**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**«Московский государственный университет путей сообщения»**

**Кафедра «Экономическая информатика»**

**В.И. МОРОЗОВА, К.Э. ВРУБЛЕВСКИЙ**

**Управление жизненным циклом информационных систем**

**УЧЕБНОЕ ПОСОБИЕ**

**М о с к в а - 2015**

**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**«Московский государственный университет путей сообщения»**

**Кафедра «Экономическая информатика»**

**В.И. МОРОЗОВА, К.Э. ВРУБЛЕВСКИЙ**

**Управление жизненным циклом информационных систем**

**Рекомендовано редакционно-издательским**

**советом университета в качестве учебного пособия**

**для бакалавров 090303 «Прикладная информатика», 380305«Бизнес-информатика»**

**М о с к в а - 2015**

**УДК 33:007**

**М-80**

Морозова В.И., Врублевский К.Э. **Управление жизненным циклом информационных систем: Учебное пособие.** – М.: МГУПС (МИИТ), 2015.  
– 109 c.

В учебном пособии изложены основные понятия проекта, методологии, методов проектирования и жизненного цикла информационных систем. Объясняется структура и модели жизненного цикла информационной системы (каскадная, спиральная). Описаны подходы к проектированию информационных систем (функционально-ориентированный и объектно-ориентированный) с использованием современных Case-средств: BPwin, ERwin, Rational Rose, методология быстрой разработки приложений (RAD).

Приводятся и подробно рассматриваются примеры заданий к лабораторным работам по моделированию информационной системы на основе современных инструментальных средств.

Учебное пособие предназначено для студентов-бакалавров экономических профилей «Прикладная информатика», «Бизнес-информатика», изучающих дисциплину «Управление жизненным циклом информационных систем».

Рецензенты: заведующий УНЛ МБП МИИТа В.Г.Дедов.

С.н.с. ВЦ РАН, к.ф.-м.н. А.Н.Соломатин.

© МГУПС (МИИТ), 2015

С о д е р ж а н и е: стр.

Введение 5

1.Методологические основы проектирования информационных систем 7

1.1. Технология проектирования ЭИС 7

1.1.1. Распределение обязанностей между участниками проекта 12

1.1.2. Модели жизненного цикла ИС 14

1.2. Автоматизированное проектирование ЭИС (CASE-технология) 25

1.2.1. Подходы к проектированию ИС 27

1.3. Объектно-ориентированное проектирование ЭИС 30

1.3.1. Методология DATARUN 32

1.3.2. Методология RAD 34

2. Создание модели данных средствами BPwin 7.0 40

2.1. Принципы построения модели (IDEF0) 41

2.1.1. Стоимостный анализ (ABC) 53

2.2. Диаграммы потов данных (DFD) 55

2.3. Метод описания процессов (IDEF3) 58

2.4. Создание отчетов 62

2.4.1. Отчеты на основе встроенных шаблонов 63

2.4.2. Отчеты с помощью шаблонов Report Template Builder 66

2.4.3. Отчеты с помощью генератора RPTwin 68

3. Создание модели данных средствами ERwin 7.0 69

3.1. Создание логической модели данных 71

3.2. Создание физической модели данных 77

3.3. Экспорт данных в СУБД 80

4. Создание модели данных средствами IBM Rational Rose 83

4.1. Особенности объектно-ориентированного подхода 83

4.2. Преимущества объектно-ориентированного подхода 86

4.3. Недостатки объектно-ориентированного подхода 88

4.4. Особенности построения информационной модели 90

4.4.1. Структура и настройка основного окна 90

4.4.2. Особенности разработки диаграмм классов 93

4.4.3. Особенности разработки диаграммы деятельности 98

5. Задания на проектирование системы 103

Заключение 106

Список литературы 108

# Введение

В настоящее время информационные системы (ИС) используются во всех сферах жизнедеятельности. Это и промышленность, сельское хозяйство, строительство, экономика, образование, медицина и т.д. ИС призваны решать такие задачи как осуществление сложнейших вычислительных процессов, разработка и управление бизнес-проектами, анализ, планирование и управление производством, оценка и прогнозирование. Развитие современных информационных технологий приводят к постоянному возрастанию сложности информационных систем, разрабатываемых для различных областей экономики. Для крупных проектов ИС характерны такие особенности, как [3]:

● сложность описания большого количества функций, процессов, элементов данных и взаимосвязей между ними;

● наличие совокупности тесно взаимодействующих подсистем, имеющих свои локальные задачи и цели функционирования;

● отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;

● необходимость интеграции существующих и вновь разрабатываемых приложений;

● функционирование в неоднородной среде на нескольких аппаратных платформах;

● разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств.

Успех реализации проекта зависит от правильности и полноты описания объекта проектирования, от полноты и непротиворечивости построения функциональных и информационных моделей ИС. Опыт проектирования ИС показывает, что это и сложная, трудоемкая и длительная по времени работа, требующая от специалистов в этой области высокой квалификации.

Еще до недавнего времени проектирование ИС осуществлялось, чаще всего опираясь на интуицию, использование опыта практических наработок, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. В процессе создания и функционирования ИС информационные потребности пользователей могут быть подвержены изменениям (уточнениям), что влечет за собой усложнение разработки ИС.

До недавнего времени проектирование ИС осуществлялось вручную, что приводило к таким проблемам, как низкое качество документации, невозможность своевременного обнаружения ошибок, продолжительный цикл разработки и, как следствие, отрицательные результаты при тестировании.

Для решения перечисленных проблем в настоящее время при разработке ИС используются программно-технологические средства специального класса - CASE-средства, которые реализуют CASE-технологию создания и сопровождения ИС.

«Под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного программного обеспечения (ПО) (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС» [5].

Появлению CASE-технологии способствовали и такие факторы, как:

● подготовка аналитиков и программистов, готовых к принятию концепций модульного и структурного программирования;

● широкое внедрение и постоянный рост производительности вычислительных систем, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;

● внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

Успешное внедрение CASE-средств должно обеспечить:

● высокий уровень технологической поддержки процессов разработки и сопровождения ПО;

● положительное воздействие на производительность, качество продукции, соблюдение стандартов, документирование;

● приемлемый уровень отдачи от инвестиций в CASE-средства.

# 1.Методологические основы проектирования информационных систем

## 1.1. Технология проектирования ЭИС

***Проект ЭИС*** – это проектно-конструкторская и технологическая документация, содержащая описание проектных решений по созданию и эксплуатации информационной системы в конкретной программно-технической среде.

***Управление проектом*** связано с вопросами планирования и организации работ, создания коллективов разработчиков и контроля за сроками и качеством выполняемых работ.

***Процесс проектирования ЭИС*** представляет собой преобразование входной информации об объекте и методах проектирования в проект ЭИС в соответствии с ГОСТом.

Объекты проектирования ЭИС – это элементы (задачи), комплекс задач функциональной и обеспечивающей частей. В состав обеспечивающей части ЭИС входят элементы и их комплексы информационного, программного и технического обеспечения системы.

Субъектами проектирования ЭИС являются проектная организация и организация-заказчик.

**I. Технология проектирования ЭИС** – это комплекс методологий и средств проектирования, а также методов и средств организации проектирования.

Технологию проектирования ЭИС можно представить графически в виде технологического процесса проектирования ЭИС, отображающего последовательность действий, необходимые средства и ресурсы для выполнения этих действий и состав исполнителей.

В процессе работы над проектом необходимо знать ЧТО, КАК, КОМУ и в КАКОЙ ПОСЛЕДОВАТЕЛЬНОСТИ это должно быть сделано.

**Требования** к выбираемой технологии проектирования:

* проект по выбранной технологии должен отвечать требованиям заказчика;
* максимальное отображение всех этапов жизненного цикла проекта;
* обеспечение минимум трудовых и стоимостных затрат на проектирование и сопровождение проекта;
* технология должна способствовать росту производительности труда проектировщика;
* простота ведения проектной документации;
* технология должна обеспечивать надежность процесса проектирования и эксплуатации проекта.

*Методология* проектирования предполагает наличие концепции, принципов проектирования, реализуемых набором методов, которые должны поддерживаться средствами проектирования.

*Методы* проектирования ЭИС можно классифицировать:

* по степени *автоматизации:*
  + ручное – без использования специальных инструментальных программных средств, а программирование на алгоритмических языках;
  + компьютерное – генерация или конфигурация (настройка) проектных решений на основе использования специальных инструментальных программных средств;
    - по степени *использования типовых проектных решений:*
      * оригинальное (индивидуальное) – разработка с «нуля»;
      * типовое – настройка ЭИС из готовых типовых проектных решений – программных модулей;
        + по степени *адаптивности проектных решений* методы проектирования классифицируются на методы:

реконструкции - адаптация, путем программирования модулей;

параметризации - настройка в соответствии с изменяемыми параметрами;

реструктуризации модели - изменение модели проблемной области, на основе которой автоматически перегенерируются проектные решения.

Сочетание различных признаков классификации методов проектирования обуславливает характер используемой технологии проектирования ЭИС, среди которых можно выделить два основных класса (см. табл.1.1):

1. каноническая – для небольших локальных ЭИС;
2. индустриальная технология, подразделяющаяся на:

* автоматизированное (использование CASE-технологий);
* типовое (параметрически-ориентированное или модельно-ориентированное).

*Таблица 1.1*

**Характеристики классов технологий проектирования**

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс технологии проектирования** | **Степень автоматизации** | **Степень типизации** | **Степень адаптивности** |
| Каноническое проектирование | Ручное проектирование | Оригинальное проектирование | Реконструкция |
| Индустриальное проектирование: *автоматизированное* | Компьютерное  проектирование | Оригинальное проектирование | Реструктуризация модели  (генерация ЭИС) |
| Индустриальное проектирование:  *типовое* | Компьютерное  проектирование | Сборочное  проектирование | Параметризация и реструктуризация модели (конфигу-  рация ЭИС) |

Конкретные виды технологий проектирования требуют выбора средств проектирования, которые по своим характеристикам максимально соответствовали бы требованиям конкретного предприятия.

Средства проектирования должны обладать следующими качествами:

* + охватывать все этапы жизненного цикла ЭИС;
  + совместимы технически, программно и информационно;
  + просты в освоении и применении;
  + экономически целесообразны.

Средства проектирования ЭИС можно разделить на два класса:

1) без использования ЭВМ на всех стадиях проектирования (средства организационно-методического обеспечения операций проектирования, куда входят стандарты, регламентирующие процесс проектирования систем, Единая Система Классификации и Кодирования информации, унифицированная система документации (УСД), модели описания и анализа потоков информации и т.п.);

2) с использованием ЭВМ, которые подразделяются на 4 подкласса (см. табл. 1.2):

*Таблица 1.2*

**Подклассы автоматизированных средств проектирования ЭИС**

|  |  |  |  |
| --- | --- | --- | --- |
| **I** | **II** | **III** | **IV** |
| ***Операционные средства:*** | ***Средства общесистемного***  ***назначения:*** | ***Функциональные средства:*** | ***Средства***  ***автоматизации проектирования ИС*** |
| Алгоязыки, библиотеки стандартных подпрограмм и классов объектов, и т.д.  Средства для тестирования и отладки программ, под­держки процесса документирования проекта и т.д. | -СУБД,  -методоориентиро-ванные программы (задачи дискретного программирования, математической статистики и др.),  -табл. процессоры,  -статистические программы,  -графические, текстовые редакторы,  -оболочки экспертных систем,  -интегрированные программы | - типовые проектные решения,  - функциональные программы,  - типовые проекты | - средства, поддерживающие разработку проекта –*CASE–технологии* |

**Жизненный цикл** (ЖЦ) **ИС *–*** совокупность стадий и этапов, которые проходит ИС в своем развитии от момента принятия решения о создании системы до момента прекращения ее функционирования. Другими словами, ЖЦ ИС – это непрерывный процесс, начинающийся с момента принятия решения о создании информационной системы и заканчивающийся в момент полного изъятия ее из эксплуатации [12].

Основным стандартом, определяющим структуру жизненного цикла, является ГОСТ Р ИСО/МЭК 12207-02 [5]. Согласно стандарту, структура жизненного цикла основывается на трех группах процессов:

- [*основные процессы*](http://edu.dvgups.ru/METDOC/GDTRAN/YAT/ITIS/PROEK_INF_SIS/METOD/UMK_DO/frame/UMK_DO/M1/L2.HTM#2_2) (заказ, поставка, разработка, эксплуатация, сопровождение);

- *вспомогательные процессы* (обеспечивают выполнение основных процессов):

- *документирование* – работы по разработке, выпуску, редактированию, распространению и сопровождению документов, в которых нуждаются все заинтересованные лица;

- *управление конфигурацией* (конфигурационное управление) включает работы: определение и установление состояния программных объектов в системе; управление изменениями и выпуском объектов; обеспечение полноты, совместимости и правильности объектов; управление хранением, обращением и поставкой объектов;

- *обеспечение качества* – работы по обеспечению соответствия создаваемой системы и реализуемых процессов жизненного цикла установленным требованиям и утвержденным планам;

- *верификация* – работы соответствующего субъекта (заказчика, поставщика или независимой стороны) по проверке соответствия создаваемых промежуточных результатов установленным требованиям по мере реализации проекта. Различают верификацию договора, процесса, требований, проекта, системы, сборки системы и документации;

- *аттестация* – работы соответствующего субъекта по проверке полного соответствия требований и конечного продукта функциональному назначению системы;

- *совместный анализ* – работы по оценке состояния или результатов какой-либо работы (системы);

- *аудит* – работы независимых (по отношению к проекту) экспертов по определению соответствия деятельности субъекта принятым требованиям, планам и условиям договора;

*- разрешение проблем* – работы по анализу и устранению проблем, обнаруженных при реализации проекта;

- *организационные*:

- *управление проектами* – работы по планированию и управлению процессами, включая контроль, проверку и оценку выполненных работ с формированием отчетности;

- *создание инфраструктуры проекта* – работы по установлению и обеспечению инфраструктуры, необходимой для любого другого процесса. Инфраструктура может содержать технические и программные средства, инструментальные средства, методики, стандарты и условия для разработки, эксплуатации или сопровождения системы;

- *усовершенствование* – работы по оценке, контролю и улучшению процессов жизненного цикла;

- *обучение* – работы по планированию и проведению обучения персонала, включая разработку учебных материалов. При этом под персоналом понимаются не только конечные пользователи, которые будут эксплуатировать систему, но и разработчики системы. Например, разработчики должны быть обучены технологиям и средствам программирования, принятым в организации, и даже обучены правильно внедрять и обучать конечных пользователей работе с системой. Как бы это ни парадоксально звучало, но обучать правильной методике и приемам обучения тоже необходимо.

### 1.1.1. Распределение обязанностей между участниками проекта

В процессе разработки и эксплуатации системы участвует определенный круг лиц (представители заказчика и разработчика), заинтересованных в успешной реализации проекта. В этом процессе между ними распределяются роли, за каждой из которых закрепляется определенный набор функций (обязанностей). При этом один и тот же человек может выступать в разных ролях (качествах). Так, например, один и тот же человек может быть проектировщиком и программистом, в то же время в проекте может принимать участие несколько экспертов, проектировщиков или программистов. В табл. 1.3 приведен типичный список ролей и их функций.

Таблица 1.3.

**Роли участников в проекте**

|  |  |
| --- | --- |
| **Роль** | **Функции** |
| Руководитель (менеджер) проекта | Ищет потенциальных заказчиков. (поиск и выбор) Заключает договор на разработку системы. Отвечает за планирование сроков и ресурсов. Выполняет управление и контроль за ходом выполнения проекта. Отвечает за взаимодействие с заказчиком |

Продолжение таблицы1.3.

|  |  |
| --- | --- |
| Эксперт-технолог | Делает постановку задачи. Определяет (совместно с системным аналитиком) основные функциональные и нефункциональные требования к системе. Определяет технологию использования разрабатываемой системы. Консультирует разработчиков в процессе создания системы. Участвует в процессе приемки системы в эксплуатацию |
| Системный аналитик (архитектор, главный конструктор) | Определяет функциональные и нефункциональные требования к системе, а также технологию ее использования. Выполняет анализ требований и ищет пути их реализации на уровне концепции системы. Задает архитектуру (скелет) системы и несет ответственность за соответствие моделей системы заданной архитектуре (отвечает за проектирование). Квалифицированный аналитик должен быть специалистом в области разработки программного обеспечения и должен быть (стать) специалистом в предметной области |
| Проектировщик | Разрабатывает модели системы на основе архитектуры |
| Программист | Реализует модели в виде программного обеспечения |
| Тестировщик | Разрабатывает тесты и тестирует модели системы и разработанное программное обеспечение |
| Технический редактор (писатель) | Готовит документацию для пользователей на разработанную систему. В комплект документации могут входить технологические инструкции, руководства пользователя, администратора системы, БД и т. д. |
| Инженер по внедрению | Внедряет разработанную систему на объекте автоматизации. В его функции может входить как первоначальная установка и настройка системы, так и обучение пользователей |
| Пользователь | Эксплуатирует систему в штатном режиме. Кроме этого, желательно, чтобы пользователь (помимо эксперта-технолога) был вовлечен в процесс формирования требований к системе |

У проекта должен быть один руководитель и, как правило, один системный аналитик. За остальные роли в крупных проектах отвечает обычно по несколько человек. Эксперты-технологи могут быть приглашены из сторонней организации. По мере необходимости в проекте могут принимать участие координатор работ (ответственный) со стороны заказчика, аудиторы и т.д.

Неотъемлемой чертой ЖЦ экономической ИС (ЭИС) является его повторяемость: **системный анализ – разработка – сопровождение – системный анализ**.

При первом выполнении стадии «разработка» создается проект ЭИС, а при повторном – осуществляется модификация проекта для поддержания его в актуальном состоянии.

### 1.1.2. Модели жизненного цикла ИС

Технологии проектирование ЭИС, определяющие порядок выполнения стадий и этапов претерпевали существенные изменения. Среди известных *моделей ЖЦ ИС* можно выделить следующие модели:

* ***Каскадная*** (до 70-х годов XX века) – последовательный переход на следующий этап после завершения предыдущего. Применима для отдельных несвязных задач, не требующая выполнения информационной интеграции и совместимости программного, технического и организационного сопровождения. Применение каскадной модели к большим и сложным проектам вследствие большой длительности процесса проектирования и изменчивости за это время требований, приводит к их практической не реализуемости (см. рис.1.1).
* ***Итерационная*** (70-80 гг. XX века) – с итерационными возвратами на предыдущие этапы после выполнения очередного этапа, т.е. осуществляется проектирование ***«снизу–вверх»***, когда проектные решения по отдельным задачам комплектуются в общие системные решения. При этом возникает рассогласованность в выполненных проектных решениях и документации из-за большого числа итераций и возникновении необходимости пересмотра ранее выдвинутых требований (см. рис.1.2).

**Рис. 1.1.** Каскадная схема разработки ПО.

**Анализ**

**Проектирование**

**Реализация**

**Внедрение**

**Сопровождение**

**Внедрение**

**Реализация**

**Проектирование**

**Анализ**

**Сопровождение**

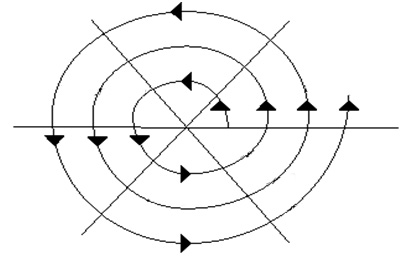
**Рис. 1.2.** Реальный процесс разработки ПО по каскадной схеме.

* ***Спиральная*** (80-90гг. XX века) – проектирование ЭИС ***«сверху – вниз»,*** при котором сначала определяется состав функциональных подсистем ЭИС (комплекс задач с высокой степенью информационных обменов (связей) между задачами),

Например, начисление зарплаты; учет материалов; оформление заказов на покупку и т.д., управление производством, сбытом продукции, МТС, финансами, персоналом, а затем постановка отдельных задач (см. рис.1.3).

Первоначально разрабатываются общесистемные вопросы:

* организация интегрированной БД;
* технология сбора, передачи, накопления, хранения информации, затем решение задач.



**Реализация и тестирование**

**Определение требований**

**Версия 1**

**Версия 2**

**Версия 3**

**Интеграция**

**Проектирование**

**Анализ**

**Рис 1.3.** Спиральная модель жизненного цикла ИС.

Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная же задача - как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали.

В основе спиральной модели ЖЦ ИС лежит прототипная технология или RAD – технология (технология быстрой разработки приложений). Разработка ЭИС при этой технологии осуществляется путем расширения программных прототипов.

Достоинство. Сокращается число итераций, уменьшается число ошибок и несоответствий, ускоряется процесс проектирования ЭИС, упрощается создание проектной документации.

Для более точного соответствия проектной документации разработанной ЭИС все большее значение придается ведению общесистемного репозитория и использованию CASE-технологий.

* 1. Основой формализации технологии проектирования ЭИС является определение технологической операции (ТО) проектирования (см. рис.1.4):

**Рис. 1.4.** Составляющие технологической операции.

#### V

**П : R, S**

#### W

#### V

**D j1**

**U j2**

**P j3**

**Gj4**

**D i1**

**U i2**

**Pi3**

**Gi4**

Компонентами входа и выхода технологической операции являются:

**П** (преобразователь) – методика или формализованный алгоритм, или машинный алгоритм. Используются ручные, автоматизированные, автоматические методы реализации преобразователей.

**D**(документ) – описатель множества взаимосвязанных фактов.

**P** (параметр) – описатель одного факта (частный случай документа). Например, срок разработки, объем финансирования, ….

**U**(универсум) – конечное и полное множество фактов (документов) одного типа. Например, описание технических, программных средств (ОС, СУБД, ППП и т.д.), технологий проектирования.

**G** (программа) – описание алгоритма решения задачи (от спецификации программы до машинного кода).

**R** (ресурсы) – людские, технические, временные, финансовые, позволяющие выполнить технологическую операцию.

**S** (средства) – специальный вид ресурса, включающий методические и программные средства выполнения ТО (в большей степени ориентировано на проектировщика).

На основе технологических операций строится технологическая сеть проектирования (ТСП) – взаимосвязанная по входам и выходам последовательность ТО, выполнение которых приводит к достижению требуемого результата – созданию проекта. ТСП могут строиться с различной степенью детализации.

Стадии разработки ЭИС:

1. *Планирование и анализ требований* (предпроектная стадия) – системный анализ. Исследуется и анализируется существующая ИС, определяются требования к новой ЭИС, оформляются технико-экономическое обоснование (**ТЭО**) и техническое задание (**ТЗ**) на разработку ЭИС.
2. *Проектирование* (техническое и логическое проектирование). Разработка состава автоматизируемых функций и состава обеспечивающих подсистем; оформление технического проекта (**ТП**) на ЭИС.
3. *Реализация* (рабочее, физическое проектирование, программирование). Разработка и настройка программ, наполнение БД, создание рабочих инструкций для персонала, оформление рабочего проекта (**РП**).
4. *Внедрение* (тестирование, опытная эксплуатация). Комплексная отладка подсистем ЭИС, обучение персонала, поэтапное внедрение ЭИС в эксплуатацию по подразделениям объекта, оформление акта о приемо-сдаточных испытаниях ЭИС.
5. *Эксплуатация* (сопровождение, модернизация). Повторение стадий 2-4 и их доработка.

Целью «**Анализа материалов обследования»** является сопоставление собранной об объекте информации выдвигаемым требованиям; выработка направлений совершенствования работы объекта на базе внедрения проекта ЭИС, выбор инструментария проектирования и оценка эффективности его применения; определение общесистемных, функциональных и локальных требований к проекту и его частям.

Анализ материалов обследования позволяет разработчикам выделить и составить *список автоматизируемых подразделений* (отделы технико-экономического планирования, оперативного управления основным производством, технической подготовки производства, МТС, реализации и сбыта готовой продукции, бухгалтерия; выявить *список автоматизируемых задач* (связь с другими задачами, трудоемкость и оперативность расчета показателей, их достаточность, достоверность выходных данных, очередность проектирования решаемых задач и т.д.); *предварительный выбор комплекса технических средств*;  *выбор типа операционных систем; выбор способа организации информационной базы (ИБ)* (локальные, интегрированные[[1]](#footnote-1)\*) *и программного средства ведения ИБ* (выбираются исходя из класса хранения данных: систем управления файлами, или СУБД); *выбор методов и средств проектирования ПО системы*, зависящий от выбранной технологии проектирования.

Выполнение всех этих операций завершается составлением **ТЭО** и формированием **ТЗ**. Цель ***ТЭО*** разработки проекта – оценка параметров: организационных, информационных и экономических. Параметризация позволяет определить требования к системе, оценить существующую ИС, определить пригодность типовых решений в проекте ЭИС, выбрать проектные решения в соответствии с требованиями к ЭИС.

На основе ТЭО разрабатываются основные требования к будущему проекту ЭИС и составляется «*Техническое задание*» (**ТЗ**), включающего следующие разделы:

* *«Общие сведения о проекте»* - наименование и код системы, код договора, наименования организации-разработчика и организации-заказчика, перечень документов, являющихся основой для создания системы, сроки начала и окончания разработки ЭИС, источники финансирования, порядок оформления и предъявления заказчику результатов работ по созданию системы (ее частей);
* *«Назначение и цели создания ЭИС»*;
* *«Характеристика объекта автоматизации»*;
* *«Требования к системе*» состоят из следующих подразделов:
* требования к системе в целом – к структуре и функционированию системы; к численности квалифицированных работников; к надежности и безопасности работы системы; эксплуатации, техническому обслуживанию; к сохранности информации при аварийных ситуациях; требования к унификации и стандартизации и т.д.;
* требования к функциям (задачам), выполняемым системой – очередность их создания; временной регламент и качество выполнения каждой функции, задачи или комплекса; форма представления выходной информации; достоверность и т.д.;
* требования к видам обеспечения – математическому, программному, техническому, информационному и методическому.
* *«Состав и содержание работ по созданию системы»* - перечень стадий и этапов работ по созданию системы.
* *«Порядок контроля приемки системы»* – виды, состав, методы испытания системы и ее частей, общие требования к приемке работ по стадиям, порядок утверждения приемных документов, статус приемочной комиссии;
* *«Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие»* – перечень мероприятий и исполнителей (приведение информации к виду, пригодному для ввода ее в ЭВМ, создание для функционирования системы подразделений и служб, сроки и порядок комплектования штатов, и обучение персонала);
* *«Требования к документированию» –* разработка документов и научно-технической документации отрасли заказчика;
* *«Источники разработки» –* приводятся документы и информационные материалы (ТЭО, отчеты о законченных научно-исследовательских разработках, данные на отечественные, зарубежные системы-аналоги и др.);

**Техническое задание** (ТЗ) включает в себя утвержденные методики, приложения с расчетами экономической эффективности системы, оценку научно-технического уровня системы [18].

На основе утвержденного ТЗ начинается стадия «***Технорабочего проектирования***», включающая два этапа работ: техническое и рабочее проектирование.

**Техническое проектирование** содержит:

* разработку общесистемных проектных решений (уточняются цели создания ЭИС и выполняемые ею функции, устанавливается ее взаимосвязь с другими системами); уточняется и изменяется организационная структура; разрабатывается функциональная архитектура ЭИС;
* разработку локальных проектных решений, к числу которых относятся следующие операции:
  + разработка блока «Постановка задачи» (см. рис.1.5), для задач, входящих в состав функциональных подсистем;
  + проектирование форм входных и выходных документов, макетов экранных форм документов, классификаторов экономической информации;
  + проектирование состава и структур файлов информационной базы;
  + проектирование внемашинной и внутримашинной технологии решения каждой задачи;
  + уточнение состава технических средств.

Перечень и описание входных документов и сообщений

#### Постановка задачи

Цель и назначение задачи

Экономическая и организационная сущность задачи

Описание алгоритмов задачи

Периодичность решения и связи с другими задачами

Требования к организации сбора информации

***Входная информация***

***Выходная информация***

Перечень структурных единиц информации

Наименование, идентификатор, форма, требования к точности, источник

Перечень и описание выходных документов и сообщений

Перечень структурных единиц информации

Наименование, идентификатор, форма, периодичность, сроки

получения

**Рис.1.5.** Функциональные подсистемы блока «Постановка задачи».

В состав блока «Характеристика задачи» входят:

* описание *цели* автоматизации решения задачи (получение экономического эффекта в сфере управления процессами системы, снижение стоимостных и трудовых затрат на обработку информации, повышение качества и достоверности получаемой информации, оперативности её обработки и т.д.;
* описание *экономической сущности* решаемой задачи, т.е. состава экономических показателей, рассчитываемых при ее решении, документов, куда заносятся эти показатели;
* описание *организационной сущности* задачи – описание порядка решения задачи, режима решения, состава файлов с постоянной и переменной информацией, способами и формами ввода и вывода результатной информации и т.д.;
* описание *алгоритма* – формализованное описание входных и результатных показателей, перечень формул расчета, описание математической модели, экономико-математических методов и т.д.

Для каждой задачи разрабатываются все компоненты информационного (см. рис. 1.6), технического, математического и лингвистического обеспечения, а также компоненты программного обеспечения.

***Информационное обеспечение***

**Форм документов**

**Структуры массивов**

**Систем классификации и кодирования**

**Входных и выходных сообщений**

**Организации ИБ**

**Технологического процесса**

**Информационного обеспечения**

**Описание**:Перечень и описание входных документов и сообщений

**Рис.1.6.** Состав подсистем информационного обеспечения ЭИС

На этапе «Рабочего проектирования» выполняется «*Кодирование и составление программной документации*», в состав которой входит:

* описание программ;
* спецификация программ;
* тексты программ;
* контрольные примеры;
* инструкции для программиста, оператора и пользователя.

В состав «Рабочего проекта» входит *технологическая документация,* включающая технологические карты, разрабатываемые на процессы обработки информации при решении задач, и инструкционные карты, составляемые на каждую технологическую операцию.

В процессе рабо­чего проектирования должен быть проведен комплекс меропри­ятий по подготовке объекта управления к внедрению ИС:

* сформирован комплекс технических средств;
* завершены созда­ние информационной базы,
* кодирование и отладка программ,
* генерация и настройка пакетов прикладных программ (ППП) (приложений);
* обеспе­чено обучение пользователей и обслуживающего персонала;
* разработаны и утверждены технология обработки и должност­ные инструкции, определяющие порядок работы соответствую­щих специалистов в условиях автоматизированной обработки.

*Примечание.* Стадии ***технического*** и ***рабочего*** проектирования могут быть совмещены в том случае, когда имеются проверенные эффективные решения по основным видам обеспечения ИС, пригодные для применения в конкретном проекте (по информационному обеспечению – способы сбора и организации данных: структура данных, классификаторы,…; функциональному, программному обеспечению – ОС, трансляторы, СУБД,…).

***Внедрение проекта*** включает три этапа:

1. *Подготовка объекта к внедрению* (изменяется организационная структура, набор квалифицированных кадров в области обработки информации, эксплуатации системы и сопровождения проектной документации; приобретение и установка ВТ, оргтехники и т.д.; установка каналов связи, разработка новых документов и классификаторов, создание информационной базы с нормативно-справочной информацией). Результатом является «*Акт готовности объекта к внедрению*»;
2. *Опытное внедрение – подготовка исходных оперативных данных; ввод данных в ЭВМ и реализация поставленных решений; анализ результатных данных на наличие ошибок. При выявлении ошибочных ситуаций, вносятся коррективы в программы, технологию обработки информации, в работу технических средств и т.п. После устранения ошибок составляется «Акт о проведении опытного внедрения»;*
3. *Сдача проекта в промышленную эксплуатацию сопровождается следующей документацией:*

* договорная документация;
* «Приказ на разработку ЭИС»;
* исправленный «ТРП»;
* «Приказ о начале промышленного внедрения»;
* «Программа проведения испытаний»;
* «Требования к научно-техническому уровню проекта системы».

В процессе сдачи проекта в промышленную эксплуатацию осуществляются такие работы, как проверка соответствия выполненной работы договорной документации, соответствие проектной документации ГОСТам и ОСТам, проверка качества функционирования информационной базы, оперативности и полноты ответов на запросы и т.д.

Приемная комиссия определяет научно-технический уровень проекта и возможности расширения проектных решений за счет включения новых компонентов. В результате выполнения работ на данном этапе осуществляется доработка «**ТРП**» за счет выявления системных и локальных ошибок и составляется «***Акт сдачи проекта в промышленную эксплуатацию***».

На стадии *«Эксплуатация и сопровождение проекта»* осуществляется устранение сбоев в системе и регистрация их в журналах, отслеживание технико-экономических характеристик системы и накопление статистики о качестве работы всех компонентов системы, определение объемов доработок, сроков и стоимости на выполнение этих работ.

## 1.2. Автоматизированное проектирование ЭИС (CASE-технология)

Большинство CASE (Computer Aided System / Software Engineering) – систем ориентировано на автоматизацию проектирования ПО и основано на методологиях структурного (в основном) или объектно-ориентированного проектирования и программирования, использующих спецификации в виде диаграмм или текстов для описания системных требований, связей между моделями системы и т.д. [5]

Необходимость в использовании CASE-систем ощущается на начальных этапах анализа и спецификации требований к ЭИС. Это вызвано тем, что цена ошибок, допущенных на начальных этапах, на несколько порядков превышает цену ошибок, выявленных на более поздних этапах разработки.

*Преимущества использования CASE-технологии* сводятся к следующему:

* улучшение качества разрабатываемого программного приложения за счет средств автоматического контроля и генерации;
* возможность повторного использования компонентов разработки;
* поддержание адаптивности и сопровождения ЭИС;
* снижение времени на создание системы, что позволит на ранних стадиях проектирования получить прототип будущей системы и оценить его;
* освобождение разработчиков от рутинной работы по документированию проекта, за счет использования встроенного документатора;
* возможность коллективной разработки ЭИС в режиме реального времени.

CASE-технология в рамках методологии включает в себя методы (см. рис. 1.7), с помощью которых на основе графической нотации строятся диаграммы, поддерживаемые инструментальной средой.

***CASE-средства***

***Репозиторий***

***(словарь данных)***

Графический редактор диаграмм

Верификатор диаграмм

Администратор проекта

Документатор

проекта

Сервис

**Рис.1.7.** Архитектура CASE-средства

Инструментальные CASE-средства– специальные программы, поддерживающие одну или несколько методологий анализа и разработки ИС.

***Репозиторий*** (словарь данных), является ядром системы – представляет собой специализированную базу данных проекта, которая предназначена для отображения состояния разрабатываемой ЭИС в каждый момент времени.

В репозитории хранятся описания таких объектов, как:

* проектировщики и их права доступа к различным компонентам системы;
* организационные структуры;
* диаграммы и связи между ними;
* компоненты диаграмм;
* структуры данных;
* программные модули;
* процедуры;
* библиотеки модулей и т.д.

Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую систему, перестраивать ее в соответствии с поставленными целями и ограничениями.

*Графический редактор* позволяет выполнять следующие операции:

* создавать элементы диаграмм, их взаимосвязи и описания;
* задавать описания элементов диаграмм и связей между ними;
* редактировать элементы диаграмм, их взаимосвязи и описания.

*Верификатор* – служит для контроля правильности построения диаграмм в заданной методологии проектирования ЭИС.

*Документатор* проекта позволяет получать информацию о состоянии проекта в виде различных отчетов, которые строятся по нескольким признакам (по времени, автору, элементам диаграмм, диаграмме или проекту в целом).

*Администратор* проекта – это инструменты, необходимые для выполнения таких функций, как:

* инициализация[[2]](#footnote-2) проекта;
* задания начальных параметров проекта;
* назначения и изменения прав доступа к элементам проекта;
* мониторинга выполнения проекта.

*Сервис* – набор системных утилит по обслуживанию репозитория (архивация данных, восстановления данных и создания нового репозитария).

Выбор CASE-систем зависит от целей и потребностей самого проекта, а также от квалификации специалистов, участвующих в процессе проектирования. Как правило, используется несколько средств, так как одно необходимо для анализа, другое – для проектирования систем. Поэтому, при выборе CASE-системы необходимо учитывать следующее:

* *наличие базы проектных данных, архива или словаря*, предоставляющие широкие возможности для централизованного сбора, хранения и распределения проектной информации между различными этапами проекта и выполняемыми операциями;
* *интерфейс с другими CASE-системами*. Использование различных методологий;
* *расширение новыми методологиями* с учетом появления новых требований;
* *возможности экспорта/импорта* спецификаций в различные CASE-системы;
* *обеспечение качества проектной документации:* проверка на полноту и непротиворечивость, на соответствие принятым в данной методологии стандартам и правилам;
* *автоматическая генерация отчетов о проектных решениях;*
* *генерация кодов программ;*
* *планирование и управление проектом.*

### 1.2.1. Подходы к проектированию ИС

В настоящее время существует два подхода к разработке ПО ЭИС, различающиеся между собой разными способами декомпозиции систем.

Первый, ***функционально-ориентированный*** или ***структурный***, в основе своей содержит принцип функциональной декомпозиции, при которой структура системы описывается в терминах иерархии ее *функций* и передачи информации между отдельными функциональными элементами. Сущность структурного подхода заключается в декомпозиции на автоматизируемые функции, затем подфункции, т.е. на задачи и т.д. до конкретных процедур. При разработке системы «снизу вверх» теряется целостность, возникают проблемы при описании информационного взаимодействия отдельных компонентов.

В структурном подходе используется в основном две группы средств, описывающих функциональную структуру системы и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными являются:

DFD – диаграммы потоков данных;

SADT – метод структурного анализа и проектирования (модели и соответствующие функциональные диаграммы);

ERD – диаграммы «сущность-связь».

Диаграммы потоков данных и диаграммы «сущность-связь» - наиболее часто используемые в CASE–средствах виды моделей.

Второй, ***объектно-ориентированный*** подход использует объектную декомпозицию. При этом структура системы описывается в терминах *объектов* и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами.

В основе *функционально-ориентированной* CASE-технологии лежит структурный анализ и проектирование информационных систем, заключающееся в следующем:

1. декомпозиция всей системы на множество иерархически подчиненных функций;
2. представление всей информации в виде графической нотации.

Методология *функционального проектирования* SADT была разработана **Дугласом Россом** в 1973 году и получила дальнейшее развитие в известной методологии **IDEF0.**

Методология SADT – это совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области.

**Функциональная модель SADT** отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями (см. рис. 1.8).

**Функция**

**Вход**

**Управление**

**Механизм**

**Выход**

**Рис.1.8.**Модель управления функцией (бизнес-процессом).

Построение **SADT–** модели начинается с представления всей системы в виде одного блока и дуг, отображающих интерфейсы с функциями вне системы. Блок, представляющий систему в целом в дальнейшем детализируется на подфункции. Каждая из подфункций может быть декомпозирована в целях большей детализации.

*Контекстная диаграмма* – самый верхний процесс (ТОР-уро­вень) декомпозиции системы, который отражает общие представ­ления о системе. В контекстной диаграмме есть один процесс, с кото­рым связаны внешние сущности.

Модель **SADT**представляет серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части (блоки).

Сущность **структурного подхода** к разработке ИС заключается в ее ***декомпозиции*** (***разбиении***) на ***автоматизируемые функции***.

Система разбивается на:

* функциональные подсистемы, которые в свою очередь делятся на
* подфункции, подразделяемые на
* задачи и так далее.

Процесс разбиения ***продолжается*** вплоть ***до конкретных процедур***. При этом автоматизируемая *система сохраняет целостное представление*, в котором *все составляющие компоненты взаимоувязаны*.

Одной из наиболее важных особенностей методологии SADT является постепенное введение все больших уровней детализации по мере создания диаграмм, отображающих модель.

Методология SADT может использоваться для моделирования широкого круга систем и определения требований и функций, а затем для разработки системы, которая удовлетворяет этим требованиям и реализует эти функции.

## 1.3. Объектно-ориентированное проектирование ЭИС

Принципиальное различие между структурным и объектно-ориентированным подходом заключается в способе декомпозиции системы. Объектно-ориентированный подход использует *объектную* декомпозицию, при этом статическая структура системы описывается в терминах *объектов* и связей между ними, а поведение системы описывается в терминах *обмена сообщениями* между объектами. Каждый объект системы обладает своим собственным поведением, моделирующим поведение объекта реального мира. Наиболее значительный вклад в объектный подход был внесен объектными и объектно-ориентированными языками программирования: Simula, Smaltalk, C++, Object Pascal. На объектный подход оказали влияние развивающиеся достаточно независимо методы моделирования баз данных, в особенности подход «сущность-связь».

Объектно-ориентированную разработку ИС характеризует использование объектно-ориентированных моделей при создании программных систем и их элементов, а точнее применение [13]:

* **объектно-ориентированной методологии** (технологии) разработки сложных ИС (программных систем);
* последние версии **CASE-средств**, поддерживающих эти технологии.

Под термином ***CASE-средства*** понимаются программные средства, поддерживающие процессы *создания* и *сопровождения информационных* систем, включая *анализ и формулировку требований*, *проектирование прикладного ПО (приложений)* и *баз данных*, *генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы.”*

В этой же работе определяются основные компоненты CASE-пакета:

1. репозиторий;
2. графические редакторы;
3. средства разработки готовых приложений, включая генераторы конечного кода по диаграммам;
4. средства конфигурационного управления проектом;
5. средства документирования проекта;
6. средства тестирования;
7. средства управления проектом;
8. средства реинжиниринга.

В **объектной методологии** проектирования ИС используются **объекты,** обладающие тремя **главными свойствами**:

1. ***наследование,*** т.е. возможность строить дерево классов с наследованием ***методов*** и ***структур данных;***
2. ***полиморфизм***, когда один и тот же метод (операция или функция) может приводить к разным результатам на различных классах;

***Полиморфизм*** – способность объектов выбирать метод на основе типов данных, применяемых в сообщении. Каждый объект может реа­гировать по-своему на одно и то же сообщение. Полиморфизм позволяет упростить исходные Тексты программ, обеспечивает их развитие за счет введения новых методов обработки.

1. ***Инкапсуляция –*** реализация метода происходит внутри объекта, для других объектов доступен только интерфейс, по которому они связываются с данным объектом.

***Инкапсуляция*** – скрытие информации. При объектно-ориетированном программировании предусмотрена возможность запрета доступа к атрибутам объектов, кроме как через его методы. Внутренняя структура объекта в этом случае скрыта от пользователя, т.е. объекты можно считать самостоятельными сущностями, отделенными от внешнего мира. Для того чтобы объект произвел некоторое действие, ему извне необходимо послать сообщение, которое инициирует вы­полнение нужного метода. Инкапсуляция позволяет изменять реали­зацию любого класса объектов без опасения, что это вызовет нежелательные побочные эффекты в программной системе. Тем самым упрощается процесс исправления ошибок и модификации программ.

При разработке сложных ИС все чаще используется ***компонентная архитектура***. Программа представляется в виде совокупности компонент ***с простыми*** и ***четко специфицированными интерфейсами***. Этот подход в объектной технологии проектирования позволяет ***разрабатывать каждую компоненту независимо***, реализовывать компоненты так, чтобы они могли работать в распределенной среде, модифицировать одну из компонент ПО, оставляя неизменными все остальные, и т.д.

**Недостаток структурного подхода**:

* + процессы и данные существуют отдельно друг от друга, причем проектирование ведется от процессов к данным. Т.е., помимо функциональной декомпозиции, существует также структура данных, находящаяся на втором плане.

В объектно-ориентированном подходе основная категория объектной модели – класс – объединяет в себе как данные, так и операции, которые над ними выполняются (методы). Данные по сравнению с процессами являются более стабильной и относительно редко изменяющейся частью системы. Поэтому, главным **достоинством объектно-ориентированного подхода** по определению Г. Буча в том, что объектно-ориентированные системы более открыты и легче поддаются внесению изменений, поскольку их конструкция базируется на устойчивых формах. Это дает возможность системе развиваться постепенно и не приводит к полной ее переработке даже в случае существенных изменений исходных требований.

### 1.3.1. Методология DATARUN

Высокая динамичность рынка требует от организаций быстрого развития информационно-технологической инфраструктуры. Одной из ее наиболее важных и дорогостоящих составляющих является информационная система, для реализации которой применяются современные технологии: архитектура клиент/сервер, распределенные базы данных, сложные сети коммуникаций, развитые интерфейсы пользователя. Все это ставит перед разработчиком проблему выбора инструментальных средств и технологий для ведения проекта. Современные методологии и реализующие их технологии поставляются в ***электронном*** виде вместе с CASE-средствами и включают библиотеки процессов, шаблонов, методов, моделей и других компонент, предназначенных для построения ПО того класса систем, на который ориентирована методология. Электронные методологии включают также средства, которые должны обеспечивать их *адаптацию* для конкретных пользователей и развитие методологии по результатам выполнения конкретных проектов.

Создание сложной информационной системы невозможно без единого интегрированного подхода к процессу разработки. Методология проектирования преследует такие цели, как: обеспечение концептуальной основы для всего процесса разработки; предоставление технологии руководства проектом. Методология DATARUN гарантирует, что на каждой стадии выполняется только важнейшая для целей проекта работа, облегчающая быстрое создание приложений. Повторения и избыточность в спецификациях исключаются, создается управляемая, основанная на моделях форма итеративной разработки. Исходные версии объектов доступны для непосредственного использования на следующих фазах проектного цикла. Создаваемая информационная система описывается рядом последовательных моделей, каждая из которых является развитием предыдущей и наследует правила и данные, определенные в более ранних моделях. Наследование свойств позволяет многократно использовать различные спецификации на всех уровнях прикладного проекта [12].

Методология DATARUN ведет заказчика и разработчика информационной системы по всем этапам жизненного цикла проекта, от стадии первоначальной экономической оценки затрат на проект до выхода реального приложения. Она позволяет координировать и контролировать работу всех групп лиц, занятых в работе над проектом.

Методология DATARUN обеспечена средствами автоматизированной поддержки:

* Для управления проектной деятельностью имеется система Software Engineering Companion, позволяющая детально расписывать ведение проекта, распределять проектные роли среди исполнителей, контролировать выполнение заданий.
* Детальное изложение техник моделирования данных и бизнес-функций, проектирования баз данных, создания приложений содержится в гипертекстовой системе Software Engineering Guidelines.
* Автоматизация проведения проектных работ обеспечивается CASE-системой SILVERRUN.

Предоставляемая этими средствами среда проектирования дает возможность руководителю проекта контролировать выполнение работ. Каждый участник проекта, подключившись к системе, может уточнить содержание и сроки выполнения порученной ему работы, изучить технику ее выполнения в гипертексте по технологиям, и вызвать инструмент (модуль SILVERRUN) для реального выполнения работы.

Такой автоматизированный комплекс поддержки выполнения проектов, основанный на современной методологии проектирования и эффективном CASE-средстве, создает все необходимые условия для быстрого создания сложных информационных систем с высоким качеством.

### 1.3.2. Методология RAD

Одним из возможных подходов к разработке ПО в рамках спиральной модели ЖЦ является получившая в последнее время широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development). Под этим термином обычно понимается процесс разработки ПО, содержащий 3 элемента:

* небольшую команду программистов (от 2 до 10 человек);
* короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
* повторяющийся цикл, при котором разработчики, по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком.

Команда разработчиков должна представлять из себя группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании ПО с использованием CASE-средств. Члены коллектива должны также уметь трансформировать в рабочие прототипы предложения конечных пользователей.

Жизненный цикл ПО по методологии RAD состоит из четырех фаз:

* фаза анализа и планирования требований;
* фаза проектирования;
* фаза построения;
* фаза внедрения.

На фазе анализа и планирования требований пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности. Определение требований выполняется в основном силами пользователей под руководством специалистов-разработчиков. Ограничивается масштаб проекта, определяются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации данного проекта в установленных рамках финансирования, на данных аппаратных средствах и т.п. Результатом данной фазы должны быть список и приоритетность функций будущей ИС, предварительные функциональные и информационные модели ИС.

На фазе проектирования часть пользователей принимает участие в техническом проектировании системы под руководством специалистов-разработчиков. CASE-средства используются для быстрого получения работающих прототипов приложений. Пользователи, непосредственно взаимодействуя с ними, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. Более подробно рассматриваются процессы системы. Анализируется и, при необходимости, корректируется функциональная модель. Каждый процесс рассматривается детально. При необходимости для каждого элементарного процесса создается частичный прототип: экран, диалог, отчет, устраняющий неясности или неоднозначности. Определяются требования разграничения доступа к данным. На этой же фазе происходит определение набора необходимой документации.

После детального определения состава процессов оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении ИС на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время – порядка 60-90 дней. С использованием CASE-средств проект распределяется между различными командами (делится функциональная модель). Результатом данной фазы должны быть:

* общая информационная модель системы;
* функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;
* точно определенные с помощью CASE-средства интерфейсы между автономно разрабатываемыми подсистемами;
* построенные прототипы экранов, отчетов, диалогов.

Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы. Данное требование вызвано тем, что в традиционном подходе при передаче информации о проекте с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет избежать этой опасности.

В отличие от традиционного подхода, при котором использовались специфические средства прототипирования, не предназначенные для построения реальных приложений, а прототипы выбрасывались после того, как выполняли задачу устранения неясностей в проекте, в подходе RAD каждый прототип развивается в часть будущей системы. Таким образом, на следующую фазу передается более полная и полезная информация.

На фазе построения выполняется непосредственно сама быстрая разработка приложения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Программный код частично формируется при помощи автоматических генераторов, получающих информацию непосредственно из репозитория CASE-средств. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется непосредственно в процессе разработки.

После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения с остальными, а затем тестирование системы в целом. Завершается физическое проектирование системы:

* определяется необходимость распределения данных;
* производится анализ использования данных;
* производится физическое проектирование базы данных;
* определяются требования к аппаратным ресурсам;
* определяются способы увеличения производительности;
* завершается разработка документации проекта.

Результатом фазы является готовая система, удовлетворяющая всем согласованным требованиям.

На фазе внедрения производится обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой). Так как фаза построения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, как правило, на этапе проектирования системы. Приведенная схема разработки ИС не является абсолютной. Возможны различные варианты, зависящие, например, от начальных условий, в которых ведется разработка: разрабатывается совершенно новая система; уже было проведено обследование предприятия и существует модель его деятельности; на предприятии уже существует некоторая ИС, которая может быть использована в качестве начального прототипа или должна быть интегрирована с разрабатываемой.

Следует, однако, отметить, что методология RAD, как и любая другая, не может претендовать на универсальность, она хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика. Если же разрабатывается типовая система, которая не является законченным продуктом, а представляет собой комплекс типовых компонент, централизованно сопровождаемых, адаптируемых к программно-техническим платформам, СУБД, средствам телекоммуникации, организационно-экономическим особенностям объектов внедрения и интегрируемых с существующими разработками, на первый план выступают такие показатели проекта, как управляемость и качество, которые могут войти в противоречие с простотой и скоростью разработки. Для таких проектов необходимы высокий уровень планирования и жесткая дисциплина проектирования, строгое следование заранее разработанным протоколам и интерфейсам, что снижает скорость разработки.

**Методология RAD не применима** для построения сложных расчетных программ, операционных систем или программ управления космическими кораблями, т.е. программ, требующих написания большого объема (сотни тысяч строк) уникального кода.

Не подходят для разработки по методологии RAD приложения, в которых отсутствует ярко выраженная интерфейсная часть, наглядно определяющая логику работы системы (например, приложения реального времени) и приложения, от которых зависит безопасность людей (например, управление самолетом или атомной электростанцией), так как итеративный подход предполагает, что первые несколько версий наверняка не будут полностью работоспособны, что в данном случае исключается.

Оценка размера приложений производится на основе так называемых функциональных элементов (экраны, сообщения, отчеты, файлы и т.п.) Подобная метрика не зависит от языка программирования, на котором ведется разработка.

Размер приложения, которое может быть выполнено по методологии RAD, для хорошо отлаженной среды разработки ИС с максимальным повторным использованием программных компонентов, определяется следующим образом:

|  |  |
| --- | --- |
| **< 1000** функциональных элементов | один человек |
| **1000-4000** функциональных элементов | одна команда разработчиков |
| **> 4000** функциональных элементов | 4000 функциональных элементов на одну команду разработчиков |

В качестве итога перечислим основные **принципы методологии RAD**:

* разработка приложений итерациями;
* необязательность полного завершения работ на каждом из этапов жизненного цикла;
* обязательное вовлечение пользователей в процесс разработки ИС;
* необходимое применение CASE-средств, обеспечивающих целостность проекта;
* применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
* необходимое использование генераторов кода;
* использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности конечного пользователя;
* тестирование и развитие проекта, осуществляемые одновременно с разработкой;
* ведение разработки немногочисленной хорошо управляемой командой профессионалов;
* грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

# 2. Создание модели данных средствами BPwin 7.0

Для успешного построения как бизнес-процессов, так и для реинжениринга, необходимо тесное взаимодействие между специалистами в сфере информационных технологий и экспертами в предметной области бизнеса, которое возможно только при использовании единого понятийного языка для обеих сторон, коим является язык контекстных диаграмм, который позволяет описать текущую структуру бизнес процессов, а также желаемые изменения. Таким мощным инструментом для создания моделей, позволяющих анализировать, документировать и планировать изменения сложных бизнес-процессов является BPwin (программный продукт компании Computer Associates).

BPwin предлагает средство для сбора всей необходимой информации о работе предприятия и графического изображения этой информации в виде целостной и непротиворечивой модели. BPwin поддерживает три методологии: IDEF0, DFD и IDEF3, позволяющие анализировать ваш бизнес с трех ключевых точек зрения [19]:

* с точки зрения функциональности системы. В рамках методологии IDEF0 (Integration Definitionfor Function Modeling) бизнес-процесс представляется в виде набора элементов-работ, которые взаимодействуют между собой, а также показывается информационные, людские и производственные ресурсы, потребляемые каждой работой;
* с точки зрения потоков информации (документооборота) в системе. Диаграммы DFD (Data Flow Diagramming) могут дополнить то, что уже отражено в модели IDEF3, поскольку они описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией между бизнес-функциями внутри системы. В тоже время диаграммы DFD оставляют без внимания взаимодействие между бизнес-функциями;
* с точки зрения последовательности выполняемых работ. И еще более точную картину можно получить, дополнив модель диаграммами IDEF3. Этот метод привлекает внимание к очередности выполнения событий. В IDEF3 включены элементы логики, что позволяет моделировать и анализировать альтернативные сценарии развития бизнес-процесса.

## 2.1. Принципы построения модели (IDEF0)

Основу методологии IDEF0 составляет графический язык описания бизнес-процессов, отображающих отношение «сущность-связь». Модель IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе [19].

Модель может содержать четыре типа диаграмм:

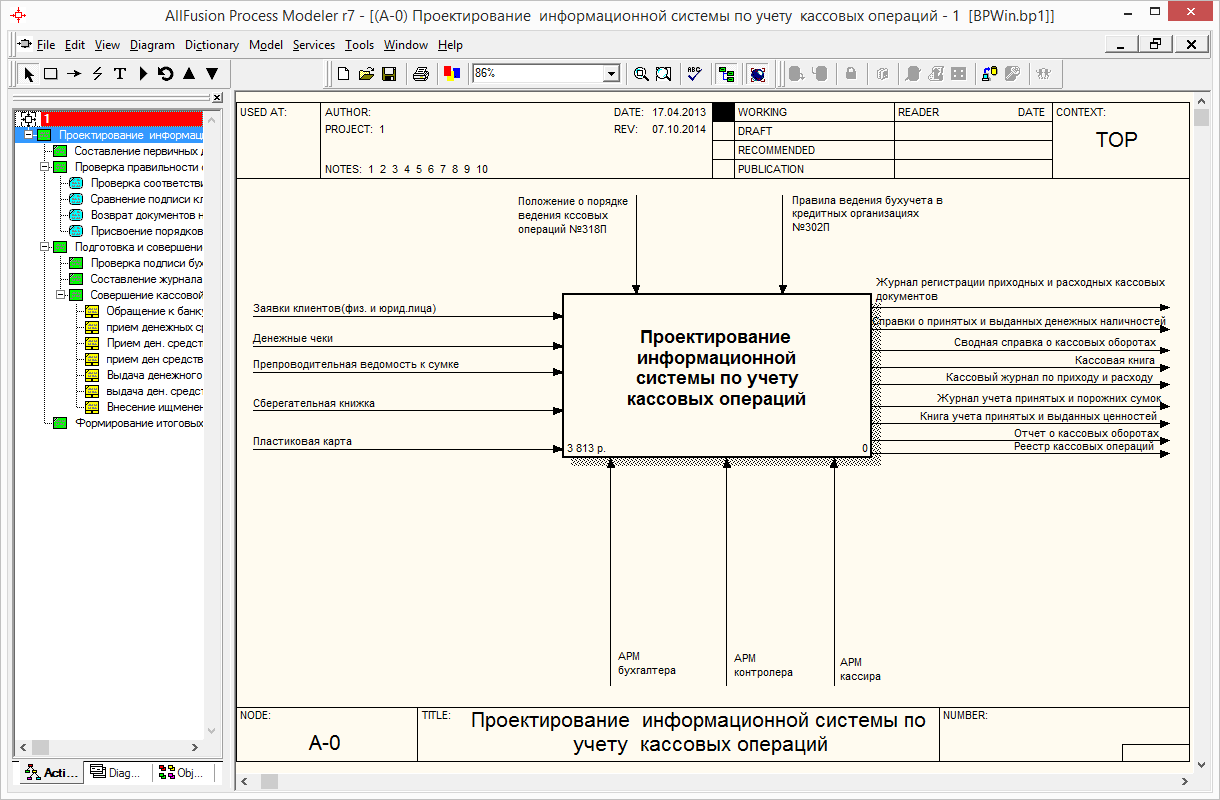
* контекстную;
* декомпозиции;
* дерева узлов;
* только для экспозиции (FEO).

В данном учебном пособии будут рассмотрены первые два типа диаграмм.

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой.

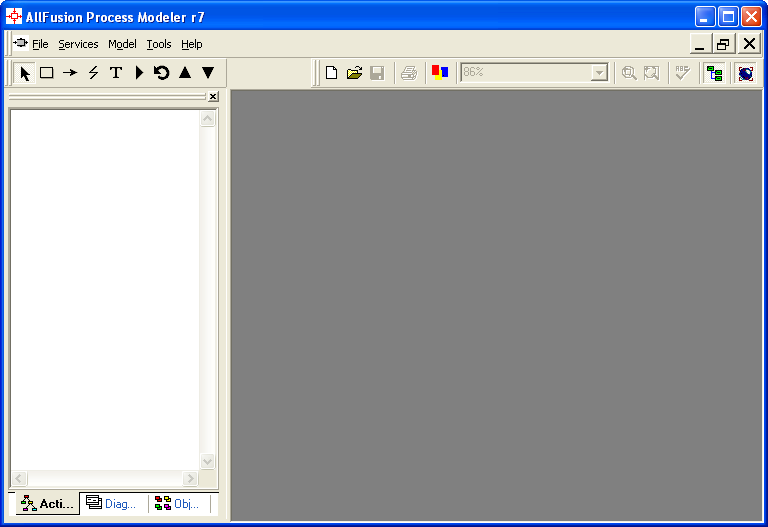
При функционально-ориентированном подходе проектирования информационных систем с использование CASE-средств используется такое средство как BPwin.

Рассмотрим процесс функциональной декомпозиции на примере работы по созданию информационной системы «Автоматизация учета кассовых операций» в программе BPwin 7.0 (см. рис. 2.1). После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции.



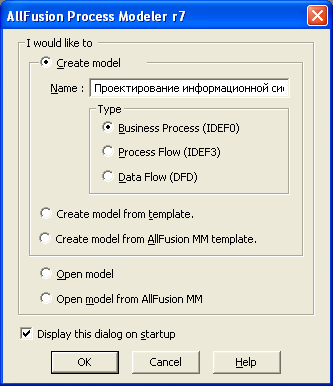
**Рис. 2.1.** Контекстная диаграмма информационной системы «Учет кассовых операций».

Сразу после запуска программы BPwin 7.0 появляется рабочее поле для создания и редактирования моделей (см. рис. 2.2).

****

**Рис. 2.2.** Главное окно программы BPwin 7.0.

Для создания новой модели бизнес-процесса (например, IDEF0) следует выбрать в меню **File** режим **New**. В открывшемся окне ***All Fusion Process Modeler r7*** следует ввести название модели, выбрать тип модели *Business Process (IDEF0)* и нажать кнопку ОК (см. рис. 2.3).



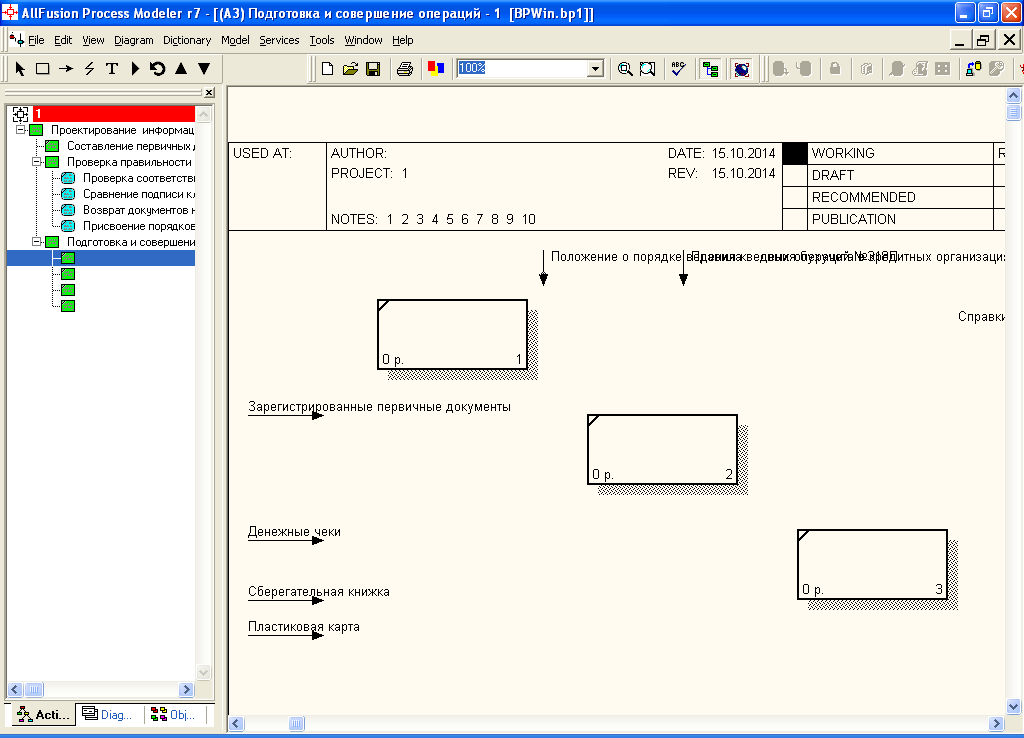
**Рис. 2.3.** Окно для выбора создаваемой модели.

После этого откроется окно для заполнения свойств новой модели. На вкладках этого окна можно настроить вид нумерации диаграмм, размеры отступов, отображаемые элементы на экране, настройки форматирования объектов, настройки денежного формата, настройки содержимого заголовка и подвала.

Начальным этапом разработки поставленной задачи по проектированию ИС является создание контекстной диаграммы, в которой определяется:

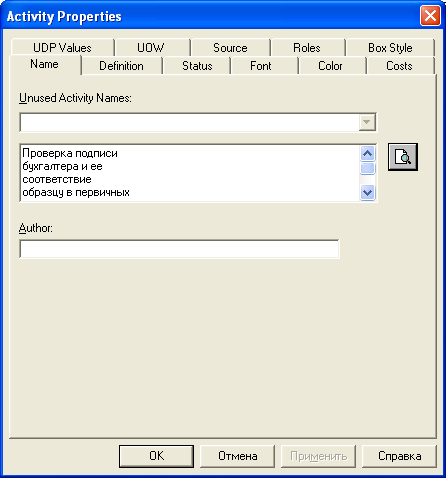
* название проектируемой ИС;
* входная и выходная информация
* механизм реализации ИС;
* нормативная документация.

Все это относится к блоку, обозначающему работу, вид которой представлен на рис. 2.4. Работы обозначают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты.



**Рис. 2.4.** Диаграмма, обозначающая работу (IDEF0).

Для добавления работы требуется нажать кнопку , а затем – щелкнуть «мышью» в свободном месте диаграммы. После этого следует щелкнуть два раза «мышью» на появившемся элементе, и в окне ***Activity Properties*** (см.  
рис. 2.5) на вкладке **Name** ввести название работы.



**Рис. 2.5.** Окно для ввода характеристик работы (IDEF0).

Работа может иметь на входе:

* (сверху) нормативная документация, документы действующего законодательства;
* (слева) входная информация;
* (снизу) механизм реализации ИС, средства для выполнения работы.

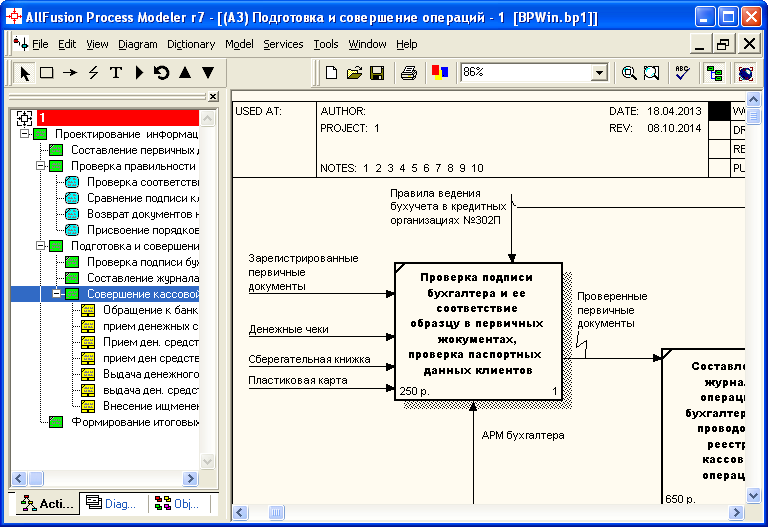
Работа может иметь на выходе:

* (справа) выходная информация.

В таком случае работа может принять вид, представленный на рис. 2.6:

На одном листе диаграммы аналогично строятся все необходимые работы.

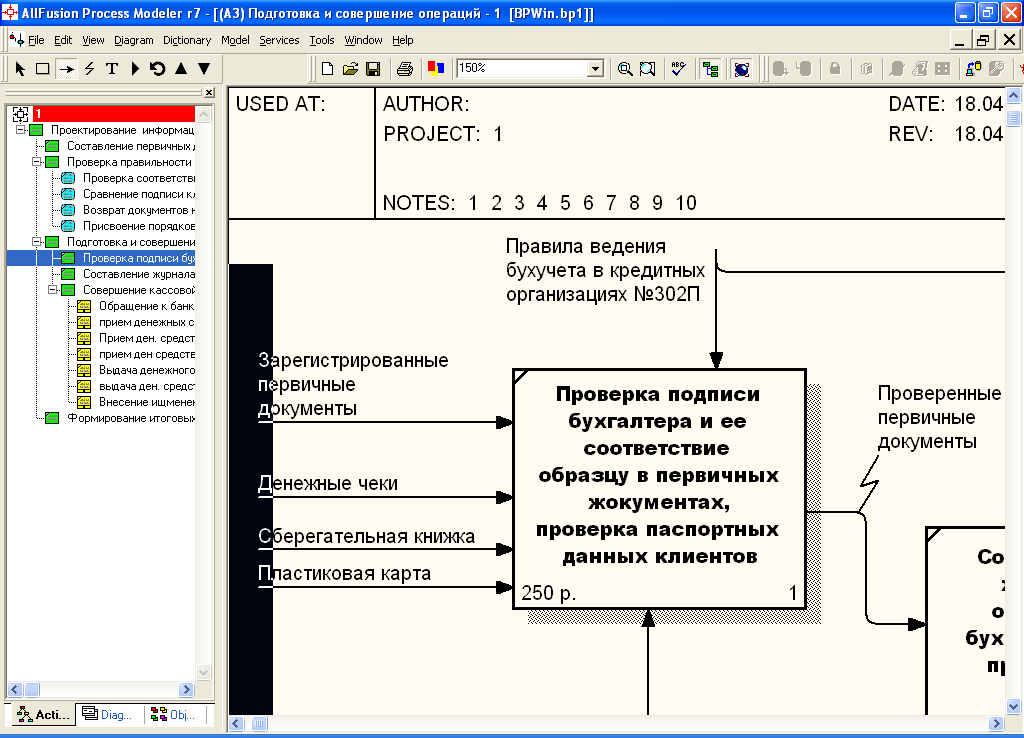
После заведения всех работ на текущем листе выполняют создание стрелок, обозначающих как связь между работами, так и входную и выходную информацию, а также воздействие из внешней среды.



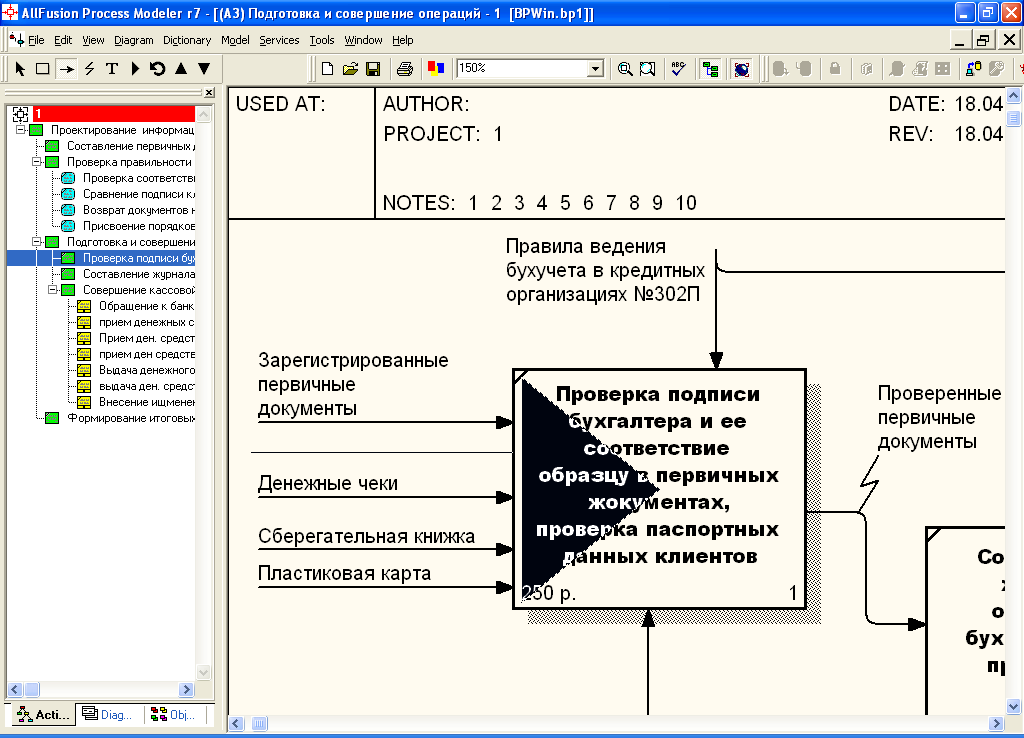
**Рис. 2.6.** Диаграмма, обозначающая работу, с учетом входов и выходов.

Для использования стрелок следует нажать кнопку . Затем следует:

* для указания взаимодействия системы из окружающего мира подвести «мышь» к левой (верхней или нижней) стороне экрана, пока не появится начальная темная полоска (см. рис. 2.7), щелкнуть левой кнопкой «мыши» один раз, затем подвести «мышь» к левой (верхней или нижней) части работы, пока не появится конечный темный треугольник и щелкнуть левой кнопкой «мыши» (см. рис. 2.8);

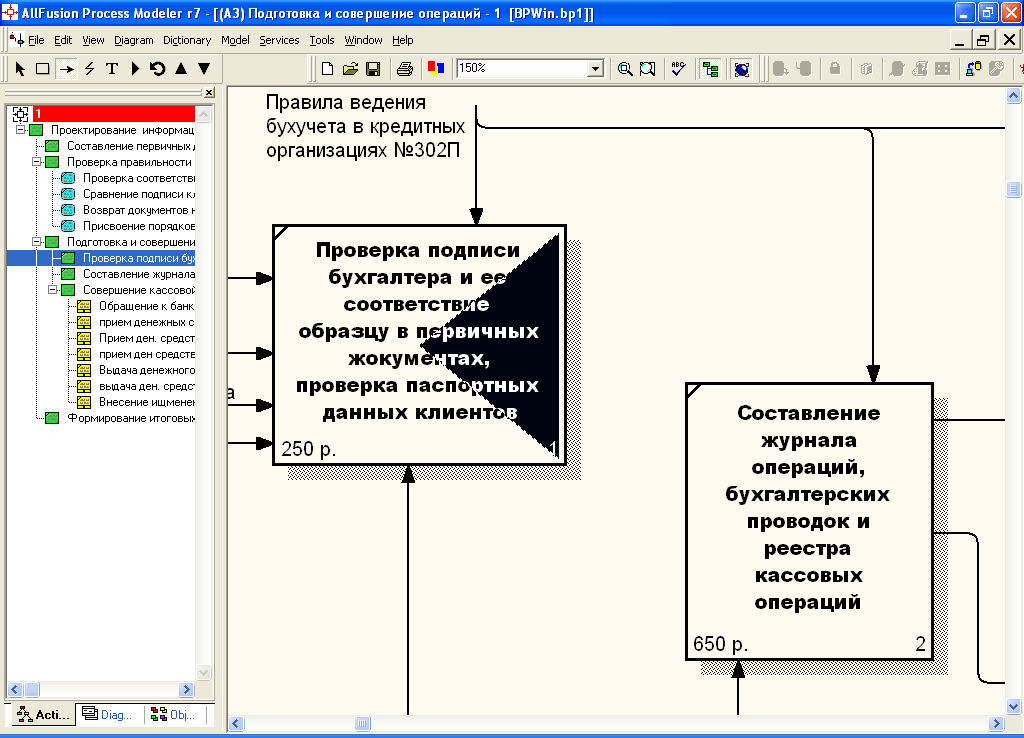


**Рис. 2.7.** Влияние на работу из внешней среды (начало).

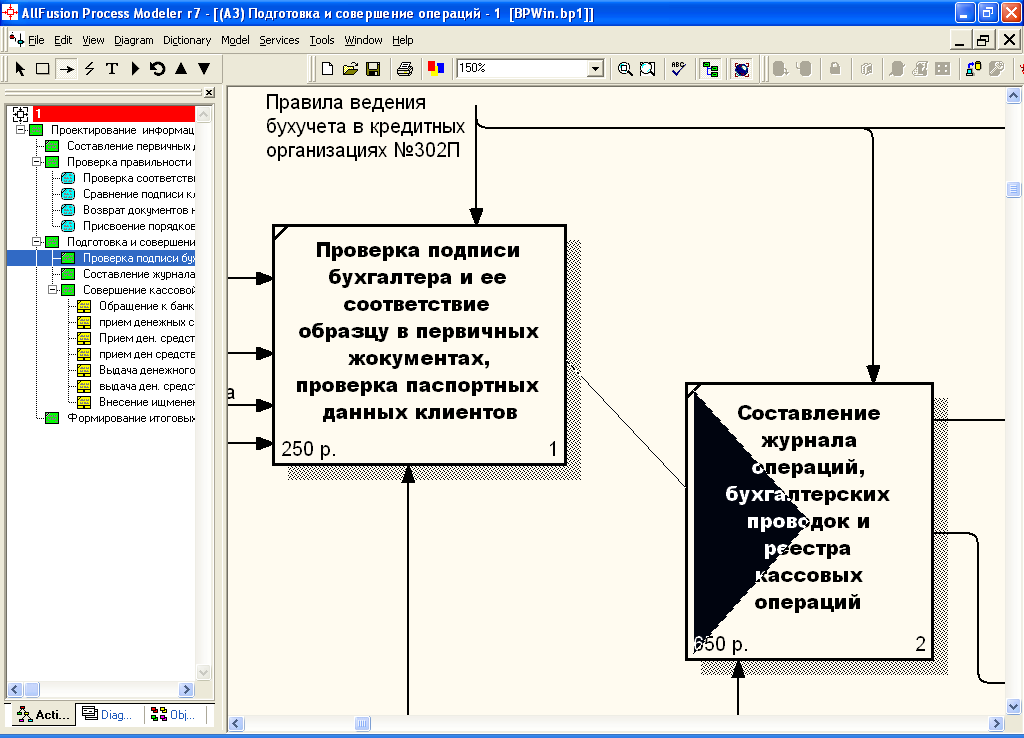
****

**Рис. 2.8.** Влияние на работу из внешней среды (конец).

* для связывания работ подвести «мышь» к правой части работы, пока не появится начальный темный треугольник, щелкнуть левой кнопкой «мыши» (см. рис. 2.9); затем подвести «мышь» к левой (верхней или нижней) части работы, пока не появится конечный темный треугольник и щелкнуть левой кнопкой «мыши» (см. рис.2.10);

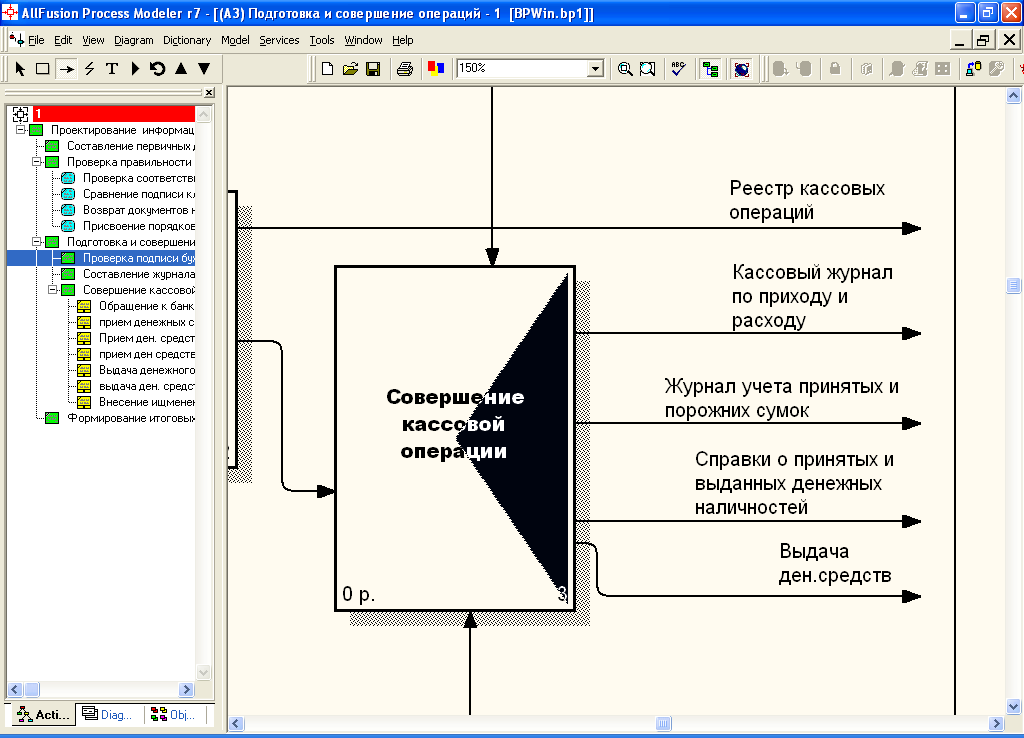


**Рис. 2.9.** Связывание работ (начало).

****

**Рис. 2.10.** Связывание работ (конец).

* для указания взаимодействия системы с внешней средой подвести «мышь» к правой части работы, пока не появится начальный темный треугольник (см. рис. 2.11), щелкнуть левой кнопкой «мыши», затем подвести «мышь» к правой части экрана, пока не появится конечная темная полоска, и щелкнуть по ней левой кнопкой «мыши» один раз (см. рис.2.12).



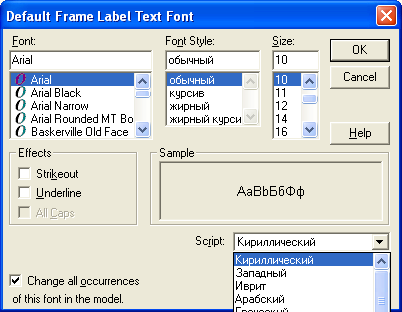
**Рис. 2.11.** Влияние работы на внешнюю среду (начало).

****

**Рис. 2.12.** Влияние работы на внешнюю среду (конец).

Для перехода в режим написания (или редактирования) текста (на стрелках, в работах) следует нажать кнопку . Затем необходимо два раза щелкнуть левой кнопкой «мыши» по нужному объекту и ввести в него текст.

Для настройки шрифта текста, используемого в названиях работ, а также текста верхней части шаблона листа, следует выбрать в меню **Model** режим **Default Fonts**, а затем подрежим **Frame System Text**. В открывшемся окне ***Default Frame Label Text Font***(см. рис. 2.13) в блоке Script необходимо выбрать *Кириллический*, а также установить флажок Change all occurrences. После этого следует нажать кнопку ОК.



**Рис. 2.13.** Настройка шрифта.

Для настройки шрифта текста, используемого в нижней части шаблона листа, следует выбрать в контекстном меню выбрать режим **Parent Diagram**, а затем подрежим **Title Text Font**. Дальнейшие действия описаны ранее (см. рис. 2.13).

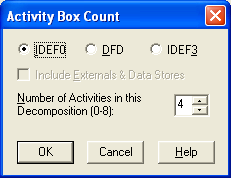
Для ввода ***комментариев*** по требуемой «стрелке» следует после создания текста-названия стрелки, сдвинуть этот текст в удобное свободное место диаграммы. Затем нажать кнопку  и щелкнуть на этой стрелке левой кнопкой «мыши» один раз.

Для настройки шрифта текста названий «стрелок» декомпозированных работ следует выбрать в меню **Model** режим **Default Fonts**, а затем подрежим **Decomposition Arrow**. Дальнейшие действия описаны ранее (см. рис. 2.13).

Альтернативным способом настройки шрифта любого элемента диаграммы является нажатие на таком элементе правой кнопке «мыши» и выбор вкладки **Font**, в которой следует выбрать шрифт кириллицы.

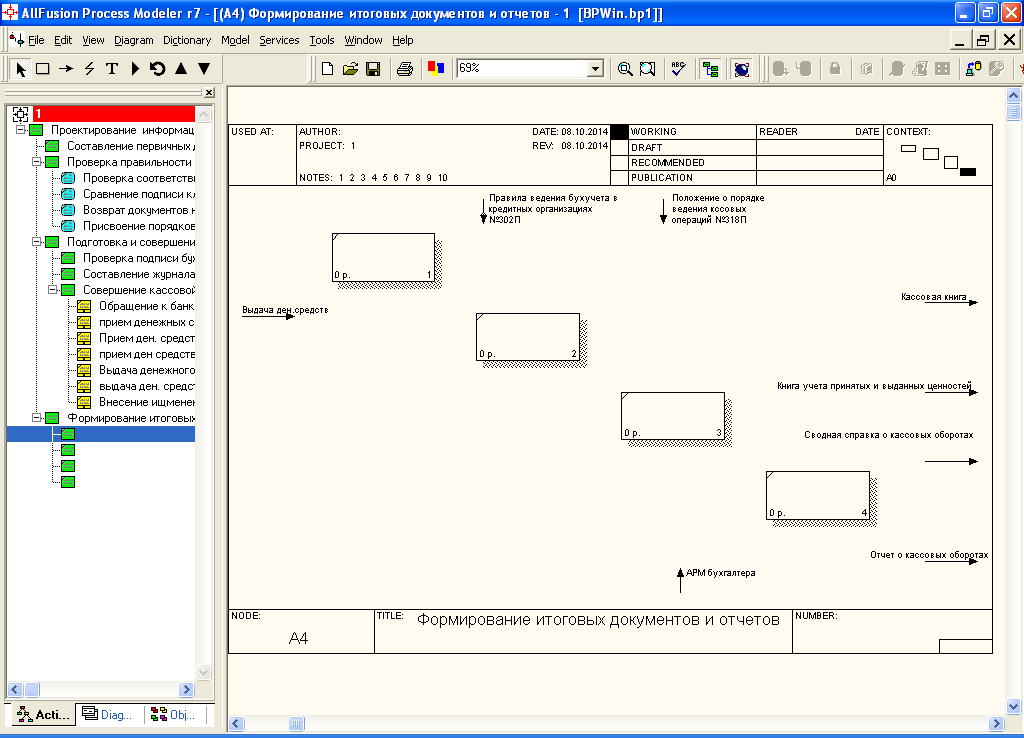
Контекстная диаграмма должна быть подвергнута декомпозиции, т.е. разбиению на более мелкие элементы, необходимого для более детальной и углубленной проработки или рассмотрения бизнес-процессов. Разбиение может быть продолжено до тех пор, пока это имеет смысл и является необходимым для пояснения технологии проектирования ИС.

Для добавления диаграммы декомпозиции, т.е. для дальнейшей детализации ИС следует, отметив работу, подвергающуюся декомпозиции, щелкнуть по кнопке . В появившемся окне следует выбрать тип добавляемой модели (например, IDEF0), установить число блоков в этой модели и нажать кнопку ОК (см. рис.2.14).



**Рис. 2.14.** Окно для выбора диаграммы декомпозиции.

После этого откроется новое поле для редактирования диаграммы декомпозиции (см. рис. 2.15).



**Рис. 2.15.** Новое окно для редактирования диаграммы декомпозиции.

Количество работ в диаграмме декомпозиции должно быть не более 6, чтобы не усложнять восприятие такой диаграммы. Следует отметить, что декомпозицию можно только от типа моделей верхнего уровня к нижнему уровню:

* от IDEF0 к IDEF3 или к DFD;
* от IDEF3 к DFD.

Каждый лист диаграммы декомпозиции состоит из следующих элементов, приведенных в таблицах 2.1 и 2.2 [19].

*Таблица 2.1*

**Поля заголовка каркаса**

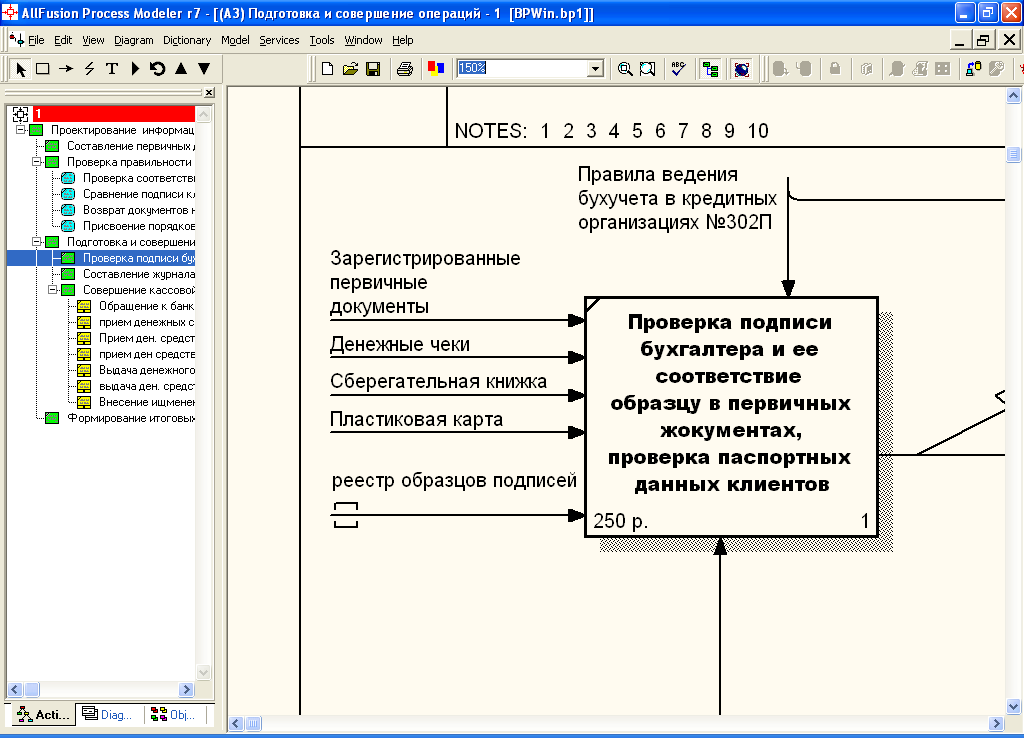
|  |  |
| --- | --- |
| ***Поля*** | ***Смысл*** |
| UsedAt | Используется для указания на родительскую работу в случае, если на текущую диаграмму ссылались посредством стрелки вызова |
| Autor, Date, Rev, Project | Имя создателя диаграммы, дата создания и имя проекта, в рамках которого была создана диаграмма. REV – дата последнего редактирования диаграммы |
| Notes 1 2 3 4 5 6 7 8 9 10 | Используется при проведении сеанса экспертизы. Эксперт должен (на бумажной копии диаграммы) указать число замечаний, вычеркивая цифру из списка каждый раз при внесении нового замечания |
| Status | Статус отображает стадию создания диаграммы, отображая все этапы публикации |
| Working | Новая диаграмма, кардинально обновленная диаграмма или новый автор диаграммы |
| Draft | Диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению |
| Recommended | Диаграмма и все ее сопровождающие документы прошли экспертизу. Новых изменений не ожидается |
| Publication | Диаграмма готова к окончательной печати и публикации |
| Reader | Имя читателя (эксперта) |
| Date | Дата прочтения (экспертизы) |
| Context | Схема расположения работ в диаграмме верхнего уровня. Работа, являющаяся родительской, показана темным прямоугольником, остальные – светлым. На контекстной диаграмме (А-0) показывается надпись TOP. В левом нижнем углу показывается номер по узлу родительской диаграммы: |

*Таблица 2.2*

**Поля подвала каркаса**

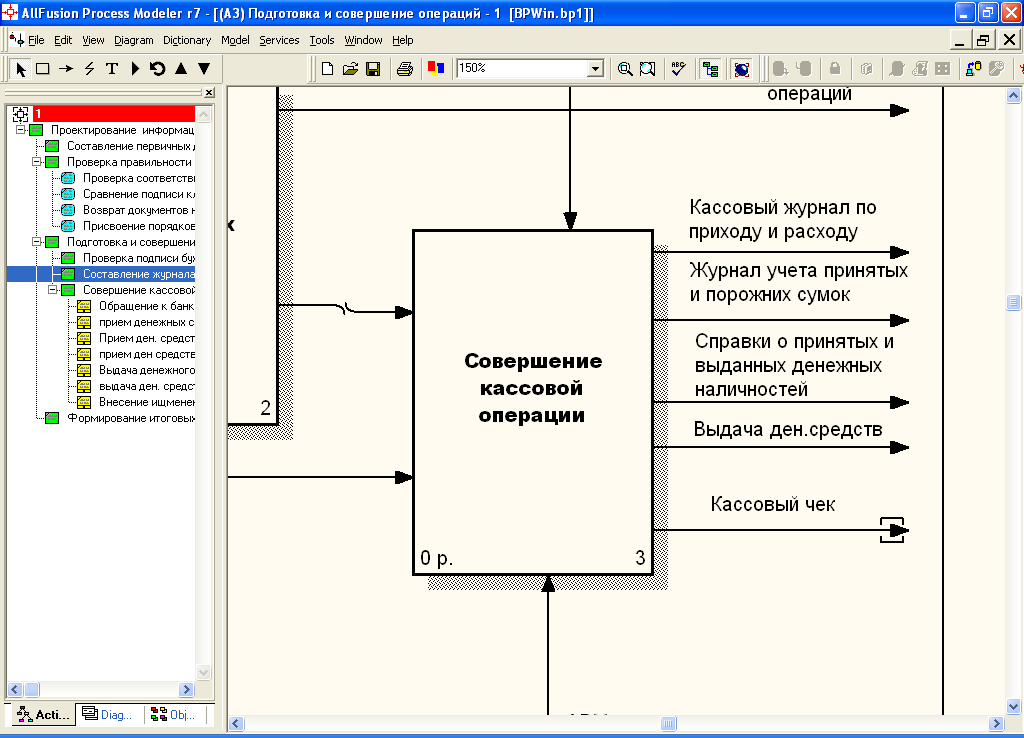
|  |  |
| --- | --- |
| ***Поле*** | ***Смысл*** |
| Node | Номер узла диаграммы (номер родительской работы) |
| Title | Имя диаграммы. По умолчанию – имя родительской работы |
| Number | C-Number, уникальный номер версии диаграммы |
| Page | Номер страницы, может использоваться как номер страницы при формировании папки |

Если одна из стрелок диаграммы отсутствует на родительской диаграмме и не связана с другими стрелками той же диаграммы, точка входа или выхода этой «стрелки» на диаграмме обозначается туннелем (см. рис. 2.16).



**Рис. 2.16.** Туннеллирование входа стрелки.

Кроме того, туннели используются для отражения ситуации, когда стрелка, присутствующая на родительской диаграмме, отсутствует в диаграмме декомпозиции соответствующего блока (см. рис. 2.17):



**Рис. 2.17.**Туннеллирование выхода стрелки.

В случае возникновения (незапланированного) туннелирования следует:

* удалить стрелку с туннелем на дочерней, а затем – на родительской диаграмме;
* добавить стрелку в родительской диаграмме на месте удаленной, а в качестве имени выбрать прежнее название;
* соединить вновь созданную стрелку как было до удаления в родительской и дочерней диаграммах.

Для перехода к работе с объектами диаграммы следует нажать кнопку .

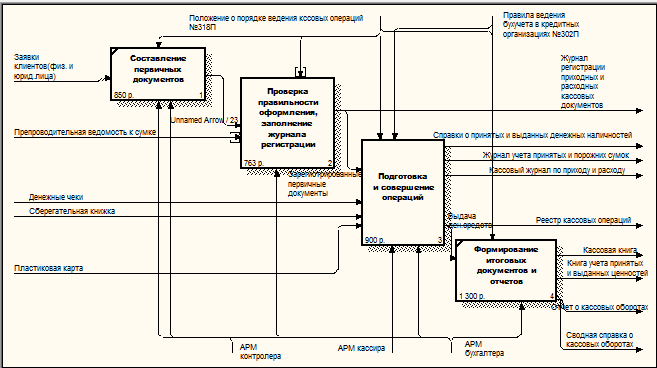
Для удаления блока диаграммы следует отметить этот блок, щелкнув по нему левой кнопкой «мыши», затем нажать клавишу Delete. В открывшемся окне следует нажать кнопку Да.

Для удаления неиспользованных стрелок в модели следует в меню **Model** выбрать **Arrow Editor**, а затем, отметив стрелку без названия (например, *Unnamed Arrow*), нажать кнопку Delete. В открывшемся окне следует нажать кнопку Да.

При построении диаграммы декомпозиции между работами можно устанавливать следующие связи:

* входящая информация может быть входной информацией для других работ;
* выходная информация может выступать нормативной документацией или быть промежуточной и выступать средством для выполнения других работ.

Пример диаграммы декомпозиции IDEF0 представлен на рис. 2.18.

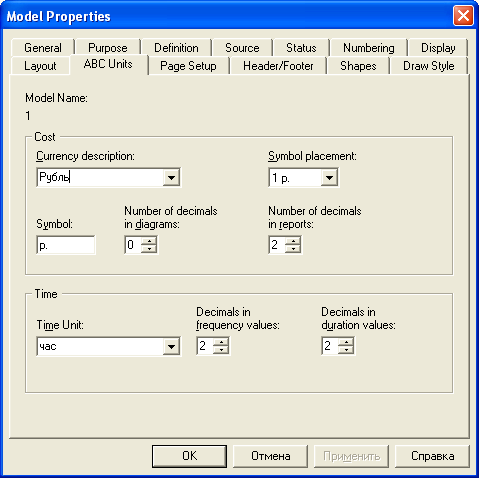
****

**Рис. 2.18.** Пример диаграммы декомпозиции IDEF0.

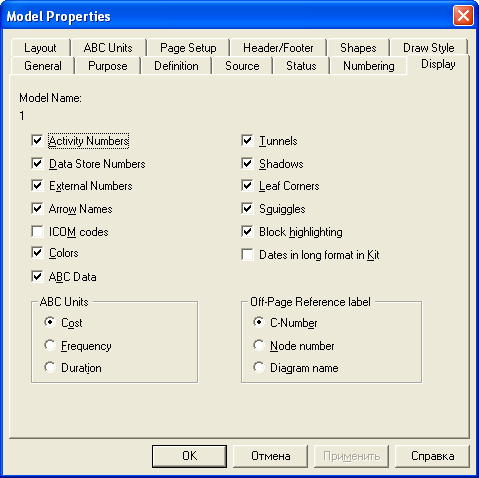
### 2.1.1. Стоимостный анализ (ABC)

Для оценки модели BPwin предоставляет разработчику такой инструмент как стоимостный анализ, основанный на работах (Activity Based Costing, ABC). АВС является широко распространенной методикой, используемой международными корпорациями и государственными организациями (в т.ч. Департаментом обороны США) для идентификации истинных движителей затрат в организации. Стоимостный анализ представляет собой соглашение об учете, используемое для сбора затрат, связанных с работами, с целью определить общую стоимость процесса. АВС применяется для того, чтобы понять происхождение выходных затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия. АВС может проводиться только тогда, когда модель работы последовательная (следует синтаксическим правилам IDEF0), корректная (отражает бизнес), полная (охватывает всю рассматриваемую область) и стабильная (проходит цикл экспертизы без изменений), другими словами, создание модели работы закончено [19].

При проведении стоимостного анализа сначала задаются единицы измерения времени и денег. Для этого в меню **Model** следует выбрать режим **Model Properties**. В открывшемся диалоговом окне ***Model Properties*** следует на вкладке ABC Units (см. рис. 2.19) установить денежную единицу (например, рубль) и временную единицу (например, час). Для настройки отображения на экране стоимости работ следует на вкладке Display для блока АВС units установить переключатель Cost (см. рис. 2.20).



**Рис. 2.19.** Настройка единиц измерения валюты и времени.



**Рис. 2.20.** Настройка отображения стоимости работ.

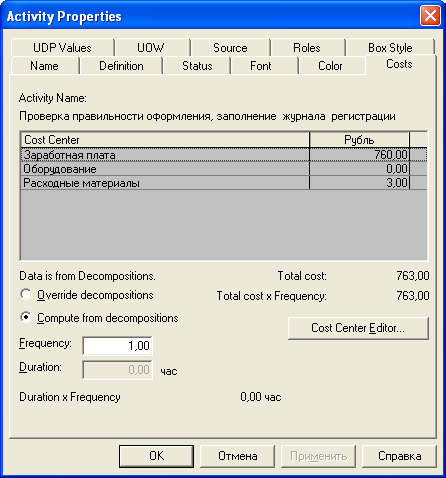
После этого следует ввести названия центров затрат. Для этого в меню **Dictonary** выбирают режим **Cost Center**. Пример названий центров затрат представлен в таблице 2.3.

*Таблица 2.3.*

**Центры затрат (АВС)**

|  |  |
| --- | --- |
| ***Центры затрат*** | ***Определение*** |
| Заработная плата | Расходы по заработной плате в день |
| Оборудование | Затраты на амортизацию оборудования в день |
| Расходные материалы | Стоимость расходных материалов (канцтовары) |

Каждой работе в разрезе центров затрат необходимо задать стоимость, для чего следует щелкнуть два раза «мышью» на такой работе. В открывшемся окне ***Activity Properties*** на вкладке Costs устанавливается переключатель Override decompositions (см. рис. 2.21), устанавливается частота (Frequency) и длительность (Duration) и задается стоимость по центрам затрат, автоматически суммирующимся в стоимость работы. Для редактирования названий центров затрат используется кнопка Cost Center Editor. Для подтверждения введенных данных нажать кнопку ОК.



**Рис. 2.21.** Диалоговое окно для задания стоимости работ.

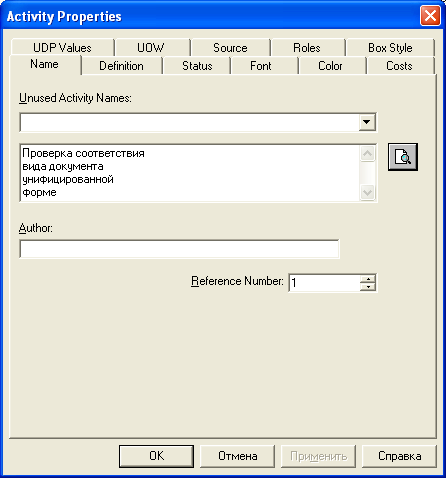
## 2.2. Диаграммы потов данных (DFD)

Диаграммы потоков данных (Data Flow Diagramming, DFD) используются для описания документооборота и обработки информации. Подобно IDEF0, DFD представляет модельную систему как взаимосвязанный набор действий, которые обрабатывают данные в «хранилища данных» как внутри, так и вне границ моделируемой системы. Их можно использовать как дополнение к модели IDEF0 для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации. DFD описывает [19]:

* функции обработки информации (работы);
* документы (стрелки), объекты, сотрудников или отделы, которые участвуют в обработке информации;
* внешние ссылки, которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы;
* таблицы для хранения документов (хранилище данных).

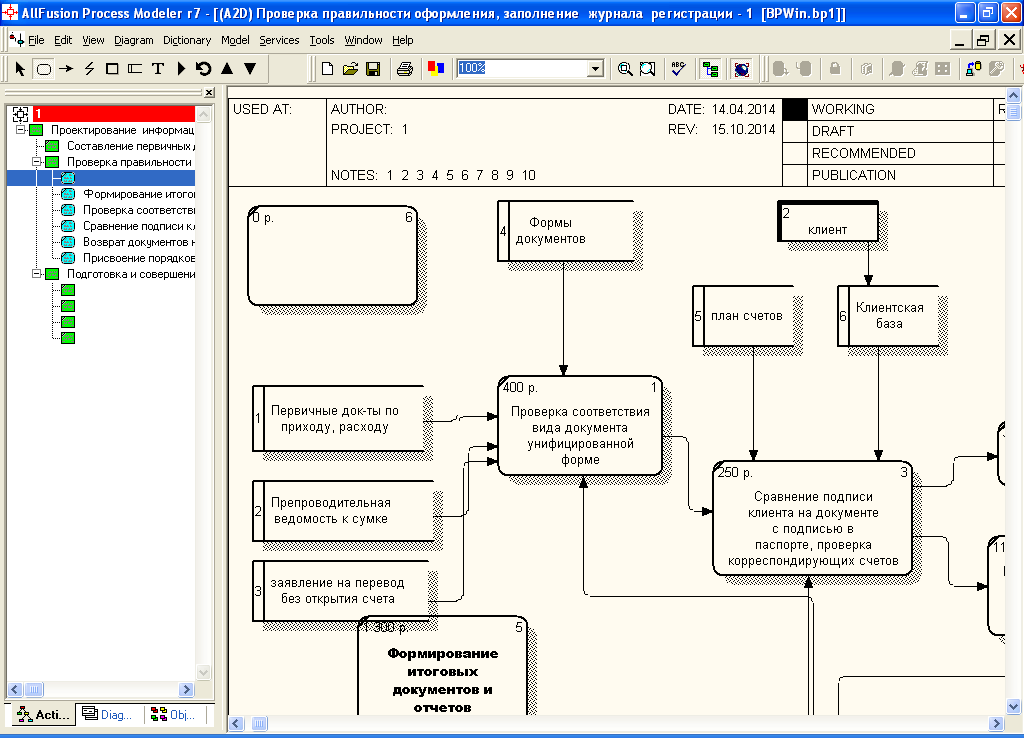
В BPwin для построения диаграмм потоков данных используется нотация Гейна-Сарсона.

Для добавления в диаграмму новой работы следует нажать кнопку , а затем – щелкнуть «мышью» в свободном месте диаграммы. После этого следует щелкнуть два раза «мышью» на появившемся элементе и в окне ***Activity Properties*** (см. рис. 2.22) на вкладке Name ввести название работы.



**Рис. 2.22.** Окно для заполнения свойств работы (DFD).

Диаграмма «Работа» обозначается в виде следующего блока (см. рис. 2.23).



**Рис. 2.23.** Диаграмма, обозначающая работу (DFD).

В DFD работы представляют собой функции системы, преобразующие входы в выходы. Работы не поддерживают управление и механизмы, как IDEF0.

Для добавления в диаграмму внешней ссылки следует нажать кнопку , а затем – щелкнуть «мышью» в свободном месте диаграммы. Внешние сущности изображают источник или приемник данных из вне модели и обычно располагаются по краям диаграммы.

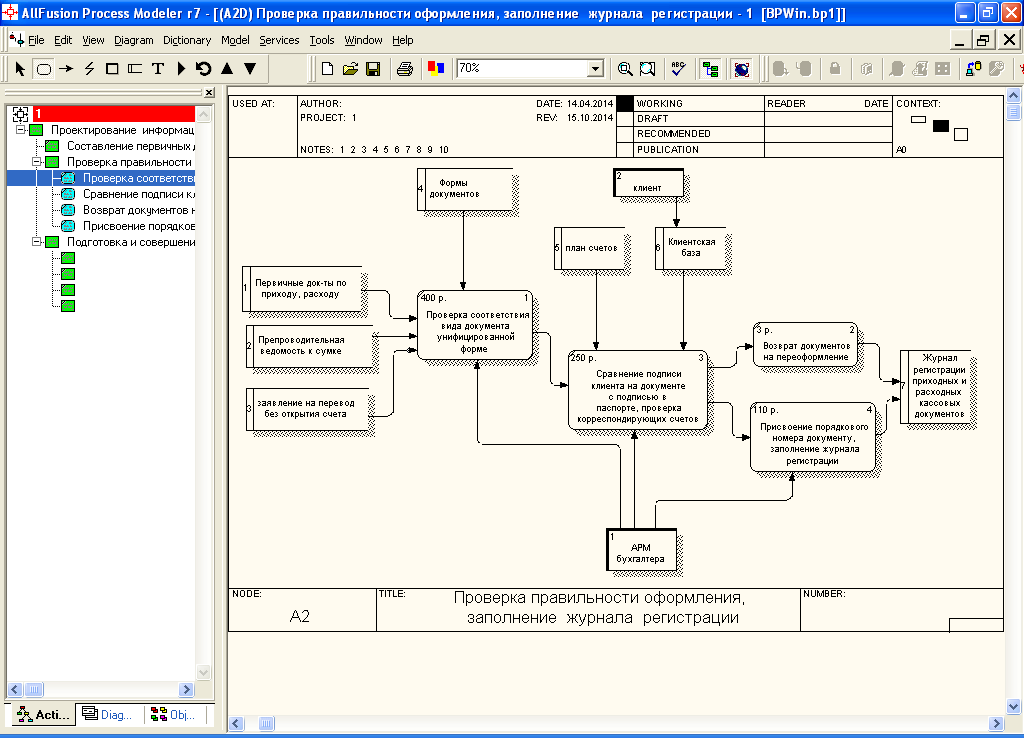
Для добавления в диаграмму хранилища данных следует нажать кнопку , а затем – щелкнуть «мышью» в свободном месте диаграммы. Хранилища данных являются механизмом, который позволяет описать данные, сохраняемые в памяти прежде, чем использовать их в работах.

Стрелки (потоки данных) описывают движение объектов из одной части системы в другую. В DFD также применяются двунаправленные стрелки для описания диалогов типа «команда-ответ» между работами, между работой и внешней сущностью, между внешними сущностями.

*Следует избегать дублирования блоков внешних сущностей или хранилищ данных на одном листе диаграммы.*

Элементы хранилищ данных и внешние сущности нумеруются автоматически на каждой диаграмме декомпозиции DFD.

Пример диаграммы декомпозиции DFD представлен на рис. 2.24.



**Рис. 2.24.** Диаграмма декомпозиции DFD.

## 2.3. Метод описания процессов (IDEF3)

Для описания логики взаимодействия информационных потоков более подходит IDEF3, называемая также workflow diagramming – методологией моделирования, предназначенная для обеспечения структурированного подхода к описанию бизнес-процесса как упорядоченной последовательности событий одновременно с описанием любых участвующих в бизнес-процессе объектов и относящихся к ним правил [19].

**IDEF3** – стандарт моделирования бизнес-процессов, поддерживающий графическое описание непосредственного механизма функционирования системы или организации. IDEF3 содержит правила разработки двух видов сетевых диаграмм:

* диаграмм потоков для бизнес-процессов;
* диаграмм изменения состояния объекта.

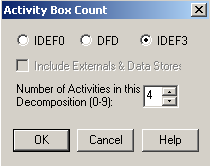
Диаграммы workflow[[3]](#footnote-3) могут быть использованы в моделировании бизнес – процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например последовательность обработки заказа или события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

IDEF3 – это метод, предоставляющий аналитикам средство для описания ситуации, в которой процессы выполняются как в определенной последовательности, так и описать объекты, участвующие совместно в одном процессе.

Техника описания набора данных IDEF3 является частью структурного анализа. В отличие от некоторых методик описаний процессов, IDEF3 не   
ограничивает аналитика чрезмерно жесткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей.

IDEF3 может быть также использован как метод создания процессов. IDEF3 дополняет IDEF0 и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

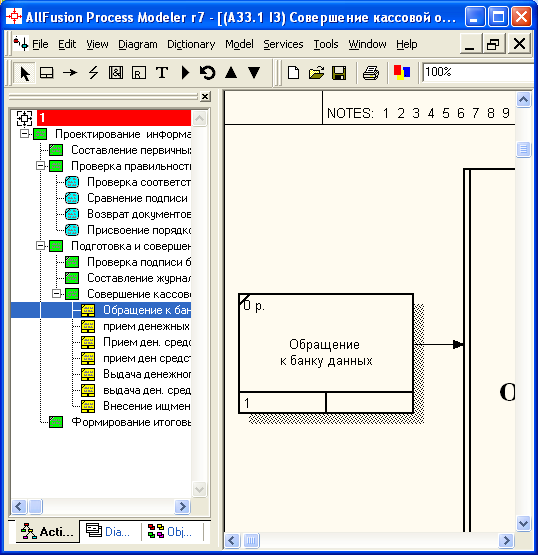
Для добавления диаграммы декомпозиции, т.е. для дальнейшей детализации ИС следует, отметив работу, подвергающуюся декомпозиции, щелкнуть по кнопке . В появившемся окне следует выбрать тип добавляемой модели (например, IDEF3), установить число блоков в этой модели и нажать кнопку ОК (см. рис. 2.25).



**Рис. 2.25.** Окно для выбора диаграммы декомпозиции (IDEF3).

Главной организационной единицей модели IDEF3 является диаграмма. Другим важным компонентом модели является единица работы (или работа). Диаграммы IDEF3 отображают действие в виде прямоугольника (см. рис. 2.26). При подборе подходящего наименования для обозначения действий бизнес-процесса применяются стандартные рекомендации по предпочтительному использованию глаголов и отглагольных существительных. Каждому из действий присваивается уникальный идентификационный номер.

Для добавления работы следует нажать кнопку , а затем – щелкнуть «мышью» на свободном месте диаграммы.



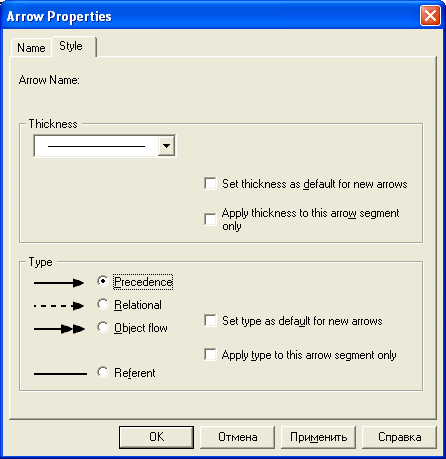
**Рис. 2.26.** Изображение действий (работ) в диаграмме IDEF3.

Работа в IDEF3 требует более подробного описания, чем работа в IDEF0. Каждая работа должна иметь ассоциативный документ, который включает текстовое описание компонентов работы: объектов (*Objects*) и фактов (*Facts*), связанных с работой, ограничений (*Constraints*), накладываемых на работу и дополнительное описание работы (*Description*).

Связи выделяют существенные взаимоотношения между работами. Все связи в IDEF3 однонаправлены (обычно слева направо). Существуют следующие типы связей:

* связь типа «временное предшествование» (исходное действие должно полностью завершиться, прежде чем начнется выполнение конечного  
  действия) – **сплошная линия**;
* связь типа «объектный поток» (исходное действие должно завершиться, прежде чем конечное действие может начать выполняться) – **пунктирная линия**;
* связь типа «нечеткое отношение» (выделение отношений между действиями, которое невозможно описать с использованием предшествующих или объектных связей, например, для отображения взаимоотношений между параллельно выполняющимися действиями) – **стрелка с двумя наконечниками**.

Для изменения типа стрелки, связывающей работы, необходимо щелкнуть по такой стрелке два раза «мышью» - откроется диалоговое окно ***Arrow Properties***, в котором следует перейти на вкладку *Style* (см. рис. 2.27) и установить переключатель (Precedence, Relational, Objectflow) в блоке *Type* в соответствии с требуемым типом связи.



**Рис. 2.27.** Окно для добавления объекта ссылки.

Для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут и должны быть завершены перед началом следующей работы используются перекрестки. Различают перекрестки для слияния (*Fan–in Junction*) и разветвления (*Fan–out Junction*) стрелок. Смысл каждого типа перекрестка приведен в таблице 2.4 [19].

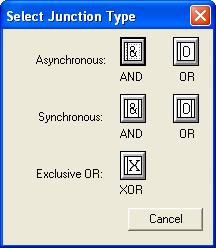
Таблица 2.4

**Типы перекрестков**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Обозна-чение*** | ***Наименование*** | ***Смысл в случае слияния стрелок (Fan-in Junction)*** | ***Смысл в случае разветвления стрелок (Fan-out Junction)*** |
| **1** | **2** | **3** | **4** |
| **&** | Асинхронное «И» (Asynchronous AND) | Все предшествующие процессы должны быть завершены | Все следующие процессы должны быть запущены |
| **&** | Синхронное «И» (Synchronous AND) | Все предшествующие процессы завершены одновременно | Все следующие процессы запускаются одновременно |
| **Ο** | Асинхронное «ИЛИ» (Asynchronous OR) | Один или несколько предшествующих процессов должны быть завершены | Один или несколько следующих процессов должны быть запущены |
| **О** | Синхронное «ИЛИ» (Synchronous OR) | Один или несколько предшествующих процессов завершены одновременно | Один или несколько следующих процессов запускаются одновременно |
| **Х** | Исключающее «ИЛИ» XOR (Exclusive OR) | Только один предшествующий процесс завершен | Только один следующий процесс запускается |

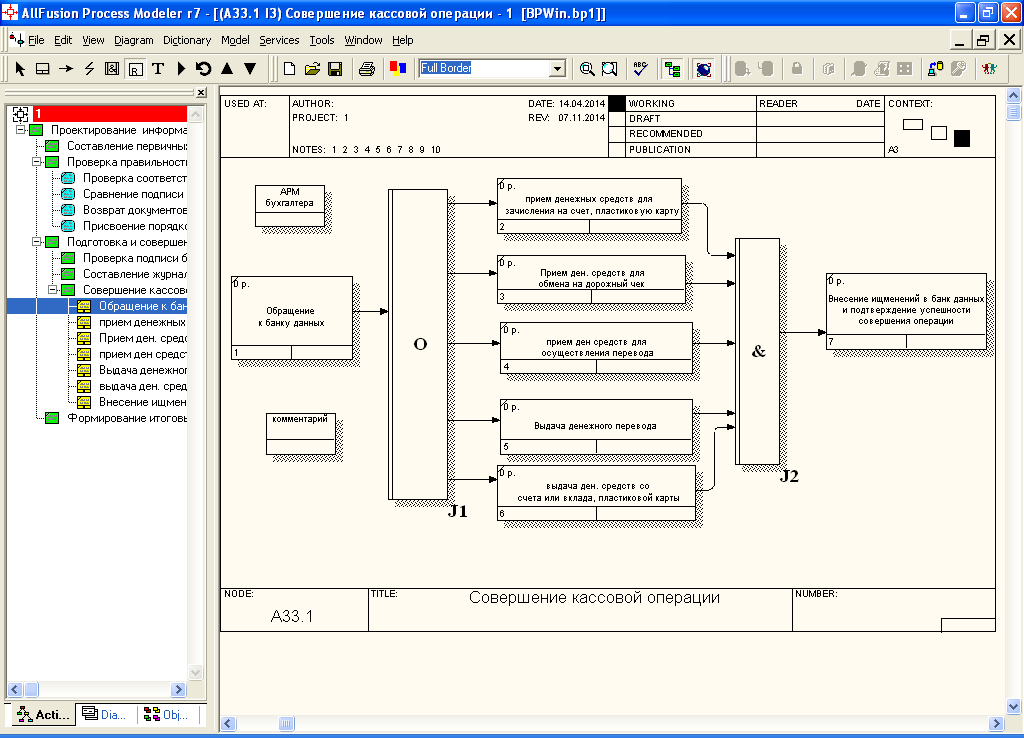
Перекресток не может использоваться одновременно для слияния и для разветвления.

Для добавления перекрестка следует нажать кнопку , а затем щелкнуть «мышью» на свободном месте диаграммы. В открывшемся окне ***Select Junction Type*** (см. рис. 2.28) следует выбрать тип перекрестка. После этого стрелками следует соединить необходимые работы.



**Рис. 2.28.** Окно для выбора типа перекрестка.

Пример диаграммы декомпозиции IDEF3 представлен на рис. 2.29.

****

**Рис. 2.29.** Диаграмма декомпозиции IDEF3.

## 2.4. Создание отчетов

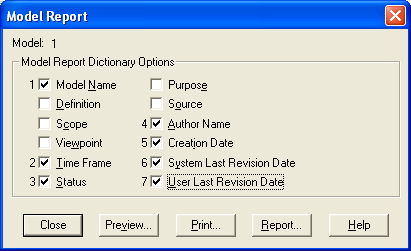
Существует три способа создания отчетов в BPwin7.0:

1. с помощью встроенных шаблонов;
2. с помощью Report Template Builder;
3. с помощью RPTwin.

### 2.4.1. Отчеты на основе встроенных шаблонов

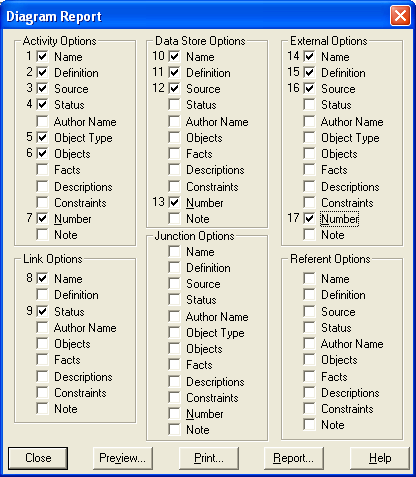
Отчеты на основе встроенных шаблонов можно создать, выбрав в меню **Tools** в режиме **Reports** режим с необходимым типом шаблона. Всего имеется семь типов шаблонов отчетов [19]:

1) **Model Report**. Отчет включает информацию о контексте модели – имя модели, точку зрения, область, цель, имя автора, дату создания и др. Пример выбранных настроек данного отчета представлен на рис. 2.30.



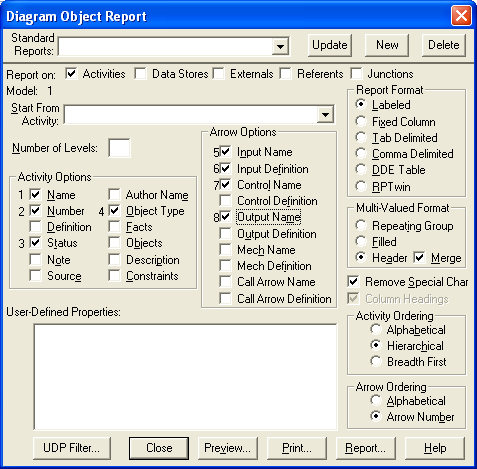
**Рис. 2.30.** Настройки отчета Model Report.

2) **Diagram Report**. Отчет по контекстной диаграмме, включает список объектов: работ, стрелок, хранилищ данных, внешних ссылок и др. Пример выбранных настроек данного отчета представлен на рис. 2.31.



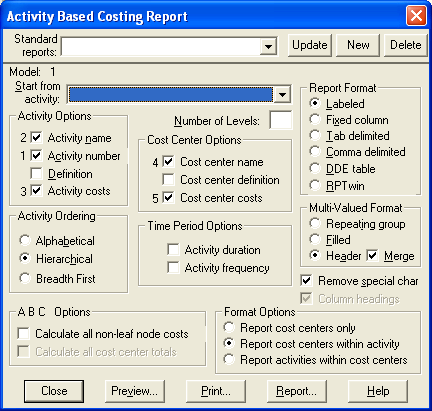
**Рис. 2.31.** Настройки отчета Diagram Report.

3) **Diagram Object Report**. Наиболее полный отчет по модели, включает полный список объектов модели: работ, стрелок с указанием их типа и др. – и свойства, определяемые пользователем. Пример выбранных настроек данного отчета представлен на рис. 2.32.



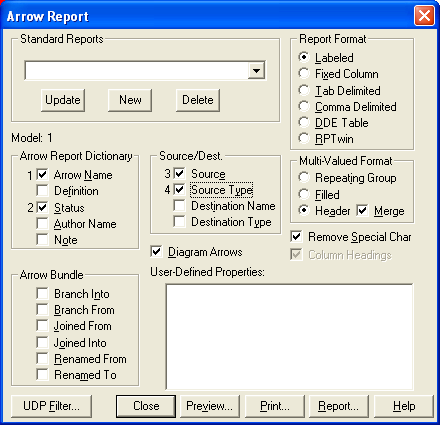
**Рис. 2.32.** Настройки отчета Diagram Object Report.

4) **Activity Cost Report**. Отчет о результатах стоимостного анализа, т.е. содержит информацию о затратах функциональных блоков и о «центрах затрат» в модели. Пример выбранных настроек данного отчета представлен на рис. 2.33.



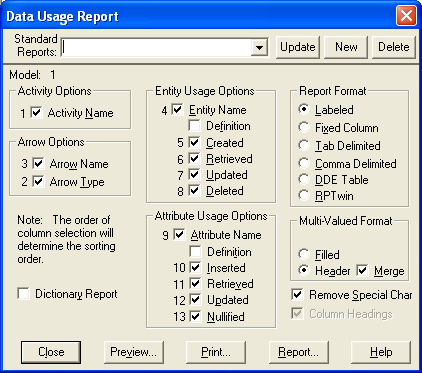
**Рис. 2.33.** Настройки отчета Activity Cost Report.

5) **Arrow Report**. Отчет по стрелкам, включает информацию из словаря стрелок, информацию о работе-источнике, работе-назначении стрелки и информацию о разветвлении и слиянии стрелок. Пример выбранных настроек данного отчета представлен на рис. 2.34.



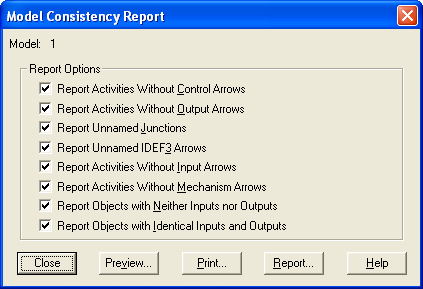
**Рис. 2.34.** Настройки отчета Arrow Report.

6) **Data Usage Report**. Отчет о результатах связывания модели процессов и модели данных, содержит информацию о таблицах базы данных или сущностях и атрибутах. Пример выбранных настроек данного отчета представлен на рис. 2.35.

****

**Рис. 2.35.** Настройки отчета Arrow Report.

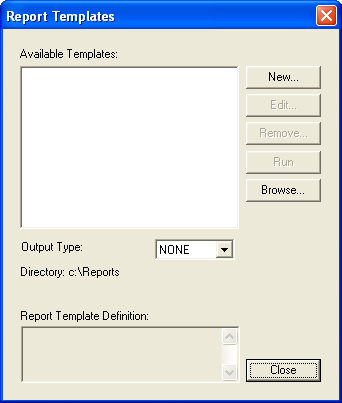
7) **Model Consistency Report**. Отчет, содержащий список синтаксических ошибок модели. Во-первых, это ошибки, которые BPwin выявить не в состоянии (синтаксис естественного языка). Во-вторых, это ошибки, которые BPwin просто не допускает (например, неправильное соединение стрелок и работ). В-третьих, это ошибки, которые BPwin позволяет допустить, но обнаруживает их. Пример выбранных настроек данного отчета представлен на рис. 2.36.

****

**Рис. 2.36.** Настройки отчета Model Consistency Report.

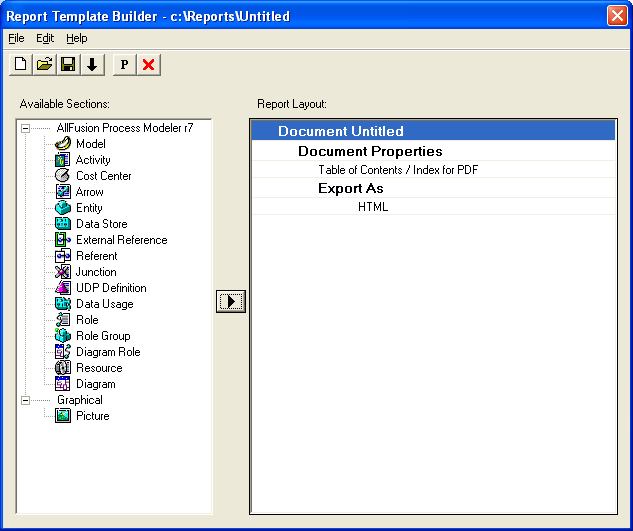
### 2.4.2. Отчеты с помощью шаблонов Report Template Builder

Собственный шаблон отчета можно создать с помощью диалога *Report Template Builder*. Для этого следует в меню **Tools** выбрать пункт **Reports Builder**. Откроется диалоговое окно ***Report Templates*** (см. рис. 2.37). Для создания нового шаблона следует нажать кнопку New (для редактирования существующего шаблона следует нажать кнопку Edit).

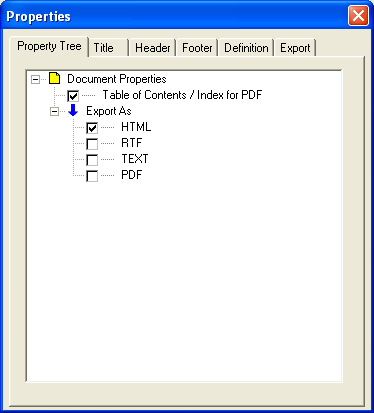
****

**Рис. 2.37.** Диалоговое окно ***Report Templates***.

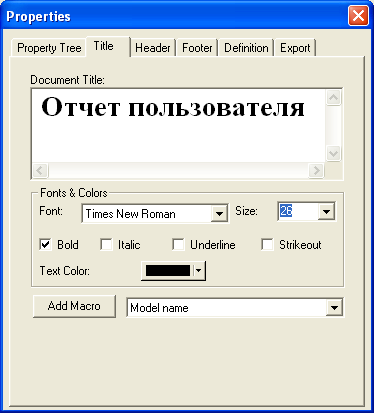
При работе с новым или существующим шаблоном откроется диалоговое окно ***Report Template Builder*** (см. рис. 2.38). Для создания новой секции отчета необходимо выбрать тип объекта модели и нажать кнопку . По умолчанию в новый шаблон включено только имя объекта. Для включения других свойств следует в меню **Edit** выбрать пункт **Properties**. Откроется диалоговое окно ***Properties*** (см. рис. 2.39). На вкладке **Property Tree** можно выбрать требуемые объекты, свойства объекта, а на вкладке **Title** – стиль, размер и цвет шрифта (см. рис. 2.40).



**Рис. 2.38.** Окно начальной настройки отчета.



**Рис. 2.39.** Окно выбора формата вывода отчета.



**Рис. 2.40.** Окно настройки заголовка отчета.

### 2.4.3. Отчеты с помощью генератора RPTwin

Специализированный генератор отчетов RPTwin позволяет создавать качественные отчеты по моделям процессов и данных благодаря включению более 40 функций, что позволяет проводить сложную обработку данных. Однако, RPTwin не входит в поставку BРwin, поэтому в данном учебном пособии он рассмотрен не будет.

# 3. Создание модели данных средствами ERwin 7.0

Для построения модели данных Computer Associates предлагает мощный и удобный инструмент – ERwin версии 7.0. ERwin имеет два уровня представления модели – логический и физический. На логическом уровне данные не связаны с конкретной системой управления базами данных (СУБД), поэтому могут быть наглядно представлены даже для неспециалистов. Физический уровень данных – это по существу отображение системного каталога, который зависит от конкретной реализации СУБД. ERwin позволяет проводить процессы прямого и обратного проектирования базы данных (БД). Это означает, что по модели данных можно сгенерировать схему БД или автоматически создать модель данных на основе информации системного каталога. Реализация моделирования в ERwin базируется на теории реляционных баз данных и на методологии IDEF1X. Методология IDEF1X была разработана для ВВС США и теперь используется, в частности, в правительственных, аэрокосмических и финансовых учреждениях, а также в большом числе частных компаний. Методология IDEFIX определяет стандарты терминологии, используемой при информационном моделировании, и графического изображения типовых элементов на диаграммах.

Построение модели данных определяет все последующие этапы разработки. Сначала осуществляют разработку логической модели, которая включает в себя следующие этапы [19]:

- выделение сущностей;

- выявление связей между сущностями и построение модели «сущность-связь» (ER Diagram);

- определение первичных ключей (Primary Key) и внешних ключей (Foreign Key);

- определение атрибутов сущностей и построение полной модели (Fully Attributed Model).

В процессе построения все сущности (отношения) должны быть приведены к *третьей* нормальной форме, после чего приступают к построению физической модели.

**Примечание.** Введено пять уровней нормализации схем отношений и соответственно пять нормальных форм отношений.

Каждая нормальная форма:

- ограничивает определенный тип функциональной зависимости;

- устраняет соответствующие аномалии при выполнении операций над отношениями БД.

Все формы подчиняются правилу вложенности по возрастанию номеров. Иными словами, если отношение находится в 4НФ, то оно будет соответствовать и 3НФ, и 2НФ, и 1НФ.

**1НФ.** Отношение находится в первой нормальной форме в том случае, если не первичные элементы отношения функционально зависят от первичных (ключевых) элементов или схема отношения находится в первой нормальной форме тогда и только тогда, когда все входящие в нее атрибуты являются атомарными (т.е. значения соответствующих доменов рассматриваются как неделимые, а не как множества или кортежи из более элементарных доменов).

**2НФ.** Отношение находится во второй нормальной форме, если оно находится в 1НФ и каждый не первичный элемент функционально полно зависит от каждого ключевого элемента или когда все элементы первичны или каждый ключ содержит один элемент.

**3НФ.** Отношение находится в третьей нормальной форме, если оно входит во 2НФ и каждый не первичный атрибут не транзитивно зависит от первичного ключа. Иными словами в отношении отсутствуют транзитивные зависимости не ключевых атрибутов от ключа.

**4НФ.** Отношение находится в четвертой нормальной форме, если оно находится в 3НФ и в нем присутствуют многозначные функциональные зависимости.

**5НФ.** Отношение находится в пятой нормальной форме, если оно находится в 4НФ и в нем устранена избыточность в отношениях со многими многозначными зависимостями, а также устранена аномалия обновления.

Перед построением физической модели необходимо определиться с тем, на какой платформе будет функционировать система, поскольку от этого зависит, какие типы данных она будет поддерживать, и какой диалект SQL использовать. Последующая работа разбивается на этапы: определение таблиц, определение и полей и их типов данных, определение ограничений на значения полей, определение связей между таблицами, разработка хранимых процедур (если они требуются).

## 3.1. Создание логической модели данных

Логическими уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире.

Объекты модели, представляемые на логическом уровне, называются *сущностями и атрибутами*. Логическая модель данных может быть построена на основе другой логической модели, например на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Различают три уровня логической модели, отличающихся по глубине представления информации о данных [19]:

* диаграмм сущность-связь (Entity Relationship Diagram, ERD);
* модель данных, основанная на ключах (Key Based model, KB);
* полная атрибутная модель (Fully Attributed model, FA).

Диаграмма сущность-связь представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС.

Диаграмма сущность-связь может включать связи многие-ко-многим и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

Модель данных, основанная на ключах, - более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

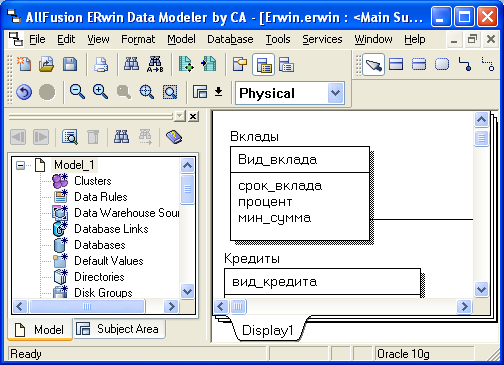
Полная атрибутивная модель – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

В данном учебном пособии будет рассмотрена модель данных, основанная на ключах.

Основные компоненты диаграммы ERwin – это сущности, атрибуты и связи.

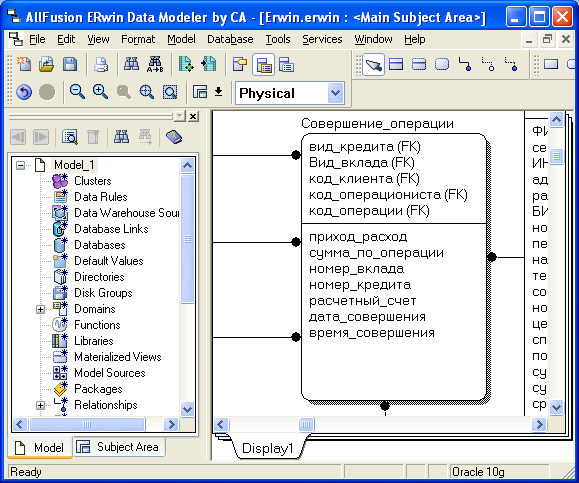
Каждая **сущность** является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Построение модели данных предполагает определение сущностей и атрибутов, т.е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. **Сущность** можно определить как объект, событие или концепцию, информация о которой должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном числе, не носить «технических» наименований и быть достаточно значимыми для того, чтобы их моделировать. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра.

Сущность на диаграмме изображается прямоугольником. В зависимости от режима представления диаграммы прямоугольник может содержать имя сущности, ее описание, список ее атрибутов и другие сведения (см. рис. 3.1).



**Рис. 3.1.** Сущность с заполненными атрибутами.

Экземпляры независимой сущности могут быть уникально идентифицированы без определения ее связей с другими сущностями; зависимая сущность, наоборот, не может быть уникально идентифицирована без определения ее связей с другими сущностями. Зависимая сущность отображается в ERwin прямоугольником с закругленными углами (см. рис. 3.2).



**Рис. 3.2.** Зависимая сущность с заполненными атрибутами.

Зависимая сущность может наследовать один и тот же внешний ключ от более чем одной родительской сущности, или от одной и той же родительской сущности через использование несколько связей. Если не введены различные роли для такого множественного наследования, ERwin считает, что в зависимой сущности атрибуты внешнего ключа появляются только один раз.

В зависимости от того, все ли возможные сущности-подтипы включены в модель, категорийная связь является полной или неполной. Например, если супертип может содержать данные об уволенных сотрудниках, то эта связь - неполной категоризации, так как для него не существует записи в сущностях - подтипах. В ERwin полная категория изображается окружностью с двумя подчеркиваниями, а неполная - окружностью с одним подчеркиванием.

**Унификация** - это объединение двух или более групп атрибутов внешних ключей в один внешний ключ (группу атрибутов), в предположении, что значения одноименных атрибутов в дочерней сущности всегда одинаковы. Рассмотрим пример: сущность "сотрудник" имеет первичный ключ "код сотрудника" и связан идентифицирующей связью с сущностями "супруга" и "дети". При этом происходит миграция первичного ключа в зависимые сущности. В свою очередь, сущность "супруга" связана не идентифицирующей связью с сущностью "дети". Имеются два пути миграции ключа, однако в сущности "дети" атрибут "код сотрудника" появляется один раз в качестве элемента первичного ключа. Существуют случаи, когда унификация атрибутов дает неверный с точки зрения предметной области результат. Для отмены унификации для атрибутов вводятся имена ролей.

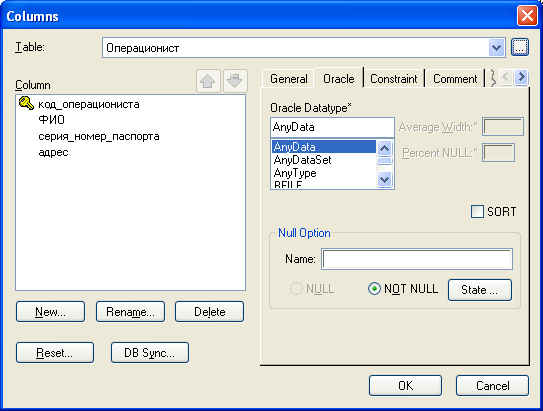
**Атрибут** выражает свойство объекта, характеризующее его экземпляр (определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы. Горизонтальная линия прямоугольника разделяет атрибуты сущности на два набора: атрибуты, составляющие первичный ключ (в верхней части) и прочие, не входящие в первичный ключ (в нижней части).

**Первичный ключ** – это атрибут или набор атрибутов, уникально идентифицирующий экземпляр сущности. Если несколько наборов атрибутов могут уникально идентифицировать сущность, то выбор одного из них осуществляется разработчиком на основании анализа предметной области. Для каждого первичного ключа ERwin создает при генерации структуры БД уникальный индекс.

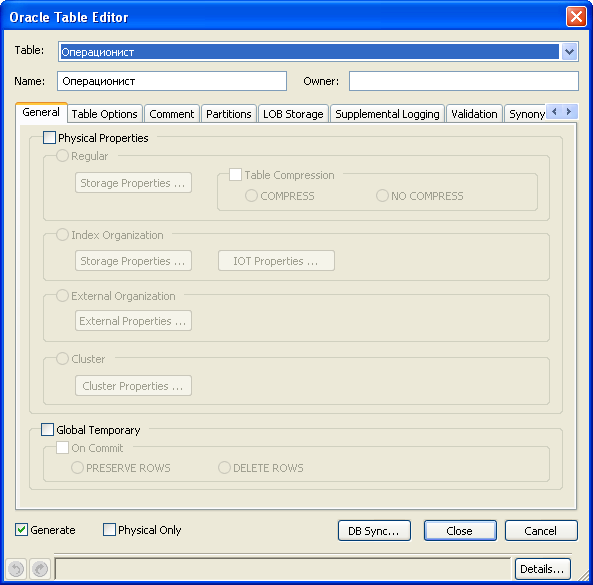
Если между некоторыми сущностями существует связь, то факты из одной сущности ссылаются или некоторым образом связаны с фактами из другой сущности. **Связь** – это функциональная зависимость между сущностями. Поддержание непротиворечивости функциональных зависимостей между сущностями называется ссылочной целостностью. Поскольку связи содержатся "внутри" реляционной модели, реализация ссылочной целостности может выполняться как приложением, так и самой СУБД (с помощью механизмов декларативной ссылочной целостности, триггеров). Связь это понятие логического уровня, которому соответствует внешний ключ на физическом уровне. Связь называется **идентифицирующей**, если экземпляр дочерней сущности идентифицируется через ее связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в первичный ключ дочерней сущности. Дочерняя сущность при идентифицирующей связи всегда является зависимой. Связь называется **не идентифицирующей**, если экземпляр дочерней сущности идентифицируется иначе, чем через связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в состав не ключевых атрибутов дочерней сущности.

Для добавления сущности следует нажать кнопку , а затем – щелкнуть «мышью» по свободному месту диаграммы. После этого по созданному элементу следует щелкнуть два раза левой кнопкой «мыши». В открывшемся диалоговом окне ***Columns*** (см. рис. 3.3) следует:

* при нажатии кнопки  в строке Table откроется диалоговое окно ***Oracle Table Editor*** (см. рис. 3.4), в котором в строке Name следует ввести название сущности;



**Рис. 3.3.** Окно для заполнения атрибутов сущности.



**Рис. 3.4.** Окно для заполнения названия сущности.

* при нажатии кнопки New ввести название и тип добавляемого атрибута (список форматов возможных типов представлен в табл. 3.1), а при необходимости установить ему ключевой признак вводом флажка *Primary Key*.

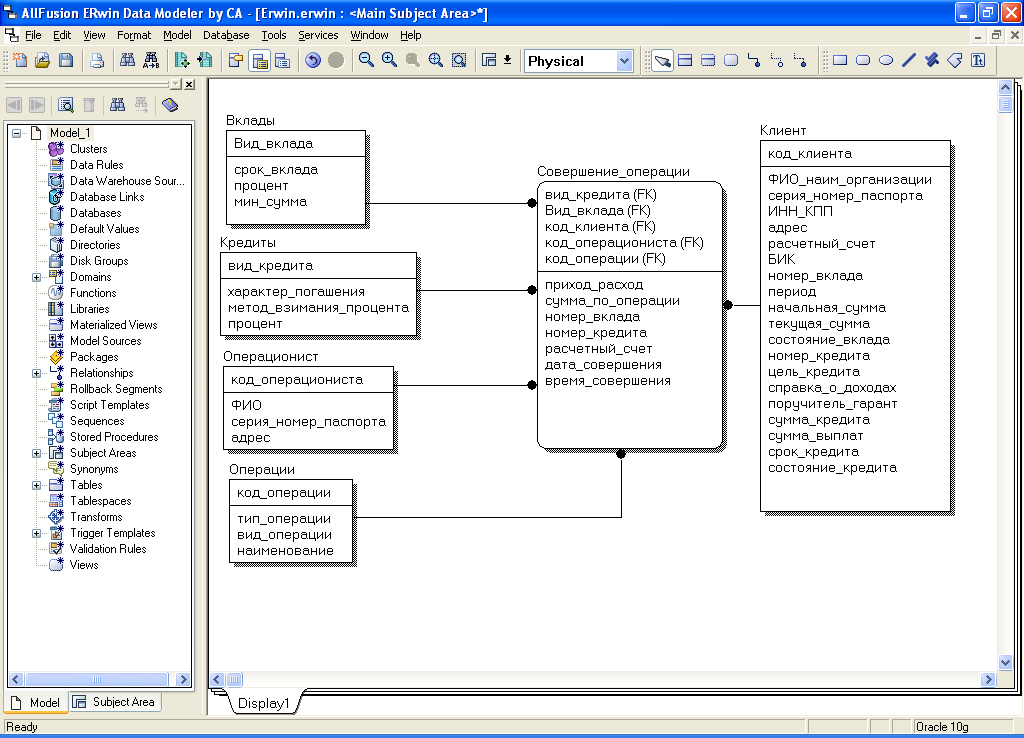
Таблица 3.1.

Расшифровка назначения типов атрибутов

|  |  |
| --- | --- |
| **Тип** | **Формат** |
| \_default\_ | Не определен |
| Blob | Счетчик |
| Datetime | Дата или время |
| Number | Числовой |
| String | Текстовый |

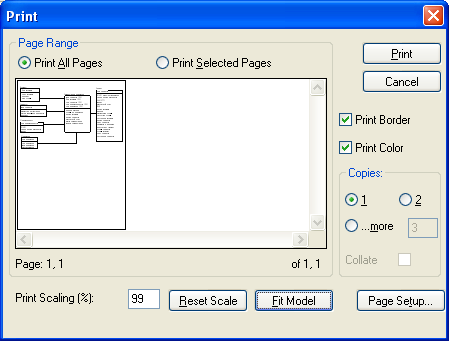
Для связывания таблиц следует, нажав кнопку (идентифицирующая связь один-ко-многим) или  (идентифицирующая связь многие-ко-многим), или  (не идентифицирующая связь), щелкнуть левой кнопкой «мыши» на одной таблице, а затем щелкнуть «мышью» на другой таблице, с которой требуется выполнить связь.

Пример логической модели данных представлен на рис. 3.5.



**Рис. 3.5.** Пример логической модели данных

Для компактного расположения модели на листе бумаги при печати следует вызвать в меню **File** режим **Print**, а в открывшемся окне ***Print*** (см. рис. 3.6) нажать кнопку Fit Model.



**Рис. 3.6.** Окно для настройки параметров печати.

## 3.2. Создание физической модели данных

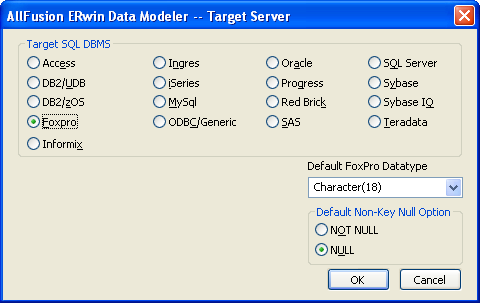
Различают два уровня физической модели:

* трансформационная модель (Transformation Model);
* модель СУБД (DBMS Model).

Физическая модель содержит всю информацию, необходимую для реализации конкретной БД. Трансформационная модель содержит информацию для реализации отдельного проекта, который может быть частью общей ИС и описывать подмножество предметной области. ERwin поддерживает ведение отдельных проектов, позволяя проектировщику выделять подмножество модели в виде предметных областей (Subject Area). Трансформационная модель позволяет проектировщикам и администраторам БД лучше представлять, какие объекты БД хранятся в словаре данных, и проверить, насколько физическая модель данных удовлетворяет требованиям к ИС.

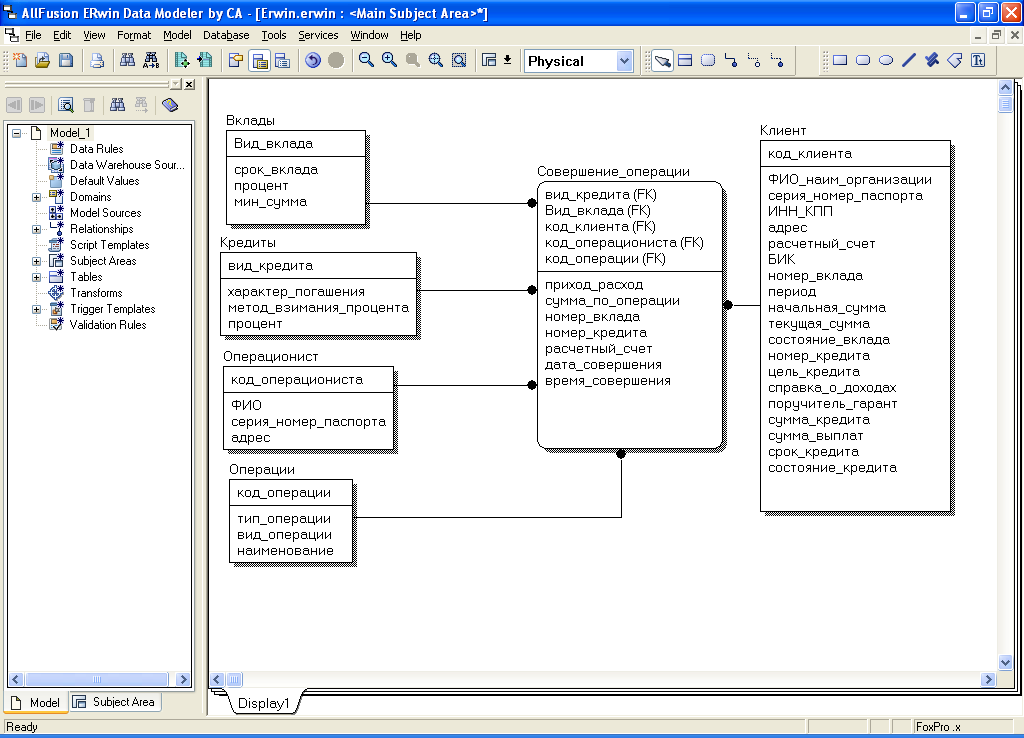
Модель СУБД автоматически генерируется из трансформационной модели и является точным отображением системного каталога СУБД. ERwin непосредственно поддерживает эту модель путем генерации системного каталога [19].

Физический уровень представления модели зависит от выбранного сервера. Для смены БД, в которой будет реализована физическая модель, следует в меню **Database** выбрать режим **Choose Database**. В открывшемся окне ***All Fusion Erwin Data Modeler – Target Server*** (см. рис. 3.7) в блоке **Target SQL DBMS** следует установить переключатель на имени требуемой БД. В нижней части данного окна сразу будет отражен стандарт данной БД по длине и формату полей.



**Рис. 3.7.** Окно для выбора базы данных (FoxPro).

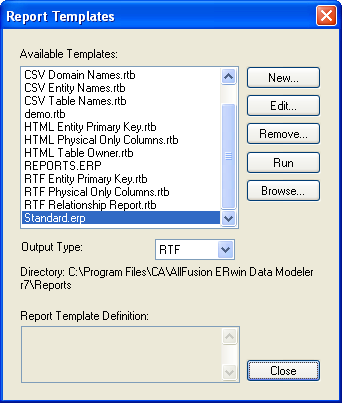
Пример физической модели данных представлен на рис. 3.8.



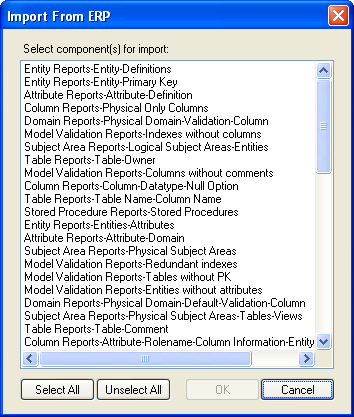
**Рис. 3.8.** Пример физической модели данных(FoxPro)

Диаграмма физической модели является необходимым, почти достаточным и очень удобным материалом для разработчиков программ.

По завершении работы над информационной моделью, как правило, распечатываются логический и физический уровни диаграммы, а также отчет по соответствиям сущность-таблица, атрибут-имя колонки, сущность-атрибуты. Для этого в меню **Tools** следует выбрать пункт **Report Template Builder**, в котором – подпункт **Report Builder**. Откроется диалоговое окно ***Report Templates*** (см. рис. 3.9), в котором следует выбрать тип отчета (например, *Standard.erp*) и тип представления выходных данных (например, RTF) и нажать кнопку Run. Откроется диалоговое окно ***Import From ERP*** (см. рис. 3.10), в котором следует нажать кнопку Select All, а затем – кнопку ОК.



**Рис. 3.9.** Окно для выбора типа отчета (Standard.erp)



**Рис. 3.10.** Окно для выбора компонентов отчета

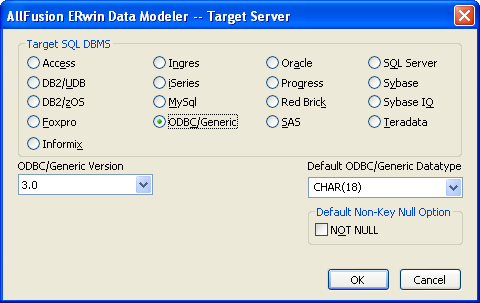
Сгенерированный отчет может быть сохранен на диск (колонки разделяются запятыми, выравниваются или разделяются табуляцией) или передан в текстовый процессор (или электронную таблицу) через интерфейс DDE.

## 3.3. Экспорт данных в СУБД

Завершением построения логической, а также физической модели данных является экспорт созданных таблиц и их полей в системы управления базами данных (СУБД).

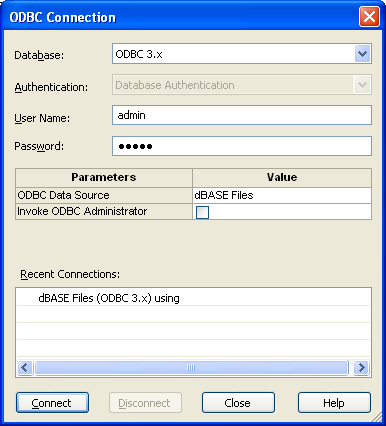
Рассмотрим экспорт данных на примере СУБД dBASE [19].

Следует установить режим просмотра физической модели данных (меню **Model** режим **Physical model**). После этого в меню **Database** необходимо выбрать режим **Choose Database**. В открывшемся окне ***All Fusion ERwin Data Modeler – Target Server*** установить переключатель на **ODBC/Generic**, выбрать версию 3.0 из списка **ODBC/Generic Version** (см. рис. 3.11) и нажать кнопку ОК.

****

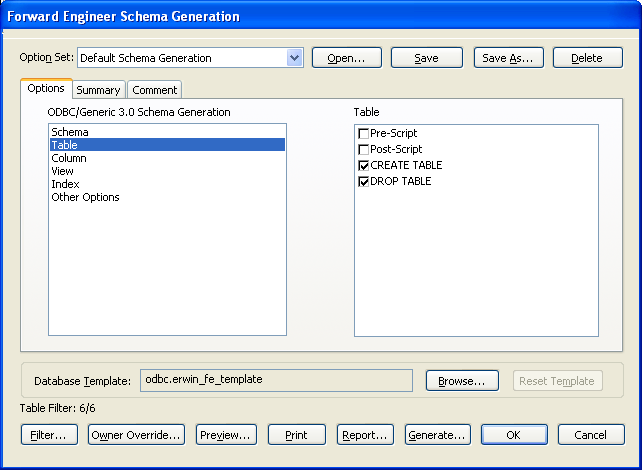
**Рис. 3.11.** Окно для настройки на экспорт данных через ODBC.

В открывшемся окне ***ODBC Connection*** (см. рис. 3.12) следует ввести: имя пользователя (User Name), пароль (Password), имя базы данных (Database) и нажать кнопку Connect.



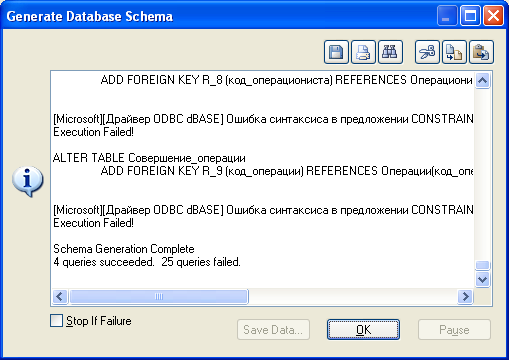
**Рис. 3.12.** Окно для настройки на экспорт данных через ODBC.

После этого в меню **Tools** необходимо выбрать в режиме **Forward Engineer** подрежим **Schema Generation**. В открывшемся окне ***Forward Engineer Schema Generation*** (см. рис. 3.13) установить на вкладке Options требуемые флажки (в блоках Table, View, Index следует установить флажки *Drop…* для удаления созданных ранее таблиц, схем, процедур) и нажать кнопку Generate.



**Рис. 3.13.** Окно для настройки экспортируемых данных.

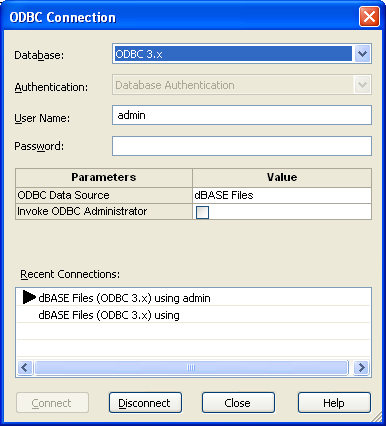
Откроется информационном окно ***Generate Database Schema*** (см. рис. 3.14) для просмотра информации по процессу генерации следует нажимать кнопку Continue, а по завершении процесса генерации – кнопку ОК.



**Рис. 3.14.** Окно для просмотра хода генерации данных в СУБД.

Сформированные файлы (формата dBASE) находятся в папке **C:\Program Files \ CA \ All Fusion ERwin Data Modeler r7** (здесь с расширением .DBF).

Для прекращения установленного соединения с СУБД следует в меню **Database** выбрать режим **Database Connection**. В открывшемся окне ***ODBC Connection*** (см. рис. 3.15) следует нажать кнопку Disconnect.



**Рис. 3.15.** Окно для настройки на разрыв соединения через ODBC.

# 4. Создание модели данных средствами IBM Rational Rose

Компьютерные и информационные технологии без преувеличения можно назвать наиболее динамичной областью современных знаний, которая концентрирует в себе самые последние достижения в сфере науки и техники.

На протяжении всего жизненного цикла существования современных корпоративных информационных систем в нем принимает участие множество специалистов различной квалификации, для которых единообразное понимание архитектуры и функциональности является серьезной проблемой [10].

Таким образом, все эти особенности приводят к настоятельной необходимости моделирования структуры и процесса функционирования программных систем до начала написания их программного кода. При этом непременным условием успешного завершения проекта становится построение предварительной модели программной системы.

## 4.1. Особенности объектно-ориентированного подхода

Стремление избавиться от недостатков структурного подхода привело к развитию новых идей, основанных на объектной декомпозиции. Такой подход к разработке программных систем получил название объектно-ориентированного. В основе его лежат понятия «объект», «класс», «инкапсуляция» и «полиморфизм». В реальном мире, а точнее, в интересующей разработчика предметной области, в качестве объектов могут рассматриваться конкретные предметы, а также абстрактные или реальные сущности. Например, объектами могут быть покупатель; фирма, производящая определенные товары; банк; заказ на поставку. Объект обладает индивидуальностью и поведением, имеет атрибуты, значения которых определяют его состояние. Так, конкретный покупатель, делая заказ, может оказаться в состоянии, когда денег на его счете не хватает для оплаты, а его «поведение» в этом случае заключается в «обращении в банк за кредитом».

Каждый объект является представителем некоторого класса однотипных объектов. Класс определяет общие свойства для всех его объектов. К таким свойствам относятся [20]:

* состав и структура данных, описывающих атрибуты класса и соответствующих объектов;
* совокупность методов — процедур, определяющих взаимодействие объектов этого класса с внешней средой.

Например, описание класса «магазины» может включать некоторые атрибуты (индивидуальные для каждого объекта этого класса — конкретного магазина): «название», «адрес», «штат сотрудников», а также методы: «формирование заказов на поставку товаров», «передача товара со склада в торговую секцию» и т.д. Объекты и классы обладают характерными свойствами, которые активно используются при объектно-ориентированном подходе и во многом определяют его преимущества.

Инкапсуляция — это скрытие информации. При объектно-ориентированном программировании предусмотрена возможность запрета доступа к атрибутам объектов, кроме как через его методы. Внутренняя структура объекта в этом случае скрыта от пользователя, т.е. объекты можно считать самостоятельными сущностями, отделенными от внешнего мира. Для того чтобы объект произвел некоторое действие, ему извне необходимо послать сообщение, которое инициирует выполнение нужного метода. Инкапсуляция позволяет изменять реализацию любого класса объектов без опасения, что это вызовет нежелательные побочные эффекты в программной системе. Тем самым упрощается процесс исправления ошибок и модификации программ.

Наследование — возможность создавать из классов новые классы по принципу «от общего к частному». Наследование позволяет новым классам при сохранении всех свойств классов-родителей (называемых в дальнейшем суперклассами) добавлять свои черты, отражающие их индивидуальность. С точки зрения программиста новый класс должен содержать только коды и данные для новых или изменяющихся методов. Сообщения, обработка которых не обеспечивается собственными методами класса, передаются суперклассу. Наследование позволяет создавать иерархию классов и является эффективным средством внесения изменений и дополнений в программные системы.

Полиморфизм — способность объектов выбирать метод на основе типов данных, применяемых в сообщении. Каждый объект может реагировать по-своему на одно и то же сообщение. Полиморфизм позволяет упростить исходные тексты программ, обеспечивает их развитие за счет введения новых методов обработки.

Объектно-ориентированная декомпозиция заключается в представлении системы в виде совокупности классов и объектов предметной области. При этом иерархический характер сложной системы отражается в виде иерархии классов, а ее функционирование рассматривается как взаимодействие объектов.

Жизненный цикл объектно-ориентированной разработки программных систем содержит несколько этапов, но в отличие от структурного подхода в нем нет строгой последовательности их выполнения. Процесс носит принципиально итеративный характер, что полностью отвечает потребностям разработчиков [10].

Разработка начинается с этапа исследования — объектно-ориентированного анализа. Здесь предъявляются требования к системе. Затем осуществляется анализ предметной области, в ходе которого определяются классы и объекты, которые составляют словарь предметной области. Результатом исследования должно быть получение достаточно полных сведений для создания модели системы.

После исследования начинается объектно-ориентированное проектирование, в ходе которого детализируется представление классов и объектов, полученных на этапе анализа. Определяются структуры данных, методы, отношения между классами, разрабатываются сценарии взаимодействия объектов. При проектировании системы могут вводиться новые классы и объекты, если это потребуется для решения поставленных проблем. В результате проектирования должна быть создана детальная модель системы, составлены спецификации объектов, классов и отношений, достаточные для их программирования.

Программирование, тестирование и сборку системы Г. Буч рассматривает как единый этап, называемый эволюцией системы. Объектно-ориентированный подход обеспечивает быстрое создание прототипов проектируемой системы, постепенное развитие которых приводит к конечному результату. На этом этапе также возможно введение новых классов, изменение структур данных, добавление новых методов. Следует заметить, что программирование и тестирование отдельных компонентов системы возможно до завершения проектирования, что экономит время разработки. Современные объектно-ориентированные инструментальные средства, применяемые при разработке программных систем, обычно обладают возможностями автоматизации процессов, выполняемых на этом этапе. Модификация системы может рассматриваться как отдельный этап. Возможность внесения изменений — естественное свойство сложных систем, обеспечивающее их развитие. При объектно-ориентированном подходе модификация не требует полного пересмотра проекта, затрагивая лишь необходимые для этого классы и объекты.

Главная особенность жизненного цикла при объектно-ориентированном подходе заключается в том, что нет строгой последовательности выполнения отдельных этапов. При разработке может выясниться необходимость дополнительного исследования; программирование и последующее тестирование могут потребовать возврата к проектированию. Такой метод, названный Г. Бучем возвратным, отражает итеративный характер разработки приложения.

## 4.2. Преимущества объектно-ориентированного подхода

**Распараллеливание работ.** При проектировании может возникнуть необходимость внесения изменений в существующие классы или потребоваться введение новых объектов или классов. В этом случае, вернувшись к этапу проектирования или даже к анализу, можно внести изменения и дополнения, не подвергая проект полной переработке [10].

***Упрощение внесения изменений***. В отличие от структурного подхода, в объектно-ориентированном подходе внесение изменений в проект имеет более локальный характер. В тех случаях, когда изменение носит характер уточнения, локализации, вводятся новые классы, наследующие поведение ранее созданных. Наследование — одно из основных свойств классов — позволяет в этих случаях не только не пересматривать ранее созданные объекты и классы, но даже обойтись без их повторной трансляции. В более сложных случаях, когда меняются методы, определяющие интерфейс классов, изменения в проекте будут более значительными, но и тогда они будут локализованы, затрагивая лишь классы, использующие эти методы.

***Гибкая архитектура и переносимость***. Объектно-ориентированная декомпозиция, в результате которой приложение представляется в виде совокупности классов и объектов, обеспечивает гибкость архитектуры системы. В клиент-серверной системе объекты могут размещаться как на местах клиента, так и на сервере. В гетерогенных (разнородных) сетях возможна реализация классов на компьютерах разных типов, а фиксированный интерфейс каждого класса, определяемый набором его методов, обеспечит правильность функционирования системы. Изменения конфигурации оборудования не требуют внесения изменений в проект.

***Повторное использование программных компонентов***. Разрабатываемые в рамках некоторого приложения классы обычно отражают типовые решения, поэтому их использование возможно и в других приложениях. Возможность повторного использования программных компонентов — одна из самых привлекательных черт объектно-ориентированного подхода. Библиотеки классов, отражающие опыт в определенной области, позволяют значительно снизить объем программирования при разработке новых приложений. При наличии развитых библиотек классов проектирование и программирование новых приложений будет в основном сводиться к сборке системы из готовых компонентов.

Для того чтобы повторное использование компонентов приносило свои плоды, разработчики программных систем должны:

* осознавать выгоды такого подхода;
* знать, какие части задачи могут быть решены с применением существующих программных средств;
* заниматься поиском подходящих для повторного использования программ;
* стремиться найти такие программы;
* использовать их даже в том случае, если они лишь частично совпадают с тем, что программист написал бы сам.

Следует заметить, что основные свойства классов и объектов — инкапсуляция, наследование и полиморфизм — полностью отвечают задаче повторного использования. В связи с этим в последние годы значительно возрос интерес к объектно-ориентированным информационным системам, которые представляют удобные услуги, связанные с повторным использованием программных компонентов.

***Естественность описания***. Объектно-ориентированный подход позволяет описывать как статические, так и динамические отношения между объектами модели. По описанию предметной области, выполненному на естественном языке, легко выделить объекты и статические связи между ними. Объекты соответствуют существительным, а связи — глаголам и отглагольным формам. Например, фраза «фирмы выполняют заказ» позволяет выделить классы объектов «фирма» и «заказ» и отношение между ними типа M:N, так как фирма может выполнять много заказов, а заказ может быть реализован разными фирмами.

Кроме того, свойства наследования и инкапсуляции предоставляют возможность каждому участнику проекта рассматривать модель с удобным для него уровнем детализации. Руководители проекта могут работать с верхним уровнем модели, где отражаются только основные классы, объекты и связи. Другие разработчики или эксперты имеют возможность работать с более мелкими терминальными объектами, их свойствами, связями, методами.

## 4.3. Недостатки объектно-ориентированного подхода

Недостатки самого объектно-ориентированного подхода лежат в области программирования. Динамическое связывание, предполагающее поиск метода в классе, которому принадлежит получающий сообщение объект, приводит к тому, что обращение к методу занимает в 1,75-2,5 раза больше времени, чем к обычной подпрограмме. Это, конечно, замедляет работу приложения. Однако, как указывает Г. Буч, динамическое связывание при использовании строго типизированных языков применяется примерно в 20% случаев вызовов методов. В результате снижаются непроизводительные потери времени. В приложениях, где такие потери критичны, приходится прибегать к специальным программистским приемам [10].

Другой недостаток связан с многочисленностью методов и их излишними вызовами. Это объясняется тем, что для доступа ко многим атрибутам объектов (а к защищенным — всегда) используются специальные методы. Вызов метода высокого уровня абстракции приводит к тому, что в системе происходит каскад вызовов — от методов более высоких уровней иерархии к методам более низких уровней. Если время является ограничивающим фактором, такая ситуация может оказаться неприемлемой. Выход здесь видится в том, что после создания начального варианта системы производится его оптимизация для сокращения количества вызовов. Например, защищенные переменные можно сделать общедоступными и обращаться к ним напрямую, уменьшая тем самым число вызовов.

На компьютерах с сегментированной организацией памяти объектно-ориентированные системы при работе могут осуществлять интенсивный межсегментный обмен, что отрицательно сказывается на их производительности. Это вызвано тем, что классы обычно объявляются в разных файлах и соответственно реализуются в разных сегментах. Проблема решается путем перераспределения классов по модулям, логическое описание модели не изменяется.

Для задач реального времени, выполняющихся в высоком темпе, нежелательным является динамическое создание и удаление объектов, что также активно используется в объектно-ориентированных языках. Можно размещать такие объекты при создании программы, а не во время работы критичных по времени алгоритмов. Преодоление перечисленных затруднений связано с дополнительной работой программистов, но в то же время не требует очень больших усилий, так как действия, которые надо предпринять, достаточно, очевидны. Кроме того, подобные проблемы возникают весьма редко. Следует также заметить, что объектно-ориентированные языки обладают средствами, позволяющими достичь более высокого быстродействия программ по сравнению с традиционными языками.

Существует расхожее мнение, что объектно-ориентированный подход труден для понимания, поэтому переход на объектно-ориентированные технологии связан с большими затратами, которые не окупаются. В действительности дело обстоит по-другому. Традиционная и объектно-ориентированная технологии с точки зрения получаемых результатов по-разному ведут себя по отношению к затратам на их освоение. При использовании традиционных технологий некоторые результаты можно получить и при сравнительно небольших затратах, однако на определенной стадии наступает насыщение, когда даже значительные дополнительные затраты не приводят к существенному повышению эффективности.

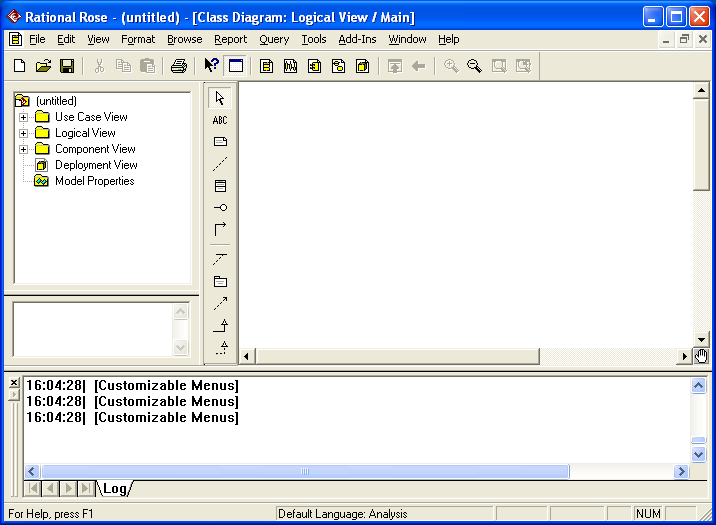
## 4.4. Особенности построения информационной модели

Работа над моделью в среде IBM Rational Rose начинается с общего анализа проблемы и построения диаграммы вариантов использования, которая отражает функциональное назначение проектируемой программной системы. Для вновь создаваемого проекта можно воспользоваться мастером типовых проектов, если он установлен в данной конфигурации. Мастер типовых проектов доступен из меню **File** в режиме **New** или при первоначальной загрузке программы IBM Rational Rose 2007. В случае разработки проекта, для которого неизвестна или не выбрана технология его реализации, следует отказаться от мастера, в результате чего появится рабочий интерфейс программы IBM Rational Rose 2007 с чистым окном активной диаграммы классов и именем проекта ***untitled*** по умолчанию.



### 4.4.1. Структура и настройка основного окна

Рабочий интерфейс программы IBM Rational Rose 2007 (см. рис. 4.1) состоит из различных элементов, основными из которых являются:



**Главное меню**

**Стандартная панель инструментов**

**Окно браузера проекта**

**Специальная панель инструментов**

**Окно документации**

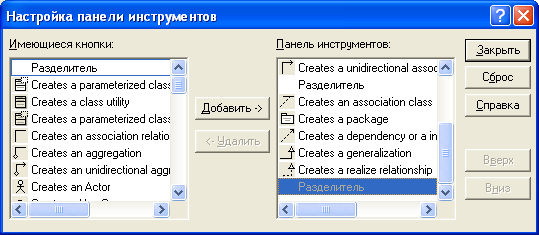
**Рабочая область изображения диаграммы**

**Окно журнала**

**Рис. 4.1.** Главное окно программы Rational Rose 2007.

* главное меню;
* стандартная панель инструментов;
* специальная панель инструментов;
* окно браузера проекта (организует представления модели в виде иерархической структуры, которая упрощает навигацию и позволяет отыскать любой элемент модели в проекте);
* рабочая область изображения диаграммы или окно диаграммы;
* окно документации (для документирования элементов разрабатываемой модели, вводится на русском языке);
* окно журнала (для служебной информации: фиксируется время и характер выполняемых разработчиком действий, таких как обновление модели, настройка меню и панелей инструментов, а также сообщений об ошибках, возникающих при генерации программного кода).

Для добавления или удаления кнопок на стандартной или специальной панелям инструментов следует нажать правой кнопкой мыши в любом месте требуемой панели и выбрать режим **Customize**. При настройке специальной панели инструментов открывается окно ***Настройка панели инструментов*** (см. рис. 4.2). Для добавления необходимых кнопок на панель следует выделить их в левом окне со списком пиктограмм графических элементов, после чего нажать кнопку Добавить в центре диалогового окна. Для удаления ненужных кнопок с панели инструментов следует выделить их в правом окне со списком пиктограмм графических элементов, после чего нажать кнопку Удалить в центре диалогового окна. Для восстановления набора пиктограмм по умолчанию можно нажать кнопку Сброс. После настройки специальной панели инструментов соответствующее окно следует закрыть нажатием на кнопку Закрыть.



**Рис. 4.2.** Окно настройки панели инструментов.

Иерархическое представление структуры каждого разрабатываемого проекта (см. ***Окно браузера проекта*** на рис.4.1) организовано в форме 4-х представлений:

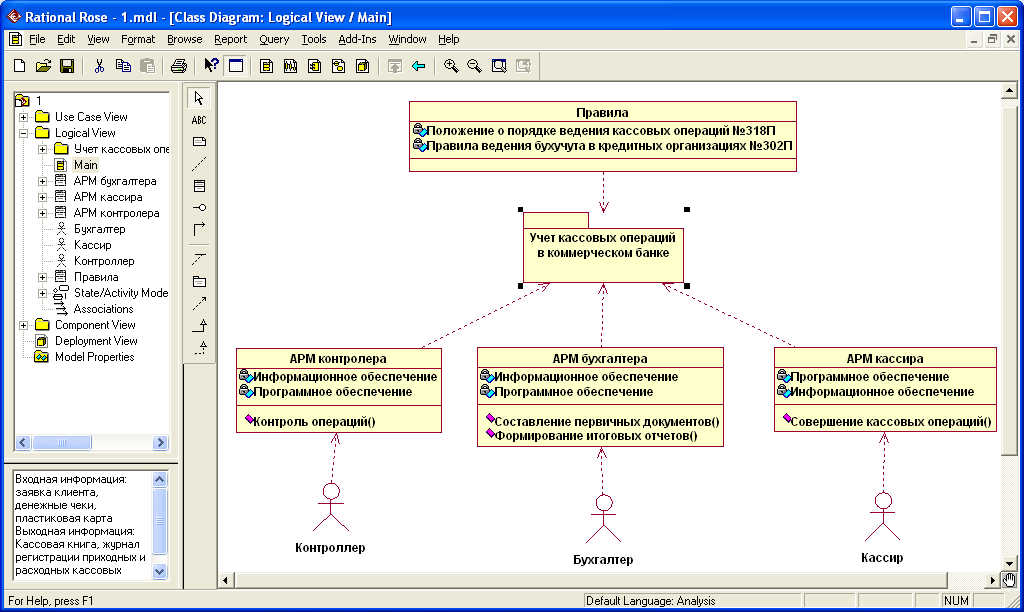
• *Use Case View* – представление вариантов использования, в котором содержатся диаграммы вариантов использования и их реализации в виде вариантов взаимодействия;

• *Logical View* – логическое представление, в котором содержатся диаграммы классов, диаграммы состояний и диаграммы деятельности;

• *Component View* – представление компонентов, в котором содержатся диаграммы компонентов разрабатываемой модели;

• *Deployment View* – представление развертывания, в котором содержится единственная диаграмма развертывания разрабатываемой модели.

Примерный вид диаграммы, относящийся к теме учета кассовых операций в коммерческом банке, содержится на рис. 4.3.



**Рис. 4.3.** Основная диаграмма по учету кассовых операций в коммерческом банке.

### 4.4.2. Особенности разработки диаграмм классов

Диаграмма классов является основным логическим представлением модели и содержит детальную информацию о внутреннем устройстве объектно-ориентированной программной системы или, используя современную терминологию, об архитектуре программной системы. Активизировать рабочее окно диаграммы классов можно несколькими способами [20]:

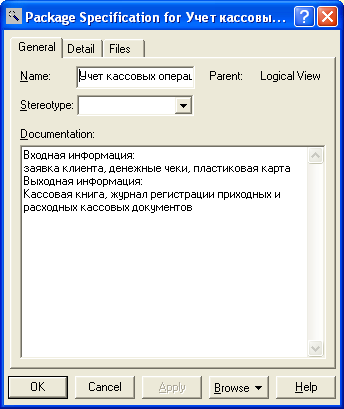
* окно диаграммы классов появляется по умолчанию в рабочем окне диаграммы после создания нового проекта;
* щелкнуть на кнопке с изображением диаграммы классов на стандартной панели инструментов;
* раскрыть логическое представление (*Logical View*) в ***Окне браузере проекта*** и дважды щелкнуть на пиктограмме **Main**;
* выполнить операцию главного меню: **Browse🡪Class Diagram**.

При этом появляется новое окно с чистым рабочим листом диаграммы классов и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы классов. Назначение кнопок специальной панели инструментов для диаграммы классов представлено в таблице 4.1 [20].

Таблица 4.1.

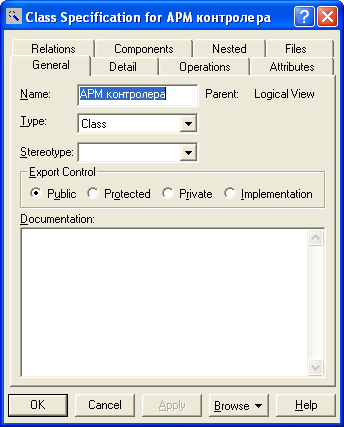
**Назначение кнопок специальной панели инструментов   
для диаграммы классов.**

|  |  |
| --- | --- |
| **Графическое изображение** | **Назначение кнопки** |
| 1 | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| 2 | Добавляет на диаграмму текстовую область |
| 3 | Добавляет на диаграмму примечание |
| 4 | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| 5 | Добавляет на диаграмму класс |
| 6 | Добавляет на диаграмму интерфейс |
| 7 | Добавляет на диаграмму направленную ассоциацию |
| 8 | Добавляет на диаграмму ассоциацию класс |
| 9 | Добавляет на диаграмму пакет (блок деятельности) |
| a | Добавляет на диаграмму отношение зависимости |
| b | Добавляет на диаграмму отношение обобщения |
| c | Добавляет на диаграмму отношение реализации |
| Актер | Добавляет на диаграмму актера (**необходимо предварительно добавить**) |
| Вариант_использования | Добавляет на диаграмму вариант использования (**необходимо предварительно добавить**) |

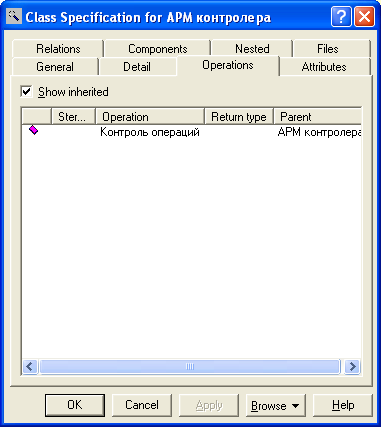
Для добавления в рабочую область *пакета* (*блок деятельности*) нужно на специальной панели инструментов выбрать кнопку9и щелкнуть мышью в рабочую область диаграммы. Затем следует ввести название блока. Для добавления реквизитов данного блока следует щелкнуть по нему правой кнопкой мыши и выбрать режим **Open Specification**. Откроется окно ***Package Specification*** (см. рис. 4.4). В данном окне на вкладке **General** вводится текстовая документация (дублируется в ***Окне документации*** на рис. 4.1).

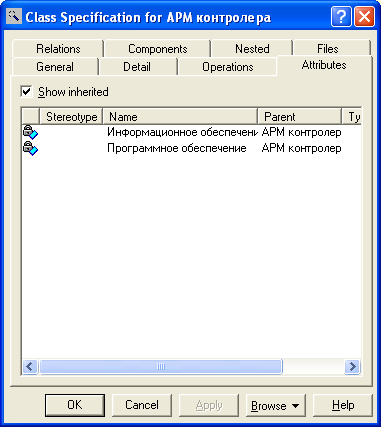
**Рис. 4.4.** Окно спецификации блока деятельности.

Для добавления в рабочую область *класса* нужно на специальной панели инструментов выбрать кнопку 5 и щелкнуть мышью в рабочую область диаграммы. Затем следует ввести название блока. Для изменения реквизитов данного блока следует щелкнуть по нему правой кнопкой мыши и выбрать режим **Open Specification**. Откроется окно ***Class Specification*** (см. рис. 4.5). В данном окне на вкладке **General** вводится текстовая документация (дублируется в ***Окне документации*** на рис. 4.1), на вкладке **Operations** вводятся названия операций класса (см. рис. 4.6), на вкладке **Attributes** вводятся названия атрибутов класса (см. рис. 4.7). Для добавления элементов на этих вкладках следует нажать правую кнопку мыши и выбрать режим **Insert**.



**Рис. 4.5.** Окно основной информации класса.



**Рис. 4.6.** Окно операций класса.

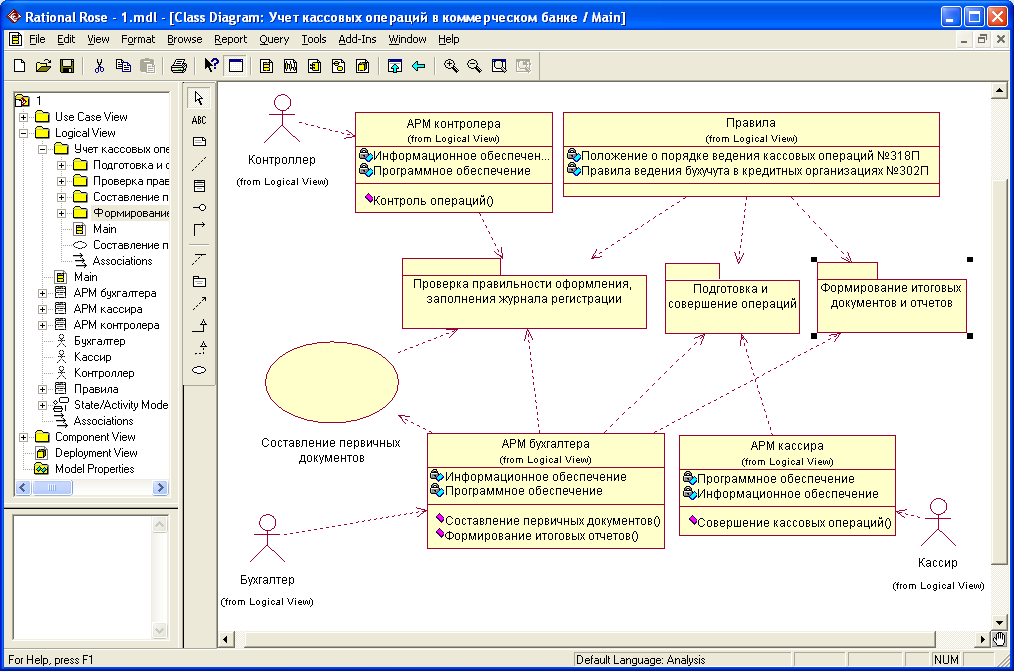
**Рис. 4.7.** Окно атрибутов класса.

Для добавления *отношение зависимости* между требуемыми элементами диаграммы нужно на специальной панели инструментов выбрать кнопку a, щелкнуть на зависимом элементе диаграммы (например, класс АРМ контролера на рис. 4.3), а затем – щелкнуть на управляющем элементе диаграммы (например, блок деятельности Учет кассовых операций в коммерческом банке на рис. 4.3).

Для того, чтобы перейти на следующий уровень иерархии, следует дважды щелкнуть мышью на требуемом блоке деятельности. При этом возврат назад осуществляется кнопкой Презентация1 или выбором требуемого уровня в ***Окне браузера проекта***, или выбором требуемого окна в **Главном меню** в режиме **Window**.

Для добавления элемента *вариант использования* на специальной панели инструментов выбрать кнопкуВариант_использованияи щелкнуть мышью в рабочую область диаграммы. Затем следует ввести название блока (см. блок *Составление первичных документов* на рис. 4.8). Элемент *вариант использования* следует использовать в том случае, если у данной работы нет иерархии.

**Примечание.** Элемент *вариант использования* необходимо предварительно добавить на специальную панель инструментов.



**Рис. 4.8.** Диаграмма по учету кассовых операций в коммерческом банке.

Для добавления элемента *актер* на специальной панели инструментов выбрать кнопкуАктери щелкнуть мышью в рабочую область диаграммы. Затем следует ввести название блока (см. блок *Бухгалтер* на рис. 4.3). Элемент *актер* следует использовать для обозначения управляющего воздействия.

**Примечание.** Элемент *актер* необходимо предварительно добавить на специальную панель инструментов.

### 4.4.3. Особенности разработки диаграммы деятельности

Для добавления в рабочую область *диаграммы деятельности* в **Браузере проекта** необходимо в открытом логическом представлении (Logical View) выбрать требуемый блок деятельности (например, на рис. 4.3 блок *Учет кассовых операций в коммерческом банке*). Затем на этом блоке в контекстном меню выбрать **New**🡪**Activity Diagram**. Теперь необходимо ввести название *диаграммы деятельности* (например, *итоговые документы*). После этого следует щелкнуть два раза левой кнопкой мыши по созданной диаграмме.

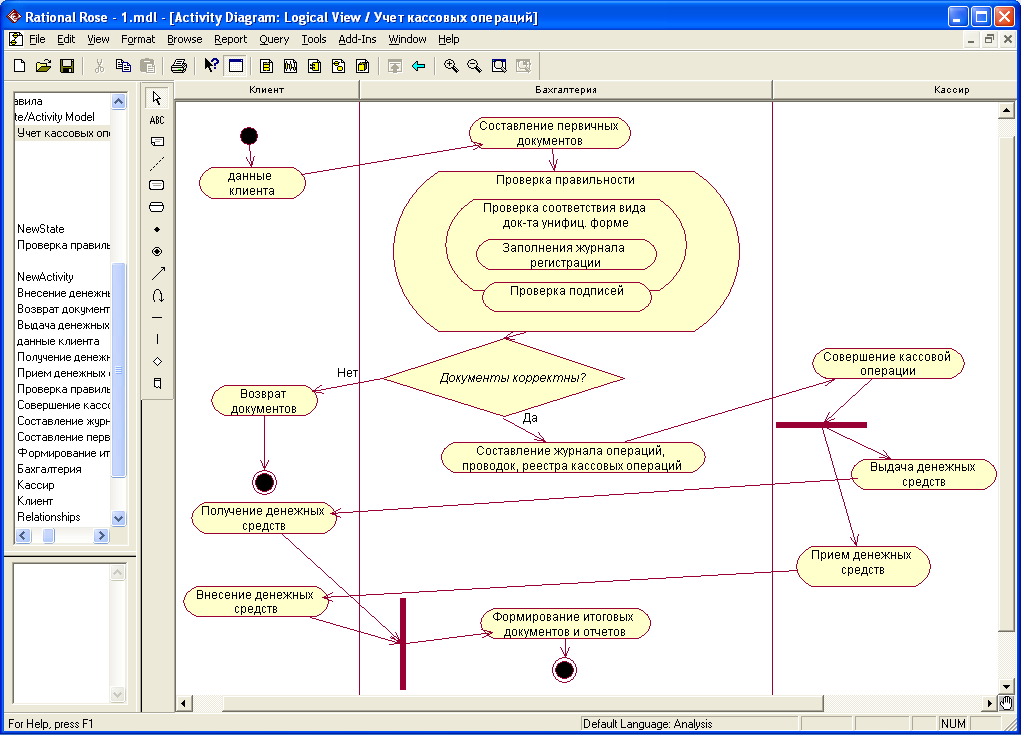
При этом появляется новое окно с чистым рабочим листом диаграммы деятельности и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы деятельности. Назначение кнопок специальной панели инструментов для диаграммы деятельности представлено в таблице 4.2 [20].

Таблица 4.2.

**Назначение кнопок специальной панели инструментов   
для диаграммы деятельности.**

|  |  |
| --- | --- |
| **Графическое изображение** | **Назначение кнопки** |
| 1 | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| 2 | Добавляет на диаграмму текстовую область |
| 3 | Добавляет на диаграмму примечание |
| 4 | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| 1 | Добавляет на диаграмму состояние |
| 2 | Добавляет на диаграмму *деятельность* |
| 3 | Добавляет на диаграмму начальное состояние |
| 4 | Добавляет на диаграмму конечное состояние |
| 5 | Добавляет на диаграмму *переход* |
| 6 | Добавляет на диаграмму рефлексивный *переход* |
| 7 | Добавляет на диаграмму горизонтально расположенный символ синхронизации |
| 8 | Добавляет на диаграмму вертикально расположенный символ синхронизации |
| 9 | Добавляет на диаграмму *символ принятия решения* для альтернативных *переходов* |
| 10 | Добавляет на диаграмму дорожку |

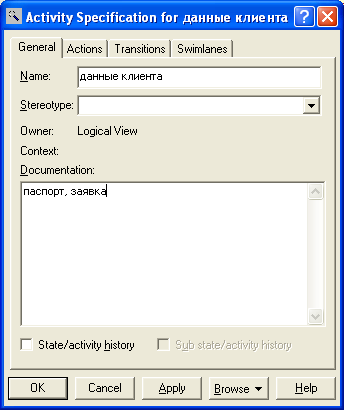
Для добавления *дорожки* на диаграмму деятельности нужно на специальной панели инструментов выбрать кнопку 10 и щелкнуть мышью на свободном месте рабочего листа диаграммы. В результате этих действий на диаграмме в области диаграммы появится изображение *дорожки* с вертикальной линией и именем дорожки *New Swimlane* в верхней части, предложенное программой по умолчанию. Следует ввести требуемое название *дорожки* (например, *Клиент*). Следует создать все дорожки (например, *Бухгалтерия*, *Кассир*) (см. рис. 4.9.).



**Рис. 4.9.** Окно диаграммы деятельности.

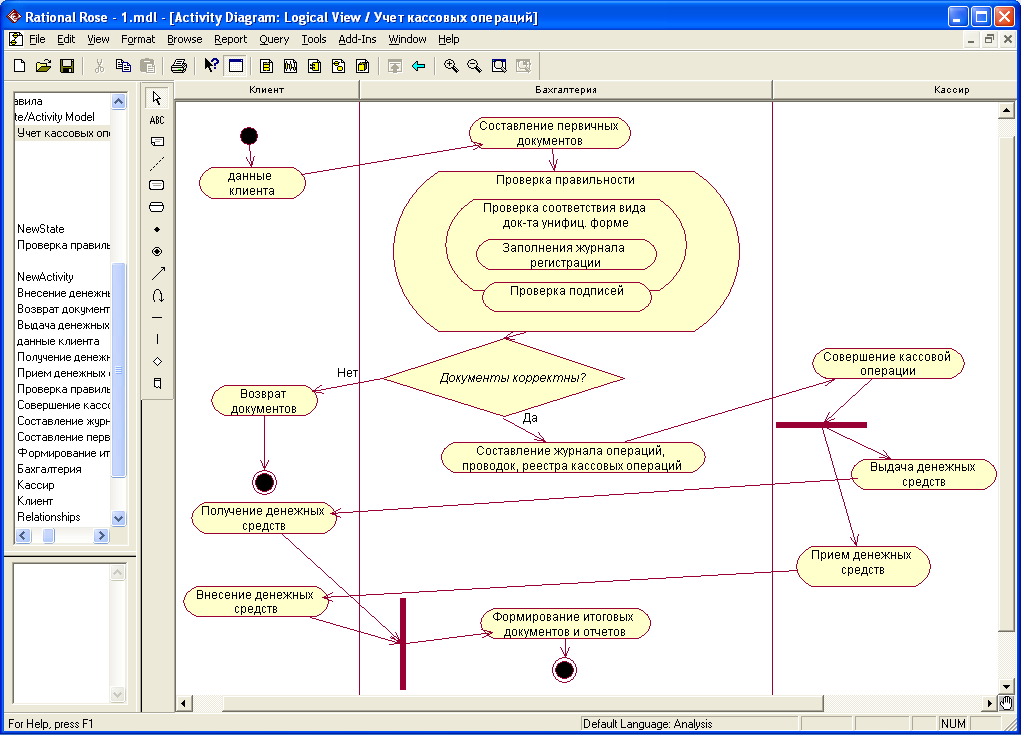
Следует отметить, что диаграмма деятельности должна начинаться с блока 3 и заканчиваться блоком 4 (элементы берутся из специальной панели инструментов).

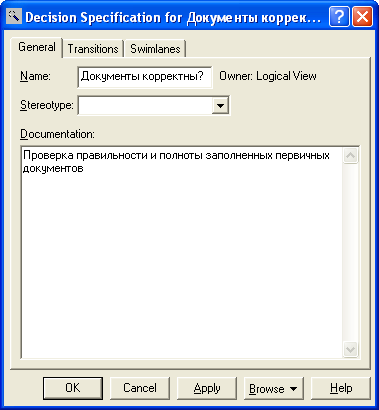
Для добавления в рабочую область *деятельности* нужно на специальной панели инструментов выбрать кнопку 2 и щелкнуть мышью в рабочую область диаграммы. Затем следует ввести название блока (например, *данные клиента*). Для изменения реквизитов данного блока следует щелкнуть по нему правой кнопкой мыши и выбрать режим **Open Specification**. Откроется окно ***Activity Specification*** (см. рис. 4.10). В данном окне на вкладке **General** вводится текстовая документация.



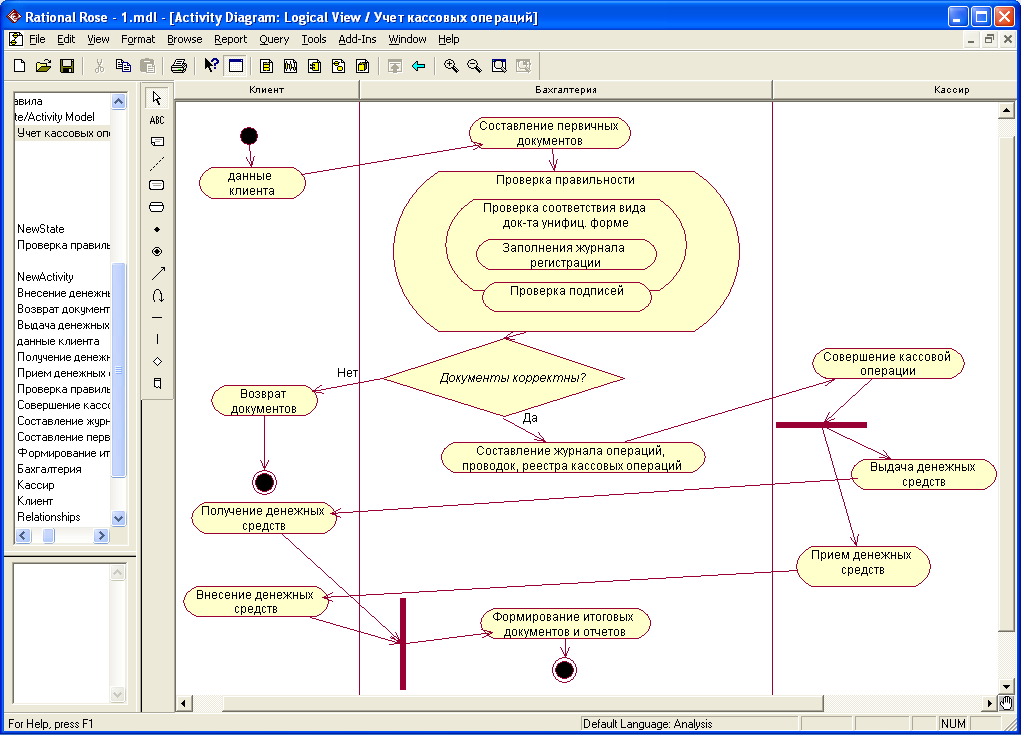
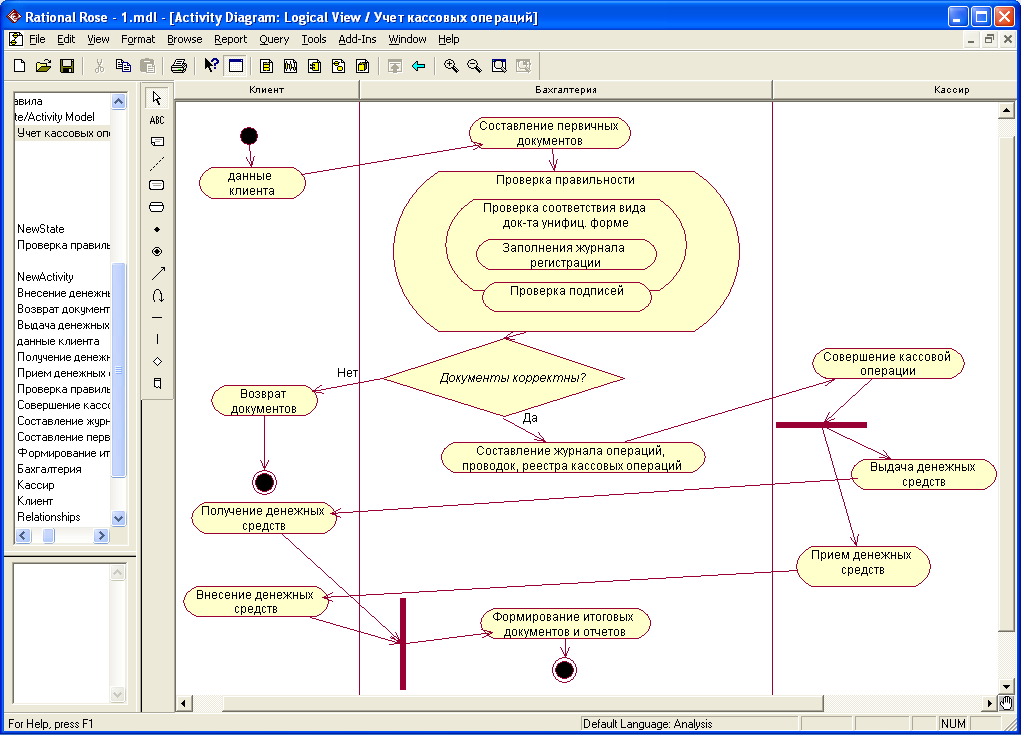
**Рис. 4.10.** Основное окно спецификации деятельности.

Чтобы обычную *деятельность* превратить в *композит*, следует при добавлении новой *деятельности* поместить ее внутри границы той *деятельности*, которую необходимо сделать *композитной*. В результате внутри исходной *деятельности* появится новая вложенная *деятельность*, которая при перемещении композита в рабочей области всегда будет находиться внутри своего композита (см. на рис. 4.9 элемент *Проверка правильности*).

Для добавления элемента *принятия решения* следует на специальной панели инструментов выбрать кнопку  и щелкнуть мышью в рабочую область диаграммы. Затем следует ввести название блока (например, *Документы корректны?*). Для изменения реквизитов данного блока следует щелкнуть по нему правой кнопкой мыши и выбрать режим **Open Specification**. Откроется окно ***Decision Specification*** (см. рис. 4.11). В данном окне на вкладке **General** вводится текстовая документация.



**Рис. 4.11.** Основное окно спецификации блока принятия решения.

Для добавления элемента с вертикальной (горизонтальной) синхронизацией (одновременное выполнение действий) следует на специальной панели инструментов выбрать кнопку  или  и щелкнуть мышью в рабочую область диаграммы. Затем следует соединить необходимые действия с этим элементом диаграммы (см. рис. 4.9).

Размещение блоков диаграммы следует выполнять в соответствующих дорожках диаграммы.

Дальнейшая работа с моделью зависит от целей выполнения проекта. Если проект не предполагает программную реализацию, то можно ограничиться формированием проектной документации. С этой целью следует выполнить операцию главного меню: ***Report*** 🡪 ***So DA Report***, в результате чего будет открыто диалоговое окно свойств по выбору шаблонов генерации отчета. После выбора шаблонов будет автоматически сгенерирован отчет о разрабатываемой модели в формате MicroSoft Word с использованием специального средства IBM Rational So DA, если оно доступно в системе после инсталляции IBM Rational Rose 2007.

# 5. Задания на проектирование системы

Для проектирования экономической ИС по анализу работы условной фирмы требуется описать бизнес-процессы деятельности этой фирмы при помощи пакета BPwin, построить логическую и физическую модели данных при помощи пакета ERwin, а также описать бизнес-процессы деятельности этой фирмы при помощи пакета IBM Rational Rose 2007.

Список тем для проектирования:

**1**. Учет расчета прибыли

**2**. Учет материальных ценностей и их оценка

**3**. Учет расчетов нематериальных активов

**4**. Учет движения арендованных основных средств

**5**. Учет реализации продукции

**6**. Учет расчетов финансовых вложений

**7**. Учет расчетов с бюджетом и внебюджетными фондами

**8**. Учет реализации готовой продукции

**9**. Учет расчетов с учредителями

**10**. Учет расчетов затрат на производство

**11**. Учет ведения валютных операций

**12**. Учет расчетов с органами социального страхования и обеспечения

**13**. Учет ведения расчетов по векселям

**14**. Учет реализации продукции и расчет себестоимости

**15**. Учет расчетов по НДС

**16**. Учет движения МБП

**17**. Учет ведения кассовых операций

**18**. Учет ведения расчетов с поставщиками и подрядчиками

**19**. Учет движения материальных ценностей

**20**. Учет расчетов с бюджетом по налогам

**21**. Учет расчетов по заработной плате

**22**. Учет инвентаризация материальных ценностей

**23**. Учет расчетов с подотчетными лицами

**24**. Учет расчетов по кредитам банков

**25**. Учет движения основных средств

**26**. Учет движения материалов, отпускаемых на производство

**27**. Учет расчетов с разными дебиторами и кредиторами

**28**. Учет ведения долгосрочной аренды основных средств

**Требования к построению модели.**

BPwin:

* в контекстной диаграмме входных (и выходных) документов должно быть не менее четырех;
* диаграмм декомпозиции должно быть не менее пяти, с учетом вложенности;
* каждая диаграмма декомпозиции должна состоять не менее чем из трех элементов «работа»;
* диаграмм потоков данных должно быть не менее двух;
* каждая диаграмма потоков данных должна состоять не менее чем из трех элементов «работа».

ERwin:

* полей в таблицах должно быть не менее семи;
* таблицы должны включать первичные ключи (связанные – еще и внешние ключи);
* связи таблиц должны быть как идентифицирующие так и не идентифицирующие;

Rational Rose:

* в каждой диаграмме должно быть не менее четырех входных (и выходных) документов;
* в диаграмме деятельности должно быть не менее трех отделов;
* диаграмм должно быть не менее пяти, с учетом иерархии;
* каждая диаграмма должна состоять не менее чем из трех элементов «класс».

**Структура отчета следующая:**

* титульный лист (тема: «Разработка информационной системы по … CASE-средствами BPwin, ERwin, Rational Rose),
* введение (преимущества информационного и визуального моделирования, что будет выполнено в работе, какими средствами),
* постановка задачи (требования к диаграммам, основные документы по задаче, требования к техническому заданию, требование построить не менее трех отчетов, требование добавить стоимостной анализ),
* исходные и выходные данные (формы документов),
* описание методологии функциональной модели IDEF0, IDEF3, DFD (BPwin), модели данных (ERwin), визуального моделирования (Rational Rose),
* распечатка (функциональная модель, бизнес-процессы, диаграмма потоков данных, отчеты по BPwin с фамилией автора в каждом, логическая и физическая модели данных Erwin, диаграммы деятельности и классов),
* описании схем декомпозиции (BPwin) по каждому блоку в диаграмме (вход, выход, управляющее воздействие, нормативы),
* заключение (что было выполнено, какими средствами, что было создано в работе с помощью CASE-средств BPwin, ERwin, Rational Rose, преимущества управления жизненным циклом с помощью ИС).

Заключение

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

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

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

All Fusion ERwin Data Modeler (ERwin) позволяет наглядно отображать сложные структуры данных. Удобная в использовании графическая среда ERwin упрощает разработку базы данных и автоматизирует множество трудоёмких задач, уменьшая сроки создания высококачественных и высокопроизводительных транзакционных баз данных и хранилищ данных. Данное решение улучшает коммуникацию организации, обеспечивая совместную работу администраторов и разработчиков баз данных, многократное использование модели, а также наглядное представление комплексных активов данных в удобном для понимания и обслуживания формате.

Следует отметить, что построение как логической, так и физической модели данных средствами Erwin оказывает неоспоримое влияние на все этапы жизненного цикла информационной системы. На любом этапе жизненного цикла как специалист-профессионал, так и неспециалист может оценить соответствие созданной модели тем требованиям, которые были поставлены перед разработчиками.

Моделирование бизнес процессов в Rational Rose выполняется за счет применения различных аспектов. Каждый из этих аспектов концентрирует внимание на определенных характеристиках и возможностях процессов.

К таким аспектам относятся:

* Вариант использования (Use Сase). Этот аспект дает возможность понять, каким образом действуют участники процесса и за счет этого определить их взаимодействие и влияние на процесс. Для построения моделей процесса в рамках данного аспекта применяются Use Сase диаграммы, диаграммы последовательностей, диаграммы совместной работы и диаграммы действий.
* Логический аспект. С помощью этого аспекта можно определить функциональные требования процессов. Он задает логическую взаимосвязь между классами элементов процессов. Для построения моделей применяются диаграммы классов и диаграммы состояний.
* Составляющие элементы. Этот аспект обращает внимание на состав элементов процесса и их распределение при создании информационной системы. Модели в этом аспекте строятся с помощью диаграммы компонентов. Она содержит информацию об элементах процесса и программном обеспечении.
* Ввод в действие. Этот аспект показывает схему процесса в привязке к аппаратному обеспечению информационной системы. Для построения моделей применяется только одна диаграмма – диаграмма топологии.

Очевидно также, что эффект от использования средства IBM Rational Rose проявляется при разработке масштабных проектов в составе команды или проектной группы. Однако ситуация покажется не столь тривиальной, когда станет необходимо выполнить проект с несколькими десятками вариантов использования и сотней классов. Именно для подобных проектов явно выявляется преимущество использования средства IBM Rational Rose для документирования и реализации соответствующих моделей.

# Список литературы

1. Калянов Г.Н. Моделирование, анализ, реорганизация и автоматизация бизнес-процессов: Учебное пособие / под ред. Калянов Г.Н. - М.: Финансы и Статистика, 2006.

2. Вендров А.М. Практикум по проектированию программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2002.

3. Черемных С.В., Семенов И.О., Ручкин В.С. Структурный анализ систем: IDEF-технологии. - М.: Финансы и статистика, 2001.

4. Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем. Учебник. – М.: Финансы и статистика, 2001.

5. Маклаков С.В. BPwin, ERwin CASE-средства разработки информационных технологий. – М.: Диалог-МИФИ, 2000.

6. Васильева Л.Н. Моделирование микроэкономических процессов и систем. - М.: Кнорус, 2012.

7. Светлов Н.М. Информационные технологии управления проектами: Учебное пособие. 2-е изд., перераб. и доп. - М.: ИНФРА-М, 2012.

8. Дж. Лодон, К. Лодон «Управление информационными системами», изд. Питер, 7-ое издание, 2005.

9. Исаев Г.И. Проектирование информационных систем, Учебное пособие, ОМЕГА-Л, 2013.

10. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose: Учебное пособие / А.В. Леоненков. – М.: Интернет-Университет Информационных технологий; БИНОМ, 2006.

11. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – 2-е изд., перераб. и доп. - М.: Финансы и статистика, 2005.

12. Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. Проектирование информационных систем. Интернет-университет информационных технологий - ИНТУИТ.ру, 2005.

13. Грехем И. Объектно-ориентированные методы. Принципы и практика. Пер. с англ. – М.: Вильямс, 2004.

14. Тельнов Ю.Ф. Реинжиниринг бизнес-процессов. Компонентная методология. – 2-е изд., перераб. и доп. - М.: Финансы и статистика, 2004.

15. Елиферов В.Г., Репин В.В. Бизнес-процессы: регламентация и управление. - М.: ИНФРА-М, 2004.

16. Маклаков С.В. Моделирование бизнес-процессов с AII Fusion Process Modeler. – М.: Диалог-МИФИ, 2003.

17. Нейбург Э. Д., Максимчук Р.А. Проектирование баз данных с помощью UML. - М.: Издательский дом «Вильямс», 2002.

18. Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. Проектирование экономических информационных систем. - М.: Финансы и статистика, 2002.

19. Морозова В.И., Врублевский К.Э. Методические указания к выполнению лабораторных работ по разработке информационных систем с использованием CASE-средств BPwin и ERwin – М.: МИИТ, 2004.

20. Морозова В.И., Врублевский К.Э. Визуальное моделирование информационных систем: Методические указания к выполнению лабораторных работ. – М.: МИИТ, 2009.

21. Белов В.В., Чистякова В.И. Проектирование информационных систем. Учебник. Изд. Академия, 2013.

22. Емельянова Н.З., Попов И.И., Партыка Т.Л. Проектирование информационных систем. Изд. Форум, 2014.

23. Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. Управление внедрением информационных систем – М.: Интернет-университет информационных технологий, 2010.

23. Зараменских Е.П. Управление жизненным циклом информационных систем. - Новосибирск: НГТУ, 2014.

24. http://alice.stup.ac.ru/case/caseinfo/bpwin/part1.html

25. http://www.info-system.ru/designing/methodology/sadt/sadt.html

Св.план 2015г., поз. 263

МОРОЗОВА Вера Ивановна

ВРУБЛЕВСКИЙ Константин Эдуардович

Управление жизненным циклом информационных систем

УЧЕБНОЕ ПОСОБИЕ

Подписано в печать Формат *60х84/16*

Усл.-печ. л. *4 Заказ №* Тираж  *100 экз.*

1. \* Интегрированная БД – совокупность взаимосвязанных, хранящихся вместе данных, используемых для одного или нескольких приложений. [↑](#footnote-ref-1)
2. Инициализация – проверка целостности и работоспособности проекта (системы) [↑](#footnote-ref-2)
3. Workflow – методики и программное обеспечение для описания и проектирования бизнес-процессов. [↑](#footnote-ref-3)