**Применение средств CASE-анализа для решения задач обеспечения безопасности**

**7.1. Введение в CASE-средства**

CASE – информационная технология, представляющая собой совокупность средств проектирования и сопровождения ИС на всех этапах жизненного цикла (ЖЦ), поддержанную комплексом взаимоувязанных средств автоматизации. CASE - это инструментарий для программистов, аналитиков и разработчиков, заменяющий им бумагу и карандаш на компьютер для автоматизации процесса проектирования и разработки ПО. Технология направлена на ускорение процесса разработки систем и на улучшение качества разработки результитрующей системы. CASE не является методологией или альтернативой методологиям – CASE доступная технология, которая поддерживает стратегический подход к разработке ИС. Обычно к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла разработки ИС и обладающее следующими основными характерными особенностями:

* мощные графические средства для описания и документирования ИС, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности;
* интеграция отдельных компонентов CASE-средств, обеспечивающая управляемость процессом разработки ИС;
* использование специальным образом организованного хранилища проектных метаданных (репозитория).

Структура инструментальных средств CASE классифицируется в соответствии с фазами поддержки ЖЦ разработки ИС. Инструментальные средства высокого уровня автоматизируют поддержку ранних фаз разработки системы-обследование и изучение. Они включают построение словарей данных, диаграмму потоков данных, E-R-диаграмму для моделирования данных. Результатом применения CASE-средств этого уровня является модель предметной области, а также спецификации компонентов и интерфейсов системы, архтектуры системы, алгоритсмов и структур данных. CASE-средства низкого уровня автоматизируют или поддерживают более поздние стадии разработки систем – детальное проектирование, средства разработки приложений, применение и сопровождение. Кроме того, современные средства поддерживают реинжиниринг, обеспечивающий анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций.

**7.2. Архитектура CASE - средств**

Основой любых CASE-средств является база данных, наывающаяся репозиторием. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость. Вокруг репозитория располагается набор инструментальных средств, используемых для создания документации и других компонентов системы. CASE-средства обеспечивают следующие возможности:

1. Графические средства построения диаграмм, предназначенные для разработки моделей системы (DFD, ERD и др.), требуемых и рекомендуемых большинством методологий для выполнения анализа и проектирования ИС.
2. Средства описани, предазначенные для записи, удаления, редактирования и вывода неграфических документов и спецификаций.
3. Средства прототипирования, предназначенные для создания компонентов системы, включая входы, выходы и программы.
4. Средства реализации запросов и отчетов, предназначенные для моделей, описаний и спецификаций из репозитория.
5. Средства управления качеством, предназначенные для анализа моделей, описаний и прототипов на непротиворечивость, полноту или удовлетворение принятым правилам и стандартам.
6. Средства поддержки принятия решений, предназначенные для обеспечения информацией для принятия решений, которые могут возникнуть в процессе разработки.
7. Средства документирования, предназначенные для сборки, организации и вывода отчетов из репозитория, которые могут быть рассмотрены руководителями, пользователями, проектировщиками и разработчиками.
8. Средства генерации, предназначенные для создания первоначального проектирования развичных компонентов системы, основанных на требованиях, записанных в репозитории и технологических стандартах, обеспечиваемых разработчиком сиссемы.
9. Средства генерации кода, предназначенные для генерации программ приложения или существенных фрагментов этих программ.
10. Средства тестирования, предназначенные для помощи проектировщикам и разработчикам системы тестировать базы данных и программы приложения.
11. Средства распределения данных, предназначенные для обеспечения импорта и экспорта информации репозитория с другими программными средствами, которые не имеют прямого досиупа к репозиторию.
12. Средства безопасности и управления версиями, предазначенные для обеспечения целостности репозитория, предотвращения неавторизованных и небрежных изменений и сохранения предыдущих версий различной информации, сохраненной в репозитории.
13. Средства администрирования, предназначенные для установки пользовательских имен и привилегий, установки средств по умолчанию, выполнения процедур сохранения и восстановления данных репозитория и других существенных возможностей.

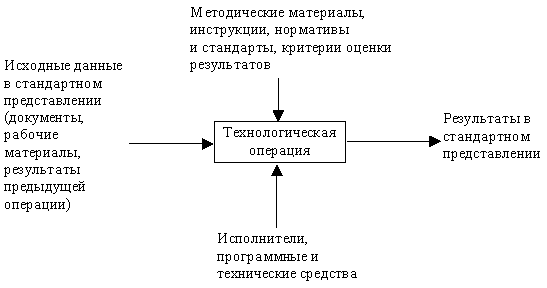
**7.3. Общие требования к методологии и технологии**

Методологии, технологии и инструментальные средства проектирования (CASE-средства) составляют основу проекта любой ИС. Методология реализуется через конкретные технологии и поддерживающие их стандарты, методики и инструментальные средства, которые обеспечивают выполнение процессов ЖЦ.

Технология проектирования определяется как совокупность трех составляющих:

* пошаговой процедуры, определяющей последовательность технологических операций проектирования (рис.7.1.);
* критериев и правил, используемых для оценки результатов выполнения технологических операций;
* нотаций (графических и текстовых средств), используемых для описания проектируемой системы.

Технологические инструкции, составляющие основное содержание технологии, должны состоять из описания последовательности технологических операций, условий, в зависимости от которых выполняется та или иная операция, и описаний самих операций.



*Рис. 7.1. Представление технологической операции проектирования*

Технология проектирования, разработки и сопровождения ИС должна удовлетворять следующим общим требованиям:

* технология должна поддерживать полный ЖЦ ИС;
* технология должна обеспечивать гарантированное достижение целей разработки ИС с заданным качеством и в установленное время;
* технология должна обеспечивать возможность выполнения крупных проектов в виде подсистем (т.е. возможность декомпозиции проекта на составные части, разрабатываемые группами исполнителей ограниченной численности, с последующей интеграцией составных частей). Опыт разработки крупных ИС показывает, что для повышения эффективности работ необходимо разбить проект на отдельные слабо связанные по данным и функциям подсистемы. Реализация подсистем должна выполняться отдельными группами специалистов. При этом необходимо обеспечить координацию ведения общего проекта и исключить дублирование результатов работ каждой проектной группы, которое может возникнуть в силу наличия общих данных и функций;
* технология должна обеспечивать возможность ведения работ по проектированию отдельных подсистем небольшими группами (3-7 человек). Это обусловлено принципами управляемости коллектива и повышения производительности за счет минимизации числа внешних связей;
* технология должна обеспечивать минимальное время получения работоспособной ИС. Речь идет не о сроках готовности всей ИС, а о сроках реализации отдельных подсистем. Реализация ИС в целом в короткие сроки может потребовать привлечения большого числа разработчиков, при этом эффект может оказаться ниже, чем при реализации в более короткие сроки отдельных подсистем меньшим числом разработчиков. Практика показывает, что даже при наличии полностью завершенного проекта, внедрение идет последовательно по отдельным подсистемам;
* технология должна предусматривать возможность управления конфигурацией проекта, ведения версий проекта и его составляющих, возможность автоматического выпуска проектной документации и синхронизацию ее версий с версиями проекта;
* технология должна обеспечивать независимость выполняемых проектных решений от средств реализации ИС (систем управления базами данных (СУБД), операционных систем, языков и систем программирования);
* технология должна быть поддержана комплексом согласованных CASE-средств, обеспечивающих автоматизацию процессов, выполняемых на всех стадиях ЖЦ.

Реальное применение любой технологии проектирования, разработки и сопровождения ИС в конкретной организации и конкретном проекте невозможно без выработки ряда стандартов (правил, соглашений), которые должны соблюдаться всеми участниками проекта. К таким стандартам относятся следующие:

* стандарт проектирования;
* стандарт оформления проектной документации;
* стандарт пользовательского интерфейса.

Стандарт проектирования должен устанавливать:

* набор необходимых моделей (диаграмм) на каждой стадии проектирования и степень их детализации;
* правила фиксации проектных решений на диаграммах, в том числе: правила именования объектов (включая соглашения по терминологии), набор атрибутов для всех объектов и правила их заполнения на каждой стадии, правила оформления диаграмм, включая требования к форме и размерам объектов, и т.д.;
* требования к конфигурации рабочих мест разработчиков, включая настройки операционной системы, настройки CASE-средств, общие настройки проекта и т.д.;
* механизм обеспечения совместной работы над проектом, в том числе: правила интеграции подсистем проекта, правила поддержания проекта в одинаковом для всех разработчиков состоянии (регламент обмена проектной информацией, механизм фиксации общих объектов и т.д.), правила проверки проектных решений на непротиворечивость и т. д.

Стандарт оформления проектной документации должен устанавливать:

* комплектность, состав и структуру документации на каждой стадии проектирования;
* требования к ее оформлению (включая требования к содержанию разделов, подразделов, пунктов, таблиц и т.д.);
* правила подготовки, рассмотрения, согласования и утверждения документации с указанием предельных сроков для каждой стадии;
* требования к настройке издательской системы, используемой в качестве встроенного средства подготовки документации;
* требования к настройке CASE-средств для обеспечения подготовки документации в соответствии с установленными требованиями.

Стандарт интерфейса пользователя должен устанавливать:

* правила оформления экранов (шрифты и цветовая палитра), состав и расположение окон и элементов управления;
* правила использования клавиатуры и мыши;
* правила оформления текстов помощи;
* перечень стандартных сообщений;
* правила обработки реакции пользователя.

**7.4. Методология функционального моделирования SADT**

Методология SADT разработана Дугласом Россом и получила дальнейшее развитие во многих работах. На ее основе разработана, в частности, известная методология IDEF0 (*Icam DEFinition*), которая является основной частью программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой по инициативе ВВС США.

Методология SADT представляет собой 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. Основные элементы этой методологии основываются на следующих концепциях:

* графическое представление блочного моделирования. Графика блоков и дуг SADT-диаграммы отображает функцию в виде блока, а интерфейсы входа/выхода представляются дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описываются посредством интерфейсных дуг, выражающих "ограничения", которые, в свою очередь, определяют, когда и каким образом функции выполняются и управляются;
* строгость и точность. Выполнение правил SADT требует достаточной строгости и точности, не накладывая в то же время чрезмерных ограничений на действия аналитика.

Правила SADT включают:

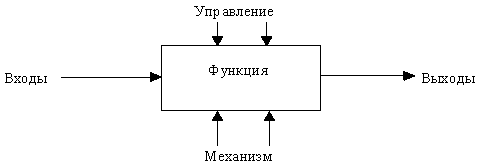
* ограничение количества блоков на каждом уровне декомпозиции (правило 3-6 блоков);
* связность диаграмм (номера блоков);
* уникальность меток и наименований (отсутствие повторяющихся имен);
* синтаксические правила для графики (блоков и дуг);
* разделение входов и управлений (правило определения роли данных);
* отделение организации от функции, т.е. исключение влияния организационной структуры на функциональную 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. Для уже существующих систем SADT может быть использована для анализа функций, выполняемых системой, а также для указания механизмов, посредством которых они осуществляются.

***7.4.1.Состав функциональной модели***

Результатом применения методологии SADT является модель, которая состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга. Диаграммы - главные компоненты модели, все функции ИС и интерфейсы на них представлены как блоки и дуги. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке, показана с левой стороны блока, а результаты выхода показаны с правой стороны. Механизм (человек или автоматизированная система), который осуществляет операцию, представляется дугой, входящей в блок снизу (рис.7.2.).

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

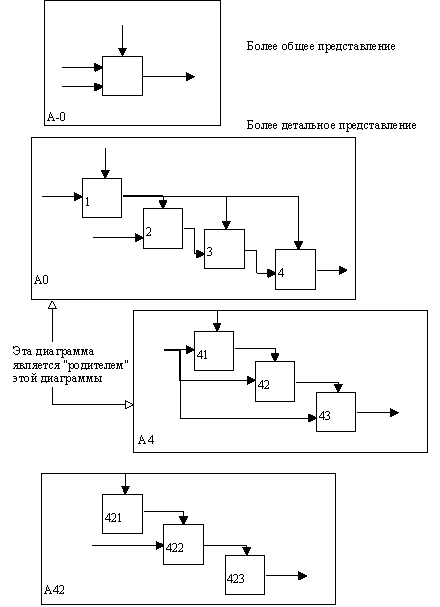


*Рис.7.2. Функциональный блок и интерфейсные дуги*

На рис.7.3, где приведены четыре диаграммы и их взаимосвязи, показана структура SADT-модели. Каждый компонент модели может быть декомпозирован на другой диаграмме. Каждая диаграмма иллюстрирует "внутреннее строение" блока на родительской диаграмме.

***7.5.2 Иерархия диаграмм***

Построение SADT-модели начинается с представления всей системы в виде простейшего компонента - одного блока и дуг, изображающих интерфейсы с функциями вне системы. Поскольку единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг - они также представляют полный набор внешних интерфейсов системы в целом.



*Рис.7.3.Структура SADT-модели. Декомпозиция диаграмм*

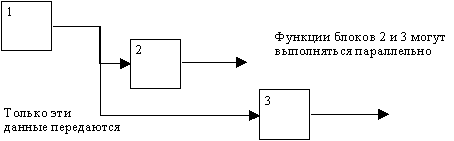
Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых представлена как блок, границы которого определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления.

Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию. Кроме того, модель не может опустить какие-либо элементы, т.е., как уже отмечалось, родительский блок и его интерфейсы обеспечивают контекст. К нему нельзя ничего добавить, и из него не может быть ничего удалено.

Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других диаграммах. Каждая детальная диаграмма является декомпозицией блока из более общей диаграммы. На каждом шаге декомпозиции более общая диаграмма называется родительской для более детальной диаграммы.

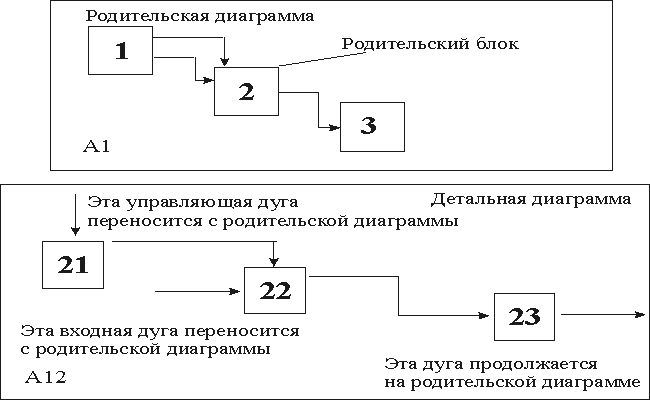
Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются точно теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.

На рис.7.4 - 7.6. представлены различные варианты выполнения функций и соединения дуг с блоками.

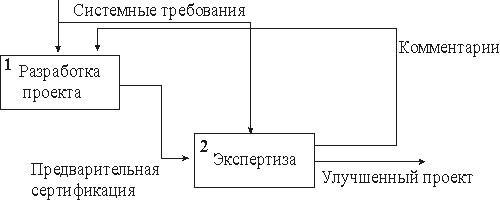


*Рис.7.4. Одновременное выполнение*

Некоторые дуги присоединены к блокам диаграммы обоими концами, у других же один конец остается неприсоединенным. Неприсоединенные дуги соответствуют входам, управлениям и выходам родительского блока. Источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме. Неприсоединенные концы должны соответствовать дугам на исходной диаграмме. Все граничные дуги должны продолжаться на родительской диаграмме, чтобы она была полной и непротиворечивой. На SADT-диаграммах не указаны явно ни последовательность, ни время. Обратные связи, итерации, продолжающиеся процессы и перекрывающиеся (по времени) функции могут быть изображены с помощью дуг. Обратные связи могут выступать в виде комментариев, замечаний, исправлений и т.д. (рис.7.6).

**

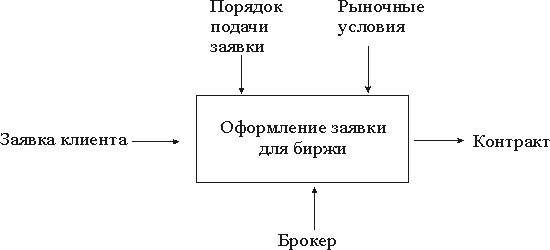
*Рис. 7.5. Соответствие должно быть полным и непротиворечивым*



*Рис. 7.6. Пример обратной связи*

Как было отмечено, механизмы (дуги с нижней стороны) показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию (рис.7.7).

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть далее описан диаграммой нижнего уровня, которая, в свою очередь, может быть далее детализирована с помощью необходимого числа диаграмм. Таким образом, формируется иерархия диаграмм.

****

*Рис.7.7 Пример механизма*

Для того чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм. Например, А21 является диаграммой, которая детализирует блок 1 на диаграмме А2. Аналогично А2 детализирует блок 2 на диаграмме А0, которая является самой верхней диаграммой модели. На рис.7.8 показано типичное дерево диаграмм.



*Рис.7.8. Иерархия диаграмм*

***7.4.3. Методология IDEF1X***

В последнее время ведущие корпорации убеждались в необходимости управления данными как ресурсом. Для успешного управления данными в условиях конкурентной борьбы и обеспечения гибкости инфраструктуры бизнеса следует владеть данными и знаниями , необходимыми для ведения дел. Трудности управления данными усугубляются стремительным ростом объема данных и их разнообразием. Исследования показывают, что данные являются несогласованными, несвоевременными, негибкими, недоступными и не соответствующими текущим деловым потребностям.

Для управления данными необходимо понимать их характеристики и уметь интерпретировать факты. Факт без интерпретации не имеет ценности, а факт с неправильной интерпретацией может привести к губительным последствиям. Поэтому управление данными должно главным образом ориентироваться на интерпретацию данных.

Как известно, существуют две различные точки зрения к построению информационных систем: точка зрения пользователя и точка зрения компьютера. С точки зрения пользователя, которая называется внешней схемой, определение данных представляется в контексте отчетов и выборок предназначенных для анализа и принятия решения. В этом случае структура данных изменяется в зависимости от видов деятельности и особенностей конкретных пользователей. С точки зрения компьютера, называемой внутренней схемой, данные определяются в терминах структур файлов для хранения и поиска. Структура данных зависит от конкретной компьютерной технологии и от потребности в эффективной обработке данных.

Стремление к гибкости в организации данных привело к созданию СУБД, позволяющих осуществлять прямой доступ к логически связанным порциям данных. Однако применение СУБД еще не гарантирует непротиворечивость и избыточность определения данных. Для создания идеальной среды управления данными в настоящее время используется концептуальная схема, определяющая данные в рамках системы, не ориентированная на какое либо конкретное использование и не зависящая от организации физического хранения и доступа к ним.

Концептуальная схема должна удовлетворять следующим требованиям:

* быть согласованной с инфраструктурой бизнеса и верной во всех сферах применения;
* при расширении новые данные должны определяться без изменения ранее определенных;
* должна адаптироваться как к точкам зрения пользователей, так и к многообразию структур хранения данных и доступа к ним.

Необходимость определения данных на концептуальном уровне привела к разработке методологии моделирования данных, основанной на семантике, то есть к трактовке данных в контексте их взаимодействия с другими данными.

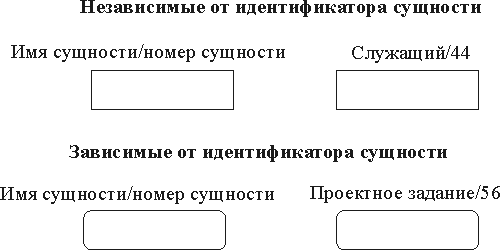
Методология IDEF1 – это методология семантического моделирования данных, с учетом таких требований, как простота изучения и возможность автоматизации. IDEF1X-диаграммы используются рядом распространенных CASE-средств (в частности, ERwin, Design/IDEF).

IDEF1X использует подход сущностей-отношений к семантическому моделированию. Основными конструкциями IDEF1X-модели являются:

* предметы, к которым относятся данные, т.е. люди, места, идеи, события и т.д. Они изображаются блоками;
* отношения между этими предметами, изображаемые соединяющими блоки линиями;
* характеристики этих предметов, изображаемые именами атрибутов внутри блоков.

Сущность представляет множество реальных или абстрактных предметов (людей, объектов, мест, событий, состояний и т.д.), обладающих общими атрибутами и характеристиками. Отдельны элемент этого множества называется экземпляром сущности. Реально существующий объект может быть представлен в нескольких сущностях модели данных.

Сущность в методологии IDEF1X является независимой от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности (рис.7.9).



*Рис. 7.9 Синтаксис сущности*

Каждой сущности присваиваются уникальное имя и номер, разделяемые косой чертой "/" и помещаемые над блоком.

*Правила, связанные с сущностью*

Каждая сущность должна иметь уникальное имя, и к одному и тому же имени должна применяться одна и та же интерпретация.

Сущность обладает одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через отношение.

Сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности.

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

Если внешний ключ целиком используется в качестве первичного ключа сущности или его части, то сущность является зависимой от идентификатора. И наоборот, если если используется только часть внешний ключа или вообще не используются внешние ключи, то сущность является независимой от идентификатора

Отношение связи - это связь между сущностями, при которой каждый экземпляр одной сущности, называемой родительской сущностью, ассоциирован с произвольным количеством экземпляров второй сущности, называемой сущностью-потомком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя.

Отношение связи может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может существовать для каждого экземпляра сущности-родителя). В IDEF1X могут быть выражены следующие мощности связей:

каждый экземпляр сущности-родителя может иметь ноль, один или более связанных с ним экземпляров сущности-потомка;

каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;

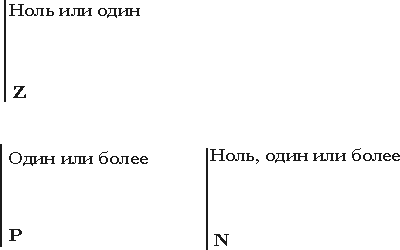
каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;

каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

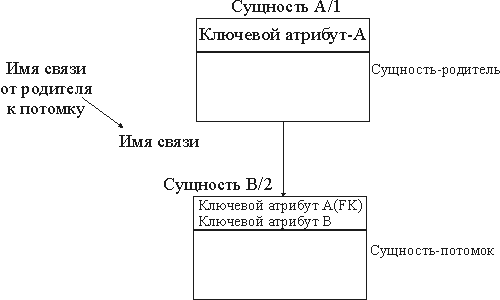
Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае - неидентифицирующей.

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка. Мощность по умолчанию «ноль, один или много». Мощность связи обозначается как показано на рис.7.10 (мощность по умолчанию - N).

Идентифицирующая связь между сущностью-родителем и сущностью-потомком изображается сплошной линией (рис. 7.11). Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее связями с другими сущностями).



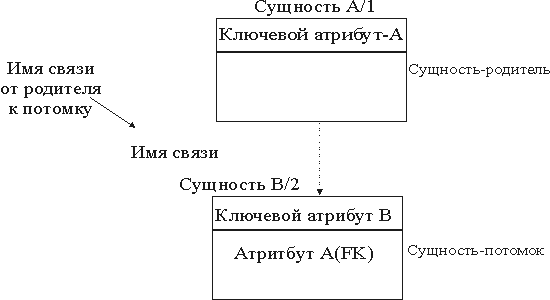
*Рис. 7.10 Синтаксис мощности отношения*



*Рис.7.11 Идентифицирующая связь*

Пунктирная линия изображает неидентифицирующую связь (рис.7.12). Сущность-потомок в неидентифицирующей связи будет независимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

Отношению присваивается имя и помещается возле линии отношения. Имя формируется с точки зрения родителя. Например, если утверждение «Проект состоит из одного или более заданий» выводится из отношения, то в качестве имени отношения выступает СОСТОИТ\_ИЗ.



*Рис. 7.12 Неидентифицирующая связь*

*Правила отношений*

Специфическое отношение всегда имеет место между двумя сущностями - сущностью-родителем и сущностью-потомком.

Экземпляр сущности-потомка всегда должен быть связан в точности с одним экземпляром сущности-родителя.

Экземпляр сущности-родителя может быть связан с любым числом (от нуля и более) экземпляров сущностей-потмков, где число зависит от указанной мощности.

В идентифицирующем отношени сущность-потомок всегда является зависимой от идентификатора сущностью.

Сущность может быть связана с любым количеством других сущностей как в качестве потомеа, так и в качсестве родителя.

Неекоторые реально существующие сущности являются категориями других сущностей. Так, ШТАТНЫЙ\_СЛУЖАЩИЙ и СЛУЖАЩИЙ\_ПОЧАСОВИК являются категориями сущности СЛУЖАЩИЙ. В IDEF1X-модели они связаны друг с другом через категориальное отношение. Сущности-категории являются взаимоисключающими. Существует также отношение неполной категоризации, когда имеется экземпляр общей сущности, не связанный ни с каким экземпляром из сущностей-категорий. Пример отношений категоризации приведен на рис.7.13.

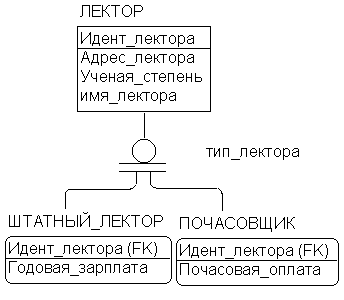
*Правила отношений категоризации*

Сущность-категория может иметь только одну общую сущность.

Сущность-категория не может быть сущностью- потомком в идентифицирующем отношении.

Атрибуты первичного ключа сущности-категории должны совпадать с атрибутами первичного ключа общей сущности.

Атрибуты представляют тип характеристик или свойств, ассоциированных со множеством реальных или абстрактных объектов. Экземпляр атрибута определяется типом хараетеристики и ее значением. Атрибуты изображаются в виде списка имен внутри блока сущности.



*Рис.7.13. Отношение категоризации*

*Правила атрибутов*

Каждый атрибут должен иметь уникальное имя.

Сущность может обладать любым количеством атрибутов.

Сущность может обладать любым количеством наследуемых атрибутов.

Каждый экземпляр сущности должен иметь значение для каждого атрибута (правило необращения в ноль).

Ни один из экземпляров сущности не может обладать более чем одним значением для связанного с сущностью атрибута (правило неповторяемости).

Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой (рис. 7.14).

**

*Рис. 7.14 Атрибуты и первичные ключи*

*Правила первичных и альтернативных ключей*

Каждая сущность должна обладать первичным ключом.

Каждая сущность может обладать любым числом альтернативных ключей.

Первичный или альтернативный ключ могут состоять из одного атрибута или комбинации атрибутов.

Один атрибут может быть частью более чем одного ключа, первичного или альтернативного.

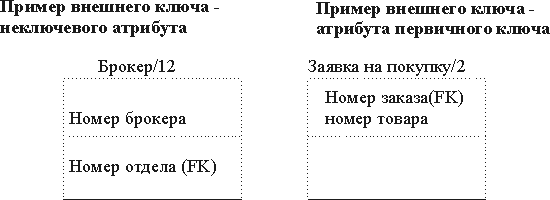
Атрибуты, входящие в первичные ключи сущности, могут быть собственными для сущности или наследоваться через отношения.

Первичные и альтернативные ключи должны содержать только необходимые для однозначной идентификации атрибуты, т.е. при исключении из ключа любого атрибута не все экземпляры сущности могут быть однозначно определены.

Если первичный ключ состоит более чем из одного атрибута, то значение любого неключевого атрибута должно функционально зависеть от всего первичного ключа (правило полной функциональной зависимости).

Каждый неключевой атрибут должен функционально зависеть только от первичного и альтернативных ключей, т.е. значение неключевого атрибута не может определяться значением другого неключевого атрибута (правило отсутствия транзитивной зависимости).

Сущности могут иметь также внешние ключи (*Foreign Key*), которые могут использоваться в качестве части или целого первичного ключа или неключевого атрибута. Внешний ключ изображается с помощью помещения внутрь блока сущности имен атрибутов, после которых следуют буквы FK в скобках (рис. 7.15).



*Рис.7.15. Примеры внешних ключей*

*Правила внешних ключей.*

Каждая сущность должна содержать отдельный внешний ключ для каждого отношения связи или категоризации, в котором эта сущность является сущностью-потомком или сущностью-категорией.

Первичный ключ общей сущности должен наследоваться в качестве первичного ключа для каждой сущности-категории. Каждый наследуемый атрибут сущности-потомка или сущности-категории должен представлять атрибут из первичного ключа связанной родительской или общей сущности. Наоборот, каждый атрибут первичного ключа родительской или общей сущности должен быть наследуемым атрибутом в связанной сущности-потомке или сущности –категории. Этапы и задачи моделирования IDEF1X можно представить в следующей последовательности:

планирование проекта;

сбор данных;

определение сущностей;

определение отношений;

определение ключевых атрибутов;

заполнение неключевых атрибутов;

проверка правильности модели;

приемка модели.

Для реализации этапов информационного моделирования и обеспечения управления разработкой рекомендуется пять основных ролей:

менеджер проекта;

разработчик;

источники информации;

эксперты в рассматриваемой предметной области;

комитет рецензирования.

Допускается совмещение обязанностей в коллективе, но недостаточное рассмотрение точек зрения может привести к неадекватному представлению проблемы и неполному достижению целей моделирования.

Менеджер проекта осуществляет общее руководство проектом моделирования и выполняет следующие основные функции:

* назначает разработчика модели и согласовывает основные правила процесса моделирования (степень контроля, область действия и ориентацию модели);
* выявляет источники информации и обеспечивает ею разработчиков;
* выбирает экспертов и формирует комитет рецензирования и одобрения.

Одной из первых проблем в процессе разработки модели является определение потребности информации из различных источников. Область действия и контекст модели устанавливаются на основе анализа функциональной IDEF0-модели. Анализ функций и связей с ними выявляет совокупность информации, необходимой для ее реализации. Кроме того, исходный материал может иметь разнообразные формы, включающие:

результаты опроса;

результаты наблюдения;

технологию и процедуры обработки;

выходные данные существующих систем (отчеты и выборки);

входные данные для существующих систем (бланки входных данных и выборки);

спецификации баз данных и файлов для существующих систем.

В результате выполнения модели определяются семантические характеристики данных, котрые могут быть использованы для организации потока данных и управления при создании распределенных баз данных и интеграции информационных систем. Использование программного обеспечения IDEF1X значительно облегчает создание документации и конфигурации модели, равно как и взаимодействие коллектива разработчиков.

**IDEF0**  — [методология](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) функционального моделирования ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *function modeling*) и графическая нотация, предназначенная для формализации и описания [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81). Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временна́я последовательность ([поток работ](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D1%80%D0%B0%D0%B1%D0%BE%D1%82)).

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.

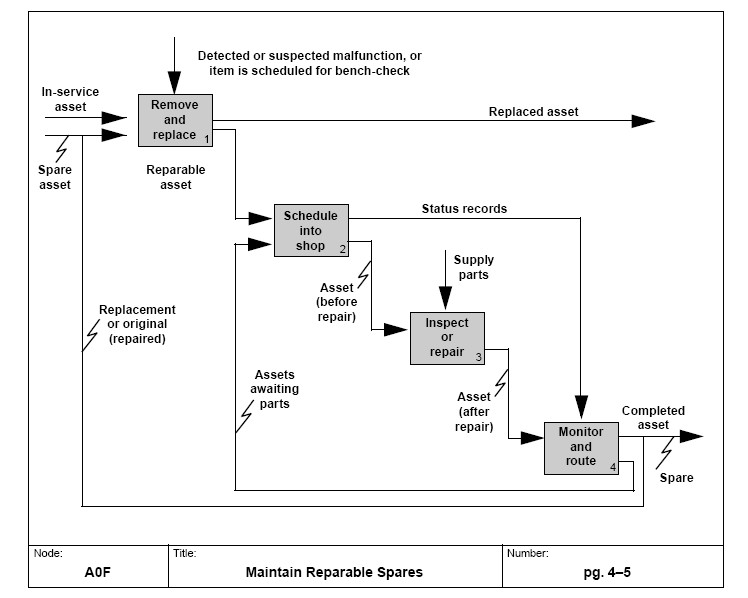
Описание выглядит как «[чёрный ящик](https://ru.wikipedia.org/wiki/%D0%A7%D1%91%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D1%89%D0%B8%D0%BA)» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того чтобы быть правильно понятым, существуют словари описания 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](https://ru.wikipedia.org/wiki/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.

IDEF0 как стандарт был разработан в 1981 году департаментом [Военно-воздушных сил США](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B5%D0%BD%D0%BD%D0%BE-%D0%B2%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D0%BB%D1%8B_%D0%A1%D0%A8%D0%90) в рамках программы [автоматизации](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) промышленных предприятий, которая носила обозначение [ICAM](https://ru.wikipedia.org/w/index.php?title=ICAM&action=edit&redlink=1) (*Integrated Computer Aided Manufacturing*). Набор стандартов [IDEF](https://ru.wikipedia.org/wiki/IDEF) унаследовал своё название от этой программы (IDEF расшифровывается как *ICAM Definition*). В процессе практической реализации, участники программы ICAM столкнулись с необходимостью разработки новых методов анализа процессов взаимодействия в промышленных системах. При этом кроме усовершенствованного набора функций для описания бизнес-процессов, одним из требований к новому стандарту было наличие эффективной методологии взаимодействия в рамках «аналитик-специалист». Другими словами, новый метод должен был обеспечить групповую работу над созданием модели, с непосредственным участием всех аналитиков и специалистов, занятых в рамках проекта.

В результате поиска соответствующих решений родилась методология функционального моделирования IDEF0. С 1981 года стандарт IDEF0 претерпел несколько незначительных изменений, в основном, ограничивающего характера, и последняя его редакция была выпущена в декабре 1993 года [Национальным институтом по стандартам и технологиям США](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2_%D0%B8_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B9).

**IDEF** — методологии создавались в рамках предложенной [ВВС США](https://ru.wikipedia.org/wiki/%D0%92%D0%92%D0%A1_%D0%A1%D0%A8%D0%90) программы компьютеризации промышленности — ICAM, в ходе реализации которой выявилась потребность в разработке методов анализа процессов взаимодействия в производственных (промышленных) системах. Принципиальным требованием при разработке рассматриваемого семейства методологий была возможность эффективного обмена информацией между всеми специалистами — участниками программы ICAM (отсюда название: Icam DEFinition — IDEF другой вариант — Integrated DEFinition). После опубликования стандарта он был успешно применен в самых различных областях бизнеса, показав себя эффективным средством анализа, конструирования и отображения [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81). Более того, собственно с широким применением IDEF (и предшествующей методологии — [SADT](https://ru.wikipedia.org/wiki/SADT)) и связано возникновение основных идей популярного ныне понятия — [BPR](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B8%D0%BD%D0%B6%D0%B8%D0%BD%D0%B8%D1%80%D0%B8%D0%BD%D0%B3_%D0%B1%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D0%B2) ([бизнес-процесс](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81) [реинжиниринг](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0)).

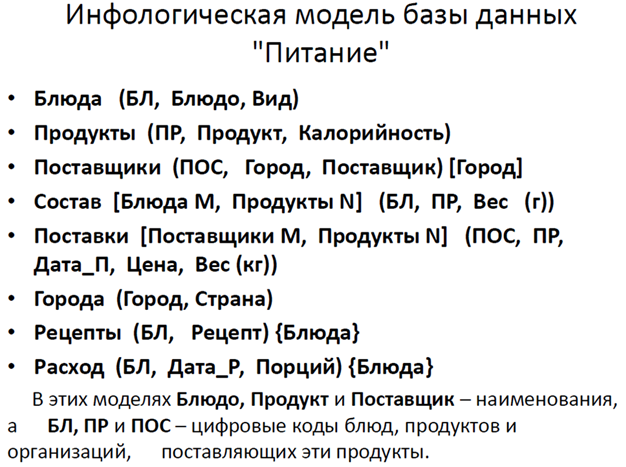


Пример диаграммы [IDEF0](https://ru.wikipedia.org/wiki/IDEF0): [функциональная модель](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) процесса «Поддержка запчастей, подлежащих починке».

[Function Modeling](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81) — методология функционального моделирования. С помощью наглядного графического языка IDEF0 изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков — в терминах IDEF0). Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы. Методологию IDEF0 можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем [SADT](https://ru.wikipedia.org/wiki/SADT) (Structured Analysis and Design Technique);

[**IDEF1**](https://ru.wikipedia.org/wiki/IDEF1)

[Information Modeling](https://ru.wikipedia.org/wiki/DFD) — методология моделирования информационных потоков внутри системы, позволяющая отображать и анализировать их структуру и взаимосвязи. [IDEF1X](https://ru.wikipedia.org/w/index.php?title=IDEF1X&action=edit&redlink=1) (IDEF1 Extended) — [Data Modeling](https://ru.wikipedia.org/wiki/ER-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) — методология моделирования баз данных на основе модели «сущность-связь». Применяется для построения информационной модели, которая представляет структуру информации, необходимой для поддержки функций производственной системы или среды. Метод IDEF1, разработанный Т. Рэмей (T. Ramey), также основан на подходе П. Чена и позволяет построить модель данных, эквивалентную реляционной модели в [третьей нормальной форме](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D1%82%D1%8C%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0). В настоящее время на основе совершенствования методологии IDEF1 создана её новая версия — методология IDEF1X. IDEF1X разработана с учетом таких требований, как простота изучения и возможность автоматизации. IDEF1X-диаграммы используются рядом распространённых CASE-средств (в частности, [ERwin](https://ru.wikipedia.org/wiki/ERwin_Data_Modeler), Design/IDEF).



Пример модели [IDEF1X](https://ru.wikipedia.org/w/index.php?title=IDEF1X&action=edit&redlink=1). на ЯИМ



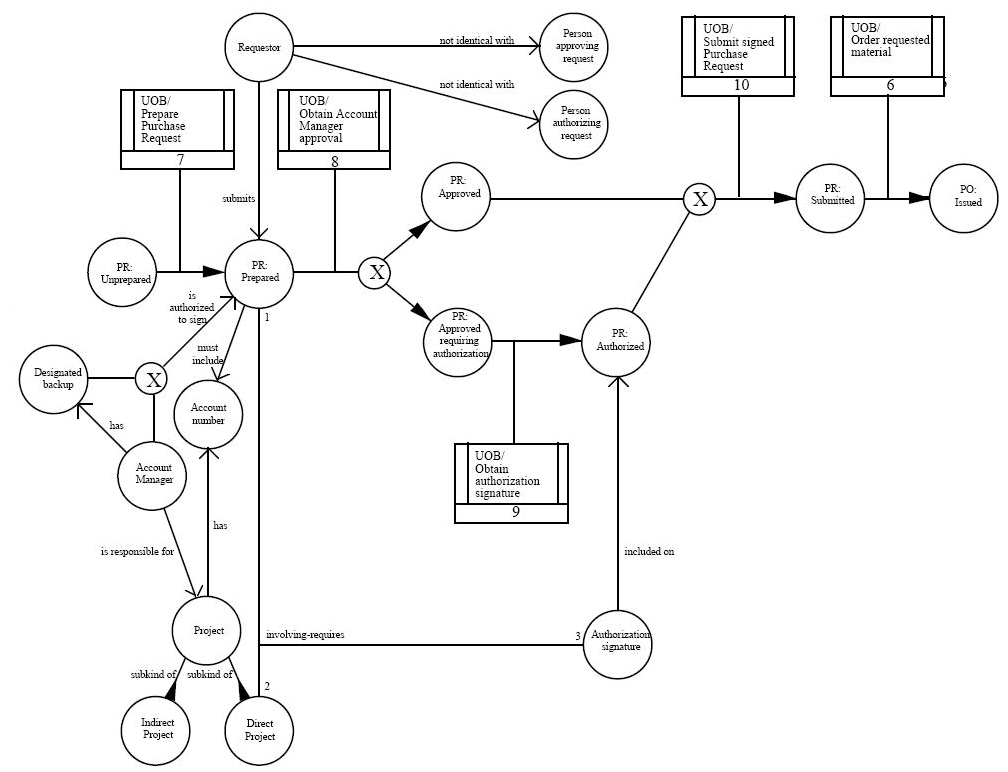
Пример ER-диаграммы [IDEF1X](https://ru.wikipedia.org/w/index.php?title=IDEF1X&action=edit&redlink=1).

[**IDEF2**](https://ru.wikipedia.org/w/index.php?title=IDEF2&action=edit&redlink=1)

Simulation Model Design — методология динамического моделирования развития систем. В связи с весьма серьёзными сложностями анализа динамических систем от этого стандарта практически отказались, и его развитие приостановилось на самом начальном этапе. В настоящее время присутствуют алгоритмы и их компьютерные реализации, позволяющие превращать набор статических диаграмм IDEF0 в динамические модели, построенные на базе «раскрашенных [сетей Петри](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B8_%D0%9F%D0%B5%D1%82%D1%80%D0%B8)» (CPN — Color Petri Nets);

[**IDEF3**](https://ru.wikipedia.org/wiki/IDEF3)

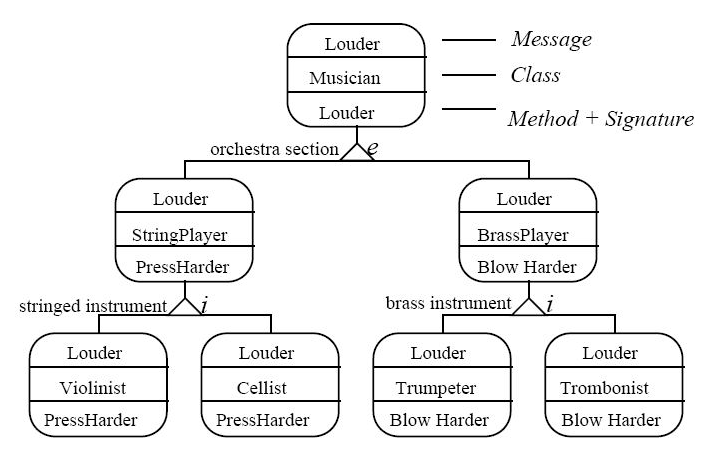
Process Description Capture (Документирование технологических процессов) — методология документирования процессов, происходящих в системе (например, на предприятии), описываются сценарий и последовательность операций для каждого процесса. IDEF3 имеет прямую взаимосвязь с методологией IDEF0 — каждая функция (функциональный блок) может быть представлена в виде отдельного процесса средствами IDEF3.



Пример расширенной схемы перехода, модель [IDEF3](https://ru.wikipedia.org/wiki/IDEF3).

[IDEF4](http://en.wikipedia.org/wiki/IDEF4)

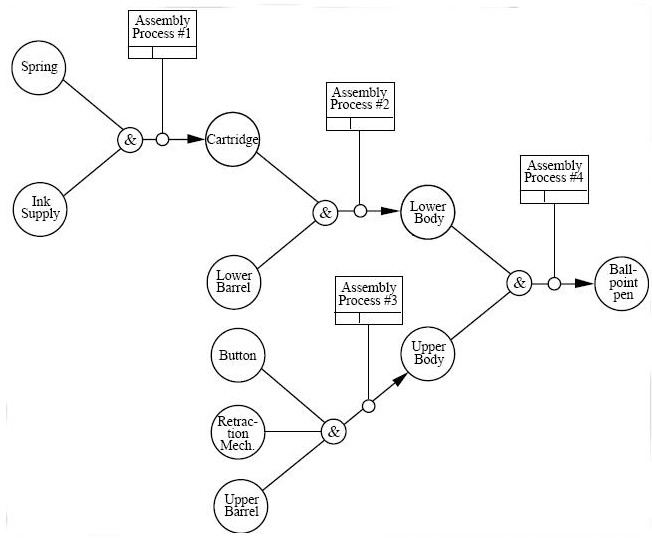
[Object-Oriented Design](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — методология построения объектно-ориентированных систем, позволяют отображать структуру объектов и заложенные принципы их взаимодействия, тем самым позволяя анализировать и оптимизировать сложные объектно-ориентированные системы.



Пример методологии проектирования [IDEF4](https://ru.wikipedia.org/w/index.php?title=IDEF4&action=edit&redlink=1): диаграмма поведения для объектов, реализующих метод «Громче».

[Object-Oriented Design](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — методология построения объектно-ориентированных систем, позволяют отображать структуру объектов и заложенные принципы их взаимодействия, тем самым позволяя анализировать и оптимизировать сложные объектно-ориентированные системы.

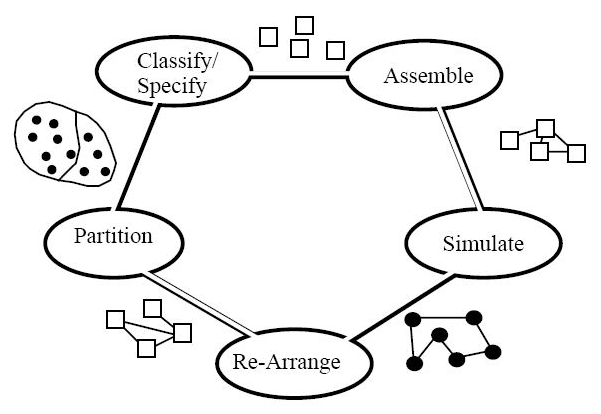
[**IDEF5**](http://en.wikipedia.org/wiki/IDEF5)



Пример [IDEF5](https://ru.wikipedia.org/w/index.php?title=IDEF5&action=edit&redlink=1)-схемы состава шариковой ручки.

Ontology Description Capture — Стандарт онтологического исследования сложных систем. С помощью методологии IDEF5 [онтология](https://ru.wikipedia.org/wiki/%D0%9E%D0%BD%D1%82%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) системы может быть описана при помощи определенного словаря терминов и правил, на основании которых могут быть сформированы достоверные утверждения о состоянии рассматриваемой системы в некоторый момент времени. На основе этих утверждений формируются выводы о дальнейшем развитии системы и производится её оптимизация;

[**IDEF6**](http://en.wikipedia.org/wiki/IDEF6)



Модель IDEF6 деятельностей проектирования IDEF4

Design Rationale Capture — Обоснование проектных действий. Назначение IDEF6 состоит в облегчении получения «знаний о способе» моделирования, их представления и использования при разработке систем управления предприятиями. Под «знаниями о способе» понимаются причины, обстоятельства, скрытые мотивы, которые обуславливают выбранные методы моделирования. Проще говоря, «знания о способе» интерпретируются как ответ на вопрос: «почему модель получилась такой, какой получилась?» Большинство методов моделирования фокусируются на собственно получаемых моделях, а не на процессе их создания. Метод IDEF6 акцентирует внимание именно на процессе создания модели;

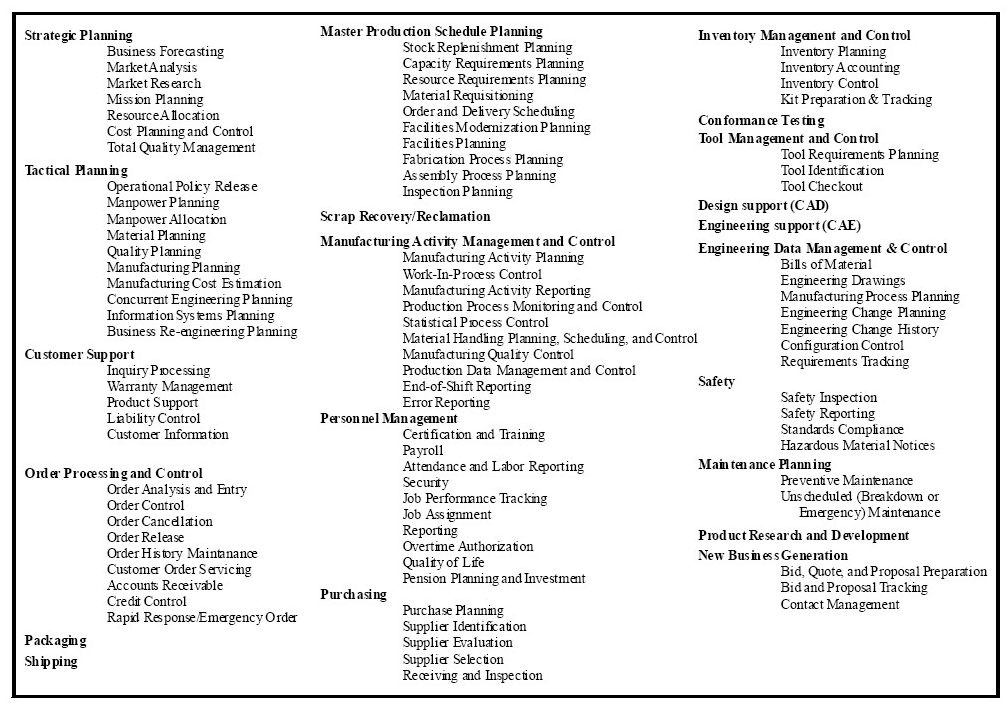
[**IDEF7**](https://ru.wikipedia.org/w/index.php?title=IDEF7&action=edit&redlink=1)

Information System Auditing — Аудит информационных систем. Этот метод определён как востребованный, однако так и не был полностью разработан;

[**IDEF8**](https://ru.wikipedia.org/w/index.php?title=IDEF8&action=edit&redlink=1)

User Interface Modeling — Метод разработки интерфейсов взаимодействия оператора и системы (пользовательских интерфейсов). Современные среды разработки пользовательских интерфейсов в большей степени создают внешний вид интерфейса. IDEF8 фокусирует внимание разработчиков интерфейса на программировании желаемого взаимного поведения интерфейса и пользователя на трех уровнях: выполняемой операции (что это за операция); сценарии взаимодействия, определяемом специфической ролью пользователя (по какому сценарию она должна выполняться тем или иным пользователем); и, наконец, на деталях интерфейса (какие элементы управления, предлагает интерфейс для выполнения операции);

[**IDEF9**](https://ru.wikipedia.org/w/index.php?title=IDEF9&action=edit&redlink=1)



Типичные бизнес-системы.

Scenario-Driven IS Design (Business Constraint Discovery method) — Метод исследования бизнес-ограничений был разработан для облегчения обнаружения и анализа ограничений в условиях, которых действует предприятие. Обычно, при построении моделей описанию ограничений, оказывающих влияние на протекание процессов на предприятии, уделяется недостаточное внимание. Знания об основных ограничениях и характере их влияния, закладываемые в модели, в лучшем случае остаются неполными, несогласованными, распределенными нерационально, но часто их вовсе нет. Это не обязательно приводит к тому, что построенные модели нежизнеспособны, просто их реализация столкнется с непредвиденными трудностями, в результате чего их потенциал будет не реализован. Тем не менее, в случаях, когда речь идет именно о совершенствовании структур или адаптации к предсказываемым изменениям, знания о существующих ограничениях имеют критическое значение;

**IDEF10 — IDEF14**

* [IDEF10](https://ru.wikipedia.org/w/index.php?title=IDEF10&action=edit&redlink=1) — Implementation Architecture Modeling — Моделирование архитектуры выполнения. Этот метод определён как востребованный, однако так и не был полностью разработан;
* [IDEF11](https://ru.wikipedia.org/w/index.php?title=IDEF11&action=edit&redlink=1) — Information Artifact Modeling. Этот метод определён как востребованный, однако так и не был полностью разработан;
* [IDEF12](https://ru.wikipedia.org/w/index.php?title=IDEF12&action=edit&redlink=1) — Organization Modeling — Организационное моделирование. Этот метод определён как востребованный, однако так и не был полностью разработан;
* [IDEF13](https://ru.wikipedia.org/w/index.php?title=IDEF13&action=edit&redlink=1) — Three Schema Mapping Design — Трёхсхемное проектирование преобразования данных. Этот метод определён как востребованный, однако так и не был полностью разработан;
* [IDEF14](https://ru.wikipedia.org/w/index.php?title=IDEF14&action=edit&redlink=1) — Network Design — Метод проектирования компьютерных сетей, основанный на анализе требований, специфических сетевых компонентов, существующих конфигураций сетей. Также он обеспечивает поддержку решений, связанных с рациональным управлением материальными ресурсами, что позволяет достичь существенной экономии.

**Применение средств CASE-анализа для решения задач обеспечения безопасности**

В процессе разработки и реализации плана защиты информационных ресурсов админист-

ратор безопасности сталкивается с необходимостью изучения и формализации информационных

потоков предприятия с целью разграничения двух групп рисков: рисков, связанных с несовершен-

ством организационной структуры предприятия, и рисков, связанных с несовершенством компью-

терных процедур обработки информации в рамках корпоративной сети. В целях минимизации

второй группы рисков и разрабатывается план защиты корпоративной сети.

Администратор безопасности сталкивается с необходимостью определить круг своих обя-

занностей в соответствии с формализованной моделью информационных процессов предприятия

и принятой на предприятии политикой безопасности. Следует отметить, что разработать эту мо-

дель невозможно без применения средств системного анализа. К этим средствам относятся рас-

пространенные в настоящее время CASE-системы типа CASE-Аналитик.

Как показано на рис. 1, практическая реализация политики безопасности на предприятии

предполагает наличие у администратора безопасности трех актуальных моделей: технологии обра-

ботки информации, политики безопасности и вероятного нарушителя.



**Рис. 1. Соотношение между информационной базой предприятия, администратором**

**безопасности и CASE-системой**

Очевидно, что разработка этих моделей и поддержание их в рабочем состоянии представ-

ляет собой нетривиальную задачу, выполнить которую можно только при наличии инструмен-

тальных средств системного анализа. Остановимся на описании базовых подходов к процессу раз-

работки этих трех моделей. Отметим, что нижеописанный процесс является итеративным и перио-

дически возобновляется по мере развития корпоративной сети.

Итак, на первом этапе разработки моделей администратор безопасности осуществляет ана-

лиз деятельности структурных подразделений организации.

На этом этапе выявляются схемы функциональных взаимосвязей подразделений, потоки

информации и строится схема соответствующего документооборота. Итогом работы является мо-

дель технологии обработки информации, актуальная на дату проведении обследования. Для вы-

полнения этой задачи в некоторых случаях целесообразно использовать программный пакет сис-

темного анализа Design/IDEF, в основу которого положены следующие методологии системного

анализа'. fDEfV) — методология функционального моделирования, IDEF1X — методология ин-

формационного моделирования, IDEFO/CPN — методология перехода к динамическому модели-

рованию.

На основе той же методологии Design/IDEF разрабатывается модель вероятного нарушите-

ля информационной безопасности и модель политики безопасности в виде эталонной безопасной

информационной технологии. Далее на основе трех моделей, выполненных на единой методоло-

гической базе, осуществляется оценка существующей информационной технологии на предмет ее

информационной безопасности.

На данном этапе целесообразно использовать следующие инструментальные средства ме-

тодологии системного анализа IDEF: модуль реализации стоимостного анализа АВС (Activity Base

Costing) — EasyABC и модуль динамического моделирования посредством раскрашенных сетей

Петри CPN (Colored Petri Nets) — Design/CPN. После выполнения процедуры анализа уровня

безопасности существующей информационной технологии разрабатывается оптимальная модель

информационных процессов предприятия. Эта модель и выступает в виде основы для разработки

соответствующих организационных решений по совершенствованию правил безопасности.

Очевидно, что совершенствование основного технологического цикла обработки инфор-

мации по критерию оптимизации безопасности не должно наносить ущерб другим показателям

технологического цикла, включая производительность аппаратных и программных средств.

**Таблица 1. Сравнительные характеристики наиболее популярных в настоящий момент (2000/2010 гг.) средств CASE-анализа**