

Введение

«Мир и все в нем, включая наши мысли о нем, можно рассматривать как систему взаимодействующих систем...». Эти слова принадлежат математику Дугласу Т. Росу, автору методологии структурного анализа и проектирования *SADT* (*Structured Analysis and Design Technique*), положившему начало истории функционального моделирования, сформировав в 1950-х гг. понятие функционального блока – графического изображения функции, оснащенного тремя входящими и одной исходящей связью.

В процессе создания промышленных крупномасштабных систем специалистами было замечено, что большой процент ошибок возникает на стадии анализа и проектирования систем, а их исправление на более поздних стадиях проекта: установки и эксплуатации, очень затратное. Описание функционирования системы, предопределяющее результаты ее работы, позволило бы уменьшить количество дорогостоящих ошибок, улучшить контакты между пользователями и разработчиками и сгладить переход от анализа к проектированию. Термин «описание» подразумевает в данном случае термин «создание модели». Идея того, что функционирование любой системы может быть описано иерархической структурой массива функций и легла в основу функционального моделирования (ФМ).

Существует два основных типа моделирования – динамическое и статическое. Динамическое рассматривает изменение моделируемых процессов во времени, которым нельзя пренебречь с точки зрения решаемых задач. Статическое предполагает создание моделей, в которых можно пренебречь временем. Функциональная модель описывает систему в статическом состоянии, помогает ответить на вопрос «что надо делать для достижения поставленной цели?». Ответом является перечень всех действий, которые необходимо выполнить, чтобы добиться запланированного результата.

Известно, что сложные вещи легче понимать, если они тем или иным образом зрительно представлены. Функциональное моделирование использует графические и естественные языки. Функциональная модель системы представляет собой совокупность структурных схем и носит качественный, описательный, декларативный характер. Она принципиально не может ответить на вопросы о том, как протекают процессы в системе во времени и в пространстве, каковы их характеристики, и в какой мере удовлетворяются (или не удовлетворяются) требования, предъявляемые к системе. На эти вопросы отвечают другие модели – математические, имитационные, описывающие

разворачивающиеся во времени физические, экономические, организационные, финансовые, логические и т.п. отношения между сущностями, входящими в функциональную модель.

Дуглас Т. Рос дал такое определение: «*М моделирует А, если М отвечает на вопросы относительно А с точностью В*». Но каждая модель ограничена в своих ответах, поэтому ФМ поддерживает достаточно большой перечень методологий, позволяющих анализировать и проектировать и информационные, и материально-информационные системы (в том числе и автоматизированные материальные системы) различных предметных областей. Назначение методологий в том, чтобы регламентировать основы разработки сложных систем, помогать справляться с проблемами размерности, обеспечивать организационную поддержку, позволяющую большим коллективам разработчиков действовать наиболее согласованно.

Одним из первых примеров осознания эффективности ФМ, применяемого с целью поддержки создания сложных систем, можно считать программу интегрированной компьютеризации производства *ICAM (Integrated Computer Aided Manufacturing)* в США в 1970-х гг. Ее выполнение потребовало новых методов работы, что и привело к успешному применению методологии функционального моделирования *SADT*. Поэтому и в СССР аналогичная научно-техническая программа «Технологии, машины и производства будущего» была принята уже в 1988 г. Тенденция поддержки производства вычислительной техникой получила свое развитие созданием технологий *CALS (Continuous Acquisition and Lifecycle Support)* позволяющих на всех стадиях жизненного цикла сложных изделий организовать обмен инженерной информацией в электронном виде от этапа научной деятельности по их созданию, до этапа утилизации вышедших из эксплуатации экземпляров. В связи с освоением этих технологий предприятиями Российской Федерации Госстандарт России разработал и ввёл в действие регламент на функциональное моделирование «Р50.1.028 - 2001. Рекомендации по стандартизации. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования».

В некоторых источниках авторы используют термин «структурный системный анализ» подразумевая ФМ. Связано это с тем, что ФМ является инструментом системного анализа. Действительно, скажем, грамотный системный анализ, проведенный на этапе формулировки требований к разрабатываемой системе, окажет существенное влияние и на все последующие этапы. Системному анализу необходимо, во-первых, понять, что предполагается сделать,

структурируя и обобщая данные о предметной области, во-вторых, задокументировать это, т.к. если требования не зафиксированы и не сделаны доступными для участников проекта, то они вроде бы и не существуют. При этом язык, на котором формулируются требования, должен быть понятен и заказчику, и исполнителю. С этими задачами помогает справиться ФМ.

Вообще, в какой бы предметной области не использовалось бы ФМ, оно способствует упорядочению и накоплению знаний, тем самым оптимизируя деятельность, связанную с анализом и проектированием сложных систем.

Учебное пособие состоит из шести разделов и приложений. В первых трех разделах рассмотрены теоретические основы методологий:

- структурного моделирования материально-информационных систем *IDEF0*;
- моделирования информационных систем *DFD*;
- потокового моделирования *IDEF3*.

Четвертый раздел посвящен правилам создания гибридных моделей. В качестве примера рассмотрено моделирование процесса функционирования *TLS*-сервера с совместным использованием методологий *IDEF0*, *DFD*, *IDEF3*.

Пятый раздел «Функциональное моделирование и промышленное предприятие» дает представление о функционировании промышленного предприятия, раскрывает содержание таких понятий как «бизнес-процесс», «бизнес-моделирование», «бизнес-консалтинг».

В шестом разделе приведен обзор инструментальных средств ФМ.

Приведенные в примерах модели были выполнены с помощью инструментальной среды *CA Erwin Process Modeler*. Технология работы с данным программным продуктом не рассматривается.

В пособии были использованы материалы публикаций известных специалистов в области ФМ: Г. Н. Калянова, В. И. Дубейковского, С. В. Маклакова, С. В. Черемных, Г. Г. Верникова [2, 4, 5, 7, 13, 14].

1. Методология *IDEF0*

История создания методологии функционального моделирования *IDEF0* (*Integration Definition for Function Modeling*) начинается с создания методологии структурного анализа и проектирования *SADT* (*Structured Analysis and Design Technique*) американским ученым Дугласом Т. Россом. Успешное применение *SADT* в военной и гражданской практике привело к тому, что часть этой методологии была положена в основу федерального стандарта США под названием *IDEF0*. Но в некоторых работах и сегодня *IDEF0* продолжают именовать как *SADT*.

Обозначение «SA-блок» (*Structured Analysis*) и понятие «иерархическая декомпозиция сверху вниз» Дуглас Т. Росс начал использовать еще с конца 50-х гг. в работах предшествовавших большинству технических средств и методов в области разработки программного обеспечения и в области систем автоматизированного проектирования. Но осмысление этих базовых для *SADT* понятий в практическом аспекте произошло только в 1972 г., в работе над проектом завода будущего. Передовое производство предполагало применение интегрированных систем, системы распределенной базы данных, компьютеры для инженерных расчетов и офисных работ, системы управления, механизмы учета и обработки сырья, деталей и инструментов, в сочетании со станками и людьми. Выделяя интерфейсные дуги между подсистемами Дуглас Т. Росс свел годами апробируемые идеи и опыт в соответствующую методологию проектирования и завершил проект в срок. А уже в следующем году новый инструмент использовался аналитиками в программе интегрированной компьютеризации производства *ICAM* для нужд ВВС США. К 1981 г. *SADT* использовали более чем в 50 компаниях при работе более чем над 200 проектами. С появлением персональных компьютеров и, соответственно, инструментальных средств проектирования систем, позиции *SADT* только упрочились. Причина такого успеха заключается в том, что *SADT* является полной методологией для создания описания систем, основанной на концепциях системного моделирования. Методология в усеченном варианте под названием *Integration Definition for Function Modeling – IDEF0* в 1993 году была утверждена в качестве федерального стандарта США. После опубликования стандарта он был успешно применен в самых различных областях бизнеса, показав себя эффективным средством анализа, конструирования и отображения бизнес-процессов. Подробные спецификации *IDEF0* можно найти на сайте <http://www.idef.com>. А в 2000 г. Госстандарт

России принял Руководящий документ «Методология функционального моделирования *IDEF0*» для целей реинжиниринга бизнес-процессов и процессов менеджмента качества.

1.1. *IDEF0*-модель

Согласно методологии *IDEF0*: «*М* есть модель системы *А*, если *М* может быть использована для получения ответов на вопросы относительно *А* с точностью *В*». Таким образом, в процессе анализа системы формируется ряд вопросов, ответы на которые и являются *целью моделирования*, они руководят созданием модели и направляют его. И только четко сформулированная цель «подскажет» когда процесс моделирования можно считать завершенным. Например, целью может быть написание должностных инструкций работников, выполняющих моделируемые бизнес-процессы, разработка технического задания на разрабатываемую информационную систему, улучшение системы документооборота, выявление причины брака в производстве продукции, выявление бизнес-процессов, уязвимых с точки зрения информационной безопасности.

Моделируемая система всегда связана с окружающей средой. Причем часто трудно сказать, где кончается система и начинается среда. По этой причине методология предписывает точное определение *границ системы*, необходимо решить, что входит в систему, а что лежит за ее пределами. Границы моделирования предназначены для обозначения ширины охвата предметной области и глубины детализации. Глубина детализации определяет степень подробности, с которой нужно проводить декомпозицию. Обычно, определяя входы и выходы системы на контекстной диаграмме, аналитик этой процедурой определяет и границы моделирования.

С определением модели тесно связана позиция, с которой наблюдается система и создается ее модель. Эта позиция называется *точкой зрения* данной модели. Скажем, описать работу механического цеха можно с точки зрения и механика, и начальника цеха. Но, если целью моделирования является написание учебного руководства для всего персонала, то только с позиции начальника цеха можно увидеть все виды работ, выполняемых в цехе. Основой для выбора точки зрения должна служить поставленная цель моделирования. И цель моделирования, и точка зрения выносятся на контекстную диаграмму. Однажды выбранная точка зрения остается неизменной для всех элементов модели. При необходимости могут быть созданы другие модели, отображающие систему с других точек зрения, однако каждая

отдельно взятая модель должна разрабатываться исходя из единственной заранее определенной точки зрения. Вот несколько примеров точек зрения при построении моделей: менеджер по продажам, клиент, поставщик, начальник службы безопасности.

После того как определены цель, границы моделирования и точка зрения модели, аналитик определяет выбор нужной информации о системе и форму ее подачи. Цель становится критерием окончания моделирования. Конечным результатом этого процесса является набор тщательно взаимосвязанных диаграмм, начиная с описания самого верхнего уровня всей системы и кончая подробным описанием деталей или операций системы. Итак, сочетая естественный и графический язык, *IDEF0*-модель представляет собой иерархически увязанные диаграммы, текст и глоссарий, имеющих ссылки друг на друга. Диаграммы могут быть четырех типов: контекстная (в каждой модели может быть только одна контекстная диаграмма), декомпозиции, дерева узлов, *FEO*- диаграммы (только для экспозиции).

1.2. Синтаксис и семантика графического языка *IDEF0*

Компонентами графического языка *IDEF0* (или нотации ¹ *IDEF0*) являются блоки, стрелки, диаграммы и правила.

Блоки представляют **функции**, определяемые как работа (*activity*), деятельность, процесс, операция, действие или преобразование. Блоки должны быть прямоугольными, с прямыми углами, нарисованы сплошными линиями. Внутри каждого блока помещается его имя и номер (рис. 1). Имя должно быть активным глаголом или глагольным оборотом, описывающим функцию. Например: «Обработать заявку», «Планировать ресурсы», «Проектировать систему». Номер блока размещается в правом нижнем углу. Номера блоков используются для их идентификации на диаграмме и в соответствующем тексте. Размеры блоков должны быть достаточными для того, чтобы включить его имя.



Рис. 1

¹ Нотация – система условных обозначений.

Стрелки (Arrow) или интерфейсные дуги показывают, какие данные или материальные объекты должны поступить на вход функции для того, чтобы эта функция могла выполняться, поэтому имя стрелки должно быть существительным или оборотом существительного. Например: «Отчет», «Клиент», «Инструкции», «Политика безопасности», «Договор». Стрелка формируется из одного или более отрезков прямых и наконечника на одном конце. Сегменты стрелок могут быть прямыми или ломаными (рис. 2). Стрелки должны быть нарисованы сплошными линиями (допускается их различная толщина) и могут состоять только из вертикальных или горизонтальных отрезков; отрезки, направленные по диагонали, не допускаются.

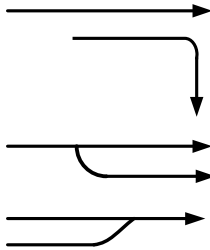


Рис. 2

Концы стрелок должны касаться внешней границы функционального блока, но не должны пересекать ее. Присоединение в углах не допускается.

Каждая сторона функционального блока имеет стандартное значение с точки зрения связи блок/стрелки. В свою очередь, сторона блока, к которой присоединена стрелка, однозначно определяет ее роль (рис. 3).

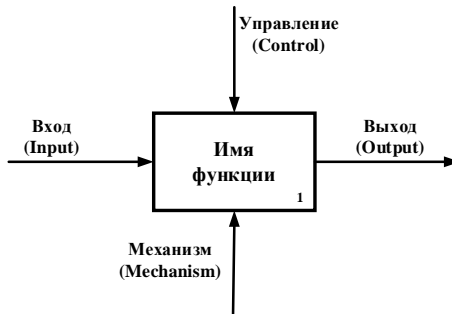


Рис. 3

Стрелки входа. Входы – это материал или информация, которые используются или преобразуются функцией для получения результата (выхода). Стрелка входа рисуется как входящая в левую грань блока. *Допускается, что функциональный блок может не иметь ни одной стрелки входа*, так как возможно, что некоторые функции ничего не преобразуют и не изменяют, например: функция «принятие решения руководством», не имеет стрелки входа, так как для принятия решения анализируется несколько факторов, но ни один из них непосредственно не преобразуется и не потребляется в результате принятия какого-либо решения. При описании технологических процессов не возникает проблем определения входов. Действительно, «Сырье» на рис. 4 – это нечто, что перерабатывается в процессе «Изготовление изделия» для получения результата.

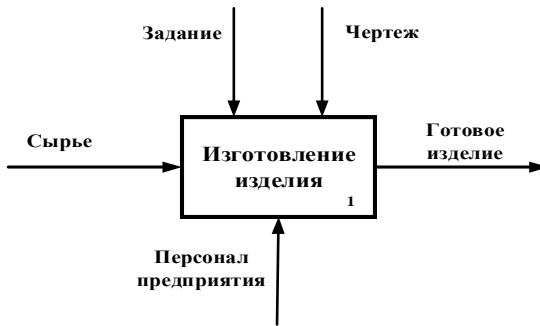


Рис. 4

При моделировании информационных систем, когда стрелками являются не физические объекты, а данные, не все так очевидно. Например, при приеме пациента врачом карта пациента может быть и на входе и на выходе, между тем качество этих данных меняется. В таком случае определение выхода должно указывать на то, что данные действительно были переработаны (например, на выходе – «Заполненная карта пациента»). Очень часто сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить то, перерабатываются/изменяются ли данные в работе или нет. Если изменяются, то скорее всего это вход, если нет – управление.

Стрелки управления. Стрелки, входящие в блок сверху – управления. Управления определяют условия, необходимые функции, чтобы произвести правильный выход. Так как управление контролирует поведение функции блока для обеспечения создания жела-

емого выхода, *каждый функциональный блок должен иметь хотя бы одну стрелку управления*. На рис. 4 стрелки «Задание» и «Чертеж» – управление для работы «Изготовление изделия». Управление влияет на функцию, но не преобразуется функцией. Управление часто существует в виде правил, инструкций, законов, политики, набора необходимых процедур или стандартов. Управление можно рассматривать как специфический вид входа.

В случаях, когда неясно, относить ли стрелку к входу или к управлению, предпочтительно относить ее к управлению до момента, пока неясность не будет разрешена. Допускается разветвление стрелки управления на вход и управление одновременно (рис. 5).

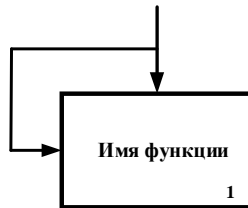


Рис. 5

Стрелки выхода. Стрелки, покидающие блок справа – выходы, т.е. данные или материальные объекты, произведенные функцией. Выход — это продукция или информация, получаемая в результате работы функционального блока. *Каждый функциональный блок должен иметь, как минимум, один выход*. Действие, которое не производит никакого четко определяемого выхода, не должно моделироваться вообще (по меньшей мере, должно рассматриваться в качестве одного из первых кандидатов на исключение из модели). При моделировании непроеизводственных предметных областей выходами, как правило, являются данные, в каком-либо виде обрабатываемые функциональным блоком. В этом случае важно, как было уже отмечено выше, чтобы названия стрелок входа и выхода были достаточно различимы по своему смыслу. Например, блок «Прием пациентов» может иметь стрелку «Данные о пациенте» как на входе, так и на выходе. В такой ситуации входящую стрелку можно назвать «Предварительные данные о пациенте», а исходящую — «Подтвержденные данные о пациенте».

Стрелки механизма. Стрелки, подключенные к нижней стороне блока, представляют механизмы. Стрелки, направленные вверх, идентифицируют средства, поддерживающие выполнение функции. Механизмы являются ресурсом, который непосредственно исполняет

моделируемое действие. С помощью механизмов исполнения могут моделироваться: ключевой персонал, техника и (или) оборудование. *Стрелки механизма исполнения могут отсутствовать в случае, если оказывается, что они не являются необходимыми для достижения поставленной цели моделирования.* На рис. 4 стрелка «Персонал предприятия» является механизмом для работы «Изготовление изделия».

Кроме стрелок механизма исполнения также могут использоваться стрелки, направленные вниз (рис. 6). Такие стрелки являются **стрелками вызова**. Стрелки вызова обозначают обращение из данной модели или из данной части модели к блоку, входящему в состав другой модели или другой части модели, обеспечивая их связь, т.е. разные модели или разные части одной и той же модели могут совместно использовать один и тот же элемент (блок).

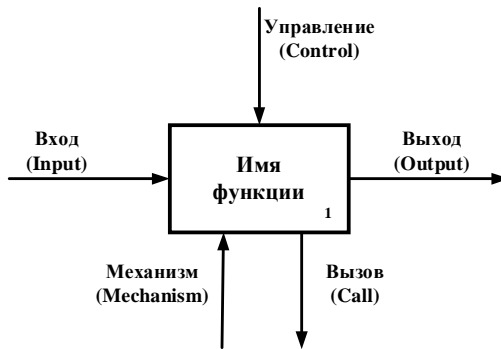


Рис. 6

В именах стрелок не должны использоваться следующие термины: функция, вход, управление, выход, механизм, вызов.

При практическом моделировании часто возникает вопрос об объектах, являющимися входами, выходами, управлениями и механизмами. В Приложении приведен словарь, содержащий некоторые универсальные связи. Но, необходимо понимать, что некоторые объекты могут быть использованы в разных группах (например, «Знания»).

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой. Каждая модель должна иметь контекстную диаграмму, на которой объект моделирования представлен единственным блоком с граничными стрелками. Эта диа-

грамма называется А-0 (А минус ноль). Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой и называются *границными* (начинаются у границы диаграммы и заканчиваются у блока или наоборот). Поскольку единственный блок представляет весь объект, его имя – общее для всего проекта. Это же справедливо и для всех стрелок диаграммы. Диаграмма А-0 устанавливает область моделирования и ее границ. Контекстная диаграмма должна содержать указание на точку зрения должностного лица или подразделения, с позиций которого создается модель. Точка зрения определяет, что и в каком разрезе можно увидеть в пределах контекста модели. Изменение точки зрения, приводит к рассмотрению других аспектов объекта. Аспекты, важные с одной точки зрения, могут не появиться в модели, разрабатываемой с другой точки зрения на тот же самый объект. Формулировка цели выражает причину создания модели, т.е. подразумевает перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет ее структуру.

В качестве примера рассмотрим моделирование одного из процессов системы пропускного режима: процесс пропуска посетителей на территорию предприятия по разовым пропускам [11]. Контекстная диаграмма приведена на рис. 7.

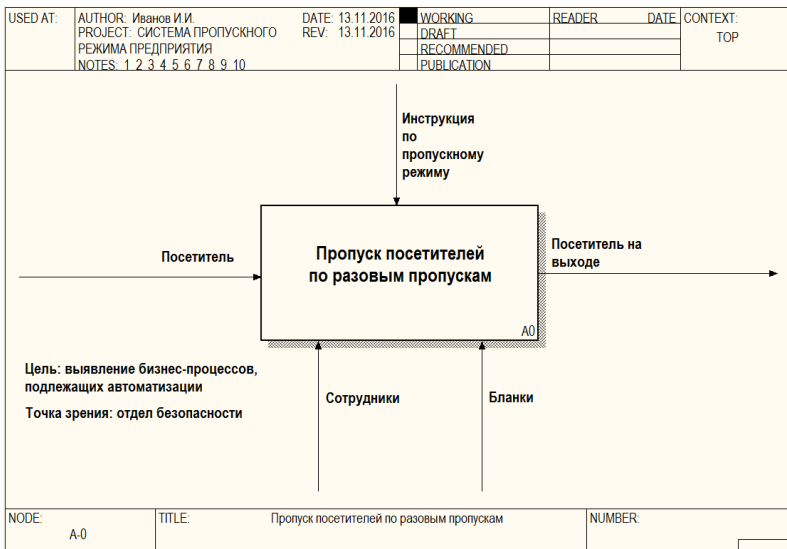


Рис. 7

После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется **функциональной декомпозицией**, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются **диаграммами декомпозиции**. Декомпозиция контекстной диаграммы именуется как **A0** (*A* ноль) или еще ее называют диаграммой основных процессов. Декомпозиция контекстной диаграммы проекта «Система пропускного режима предприятия» приведена на рис. 8.

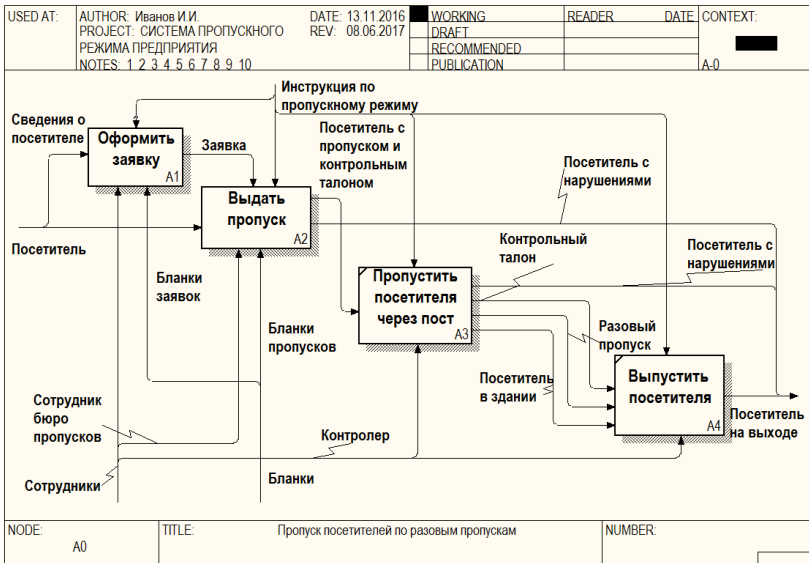


Рис. 8

После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и т. д., до достижения нужного уровня подробности описания. Декомпозиция функции «Оформить заявку» диаграммы A0 приведена на рис. 9. Допустимый интервал числа блоков на диаграмме декомпозиции 2-8. Декомпонировать работу на одну работу не имеет смысла, а диаграммы с количеством блоков более восьми получаются перенасыщенными и плохо читаются. Для обеспечения наглядности и лучшего понимания моделируемых процессов рекомендуется использовать от 3 до 6 блоков на одной диаграмме.

Диаграммы декомпозиции также могут именоваться **дочерними диаграммами** или **родительскими диаграммами**. Родительская

диаграмма – та, которая содержит один (в случае диаграммы *A-0*) или более родительских блоков. Дочерняя диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский блок, но описывает ее более подробно с помощью дочерних блоков и стрелок, обеспечивающих дополнительную детализацию родительского блока. Таким образом, дочерняя диаграмма как бы вложена в свой родительский блок.

То, что блок является дочерним и раскрывает содержание родительского блока на диаграмме предшествующего уровня, указывается специальным номером или ссылочным кодом, указанным в нижнем правом углу блока. Номер состоит из префикса и числа. Может быть использован префикс любой длины, но обычно используется префикс *A*. Функции декомпозиции диаграммы *A0* имеют номера *A1*, *A2*, *A3* и т. д. Например, показанные на рис. 10 коды означают, что диаграмма *A13* является декомпозицией 3-го блока диаграммы *A1* которая, в свою очередь является декомпозицией 1-го блока диаграммы *A0*, а сами коды образуются присоединением номера блока. Основное иерархическое отношение существует между родительским блоком и дочерней диаграммой, которая его подробно описывает (рис. 10).

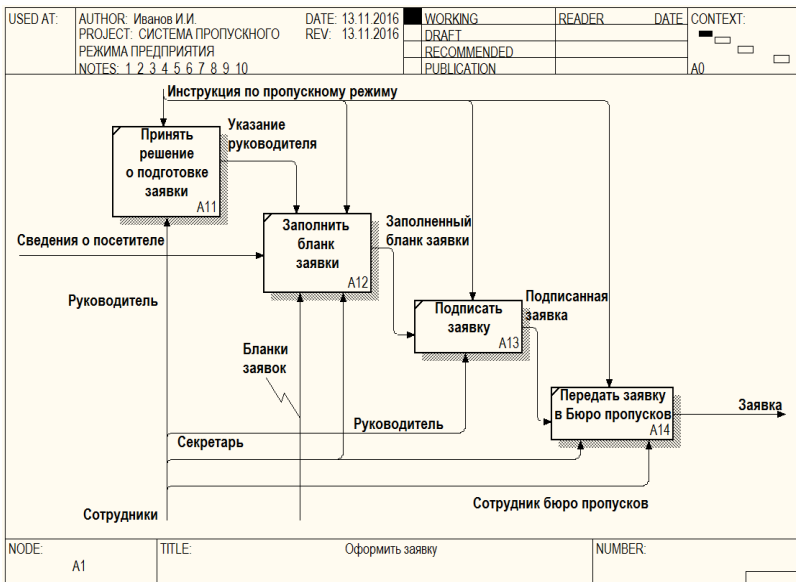


Рис. 9

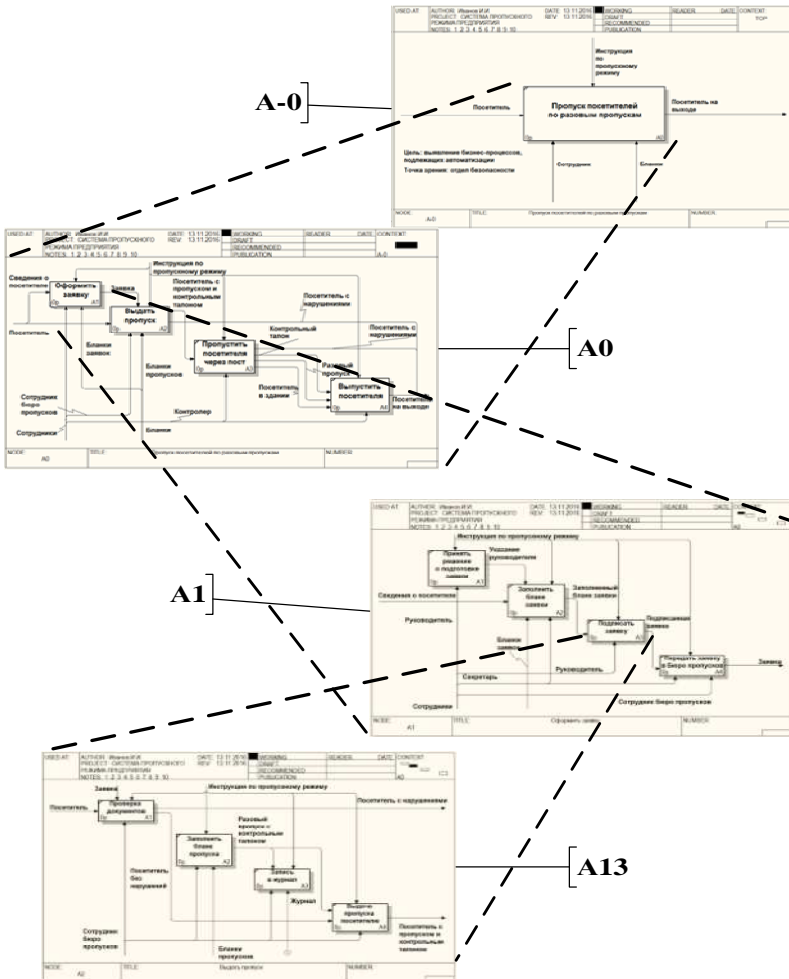


Рис. 10

В методологии *IDEF0* существуют **следующие типы отношений между функциями** в пределах одной диаграммы:

- доминирование;
- выход-управление;
- выход-вход;
- обратная связь по управлению;

- обратная связь по входу;
- выход-механизм.

Первое из перечисленных отношений определяется взаимным расположением блоков на диаграмме. Предполагается, что блоки, расположенные на диаграмме выше и левее, «доминируют» над блоками, расположенными ниже и правее. «Доминирование» понимается как влияние, которое одна функция оказывает на другие или же функция просто выполняется ранее. Остальные пять отношений описывают связи между блоками и изображаются соответствующими стрелками.

Отношение «выход-управление» (рис. 11) возникает тогда, когда выход одного блока служит управляющим воздействием на блок с меньшим доминированием.

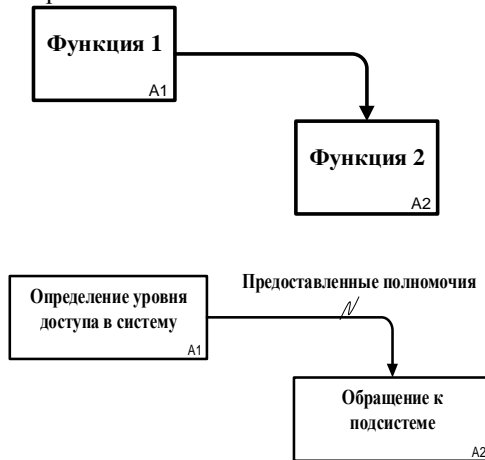


Рис. 11

Отношение «выход – вход» (рис.12, 13) возникает при соединении выхода одного блока с входом другого блока с меньшим доминированием.

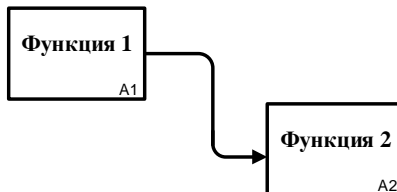


Рис. 12

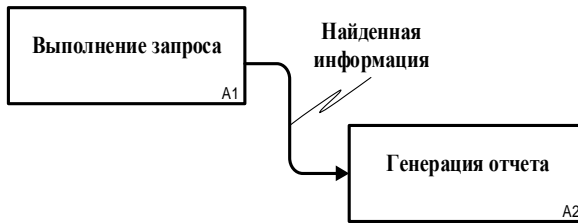


Рис. 13

Отношение «обратная связь по входу» (рис. 14) имеет место тогда, когда выход блока становится входом другого блока с большим доминированием. Обратите внимание, что связь между функциями внутри цикла обратной связи может быть не только «выход-вход», но и «выход-управление».



Рис. 14

Отношение «обратная связь по управлению» (рис. 15) возникает тогда, когда выход некоторого блока создает управляющее воздействие на блок с большим доминированием.

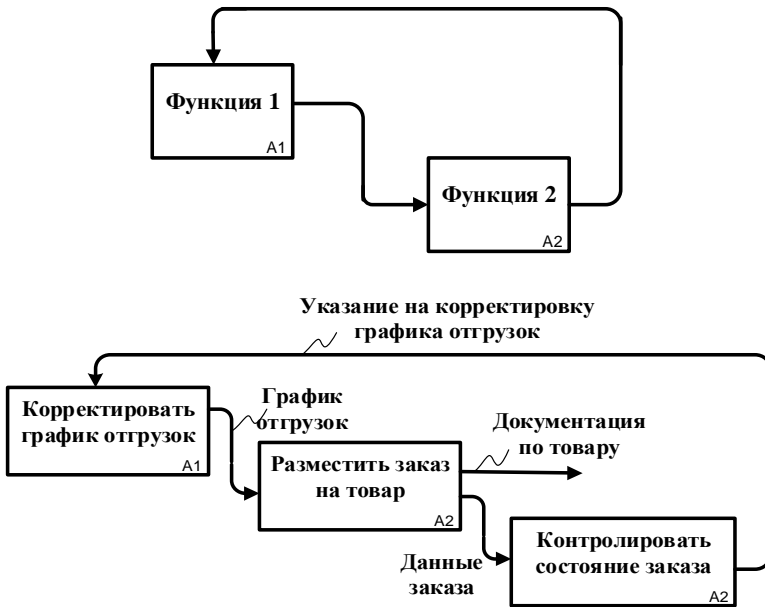


Рис. 15

Связи «выход-механизм» встречаются нечасто и представляют особый интерес. Они отражают ситуацию, при которой выход одной функции становится средством достижения цели для другой (рис. 16).

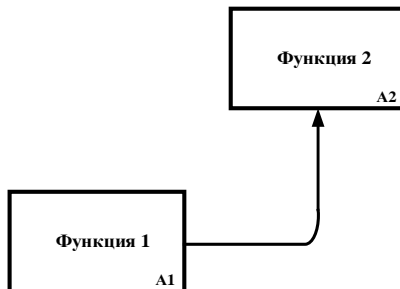


Рис. 16



Рис. 16 (Окончание)

Функция «Установить антивирусную программу», имеющая выход «Установленная антивирусная программа», который, в свою очередь, является механизмом для функции «Проверить разделы жесткого диска». Это означает, что установленная антивирусная программа необходима для того, чтобы начать процесс проверки. А в этом случае стрелка механизма обозначает строго последовательную взаимосвязь: приготовления должны быть завершены до начала работы. Поэтому связи «выход-механизм» характерны при распределении источников ресурсов (например, требуемые инструменты, обученный персонал, физическое пространство, оборудование, финансирование, материалы).

1.2.1. Стрелки на диаграмме декомпозиции

Граничные стрелки. На диаграмме декомпозиции граничные стрелки представляют входы, управления, выходы или механизмы родительского блока. Источник или потребитель граничных стрелок можно обнаружить, только изучая родительскую диаграмму. Все граничные стрелки на дочерней диаграмме (за исключением стрелок, помещенных в тоннель (см. ниже), должны соответствовать стрелкам родительского блока.

Явные стрелки. Явная стрелка имеет источником одну-единственную функцию и назначением тоже одну-единственную функцию.

Разветвляющиеся и сливающиеся стрелки. Одни и те же данные или объекты, порожденные одной функцией, могут использоваться сразу в нескольких других функциях. С другой стороны, стрелки, порожденные в разных функциях, могут представлять собой одинаковые или однородные данные или объекты, которые в даль-

нейшем используются или перерабатываются в одном месте. Для моделирования таких ситуаций в *IDEF0* используются разветвляющиеся и сливающиеся стрелки.

Смысл разветвляющихся и сливающихся стрелок передается именованием каждой ветви стрелок. Существуют определенные правила именования таких стрелок. Рассмотрим их на примере разветвляющихся стрелок. Если стрелка именована до разветвления, а после разветвления ни одна из ветвей не именована, то подразумевается, что каждая ветвь моделирует те же данные или объекты, что и ветвь до разветвления (рис. 17). Если стрелка именована до разветвления, а после разветвления какая-либо из ветвей не именована, то подразумевается, что она моделирует те же данные или объекты, что и ветвь до разветвления (рис. 18).



Рис. 17

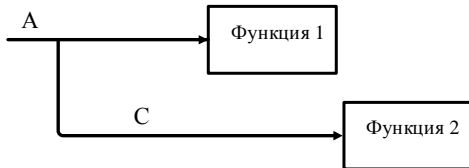


Рис. 18

Недопустима ситуация, когда стрелка до разветвления не именована, а после разветвления не именована какая-либо из ветвей (рис. 19).

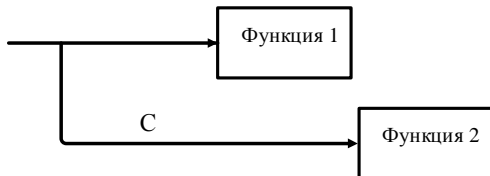


Рис. 19

Правила именования сливающихся стрелок полностью аналогичны – ошибкой будет считаться стрелка, которая после слияния не именована

Стрелки, помещенные в «тоннель». Тоннель – круглые скобки в начале или окончании стрелки. Тоннельные стрелки означают, что данные, выраженные этими стрелками, не рассматриваются на родительской диаграмме или на дочерней диаграмме. Тоннелирование может быть применено для изображения малозначимых стрелок.

Если на какой-либо диаграмме нижнего уровня необходимо изобразить малозначимые данные или объекты, которые не обрабатываются или не используются родительскими функциями, вплоть до диаграммы А-0, то в результате малозначимая стрелка будет изображена на всех уровнях и затруднит чтение всех диаграмм, на которых она присутствует. Выходом является тоннелирование стрелки на самом нижнем уровне. Другим примером тоннелирования может быть ситуация, когда стрелка мигрирует с верхнего уровня на нижний, загромождая дочерние диаграммы. В этом случае стрелка на нижнем уровне может быть удалена, а на родительской диаграмме затоннелирована. Пример использования тоннелированных стрелок показан в разделе «Гибридные функциональные модели».

1.2.2. ICOM-коды

Диаграмма декомпозиции предназначена для детализации функции. В отличие от моделей, отображающих структуру организации, функция на диаграмме верхнего уровня в *IDEFO* – это не элемент управления нижестоящими функциями. Функции нижнего уровня – это то же самое, что и функции верхнего уровня, но в более детальном изложении. Как следствие этого границы функции верхнего уровня – это то же самое, что и границы диаграммы декомпозиции. *ICOM*-коды связывают граничные стрелки на дочерней диаграмме со стрелками родительского блока. Нотация, названная *ICOM*-кодом, определяет значения соединений. Буквы *I*, *C*, *O* или *M*, написанные около несвязанного конца граничной стрелки на дочерней диаграмме идентифицируют стрелку как Вход (*Input*), Управление (*Control*), Выход (*Output*) или Механизм (*Mechanism*) в родительском блоке. Буква следует за числом, определяющим относительное положение точки подключения стрелки к родительскому блоку. Например, код «C1», написанный возле граничной стрелки на дочерней диаграмме, указывает, что эта стрелка соответствует первой (считая слева) управляющей стрелке родительского блока (рис. 20). Это кодирова-

ние связывает каждую дочернюю диаграмму со своим родительским блоком. Если блоки на дочерней диаграмме подвергаются дальнейшей декомпозиции и подробно описываются на дочерних диаграммах следующего уровня, то на каждую новую диаграмму назначаются новые *ICOM*-коды, связывающие граничные стрелки этих диаграмм со стрелками их родительских блоков. Иногда буквенные *ICOM*-коды, определяющие роли граничных стрелок (вход, управление, механизм), могут меняться при переходе от родительского блока к дочерней диаграмме. Например, управляющая стрелка в родительском блоке может быть входом на дочерней диаграмме. Аналогично, вход родительского блока может быть управлением для одного или более дочерних блоков.

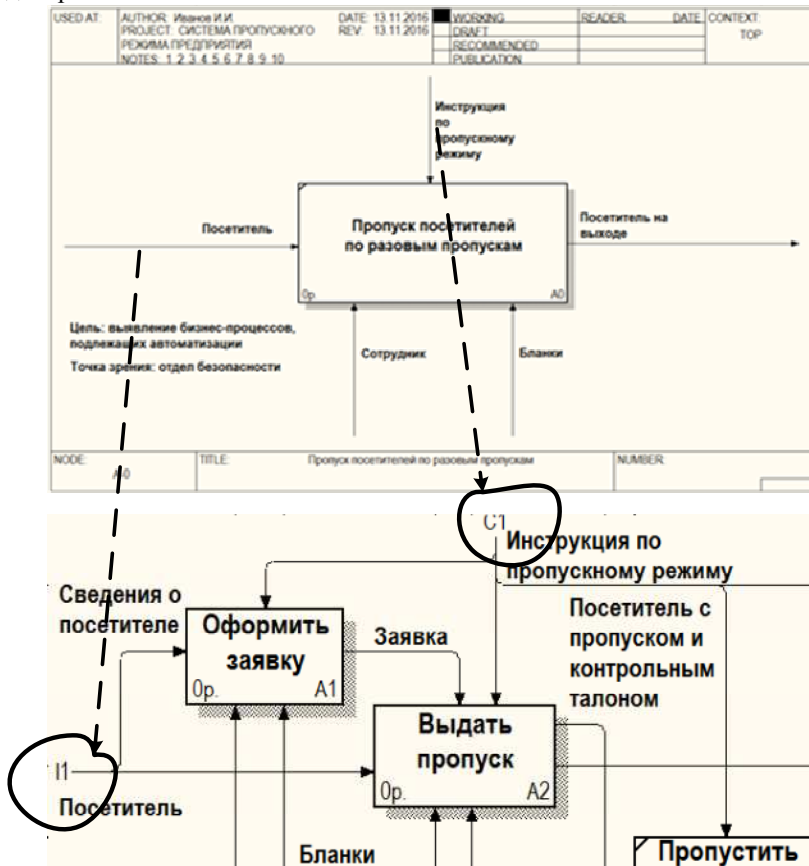


Рис. 20

1.3. Диаграммы «Дерево узлов» и FEO

Диаграмма дерева узлов показывает иерархию работ в модели и позволяет рассмотреть всю модель целиком, но не показывает взаимосвязи между функциями (стрелки) (рис. 21). Процесс создания модели является итерационным, следовательно, функции могут менять свое расположение в дереве узлов многократно. Чтобы не запутаться и проверить способ декомпозиции, следует после каждого изменения создавать диаграмму дерева узлов. Диаграмм деревьев узлов может быть в модели сколь угодно много, поскольку дерево может быть построено на произвольную глубину и не обязательно с корня (контекстной диаграммы).

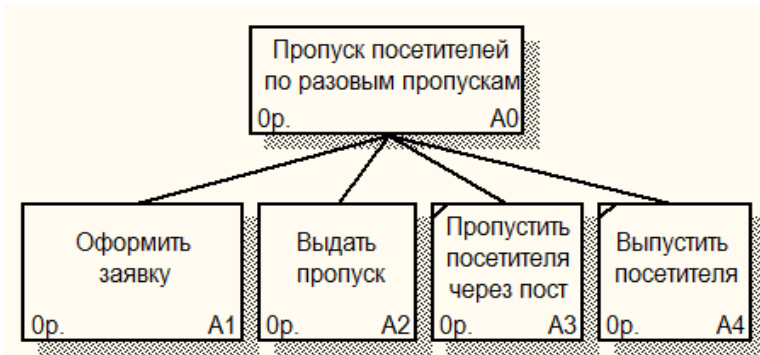


Рис. 21

Диаграммы *FEO* (*For Exposition Only*) – «только для экспозиции», строятся для иллюстрации отдельных фрагментов модели, для иллюстрации альтернативной точки зрения либо для специальных целей. Диаграммы *FEO* позволяют нарушить любое синтаксическое правило, поскольку по сути являются просто картинками. Функция на диаграмме *FEO* может не иметь стрелок управления и выхода. С целью обсуждения определенных аспектов модели с экспертом предметной области может быть создана диаграмма только с одной функцией и одной стрелкой, поскольку стандартная диаграмма декомпозиции содержит множество деталей, не относящихся к теме обсуждения и дезориентирующих эксперта. Но если *FEO* используется для иллюстрации альтернативных точек зрения, рекомендуется все-таки придерживаться синтаксиса *IDEF0*. Номер диаграммы формируется

следующим образом: номер родительской диаграммы по узлу плюс постфикс *F*. Например, с целью анализа механизмов функций диаграммы *A0* создана диаграмма *FEO A0F*, представленная на рис. 22.

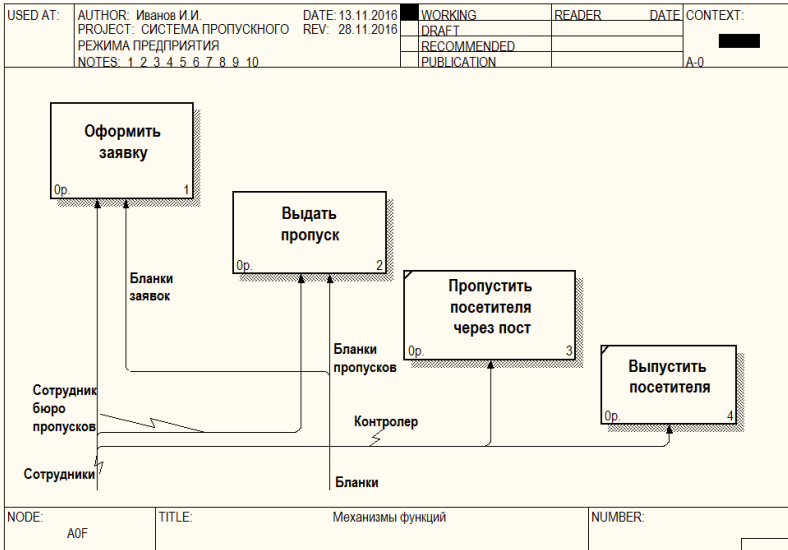


Рис. 22

1.4. Рекомендации по рисованию диаграмм

В *IDEF0* существуют соглашения по рисованию диаграмм, которые призваны облегчить чтение и экспертизу модели:

1. Функции должны располагаться по диагонали с левого верхнего в правый нижний угол (порядок доминирования). Нарушать диагональное расположение блоков по возможности не следует. Порядок доминирования подчеркивает взаимосвязь функций, позволяет минимизировать изгибы и пересечения стрелок.

2. Следует максимально увеличивать расстояние между входящими или выходящими стрелками на одной грани блока.

3. Следует максимально увеличить расстояние между блоками, поворотами и пересечениями стрелок.

4. Если две стрелки проходят параллельно (начинаются из одной и той же грани одного блока и заканчиваются на одной и той же

границы другого блока), то по возможности следует их объединить и назвать единым термином.

5. Обратные связи по входу рисуются «нижней» петлей, обратная связь по управлению – «верхней».

6. Циклические обратные связи (рис. 23) следует рисовать только в случае крайней необходимости, когда подчеркивают значение повторно используемого объекта.



Рис. 23

7. Диаграммы должны быть сбалансированы. Это означает, что у функций не должно происходить ситуации, изображенной на рис. 24: у Функции 1 входящих стрелок и стрелок управления значительно больше, чем выходящих. Но данная рекомендация может не выполняться в моделях, описывающих производственные процессы. Например, при описании процедуры сборки в блок может входить множество стрелок, описывающих компоненты изделия, а выходить одна стрелка – готовое изделие.

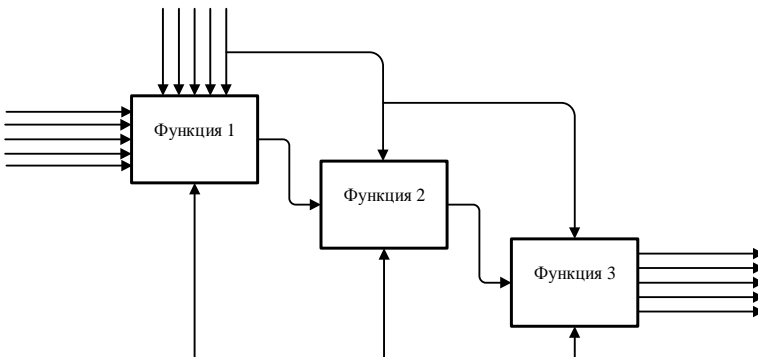


Рис. 24

8. Необходимо стремиться к тому, чтобы количество блоков на диаграммах нижних уровней было бы меньше количества блоков

на родительских диаграммах, т.к. по мере декомпозиции функции должны упрощаться.

9. При соединении большого числа блоков необходимо избегать необязательных пересечений стрелок. В реальных моделях к каждой функции может подходить и от каждой может отходить около 10 стрелок. Если диаграмма содержит 6-8 работ, то она может содержать 30 - 40 стрелок, причем они могут сливаться, разветвляться и пересекаться. Такие диаграммы могут стать очень плохо читаемыми, поэтому следует минимизировать число петель и поворотов каждой стрелки (рис. 25).

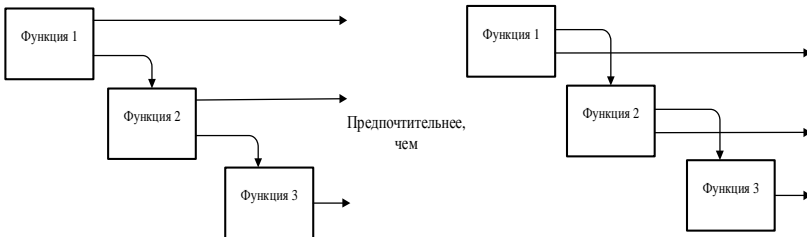


Рис. 25

1.5. Текст и глоссарий

Диаграмме может быть поставлен в соответствие структурированный текст, представляющий собой краткий комментарий к содержанию диаграммы. Текст используется для объяснений и уточнений характеристик, потоков, внутриблочных соединений и т.д. Текст не должен использоваться для описания и без того понятных блоков и стрелок на диаграммах.

Глоссарий предназначен для определения аббревиатур (акронимов), ключевых слов и фраз, используемых в качестве имен и меток на диаграммах. Глоссарий определяет понятия и термины, которые должны быть одинаково понимаемы всеми участниками разработки и пользователями модели, чтобы правильно интерпретировать ее содержание.

1.6. Цикл «Автор-Читатель»

Методология *IDEF0* является единственной, содержащей в своем активе не только средства описания систем, но и правила получе-

ния согласованных и достоверных моделей. Ведь модель есть результат скоординированной коллективной работы, при которой авторы (системные аналитики) создают первоначальные диаграммы, основанные на собранной информации об объекте моделирования, и передают их другим участникам проекта для рассмотрения и формулирования замечаний. Правила требуют, чтобы каждый читатель (эксперт предметной области), у которого есть замечания к диаграмме, сделал их письменно и передал автору диаграммы. Для больших проектов предусматривается сотрудник-координатор, организующий хранение материалов и распространение их между участниками процесса. Этот цикл – «Автор-Читатель», продолжается до тех пор, пока диаграммы, а затем и вся модель не будут приняты и экспертами, и системными аналитиками. Для именования стадий готовности диаграмм используются термины *DRAFT* (рабочая версия), *WORKING* (эскиз), *RECCOMENDED* (рекомендовано), *PUBLICATION* (публикация), составляющие, на самом деле, стандартный набор методологии *SADT*, унаследованный *IDEF0*. Диаграммы проходят через серию последовательных улучшений до тех пор, пока они в точности не будут отображать реальное положение вещей и соответствовать цели моделирования. Специальная группа людей, ответственных за то, что создаваемая в процессе анализа модель приемлема для дальнейшего использования, называется Комитетом технического контроля. Если модель признана Комитетом применимой, она утверждается и публикуется. В противном случае авторам направляются замечания для необходимой доработки. Только после прохождения экспертизы без замечаний аналитику можно приступить к следующему этапу декомпозиции.

Процесс управления моделированием иллюстрируется на рис. 26. Диаграмма с помощью отношений обратной связи отражает тот факт, что данный процесс – итеративная процедура, приводящая к точному описанию системы. Механизмом функции «Сбор информации» являются «Документация», «Эксперты» и «Аналитики». Это означает, что системные аналитики собирают необходимую информацию с помощью анкетирования, устных опросов экспертов предметной области, изучения документации. Документы, представляющие в этой связи интерес для аналитиков, перечислены далее, в разделе «Бизнес-консалтинг».

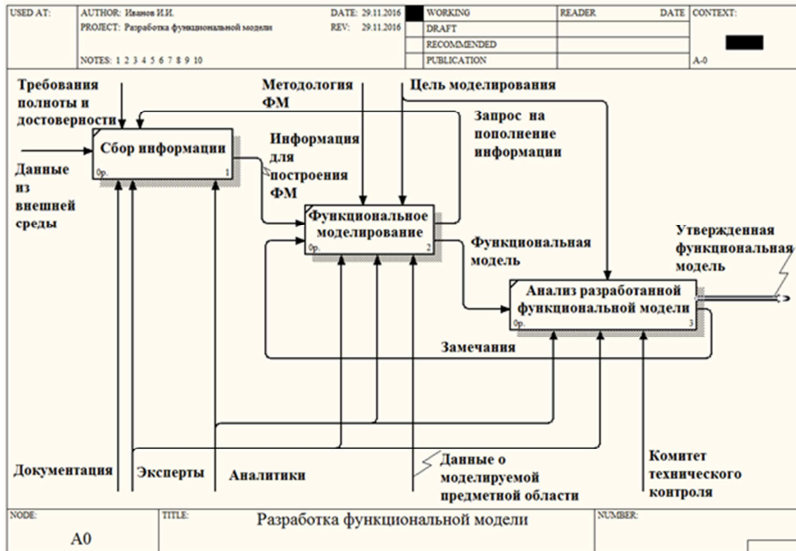


Рис. 26

2. Методология *DFD*

Методология *DFD* (*Data flow diagram* – диаграмма потоков данных) разработана независимо от *IDEF0*. Методология *DFD* изначально была создана с целью разработки программного обеспечения. Авторами одной из первых нотаций *DFD* (1979 г.) стали Эд Йордан и Том де Марко. В настоящее время наиболее распространенной является нотация, созданная Крисом Гейном и Тришем Сарсоном. Далее речь будет идти о нотации Гейна-Сарсона.

2.1. Синтаксис и семантика графического языка *DFD*

Также как в методологии *IDEF0*, компонентами графического языка *DFD* являются блоки, стрелки, диаграммы и правила. Блоками различной формы отображаются процессы (функции), внешние сущности (внешние ссылки), хранилища данных (накопители данных), системы/подсистемы. Стрелками – потоки данных.

Процессы. В *DFD* функции изображаются прямоугольниками со скругленными углами (рис. 27). Уникальный номер процесса необходим для ссылок на него внутри диаграммы. Этот номер может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели. Каждый процесс должен иметь имя в виде предложения с глаголом в неопределенной форме с последующим дополнением, например: «Ввести сведения о клиенте», «Выдать список должников». Использование таких глаголов, как «обработать», «модернизировать» или «отредактировать» означает, как правило, недостаточно глубокое понимание данного процесса и требует дальнейшего анализа.

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т.д.

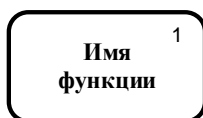


Рис. 27

Системы/подсистемы. Преобразование информации может показываться как с точки зрения процессов, так и с точки зрения систем/подсистем (рис. 28). Если вместо имени процесса «Выдать данные об успеваемости студентов» написать «Подсистема данных об успеваемости студентов», тогда этот блок на диаграмме стоит рассматривать, как подсистему. Необходимость использования данного элемента нотации, связана со сложностью моделируемой системы (подробнее см. *Контекстная диаграмма*). Имя внутри блока – наименование системы/подсистемы в виде предложения с подлежащим и соответствующими определениями и дополнениями.

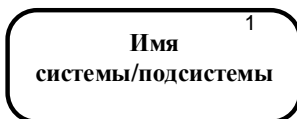


Рис. 28

Внешние сущности. Внешние сущности (или внешние ссылки) изображаются в виде прямоугольника с тенью (рис. 29) и обычно располагаются по краям диаграммы. Внешняя сущность – материальный предмет или физическое лицо, представляющее собой источник или приемник информации, например: персонал, поставщики, склад. Ее имя должно содержать существительное, например, «Клиент», «Заказчик», «Эксперт». Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой системы. Объекты, представленные внешними сущностями, не должны участвовать в процессах обработки. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь диаграммы анализируемой системы, если это необходимо, или, наоборот, часть процессов системы может быть вынесена за пределы диаграммы и представлена как внешняя сущность.



Рис. 29

Потоки данных. Потоки изображаются стрелками (рис. 30), ориентация которых указывает направление движения объектов (данных, информации). В отличие от стрелок в *IDEFO*, которые иллюстрируют отношения, стрелки в *DFD* показывают, как объекты реально перемещаются от одного действия к другому: информация, передаваемая по кабелю между двумя устройствами; пересылаемые по почте письма и т.д. Двухнаправленные стрелки (рис. 31) применяются для описания диалогов типа команды-ответа между функциями, между функцией и внешней сущностью и между внешними сущностями. Каждый поток имеет имя, отражающее его содержание. Имя должно быть существительным или оборотом существительного. Например: «Кредитная карта», «Протокол обслуживания».

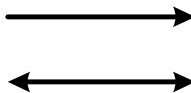


Рис. 30

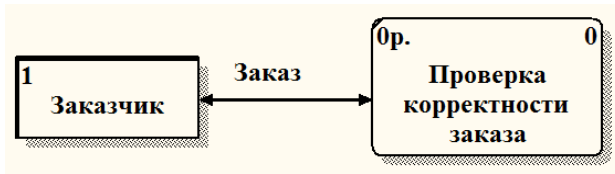


Рис. 31

В *DFD* каждая сторона блока, отображающего функцию, не имеет четкого назначения, как в *IDEF0*, поэтому все стрелки, независимо от того к какой грани они подходят или выходят, считаются стрелками входа или выхода. Так как блок моделирует функцию, преобразующую входы в выходы, очевидно, что должна быть хотя бы одна стрелка входа и одна стрелка выхода.

Методология *DFD*, строго говоря, управления и механизмы не поддерживает. Но, если проанализировать модели *информационных систем*, то можно заметить, что на самом деле, как и в *IDEF0*, *DFD* структурируется фактически тоже на пять групп операторов: функцию, вход, выход, управление, механизм. Особенности состоят лишь в способах представления управлений и механизмов на *DFD*-диаграммах.

На рис. 32 приведена, в самом общем виде, *IDEF0*-модель работы информационной системы, работающая с буквенно-цифровой информацией. Входная информация, так или иначе, преобразуется в выходную по запросу. Очевидно, что вход «запрос» играет роль управления работой информационной системы. В соответствии с определением *IDEF0* управление выражает «условия, необходимые для производства корректного выхода». Ту же роль играет в информационной системе запрос. Например, если на вход информационной системы будут поданы цифры 2, 3, то на выходе мы получим: 5 – при запросе «сложить» и 6 – при запросе «умножить».

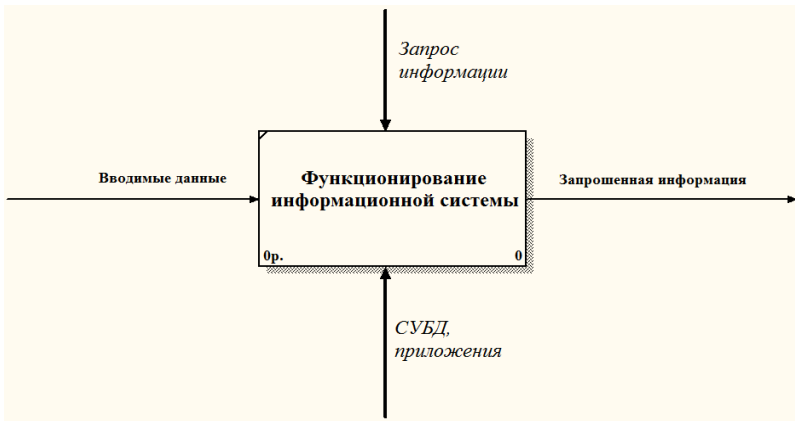


Рис. 32

Следует также добавить, что управления информационных систем имеют ту же природу, что и все *DFD*-связи, исключительно информационную (те или иные запросы информации). По этой причине, вероятно, они не выделяются в отдельную группу, в отличие от *IDEF0*, в которой управляющие воздействия могут быть как материальной, так и информационной природы. Тем не менее, не следует забывать, как при моделировании в *IDEF0*, о введении стрелок в состав входов *DFD*, символизирующих управление работой информационной системы, как в целом, так и ее фрагментов.

На рис. 33 приведена *DFD*-модель работы информационной системы. В некоторых интерпретациях нотации Гейна-Сарсона механизмы изображаются в нижней части блока.

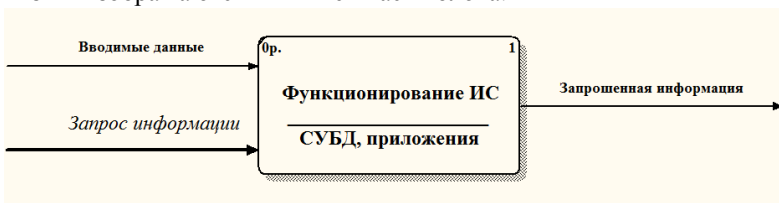


Рис. 33

Следует добавить, что механизмы информационной системы имеют как информационную, так и материальную природу (средства вычислительной техники), а в большинстве случаев еще и «челове-

ческую» для интерактивных систем.

Хранилище данных. В отличие от стрелок, описывающих объекты в движении, хранилище данных (накопитель данных) изображают объекты в покое (рис. 34). Накопитель представляет собой абстрактное устройство для хранения объектов (данных), которые можно в любой момент туда поместить и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

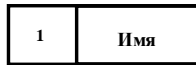


Рис. 34

В материальных системах хранилища данных изображаются там, где объекты ожидают обработки, например в очереди. В системах обработки информации хранилища данных являются инструментом, сохраняющим и извлекающим данные для последующего использования в процессах. Накопитель может быть реализован физически в виде ящика в картотеке, таблицы в оперативной памяти, файла и т.д. Хранилище данных в общем случае является прообразом будущей базы данных и описание хранящихся в нем данных должно быть увязано с информационной моделью. Имя накопителя должно идентифицировать его содержимое и быть существительным во множественном числе (рис. 35, 36), также имя выбирается из соображения наибольшей информативности для аналитика.

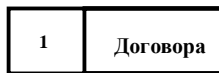


Рис. 35

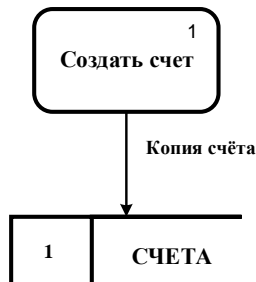


Рис. 36

В *DFD*-модели для всех потоков данных и хранилищ данных должно выполняться правило сохранения информации: все поступающие куда-либо данные должны быть считаны, а все считываемые данные должны быть записаны.

Слияние и разветвление стрелок. В *DFD* стрелки могут сливаться и разветвляться, что позволяет описать декомпозицию стрелок. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя (рис. 37, 38).

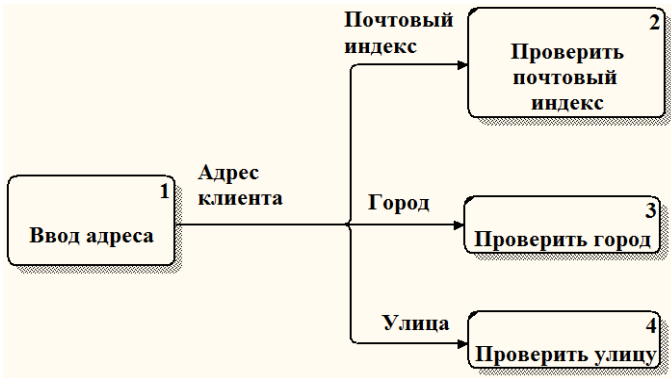


Рис. 37



Рис. 38

Нумерация объектов. В *DFD* номер каждого блока может включать префикс, номер родительской функции и номер объекта. Номер объекта – это уникальный номер блока на диаграмме, например А3.2. Уникальный номер имеют хранилища данных и внешние сущности независимо от их расположения на диаграмме. Каждое хранилище данных имеет префикс *D* и уникальный номер, например *D5*. Каждая внешняя сущность имеет префикс *E* и уникальный номер, например *E5*.

Контекстная диаграмма. Контекстная *DFD*-диаграмма часто состоит из одного функционального блока и нескольких внешних сущностей. Обычно при моделировании относительно простых систем строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится главный процесс, соединенный с приемниками и источниками информации (внешними сущностями). Например, на рис. 39 представлена контекстная диаграмма автоматизированной системы управления Завода XXX. Моделирование осуществляется с целью формирования требований к проектируемой информационной системе. Внешние сущности: Поставщик, Подрядчик, Торговый дом, Органы надзора, обмениваются с АСУП Завода с помощью обобщенных информационных потоков, название которых отражает направление обмена.

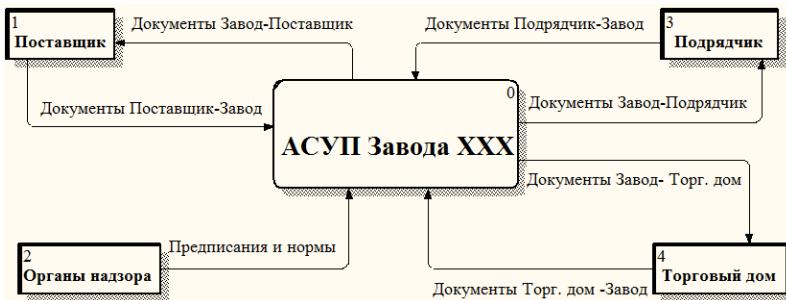


Рис. 39

Если же для сложной системы ограничиться единственной контекстной диаграммой, то она будет содержать слишком большое количество источников и приемников информации, которые трудно расположить на листе бумаги нормального формата, и кроме того, единственный главный процесс не раскроет структуры, скажем, распределенной системы. К признаками сложности отнесем:

- наличие большого количества внешних сущностей (десять и более);
- распределенная природа системы;
- многофункциональность системы с уже сложившейся или выявленной группировкой функций в отдельные подсистемы.

Для сложных информационных систем, имеющих вышеперечисленные показатели сложности, строится *иерархия контекстных диаграмм*. При этом контекстная диаграмма верхнего уровня содержит набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем. Иерархия контекстных диаграмм определяет взаимодействие основных функциональных подсистем информационной системы, как между собой, так и с внешними входными и выходными потоками данных и внешними сущностями, с которыми взаимодействует система. А это особенно полезно на самой ранней стадии проектирования.

Требование того, что модель должна строиться с единственной точки зрения и должна иметь четко определенные цель и границы, сохраняется.

После построения контекстных диаграмм полученную модель следует проверить на полноту исходных данных об объектах системы и изолированность объектов (отсутствие информационных связей с другими объектами).

Декомпозиция. Диаграммы верхних уровней иерархии – контекстные диаграммы, определяют основные процессы или подсистемы с внешними входами и выходами. Они, так же как и в *IDEF0*, детализируются при помощи диаграмм нижнего уровня. Декомпозиция продолжается, создавая многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень подробности, на котором процесс становится элементарным и детализировать его далее невозможно.

При декомпозиции должно выполняться правило: при детализации подсистемы или процесса диаграмма в качестве внешних источников/приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, хранилища данных), с которыми имеет информационную связь детализируемая подсистема или процесс на родительской диаграмме.

Правило нумерации поддерживает иерархическую структуру. Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2, 12.3 и т.д.

Построение иерархии диаграмм невозможно без предваритель-

ного определения содержания всех потоков и хранилищ данных.

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

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

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

Мини-спецификация. Конечной вершиной иерархии диаграмм является мини-спецификация (описание логики процесса). Мини-спецификация — это алгоритм описания задач, выполняемых процессами, множество всех мини-спецификаций является полной спецификацией системы. Мини-спецификации содержат номер и/или имя процесса, списки входных и выходных данных и тело(описание) процесса, являющееся спецификацией алгоритма или операции, трансформирующей входные потоки данных в выходные

Решение о завершении декомпозиции процесса и использовании мини-спецификации принимается аналитиком исходя из следующих критериев:

- наличия у процесса относительно небольшого количества входных и выходных потоков данных (2-3 потока);
- возможности описания преобразования данных процессом в виде последовательного алгоритма;

- выполнения процессом единственной логической функции преобразования входной информации в выходную;
- возможности описания логики процесса при помощи миниспецификации небольшого объема (не более 20-30 строк).

Верификация. После построения законченной модели системы ее необходимо верифицировать (проверить на полноту и согласованность). В полной модели все ее объекты (подсистемы, процессы, потоки данных) должны быть подробно описаны и детализированы. Выявленные недетализированные объекты следует детализировать, вернувшись на предыдущие шаги разработки.

Пример. Рассмотрим процесс, знакомый каждому студенту: «Сдать экзамен». Контекстная диаграмма *A-0* и диаграмма декомпозиции *A0* представлены на рис. 40. Внешними сущностями являются «Директорат» и «Студент». Для сдачи экзамена необходимо, чтобы у студента была зачетная книжка и допуск к экзамену. Недопущенные к экзамену студенты отмечены в ведомости, подготовленной директором. После экзамена ведомость с оценками передается в директорат. Результирующими потоками данных будут «Ведомость с оценками» и «Зачетная книжка с оценкой».

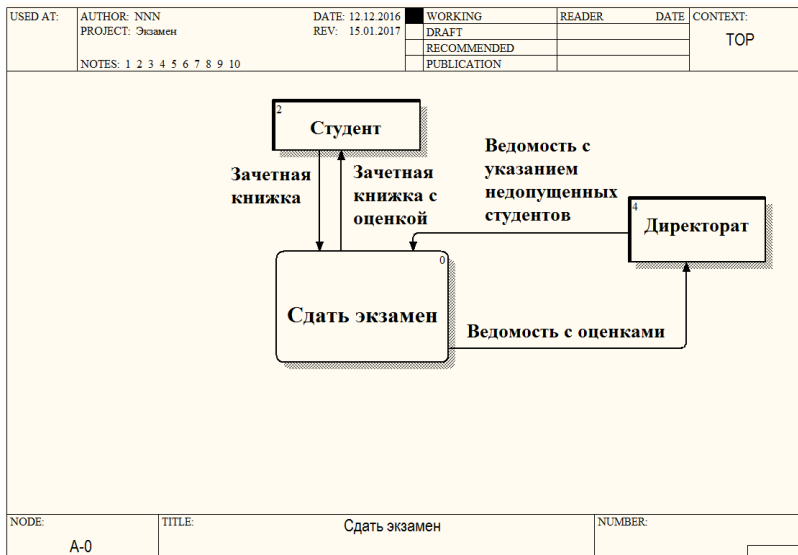


Рис. 40

Декомпозиция контекстной диаграммы представлена на рис. 41. Согласно правилам на DFD-диаграммах стрелки могут заканчиваться

на работах, хранилищах и внешних сущностях, т.е. граничные стрелки отсутствуют. Поэтому Студент и Преподаватель хотя и являются участниками процесса, представлены внешними сущностями.

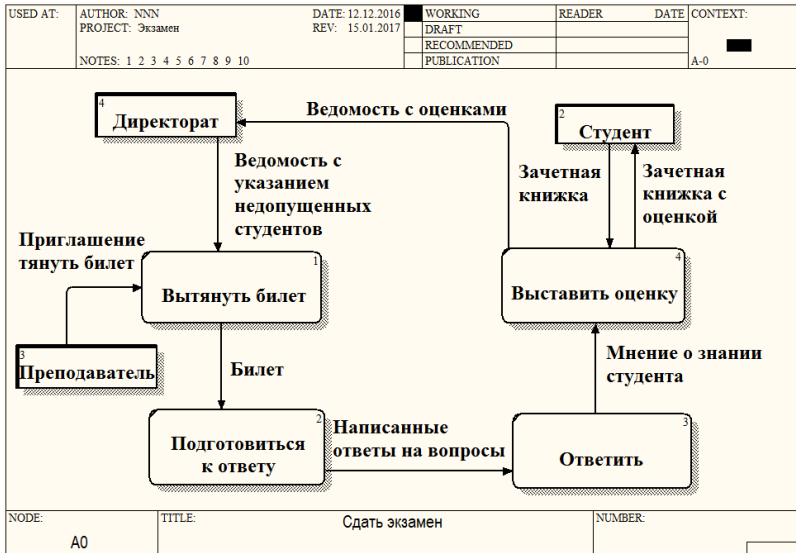


Рис. 41

2.2. Сравнительный анализ DFD и IDEF0

В отличие от *IDEF0*, являющейся основой для моделирования материально-информационных систем, *DFD*-методика поддерживает моделирование информационных систем. Отсюда и вытекают их различия, достоинства и недостатки:

1. Методология *IDEF0* успешно работает при моделировании хорошо специфицированных и стандартизованных бизнес-процессов, поэтому она и принята на Западе в качестве типовой. В российском бизнесе процессы слабо типизированы, поэтому предпочтительнее описывать системы с точки зрения организации и/или реорганизации потоков информации, используя методологию *DFD*. В *DFD* отсутствуют какие-либо ограничения на топологию связей функций.

2. Для систем обработки информации, таких как системы документооборота, управления, стирается смысловое различие между входами-выходами, с одной стороны, и управлениями-механизмами, с другой: входы, выходы и управления являются потоками данных и/или управления и правилами их трансформации. Поэтому анализ системы при помощи потоков данных и процессов, их преобразующих, является более надежным. А *IDEF0*-диаграммы значительно менее выразительны и удобны для моделирования систем обработки информации в силу жесткой типизации интерфейсных дуг (вход, выход, управление, механизм). Кроме того, методология *DFD* имеет более богатый набор элементов, способных описать информационные потоки: хранилища данных, внешние сущности.

3. В методологии *IDEF0* моделирование обрывается при достижении нужного уровня декомпозиции. Наличие мини-спецификаций *DFD*-процессов нижнего уровня позволяет построить полную функциональную спецификацию разрабатываемой системы и тем расширить возможности применения созданной модели.

4. Ограничение на количество блоков на диаграмме *IDEF0*, вынуждает искусственно детализировать систему, увеличивая объем модели.

5. *DFD*-диаграммы плохо отражают факторы, оказывающие влияние на документооборот, но стандартно не формализуемые. Например, процесс визирования документа, не отраженный в его бумажной форме.

3. Методология *IDEF3*

Методология *IDEF3* (*Integrated DEFinition for Process Description Capture Method*) разработана в конце 1980 гг. в США для моделирования последовательности действий и процессов анализируемой системы. *IDEF3*, как правило, дополняет *IDEF0*, уточняя картину процесса, привлекая внимание аналитика к очередности выполнения функций и бизнес-процессов в целом. Логика этой методологии позволяет строить и анализировать альтернативные сценарии развития изучаемых бизнес-процессов (модели типа «Что – если?»).

Нотацию *IDEF3* целесообразно применять в случае относительно простых процессов нижнего уровня, т.е. процессов уровня рабочих мест. В этом случае модель процесса может служить основой для создания документов, регламентирующих работу исполнителей.

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 технологические процессы;
- 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;
- описывать логику взаимодействия информационных потоков;
- разрабатывать поведенческие модели технологических процессов. Для создания поведенческих моделей используются методы имитационного моделирования, теория систем массового обслуживания, сети Петри.

3.1. Синтаксис и семантика графического языка *IDEF3*

Компонентами графического языка *IDEF3* являются блоки, стрелки, диаграммы, правила. Блоками различной формы отображаются единицы поведения (функции), объекты ссылок, перекрестки. Стрелками – связи, показывающие взаимоотношения функций. Методология *IDEF3* не ограничивает аналитика чрезмерно жесткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей.

Единица поведения. *UOW (Unit of Work)* обычно переводится в русскоязычных текстах как «единица работы», но слово *work*, помимо его общеупотребительного перевода как «работа», имеет еще целый ряд вариантов перевода на русский язык. Кроме того, в первоисточнике текста *IDEF3* для обозначения этого оператора используется другой термин – *UOB (Unit Of Behavior)* – «единица поведения», что вполне соответствует сути *IDEF3* – методологии, описывающей последовательность событий, происходящих при работе моделируемой системы, т. е. описывающей поведение системы. Поэтому при использовании термина *Unit of Work* правильно было бы использовать его перевод как «единица поведения, единица событий».

IDEF3 отображает *UOW* на диаграммах в виде блока (рис. 42).

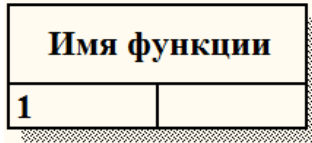


Рис. 42

Функции имеют имя, в составе которого, как правило, отглагольное существительное, обозначающее процесс и существительное, зависимое от отглагольного существительного, отображающее основной выход (результат) функции (например, «Изготовление изделия»). Номер функции присваивается при создании и не меняется никогда. Даже если блок будет удален, его номер не будет вновь использоваться. Обычно номер функции состоит из номера родительского блока и порядкового номера на текущей диаграмме.



Функция в *IDEF3* сопровождается подробным описанием ее компонентов: объектов, фактов, связанных с функцией, накладываемых ограничений и дополнительное описание функции.

Связи. Связи показывают взаимоотношения работ. Подобно *DFD*, *IDEF3* не имеет какой-либо регламентации функционального назначения стрелок-связей, кроме как «вход» и «выход». Связи выделяют существенные взаимоотношения между функциями. Все связи в *IDEF3* являются однонаправленными, и хотя стрелка может начинаться или заканчиваться на любой стороне блока, диаграммы *IDEF3* обычно организуются слева направо таким образом, что стрелки начинаются на правой и заканчиваются на левой стороне блоков. В табл. 1 приведены три возможных типа связей.

Таблица 1

Наименование	Описание	Графическое представление
1	2	3
Старшая стрелка или временное предшествование (<i>Precedence</i>)	Исходное действие должно полностью завершиться, прежде чем начнется выполнение конечного действия. Или: функция-источник должна закончиться прежде, чем функция-цель начнется. Рисуется слева направо или сверху вниз	

Окончание табл.1

1	2	3
Потоки объектов (<i>Object Flow</i>)	Применяется для описания того факта, что объект используется в двух или более <i>UOW</i> . Например, когда объект порождается в одной функции и используется в другой	
Стрелка отношения (<i>Relational Link</i>)	Показывает, что исходное действие не обязательно должно закончиться прежде, чем конечное действие начнется. Используется также между <i>UOW</i> и объектами ссылок	

Временная диаграмма выполнения работ при использовании старшей стрелки или потока объектов представлена на рис. 43, при использовании стрелки отношения – рис. 44.

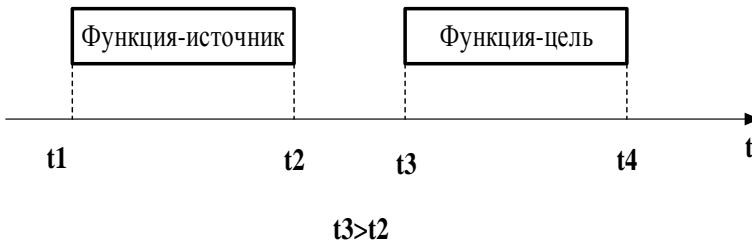


Рис. 43

Топологические ограничения отсутствуют, кроме одного: разветвление и слияние всех связей *IDEF3* осуществляется только через перекрестки (см. ниже). Введение параллельных связей допускается.

Связь должна быть поименована таким образом, чтобы человеку, просматривающему модель, была понятна причина ее появления.

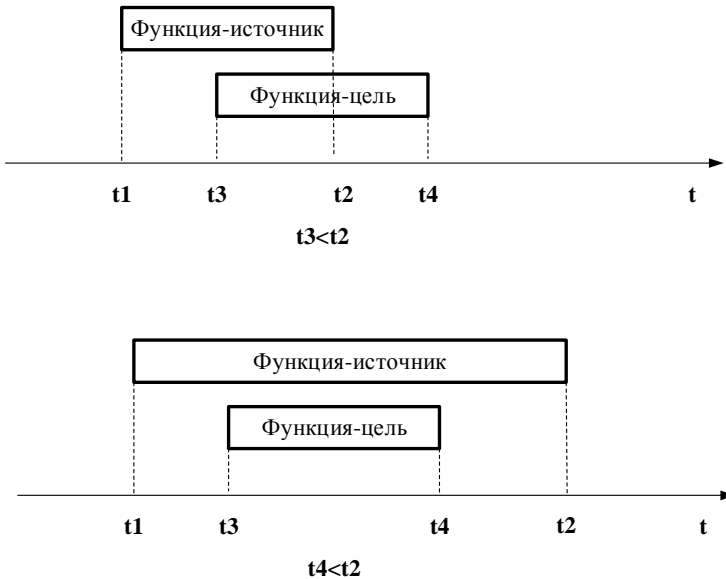


Рис. 44

Перекрестки. Окончание одной функции может служить сигналом к началу нескольких функций, или же одна функция для своего запуска может ожидать окончания нескольких функций. Перекрестки (*Junction*), используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей функции. Различают перекрестки для слияния (*Fan-in Junction*) и разветвления (*Fan-out Junction*) стрелок. Перекресток не может использоваться одновременно для слияния и разветвления.

Описание каждого типа перекрестка приведено в табл. 2.

Все перекрестки нумеруются, каждый номер имеет префикс *J*. Рассмотрим применение перекрестков каждого типа. На рис. 45 перекрестки для слияния и разветвления типа синхронного «И». Здесь после завершения Функции 1 одновременно запускаются Функции 2 и 4. Для запуска Функции 5 требуется одновременное завершение функций 3 и 4.

Таблица 2

Графическое представление	Наименование	Описание	
		В случае слияния стрелок	В случае разветвления стрелок
	Асинхронное «И»	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Синхронное «И»	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
	Асинхронное «ИЛИ»	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Синхронное «ИЛИ»	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
	Исключающее «ИЛИ»	Только один предшествующий процесс завершен	Только один следующий процесс запускается

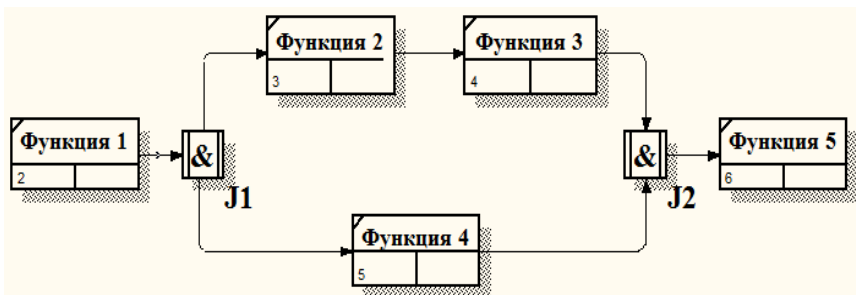


Рис. 45

На рис. 46 перекрестки для слияния и разветвления типа асинхронного «И». Здесь после завершения Функции 1 запускаются Функции 2 и 4 (не обязательно одновременно). Для запуска Функции 5 требуется завершение Функций 3 и 4 (не обязательно одновременное).

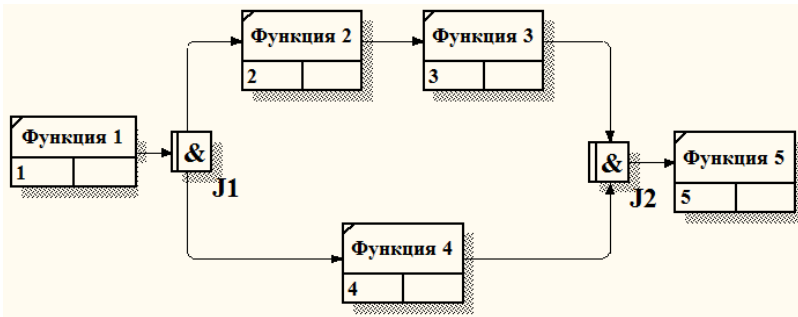


Рис. 46

На рис. 47 перекрестки для слияния и разветвления типа синхронного «ИЛИ». Здесь после завершения Функции 1 запускается либо Функция 2, либо Функция 3, либо Функция 4, либо их сочетание. Если запускается более одной функции, требуется их одновременный запуск. Для запуска Функции 5 требуется завершение любой из Функций 2, 3 и 4 или их сочетания. Если завершается более чем одна функция, требуется их одновременное завершение

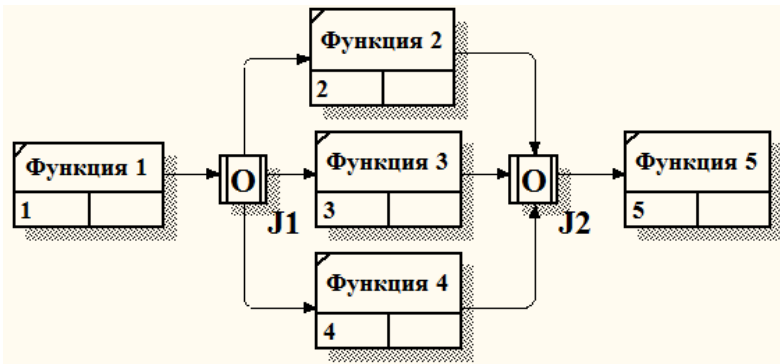


Рис. 47

На рис. 48 перекрестки для слияния и разветвления типа асинхронного «ИЛИ». Здесь после завершения Функции 1 запускается либо Функция 2, либо Функция 3, либо Функция 4, либо их сочетание (не обязательно одновременно). Для запуска Функции 5 требуется завершение любой из Функций 2, 3 и 4 или их сочетания (не обязательно одновременное).

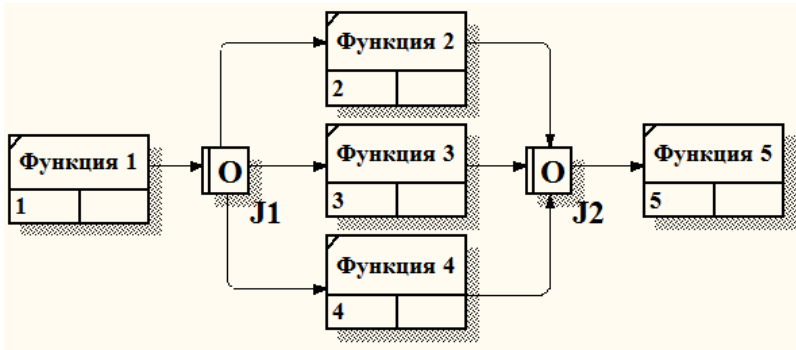


Рис. 48

На рис. 49 перекрестки для слияния и разветвления типа исключающего «ИЛИ». Здесь после завершения Функции 1 запускается только одна функция – либо Функция 3, либо Функция 4. Для запуска Функции 5 требуется завершение только одной из Функций – 3 или 4.

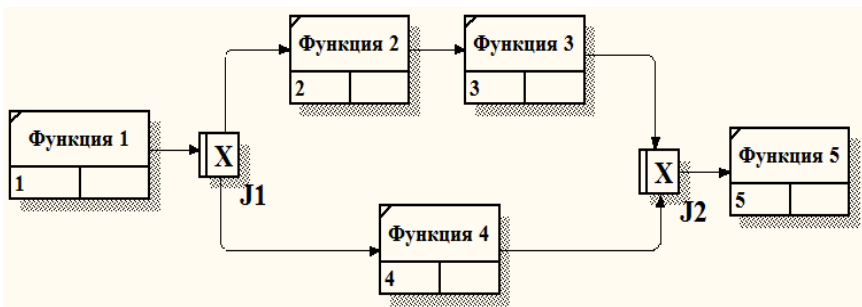


Рис. 49

Рис. 50 иллюстрирует более сложную комбинацию соединения. Но комбинации перекрестков следует использовать с осторожностью, поскольку перегруженные ветвлением диаграммы могут оказаться сложными для восприятия.

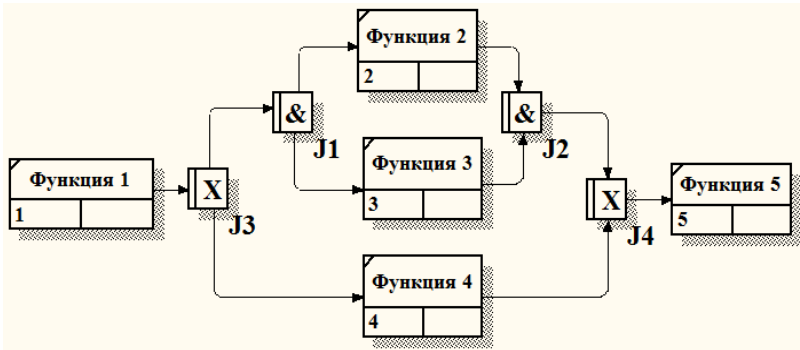


Рис. 50

Правила создания перекрестков. Определенные сочетания перекрестков могут приводить к логическим несоответствиям. Чтобы избежать конфликтов, необходимо соблюдать следующие правила:

1. Каждому перекрестку для слияния должен предшествовать перекресток для разветвления (обратное не обязательно).

2. Перекресток для слияния «И» не может следовать за перекрестком для разветвления типа синхронного или асинхронного «ИЛИ» (рис. 51). Действительно, после Функции 1 может запускаться только одна Функция – 2 или 3, а для запуска Функции 4 требуется окончание обеих Функций – 2 и 3. Такой сценарий не может реализоваться.

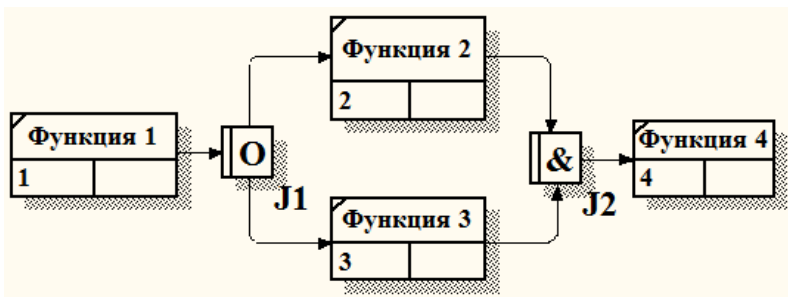


Рис. 51

3. Перекресток для слияния «И» не может следовать за перекрестком для разветвления типа исключающего «ИЛИ» (рис. 52)

4. Перекресток для слияния типа исключающего «ИЛИ» не может следовать за перекрестком для разветвления типа «И» (рис. 53). Здесь после завершения Функции 1 запускаются обе Функции – 2 и 3, а для запуска Функции 4 требуется, чтобы завершилась одна и только одна Функция – или 2, или 3.

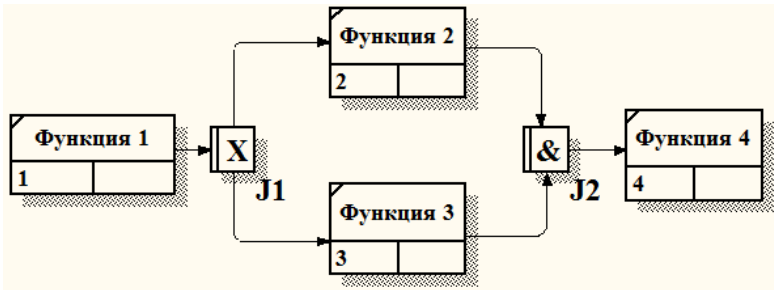


Рис. 52

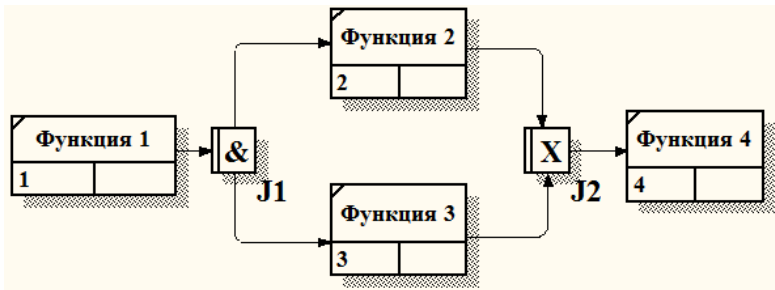


Рис. 53

5. Перекресток, имеющий одну стрелку на одной стороне, должен иметь более одной стрелки на другой (рис. 54).

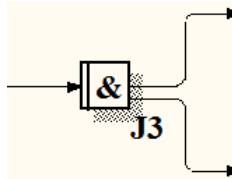


Рис. 54

Объект ссылки. Объект ссылки в *IDEF3* выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или функцией. Объект ссылки изображается в виде прямоугольника, похожего на блок функции (рис. 55).

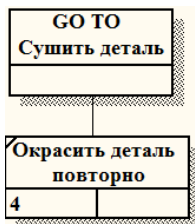


Рис. 55

Объекты ссылки должны быть связаны с *UOW* или перекрестками. Внутри блока объекта ссылки следует указывать его тип. Типы объектов ссылок приведены в табл. 3.

Таблица 3

Тип объекта ссылки	Описание
<i>OBJECT</i>	Описывает участие важного объекта в функции
<i>GOTO</i>	Инструмент циклического перехода (в повторяющейся последовательности функций), возможно на текущей диаграмме, но не обязательно. Если все функции цикла присутствуют на текущей диаграмме, цикл может также изображаться стрелкой, <i>возвращающейся</i> на запуск. <i>GOTO</i> может ссылаться на перекресток
<i>UOB</i> (<i>Unit of behavior</i>)	Применяется, когда необходимо подчеркнуть множественное использование какой-либо функции, но без цикла. Например, функция «Контроль качества» может быть использована в процессе «Изготовление изделия» несколько раз, после каждой единичной операции
<i>NOTE</i>	Применяется для документирования важной информации, относящейся к каким-либо графическим объектам на диаграмме. <i>NOTE</i> является альтернативой внесению текстового объекта в диаграмму
<i>ELAB</i> (<i>Elaboration</i>)	Употребляется для детального описания разветвления и слияния стрелок на перекрестках

Официальная спецификация *IDEF3* различает три стиля объектов ссылок – безусловные, синхронные и асинхронные. Синхронные и асинхронные объекты ссылок, используются в диаграммах переходов состояний объектов *OSTD* и здесь не рассматриваются.

Диаграммы. Методология *IDEF3* предусматривает два метода (и две нотации) для описания процессов. Первый предназначен для документирования и описания функций процесса «С точки зрения наблюдателя», второй – для описания процесса «С точки зрения объекта». Соответственно диаграммы, относящиеся к первому типу, называются диаграммами *описания последовательности этапов процесса (Process Flow Description, PFD)*, а ко второму – диаграммами *описания трансформаций состояния объекта (Object State Transition Description, OSTD)*. Проиллюстрируем это примером: предположим, требуется описать процесс окраски детали. В целом, этот процесс состоит из самой окраски и этапа контроля ее качества, который определяет, нужно ли деталь окрасить заново (в случае несоответствия стандартам и выявления брака) или отправить ее в дальнейшую обработку. Диаграмма *PFD* описывающая этапы обработки детали представлена на рис. 56.

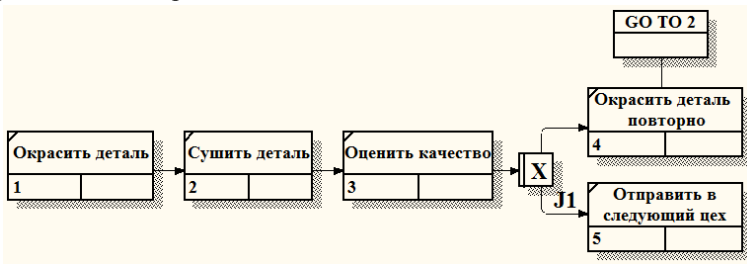


Рис. 56

Диаграмма *OSTD* (рис. 57) создается с целью описания трансформации детали, происходящей на каждой стадии обработки.

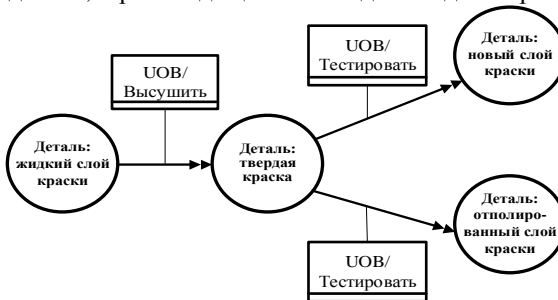


Рис. 57

Состояния объекта и *Изменение состояния* являются ключевыми понятиями *OSTD*-диаграммы. Состояния объекта отображаются окружностями, переходы между состояниями – стрелками. Каждая стрелка имеет ссылку на блок *UOB*, указывающий, в результате чего произошло изменение состояния объекта.

Надо сказать, что набор элементов нотации для диаграмм *PFD* практически совпадает с набором элементов нотации для диаграмм *OSTD*, отличаясь синтаксисом. Не только состояния объектов, как было показано выше, отображаются окружностями, но и перекрестки (рис. 58).

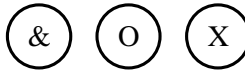


Рис. 58

IDEF3 обычно называют методикой потокового моделирования (безотносительно к особенностям объекта моделирования – материального или информационного), а инструментом графического представления функций – *Work Flow Diagrams (WFD)* – диаграммы потоков событий и здесь надо понимать, что речь идет о диаграммах *PFD*. Важным обстоятельством является то, что только функциональная модель *IDEF3*, состоящая из *WFD*-диаграмм, является основой для имитационного моделирования.

В качестве еще одного примера диаграммы потока событий рассмотрим модель процесса «Принять решение о начале движения на перекрестке» (рис. 59).

Контекстная диаграмма. В отличие от контекстных диаграмм *DFD*-моделей и *IDEF0*-моделей, контекстная *IDEF3*-диаграмма может содержать более одного функционального блока: методология не накладывает ограничений. Точка зрения на модель должна быть задокументирована. Обычно это точка зрения человека, ответственного за работу в целом. Также необходимо задокументировать цель модели – те вопросы, на которые призвана ответить модель.

Декомпозиция. Как и все другие методологии моделирования, *IDEF3* позволяет создавать иерархические функциональные модели, осуществляя декомпозиции ее *UOW*. Особенностью *IDEF3* является то, что функции могут быть декомпозированы многократно, т. е. функция может иметь множество дочерних функций. Это позволяет документировать альтернативные потоки процесса в одной модели.

Декомпозиция может быть *описанием* или *сценарием*. Описание включает все возможные пути развития процесса. Сценарий является

частным случаем описания и иллюстрирует только один путь реализации процесса.

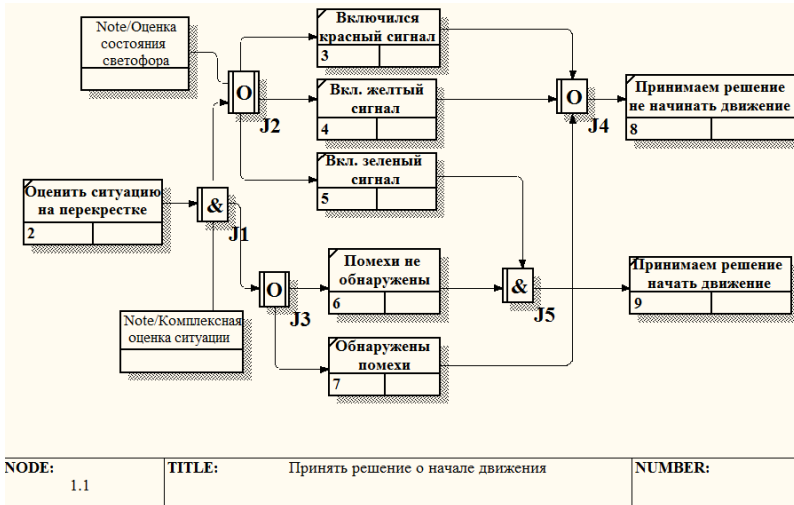


Рис. 59

Такая декомпозиция подобна диаграммам *FEO* (правила работы со сценарием те же, что и с *FEO*).

Для корректной идентификации в модели с множественными декомпозициями схема нумерации *UOW* расширяется и наряду с номерами блока и его родителя включает в себя порядковый номер декомпозиции (рис. 60). Например, в номере 3.1.5: 3 — номер родительского блока, 1 — номер декомпозиции, 5 — номер *UOW*.

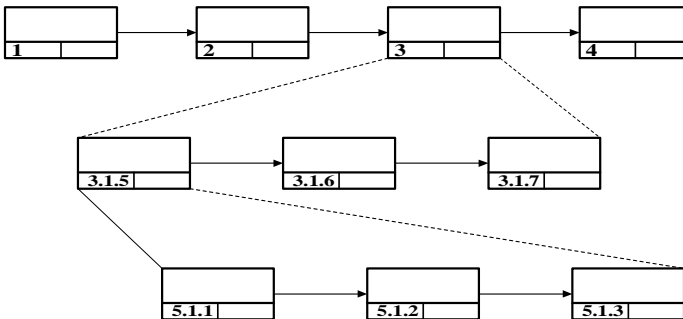


Рис. 60

4. Гибридные функциональные модели

Под гибридными подразумеваем функциональные модели, в которых на разных уровнях декомпозиции, процессы системы описаны с помощью разных методологий. Правила использования взаимосвязанного использования нескольких нотаций в пределах одной функциональной модели зависят от выбранного системным аналитиком инструмента для поддержки моделирования и продиктованы особенностями перехода от моделирования в одной методике к моделированию в другой. Изложенный ниже материал предполагает использование программного продукта *CA ERwin Process Modeler* и речь пойдет о совмещении нотаций *IDEF0*, *IDEF3* и *DFD*.

CA ERwin Process Modeler допускает следующие переходы от одной нотации к другой:

- *IDEF0* → *DFD*;
- *IDEF0* → *IDEF3*;
- *DFD* → *IDEF3*.

Нельзя декомпонировать *DFD*-функцию с помощью нотации *IDEF0* и декомпонировать *IDEF3*-функцию с помощью нотации *IDEF0* или *DFD*.

Обычно логика гибридных моделей такова: верхний уровень функциональной модели является структурным, материально-информационным, формализованным в правилах *IDEF0*. Далее, только в его функциональных границах, с помощью *DFD* и/или *IDEF3* детализируется структура, сформированная в *IDEF0*. В результате такого дополнения может быть создана гибридная (или смешанная) модель, которая наилучшим образом описывает все стороны моделируемой системы.

Декомпозиция функции *IDEF0* с помощью нотации *DFD*. Хотя нотация *DFD* не включает понятия «управление» и «механизм» и можно создавать внутренние стрелки исходящими из любой грани блока и входящими в любую грань, но стрелки управления и механизма родительского *IDEF0*-блока при миграции на дочернюю *DFD*-диаграмму, будут связаны только так, как если бы это была диаграмма *IDEF0*, т. е. входящими в верхнюю и нижнюю грани блока.

Согласно нотации *DFD*-диаграмма не должна иметь граничных стрелок – все стрелки должны начинаться и заканчиваться на функциях, хранилищах данных или внешних сущностях. Строго придерживаться этих правил при создании смешанных моделей не всегда удобно, поэтому *CA ERwin Process Modeler* позволяет создавать граничные стрелки на диаграммах *DFD* и не идентифицирует их как

синтаксическую ошибку. Но если строго следовать правилам, то следует заменить граничные стрелки на диаграмме *DFD* соответствующими внешними сущностями и хранилищами данных, а стрелки на диаграмме *IDEF0* затоннелировать.

Декомпозиция функции *IDEF0* и *DFD* с помощью нотации *IDEF3*. В отличие от *IDEF0* и *DFD*, *IDEF3*-диаграммы, интегрирующиеся в гибридные функциональные модели, не связаны с ними граничными стрелками, которые соответственно не мигрируют ни вверх, ни вниз по модели.

Пример. Рассмотрим процесс функционирования *TLS*-сервера (в контексте программного продукта). На рис. 61 представлена контекстная диаграмма процесса в нотации *IDEF0*. Предполагается, что сервер построен на базе криптографического пакета с открытым исходным кодом *OpenSSL* и предназначен для организации безопасного обмена данными с клиентским приложением по протоколу *TLS* (*Transport Layer Security*). Данный протокол относится к транспортному уровню эталонной модели *OSI* (*Open Systems Interconnection*).

Безопасность протокола *TLS* обеспечивается применением криптографических алгоритмов (шифров) двух типов: симметричных (одноключевых) и асимметричных (двухключевых, называемых также алгоритмами с открытым ключом). Первый тип используется для зашифрования/расшифрования передаваемых между клиентом и сервером данных, а второй – для согласования ключа симметричного шифра и прохождения процедуры аутентификации сторон. В асимметричных алгоритмах один из ключей общеизвестен и называется открытым, а второй известен только владельцу этой ключевой пары и называется закрытым.

Протокол позволяет осуществить одностороннюю (сервера) или двухстороннюю (и сервера, и клиента) аутентификацию участников протокола с использованием сертификатов открытого ключа стандарта X.509. Данный стандарт предполагает выпуск таких сертификатов некоей третьей доверенной стороной – центром сертификации (*ЦС*, *Certification Authority*) или удостоверяющим центром (в терминологии 63-ФЗ «Об электронной подписи»). Сертификат содержит информацию о владельце ключевой пары (например, доменное имя сервера), издатель (ЦС), сроке действия, используемых алгоритмах, а также, собственно, сам открытый ключ. ЦС размещает в составе сертификата созданную им электронную подпись, подтверждающую его подлинность и определяющую принадлежность открытого ключа указанному лицу.

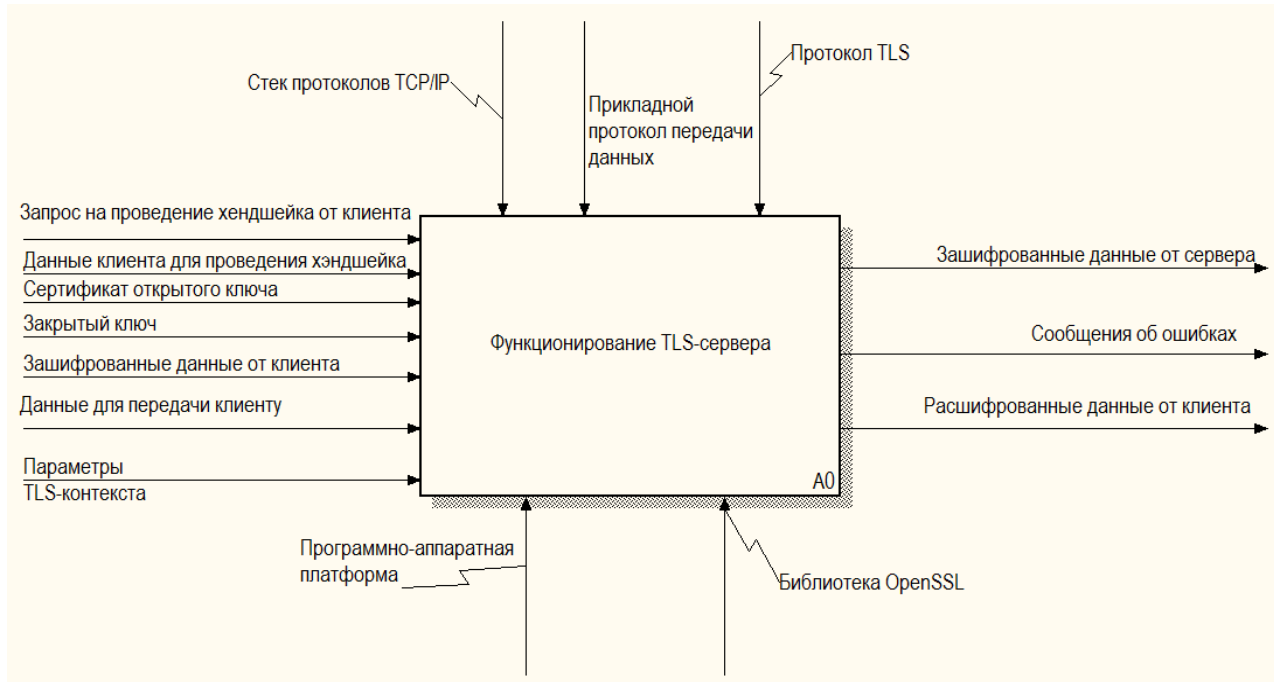


Рис. 61

Еще одним видом используемых в протоколе алгоритмов являются однонаправленные хеш-функции. Они осуществляют свертку сообщения произвольного размера в битовую строку фиксированной длины. В данном случае хеш-функции используются для построения *HMAC* (*hash-based message authentication code*) – кода, позволяющего определить целостность передаваемых данных. Для создания и верификации *HMAC* двум сторонам также необходимо согласовать общий ключ. Таким образом, клиенту и серверу для функционирования протокола требуется определенный объем согласованного *ключевого материала*, как основы для получения сеансовых ключей симметричных шифров (используемых только в одной конкретной сессии), векторов инициализации, ключей для создания и верификации *HMAC* и т.п.

Управление осуществляется с помощью стека протоколов *TCP/IP*, самого протокола *TLS*, а также протокола прикладного уровня. Стек протоколов *TCP/IP* используется потому, что сам протокол *TLS* является надстройкой над протоколом транспортного уровня *TCP*. Функционирование протокола *TLS* обязательно требует установления между клиентом и сервером обычного (незащищенного) *TCP*-соединения, для которого защищенный трафик рассматривается как обычные данные вышестоящего уровня модели *OSI*. Характер же и форматы передаваемых данных между клиентским и серверным приложением определяются протоколом прикладного уровня модели *OSI* и могут быть самыми разнообразными, решающими конкретные задачи взаимодействия компонентов информационной системы.

В качестве механизма, поддерживающего функционирование *TLS*-сервера, указана программно-аппаратная платформа, представляющая собой конкретный экземпляр вычислительной техники (например, компьютер, выполняющий роль аппаратного сервера) с установленной операционной системой, необходимым набором сервисных программ и комплектом системных библиотек подпрограмм, обеспечивающих функционирование описываемого программного продукта. Также в качестве механизма отдельно указана криптографическая библиотека *OpenSSL*, которая должна быть установлена на программно-аппаратной платформе и функции которой непосредственно вызываются сервером для реализации протокола *TLS*.

Входами являются запрос на проведение хендшейка от клиента и необходимые для этого данные. Протокол *TLS* на самом деле представляет собой двухуровневый стек протоколов, которые выполняют свои функции в процессе взаимодействия клиентского и серверных приложений. Самым сложным протоколом данного стека является

протокол установления соединения или хендшейка (*handshake*, «рукопожатие», квитирование), цель которого – выполнение процедур аутентификации сторон, согласование используемых алгоритмов, ключей и т.д. *TLS*-сервер ожидает входящие запросы от клиентских приложений на установление соединения и данные, которые они передают в процессе этого: используемую версию протокола *TLS*, используемые клиентом наборы алгоритмов (шифрсыюты), используемые клиентом алгоритмы сжатия, случайные данные, на базе которых будет вычисляться ключевой материал и т.п.

Как было сказано выше, одной из основных задач протокола установления *TLS*-соединения является аутентификация участников протокола. В данном случае рассматривается пример односторонней аутентификации (сервера), как наиболее часто встречающийся на практике. В качестве входных данных указаны сертификат открытого ключа *TLS*-сервера и соответствующий закрытый ключ. Сертификат в процессе проведения хендшейка отправляется клиенту, который проверяет его, делая вывод, к тому ли узлу в сети он обратился. Закрытый ключ используется в процессе согласования общего между клиентом и сервером ключевого материала.

Процесс функционирования протокола *TLS* предполагает защиту передаваемого трафика с помощью шифрования. Таким образом, на вход сервера поступают зашифрованные данные от клиента, которые после расшифрования и контроля целостности передаются на вышестоящие уровни модели *OSI*. Оттуда же приходят данные для передачи клиенту, которые проходят обратные процедуры и отправляются в сеть.

При построении сервера средствами криптографической библиотеки *OpenSSL*, основным объектом является *TLS*-контекст. В дальнейшем, в рамках существования одного контекста может быть создано множество *TLS*-соединений. После создания контекста, необходимо установить параметры его функционирования, которые в дальнейшем наследуются всеми создаваемыми *TLS*-соединениями. Эти параметры, к примеру, могут ограничивать используемые версии протокола или определять действия при блокировании транспортного сокета (сокеты – программный интерфейс для связи между процессами, по сути, комбинация *IP*-адреса и номера порта). Они также указаны в качестве входных данных (рис. 61).

В качестве выходов сервера на контекстной диаграмме (рис. 61) указаны: зашифрованные данные сервера, передаваемые в рамках функционирования протокола, расшифрованные данные от клиента, передаваемые протоколам вышестоящих уровней эталонной

модели *OSI*, а также сообщения клиенту об ошибках, которые могут случиться на различных стадиях работы протокола.

Теперь рассмотрим диаграмму декомпозиции *A0*, созданную также в нотации *IDEF0* (рис. 62). Для простоты изложения, рассмотрен частный случай, когда устанавливается только одно *TLS*-соединение с единственным клиентом, осуществляется обмен данными, после которого соединение разрывается, и освобождаются ресурсы, выделяемые для функционирования *TLS*-сервера.

Блок *A1* соответствует этапу создания и настройки *TLS*-контекста. На вход блока, кроме параметров контекста, поступают сертификат открытого ключа и закрытый ключ. Они должны быть установлены в контекст и в дальнейшем использоваться *TLS*-соединениями. Созданный контекст поступает на вход блока *A2*, который непосредственно определяет создание *TLS*-соединения. В отличие от предыдущего блока, здесь свою управляющую роль начинает выполнять стек протоколов *TCP/IP*, так как перед созданием *TLS*-соединения между клиентом и сервером, создается обычный *TCP*-сокет, «слушающий» определенный порт. После получения входящего запроса на установление *TCP*-соединения (рис. 62), оно устанавливается, затем создается объект *TLS*-соединения. Если в процессе установления *TCP*-соединения или создания объекта *TLS*-соединения происходит какая-либо ошибка, сервер выдает соответствующее сообщение.

Следом за успешным созданием соединения происходит непосредственно процедура хендшейка по протоколу *TLS*. На вход блока *A3* поступает созданный ранее объект *TLS*-соединения, запрос на проведение хендшейка от клиента и необходимые для этого данные. Результатом работы данного блока является либо установленное *TLS*-соединение, либо сообщение о возникшей ошибке.

В случае успешного хендшейка, установленное *TLS*-соединение передается на вход блока *A4*, который обозначает прием-передачу данных между сервером и клиентом. Также на вход этого блока передаются данные с верхних уровней модели *OSI* для передачи их клиенту. Формат передаваемых данных определяется протоколом прикладного уровня, используемым клиентом и сервером и осуществляющим управление процессом передачи информации. Используя согласованный на этапе хендшейка ключевой материал,

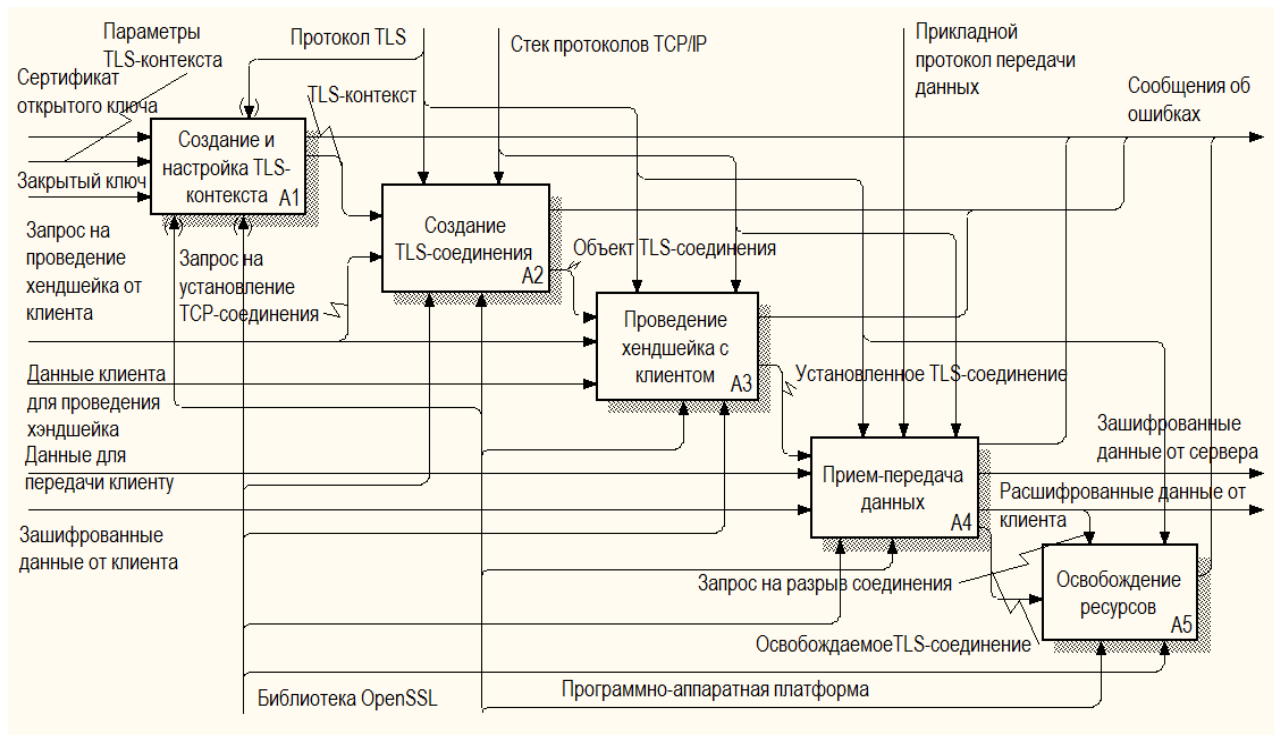


Рис. 62

осуществляется шифрование данных, создание *HMAC*, позволяющего определить клиенту, что полученная информация не была изменена в процессе пересылки, а также ее сжатие с помощью определенного на этапе хендшейка алгоритма. Полученные на этом этапе зашифрованные данные от клиента проходят обратную процедуру, в ходе которой они распаковываются, проверяется их целостность и осуществляется расшифрование. Расшифрованные данные передаются вышестоящим уровням модели *OSI*.

После того, как клиент закончил обмен информацией с сервером, он посылает запрос на разрыв соединения. Он передается в составе данных клиента и после его идентификации происходит запуск соответствующей процедуры, представленной блоком *A5*. Также как и установление соединения, это двухэтапная процедура, в которой сначала разрывается соединение по протоколу *TLS*, а затем – *TCP*. После разрыва соединений освобождаются все занимаемые ими локальные ресурсы. В случае возникновения ошибки, генерируется соответствующее сообщение.

Следующий уровень декомпозиции представлен с помощью методологии *DFD*. То есть в данном случае создается гибридная модель, использующая преимущества каждой из нотаций. В диаграмме *A0* будет декомпозировано два блока: *A1* и *A3*. Диаграмма декомпозиции блока *A1* представлена на рис. 63.

Если обратить внимание на блок *A1* на диаграмме *A0* (рис. 62), то можно увидеть, что стрелки *Протокол TLS*, *Библиотека OpenSSL*, *Программно-аппаратная платформа*, входящие в этот блок, являются туннелированными (взятыми в круглые скобки). Это означает, что на диаграмме декомпозиции этого блока данные стрелки были удалены и на родительской диаграмме *A0* они были определены как туннелированные. Это произошло потому, что эти стрелки должны подходить к каждому блоку, и, чтобы не загромождать ими диаграмму, их просто из нее удалили.

На диаграмме представлены три блока. В первом предполагается непосредственно создать контекст и установить его параметры, которые поступают на вход данного блока. Созданный контекст для полноценного функционирования требует установки закрытого ключа и сертификата открытого ключа. С точки зрения программы, контексту необходимо указать место их хранения, например, папку на локальном диске. На диаграмме для этой цели представлен накопитель данных *Хранилище ключевой информации*, в который

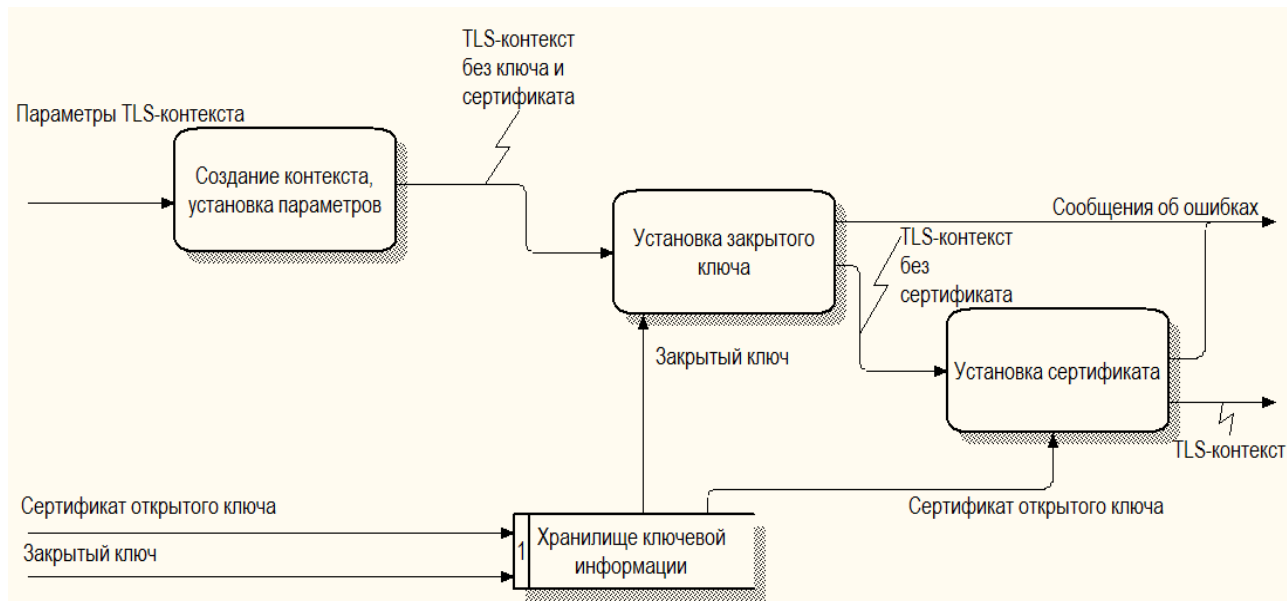


Рис. 63

предварительно загружаются сертификат и закрытый ключ, а затем они извлекаются для установки в контекст. Если процесс установки закрытого ключа или сертификата заканчивается неудачей, то выдается сообщение об ошибке и дальнейшие действия по установлению соединения между клиентом и сервером являются невозможными. В случае корректности этих операций, результатом является созданный *TLS*-контекст.

Еще одна диаграмма декомпозиции, раскрывающая содержимое блока АЗ, созданная в нотации *IDEF3*, представлена на рис. 64. Здесь укрупненно рассмотрено функционирование протокола хендшейка. Объект *TLS*-соединения проходит через ряд блоков и при успешном их завершении на выходе получается установленное *TLS*-соединение. При невыполнении некоторых условий хендшейк прекращается и генерируется соответствующее сообщение об ошибке.

В блоке с номером 1 поступающий на вход объект *TLS*-соединения переводится в режим ожидания хендшейка. После этого сервер ждет запроса на его проведение от клиента (блок с номером 2). После отправки серверу запроса, клиент также отправляет сообщение, в котором передает минимальную поддерживаемую им версию протокола *TLS* и сведения об используемых им шифрсьютах (блок номер 3).

После получения информации от клиента об используемых им версии протокола и шифрсьютах, сервер может соотнести их с используемыми им версиями и шифрсьютами и согласовать тот вариант, который и будет далее применяться при установлении соединения. Эти действия представлены в блоке с номером 4. Его выход поступает на вход перекрестка *J1* типа *Исключающее ИЛИ*. В нем принимается одно из трех взаимоисключающих решений. Одно из этих решений принимается тогда, когда версию протокола и шифрсьюты не удалось согласовать. Тогда выдается сообщение об ошибке и процесс установления соединения прекращается.

Если процесс согласования завершился успешно, то оставшиеся выходы перекрестка *J1* определяют, какой из вариантов аутентификации будет применяться в данном случае: односторонняя (блок 5) или двухсторонняя (блок 6). Их взаимоисключающие результаты объединяются перекрестком *J2*, и далее, по итогам аутентификации в блоке 7 принимается решение о продолжении хендшейка. Перекресток *J3* определяет результаты этого решения: либо выдается сообщение об ошибке, либо процесс продолжается в штатном режиме. В последнем случае, далее, осуществляется согласование с клиентом

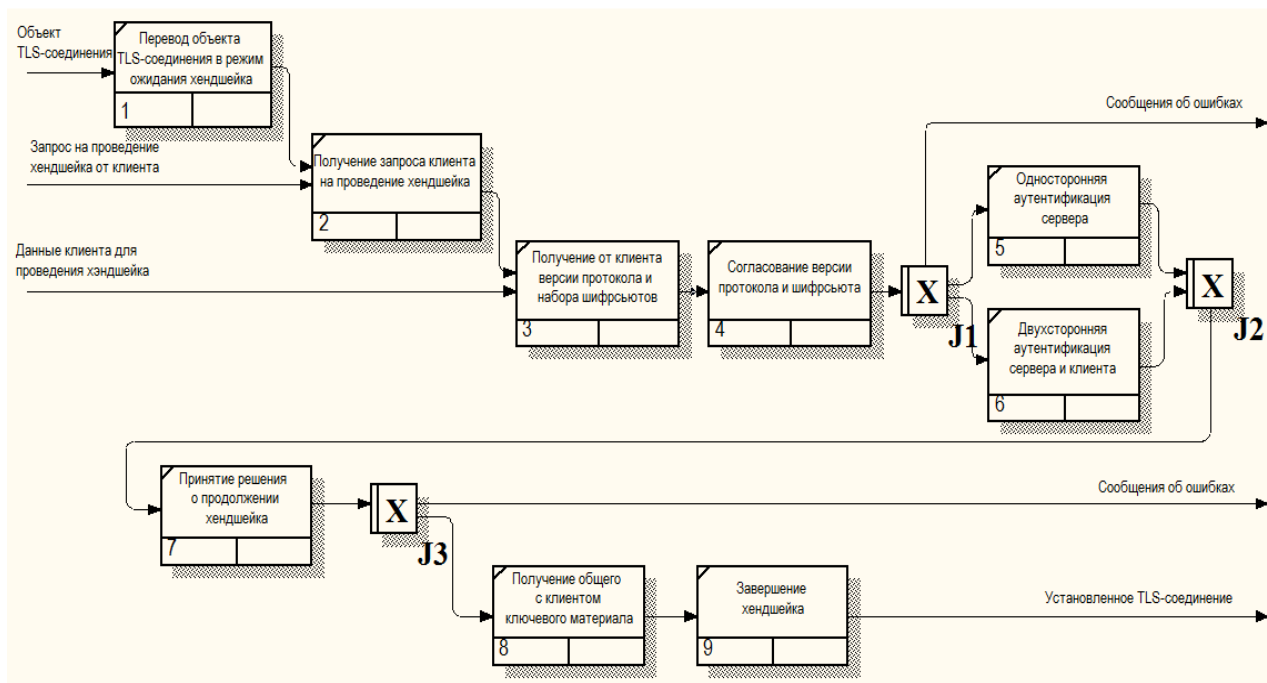


Рис. 64

общего ключевого материала (блок с номером 8) и инициируется завершение процедуры хендшейка (блок с номером 9). Результатом этой ветви является установленное *TLS*-соединение.

Безусловно, представленная здесь модель функционирования *TLS*-сервера является во многом упрощенной, но, тем не менее, отражающей основные черты данного процесса и демонстрирующей основные возможности используемых нотаций. При желании можно уточнить модель, декомпозируя ее блоки.

5. Функциональное моделирование и промышленное предприятие

Промышленное предприятие (ПП) любой отрасли производства представляет сложную многофункциональную систему. На рис. 65 приведена очень укрупненная модель функционирования ПП.

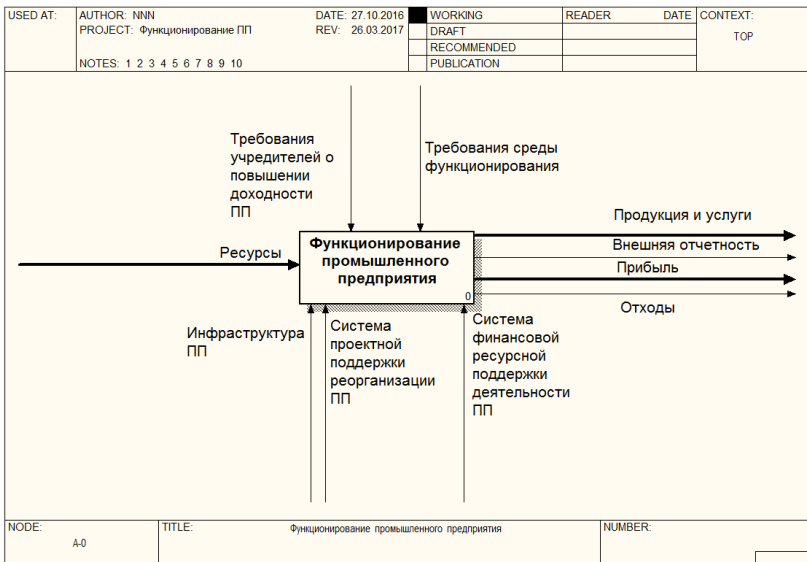


Рис. 65

Промышленное предприятие функционирует в условиях, определяющихся требованиями его учредителей и требованиями среды функционирования. Требования среды функционирования – это, в

основном, требования рынка (задача преодоления конкуренции), а также различные законодательные ограничения. Информационным выходом деятельности ПП является, разнообразная внешняя отчетность (перед налоговыми органами, перед Фондом социального страхования РФ, органами статистического наблюдения и др.). На основании данных бухгалтерской отчетности формируются выходные данные, служащие индикаторами финансового состояния ПП и являющиеся основой для активизации цикла периодической реорганизации. Предусмотренная законодательством РФ обязательная публикация годовой отчетности ПП обеспечивает контроль за его деятельностью со стороны акционеров, не входящих в состав органов управления ПП и позволяет привлечь к ПП внимание и доверие потенциальных инвесторов. Нормальное функционирование ПП сопровождается также образованием материальных и энергетических отходов. Строго говоря, также и информационных отходов – в виде потерявших по какой-либо причине актуальность данных. Важно подчеркнуть, что устойчивое, эффективное функционирование ПП складывается не только из выпуска той или другой продукции, но и из деятельности, направленной на адаптацию к изменяющимся условиям среды функционирования – реорганизации (рис. 66).

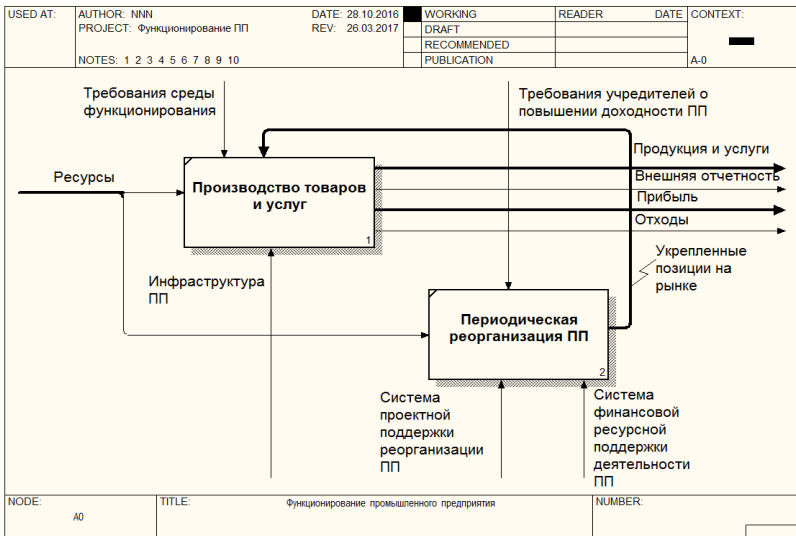


Рис. 66

Требования учредителей об увеличении доходности и капитализации предприятия приводят к периодической реорганизации ПП. Реорганизация позволяет обновлять производственно-технологические и финансовые характеристики, что обеспечивает способность ПП сохранять конкурентоспособность на рынке товаров и услуг.

5.1. Процессный подход к управлению

Управление является непременным атрибутом любого функционирования. Процессно-ориентированный подход к управлению позволяет рассматривать деятельность предприятия как систему взаимосвязанных и взаимодействующих бизнес-процессов, конечными целями выполнения которых является создание продуктов или услуг, представляющих ценность для потребителей. Управление на основе процессов позволяет точно знать, кто и за что отвечает и как каждая операция влияет на конечный результат. Процессный подход часто сравнивают (и противопоставляют, хотя это не оправдано) с функциональным подходом к управлению, согласно которому основной упор делается на результат работы подразделений предприятия, т.е. сотрудники не могут самостоятельно координировать работу в рамках процесса и решать возникающие проблемы без участия вышестоящего руководства. Тогда как процессный подход дает ряд преимуществ:

- повышение предсказуемости результатов;
- устранение барьеров между подразделениями;
- исключение неустраиваемых процессов;
- сокращение временных и материальных затрат;
- выявление возможностей для целенаправленного улучшения процессов.

Управляя процессами и постоянно их совершенствуя, предприятие добивается высокой эффективности своей деятельности. Именно поэтому, в 2000 году Международная организация по стандартизации (ISO) приняла новую версию стандартов серии 9000, содержащих перечень требований к системе качества предприятия на основе процессного подхода. В стандарте *ISO 9000:2000* используется следующее определение процесса: «Набор взаимосвязанных и взаимодействующих операций (действий), которые преобразуют входы в выходы». Входами процесса являются элементы, претерпевающие изменения в ходе выполнения действий: материалы, оборудование, документацию, различную информацию, персонал, финансы и пр. Выхо-

дами процесса являются ожидаемые результаты, ради которых предпринимаются действия. Выходом может быть как материальный продукт, так и различного рода услуги или информация. Процессный подход вводит понятие «владелец процесса». Это человек, отвечающий за конечный результат (выход) процесса. Для каждого процесса должны быть определены показатели, характеризующие его результаты.

На предприятии протекает множество процессов (или т.с. бизнес-процессов), которые можно разделить на три группы:

- основные, которые связаны непосредственно с производством продукции;
- обеспечивающие процессы осуществляют поддержку основных процессов (снабжение, управление персоналом и др.);
- управленческие процессы включают процессы по установлению целей и формированию условий для их достижения.

Процессы необходимо документировать, что позволяет их стандартизировать и получить базу для изменения или дальнейшего совершенствования и здесь возможны два подхода. Первый подход – традиционный: описание процессов в виде стандартов, методик, инструкций. Второй подход – моделирование: графическое изображение сети процессов с интегрированным первым подходом. Моделирование бизнес-процессов – основная часть процессного подхода и один из наиболее известных примеров использования ФМ.

5.2. Бизнес-моделирование

Бизнес-модель ПП должна отображать взаимодействие предприятия с окружающей средой (клиентами, партнерами и т.п.), показывать как организована деятельность на каждом рабочем месте, т.е. давать ответы на следующие вопросы:

1. Какие процедуры (функции, работы) необходимо выполнить для получения заданного конечного результата.
2. В какой последовательности выполняются эти процедуры.
3. Какие механизмы контроля существуют в рамках рассматриваемого бизнес-процесса.
4. Кто выполняет процедуры процесса.
5. Какие входящие документы/информацию использует каждая процедура процесса.
6. Какие исходящие документы/информацию генерирует процедура процесса.

7. Какие ресурсы необходимы для выполнения каждой процедуры процесса.

8. Какая документация/условия регламентирует выполнение процедуры.

9. Какие параметры характеризуют выполнение процедур и процесса в целом.

Моделирование осуществляется при помощи выбранных методологий ФМ и инструментальной среды, только в этом случае модель можно будет в дальнейшем анализировать и изменять.

Моделирование деятельности ПП является достаточно серьезной задачей. Необходимость в таком описании возникает всякий раз с внедрением информационных систем, систем менеджмента качества, систем управления безопасностью, новых технологий, т.е. с реорганизацией продиктованной теми или иными причинами.

Существует множество примеров, когда проекты внедрения корпоративных информационных систем заканчивались неудачей и именно этот негативный опыт заставляет на сегодняшний день активней использовать ФМ с целью минимизации рисков, возникающих на разных этапах жизненного цикла систем. Так разработчики программных средств, поддерживающих функционирование сложных производственных систем, так называемых *ERP*-систем (*Enterprise Resource Planning* – планирование ресурсов предприятия; группа специализированных программных средств поддержки деятельности производственных предприятий), позволяют себе в некоторых случаях поддержку своих программных комплексов специализированными средствами функционального моделирования. В качестве примера можно привести модуль *SAM* (*Strategic Application Modeler*) *ERP*-системы *iRenaissance* (разработчик – компания *ROSS Systems*). Использование этого модуля обеспечивает «гибкость» *ERP*-системы, позволяя выбрать наилучшую из альтернативных моделей производственных процессов. Обеспечивает графическое представление структуры организации, позволяет создать бизнес-модель предприятия, привязав к ней данные о стоимости и времени, о персонале и его должностных инструкциях. На протяжении всего срока эксплуатации системы, предприятие может использовать созданную в *SAM* модель для реорганизации бизнес-процессов, стратегического планирования, обучения и сертификации служащих, изменения управления, документирования процессов и т. д. В поддержку *SAM* в системе предусмотрен специализированный репозиторий знаний *ROK* (*Ross Online Knowledge*), содержащий более 1400 бизнес-процедур, накопленных в итоге работы с пользователями *ERP*-

системы и являющихся «строительным материалом» для формирования функциональных моделей при помощи SAM.

Функциональные бизнес-модели используются при создании систем менеджмента качества (СМК) в соответствии с международными стандартами *ISO 9000* и систем менеджмента информационной безопасности (СМИБ) в соответствии с международными стандартами *ISO 27001*. Следует отметить, что бизнес-процессы предприятий не являются статичными. Их высокая изменчивость обусловлена многими причинами, в частности, слиянием фирм, оптимизацией организационной структуры, внедрением автоматизированных технологий и т.п., что требует постоянного отражения происходящих изменений и в СМК, и в СМИБ. При этом изменения могут касаться всех элементов систем менеджмента: от концептуальных документов, инструкций и регламентов до конфигурации программно-технических решений. Построение и анализ бизнес-моделей дает возможность грамотно и своевременно реагировать на происходящие изменения.

Следует отметить, что бизнес-модели являются не просто промежуточным результатом, а имеют большое практическое значение:

1. Модели позволяют осуществлять обучение новых работников какому-либо направлению деятельности предприятия, так как модель содержит описание всех технологий в виде диаграмм.

2. С помощью модели можно осуществлять предварительное проектирование нового направления деятельности анализируя возможные потоки данных и объектов взаимодействующих подсистем.

5.2.1. Бизнес-консалтинг

Построение и анализ моделей деятельности ПП относится к области *бизнес-консалтинга*. Консалтинг – это деятельность, направленная на повышение эффективности компании, включающая в себя построение моделей предприятия, выработку предложений по совершенствованию его деятельности, формирование целевой программы развития предприятия и плана перехода из текущего состояния в целевое.

Для построения моделей проводится обследование и в качестве исходной информации служат:

- данные по оргштатной структуре предприятия;
- информация о принятых технологиях деятельности;
- стратегические цели и перспективы развития;
- результаты интервьюирования сотрудников (от руководителей до исполнителей нижнего звена) или анкетирования;

- предложения сотрудников по усовершенствованию бизнес-процессов предприятия;
- нормативно-справочная документация;
- данные по имеющимся на предприятии средствам и системам автоматизации;
- опыт системных аналитиков в части наличия типовых решений.

После обработки результатов обследования создаются функциональные, информационные и, по необходимости, событийные модели технологий работы предприятия следующих двух видов:

- модели текущего состояния (или другое название: модели «как есть», или *AS-IS*), представляющие собой «снимок» положения дел на предприятии на момент обследования и позволяющие понять, что делает и как функционирует данное предприятие с позиций системного анализа, а также выявление ряд ошибок и узких мест и формулировка ряда предложений по улучшению ситуации;
- модели целевого состояния (или другое название: «как должно быть», или *TO-BE*), интегрирующие перспективные предложения руководства и сотрудников предприятия, экспертов и системных аналитиков по совершенствованию деятельности предприятия.

Каждая из моделей деятельности должна включать:

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

В рамках создания моделей деятельности должен быть осуществлен:

- анализ функциональной деятельности структурных подразделений предприятия;
- анализ функционального взаимодействия структурных подразделений;
- анализ внутреннего документооборота структурных подразделений;
- анализ информационных потоков и информационного взаимодействия структурных подразделений;

– анализ применяемых в настоящее время средств автоматизации как в структурных подразделениях, так и на предприятии в целом.

По результатам анализа и моделирования осуществляется оценка эффективности деятельности структурных подразделений и предприятия в целом и формируется целевая программа развития предприятия и план мероприятий по переходу из текущего состояния в целевое. При этом переход от модели «как есть» к модели «как должно быть» обычно осуществляется с помощью технологии, называемой «*бизнес-реинжиниринг*».

6. Инструментальные средства функционального моделирования

Программное обеспечение, создаваемое для поддержки функционального моделирования относится, как правило, к разряду *CASE*-средств (*Computer Aided System Engineering* – проектирование систем с помощью компьютера). *CASE*-средства позволяют автоматизировать процесс создания моделей, отслеживая связи в диаграммах, организовать сопровождающее документирование, сохранять модели в виде файлов или в виде данных в специальной репозитории и генерировать отчеты на основе моделей.

Один из самых популярных продуктов этой категории на отечественном рынке – *BPWin*. И даже сегодня, когда в результате его доработки различными компаниями – *Logic Works*, *Platinum Technology*, *Computer Associates*, с дважды измененным названием – *AllFusion Process Modeler*, *CA ERwin Process Modeler*, многие продолжают называть его *BPWin*. Последняя версия программного продукта получила название *CA ERwin Process Modeler*. Данное средство моделирования поддерживает три методологии: *IDEF0*, *DFD*, *IDEF3* и возможность создания гибридных моделей. Изданные в России книги по этой тематике [7, 14] содержат подробные описания и набор упражнений, позволяющих самостоятельно освоить правила работы с программным продуктом.

Кроссплатформенная система моделирования и анализа бизнес-процессов *Ramus* (разработчик *Ramus Soft Group*, официальный русскоязычный сайт <http://ramussoftware.com>), поддерживает методологии *IDEF0* и *DFD*. Обеспечивается частичная совместимость *CA ERwin Process Modeler*. Некоммерческая версия *Ramus Educational*

ориентирована на использование студентами вузов, но имеет функциональные ограничения. Практикум, по работе с Ramus размещен на сайте открытого национального университета «Интуит» (<http://www.intuit.ru/studies/courses/2195/55/lecture/15043>).

Система моделирования *Business Studio* (разработчик ГК «Современные технологии управления», официальный сайт <http://www.businessstudio.ru>), поддерживает методологию *IDEF0* и ряд других, не рассмотренных в данном пособии (*Basic Flowchart*, *Cross-Functional Flowchart*, *BPMN 2.0*, *EPC*). Помимо ряда преимуществ, присущих всем *CASE*-средствам, *Business Studio* позволяет формировать техническое задание на создание информационной системы при работе над проектами по автоматизации бизнес-процессов: формировать реестр функций информационных систем, указывать функции информационных систем, поддерживающие выполнение бизнес-процессов.

Необходимо упомянуть еще один программный продукт, не являющийся *CASE*-средством – офисное приложение *Visio* компании *Microsoft*. Основными преимуществами *Visio* являются низкая стоимость и простота применения, но отсутствие возможности анализа и проверки корректности моделей, поддержки целостности и непротиворечивости данных делает этот инструмент непригодным для больших проектов.

Приведенный перечень инструментальных средств функционального моделирования далеко не полон и составлен исходя из возможности поддержки ими методологий, рассмотренных в данном учебном пособии и доступности их в учебном процессе. Более подробные сведения о рынке *CASE*-средств представлены в книге А.М. Вендрова «*CASE*-технологии. Современные методы и средства проектирования информационных систем». Также интересно ознакомиться с анализом средств бизнес-моделирования, приведенном на сайте российской компании «Бизнес Инжиниринг Групп» (<http://bigc.ru/instruments>). Компания является разработчиком методов и средств, предназначенных для повышения эффективности бизнеса.

Заключение

Сегодняшний бизнес не может существовать без информационных технологий. Известно, что около 70% мирового совокупного национального продукта зависят тем или иным образом от информа-

ции, хранящейся в информационных системах. Поэтому тема функционального моделирования важна и для студентов направлений «Программная инженерия», «Информатика и вычислительная техника» – как будущих разработчиков информационных систем, и для студентов специальности «Информационная безопасность автоматизированных систем» – как будущих специалистов по защите информации.

Любое промышленное предприятие (организация, фирма) представляет собой сложную динамическую систему. Адаптируясь к изменениям внешней среды, трансформируя свою структуру, предприятие повышает требования к качеству управления, обеспечить которое поможет информатизация бизнес-процессов. В свою очередь, чем выше информатизация, тем большее значение приобретают системы менеджмента информационной безопасности. А проектирование и внедрение информационных систем и СМИБ – задачи, решение которых, во-первых, связано, во-вторых, всегда начинается с анализа бизнес-процессов и активов предметной области. Под активом здесь понимается все, что представляет для предприятия ценность и подвергается рискам.

К уже перечисленным возможностям ФМ – ускорение и удешевление разработки систем, поддержка перестройки работы предприятий, поддержка анализа работы систем, можно еще добавить следующие:

- подготовка проектного управления (разработка планов работ). Инструментальная среда *CA ERwin Process Modeler* позволяет преобразовывать диаграммы процессов в диаграммы Ганта;

- совершенствование текстов на естественном языке за счёт конвертации их в формат функциональной модели;

- реконструктивное моделирование (воссоздание системы функций). Например, поддержка работы следователя, ведущего криминалистическое расследование;

- поддержка учебного процесса в различных отраслях знаний. Эффективность представления и усвоения знаний основывается на структурировании учебной информации и представлении её в графическом виде

- создание инструктивных материалов, в том числе – должностных инструкций. Основывается на детальной проработке функционирования системы в формате ФМ.

ПРИЛОЖЕНИЯ

Приложение 1

Примерный перечень контрольных вопросов к зачету

1. Что представляет собой *IDEF0*-модель?
2. Приведите примеры целей создания функциональной модели.
3. Раскройте содержание понятий «граница системы» и «точка зрения».
4. Перечислите основные компоненты графического языка *IDEF0*.
5. Каковы правила отображения функций в методологии *IDEF0*?
6. Каково назначение сторон блоков в методологии *IDEF0*?
7. Каковы правила отображения стрелок в методологии *IDEF0*?
8. Охарактеризуйте стрелки входа, выхода, управления, механизма.
9. Что такое «контекстная диаграмма»?
10. Что такое «функциональная декомпозиция»?
11. Что такое «родительская диаграмма», «дочерняя диаграмма»?
12. Как формируется ссылочный код функций?
13. Каково ограничение на количество функций на одной диаграмме?
14. Назовите виды отношений функций на диаграмме *IDEF0*?
15. Что называется порядком доминирования?
16. Как располагаются работы по принципу доминирования?
17. Перечислите типы стрелок на *IDEF0*-диаграмме декомпозиции.
18. Каково назначение тоннелированных стрелок?
19. Объясните принцип именования разветвляющихся и сливающихся стрелок.
20. Назначение *ICOM*-кодов.
21. Каково назначение диаграмм «Дерево узлов» и *FEO*?
22. Перечислите основные рекомендации по рисованию диаграмм.
23. Назначение текста и глоссария в *IDEF0*-моделях?
24. Как осуществляется процесс управления моделированием?
25. Перечислите основные компоненты графического языка *DFD*.

26. Каковы правила отображения функций в методологии *DFD*?
27. Каковы правила отображения потоков данных в методологии *DFD*?
28. Что моделируют внешние сущности и хранилища данных?
29. Правила декомпозиции в методологии *DFD*?
30. Особенности построения контекстной диаграммы в методологии *DFD*?
31. Перечислите правила нумерации блоков на диаграммах *DFD*.
32. Назначение словаря данных в *DFD*-модели.
33. Что такое «мини-спецификация» *DFD*-модели?
34. В чем особенности *DFD*-моделирования по сравнению с *IDEF0*?
35. В чем назначение методологии *IDEF3*?
36. Перечислите основные компоненты графического языка *IDEF3*.
37. Каковы правила отображения функций в методологии *IDEF3*?
38. Перечислите типы связей функций в методологии *IDEF3*?
39. Каковы правила отображения связей функций в методологии *IDEF3*?
40. Перечислите типы перекрестков.
41. Перечислите правила создания перекрестков.
42. В чем назначение объекта ссылки?
43. Что такое «*WFD*-диаграмма»?
44. Правила декомпозиции в методологии *IDEF3*.
45. Особенности построения контекстной диаграммы в методологии *IDEF3*.
46. Что такое «гибридная модель»?
47. Правила построения гибридной модели.
48. Что такое «процесс»?
49. В чем преимущество управления на основании процессного подхода?
50. Что такое «бизнес-моделирование»?
51. Что такое «бизнес-консалтинг»?
52. Что такое «модель текущего состояния» и «модель целевого состояния»?
53. Назначение *CASE*-средств.
54. Перечислите области применения функционального моделирования.

Список некоторых универсальных связей***CONTROL***

- различная документация;
- законодательные акты различных уровней;
- знания;
- разные нормативные документы;
- готовность оборудования к работе;
- требования среды функционирования (законодательные, экономические, этические, эстетические и др. ограничения);
- ожидания учредителей, акционеров, инвесторов, работников ПП;
- ожидания клиентов;
- требования рынка товаров и услуг;
- сроки выполнения работ;
- методики;
- технические задания;
- задания на проектирование;
- деньги и другие ограниченные ресурсы;
- нормативы;
- стандарты;
- планы работ;
- различные схемы как ограничения;
- правила управления процессами.

INPUT

- эксплуатационные материалы;
- ресурсы, поглощаемые или перерабатываемые: материальные, вые, финансовые, энергетические, информационные;
- параметры индикаторов финансового состояния предприятия;
- данные о доступных ресурсах;
- информация из всех доступных источников;
- инвестиции;
- инновации;
- бизнес-план;
- технические задания;
- расходуемые ресурсы;

- предметы, обрабатываемые в процессе функционирования;
- чертежи, технологическая информация;
- данные из различных источников.

OUTPUT

- отходы всех видов, в том числе энергетические и информационные;
- готовые детали;
- финансовые результаты деятельности ПП;
- внешняя отчетность ПП;
- продукция;
- информация выходная: научная, техническая, общественно-политическая, экономическая и др.;
- бизнес-план как результат его разработки;
- преобразованные данные, в том числе результаты контроля;
- отчеты о результатах различной деятельности;
- результаты деятельности по координации управлений;
- различные обработанные предметы;
- стоимостные характеристики работы системы;
- разработанные управляющие программы для различного оборудования;
- различные проекты;
- различные файлы и др.;
- люди как объекты политического, экономического, социального воздействия на них.

MECHANISM

- люди в качестве управляющих машинами, менеджеры, руководители всех ветвей власти и др.;
- различное программное обеспечение;
- технологическое оборудование;
- вычислительная техника;
- различные библиотеки;
- бизнес-партнеры;
- Интернет-ресурсы;
- CASE-средства разработки программного обеспечения;
- в целом инфраструктура организации, предприятия.

Библиографический список

1. *Андреева Н. В.* Функциональное моделирование системы управления информационной безопасности организации по семейству стандартов *ISO/IEC 2700X*: [Электронный ресурс] / Н. В. Андреева, С. В. Шустиков // Научно-технический вестник информационных технологий, механики и оптики / Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики. – 2008. – № 52. – С. 251-257. – Режим доступа: https://elibrary.ru/publisher_titles.asp?publishid=800
2. *Верников Г. Г.* Стандарт *IDEF3*: [Электронный ресурс] // Сайт ВЕРНИКОВ 7 тонн менеджмента / Г. Г. Верников. – 2009. – Режим доступа: <http://vernikov.ru/biznes-modelirovanie/tehnologii-i-standarty/item/32--idef3.html>
3. *Дорофеев А. В.* Менеджмент информационной безопасности: основные концепции: [Электронный ресурс] / А. В. Дорофеев, А. С. Марков // Вопросы кибербезопасности. – 2014. – Режим доступа: <https://cyberleninka.ru/article/n/menedzhment-informatsionnoy-bezopasnosti-osnovnye-kontseptsii>
4. *Дубейковский В. И.* Практика функционального моделирования с *AllFusion Process Modeler 4.1*. Где? Зачем? Как? / В. И. Дубейковский. – М.: ДИАЛОГ-МИФИ, 2004 .
5. *Калянов Г. Н.* Консалтинг при автоматизации предприятий / Г. Н. Калянов. – М.: Синтег, 1997.
6. *Калашян А. Н.* Структурные модели бизнеса: *DFD*-технологии; под ред. Калянова Г. Н. / А. Н. Калашян, Г. Н. Калянов. – М.: Финансы и статистика, 2003.
7. *Маклаков С. В.* *BPWin, ERWin. CASE*-средства разработки информационных систем / С. В. Маклаков. М.: ДИАЛОГ-МИФИ, 1999.
8. *Марка Д. А.* *SADT* – методология структурного анализа и проектирования / Д. А. Марка, К. МакГоуэн. – М.: Метатехнология, 1993.
9. *Пестунова Т. М.* Анализ аспектов информационной безопасности на основе формальных моделей бизнес-процессов: [Электронный ресурс] / Т. М. Пестунова, З. В. Родионова, С. Д. Горинова // Доклады Томского государственного университета систем управления и радиоэлектроники. – 2014. – № 2(32). – С. 150-156. – Режим доступа: <http://old.tusur.ru/filearchive/reports-magazine/2014-32-2/29.pdf>
10. *Репин В. В.* Процессный подход к управлению. Моделирование бизнес-процессов / В. В. Репин, В. Г. Елиферов. – М.: РИА «Стандарт и качество», 2013.

11. *Свечников А.* Модернизация системы пропускного режима предприятия: [Электронный ресурс] / А. Свечников // Сайт ВЕРНИКОВ 7 тонн менеджмента. – 2010. – Режим доступа: <http://vernikov.ru/biznes-processy/item/373-modernizacija-sistemy-propusknogo-rezhima-predprijatija.html>
12. *Федотова Д.Э.* CASE-технологии: практикум.// Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик. – М.: Горячая линия – Телеком, 2005.
13. *Черемных С. В.* Структурный анализ систем. *IDEF*-технологии / С. В. Черемных, В. С. Ручкин, И. О. Семенов. – М.: Финансы и статистика, 2001.
14. *Черемных С. В.* Моделирование и анализ систем. *IDEF*-технологии: практикум / С. В. Черемных, И. О. Семенов, В. С. Ручкин. – М.: Финансы и статистика, 2006.
15. Методология функционального моделирования *IDEF0*. Руководящий документ [Электронный ресурс]. – Режим доступа: <http://www.nsu.ru/smk/files/idef.pdf>

Оглавление

Введение	3
1. Методология <i>IDEF0</i>	6
1.1. <i>IDEF0</i> -модель	7
1.2. Синтаксис и семантика графического языка <i>IDEF0</i>	8
1.2.1. Стрелки на диаграмме декомпозиции	20
1.2.2. <i>ICOM</i> -коды	22
1.3. Диаграммы «Дерево узлов» и <i>FEO</i>	24
1.4. Рекомендации по рисованию диаграмм	25
1.5. Текст и глоссарий	27
1.6. Цикл «Автор-Читатель»	27
2. Методология <i>DFD</i>	29
2.1. Синтаксис и семантика графического языка <i>DFD</i>	29
2.2. Сравнительный анализ <i>DFD</i> и <i>IDEF0</i>	40
3. Методология <i>IDEF3</i>	41
3.1. Синтаксис и семантика графического языка <i>IDEF3</i>	42
4. Гибридные функциональные модели	55
5. Функциональное моделирование и промышленное предприятие	66
5.1. Процессный подход к управлению	68
5.2. Бизнес-моделирование	69
5.2.1. Бизнес-консалтинг	71
6. Инструментальные средства функционального моделирования	73
Заключение	74
ПРИЛОЖЕНИЯ	76
Библиографический список	81