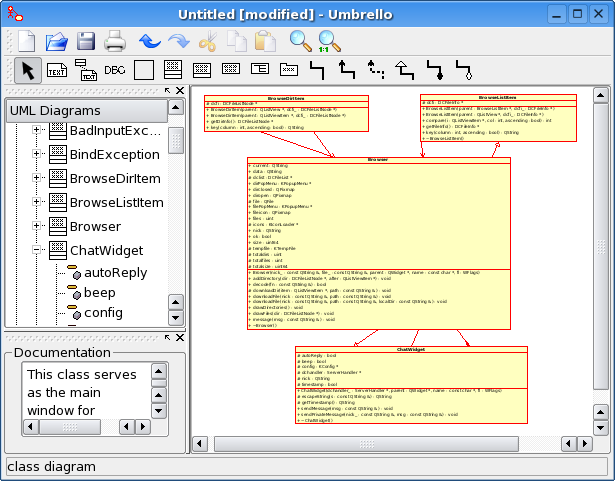
# ЛЕКЦИЯ 3. CASE-СРЕДСТВА И МЕТОДОЛОГИИ ПРОЕКТИРОВАНИЯ

3.1. CASE-СРЕДСТВА

CASE (англ. computer-aided software engineering) — набор инструментов и методов программной инженерии для проектирования программного обеспечения, который помогает обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов. Также под CASE понимают совокупность методов и средств проектирования информационных систем с использованием CASE-инструментов.

Средства автоматизации разработки программ (CASE-средства) — инструменты автоматизации процессов проектирования и разработки программного обеспечения для системного аналитика, разработчика ПО и программиста. Первоначально под CASE-средствами понимались только инструменты для упрощения наиболее трудоёмких процессов анализа и проектирования, но с приходом стандарта ISO/IEC 14102 CASE-средства стали определять, как программные средства для поддержки процессов жизненного цикла ПО.



Основной целью CASE-технологии является разграничение процесса проектирования программных продуктов от процесса кодирования и последующих этапов разработки, максимально автоматизировать процесс разработки. Для выполнения поставленной цели CASE-технологии используют два принципиально разных подхода к проектированию: структурный и объектно-ориентированный.

Структурный подход предполагает декомпозицию (разделение) поставленной задачи на функции, которые необходимо автоматизировать. В свою очередь, функции также разбиваются на подфункции, задачи, процедуры. В результате получается упорядоченная иерархия функций и передаваемой информацией между функциями.

Структурный подход подразумевает использование определенных общепринятых методологий при моделировании различных информационных систем:

* SADT (structured analysis and design technique);
* DFD (data flow diagrams);
* ERD (entity-relationship diagrams).

Существует три основных типа моделей, используемых при структурном подходе: функциональные, информационные и структурные.

Основным инструментом объектно-ориентированного подхода является язык UML — унифицированный язык моделирования, который предназначен для визуализации и документирования объектно-ориентированных систем с ориентацией их на разработку программного обеспечения. Данный язык включает в себя систему различных диаграмм, на основании которых может быть построено представление о проектируемой системе.

3.1.1 Классификация CASE-средств

В функции CASE входят средства анализа, проектирования и программирования программных средств, проектирования интерфейсов, документирования и производства структурированного кода на каком-либо языке программирования.

CASE-инструменты классифицируются по типам и категориям.

Классификация по типам отражает функциональную ориентацию средств на те или иные процессы жизненного цикла разработки программного обеспечения, и, в основном, совпадают с компонентным составом крупных интегрированных CASE-систем, и включает следующие типы:

* средства анализа — предназначены для построения и анализа модели предметной области;
* средства проектирования баз данных;
* средства разработки приложений;
* средства реинжиниринга процессов;
* средства планирования и управления проектом;
* средства тестирования;
* средства документирования.

Классификация по категориям определяет степень интегрированности по выполняемым функциям и включают — отдельные локальные средства, решающие небольшие автономные задачи, набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла и полностью интегрированных средств, охватывающий весь жизненный цикл информационной системы и связанных общим репозиторием.

Типичными CASE-инструментами являются:

* инструменты управления конфигурацией;
* инструменты моделирования данных;
* инструменты анализа и проектирования;
* инструменты преобразования моделей;
* инструменты редактирования программного кода;
* инструменты рефакторинга кода;
* генераторы кода;
* инструменты для построения UML-диаграмм.

3.1.2. Программные продукты CASE-средств

**erwin Modeling Suite:**

1. erwin Modeling Suite (ранее: ERwin Modeling Suite) - интегрированный комплекс CASE-средств, обеспечивающий все потребности компаний-разработчиков ПО. Данный пакет служит для проектирования и анализа баз данных, бизнес-процессов и информационных систем и включает продукты: erwin Business Process (ранее: BPwin), ERwin Data Modeler (ранее: ERwin), erwin Data Model Validator (ERwin Examiner), erwin Model Manager (ранее: ModelMart), erwin Component Modeler (Paradigm Plus), использование которых позволяет сократить расходы и повысить продуктивность процесса разработки.
2. erwin Business Process (ранее: BPwin) - ведущий инструмент для моделирования бизнес-процессов. Позволяет оптимизировать деятельность организации и проверить ее на соответствие стандартам ISO9000, спроектировать оргструктуру, снизить издержки, исключить ненужные операции и повысить эффективность. Являясь стандартом де-факто, BPwin поддерживает сразу три нотации моделирования: IDEF0 (федеральный стандарт США), IDEF3 и DFD.
3. erwin Data Modeler (ранее: ERwin) - лидер среди средств моделирования баз данных и хранилищ данных. Позволяет проектировать, документировать и сопровождать базы данных различных типов. Поддерживая прямое и обратное проектирование для 20 типов СУБД, erwin повышает качество разрабатываемой БД, производительность труда и скорость разработки.
4. erwin Data Model Validator (ранее: ERwin Examiner) - инструмент для проверки структуры баз данных и создаваемых в erwin моделей, позволяющий выявлять недочеты и ошибки проектирования. erwin Examiner дополняет функциональность erwin, автоматизируя трудоемкую задачу поиска и исправления ошибок, одновременно повышая квалификацию проектировщиков баз данных благодаря встроенной системе обучения.

**IBM Rational Software Modeler:**

1. IBM Rational Software Modeler - визуальный инструмент моделирования и проектирования, который позволяет пользователям четко документировать эти различные представления системы и доводить их до сведения заинтересованных лиц.
2. IBM Rational Software Architect - новое средство проектирования и разработки, которое поможет архитекторам более эффективно определять и поддерживать все аспекты архитектурных решений при разработке программного обеспечения. Является развитием программы Rational Rose (средство моделирования).

**Oracle:**

1. Oracle Designer (входит в Oracle9i Developer Suite) - высоко функциональное средство проектирования программных систем и баз данных, реализующее технологию CASE и собственную методологию Oracle - "CDM". Позволяет команде разработчиков полностью провести проект, начиная от анализа бизнес-процессов через моделирование к генерации кода и получению прототипа, а в дальнейшем и окончательного продукта. Сложное CASE-средство, имеет смысл использовать при ориентации на линейку продуктов Oracle.

**Microsoft:**

1. Microsoft Visio — векторный графический редактор, редактор диаграмм и блок-схем для Windows. Шаблоны UML используются в Visio для проектирования новых схем или документирования существующих. Visio содержит шаблоны нескольких типов схем версий UML 2.0:

* статическая структура;
* нотация баз данных UML;
* схема последовательностей;
* схема вариантов использования;
* схема деятельности;
* схема конечного автомата UML.

3.2. МЕТОДОЛОГИИ ПРОЕКТИРОВАНИЯ

3.2.1. Модели AS-IS и TO-BE

Обычно сначала строится модель существующей организации работы - AS-IS (как есть). Анализ функциональной модели позволяет определить:

* наиболее слабые места преимущества новых бизнес-процессов;
* глубину изменений, которым подвергнется существующая структура организации бизнеса.

Признаками неэффективной работы деятельности могут быть:

* бесполезные, неуправляемые и дублирующиеся работы;
* неэффективный документооборот;
* отсутствие обратных связей по управлению;
* отсутствие обратных связей по входу;

Найденные в модели AS-IS недостатки можно исправить при создании модели TO-BE (как будет) - модели новой организации бизнес-процессов. Модель TO-BE нужна для анализа альтернативных путей выполнения работы и документирования того, как компания будет делать бизнес в будущем.

При создании модели AS-IS разработчиком может быть допущена распространенная ошибка - создание идеализированной модели (например, модель, созданная на основе знаний руководителя, а не конкретного исполнителя работ). Такая модель несет ложную, искаженную информацию и называется SHOULD-BE (как должно бы быть).

Технология проектирования ИС подразумевает сначала создание модели AS-IS, затем ее анализ и улучшение бизнес-процессов, т.е. создание модели TO-BE. И только на основе модели TO-BE строится модель данных, прототип и затем окончательный вариант ИС.

Иногда текущая AS-IS и будущая TO-BE модели различаются очень сильно, так что переход от начального состояния к конечному состоянию становится неочевидным. В этом случае необходима третья модель, описывающая процесс перехода от начального состояния системы к конечному.

3.2.2. Методология функционального моделирования IDEF0

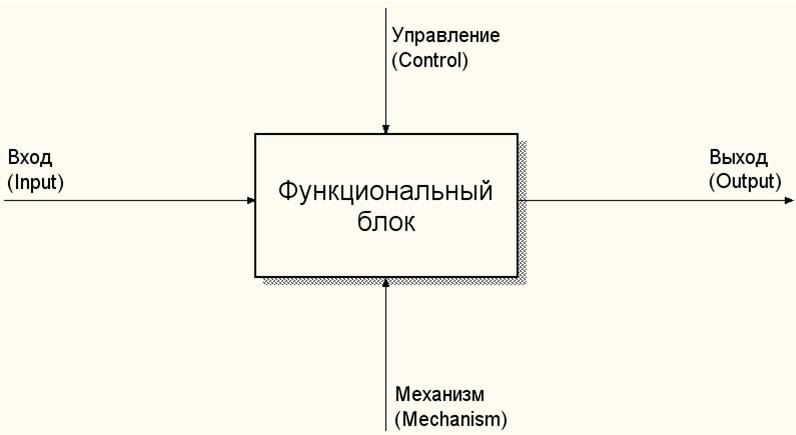
IDEF0  — методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временна́я последовательность (поток работ).

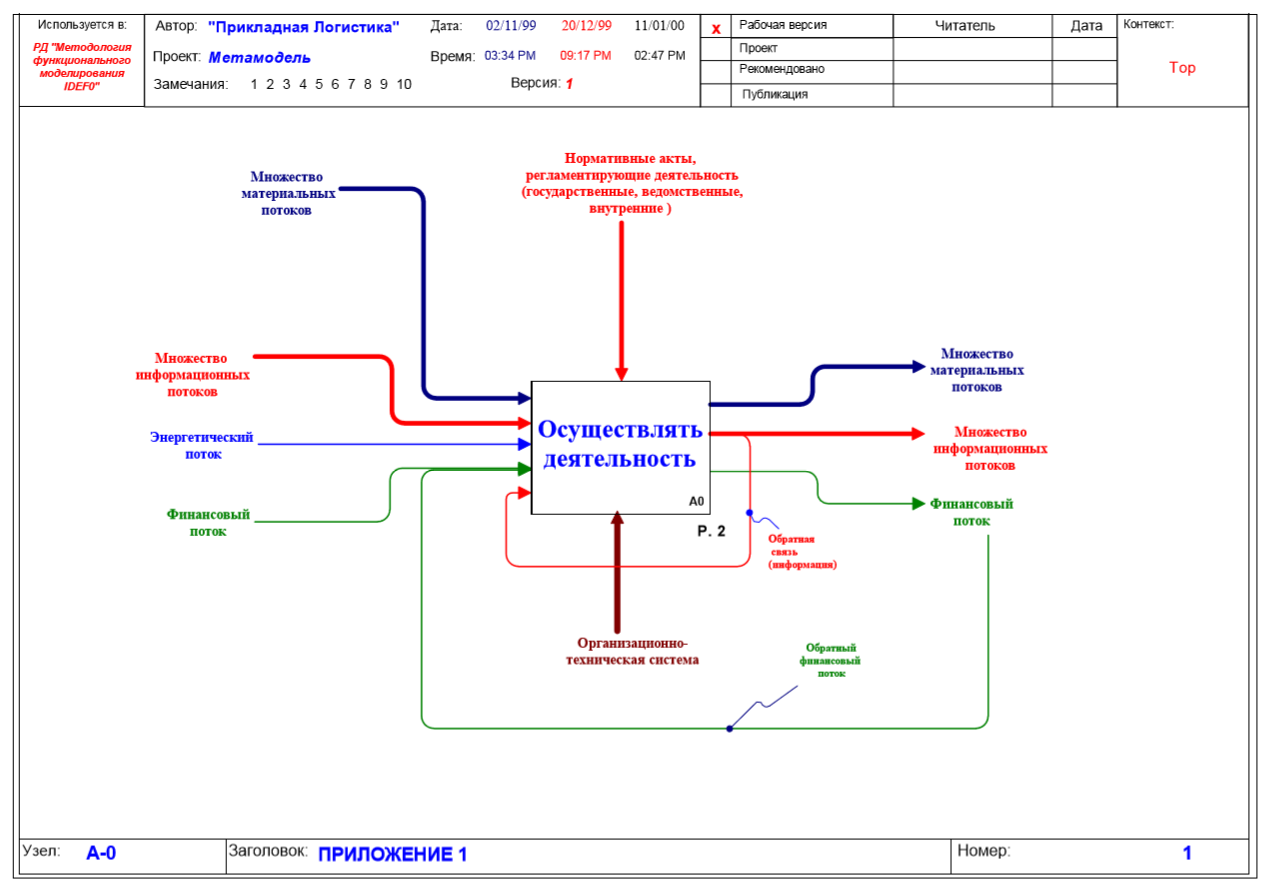
Стандарт IDEF0 представляет организацию как набор модулей, здесь существует правило — наиболее важная функция находится в верхнем левом углу, кроме того есть правило стороны:

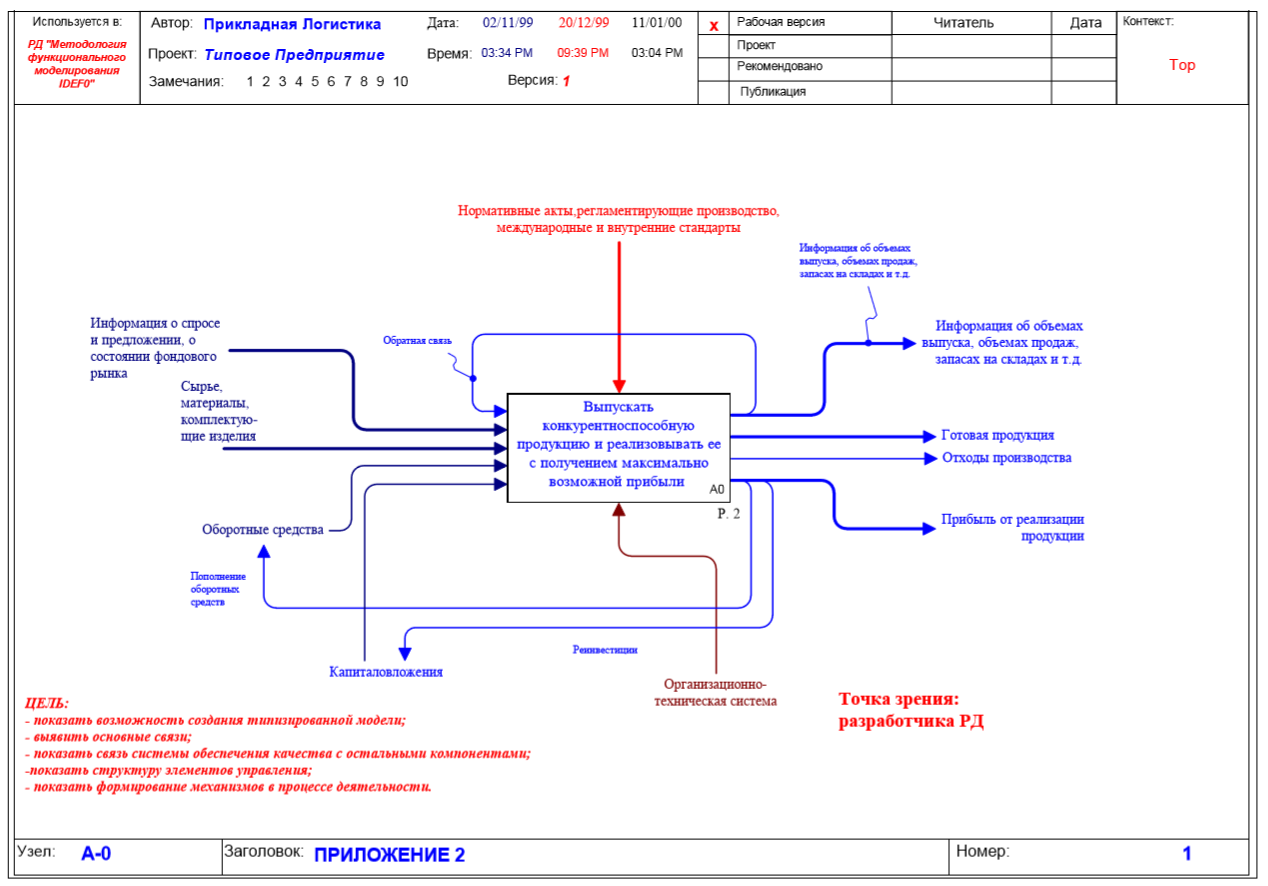
* стрелка входа приходит всегда в левую кромку активности,
* стрелка управления — в верхнюю кромку,
* стрелка механизма — нижняя кромка,
* стрелка выхода — правая кромка.

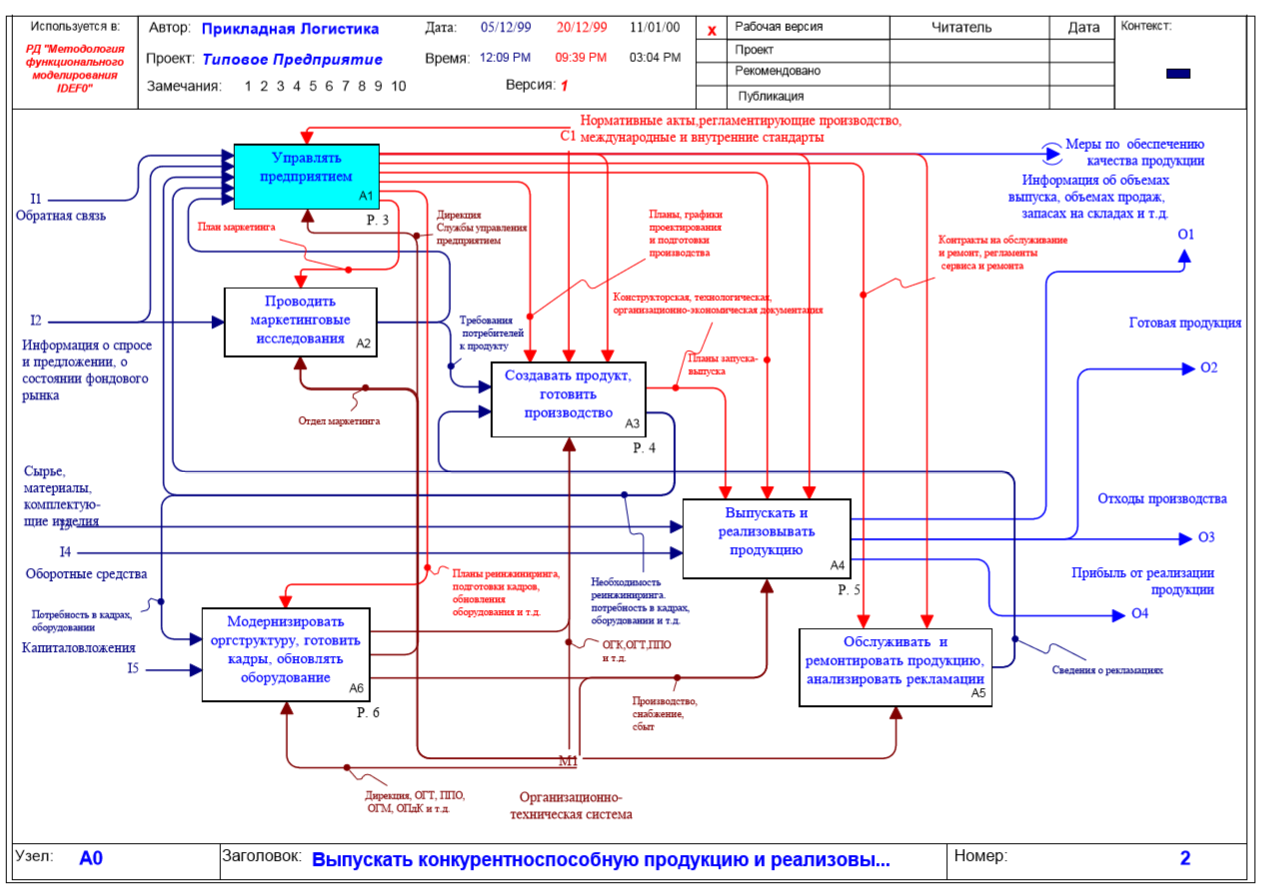
Описание выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

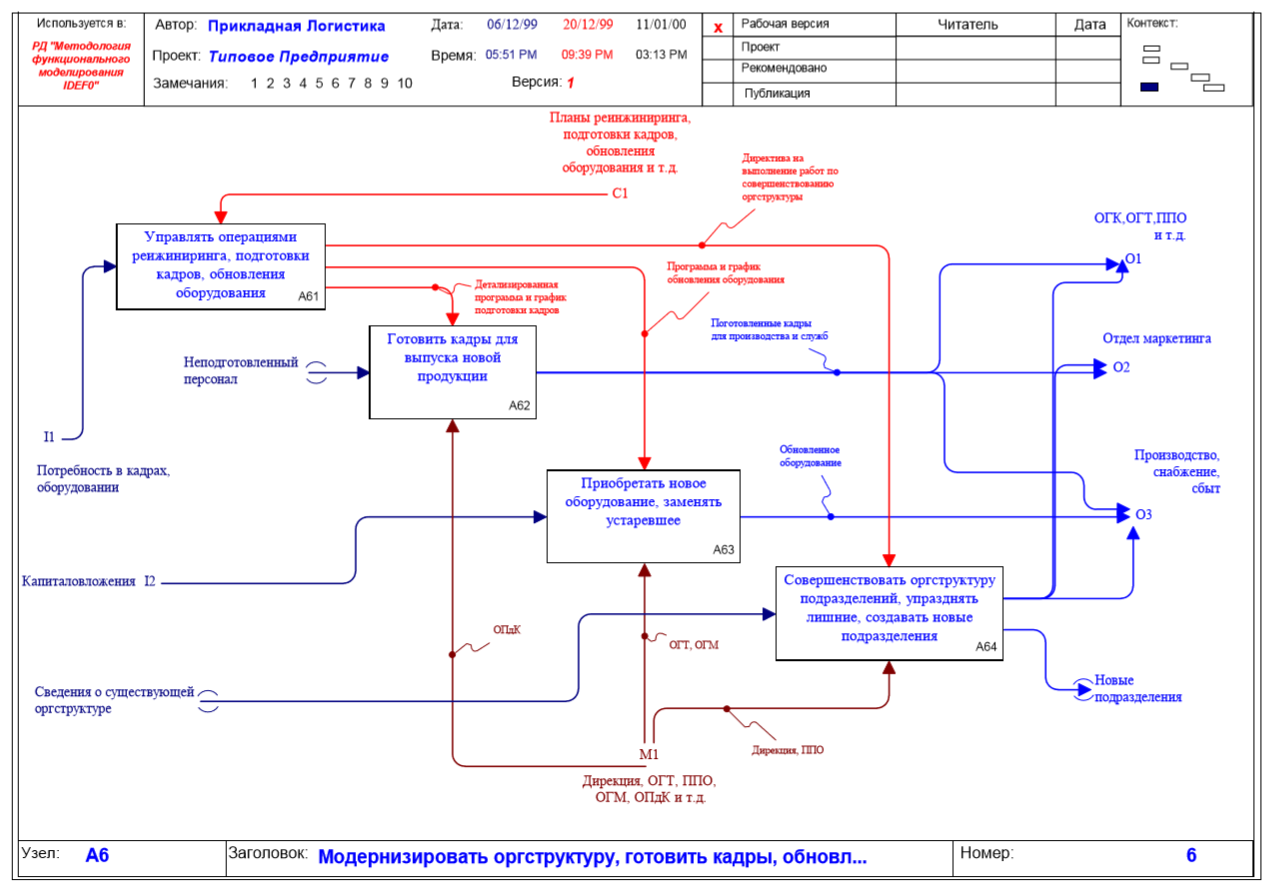
Также отображаются все сигналы управления, которые на DFD (диаграмме потоков данных) не отображались. Данная модель используется при организации бизнес-процессов и проектов, основанных на моделировании всех процессов: как административных, так и организационных.

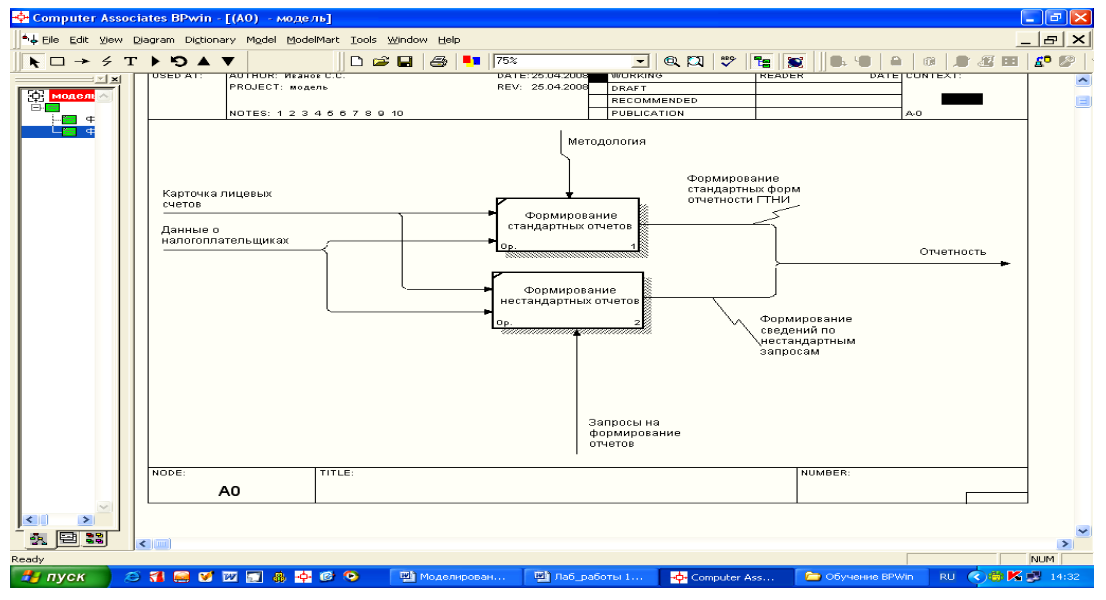












3.2.3. Методология моделирования IDEF3

IDEF3 (англ. Integrated DEFinition for Process Description Capture Method) — методология моделирования и стандарт документирования процессов, происходящих в системе. Метод документирования технологических процессов представляет собой механизм документирования и сбора информации о процессах. IDEF3 показывает причинно-следственные связи между ситуациями и событиями в понятной эксперту форме, используя структурный метод выражения знаний о том, как функционирует система, процесс или предприятие.

IDEF3 широко применяется при разработке информационных систем. При этом используется инструмент визуального моделирования бизнес-процессов. Система описывается как упорядоченная последовательность событий с одновременным описанием объектов, имеющих отношение к моделируемому процессу.

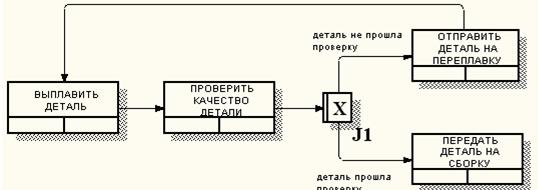
IDEF3 состоит из двух методов. Process Flow Description (PFD) — Описание технологических процессов, с указанием того, что происходит на каждом этапе технологического процесса. Object State Transition Description (OSTD) — описание переходов состояний объектов, с указанием того, какие существуют промежуточные состояния у объектов в моделируемой системе.

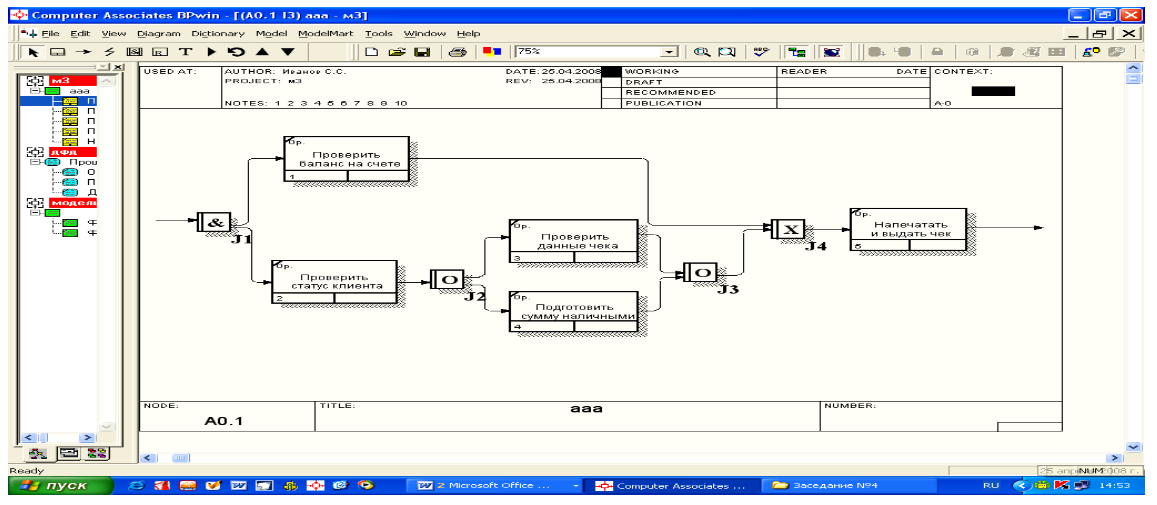
Основу методологии IDEF3 составляет графический язык описания процессов. Модель в нотации IDEF3 может содержать два типа диаграмм:

* диаграмму Описания Последовательности Этапов Процесса (Process Flow Description Diagrams, PFDD);
* диаграмму Сети Трансформаций Состояния Объекта (Object State Transition Network, OSTN).

Диаграмма IDEF3 Process Flow Description может состоять из 7 основных описательных блоков:

* работы (boxes, activities);
* стрелки или связи (arrows, links);
* перекрёстки (junctions);
* объекты ссылок;
* Unit of Behavior;
* Decomposition;
* Elaboration.





3.2.4. Методика диаграммы потоков данных DFD

DFD — общепринятое сокращение от англ. data flow diagrams — диаграммы потоков данных. Так называется методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

Диаграмма потоков данных (data flow diagram, DFD) — один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML. Несмотря на имеющее место в современных условиях смещение акцентов от структурного к объектно-ориентированному подходу к анализу и проектированию систем, «старинные» структурные нотации по-прежнему широко и эффективно используются как в бизнес-анализе, так и в анализе информационных систем.

Исторически сложилось так, что для описания диаграмм DFD используются две нотации — Йордана (Yourdon) и Гейна-Сарсона (Gane-Sarson), отличающиеся синтаксисом.

Информационная система принимает извне потоки данных. Для обозначения элементов среды функционирования системы используется понятие внешней сущности. Внутри системы существуют процессы преобразования информации, порождающие новые потоки данных. Потоки данных могут поступать на вход к другим процессам, помещаться (и извлекаться) в накопители данных, передаваться к внешним сущностям.

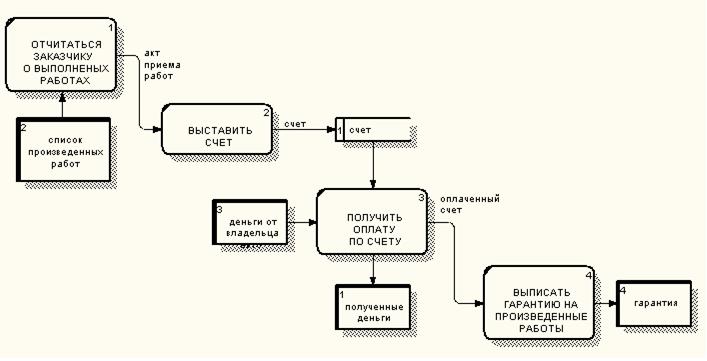
Модель DFD, как и большинство других структурных моделей — иерархическая модель. Каждый процесс может быть подвергнут декомпозиции, то есть разбиению на структурные составляющие, отношения между которыми в той же нотации могут быть показаны на отдельной диаграмме. Когда достигнута требуемая глубина декомпозиции — процесс нижнего уровня сопровождается мини-спецификацией (текстовым описанием).

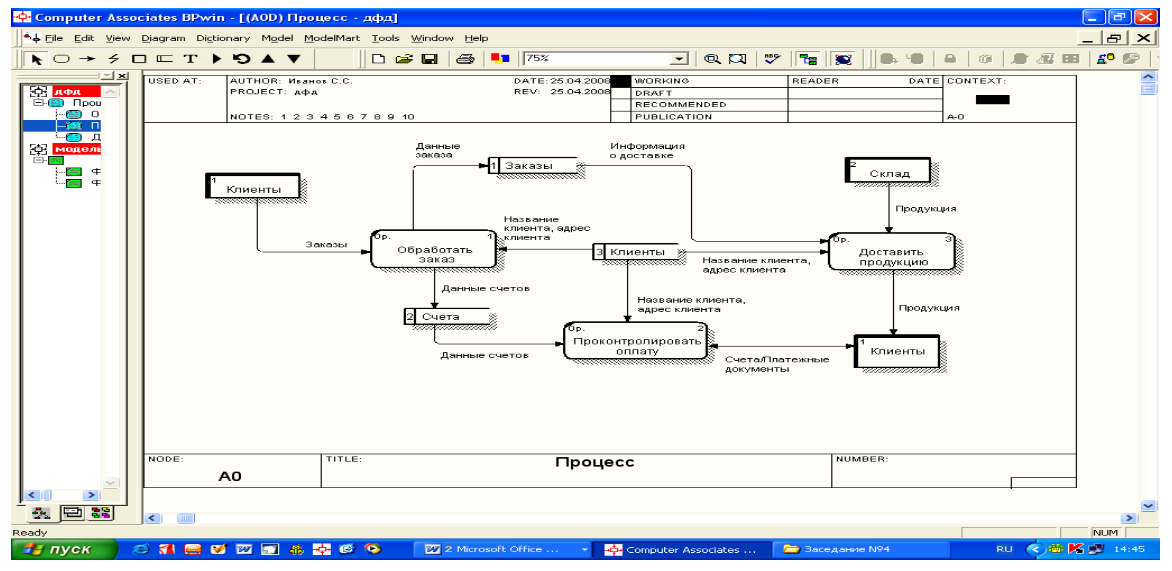
Кроме того, нотация DFD поддерживает понятие подсистемы — структурного компонента разрабатываемой системы.

Нотация DFD — удобное средство для формирования контекстной диаграммы, то есть диаграммы, показывающей разрабатываемую АИС в коммуникации с внешней средой. Это — диаграмма верхнего уровня в иерархии диаграмм DFD. Её назначение — ограничить рамки системы, определить, где заканчивается разрабатываемая система и начинается среда. Другие нотации, часто используемые при формировании контекстной диаграммы — диаграмма SADT, Диаграмма вариантов использования.

External entity (внешняя сущность) представляет собой материальный предмет или физическое лицо, являющиеся источником или приёмником информации, например, заказчики, персонал, поставщики, клиенты, склад. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой ИС. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь диаграммы анализируемой ИС, если это необходимо, или, наоборот, часть процессов ИС может быть вынесена за пределы диаграммы и представлена как внешняя сущность. Внешняя сущность обозначается квадратом, расположенным как бы «над» диаграммой и бросающим на неё тень, для того, чтобы можно было выделить этот символ среди других обозначений.

Process (системы и подсистемы) — при построении модели сложной ИС она может быть представлена в самом общем виде на так называемой контекстной диаграмме в виде одной системы как единого целого, либо может быть декомпозирована на ряд подсистем. Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т. д.

****

****

3.2.5. Rational Unified Process

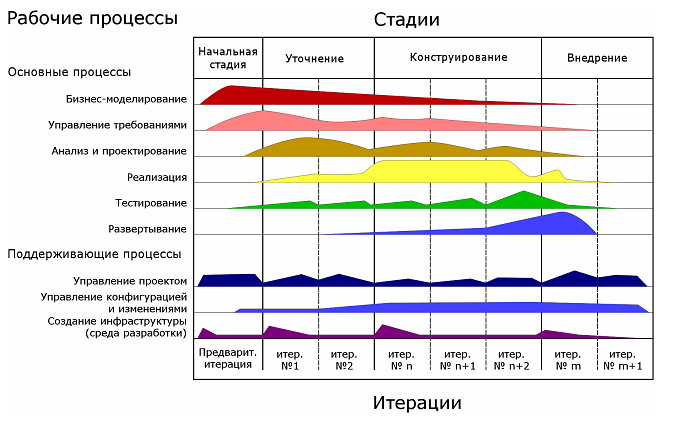
Rational Unified Process (RUP) — методология разработки программного обеспечения, созданная компанией Rational Software.

В основе RUP лежат следующие принципы:

1. Ранняя идентификация и непрерывное (до окончания проекта) устранение основных рисков.
2. Концентрация на выполнении требований заказчиков к исполняемой программе (анализ и построение модели прецедентов (вариантов использования)).
3. Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки.
4. Компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта.
5. Постоянное обеспечение качества на всех этапах разработки проекта (продукта).
6. Работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам.

RUP использует итеративную модель разработки. В конце каждой итерации (в идеале продолжающейся от 2 до 6 недель) проектная команда должна достичь запланированных на данную итерацию целей, создать или доработать проектные артефакты и получить промежуточную, но функциональную версию конечного продукта. Итеративная разработка позволяет быстро реагировать на меняющиеся требования, обнаруживать и устранять риски на ранних стадиях проекта, а также эффективно контролировать качество создаваемого продукта. Первые идеи итеративной модели разработки были заложены в "спиральной модели".

Полный жизненный цикл разработки продукта состоит из четырёх фаз, каждая из которых включает в себя одну или несколько итераций:



В фазе начальной стадии:

1. Формируются видение и границы проекта.
2. Создается экономическое обоснование (business case).
3. Определяются основные требования, ограничения и ключевая функциональность продукта.
4. Создается базовая версия модели прецедентов.
5. Оцениваются риски.

При завершении начальной фазы оценивается достижение этапа жизненного цикла цели (англ. Lifecycle Objective Milestone), которое предполагает соглашение заинтересованных сторон о продолжении проекта.

В фазе «Уточнение» производится анализ предметной области и построение исполняемой архитектуры. Это включает в себя:

1. Документирование требований (включая детальное описание для большинства прецедентов).
2. Спроектированную, реализованную и оттестированную исполняемую архитектуру.
3. Обновленное экономическое обоснование и более точные оценки сроков и стоимости.
4. Сниженные основные риски.

Успешное выполнение фазы уточнения означает достижение этапа жизненного цикла архитектуры (англ. Lifecycle Architecture Milestone).

В фазе «Построение» происходит реализация большей части функциональности продукта. Фаза Построение завершается первым внешним релизом системы и вехой начальной функциональной готовности (Initial Operational Capability).

В фазе «Внедрение» создается финальная версия продукта и передается от разработчика к заказчику. Это включает в себя программу бета-тестирования, обучение пользователей, а также определение качества продукта. В случае, если качество не соответствует ожиданиям пользователей или критериям, установленным в фазе Начало, фаза Внедрение повторяется снова. Выполнение всех целей означает достижение вехи готового продукта (Product Release) и завершение полного цикла разработки.

3.2.6. ERwin Process Modeler

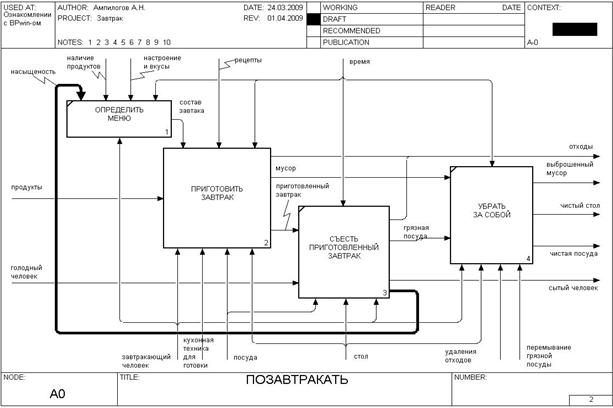
AllFusion ERwin Data Modeler (ранее ERwin) — CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных. Модели данных помогают визуализировать структуру данных, обеспечивая эффективный процесс организации, управления и администрирования таких аспектов деятельности предприятия, как уровень сложности данных, технологий баз данных и среды развертывания.

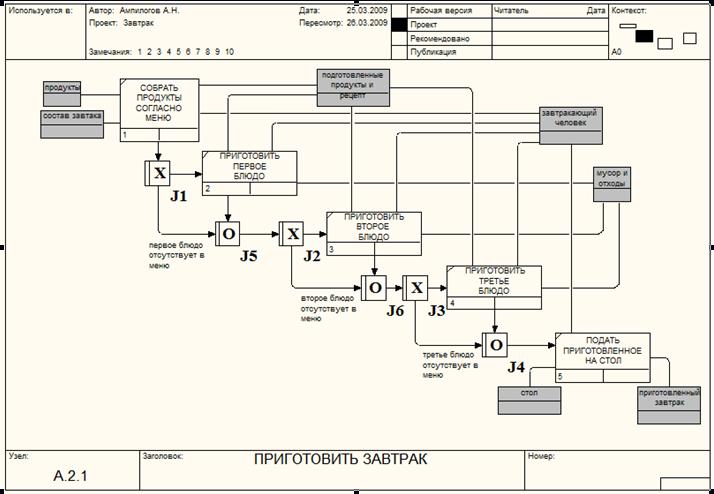
AllFusion ERwin Data Modeler (ERwin) предназначен для всех компаний, разрабатывающих и использующих базы данных, для администраторов баз данных, системных аналитиков, проектировщиков баз данных, разработчиков, руководителей проектов, — и позволяет управлять данными в процессе корпоративных изменений, а также в условиях стремительно изменяющихся технологий.

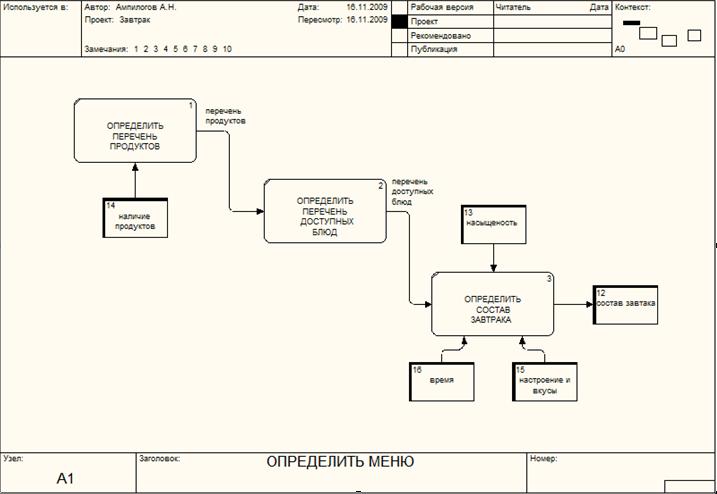
AllFusion ERwin Data Modeler (ERwin) позволяет наглядно отображать сложные структуры данных. Удобная в использовании графическая среда системы упрощает разработку базы данных и автоматизирует множество трудоёмких задач, уменьшая сроки создания высококачественных и высокопроизводительных транзакционных баз данных и хранилищ данных. Продукт 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.

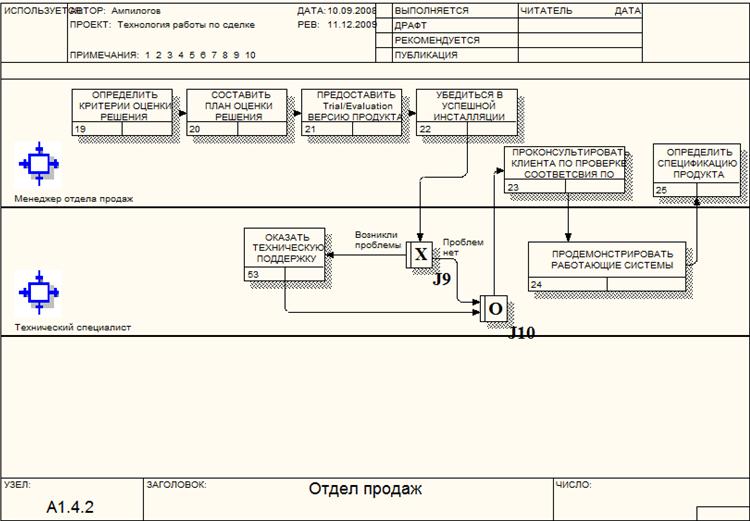
3.2.6.1. Примеры схем в ERwin











3.2.7. ModelMart

Model Mart - CASE-средство для рабочих групп. Это система управления моделями для групповой разработки при создании приложений для архитектуры "клиент-сервер", хранилищ данных, Web. Обеспечивает многопользовательский доступ к моделям, созданным с помощью ERwin и BPwin. Модели хранятся на центральном сервере и доступны для всех участников группы проектирования, при этом обеспечивается возможность коллективного создания сложных и объемных моделей.

Основные особенности ModelMart:

* поддерживается несколько версий модели, участники коллектива оектировщиков имеют доступ к самой последней версии модели и могут видеть результаты сделанных изменений;
* используются гибкие средства разграничения доступа;
* пользователь может получить список различий между двумя версиями модели и, при необходимости, вернуться к предыдущей версии;
* производится анализ влияния вносимых в модель изменений;
* поддерживается возможность слияния нескольких моделей в одну;
* обеспечивается гибкая блокировка доступа к моделям;
* специальное средство интеллектуального разрешения конфликтов автоматически идентифицирует любые конфликты, возникающие при одновременной модификации одной и той же модели несколькими пользователями.

ModelMart построен по правилам архитектуры "клиент-сервер": устанавливается на сервере в среде СУБД, в качестве которой в настоящее время могут выступать Sybase, MS SQL Server, Oracle , Informix. В качестве клиента могут выступать ERwin/ERX for ModelMart,ERwin/Navigator for ModelMart версий 2.6 и выше, и BPwin версий 2.01 и выше.

Модели в ModelMart хранятся в определяемых пользователем библиотеках. Все модели, хранящиеся в одной библиотеке, имеют в совместном владении некоторое множество объектов: это домены, триггеры, шаблоны для них, правила проверки и физические свойства. При создании в этой библиотеке новой модели, она автоматически получает доступ к этим разделяемым объектам. При внесении изменений в такой объект, эти изменения отображаются во всех моделях библиотеки, обеспечивая непротиворечивость проекта.

ModelMart поддерживает одновременную работу с моделями данных нескольких пользователей. Перед началом работы с моделью ModelMart сохраняет ее исходное состояние, которое и будет затем использоваться для сравнения изменений, внесенных конкретным пользователем, с изменениями, сделанными другими.

При одновременном доступе к модели, компонент "Интеллектуальное разрешение конфликтов" автоматически идентифицирует любые конфликты, которые могут возникнуть при попытках одновременно модифицировать модель.

Поддерживается возможность хранить несколько версий одной и той же модели.Раннюю версию модели можно сравнить с более поздней, увидеть все различия и, при необходимости, вернуться к старой версии.

Поддерживается средство слияния моделей. Независимые модели данных могут быть интеллектуально слиты в новую большую модель. Проектировщики могут интерактивно просматривать все производимые в модели изменения и оценивать их потенциальное влияние.

Специальный компонент - "менеджер управления изменениями", позволяет принять или отвергнуть каждое изменение. Возможность производить "откат" по отношению к конкретным изменениям позволяет проектировщику исследовать свою модель по принципу "а что, если". ModelMart также фиксирует, какой пользователь ответственен за конкретное изменение и когда он это сделал.

ModelMart включает специальную утилиту - ModelMart Synchronizer, позволяющую проводить синхронизацию моделей процессов (BPwin) и данных (ERwin), хранящихся в библиотеках ModelMart.

Разграничение доступа осуществляется на основе специальных профилей доступа. Каждому пользователю могут быть назначены предопределенные, встроенные в ModelMart, и специальные профили, настроенные для конкретной ситуации. Изменения, которые может внести конкретный пользователь, могут быть ограничены уровнем библиотеки, диаграммы или предметной области в пределах диаграммы. Это дает возможность строго разграничить ответственность членов коллектива.