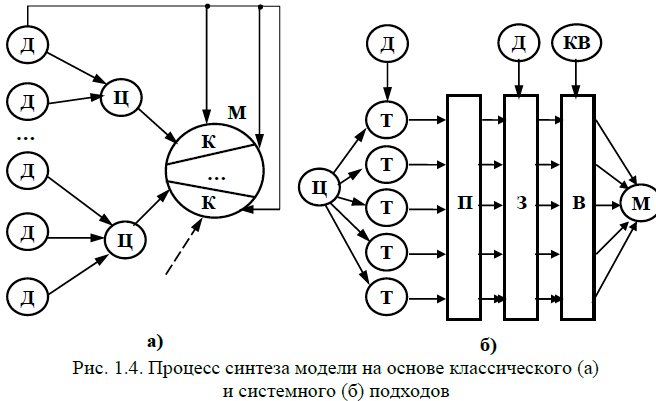
**4.Подходы к исследованию систем**.

***структуры системы*** – совокупности связей между элементами системы, отражающих их взаимодействие. Структура системы может изучаться извне с точки зрения состава отдельных подсистем и отношений между ними, а также изнутри, когда анализируются отдельные свойства, позволяющие системе достигать заданной цели, т.е. когда изучаются функции системы. В соответствии с этим наметился ряд подходов к исследованию структуры системы с ее свойствами, к которым следует прежде всего отнести структурный и функциональный.

При ***структурном подходе*** выявляются состав выделенных элементов системы ***S*** и связи между ними. Совокупность элементов и связи между ними позволяют судить о структуре системы. Последняя в зависимости от цели исследования может быть описана на разных уровнях рассмотрения. Наиболее общее описание структуры – это топологическое описание, позволяющее определить в самых общих понятиях составные части системы и хорошо формализуемое на базе теории графов.

Менее общим является функциональное описание, когда рассматриваются отдельные функции, т.е. алгоритмы поведения системы, и реализуется ***функциональный подход***, оценивающий функции, которые выполняет система, причем под функцией понимается свойство, приводящее к достижению цели. Поскольку функция отображает свойство, а свойство отображает взаимодействие системы ***S*** с внешней средой ***Е***, то свойства могут быть выражены в виде либо некоторых характеристик элементов ***Si(j)*** и подсистем ***Si*** системы, либо системы ***S*** в целом.

При наличии некоторого эталона сравнения можно ввести количественные и качественные характеристики систем. Для количественной характеристики вводятся числа, выражающие отношения между данной характеристикой и эталоном. Качественные характеристики системы находятся, например, с помощью метода экспертных оценок.

Проявление функций системы во времени ***S(t)***, т.е. ***функционирование системы***, означает переход системы из одного состояния в другое, т.е. движение в пространстве состояний ***Z***. При эксплуатации системы ***S*** весьма важно качество ее функционирования, определяемое показателем эффективности и являющееся значением критерия оценки эффективности. Существуют различные подходы к выбору критериев оценки эффективности. Система ***S*** может оцениваться либо совокупностью частных критериев, либо некоторым общим интегральным критерием.

Следует отметить, что создаваемая модель ***M*** с точки зрения системного подхода также является системой, т.е. C:\Users\Lenovo\Pictures\Screenshots\Снимок экрана (6).png, и может рассматриваться по отношению к внешней среде ***Е.***

Простой подход к изучению взаимосвязей между отдельными частями модели предусматривает рассмотрение их как отражение связей между отдельными подсистемами объекта. Такой классический подход может быть использован при создании достаточно простых моделей. Процесс синтеза модели ***M*** на основе классического (индуктивного) подхода представлен на рис. 1.4а

Реальный объект, подлежащий моделированию разбивается на отдельные подсистемы, т.е. выбираются исходные данные ***Д*** для моделирования и ставятся цели ***Ц*** , отображающие отдельные стороны процесса. По отдельной совокупности исходных данных ***Д*** ставится цель моделирования отдельной стороны функционирования системы, на базе этой цели формируется некоторая компонента ***К*** будущей модели. Совокупность компонент объединяется в модель ***M*** .

Можно отметить две отличительные стороны классического подхода: наблюдается движение от частного к общему, создаваемая модель (система) образуется путем суммирования отдельных ее компонент и не учитывается возникновение нового системного эффекта.

Системный подход получил применение в системотехнике в связи с необходимостью исследования больших реальных систем, когда сказалась недостаточность, а иногда ошибочность принятия каких-либо частных решений. На возникновение системного подхода повлияли увеличивающееся количество исходных данных при разработке, необходимость учета сложных стохастических связей в системе и воздействия внешней среды ***Е***. Все это заставило исследователей изучать сложный объект не изолировано, а во взаимодействии с внешней средой, а также в совокупности с другими системами некоторой метасистемы.

Системный подход позволяет решить проблему с учетом всех факторов и возможностей, пропорциональных их значимости, на всех этапах исследования системы ***S*** и построения модели ***M*** . системный подход означает, что каждая система ***S*** является интегрированным целым даже тогда, когда она состоит из отдельных разобщенных подсистем. Таким образом, в основе системного подхода лежит рассмотрение системы как интегрированного целого, причем это рассмотрение при разработке начинается с главного – формулировки цели функционирования. Процесс синтеза модели ***М*** на базе системного подхода условно представлен на рис. 1.4б. На основе исходных данных ***Д*** , которые известны из анализа внешней системы, тех ограничений, которые накладываются на систему сверху либо исходя из возможностей ее реализации, и на основе цели функционирования формулируются исходные требования ***Т*** к модели системы ***S*** . На базе этих требований формируются ориентировочно некоторые подсистемы ***П*** , элементы ***Э*** и осуществляется наиболее сложный этап синтеза – выбор ***В*** составляющих системы, для чего используются специальные критерии выбора ***КВ*** .

**Стадии разработки моделей.**

На стадии ***макропроектирования*** на основе данных о реальной системе ***S*** и внешней среде ***Е*** строится модель внешней среды, выявляются ресурсы и ограничения для построения модели системы, выбирается модель системы и критерии, позволяющие оценить адекватность модели ***М*** реальной системы ***S***. Построив модель системы и модель внешней среды на основе критерия эффективности функционирования системы в процессе моделирования выбирают оптимальную стратегию управления, что позволяет реализовать возможности модели по воспроизведению отдельных сторон функционирования реальной системы ***S*** .

Стадия ***микропроектирования*** в значительной степени зависит от конкретного типа выбранной модели. В случае имитационной модели необходимо обеспечить создание информационного, математического, технического и программного обеспечения системы моделирования. На этой стадии можно установить основные характеристики созданной модели, оценить время работы с ним и затраты ресурсов для получения заданного качества соответствия модели процессу функционирования системы ***S.***

Независимо от типа используемой модели ***М*** при ее построении необходимо руководствоваться рядом принципов системного подхода:1) пропорционально-последовательное продвижение по этапам и направлениям создания модели; 2) согласование информационных, ресурсных, надежностных и других характеристик; 3) правильное соотношение отдельных уровней иерархии в системе моделирования; 4) целостность отдельных обособленных стадий построения модели.

**6.Теория множеств**

**Исходным понятием теории множеств** является само понятие множество, под которым принято понимать некоторую совокупность объектов, хорошо различимых нашей мыслью или интуицией. При этом не делается никаких предположений ни о природе этих объектов, ни о способе их включения в данную совокупность. Отдельные объекты, составляющие то или иное множество, называют элементами данного множества.

В теории множеств множества обозначаются прописными буквами латинского алфавита. При этом отдельные элементы обозначаются строчными буквами, иногда с индексами, которые вносят некоторую упорядоченность в последовательность рассмотрения этих элементов. Принято называть элементы отдельного множества принадлежащими данному множеству.

Следующим важным понятием, которое служит прототипом многих более конкретных терминов при моделировании сложных систем, является понятие подмножества. Для обозначения подмножества используется специальный символ.

Последнее, на что следовало бы обратить внимание при столь кратком знакомстве с основами теории множеств – это на так называемые понятия мощности множества и отношения множеств. Что касается понятия мощности множества, то данный термин важен для анализа кратности связей, поскольку ассоциируется с количеством элементов отдельного множества. В случае конечного множества ситуация очень простая, поскольку мощность конечного множества равна количеству элементов этого множества

Ситуация усложняется, когда рассматриваются бесконечные множества, т. е. множества, не являющиеся конечными.ограничим наше рассмотрение бесконечными множествами счетной мощности. Такими множествами принято считать множества, содержащие бесконечное число элементов, которые, однако, можно перенумеровать натуральными числами 1, 2, 3 и т. д. При этом важно иметь в виду, что достичь последнего элемента при такой нумерации принципиально невозможно, иначе множество окажется конечным.

Отношения множеств, которое часто заменяется терминами связь или соотношение. Данный термин ведет свое происхождение от теории множеств и служит для обозначения любого подмножества упорядоченных кортежей, построенных из элементов некоторых исходных множеств. При этом под кортежем понимается просто набор или список элементов, важно только, чтобы они были упорядочены. Другими словами, если рассматривать первый элемент кортежа, то он всегда будет первым в списке элементов, второй элемент кортежа будет вторым элементом в списке и т. д.

Хотя и существует некоторая неоднозначность в принятых обозначениях, кортеж из двух элементов удобно обозначать как <a1, a2>, из трех элементов – <a1, a2, a3> и т. д. При этом отдельные элементы могут принадлежать как одному и тому же множеству, так и различным множествам. Важно иметь в виду, что порядок выбора элементов для построения кортежей строго фиксирован для конкретной задачи. Речь идет о том, что первый элемент всегда выбирается из первого множества, второй – из второго, и т. д:

Отношение в этом случае будет характеризовать способ или семантику выбора отдельных элементов из одного или нескольких множеств для подобного упорядоченного списка. В этом смысле взаимосвязь является частным случаем отношения, о чем будет сказано в последующем.

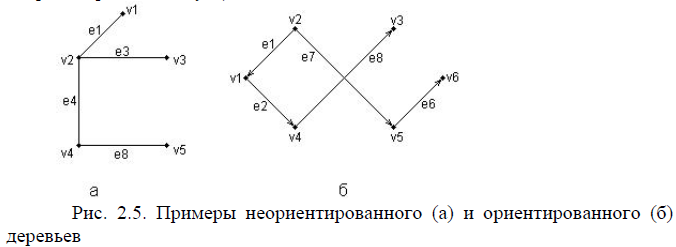
**7.Теория графов**

Графом называется совокупность двух множеств: множества точек или вершин и множества соединяющих их линий или ребер. Формально граф задается в виде двух множеств: G=(V, Е), где V={v1v2, ..., vn} – множество вершин графа, а Е={е1, е2, ..., еm} – множество ребер графа. Следует заметить, в общем случае не все вершины графа могут соединяться между собой, что ставит в соответствие каждому графу некоторое бинарное отношение PQ, состоящее из всех пар вида <vi, vj>, где vi, vj = V. При этом пара <vi, vj> и, соответственно, пара <vj, vi> принадлежат отношению PG в том и только в том случае, если вершины vi и vj соединяются в графе G некоторым ребром ek=Е.

Графы широко применяются для представления различной информации о структуре систем и процессов. Примерами подобных графических моделей могут служить: схемы автомобильных дорог; схемы телекоммуникаций

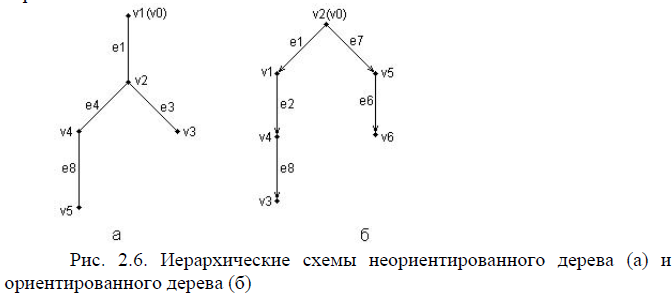
Важными понятиями теории графов являются понятия маршрута и пути, которые ассоциируются с последовательным перемещением от вершины к вершине по соединяющим их ребрам или дугам. Для неориентированного графа маршрут определяется как конечная или бесконечная упорядоченная последовательность ребер S=<, esl, es2, ..., esk>>, таких, что каждые два соседних ребра имеют общую вершину. Нас будут интересовать только конечные маршруты S=<es1, es2, ..., esk>, т. е. такие маршруты, которые состоят из конечного числа ребер. При этом ребро esl принято считать началом маршрута S, а ребро esk – концом маршрута S. Для ориентированного графа соответствующая последовательность дуг S=<es1, es2, ..., esk> называется ориентированным маршрутом, если две соседние дуги имеют общую вершину, которая является концом предыдущей и началом последующей дуги.

Если в ориентированном маршруте не повторяются ни ребра, ни вершины, как в случае S1 и S2, то такой ориентированный маршрут называется путем. Последнее понятие также иногда применяется для обозначения простой цепи в неориентированных графах и для определения специального класса графов, так называемых деревьев. В общем случае деревья служат для графического представления иерархических структур или иерархий, занимающих важное место в ООАП.

Деревом в теории графов называется такой граф D=<V, E>, между любыми двумя вершинами которого существует единственная простая цепь, т. е. неориентированный маршрут, у которого вершины и ребра не повторяются. Применительно к ориентированным графам соответствующее определение является более сложным, поскольку основывается на выделении некоторой специальной вершины v0, которая получила специальное название корневой вершины или просто – корня. В этом случае ориентированный граф D=<V, Е> называется ориентированным деревом или сокращенно – деревом, если между корнем дерева v0 и любой другой вершиной существует единственный путь, берущий начало в v0. Ниже представлены два примера деревьев: неориентированного дерева (рис. 2.5, а) и ориентированного дерева (рис. 2.5, б).

В случае неориентированного дерева (рис. 2.5, а) любая из вершин графа может быть выбрана в качестве корня. Подобный выбор определяется специфическими особенностями решаемой задачи. Так, вершина v1 может рассматриваться в качестве корня неориентированного дерева, поскольку между v1 и любой другой вершиной дерева всегда существует единственная простая цепь по определению (или, что менее строго, единственный неориентированный путь).

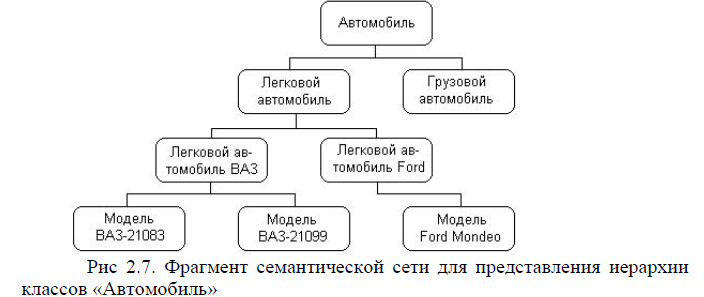
Для случая ориентированного дерева (рис. 2.5, б) вершина v2 является единственным его корнем и имеет специальное обозначение v0. Единственность корня в ориентированном дереве следует из того факта, что ориентированный путь всегда имеет единственную вершину, которая является его началом. Поскольку в теории графов имеет значение только наличие или отсутствие связей между отдельными вершинами, деревья, как правило, изображаются специальным образом в виде иерархической структуры. При этом корень дерева изображается самой верхней вершиной в данной иерархии. Далее следуют вершины уровня 1, которые связаны с корнем одним ребром или одной дугой. Следующий уровень будет иметь номер 2, поскольку соответствующие вершины должны быть связаны с корнем двумя последовательными ребрами или дугами. Процесс построения иерархического дерева продолжается до тех пор, пока не будут рассмотрены вершины, которые не связаны с другими вершинами, кроме рассмотренных, или из которых не выходит ни одна дуга. В этом случае самые нижние вершины иногда называют листьями дерева. Важно иметь в виду, что в теории графов дерево «растет» вниз, а не вверх, как в реальной жизни.

Изображенные выше деревья (рис. 2.5) можно преобразовать к виду иерархий. Например, неориентированное дерево (рис. 2.5, а) может быть представлено в виде иерархического дерева следующим образом (рис. 2.6, а). В этом случае корнем иерархии является вершина v1. Ориентированное дерево (рис. 2.5, б) также может быть изображено в форме иерархического дерева (рис. 2.6, б), однако такое представление является единственным. листьями данного неориентированного дерева являются вершины v3 и v5.

**8.Семантические сети.**

В общем случае под семантической сетью понимают некоторый граф Gs= =(Vs, Es), в котором множество вершин Vs и множество ребер Es разделены на отдельные типы, обладающие специальной семантикой, характерной для той или иной предметной области. В данной ситуации множество вершин может соответствовать объектам или сущностям рассматриваемой предметной области и иметь вместо номеров вершин соответствующие явные имена этих сущностей. Подобные имена должны позволять однозначно идентифицировать соответствующие объекты, при этом общих формальных правил записи имен не существует. Множество ребер также делится на различные типы, которые соответствуют различным видам связей между сущностями рассматриваемой предметной области.

Важной особенностью семантических сетей является разработка специальных графических обозначений для представления отдельных типов вершин и ребер. При этом вершины не изображаются, как ранее – точками, а имеют вид прямоугольников, овалов, окружностей и других геометрических фигур, конкретный вид которых определяет тот или иной тип сущностей предметной области. Более разнообразным становится и изображение ребер, приобретающих вид различных линий со стрелками или без них, а также имеющих специальные обозначения или украшения в виде условных значков. Соответствующая система обозначений, предназначенная для представления информации об отдельных аспектах моделируемой предметной области, получила название графической нотации.

В качестве конкретного варианта представления информации в виде семантической сети рассмотрим дальнейшее развитие примера с классом «Автомобиль» из главы 1

**9 Методология структурного анализа и моделирования.**

Со временем разработка больших программ превратилась в серьезную проблему и потребовала их разбиения на более мелкие фрагменты. Основой для такого разбиения как раз и стала процедурная декомпозиция, при которой отдельные части программы или модули представляли собой совокупность процедур для решения некоторой совокупности задач. Главная особенность процедурного программирования заключается в том, что программа' всегда имеет начало во времени или начальную процедуру (начальный блок) и окончание (конечный блок). При этом вся программа может быть представлена визуально в виде направленной последовательности графических примитивов или блоков

Важным свойством таких программ является необходимость завершения всех действий предшествующей процедуры для начала действий последующей процедуры. Изменение порядка выполнения этих действий даже в пределах одной процедуры потребовало включения в языки программирования специальных условных операторов типа if-then-eise и Goto для реализации ветвления вычислительного процесса в зависимости от промежуточных результатов решения задачи.

Рассмотренные идеи способствовали становлению некоторой системы взглядов на процесс разработки программ и написания программных кодов, которая Получила название методологии структурного программирования. Основой данной методологии является процедурная декомпозиция программной системы и организация отдельных модулей в виде совокупности выполняемых процедур. В рамках данной методологии получило развитие нисходящее проектирование программ или программирование «сверху-вниз». Как вспомогательное средство структуризации программного кода было рекомендовано использование отступов в начале каждой строки, которые должны выделять вложенные циклы и условные операторы. Все это призвано способствовать пониманию или читабельности самой программы. Данное правило со временем было реализовано в современных инструментариях разработки программ. В этот период основным показателем сложности разработки программ считали ее размер. Вполне серьезно обсуждались такие оценки сложности программ, как количество строк программного кода. А профессионализм программиста напрямую связывался с количеством строк программного кода, который он мог написать и отладить в течение, скажем, месяца.

**10. Методология объектно-ориентированного анализа и проектирования**

Для выделения или идентификации компонентов предметной области было предложено несколько способов и правил. Сам этот процесс получил название концептуализации предметной области. При этом под компонентой понимают некоторую абстрактную единицу, которая обладает функциональностью, т. е. может выполнять определенные действия, связанные с решением поставленных задач. На предварительном этапе концептуализации рекомендуется использовать так называемые CRC-карточки (Component, Responsibility, Collaborator– компонента, обязанность, сотрудники) [1]. Для каждой выделенной компоненты предметной области разрабатывается собственная CRC-карточка (рис. 1.6).

Появление методологии ООАП потребовало, с одной стороны, разработки различных средств концептуализации предметной области, а с другой – соответствующих специалистов, которые владели бы этой методологией. На данном этапе появляется относительно новый тип специалиста, который получил название аналитика или архитектора. Наряду со специалистами по предметной области аналитик участвует в построении концептуальной схемы будущей программы, которая затем преобразуется программистами в код. При этом отдельные компоненты выбираются таким образом, чтобы при последующей разработке их было удобно представить в форме классов и объектов. В этом случае немаловажное значение приобретает и сам язык представления информации о концептуальной схеме предметной области.

Разделение процесса разработки сложных программных приложений на отдельные этапы способствовало становлению концепции жизненного цикла программы. Под жизненным циклом (ЖЦ) программы понимают совокупность взаимосвязанных и следующих во времени этапов, начиная от разработки требований к ней и заканчивая полным отказом от ее использования. Согласно принятым взглядам ЖЦ программы состоит из следующих этапов:

• Анализа предметной области и формулировки требований к программе

• Проектирования структуры программы

• Реализации программы в кодах (собственно программирования)

• Внедрения программы

• Сопровождения программы

• Отказа от использования программы

На этапе анализа предметной области и формулировки требований осуществляется определение функций, которые должна выполнять разрабатываемая программа, а также концептуализация предметной области. Эту работу выполняют аналитики совместно со специалистами предметной области. Результатом данного этапа должна являться некоторая концептуальная схема, содержащая описание основных компонентов и тех функций, которые они должны выполнять.

Этап проектирования структуры программы заключается в разработке детальной схемы будущей программы, на которой указываются классы, их свойства и методы, а также различные взаимосвязи между ними. Как правило, на этом этапе могут участвовать в работе аналитики, архитекторы и отдельные квалифицированные программисты. Результатом данного этапа должна стать детализированная схема программы, на которой указываются все классы и взаимосвязи между ними в процессе функционирования программы

Этапы внедрения и сопровождения программы связаны с необходимостью настройки и конфигурирования среды программы, а также с устранением возникших в процессе ее использования ошибок. Иногда в качестве отдельного этапа выделяют тестирование программы, под которым понимают проверку работоспособности программы на некоторой совокупности исходных данных или при некоторых специальных режимах эксплуатации. Результатом этих этапов является повышение надежности Программного приложения, исключающего возникновение критических ситуаций или нанесение ущерба компании, использующей данное приложение.

**11. Методология системного анализа и системного моделирования**

Системы, под которой понимается совокупность объектов, компонентов или элементов произвольной природы, образующих некоторую целостность. Определяющей предпосылкой выделения некоторой совокупности как системы является возникновение у нее новых свойств, которых не имеют составляющие ее элементы. Примеров систем можно привести достаточно много – это персональный компьютер, автомобиль, человек, биосфера, программа и др. Более ортодоксальная точка зрения предполагает, что все окружающие нас предметы являются системами.

Важнейшими характеристиками любой системы являются ее структура и процесс функционирования. Под структурой системы понимают устойчивую во времени совокупность взаимосвязей между ее элементами или компонентами. Именно структура связывает воедино все элементы и препятствует распаду системы на отдельные компоненты. Структура системы может отражать самые различные взаимосвязи, в том числе и вложенность элементов одной системы в другую. В этом случае принято называть более мелкую или вложенную систему подсистемой, а более крупную – метасистемой.

Процесс функционирования системы тесно связан с изменением ее свойств или поведения во времени. При этом важной характеристикой системы является ее состояние, под которым понимается совокупность свойств или признаков, которые в каждый момент времени отражают наиболее существенные особенности поведения системы.

Процесс функционирования системы отражает поведение системы во времени и может быть представлен как последовательное изменение ее состояний: Если система изменяет одно свое состояние на другое, то принято говорить, что система переходит из одного состояния в другое. Совокупность признаков или условий изменения состояний системы в этом случае называется переходом. Для системы с дискретными состояниями процесс функционирования может быть представлен в виде последовательности состояний с соответствующими переходами

Под моделью будем понимать некоторое представление о системе, отражающее наиболее существенные закономерности ее структуры и процесса функционирования и зафиксированное на некотором языке или в другой форме.

Общим свойством всех моделей является их подобие оригинальной системе или системе-оригиналу. Важность построения моделей заключается в возможности их использования для получения информации о свойствах или поведении системы-оригинала. При этом процесс построения и последующего применения моделей для получения информации о системе-оригинале получил название моделирование.

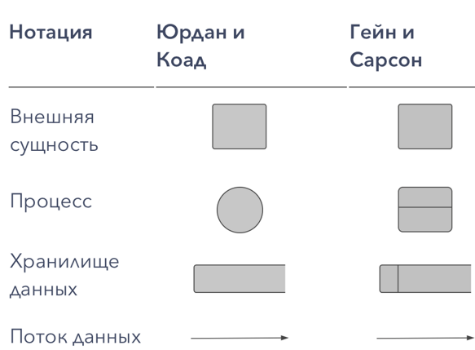
Наиболее общей моделью системы является так называемая модель «черного ящика». В этом случае система представляется в виде прямоугольника, внутреннее устройство которого скрыто от аналитика или неизвестно. Однако система не является полностью изолированной от внешней среды, поскольку последняя оказывает на систему некоторые информационные или материальные воздействия. Такие воздействия получили название входных воздействий. В свою очередь, система также оказывает на среду или другие системы определенные информационные или материальные воздействия, которые получили название выходных воздействий.

Ценность моделей, подобных модели «черного ящика», весьма условна. Невольно может возникнуть ассоциация с «Черным квадратом». Однако если оценка изобразительных особенностей последнего не входит в задачи системного анализа, то общая модель системы содержит некоторую важную информацию о функциональных особенностях данной системы, которые дают представление о ее поведении. Действительно, кроме самой общей информации о том, на какие воздействия реагирует система, и как проявляется эта реакция на окружающие объекты и системы, другой информации мы получить не можем. В рамках системного анализа разработаны определенные методологические средства, позволяющие выполнить дальнейшую конкретизацию общей модели системы.

Для построения моделей были разработаны достаточно серьезные теоретические методы, основанные на развитии математических и логических средств моделирования, а также предложены различные формальные и графические нотации, отражающие специфику решаемых задач. Важно представлять, что унификация любого языка моделирования тесно связана с методологией системного моделирования, т. е. с системой воззрений и принципов рассмотрения сложных явлений и объектов как моделей сложных систем.

Сложность системы и, соответственно, ее модели может быть рассмотрена с различных точек зрения. Прежде всего, можно выделить сложность структуры системы, которая характеризуется количеством элементов системы и различными типами взаимосвязей между этими элементами. Если количество элементов превышает некоторое пороговое значение, которое не является строго фиксированным, то такая система может быть названа сложной. Например, если программная СУБД насчитывает более 100 отдельных форм ввода и вывода информации, то многие программисты сочтут ее сложной. Транспортная система современных мегаполисов также может служить примером сложной системы.

Вторым аспектом сложности является сложность процесса функционирования системы. Это может быть связано как с непредсказуемым характером поведения системы, так и невозможностью формального представления правил преобразования входных воздействий в выходные. В качестве примеров сложных программных систем можно привести современные операционные системы, которым присущи черты сложности как структуры, так и поведения.

**15.Моделирование потоков данных с использованием нотации DFD.**

**Диаграмма потоков данных (DFD Data Float Diagram)** описывает асинхронный процесс преобразования информации от её входа в систему до выдачи пользователю**. Главная цель** построения диаграммы – продемонстрировать как каждый процесс преобразует входные данные в выходные, а также выявить отношения между этими процессами.DFD модели могут быть использованы в дополнение к моделям IDEF0 для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации.

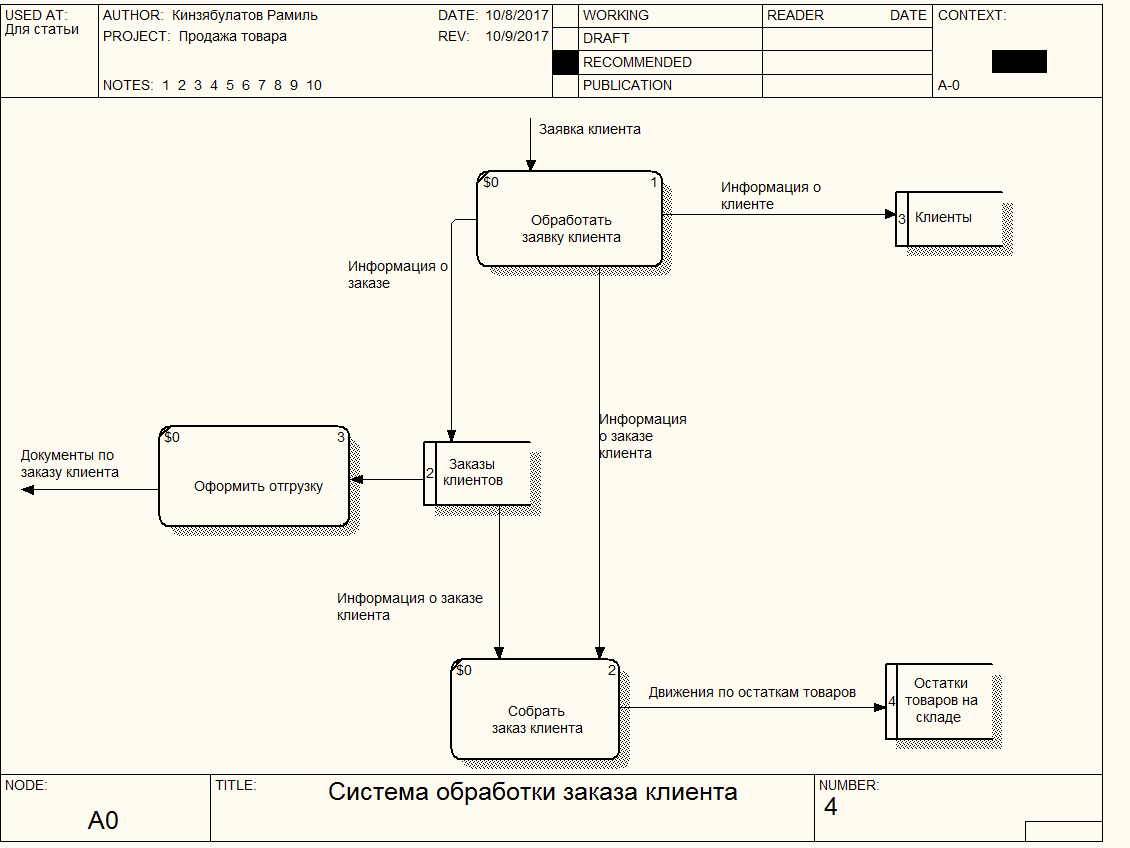
Применяется этот вид нотации в случае, когда требуется описание системы как хранилища данных. Т.е. нотация должна наглядно ответить на вопросы: Из чего состоит информационная система? 2.Что нужно, чтобы обработать информацию?

Исторически существует два варианта синтаксиса элементов данной диаграммы.

**Процесс (англ. Process)**, т.е. функция или последовательность действий, которые нужно предпринять, чтобы данные были обработаны. Это может быть создание заказа, регистрация клиента и т.д. В названиях процессов принято использовать глаголы, т.е. «Создать клиента» (а не «создание клиента») или «обработать заказ» (а не «проведение заказа»).

**Внешние сущности (англ. External Entity)**

Это любые объекты, которые не входят в саму систему, но являются для нее источником информации либо получателями какой-либо информации из системы после обработки данных. Это может быть человек, внешняя система, какие-либо носители информации и хранилища данных.

**Хранилище данных (англ. Data store)**.

Внутреннее хранилище данных для процессов в системе. Поступившие данные перед обработкой и результат после обработки, а также промежуточные значения должны где-то храниться. Это и есть базы данных, таблицы или любой другой вариант организации и хранения данных. Здесь будут храниться данные о клиентах, заявки клиентов, расходные накладные и любые другие данные, которые поступили в систему или являются результатом обработки процессов

**Поток данных (англ. Data flow)**.

В нотации отображается в виде стрелок, которые показывают, какая информация входит, а какая исходит из того или иного блока на диаграмме.

**Правила построения диаграммы потоков данных**

* Каждый процесс должен иметь хотя бы один вход и один выход. Смысл процессов здесь заключается в обработке данных, а потому процесс должен получить данные (входящая стрелка) и отдать куда-то после обработки (исходящая стрелка);
* Процесс обработки данных должен иметь внешнюю входящую стрелку (данные от внешней сущности). Для того, чтобы любой подобный процесс начал работать, мало использовать данные из хранилища, должна поступить новая информация для последующей обработки;
* Стрелки не могут связывать напрямую хранилища данных, все связи идут через процессы. Нет смысла просто перемещать данные из одного места в другое, а именно так читается прямая связь двух хранилищ стрелкой. Данные поступают для того, чтобы производились какие-то действия, в нашем примере – осуществлялся процесс продажи. А это возможно только посредством обработки (процесса);
* Все процессы должны быть связаны либо с другими процессами, либо с другими хранилищами данных. Процессы не существуют сами по себе, а потому результат должен куда-то передаваться;

**26.Инструментальные средства, используемые для моделирования.**

Инструментальные средства, предназначенные для моделирования информационных систем, могут быть отнесены к одной из следующих категорий:

- локальные, поддерживающие один тип моделей и методов (Design/IDEF, ProCap, S-Designor, «CASE. Аналитик»);

- малые интегрированные, поддерживающие несколько типов (до 5) моделей и методов (ERwin, BPwin);

- средние интегрированные, поддерживающие 5-15 типов моделей и методов (Rational Rose, Paradigm Plus, Designer/2000);

- крупные интегрированные, поддерживающие более 15 типов моделей и методов (ARIS Toolset).

Локальные средства моделирования могут быть использованы только на концептуальном уровне для предварительного анализа или как средство демонстрации заказчику общих предложений по будущему проекту. Задача комплексного анализа системы локальными средствами не может быть решена.

BPwin поддерживает 3 методологии моделирования (IDEFO, IDEF3 и DFD) и обеспечивает интеграцию моделей трех типов без экспорта или импорта данных. Интеграция выполняется как слиянием нескольких моделей, так и переключением на различные методологии в процессе разработки отдельных диаграмм модели. Предусмотрено расширение возможностей анализа систем как в самом пакете BPwin (функционально-стоимостный анализ), так и с помощью экспорта данных в другие пакеты. В ERwin поддерживается несколько разновидностей методологии информационного моделирования, основанной на ER-диаграммах («сущность-связь»).

При создании среднихинтегрированных средств моделирования в них были заложены требования комплексного использования различных методов и типов моделей. Продукты средней категории имеют единую среду для разработки всех поддерживаемых типов моделей, что позволяет применять одни и те же объекты в разных моделях. Так, например, последние версии Rational Roseпозволяют строить восемь типов диаграмм UML:

- диаграммы прецедентов (Use Cases Diagrams);

- диаграммы классов (Class Diagrams);

- диаграммы последовательности (Sequence Diagrams);

- диаграммы сотрудничества (Collaboration Diagrams);

- диаграммы состояний (State Diagrams);

- диаграммы действий (Activity Diagrams);

- компонентные диаграммы (Component Diagrams);

- диаграммы развертывания (Deployment Diagram).

Пакет Paradigm Plusориентирован на методологию OOCL (Object Oriented Change and Learning) и компонентную технологию проектирования и разработки, обеспечивая поддержку диаграмм различных методов (UML, CLIPP, TeamFusion, ОМТ, Booch, OOCL, Martin/Odell, Shlaer/ Mellor, Coad/Yourdon). В состав Designer/2000 входят Process Modeller (разработки моделей процессов) и System Modeller (модели иерархии функций (Function Hierarchy Diagrammer), модели потоков данных (Dataflow Diagrammer) и модели типа «сущность-отношение» (Entity Relationship Diagrammer)).

К крупным интегрированным средствам моделирования относят систему, предназначенную для проектирования крупных АСУП класса ERP. Это семейство ARIS(ARIS Toolset, ARIS Easy Design) от компании IDS Sheer AG. Принадлежность к категории ERP для средства моделирования означает, что оно предназначено для выполнения комплексного анализа на всех стадиях разработки АСУП класса ERP.

ARIS обеспечивает четыре различных «взгляда» на моделирование и анализ:

- Процессы;

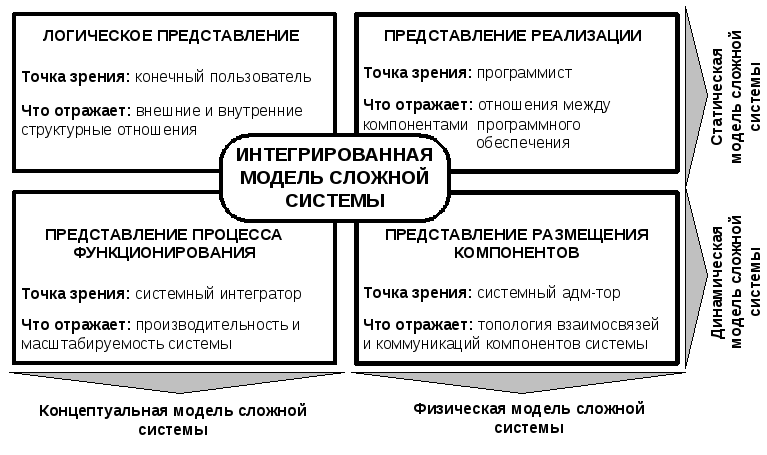
- Функции;

- Данные;

- Организация.

Для каждого «взгляда» поддерживаются три уровня анализа (требования, спецификации, внедрение). Каждый из уровней анализа состоит из своего комплекта моделей различных типов, в том числе диаграмм UML, диаграмм SAP/R3 и др. Каждый объект моделей ARIS имеет множество атрибутов, которые позволяют контролировать процесс разработки моделей, определять условия для выполнения функционально-стоимостного анализа, имитационного моделирования, взаимодействия с системами документооборота и т.д.

«Взгляды» ARIS являются «комнатами», из которых состоит так называемый домик ARIS. Этот домик используется и в процессе моделирования для выбора комплекта моделей, соответствующего «взгляду» и уровню анализа. Главная «комната» домика ARIS - это Процессы, для моделирования которых предназначено 57 типов моделей из 85.

**25.Взаимосвязь между различными видами диаграмм UML.**

**1. Принцип абстрагирования**, который предписывает включать в модель только те аспекты проектируемой системы, которые имеют непосредственное отношение к выполнению системой своих функций или своего целевого назначения.

**2. Принцип многомодельности**, который сводится к утверждению о том, что никакая отдельно взятая модель не может с достаточной степенью адекватности описать различные аспекты сложной системы.

**3.Принцип иерархического построения** моделей сложных систем, который предписывает рассматривать процесс построения модели на разных уровнях абстрагирования или детализации в рамках фиксированных представлений.