НОУ ВПО «ТОЛЬЯТТИНСКАЯ АКАДЕМИЯ УПРАВЛЕНИЯ»

А.Р. Диязитдинова, А.В. Иващенко, М.В. Фролова

МЕТОДОЛОГИИ ПРОЕКТИРОВАНИЯ СИСТЕМ ОРГАНИЗАЦИОННОГО УПРАВЛЕНИЯ

Учебное пособие

Самара – Тольятти
2015

УДК 007.51
ББК 32.965

Рецензенты:
Погорелова Е.В. – начальник отдела по научным исследованиям и инновациям ФГБОУ ВО «Самарский государственный экономический университет», д.эконом.н., доцент.
Прохоров С.А. – зав. кафедрой информационных систем и технологий Самарского государственного аэрокосмического университета им. академика С.П. Королева (национальный исследовательский университет), д.т.н., профессор.

Диязитдинова А.Р.
Методологии проектирования систем организационного управления: учебное пособие /
А.Р. Диязитдинова, А.В. Иващенко, М.В. Фролова. – Тольятти: Тольяттинская академия управления, 2015.– 84 с.

ISBN 978-5-8146-0057-8

В учебном пособии раскрываются теоретические аспекты управления бизнес-процессами, рассматриваются наиболее распространенные методологии моделирования бизнес-процессов (таких как IDEF0, DFD, ARIS, UML и пр.), содержатся рекомендации по решению практических задач проектирования систем организационного управления с использованием современных информационных технологий.

Также в учебное пособие включен лабораторный практикум и тестирующий комплекс для самостоятельной оценки студентом степени проработанности изученного материала.

Учебное пособие предназначено для преподавания дисциплин профессионального цикла студентам вузов очной и заочной форм обучения по направлениям подготовки бакалавриата 080100 «Экономика», 080200 «Менеджмент», 080500 «Бизнес-информатика», а также студентам других компьютерных специальностей.

УДК 007.51
ББК 32.965

ISBN 978-5-8146-0057-8

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Создание систем эффективного управления организациями самого разного характера и сферы деятельности – одна из проблем, стоящих перед современным динамичным менеджментом. Универсального алгоритма для создания таких систем управления не существует, однако возможна разработка общих принципов построения систем управления бизнесом. В число наиболее передовых методов построения систем эффективного управления входит так называемый процессный подход к управлению, который заключается в выделении в организации сети процессов и управлении этими процессами для достижения максимальной эффективности деятельности организации.

В международном стандарте ISO 9000:2000 принят термин **«процесс»**, однако в современном менеджменте произошло постепенное сближение понятий «процесс» и «бизнес-процесс», поэтому оба термина здесь будут использоваться как синонимы.

Бизнес-процесс – это логичный, последовательный, взаимосвязанный набор мероприятий, который потребляет ресурсы, создает ценность и выдает результат.

Процесс – это устойчивая, целенаправленная совокупность взаимосвязанных видов деятельности, которая по определенной технологии преобразует входы в выходы, представляющие ценность для потребителя.

Как видно из определений, эти понятия действительно весьма близки. Процесс включает одну или более связанных между собой процедур или функций, которые совместно реализуют некую задачу бизнеса – обычно в рамках организационной структуры. Он может выполняться в пределах одной организационной единицы, охватывать несколько единиц или даже несколько различных организаций, например, в системе «покупатель – поставщик».

Целью моделирования является систематизация знаний о компании и ее бизнес-процессах в наглядной графической форме, более удобной для аналитической обработки полученной информации. Модель должна отражать структуру бизнес-процессов организации, детали их выполнения и последовательность документооборота.

Главное достоинство идеи анализа бизнес-процессов предприятия посредством создания его модели – ее *универсальность*:

1) Во-первых, моделирование бизнес-процессов – это ответ практически на все вопросы, касающиеся совершенствования деятельности предприятия и повышения его конкурентоспособности.
2) Во-вторых, руководитель или руководство предприятия, внедрившие у себя конкретную методологию, будет иметь информацию, которая позволит самостоятельно совершенствовать свое предприятие и прогнозировать его будущее.

В настоящее время на рынке компьютерных технологий представлено множество специальных программ, позволяющих обследовать предприятие и построить модель. Выбор методологии и инструментов, с помощью которых проводится моделирование бизнес-процессов, основополагающего значения не имеет. Существуют стандартизированные, опробованные временем методологии и инструментальные средства, с помощью которых можно обследовать предприятие и построить его модель. Ключевое их преимущество – простота и доступность овладения.

В данном пособии рассмотрены наиболее часто используемые методологии моделирования бизнес-процессов и проектирования систем организационного управления.

1. ОСНОВНЫЕ ПОНЯТИЯ

Профессиональный подход к задачам проектирования и совершенствования производственных, промышленно-торговых и других социально-экономических систем является одной из важнейших предпосылок их успешного функционирования. В процессе проектирования совершенствуется как организация основной деятельности экономического объекта (производственной, хозяйственной), так и организация управленческих процедур. Именно качественное проектирование обеспечивает создание такой системы, которая способна функционировать при постоянном совершенствовании ее технических, программных, информационных составляющих, т.е. ее технологической основы, и расширять спектр реализуемых управленческих функций и объектов взаимодействия.

Само понятие «моделирование бизнес-процессов» пришло в жизнь большинства аналитиков одновременно с появлением на рынке сложных программных продуктов, предназначенных для комплексной автоматизации управления предприятием. Подобные системы всегда подразумевают проведение глубокого предпроектного обследования деятельности компании. Результатом этого обследования является экспертное заключение, в котором отдельными пунктами выносятся рекомендации по устранению «узких мест» в управлении деятельностью. На основании этого заключения непосредственно перед проектом внедрения системы автоматизации, проводится так называемая реорганизация бизнес-процессов, иногда достаточно серьезная и болезненная для компании. Подобные комплексные обследования предприятий всегда являются сложными и существенно отличающимися от случая к случаю задачами. Для решения подобных задач моделирования сложных систем существуют хорошо обкатанные методологии и стандарты.

Решение по моделированию бизнес-процессов обычно принимается по следующим причинам:

- существенный рост бизнеса компании за счет расширения направлений деятельности (и, как следствие, экстенсивный рост затрат);
- исчерпание экстенсивного пути развития компании;
- потеря «технологической прозрачности» деятельности компании;
- позиция руководства компании, осознавшего необходимость изменений и развития и осознающего перспективу и стратегию развития.

Моделирование бизнес-процессов затрагивает многие аспекты деятельности компании:

- изменение организационной структуры;
- оптимизацию функций подразделений и сотрудников;
- перераспределение прав и обязанностей руководителей;
- изменение внутренних нормативных документов и технологии проведения операций;
- новые требования к автоматизации выполняемых процессов и т.д.

Существует несколько подходов к определению понятия «моделирование бизнес-процессов»:

1. Моделирование бизнес-процессов – это описание бизнес-процессов предприятия, позволяющее руководителю знать, как работают рядовые сотрудники, а рядовым сотрудникам – как работают их коллеги и на какой конечный результат направлена вся их деятельность.
2. Моделирование бизнес-процессов – это эффективное средство поиска путей оптимизации деятельности компании, позволяющее определить, как компания работает в целом и как организована деятельность на каждом рабочем месте.
3. Моделирование бизнес-процессов – это средство, позволяющее предвидеть и минимизировать риски, возникающие на различных этапах реорганизации деятельности предприятия.
4. Моделирование бизнес-процессов – это метод, позволяющий дать оценку текущей деятельности предприятия по отношению к требованиям, предъявляемым к его функционированию, управлению, эффективности, конечным результатам деятельности и степени удовлетворенности клиента.
5. Моделирование бизнес-процессов – это метод, позволяющий дать стоимостную оценку каждому процессу, взятому в отдельности, и всем бизнес-процессам на предприятии, взятым в совокупности.
6. Моделирование бизнес-процессов – это всегда верный способ выявления текущих проблем на предприятии и предвидения будущих.

Моделирование бизнес-процессов позволяет проанализировать не только, как работает предприятие в целом, как оно взаимодействует с внешними организациями, заказчиками и поставщиками, но и как организована деятельность на каждом отдельно взятом рабочем месте. Таким образом, системный подход к проектированию и разработке автоматизированных систем управления организацией заключается, в основном, в моделировании и всестороннем анализе требований к этой системе.

В данном учебном пособии под моделированием будет пониматься процесс создания достаточно точного и адекватного графического описания системы, а также интерпретация полученного описания для определения оценочных значений ее некоторых характеристик [13].

Модель (model) – это искусственный объект, представляющий собой отображение (образ) системы и ее компонентов. Модель может описывать существующие (AS-IS), идеализированные (SHOULD-BE) и вновь создаваемые (TO-BE) процессы и функции. На разных стадиях разработки используются различные уровни детализации модели.

Процесс бизнес-моделирования может быть реализован в рамках различных методик. В настоящее время для описания бизнес-процессов существует несколько основных нотаций, получивших наибольшее распространение, и соответствующее им программное обеспечение, что позволяет моделировать бизнес-процессы любой сложности. Вопросы техники описания и моделирования бизнес-процессов при помощи различных программных продуктов и стандартов (IDEF0, IDEF3, ARIS, UML и пр.) хорошо изложены в методической литературе [например, 3, 10, 11, 14, 20].

Говорить о преимуществе той или иной методики нецелесообразно, пока не определены тип и рамки проекта, требований к информации, необходимой для анализа и принятия решений в рамках конкретного проекта, а также основные задачи, которые данный проект должен решить. Как правило, проводить бизнес-моделирование ради самого моделирования – занятие малоэффективное. Моделирование бизнес-процесса всегда должно преследовать некоторую конкретную цель (например, документирование для описания процессов в виде регламентов, регулирующих деятельность компании, или анализа и реорганизации процессов для повышения эффективности управления процессами и т.п.). В соответствии с системным подходом осуществляется детализация данных целей на конкретные задачи со своим набором требований и знаний по бизнес-процессу. От задачи к задаче требования к описанию бизнес-процессов могут меняться.

Необходимо понимать, что никакая единственная модель не может с достаточной степенью адекватности описывать различные аспекты сложной системы.

Описание бизнес-процесса формируется при помощи методик (нотаций) и программных продуктов, позволяющих отразить все указанные выше аспекты. Только в этом случае модель бизнес-процесса окажется полезной для организации. Ниже кратко рассмотрены следующие методики:

1) семейство стандартов обследования организаций и проектирования информационных систем IDEF, основанное на технологии структурного анализа и проектирования SADT (Structured Analysis and Design Technique);
2) DFD (Data Flow Diagrams), диаграммы потоков данных совместно со словарями данных и спецификациями процессов или миниспецификациями;
3) UML (Unified Modeling Language, унифицированный язык моделирования);
4) ARIS (Architecture of Integrated Information Systems, архитектура интегрированных информационных систем) доктора Шеера (IDS Scheer AG);
5) BPMN (Business Process Modeling Notation), стандарт «Нотация моделирования бизнес-процессов».

2. СТАНДАРТЫ IDEF. МЕТОДОЛОГИЯ SADT. IDEF0

В США в конце 70-х годов XX века была предложена и реализована Программа интегрированной компьютеризации производства ICAM (Integrated Computer Aided Manufacturing), направленная на увеличение эффективности промышленных предприятий посредством широкого внедрения компьютерных (информационных) технологий.

В процессе практической реализации, участники программы ICAM столкнулись с необходимостью разработки новых методов анализа процессов взаимодействия в промышленных системах. При этом, кроме усовершенствованного набора функций для описания бизнес-процессов, одним из требований к новому стандарту было требование обеспечить групповую работу над созданием модели с непосредственным участием всех аналитиков и специалистов, занятых в рамках проекта. Для удовлетворения этой потребности в рамках программы ICAM была разработана методология моделирования IDEF, позволяющая исследовать структуру, параметры и характеристики производственно-технических и организационных систем.

В настоящее время к семейству IDEF относят следующие стандарты:

1) IDEF0 – Integration Definition for Function Modeling – методология функционального моделирования. С помощью наглядного графического языка IDEF0, изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций. Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы;
2) IDEF1 – Information Modeling – методология моделирования информационных потоков внутри системы, позволяющая отображать и анализировать их структуру и взаимосвязи;
3) IDEF1X (IDEF1 Extended) – Data Modeling – методология построения реляционных структур. IDEF1X относится к типу методологий «Сущность-взаимосвязь» (ER – Entity-Relationship) и, как правило, используется для моделирования реляционных баз данных, имеющих отношение к рассматриваемой системе;
4) IDEF2 – Simulation Modeling – методология динамического моделирования развития систем. В связи с весьма серьезными сложностями анализа динамических систем от этого стандарта практически отказались, и его развитие приостановилось на самом начальном этапе. Однако в настоящее время присутствуют алгоритмы и их компьютерные реализации, позволяющие превращать набор статических диаграмм IDEF0 в динамические модели, построенные на базе «раскрашенных сетей Петри» (CPN – Color Petri Nets);
5) IDEF3 – Process Description Capture – методология документирования процессов, происходящих в системе, которая используется, например, при исследовании технологических процессов на предприятиях. С помощью IDEF3 описываются сценарий и последовательность операций для каждого процесса. IDEF3 имеет прямую взаимосвязь с методологией IDEF0 – каждая функция (функциональный блок) может быть представлена в виде отдельного процесса средствами IDEF3;
6) IDEF4 – Object-oriented Design – методология построения объектно-ориентированных систем. Средства IDEF4 позволяют наглядно отображать структуру объектов и заложенные принципы их взаимодействия, тем самым позволяя анализировать и оптимизировать сложные объектно-ориентированные системы;
7) IDEF5 – Ontology Description Capture – методология онтологического исследования сложных систем. С помощью методологии IDEF5 онтология системы может быть описана при помощи определенного словаря терминов и правил, на основании которых могут быть сформированы достоверные утверждения о состоянии рассматриваемой системы в некоторый момент времени. На основе этих утверждений формируются выводы о дальнейшем развитии системы, и производится ее оптимизация;
8) IDEF6 – Design Rationale Capture;
9) IDEF7 – Information System Audit Method;
10) IDEF8 – User Interface Modeling;
11) IDEF9 – Scenario-driven Info Sys Design Spec;
12) IDEF10 – Implementation Architecture Modeling;
13) IDEF11 – Information Artifact Modeling;
14) IDEF12 – Organization Modeling;
15) IDEF13 – Three Schema Mapping Design;
16) IDEF14 – Network Design.

Методология IDEF0 является, пожалуй, одной из наиболее полно освещенных в литературе [5, 11, 12]. В частности, существует ГОСТ Р 50.1.028-2001, посвященный методологии функционального моделирования.

В IDEF0 система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

Процесс моделирования системы в IDEF0 начинается с определения контекста, то есть наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система, при этом необходимо точно определить, что входит в систему, а что лежит за ее пределами. Однако моделируемая система никогда не существует изолированно: она всегда связана с окружающей средой. По этой причине в методологии SADT подчеркивается необходимость точного определения границ системы, т.е. модель устанавливает точно, что является и что не является субъектом моделирования, описывая то, что входит в систему, и подразумевая то, что лежит за ее пределами. SADT-модель должна иметь единственный субъект.

Точка зрения (viewpoint) – указание на должностное лицо, с позиции которого разрабатывается модель. У модели может быть только одна точка зрения. 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.

Формулировка цели (purpose) выражает причину создания модели, то есть содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет ее структуру. После того как определены субъект, цель и точка зрения модели, начинается первая интеграция процесса моделирования по методологии SADT. Субъект определяет, что включить в модель, а что исключить из нее. Точка зрения диктует автору модели выбор нужной информации о субъекте и форму ее представления. Цель становится критерием окончания моделирования. Конечным результатом этого процесса является набор тщательно взаимоувязанных описаний, начиная с описания самого верхнего уровня системы и заканчивая подробным описанием ее деталей или отдельных операций.

Область моделирования (scope) описывает круг функций системы. При формулировании области необходимо учитывать два компонента: широту и глубину. Широта подразумевает определение границ моделирования, а глубина определяет, на каком уровне детализации модель является завершенной.

Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм, разработанных с определенной целью и с выбранной точки зрения.

Каждая такая диаграмма содержит работы и стрелки. Работы или функциональные блоки (activity) обозначают поименованные процессы, функции или задачи, которые выполняются в течение определенного времени и имеют распознаваемые результаты. Поскольку IDEF0 представляет собой методологию функционального моделирования, то имя блока, описывающее функцию, должно быть глаголом или глагольным оборотом.

Взаимодействие работ с внешним миром и между собой описывается в виде стрелок. Стрелка (arrow) – направленная линия, состоящая из одного или нескольких сегментов, которая моделирует канал, передающий данные от источника (начальная точка стрелки) к потребителю (конечная точка с «наконечником»).

В IDEF0 различают пять типов стрелок (рис. 2.1):

- вход (input) – материал или информация, которые используются или преобразуются работой для получения результата. Допускается, что работа может не иметь ни одной стрелки входа. Входные стрелки рисуются как входящие в левую грань работы;
- выход (output) – материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Выходные стрелки рисуются как исходящие из правой грани работы;
- управление (control) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Стрелки управления рисуются как входящие в верхнюю грань работы;
- механизм (mechanism) – ресурсы, которые выполняют работу. Стрелки механизма рисуются как входящие в нижнюю грань работы;
- вызов (call) – специальная стрелка, указывающая на другую модель. Стрелка вызова рисуется как исходящая из нижней грани работы. 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, то есть разные модели или разные части одной и той же модели могут совместно использовать один и тот же элемент (блок).

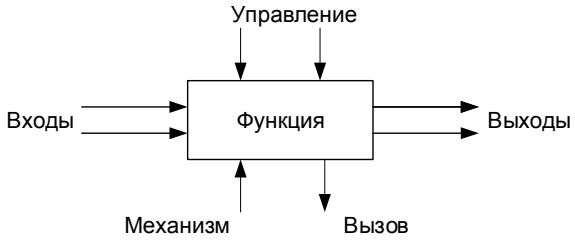


Рисунок 2.1. **Структура модели**

Все работы на диаграмме являются преобразующими, то есть преобразуют входы в выходы под действием управлений при помощи механизмов.

Стрелка обычно представляет набор объектов. Поэтому они могут разветвляться и соединяться различными способами. Сегменты стрелок, за исключением стрелок вызова, помечаются существительным или оборотом существительного.

Внутренние стрелки используются для связи работ между собой. В IDEF0 требуется только пять типов взаимосвязей между блоками с помощью стрелок для описания их отношений: а) управление, б) вход, в) обратная связь по управлению, г) обратная связь по входу, д) выход-механизм (рис. 2.2).

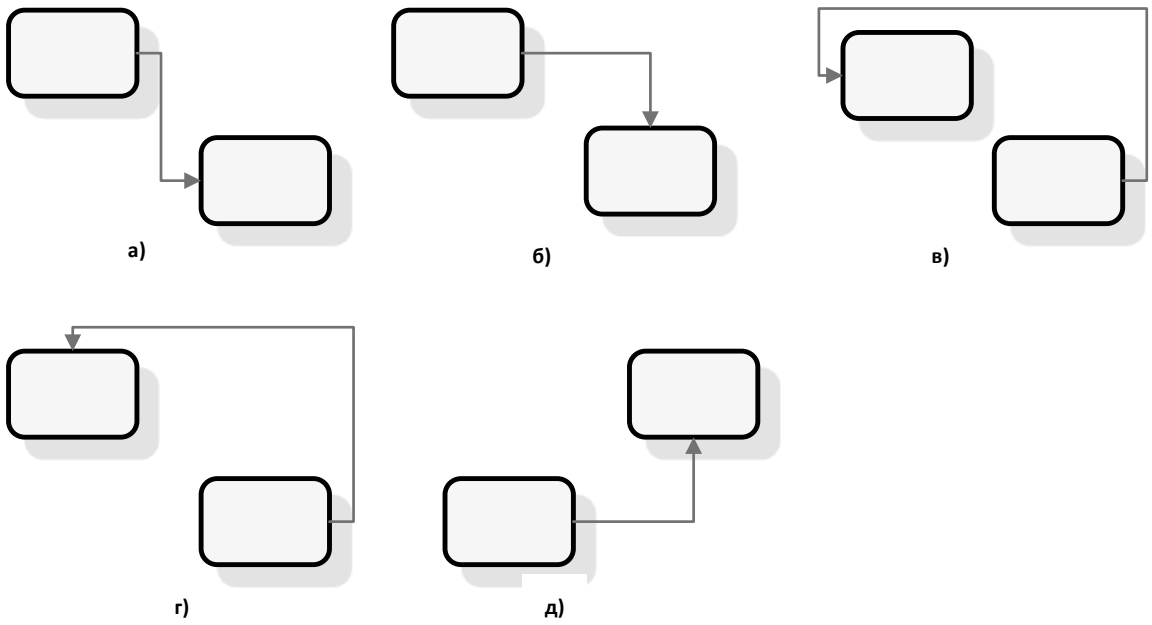


Рисунок 2.2. **Типы взаимосвязей между работами**

На каждой диаграмме отображается от трех до шести работ. Работы имеют доминирование – они размещаются на диаграмме по степени важности. Самой доминирующей работой диаграммы может быть либо первая из требуемой последовательности, либо планирующая или управляющая. Наиболее доминирующая работа располагается в левом верхнем углу диаграммы, наименее доминирующая – в правом нижнем. Работы на одной диаграмме нумеруются последовательно в порядке их доминирования.

Каждая работа (функциональный блок) может быть разбита на более мелкие работы, взаимосвязанные между собой. Этот процесс называется функциональной декомпозицией (decomposition), а диаграммы, которые описывают каждую работу (функциональный блок) и взаимодействие работ, называются диаграммами декомпозиции. Другими словами, модель SADT можно представить в виде древовидной структуры диаграмм, где верхняя диаграмма является наиболее общей, а самые нижние – максимально детализированы.

Каждый блок на диаграмме имеет свой номер. Вершиной древовидной структуры взаимосвязанных диаграмм является контекстная диаграмма, которая имеет шифр А-0. Контекстная диа-

грамма (context diagram) представляет собой самое общее описание системы и ее взаимодействия с внешней средой. Контекстная диаграмма содержит только один функциональный блок, соединенный с границами диаграммы при помощи граничных стрелок. Блок любой диаграммы может быть детализирован диаграммой нижнего уровня, которая, в свою очередь, также может детализироваться с помощью необходимого числа диаграмм. Таким образом, формируется иерархия диаграмм. Для того чтобы указать положение любой диаграммы или блока в иерархии, им присваивают уникальные обозначения. Например, **A41** (**A** сокр. от **Activity**) является диаграммой, которая детализирует блок **1** на диаграмме **A4**. Аналогично, **A4** детализирует блок **4** на диаграмме **A0**, которая является самой верхней (родительской) диаграммой модели.

Некоторые дуги имеют начало в одном из блоков диаграммы и завершение в другом, у других же начало может исходить от границ диаграммы – дуги управления, механизма, дуги входа и выхода, перенесенные с родительской (верхнего уровня) диаграммы. Таким образом, источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме.

Также следует упомянуть о так называемых «туннельных дугах». Туннельные дуги означают, что данные, выраженные этими дугами, не рассматриваются на следующем уровне детализации (как бы проходят «насквозь»). Если «туннель» расположен в месте соединения дуги с блоком «», то данные этой дуги необязательны на следующем уровне детализации. Если же «туннель» находится на противоположном конце дуги «  » – это значит, что данные дуги не описываются на родительской диаграмме. Граничные дуги должны продолжаться (дублироваться) на родительской диаграмме, делая ее полной и непротиворечивой (см. рис. 2.3).

Для упрощения понимания приведенных диаграмм следует расшифровать применяемую в IDEF систему обозначений, позволяющую аналитику точно идентифицировать и проверять по дугам связи между диаграммами. Эта схема кодирования дуг – **ICOM** – получила название по первым буквам английских эквивалентов слов вход (**Input**), управление (**Control**), выход (**Output**), механизм (**Mechanism**).

Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования. Наиболее важные свойства системы обычно выявляются на верхних уровнях иерархии и уточняются по мере декомпозиции.

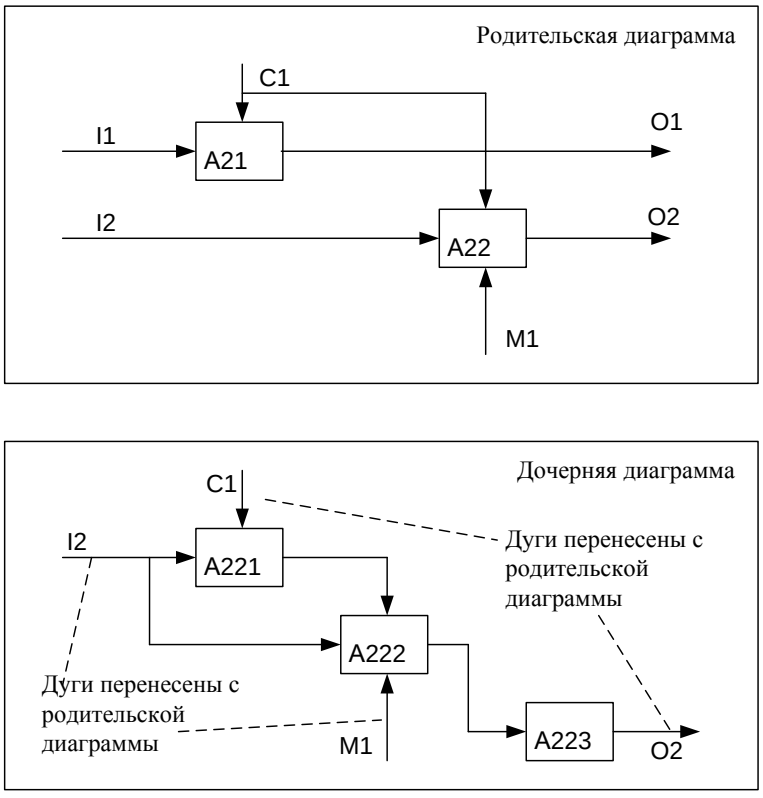


Рисунок 2.3. **Соответствие дуг родительской и дочерних диаграмм**

Если говорить о недостатках данного стандарта, то необходимо упомянуть о некоторой субъективности получаемых моделей: полученные модели напрямую зависят от компетенции разработчика. Впрочем, данный недостаток в той или иной мере присущ абсолютно всем методологиям моделирования бизнес-процессов. Например, поток «Подготовленное рабочее место» можно трактовать и как вход, и как механизм (рис. 2.4).

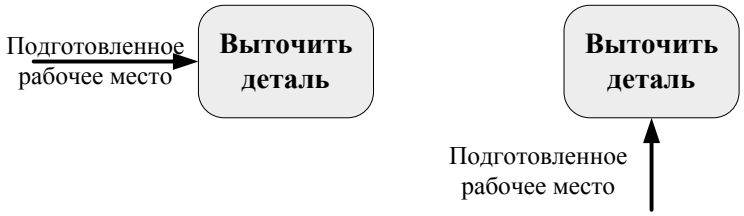


Рисунок 2.4. **Субъективность IDEF0-диаграмм**

Ценность модели (проекта) определяется ее приемлемостью для экспертов. Данная приемлемость достигается следующими путями:

- постоянным рецензированием экспертами развивающейся модели, что обеспечивает необходимый уровень соответствия (адекватности) модели существующему или предполагаемому моделируемому объекту в том понимании, которое соответствует мнению экспертов;
- периодическим обсуждением диаграмм, частей модели в целом на техническом совете, решение которого (оформленное в виде протокола) позволяет автору продолжить уточняющее моделирование или закончить его ввиду достаточности детализации и приемлемости проекта (модели).

3. ДИАГРАММЫ ПОТОКОВ ДАННЫХ DFD

Диаграммы потоков данных (Data Flow Diagrams, DFD) являются основным средством моделирования функциональных требований проектируемой системы. С их помощью эти требования разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами. Так же как и диаграммы IDEF0, диаграммы потоков данных моделируют систему как набор действий, соединенных друг с другом стрелками.

Диаграммы потоков данных (Data Flow Diagrams, DFD) описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией как внутри системы между бизнес-функциями, так и системы в целом с внешней информационной средой. Модель DFD представляет собой набор диаграмм, отражающих различные аспекты модели.

Работы в DFD представляют собой функции системы, преобразующие входную информацию в выходную в соответствии с действиями, задаваемыми именами работ. Каждая работа имеет уникальный номер для ссылок на него внутри диаграммы. Этот номер может использоваться совместно с номером диаграммы для получения уникального индекса работы во всей модели.

Внешние сущности (ссылки) указывают на место, организацию или человека, которые участвуют в процессе обмена информацией с системой, но располагаются за рамками данной модели. Внешняя сущность является источником или приемником данных извне модели. Внешние сущности обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах с целью повышения наглядности.

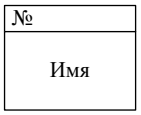
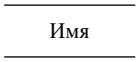
Потоки данных используются для моделирования передачи информации (или физических компонентов) из одной части системы в другую. Потоки изображаются на диаграмме именованными стрелками, ориентация которых указывает направление движения информации. Поскольку в DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, стрелки могут подходить и выходить из любой грани прямоугольника работы. В DFD также применяются двунаправленные стрелки для описания диалогов типа «команда-ответ» между работами, между работой и внешней сущностью и между внешними сущностями. В DFD стрелки могут сливаться и разветвляться, что позволяет описать декомпозицию стрелок. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя. В отличие от стрелок IDEF0, которые представляют собой жесткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) двигаются от одной работы к другой.

Хранилище данных – это место накопления информации внутри системы. Хранилище данных позволяет на определенных участках определять данные, которые будут сохраняться в памяти между работами. В отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое. Фактически хранилище представляет «срезы» потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее определения, при этом данные могут выбираться в любом порядке. Хранилище данных может содержать информацию длительного хранения или временную информацию. Имя хранилища должно идентифицировать его содержимое. На одной диаграмме может присутствовать несколько копий одного и того же хранилища данных.

Для изображения DFD традиционно используются две различные нотации: Йордана – Де Марко (Yourdon-DeMarco) и Гейна-Сарсона (Gane-Sarson, названа в честь Криса Гейна (Chris Gane) и Триша Сарсона (Trish Sarson)). Нотации аналогичны, за исключением форм объектов. Подробно ознакомиться с особенностями нотаций можно в [7, 11, 12].

Таблица 3.1. **Основные элементы DFD-диаграммы**

Название	**Определение**	**Нотация Yourdon**	**Нотация Gane-Sarson**
Внешняя сущность	Материальный предмет или физическое лицо, представляющее собой источник или приемник информации	Имя	Имя

Название	Определение	Нотация Yourdon	Нотация Gane-Sarson
Процесс	Преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом.		
Поток	Информация, передаваемая через некоторое соединение от источника к приемнику		
Хранилище	Абстрактное устройство для хранения информации (жесткий диск, база данных и т.п.), которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми		

Построение диаграмм DFD производится «сверху вниз» в соответствии с целью моделирования. Сначала строится диаграмма контекста, описывающая исследуемую систему целиком. Затем следует диаграммы первого уровня для наиболее важных бизнес-функций. При этом внешние сущности дублируются на диаграмме декомпозиции, а работы разбиваются. Дальнейшее уточнение может при необходимости производиться с помощью более подробных диаграмм на следующих уровнях.

Результатом моделирования является набор диаграмм, отражающих различные аспекты модели, которая объединяет в себе одну или несколько диаграмм потоков данных.

4. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ НА ЯЗЫКЕ UML

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем. Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения [10].

UML позволяет представить модели сложной системы в различных взаимосвязанных аспектах. Для иллюстрации использована модель «4+1» (The 4+1 View Model of Architecture), которая была предложена Филиппом Кручтеном (Philippe Kruchten) из компании Rational в 1995 году (см. рис. 4.1)

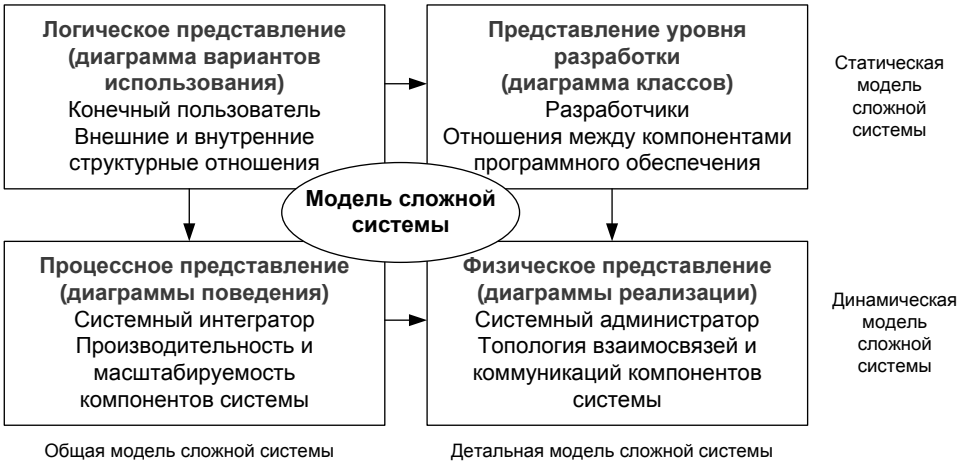


Рисунок 4.1. **Общая схема взаимосвязей моделей**

Модель предлагает простой и понятный способ описания архитектуры сложных систем, который состоит в использовании пяти различных категорий или представлений (см. табл. 4.1)

Таблица 4.1. Описание модели «4+1»

Представление	Краткое описание	Аудитория	Приоритетная область	Диаграмма UML
Логическое	Что система должна выполнять в терминах конечных пользователей	Конечный пользователь, системный аналитик	Внешние и внутренние структурные отношения	Диаграмма вариантов использования
Процесса функционирования	Учитывает некоторые нефункциональные требования к системе, включая производительность и доступность	Системный аналитик, архитектор системы	Производительность и масштабируемость компонентов системы	Диаграммы поведения
Разработки	Описывает фактическую организацию модулей системы, разделение ее на подсистемы, которые могут разрабатываться независимо	Программисты, архитектор системы	Программные компоненты	Диаграмма классов

Представление	Краткое описание	Аудитория	Приоритетная область	Диаграмма UML
Размещения компонентов	Рассматривает вопросы коммуникаций компонентов системы	Менеджеры по внедрению, системный администратор	Физические узлы	Диаграммы реализации
Сценарии (модель сложной системы)	Объединяют все представления вместе. Сценарии использования описываются как последовательность взаимодействия объектов и процессов. Они отражают наиболее важные требования, которым должна удовлетворять система			

4.1. Общая структура языка UML

Общая структура UML показана на рисунке 4.2.

С самой общей точки зрения описание языка UML состоит из двух взаимодействующих частей:

- Семантика и синтаксис языка UML. Представляет собой некоторую метамодель, которая определяет абстрактный синтаксис и семантику понятий объектного моделирования на языке UML.
- Нотация языка UML. Представляет собой графическую нотацию для визуального представления семантики языка UML.

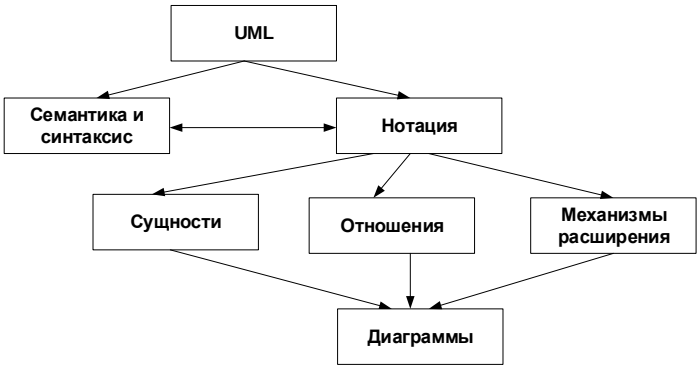


Рисунок 4.2. **Структура UML**

Абстрактный синтаксис и семантика языка UML описываются с использованием некоторого подмножества нотации UML. В дополнение к этому, нотация UML описывает соответствие или отображение графической нотации в базовые понятия семантики. Таким образом, с функциональной точки зрения эти две части дополняют друг друга. При этом семантика языка UML описывается на основе некоторой метамодели, имеющей три отдельных представления: абстрактный синтаксис, правила корректного построения выражений и семантику.

Семантика определяется для двух видов объектных моделей: структурных моделей и моделей поведения. Структурные модели, известные также как статические модели, описывают структуру сущностей или компонентов некоторой системы, включая их классы, интерфейсы, атрибуты и отношения. Модели поведения, называемые иногда динамическими моделями, описывают поведение или функционирование объектов системы, включая их методы, взаимодействие и сотрудничество между ними, а также процесс изменения состояний отдельных компонентов и системы в целом.

Для решения столь широкого диапазона задач моделирования разработана достаточно полная семантика для всех компонентов графической нотации. Нотация языка UML включает в себя описание отдельных семантических элементов, которые могут применяться при построении диаграмм (см. следующий раздел).

Механизмы расширения UML предназначены для того, чтобы разработчики могли адаптировать язык моделирования к своим конкретным нуждам, не меняя при этом его основу (метамодель). Наличие механизмов расширения принципиально отличает UML от других средств моделирования и позволяет расширять его область применения.

К механизмам расширения UML относятся: стереотипы; именованные значения; ограничения. Стереотип – это дополнительный тип элемента модели, который определяется на основе уже существующего элемента. Стереотипы расширяют нотацию модели. Любой элемент UML может быть расширен стереотипом. На диаграммах стереотип представляется в виде текстовой метки или пиктограммы. На диаграмме классам приписаны стереотипы: <<boundary>> (граничный класс), <<entity>> (класс-сущность) и <<control>> (управляющий класс). Тем самым элемент UML отображает не только класс, но и его ответственность, т.е. роль, которую он играет в системе.

4.2. Соглашение о моделировании (нотация)

Нотация представляет собой графическую интерпретацию семантики для ее визуального представления, т.е. нотация – это правила рисовании моделей. UML является графическим языком и, следовательно, его отличительной особенностью является то, что словарь языка образуют графические элементы. Каждому графическому символу соответствует конкретная семантика, поэтому модель, созданная одним разработчиком, может однозначно быть понята другим, а также программным средством, интерпретирующим UML.

В качестве основных графических элементов были выбраны:
- фигуры;
- линии;
- значки (пиктограммы);
- текст.

Фигуры в UML используются двумерные (т.е. их можно нарисовать на плоскости) и замкнутые (т.е. есть внутренняя и внешняя части). Фигуры могут менять свои размеры и форму, сохраняя при этом свои интуитивные отличительные признаки. Внутри фигур могут помещаться другие элементы нотации: тексты, линии, значки и даже другие фигуры. Единственное требование: должно быть однозначно понятно, что элемент нотации находится внутри фигуры, в частности, его изображение не должно пересекать границу фигуры.

Линии в UML одномерные, всегда присоединяются своими концами к фигурам или значкам, они не могут быть нарисованы сами по себе. Форма линий произвольна: это могут быть прямые, ломаные, плавные кривые. Толщина линий также произвольна. Стиль линии имеет значение. В UML используется только два стиля линий: сплошные и пунктирные линии. К линиям могут быть пририсованы различные дополнительные элементы: стрелки на концах, тексты и т.д. Линии могут пересекаться, но рекомендуется избегать таких случаев, поскольку это затрудняет восприятие.

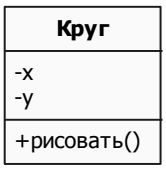
Значки (пиктограммы) в UML похожи на фигуры тем, что они двумерные, а отличаются тем, что не имеют внутренности, в которую можно что-то поместить, и, как правило, не меняют свою форму и размеры.

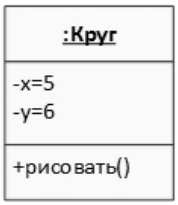
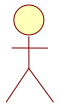
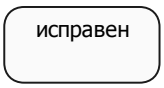
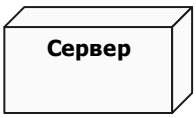
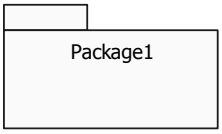
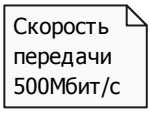
Текст в UML – это, как обычно, последовательности различимых символов некоторого алфавита. Алфавит не фиксирован – он только должен быть понятен читателю модели. Гарнитура, размер и цвет шрифта не имеют значения. Начертание шрифта имеет: в UML различаются прямые, курсивные и подчеркнутые тексты.

В UML определено три типа сущностей:
- структурная – абстракция, являющаяся отражением концептуального или физического объекта;
- группирующая – элемент, используемый для некоторого смыслового объединения элементов диаграммы;
- поясняющая (аннотационная) – комментарий к элементу диаграммы.

В таблице 4.2 приведено краткое описание сущностей, используемых в графической нотации.

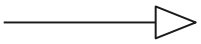
***Таблица 4.2.*Сущности UML**

Тип	Наименование	Обозначение	Определение (семантика)
Структурная	Класс (class)		Множество объектов, имеющих общую структуру и поведение

Тип	Наименование	Обозначение	Определение (семантика)
	Объект (object)		Абстракция реальной или воображаемой сущности с четко выраженными концептуальными границами, индивидуальностью (идентичностью), состоянием и поведением. С точки зрения UML, объекты являются экземплярами класса (экземплярами сущности)
	Интерфейс (interface)		Совокупность операций, определяющая сервис (набор услуг), предоставляемый классом или компонентом
	Актер (actor)		Внешняя по отношению к системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. Таким образом, актер – это внешний источник или приемник информации
	Вариант использования (use case)		Описание последовательности выполняемых системой действий, которая приводит к значимому для актера результату
	Состояние (state)		Описание момента в ходе жизни сущности, когда она удовлетворяет некоторому условию, выполняет некоторую деятельность или ждет наступления некоторого события
	Кооперация (collaboration)		Описание совокупности экземпляров актеров, объектов и их взаимодействия в процессе решения некоторой задачи
	Компонент (component)		Физическая часть системы (файл), в том числе модули системы, обеспечивающие реализацию согласованного набора интерфейсов
	Узел (node)		Физическая часть системы (компьютер, принтер и т.д.), предоставляющая ресурсы для решения задачи
Группирующая	Пакет (packages)		Общий механизм группировки элементов. В отличие от компонента, пакет – чисто концептуальное (абстрактное) понятие. Частными случаями пакета являются система и модель
Поясняющая	Примечание (comment)		Комментарий к элементу

В таблице 4.3. приведено описание всех видов отношений UML, используемых на диаграммах для указания связей между сущностями.

Таблица 4.3. **Отношения UML**

Наименование	**Обозначение**	**Определение (семантика)**
Ассоциация (association)		Отношение, описывающее значимую связь между двумя и более сущностями. Наиболее общий вид отношения
Агрегация (aggregation)		Подвид ассоциации, описывающей связь «часть» – «целое», в котором «часть» может существовать отдельно от «целого». Ромб указывается со стороны «целого». Отношение указывается только между сущностями одного типа
Композиция (composition)		Подвид агрегации, в которой «части» не могут существовать отдельно от «целого». Как правило, «части» создаются и уничтожаются одновременно с «целым»
Зависимость (dependency)		Отношение между двумя сущностями, в котором изменение в одной сущности (независимой) может влиять на состояние или поведение другой сущности (зависимой). Со стороны стрелки указывается независимая сущность
Обобщение (generalization)		Отношение между обобщенной сущностью (предком, родителем) и специализированной сущностью (потомком, дочкой). Треугольник указывается со стороны родителя. Отношение указывается только между сущностями одного типа
Реализация (realization)		Отношение между сущностями, где одна сущность определяет действие, которое другая сущность обязуется выполнить. Отношения используются в двух случаях: между интерфейсами и классами (или компонентами), между вариантами использования и кооперациями. Со стороны стрелки указывается сущность, определяющее действие (интерфейс или вариант использования)

Для ассоциации, агрегации и композиции может указываться кратность (англ. multiplicity), характеризующая общее количество экземпляров сущностей, участвующих в отношении. Она, как правило, указывается с каждой стороны отношения около соответствующей сущности. Кратность может указываться следующими способами:

- * – любое количество экземпляров, в том числе и ни одного;
- целое неотрицательное число – кратность строго фиксирована и равна указанному числу (например: 1, 2 или 5);
- диапазон целых неотрицательных чисел «первое число .. второе число» (например: 1..5, 2..10 или 0..5);
- диапазон чисел от конкретного начального значения до произвольного конечного «первое число .. *» (например: 1..*, 5..* или 0..*);
- перечисление целых неотрицательных чисел и диапазонов через запятую (например: 1, 3..5, 10, 15..*).

Если кратность не указана, то принимается ее значение, равное 1. Кратность экземпляров сущностей, участвующих в зависимости, обобщении и реализации, всегда принимается равной 1.

В таблице 4.4. приведено описание механизмов расширения, применяемых для уточнения семантики сущностей и отношений. В общем случае механизм расширения представляет собой строку текста, заключенную в скобки или кавычки.

Таблица. 4.4. **Механизмы расширения UML**

Наименование	**Обозначение**	**Определение (семантика)**
Стереотип (stereotype)	« »	Обозначение, уточняющее семантику элемента нотации (например, зависимость со стереотипом «include» рассматривается как отношение включения, а класс со стереотипом «boundary» – граничный класс)

Наименование	Обозначение	Определение (семантика)
Сторожевое условие (guard condition)	[ ]	Логическое условие (например: [A > B] или [идентификация выполнена])
Ограничение (constraint)	{ }	Правило, ограничивающее семантику элемента модели (например, {время выполнения менее 10 мс})
Помеченное значение (tagged value)	{ }	Новое или уточняющее свойство элемента нотации (например: {version = 3.2})

Диаграмма представляет собой группировку элементов нотации для отображения некоторого аспекта разрабатываемой информационной системы. Диаграммы представляют собой, как правило, связный граф, в котором сущности являются вершинами, а отношения – дугами. В следующей таблице дана краткая характеристика диаграмм UML (см. табл. 4.5).

Таблица 4.5. **Механизмы**

<table>
<tr><th colspan="3" rowspan="2">Диаграмма</th><th rowspan="2">Назначение</th><th colspan="3">Тип диаграммы (модели системы)</th></tr>
<tr><th>По степени физической реализации</th><th>По отображению динамики</th><th>По отображаемому аспекту</th></tr>
<tr><td colspan="3">Вариантов использования (use case)</td><td>Отображает функции системы, взаимодействие между актерами и функциями</td><td>Логическая</td><td>Статическая</td><td>Функциональная</td></tr>
<tr><td colspan="3">Классов (class)</td><td>Отображает набор классов, интерфейсов и отношений между ними</td><td>Логическая или физическая</td><td>Статическая</td><td>Функционально-информационная</td></tr>
<tr><td rowspan="4">Поведения (behavior)</td><td colspan="2">Состояний (statechart)</td><td>Отображает состояния сущности и переходы между ними в процессе ее жизненного цикла</td><td rowspan="4">Логическая</td><td rowspan="4">Динамическая</td><td rowspan="4">Поведенческая</td></tr>
<tr><td colspan="2">Деятельности (activity)</td><td>Отображает бизнес-процессы в системе (описание алгоритмов поведения)</td></tr>
<tr><td rowspan="2">Взаимодействия (interaction)</td><td>Последовательности (sequence)</td><td>Отображает последовательность передачи сообщений между объектами и актерами</td></tr>
<tr><td>Кооперации (collaboration)</td><td>Аналогична диаграмме последовательности, но основной акцент делается на структуру взаимодействия между объектами</td></tr>
<tr><td rowspan="2">Реализации (implementation)</td><td colspan="2">Компонентов (component)</td><td>Отображает компоненты системы (программы, библиотеки, таблицы и т.д.) и связи между ними</td><td rowspan="2">Физическая</td><td rowspan="2">Статическая</td><td rowspan="2">Компонентная (структурная)</td></tr>
<tr><td colspan="2">Развертывания (deployment)</td><td>Отображает размещение компонентов по узлам сети, а также ее конфигурацию</td></tr>
</table>

4.3. Объектный язык ограничений (OCL)

В состав UML входит объектный язык ограничений (Object Constraint Language – OCL). Ограничение (constraint) – это условие, накладываемое на значения одного или нескольких элементов модели. Ограничение не является инструкцией или командой, которую следует выполнить, оно формулируется как утверждение, которое должно быть истинным. Под элементом модели здесь имеется в виду объект или класс, или пакет, или подсистема, или атрибут, или операция, или связь. Язык OCL позволяет прослеживать конструкции в моделях классов.

Эффективность OCL обусловлена возможностью построения сложных выражений из простых конструкций. Например, выражение OCL может последовательно прослеживать несколько ассоциаций. В нем может быть несколько квалификаторов, фильтров, операторов и т.д.

Рассмотрим пример:

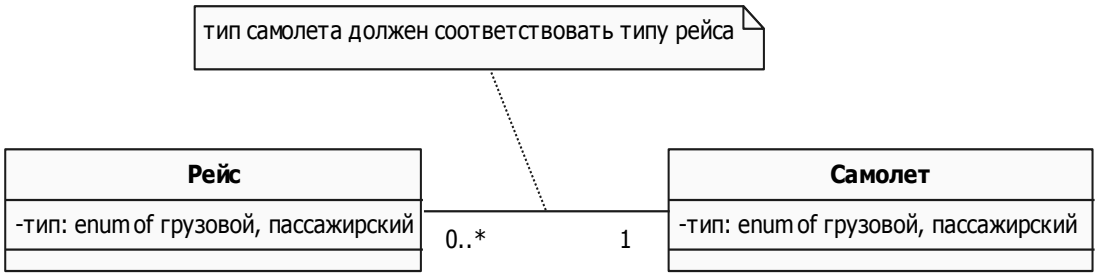


Рисунок 4.3. **Пример ограничения для ассоциаций**

Диаграмма содержит большое количество связей (ассоциаций и обобщений) необходимых для указания, что тип самолета должен соответствовать типу рейса, т.е. что пассажиров нельзя перевозить грузовым самолетом. Это ограничение можно зафиксировать иначе, упростить диаграмму, сделать ее более наглядной.

Ограничение, записанное на естественном языке, неформально, его можно неправильно трактовать (например, что чартерные рейсы должны выполняться старыми самолетами, а регулярные – новыми). Поэтому имеет смысл использовать для записи формальный язык, который не допускает произвольных толкований и имеет стандартный синтаксис и семантику. Таковым является объектный язык ограничений OCL (Object Constraint Language), являющийся одним из расширений UML. С использованием OCL диаграмма будет выглядеть так:

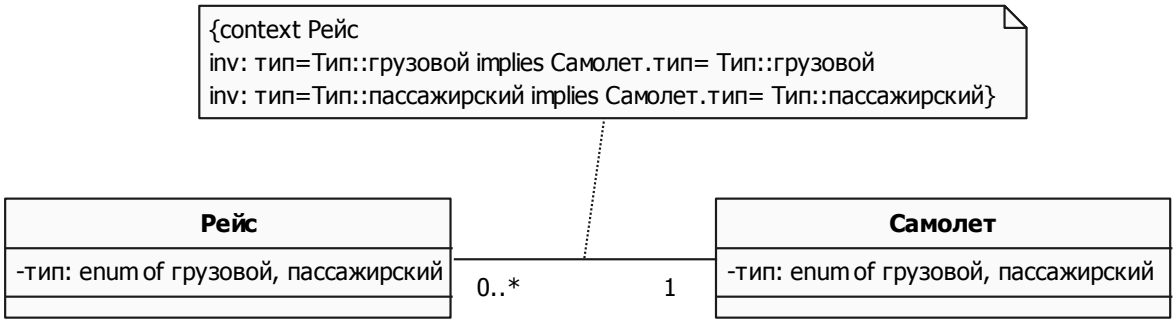


Рисунок 4.4. **Формализация ограничения на OCL**

Слово implies означает логическую операцию импликации ($a \rightarrow b$, читается так: из a следует b, это выражение ложно лишь при a – истина и b – ложь, в остальных случаях оно истинно).

Классификация ограничений:

- Инвариант класса – условие, которое всегда справедливо для всех экземпляров класса (ключевое слово **inv:**).
- Предусловие операции – условие, которое должно быть истинно перед выполнением операции (ключевое слово **pre:**).

- Постусловие операции – условие, истинное всегда после выполнения операции (ключевое слово **post:**).
- Тело запроса – описание результата операции-запроса, не модифицирующей объекты (ключевое слово **body:**)

Начальное значение атрибута или соединения (ключевое слово **init:)**

Правило вывода, описывающее производные атрибуты, связи или классы (ключевое слово **derive:**).

Дополнительное ограничение, введенное для записи других ограничений (ключевое слово **def**:).

Синтаксис OCL-выражения

<OCL-выражение> ::=
<указание контекста>
[(**inv** | **pre** | **post** | **body** | **init** | **derive** | **def**) : <тело выражения>]

В записи использованы символы языка БНФ: <> выделяют нетерминалы, (|) – вхождение одной из указанных альтернатив, [] вхождение 1 или более раз, {} – вхождение 0 или более раз. Терминалы записаны жирным шрифтом.

Контекст. В любом OCL-выражении указывается определенный контекст. Как правило, контекстом является элемент модели (пакет, класс, атрибут, операция), с которым связано ограничение.

<указание контекста> ::= context <имя элемента модели>

Для того чтобы сослаться на контекст в теле выражения используется слово self. Чтобы много раз не писать self, оно часто опускается.

OCL помимо диаграмм классов может использоваться на других диаграммах. На диаграммах взаимодействия OCL применяют для записи сторожевых условий (от выполнения которых зависит, будет ли послано сообщение). На диаграммах деятельности OCL применяют для описания деятельностей, узлов принятия решения, сторожевых условий на потоках. На диаграммах состояний OCL используется для описания состояний и сторожевых условий на переходах между состояниями.

4.4. Диаграммы вариантов использования

Диаграммы вариантов использования используются для моделирования бизнес-процессов организации и требований к создаваемой системе. Диаграммы вариантов использования описывают функциональное назначение системы или то, что система должна делать. Разработка диаграммы преследует следующие цели:

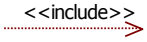
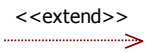
- определить общие границы и контекст моделируемой предметной области;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Разрабатывая диаграммы вариантов использования, придерживаются следующих правил:

- между действующими лицами связи не моделируют;
- не соединяют стрелкой два варианта использования непосредственно. Диаграмма данного типа описывает только, какие варианты использования доступны системе, а не порядок их выполнения;
- каждый вариант использования должен быть инициирован актером.

На диаграмме вариантов использования показано взаимодействие между вариантами использования и действующими лицами. Таким образом, варианты использования – это функции, выполняемые системой, а действующие лица – это заинтересованные лица по отношению к создаваемой системе. Такие диаграммы показывают, какие действующие лица инициируют варианты использования. Из них также видно, когда действующее лицо получает информацию от варианта использования. Основные графические примитивы представлены в таблице 4.6.

Таблица 4.6. **Графические примитивы диаграммы вариантов использования**

Обозначение	Название	Описание
	Актер	Роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Действующие лица делятся на три основных типа – пользователи системы, другие системы, взаимодействующие с данной, и время. Время становится действующим лицом, если от него зависит запуск каких-либо событий в системе
	Вариант использования	Последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой
	Отношение ассоциации	Служит для обозначения специфической роли актера в отдельном варианте использования
	Отношение включения	Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования (отношение определяется стрелкой от включающего к включаемому). Включаемый use case всегда выполняется в ходе выполнения включающего use case
	Отношение расширения	Отношение расширения добавляет к варианту использования дополнительное поведение. В случае расширения один вариант использования (расширяющий) добавляет себя к другому варианту использования (базовому). Расширяющий и расширяемый use case не зависят друг от друга. Обобщения (отношение «Является») в этом отношении нет. Расширяющий use case содержит дополнительный функционал (часто условный)
	Отношение обобщения	Отношение обобщения позволяет описывать вариации базового варианта использования. Вариант использования, являющийся предком, описывает общую последовательность поведения

Хорошим источником для идентификации вариантов использования служат внешние события. Следует начать с перечисления всех событий, происходящих во внешнем мире, на которые система должна каким-то образом реагировать. Какое-либо конкретное событие может повлечь за собой реакцию системы, не требующую вмешательства пользователей, или, наоборот, вызвать чисто пользовательскую реакцию. Идентификация событий, на которые необходимо реагировать, помогает идентифицировать варианты использования.

Диаграмма вариантов использования является самым общим представлением функциональных требований к системе. Детально функциональные требования описываются в документах, называемых «описание варианта использования». Каждое такое описание содержит один и более сценариев или потоков событий варианта использования. Описание подробно документирует взаимодействие действующих лиц с системой, осуществляемое в рамках варианта использования. Диаграмма вариантов использования сама по себе в отрыве от их описаний имеет мало смысла.

Этот документ подробно описывает, что будут делать пользователи системы и что – сама система. Цель – описать то, что будет делать система, а не как она будет делать это. Обычно поток событий включает:

- краткое описание;
- предусловия (pre-conditions) – это такие условия, которые должны быть выполнены, прежде чем вариант использования начнет выполняться сам. Не у всех вариантов использования бывают предусловия;

- основной поток событий;
- альтернативный поток событий;
- постусловия (post-conditions) – это условия, которые всегда должны быть выполнены после завершения варианта использования. Постусловия имеются не у каждого варианта использования.

Варианты использования являются необходимым средством на стадии формирования требований к ПО. Каждый вариант использования – это потенциальное требование к системе, и пока оно не выявлено, невозможно запланировать его реализацию.

4.5. Диаграммы классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. Диаграмма классов состоит из множества элементов, которые в совокупности отражают декларативные знания о предметной области. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами (см. табл. 4.7). На данной диаграмме не указывается информация о временных аспектах функционирования системы.

Таблица 4.7. **Графические примитивы диаграммы классов**

Обозначение	Название	Описание
	Класс	Служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Описание класса состоит в определении атрибутов (свойств) и методов (операций или сервисов)
	Отношение зависимости	Указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов, выраженное в том, что некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели
	Отношение ассоциации	Соответствует наличию некоторого отношения между классами
	Отношение агрегации	Частный случай ассоциации, описывающий объекты, состоящие из частей. Агрегация представляет собой отношение «часть – целое»
	Отношение композиции	Частный случай агрегации. Служит для выделения специальной формы отношения «часть – целое», при которой составляющие части в некотором смысле находятся внутри целого. Части не могут выступать в отрыве от целого, т.е. с уничтожением целого уничтожаются и все его составные части
	Отношение обобщения	Является отношением между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком). Данное отношение описывает иерархическое строение классов и наследование их свойств и поведения

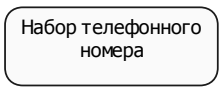
4.6. Диаграммы состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий. Таким образом, диаграмма состояний используется для моделирования поведения объектов системы при переходе из одного состояния в другое.

В отличие от других диаграмм диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее – одного экземпляра определенного класса, т.е. моделирует все возможные изменения в состоянии конкретного объекта. При этом изменение состояния объекта может быть вызвано внешними воздействиями со стороны других объектов или извне. Смена текущего состояния происходит в результате перехода, который вызывается триггером, т.е. наступлением события, приписанного переходу, ведущему из текущего состояния. Будет ли осуществлен переход, зависит не только от срабатывания триггера. Если сторожевое условие при переходе не выполнено, переход не выполняется, даже если произошло событие-триггер. События бывают разных видов: событие вызова (наступает, когда вызывается операция объекта, например, `включитьАвтопилот`); событие изменения (наступает, когда становится истинным условие, например, `when(NumUser=0)`, запись события изменения всегда начинается со слова `when`); событие времени (наступает в заданный момент времени – `at00-00`, или после окончания заданного периода времени, например, `after 5min`). События, приписанные переходам, могут вызвать смену состояний. Если переходу не приписано событие, то это переход по завершению, который может осуществиться по окончании деятельности внутри события или при отсутствии внутренней деятельности сразу после входа в состояние.

Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Они называются деятельностями состояния и указываются на диаграмме. Деятельность состояния – это прерываемое поведение. Оно может выполняться до своего завершения, пока объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Основные графические примитивы представлены в таблице 4.8.

Таблица 4.8. **Графические примитивы диаграммы состояний**

Обозначение	Название	Описание
	Состояние	Абстрактный метакласс, используемый для моделирования отдельной ситуации, в течение которой имеет место выполнение некоторого условия. Состояние определяется именем и списком внутренних действий (деятельностей), которые выполняются в процессе нахождения моделируемого элемента в данном состоянии и характеризуются меткой действия (entry, exit, do, include)
	Начальное состояние	Частный случай состояния, которое не содержит никаких внутренних действий (псевдосостояния). В этом состоянии находится объект по умолчанию в начальный момент времени
	Конечное состояние	Частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояния). В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени
	Переход	Отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния другим. Объект может перейти в то же состояние, в котором он в настоящий момент находится. Рефлексивные переходы изображают в виде стрелки, начинающейся и завершающейся на одном и том же состоянии

Диаграммы состояний могут быть вложены друг в друга, образуя составные состояния. Составные состояния могут быть последовательными, параллельными и историческими.

По своему назначению диаграмма состояний не является обязательным представлением в модели. Наличие у системы нескольких состояний, отличающихся от простой дихотомии «исправен – неисправен», «активен – неактивен», «ожидание – реакция на внешние действия», уже служит признаком необходимости построения диаграммы состояний. Каждое из состояний должно характеризоваться определенной устойчивостью во времени. Другими словами, из каждого состояния на диаграмме не может быть самопроизвольного перехода в какое бы то ни было другое состояние. Все переходы должны быть явно специфицированы, в противном случае построенная диаграмма состояний является либо неполной, либо ошибочной.

Диаграммы состояний создают лишь для классов, экземпляры которых имеют сложное поведение, т.е. они по-разному обрабатывают одни и те же получаемые сообщения в зависимости от своего внутреннего состояния.

4.7. Диаграммы деятельностей

Диаграммы деятельности полезны в описании поведения, включающего большое количество параллельных процессов. В UML 2 проведена граница между диаграммами состояний, базирующимися на формализме конечных автоматов, и диаграммами деятельности, основанными на сетях Петри. Основным элементом диаграмм деятельности является узел действия.

Диаграмма деятельности может содержать поток управления и поток объектов (рис. 4.5). Поток объектов определяет объекты вместе с их состояниями в результате выполнения действий. Диаграммы деятельности можно рассматривать как вольную трактовку формализма сетей Петри. Начальная точка порождает один курсор управления (или маркер) для каждого исходящего перехода. Если переход имеет вес (кратность) больше единицы, то для него порождается столько курсоров, каков его вес. По умолчанию кратность любого ребра равна 1. Если для ребра определено событие, то курсор достигает конца ребра только после наступления такого события. Ограничивающее (сторожевое) условие также определяет, возможно ли перемещение курсора по ребру. При попадании курсора в узел действия происходит ожидание курсоров на всех входящих ребрах и лишь потом запускается единица работы.

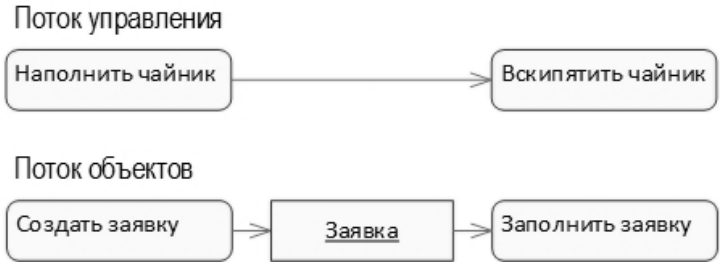
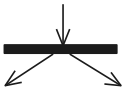


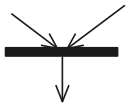
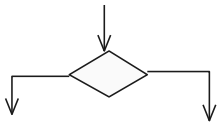
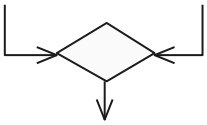
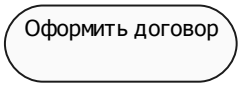
Рисунок 4.5. **Потоки на диаграмме деятельностей**

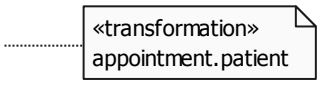
По завершении выполнения работы генерируются курсоры на всех исходящих из узла ребрах. Основные графические примитивы представлены в таблице 4.9.

Таблица 4.9. **Графические примитивы диаграммы деятельностей**

Обозначение	Название	Описание
	Начальный узел	Старт диаграммы
	Ветвление	Ветвление (fork), имеет один входной поток и несколько выходных параллельных потоков

Продолжение таблицы 4.9

Обозначение	Название	Описание
	Объединение	При наличии параллелизма необходима синхронизация.
	Поток / ребро	В UML 2 параллельно употребляются термины поток (flow) и ребро (edge) для обозначения связи между двумя операциями. Самый простой вид ребра – это обычная стрелка между двумя операциями
	Решение	Встречается, когда последовательно выполняемая деятельность должна разделиться на альтернативные ветви в зависимости от значения некоторого промежуточного результата
	Слияние	Слияние обозначает завершение условного поведения, которое было начато решением
	Конец деятельности	Окончание диаграммы
	Узел действия	Узел представляет собой элементарную единицу работы (это может быть решение некоторой задачи, которую необходимо выполнить вручную или автоматизированным способом, или выполнение метода класса)
	Входной параметр	
	Выходной параметр	
	Временной сигнал	Временной сигнал (time signal) приходит по прошествии времени. Такие сигналы могут означать конец месяца в отчетном периоде или приходить каждую секунду в контроллере реального времени
	Принятие сигнала	
	Посылка сигнала	
	Контакт	Процедуры, как и методы, могут иметь параметры. Показывать на диаграмме деятельности информацию о параметрах необязательно, но при желании можно отобразить параметры с помощью контактов (pins). Контакты удобны, когда требуется увидеть данные, принимаемые и передаваемые различными процедурами

Обозначение	Название	Описание
	Выражение преобразования	Если требуется нарисовать точную диаграмму деятельности, то необходимо обеспечить соответствие выходных параметров одной процедуры входным параметрам другой. Если они не совпадают, то можно указать преобразование (transformation) для перехода от одной процедуры к другой
	Окончание потока	Окончание потока (flow final) означает завершение конкретного потока без завершения всей активности

Диаграммы деятельностей предпочтительнее использовать в следующих ситуациях:

- анализ варианта использования. На этой стадии нужно только понять, какие действия должны иметь место и каковы зависимости в поведении системы. Связывание методов и объектов выполняется позднее с помощью диаграмм взаимодействия;
- анализ потоков работ в различных вариантах использования. Когда варианты использования взаимодействуют друг с другом, диаграммы деятельностей являются мощным средством представления и анализа их поведения.

4.8. Диаграммы взаимодействия

Диаграммы взаимодействия описывают поведение совместно действующих групп объектов в рамках потока событий. Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках только одного варианта использования. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой. Сообщение – это средство, с помощью которого объект-отправитель запрашивает у объекта-получателя выполнение одной из его операций.

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы взаимодействия. Говоря об этих диаграммах, имеют в виду два аспекта взаимодействия.

Во-первых, взаимодействия объектов можно рассматривать во времени, и тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности.

Во-вторых, можно рассматривать структурные особенности взаимодействия объектов. Для представления структурных особенностей передачи и приема сообщений между объектами используется коммуникационная диаграмма.

4.8.1. Диаграммы последовательности

Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования. Диаграмма последовательности применяется для рассмотрения взаимодействия объектов во времени. На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами.

Построение диаграммы последовательности целесообразно начинать с выделения из всей совокупности тех и только тех классов, объекты которых участвуют в моделируемом взаимодействии. После этого все объекты наносятся на диаграмму с соблюдением некоторого порядка инициализации сообщений. Здесь необходимо установить, какие объекты будут существовать постоянно, а какие временно – только на период выполнения ими требуемых действий.

Каждое сообщение может быть описано в следующем формате:

номер: переменная = операция (аргументы): возвращаемое значение

номер: – порядковый номер сообщения;

переменная = – указание, где будет сохранен результат;

операция (аргументы) – указание вызываемой операции и фактических параметров;

возвращаемое значение – явное указание возвращаемого значения.

Любая из частей описания может отсутствовать.

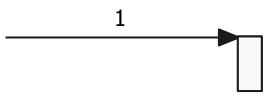
Общая проблема диаграмм последовательности заключается в том, как отображать циклы и условные конструкции. Прежде всего, надо усвоить, что диаграммы последовательности для этого не предназначены. Подобные управляющие структуры лучше показывать с помощью диаграммы деятельности или собственно кода. Диаграммы последовательности применяются для визуализации процесса взаимодействия объектов, а не как средство моделирования алгоритма управления. Существуют дополнительные обозначения: и для циклов, и для условий используются фреймы взаимодействий, представляющие собой средство разметки диаграммы взаимодействия.

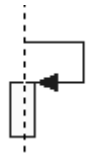
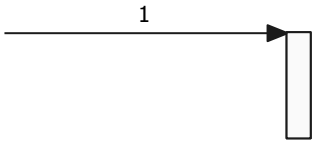
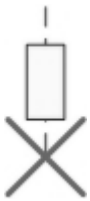
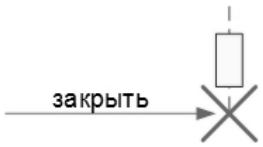
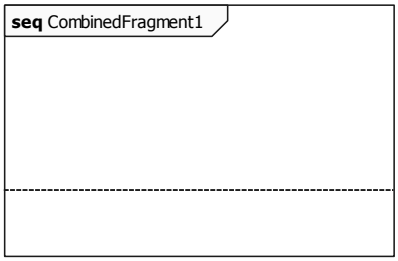
Перечень операторов взаимодействия:

- **alt** – несколько альтернатив (каждая альтернатива представляется отдельным операндом, каждый операнд снабжается сторожевым условием, фрагмент предписывает вычислить все сторожевые условия, в зависимости от истинности сторожевых условий выбрать один операнд взаимодействия и выполнить его, если есть несколько альтернатив с истинными сторожевыми условиями, выбор осуществляется недетерминировано);
- **opt** – необязательное взаимодействие (единственный заданный операнд выполняется только тогда, когда сторожевое условие истинно, иначе не делается ничего);
- **par** – параллельно выполняемые операнды взаимодействия;
- **loop** – цикл (единственный операнд – тело цикла, в сторожевом условии может быть указано минимальное и максимальное количество итераций и/или while-условие);
- **region** – критический участок;
- **neg** – неверное, недопустимое взаимодействие;
- **ref** – использование взаимодействия – ссылка на другую диаграмму, подробно моделирующую один из этапов взаимодействия;
- **sd** – блок, включающий диаграмму последовательности целиком;
- **break** – досрочный выход из цикла (встречается внутри loop-фрагмента, обеспечивает выполнение своего единственного операнда взаимодействия и досрочный выход из обрамляющего цикла при истинном сторожевом условии);
- **strict, seq** – строгое или слабое упорядочение операндов взаимодействия (при строгом упорядочении все выполняется последовательно – первый операнд, за ним второй; при слабом упорядочении упорядочиваются только сообщения, получаемые каждой линией жизни. Сами по себе наборы сообщений, относящиеся к разным линиям жизни, могут рассылаться и обрабатываться параллельно).

Основные графические примитивы представлены в таблице 4.10.

Таблица 4.10. **Графические примитивы диаграммы последовательности**

Обозначение	Название	Описание
	Найденное сообщение	У первого сообщения нет участника, пославшего его, поскольку оно приходит из неизвестного источника. Оно называется найденным сообщением (found message)
	Сообщение	Команда, отправляемая другому участнику. Может содержать только передаваемые данные
	Линия жизни	Служит для обозначения периода времени, в течение которого объект существует в системе и, следовательно, может потенциально участвовать во всех ее взаимодействиях. Каждая линия жизни имеет полосу активности, которая показывает интервал активности участника при взаимодействии
	Участник	В большинстве случаев можно считать участников диаграммы взаимодействия объектами, как это и было в действительности в UML 1. Но в UML 2 их роль значительно сложнее. Поэтому здесь употребляется термин «участники» (participants), который формально не входит в спецификацию UML

Обозначение	Название	Описание
	Самовызов	Участник отправляет сообщение (команду) самому себе
	Возврат	Передача управления обратно участнику, который до этого инициировал сообщение
	Активация	На изображении это – белый вертикальный прямоугольник. Момент, когда участник начинает действовать в ответ на принятое сообщение
	Само-удаление	Удаление участника обозначается большим крестом (X). X в конце линии жизни показывает, что участник удаляет сам себя
	Удаление из другого объекта	Удаление участника обозначается большим крестом (X). Стрелка сообщения, идущая в X, означает, что один участник явным образом удаляет другого
	Фреймы взаимодействия	Для циклов, и для условий используются фреймы взаимодействий (inter action frames), представляющие собой средство разметки диаграммы взаимодействия

4.8.2. Коммуникационные диаграммы

Коммуникационные диаграммы отображают поток событий через конкретный сценарий варианта использования и заостряют внимание на связях между объектами. В отличие от диаграммы последовательности, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии. С другой стороны, на этой диаграмме не указывается время в виде отдельного измерения. Поэтому последовательность взаимодействий и параллельных потоков может быть определена с помощью порядковых номеров. Следовательно, если необходимо явно специфицировать взаимосвязи между объектами в реальном времени, лучше это делать на диаграмме последовательности.

Кооперация может быть представлена на двух уровнях:

- Диаграмма кооперации уровня спецификации показывает роли, которые играют участвующие во взаимодействии элементы. Элементами кооперации на этом уровне являются классы и ассоциации, которые обозначают отдельные роли классификаторов и ассоциации между участниками кооперации.
- Диаграмма кооперации уровня примеров представляется совокупностью объектов (экземпляры классов) и связей (экземпляры ассоциаций). При этом связи дополняются стрелками сообщений. На данном уровне показываются только релевантные объекты.

На кооперативной диаграмме, так же как и на диаграмме последовательности, стрелки обозначают сообщения, обмен которыми осуществляется в рамках данного варианта использования. Их временная последовательность, однако, указывается путем нумерации сообщений.

Из кооперативной диаграммы легче понять связи между объектами, однако труднее увидеть последовательность событий. По этой причине часто для какого-либо сценария создают диаграммы обоих типов. Хотя они служат одной и той же цели и содержат одну и ту же информацию, но представляют ее с разных точек зрения.

4.9. Диаграммы компонентов

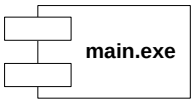
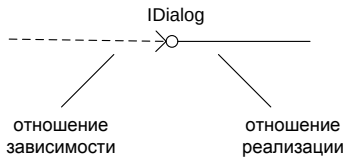
Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки кода.

Диаграмма компонентов разрабатывается для следующих целей:

– визуализации общей структуры исходного кода программной системы;
– спецификации исполнимого варианта программной системы;
– обеспечения многократного использования отдельных фрагментов программного кода;
– представления концептуальной и физической схем баз данных.

Основные графические примитивы представлены в таблице 4.11.

Таблица. 4.11. **Графические примитивы диаграммы компонентов**

Обозначение	Название	Описание
	Компоненты	Реализует некоторый набор интерфейсов и служит для общего обозначения элементов физического представления модели
	Интерфейс	Служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры
	Зависимость	Не является ассоциацией, а служит для представления только факта наличия такой связи, когда изменение одного элемента модели оказывает влияние или приводит к изменению другого элемента модели. Отношение зависимости на диаграмме компонентов изображается пунктирной линией со стрелкой, направленной от клиента (зависимого элемента) к источнику (независимому элементу)

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

После этого можно приступать к общей структуризации диаграммы компонентов. В первую очередь, необходимо решить, из каких физических частей (файлов) будет состоять программная система.

После общей структуризации физического представления системы необходимо дополнить модель интерфейсами и схемами базы данных. При разработке интерфейсов следует обращать внимание на согласование (стыковку) различных частей программной системы. Включение в мо-

дель схемы базы данных предполагает спецификацию отдельных таблиц и установление информационных связей между таблицами.

Наконец, завершающий этап построения диаграммы компонентов связан с установлением и нанесением на диаграмму взаимосвязей между компонентами, а также отношений реализации. Эти отношения должны иллюстрировать все важнейшие аспекты физической реализации системы.

4.10. Диаграммы размещения

Диаграмма размещения отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она показывает размещение объектов и компонентов в распределенной системе. Применяется для представления общей конфигурации и топологии распределенной программной системы и содержит распределение компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений – маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются. Так, компоненты с исходными текстами программ могут присутствовать только на диаграмме компонентов. На диаграмме развертывания они не указываются. Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации.

Основные графические примитивы представлены в таблице 4.12.

Таблица 4.12. **Графические примитивы диаграммы развертывания**

Обозначение	Название	Описание
	Узел	Некоторый физически существующий элемент системы, обладающий некоторым вычислительным ресурсом
	Соединения	Являются разновидностью ассоциации. Наличие такой линии указывает на необходимость организации физического канала для обмена информацией между соответствующими узлами. Характер соединения может быть дополнительно специфицирован примечанием, помеченным значением или ограничением.

5. ПРОЦЕССНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ ARIS

5.1. Общие сведения

Методология ARIS основана на разработанной профессором А.В. Шеером теории «Архитектура интегрированных информационных систем» (Architecture of Integrated Information System – ARIS). Методология ARIS основывается на концепции интеграции, предлагающей целостный взгляд на бизнес-процессы, и представляет собой множество различных методологий, интегрированных в рамках единого системного подхода. Данная методология определяет принципы моделирования практически всех аспектов деятельности организаций. Отличительными особенностями методологии ARIS являются взаимосвязанность и взаимосогласованность моделей. Методология ARIS дает возможность описывать достаточно разнородные подсистемы в виде взаимоувязанной и взаимосогласованной совокупности различных моделей, которые хранятся в едином репозитории.

ARIS поддерживает четыре типа моделей, отражающих различные аспекты исследуемой системы [13,22]:

- функциональное представление содержит описание функций бизнес-процесса или системы, отдельных подфункций (операций) и их взаимосвязи между собой;
- информационное представление описывает состояния информационных объектов (данных) и события, приводящие к их изменению;
- организационное представление определяет совокупность организационных единиц и их взаимосвязей;
- управляющее представление описывает взаимосвязи между указанными представлениями.

Каждое представление содержит разные диаграммы (ARIS поддерживает разнообразные графические нотации), которые по времени их возникновения относят к трем последовательным уровням или этапам проработки представления:

- на уровне описания требований происходит определение целей моделирования, языка предметной области и программного решения рассматриваемой задачи, которое базируется на результатах анализа проблем бизнеса и позволяет описать формализованные требования к системе;
- на уровне спецификации проекта концептуальные понятия, сформулированные на предыдущем уровне формулировки требований, трансформируются в категории, методы и алгоритмы в терминах информационных технологий;
- на уровне описания реализации спецификация проекта трансформируется в конкретные аппаратные и программные компоненты.

Управляющее представление представлено нотациями VAD и eEPC/PCD, которые описаны ниже.

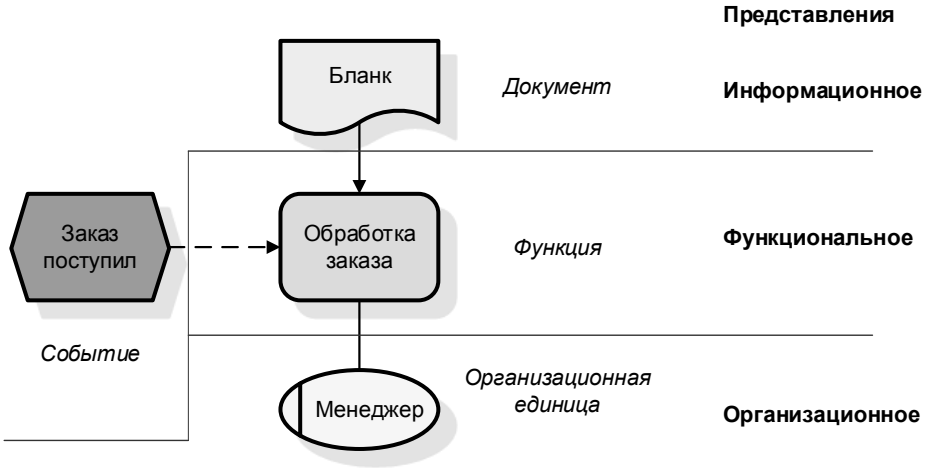


Рисунок 5.1. **Представления ARIS и базовые графические примитивы**

Для построения перечисленных типов моделей используются как собственные методы моделирования ARIS, так и различные известные методы и языки моделирования – ERM, UML, OMT и др.

Методология ARIS позиционирует себя как конструктор, из которого под конкретный проект в зависимости от его целей и задач разрабатывается локальная методология, состоящая из небольшого количества требуемых бизнес-моделей и объектов. В методологии ARIS смысловое значение имеет цвет, что повышает восприимчивость и читабельность схем бизнес-моделей. Например, структурные подразделения по умолчанию изображаются желтым цветом, бизнес-процессы и операции – зеленым. За счет высокой степени визуализации бизнес-моделей методологию ARIS могут использовать все сотрудники: начиная от топ-менеджеров и заканчивая рядовыми сотрудниками. Помимо большего количества моделей по сравнению с другими методологиями, методология ARIS имеет наибольшее количество различных объектов, используемых при построении бизнес-моделей, что увеличивает их аналитичность. Например, материальные и информационные потоки на процессных схемах обозначаются разными по форме и цвету объектами, что позволяет быстро определить тип потока.

Можно выделить следующие диаграммы:

1) Организационные модели:
   - Организационная схема – Organizational chart;
2) Функциональные модели:
   - Дерево функций – Function tree;
   - Диаграмма целей – Objective diagram;
   - Диаграмма типа прикладной системы – Application system type diagram (ASTD);
3) Модели данных (информационные модели):
   - Модель технических терминов – Technical Term Models;
   - Расширенная модель «сущность-отношение» – Extended entity-relationship model (eERM);
   - Диаграмма атрибутов eERM-модели – eERM Attribute allocation diagram;
   - Диаграмма структуры знаний – Knowledge structure diagram;
4) Модели процессов / управления:
   - Событийная цепочка процесса – Extended event driven process chain (eEPC);
   - Диаграмма окружения функции – Function allocation diagram;
   - Производственный и офисный процессы – Industrial and Office process;
   - Диаграмма цепочек добавленного качества – Value-added chain diagram (VAD);
   - Диаграмма информационных потоков – Information flow diagram;
   - Матрица выбора процессов – Process selection matrix;
   - Диаграмма eEPC (в виде столбцов) – eEPC (column display);
   - Карта знаний – Knowledge map;
   - Диаграмма цепочки процесса – Process Chain Diagram (PCD);
   - Диаграмма движения продуктов/услуг – Product/ Service exchange diagram;
   - Дерево продуктов/услуг – Product/Service tree;
   - UML-диаграмма действий – UML Activity diagram;
   - UML-диаграмма класса – UML Class diagram;
   - UML-диаграмма описания класса – UML Class description diagram;
   - UML-диаграмма взаимодействия – UML Collaboration diagram;
   - UML-диаграмма компонентов – UML Component diagram;
   - UML-диаграмма состояний – UML State chart diagram;
   - UML-диаграмма использования приложений – UML Use case diagram.

Несмотря на большее количество моделей в методологии ARIS в проектах по описанию и оптимизации деятельности в общем случае их используется не более десяти. Методически технологию применения ARIS для задач моделирования бизнес-процесса можно представить следующим образом (при необходимости детального освещения любого из четырех представлений список диаграмм можно расширить по желанию аналитика):

1) Определение цели моделирования и точки зрения на проект.
2) Построение диаграммы VAD и определение управляющей модели на высоком уровне.
3) Построение диаграмм eEPC/PCD, описывающих логику бизнес-процессов в виде взаимодействующих событий, функций, информационных объектов и организационных единиц.

4) Построение иерархий функций, данных и организационных единиц на уровне описания требований путем обобщения или уточнения абстракций, появившихся на диаграммах VAD и eEPC/PCD.
5) Проработка функционального, информационного и организационного представлений на уровнях спецификации и реализации.
6) Корректировка eEPC/PCD и VAD диаграмм по результатам проработки функционального, информационного и организационного представлений.

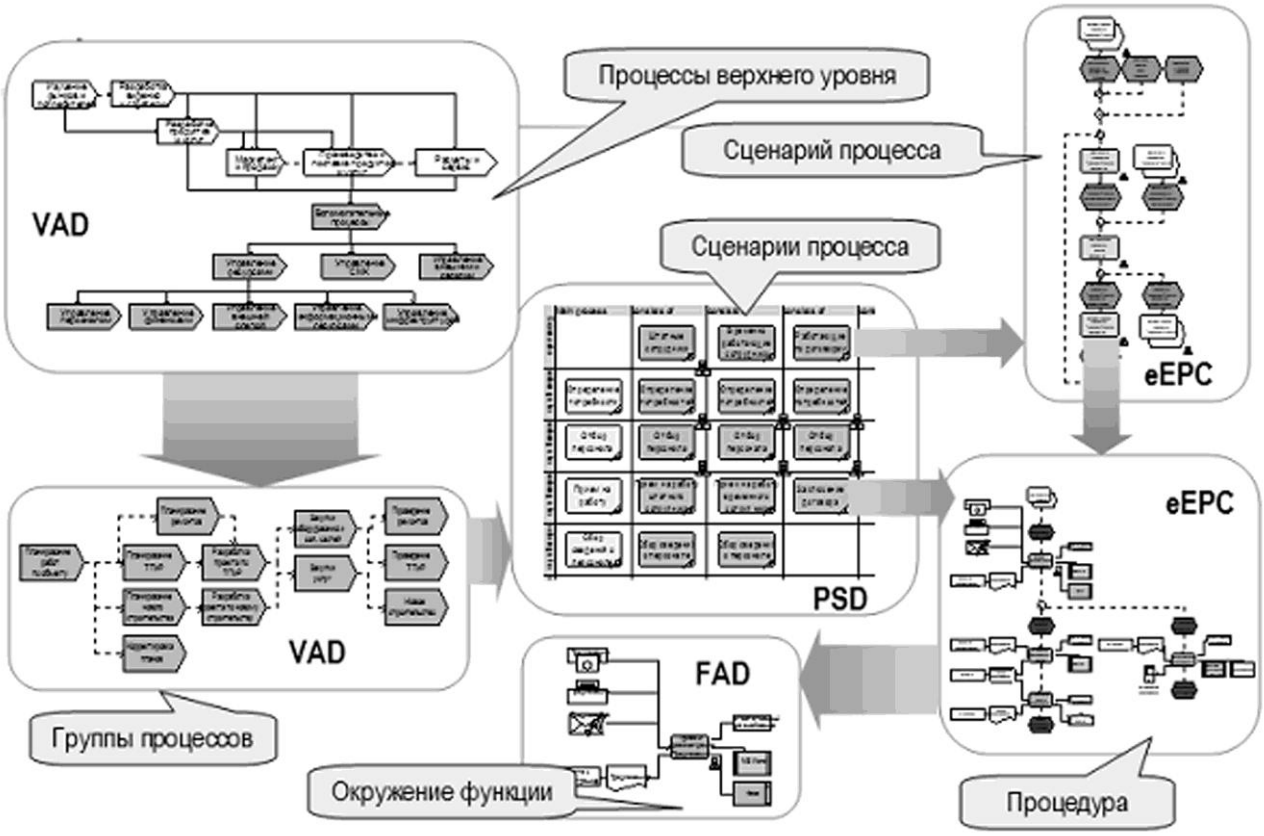


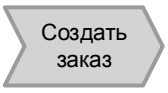
Рисунок 5.2. **Описание процессов**

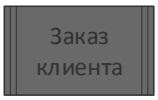
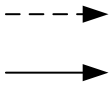
5.2. Диаграмма цепочки добавленной стоимости (VAD)

Диаграммы VAD (Value Added Chain Diagrams) описывают цепочки процессов, добавляющих стоимость, и используются для представления процесса на верхнем уровне. Диаграмма цепочек добавленного качества описывает функции организации, которые непосредственно влияют на реальный выход ее продукции.

В таблице 5.1 приведены основные элементы диаграммы VAD.

Таблица 5.1. **Графические примитивы VAD-диаграмм**

Обозначение	Название
	Процесс
	Ресурс

Обозначение	Название
	Кластер данных
	Тип прикладной информационной системы
	Связи

Между процессами передаются информационные и материальные объекты. Процессы или некоторые группы функций соединяются между собой пунктирной стрелкой, которая имеет тип «is predecessor of» (является предшественником). При этом возможно существование и обратной связи.

Между процессами отображаются потоки материальных ресурсов и информации. Для описания инфраструктуры, необходимой для выполнения процесса, используются организационные единица и ресурсы (типа «Product Service» и «Information Service»). Связь с процессами обозначается сплошной линией – при этом возможно указание типа связи.

5.3. Расширенные цепочки процессов eEPC

Расширенные цепочки процессов, управляемых событиями (eEPC – Extended Event Driven Process Chains) предназначены для описания сложных процессов и содержат события, функции, информационные объекты и организационные единицы.

Модель предназначена для детального описания процессов, выполняемых в рамках одного подразделения, несколькими подразделениями или конкретными сотрудниками. Она позволяет выявлять взаимосвязи между организационной и функциональной моделями. Модель eEPC отражает последовательность функциональных шагов (действий) в рамках одного бизнес-процесса, которые выполняются организационными единицами, а также ограничения по времени, налагаемые на отдельные функции.

Для каждой функции могут быть определены начальное и конечное события, ответственные исполнители, материальные и документарные потоки, сопровождающие модель, а также проведена декомпозиция на более низкие уровни (подфункции и т.д.).

Модель eEPC является наиболее информативной и удобной при описании деятельности подразделений организации.

Таблица. 5.2. **Графические примитивы eEPC-диаграмм**

Обозначение	Название
	Событие
	Функция
	Главный процесс

Продолжение таблицы 5.2

Обозначение	**Название**
	Подразделение (организационная единица)
	Документ
	Связь с функциями
	Связь между событиями и функциями

Операторы		
 И	 ИЛИ	 Исключающее ИЛИ

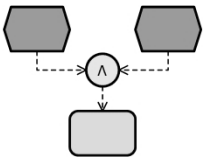
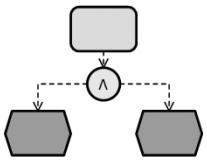
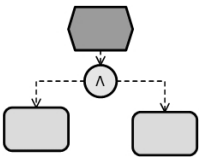
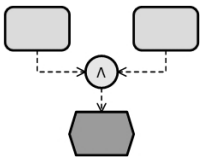
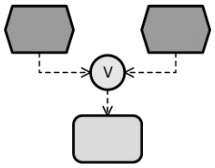
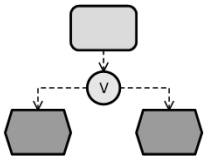
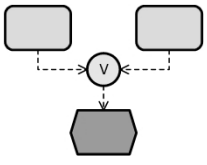
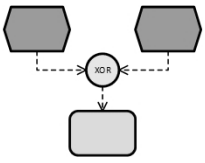
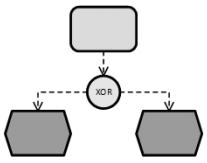
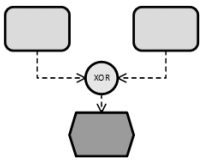
Событие служит для отображения возможных результатов выполнения функций и инициирует выполнение новых функций. Центральным элементом диаграмм нотации EPC являются события. События порождают выполнение неких действий некими участниками. Завершение выполнения действий, в свою очередь, генерирует другое событие и так далее, пока система не придет в состояние, появление которого в рамках процесса считается конечным событием. Каждая функция должна быть инициирована событием и завершена событием. В каждую функцию не может входить более одной стрелки, инициирующей ее выполнение, и выходить не более одной стрелки, описывающей завершение ее выполнения.

Для описания ветвления используются операторы. Возможны разные соединения операторов и функций кроме двух: после события нельзя использовать операторы ИЛИ и исключающего ИЛИ, так как решение может приниматься только в процессе выполнения функции (табл. 5.3).

Для соединения событий и функций используются пунктирные линии, для соединения функций, информационных объектов и организационных единиц – сплошные.

Графические элементы на диаграмме eEPC могут располагаться в произвольном порядке, однако обычно выбирают направление слева направо или сверху вниз.

Таблица 5.3. **Типы соединений**

Соединение	**Связывание событий**		**Связывание функций**	
	Инициирующие события	*Инициируемые события*	*инициирующие события*	*Инициируемые события*
 И				
 ИЛИ			запрещено	
 Исключающее ИЛИ			запрещено	

5.4. Диаграммы цепочки процессов PCD

Диаграммы цепочки процессов (PCD – Process Chain Diagrams) являются разновидностью диаграмм eEPC. Нотации объектов моделей диаграмм PCD и eEPC совпадают (см. раздел 5.3).

PCD – диаграмма цепочки процесса разделена на ряд столбцов, имеющих следующие названия:

- event – событие;
- function – функция;
- data – данные;
- medium – носитель;
- application system – прикладная система;
- organizational unit – организационная единица;
- screen/list – экран/список;
- batch – пакетная обработка;
- dialog – интерактивная обработка;
- manual – ручная обработка;
- objects – объект;
- product/service – продукт/услуга;
- objective – цель;
- other – прочее.

В этих столбцах располагаются соответствующие объекты, тем самым диаграмма приобретает четкую структуру, что делает ее легко читаемой. Отдельные столбцы могут быть опущены. Заголовки также можно изменить.

Распределение графических элементов по столбцам в некоторых случаях облегчает читаемость диаграмм. Такое представление лучше подходит при документировании, но неудобно при описании процессов с разветвленной логикой.

Первый и второй столбцы отображают логическую последовательность выполнения процесса. Отдельные функции процесса, представленные во втором столбце, связаны с событиями, которые инициируют их выполнение или переключение между ними.

Функции и события связаны пунктирными стрелками, отображающими поток управления функциями. Отношение между событиями и функциями составляет процедурную последовательность функций как логическую цепочку событий, которую называют цепочкой процесса. Логическая взаимозависимость возможных точек ветвления и циклов потока управления может быть выражена посредством логических операций, связывающих функции и события.

Входные и выходные данные, требующиеся функциям, показаны в третьем столбце в виде сущностей. Отображаться могут не только информационные объекты, но и носители (четвертый столбец). Это может быть документ, список или устройство памяти.

В пятом столбце могут быть показаны задействованные в моделируемом процессе прикладные информационные системы, а также их отдельные компоненты.

Организационные единицы, ответственные за выполнение отдельных функций моделируемого процесса, показаны в шестом столбце.

Столбец «экран/список» может содержать объекты, характеризующие представление информации в ходе выполнения процесса.

Столбцы «пакетная обработка», «интерактивная обработка», «ручная обработка» и столбец «прикладная система» представляют дополнительную информацию о степени использования информационных технологий для поддержки отдельной функции процесса. В остальных столбцах может содержаться дополнительная информация, характеризующая цели процесса и его отдельные функции, а также задействованные и производимые продукты и услуги.

Диаграммы процесса могут быть использованы при анализе реальных бизнес-процессов, определении недостатков их организации или причин неэффективности. Такими недостатками могут быть дезинтеграция между ручной обработкой и обработкой с помощью информационных технологий или организационная дезинтеграция. Кроме того, проявляются лишние входы (процедурная избыточность данных) и задержки в выполнении.

6. НОТАЦИЯ ПРОЕКТИРОВАНИЯ БИЗНЕС-ПРОЦЕССОВ BPMN

Инициативная группа по управлению бизнес-процессами (Business Process Management Initiative (BMPI)) разработала стандарт «Нотация моделирования бизнес-процессов (Business Process Modeling Notation (BPMN))». Основное назначение стандарта [18] – создание нотации, понятной всем участникам бизнес-сферы, от бизнес-аналитиков, создающих первоначальные эскизы процессов, технических разработчиков, ответственных за внедрение технологии, в которой будут представлены данные процессы, и, наконец, до бизнесменов, которые будут управлять этими процессами, а также осуществлять их мониторинг. Таким образом, BMPN является стандартизованным связующим звеном между разработкой бизнес-процессов и их реализацией.

Другой не менее важной целью является визуализация посредством бизнес-ориентированной нотации языков XML, таких как BPEL4WS (Business Process Execution Language for Web Services – язык реализации бизнес-процессов для веб-служб), разработанных для выполнения бизнес-процессов.

Данная спецификация раскрывает понятие и определяет семантику схем бизнес-процессов (Business Process Diagram (BPD)). Цель BPMN – стандартизировать нотацию моделирования бизнес-процессов при наличии множества различных нотаций и точек зрения на моделирование. Предполагается, что использование BPMN обеспечит легкую передачу информации по процессам другим участникам бизнес-сферы, специалистам по внедрению процессов, клиентам и поставщикам.

При разработке данного стандарта были рассмотрены следующие нотации и методологии: UML Activity Diagram, UML EDOC Business Processes, IDEF, ebXML BPSS, Activity-Decision Flow (ADF) Diagram, RosettaNet, LOVeM, and Event-Process Chains (EPCs).

В BPMN не включаются следующие типы моделирования [19]:

- организационные структуры и ресурсы;
- функциональные схемы;
- модели данных и информационные модели;
- стратегии;
- бизнес-правила.

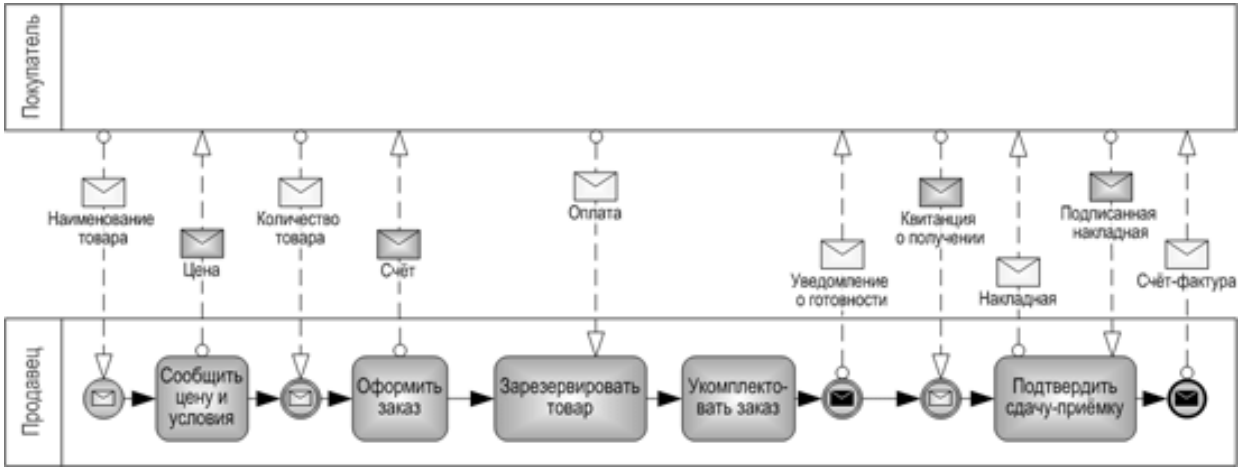
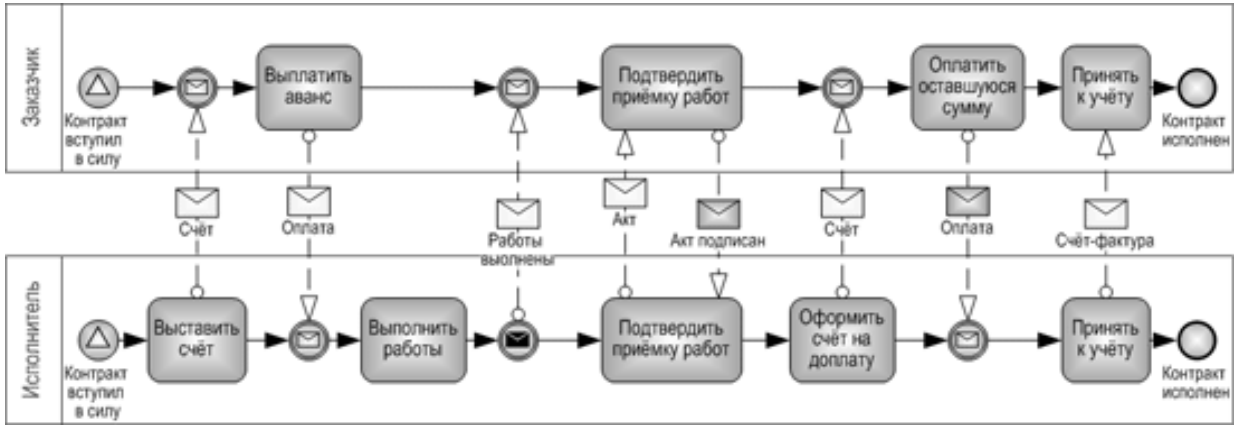
Можно выделить три основных аспекта соответствия спецификации BPMN:

- Внешний вид графических элементов BPMN. Ключевой элемент BPMN – это выбор форм и значков, используемых в графических элементах, указанных в данной спецификации. Цель – создание стандартного визуального языка, который будет узнаваем и понятен для всех разработчиков процессов, вне зависимости от источника схемы.
- Семантика элементов BPMN. Данная спецификация также определяет способ взаимодействия графических элементов друг с другом, включая условные взаимодействия, основанные на атрибутах, создающих поведенческие изменения элементов. Инструмент соответствия должен соотноситься с данными семантическими описаниями.
- Обмен схемами BPMN между инструментами соответствия. Данный проект спецификации не содержит стандартного механизма обмена схемами. Характер этого механизма еще не определен.

Моделирование бизнес-процессов предназначено для сообщения разнообразной информации широкой аудитории. BPMN описывает множество типов моделирования и допускает создание сквозных бизнес-процессов. Существует три основных типа подмоделей в рамках сквозной модели BPMN:

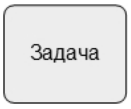
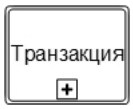
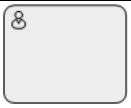
- частные (внутренние) бизнес-процессы являются внутренними для определенной организации, данный тип бизнес-процессов обычно называют workflow или процессы BPM (управление деловыми процессами);
- абстрактные (открытые) процессы представляют собой взаимодействие между частным бизнес-процессом и другим процессом или участником. Абстрактными считаются только те процессы, действия которых имеют связи за пределами частного бизнес-процесса. Абстрактный процесс показывает последовательность сообщений, которые должны взаимодействовать с данным бизнес-процессом;
- совместные (глобальные) процессы отображают взаимодействие между двумя и более бизнес-объектами. Эти взаимодействия состоят в обмене сообщениями между данными объектами. Совместный процесс можно изобразить в виде двух или более взаимодействующих абстрактных процессов.

Таблица 6.1. **Основные модели BPMN [19]**

Наименование	Пример
Частный бизнес-процесс	
Совместный процесс	
Абстрактный процесс	

Ниже приведено краткое описание наиболее часто используемых элементов нотации BPMN.

Таблица 6.2. **Графические примитивы BPMN**

Обозначение	Краткое описание		
Действия			
	Задача – это единица работы. Если задача помечена символом ⊞, то задача является подпроцессом и может быть детализирована		
	Транзакция – набор логически связанных действий. Для транзакции может быть определен протокол выполнения		
	Событийный подпроцесс помещается внутри другого процесса. Он начинает выполняться, если инициируется его начальное событие. Событийный подпроцесс может прерывать родительский подпроцесс или выполняться параллельно с ним		
	Вызывающее действие является точкой входа для глобально определенного подпроцесса, который повторно используется в данном процессе		
Маркеры действий – маркер отражает поведение действия во время выполнения			
	Маркер подпроцесса		Маркер последовательных множественных экземпляров
	Маркер цикла		Маркер ad hoc
	Маркер параллельных множественных экземпляров		Маркер компенсации
Типы задач – тип определяет природу действия, которое будет выполнено			
	Задача отправки сообщения		Задача бизнес-правило
	Задача получения сообщения		Задача-сервис
	Пользовательская задача		Задача-сценарий
	Неавтоматизированная задача		Получить создание экземпляра
Потоки			
	Поток управления: определяет порядок выполнения действий		
	Поток по умолчанию: определяет ветвь, выполняемую, когда все условия ветвления не выполнены		
	Условный поток: связан с условием, определяющим, будет ли выполнен данный поток		

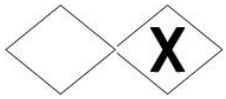
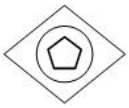
Обозначение	Краткое описание
Логические операторы	
Оператор исключающего ИЛИ, управляемый данными 	При ветвлении направляет поток лишь по одной из исходящих ветвей. При синхронизации потоков оператор ожидает завершения одной входящей ветви и активирует исходящий поток управления
Оператор исключающего ИЛИ, событийный 	Предшествует только событиям обработки или заданиям обработчикам сообщений. Поток управления направляется по той ветви, где событие произошло ранее
Оператор И 	При разделении на параллельные потоки все ветви активируются одновременно. При синхронизации параллельных ветвей оператор ждет завершения всех входящих ветвей и затем активирует исходящий поток
Оператор ИЛИ 	При ветвлении активируется одна или более ветвей. При слиянии все выполняющиеся входящие ветви должны быть завершены
Сложный оператор 	Моделирует сложные условия ветвления и слияния
Оператор исключающего ИЛИ, событийный (создает новый экземпляр) 	Наступление каждого из последующих событий создает экземпляр процесса
Оператор И, событийный (создает новый экземпляр) 	Наступление всех последующих событий создает экземпляр процесса

Таблица 6.3. **Обозначение событий в BPMN**

События	**Начальные**			**Промежуточные**				**Завершающие**
	Высоко-уровневые	**Прерывающие событийный подпроцесс**	**Непрерывающие событийный подпроцесс**	**Обрабатывающие**	**Граничные прерывающие**	**Граничные непрерывающие**	**Генерирующие**	
Простое: нетипизированное событие, обычно показывающее начало или окончание процесса								
Сообщение: получение и отправка сообщений								
Таймер: цикличные события, моменты времени, временные периоды и таймауты								
Эскалация: перенос рассмотрения вопроса на более высокий уровень организационной иерархии								
Условное: реакция на изменение бизнес-условий или интеграция бизнес-правил								
Ссылка: пара соответствующих ссылок эквивалентна потоку последовательности								
Ошибка: генерация и обработка заданного типа ошибок								
Отмена: обработка отмены транзакции или инициирование отмены								
Компенсация: обработка или инициирование компенсации								
Сигнал: передается между процессами и может обрабатываться многими получателями								
Составное: обработка одного события из множества или генерация всех определенных событий								
Параллельное составное: обработка всего множества параллельных событий								
Остановка: вызывает немедленное прекращение выполнения процесса								

7. ПРАКТИКУМ

Практикум включает в себя пять лабораторных работ, ориентированных на изучение особенностей рассмотренных выше методологий путем моделирования конкретного бизнес-процесса.

Рассмотренные методологии моделирования обладают рядом общих свойств:

1) Проектирование начинается с определения цели и точки зрения.
2) Модель представляется в виде иерархии упорядоченных диаграмм, описывающих разные аспекты модели.
3) При этом происходит описание функций (IDEF0 и DFD, диаграммы вариантов использования, последовательности и кооперации UML, VAD диаграмма и функциональное представление ARIS), процессов (диаграммы деятельности и состояний UML, eEPC/PCD диаграммы ARIS) и статической структуры (диаграммы классов UML, информационное и организационное представление ARIS).
4) При построении диаграмм необходимо строго придерживаться выбранной графической нотации, согласно которой абстракции одного класса имеют одинаковое графическое отображение.
5) Обязательным компонентом модели является краткое текстовое описание диаграмм.

Также при применении описанных в данном пособии методологий необходимо руководствоваться общим принципом:

1) Сначала сформулировать идею, которая должна быть отражена на диаграмме (в терминах семантически корректной фразы на русском языке без ошибок).
2) Только затем отобразить основное содержание этой фразы на диаграмме с использованием графической нотации.

Следует избегать формального подхода к проектированию, когда в модель включаются все необходимые составляющие только для того, чтобы соблюсти процедуру проектирования, но не для того, чтобы описать важные идеи. Диаграммы или их фрагменты, которые не содержат существенной семантической нагрузки или содержат «банальные» сведения, должны исключаться из модели – их присутствие является грубой ошибкой.

В целом, необходимо соблюдать два основных требования:

- информационно-логическая модель должна быть адекватной и достаточно детальной;
- описание модели должно быть ясным и наглядным, одновременно понятным заказчику и исполнителю.

Использование современных методологий проектирования и CASE-технологий позволяет выполнить эти требования, а грамотная организация процесса разработки обеспечивает высокую эффективность применения результатов проектирования.

7.1. Описание примера для практической части

Для того чтобы корректно создать систему, отвечающую всем требованиям заказчика, аналитик должен абсолютно четко представить себе ее основные бизнес-функции и выяснить предъявляемые к системе требования. Для этого необходимо провести обследование компании и построить ее полную бизнес-модель.

Рассмотрим условный пример – «Диспетчерская служба управления такси». Диспетчерская служба управления такси предназначена для автоматизации управления приемом заказов, диспетчеризацией бортов и водителей и ведения необходимых для этого справочников и отчетов.

Служба управления такси оказывает услуги по перевозке пассажиров. Схема взаимодействия с потенциальными клиентами следующая:

- клиент звонит оператору, при этом через call-центр определяется номер телефона, с которого делается заказ;
- оператор записывает координаты и контактные телефоны клиента, а также адрес посадки и адрес высадки. При необходимости клиент может заказать дополнительные параметры борта (например, наличие детского кресла или вместительного багажника). После чего заказ поступает диспетчеру на обработку;
- диспетчер осуществляет мониторинг свободных такси (бортов) и передает заказ ближайшему к адресу заказчика водителю;
- водитель осуществляет доставку клиента до места назначения и получает деньги.

7.2. Лабораторная работа 1

Тема: Методология моделирования бизнес-процессов IDEF0.

Цель работы: Изучить теоретические основы методологии моделирования бизнес-процессов IDEF0. Освоить принципы построения IDEF0-диаграммы на примере программной среды Ramus Educational.

Задачи:

1) Ознакомиться с теоретическими аспектами методологии моделирования бизнес-процессов IDEF0.
2) Разработать IDEF0-модель согласно индивидуальной предметной области (вариант задания получить у преподавателя).

7.2.1. Краткий обзор CASE-средств для построения IDEF0-модели

В настоящее время существует значительное количество программных продуктов, которые можно использовать при построении IDEF0-модели. Каждое из программных средств обладает как достоинствами, так и недостатками. К наиболее известным средствам следует отнести CA ERwin Process Modeler 7.3, CASE.Аналитик, Ramus Educational.

CA ERwin Process Modeler 7.3 (старое название – BPwin) представляет собой мощное средство моделирования, которое поддерживает моделирование процессов (методология IDEF0), моделирование потоков данных (методология DFD) и моделирование технологических процессов (методология IDEF3).

CASE-Аналитик является единственной отечественной разработкой, доведенной до рынка, который относится к CASE-средствам первой генерации. В основе пакета лежит методология структурного системного анализа Гейне-Сарсона.

Ramus Educational предназначен для использования в проектах, в которых необходимо описание бизнес-процессов предприятия. Ramus поддерживает методологии моделирования бизнес-процессов IDEF0 и DFD, а также имеет ряд дополнительных возможностей, призванных удовлетворить потребности команд разработчиков систем управления предприятиями. Условно свободно распространяемое ПО.

7.2.2. Разработка модели IDEF0 в системе Ramus Educational

Ramus Educational обладает гибкими возможностями построения отчетности по графическим моделям, позволяющим создавать отчеты в форме документов, регламентирующих деятельность предприятия. Ramus Educational имеет достаточно интуитивный интерфейс пользователя, позволяющий быстро и просто создавать сложные модели.

Начало работы

1. Запустите программу Ramus Educational. В появившемся окне (рис. 7.1) предлагается создать новый проект или открыть уже существующий.

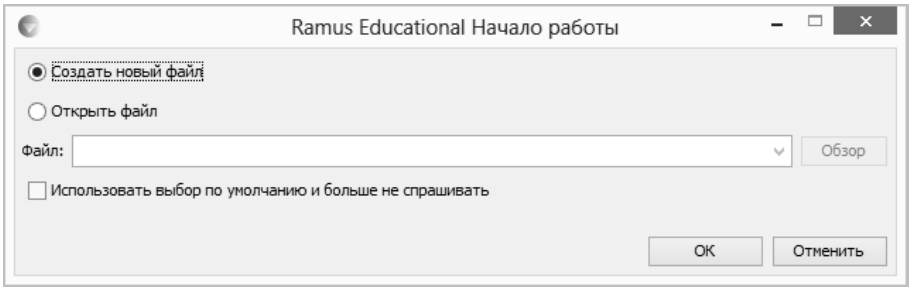


Рисунок 7.1. **Окно запуска Ramus Educational**

2. После нажатия на кнопку «**OK**» осуществляется запуск мастера проекта.

– На первом шаге (рис. 7.2) в соответствующие поля необходимо внести сведения об авторе, названии проекта и модели, а также выбрать тип нотации модели (IDEF0 или DFD).
– На втором шаге вводится название организации, использующей данный проект.
– На третьем – дается краткое описание будущего проекта.
– Четвертый шаг позволяет создать несколько основных классификаторов (в данном случае можно пропустить этот шаг). Так как модели процессов реальных предприятий могут содержать значительное количество объектов (документы, персонал, функции и т.д.), то в Ramus предусмотрена возможность упорядочено хранить информацию об этих объектах в виде системы классификаторов. Классификация объектов упрощает поиск и обработку информации об объектах модели, а также и об объектах, непосредственно не представленных на диаграммах процессов, но относящихся к процессам предприятия.

– На пятом, заключительном, предлагается выбрать те из созданных классификаторов, элементы которых будут содержаться в перечне собственников процессов (пропустить данный шаг).

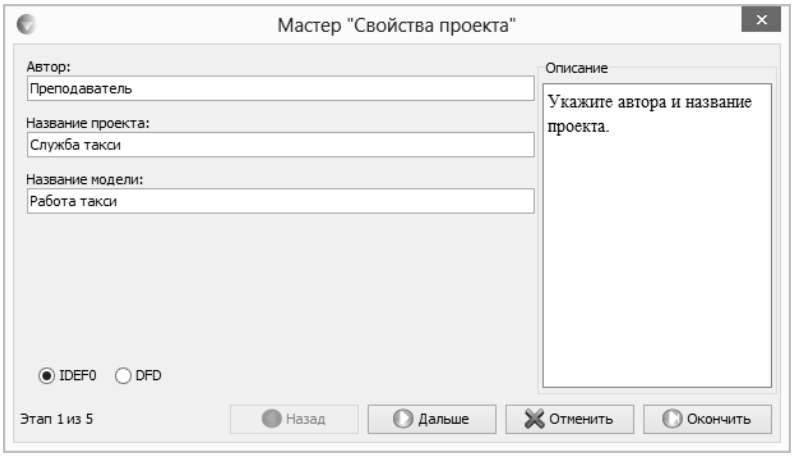


Рисунок 7.2. **Окно мастера создания проекта. Шаг 1**

При необходимости можно завершить работу мастера, нажав кнопку «**Окончить**».

После завершения работы мастера откроется рабочее пространство «**Диаграммы**», в котором можно приступить к рисованию графической модели (рис. 7.3). В верхней части приводятся сведения о проекте, введенные пользователем посредством мастера диаграмм.

Программа Ramus Educational обладает гибким графическим интерфейсом, который можно настроить под нужды и предпочтения конкретного пользователя: ненужные окна можно закрыть/свернуть; можно менять их размеры и месторасположение; также можно группировать два и более окон в одном, при этом содержимое вложенных окон будет размещено на вкладках общего окна (данный функционал возможен не для всех комбинаций окон).

3. Сохраните созданную модель, выбрав опцию меню «**Файл**» → «**Сохранить как**».

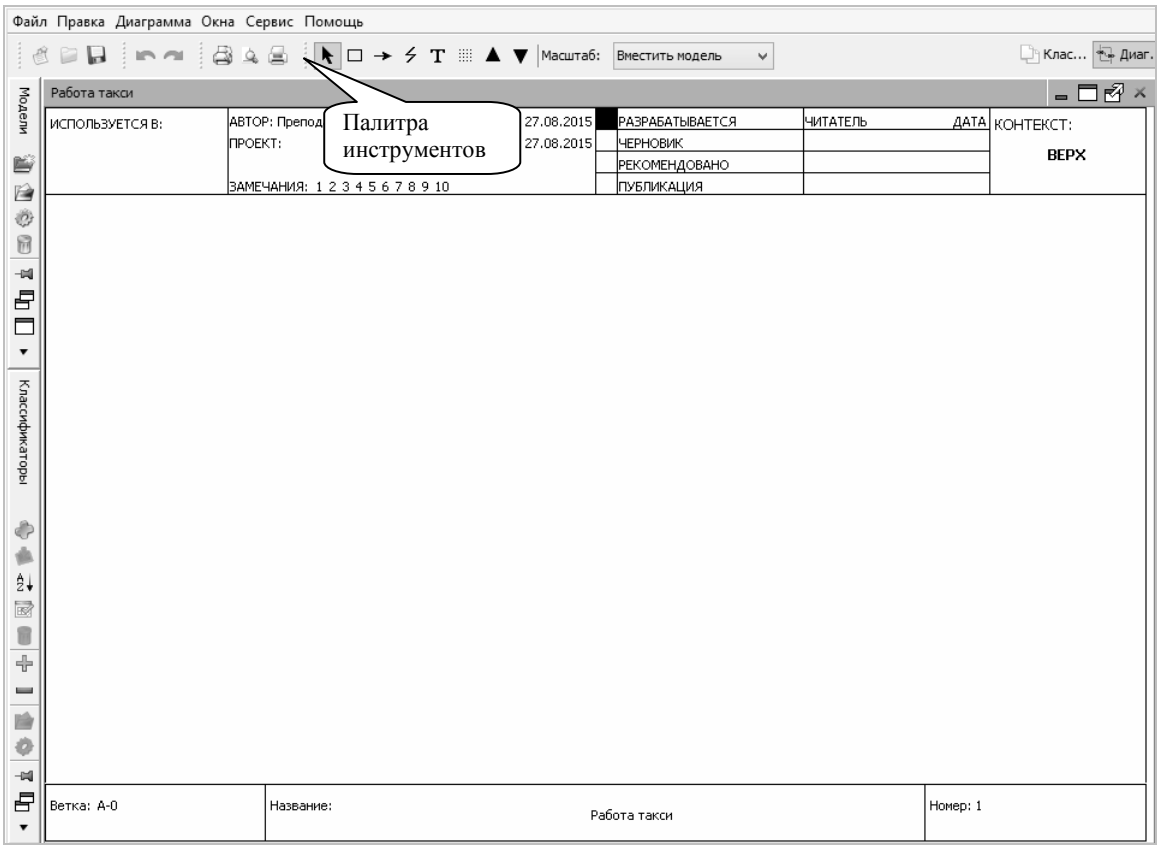


Рисунок 7.3. **Рабочее пространство «Диаграммы»**

Создание контекстной диаграммы

1. На панели инструментов выберите пиктограмму функции (□) и мышью укажите местоположение на рабочем пространстве.
2. Дайте данному функциональному блоку имя «**Управлять службой такси**».
3. Используя режим добавления текста T, добавьте цель и точку зрения модели:
   – *Цель*: Повысить оперативность оказания транспортных услуг.
   – *Точка зрения*: Главный диспетчер.
   – *Области моделирования (Scope, границы модели)*: Описание работы диспетчерской службы такси, включающей прием заказов, учет водителей и бортов, а также мониторинг исполнения заказов. В границу модели не входят вопросы формирования финансовых отчетов.
4. Используя пиктограмму панели инструментов →, создайте соответствующие потоки на контекстной диаграмме. В результате должна получиться контекстная диаграмма, показанная на рисунке 7.4.

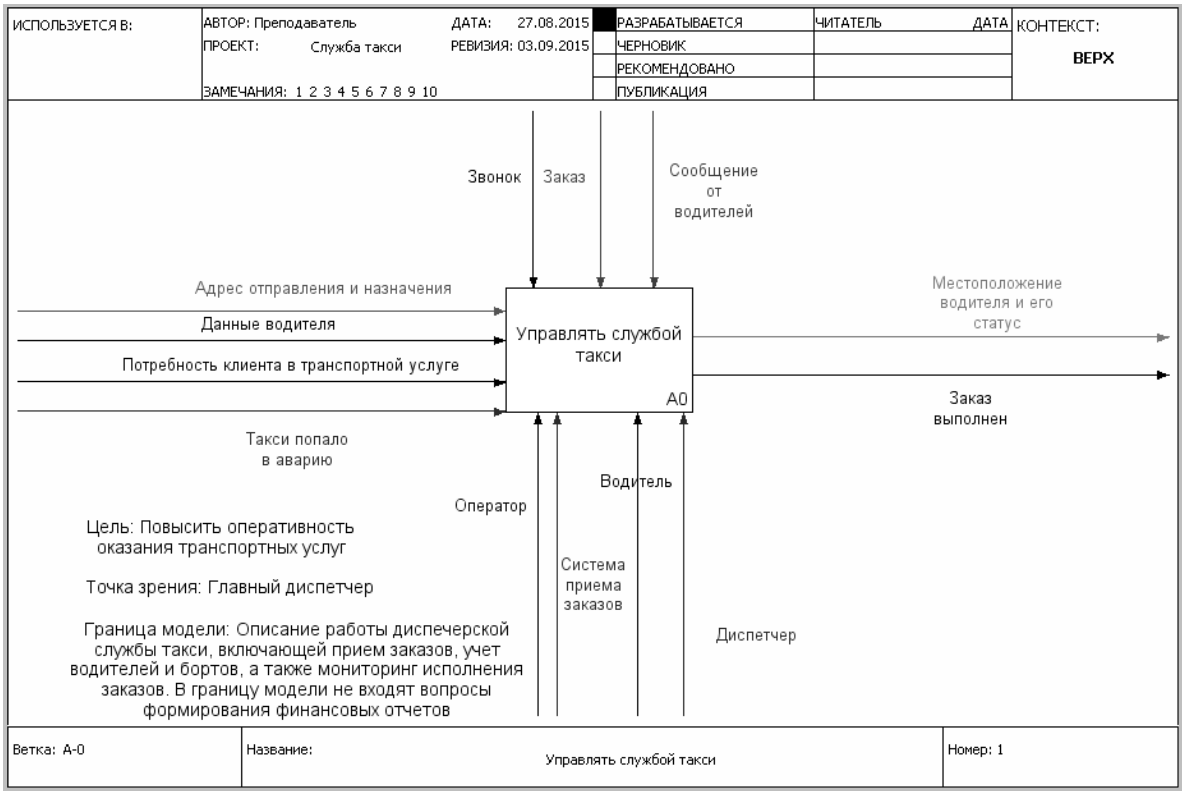


Рисунок 7.4. **Контекстная диаграмма «Управлять службой такси»**

Создание диаграммы декомпозиции

1. Выберите в палитре инструментов кнопку перехода на нижний уровень ▼, в диалоговом окне «**Создание новой диаграммы**» (рис. 7.5) установите количество функциональных блоков 4, укажите тип диаграммы (IDEF0) и нажмите кнопку «**ОК**».

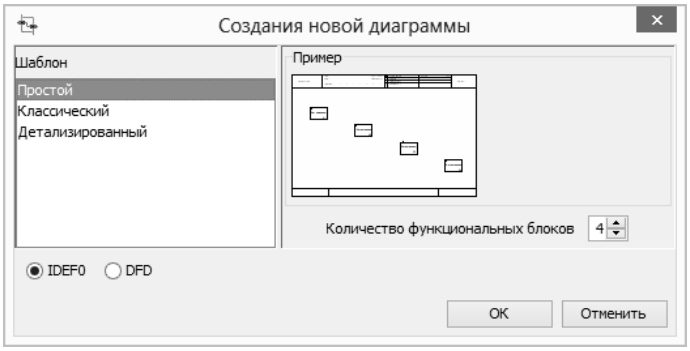


Рисунок 7.5. **Диалоговое окно создания детализирующей диаграммы**

2. Автоматически будет создана диаграмма первого уровня декомпозиции (см. рис. 7.6) с перенесенными в нее потоками родительской диаграммы.

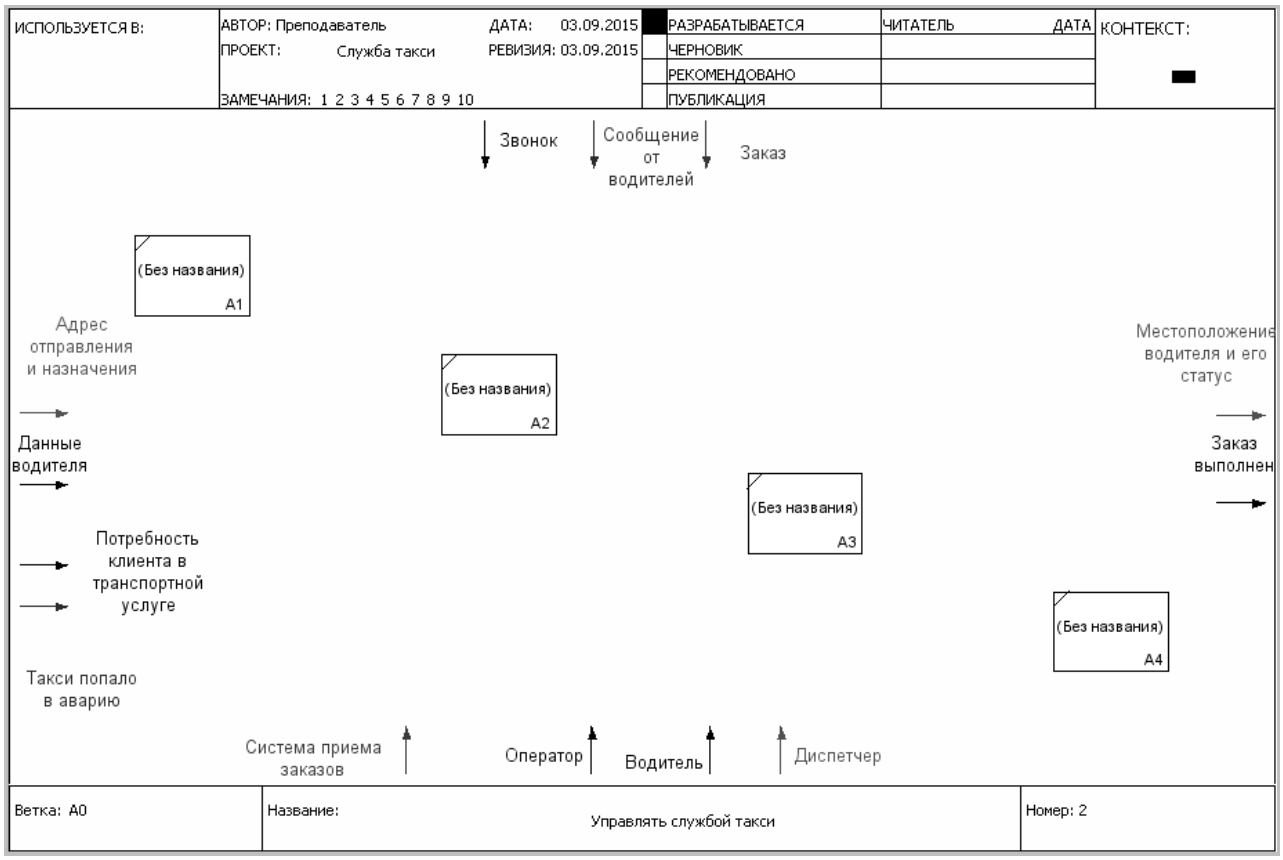


Рисунок 7.6. **Рабочее пространство детализирующей диаграммы**

3. В случае, если на дочернюю диаграмму добавляется внешний поток (т.е. поток, продолжающийся на вышележащей диаграмме), то образуется туннель. Для его устранения необходимо выделить туннель, вызвать контекстное меню и выбрать пункт «**Туннель**» → «**Создать стрелку**» (рис. 7.7). В результате символ туннеля будет удален, а на вышележащей диаграмме появится соответствующий поток.

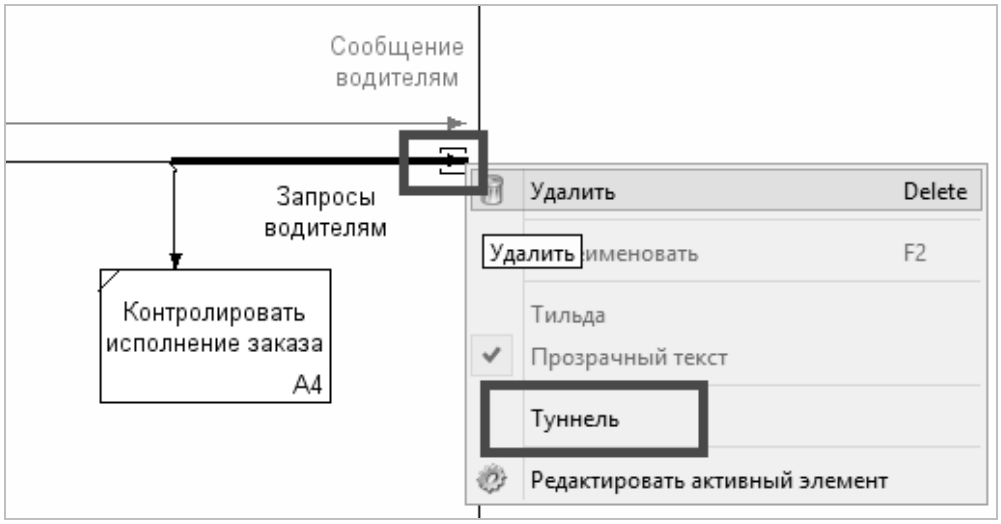


Рисунок 7.7. **Нарушение правила балансировки**

4. На детализированной диаграмме должны отсутствовать туннели (рис. 7.8).

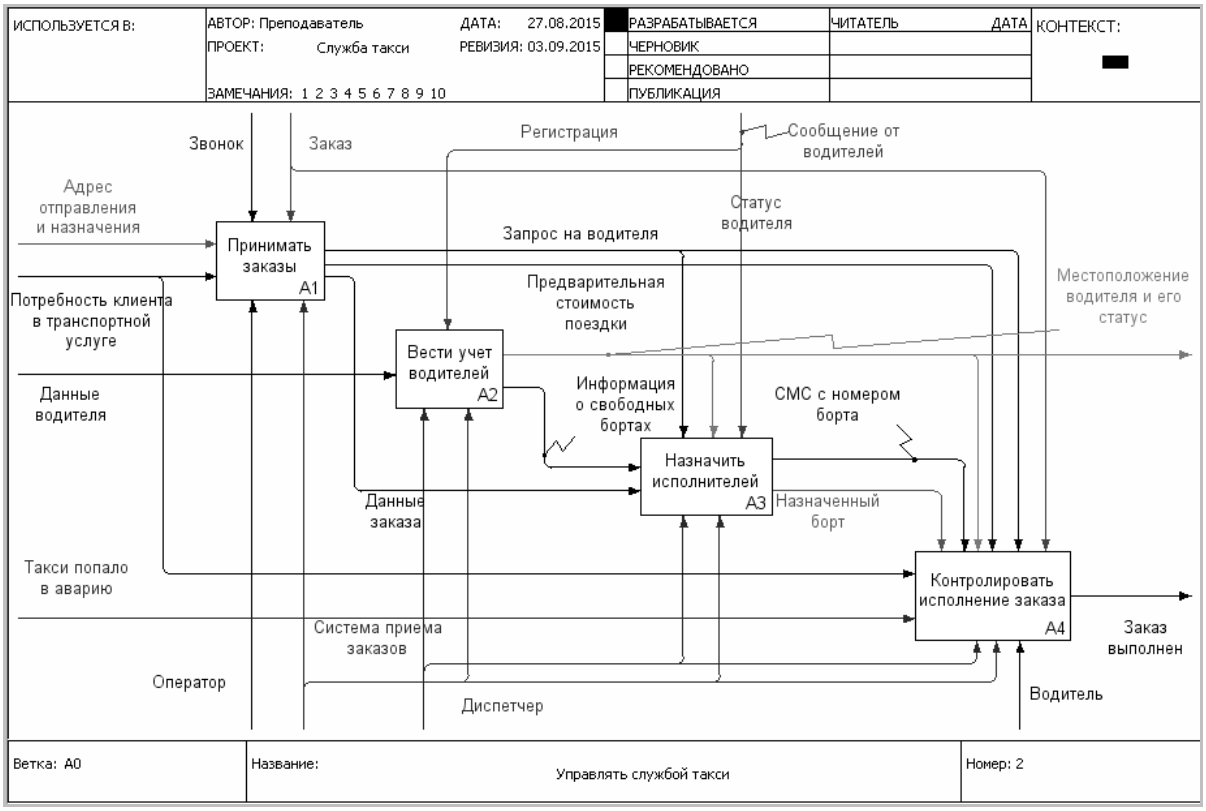


Рисунок 7.8. **Детализированная диаграмма первого уровня**

5. Осуществите построение диаграмм декомпозиции для функциональных блоков согласно представленной иерархии (рис. 7.9)

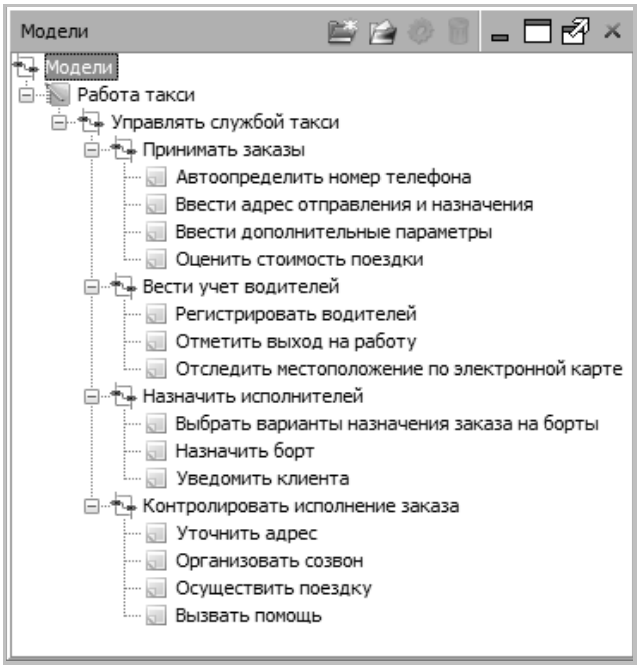


Рисунок 7.9. **Иерархия функциональных блоков модели**

6. В результате должны быть разработаны IDEF0-диаграммы, представленные на рисунках 7.10 – 7.13.

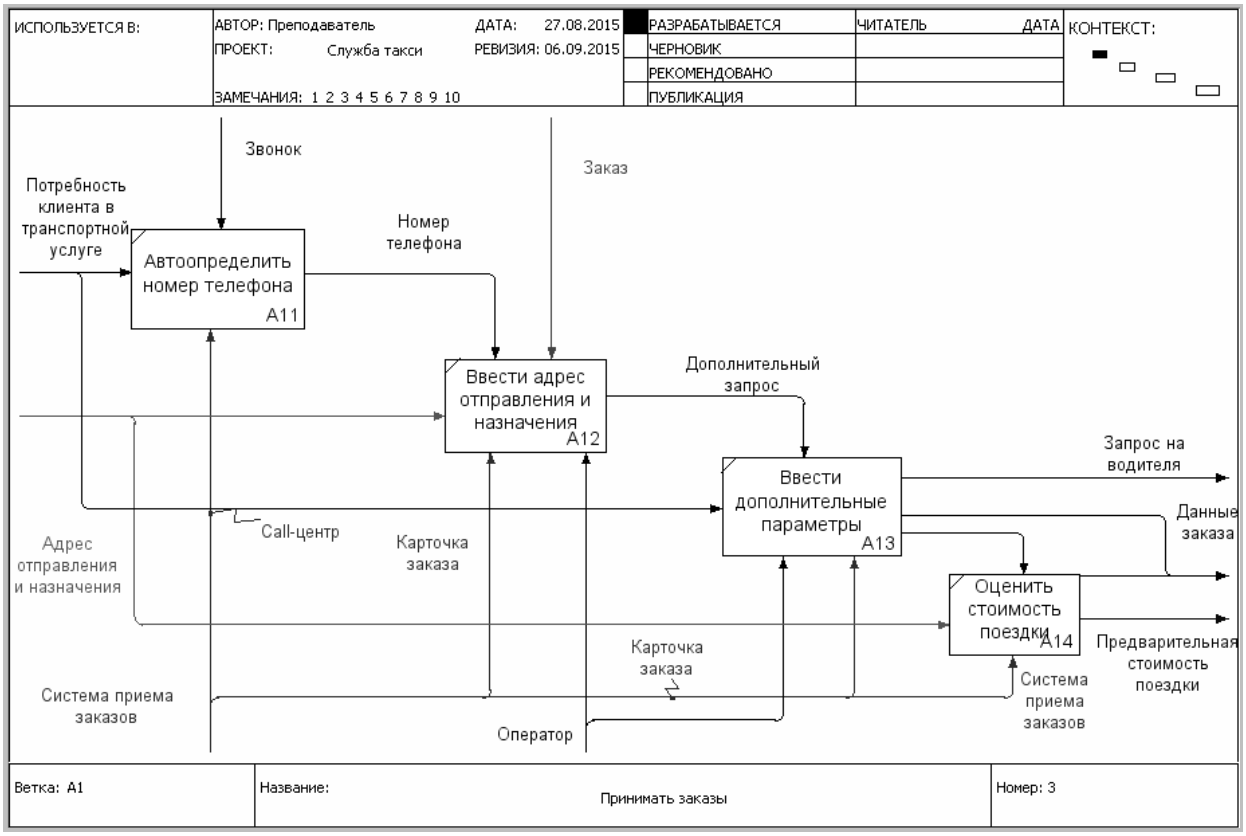


Рисунок 7.10. **Детализированная диаграмма работы «Принимать заказы»**

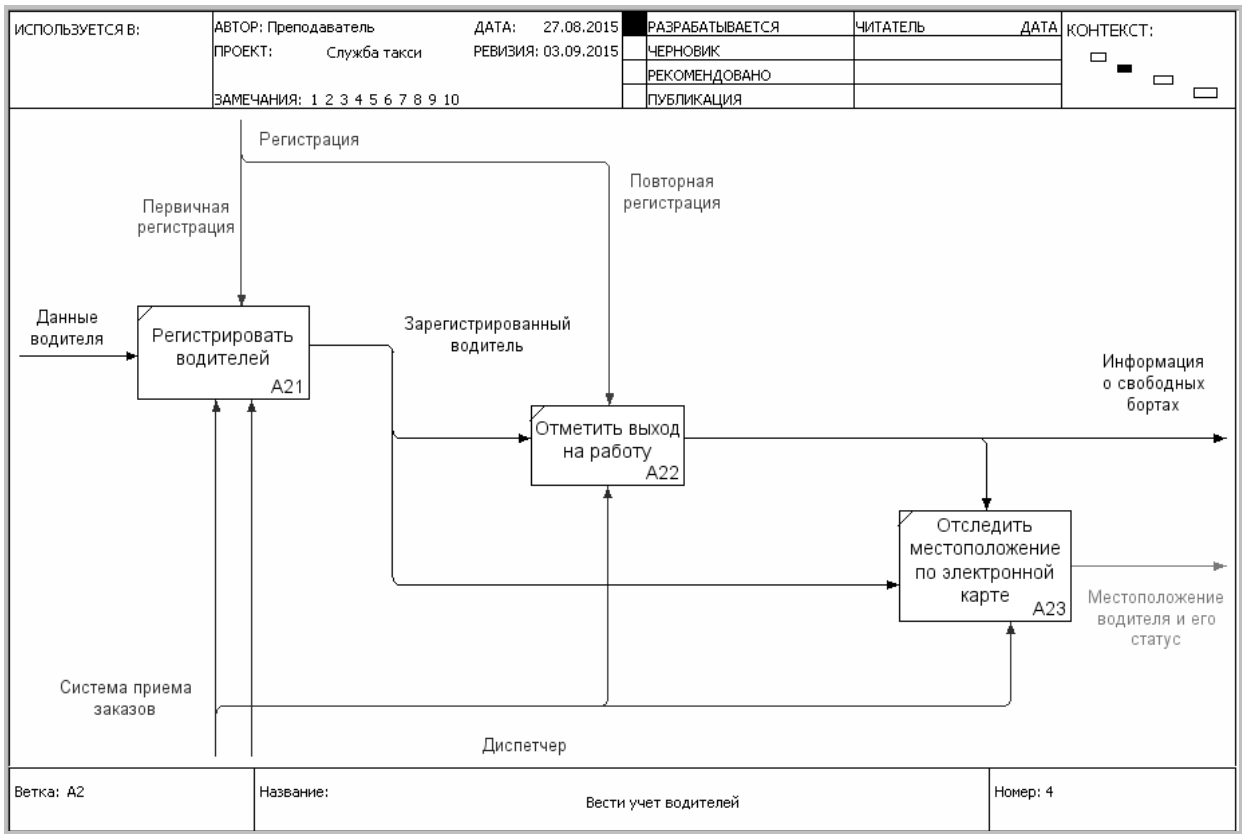


Рисунок 7.11. **Детализированная диаграмма работы «Вести учет водителей»**

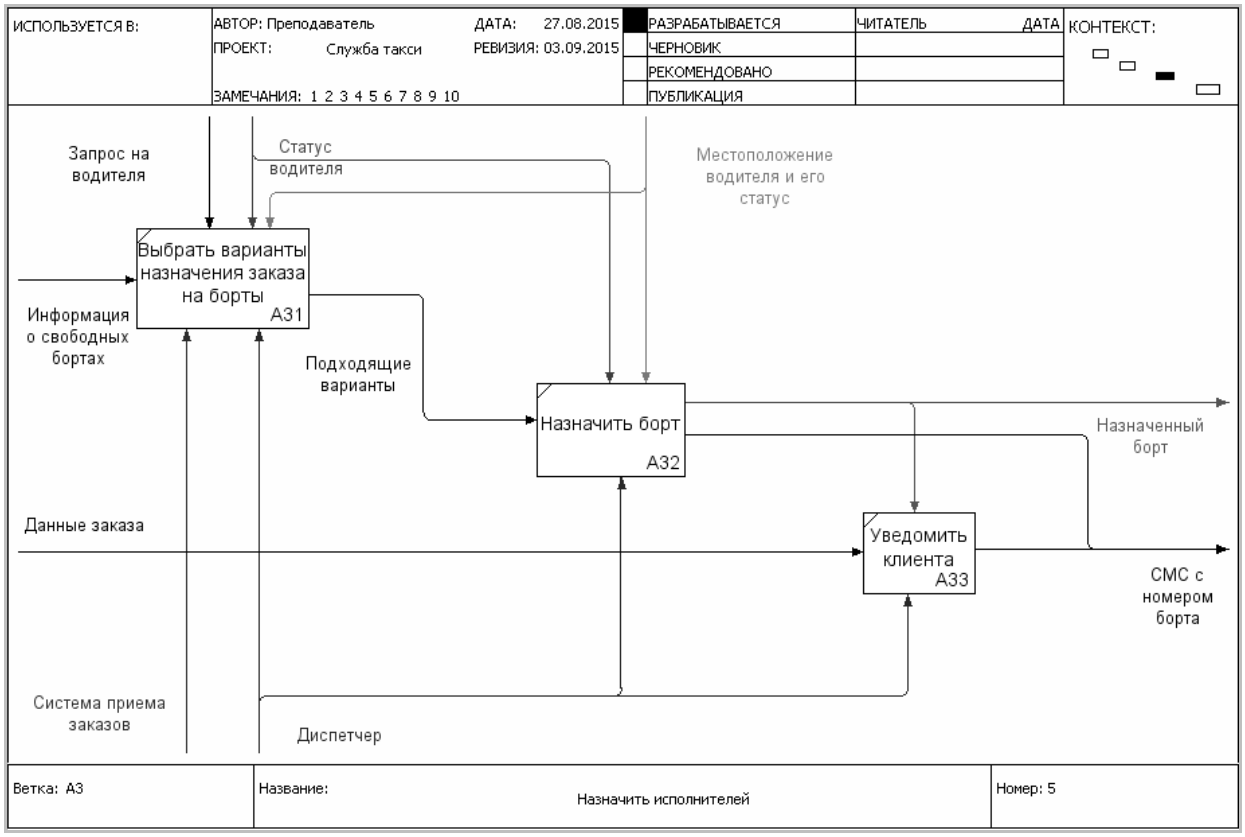


Рисунок 7.12. **Детализированная диаграмма работы «Назначить исполнителей»**

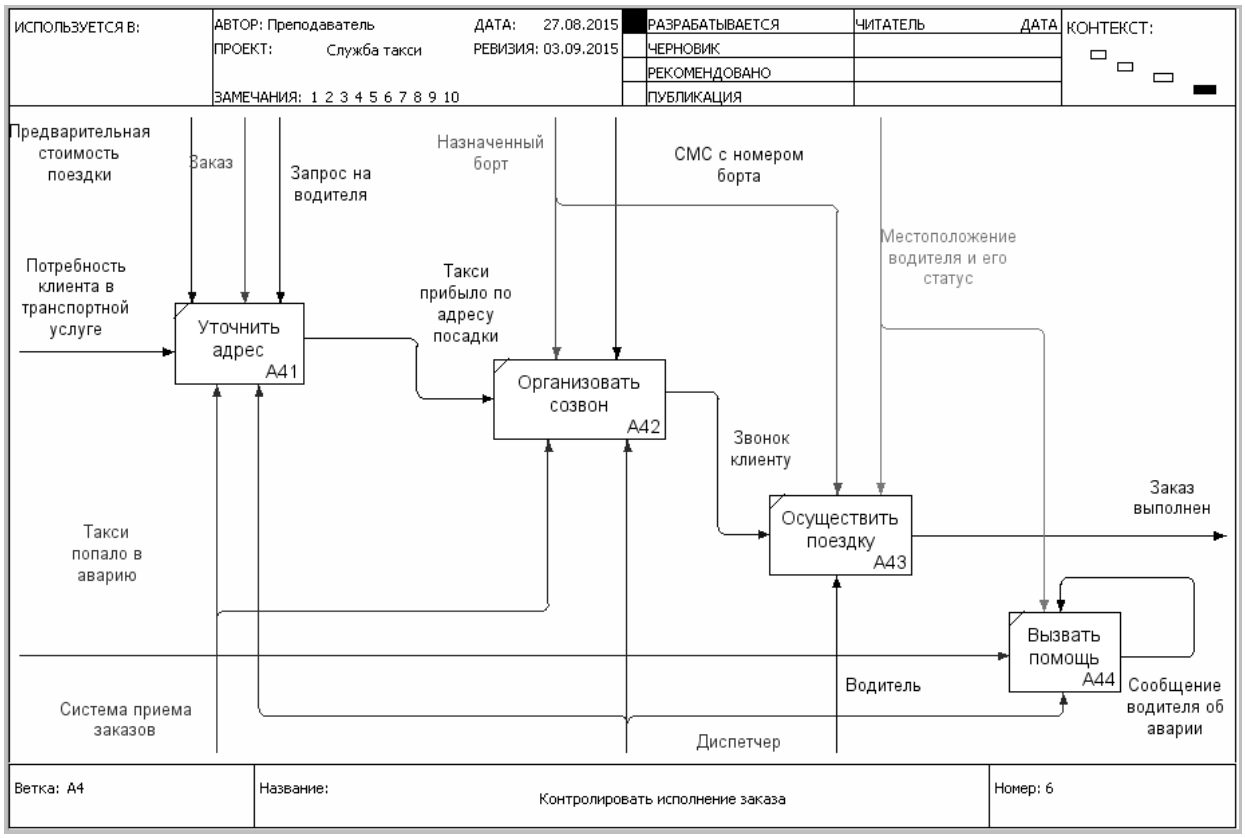


Рисунок 7.13. **Детализированная диаграмма работы «Контролировать исполнение заказа»**

7. В программе Ramus Educational предусмотрена возможность экспорта разработанных диаграмм в виде рисунков формата *.png, *.bmp или *.jpeg. Для этого в главном меню необходимо выбрать команду «**Диаграммы**» → «**Экспортировать как рисунки**». В появившемся окне указывается список экспортируемых рисунков, выбирается их формат и размер, а также путь для сохранения (см. рис. 7.14).

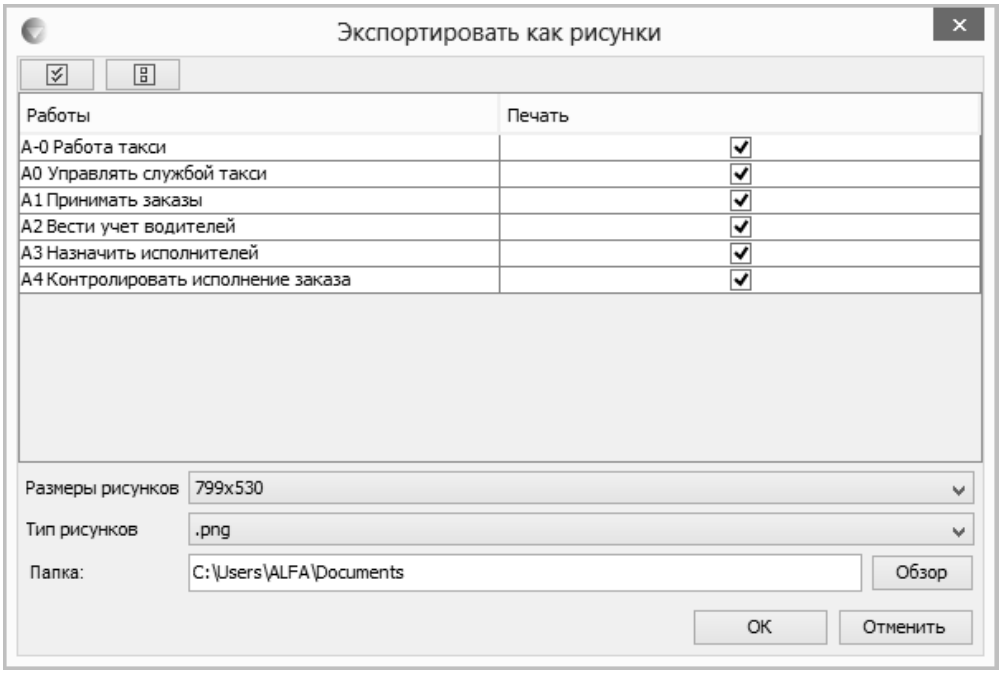


Рисунок 7.14. **Диалоговое окно экспорта диаграмм**

8. Покажите преподавателю разработанные диаграммы.

Содержание работы

1) Ознакомиться с теоретическими вопросами построения IDEF0-диаграммы.
2) Изучить диаграмму IDEF0 для предметной области «Служба такси».
3) Осуществить построение IDEF0-диаграммы с помощью программного средства Ramus Educational согласно индивидуальному заданию (вариант получить у преподавателя).

Требования к отчету

Отчет по лабораторной работе оформляется в печатном виде. Защита работы включает в себя проверку знания студентом теоретического материала, а также практической части лабораторной работы.

Отчет должен включать:

– Разработанные IDEF0-диаграммы.
– Описание разработанных IDEF0-диаграмм.

7.3. Лабораторная работа 2

Тема: Моделирование потоков данных (процессов): DFD. Спецификация процессов.

Цель работы: Изучить теоретические основы создания диаграммы потоков данных, методы построения спецификации процессов. Освоить принципы построения диаграммы потоков данных в Ramus Educational.

Задачи:

1) Ознакомиться с теоретическими вопросами моделирования потоков данных и методами построения спецификации процессов.
2) Изучить DFD-диаграммы для предметной области «Управлять службой такси».
3) Построить с помощью программного средства Ramus Educational DFD-диаграммы согласно индивидуальному заданию (вариант получить у преподавателя).
4) Разработать спецификации процессов согласно выданному индивидуальному заданию.

Краткие теоретические сведения о спецификации процесса

Спецификация процесса используется для описания функционирования процесса в случае отсутствия необходимости детализировать его с помощью DFD (если он недостаточно велик, то его описание может занимать до одной страницы текста). Фактически, спецификация представляет собой алгоритмы описания задач, выполняемых процессами. Множество всех спецификаций процессов является полной спецификацией системы. Спецификации процесса содержат номер и/или имя процесса, списки входных и выходных данных и тело (описание) процесса, являющееся спецификацией конкретного алгоритма или операции, трансформирующей входные потоки данных в выходные.

Независимо от используемой нотации спецификация процесса должна начинаться с ключевого слова (например, @СПЕЦПРОЦ). Требуемые входные и выходные данные должны быть специфицированы следующим образом:

@ВХОД = <имя символа данных>

@ВЫХОД = <имя символа данных>

@ВХОДВЫХОД = <имя символа данных>,

где имя символа данных – соответствующее имя словаря 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 компонентов хранилищ.

Разработка DFD-модели в системе Ramus Educational

1. Запустите программу Ramus Educational.
2. Выбор флага «**Создать новый файл**» и нажатие кнопки «**ОК**» осуществляет запуск мастера проекта.
   - На первом шаге (рис. 7.15) необходимо внести сведения об авторе, названии проекта и модели и выбрать тип нотации модели – в данном случае – **DFD**.

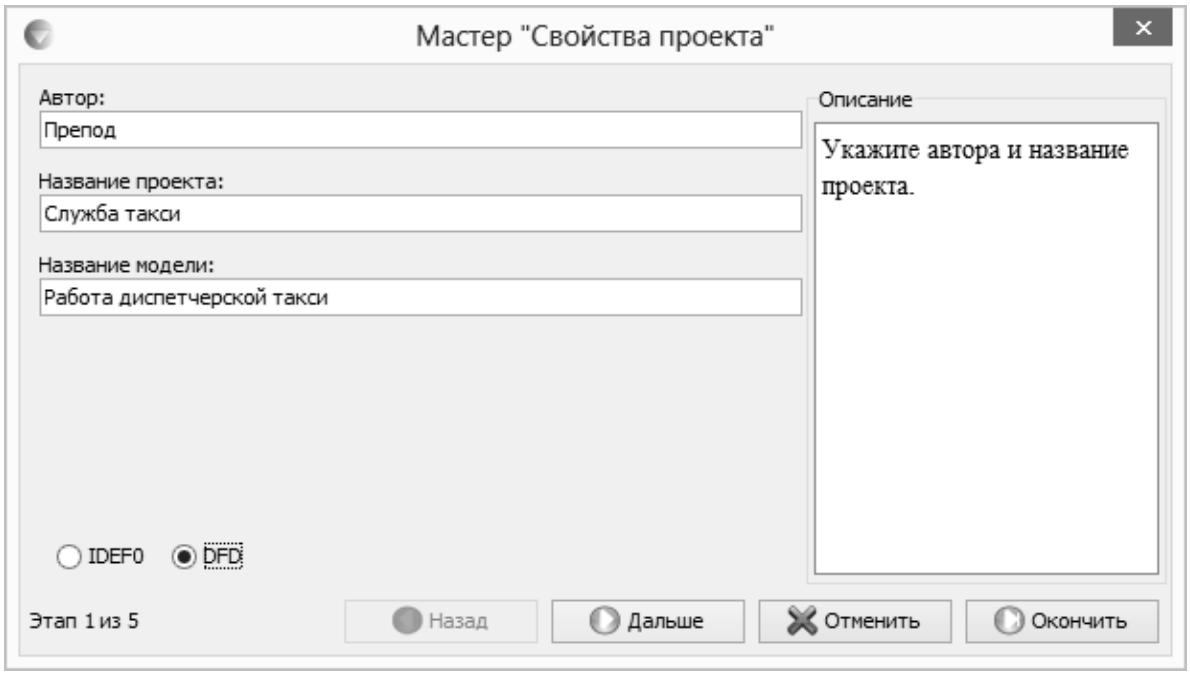


Рисунок 7.15. **Окно мастера создания проекта. Шаг 1**

- Второй шаг позволяет ввести название организации, для которой данный проект создается.
- Третий шаг – необходимо дать тезисное описание проекта.
- На четвертом шаге предлагается создать несколько основных классификаторов (рис. 7.16).

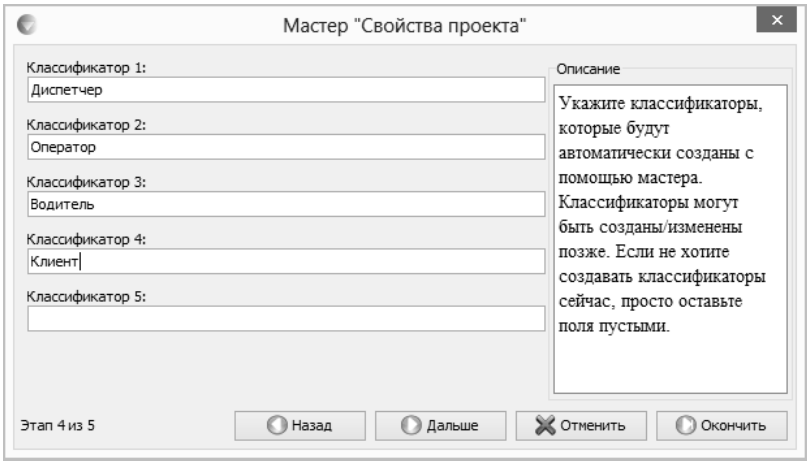


Рисунок 7.16. **Окно мастера создания проекта. Шаг 4**

– На пятом, заключительном этапе, предлагается выбрать из созданных классификаторов те, элементы которых будут содержаться в перечне собственников процессов (рис. 7.17).

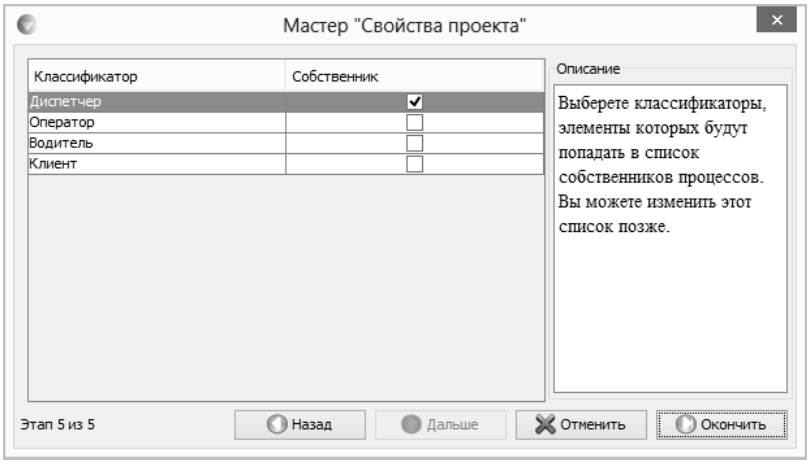


Рисунок 7.17. **Окно мастера создания проекта. Шаг 5**

При необходимости можно закрыть мастер, нажав на любом из этапов кнопку «**Окончить**».

После завершения работы мастера, откроется рабочее пространство «**Диаграммы**», в котором можно приступить к построению графической модели. На панели инструментов, в верхней части окна рабочего пространства программы, содержатся элементы диаграммы потоков данных в нотации Gane-Sarson.

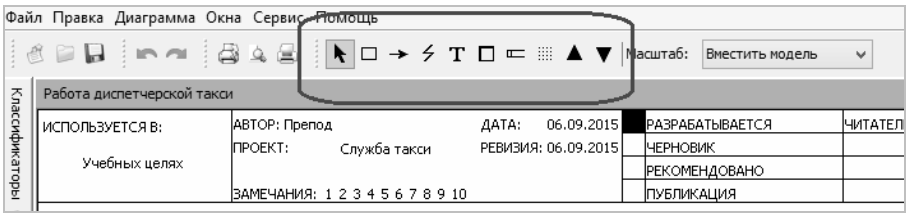


Рисунок 7.18. **Элементы DFD в нотации Gane-Sarson**

3. Сохраните созданную модель, выбрав опцию меню «**Файл**» → «**Сохранить как**».

Создание контекстной диаграммы

1. На панели выберите инструмент создания процесса ▭ и укажите мышью в произвольном месте области построения диаграммы месторасположение нового процесса.

2. Выделив процесс, выберите в контекстном меню опцию «**Редактировать активный компонент**». В появившемся диалоговом окне на вкладке «**Название**» присвойте процессу имя «*Управлять службой такси*»; на вкладке «**Тип функционального блока**» укажите тип элемента – «*Процесс*» (рис. 7.19).

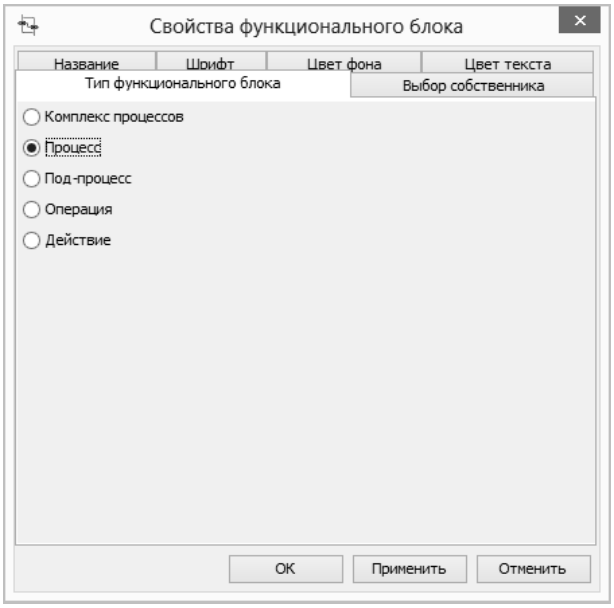


Рисунок 7.19. **Окно «Свойства функционального блока»**

3. На панели инструментов выберите инструмент создания внешней сущности ◻ и мышью укажите произвольное ее месторасположение в области построения.

4. В контекстном меню созданной внешней сущности выберите опцию «**Редактировать активный компонент**», на вкладке «**Объект**» нажмите «**Задать DFD объект**», после чего в появившемся окне выделите классификатор «*Диспетчер*» (рис. 7.20) и нажмите «**ОК**».

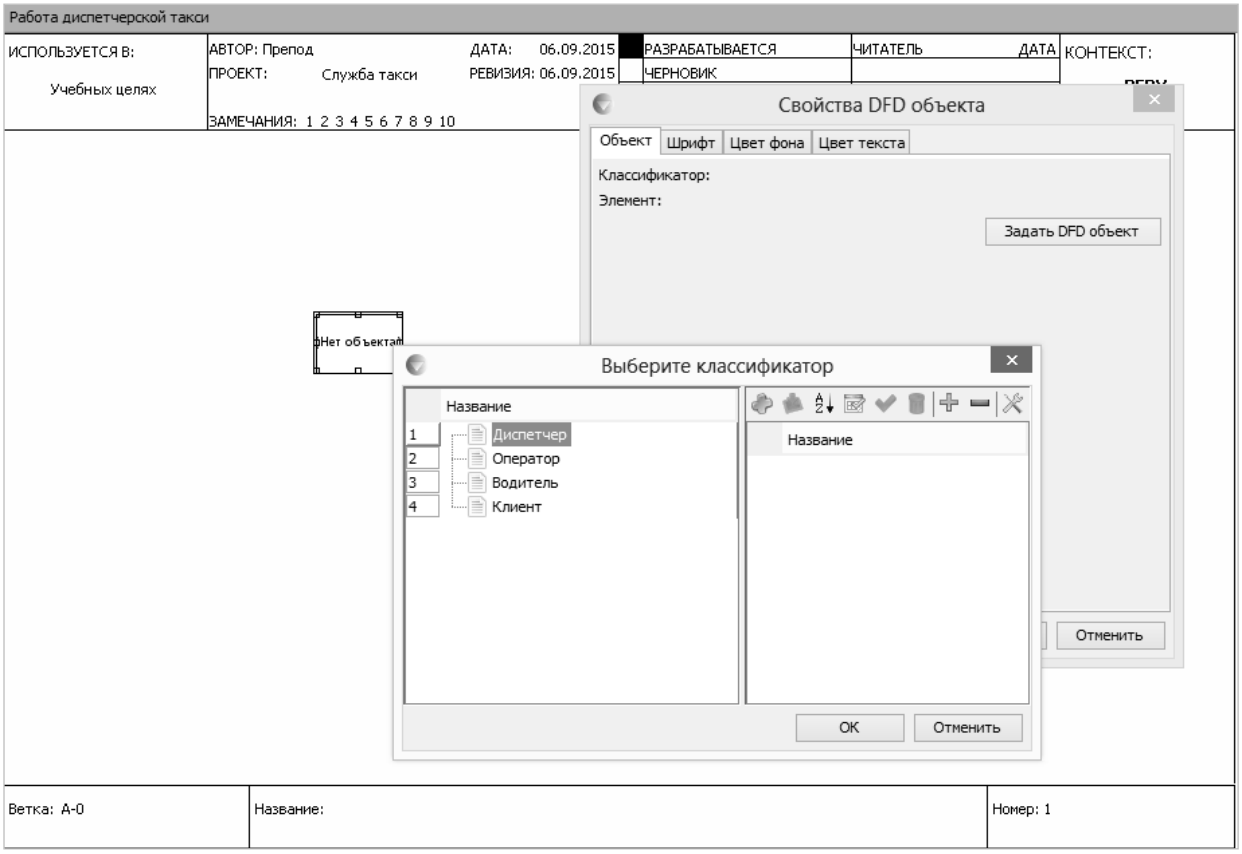


Рисунок 7.20. **Назначение классификатора**

5. Выбрав на панели инструментов элемент →, создайте стрелки на контекстной диаграмме согласно рисунку 7.21.

6. В результате должна получиться контекстная диаграмма, показанная на рисунке 7.21.

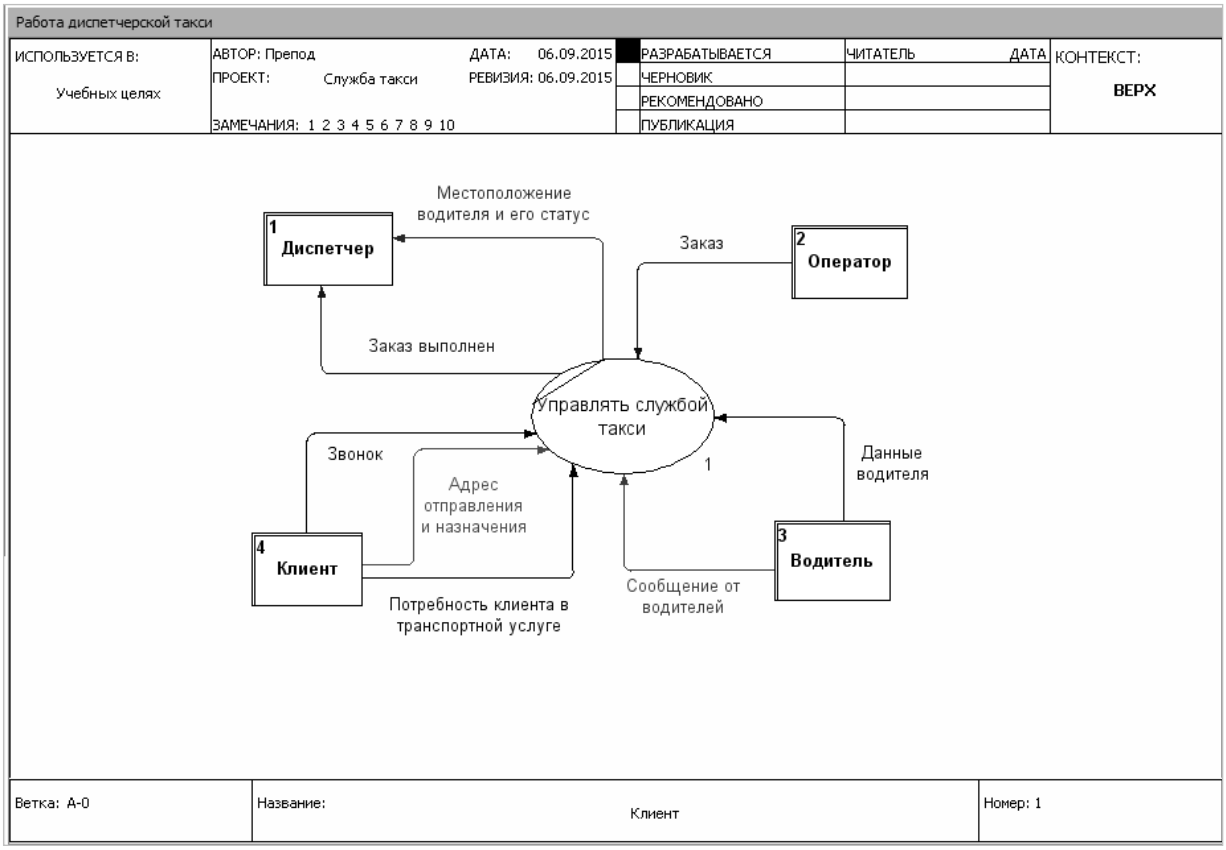


Рисунок 7.21. **Контекстная DFD-диаграмма**

Создание диаграммы декомпозиции

1. Выберите в палитре инструментов кнопку перехода на нижний уровень ▼, в диалоговом окне «**Создание новой диаграммы**» (рис. 7.22) установите количество функциональных блоков (**3**), укажите тип диаграммы (**DFD**) и нажмите кнопку «**OK**».

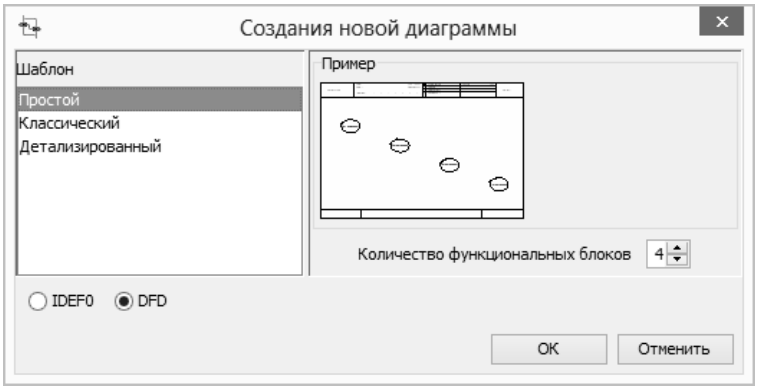


Рисунок7.22. **Диалоговое окно создания диаграммы декомпозиции**

2. Автоматически будет создана диаграмма декомпозиции (рис.7.23). В нее будут перенесены все потоки родительской диаграммы.

3. Двойным щелчком мыши на одном из процессов, или выбрав в контекстном меню процесса пункт «**Редактировать активный элемент**», откройте окно редактирования свойств и задайте процессу имя. Повторите операцию для оставшихся процессов.

4. Выделив необходимый поток (стрелку) и, удерживая левую клавишу мыши, соедините его требуемым образом с соответствующим процессом. В результате должна получиться диаграмма декомпозиции (рис. 7.24).

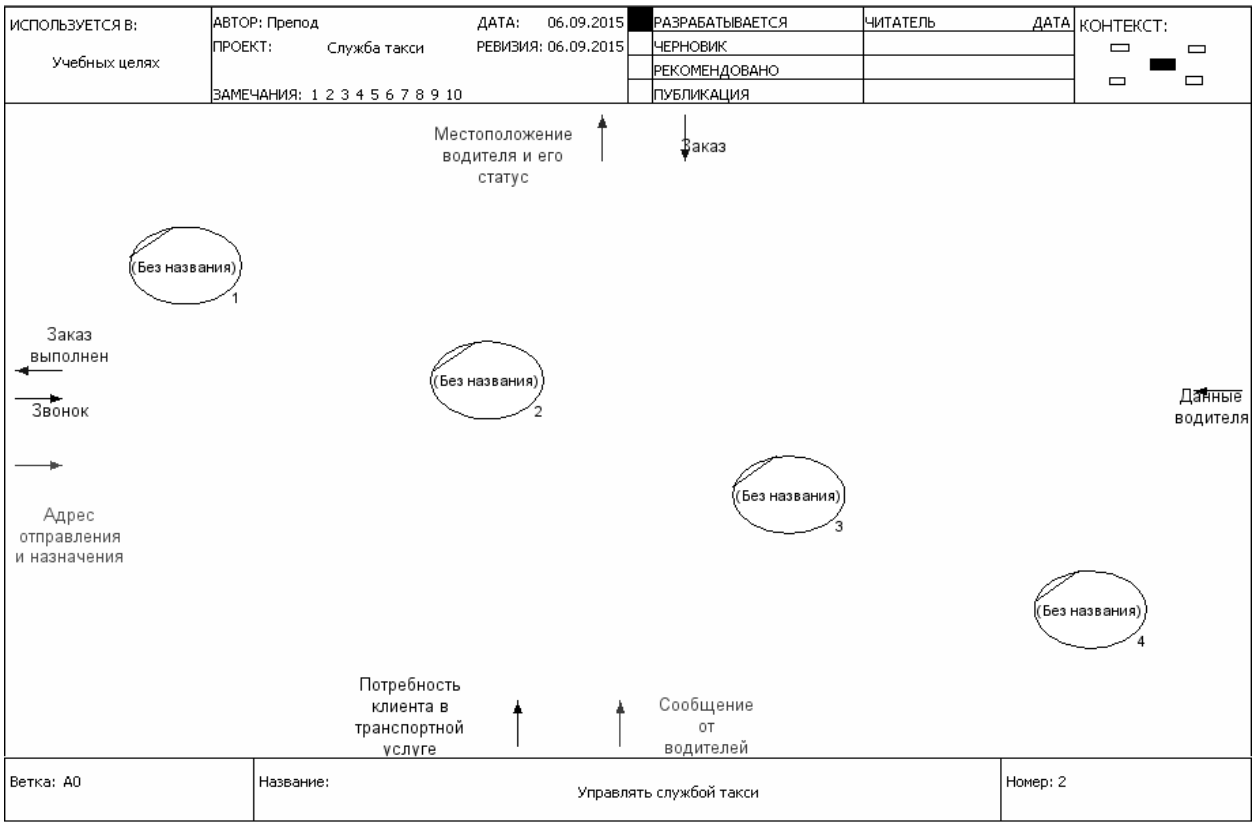


Рисунок 7.23. **Рабочее пространство диаграммы декомпозиции**

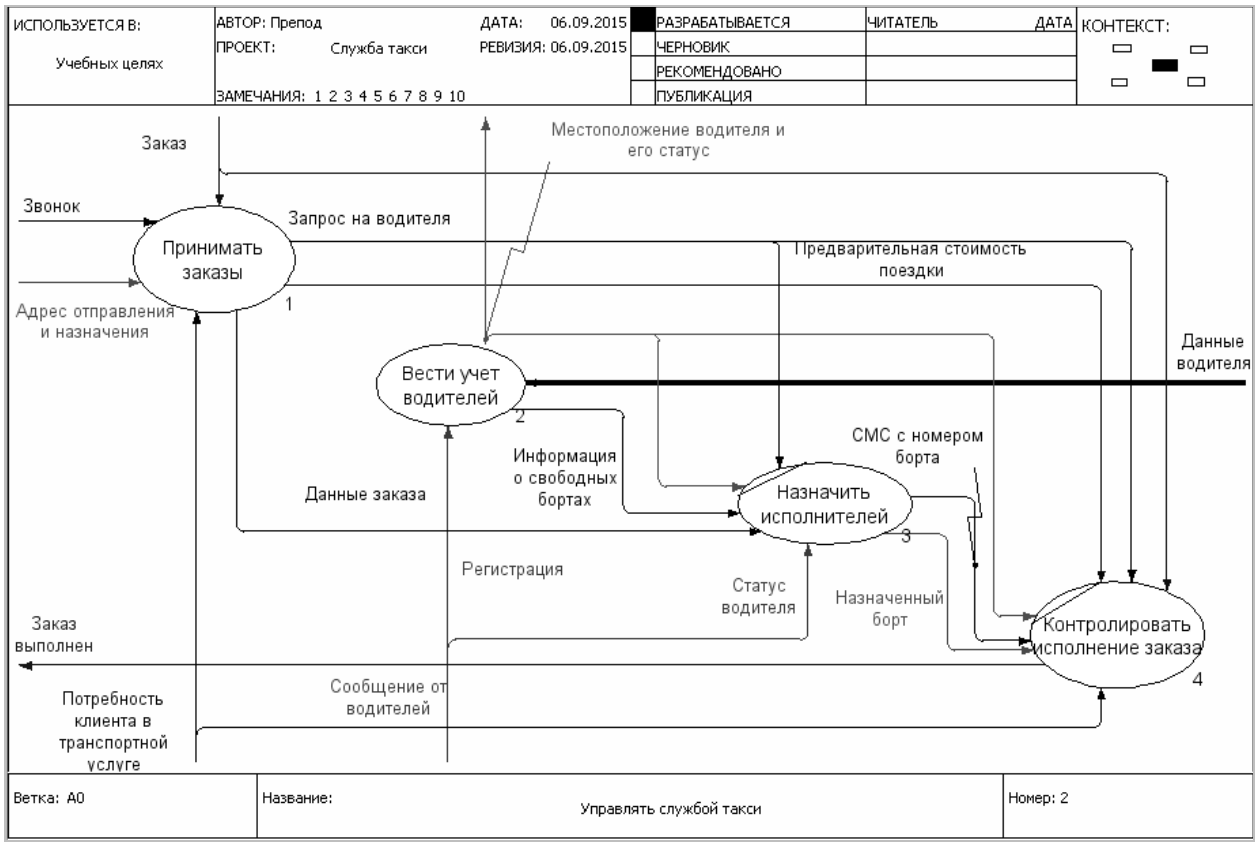


Рисунок 7.24. **Диаграмма декомпозиции первого уровня**

6. Детализируйте процесс «*Принимать заказы*».

7. Выберите инструмент создания хранилища и укажите курсором его местоположение в области построения диаграммы.

8. В контекстном меню хранилища выберите «**Редактировать активный элемент**», затем в открывшемся окне – «**Задать DFD объект**» и определите требуемый классификатор.

9. Если классификатор отсутствует, его можно добавить командой «**Создать элемент**» контекстного меню списка классификаторов (рис. 7.25), или вводом имени непосредственно в список панели инструментов классификаторы (рис. 7.26).

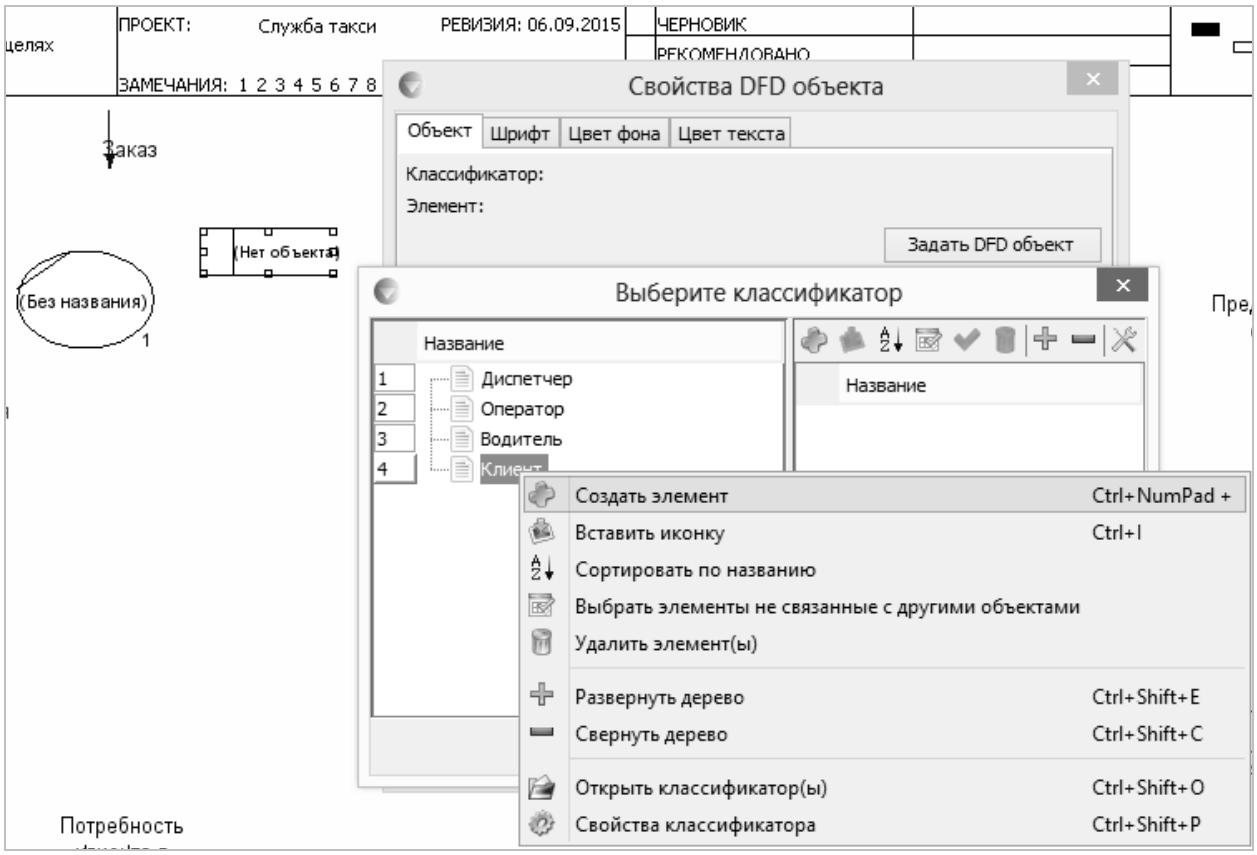


Рисунок 7.25. **Добавление нового классификатора через контекстное меню**

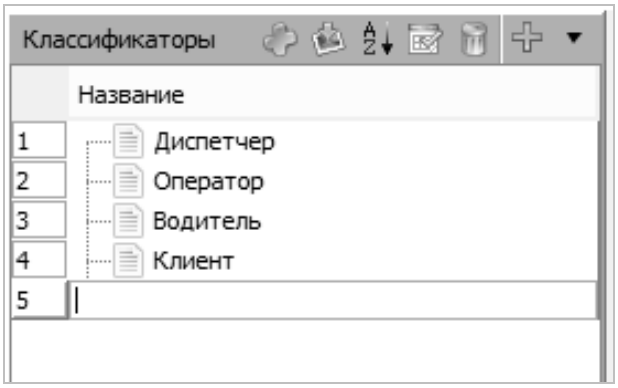


Рисунок 7.26. **Добавление нового классификатора в окне панели инструментов «*Классификаторы*»**

10. Аналогичным образом могут быть построены DFD-диаграммы, представленные на рисунках 7.27 – 7.30.

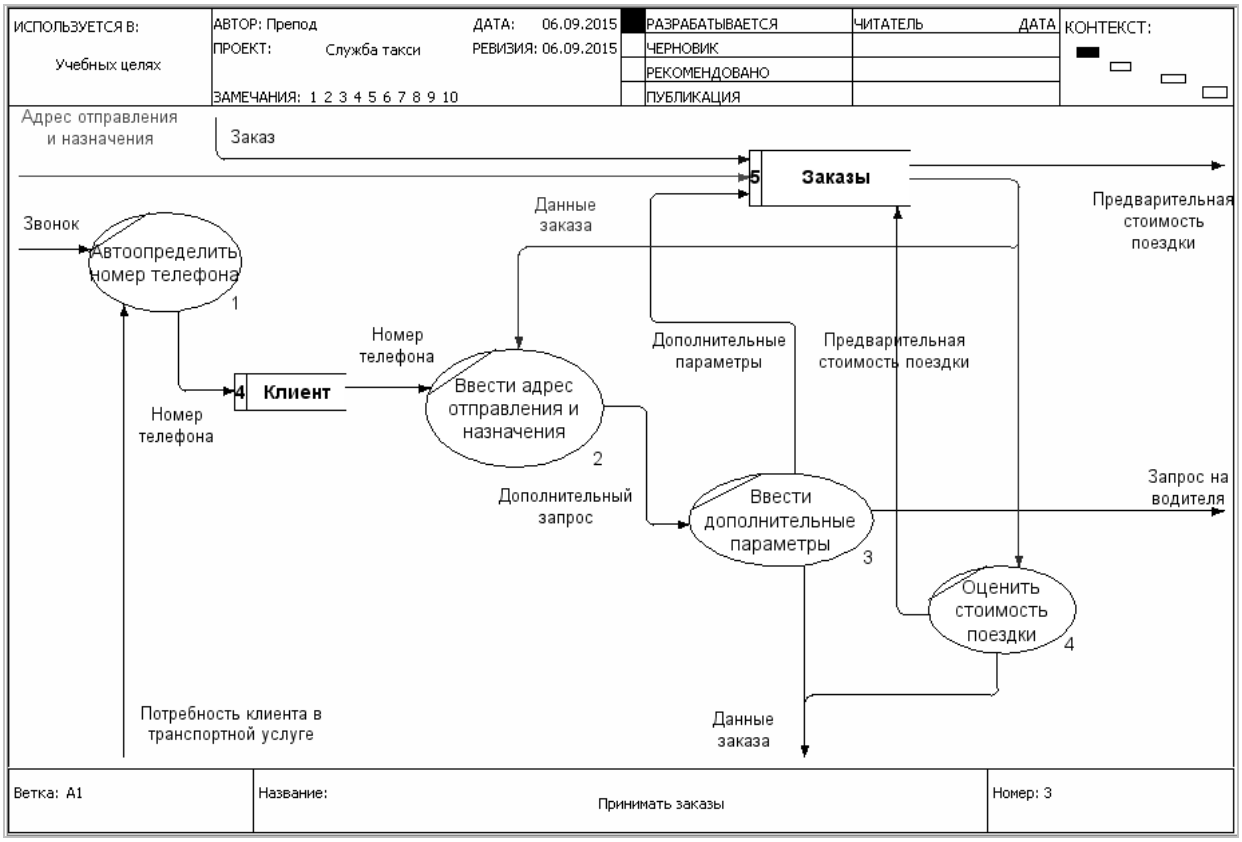


Рисунок 7.27. **Детализированный процесс «Принимать заказы»**

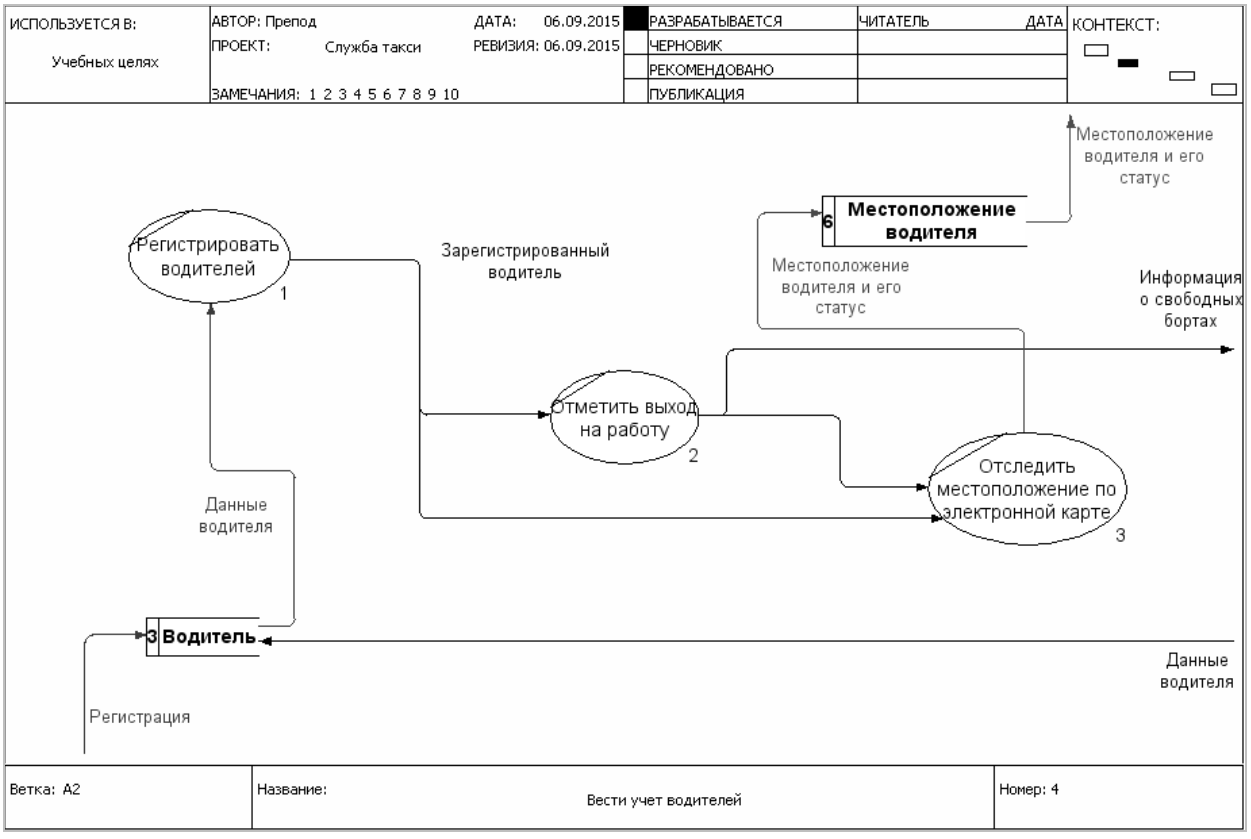


Рисунок.7.28. **Детализированный процесс «Вести учет водителей»**

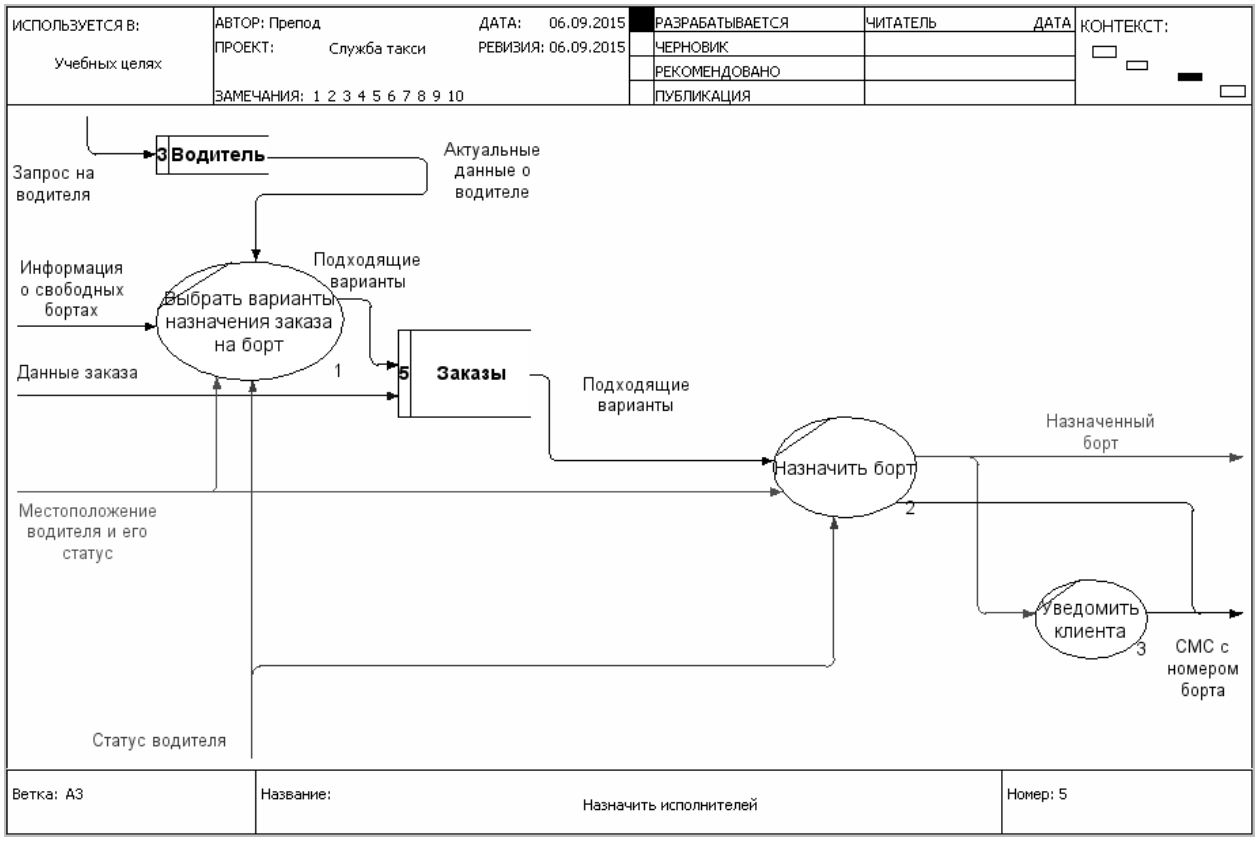


Рисунок 7.29. **Детализированный процесс «Назначить исполнителей»**

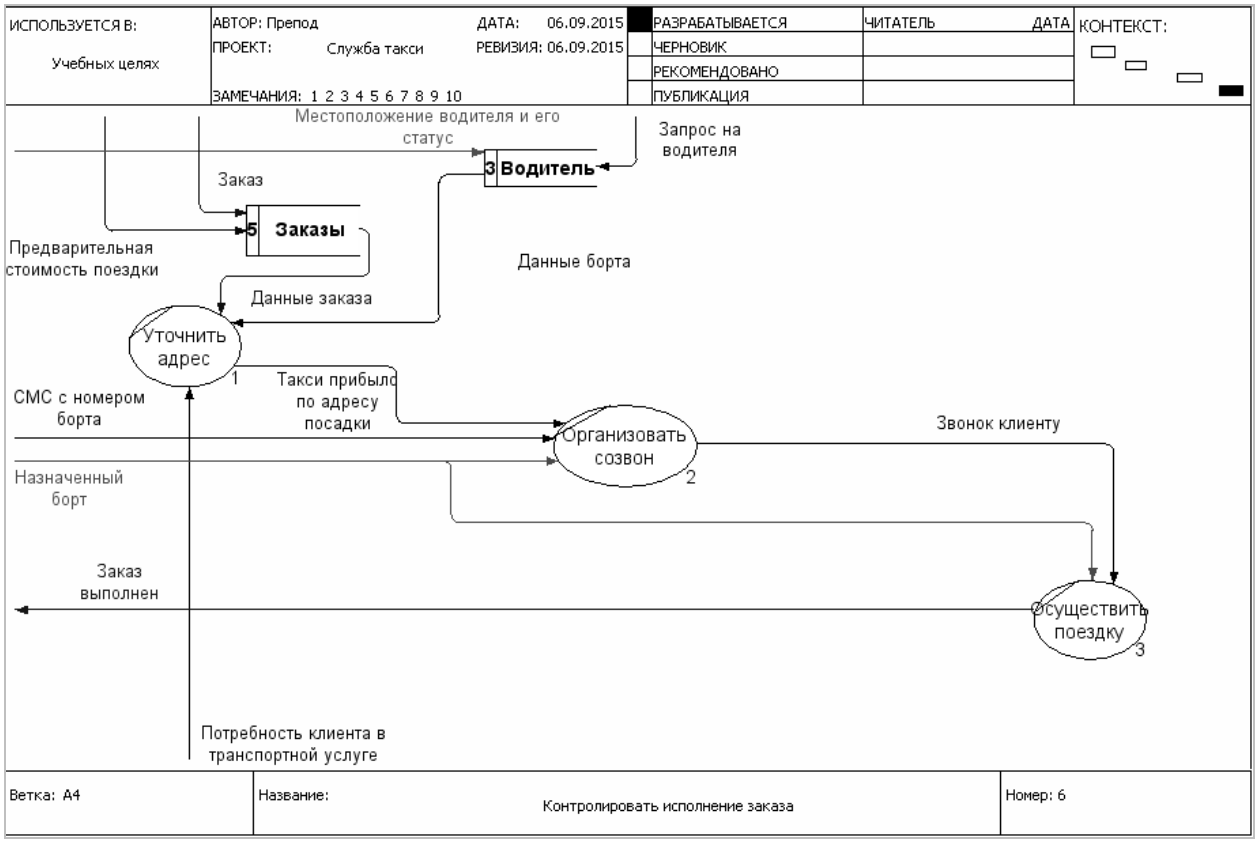


Рисунок 7.30. **Детализированный процесс «Контролировать исполнение заказа»**

11. Экспортируйте полученные диаграммы в виде графических рисунков (главное меню «**Диаграммы**» → «**Экспортировать как рисунки**»).

Спецификации процессов

Ниже приведен пример спецификации процессов

1. Спецификация процесса 1.1 «Принимать заказы»

Спецификация процесса 1.1.1 «Автоопределить номер телефона»

@ВХОД = ЗВОНОК
@ВХОД = ПОТРЕБНОСТЬ КЛИЕНТА В ТРАНСПОРТНОЙ УСЛУГЕ
@ВЫХОД = НОМЕР ТЕЛЕФОНА
@СПЕЦПРОЦ 1.1.1 = АВТООПРЕДЕЛИТЬ НОМЕР ТЕЛЕФОНА
ВЫПОЛНИТЬ
принять ЗВОНОК клиента на получение транспортной услуги
определить НОМЕР ТЕЛЕФОНА КЛИЕНТА, с которого сделан ЗВОНОК
обновить ДАННЫЕ О КЛИЕНТЕ
создать КАРТОЧКУ КЛИЕНТА
@КОНЕЦ СПЕЦИФИКАЦИИ ПРОЦЕССА 1.1.1

Спецификация процесса 1.1.2

@ВХОД = НОМЕР ТЕЛЕФОНА
@ВХОД = ДАННЫЕ ЗАКАЗА
@ВЫХОД = ДОПОЛНИТЕЛЬНЫЙ ЗАПРОС на параметры заказа
@СПЕЦПРОЦ 1.1.2 = ВВЕСТИ АДРЕС ОТПРАВЛЕНИЯ И НАЗНАЧЕНИЯ
проверить наличие ОФОРМЛЕННОЙ КАРТОЧКИ ЗАКАЗА
ЕСЛИ ОФОРМЛЕННЫЕ КАРТОЧКИ ЗАКАЗА есть,
ТО увеличить количество записей в базы данных «ЗАКАЗЫ»
КОНЕЦЕСЛИ
в КАРТОЧКУ ЗАКАЗА ввести данные ЗАКАЗА: АДРЕС ОТПРАВЛЕНИЯ, АДРЕС НАЗНАЧЕНИЯ, ВРЕМЯ ИСПОЛНЕНИЯ ЗАКАЗА
@КОНЕЦ СПЕЦИФИКАЦИИ ПРОЦЕССА 1.1.2

Спецификация процесса 1.1.3 «Ввести дополнительные параметры»

@ВХОД = ДОПОЛНИТЕЛЬНЫЙ ЗАПРОС
@ВХОД = ПОТРЕБНОСТЬ КЛИЕНТА В ТРАНСПОРТНОЙ УСЛУГЕ
@ВЫХОД = ДОПОЛНИТЕЛЬНЫЕ ПАРАМЕТРЫ
@ВЫХОД = ЗАПРОС НА ВОДИТЕЛЯ
@ВЫХОД = ДАННЫЕ ЗАКАЗА
@СПЕЦПРОЦ 1.1.3 = ВВЕСТИ ДОПОЛНИТЕЛЬНЫЕ ПАРАМЕТРЫ
ВЫПОЛНИТЬ
открыть КАРТОЧКУ ЗАКАЗА
ЕСЛИ ДОПОЛНИТЕЛЬНЫЕ ПАРАМЕТРЫ ИСПОЛНЕНИЯ ЗАКАЗА есть,
ТО внести требования в КАРТОЧКУ ЗАКАЗА
КОНЕЦЕСЛИ
обновить данные КАРТОЧКИ ЗАКАЗА
@КОНЕЦ СПЕЦИФИКАЦИИ ПРОЦЕССА 1.1.4

Спецификация процесса 1.1.4 «Оценить стоимость поездки»

@ВХОДВЫХОД = ДАННЫЕ ЗАКАЗА
@ВЫХОД = ПРЕДВАРИТЕЛЬНАЯ СТОИМОСТЬ ПОЕЗДКИ
@СПЕЦПРОЦ 1.1.4 = ОЦЕНИТЬ СТОИМОСТЬ ПОЕЗДКИ
ВЫПОЛНИТЬ
открыть КАРТОЧКУ ЗАКАЗА
просмотреть АДРЕС ОТПРАВЛЕНИЯ
просмотреть АДРЕС НАЗНАЧЕНИЯ
открыть тарифы поездок
рассчитать ПРЕДВАРИТЕЛЬНУЮ СТОИМОСТЬ ПОЕЗДКИ
озвучить КЛИЕНТУ ПРЕДВАРИТЕЛЬНУЮ СТОИМОСТЬ ПОЕЗДКИ
@КОНЕЦ СПЕЦИФИКАЦИИ ПРОЦЕССА 1.1.4

Содержание работы

1) Ознакомиться с теоретическими вопросами моделирования потоков данных и методами построения спецификации процессов.
2) Изучить DFD-диаграммы для предметной области «Управлять службой такси».
3) Построить с помощью программного средства Ramus Educational DFD-диаграммы согласно индивидуальному заданию (вариант получить у преподавателя).
4) Разработать спецификации процессов согласно выданному индивидуальному заданию.

Требования к отчету

Отчет по лабораторной работе оформляется в печатном виде. Защита работы включает в себя проверку знания студентом теоретического материала, а также практической части лабораторной работы.

Отчет должен включать:

– разработанные DFD-диаграммы согласно индивидуальному заданию;
– описание разработанных DFD-диаграмм;
– спецификации процессов.

7.4. Лабораторная работа 3

Тема: Объектно-ориентированное моделирование с помощью языка UML.

Цель работы: Изучить теоретические основы методологии моделирования UML. Освоить принципы построения UML-диаграмм на примере программного продукта StarUML.

Задачи:

1) Ознакомиться с теоретическими аспектами методологии объектно-ориентированной методологии UML.
2) Разработать все UML-диаграммы согласно индивидуальной предметной области (вариант задания получить у преподавателя).

7.4.1. Краткий обзор CASE-средств для построения UML-диаграмм

UML-диаграммы можно создавать с помощью различных средств, например, таких как MS Visio, Visual Paradigm for UML, StarUML, Magic Draw, Rational и пр.

Возможности MS Visio в области создания UML-диаграмм достаточно сильно зависит от версии и пакета. Наиболее корректные UML-диаграммы можно создавать в MS Visio Professional 2003 (при создании нового файла обязательно указать, что тип создаваемой модели UML). В версии 2013 в пакете MS Visio Standard UML-диаграммы отсутствуют, диаграммы можно создавать только в MS Visio Professional. Кроме того, следует отметить, что в MS Visio Professional 2013 присутствуют не все диаграммы UML.

Visual Paradigm for UML 7.1 (VP-UML) представляет собой CASE-средство визуального UML-моделирования. VP 7.1 предлагает объектно-ориентированный подход к анализу и проектированию систем различной сложности и позволяет создавать множество типов диаграмм в полностью визуализированной среде разработки посредством простых drag&drop операций. Условно свободно распространяемое ПО (не для коммерческих целей, только для учебных).

Программная платформа StarUML имеет свободную лицензию и доступна для установки с официального сайта. StarUML поддерживает одиннадцать различных типов диаграмм, принятых в нотации UML 2.0, а также подход MDA (модельно-настраиваемая архитектура), предлагает настройку параметров пользователя для адаптации среды разработки, поддерживает расширения, предоставляет различного рода модули, расширяющие возможности StarUML.

7.4.2. Пример разработки UML-диаграмм в StarUML

В данном разделе приводятся UML-диаграммы с использованием StarUML.

Краткие сведения о StarUML

Основная структурная единица в StarUML – это проект. Проект сохраняется в одном файле в формате XML с расширением .UML. Проект может содержать одну или несколько моделей и различные представления этих моделей (View) – визуальные выражения информации, содержащейся в моделях. Каждое представление модели содержит диаграммы – визуальные образы, отображающие определенные аспекты модели.

Новый проект будет автоматически создан при запуске программы StarUML. При этом вам будет предложено в диалоговом окне выбрать один из подходов (Approaches), поддерживаемых StarUML (см. рис. 7.31).

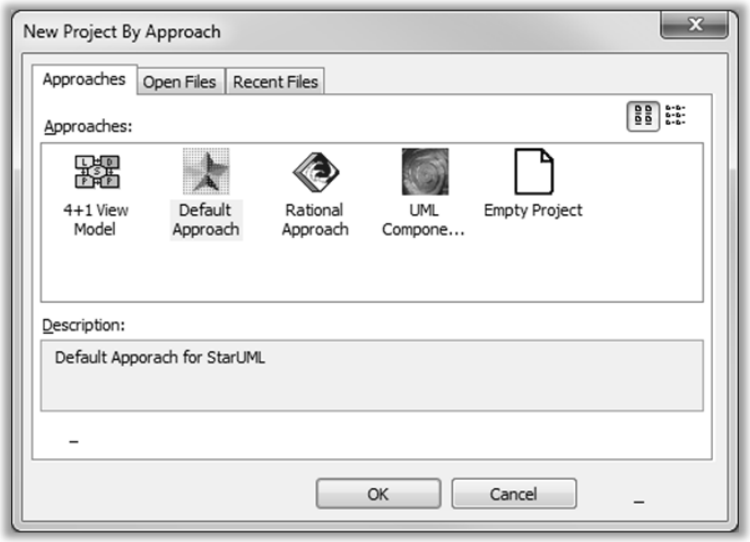


Рисунок 7.31. **Начальное окно запуска**

После того как выбран один из предложенных подходов, появляется основное окно программы.

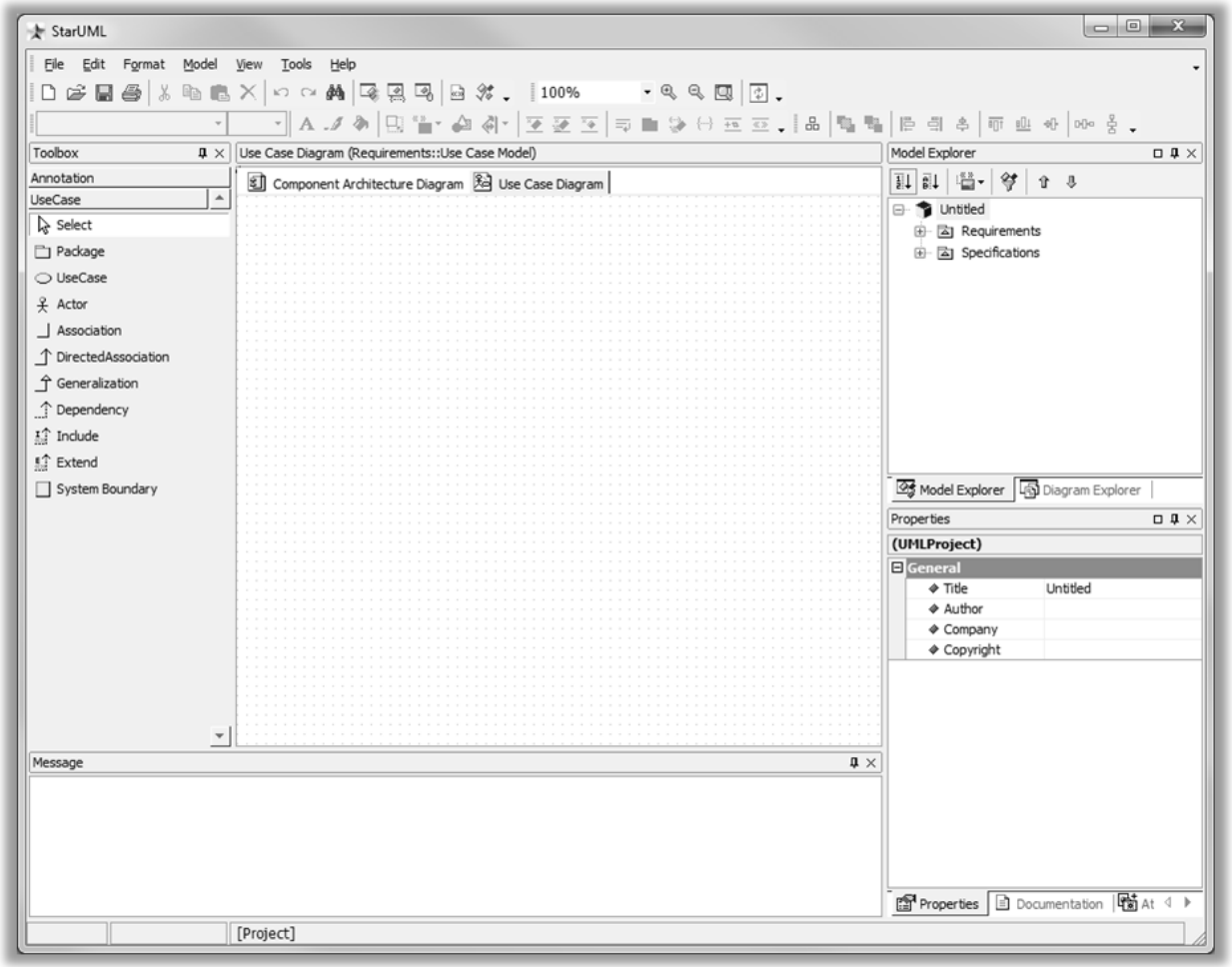


Рисунок 7.32. **Основное окно программы**

Для того чтобы создать еще одну диаграмму (любого типа), например, детализирующую прецедент, щелкните правой кнопкой мыши по папке Use Case Model и в появившемся контекстном меню выберите Add Diagram, затем выберите из списка диаграмму, которую вы хотите добавить. Например, можно создать дополнительную диаграмму прецедентов, выбрав пункт Use Case Diagram (см. рис. 7.33).

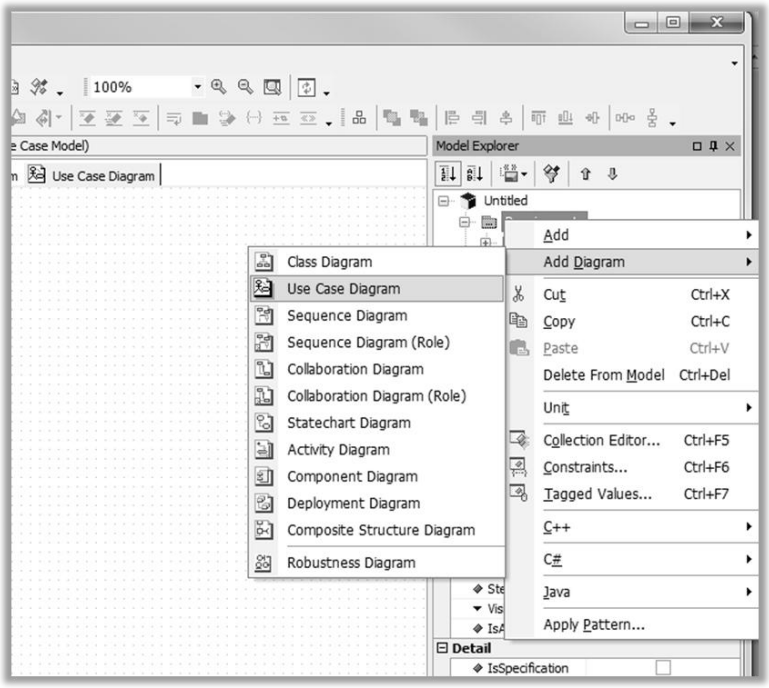


Рисунок 7.33. **Создание новой диаграммы**

Диаграмма вариантов использования

В StarUML диаграмма вариантов использования называется Main и располагается в представлении Use Case. Если в навигаторе модели щелкнуть два раза по имени этой диаграммы, то откроется ее рабочее поле. Для того чтобы создать вариант использования (прецедент), щелкните по овальному символу прецедента на панели элементов слева от рабочего поля диаграммы, а затем щелкните по тому месту на рабочем поле диаграммы, в которое необходимо поместить вариант использования. Аналогичным образом создается актер. Когда элемент помещается на поле диаграммы, он становится доступен для редактирования имени и некоторых свойств. В выделенное поле введите новое имя варианта использования или актера (рис.7.34).

Для создания отношения между элементами диаграммы щелкните по изображению соответствующего отношения на панели элементов справа, а затем проведите линию от одного элемента к другому, удерживая левую кнопку мыши.

В модель нужно включить краткое описание каждого актера или прецедента, делается это для того, чтобы между разработчиком и заказчиком системы не оставалось «белых пятен» и расхождений в понимании функциональности системы и ролей взаимодействующих с ней актеров. Для каждого актера описывается роль, которую он играет в системе, а для каждого прецедента – его назначение и функциональность. Также можно уточнить, каким актером запускается вариант использования.

В StarUML добавление описания к элементам модели делается следующим образом. Выделите элемент модели, щелкнув по нему мышкой, и откройте редактор **Documentation**. Если он не отображается справа на одной из вкладок инспектора модели, то откройте его, используя меню **View → Documentation**. Напротив пункта **Documentation** должна стоять галочка.

Введите описание каждого элемента в окно документирования (рис. 7.35).

На рисунке 7.36 приведена диаграмма вариантов использования. Основными актерами выступают Оператор, Клиент, Диспетчер и Водитель. Клиент, в свою очередь, имеет два потомка – физическое лицо и юридическое лицо. Такое выделение вызвано тем, что данные актеры имеют различные признаки. Основные виды деятельности каждого актера представлены соответствующими вариантами использования.

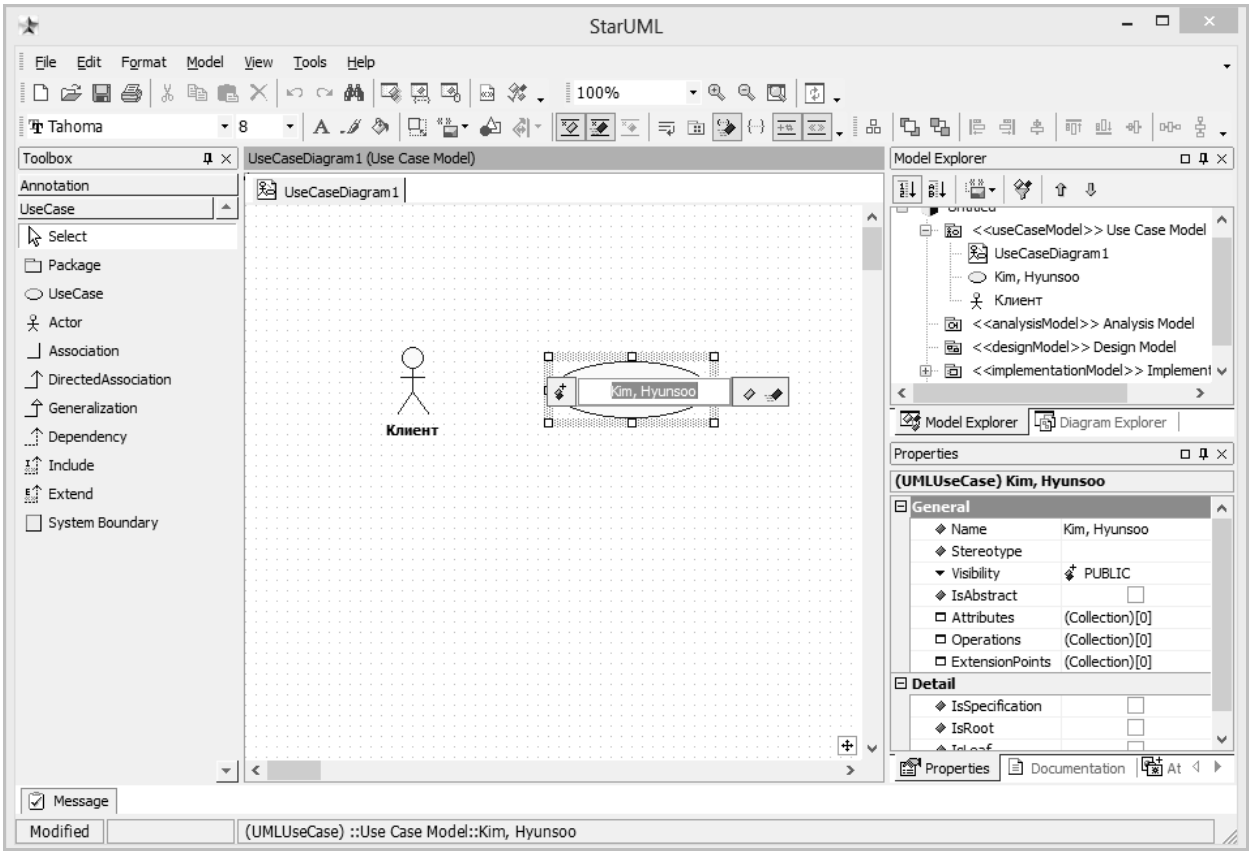


Рисунок 7.34. **Именование элементов в StarUML**

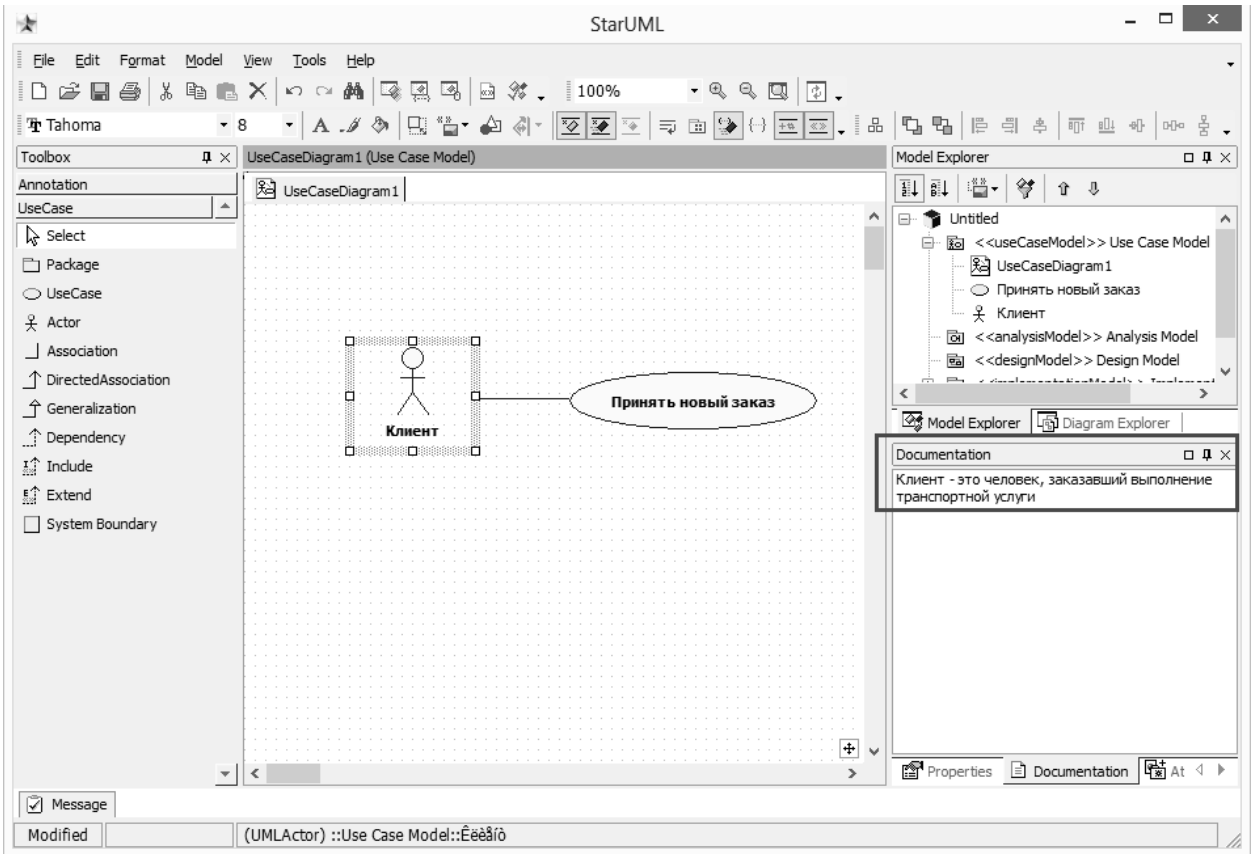


Рисунок 7.35. **Документирование элемента модели в StarUML**

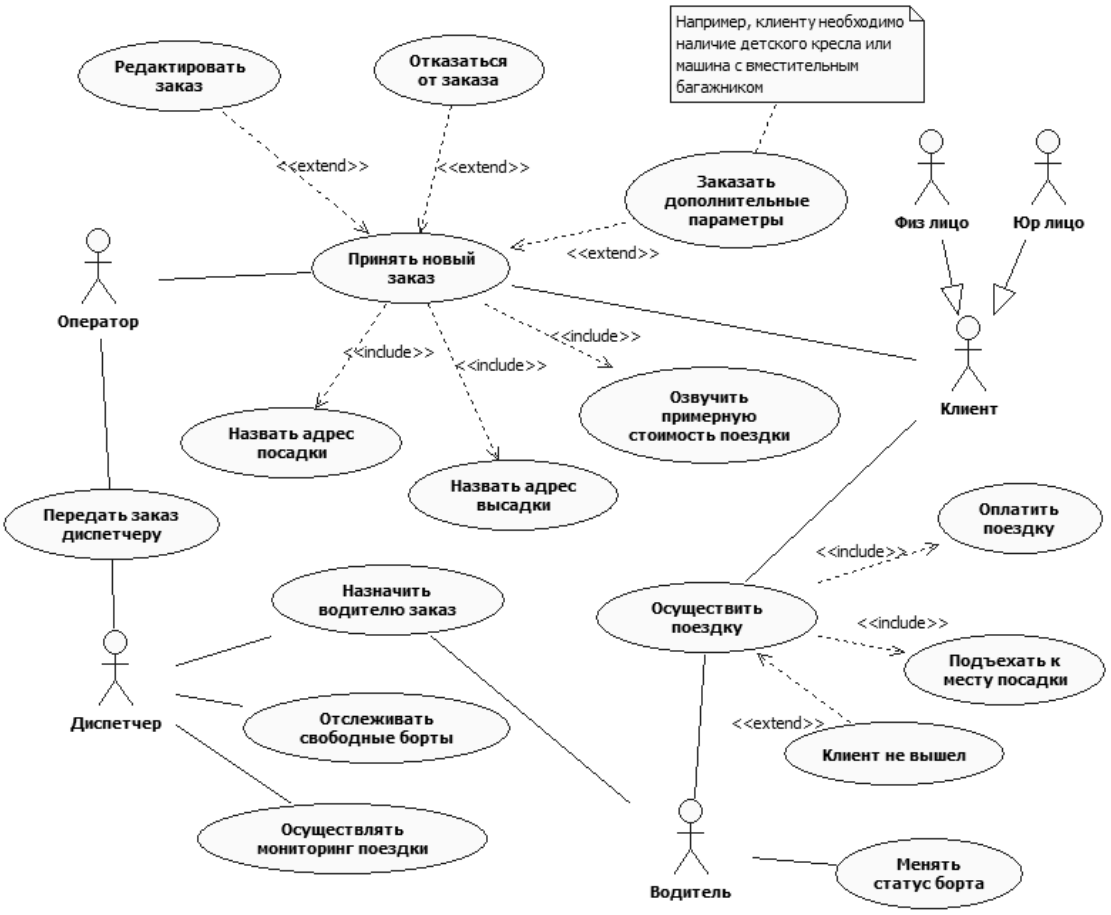


Рисунок 7.36. **Диаграмма вариантов использования**

Одним из требований языка UML является самодостаточность диаграмм для представления информации о моделях проектируемых систем. Однако диаграммы вариантов использования описывают то, что делает система, без уточнения того, как она это делает. Для реального описания системы потребуются более специфические данные, которые отражены в потоке событий. Потоки событий уточняют или детализируют последовательность действий, совершаемых системой при выполнении ее вариантов использования, а также описывают логику переходов через варианты использования.

Поток событий – это определенная последовательность действий, которая описывает действия актеров и поведение моделируемой системы в форме обычного текста. Потоки событий – это текстовые описания пошагового выполнения прецедентов, они понятны не только разработчику, но и стороннему читателю. Их задача – еще больше детализировать описание функциональности системы до того, как разработчики приступят к написанию программного кода, и устранить возможное недопонимание требуемой функциональности, как можно больше сблизить представления разработчика и заказчика о системе.

Потоки событий бывают трех типов: основной, альтернативный и поток ошибок. Поток событий формируется для каждого варианта использования.

Ниже представлен пример такого документа для варианта использования «Принять новый заказ».

Краткое описание

Вариант использования «Принять новый заказ» позволяет клиенту (физическому или юридическому лицу) заказать такси для получения транспортной услуги по перевозке с последующей оплатой полученной услуги.

Предусловия

Получение транспортной услуги возможно, если служба управления такси имеет свободные борта.

Основной поток событий

Например, поток событий варианта использования «Принять новый заказ» может выглядеть следующим образом:

Основной поток

1. Вариант использования начинается, когда клиент звонит оператору.
2. Оператор открывает карточку нового заказа. В карточку нового заказа вводится информация об адресе посадки, адресе высадки и озвучивается примерная стоимость поездки. При необходимости может запуститься альтернативный поток А1.
3. Статус заказа меняется с «Новый» на «Принят».
4. Оператор передает заказ диспетчеру на исполнение.
5. Вариант использования завершен.

Альтернативный поток А1. Заказ дополнительных параметров поездки.

1. Клиент заказывает дополнительные параметры борта.
2. Оператор делает соответствующие пометки в карточке заказа.
3. Вариант использования завершен.

Альтернативный поток Е1. Звонок клиента прервался, и заказ не принят.

1. Во время приема заказа звонок клиента прервался.
2. Заказ не принят.
3. Вариант использования завершается.

Поток ошибок Е2. Услуга не может быть предоставлена.

1. Нет свободных бортов и клиент не согласен ожидать, когда освободится такси.
2. Заказ отменен.
3. Оператор фиксирует факт отказа от поездки.
4. Вариант использования завершается.

Постусловия

После принятия нового заказа, его статус меняется со значения «Новый» на «Принят» и передается для последующего исполнения диспетчеру.

Чтобы добавить поток событий к модели, нужно выделить вариант использования, который он детализирует, в инспекторе модели открыть редактор вложений **Attachments**, нажать на значок, расположенный в верхней части редактора вложений, в появившемся окне указать пусть к соответствующему файлу, содержащему описание потока событий (рис. 7.37).

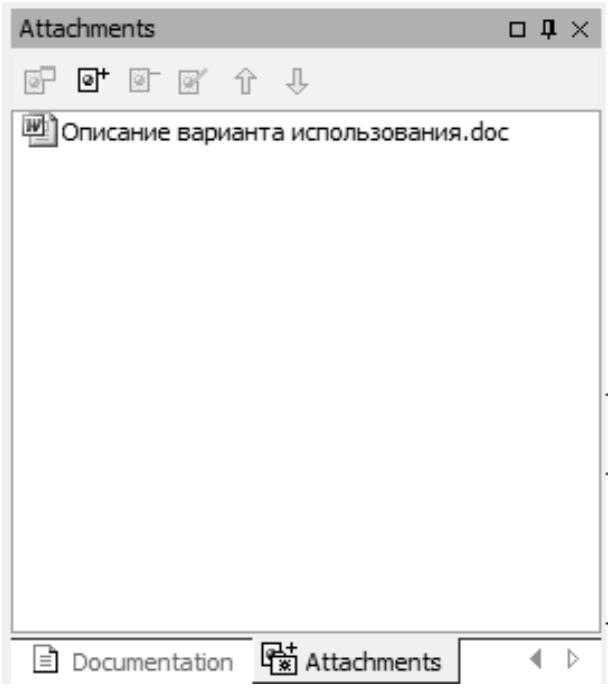


***Рисунок 7.37.* Добавление вложений в модель**

Диаграмма классов

На рисунке 7.38 представлена диаграмма классов, в которой приведены основные элементы предметной области, а также их атрибуты. Для связей между классами указана кратность. Так, например, кратность связи между классами «Клиент» и «Заказ» означает, что любой клиент может сделать от 1 до n количества заказов.

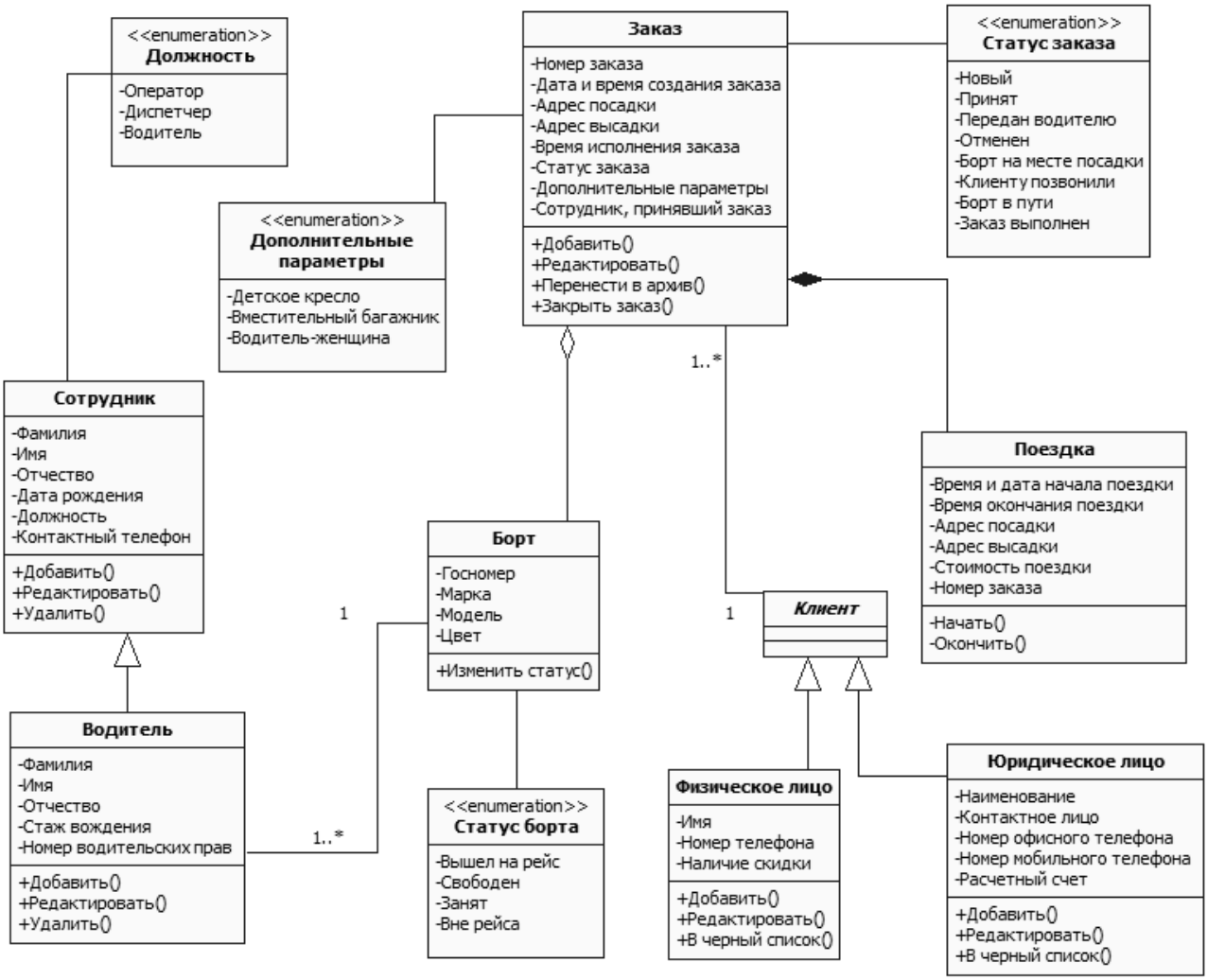


Рисунок 7.38. **Диаграмма классов**

Диаграмма состояний

Для добавления диаграммы состояний в модель нужно выполнить следующие шаги: щелкнуть правой кнопкой мыши по папке представления **Logical View** в навигаторе модели, в контекстном меню выбрать пункт **Add Diagram**, в списке выбрать диаграмму состояний **Statechart Diagram**.

Также можно связать диаграмму состояний с тем классом, состояния объекта которого она описывает. Для этого нужно щелкнуть правой кнопкой мыши по соответствующему классу, а не по папке **Logical View**.

Для рассматриваемой предметной области было построено две диаграммы состояний – для заказа (рис. 7.39) и для борта (рис. 7.40)

Состояние заказа. Заказ на транспортную услугу считается принятым в момент, когда он зафиксирован оператором в карточке заказа. Затем заказ диспетчером передается свободному водителю для исполнения. В момент прибытия борта на место посадки осуществляется дозвон до клиента и последующая доставка пассажира на место высадки (состояние *«Борт в пути»*). После того как клиент доставлен и осуществлен расчет, заказ считается выполненным. Состояние «*Отменен*» не

имеет никаких связей с другими состояниями, что означает, что в данное состояние объект может попасть из любого другого состояния.

Для отдельных состояний указаны условия входа (entry), действия (do), выхода (exit).

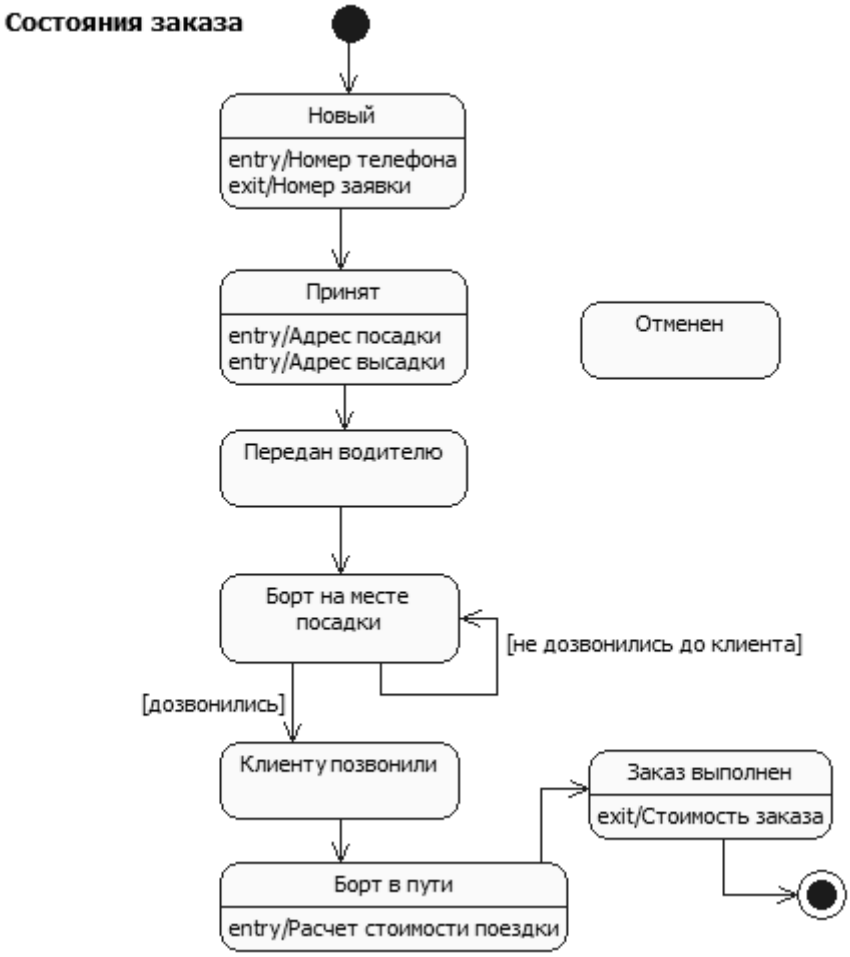


Рисунок 7.39. **Диаграмма состояний заказа**

Состояние борта. Борт может находиться только в двух состояниях – «*Вышел на рейс*» и «*Вне смены*». Условия перехода из одного состояния в другое специфицированы триггерными переходами. Состояние «Вышел на рейс» является составным и включает два состояния: «Свободен» и «Занят» (т.е. водитель выполняет заказ).

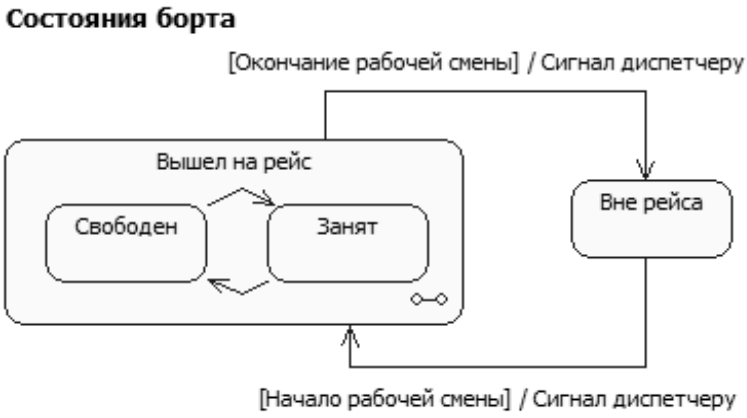


Рисунок 7.40. **Диаграмма состояний борта**

Диаграмма деятельностей

Диаграммы деятельностей предназначены для отражения последовательности действий. Диаграмма деятельностей может описывать последовательность действий как конкретного актора, так и последовательность исполнения бизнес-процесса целиком.

Чтобы построить диаграмму деятельностей для некоторого варианта использования в StarUML, нужно щелкнуть правой кнопкой мыши по этому прецеденту, в выпавшем контекстном меню выбрать пункт **Add Diagram**, затем в появившемся списке выбрать **Activity Diagram**.

Поле для создания диаграммы деятельностей появится в окне программы, изменится панель инструментов слева, и новая диаграмма отобразится на навигаторе модели.

Диаграмма деятельностей, описывающая деятельность оператора, представлена на рисунке 7.41.

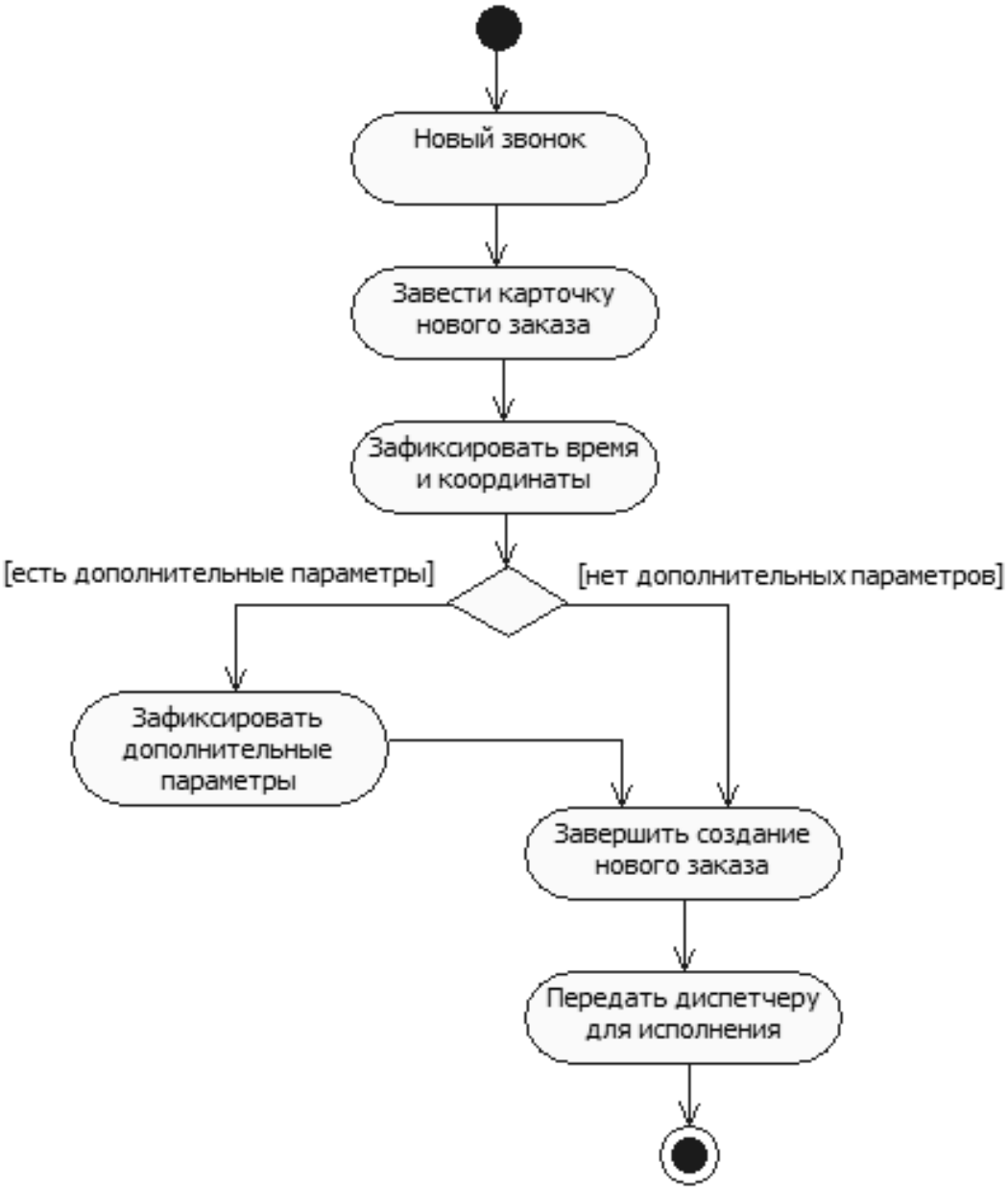


Рисунок 7.41. **Диаграмма деятельностей оператора**

Диаграмма деятельностей, показывающая последовательность бизнес-процесса «Исполнение заказа», представлена на рисунке 7.42.

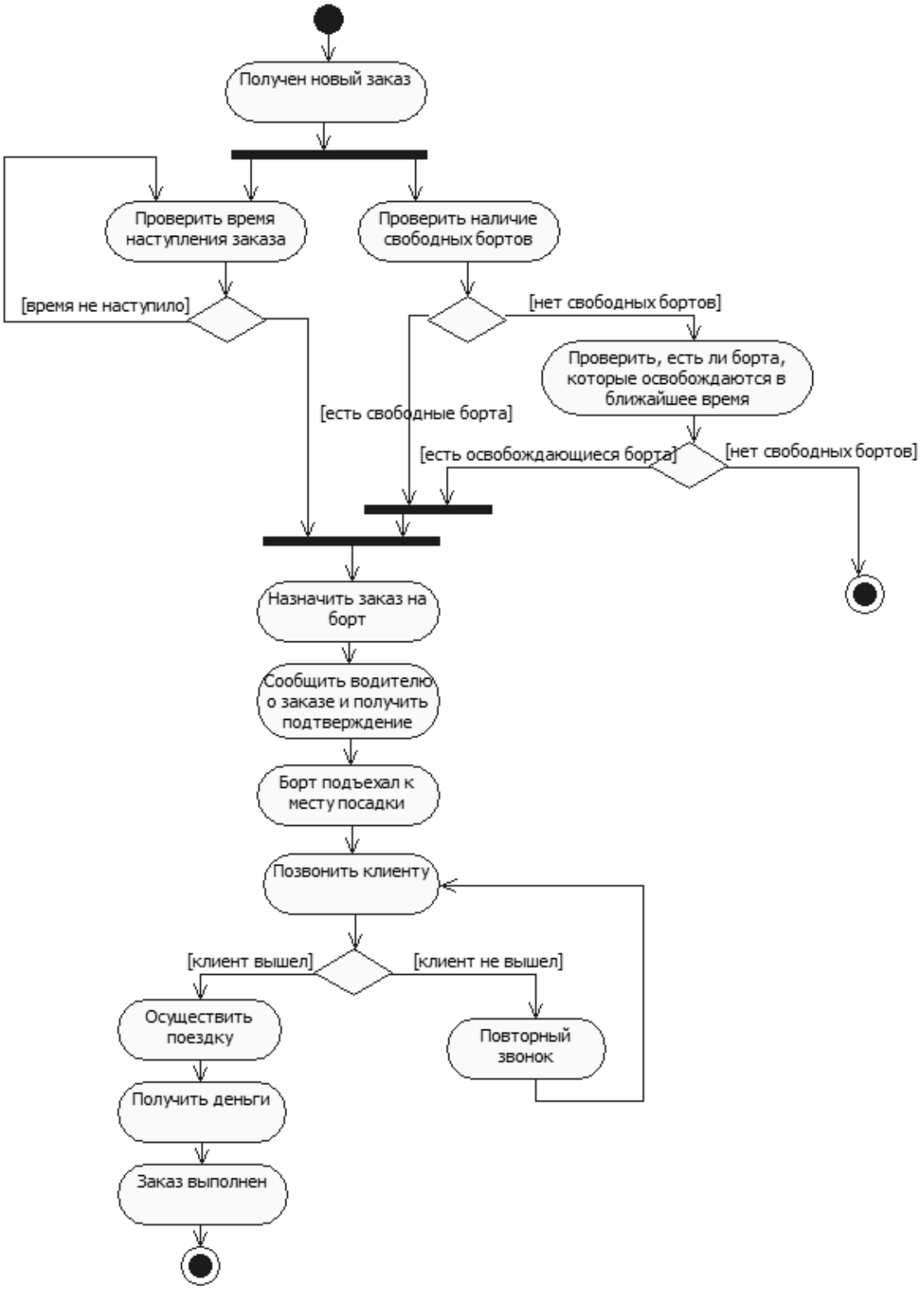


Рисунок 7.42. **Диаграмма деятельностей бизнес-процесса**

Диаграмма последовательности

Для создания новой диаграммы последовательности нужно выполнить следующие шаги: щелкнуть правой кнопкой мыши по папке представления **Logical View** в навигаторе модели, в контекстном меню выбрать пункт **Add Diagram**, в списке выбрать диаграмму последовательности **Sequence Diagram**.

Диаграмма последовательности для варианта использования «Принять новый заказ» представлена на рисунке 7.44. Как видно из нее, инициатором выступает клиент, осуществляющий звонок оператору. На основании звонка оператор посылает сообщение на создание новой карточки заказа (стереотип <<create>>).

Для определения типа сообщения в StarUML нужно выполнить следующее: выделите сообщение, щелкнув по соответствующей стрелке один раз левой кнопкой мыши, откройте редактор свойств, выберите на нем раздел **ActionKind** и в выпадающем списке выберите тот тип синхронизации, который вы хотите установить (рис. 7.43).

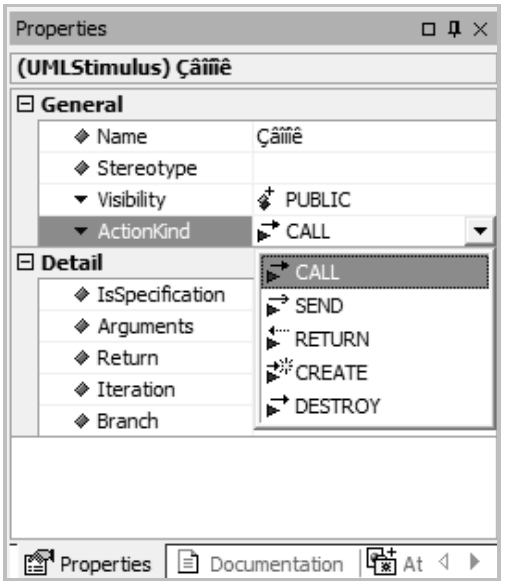


Рисунок 7.43. **Выбор типа сообщения**

Также на диаграмме представлен альтернативный поток **alt**.

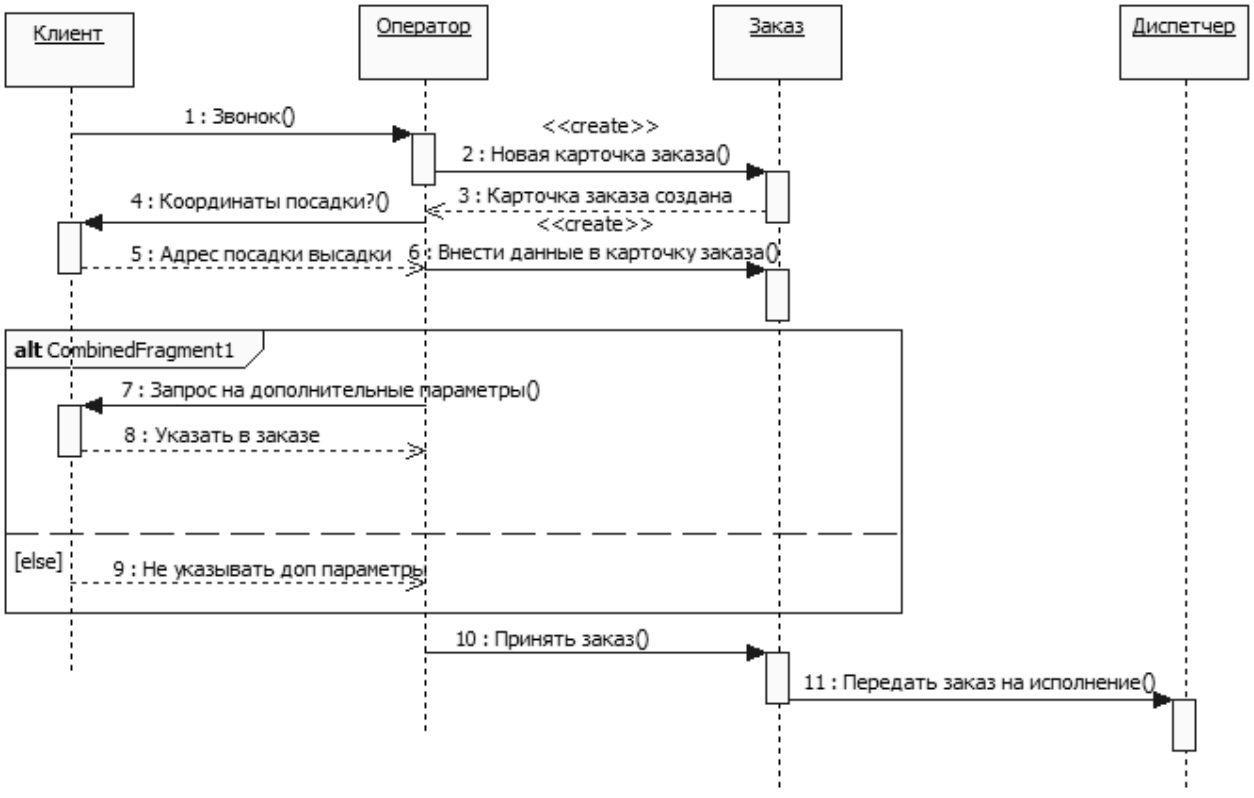


Рисунок 7.44. **Диаграмма последовательности**

Диаграмма кооперации

Диаграмма кооперации – это альтернативный способ изображения сценария варианта использования.

Для того чтобы добавить диаграмму кооперации в представление **Logical View**, щелкните правой кнопкой мыши, в контекстном меню выберите пункт **Add Diagram**, в списке выберите диаграмму кооперации **Collaboration diagram**.

Этот тип диаграмм заостряет внимание на связях между объектами, отображая обмен данными в системе. Однако здесь больше внимания уделено собственно существующим взаимосвязям. Стрелками показаны направления передаваемых сообщений, цифрами – последовательность (рис. 7.45).

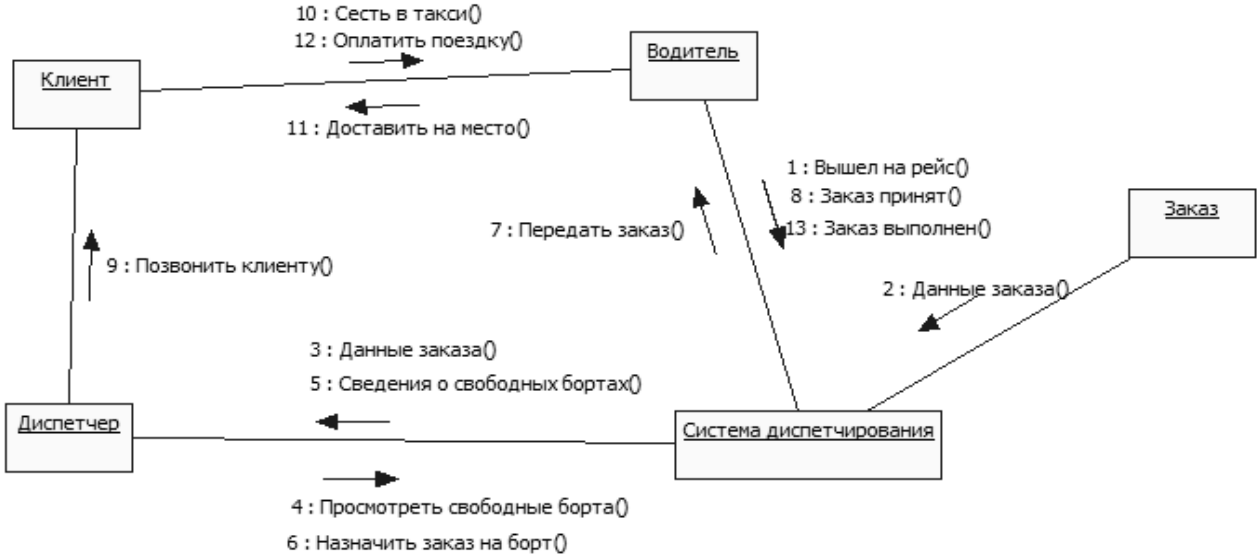


Рисунок 7.45. **Диаграмма кооперации**

Диаграмма компонентов

На диаграмме компонентов отображаются компоненты программного обеспечения и связи между ними. При разработке диаграммы в данном случае акцент был сделан на процесс отслеживания местонахождения борта.

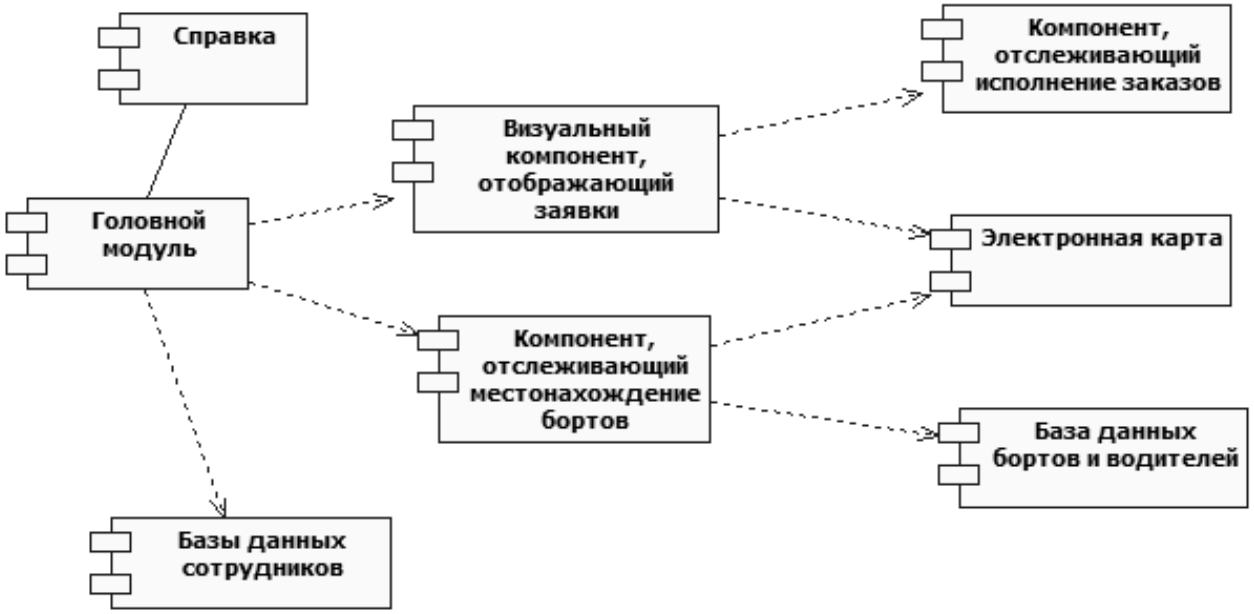


Рисунок 7.46. **Диаграмма компонентов**

Диаграмма размещения

Применяется трехзвенная архитектура: клиенты общаются с сервером приложений. Клиенты посылают серверу приложений запросы, а получают ответы. Клиенты могут обратиться и непосредственно к серверу базы данных за теми или иными данными. Обращение за данными к серверу базы данных может производить и сервер приложений.

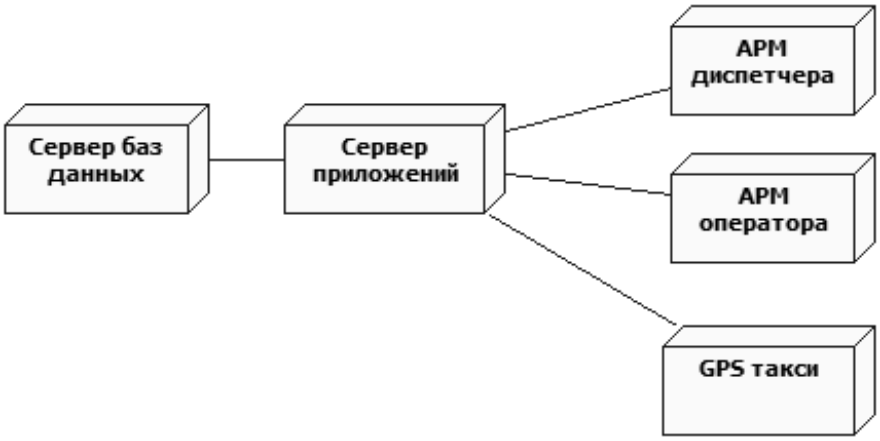


Рисунок 7.47. **Диаграмма размещения**

Требования к отчету

Отчет по лабораторной работе оформляется в печатном виде. Защита работы включает в себя проверку знания студентом теоретического материала, а также практической части лабораторной работы.

Отчет должен включать:

- разработанные UML-диаграммы, описывающие заданную предметную область;
- краткое описание для каждой из разработанных диаграмм (для диаграммы вариантов использования обязательно должно быть представлено описание потоков событий, для диаграммы состояний должно быть приведено краткое пояснение по каждому состоянию).

7.5. Лабораторная работа 4

Цель: изучить методологию моделирования бизнес-процессов и на основании выделенного бизнес-процесса разработать прототипы экранных форм.

Задание: на основании заданной предметной области в методологии ARIS разработать схему бизнес-процесса, используя VAD и eEPC диаграммы.

7.5.1. Краткий обзор CASE-средств для построения диаграмм в методологии ARIS

ARIS-диаграммы можно создавать с помощью различных средств, таких например, как MS Visio, ARIS Express и пр.

ARIS Express – это бесплатный инструмент для моделирования бизнес-процессов. Данный продукт принадлежит к семейству средств моделирования ARIS (ARchitecture of Integrated Information Systems) компании IDS Scheer (в настоящее время являющейся частью фирмы Software AG, http://www.ariscommunity.com).

Возможности MS Visio в области создания ARIS-диаграмм достаточно сильно зависят от версии и пакета. Однако, учитывая универсальность продукта, MS Visio можно рекомендовать для построения практически любых диаграмм.

Краткие сведения о работе в пакете ARIS Express

ARIS Express поддерживает общепринятые стандартные нотации для описания процессов, оргструктуры, информационных систем и моделей данных (рис. 7.48).

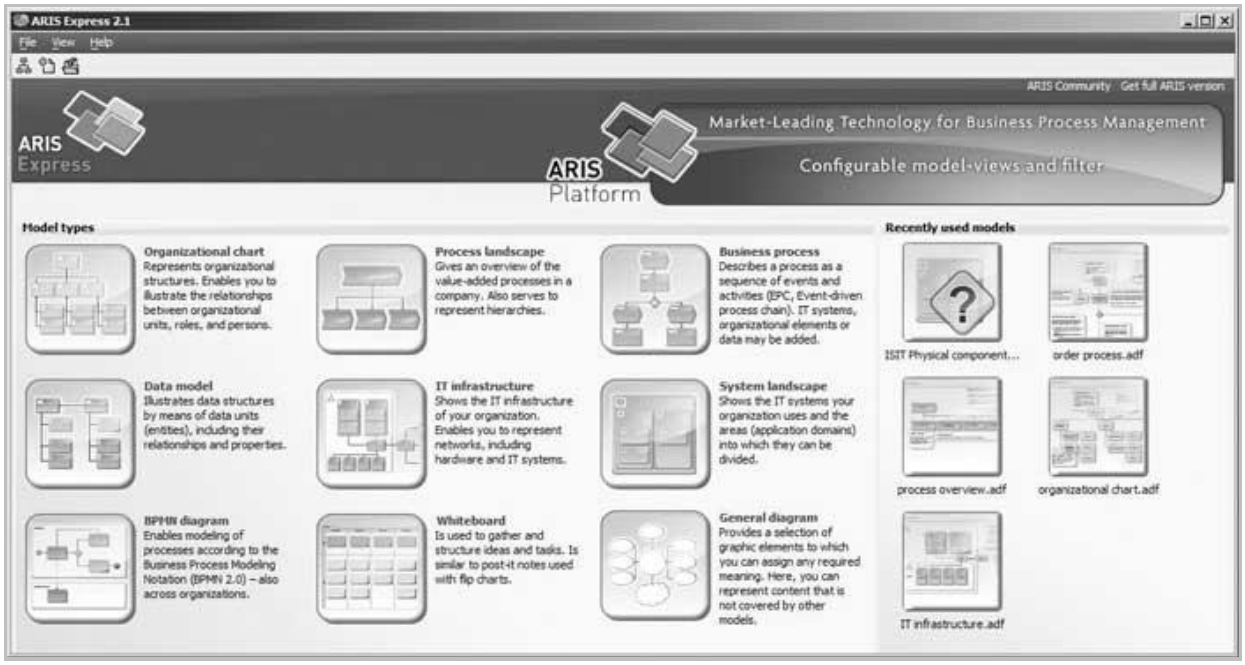


Рисунок 7.48. **Поддерживаемые инструментом ARIS Express типы моделей**

Каждый из поддерживаемых в ARIS Express типов моделей может содержать определенный набор типов объектов, наиболее распространенных при создании моделей данного типа. С целью повышения эффективности работы аналитика у него есть возможность создавать фрагменты моделей, которые в дальнейшем можно использовать повторно. Также можно изменять в соответствии с корпоративными требованиями внешний вид модели, цвет, расположение атрибутов моделей и объектов, тип шрифта.

Краткие сведения о создании eEPC-диаграмм в пакете MS Visio

Ниже приведено краткое описание последовательности действий для создания ARIS-диаграмм в пакете MS Visio.

Запустите MS Visio и откройте новый документ.

Для создания eEPC: отобразите нужную панель (рис. 7.49), в результате чего отобразится панель для рисования eEPC-диаграмм (рис. 7.50)

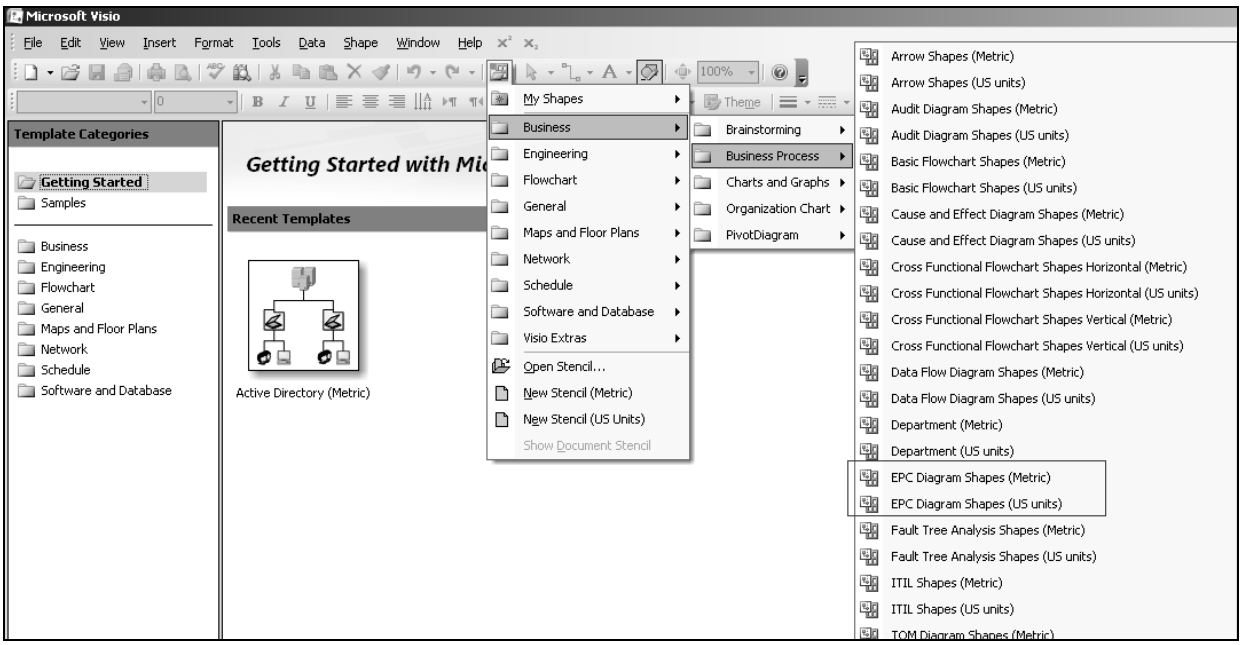


Рисунок 7.49. **Панели MS Visio**

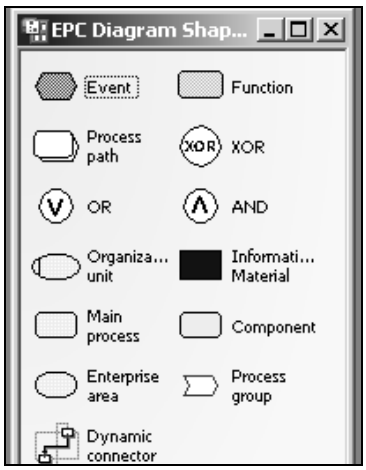


Рисунок 7.50. **Панель элементов eEPC**

Пример диаграмм

На диаграмме процесса добавленной стоимости VAD описывается модель основного бизнес-процесса в верхних уровнях декомпозиции. На диаграмме показаны 5 основных процессов: прием заявки, провести учет водителей, обработать заказ на поездку, осуществить мониторинг выполнения заказа и осуществление расчета за поездку. Показана информация, которая передается между процессами, цели, которые ставятся перед процессом, а также лица, участвующие в процессе.

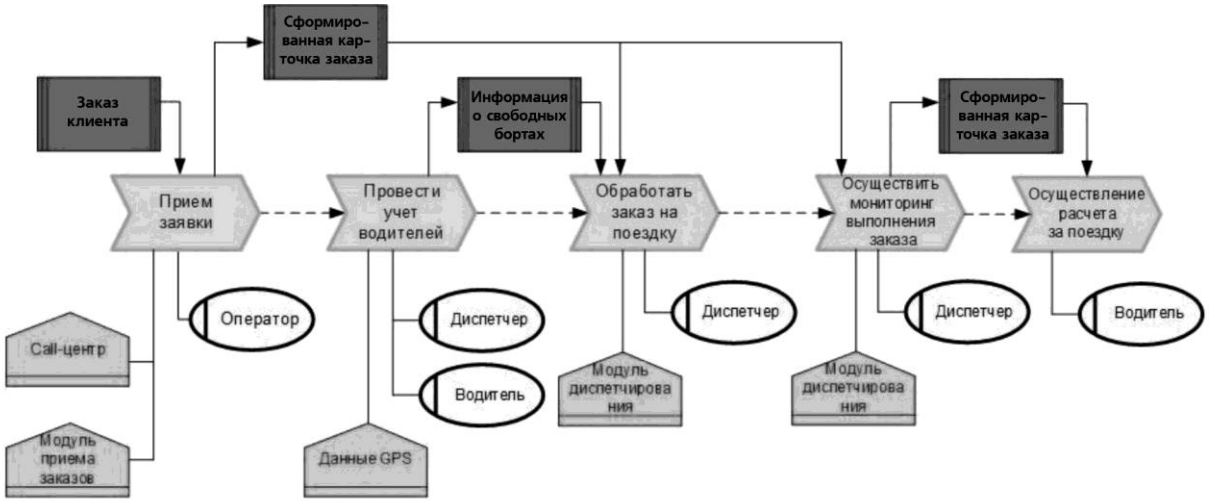


Рисунок 7.51. **Диаграмма процесса добавленной стоимости основного бизнес-процесса**

Диаграмма событийно-управляемого процесса (eEPC) используется для дальнейшей декомпозиции процессов. Строятся следующие диаграммы:

- прием заявки;
  - подпроцесс «формирование карточки заказа»;
  - подпроцесс «уточнение требований у клиента»;
- провести учет водителей:
  - подпроцесс «проверка бортов, вышедших на линию»;
  - подпроцесс «проверка свободных бортов»;
- обработать заказ на поездку:
  - подпроцесс «выбрать вариант размещения заказа»;
  - подпроцесс «назначение заказа на борт»;
- осуществить мониторинг выполнения заказа:
  - подпроцесс «уведомление клиента о подъехавшем борте»;
  - подпроцесс «контролировать исполнение заказа»;
  - осуществление расчета за поездку.

Ниже представлена eEPC-диаграмма процесса «Обработать заказ на поездку».

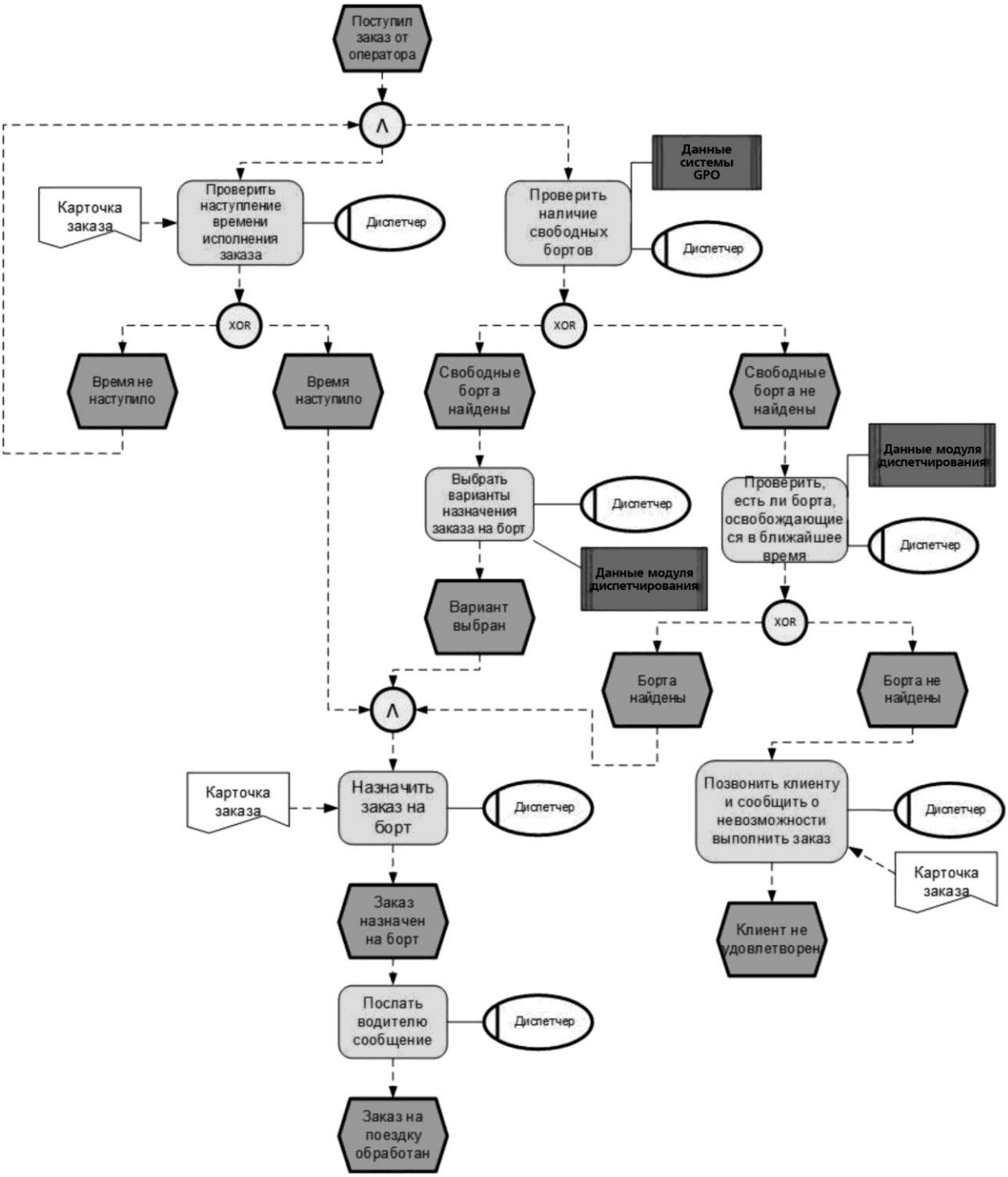


Рисунок 7.52. **eEPC-диаграмма процесса**

На рисунке 7.53 представлен пример диаграммы цепочки процессов.

Требования к отчету

Отчет по лабораторной работе оформляется в печатном виде. Защита работы включает в себя проверку знания студентом теоретического материала, а также практической части лабораторной работы.

Отчет должен включать:

– разработанные диаграммы (VAD, eEPC и PCD), описывающие заданную предметную область;
– краткое описание для каждой из разработанных диаграмм.

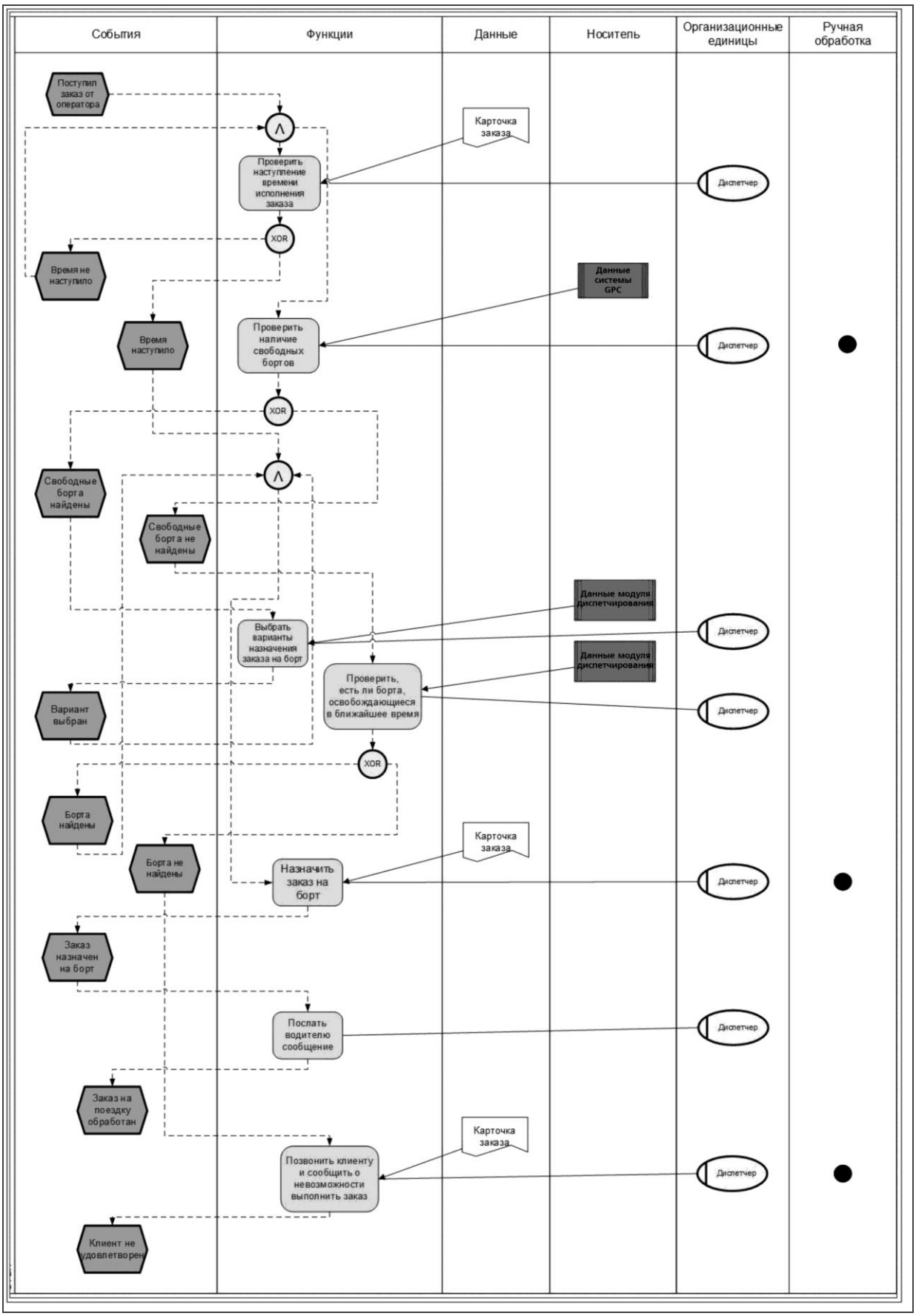


Рисунок 7.53. **PCD-диаграмма**

7.6. Лабораторная работа 5

Цель: изучить методологию моделирования бизнес-процессов BPMN.

Задание: на основании заданной предметной области в методологии BPMN разработать схему бизнес-процесса.

Краткие сведения о создании BPMN-диаграммы в MS Visio

Ниже приведено краткое описание последовательности действий для создания BPMN-диаграмм в пакете MS Visio.

1) Запустите MS Visio и откройте новый документ.
2) Для создания BPMN: отобразите нужную панель (рис. 7.54), в результате чего отобразится панель для рисования BPMN-диаграмм.

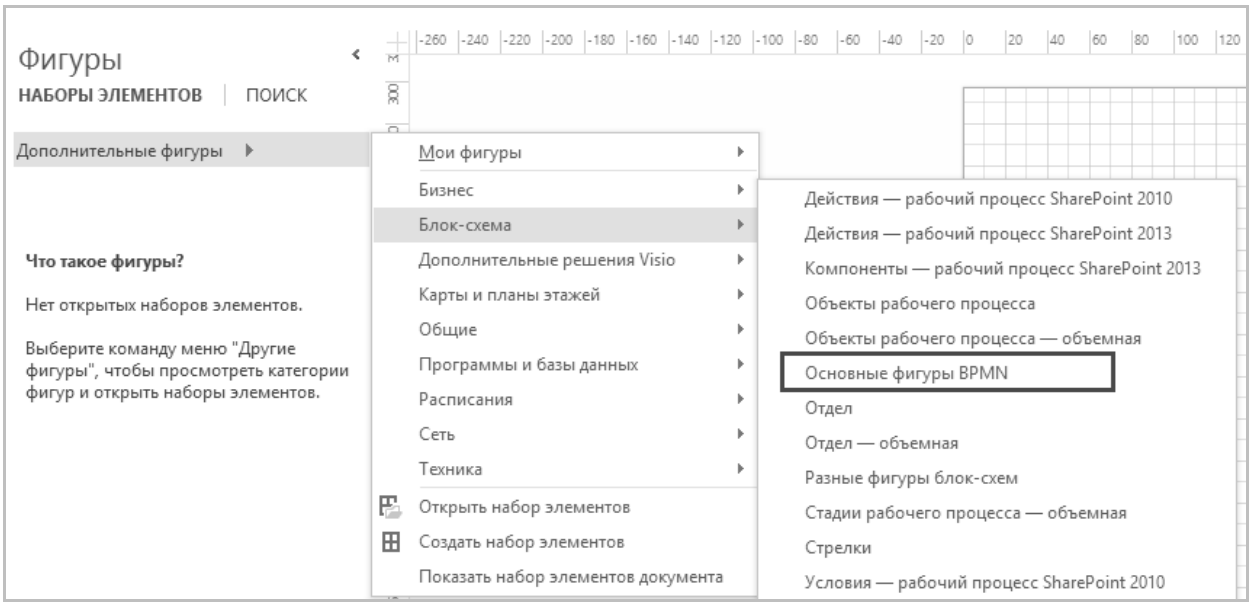


Рисунок 7.54. **Панель элементов BPMN**

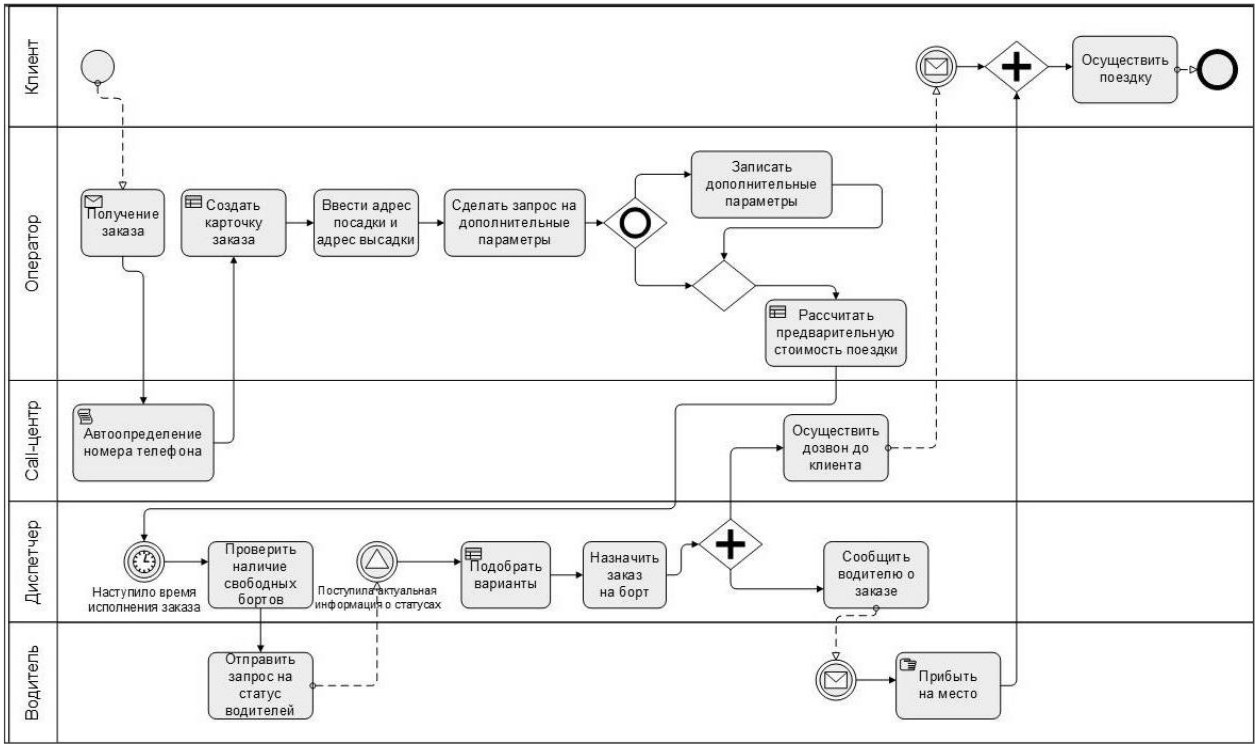


Рисунок 7.55. **BPMN-диаграмма**

8. ВАРИАНТЫ ЗАДАНИЙ

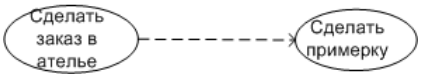
Вариант задания указывается преподавателем.

1. Аренда автомобиля в агентстве по сдаче автомобилей в аренду.
2. Бронирование и покупка билета на поезд.
3. Ведение электронного дневника: учитель ставит оценку школьнику, сообщение об оценке приходит на сотовый телефон родителей, в случае, если ребенок забыл задание, у родителей есть возможность узнать задание.
4. Ветклиника: хозяин привел питомца к врачу.
5. Выполнение заказа на пошив платья в ателье.
6. Заказ в Макдоналдс через электронное табло.
7. Заказ визитки (рекламного буклета) в типографии с предварительной подготовкой макета визитки.
8. Интернет-магазин: покупка обуви через веб-портал.
9. Машиностроительное предприятие: система по разработке и модификации изделий (ведение архива, стандартов и пр.).
10. Организация единой системы управления зоопарком, в том числе организация продажи билетов, работа ветеринаров, мониторинг кормления животных, контроль чистоты клеток и пр.
11. Организация работы библиотеки с возможностью предоставления читателям не только бумажных носителей, но и полнотекстовых электронных копий.
12. Организация работы ресторана: бронирование столика, обслуживание клиента, учет расхода ингредиентов.
13. Организация работы санатория для пациентов с полным и частичным пансионом.
14. Организация работы страховой компании по работе с физическими лицами.
15. Организация работы управляющей компании: формирование счетов, оплата коммунальных услуг квартиросъемщиком, управление капитальным ремонтом дома.
16. Организация туристическим агентством путешествий за рубеж.
17. Организация туристическим агентством путешествий индивидуальных и типовых туров по России.
18. Прием пациента в поликлинике с точки зрения пациента.
19. Прием пациента в стоматологической клинике с точки зрения врача.
20. Проведение приемной кампании в вузе.
21. Проведение рекламной кампании по продвижению нового продукта.
22. Производство мебели: прием индивидуальных и типовых заказов и изготовление мебели с доставкой клиенту на дом.
23. Работа оператора на АЗС.
24. Риэлтерское агентство: аренда; продажа первичного и вторичного жилья.
25. Система контроля доступа (СКД, СКУД) для средней компании (численность сотрудников больше 30 человек).
26. Система управления проектом для IT-компании: ведение детальной информации по каждому проекту, мониторинг назначения и контроля исполнения заданий каждому сотруднику, учет использованного времени.
27. Система управления транспортной логистикой с возможностью осуществления мультиперевозок.
28. Складская логистика для крупной сети продуктовых магазинов.
29. Составление афиши в филармонии с учетом абонементов и разовых концертов.
30. Учет оборудования на крупном промышленном предприятии.

9. ТЕСТИРУЮЩИЙ КОМПЛЕКС

Данный раздел содержит пример теста, позволяющего студенту самостоятельно оценить степень проработанности изученного материала. В конце теста приведены правильные варианты ответов.

1. Укажите тип отношения между вариантами использования

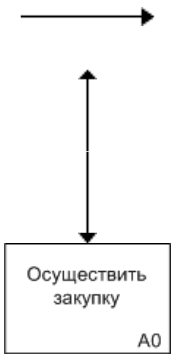


   а) отношение обобщения
   б) отношение включения
   в) отношение расширения
   г) отношение ассоциации

2. Укажите, какие цели преследует диаграмма компонентов
   а) представление временных особенностей передачи и приема сообщений между объектами
   б) визуализация общей структуры исходного кода программной системы
   в) представление концептуальной и физической схем баз данных
   г) описание поведения системы в рамках различных вариантов использования
   д) спецификация исполнимого варианта программной системы
   е) обеспечение многократного использования отдельных фрагментов программного кода

3. Диаграммы взаимодействия применяются для моделирования
   а) статической структуры классов системы и связей между ними
   б) бизнес-процессов организации (требований к системе)
   в) поведения объектов системы при переходе из одного состояния в другое
   г) иерархии компонентов (подсистем) системы
   д) процесса обмена сообщениями между объектами

4. Поставьте в соответствие графические примитивы и обозначения IDEF0-диаграммы



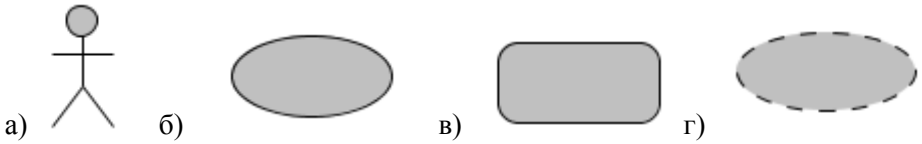
блок

управление

вход

механизм

5. Укажите, как изображают актера на диаграмме вариантов использования



6. Основными элементами диаграммы классов являются
   а) класс
   б) объект
   в) состояние
   г) актер
   д) интерфейс

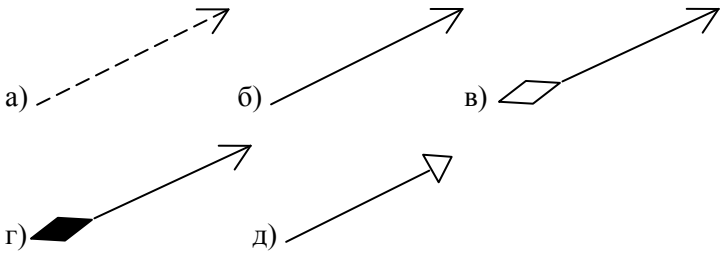
7. Назначение DFD-диаграммы заключается в том, что
   а) диаграммы потоков данных предназначены для моделирования и документирования аспектов систем, зависящих от времени или реакции на событие
   б) с помощью DFD требования пользователя разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных
   в) с помощью DFD осуществляется детализация хранилищ данных проектируемой системы, а также документируются сущности системы и способы их взаимодействия, включая идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их отношений с другими объектами (связей)

8. Концепцию SADT (Structured Analysis and Design Technique) предложил …
   а) Е. Кодд
   б) Гради Буч
   в) Дуглас Т. Росс
   г) Ивар Якобсон

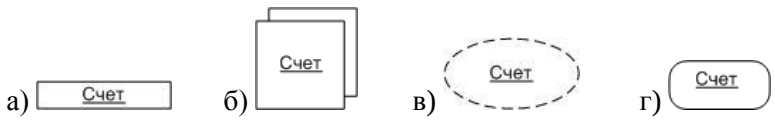
9. Блоки на SADT-модели размещаются на диаграмме …
   а) случайным образом
   б) с учетом доминирования
   в) в хронологическом порядке

10. Выберите определение, соответствующее описанию термина «фокус управления»
    а) фрагмент жизненного цикла объекта в процессе взаимодействия
    б) сообщение, которое объект посылает самому себе
    в) законченный фрагмент информации, который отправляется одним объектом другому
    г) обозначение, показывающее, что объект находится в активном состоянии, непосредственно выполняя определенные действия

11. Укажите, как в рамках диаграммы классов выглядит отношение агрегации



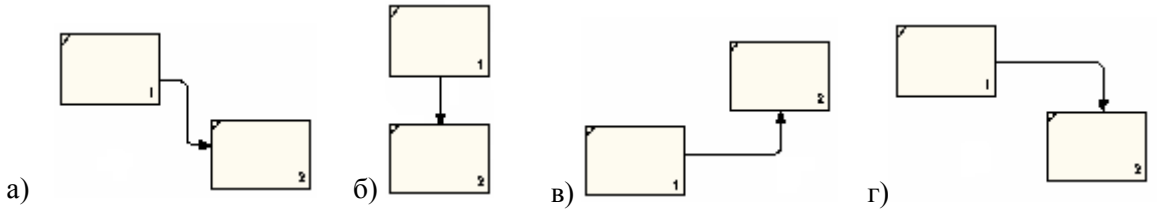
12. Как на диаграмме кооперации изображается мультиобъект?



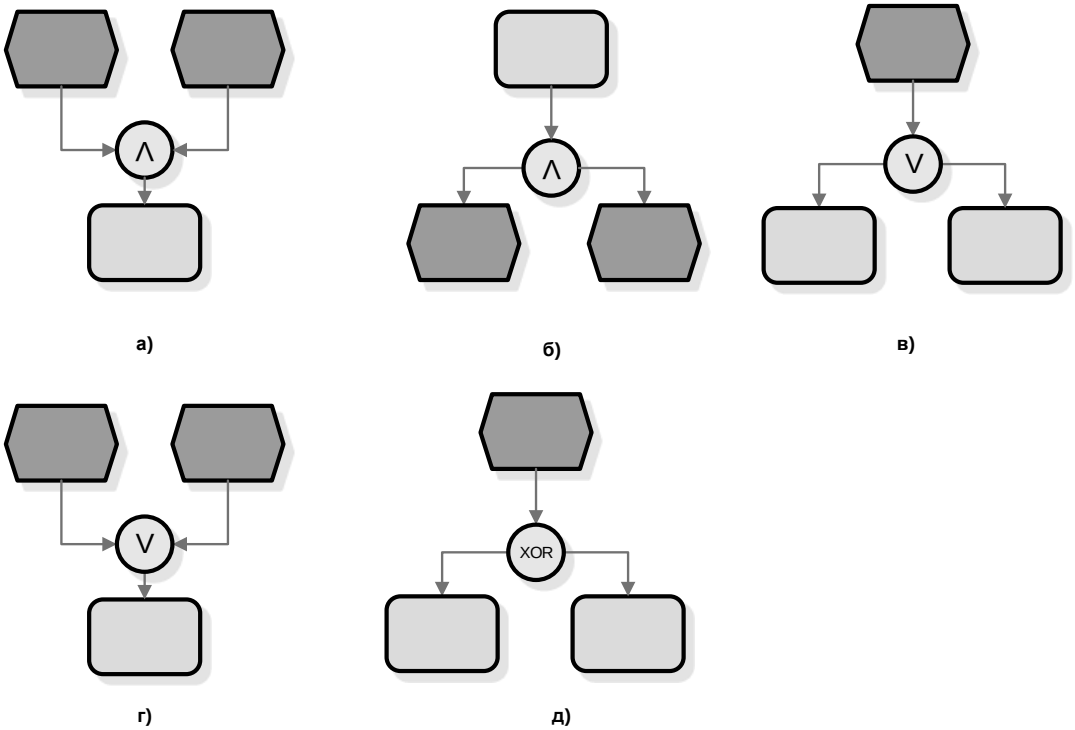
13. Основными элементами диаграммы вариантов использования являются
    а) актер (актор)
    б) вариант использования
    в) состояние
    г) узел
    д) компонент

14. Выберите корректное графическое изображение класса

15. Когда возникает эффект «туннеля» на SADT-модели?
    а) в случае неправильной декомпозиции диаграммы
    б) когда нарушается правило нумерации
    в) когда нарушается правило балансировки
16. На каждой DFD-диаграмме должно быть расположено …
    а) не менее 7 процессов
    б) от 3 до 6–7 процессов
    в) не более 3 (для контекстной диаграммы) или не меньше 6–7 процессов (для детализируемой диаграммы)
17. Какое(ие) из следующих соединений функциональных блоков неправильно?



18. Какое из следующих семантических правил описания справедливо для нотации ARIS eEPC?
    а) логические операторы могут быть определены только между событиями
    б) каждая функция должна быть инициирована событием и должна завершаться событием
    в) каждая функция должна быть связана с обработкой документа
19. Укажите, какие из следующих объектов, используются в рамках нотации ARIS eEPC
    а) функция
    б) событие
    в) компонент
    г) организационная единица
    д) документ
    е) актор
    ж) кластер информации
    з) класс
20. Какие виды соединений в eEPC-диаграмме методологии ARIS разрешены?



ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Буч Г., Рамбо Дж., Джекобсон А.* Язык UML. Руководство пользователя: Пер. с англ. – М.: ДМК Пресс; СПб.: Питер, 2004. – 432 с.
2. *Буч Г., Рамбо Дж., Джекобсон А.* UML. Проектирование программных комплексов, информационных систем. – М.: ДМК Пресс, СПб.: Питер, 2003. – 432 с.
3. *Вендров А.М.* Практикум по проектированию программного обеспечения экономических информационных систем: Учеб. пособие. – М.: Финансы и статистика, 2004. – 192 с.: ил.
4. *Гома Хасан.* UML. Проектирование систем реального времени, параллельных и распределенных приложений: Пер. с англ. – М.: ДМК Пресс, 2002. – 704 с.
5. ГОСТ Р 50.1.028-2001 Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования.
6. *Елиферов В.Г., Репин В.В.* Бизнес-процессы: Регламентация и управление: Учебник. – М.: ИНФРА-М, 2007. – 319 с.
7. *Калянов Г.Н.* CASE. Структурный системный анализ (автоматизация и применение). – М.: Лори, 1996.
8. *Каюмова А.В.* Визуальное моделирование систем в StarUML: Учебное пособие / А.В. Каюмова. – Казань: Казанский федеральный университет, 2013. – 104 с.
9. *Кватрани Т.* Визуальное моделирование с помощью Rational Rose 2002 и UML.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 192 с.
10. *Леоненков А.В.* Объектно-ориентированный анализ и проектирование с использованием UML Rational Rose: Учебное пособие / А.В. Леоненков. – И.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2006. – 320 с.: ил. (Серия «Основы информационных технологий).
11. *Марка Д.А., МакГоуэн К.* Методология структурного анализа и проектирования. – М.: МетаТехнология, 1993. – 111 с.
12. Моделирование и анализ систем. IDEF-технологии: практикум / С.В. Черемных, И.О. Семенов, В.С. Ручкин. – М.: Финансы и статистика, 2006. – 192 с: ил.
13. Основы методологий проектирования автоматизированных систем обработки информации и управления / Составитель: А.В. Иващенко / Самара: СНЦ РАН, 2009 – 40 с., ил.
14. *Рамбо Дж., Блоха М.* UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд. – СПб.: Питер, 2007. – 544 с.: ил.
15. *Рамбо Дж., Якобсон А., Буч Г.* UML: специальный справочник. – СПб.: Питер, 2002. – 656 с.
16. *Репин В.В.* Бизнес-процессы компании: построение, анализ, регламентация / В.В.Репин. – М.: РИА «Стандарты и качество», 2007. – 240 с., ил. (Серия «Деловое совершенство»).
17. *Репин В.В., Елиферов В.Г.* Процессный подход к управлению. Моделирование бизнес-процессов. – 5-е изд. – М.: РИА «Стандарты и качество», 2007. – 408 с.: ил.
18. Стандарт «Нотация моделирования бизнес-процессов» BPMN (Business Process Modeling Notation). – Режим доступа: http://www.DIRECTUM-Journal.ru/docs/1624827.html. – Загл. с экрана.
19. Учебник по BPMN. – Режим доступа: http://www.clientprav.ru/ technology/bpmn/. – Загл. с экрана.
20. *Фаулер М., Скотт К.* UML. Основы: Пер. с англ. – СПб.: Символ-Плюс, 2002. – 192 с.: ил.
21. *Шлеер С., Меллер С.* Объектно-ориентированный анализ: моделирование мира в состояниях: Пер. с англ. – Киев: Диалектика, 1993. – 240 с.
22. *Шматалюк А.* и др. Моделирование бизнеса. Методология ARIS: Практическое руководство. – С.: Серебряные нити, 2001.
23. *Шмуллер Дж.* Освой самостоятельно UML. 2-е издание: Пер. с английского – М.: Издательский дом «Вильямс», 2002. – 352 с.