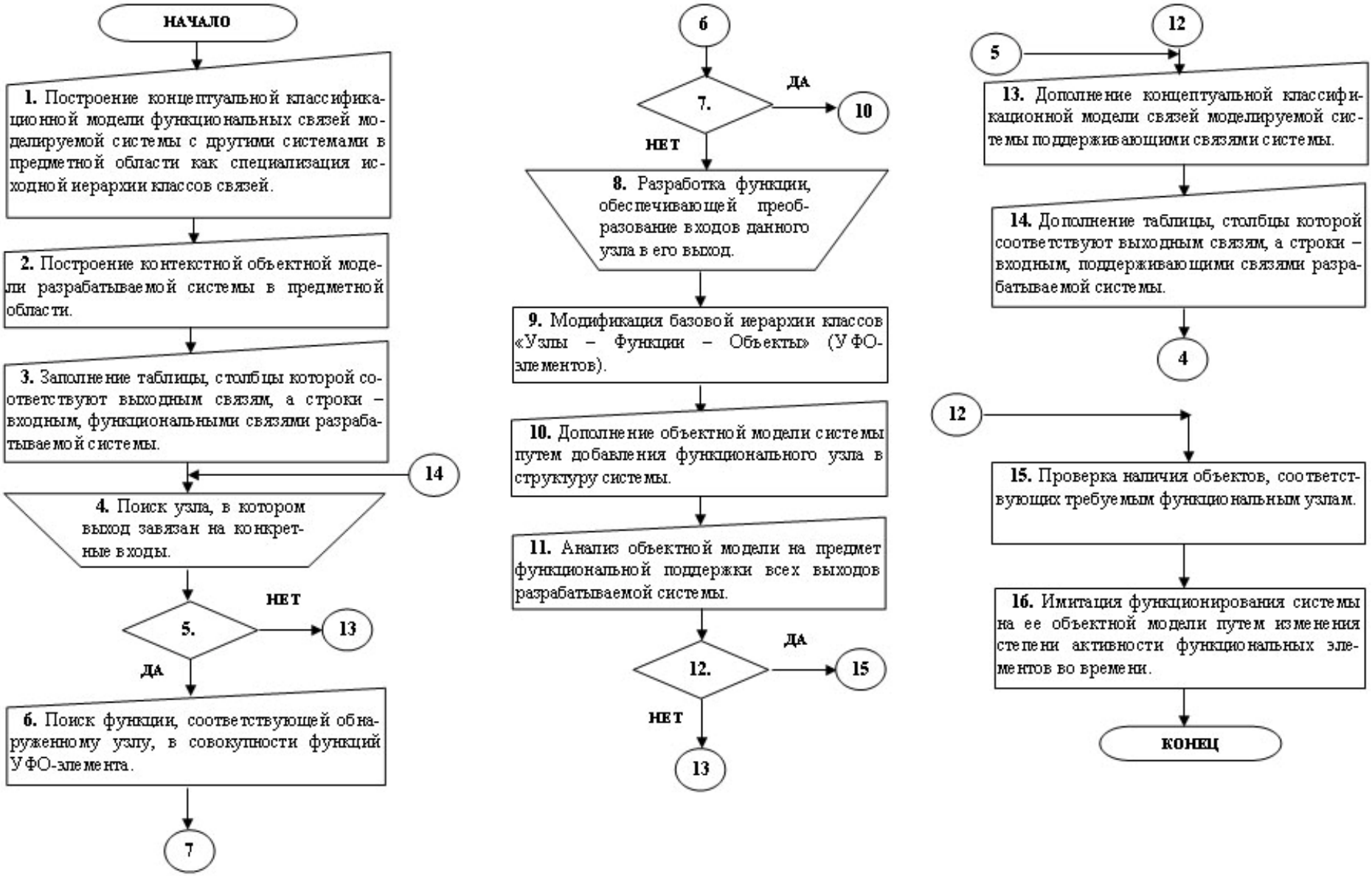


Рис. 5.5. - Алгоритм УФО-анализа.

Непосредственно процедура анализа начинается с построения контекстной модели системы, показывающей ее функциональные связи, в максимально возможной степени обозначенные знаками формально-семантического алфавита, т.е. знаками заготовленных конкретных классов связей. Данная модель соответствует диаграмме прецедентов системы на языке объектного моделирования UML, а также контекстной диаграмме SADT (IDEF0) или DFD.

Затем можно использовать таблицу, включающую выявленные функциональные связи моделируемой системы. Строки данной таблицы соответствуют входным связям системы, а столбцы – выходным. Анализ этой таблицы представляет собой итеративный процесс выявления узлов в структуре системы. Этот процесс направляется необходимостью обнаружения таких узлов, для которых может быть определена (найдена среди известных для данного УФО-элемента или разработана) конкретная функция, обеспечивающая в данном узле баланс втекающих и вытекающих потоков. Невозможность обнаружить узлы в структуре системы с известной функциональностью приводит к необходимости добавления новых видов связей, поддерживающих функциональные на все более глубоком ярусе иерархии данной системы. Добавленные связи также заносятся в таблицу и анализ ее повторяется до тех пор, пока разрабатываемая система не будет представлена в виде сети проточных узлов, для которых могут быть найдены известные или разработаны новые функции.

Завершается алгоритм УФО-анализа поиском подходящих объектов, реализующих полученную функциональность, среди заготовленных в УФО-библиотеках или же «на стороне» путем приобретения или заказа. Критерием адекватности определения функциональных объектов для идентифицированных узлов является удовлетворяющая заказчика имитация функционирования системы. При этом, в данном случае, имитационная модель системы создается в результате совершенно адекватного описания рыночных взаимоотношений организационной системы с помощью понятий системологии.

Упомянутая на начальных шагах алгоритма адаптация УФО-библиотеки (классификации) состоит в том, что она должна быть развернута (специализирована) до конкретных узлов, функций и объектов рассматриваемой предметной области, т.е. представлять собой ее ***модель онтологии***. Спецификации УФО-элементов должны содержать следующую информацию:

- Для ***объекта*** (параметры субстанции): технические и эксплуатационные характеристики (конструктивное исполнение; климатические и механические условия эксплуатации; надежность; имеемые ресурсы (запасы) энергии, материалов и информации; производительность по входу и выходу, пропускная способность и т.д.); стоимость и время эксплуатации.
- Для ***функции*** (параметры процесса): содержательное описание преобразования входа в выход, т.е. протокол функционирования; формальное описание функциональной зависимости, если оно существует и необходимо, в виде скрипта или макроса. По сути дела это данные о внутренней детерминанте соответствующей системы.

– Для ***узла*** (параметры структуры): качественные характеристики потоков. По сути дела это данные о вариантах использования объекта, т.е. о внешней детерминанте соответствующей системы.

Использование в процессе моделирования конкретных узлов, функций и объектов, для которых известна перечисленная информация, позволяет опознавать узлы и автоматически определять для них функции и объекты, если используются знаки формально-семантического алфавита. При этом, если алфавитными элементами являются программные объекты, реализованные в виде готовых классов, то можно говорить об УФО-анализе как составной части компонентных технологий и технологии бизнес-объектов CORBA (Business Object Facility – BOF). В последнем случае программное CASE-средство, автоматизирующее процедуры УФО-анализа, может функционировать в рамках компонентной архитектуры бизнес-объектов (Business Object Component Architecture – BOCA) в роли организатора (Framework), который «работает как инструмент объединения бизнес-объектов в действующую систему, предоставляя им своего рода удобные рабочие «места-кабинеты» для выполнения возложенных на них задач» [110]. Если же в качестве алфавитных элементов рассматриваются объекты техники, то УФО-анализ будет согласованным с CALS-технологией.

Таким образом, предложена схема УФО-анализа, основанного на использовании библиотеки классов (классификации) «Узлы – Функции – Объекты», представляющего собой новый метод системного анализа, процедуры и результаты которого согласуются с требованиями OOD. Метод позволяет использовать формализованные правила выявления классов и объектов предметной области в процессе OOA и OOD, что значительно облегчает процесс обнаружения подходящих классов и объектов и повышает эффективность моделирования и проектирования сложных систем.

Кроме того, представленный метод анализа, являясь результатом интеграции системного и объектного подходов, обеспечивает пользователю:

– объективность процедур анализа и синтеза организационных систем (бизнес-систем, производства);
– экономию трудоемкости анализа и моделирования, так как эти процедуры сводятся к построению всего одной модели;
– простоту моделирования бизнес-процессов специалистами без специальной подготовки в области системного анализа и OOAD;
– единообразное представление внешней и внутренней моделей бизнес-системы, описываемых одним языком моделирования;
– простоту адаптации моделей к конкретной предметной области (учета семантики предметной области);
– возможность создания и использования библиотек (репозитариев) модельных компонент для различных предметных областей.

Следовательно, УФО-технология обладает следующими достоинствами:

– впервые обеспечивает согласование результатов системного анализа с

требованиями объектно-ориентированного проектирования, которые ранее рассматривались как ортогональные;

- впервые обеспечивает возможность непосредственного использования результатов системного анализа при создании объектно-ориентированного программного обеспечения;
- повышает уровень формальности и автоматизации процедур моделирования и анализа организаций и предприятий;
- гарантирует согласование всех характеристик системы за счет объединения в одной модели различных аспектов ее рассмотрения;
- обеспечивает простоту построения визуальных моделей разного уровня абстракции, представляющих одновременно функциональную и объектную структуру системы (бизнес-системы, производства);
- обеспечивает возможность моделирования функциональных характеристик системы, интерпретируемых любым математическим аппаратом или не имеющих математической интерпретации, а также возможность имитации функционирования системы без специального алгоритма.

Таким образом, представленная технология анализа и моделирования может быть использована для ***корректирующего информационно-аналитического сопровождения бизнес-систем*** (организаций, предприятий и т.п.) и обеспечения существенного повышение эффективности их деятельности.

5.2.2. Примеры УФО-моделей.

Свойства и особенности УФО-моделей рассмотрим на примере моделирования сервисной службы телерадиосети в интересах федерального государственного унитарного предприятия «Российские телерадиосети» (ФГУП РТРС) в соотвествии, например, с работой [111].

Согласно процедурам и методам УФО-подхода для создания моделей необходимо построить **базовую иерархию связей**. Связи это материальные и информационные потоки, необходимые для обеспечения деятельности РТРС, в том числе и для сервисных работ. При создании модели рассматриваются следующие виды связей.

Средства обслуживания (СО) рассматриваются как материальные потоки.

Услуги РТРС, такие как телевизионный сигнал также рассматриваются как материальные потоки.

К управляющей информации относятся руководящие указания руководителей различного уровня и комплект документов по которому работает сервисная служба

Данные разделены на следующие подвиды: заказ, финансовая информация, отчёты о проделанной работе, сервисные данные. Каждый из них также разделяется на свои подвиды. Группа связей, содержащая информацию о состоянии вещательного узла (ВУ), относиться к сервисным данным.

Базовая иерархия связей для моделирования сервисной службы выглядит следующим образом (рис. 5.6):

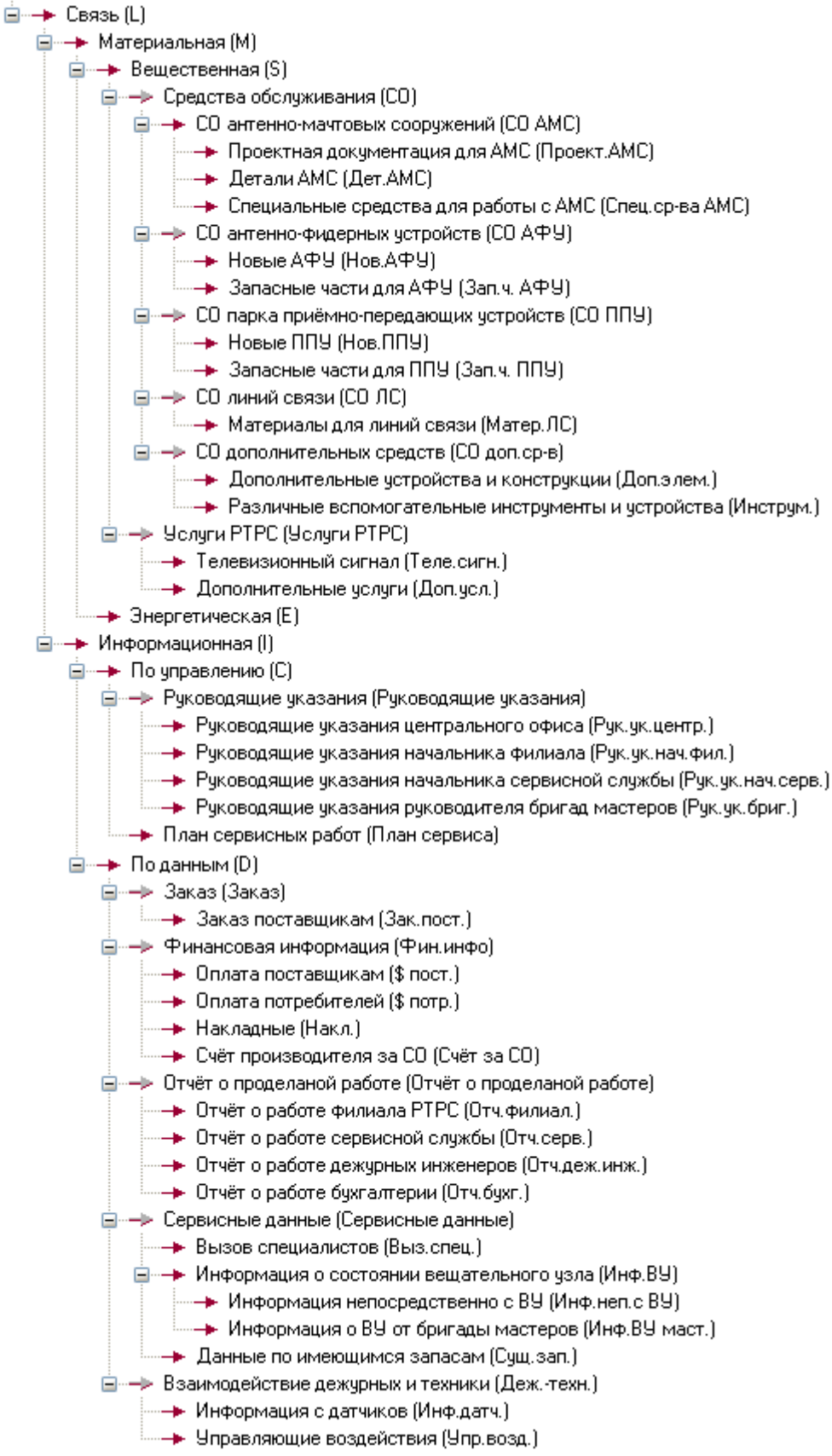


Рис. 5.6. – Базовая иерархия связей.

Моделирование структуры и процессов сервисной службы осуществляется следующим образом:

- элементы логистической цепочки сервиса представляются как УФО-элементы;
- потоки материалов и инструментов представляются как связи;
- перекрёстки связей представляются как узлы;
- сервисные работы представляются как функции соответствующих узлов;
- непосредственные исполнители, т.е. отделы и бригады мастеров представляются как объекты.

Контекстная диаграмма ФГУП РТРС, выполненная с учетом приведенной выше информации, представлена на рисунке 5.7.

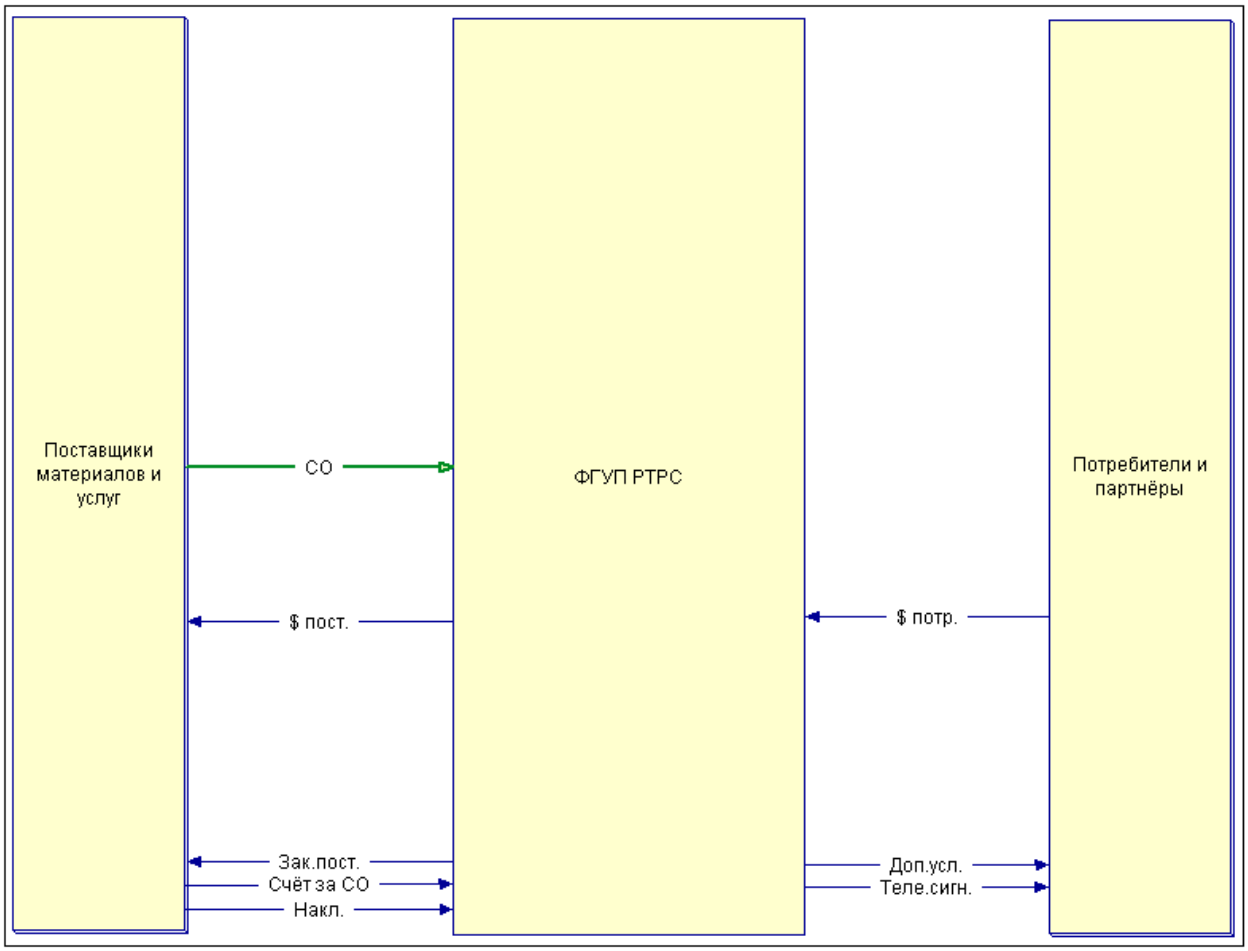


Рис. 5.7. – Контекстная диаграмма ФГУП РТРС.

Декомпозиция контекстной диаграммы, представлена на рисунке 5.8:

«Центральное управление РТРС» - это центральный офис компании в функции которого входит управление и координация деятельности всех филиалов. «Консолидированные склады» (КС) необходимы сервисной службе как временные пункты хранения при транспортировке средств обслуживания от производителей в филиалы. Элемент «Филиал РТРС» имеется на диаграмме в

одном экземпляре и представляет собой типовой филиал РТРС из общей совокупности филиалов. Это сделано исходя из того, что центральный офис и КС взаимодействуют с различными филиалами одинаково.

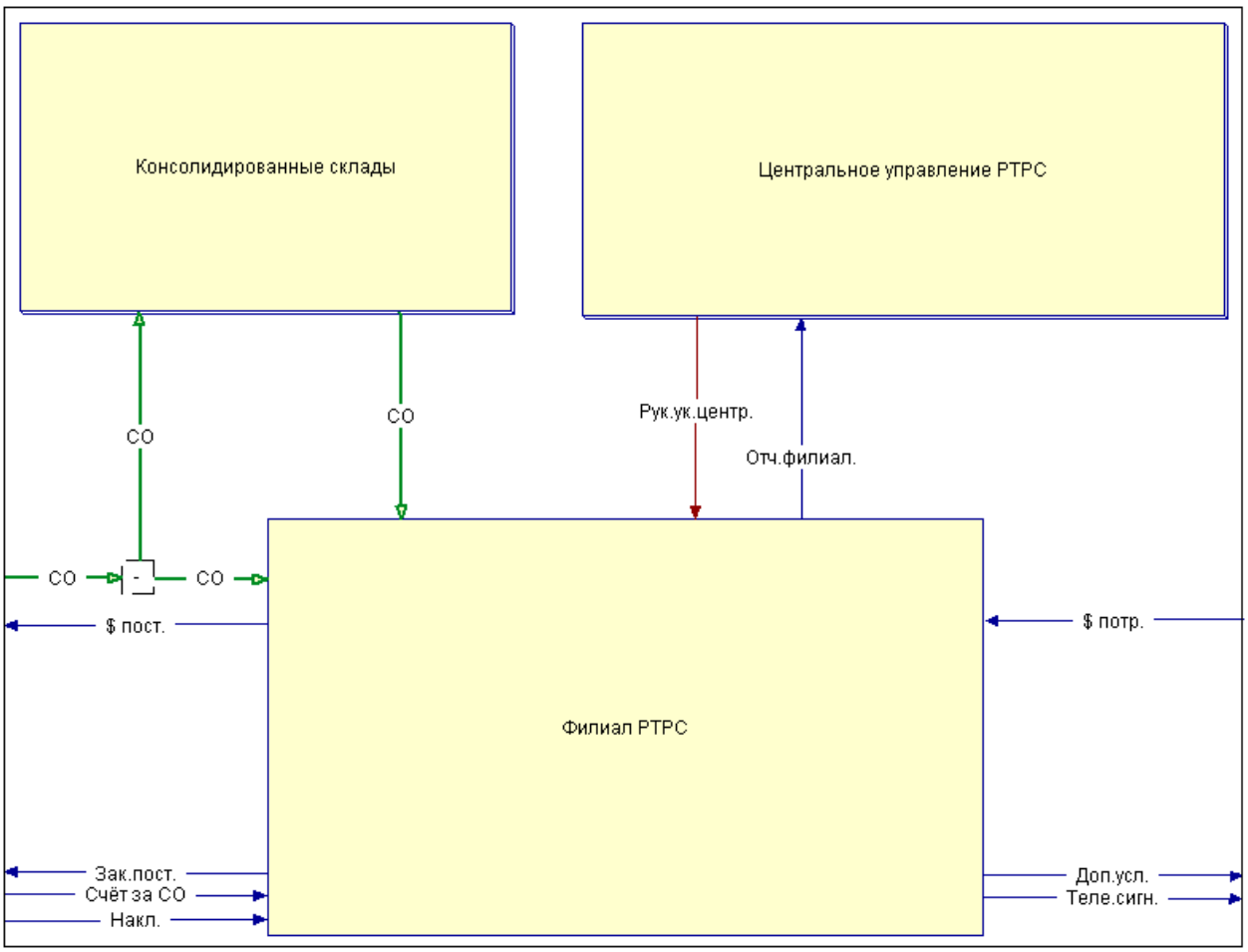


Рис. 5.8. – Декомпозиция контекстной диаграммы ФГУП РТРС.

Филиал РТРС рассматривается в контексте сервисного обслуживания ВУ. Реальный филиал РТРС представляет собой очень большую и сложную организационную структуру географически распределённую структуру с большим количеством подразделения и сотрудников, выполняющих большое количество различных задач. Однако, в данном случае, нашей задачей является моделирование лишь одного подразделения филиала, выполняющего функции сервисного обслуживания. Поэтому филиал не будет рассмотрен целиком, а будут рассмотрены лишь те его структурные подразделения и процессы, которые имеют значение для моделирования сервисного обслуживания, т.е. в данной модели филиал будет представлен в упрощённом виде.

Рассмотрим внутреннюю структуру Филиала РТРС (рис. 5.9).При рассмотрении филиала РТРС с точки зрения сервисного обслуживания целесообразно выделить в нём три структурных элемента. Это непосредственно «Подразделение сервисного обслуживания» (ПСО) филиала, а также «Управление филиалом» и «ВУ». Рассмотрим эти элементы подробнее.

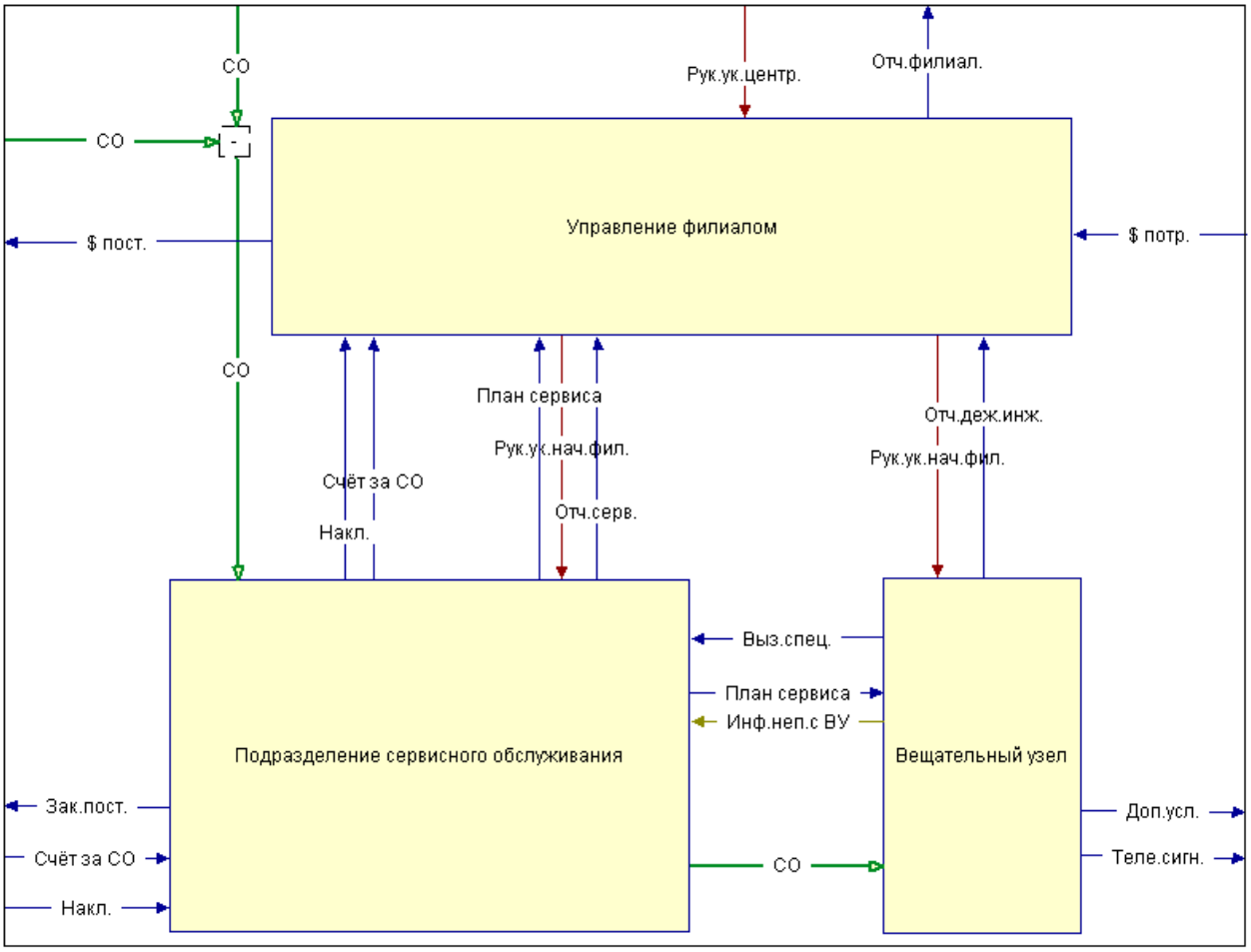


Рис. 5.9. – Структура филиала ФГУП РТРС.

«ПСО» филиала занимается сервисным обслуживанием и обновлением материально-технической базы (МТБ) филиала. Сервисным обслуживанием управляют администрация филиала и начальник ПСО. Объектом обслуживания является вещательный узел. Его структура представлена на рисунке 5.10.

«Управление филиалом» это администрация филиала и бухгалтерия филиала. Они принимают следующее участие в сервисном обслуживании: администрация контролирует и направляет деятельность ПСО, а бухгалтерия осуществляет оплату необходимых для сервиса СО.

«ВУ» это объект сервиса. Стоит отметить, что именно ВУ выполняет основные функции самого филиала РТРС. Именно поэтому сервисное обслуживание ВУ для поддержания его работоспособности является важной задачей. УФО-элемент ВУ представляет собой совокупность технических средств и обслуживающего персонала, который обеспечивает бесперебойную работу этих средств.

ПСО проводит регулярные работы на объектах филиала, устраняет последствия аварийных ситуаций, закупает необходимые средства обслуживания и обновляет МТБ филиала. Определённая часть сервисных функций выполняется непосредственно персоналом филиала. Одна сервисная служба может обслуживать как один филиал, так и несколько.

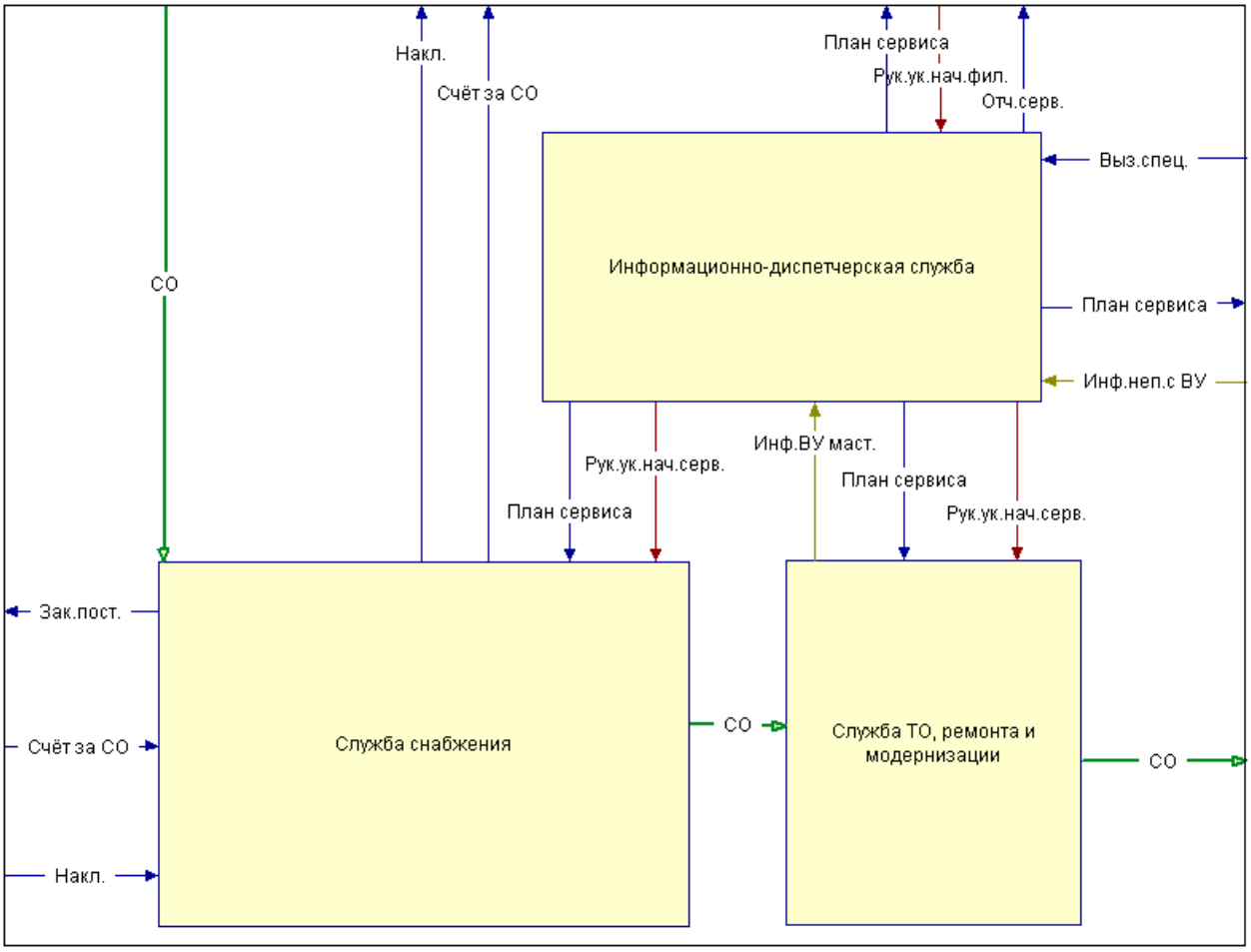


Рис. 5.10. – Структура подразделения сервисного обслуживания.

Это связано с тем, что каждый из почти сотни филиалов РТРС имеет свои географические и технические особенности. Например, в западной части России в зоне вещания «М» плотность насёленных пунктов достаточно велика и, следовательно, местные филиалы РТРС имеют большую плотность вещательных узлов. При этом одна сервисная служба может обслуживать сразу несколько филиалов в связи с низкой удалённостью объектов обслуживания друг от друга. В отличие от этого, в северо-восточной части России в зонах вещания «А», «Б» и «В» плотность населённых пунктов, а следовательно и ВУ, очень низка, поэтому один филиал (занимающий на востоке значительно большую площадь чем на западе) может иметь в своём составе несколько ПСО. Это необходимо для уменьшения расстояния между субъектами и объектами сервиса, например, для более быстрого устранения последствий аварийных ситуаций. Кроме этого, приграничные ВУ могут иметь большее количество различного оборудования, т.к. могут использоваться для таких дополнительных целей как подавление иностранных сигналов или оказание помощь пограничным службам. Это влечёт за собой усложнение сервисного обслуживания, что, в свою очередь, влечёт за собой необходимость более близкого размещения обслуживающих бригад или создание дополнительных ПСО. От данных факторов также зависит количество и состав бригад различной специализации, количество и

размеры складов.

ПСО, в свою очередь, разделено на 3 элемента: «Информационно-диспетчерскую службу», «Службу снабжения» и «Службу ТО, ремонта и модернизации». Рассмотрим их подробно.

«Информационно-диспетчерская служба» (ИДС) занимается сбором и анализом информации и руководит сервисными работами.

Функции ИДС:

- Сбор информации о ВУ от бригад мастеров и дежурных инженеров филиала.
- На основе информации о ВУ разработка плана сервиса для работы ПСО.
- Выполнение указаний администрации филиала, составление отчёта о работе сервисной службы и его передача администрации филиала.
- Управление службой снабжения и службой ТО, ремонта и модернизации.

Структура ИДС, представленная на рисунке 5.11.

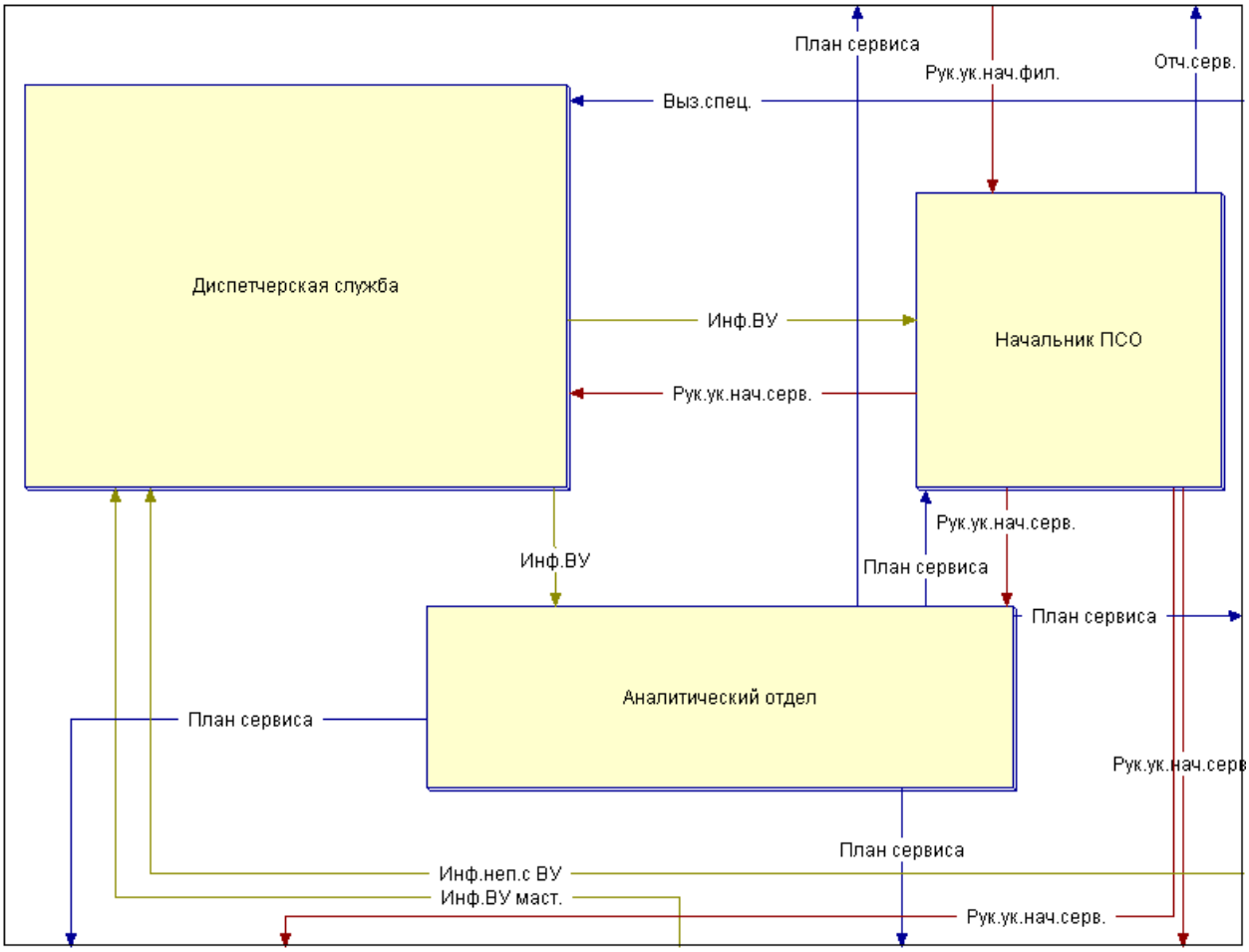


Рис. 5.11. – Структура информационно-диспетчерской службы.

Для реализации описанных функций ИДС необходима декомпозиция ИДС на три элемента. «Диспетчерская служба» внутри ИДС выполняет сбор и первичную обработку информации, «Аналитический отдел» выполняет анализ

информации и составление плана сервиса, а «Начальник ПСО» руководит работой подразделений.

«Служба ТО, ремонта и модернизации» занимается непосредственно выполнением основных сервисных работ.

Функции «Службы ТО, ремонта и модернизации»:

- Выполнение распоряжений начальника ПСО.
- Выполнение сервисных работ при помощи поступающих СО.
- Сбор информации о ВУ при работе и передача этой информации в диспетчерскую службу.
- Получение из аналитического отдела и выполнение плана сервиса.

Главной функцией службы ТО, ремонта и модернизации является выполнение сервисных работ. Именно служба ТО, ремонта и модернизации реализует главную функцию всего ПСО. Структура элемента «Служба ТО, ремонта и модернизации» представлена на рисунке 5.12.

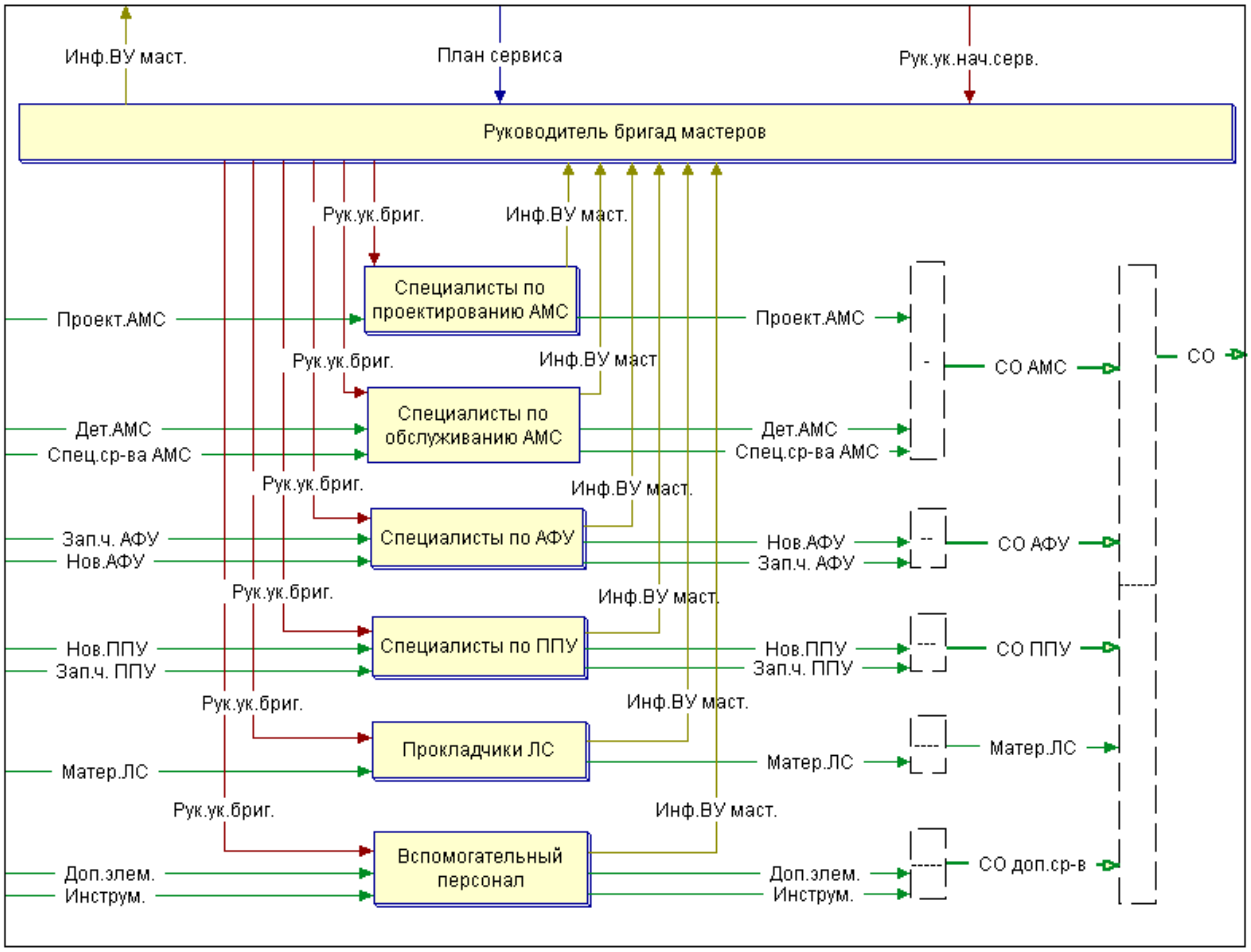


Рис. 5.12. – Структура службы ТО, ремонта и модернизации.

Главная задача «Службы снабжения» это обеспечение ПСО необходимыми средствами для проведения сервисных работ. Закупка СО производится в соответствии с планом сервиса и распоряжениями начальника ПСО. «Служба

снабжения» формирует и отправляет заказ поставщикам на необходимые СО. От поставщиков служба снабжения получает счёт за СО и отправляет его в бухгалтерию филиала. После оплаты бухгалтерией счёта на склады службы снабжения поступают СО и накладные. Накладные передаются в бухгалтерию, а СО поступают в службу ТО, ремонта и модернизации для проведения СР. Службой снабжения управляет начальник ПСО. Структура службы снабжения представлена на рисунке 5.13.

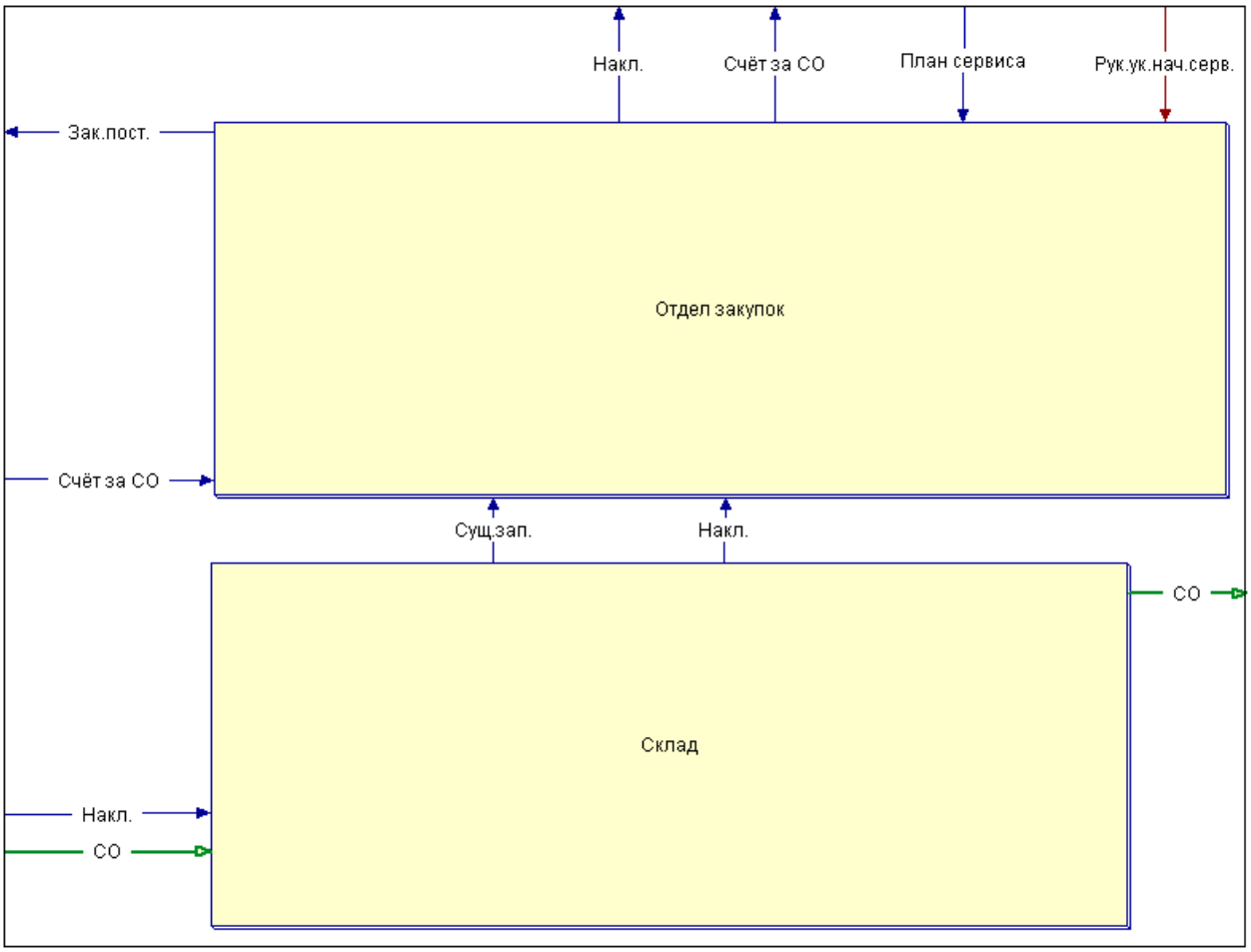


Рис. 5.13. – Структура службы снабжения.

Функции службы снабжения:

- Выполнение распоряжений начальника ПСО.
- Получение плана сервиса от аналитического отдела.
- На основании плана сервиса и распоряжений начальника ПСО анализ рынка и поиск поставщиков необходимого СО.
- Подготовка и отправка поставщикам заказа на необходимые СО.
- Получение счёта, выставленного поставщиками, и его отправка в бухгалтерию филиала.
- Получение накладных вместе с поступающими СО.
- После поступления СО на склад передача накладных в бухгалтерию фи-

лиала.

- Получение СО.
- Хранение СО.
- Выдача СО службе ТО, ремонта и модернизации.

Как видно из диаграммы службы снабжения она состоит из двух подразделений – «Склада» и «Отдела закупок». Главная задача «Отдела закупок» закупка необходимых для сервиса СО. Закупка производится в соответствии с планом сервиса, распоряжениями начальника ПСО и существующими запасами СО на складе. Проводится тщательный анализ рынка. Отдел закупок формирует и отправляет заказ поставщикам на необходимые СО. С наиболее приемлемыми поставщиками заключаются долгосрочные соглашения. Затем отдел закупок получает счёт за СО и отправляет его в бухгалтерию филиала для оплаты. После оплаты бухгалтерией счёта и поступления СО на склад в отдел закупок со склада поступают накладные, которые также передаются в бухгалтерию. Бракованные СО возвращаются назад.

Представленные диаграммы (УФО-модели) использованы для регламентации процессов сервисного обслуживания ФГУП РТРС.

5.3. CASE-инструментарий системно-объектного моделирования и анализа

5.3.1. Назначение и возможности «UFO-toolkit»

Представление о Business Intelligence отечественных специалистов, например [112], позволяет рассматривать программный CASE-инструментарий моделирования бизнес-систем и бизнес-процессов как разновидность BI-инструментария при условии, если он:

a) обеспечивает «*процесс превращения данных в информацию и знания для поддержки принятия улучшенных и неформальных решений*»;
b) встраивается в «*информационные технологии (методы и средства) сбора данных, консолидации информации и обеспечения доступа бизнес-пользователей к знаниям*»;
c) позволяет представлять и систематизировать «*знания о бизнесе, добытые в результате углубленного анализа детальных данных и консолидированной информации*».

При этом в работах [40, 113] отмечается необходимость разработки «CASE-*инструментария, ориентированного на рассматриваемую предметную область*», а также то, что «*такой инструментарий должен обеспечивать*:

1) *регистрацию информации по бизнес-процессам*;
2) *продуцирование высокоуровневых представлений бизнес-процессов*;
3) *сопровождение репозитария*;
4) *контроль синтаксиса описания бизнес-процессов*;

5) *контроль его полноты и состоятельности*;
6) *анализ и верификацию описаний процессов и формирование соответствующих отчетов*;
7) *продуцирование спецификаций бизнес-процессов*».

Хорошо видно, что реализация при разработке CASE-инструментария требований **1)–7)** приводит к удовлетворению условий **b)** и **c)** его соответствия Business Intelligence.

Кроме того, в статье [25] отмечается, что наименьший вред организации принесет инструментарий моделирования, «*лишающий разработчика той части «творческих» возможностей, которые ведут к разнообразию представления организационных моделей*». При этом степень соответствия этому требованию инструментария, использующего нотацию SADT (IDEF0), оценивается как крайне низкая. Последнее требование непосредственным образом связано с тем, что инструментарий моделирования должен быть средством поддержки принятия решений, а не художественного творчества. Таким образом, реализация этого требования при разработке CASE-инструментария приводит к выполнению условия **a)** его соответствия Business Intelligence.

Следовательно, можно утверждать, что в настоящее время происходит эволюция части CASE-средств, предназначенных для анализа и моделирования. И в результате этой эволюции CASE-инструментарий моделирования бизнес-систем становится средством для Business Intelligence, т.е. BI-инструментарием.

Рассмотрим инструментарий моделирования бизнеса, олицетворяющий этот эволюционный процесс на практике, т.е. программный инструментарий, автоматизирующий УФО-анализ («UFO-toolkit»). Хотя название данного инструмента легко ассоциируется с аббревиатурой УФО (как и было задумано), в данном случае «UFO» – есть сокращение словосочетания «*User Familiar Object*» из английского компьютерного словаря, означающего «знакомый пользователю объект» (что также полностью соответствует сути дела).

Данный инструментарий, обеспечивающий проведение системологического (системно-объектного) анализа с помощью концептуальных классификационных моделей, является принципиально новым CASE-средством, впервые представляющим собой программную систему, основанную на знаниях.

Условия работы пользователей CASE-средствами (менеджеров, специалистов по консалтингу, аналитиков, инженеров по знаниям, специалистов по информационным технологиям, архитекторов проектов, разработчиков программного обеспечения), как правило, характеризуются следующим образом:

- преобладание сложных слабоформализованных задач в слабоструктурированных проблемных областях;
- эвристическая природа методов декомпозиции сложных систем и выбора набора абстракций для ООА и моделирования;
- сложность взаимосогласованного учета структурных, субстанциальных и функциональных характеристик систем в динамических моделях;
- отсутствие на рынке «хороших» (подходящих для таких условий) инст-

рументов и высокая цена «удовлетворительных».

Проблемы, с которыми эти пользователи сталкиваются при осуществлении своей профессиональной деятельности следующие:

- отсутствие инструментальных средств системного анализа, которые могут быть применены при объектно-ориентированном проектировании (OOD);
- отсутствие инструментальных средств OOA и OOD, обеспечивающих решение задачи выбора подходящего набора абстракций для объектной декомпозиции и моделирования;
- отсутствие инструментальных средств OOA и OOD, адаптирующихся к предметной области решаемой задачи, т.е. учитывающих семантику предметной области;
- недостаточный уровень автоматизации аналитической деятельности.

Таким образом, наиболее общие и фундаментальные потребности аналитиков и проектировщиков, использующих системный анализ, CASE-технологию и объектно-ориентированный подход, могут быть сформулированы следующим образом:

- производить процедуры анализа и синтеза сложных динамических систем наиболее объективным образом;
- адекватно и взаимосвязано представлять структуру, состав элементов и функций моделируемых систем;
- имитировать функционирование системы на ее объектной модели;
- снизить трудоемкость проектирования (за счет уменьшения числа создаваемых моделей, а также, в частности, за счет единообразного (одними и теми же средствами) представления различных (внешних и внутренних) аспектов организационных систем (бизнес-процессов));
- иметь возможность учета семантики предметной области и семантического взаимодействия с инструментальным средством;
- использовать компонентные технологии при анализе и моделировании.

Описанные выше проблемы и потребности пользователей обусловили основное предназначение инструмента UFO-toolkit и его функциональные возможности. В самых общих чертах инструмент обеспечивает представление любой системы (а также любой ее подсистемы и т.д.) в виде трехэлементной конструкции: «Узел – Функция – Объект» (УФО-элемент). Для выполнения названной функции UFO-toolkit поддерживает классификацию (библиотеку) УФО-элементов, основанную на классификации связей, пересечения которых и образуют «узлы». Таким образом, назначение UFO-toolkit может быть представлено виде UML-диаграммы вариантов использования как показано ниже (см. рис. 5.14).

Рис. 5.14. Диаграмма вариантов использования UFO-toolkit.

Использование UFO-toolkit для моделирования предполагает предварительную специализацию классификации связей и УФО-библиотеки с учетом конкретной предметной области. Это позволяет создавать шаблоны классификаций и библиотеки для различных предметных областей, которые будут обеспечивать процесс моделирования множествами алфавитных УФО-элементов.

Использование инструмента непосредственно для моделирования организационных систем осуществляется следующим образом:

- Построение контекстной модели (объектной К-модели самого верхнего уровня иерархии) анализируемой/проектируемой системы в виде «черного ящика» с указанием входных и выходных связей (функциональных), которые должны быть представлены в классификации связей. Процесс может начинаться либо со стороны классификации связей, либо со стороны контекстной модели.
- Выявление функциональных узлов в структуре моделируемой системы, т.е. узлов, функция которых либо уже известна, либо может быть сформулирована в результате проектирования. Для этого используется таблица, в которой строки обозначаются входными связями, а столбцы – выходными. Эти связи также должны быть представлены в классификации связей. При этом процесс начинается с функциональных связей, показанных на контекстной модели. Если для моделирования используются шаблоны (алфавит) УФО-элементов, то обеспечивается возможность автоматической идентификации известных Инструменту узлов. Если шаблоны не используются или в них нет необходимых в данном случае узлов, то в таблицу добавляются внутренние связи, поддерживающие функциональные, показанные на контекстной модели. Этот процесс продолжается до тех пор, пока не будет обеспечен баланс «втекающих» и «вытекающих» потоков/связей контекстной модели. При этом ручная идентификация узлов должна сопровождаться модификацией классификации связей и УФО-библиотеке.
- Построение иерархической объектной О-модели анализируемой или проектируемой системы. Данная модель в процессе анализа/проектирования представляет собой (на каждом уровне иерархии) совокупность взаимосвязанных функциональных узлов, идентифицированных с помощью таблицы. При этом каждому узлу должна назначаться функция (из всех известных и хранимых в УФО-библиотеке вариантов) в максимально возможной степени точно балансирующая данный узел. Для каждой же функции, в конце концов, должен быть указан объект (из всех известных и хранимых в УФО-библиотеке вариантов), реализующий ее оптимальным с точки зрения данного проекта образом. В результате, следовательно, объектная модель системы представляет собой (на каждом уровне иерархии) совокупность взаимосвязанных функциональных объектов.
- Имитация функционирования системы. Данный процесс обеспечивается путем анимации объектной О-модели, которая сводится к визуализации изменения во времени (с возможностью его масштабирования) активностей функциональных объектов.

Схема функционирования UFO-toolkit представлена в виде UML-диаграммы деятельности на рис. 5.15.

В инструменте обеспечена возможность учета при анимации характеристик самих объектов (например: надежности), а также возможность подсчета показателей, позволяющих сравнивать различные варианты объектной модели с точки зрения выбранных для данных узлов функций и для данных функций – объектов (например: временных, стоимостных).

Кроме того, для функций, которые могут быть описаны математически, существует возможность вычисления их значений в ходе имитации, с использованием механизма скриптов. Целью имитации является обнаружение тормозящих или простаивающих связей, функций или объектов, а также определение наилучших структуры и состава проектируемой системы.

5.3.2. Особенности функционирования «UFO-toolkit»

Рассматриваемое программное средство поддержки УФО-технологии системологического (системно-объектного) анализа и моделирования представляет собой CASE-инструмент категории toolkit, использующий базу знаний специальной конфигурации. Эта база знаний включает в себя библиотеку УФО-элементов и классификацию связей, имеет сетевую структуру и хранится в формате XML. Единицей хранения в этой базе является УФО-элемент, который, в свою очередь, является основой как для декомпозиции сложной системы на составные части, так и для процедуры синтеза сложной системы из более простых частей. При этом УФО-элементами моделируются не только функциональные компоненты системы, но и компоненты, выполняющие роль связей. Формально УФО-элемент может быть представлен как класс языка объектного моделирования UML (см. рис. 5.16).

UFO-toolkit, решая задачу построения объектных и имитационных моделей сложных динамических (организационных) систем, характеризуется следующими основными принципиальными особенностями:

- значительное снижение трудоемкости проектирования, за счет увеличения степени автоматизации аналитической деятельности;
- повышение объективности анализа и адекватности моделирования;
- использование при анализе и моделировании компонентной технологии, автоматизирующей процесс создания моделей, путем использования готовых (алфавитных) функциональных объектов, представленных в базе знаний инструмента в виде УФО-элементов;
- обеспечение интеллектуального взаимодействия с пользователем, в частности, путем «узнавания» готовых компонент (УФО-элементов).

UFO-toolkit может быть применен в следующих случаях:

- Построение моделей существующего и планируемого бизнеса при проведении реинжиниринга бизнес-процессов.
- Выполнение консалтинговых проектов.
- Разработка распределенных информационных систем с применением средства бизнес-объектов CORBA (BOF).

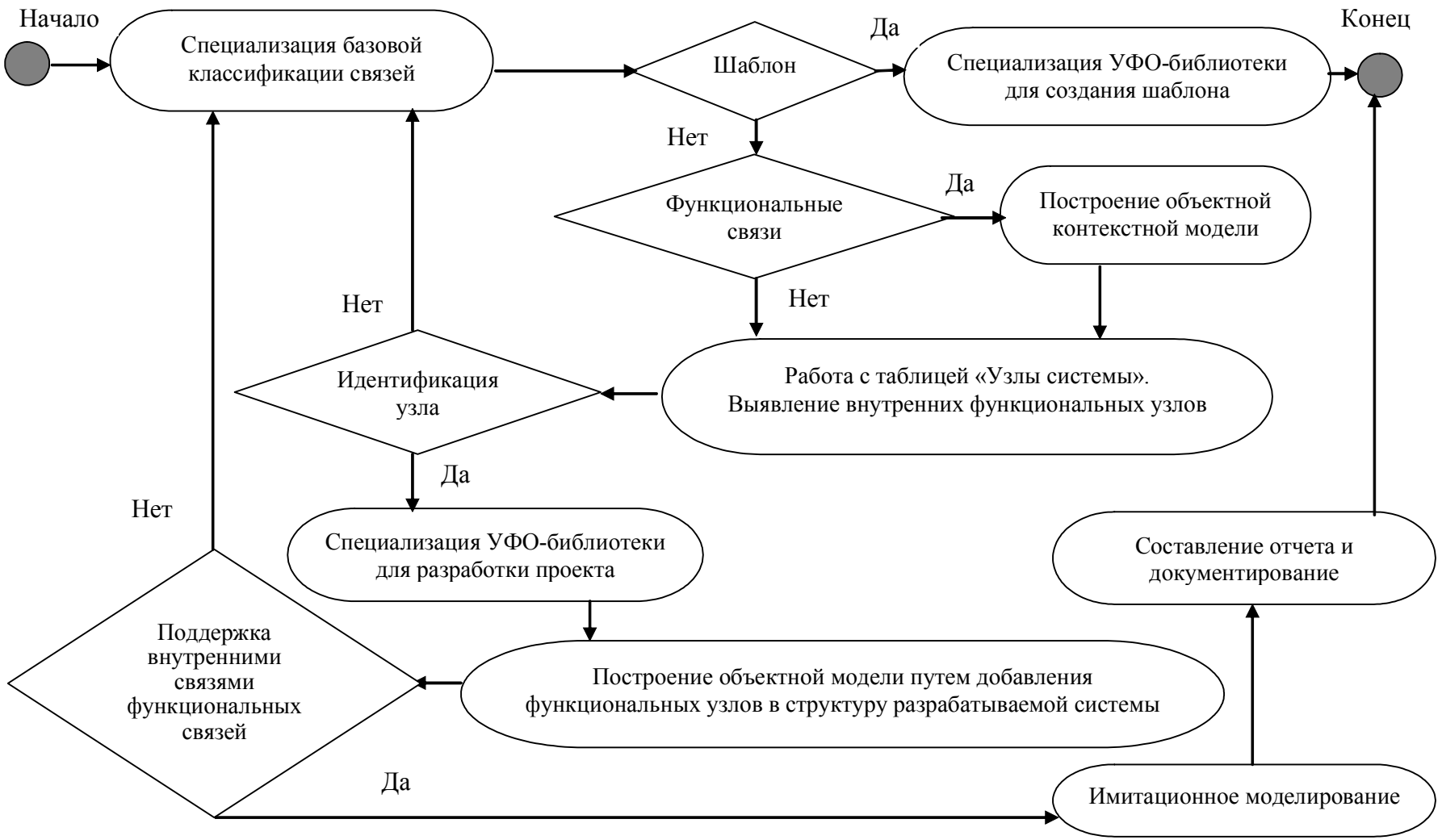


Рис. 5.15. - UML-диаграмма деятельности (функционирования) UFO-toolkit.

- Разработка технических систем с применением CALS-технологии и системы стандартов STEP.

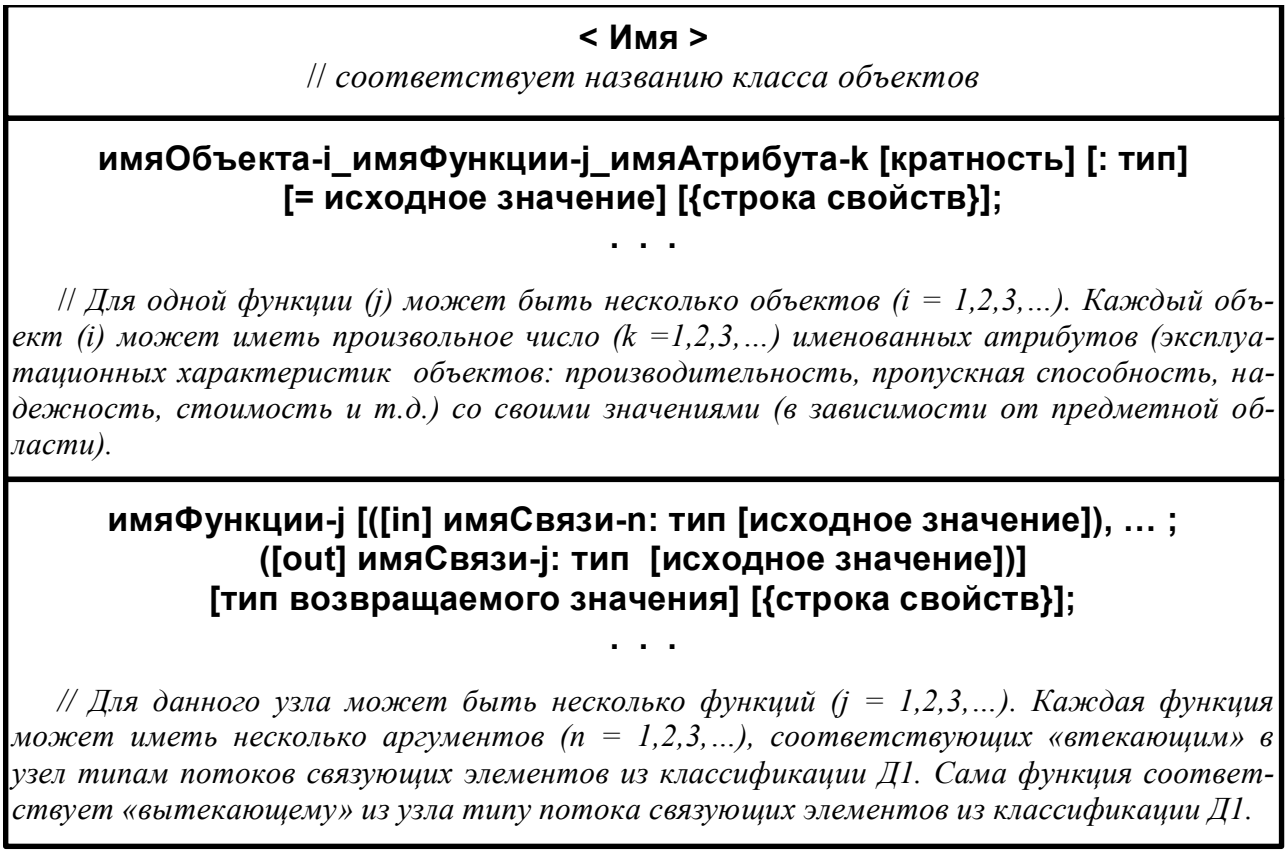


Рис. 5.16. - УФО-элемент, как класс языка UML

Наиболее существенные возможности инструмента:

- Объективизация процедур анализа и синтеза сложных динамических систем (в частности, организационных).
- Взаимосвязанные анализ, моделирование и проектирование структуры, состава элементов и функциональных характеристик систем и процессов, в том числе и не имеющих математической интерпретации.
- Возможность объединения различных аспектов рассмотрения системы в одной объектной модели (диаграмме взаимодействия объектов).
- Возможность единообразного построения внешней и внутренней моделей бизнес-системы, описываемых одним и тем же языком моделирования.
- Обеспечение имитации функционирования проектируемой системы в различных вариантах для определения «заторов» и «простоев», а также расчета различных интегральных показателей.

Впервые в практике системных исследований и объектно-ориентированного подхода к разработке информационных систем решена проблема согласования процедур и результатов системного анализа и OOD.

Предложены методы и средства, реализующие процедуры УФО-анализа сложных динамических объектов. Данные результаты позволяют использовать CASE-средства нового поколения, представляющие собой программные системы, основанные на знаниях.

5.3.3 Технология представление моделей в «UFO-toolkit»

Первой задачей, решаемой с помощью данного инструмента, в соответствии с методологией УФО-анализа, является построение классификации внешних (функциональных) и внутренних (поддерживающих) связей моделируемой системы путем специализации (итеративной) базовой иерархии связей. На основе данной классификации UFO-toolkit обеспечивает представление любой бизнес-системы (подсистемы и т. д.) в виде УФО-элемента, т.е. трехэлементной конструкции «Узел – Функция – Объект».

Применение системологии к анализу и моделированию бизнеса позволяет создать эффективный метод моделирования бизнес-систем, удовлетворяющий требованиям [25, 40, 113].

Например, АООЗТ (акционерное общество очень закрытого типа) «*Рога и копыта*» может быть представлено как УФО-элемент следующим образом.

В целом, как **узел,** АООЗТ является перекрестком связей/потоков представленных на рисунке 5.17 (в наименованиях связей через буквенно-цифровые обозначения показано, как эти связи вписываются в базовую классификацию связей).

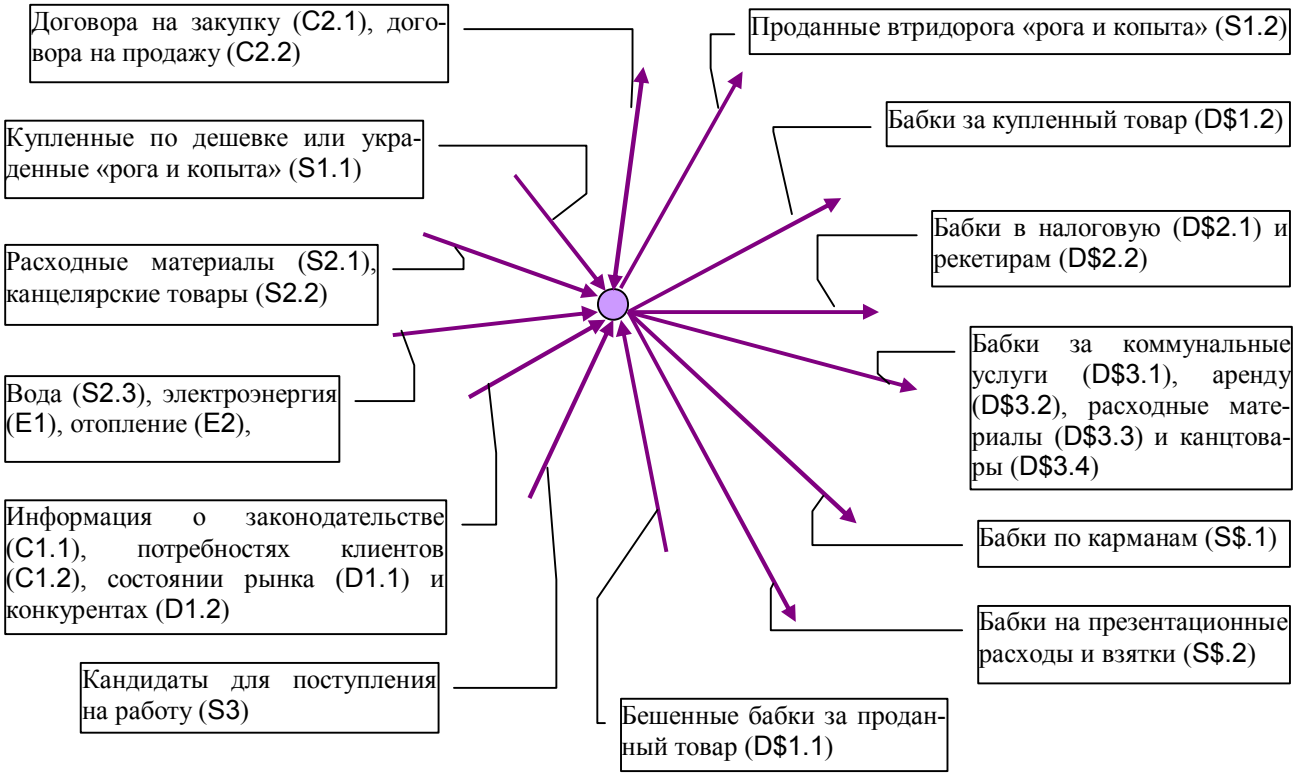


Рис. 5.17. - АООЗТ «*Рога и копыта*» как перекресток входных и выходных связей, т.е. **Узел**.

АООЗТ «*Рога и копыта*» в целом, как **функция,** в самом общем виде может быть представлена в виде процесса, изображенного на рисунке 5.18.

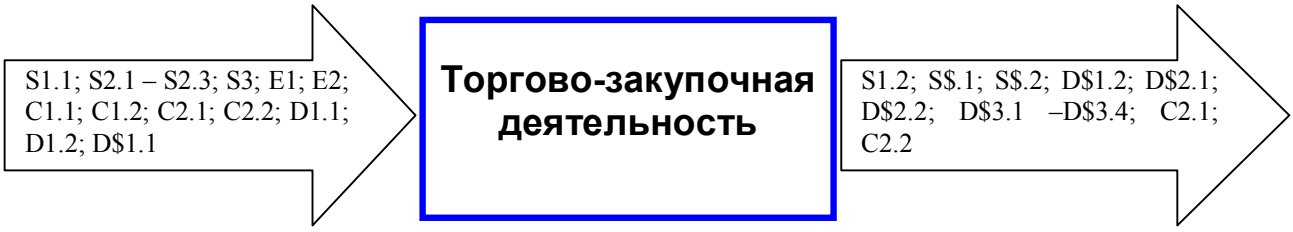


Рис. 5.18. - АООЗТ «*Рога и копыта*» как процесс преобразования входа в выход, т.е. **Функция**.

Естественно, это представление может и должно быть дополнено описанием этой деятельности как процесса такой степени подробности и формальности, которые соответствуют имеющейся информации и целям анализа.

Как **объект** АООЗТ «*Рога и копыта*» в целом может иметь, например, следующие общие характеристики, которые также могут изменяться и дополняться (см. рисунок 5.20):

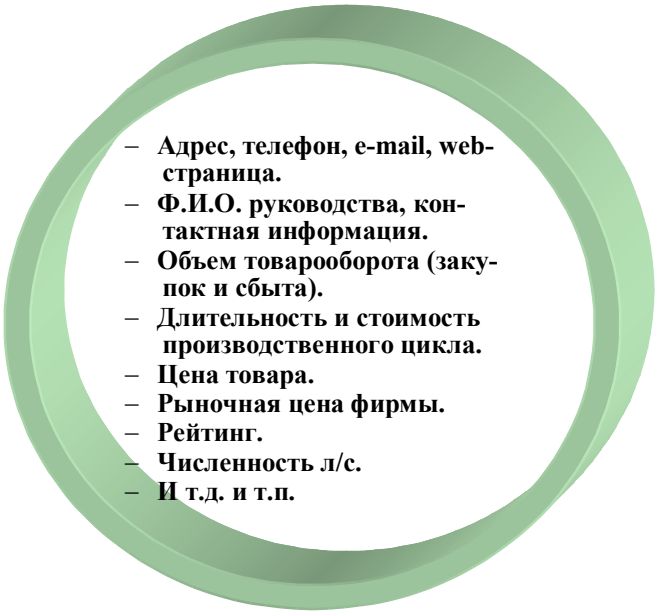


Рис. 5.20. - АООЗТ «*Рога и копыта*» как **объект**, осуществляющий торгово-закупочную деятельность.

Декомпозиция АООЗТ «*Рога и копыта*» на УФО-элементы нижнего уровня может быть осуществлена следующим образом.

С точки зрения **узлов** она может быть представлена, например, как показано на рисунке 5.21. Естественно, на данном уровне декомпозиции появляются новые связи, не использовавшиеся на уровне общего представления АООЗТ, но также включающиеся теперь в классификацию. В данном случае это: C3.1 – Заявки производственников на расходные материалы и канцтовары; C3.2 – Заявки управленцев на расходные материалы, канцтовары и ком-

мунальные услуги; С4 – Контроль со стороны управления за основной и вспомогательной деятельностью; D2.1 – Отчетность производственного отделения; S4 – Ремонтно-технический персонал для обслуживания. Кроме того, поток Е представляет собой объединение потоков Е1 и Е2 (т.е. родовой по отношению к ним поток), а поток S2 – объединение потоков S2.1 – S2.3 (так же родовой поток).

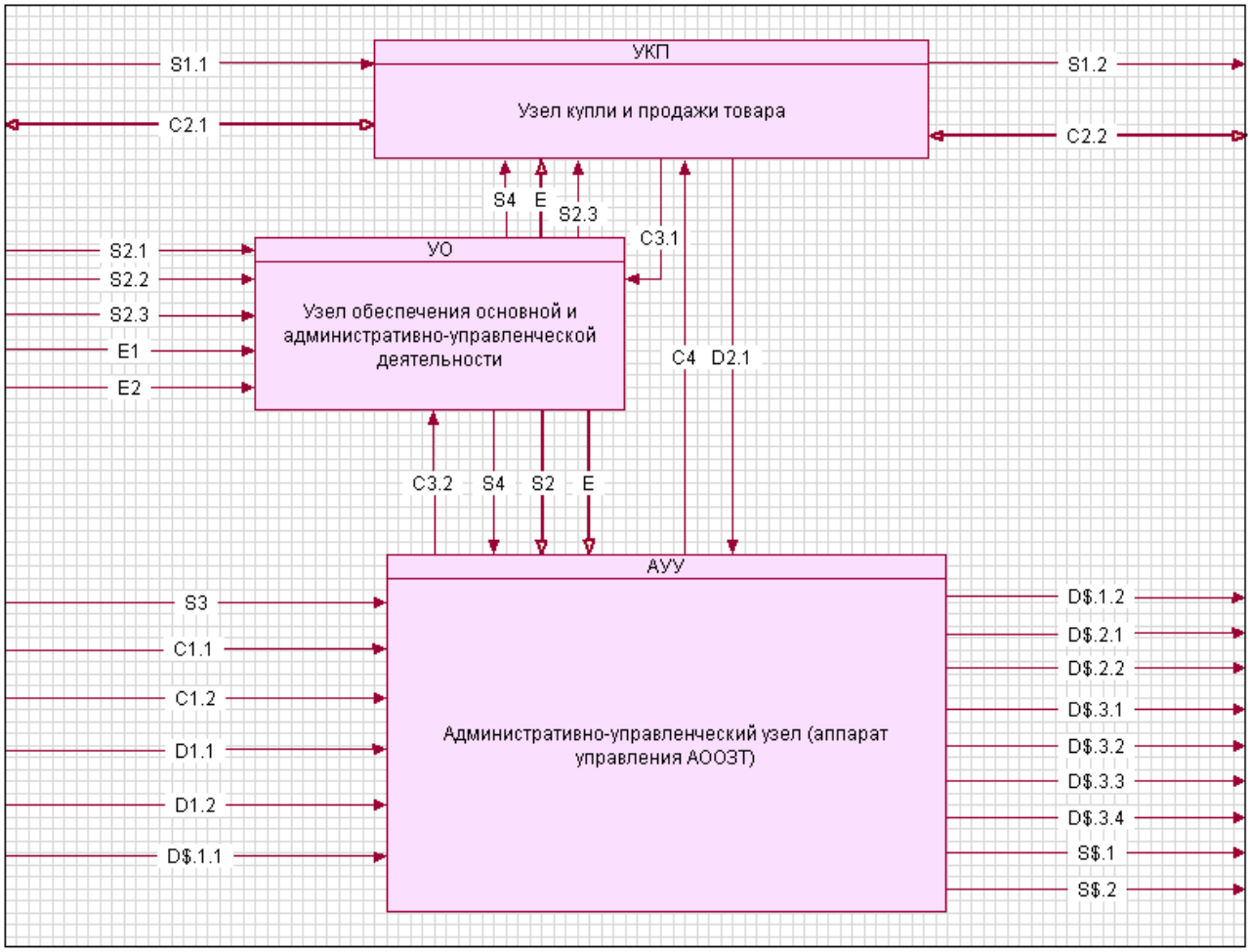


Рис. 5.21.- Декомпозиция узла, занимаемого АООЗТ «*Рога и копыта*».

При этом функция в узле **УКП**, соответствующая **основной деятельности** АООЗТ, может быть описана, например, как «деятельность по приобретению, транспортировке, хранению и сбыту товара». Объектом, который осуществляет эту деятельность фактически, может быть, например, **производственное отделение** (подразделения и должностные лица, занятые куплей, транспортировкой, хранением и сбытом товара).

Функция в узле **УО**, соответствующая **вспомогательной деятельности** АООЗТ, может быть описана, например как «работа по материально-техническому обеспечению основной и административно-управленческой деятельности АООЗТ». Объектом, который осуществляет эту деятельность фактически, может быть, например, **вспомогательное отделение** (подразде-

ления и должностные лица, занятые вспомогательной, обеспечивающей деятельностью).

Функция в узле **АУУ**, соответствующая **управленческой деятельности** АОЗТ, может быть описана, например как «работа по координации и организации деятельности торгово-закупочного предприятия». Объектом, который осуществляет эту деятельность фактически, может быть, например, **управление** (подразделения и должностные лица, занимающиеся организацией, планированием, контролем, учетом, отчетностью, а также кадровыми вопросами).

Дальнейшая декомпозиция на УФО-элементы более глубоких ярусов (на узлы функциональные и функциональные объекты) осуществляется до тех пор, пока не будут выявлены узлы, функции в которых имеют четкое (по возможности формализованное) описание, позволяющее сформировать либо положения о конкретных подразделениях, либо инструкции конкретным должностным лицам. При этом должна быть гарантирована возможность существования объектов (людей, подразделений или технических средств), способных выполнить упомянутые функции.

Общее представление об интерфейсе UFO-toolkit отображено на экранной форме (см. рисунок 5.22), показывающей вариант декомпозиции узла купли и продажи товара (УКП) АООЗТ «*Рога и копыта*».

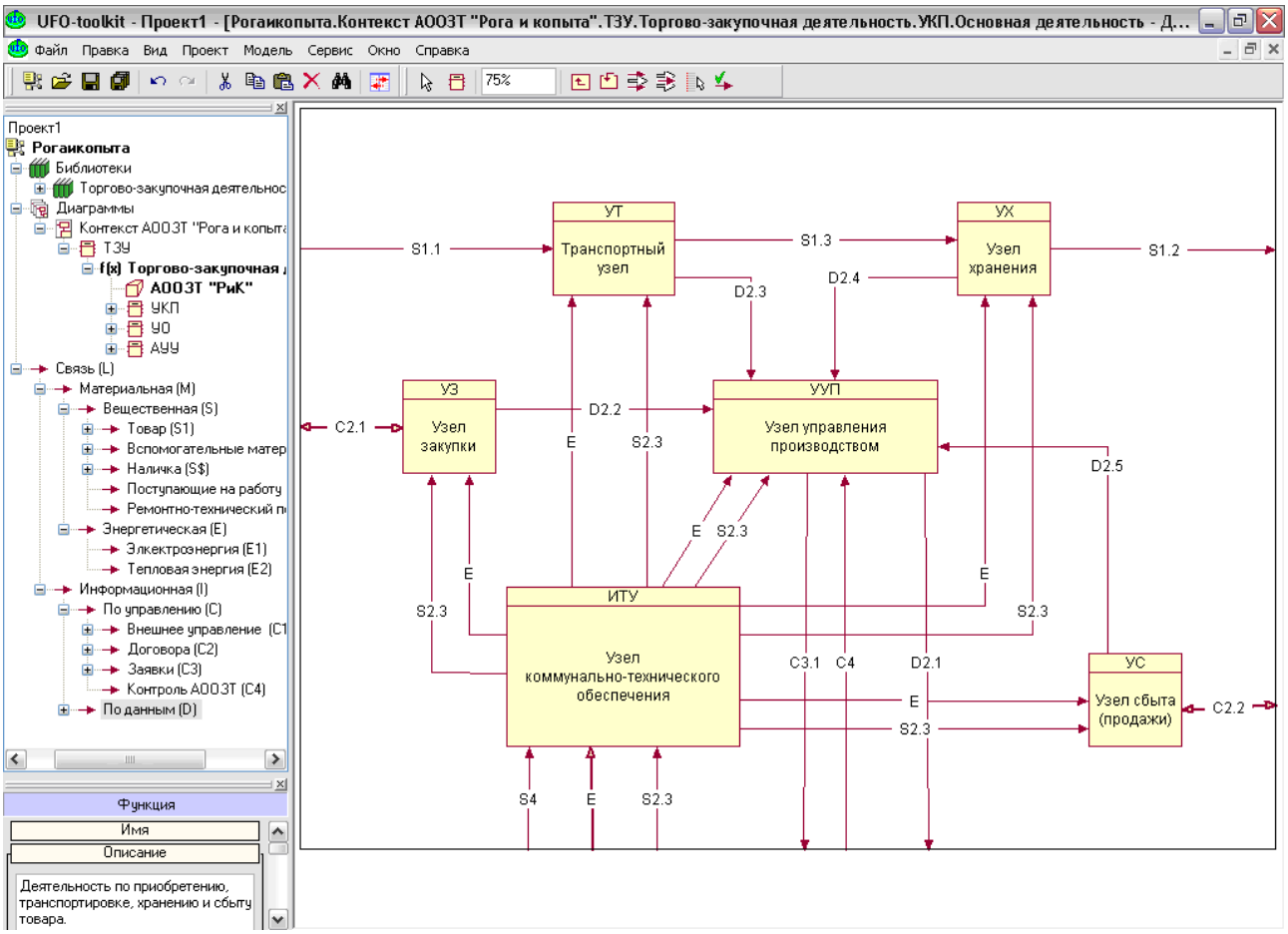


Рис. 5.22. - Пример экранной формы UFO-toolkit.

Практически могут заранее разрабатываться, сохраняться в библиотеке (репозитарии) и затем использоваться УФО-элементы, моделирующие различные составные части бизнес-системы. Например, для рассматриваемого АООЗТ, могут быть использованы модели в виде УФО-элементов (если они заранее разрабатывались) бухгалтерии, отдела снабжения, отдела сбыта, отдела кадров и т.д. Сборка **УФО-модели** бизнес-системы из готовых УФО-элементов осуществляется путем использования правил комбинирования УФО-элементами, т.е. **правил системной декомпозиции** (см. пп. 5.1.2).

При этом обеспечивается учет в одной модели и структурных, и функциональных, и объектных (субстанциальных) характеристик бизнеса (любой системы вообще). Кроме того, представленный метод впервые в практике системного анализа и объектно-ориентированного проектирования информационных систем решает проблему согласования их процедур и результатов.

Компонентный подход к моделированию бизнеса позволяет в значительной степени автоматизировать аналитическую деятельность за счет формирования и использования библиотек (репозитариев) УФО-элементов, а также за счет использования формальных правил сборки конфигураций из этих элементов (построения диаграмм).

На концептуальном уровне библиотеки представляют собой концептуальные модели соответствующих отраслей или частей бизнеса, в которых хранятся их структурные, функциональные и субстанциальные (объектные) характеристики. При этом библиотеки могут содержать не только одиночные УФО-элементы, но и их иерархии, что позволяет повторно использовать готовые подсистемы и системы. Таким образом, библиотеки представляют собой базу знаний специальной конфигурации, в которой хранятся УФО-элементы, соответствующие определенным классам бизнес-систем.

На логическом уровне библиотеки представляют собой фасетные классификации УФО-элементов, основанные на классификации связей. UFO-toolkit строит эти фасетные классификации автоматически, используя заданные диаграммы взаимодействия. Автоматическое построение библиотек на основе модели системы в значительной степени упрощает повторное использование УФО-элементов.

Использование библиотек позволяет накапливать, систематизировать и передавать знания о конкретных предметных областях. Решение конкретной задачи с использованием готовой библиотеки представляет собой достаточно простой и формализованный процесс, что позволяет рассматривать библиотеки УФО-элементов как самостоятельные, имеющие потребительскую стоимость продукты.

Таким образом, можно утверждать, что метод УФО-анализа является знаниеориентированным, а также системным и объектным одновременно.

Как знаниеориентированный, системно-объектный CASE/BI-инструмент нового поколения, UFO-toolkit обладает рядом преимуществ в сравнении, например с BPwin, поскольку позволяет накапливать, систематизировать и использовать в дальнейшем знания о предметных областях, а также полноценно использовать результаты системного анализа бизнеса в ходе объектно-

ориентированного проектирования информационной системы.

Сравнение BPwin и UFO-toolkit (с использованием данных о SADT (IDEF0) и DFD нотациях из работ [20, 32]) представлено в таблице.

Таблица. 5.5. Сравнение BPwin и UFO-toolkit

UFO-toolkit	**BPwin**
Использование формализованных средств (правил) для построения и модификации визуальных графоаналитических моделей, что существенно сокращает разнообразие представления организационных систем (бизнес-систем).	Отсутствие правил и методических рекомендаций по построению моделей организационных систем (бизнес-систем), которые бы сокращали разнообразие получаемых результатов.
Возможность поддержки содержательной классификации связей, что позволяет сориентировать инструмент на любую конкретную предметную область. Модели являются формально-семантическими.	Не имеется средств ориентирования на конкретную предметную область. Модели имеют совершенно формальный характер.
Поддержка компонентной технологии моделирования и проектирования вследствие наличия репозитария/библиотеки, что обеспечивает возможность учета, систематизации и передачи знаний о предметной области.	Отсутствует возможность применение компонентной технологии моделирования, а также возможность учета, систематизации и передачи знаний о предметной области.
УФО-анализ и инструмент UFO-toolkit согласуются с требованиями объектно-ориентированной технологии проектирования информационных систем и позволяют упростить начальные технологические процессы разработки объектных приложений.	Результаты, полученные при моделировании бизнес-процессов в BPwin малопригодны для использования при создании объектно-ориентированного программного обеспечения.
Структурные, функциональные и объектные (субстанциальные) аспекты рассмотрения бизнес-системы объединены в одной системно-объектной УФО-модели.	Все системно-структурные методы, реализованные в BPwin, требуют построения двух или трех моделей одного и того же объекта: функциональной (активностной), информационной (данных), а также динамической.
УФО-анализ обеспечивает автоматизацию построения диаграмм взаимодействия УФО-элементов (декомпозиции) с использованием библиотек по заданной контекстной УФО-модели.	Не существует перспектив автоматизации декомпозиции моделей.
Динамическая модель есть результат активизации (анимации) статической модели взаимодействия объектов системы; привлечения других средств не требуется.	Для создания динамических моделей требуется использование дополнительных специальных расширений или других средств, с которыми технологии, реализованные в BPwin, плохо согласуются.

Вопросы для повторения

1. Каким образом система может быть представлена с трех сторон: как «узел», «функция» и «объект»?
2. Как могут быть расклассифицированы связи между системами с содержательной и формальной точки зрения?
3. Приведите классификацию организационных систем, связанную с классификацией связей.
4. Опишите алгоритм построение моделей систем в виде иерархии «Узлов», «Функций» и «Объектов».
5. Приведите пример описания системы в терминах УФО-подхода.
6. Опишите особенности функционирования пакета «UFO-toolkit» как инструментария визуального графоаналитического моделирования.
7. Почему пакет «UFO-toolkit» может рассматриваться как система, основанная на знаниях?

Резюме по теме

В данном разделе рассмотрена технология системно-объектного моделирования и анализа сложных систем: системологический подход, с помощью которого система представляется в виде трехсторонней конструкции «Узел-Функция-Объект»; алгоритм построения УФО-моделей и средства формализации системно-объектного УФО-анализа; а также CASE-инструментарий системно-объектного моделирования и анализа (UFO-toolkit).

Тема 6. Графический язык моделирования бизнес-процессов BPMN.

Цели и задачи изучения темы

Целью изучения данной темы является теоретическое и практическое освоение технологии моделирования и анализа бизнес-процессов с помощью языка графического моделирования BPMN.

При этом ставятся следующие задачи:

- изучение спецификации BPMN;
- изучение правил построения диаграмм бизнес-процессов (BPD) в нотации BPMN;
- изучение CASE-инструментария построения BPD-диаграмм.

6.1. Назначение и область применения.

«Нотация моделирования бизнес-процессов (Business Process Modeling Notation (BPMN))» разработана инициативной группой по управлению бизнес-процессами (Business Process Management Initiative (BMPI)), которая проанализировала опыт многих существующих нотаций и попробовала объединить лучшие концепции разных нотаций в одну стандартную. Основное назначение получившегося стандарта - обеспечение пользователей нотацией, понятной всем участникам бизнес-сферы, от бизнес-аналитиков, создающих первоначальные эскизы процессов, технических разработчиков, ответственных за внедрение технологии, в которой будут представлены данные процессы, и, наконец, до бизнесменов, которые будут управлять этими процессами. Таким образом, BMPN является стандартизованным связующим звеном между разработкой бизнес-процессов и их реализацией.

Другой не менее важной целью создания стандарта бизнес-моделирования является визуализация бизнес-процессов посредством бизнес-ориентированного подмножества языков XML (например, такого как BPEL (Business Process Execution Language – язык выполнения бизнес процессов)), разработанных для их исполнения.

Слияние концерна OMG с группой BMPI и его активная поддержка графической нотации BPMN свидетельствуют о признании объектно-ориентированном сообществом фактической непригодности использования языка UML для моделирования бизнес-процессов.

6.2. Диаграммы бизнес-процессов (BPD).

Одной из причин создания BPMN явилась необходимость использования единого простого механизма для проектирования как простых, так и сложных моделей бизнес-процессов. Для удовлетворения этих двух противоречащих друг другу требований была осуществлена систематизация графических элементов по категориям. Результатом явился небольшой перечень категорий элементов графической нотаций, позволивший людям, работающим с диаграммами BPMN, без труда распознавать основные типы элементов и

осуществлять корректное чтение схем (BPD-диаграмм). Основные категории элементов допускают внутренние вариации, а также добавление информации для удовлетворения требований сложности без внесения значительных изменений в общую структуру диаграммы для легкости её понимания.

Существуют четыре основные категории элементов BPMN [114]:

• *Элементы потока* (Flow Objects);

• *Соединяющие элементы* (Connecting Objects);

• *Зоны ответственности* (Swimlanes);

• *Артефакты* (Artifacts).

Элементы потока являются важнейшими графическими элементами, определяющими ход бизнес-процесса. Они, в свою очередь, делятся на:

• *События* (Events);

• *Действия* (Activities);

• *Шлюзы* (Gateways).

Выделяют три вида Соединяющих элементов:

• *Поток операций* (Sequence Flow);• *Поток сообщений* (Message Flow); • *Ассоциация* (Association).

Используются два способа группировки основных элементов моделирования с помощью Зон ответственности: • *Группировка с помощью Пула* (Pool); • *Группировка с помощью Дорожки* (Lane).

Артефакты используются для добавления дополнительной информации о процессе. Выделяют три типовых Артефакта, что, однако, не запрещает разработчикам моделей бизнес-процессов либо программам моделирования добавлять необходимое количество Артефактов. Для широкого круга пользователей, а также для вертикальных рынков существует возможность стандартизации более полного перечня Артефактов. Таким образом, текущий перечень Артефактов включает в себя следующие элементы:

• *Объект данных* (Data object); • *Группа* (Gruop); • *Аннотация* (Annotation).

6.2.1. Элементы потока.

В таблице 6.1 представлены графические элементы BPMN, относящиеся к категории «Элементы потока», которые обозначают различные события.

Таблица 6.1. Графические элементы BPMN «Событие».

Описание элемента	***Основные графические элементы***
СОБЫТИЕ (EVENT) Событие – это то, что происходит в течение бизнес-процесса и оказывает влияние на его ход. Событие имеет причину (триггер) или воздействие (результат), маркеры которых представлены на рисунке 6.1. Согласно влиянию Событий на ход бизнес-процесса, выделяют три типа: Стартовое событие (Start), Промежуточное событие (Intermediate) и Конечное событие (End).	Start Intermediate End

Следующий рисунок поясняет смысл маркеров (триггеров) событий.

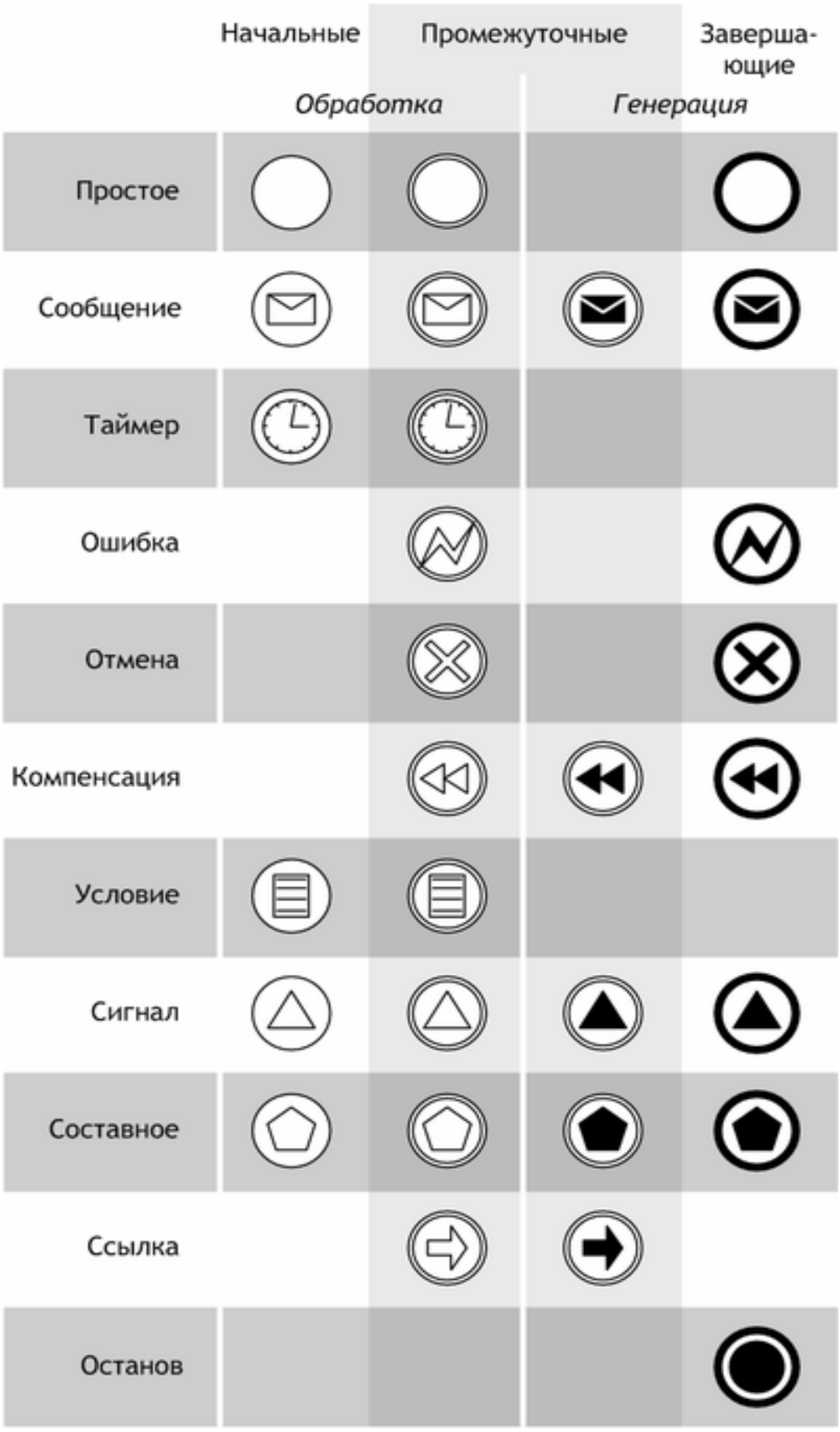
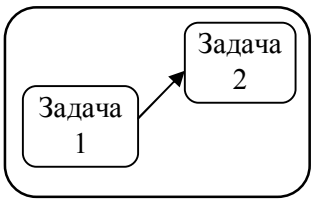
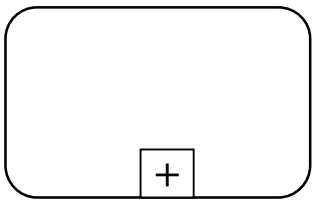
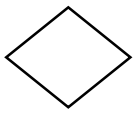


Рис. 6.1. - Маркеры событий.

В таблице 6.2 представлены графические элементы BPMN, относящиеся к категории «Элементы потока», которые обозначают различные действия и операции с потоками.

Таблица 6.2. Графические элементы BPMN «Действие» и «Шлюз».

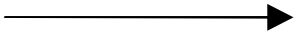
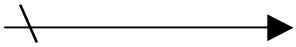
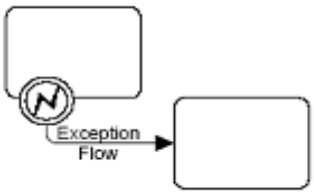
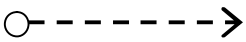
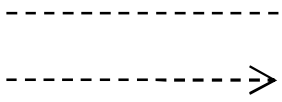
Описание элемента	*Основные графические элементы*
ДЕЙСТВИЕ (ACTIVITY) Действие – общий термин, обозначающий работу, выполняемую исполнителем. Действия могут быть либо элементарными, либо неэлементарными (составными). Выделяют следующие виды действий, являющихся частью модели Процесса: Процесс (Process), Подпроцесс (Sub-Process) и Задача (Task). Задача и Подпроцесс изображаются в виде прямоугольника с закругленными углами. Процесс либо не имеют границ, либо находятся внутри Пула. Для свернутых подпроцессов могут применяться следующие маркеры: ↻- циклический подпроцесс; ‖ - многоэкземплярный подпроцесс; ~ - подпроцесс ad-hoc; ◀◀ - компенсирующий подпроцесс	 Развернутый подпроцесс  Свернутый подпроцесс
ШЛЮЗ (GATEWAY) Шлюзы используются для контроля расхождений и схождений потока операций. Таким образом, данный термин подразумевает ветвление, раздвоение, слияние и соединение маршрутов. Внутренние маркеры указывают тип контроля развития бизнес-процесса: ✖ - эксклюзивное ИЛИ (XOR); ○ - ИЛИ (OR); ✚ - И (AND); ✱- Комплексные/сложные (Complex); ◉ - основано на событиях (Event-based). Шлюзы каждого из типов оказывают влияние, как на входящие, так и на исходящие потоки.	

6.2.2. Соединяющие элементы.

В таблице 6.3 представлены графические элементы BPMN, относящиеся к категории «Соединяющие элементы», которые обозначают различные потоки, связывающие между собой события, действия и шлюзы.

Таблица 6.3. Соединяющие графические элементы BPMN.

Описание элемента	*Основные графические элементы*
ПОТОК ОПЕРАЦИЙ (SEQUENCE FLOW) Поток операций служит для отображения того порядка, в котором организованы действия Процесса:	**СМ. НИЖЕ**

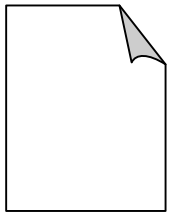
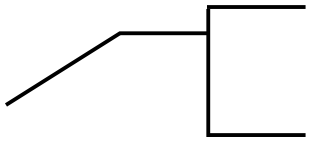
Описание элемента	*Основные графические элементы*
Стандартный поток (Normal Flow)	
Неконтролируемый поток (Uncontrolled Flow) Неконтролируемый поток операций относится либо к потокам, на которые не воздействую никакие условия, либо к потокам, не проходящим через Шлюзы.	
Условный поток (Conditional Flow) Поток операций может зависеть от условных выражений, оценивающихся согласно времени выполнения для того, чтобы определить, будет ли использоваться поток или нет.	
Поток по умолчанию (Default Flow) Поток операций данного типа используется в том случае, если все остальные исходящие Условные потоки операций не являются верными во время выполнения действия.	
Исключающий поток (Exception Flow) Поток исключений встречается за пределами Стандартного потока операций. Основывается на Промежуточных событиях, возникающих в ходе Процесса.	
Поток сообщений (Message Flow) Поток сообщений служит для отображения обмена сообщениями между двумя участниками, готовыми эти сообщения отсылать и принимать. На диаграмме BPMN два отдельно взятых Пула представляют собой двух участников процесса (бизнес-объекты или бизнес-роли).	
Ассоциация (Association) Ассоциация служит для установления связи между информацией и элементами потока. Текстовые объекты, а также графические объекты, не относящиеся к элементам потока, могут соотноситься с Элементами потока.	

6.2.3. Зоны ответственности и артефакты.

В таблице 6.4 представлены графические элементы BPMN, относящиеся к категориям «Зоны ответственности» и «Артефакты».

Таблица 6.4. Элементы BPMN «Зоны ответственности» и «Артефакты».

Описание элемента	*Основные графические элементы*
Пул (Pool) Пул представляет собой Участника Процесса. Он также может выступать в качестве Зоны ответственности или графического контейнера, отвечающего за разделение определенного набора действий.	

Описание элемента	*Основные графические элементы*
ДОРОЖКА (LANE). Дорожка – это подраздел в пределах пула, его протяженность равна длине пула, как по вертикали, так и по горизонтали. Дорожки организовывают и классифицируют действия.	
ОБЪЕКТ ДАННЫХ (DATA OBJECT) Объект данных относится к Артефактам, т.к. не оказывает непосредственного влияния ни на Поток операций, ни на Поток сообщений. Однако Объект данных предоставляет информацию о том, какие действия необходимо выполнить и/или каков результат этих действий.	
ГРУППА (GRUOP) Группировка действий, не оказывающая влияние на последовательный поток. Группировка может использоваться для документирования или анализа. Группы могут так же использоваться для обозначения действий распределенных групповых операций, расположенных по ширине областей.	
АННОТАЦИЯ (ANNOTATION) Текстовая аннотация – это возможность для разработчика привести дополнительную информацию для читателя схемы BPMN.	

Текстовые аннотации элементов BPD-диаграм в BPMN-нотации могут отражать дополнительную информацию о процессе или атрибутах элементов в рамках процесса [114].

- У элементов и потоков МОГУТ быть признаки (например, название и/или другие атрибуты), находящиеся внутри формы, а также над, либо под формой, в любом направлении и месте, в зависимости от пожеланий разработчика или продавца инструмента моделирования.
- Заливка, предназначенная для графических объектов, МОЖЕТ быть белой или прозрачной.
- МОГУТ использоваться другие цвета заливки в соответствии с целями разработчика или продавца инструмента моделирования (например, в целях подчеркивания значимости атрибута объекта).
- Элементы схемы и маркеры МОГУТ быть любого размера, в соответствии с целями разработчика или инструмента моделирования.
- Линии, используемые для рисования графических объектов, МОГУТ быть черными.
- МОГУТ использоваться другие цвета линии в соответствии с целями разработчика или инструмента (например, в целях подчеркивания значимости атрибута объекта).
- МОГУТ использоваться другие стили линии в соответствии с целями

разработчика или инструмента (например, в целях подчеркивания значимости атрибута объекта) при условии, что стиль линии НЕ ДОЛЖЕН вступать в конфликт ни с одним стилем линии, определенным BPMN. Таким образом, НЕ ДОЛЖНЫ изменяться стили линий последовательного потока, потока сообщений и ассоциаций.

6.2.4. Правила соединения Элементов потока.

Входящий Поток операций может быть присоединен к любой точкой Элемента потока (слева, справа, сверху, снизу). Подобно ему, Исходящий поток операций может брать начало из любой точки Элемента потока (слева, справа, сверху, снизу). Поток сообщений обладает теми же свойствами, что и Поток операций.

Для соединения Элементов потока разработчикам моделей настоятельно рекомендуется использовать имеющийся опыт, что облегчит понимание создаваемых диаграмм и сделает ход изображаемого бизнес-процесса прозрачным и доступным для понимания. Это особенно важно в том случае, если в диаграмме присутствуют такие графические элементы, как Поток операций или Поток сообщений. В данном случае оптимальным вариантом является выбор направления Потока сообщений, который необходимо расположить под углом в 90° по отношению к уже выбранному Потоку операций.

Таблица 6.5 содержит изображения Элементов потока, используемые языком BPMN, и указывает, каким образом данные графические элементы соединяются друг с другом посредством Потока операций [114].

Таблица. 6.5. Правила соединения с помощью Потока операций.

От/К						
		↗	↗	↗	↗	↗
		↗	↗	↗	↗	↗
		↗	↗	↗	↗	↗
		↗	↗	↗	↗	↗
		↗	↗	↗	↗	↗

Символ ↗ обозначает, что графический элемент, изображенный в одной из строк таблицы, может соединяться с графическим элементом, изображенным в соответствующей колонке. В таблице не указывается количество входящих и исходящих соединений графического элемента, зависящее от различных конфигураций.

Таблица 6.6 содержит изображения объектов моделирования BPMN, а также указывает, каким образом данные объекты соединяются друг с другом посредством Потока сообщений. Символ ⇗ обозначает, что графический элемент, изображенный в одной из строк таблицы, может соединяться с графическим элементом, изображенным в соответствующей колонке. В таблице не указывается количество входящих и исходящих соединений графического элемента, зависящее от различных конфигураций [114].

Таблица 6.6. Правила соединения с помощью Потока сообщений.

От/К		Name (Pool)	Name	Name		
Name (Pool)	⇗	⇗	⇗	⇗	⇗	
Name	⇗	⇗	⇗	⇗	⇗	
Name	⇗	⇗	⇗	⇗	⇗	
	⇗	⇗	⇗	⇗	⇗	

6.3. Соотношение BPMN, XPDL, BPEL, BPML.

«Знание BPMN без знания BPEL и его места в архитектуре SOA (хотя бы даже в общих чертах) - это вообще незнание» [см., например, http://chevalry.livejournal. com/124062.html]. Действительно, очевидно, что для правильного понимания и использования кокретного языка моделирования необходимы хотя бы общие знания связанных с ним языков.

Для получение же представления о XPDL, BPEL и BPML необходимы знания стандартов, на основе которых эти языки разработаны.

6.3.1. Стандарты SGML и XML

SGML (Standard Generalized Markup Language) — стандартный обобщённый язык разметки. Это метаязык, на котором можно определять язык разметки для документов. Изначально SGML был разработан для совместного использования машинно-читаемых документов в больших правительственных и аэрокосмических проектах. Он широко использовался в печатной и издательской сфере, но его сложность затруднила его широкое распространение для повседневного использования. SGML предоставляет множество вариантов синтаксической разметки для использования различными приложениями. SGML стандартизован ISO: «ISO 8879:1986 Information processing—Text and office systems»

Основные части документа SGML:

- SGML-декларация — определяет, какие символы и ограничители могут появляться в приложении;
- Document Type Definition — определяет синтаксис конструкций разметки. DTD может включать дополнительные определения, такие, как символьные ссылки-мнемоники;
- Спецификация семантики, относится к разметке — также даёт ограничения синтаксиса, которые не могут быть выражены внутри DTD;
- Содержимое SGML-документа — по крайней мере, должен быть корневой элемент.

Пример синтаксиса SGML:

```
<QUOTE TYPE="example">
typically something like <ITALICS>this</ITALICS>
</QUOTE>
```

Создание SGML можно с уверенностью назвать попыткой объять необъятное, так как он объединяет в себе такие возможности, которые крайне редко используются все вместе. В этом и состоит его главный недостаток — сложность и, как следствие, дороговизна этого языка ограничивает его использование только крупными компаниями, которые могут позволить себе купить соответствующее программное обеспечение и нанять высокооплачиваемых специалистов. Кроме того, у небольших компаний редко возникают настолько сложные задачи, чтобы привлекать к их решению SGML.

Наиболее широко SGML применяется для создания других языков разметки, именно с его помощью был создан язык разметки гипертекстовых документов — HTML. Его появление было связано с необходимостью организации стремительно увеличивающегося массива документов в сети Интернет. Бурный рост количества подключений к Интернету и, соответственно, вебсерверов повлек за собой такую потребность в кодировке электронных документов, с которой не мог справиться SGML вследствие высокой трудности освоения. Появление HTML — очень простого языка разметки — быстро решило эту проблему: лёгкость в изучении и богатство средств оформления документов сделали его самым популярным языком для пользователей Ин-

тернет. Но, по мере роста количества и изменения качества документов в Сети, росли и предъявляемые к ним требования, и простота HTML превратилась в его главный недостаток. Ограниченность количества тегов и полное безразличие к структуре документа побудили разработчиков в лице консорциума W3C к созданию такого языка разметки, который был бы не столь сложен, как SGML, и не настолько примитивен, как HTML. В результате на свет появился язык XML, сочетающий в себе простоту HTML, логику разметки SGML и удовлетворяющий требованиям Интернета. Таким образом, HTML — это приложение SGML, а XML — это подмножество SGML, разработанное для упрощения процесса машинного разбора документа.

XML (**Extensible Markup Language**) - это новый SGML-производный язык разметки документов, позволяющий структурировать информацию разного типа, используя для этого произвольный набор инструкций.

XML-документ представляет собой обычный текстовый файл, в котором при помощи специальных маркеров создаются элементы данных, последовательность и вложенность которых определяет структуру документа и его содержание. Основным достоинством XML документов является то, что при относительно простом способе создания и обработки (обычный текст может редактироваться любым тестовым процессором и обрабатываться стандартными XML-анализаторами), они позволяют создавать структурированную информацию, которую хорошо "понимают" компьютеры.

При создании собственного языка разметки вы можете придумывать любые названия элементов, соответствующих контексту их использования. В этом и заключается гибкость и расширяемость XML-производных языков - они создаются разработчиком "на лету", согласно его представлениям о структуре документа, и могут затем использоваться универсальными программами просмотра наравне с любыми другими XML-производными языками, т.к. вся необходимая для синтаксического анализа информация заключена внутри документа.

Создавая новый формат, необходимо учитывать тот факт, что документов, "написанных на XML", не может быть в принципе - в любом случае авторы документа для его разметки используют основанный на стандарте XML (т.н. XML-производный) язык, но не сам XML. Поэтому при сохранении созданного файла можно выбрать для него какое-то подходящее названию расширение (например, noteML).

Сегодня XML может использоваться в любых приложениях, которым нужна структурированная информация - от сложных геоинформационных систем, с гигантскими объемами передаваемой информации до обычных "однокомпьютерных" программ, использующих этот язык для описания служебной информации. При внимательном взгляде на окружающий нас информационный мир можно выделить множество задач, связанных с созданием и обработкой структурированной информации, для решения которых может использоваться XML:

– В первую очередь, эта технология может оказаться полезной для разработчиков сложных информационных систем, с большим количеством

приложений, связанных потоками информации самой различной структурой. В этом случае XML-документы выполняют роль универсального формата для обмена информацией между отдельными компонентами большой программы.

- XML является базовым стандартом для нового языка описания ресурсов, RDF, позволяющего упростить многие проблемы в Web, связанные с поиском нужной информации, обеспечением контроля за содержимым сетевых ресурсов, создания электронных библиотек и т.д.
- Язык XML позволяет описывать данные произвольного типа и используется для представления специализированной информации, например химических, математических, физических формул, медицинских рецептов, нотных записей, и т.д. Это означает, что XML может служить мощным дополнением к HTML для распространения в Web "нестандартной" информации. Возможно, в самом ближайшем будущем XML полностью заменит собой HTML, по крайней мере, первые попытки интеграции этих двух языков уже делаются (спецификация XHTML).
- XML-документы могут использоваться в качестве промежуточного формата данных в трехзвенных системах. Обычно схема взаимодействия между серверами приложений и баз данных зависит от конкретной СУБД и диалекта SQL, используемого для доступа к данным. Если же результаты запроса будут представлены в некотором универсальном текстовом формате, то звено СУБД, как таковое, станет "прозрачным" для приложения. Кроме того, сегодня на рассмотрение W3C предложена спецификация нового языка запросов к базам данных XQL, который в будущем может стать альтернативой SQL.
- Информация, содержащаяся в XML-документах, может изменяться, передаваться на машину клиента и обновляться по частям. Разрабатываемые спецификации XLink и Xpointer поволят ссылаться на отдельные элементы документа, с учетом их вложенности и значений атрибутов.
- Использование стилевых таблиц (XSL) позволяет обеспечить независимое от конкретного устройства вывода отображение XML-документов.
- XML может использоваться в обычных приложениях для хранения и обработки структурированных данных в едином формате.

6.3.2. XPDL

XPDL (XML Process Definition Language) - это язык на основе XML, предназначенный для описания определений и реализаций рабочих процессов. Спецификация XPDL, предложенная Workflow Management Coalition (WfMC), представляет собой формальную модель для описания рабочих процессов, относящихся к любым сферам деятельности. В соответствии с ней каждый поток работ разбивается на следующий набор взаимодействующих между собой компонент (см. рисунок 6.2).

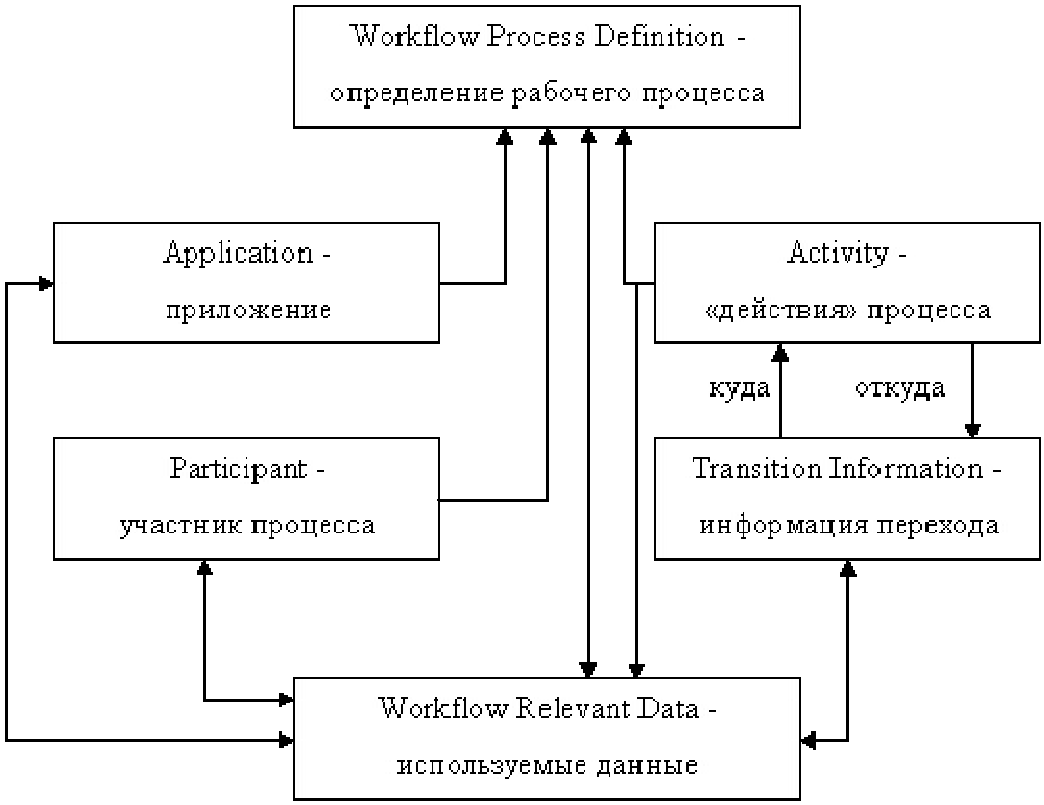


Рис. 6.2. – Компоненты потока работ в XPDL.

- *WorkflowProcessDefinition* - представляет собой контекст выполняющегося процесса, и его данные могут быть доступны всем остальным компонентам.
- *Activity* - <действие> процесса, представляющее собой этап, на котором происходит изменение содержания объектов процесса.
- *TransitionInformation* - переходы между заданиями (могут быть условными и безусловными).
- *WorkflowRelevantData* - оперативные данные, доступные всем компонентам процесса в ходе его выполнения.
- *Participant* - участник процесса, производящий <действия> над объектами и осуществляющий переходы (участники могут являться как человеческими, так и машинными ресурсами).
- *Application* - внешнее IT- или другое приложение, используемое для выполнения <действий>.

В языке XPDL рабочий процесс представляет собой направленный граф, узлами которого являются <действия>, связанные между собой переходами. Переходы могут быть условными, причем условие проверяется на этапе выполнения конкретного <действия>. В языке существует возможность выделения <блоков>, возможность объединения <действий> в блок <действий> со своими отдельными условными или безусловными точками входа и выхода. Так же имеется возможность определять вложенные подпроцессы внутри родительского процесса, которые сами по себе представляют полно-

ценные потоки работ. Спецификация поддерживает возможность экспорта некоторых блоков описания одного процесса в описание другого с возможностью переопределения части импортируемого описания, что исключает необходимость дублирования идентичных фрагментов описания в нескольких процессах. XPDL является расширяемым стандартом. Он позволяет определять набор элементов и атрибутов, специфичных для конкретной сферы его применения. Элементы описания процессов XPDL имеют обширный набор атрибутов, определяющих ход выполнения процесса. К ним можно отнести условные выражения для переходов, временные рамки, задание множественных исполнителей <действий> и т.д.

Рассмотрим в соотвествии с [115] основные элементы языка XPDL и правила их использования.

Основные элементы языка:

- *Activity* (узел-действие);
- *Transition* (переход);
- *Participant* (участник бизнес-процесса);
- *Application* (внешнее приложение);
- *DataType* (тип переменной);
- *DataField* (переменная).

Граф бизнес-процесса определяется наборами элементов *Activity* и *Transition*. *Activity* - это основной элемент бизнес-процесса. Элементы *Activity* соединяются при помощи элементов *Transition*.

Существует три типа элементов *Activity*: • *Route*; • *Implementaion*; • *BlockActivity*.

Route - узлы, не выполняющие действий. Они используются в целях маршрутизации точек управления.

Inplementation - узлы, с которыми связаны действия в бизнес-процессе. Существует три варианта *Inplementation*:

- узел взаимодействия с пользователем;
- узел взаимодействия с внешним приложением;
- узел, запускающий подпроцесс.

BlockActivity - узел-контейнер, содержащий в себе не имеющую разветвлений последовательность узлов.

Элемент *Transition* используется для описания переходов между элементами *Activity*. Каждый такой элемент содержит информацию о том, между какими *Activtiy* и при каких условиях осуществляется переход (переходы бывают условные и безусловные).

В начале описания workflow-процесса находятся спецификации типов (тег TypeDeclaration). Для описания данных, относящихся к процессу, и параметров, передаваемых и возвращаемых приложениями, используются элементы *DataField* и *DataType*. Кроме того, в XPDL существует понятие ExtendedAtributes - оно дает возможность расширять язык путем ввода дополнительных типов переменных.

Для описания участников бизнес-процесса, т. е. сущностей, которые могут выполнять работу, используется элемент *Participant*. Существует шесть подтипов элемента *Participant*:

- *Role* - соответствует роли участника бизнес-процесса. Для каждой роли должен существовать список людей, которые могут быть “назначены” на эту роль;
- *OrganisationUnit* - соответствует административному подразделению организации;
- *Human* - конкретный человек, который будет взаимодействовать с бизнес-процессом при помощи графического интерфейса;
- *System* - конкретное приложение;
- *Resource* - некоторый ресурс, который может предложить исполнителя работы;
- *RecourceSet* - набор ресурсов.

В языке XPDL задаются спецификации внешних приложений - фактически это описание функций: их названия и параметры (при помощи тега *Application*). Внутри *Activity* конкретное приложение указывается в виде параметра тега Tool, и внутри этого тега также производится отображение формальных параметров на фактические.

Упрощенно описание бизнес-процесса в XPDL выглядит следующим образом.

- Определяются переменные бизнес-процесса (и их типы).
- Описываются участники бизнес-процесса (роли и т. д.).
- Задается множество внешних приложений, вызываемых бизнес-процессом (имена функций и типы их параметров).
- Описывается множество всех узлов графа бизнес-процесса, для узла задаются исполнитель, фактические параметры, набор исходящих переходов и т. д.
- Определяются все переходы. Для каждого перехода указывается, какие узлы он связывает, и в случае необходимости условие, при котором по данной связи осуществляется передача управления.

Переход может соединять любые два узла, то есть бизнес-процессу может соответствовать граф любой сложности и топологии. В частности, в графе бизнес-процесса допустимы циклы. В языках, описывающих поток работ (workflow(WF)-языках), к которым относится XPDL, цикл рассматривается как workflow(WF)-паттерн “произвольный цикл”. Поддержка “произвольных циклов” в WF-языке аналогична допустимости использования оператора “goto” в обычных языках программирования. Однако в силу того, что переход управления осуществляется только по ребрам графа, в XPDL нельзя реализовать WF-паттерн “отложенный выбор”.

Технология работы с определениями и экземплярами бизнес-процессов, записанных на языке XPDL, определяется другими спецификациями коалиций WfMC и OMG:

– OMG - Workflow Management Facility Specification;
– WfMC - WAPI (Workflow Application Programming Interface).

В этих спецификациях описывается общая архитектура workflow-системы и интерфейсы взаимодействия различных компонентов системы друг с другом. В частности, спецификации определяют интерфейсы взаимодействия клиентского приложения и внешней системы с ядром workflow-системы, в которое загружен бизнес-процесс. Эти интерфейсы содержат такие команды, как “запустить процесс”, “посмотреть состояние процесса” и т. д. Основными команды, относящимися к работе с экземпляром бизнес-процесса, являются:

– “Сгенерировать список текущих заданий”;
– “Сообщить ядру, что данное задание выполнено”.

То есть предполагается, что ядро системы, реализующей XPDL, полностью пассивно - оно только отвечает на действия взаимодействующих с ним субъектов.

6.3.3. BPEL

BPEL (**Business Process Execution Language**) — язык на основе XML, предназначенный для формального описания бизнес-процессов и протоколов их взаимодействия между собой. BPEL расширяет модель взаимодействия веб-служб и включает в эту модель поддержку транзакций. Данный язык часто рассматривается как ключевая составляющая сервис-ориентированной архитектуры приложений (SOA). Действительно, BPEL позволяет эффективно управлять вызовами сервисов; в особенности он удобен при работе с Web-сервисами.

Одно из главных направлений развития современных информационных систем масштаба предприятия связано с концепцией сервис-ориентированной архитектуры (**services-oriented architecture, SOA**), которая, являясь идеей компонентного построения распределенных компьютерных систем, совсем не нова. Качественно новым элементом идеи стала ориентация SOA на применение, появившихся относительно недавно, технологий, позволяющих создавать распределенные системы на базе Web-сервисов.

В несколько упрощенном виде новизна Web Services заключается в использовании Интернет-технологий на базе открытых отраслевых стандартов, что, в свою очередь, позволяет создавать гетерогенные (платформно независимые), масштабируемые (от локальных до глобальных) решения. Web Services вполне допустимо назвать технологиями XXI века - за точку отсчета их истории, хотя и с некоторой долей условности, можно принять 2000 г. Именно тогда в специализированной прессе стали появляться названия первых WS-стандартов: SOAP (Simple Object Access Protocol) для обмена данными между приложениями, WSDL (Web Services Description Language) для описания программных интерфейсов сервисов и UDDI (Universal Description Discovery & Integration) для хранения и получения WSDL-описаний. Этих

стандартов вполне хватает для создания несложных распределенных решений, но явно недостаточно для построения корпоративных систем. Именно потому наряду с модернизацией базовых стандартов стали появляться специализированные технологии для решения таких задач, как гарантированная доставка сообщений, шифрование и обеспечение безопасности, управление транзакциями и т. п. Все они реализованы на основе XML.

В результате к сегодняшнему моменту сложилась целая система WS-стандартов. Подавляющее большинство этих стандартов существуют лишь на бумаге и пока не поддерживаются в программных продуктах. Более того, здесь наблюдается неприятная тенденция - создание конкурирующих между собой спецификаций, подготовкой которых занимаются две группировки ИТ-компаний: одна во главе с IBM и Microsoft и другая, возглавляемая Sun Microsystems и Oracle.

Основу структуры WS-технологий (см. рис. 6.3) составляет все та же базовая тройка - SOAP, WSDL, UDDI, которой пока, к счастью, не коснулась конкурентная борьба ИТ-вендоров. Над ними располагаются слои "Безопасность", "Обмен сообщениями", "Управление контекстом", "Координация", "Управление транзакциями". Выше находятся еще два слоя - "Оркестровка" (Orchestration) и "Хореография" (Choreography). Первый из них стали использовать в лексиконе WS-технологий для обозначения задач управления бизнес-процессами. Второй появился относительно недавно и также относится к проблематике автоматизации процессов, но акцент при этом делается на интеграцию разнородных приложений.

Рис. 6.3. – Структура стандартов и спецификаций Web Services.

Для решения задач "оркестровки" и "хореографии" также разработан целый набор стандартов. На лидирующие позиции в этой сфере сейчас претендует в первую очередь Business Process Execution Language for Web Services (язык исполнения бизнес-процессов для Web-сервисов), обозначаемый как BPEL4WS, или просто BPEL. Разработкой этого стандарта занимались в основном специалисты IBM и Microsoft. Более того, BPEL фактически стал прямым наследником ранее созданных спецификаций IBM WSFL и Microsoft XLANG. В нем используются сразу несколько XML-спецификаций - WSDL 1.1, XML Schema 1.0, XPath 1.0.

BPEL - это достаточно простой в изучении, но вполне мощный язык, реализованный на базе XML, позволяющий определить последовательность выполнения функционала Web-сервисов в ходе различных потоков операций (транзакций). И здесь наиболее важен тот факт, что BPEL поддерживают ведущие поставщики ПО, предлагающие BPEL-совместимые продукты. И пусть число пользователей этих продуктов, пока, не слишком велико, но можно ожидать, что оно будет очень быстро расти.

В то же время нужно подчеркнуть, что, решая задачи интеграции разнородных приложений в общей цепочке выполнения бизнес-процессов, BPEL совершенно не учитывает, как Web-сервисы выполняют порученные им функции, занимаясь исключительно координацией их работы ("оркестровкой" или даже "хореографией" отдельных исполнителей) в ходе делового потока.

Для иллюстрации возможностей BPEL рассмотрим простой пример. Предположим, что есть некоторый онлайновый продавец антиквариата. Когда он через свой Web-сайт получает запрос на покупку, ему необходимо запустить процессы для автоматического определения кредитоспособности клиента и поиска у оптовых торговцев нужного товара по наименьшей цене. Для этого программа должна отправить запрос в соответствующую учетную систему и в целый ряд профильных электронных магазинов, например, в Amazon и eBay, у которых есть общедоступные интерфейсы Web-сервисов. После получения ответной информации выбирается вариант с наименьшей ценой. Диаграмма этого бизнес-процесса определения стоимости товара приведена на рисунке 6.4. Разумеется, за ним может следовать автоматическое продолжение - заказ товара, отправка его покупателю и т. д.

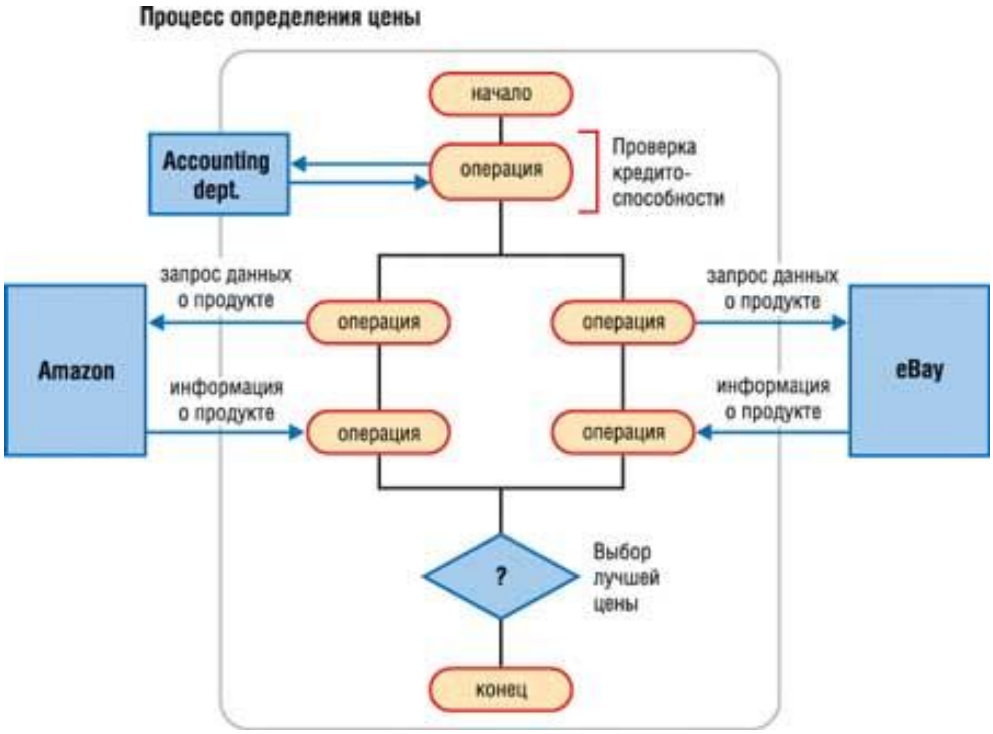


Рис. 6.4. – Определение стоимости товара.

Анализируя эту схему, нужно обратить внимание на то, что она показывает общую логику процесса, передачу запросов и ответов, никак не детализируя функционирование самих удаленных Web-сервисов. На BPEL нужно только описать последовательность обращений и способы обработки получаемой информации. Для написания соответствующей программы можно использовать один из многих BPEL-инструментов, которые на основе такой визуальной диаграммы автоматически сгенерируют код на языке BPEL, создав, таким образом, простейшее приложение класса SOA.

Каждая операция (invoke), представленная на диаграмме прямоугольным блоком, описывается простой BPEL-структурой, которая включает элементы, соответствующие таким действиям как: отправка запроса, ожидание, получение ответа и т.д. Например, операция запроса о кредитоспособности (которая может выполняться в синхронном или асинхронном режимах с переход к последующим действиям с ожиданием или без ожидания ответа) может быть представлена в коде BPEL следующим образом:

```
<invoke name="invoke-1"
   partnerLink="AccountingDept"
  portType="services:CreditRatingService"
  operation="process" inputVariable="crIn"
  outputVariable="crOut">
</invoke>
```

Разумеется, приведенный пример BPEL-программы очень далек от реальных бизнес-приложений и служат здесь только иллюстрацией основных идей, заложенных в этот стандарт. BPEL позволяет применять условные ветвления, организовывать потоки параллельных вычислений, описывать правила соединения потоков, обмениваться данными между потоками, применять синхронные и асинхронные режимы взаимодействия, обрабатывать исключительные ситуации и т. п. При этом BPEL использует традиционную, центрическую модель исполнения: любые внешние сообщения из внешнего мира ожидаются только в том случае, если они ожидаются по ходу процесса.

Создаваемые с помощью BPEL приложения относятся к категории "процессно-ориентированных" (process-based applications). Фактически они состоят из двух отдельных слоев исполнения. Верхний слой описывает бизнес-логику процесса, представленную на языке BPEL, нижний слой выполняет собственно все функциональные операции с помощью различных Web-сервисов. BPEL-приложение может выполняться на любом сервере приложений, имеющем механизм исполнения BPEL.

Развитые инструменты позволяют визуально проектировать полнофункциональные BPEL-приложения, не требуя написания кода вручную. Эти средства, кроме того, включают функции автономного тестирования программы. Однако для работы в реальных условиях под управлением BPEL-сервера требуются правильные установки для всех используемых Web-сервисов в WSDL-файлах, а также конфигурирование необходимых коммуникационных протоколов (например, Java Message Service или HTTP).

Рассмотрим в соотвествии с [115] основные элементы языка BPEL (BPEL4WS) и правила их использования.

Язык BPEL4WS во многом он похож на BPML, однако существенно сложнее его. Как уже было отмечено выше, появился BPEL4WS путем слияния WF-языков WSFL и XLANG. Эти языки основаны на разных моделях: WSFL базируется на теории графов, XLANG - на иерархии тегов XML. BPEL4WS унаследовал конструкции обоих языков. Например, он допускает реализацию некоторых WF-паттернов в двух вариантах: в стиле WSFL и в стиле XLANG. В некоторых случаях допустим и смешанный стиль. Это делает BPEL4WS трудным для изучения.

Несмотря на то что BPEL4WS является наследником языков различной природы, его можно отнести к классу структурно-ориентированных: унаследованные "граф-ориентированные" конструкции реально соответствуют некоторым ограничениям на порядок выполнения *Activities* внутри параллельного блока.

BPEL4WS определяет два вида процессов - абстрактный и исполняемый. Абстрактный процесс определяет протокол обмена сообщениями между различными участниками, не "открывая" алгоритмы их "внутреннего" поведения. В отличие от абстрактного исполняемый процесс содержит в себе алгоритмы, определяющие порядок выполнения *Activities*, назначение исполнителей, обмен сообщениями, правила обработки исключений и т. д. *Activities* в BPEL4WS делятся на примитивные и структурные.

Примитивные *Activities*:

- *Receive* - ожидает сообщения от внешнего источника;
- *Reply* - отвечает внешнему источнику;
- *Invoke* - "запускает" операцию какого-либо Web-сервиса;
- *Wait* - ждет в течение определенного периода времени;
- *Assign* - копирует значение одной переменной в другую;
- *Throw* - отбрасывает исключение в случае ошибки;
- *Terminate* - принудительно завершает выполнение службы;
- *Empty* - не выполняет никаких действий.

Структурные *Activities*:

- *Sequence* - соответствует последовательному выполнению *Activities*, содержащихся внутри этого элемента.
- *Switch* - условная передача управления (соответствует оператору Switch языков программирования C++, Java и т. д.);
- *While* - организует цикл типа "While";
- *Pick* - запускает обработку событий и исключительных ситуаций;
- *Flow* - соответствует параллельному выполнению *Activities*, содержащихся внутри этого элемента;
- *Scope* - группирует узлы для программы -- обработчика ошибок.

Кроме того, в языке присутствует понятие "связь" (*link*). Эта конструкция унаследована из граф-ориентированного "предка" BPEL4WS. Как прави-

ло, применяется она к *Activities*, находящимся внутри параллельного блока, и накладывает ограничения на порядок их выполнения. К конструкции языка, использующей *link*, может быть применено, например, такое ограничение: «*Activities*, соединенные при помощи этого элемента, не могут образовывать циклы».

Переменные описываются при помощи тега “variables”. Для задания исполнителя используется тег “partnerLink”. В некоторые теги добавлен параметр “variable”.

Общение с “внешним миром” в BPEL4WS определяется технологией Web-сервисов.

6.3.4. BPML

BPML (Business Process Modeling Language) - язык на основе XML, предназначенный для определения формальной модели, выражающей выполнимые процессы, которые описывают все аспекты корпоративных бизнес-процессов, в том числе с использованием Web-сервисов. В отличие от XPDL, он принадлежит (как и BPEL) к так называемым структурно-ориентированным языкам: бизнес-процесс в BPML соответствует не математическому графу, а иерархическому набору вложенных и последовательных тегов.

Рассмотрим общее описание языка в соотвествии с работой [116].

Язык BPML дополняет язык реализации бизнес-процессов BPEL. BPML может использоваться для определения детальных бизнес-процессов, исполняемых при вызове каждого Web-сервиса. BPML преобразует ("мэппирует") бизнес-операции в обменные сообщения. Этот язык может использоваться для определения корпоративных бизнес-процессов, комплексных Web-сервисов и многостороннего сотрудничества. В разработке BPML-спецификаций участвует целый ряд организаций: CSC, Intalio, SAP, Sun, SeeBeyond, Versata и др.

Как следует из BPML-спецификаций, BPML определяет операции разного уровня сложности, транзакции и компенсации, управление данными, параллелизм, обработку исключений и операционную семантику. Грамматика BPML оформляется в виде XML-схемы, что обеспечивает постоянство определений и их обмен между гетерогенными системами и инструментами моделирования.

BPML - это богатый и зрелый язык, с помощью которого можно описывать как простые, так и сложные бизнес-процессы. Поскольку BPML и BPEL - это языки с блочной структурой, то у них одинаковый набор выражений и похожий синтаксис. По сравнению с операциями, которые поддерживает BPEL, возможности BPML шире. Синтаксис BPML поддерживает операции и их типы, процессы, свойства, сигналы, расписания и нестандартные ситуации.

Простые типы операций BPML:

- *Action* - выполняет или вызывает выполнение операции, включающей

обмен входящими и исходящими сообщениями;

- *Assign* - присваивает новое значение показателю;
- *Call* - запускает процесс и ждет его завершения;
- *Compensate* - инициирует компенсацию для указанных процессов;
- *Delay* - выражает промежуток времени;
- *Empty* - ничего не делает;
- *Fault* - выдает сообщение об ошибке в текущем контексте;
- *Raise* - активизирует сигнал;
- *Spawn* - запускает процесс без ожидания его завершения;
- *Synch* - синхронизирует по сигналу.

Сложные типы операций BPML:

- *All* - выполняет операции параллельно;
- *Choice* - выполняет операции из одного из составных комплектов, выбранного в ответ на событие;
- *Foreach* - однократно выполняет операции для каждого пункта из списка;
- *Sequence* - выполняет операции в последовательном порядке;
- *Switch* - выполняет операции из одного из составных комплектов, выбранного на основе истинного значения условия;
- *Until* - выполняет операции один или более раз на основе истинного значения условия;
- *While* - не выполняет операции или выполняет их один или более раз на основе истинного значения условия.

Простые операции - это операции, которые могут привести к выполнению множественных операций, в частности такие, как *action*, *call*, *compensate* и *spawn*. Но сама простая операция не определяет контекст для выполнения других операций. BPML включает все логические конструкции строгого языка программирования. Простые операции прерывают выполнение (*abort*) или выдают сообщение об ошибке (*fault*), если их завершению препятствует неожиданная ошибка.

Сложная операция - это операция, включающая в себя одну или более дочерних операций. Она устанавливает контекст для выполнения действий и направляет это выполнение. Сложные операции определяют иерархическую организацию. Она может быть простой - например, повторяющееся выполнение одной и той же операции, или более сложной - например, установление вложенного контекста для выполнения множественных операций. BPML также поддерживает и другие формы организации, в том числе циклические графы и рекурсивные операции. Сложные операции используются в тех случаях, когда требуется иерархическая организация, в частности, для установления нового контекста, необходимого при выполнении дочерних операций.

Сложная операция, включающая комплекты множественных операций, должна выбирать, какой из них использовать. Для этого применяется несколько стандартных логических конструкций. Операция *choice* ждет собы-

тия, которое должно быть инициировано, а затем выбирает комплект операций, связанный с обработчиком этого события. Операция *switch* оценивает условия и выбирает комплект операций, связанный с тем условием, значение которого является истинным. Все остальные сложные операции, определенные в спецификации BPML, включают только один комплект операций, поэтому им не приходится принимать подобные решения.

Сложная операция также определяет, сколько раз должны быть выполнены операции из общего набора операций. Для этого используются следующие стандартные логические конструкции: операция *until* - повторяет выполнение операций, пока значение условия не станет истинным; операция *while* - повторяет выполнение операций, пока значение условия остается истинным; и операция *foreach* - выполняет операции однократно для каждого пункта списка. Все остальные названные выше сложные операции выполняют действия из комплекта операций однократно.

Помимо этого, сложная операция определяет порядок выполнения других операций. Операция *sequence* обеспечивает выполнение всех действий из комплекта операций в последовательном порядке. Операция *all* обеспечивает выполнение всех действий из комплекта операций одновременно. Остальные сложные операции языка BPML обеспечивают выполнение операций в последовательном порядке.

Сложная операция считается завершенной, когда закончено выполнение всех действий из комплекта операций. Это включает все действия, перечисленные в списке операций, и все процессы, запускаемые из определения, сделанного в контексте комплекта операций. Вложенные процессы и процессы обработки нестандартных ситуаций рассматриваются как действия из комплекта операций.

Сложные операции прерываются и разрываются, если одно из действий, входящих в их состав, разрывается таким образом, что его восстановление невозможно.

Рассмотрим в соотвествии с [115] основные элементы языка BPML и правила их использования.

Основные элементы, при помощи которых определяется workflow-процесс в BPML:

- . *Activity* (узел - действие);
- . *Context* (контекст);
- . *Property* (свойство);
- . *Signal* (сигнал);
- . *Exception* (исключение).

Activity - основной элемент бизнес-процесса. Элементы *Activity* могут соединяться последовательно или вкладываться один в другой. Соответственно *Activity* могут быть простыми (не содержащими других *Activity*) или сложными. В описании языка указано, что бизнес-процесс является специальным типом сложной *Activity*. Всего существует 17 типов простых *Activity*. Основной тип простой *Activity* называется *Action*. Когда управление бизнес-

процессом попадает в *Activity* этого типа, происходит вызов описанных там Web-сервисов. Есть типы *Activity*, соответствующие ветвлению процесса (ветвление относится только к содержащимся внутри них *Activities*), - это *Switch* и *All Activities*. Также существуют типы *Activities*, которые запускают дочерние процессы (как с ожиданием их окончания, так и без), организуют задержки выполнения процесса и т. д. Кроме того, есть несколько типов *Activities*, реализующих разного вида циклы. Для синхронизации точек управления, находящихся в *Activities* одного уровня вложенности, в языке используются сигналы (*Signals*).

В BPML также введены понятия “исключение” (*Exception*) и “компенсация” (*Compensation*). “Исключение” соответствует возникновению нештатной ситуации, когда оказывается, что выполнять некоторый участок бизнес-процесса уже не требуется. Например, во время выполнения бизнес-процесса оформления туристической поездки клиент позвонил в туристическую компанию и сообщил, что он отказывается от поездки.

“Компенсация” соответствует необходимым действиям по корректному завершению ситуации, возникшей в связи с *Exception*. Если в вышеприведенном примере для клиента были забронированы билеты на самолет и номер в гостинице, то задачей *Compensation* будет отменить бронирование.

На языке BPML данные описываются в рамках контекста (*Context*). Контекст содержит относящиеся к процессу переменные, локальные определения процессов, сигналов и т. д. и служит для передачи информации между узлами и синхронизации. Переменные определяются тегом Property. Они могут быть локальными или глобальными по отношению к данному контексту.

6.3.5. Соотношение языков.

В 2000 г. появилась коалиция BPMI, которая достаточно быстро разработала основанный на технологии Web-сервисов язык определения бизнес-процессов BPML и начала создание других полезных стандартов (не совместимых со стандартами WfMC). Через некоторое время коалиция BPMI подготовила стандарт графических диаграмм, описывающих WF-процесс - BPMN. Язык также содержал правила автоматического перевода графических диаграмм BPMN в язык BPML [117].

Однако вслед за объединением IBM, Microsoft и BEA и созданием новой коалицией другого WF-языка, также основанного на технологии Web-сервисов (BPEL4WS), в рядах коалиции BPMI началась паника. Прогнозы абсолютного большинства экспертов говорили о том, что IBM, Microsoft и BEA “продавят” свою спецификацию и именно BPEL4WS станет стандартом де-факто в качестве языка определения бизнес-процессов. Был период, когда BPMI позиционировало язык графических нотаций BPMN как графическую оболочку для BPEL4WS. Однако в настоящее время коалиция реанимировала BPML и предлагает экспорт из BPMN как в BPML, так и в BPEL4WS [117].

BPEL и BPML - это похожие подходы к решению одной и той же проблемы: определение логики процессов в языке XML таким образом, чтобы

результат мог использоваться как исполняемый код программными продуктами на основе BPM. Обладая средствами дополнительной поддержки вложенных процессов и другого синтаксиса, BPML может считаться расширенным вариантом языка BPEL. В тех случаях, когда эти языки используются совместно, сквозной обзор показывает роль каждого бизнес-процесса в общей картине и то, какие бизнес-операции он выполняет.

Однако, при изучении стандарта BPEL возникает впечатление, что он неоправданно усложнен. Понятия, относящиеся к предметной области, находятся в нем на одном уровне с техническими понятиями, специфическими для технологии Web-сервисов. Это сильно ухудшает читаемость языка (например, по сравнению с XPDL).

Идеологически языки BPEL4WS и BPML очень похожи. Нам кажется, что BPML проще и удобнее, чем BPEL4WS, однако за BPEL4WS стоят такие корпорации-гиганты, как IBM, Microsoft, BEA, SAP и Siebel, и вполне возможно, что благодаря “маркетинговой мощи” этих компаний язык BPML будет полностью вытеснен языком BPEL4WS.

Ситуация с BPMN оказалась тоже далеко не безоблачной. OMG-группа разработала диаграмму Activity в языке UML 2.0, эта диаграмма - в некотором смысле альтернатива языку BPMN, а по графической выразительной силе эти нотации примерно одинаковы. Мы считаем, что для описания бизнес-процессов в настоящее время BPMN все же удобнее, чем Activity-диаграмма языка UML 2.0, однако вполне можно ожидать, что в следующей версии языка UML Activity-диаграмма вберет в себя все текущие преимущества BPMN и за счет маркетингового веса OMG именно диаграмма Activity UML, а не BPMN может стать фактическим стандартом графической нотации.

При этом сама спецификация BPMN не определяет формата файла, в котором можно сохранять описание и которым можно обмениваться, однако уже есть как минимум одна спецификация, описывающая этот формат - это XPDL. В спецификации XPDL v.2.00 [http://yurivolkov.com/ articles/Diagrams_for_business_ processes_ru.html#_Ref4] явно указано, что одно из её назначений - служить описанием формата файла для нотации BPMN. Т.е. XPDL позволяет хранить не только логику процесса (как и BPEL, другая спецификация формата описания бизнес-процесса, понимаемого машиной), но и его графическое BPMN-представление. Таким образом, формат файла BPMN уже существует, что позволяет "понимать друг друга" различным инструментальным средствам моделирования.

При этом шум, создаваемый в основном усилиями Microsoft, IBM и Oracle, привел к тому, что BPEL воспринимается как если не единственный, то основной стандарт в области систем менеджмента бизнес-процессов. В связи с этим представляют интерес две недавние заметки в ebizq.net.

Первая принадлежит Sandy Kemsley, известному аналитику в области BPM: «*Assorted thoughts on BPEL*» («*Избранные мысли о BPEL*»). Отмечая, что BPEL не продвинулся ни на шаг с 2005 г. и так остался в состоянии whitepaper, а также тот факт, что сложность BPEL не допускает использования его аналитиками, Sandy приходит к следующему выводу: «BPEL стано-

вится скорее формальным пунктом в опросном листе, чем реальным требованием – большинство организаций-заказчиков слабо представляют себе для чего он им нужен.» С другой стороны, тем, кто разрабатывает сложные сценарии взаимодействия веб-сервисов, возможностей BPEL не хватает и они возлагают надежды на зарождающийся стандарт WS-CDL: Web Services Choreography Description Language [http://www.ebizq.net/blogs /column2/2007/03/assorted_though.php].

Во второй заметке с красноречивым названием «*Is XPDL the Silent Workhorse of BPM?*» («*XPDL – рабочая лошадка BPM?*») обосновываются следующие тезисы [http://www.ebizq.net/topics/human_centric_bpm/features/ 7852.html]:

- Из 10 стандартов, работа над которыми велась в 2003 г., сегодня, в результате прекращения работы над одними и консолидации других, осталось три: BPMN, XPDL, BPEL. Вопреки распространенному мнению, эти стандарты не конкурируют друг с другом. Явно отдельное положение занимает BPMN: это система условных обозначений и правил рисования диаграмм, доступная для понимания всех заинтересованных лиц – и бизнес-аналитиков, и менеджеров, и технических специалистов.
- Разница между BPEL и XPDL не столь очевидна, но тоже существенна. BPEL – это своего рода «машинный код», исполняемый язык, при помощи которого программируется последовательность вызова веб-сервисов. В конечном счете, BPEL манипулирует битами и байтами, пересылаемыми из одной точки в другую.
- XPDL был разработан WfMC для хранения и обмена диаграммами процессов между программными инструментами, один из которых предназначен для моделирования процесса, другой для чтения и редактирования, третий для исполнения процесса внутри BPM-«движка» и т.д. Примечательное отличие XPDL от BPEL в том, что он обеспечивает взаимно-однозначное представление для BPMN-диаграмм. Преобразование же из BPMN в BPEL является односторонним, аналогично компиляции из языка высокого уровня в машинный код.
- Иногда высказываемое мнение о том, что XPDL мертв или не относится к делу, свидетельствует только о плохой информированности. На сегодня XPDL поддерживает около 50 вендоров, включая таких энтузиастов BPEL, как IBM и Oracle, и 8 из 11 лидеров рынка BPMS по версии Gartner'2006. XPDL является стабильным стандартом на протяжении многих лет, и вы можете быть уверены, что XPDL V.3, когда бы он не появился, и инструментарий, который с ним будет работать, сможет прочесть те схемы, которые вы создаете при помощи XPDL сегодня. К тому же он общедоступен, не будучи связан никакими лицензионными ограничениями.

Не получая такой же шумной рекламы, как «некоторые другие стандарты», заключает автор Jon Pyke, XPDL молча делает то, что ему положено.

Сравненивая же BPML и XPDL, специалисты говорят следующее (см., например, [115]).

Язык BPML существенно “легче” языка XPDL. В нем не надо описывать внешние приложения (Applications в XPDL). Эти функции “перекладываются” на технологию Web-сервисов. Не надо определять и “присоединять к Activities” переходы. Архитектура связей определяется вложенностью тегов, соответствующих элементам Activity. В BPML нет понятия Transition. Не надо в рамках языка специально определять описание участника бизнес-процесса. Все участники - это Web-сервисы. Следовательно, соответствующие описания отвечают спецификации Web-сервисов.

Кроме того, для работы с бизнес-процессами, написанными на XPDL, требуются дополнительные спецификации. В частности, они указывают, каким образом можно “сообщить” определенной Activity, что она выполнена и управление может двигаться дальше, и т. д. В соответствии с идеологией языка XPDL среда, в которой выполняется бизнес-процесс, не является активной; активными должны быть внешние участники, а среда выполнения процесса только реагирует на их действия.

“Идеология” языка BPML скорее противоположна - в ней бизнес-процесс может быть активным и давать задания участникам-исполнителям. Исполнители, являясь Web-сервисами, могут “ничего не знать” о бизнес-процессе, в котором они участвуют.

Однако отказ от понятия Transition и замена его вложенностью тегов приводят к тому, что граф, соответствующий бизнес-процессу, в BPML не может быть графом произвольной структуры, например, он не может содержать сложные циклы. Вследствие этого в BPML невозможно реализовать WF-паттерн “Произвольный цикл”. Для того чтобы выполнять повторяющиеся последовательности шагов, в BPML введено несколько типов Activity-циклов, однако в этом случае циклически повторяться может только содержащаяся внутри данной Activity последовательность узлов. Отсутствию произвольных циклов в бизнес-процессе можно поставить в соответствие аналогию программирования “без goto”.

В BPML параллельно выполняющиеся ветви процесса могут дополнительно обмениваться сигналами, производя, таким образом, синхронизацию. Трудно сказать, облегчает это понимание бизнес-процесса или, наоборот, затрудняет. В силу того, что язык BPML основан на тегах, в нем легко организовать генерацию и обработку исключений. А не будучи явно основанным на теории графов, он позволяет реализовать WF-паттерн “отложенный выбор”. В BPML поддерживаются такие концепции, как транзакции и расписания. В нем можно устанавливать задержки и deadlines.

Недавно и в XPDL появились понятия TimeOuts и Exceptions, однако функциональность их заметно ниже соответствующих конструкций BPML.

6.4. CASE-инструментарий бизнес-моделирования в нотации BPMN.

В настоящее время уже существует множество CASE-средств, автоматизирующих построение BPD-диаграмм в BPMN-нотации. Учитывая цели данного учебно-практического пособия, воспользуемся информацией, представленной на сайте отечественной консалтинговой группы «Руна» http://wf.runa.ru/rus, представляющем CASE-инструментарий RunaWFE.

Одним из способов организации управления бизнес-процессами является использование BPMS (Business Process Management System/Suite) — системы управления бизнес-процессами. Целями использования таких систем являются повышение качества исполнения бизнес-процессов, сокращение временных затрат, получение возможности контроля деятельности для повышения качества управления, а также непрерывное совершенствование внутренних бизнес-процессов с возможностью их изменения "на лету".

BPMS предназначена для автоматизации бизнес-процессов, но автоматизация бизнес-процессов не является основной целью внедрения системы. С внедрением BPM-системы у компании появляется инструмент для управления бизнес-процессами, что позволяет повысить исполнительскую дисциплину, заставить компанию работать по установленным правилам и непрерывно совершенствовать бизнес-процессы. Бизнес-процессы, без должного управления и контроля, склонны терять свою эффективность.

BPM System в обязательном порядке включает моделирование бизнес-процессов, их исполнение (process engine, процессный “движок”) и мониторинг/анализ. Опционально может включать имитационное моделирование, движок бизнес-правил и многое другое, но в этом случае она уже является BPM Suite.

Одной из информационных технологий реализации BPMS является RunaWFE - информационная система управления бизнес-процессами, позволяющая построить эффективное взаимодействие сотрудников компании и контролировать их деятельность с целью повышения качества работы всей компании с помощью ресурсов локальной вычислительной сети организации. То есть система RunaWFE реализует концепцию BPM, что позволяет строить гибкие адаптивные информационные системы, способные оперативно меняться вместе с изменением бизнес-процессов компании.

Система RunaWFE является свободным программным продуктом с открытым кодом. Ее дистрибутивы, исходный код, а также документацию можно скачать с сайта разработчиков свободного программного обеспечения по адресу http://sourceforge.net/projects/runawfe/files. Систему можно свободно установить на любое количество компьютеров без каких-либо ограничений.

6.4.1. Назначение и возможности.

RunaWFE - свободная, масштабируемая, ориентированная на конечного пользователя система проектирования бизнес-процессов предприятия и

управления ими, а также административными регламентами при их реализации.

Основная задача системы: проектировать модели бизнес-процессов, раздавать задания исполнителям и контролировать их выполнение. Последовательность заданий определяется графом бизнес-процесса, который менеджер или бизнес-аналитик может быстро изменять при помощи редактора бизнес-процессов.

При помощи переменных бизнес-процесса в системе происходит передача информации между исполнителями заданий. В случае хранения документов в переменных бизнес-процесса, систему можно использовать для автоматизации документооборота предприятия.

Система RunaWFE состоит из серверной и клиентской частей. Компоненты, относящиеся к серверной части системы, это:

- RunaWFE – сервер;
- Бот-станция.

Серверная часть системы может быть установлена на уже существующий аппаратный сервер. Платформой, на которой разворачивается RunaWFE-сервер является сервер JBoss. Пример состава программного обеспечения на сервере показан на рисунке 6.5.

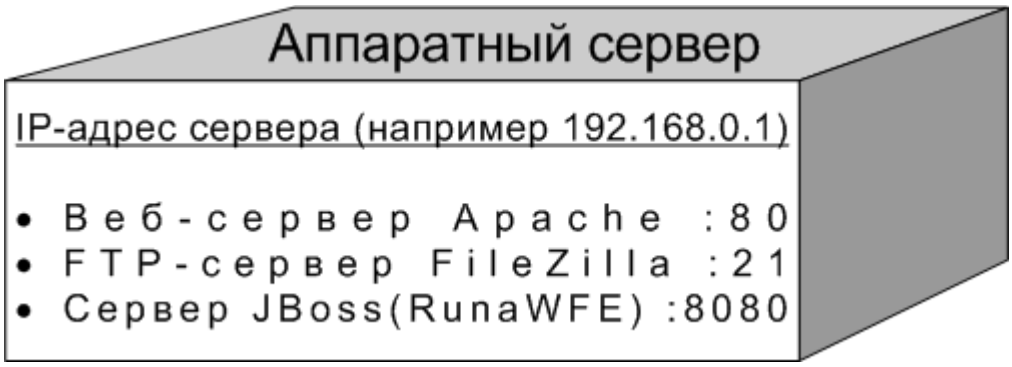


Рис. 6.5. – Структура программного обеспечения сервера.

На одном сервере может располагаться несколько серверных сетевых приложений. Во избежание конфликтов на уровне сетевого адаптера каждому из приложений присваивается один или несколько портов. В результате сетевой адрес этих приложений выглядит так: IP-адрес:порт. Например, чтобы обратиться к веб-интерфейсу сервера JBoss нужно будет в поле адреса набрать http://192.168.0.1:8080. По умолчанию большинство браузеров (IE, Mozilla Firefox, Opera, Google Chrome и др.) обращаются на порт 80.

Компоненты, относящиеся к клиентской части системы:

- Клиент (web-интерфейс);
- Клиент-оповещатель о поступивших заданиях;
- Графический редактор бизнес-процессов;
- Симулятор бизнес-процессов.

Взаимодействие между клиентской и серверной частями системы производится посредством локальной вычислительной сети, как показано на рисунке 6.6.

Рис. 6.6. – Схема взаимодействия между клиентами и сервером RunaWFE

При помощи Web-интерфейса системы пользователь может:

- получать, фильтровать, выполнять задачи, генерируемые экземплярами бизнес-процессов;
- запускать новые экземпляры бизнес-процессов;
- просматривать состояния выполняющихся экземпляров бизнес-процессов;
- загружать файлы-архивы, содержащие определения бизнес-процессов в систему.

При помощи web-интерфейса системы администратор может:

- создавать-удалять пользователей и группы пользователей;
- включать (исключать) пользователей в группы;
- раздавать права на объекты системы пользователям и группам пользователей;
- принудительно останавливать экземпляры бизнес-процессов.

При помощи графического редактора бизнес-процессов аналитик может разрабатывать бизнес-процессы и экспортировать их в файлы-архивы в файловую систему.

При помощи клиента-оповещателя о поступивших заданиях пользователь может получать оповещения о поступивших заданиях.

При помощи симулятора бизнес-процессов можно тестировать разработанные бизнес-процессы на клиентском компьютере аналитика, не загружая их в промышленную систему.

Основные возможности системы:

- Работа с определениями и экземплярами бизнес-процессов;
- Работа со списками заданий;

- Визуализация форм, соответствующих заданиям;
- Работа с системой через web-интерфейс;
- Предоставление возможности работы с системой приложениям специального вида (ботам);
- Авторизация и аутентификация пользователей.
- Возможности графического редактора:
- Редактирование графа бизнес-процесса;
- Создание и редактирование графических форм заданий;
- Создание и назначение ролей;
- Создание переменных.

Система является как бы конвейером, перенесенным с производства в офис и позволяет работнику выполнять задачи, не отвлекаясь на:

- получение необходимой для выполнения задания информации;
- передачу результатов своего труда другим работникам;
- изучение должностных инструкций.

Все необходимое возникает на экране пользователя при "клике" на задание (в частности на экране может быть написана инструкция - как надо выполнять это задание).

Исполнителями могут быть как люди, так и специальные компьютерные приложения - боты. Используя боты, можно при помощи системы решить задачу интеграции разнородных приложений предприятия в единую систему (КИС).

В случае заданий, выполняемых только людьми (без ботов), систему можно распространять в виде коробочной версии. Пользователь сможет сам установить систему, запустив дистрибутив и сразу после установки начать с ней работать. Наличия программиста при этом не потребуется. Установка и интерфейс RunaWFE изучаются в лабораторных работах заимствованных из практикума [118] (см. лабораторный практикум по дангному курсу).

С нашей точки зрения вашнейшей составной частью проекта RunaWFE является графический редактор бизнес-процессов RunaGPD.

RunaGPD – это графический редактор бизнес-процессов для открытой системы управления бизнес-процессами RunaWFE. RunaGPD является частью открытого проекта RunaWFE, свободно распространяется под LGPL лицензией. RunaGPD может выполняться на различных платформах (Linux, Windows и т.д.). Редактор можно свободно загрузить вместе с исходными кодами с портала sourceforge по адресу http://sourceforge.net/projects/runawfe/files. Установка RunaGPD подробно описна в документе «RunaWFE. Графический редактор бизнес-процессов. Руководство разработчика».

6.4.2. Особенности функционирования и интерфейса.

На одном сервере запускается RunaWFE – сервер. На нескольких серверах могут быть запущены бот-станции. На клиентских компьютерах запус-

кается клиент-оповещатель о поступивших заданиях или браузер, в котором открывается web-интерфейс системы. На клиентских компьютерах может быть запущен графический редактор бизнес-процессов, также на клиентских компьютерах может быть запущен симулятор бизнес-процессов.

RunaWFE – сервер содержит определения загруженных в него бизнес-процессов и выполняющиеся экземпляры бизнес-процессов. Бот-станции содержат боты, которые периодически опрашивают RunaWFE – сервер. Если выполняющиеся на RunaWFE – сервере экземпляры бизнес-процессов содержат задачи для ботов, загруженных в бот-станцию, то боты выполняют эти задачи и возвращают результаты работы на RunaWFE – сервер.

Для того, чтобы можно было войти в систему RunaWFE – сервер (симулятор) должен быть запущен. Симулятор можно запустить, например, иконкой на рабочем столе или командой меню «...RunaWFE / Start Simulation». При запуске симулятора появится консольное окно (см. рис. 6.7).

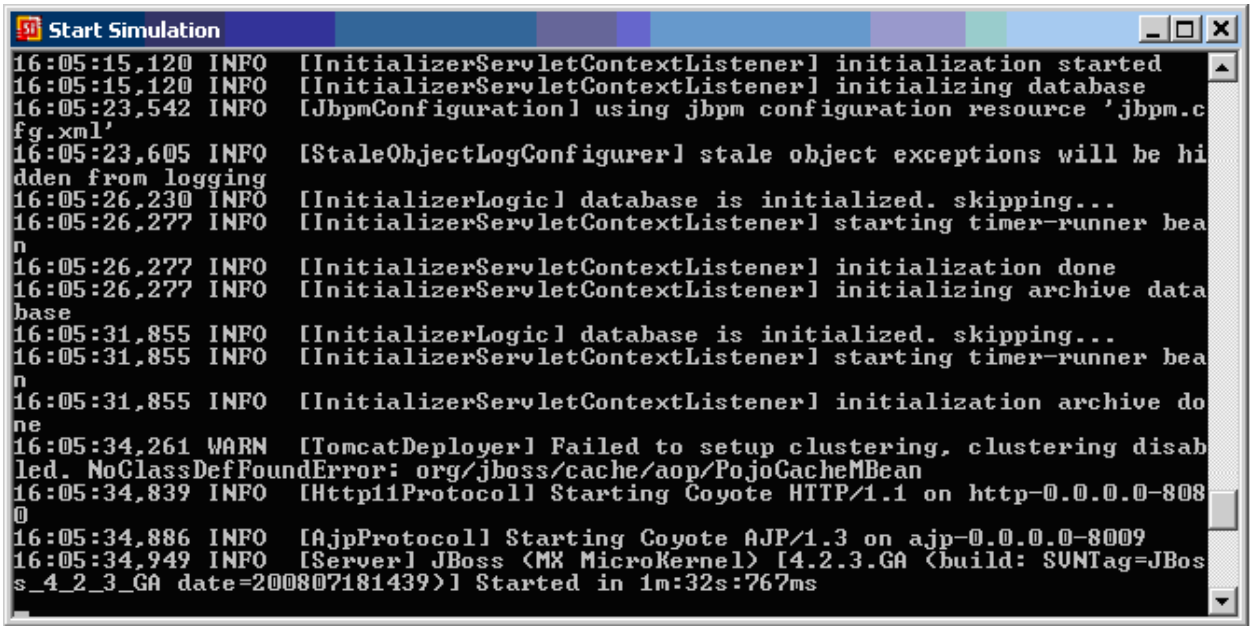


Рис. 6.7. – Окно запуска симулятора RunaWFE

Надпись «…INFO [Server] JBoss … Started in …» означает, что симулятор запущен. После этого с системой можно работать через web-интерфейс. Это можно сделать как через клиент-оповещатель о поступивших заданиях, так и через обычный браузер.

После того, как система запущена, графический интерфейс системы доступен в окне клиента-оповещателя о поступивших заданиях или просто в web-браузере по адресу: http://<servername>:8080/wfe. Здесь <servername>-адрес сервера, на котором установлена система. В данном случае работа с системой будет показана в варианте использования web-браузера. (В случае использования протокола SSL для работы с графическим интерфейсом надо использовать URL: https://<servername>:8443/wfe).

Если web-браузер открыть по указанному выше адресу, то он показает

страницу ввода логина и пароля пользователя (см. рис. 6.8).

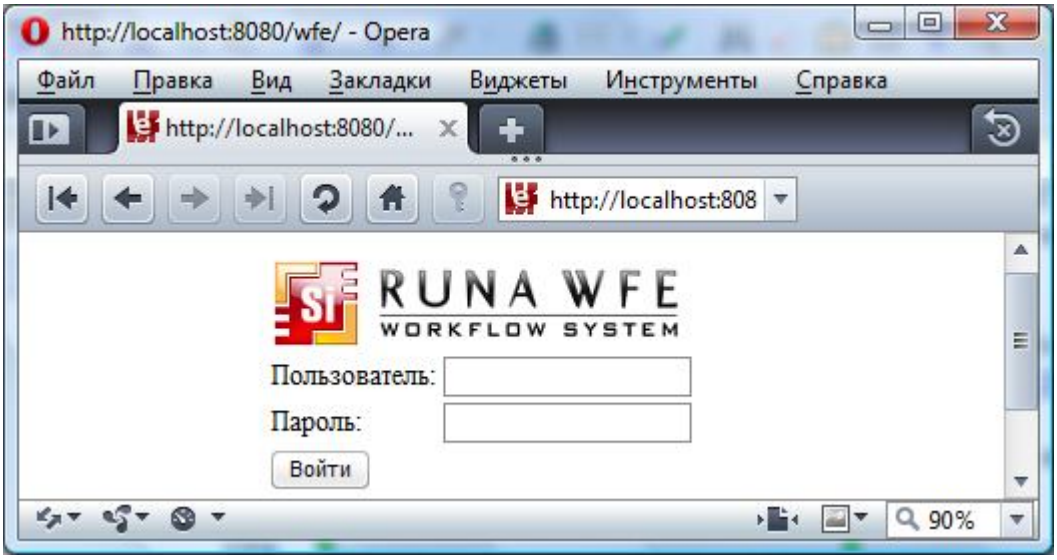


Рис. 6.8. – Окно ввода логина и пароля RunaWFE

Логин администратора системы по умолчанию – «Administrator» (существенно, что с большой буквы), пароль администратора – «wf».

В левой верхней части появившейся после входа в систему страницы (см. рис. 6.9) находится меню системы, состоящее из следующих элементов:

- Список заданий.
- Запустить процесс.
- Запущенные процессы.
- Бот-станции.
- Исполнители.
- Система.

Дадим краткое описание пунктов меню системы RunaWFE.

Меню «Список заданий». При выполнении команды меню «Список заданий» открывается форма списка заданий для данного пользователя. Здесь пользователь может, кликнув на задание, открыть форму задания, ввести в нее данные, а также отметить выполнение задания. Также в списке заданий пользователь может искать, фильтровать задания, выводить в строках задания значения переменных бизнес-процессов.

Меню «Запустить процесс». На странице, соответствующей пункту меню «Запустить процесс» находится список определений бизнес-процессов. Здесь пользователь может запустить бизнес-процесс, посмотреть схему и другие свойства бизнес-процесса, посмотреть описание бизнес-процесса. Если у пользователя есть соответствующие права, он может загрузить новый бизнес-процесс в систему или загрузить новую версию уже существующего процесса.

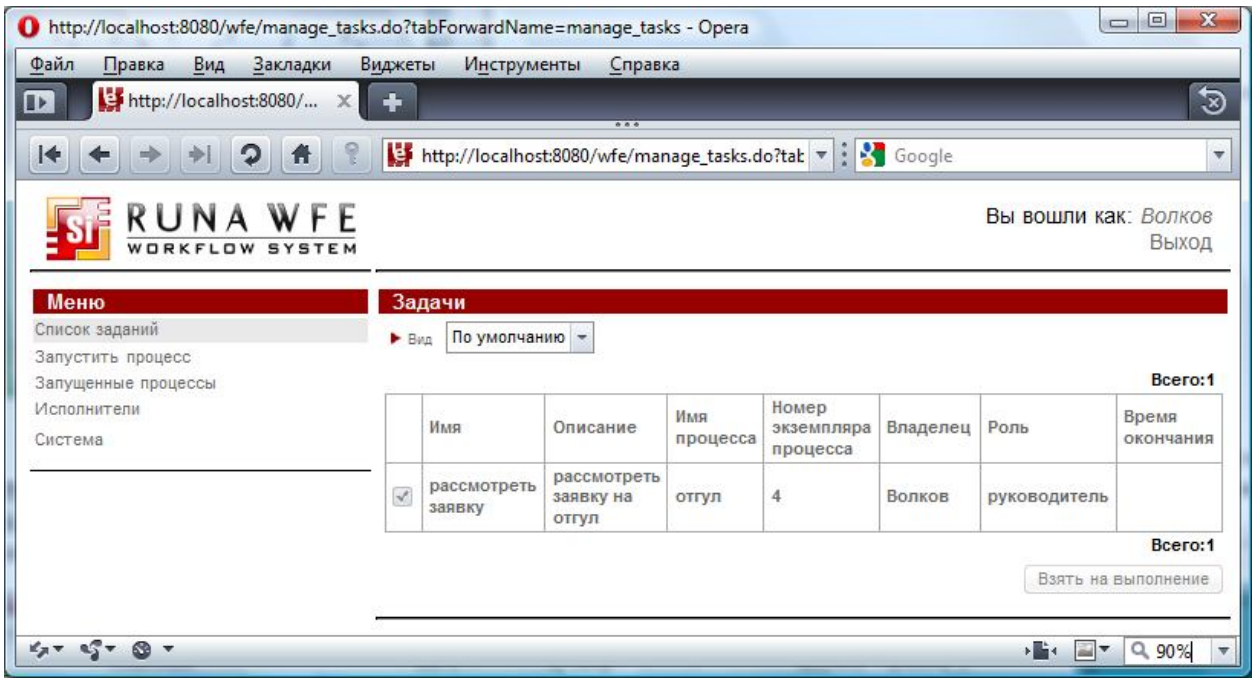


Рис. 6.9. – Главное окно RunaWFE

Меню «Запущенные процессы». На странице, соответствующей пункту меню «Запущенные процессы», находится список экземпляров бизнес-процессов, доступных для чтения данному пользователю. Здесь пользователь может посмотреть состояния выполняющихся экземпляров бизнес-процессов, в частности – положение текущих точек управления на схеме бизнес-процесса, текущие значения переменных и ролей экземпляра бизнес-процесса, а также историю событий экземпляра бизнес-процесса. Если у пользователя есть соответствующие права, он может остановить выполнение экземпляра бизнес-процесса. Также в списке экземпляров бизнес-процессов пользователь может искать, группировать, фильтровать экземпляры бизнес-процессов, выводить в строках значения переменных бизнес-процессов.

Меню «Исполнители». На странице, соответствующей пункту меню «Исполнители», находится список потенциальных исполнителей заданий (пользователей и групп пользователей), доступных для чтения данному пользователю. На этой странице можно завести или удалить исполнителя, завести или удалить группу исполнителей, включить (исключить) исполнителя или группу исполнителей в другую группу. Также для исполнителя можно установить статус (Активен / Не активен) настроить список замещений.

Меню «Бот-станции». Боты в системе RunaWFE – это специальные компьютерные приложения, которые также как и люди могут быть исполнителями заданий. Бот-станция – это компьютерная среда, в которой функционируют боты. Находящиеся в бот-станции боты периодически опрашивают RunaWFE - сервер. Если выполняющиеся на сервере экземпляры бизнес-процессов содержат задачи для исполнителей - ботов, то боты выполняют эти задачи и возвращают результаты работы на RunaWFE - сервер. На странице, соответствующей пункту меню «Бот-станции», находится список зарегист-

рированных бот-станций. Здесь пользователь может посмотреть совйства бот-станций состояния бот-станций, свойства входящих в бот-станцию ботов, а также задачи, которые они могут выполнять. Также в меню «Бот-станции» можно завести новую бот-станцию, изменить параметры бот-станции, запустить/остановить периодическую активацию бот-станции, а также изменять свойства входящих в бот-станцию ботов. В частности можно добавить новое задание боту, или изменить/удалить уже существующее задание.

Меню «Система». На странице, соответствующей пункту меню «Система» находится список полномочий исполнителей на действия с системой, которые настраивает администратор.

Рассмотрим подробнее работу с RunaGPD в соотвествии с «RunaWFE. Графический редактор бизнес-процессов. Руководство пользователя».

Для работы с RunaGPD запустите на выполнение файл RunaWFE-Installer.exe (в случае дистрибутива в виде исполняемого файла). Следуйте инструкциям появившегося на экране мастера установки. В процессе установки не забудьте отметить галочкой компонент — графический редактор бизнес-процессов.

После установки RunaGPD с графическим редактором можно работать через меню системы: “Пуск/Программы/RunaWFE/Process designer” и значок с надписью “Process designer” на рабочем столе.

Для создания нового проекта бизнес-процесса необходимо выполнить следующие действия.

Выберите пункт меню **Файл > Создать > Новый проект** (см. рис. 6.10). В результате активации данного пункта появится Wizard заведения нового проекта процессов (см. рис. 6.11).

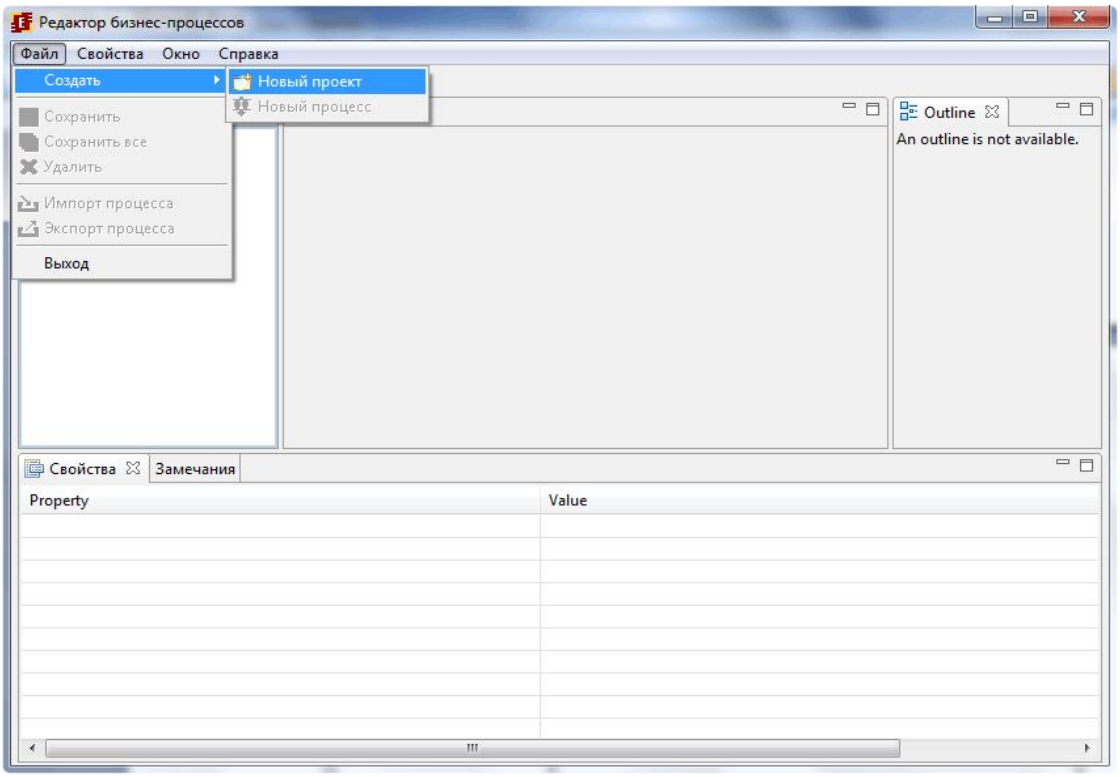


Рис. 6.10. – Начало создания нового проекта в RunaGPD.

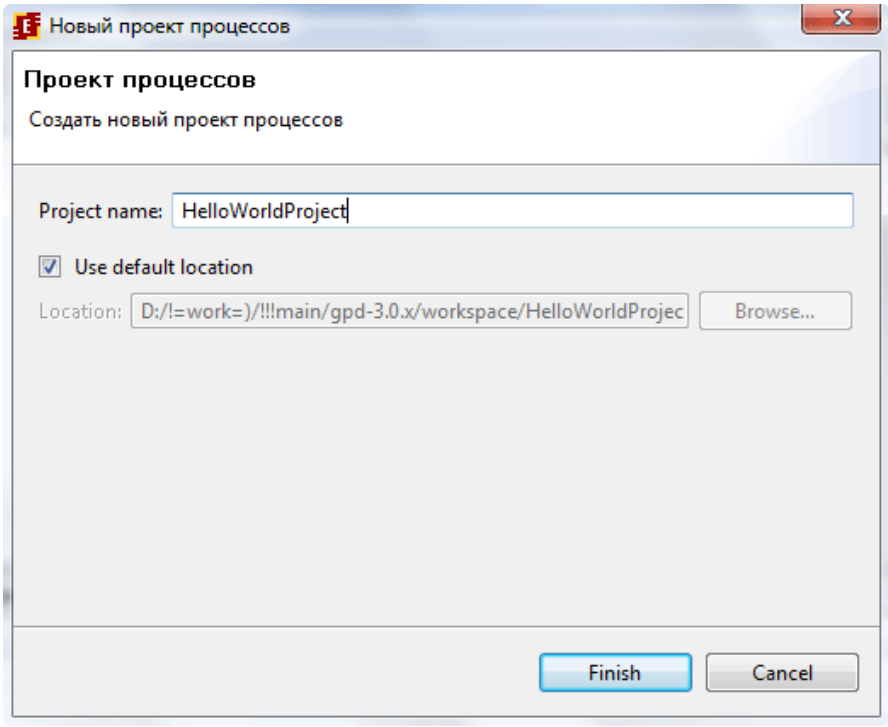


Рис. 6.11. – Wizard заведения нового проекта процессов.

Введите в соотвествующее поле имя проекта (например, “HelloWorldProject”, см. рис. 6.11). В результате этого действия будет создан новый проект с именем «HelloWorldProject» (см. рис. 6.12).

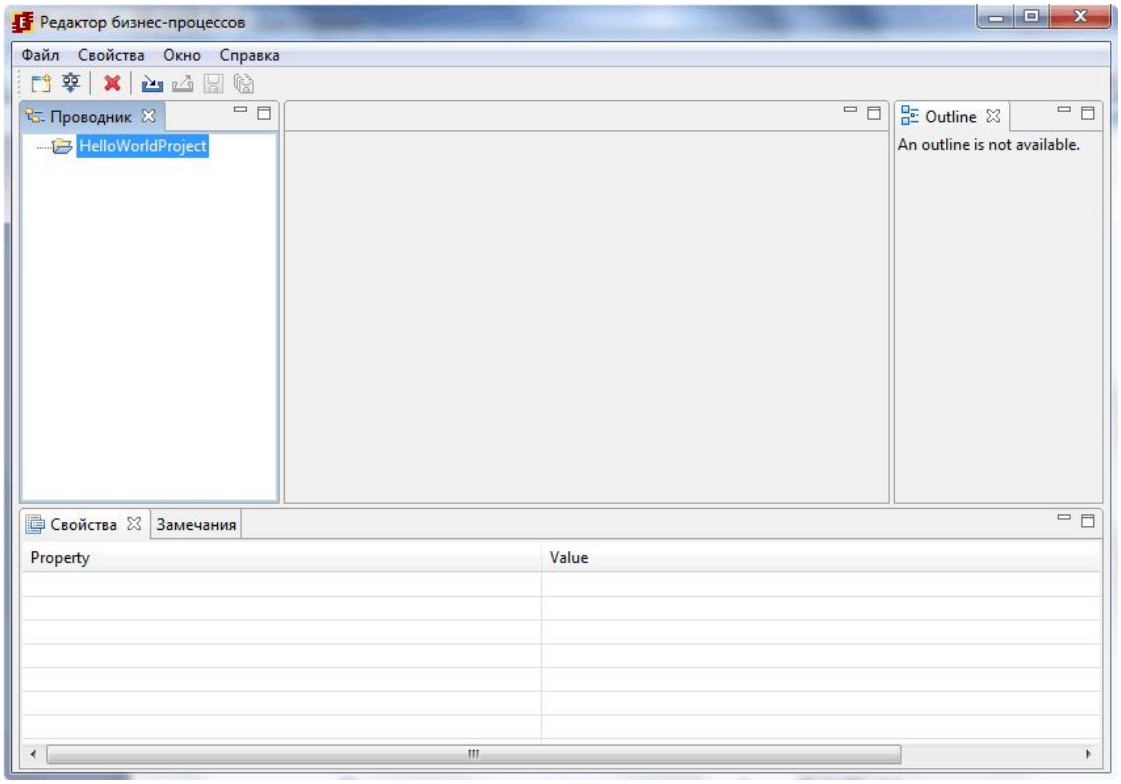


Рис. 6.12. – Новый проект бизнес-процесса создан.

Для создания графа нового бизнес-процесса необходимо выполнить следующие действия.

Откройте контекстное меню правой кнопкой мыши, кликнув на HelloWorldProject, затем кликните на «**Новый процесс**» (см. рис. 6.13).

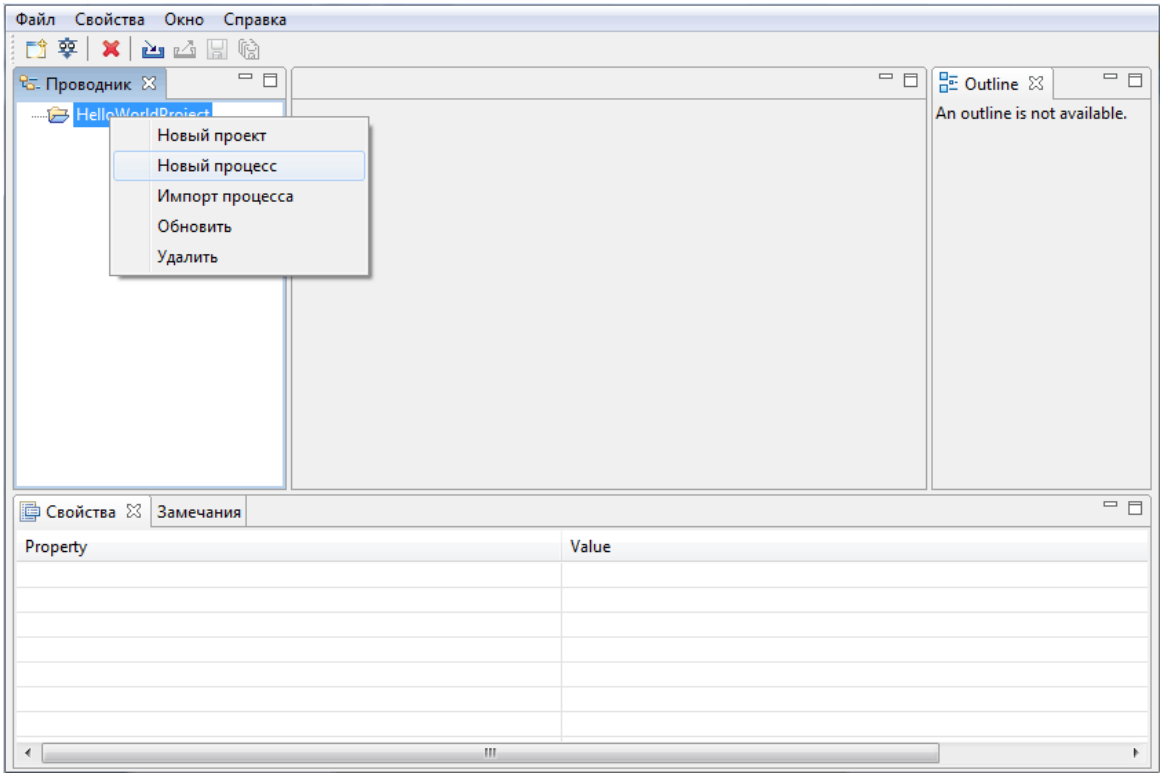


Рис. 6.13. – Начало создания графа нового бизнес-процесса в RunaGPD.

В результате активации данного пункта появится Wizard создания графа нового бизнес-процесса (см. рис. 6.14). Введите «HelloWorldProcess» в качестве имени процесса как показано на рис. 6.14.

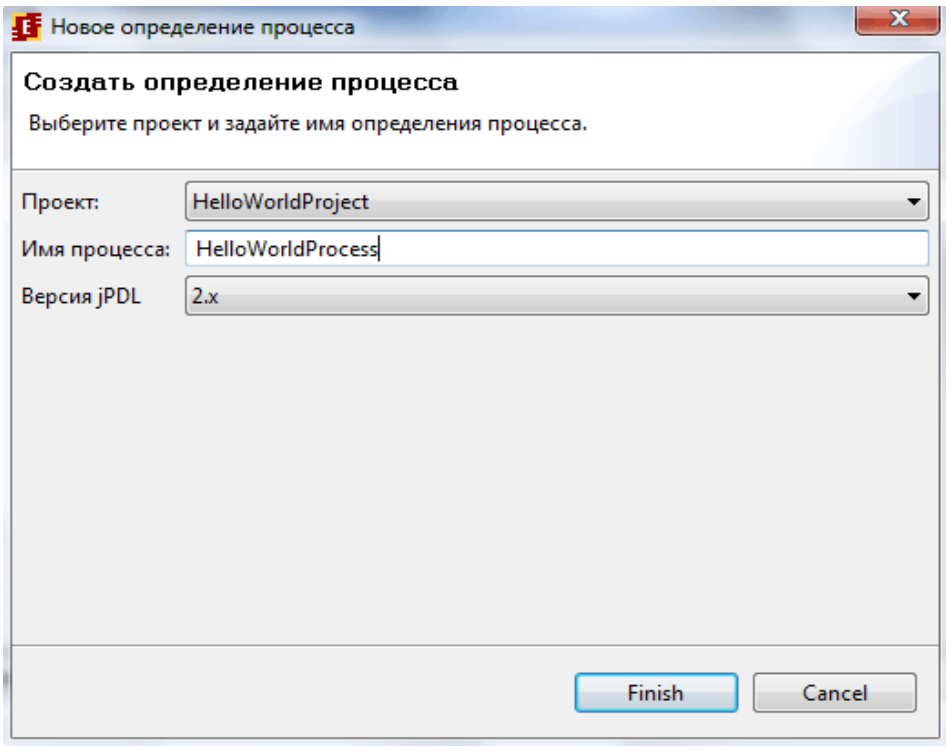


Рис. 6.14. – Wizard создания графа нового бизнес-процесса.

В результате нажатия кнопки «**Finish**». Будет создан процесс «HelloWorldProcess».

В результате двойного щелчка на HelloWorldProcess откроется диаграмма процесса (см. рис. 6.15).

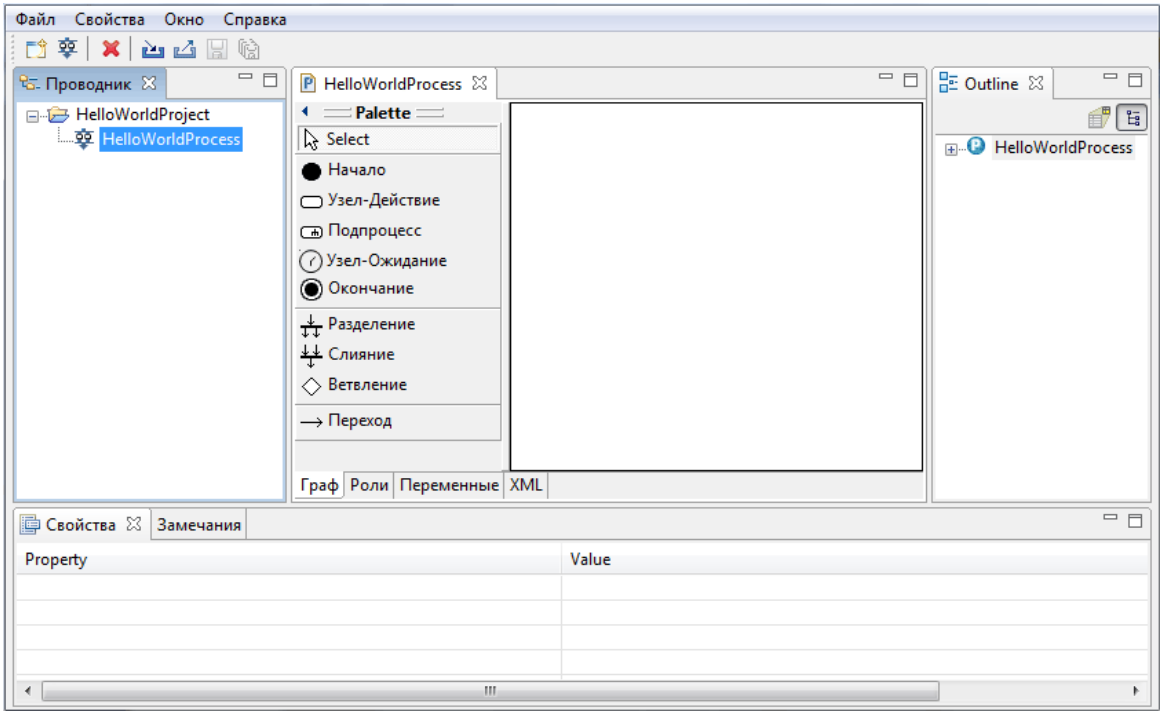


Рис. 6.15. – Окно диаграммы нового бизнес-процесса.

В результате клика на **Свойства/Показать сетку**, появится сетка (см. рис. 6.16).

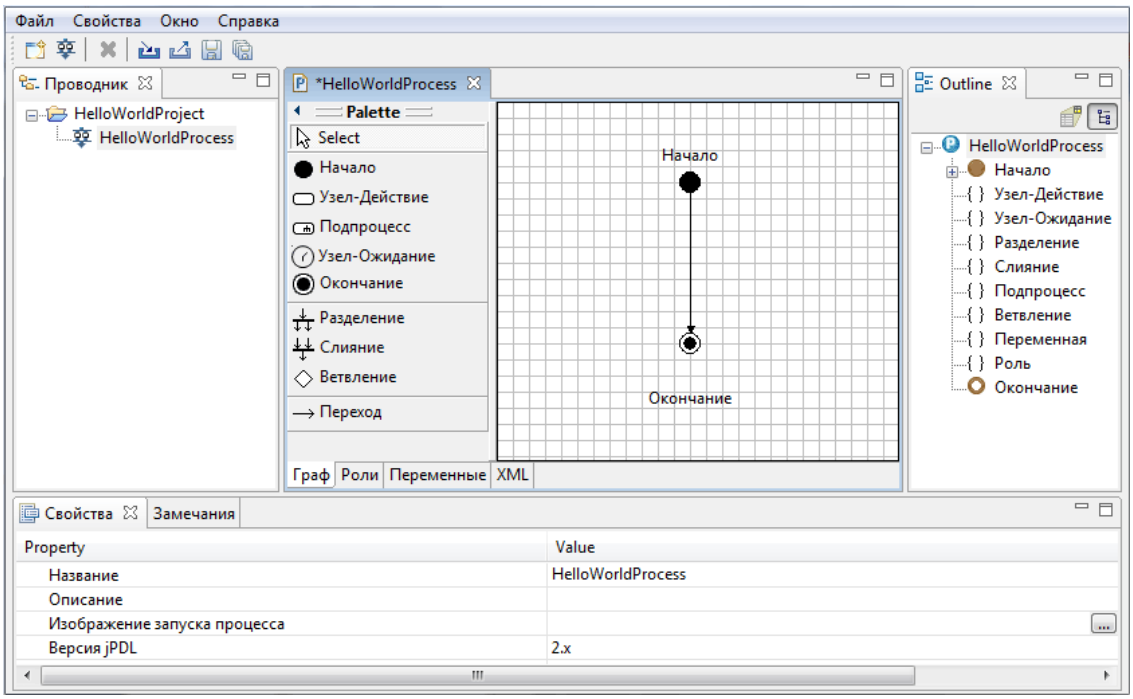


Рис. 6.16. – Диаграмма бизнес-процесса.

Кликните на элементе “**Начало**” палитры, затем кликните на окне диаграммы. Стартовый узел процесса появится в окне диаграммы. Аналогично поместите на диаграмму завершающий узел процесса “**Окончание**”. Кликните на элемент “**Переход**” и соедините узлы «Начало» и «Окончание» (см. рис. 6.16). В соответствующих полях свойств можно ввести краткое описание процесса и пиктограмму изображения процесса. Граф бизнес-процесса готов.

Подробнее разработка диаграмм в BPMN в среде RunaGPD рассмотрена в соотвествующей лабораторной работе (см. лабораторный практикум по данному курсу).

6.4.3. Примеры моделей в нотации BPMN.

Моделирование бизнес-процессов в нотации BPMN используется для донесения широкого спектра информации до различных категорий пользователей. BPD-диаграммы бизнес-процессов позволяют создавать три типа моделей для описания бизнес-процессов:

- Частные (внутренние) бизнес-процессы.
- Абстрактные (открытые) бизнес-процессы.
- Процессы взаимодействия (глобальные).

Частные бизнес-процессы описывают внутреннюю деятельность организации. Они представляют бизнес-процессы в общепринятом понимании (business processes или workflows). При использовании ролей частный бизнес-процесс помещается в отдельный пул. Поэтому поток управления находится внутри одного пула и не может пересекать его границ. Поток сообщений, напротив, пересекает границы пулов для отображения взаимодействия между разными частными бизнес-процессами.

Абстрактные (открытые) бизнес-процессы служат для отображения взаимодействия между двумя частным бизнес-процессами (то есть между двумя участниками взаимодействия) В открытом бизнес-процессе показываются только те действия, которые участвуют в коммуникации с другими процессами. Все другие, «внутренние», действия частного бизнес-процесса не показываются в абстрактном процессе. Таким образом, абстрактный процесс показывает окружающим последовательность событий, с помощью которой можно взаимодействовать с данным бизнес-процессом. Абстрактные процессы помещаются в пулы и могут моделироваться как отдельно, так и внутри большей диаграммы для отображения потока сообщений между действиями абстрактного процесса с другими элементами. Если абстрактный процесс и соответствующий частный процесс находятся в одной диаграмме, то действия, отображённые в обоих процессах, могут быть связаны ассоциациями.

Процессы взаимодействия (глобальные) отображает взаимодействия между двумя и более сущностями. Эти взаимодействия определяются последовательностью действий, обрабатывающих сообщения между участниками. Процессы взаимодействия могут помещаться в пул. Эти процессы могут моделироваться как отдельно, так и внутри большей диаграммы для отображе-

ния ассоциаций между действиями и другими сущностями. Если процесс взаимодействия и соответствующий частный процесс находятся в одной диаграмме, то действия, отображённые в обоих процессах, могут быть связаны ассоциациями.

Рассматрим далее пример бизнес-процесса «Регистрация на рейс».

Сначала приводится словесное описание процесса, а потом один из вариантов его представления в BPMN. Данный пример не гарантирует максимального приближёния к реальному процессу, а ставит целью показать использование конструкций нотации BPMN.

Когда пассажир прибывает в аэропорт, его приоритетной задачей является регистрация на рейс. Сотрудник на стойке регистрации приветствует клиента и берёт у него документы: билет на рейс и паспорт. Если документы клиента не в порядке (например, истёк срок действия паспорта), он не может быть зарегистрирован на рейс и процесс завершается. При этом клиент получает документы обратно.

Если паспорт и билет в порядке, то сотрудник авиакомпании регистрирует клиента на рейс и распечатывает посадочный талон. При этом он взаимодействует с информационной системой авиакомпании. Сотрудник отдаёт пассажиру посадочный талон и паспорт, после чего уточняет, нет ли в багаже пассажира запрещённых грузов (например, воспламеняющихся веществ). Если таковые есть, то они изымаются из багажа. Сотрудник авиакомпании забирает багаж и ручную кладь пассажира и регистрирует её. При этом сотрудник снова взаимодействует с информационной системой авиакомпании. Если выясняется, что есть перевес, то сотрудник уведомляет об этом пассажира и сообщает сколько необходимо заплатить. После получения денег от пассажира, сотрудник регистрирует оплату в системе.

В итоге, пассажир получает багажную квитанцию. Сотрудник желает пассажиру приятного полёта, и процесс завершается.

На рисунке 6.4 приведен пример представления упомянутого выше словесного описания бизнес-процесса в виде BPD-диаграммы.

Далее на рисунках 6.5 – 6.7 приведены примеры BPD-диаграмм других бизнес-процессов.

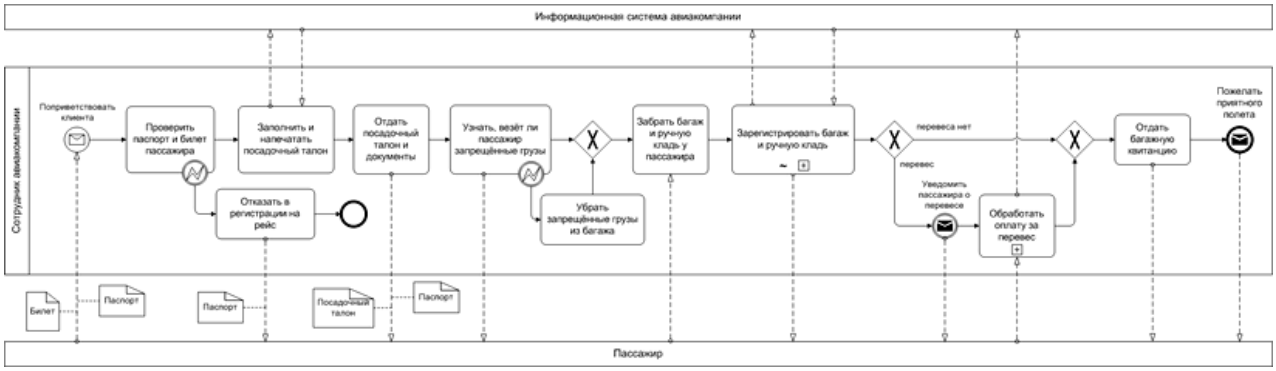


Рис.6.4. – Пример диаграммы в нотации BPMN. Регистрация на рейс.

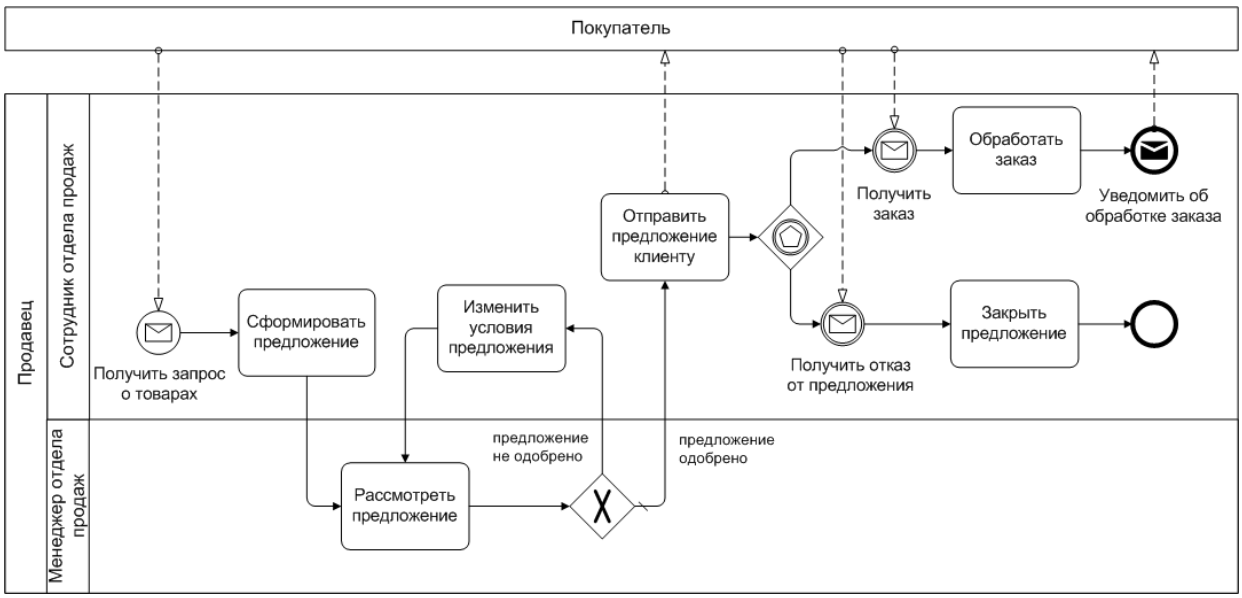


Рис. 6.5. – Пример диаграммы в нотации BPMN. Обработка запроса о товарах.

6.4.4. Недостатки моделирования в нотации BPMN.

Анализируя графический язык моделирования бизнес-процессов в BPMN-нотации, можно обратить внимание на следующие недостатки:

1. Для освоения данной нотации требуются курсы, консультации и время для ее изучения на уровне достаточном для практического использования. Маловероятно, что эта нотация будет хорошо понимаема управленцами.

2. Очень сложно моделировать большие иерархические системы, к которым относятся серьезные организации. А ведь именно для их моделирования такие средства и нужны в первую очередь.

3. Соединяющие элементы предназначены только для отображения порядка выполенения действий или появления событий и не отображают материальные и информационные потоки. Бизнес-процессы же, по сути своей, всегда являются проточными элементами, пропускающими через себя потоки материи и информации.

4. Введение элементов «Событие» и «Объект данных», представляющих, по сути дела, некоторые специфические виды связей. Использование, таким образом, избыточных сущностей, затрудняющих понимание диаграмм.

5. Авторы нотации BPMN утверждают, что данная нотация не предназначена для построения функциональных диаграмм и представления бизнес-правил. Но это может означать только то, что данная нотация вообще не предназначена для моделирования бизнес-процессов (хотя она, вроде бы, для этого и сделана), так как бизнес-процессы, по сути своей, всегда функциональны, а удачные модели бизнес-процессов представляют собой бизнес-правила.

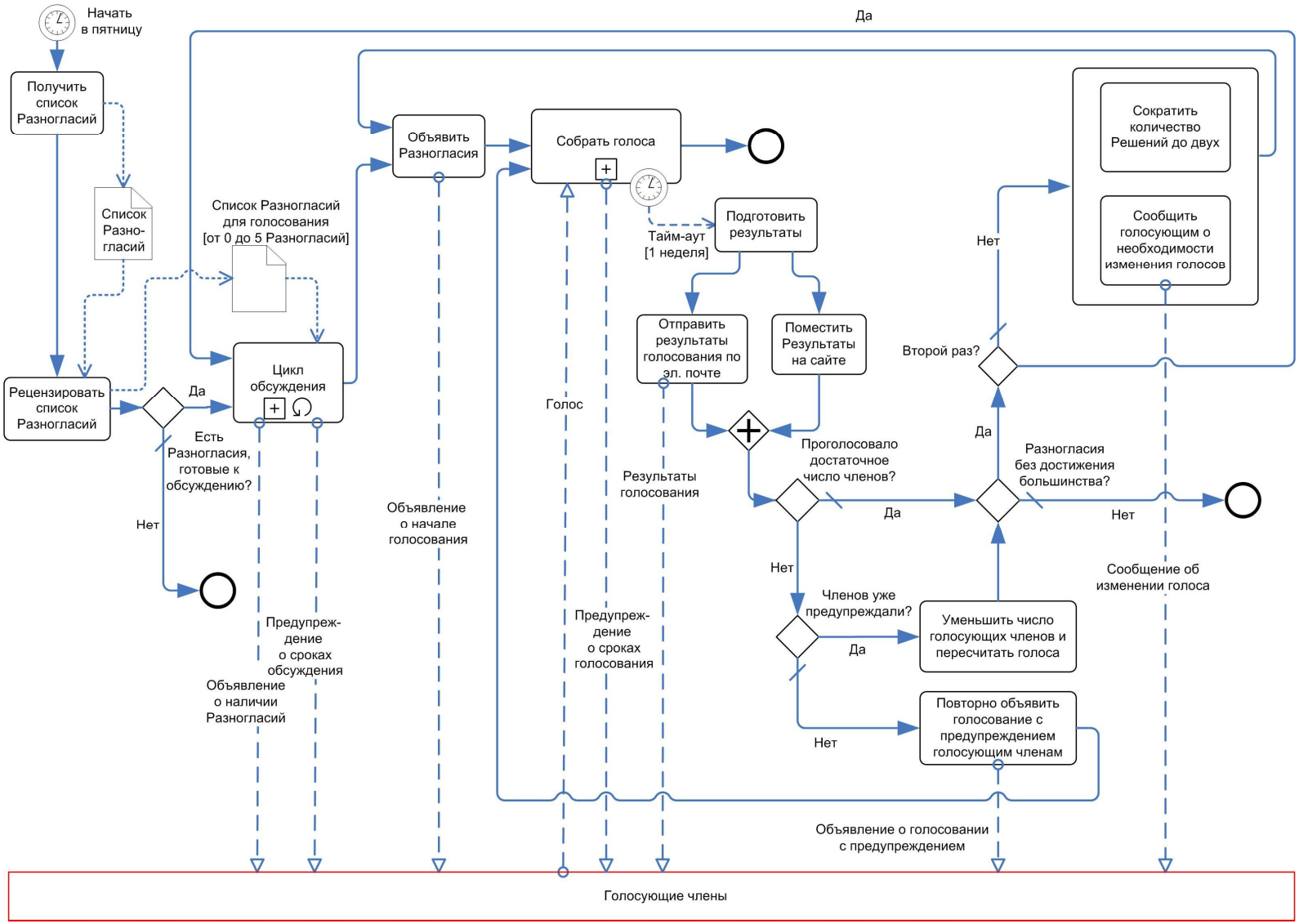


Рис. 6.6. – Пример диаграммы в нотации BPMN. Голосование по электронной почте.

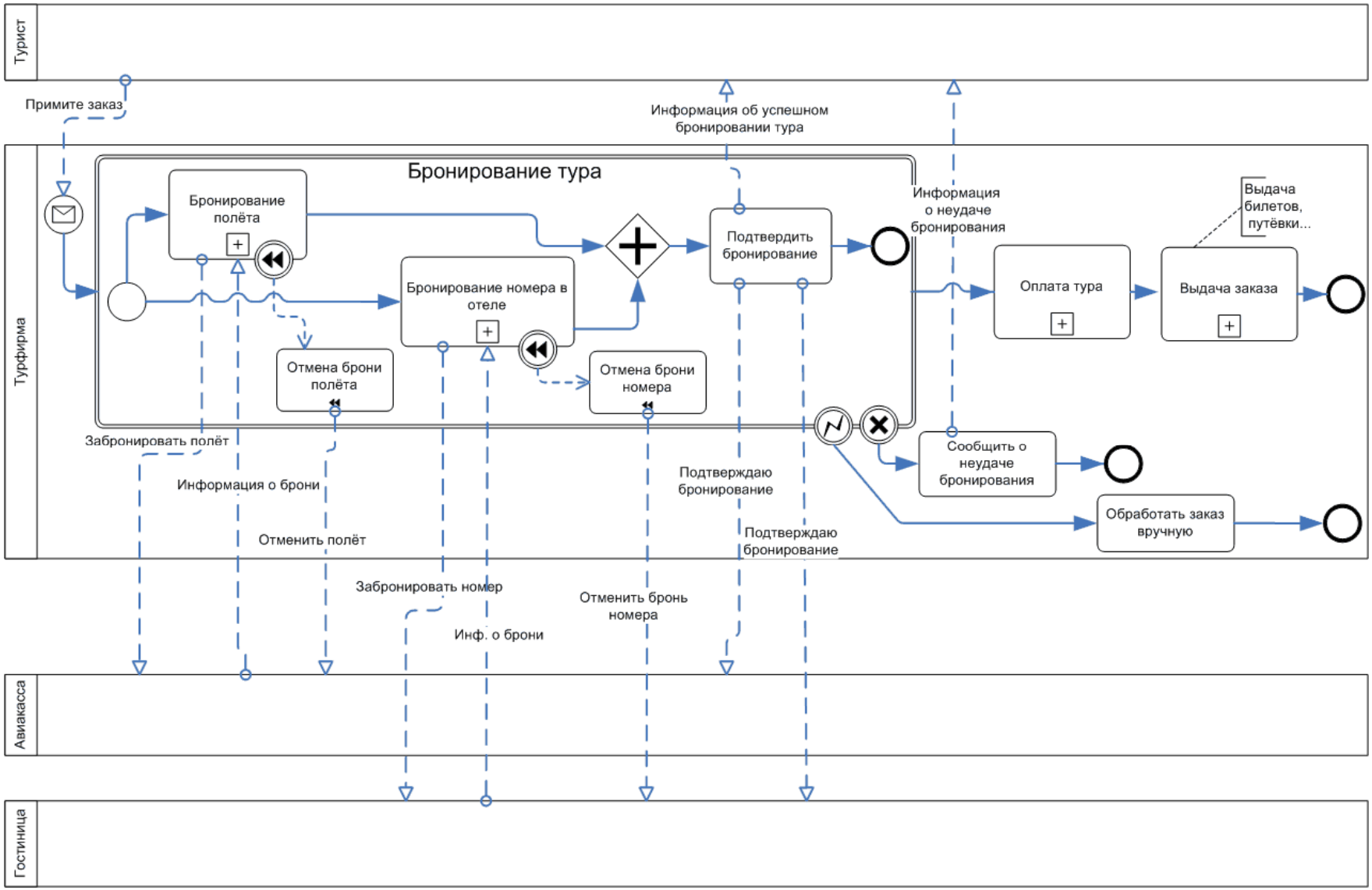


Рис. 6.7. – Пример диаграммы в нотации BPMN. Деятельность турфирмы.

Вопросы для повторения

1. Назовите основные категории элементов BPMN.
2. Опишите элементы потока BPD-диаграмм «События».
3. Опишите элементы потока BPD-диаграмм «Действия».
4. Опишите элементы потока BPD-диаграмм «Шлюзы».
5. Опишите соединяющие элементы BPD-диаграмм.
6. Опишите зоны отвественности и артефекты BPD-диаграмм.
7. Назовите правила соединения элементов потока с помощью «Потока операций».
8. Назовите правила соединения элементов потока с помощью «Потока сообщений».
9. Что такое BPEL?
10. Что такое XPDL?
11. Как взаимосвязаны стандарты BPMN, BPEL и XPDL?
12. Привидите пример диаграммы в нотации BPMN.
13. Нозовите примеры CASE-инструментария, поддерживающего нотацию BPMN.
14. Назовите и объясните недостатки нотации BPMN.

Резюме по теме

В данном разделе подробно рассмотрен стандарт моделирования бизнес-процессов BPMN. Описаны элементы BPD-диаграмм (нотация BPMN). Приведены примеры моделей бизнес-процессов в данной нотации. Рассмотрены программные CASE-средства, поддерживающие построение диаграмм в нотации BPMN. Проанализированы связь стандарта BPMN со стандартами BPEL и XPDL, а также недостатки BPMN-нотации.

Вместо заключения

В настоящее время актуальной является задача создания единых теоретических основ представления организационных знаний и управления ими за счет интеграции и универсализации существующих способов представления таких знаний на единой основе. Для решения данной задачи предлагается способ преобразования знаний, представляемых в настоящее время в виде DFD-моделей, моделей в стандартах серии IDEF, а также моделей в стандарте BPMN к единому виду на основе системно-объектных моделей в терминах «Узел-Функция-Объект» (УФО-моделей). Формализация данного единого универсального системного способа представления знаний позволит создать соответствующие алгоритмы и инструментальных средства на базе оригинального пакета «UFO-toolkit» для обработки знаний различного типа единообразным способом и с помощью единого формального аппарата.

Представление DFD-диаграммы с помощью УФО-модели

Для обеспечения такого представления используем соответствие между графическими элементами DFD-нотации и УФО-моделей показанное в таблице 3.1.

Таблица 3.1. Соответствие графических элементов DFD и УФО.

Описание элемента	*Элементы DFD*	*Элементы УФО*
Поток данных Используется для моделирования передачи информации (или даже физических компонент) из одной части системы в другую.	«имя»	Имя
Процесс Используется для моделирования процесса преобразования входного потока в выходной.	«имя» «номер»	Номер Имя
Хранилище (накопитель) данных Используется для моделирования данных (или даже физических компонент), которые будут сохраняться между процессами.	«имя»	Хранилище Имя
Внешняя сущность (терминатор) Используется для моделирования сущностей вне системы (контекстных сущностей), являющихся источником или приемником системных данных.	«имя»	Имя

Рассмотрим пример модели в нотации DFD (см. рисунки 3.1 и 3.2).

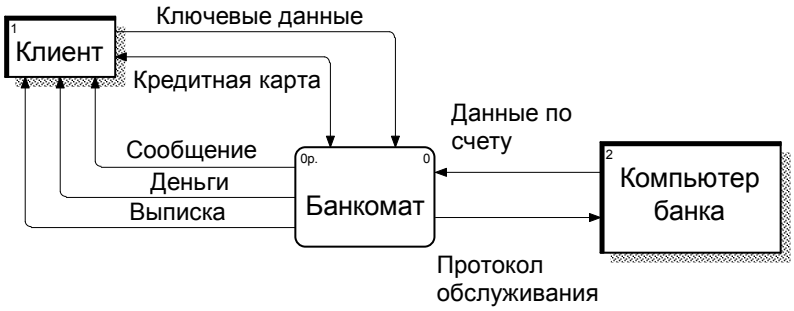


Рис. 3.1. - Пример контекстной диаграммы в нотации DFD.

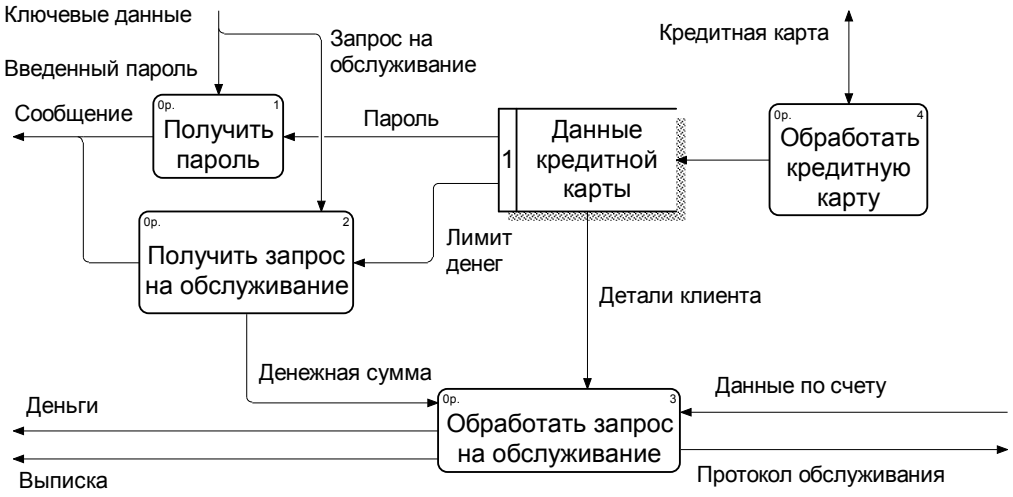


Рис. 3.2. - Пример диаграммы декомпозиции в нотации DFD.

Преобразуем представленные на рисунках 3.1 и 3.2 DFD-диаграммы в УФО-модели, используя соответствия между графическими элементами (см.табл. 3.1) В результате преобразования (см. рис. 3.3 и 3.4) можно утверждать, что УФО-модель будет соответствовать DFD-диаграмме если в ней:

- для всех УФО-элементов определены функции;
- для контекстных УФО-элементов определены еще и объекты;
- выделен специфический УФО-элемент, представляющий собой функциональный узел для отображения какого-либо хранилища;
- введены служебные УФО-элементы, определенные только на уровне узлов, для обеспечения соединения и разветвления потоков.

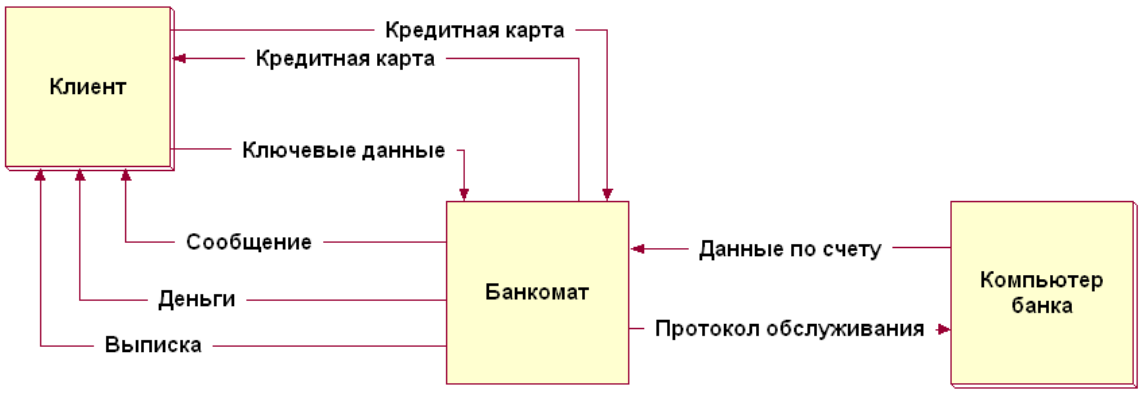


Рис. 3.3. – Диаграмма на рис. 3.1 в виде модели «Узел-Функция-Объект».

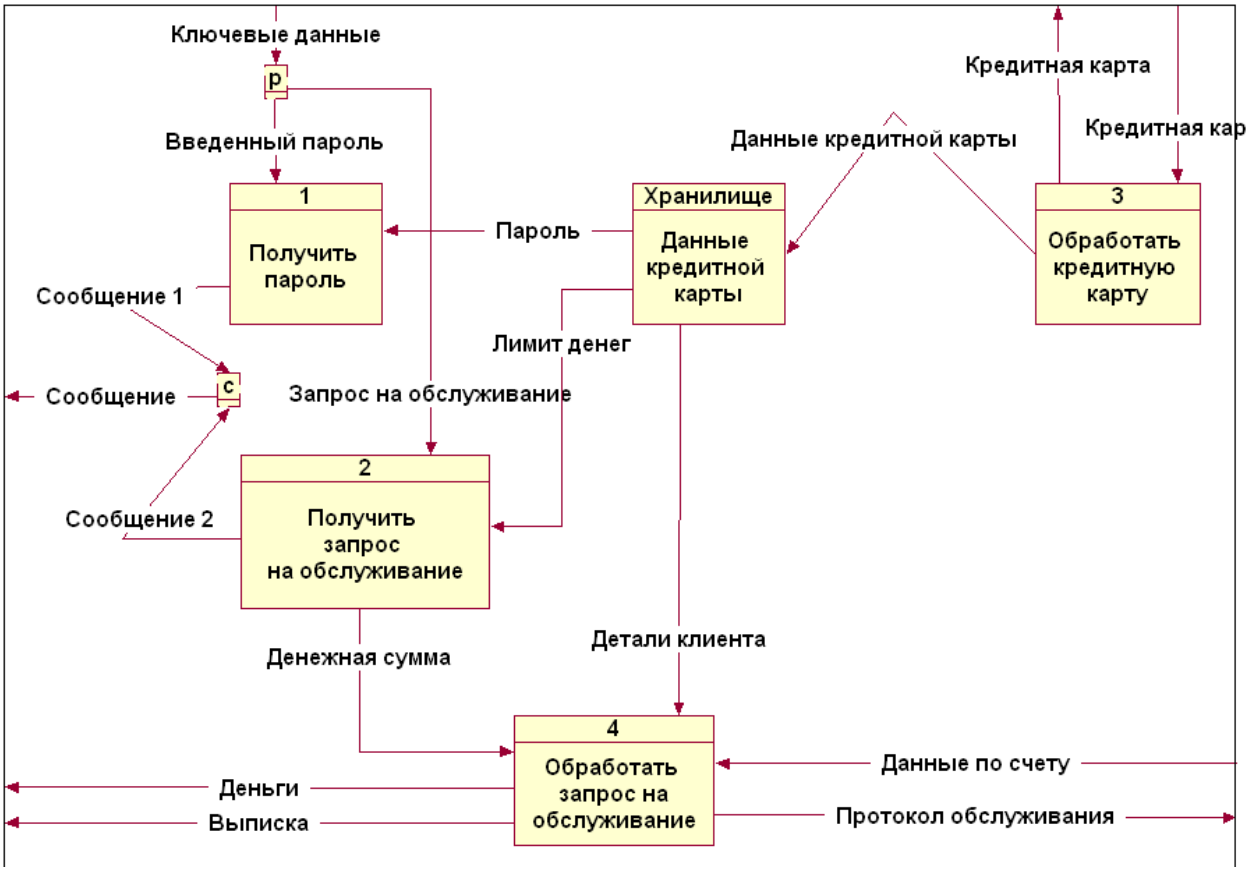


Рис. 3.4. - Диаграмма на рис. 3.2 в виде модели «Узел-Функция-Объект».

Представление IDEF0-диаграммы с помощью УФО-модели.

Для обеспечения такого представления используем соответствие между графическими элементами IDEF0-нотации и УФО-моделей показанное на рисунках 3.5 и 3.6.

Рис. 3.5. – Функциональный блок в нотации IDEF0.

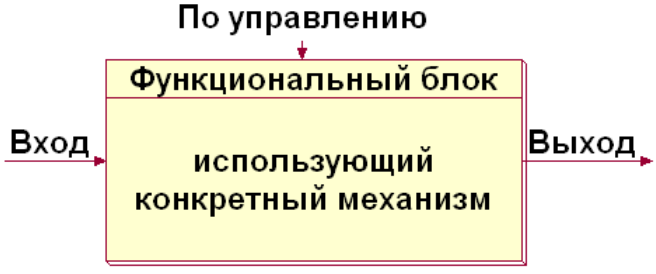


Рис. 3.6. – Функциональный блок IDEF0 в виде модели «Узел-Функция-Объект».

Рассмотрим пример модели в нотации IDEF0 (см. рис. 3.7 и 3.8).

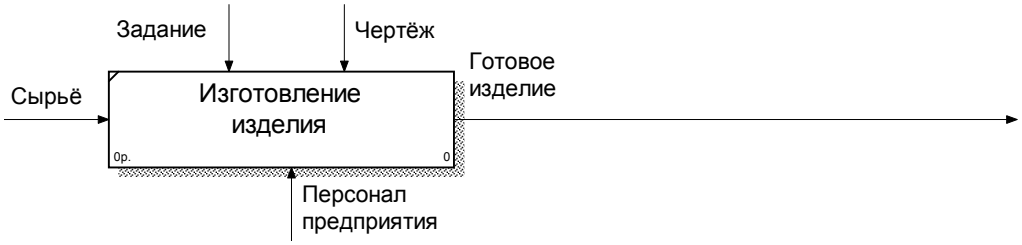


Рис. 3.7. - Пример контекстной диаграммы в нотации IDEF0.

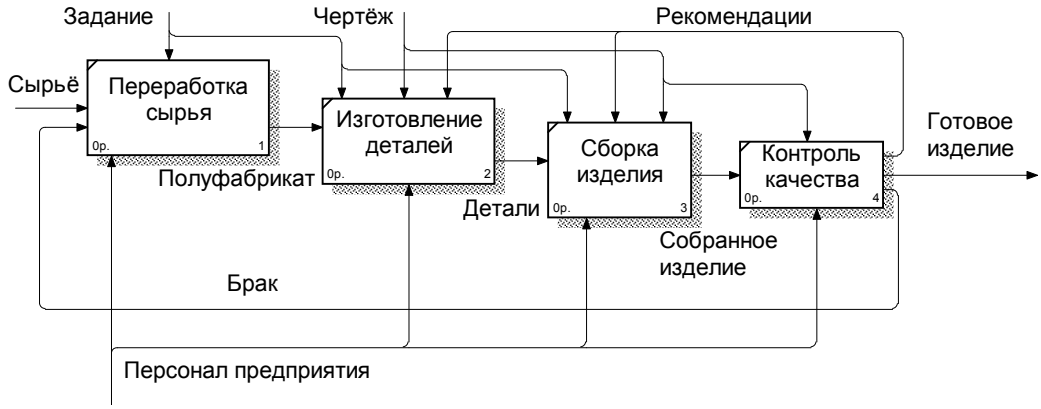


Рис. 3.8. - Пример диаграммы декомпозиции в нотации IDEF0.

Преобразуем представленные на рисунках 3.7 и 3.8 IDEF0-диаграммы в УФО-модели, используя соответствия между графическими элементами. В результате преобразования (см. рис. 3.9 и 3.10) можно утверждать, что УФО-модель будет соответствовать IDEF0-диаграмме если в ней:

- для всех УФО-элементов определены функции;
- для всех УФО-элементов определены объекты и их определения соответствуют связи «Механизм»;
- нижняя граница УФО-элемента для связи не используется;
- все управляющие связи прикрепляются только к верхней границе УФО-элемента;
- для входов в УФО-элементы используется только левая граница;
- для выходов из УФО-элементов используется только правая граница;
- введены служебные УФО-элементы, определенные только на уровне узлов, для обеспечения соединения и разветвления потоков.

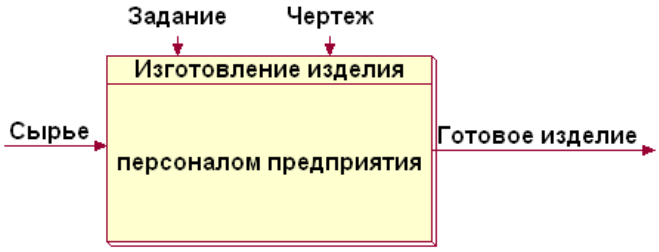


Рис. 3.9. - Диаграмма на рис. 3.7 в виде модели «Узел-Функция-Объект».

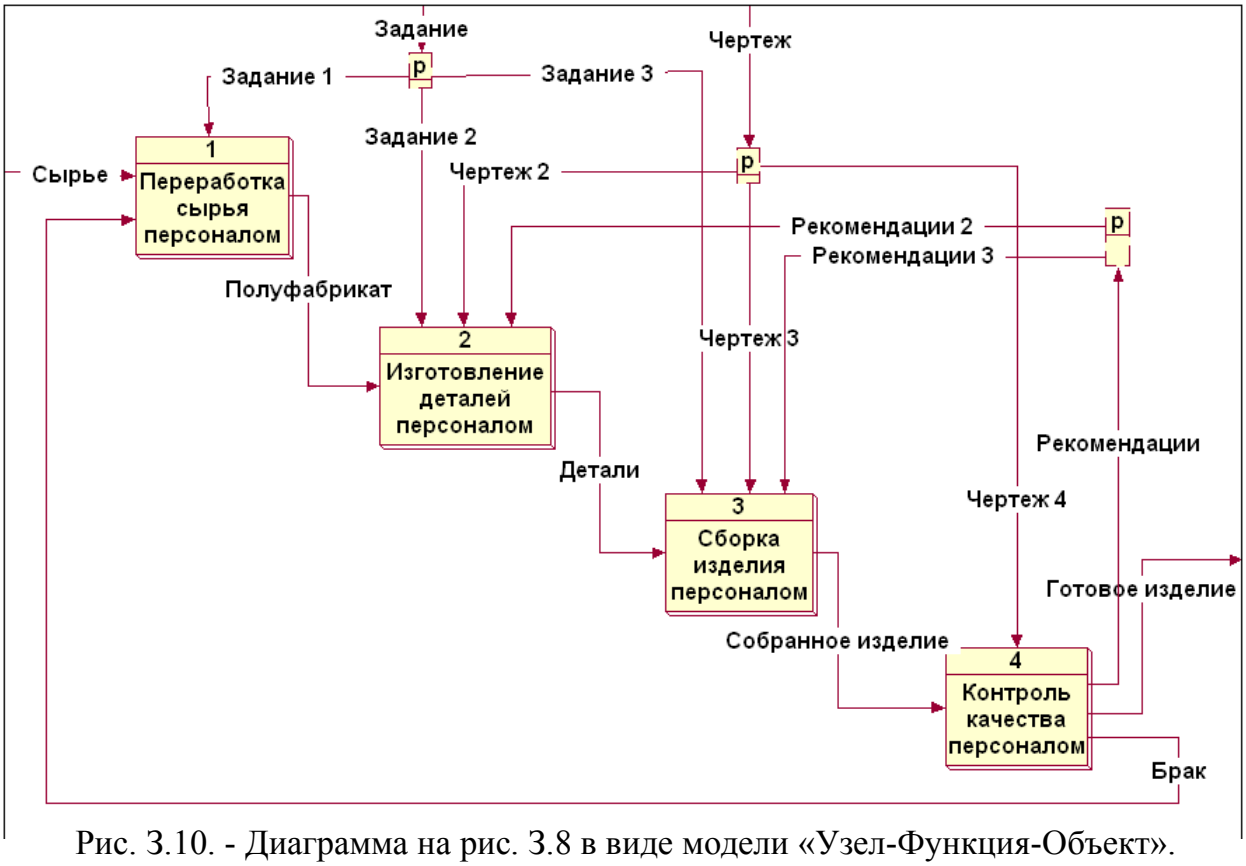


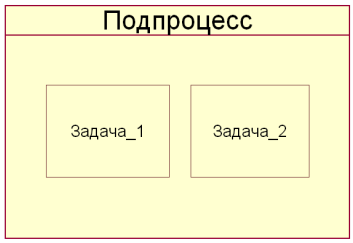
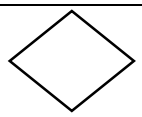
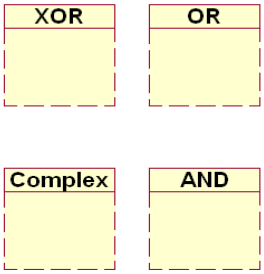
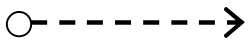
Рис. 3.10. - Диаграмма на рис. 3.8 в виде модели «Узел-Функция-Объект».

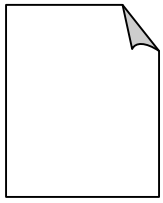
Представление BPMN-диаграммы с помощью УФО-модели.

Для обеспечения такого представления используем соответствие между графическими элементами BPMN-нотации и УФО-моделей показанное в таблице 3.2.

Таблица 3.2. Соответствие графических элементов BPMN и УФО.

Описание элемента	***Элементы BPMN***	***Элементы УФО***
Событие (Event) Событие – это то, что происходит в течение бизнес-процесса и оказывает влияние на его ход. Чаще всего событие имеет причину (триггер) или воздействие (результат). Согласно влиянию Событий на ход бизнес-процесса, выделяют три типа: Стартовое событие (Start), Промежуточное событие (Intermediate) и Конечное событие (End).	**Маркеры (триггеры) событий:** -сообщение, -таймер, -ошибка, -отмена, -компенсация, -условие\правило, -сигнал.	Сообщение Таймер Ошибка Отмена Компенсация Условие/правило Сигнал

Описание элемента	*Элементы BPMN*	*Элементы УФО*
Действие (Activity) Действие – общий термин, обозначающий работу, выполняемую исполнителем. Действия могут быть либо элементарными, либо неэлементарными (составными). Выделяют следующие виды действий, являющихся частью модели Процесса: Процесс (Process), Подпроцесс (Sub-Process) и Задача (Task).		
Шлюз (Gateway) Шлюзы используются для контроля расхождений и схождений потока операций. Таким образом, данный термин подразумевает ветвление, раздвоение, слияние и соединение маршрутов. Внутренние маркеры указывают тип контроля развития бизнес-процесса.	 **Типы шлюзов:** -Эксклюзивные ИЛИ (XOR); -ИЛИ (OR); -Комплексные (Complex); -И (AND).	
Поток операций (Sequence Flow) Поток операций служит для отображения того порядка, в котором организованы действия Процесса.		
Поток сообщений (Message Flow) Поток сообщений служит для отображения обмена сообщениями между двумя участниками, готовыми эти сообщения отсылать и принимать. На диаграмме BPMN два отдельно взятых Пула представляют собой двух участников процесса.		

Описание элемента	*Элементы BPMN*	*Элементы УФО*
Объект данных (Data Object) Объекты данных рассматриваются как артефакты, так как они не влияют непосредственно на последовательный поток или поток сообщений процесса, но они обеспечивают ввод информации о том, какие действия требуют выполнения и/или что они производят.		

Рассмотрим пример модели в нотации BPMN (см. рис. 3.11).

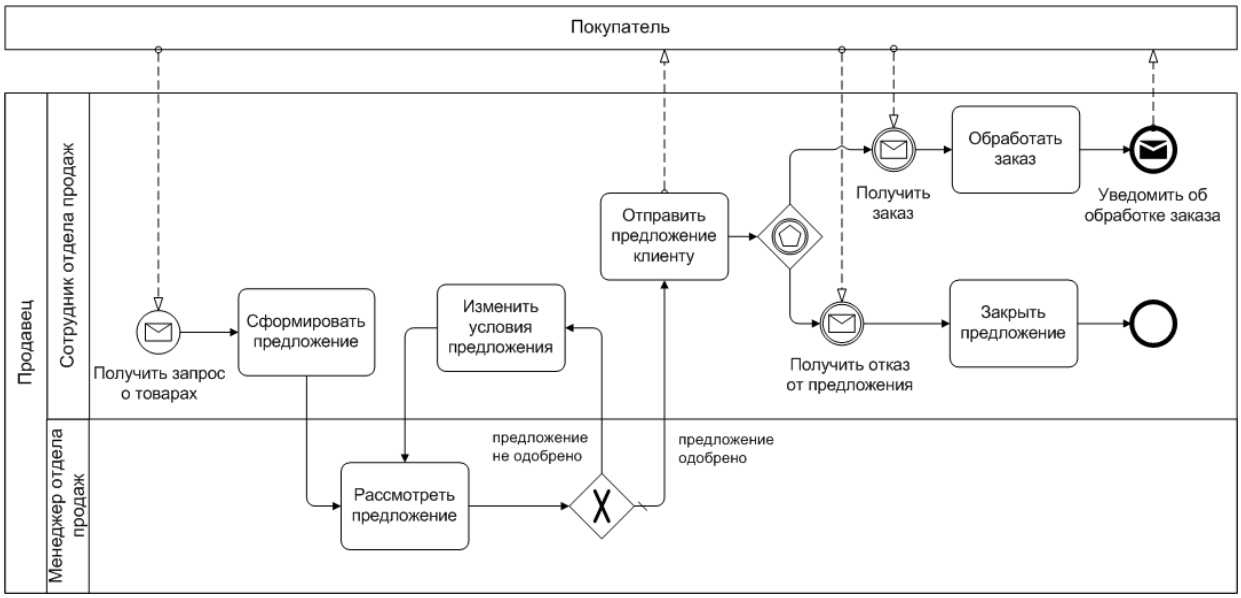


Рис. 3.11. – Пример диаграммы в нотации BPMN.

Преобразуем представленную на рисунке 3.11 BPMN-диаграмму в УФО-модель, используя соответствия между графическими элементами. Результаты представлены на рисунках 3.12 – 3.14. В результате выполненного преобразования можно утверждать, что УФО-модель будет соответствовать BPMN-диаграмме если в ней:

– в классификацию, в категорию связей «По управлению (С)» введен абстрактный класс связей «Событие», разделенный на подклассы связей, соответствующие маркерам (триггерам) событий (так как элемент «Событие» в нотации BPMN, по сути дела, представляет связи/потоки или поступающие на обработку (на вход процесса), или генерируемые процессом (поступающие на выход));

– УФО-элементы в модели определены на уровне функций;
– введены специальные/служебные УФО-элементы, определенные на уровне узлов, обозначающие логические операции, обеспечивающие схождение и расхождение потоков;
– все потоки в BPMN-модели (операций и сообщений) представляются в УФО-модели связями из классификации (так как действия в процессах не могут просто так переходить одно в другое, они всегда обмениваются материей и информацией);
– элемент BPMN-модели «Объект данных» представляется в УФО-модели определенного вида связью из категории связей «По данным (D)»;
– пулы и дорожки BPMN-диаграммы представляются в УФО-модели УФО-элементами, определенными на функциональном уровне.

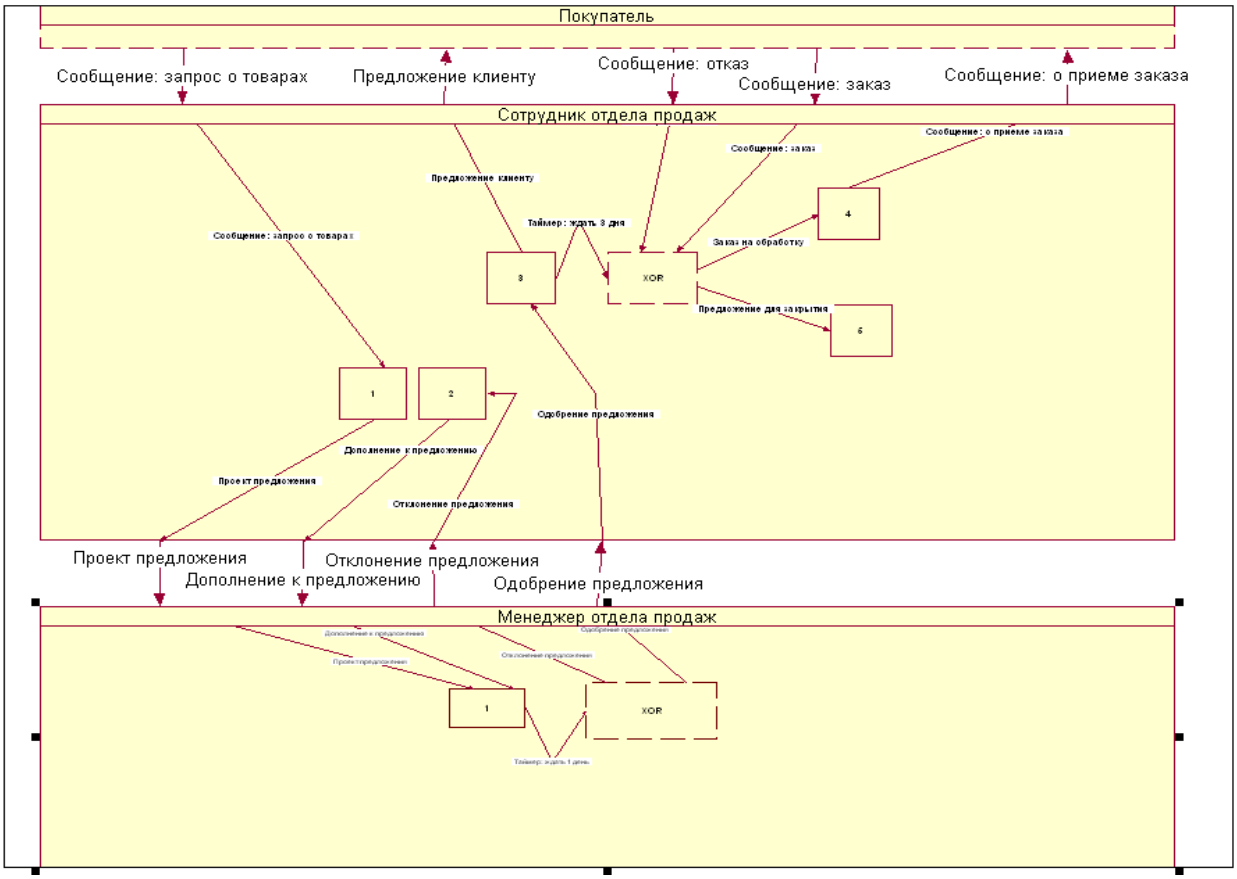


Рис. 3.12. - Диаграмма на рис.3.11 в виде модели «Узел-Функция-Объект».

Показанное соответствие графических элементов некоторых графоаналитических нотаций (так называемых WF-спецификаций) элементам системно-объектных моделей, а также приведенные примеры преобразования диаграмм в этих нотациях в модели «Узел-Функция-Объект» показывают универсальность УФО-моделей. Таким образом, УФО-подход позволяет моделировать любые процессы и системы без ограничений и способен заменить собой любую существующую нотацию бизнес-моделирования. Учитывая, что

возможна формализация УФО-подхода с помощью алгебраических средств (теории паттернов и теории процессов), можно говорить об УФО-моделировании как о едином универсальном способе представления организационных знаний. Данное обстоятельство обосновывает мнение отечественных специалистов по WF-языкам о том, что: "Еще нет WF-спецификации, с которой не было бы связано серьезных проблем, лидеры в этой области пока выглядят неоправданно сложными. Возможно, реальным WF-стандартом станет еще только разрабатываемая спецификация» [116].

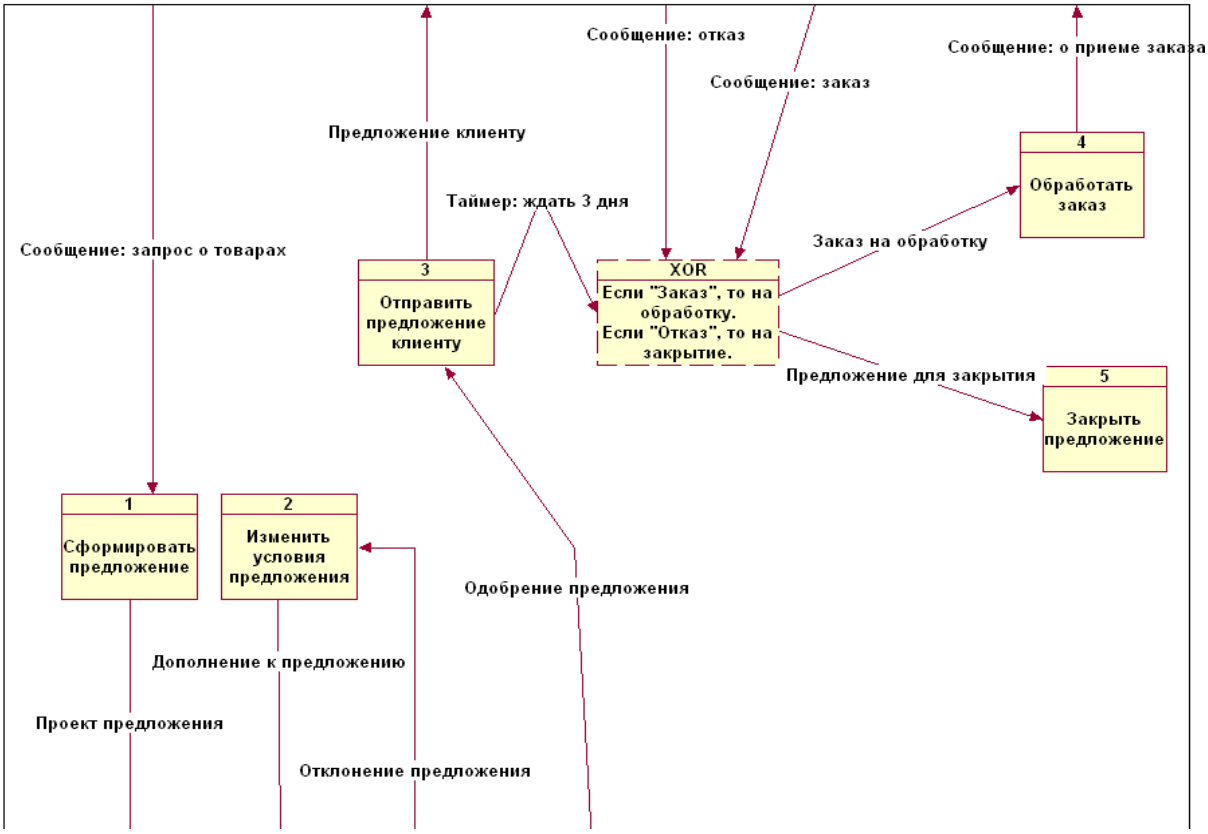


Рис. 3.13. – Дополнение к диаграмме на рис. 3.12: «Сотрудник отдела продаж».

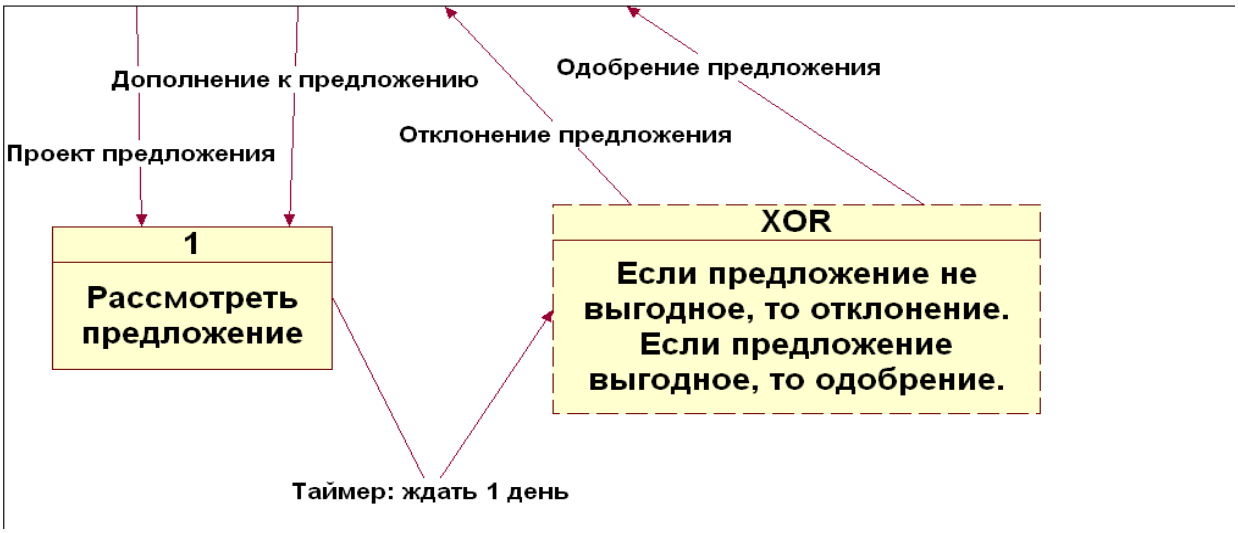


Рис. 3.14. – Дополнение к диаграмме на рис.3.12: «Менеджер отдела продаж».

Глоссарий

Activity-Based Costing	Технология, применяемая для оценки затрат и используемых ресурсов.
AllFusion	Линейка CASE-инструментов для поддержки всех стадий разработки программного обеспечения. Туда, в частности, входит линейка CASE-средств AllFusion Modeling Suite (ERwin, BPwin, ModelMart, Paradigm Plus, ERwin, Examiner) и средства управления проектами.
AllFusion Process Modeler (BPwin)	CASE-инструмент моделирования, который используется для анализа, документирования и реорганизации сложных бизнес-процессов.
Arena	CASE-средство имитационного моделирования.
CASE (Computer-Aided Software Engineering)	Технология поддержки жизненного цикла программного обеспечения. Программные инструменты для анализа, моделирования, проектирования и разработки информационных систем.
CASE-средство	Инструментальная программная система, обеспечивающая поддержку различных (одного или нескольких) этапов жизненного цикла программного обеспечения.
DFD (Data Flow Diagrams)	Диаграммы потоков данных.
ERD (Entity-Relationship Diagrams)	Диаграммы сущность связь.
ERwin	CASE-средство моделирования баз данных.
ICAM DEFinition	Система аналитических стандартов США.
IDEF0	Составная часть стандартов ICAM DEFinition. Стандарт предназначен для создания функциональной модели, отображающей структуру и функции системы.
IDEF1	Составная часть стандартов ICAM DEFinition. Стандарт предназначен для построения информационной модели, отображающей структуру и содержание информационных потоков, необходимых для поддержки функций системы.
IDEF1X	Составная часть стандартов ICAM DEFinition. Стандарт предназначен для разработки структуры реляционных баз данных. Создан на основе совершенствования стандарта IDEFI с учетом таких требований, как простота для изучения и возможность автоматизации. IDEFIX-диаграммы используются в ряде распространенных CASE-средств (в частности, ERwin, Design/IDEF).

IDEF3	Составная часть стандартов ICAM DEFinition. Стандарт предназначен для документирования технологических процессов, происходящих на предприятии, и предоставляет инструментарий для наглядного исследования и моделирования их сценариев.
IDEF5	Составная часть стандартов ICAM DEFinition. Стандарт обеспечивает наглядное представление данных, полученных в результате обработки онтологических знаний, в простой естественной графической форме.
Model Navigator	CASE-средство для просмотра моделей ERwin и BPwin с возможностью генерации отчетов.
ModelMart	Системой управления моделированием, обеспечивающая совместное (групповое) проектирование и разработку программных систем, включая механизмы объединения моделей и анализа изменений, контроль версий, возможность создания «компонент» модели и т.д.
Object Management Group – OMG	Международная некоммерческая организация, занимающаяся вопросами создания, развития и распространения стандартов, методов и средств построения информационных систем и технологий в рамках объектно-ориентированного подхода.
Object-oriented analysis – OOA	Объектно-ориентированный анализ.
Object-oriented design – OOD	Объектно-ориентированное проектирование.
Object-oriented programming – OOP	Объектно-ориентированное программирование.
OSTN (Object State Transition Network)	Диаграммами Состояния Объекта в Процессе его Трансформации.
Paradigm Plus	CASE-средство проектирования компонентов программного обеспечения и кодогенерации.
PFDD (Process Flow Description Diagrams)	Диаграммами Описания Последовательности Этапов Процесса.
Rational Rose	CASE-средство, разработанное фирмой Rational Software. Предназначено для автоматизации этапов анализа и проектирования ПО, а также для генерации кодов на различных языках и выпуска проектной документации. Rational Rose использует методологию объектно-ориентированного анализа и проектирования.
Rational Unified Process – RUP	Унифицированный процесс создания ПО, разработанный фирмой Rational Software.
Report Template Builder (RPTwin)	Составная, но автономная часть BPwin (общая для ERwin и BPwin), представляющая собой генератор отчетов, который позволяет генерировать подробные и многогранные отчеты по модели.

SADT	Технология структурного анализа и проектирования, стандартизованная в США.
SSADM	Методология структурного системного анализа и проектирования, стандартизованная в Великобритании.
STD (State Transition Diagrams)	Диаграммы переходов состояний.
Swim Lane	Механизм визуализации и оптимизации сложных бизнес-процессов, который позволяет на диаграммах координировать сложные процессы и функциональные ограничения.
TQM (Total Quality Management)	Японский аналог CPI; концепция глобального управления качеством (Широко распространена в Европе).
Unified Modelling Language – UML	Унифицированный язык объектного моделирования.
Абстрагирование	Способ выделения существенных характеристик некоторого объекта (абстракций), отличающих его от всех других видов объектов и, таким образом, четко определяющих его концептуальные границы.
Адаптация	Возрастание согласованности между свойствами системы (внутренней детерминантой) и определенной заданной функцией (функциональным запросом надсистемы = внешней детерминантой) в узле сети связей надсистемы.
Актор	Объект (класс), который может воздействовать на другие объекты, но сам никогда не подвергается их воздействию.
Алфавит формально-семантический	Алфавит формальной системы, знаки которого обладают предметным смыслом до построения из них каких-либо выражений.
Аналитический этап развития науки	Уходящий этап развития науки, характеризующийся индуктивностью (операционализмом), элементаризмом (редукционизмом), антителеологичностью, дифференциацией знаний.
Атрибут	Характеризует состояние объектов данного класса.
Базовая иерархия классов	Высокоуровневая концептуальная классификационная модель видов систем, рассматриваемых как функциональные объекты, а также видов их функциональных и структурных свойств.
Балансирование диаграмм	Проведение специального сквозного контроля диаграмм одного или разных типов.
Бизнес-модель	Модель бизнес-процесса, осуществляемого бизнес-системой.
Бизнес-процесс	Множество внутренних шагов (видов) деятельности, начинающихся с одного или более входов и заканчивающихся созданием продукции, необходимой клиенту.

Бизнес-система	см. Система организационная.
Браузер	Иерархическая структура, позволяющая осуществлять навигацию по модели.
Валентность	Свойство системы.
Валентность интенциальная	Непроявленное в связях свойство, но возможное (способность сильная).
Валентность потенциальная	Непроявленное в связях свойство, но возможное (способность слабая).
Валентность экстенциальная	Свойство, проявленное в связях.
Вариант использования (use cases) или прецедент	Описание последовательности выполняемых системой действий, в результате которых образуется наблюдаемый результат, значимый для какого-нибудь определенного актора.
Внешняя модель бизнес-системы	Прецедент-модель (П-модель). Она описывает бизнес и его окружение, предприятие в целом и его внешний мир (сегмент рынка), а также процессы, которые удовлетворяют интересы клиентов и интересы вне предприятия. Процессы моделируются при помощи прецедентов, а окружение моделируется при помощи так называемых действующих лиц или субъектов.
Внутренняя модель бизнес-системы	Объект-модель (О-модель). Она описывает, как строится каждый бизнес-процесс предприятия из различных рабочих задач (внутренних процессов) и какие ресурсы он использует.
Генетический способ задания формальной системы	Способ построения формальной системы путем задания некоторого алфавита и правил манипулирования им (синтаксиса), без использования исходных постулируемых утверждений (аксиом).
Гомеостатическое регулирование	Поддержание постоянства характеристик внутренней среды.
Декомпозиция объектная	Разделение системы на составляющие ее объекты.
Декомпозиция процедурная	Разделение общей функциональности системы на частные процедуры.
Детерминанта системы внешняя (функциональный запрос надсистемы)	Причина выбора внутренней детерминанты (функциональный запрос надсистемы). Потребность надсистемы в системе с определенной функцией.
Детерминанта системы внутренняя	Функционирование системы, определяющее ее внутренние характеристики.
Детерминанта системы внутренняя предельная	Детерминанта, характеризующая необходимую в конечном счете надсистеме функциональную способность данной системы.
Детерминанта системы внутренняя текущая	Детерминанта, характеризующая достигнутую к данному моменту текущую функциональную способность системы.
Детерминантный анализ	Анализ, направленный на выявление внутренней и внешней детерминант системы.

Диаграмма	Совокупность правил соединения различных символов языка моделирования. Диаграмма представляет собой связанный граф, вершинами которого являются сущности, а ребрами – отношения.
Диаграмма «сущность связь»	Средство моделирования данных и связей между ними на концептуальном уровне.
Диаграмма вариантов использования (прецедентов)	Диаграмма поведения, показывающая функции моделируемой системы и ее связи с другими системами, т.е., в случае разработки программного обеспечения, требования пользователей.
Диаграмма классификации	Инструмент онтологического анализа для логической систематизации знаний, накопленных при изучении системы.
Диаграмма переходов состояний	Средство моделирования и документирования аспектов систем, зависящих от времени или реакции на событие. Данные диаграммы позволяют осуществлять декомпозицию управляющих процессов и описывают отношения между входными и выходными управляющими потоками.
Диаграмма потоков данных	Средство моделирования свойств систем в виде иерархии процессов, связанных потоками данных.
Диаграмма состояния объекта	Инструмент онтологического анализа, позволяющий документировать тот или иной процесс с точки зрения изменения состояния объекта.
Жизненный цикл организационной системы	Состоит из следующих этапов: 1 – зона нечувствительности; 2 – внедрение; 3 – рост; 4 – зрелость; 5 – стабильность; 6 – спад; 7 – крах; 8 – старение; 9 – утилизация.
Закон системной декомпозиции	Элементы на i-ом ярусе системы должны находиться в отношении поддержания функциональной способности i+1-го яруса системы (системы должны поддерживать надсистему, подсистемы – систему и т.д.).
Запас	Совокупность объектов, имеющуюся в наличии в конкретный момент времени и измеряемую в абсолютных статических единицах.
Идеальная О-модель	Модель, не учитывающая, как бизнес должен реализоваться на практике.
Иерархия	Способ упорядочения абстракций (классов) или объектов по уровням или ярусам.
Иерархия наследования (Иерархия классов)	Способ упорядочения абстракций с помощью отношения обобщения (род – вид).
Иерархия объектов	Способ упорядочения объектов с помощью отношения агрегирования (целое – часть).
Измеримая потребительская ценность	Характеристика варианта использования (прецедента) бизнеса, позволяющая оценить его эффективность с точки зрения клиента.

Индивидуальный субъект	Конкретный класс субъектов или экземпляр класса субъект.
Инжиниринг бизнеса	Набор приемов и методов, которые организация использует для анализа и проектирования бизнеса в соответствии со своими целями.
Инжиниринг бизнеса обратный	Создание модели существующего предприятия, разработку его детального описания, идентификацию и документирование основных процессов, оценку их эффективности.
Инжиниринг бизнеса прямой	Разработка образа будущего предприятия и спецификации основных целей, исходя из потребностей клиентов и текущего состояния предприятия. Кроме того, создание более эффективных рабочих процедур, идентификация необходимых изменений в работе персонала, создание новой организационной структуры и системы мотивации, создание и внедрение поддерживающих информационных технологий и систем.
Инкапсуляция	Способ отделения элементов объекта (класса), определяющих его устройство, от элементов, определяющих его поведение.
Интерфейс	Описание поведения объекта (системы). Описание того, что делает система.
Информационная область предприятия	Совокупность информации об области деятельности предприятия; в первую очередь, содержащейся во всех видах документов, касающихся данного предприятия.
Информационный менеджмент	Использование информационных потоков, а также внесение в них каких-либо изменений.
Информационный поток	Движение информации; в первую очередь, документооборот.
Исходный материал	Части системы в исходном состоянии до их включения в состав целого и до адаптации в составе этого целого.
Каноническая форма представления системы	Форма представления системы, включающая в себя две ортогональных иерархии: иерархию классов и иерархию объектов.
Класс	Множество объектов (экземпляров), имеющих общую структуру и общее поведение.
Класс абстрактный	Класс, от которого экземпляры не производятся.
Класс анализа	Класс, выявленный на этапе анализа и входящий в словарь предметной области. Он не содержит специфические для технологии программирования детали.
Класс базовый	Самый общий класс в иерархии классов. Корневой класс.
Класс конкретный	Узкий, специализированный класс в иерархии, от которого создается экземпляры (объекты).

Класс прецедентов	Описание прецедента (варианта использования).
Класс проекта	Класс, в который преобразуется на этапе проектирования класс анализа или дополнительно вводимый в ходе проектирования класс, учитывающий специфические для технологии программирования детали.
Классификация генетическая	Классификация состояний системы. Предназначена для отслеживания фаз изменения функциональной способности системы от состояния исходного материала до состояния, соответствующего текущей внутренней детерминанте.
Классификация естественная	Таксономическая параметрическая классификация, в которой родо-видовым отношениям между любыми классифицируемыми элементами, соответствуют такие же отношения между учтенными в этой же классификации свойствами этих элементов.
Классификация мерономическая	Классификация частей системы. Предназначена для выявления сложившихся в результате адаптации к запросу надсистемы на данный момент времени внутренних (поддерживающих) свойства системы путем анализа частей (компонентов и элементов, то есть подсистем) данной системы.
Классификация параметрическая	Классификация, в которой элементы систематизированы на основании определенного свойства (параметра).
Классификация партитивная (мерономическая)	Классификация, получаемая путем выявления частей (меронов) на основании отношений часть-целое.
Классификация родовидовая (таксономическая)	Классификация, получаемая путем выявления видов (таксонов) на основании отношения род-вид.
Классификация стадиальная	См. классификация генетическая.
Клиент	Объект (система), использующий услуги другого объекта (системы).
Когнитивные структуры и процессы	Структуры и процессы существующие и происходящие в сознании человека.
Команда процесса	Группа людей, выполняющих совместно законченную часть работы – бизнес-процесс. Команды процессов заменяют старые структурные подразделения.
Композиционная схема	Инструмент онтологического анализа по принципу «Что из чего состоит». Средство графического представления состава объекта какого-либо класса.
Концептуальное классификационное моделирование	Концептуальное моделирование с помощью классификационных схем.
Концептуальное моделирование	Моделирование понятийных структур, описывающих предметную область или систему.

Критерий оптимальности В. Парето	Хорошо делать так, чтобы кому-нибудь стало лучше, если при этом никому другому не становится хуже.
Логистика	Теория и практика планирования, организации, управления и контроля процессов движения совокупности материальных, трудовых, финансовых и информационных потоков в системе рыночной экономики.
Логистика внутрипроизводственная	Теория и практика планирования, организации, управления и контроля процессов движения совокупности материальных, трудовых, финансовых и информационных потоков, осуществляемых в рамках производства и хранения продуктов производства.
Логистика заготовительная	Теория и практика планирования, организации, управления и контроля процессов движения совокупности материальных, трудовых, финансовых и информационных потоков, осуществляемых в рамках снабжения.
Логистика распределительная	Теория и практика планирования, организации, управления и контроля процессов движения совокупности материальных, трудовых, финансовых и информационных потоков, осуществляемых в рамках распределения и потребления.
Логистическая конфигурация	Конфигурация, любой выход каждого элемента которой или повторяет его вход, или является выходом такого типа, которого еще не было во всей этой конфигурации, начиная с входа первого элемента.
Логистическая цепь	Снабжение – производство – хранение – распределение – транспорт – спрос – потребление.
Логистический показатель	Показатель, характеризующий какую-либо сторону логистической цепи или логистического процесса.
Логистический процесс	Транспортно-заготовительные процессы и процессы хранение запасов.
Мера системности	Отношение области требуемых функциональных состояний к области возможных состояний системы.
Метод	Реализация операции класса, экземпляром которого является данный объект.
Механизмы расширения	Специальные средства UML, обеспечивающие расширение синтаксиса и семантики языка.
Миссия организации	Набор концептуальных положений, в обобщенной форме раскрывающих то, чему решила посвятить себя организация; своеобразная философская, социальная установка организации, ведущее направление ее деятельности.

Модель взаимодействия объектов	Модель, описывающая взаимодействие функциональных объектов (как экземпляров), а также материальные и информационные потоки их связывающие.
Модель онтологии	Концептуальная модель предметной области.
Модульность	Способ разложения системы на связанные, но относительно самостоятельные части (модули).
Ноосферный этап развития науки	Новый этап научного развития, характеризующийся дедуктивностью, эмерджентностью (нередукционизмом), целеполаганием (телеологичностью), интеграцией знаний.
Нормативная система	Формальная система, построенная не аксиоматическим, а генетическим способом.
Нормативная система формально-семантическая	Нормативная система, использующая формально-семантический алфавит.
Область возможных состояний	Функциональные состояния, определяемые внутренними поддерживающими свойствами системы.
Область требуемых функциональных состояний	Функциональные состояния, определяемые запросом надсистемы (вакантным узлом).
Объект	Конкретный опознаваемый предмет, единица или сущность (реальная или абстрактная), имеющая четко определенное функциональное назначение в данной предметной области. Экземпляр класса. Системный аспект, с точки зрения которого система представляет собой определенную материальную субстанцию, реализующую некоторую функцию.
Объект интерфейсный	Объект, выполняющий часть бизнес-процесса, состоящую во взаимодействии с окружением бизнеса.
Объект управляющий	Объект, выполняющий часть бизнес-процесса и не имеющий непосредственного контакта с окружением бизнеса.
Объект-модель (О-модель) предприятия	см. Внутренняя модель организационной системы.
Объектная декомпозиция системы	Процесс разбиения системы на части, соответствующие классам и объектам предметной области.
Объектная модель системы	Модель системы, представляющая ее в виде совокупности взаимодействующих объектов и классов, согласованно действующих для обеспечения требуемого поведения.
Объектное моделирование	Процесс моделирования, в результате которого создается объектная модель системы.
Объектно-ориентированная методология	Совокупность основополагающих принципов, основанных на абстрагировании, инкапсуляции, иерархичности, модульности и т.д.

Объектно-ориентированное программирование	Подход к программированию, в рамках которого программа представляется в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.
Объектно-ориентированное проектирование	Подход к проектированию, в рамках которого путем объектной декомпозиции создается объектная модель разрабатываемой системы.
Объектно-ориентированный анализ	Подход к анализу, при котором моделируемая и разрабатываемая системы анализируются с точки зрения классов и объектов, выявленных в предметной области.
Объект-сущность	Объект, который обрабатывается бизнесом.
Окрестностные условия	Совокупность систем, с которыми связана данная система.
Онтологический анализ	Разделение реального мира на составляющие его классы объектов и определение их онтологий, или же совокупности фундаментальных свойств, которые определяют их изменения и поведение.
Операциональные критерии естественной классификации	Критерии, позволяющие осуществлять построение классификационных схем, учитывающих закономерности естественной классификации.
Операция	Характеризует функционирование (поведение) объектов данного класса.
Организмический подход	Подход к анализу, моделированию и проектированию бизнеса, при котором бизнес-система рассматривается как живой организм с использованием биологической метафоры.
Ответственность	Предназначение и место класса (объекта) в системе; совокупность всех его услуг и всех контрактных обязательств (см. Роль).
Отношение	Символы, связывающие различные сущности на языке UML.
Отношение поддержания функциональной способности целого	Отношение между подсистемой и системой, при котором система приобретает свойства, поддерживаемые подсистемой.
Партитивная модель деловой активности бизнес-системы	Модели, представляющие систему переработки информации организацией в виде четырех подсистем, соответствующих уровням деловой активности, аналогичным уровням сенсомоторной активности человека.
Партитивная модель сенсомоторной активности человека	Модели, представляющие систему переработки информации человека в виде четырех подсистем, соответствующих уровням сенсомоторной активности.
Поведение	Динамические (функциональные) характеристики объекта.

Подход «Узел-Функция-Объект»	Подход, при котором любая система (любое явление действительности) рассматривается с точки зрения трех аспектов: как структурная часть еще более целого; как функциональный элемент и как некоторое материальное образование.
Потенциал	Способность организационной системы к деятельности.
Потенциал ликвидации	Потенциал, способствующий ликвидации организации.
Потенциал удержания	Потенциал, способствующий существованию и развитию организации.
Поток	Совокупность объектов, рассматриваемая как единое целое, существующая как процесс на некотором временном интервале и измеряемая в абсолютных единицах за определенный период.
Поток событий	Последовательность событий, необходимых для обеспечения требуемого поведения.
Правила системной декомпозиции	Правила взаимодействия компонент системы как функционального объекта: правило присоединения, правило баланса, правило реализации, правило замкнутости.
Представление вариантов использования	Представление модели системы в Rational Rose, показывающее варианты использования системы и ее окружение.
Представление компонентов	Представление модели системы в Rational Rose, показывающее информацию о библиотеках кода, исполняемых файлах, динамических библиотеках и других компонентах (физических модулях кода) модели.
Представление логическое	Представление модели системы в Rational Rose, показывающее как система будет реализовывать поведение, описанное в вариантах использования. Оно дает подробную картину составных частей системы и описывает взаимодействие этих частей.
Представление размещения	Представление модели системы в Rational Rose, показывающее физическое размещение объектов системы, которое может отличаться от ее логической архитектуры.
Прецедент-модель (П-модель) предприятия	см. Внешняя модель организационной системы.
Расширение	Отношение между вариантами использования (прецедентами). Применяется в случае, при котором один вариант использования подобен другому, но несет несколько другую нагрузку и, в связи с этим, необходимо описать изменение в обычном поведении системы.

Рациональный Унифицированный Процесс	Пошаговый процесс создания ПО, предполагающий постепенное проникновение в суть проблемы с помощью последовательных уточнений, в котором ПО разрабатывается и реализуется по частям, путем получения все более емкого решения.
Реализация	Описание устройства системы. Описание того, как система реализует то, что предполагает ее интерфейс.
Реальная О-модель	Модель, учитывающая, как бизнес может реализоваться на практике.
Реинжиниринг бизнеса	Революционная составляющая инжиниринга бизнеса. Обратный инжиниринг плюс прямой инжиниринг.
Репозитарий	В данном случае, объектно-ориентированная база данных.
Роль	Предназначение и место класса (объекта) в системе; совокупность всех его услуг и всех контрактных обязательств (см. Ответственность).
Свойство	Способность поддерживать (при определенных условиях) связи одних видов и препятствовать осуществлению связей других видов (валентность).
Свойство поддерживающее	Свойство, необходимое для поддержания и обеспечения устойчивости функциональных свойств, т.е. способность поддерживать связи (потоки), служащие средством внутреннего поддержания, стабилизации функциональных свойств (связей).
Свойство функциональное	Свойство, которым обязательно должна обладать система для выполнения своих функций, т.е. способность поддерживать связи (потоки), на основе которых протекают важные для надсистемы взаимодействия системы с окрестностными системами.
Связь	Проявление процесса обмена (т.е. потока) между системами элементами, представляющими собой субстанции определенных глубинных ярусов связанных систем. В объектно-ориентированном подходе: «Физическое или концептуальное соединение между объектами».
Связь поддерживающая	Внутренняя связь данной системы, т.е. функциональная связь ее подсистемы.
Связь функциональная	Внешняя связь данной системы с одной из окрестностных систем.
Сенсомоторная активность	Деятельность системы, осуществляемая за счет сенсорного распознавания сигналов и состояния внешней и внутренней среды, принятия в связи с этими сигналами и состояниями решений и управления, а также моторной (физической) активности в соответствии с принятыми решениями.

Сервер	Объект (система), предоставляющий услуги другому объекту (системе).
Сжатие процессов вертикальное	Реинжиниринг бизнес-процессов, при котором в тех точках процесса, где при традиционной организации работ исполнитель должен был обращаться к управленческой иерархии для принятия решений, он принимает решения самостоятельно.
Сжатие процессов горизонтальное	Реинжиниринг бизнес-процессов, при котором процедуры, выполнявшиеся различными сотрудниками, интегрируются в одну.
Синергия	Явление резкого усиления общего потенциала организации, когда этот потенциал становится больше суммы частных потенциалов.
Система (системологическое понимание)	Функциональный объект, функция которого обусловлена функцией объекта более высокого яруса, т.е. надсистемой.
Система (традиционное теоретико-множественное понимание)	Нечто целое в принципе структурируемое, т.е. состоящее из взаимосвязанных частей; имеющее вход и выход; существующее или даже функционирующее в некоторой среде. Выходы системы отождествляются с ее целью или определяются как воздействия системы на среду. Вход рассматриваться как поступление ресурсов из среды. Системе приписываться некоторый процесс перевода входа в выход.
Система логистическая	Система, в которой осуществляются планирование и реализация движения и развития совокупного ресурсного потенциала, организованного в виде логистического потока, начиная с отчуждения ресурсов у окружающей среды и до реализации конечной продукции.
Система логистическая толкающая	Внутрипроизводственная логистическая система подачи материалов, деталей или узлов в производственный процесс или с предыдущей технологической операции на последующую независимо от того, нужны ли они в данный момент и в данном количестве на этой операции.
Система логистическая тянущая	Внутрипроизводственная логистическая система подачи материалов, деталей или узлов с предыдущей технологической операции на последующую по мере необходимости.
Система оптимально адаптированная (совершенная)	Система, мера системности которой приближается к единице.
Система организационная	Социальная система, характеризующаяся наличием социально-экономических (хозяйственных) связей и отношений.

Система социальная	Система, характеризующаяся наличием человека в совокупности взаимосвязанных элементов.
Система-класс (внешняя система)	Класс объектов общей природы, объединенных некоторой целостной сущностью. Элементы такой системы могут не обладать ни пространственной, ни временной общностью, ни даже генетической связью, важна лишь общность природы образующих систему объектов.
Система-явление (внутренняя система)	Целостное образование, к которому можно применить процедуры членения, представляя эту систему в виде некоторой структуры составляющих частей.
Системные исследования	Научное направление, изучающее свойства сложных (слабоструктурированных и слабоформализуемых) объектов и процессов с помощью средств системного анализа и методов системного подхода. Теоретически оформлено в виде общей теории систем.
Системный анализ	Совокупность методов и средств, используемых при исследовании и конструировании сложных и сверхсложных объектов, прежде всего методов выработки, принятия и обоснования решений при проектировании, создании и управлении социальными, экономическими, человеко-машинными и техническими системами.
Системный подход	Направление методологии научного познания, в основе которого лежит рассмотрение объектов как систем; ориентирует исследователя на раскрытие целостности объекта, на выявление многообразных видов связей в нем и сведение их в единую теоретическую картину.
Системный эффект (эмерджентность)	Наличие (возникновение) у системы целостных свойств, представляющих собой принципиально новое качество, несводимое к свойствам составляющих систему частей.
Системологический классификационный анализ	Анализ, направленный на обеспечение учета свойств и закономерностей естественной классификации при построении концептуальных классификационных моделей.
Системологический когнитивный подход	Подход к изучению системы или предметной области, учитывающий их системные и когнитивные свойства и закономерности.
Системология (функциональная)	Системный подход, при котором система рассматривается как функциональный объект.
Словарь данных	Определенным образом организованный список всех элементов, составляющих потоки данных системы с их точными определениями.

Словарь предметной области	Совокупность классов и объектов предметной области, выявляемых (обнаруживаемых, открываемых) на этапе объектно-ориентированного анализа.
Состояние	Статические (субстанциальные и структурные) характеристики объекта.
Спецификация процесса	Описание алгоритма, соответствующего данному процессу и трансформирующего входные потоки данных в выходные.
Структура	Совокупность связей и их пересечений (узлов).
Субстанция	Наполнение структуры, т.е. то, что находится в узлах и передается по связям.
Субъект	Роль, которую кто-то или что-то может играть по отношению к бизнесу.
Сущностные свойства	Функциональные свойства, ради наличия и для поддержания которых, сформировалась данная система.
Сущность	Символы, являющиеся основными элементами объектной модели, представленной на языке UML.
Схема взаимосвязей	Инструмент онтологического анализа, позволяющий визуализировать и изучать взаимосвязи между различными классами объектов в системе.
Сценарий	Описание последовательности изменений свойств объекта.
Теория организации	Составная часть науки (теории) управления, изучающая принципы, законы и закономерности создания, функционирования, реорганизации и ликвидации организаций.
Транзакция	Неделимое множество действий, которые или выполняются все целиком, или не выполняются вообще.
Узел	Элемент структуры, в котором пересекаются связи (экстенциальные валентности) окрестностных систем. Системный аспект, с точки зрения которого система представляет собой перекресток определенных входящих и выходящих связей/потоков.
Узел вакантный	Элемент структуры, в котором пересекаются интенциальные валентности окрестностных систем.
Уровень самосохранения	Отношение разности потенциалов удержания и ликвидации к потенциалу удержания.
Усовершенствование бизнеса	Эволюционная составляющая инжиниринга бизнеса.
УФО-анализ	Анализ, представляющий собой реализацию объектно-ориентированной методологии системологического анализа и проектирования (OMSAD), основанную на подходе «Узел – Функция – Объект».
УФО-библиотека	Библиотека (репозитарий; фасетная классификация), в которой хранятся УФО-элементы, соответствующие определенной предметной области.

УФО-иерархия	Концептуальная классификационная модель УФО-элементов предметной области или системы.
УФО-конфигурация	Совокупность (сборка) взаимосвязанных по правилам системной декомпозиции УФО-элементов (см. Модель взаимодействия объектов).
УФО-модель	Модель, представляющая систему или предметную область в виде УФО-конфигурации.
УФО-подход	Подход «Узел-Функция-Объект».
УФО-элемент	Система, которой соответствует определенный Узел (пересечение связей/потоков) в структуре надсистемы, определенная Функция (в общем случае не единственная), балансирующая потоки данного узла, и определенный Объект (в общем случае для каждой функции не единственный), реализующий данную функцию.
Функция	Системный аспект, с точки зрения которого система представляет собой определенную функцию преобразования входных ресурсов конкретного узла в выходные (обеспечения баланса «притока» и «оттока»).
Эволюция	Возрастание согласованности между свойствами системы (внутренней детерминантой) и определенной заданной функцией (функциональным запросом надсистемы = внешней детерминантой) в узле сети связей надсистемы с учетом изменения функционального запроса (внешней детерминанты).
Экземпляр прецедента	Действия, которые выполняются, когда поток событий, соответствующий описанию прецедента, проходит через систему.
Язык объектного моделирования	Язык для визуализации, специфицирования, конструирования и документирования различных аспектов анализируемых и проектируемых систем произвольной природы в рамках объектно-ориентированной парадигмы.

Список литературы

1. *Гвишиани, Д. М.* Материалистическая диалектика – философия основы системных исследований [Текст] / Д.М. Гвишиани // Системные исследования: Ежегодник, 1979. – М.: Наука, 1980. - С. 7-28.
2. *Бреховских, С. М.* Основы функциональной системологии материальных объектов [Текст] / С.М. Бреховских. – М.: Наука, 1986. - 192 с.
3. *Разработка* подсистемы лингвистического обеспечения АИС документально-факто-графического типа в области материаловедения [Текст]: отчёт о НИР (заключ.) / СКТБ ИС ИПМ АН УССР; рук. Г.П. Мельников. – М., 1986. - № рег. 01.86.0103243.
4. *Мельников, Г. П.* Методология лингвистики [Текст] / Г.П. Мельников, С.Ю. Преображенский. – М.: Изд-во Ун-та дружбы народов, 1989. - 83 с.
5. *Мельников, Г. П.* Системология и языковые аспекты кибернетики [Текст] / Г.П. Мельников. – М.: Сов. радио, 1978. - 368 с.
6. *Шрейдер, Ю. А.* Системы и модели [Текст] / Ю. А. Шрейдер, А. А. Шаров. – М.: Радио и связь, 1982. - 152 с.
7. *Перегудов, Ф. И.* Введение в системный анализ [Текст] / Ф. И. Перегудов, Ф. П. Тарасенко. – М.: Высш. шк., 1989. - 367 с.
8. *Месарович, М.* Общая теория систем: математические основы [Текст] / М. Месарович, Д. Михайло, Я. Такахара. – М.: Мир, 1978. - 311 с.
9. *Клир, Д.* Системология: автоматизация решения системных задач [Текст] / Д. Клир. – М.: Радио и связь, 1990. - 539 с.
10. *Косарев, Ю. Г.* Вступительная статья [Текст] / Ю.Г. Косарев // Системология и языковые аспекты кибернетики / Г.П. Мельников. – М.: Сов. радио, 1978.
11. *Полищук, Д. М.* Теория автоматизированных банков информации [Текст] / Д. М. Полищук, В. Б. Хон. – М., 1989. - 184 с.
12. *Шрейдер, Ю. А.* Теория познания и феномен науки [Текст] / Ю.А. Шрейдер // Гносеология в системе философского мировоззрения. – М.: Наука, 1983. - С. 173-193.
13. *Буч, Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на С++ [Текст] / Г. Буч. - 2-е изд. - М.: «Издательство Бином», СПб.: «Невский диалект», 1998. – 560 с.
14. *Никаноров, С.П.* Системный анализ: этап развития методологии решения проблем в США [Текст] / С.П. Никаноров // Системный анализ для решения деловых и промышленных проблем / С.Л. Оптнер. - М.: Советское радио, 1969. - С.7-45.
15. *Никаноров, С.П.* Совершенствование, создание и развитие организаций на основе теории систем [Текст] / С.П. Никаноров // Кибернетику на службу коммунизму. - М.: Наука, 1977. - Т.8. - С. 45-52.
16. *IDEF* [Электронный ресурс] / Режим доступа: http:// www.idef.com
17. *IS.KIEV* [Электронный ресурс] / Режим доступа: http:// www.is.kiev.ua
18. *CITFORUM* [Электронный ресурс] / Режим доступа: http:// www.citforum.ru
19. *Лямец, В.И.* Системный анализ [Текст] / В.И. Лямец, А.Д. Тевяшев. - Харьков: ХТУРЭ, 1998. – 252 с.
20. *Калянов, Г.Н.* Консалтинг при автоматизации предприятий [Текст] / Г.Н. Калянов. - М.: СИНТЕГ, 1997. – 316 с.
21. *Сапегин, А.* Реорганизация бизнес-процессов: как выглядит наше будущее? [Электронный ресурс] / А. Сапегин. - Режим доступа: http://www.interfase.ru
22. *Вендров, А.М.* Проектирование программного обеспечения экономических информационных систем [Текст] / А.М. Вендров. - М.: Финансы и статистика, 2000. – 352 с.
23. *Колесников, С.* Методики моделирования в бизнесе [Электронный ресурс] / С. Колесников. - Режим доступа: http:// www.consulting.ru/main/

mgmt/texts/m3/032_method1.htm.
24. *Машков, Д.А.* Моделирование бизнес-процессов при проектировании КИС [Электронный ресурс] / Д.А. Машков. - Режим доступа: http:// www.soft.implozia.ru/news/seminar02_04.html.
25. *Рубцов, С.* Какой CASE-инструментарий нанесет наименьший вред организации? [Электронный ресурс] / С. Рубцов // Директору информационной службы: электронный журн. - 2002. - №1. - Режим доступа: http:// www.osp.ru/cio/2002/008.htm.
26. *Маторин, С.И.* О новом научном направлении системологического анализа сложных динамических объектов [Текст] / С.И. Маторин // Вестник ХГПУ. Новые решения в современных технологиях. - 2000. - №81. - С. 15-18.
27. *Маторин, С.И.* Системология и объектно-ориентированный подход (проблемы формализации и перспективы стыковки) [Текст] / С.И. Маторин // Научно-техническая информация. Сер. 2. - 2001. - № 8. - С. 1-8.
28. *Кисель, Е.Б.* Опыт использования средств искусственного интеллекта в моделировании бизнес-процессов [Текст] / Е.Б. Кисель, Е.Н. Кондрашева, М.Б. Шинкарев // Новости искусственного интеллекта. – 1996. - №4. - С. 85-120.
29. Системный анализ в экономике и организации производства [Текст]: учеб. пособие вузов / под ред. С.А. Валуева, В.Н. Волковой. - Л.: Политехника. - 1991. – 398 с.
30. *Калашников, В.В.* Сложные системы и методы их анализа [Текст] / В.В. Калашников. - М.: Знание, 1980. – 64 с.
31. *Бусленко, Н.П.* Моделирование сложных систем [Текст] / Н.П. Бусленко. - М.: Наука, 1978. – 399 с.
32. *Йордан, Э.* Структурные модели в объектно-ориентированном анализе и проектировании [Текст] / Э. Йордан, К. Аргила. - М.: «ЛОРИ», 1999. – 264 с.
33. *Петров, Э.Г.* Методология структурного системного анализа и проектирования крупномасштабных ИУС [Текст] / Э.Г. Петров, С.И. Чайников, А.О. Овезгельдыев. - Харьков: «Рубикон», 1997. – 140 с.
34. *Чеботарев, В.* Моделирование бизнеса: средства и методы [Электронный ресурс] / В. Чеботарев. - Режим доступа: http://www.pcweek.ru/year2000/N9/CP1251/CorporationSystems/chapt1.htm.
35. *Лысенко, М.А.* Методика анализа и проектирования при построении корпоративных информационных систем. Часть 1 [Электронный ресурс] / М.А. Лысенко, М.Г. Осипов. - Режим доступа: http:// www.interface.ru/misc/metan_01.htm.
36. *Лысенко, М.А.* Методика анализа и проектирования при построении корпоративных информационных систем. Часть 2 [Электронный ресурс] / М.А. Лысенко, М.Г. Осипов. - Режим доступа: http:// www.interface.ru/misc/metan_02.htm.
37. *Сахаров, П.* Rational Rose, BPwin и другие – аспект анализа бизнес-процессов [Электронный ресурс] / П. Сахаров // Директору информационной службы: электронный журн. - 2000. - №11.- Режим доступа: http: // www.osp.ru/cw/cio/2000/011_0.htm.
38. *Гаврилова, Т.А.* Базы знаний интеллектуальных систем [Текст] / Т.А Гаврилова, В.Ф. Хорошевский. - СПб: Питер, 2000. – 384 с.
39. *Юдицкий, С.А.* Сценарный подход к моделированию поведения бизнес-систем [Текст] / С.А. Юдицкий. - М.: СИНТЕГ, 2001. – 112 с.
40. *Калянов, Г.Н.* Теория и практика реорганизации бизнес-процессов [Текст] / Г.Н. Калянов. - М.: СИНТЕГ, 2000. – 212 с.
41. *Теория* систем и методы системного анализа в управлении и связи [Текст]: учеб. пособие вузов / под ред. В.Н. Волковой, В.А. Воронкова, А.А. Денисова и др. - М.: Радио и связь, 1983. – 248 с.
42. *Маторин, С.И.* О новом методе системологического анализа, согласованном с процедурой объектно-ориентированного проектирования. Ч.1 [Текст] / С.И. Маторин // Кибернетика и системный анализ. - 2001. - №4. - С. 119-132.

43. *Беляев, И.П.* Системный анализ для разработки и внедрения информационных технологий [Текст] / И.П. Беляев, В.М. Капустян.– М.: МГСУ, 2007.
44. *Волкова, В.Н.* Основы теории систем и системного анализа [Текст] / В.Н. Волкова, А.А. Денисов. - СПб.: Изд-во Политехнического университета, 2005.
45. *Бондаренко, М.Ф.* Основы системологии [Текст]: учеб. пособие / М.Ф. Бондаренко, Е.А. Соловьева, С.И. Маторин. - Харьков: ХТУРЭ, 1998. - 118 с.
46. *Верников, Г.* Основные методологии обследования организации. Стандарт IDEF0 [Электронный ресурс] / Г. Верников. - Режим доступа: http:// www.consulting.ru/main/mgmt/texts/m7/079_idef.shtml.
47. *Верников, Г.* Основные методологии обследования организации. Стандарт IDEF0 [Электронный ресурс] / Г. Верников. - Режим доступа: http:// www.consulting.ru/main/mgmt/texts/m7/080_idef.shtml.
48. *Верников, Г.* Основы методологии IDEF1, IDEF1X, IDEF3, IDEF5 [Электронный ресурс] / Г. Верников. - Режим доступа: http:// www.citforum.ru/cfin/vernikov.
49. *Гиг, Дж., ван* Прикладная общая теория систем [Текст] / Дж. Гиг, ванн. - М.: Мир, 1981. - Т.1. – 336 с.
50. *Колесников, С.* Что такое КИС и как с ней бороться … [Электронный ресурс] / С. Колесников. - Режим доступа: http:// www.consulting.ru/main/soft/texts/m4/043_s_red0-2.htm.
51. *Суслов, И.П.* Основы теории достоверности статистических показателей [Текст] / И.П. Суслов. - Новосибирск: Наука, Сибирское отделение, 1979. - 304 с.
52. *Цвиркун, А.Д.* Имитационное моделирование в задачах синтеза структуры сложных систем [Текст] / А.Д. Цвиркун, В.К Акинфиев., В.А. Филиппов. - М.: Наука, 1985. – 174 с.
53. *Глушков, В.М.* Введение в АСУ [Текст] / В.М. Глушков. - К.: Техніка, 1972. – 312 с.
54. *Казачков, Л.С.* Прикладная логика информатики [Текст] / Л.С. Казачков. - К.: Наук. думка, 1990. – 256 с.
55. *Винер, Н.* Творец и робот [Текст] / Н. Винер. - М.: Прогресс, 1960. – 104 с.
56. *Гернек, Ф.* Альберт Эйнштейн [Текст] / Ф. Гернек. - М.: Прогресс, 1966. – 107 с.
57. *Агошкова, Е.Б.* Системология: сущность и место в научном знании [Текст] / Е.Б. Агошкова, Б.В. Ахлибинский, Б.С. Флейшман // Синергетика и методы науки. - СПб.: Наука, 1998. - С. 63-76.
58. *Уемов, А.И.* Системный подход и общая теория систем [Текст] / А.И. Уемов. - М.: Мысль, 1978. – 272 с.
59. *Оптнер, Станфорд, Л.* Системный анализ для решения деловых и промышленных проблем [Текст] / Станфорд Л. Оптнер.- М.: Советское радио, 1969. – 216 с.
60. *Тюхтин, В.С.* Отражение, системы, кибернетика [Текст] / В.С. Тюхтин. - М.: Наука, 1972. – 256 с.
61. *Садовский, В.Н.* Основания общей теории систем. Логико-методологический анализ [Текст] / В.Н. Садовский. - М., 1974. – 106 с.
62. *Смирнов, Э.А.* Основы теории организации [Текст] / Э.А. Смирнов. - М.: «Аудит», 1998. – 375 с.
63. *Зубенко, Ю.Д.* Менеджмент: на базе системного анализа [Текст] / Ю.Д. Зубенко, А.К. Носач; под ред. А.Д. Шарапова. - Донецк: ДонГТУ, 1998. – 415 с.
64. *Левин, В.И.* Структурно-логические методы исследования сложных систем с применением ЭВМ. (Теория и методы системного анализа) [Текст] / В.И. Левин. - М.: Наука, 1987. – 304 с.
65. *Жиляков, Е.Г.* Модели и методы системного анализа в экономике: учебно-практическое пособие [Текст] / Е.Г. Жиляков, Н.В. Щербинина. - Белгород: Изд-во БелГУ, 2006.
66. *Хомяков, П.М.* Системный анализ: Краткий курс лекций [Текст] / П.М. Хомяков. - М.: КомКнига, 2006.

67. *Бурков, В.Н.* Механизмы функционирования организационных систем [Текст] / В.Н. Бурков, В.В. Кондратьев. - М.: Наука, 1981. – 384 с.
68. *Бурков, В.Н.* Теория активных систем: состояние и перспективы [Текст] / В.Н. Бурков, Д.А. Новиков. - М.: СИНТЕГ, 1999. – 128 с.
69. *Новиков, Д.А.* Курс теории активных систем [Текст] / Д.А. Новиков, С.Н. Петраков. - М.: СИНТЕГ, 1999. – 108 с.
70. *Современный* философский словарь / под ред. В.Е. Кемерова. - М.: «Одиссей», 1996. – 608 с.
71. *Игнатьев, М.Б.* Моделирование сложных систем [Текст] / М.Б. Игнатьев // Синергетика и методы науки. - СПб.: Наука, 1998. - С. 425-431.
72. *Заец, Р.В.* Содержание системного анализа проблем городского функционирования и развития [Текст] / Р.В. Заец // Проблемы информатики города. К.: Наук. Думка, 1990. - С. 91-110.
73. *Моисеев, Н. Н.* Универсальный эволюционизм и коэволюция [Текст] / Н.Н. Моисеев // Природа. - 1989. - №4. - С. 3-8.
74. *Бондаренко, М.Ф.* Моделирование и проектирование бизнес-систем: методы, стандарты, технологии [Текст]: учеб. пособие вузов / М.Ф. Бондаренко, С.И. Маторин, Е.А. Соловьева; предисл. Э.В. Попова. - Харьков: «Компания СМИТ», 2004. – 272 с.
75. *Кондратьев, В. В.* Показываем бизнес-процессы [Текст] / В.В. Кондратьев, М.Н. Кузнецов. – М.: Эксмо, 2008. – 256 с.
76. *Вилкас, Є.Й.* Решения: теория, информация, моделирование [Текст] / Є.Й. Вилкас, Е.З. Майминас. - М.: «Радио и связь», 1981. – 328 с.
77. *Маклаков, С.В.* BPwin и ERwin. CASE – средства разработки информационных систем [Текст] / С.В. Маклаков. - М.: ДИАЛОГ–МИФИ, 2000. – 256 с.
78. *Буч, Г.* Объектно-ориентированный анализ и проектирование [Текст] / Г. Буч. - М.: «Бином», 1998. – 356 с.
79. *Буч, Г.* Язык UML. Руководство пользователя [Текст] / Г. Буч, Д. Рамбо, А. Джекобсон. - М.: ДМК, 2000. – 432 с.
80. *Гайсарян С.С.* Объектно-ориентированные технологии проектирования прикладных программных систем [Электронный ресурс] / С.С. Гайсарян. - Режим доступа: http:// www.citforum.ru.
81. *Фаулер, М.* UML в кратком изложении. Применение стандартного языка объектного моделирования [Текст] / М. Фаулер, К. Скот. - М.: Мир, 1999. – 191 с.
82. *Вендров, А.М.* Объектно-ориентированный анализ и проектирование информационных систем с помощью Rational Rose [Текст] / А.М. Вендров. - М.: «Академия АйТи», 2000. – 210 с.
83. *Кватрани, Т.* Rational Rose 2000 и UML. Визуальное моделирование [Текст] / Т. Кватрани. - М.: ДМК Пресс, 2001. – 176 с.
84. *Ларман, Крег* Применение UML и шаблонов проектирования [Текст] / Крег Ларман. - М.: Издательский дом «Вильямс», 2001. – 496 с.
85. *Леоненков, А.* Самоучитель UML [Текст] / А. Леоненков. - СПб.: «БХВ-Петербург», 2001. – 304 с.
86. *Трофимов, С.А.* CASE-технологии: практическая работа в Rational Rose [Текст] / С.А. Трофимов. - М.: ЗАО «Издательство БИНОМ», 2001. – 272 с.
87. *Новоженов, Ю.В.* Объектно-ориентированный подход к разработке прикладных программных систем [Текст] / Ю.В. Новоженов // PC magazine.- 1995.- №12.- С. 13-18.
88. *Шлеер, С.* Объектно-ориентированный анализ: моделирование мира в состояниях [Текст] / С. Шлеер, С. Меллор. - К.: Диалектика, 1993. – 240 с.
89. *Ross, R.* Entity Modeling: Techniques and Application [Текст] / R. Ross. - Boston, MA: Database Research Group, 1987.

90. *Coad, P.* Object-Oriented Analysis [Текст] / P. Coad, E. Yourdon. - New Jersey: Prentice-Hall, 1990.
91. *Кондаков, Н.И.* Логический словарь-справочник [Текст] / Н.И. Кондаков. - М.: Наука, 1975. – 720 с.
92. *Крачтен, Ф.* Введение в Rational Unified Process [Текст] / Ф. Крачтен. -М.: Издательский дом «Вильямс», 2002. – 240 с.
93. *Соловьева, Е.А.* Естественная классификация: системологические основания [Текст] / Е.А. Соловьева. - Харьков: ХТУРЭ, 1999. – 222 с.
94. *Бондаренко, М.Ф.* Анализ системологического инструментария концептуального моделирования проблемных областей [Текст] / М.Ф. Бондаренко, С.И. Маторин, Е.А. Соловьева // Научно-техническая информация. Сер. 2. - 1996. - №4. - С. 1-11.
95. *Маторин, С.И.* Анализ и моделирование бизнес-систем: системологическая объектно-ориентированная технология [Текст] / С.И. Маторин; предисл. Э.В. Попова. – Харьков: ХНУРЭ, 2002. – 322 с.
96. *Маторин, С.И.* Миссия возможна [Электронный ресурс] / С.И. Маторин. - Режим доступа: http://management.com.ua/stat/count_pdf.php3? art_id=str099.pdf
97. *Ойхман, Е.Г.* Реинжиниринг бизнеса [Текст] / Е.Г. Ойхман, Э.В. Попов. - М.: Финансы и статистика, 1997. – 336 с.
98. *Технология* моделирования бизнес-процессов [Электронный ресурс] / НИЦ CALS-технологий «Прикладная логистика». - Режим доступа: http:// http://www.cals.ru.
99. *Маклаков, С.В.* Моделирование бизнес-процессов с BPwin 4.0 [Текст] / С.В. Маклаков. - М.: ДИАЛОГ–МИФИ, 2002. – 224 с.
100. *Кинжалин, А.* BPwin – инструмент системного анализа [Электронный ресурс] / А. Кинжалин. - Режим доступа: http:// www.astrosoft.spb.ru.
101. *Маклаков, С.В.* Моделирование бизнес-процессов с AllFusion Process Modeler (BPwin 4.1) [Текст] / С.В. Маклаков. - М.: ДИАЛОГ–МИФИ, 2004. – 240 с.
102. *Притыкин, Д.А.* BPwin 4.0: пришел, увидел, реорганизовал [Электронный ресурс] / Д.А. Притыкин. - Режим доступа: http:// www.info-system.ru/designing/methodology/bpwin/bpwin.html.
103. *Бондаренко, М.Ф.* Системная технология моделирования информационных и организационных систем [Текст]: учеб. пособие / М.Ф. Бондаренко, С.И. Маторин, Д.Б. Ельчанинов. - Харьков: ХНУРЭ, 2005. – 116 с.
104. *Бондаренко, М.Ф.* Объектная технология моделирования информационных и организационных систем [Текст]: учеб. пособие / М.Ф. Бондаренко, С.И. Маторин, Е.А. Соловьева, Д.Б. Ельчанинов. - Харьков: ХНУРЭ, 2005. – 160 с.
105. *Мацяшек, Л.А.* Анализ требований и проектирование систем: разработка информационных систем с использованием UML [Текст] / Л.А. Мацяшек. - М.: Издательский дом «Вильямс», 2002. – 432 с.
106. *Франс, Р.* Разработка на базе моделей с использованием UML 2.0: обещания и просчеты [Текст] / Р. Франс, С. Гош, Т. Дин-Тронг, Э. Соулберг // Открытые системы. – 2006 - №3. - С. 34-43.
107. *Каныгин, Ю.М.* Основы теоретической информатики [Текст] / Ю.М. Каныгин, Г.И. Калитич. - К.: Наук. думка, 1990. – 232 с.
108. *Маторин, С.И.* Моделирование организационных систем в свете нового подхода «Узел-Функция-Объект» [Текст] / С.И. Маторин, А.С. Попов, В.С. Маторин // Научно-техническая информация. Сер.2. – 2005. – N1. – С. 1-8.
109. *Маторин, С.И.* «UFO-toolkit» – BI-инструментарий нового поколения [Электронный ресурс] / С.И. Маторин, А.С. Попов. - Режим доступа: http:// www.citforum.ru/consulting/BI/UFO/
110. *Аншина М.* Биография бизнес-объекта [Электронный ресурс] / М. Аншина. - Режим доступа: http://www.tops.ru\publishing\pub_021.html

111. *Маторин, С.И.* Визуальные графоаналитические модели для представления о сервисном обслуживании телерадиосети [Текст] / С.И. Маторин, О.А. Зимовец, С.Н. Трубицин // Искусственный интеллект и принятие решений. - 2008. – №3. – С. 52-63.
112. *Артемьев В.* Что такое Business Intelligence? [Электронный ресурс] / В. Артемьев // Открытые системы: электронный журн. – 2003. - №4. - Режим доступа: http://www.osp.ru/os/2003/04/020.htm
113. *Калянов Г.* CASE: все только начинается... [Электронный ресурс] / В. Артемьев // Директор ИС: электронный журн. – 2001. - №3. - Режим доступа: http://www.osp.ru/cio/2001/03/016.htm
114. *Выдержки* из перевода спецификации к нотации BPMN компании DIRECTUM [Электронный ресурс]. - Режим доступа: http://www.DIRECTUM-Journal.ru/docs/1624827.html
115. *Михеев А.* Перспективы WorkFiow систем. Сравнение workfiow языков [Электронный ресурс] / А. Михеев., М. Орлов // PC Week/RE. – 2005. - №36. - Режим доступа: http://wf.runa.ru/rus/images/c/c1/Stat_ya4.pdf
116. *Корпорация*: языки управления бизнес-процессами. BPML [Электронный ресурс]. - Режим доступа: http://citforum.ru/internet/xml/bpml/
117. *Михеев А.* Война стандартов в мире workfiow [Электронный ресурс] / А. Михеев., М. Орлов // PC Week/RE. – 2004. - №28. - Режим доступа: http://wf.runa.ru/rus/images/c/ce/Stat_ya2.pdf
118. *Куликов, Г.Г.* Лабораторный практикум по дисциплине “Автоматизированные информационные системы в производстве” [Электронный ресурс] / Уфимск. гос. авиац. техн. ун-т; Сост: Г.Г. Куликов, А.Г. Михеев, М. В. Орлов, Р.К. Габбасов, Д.В. Антонов. - Режим доступа: http://wf.runa.ru/rus/doc