1.1.1.1 Современные интеллектуальные системы: концептуальные основы и стандарты в области создания информационных систем.

Современные интеллектуальные информационные системы используются во всех сферах и бизнес-процессах человеческой деятельности. Поэтому исследования по использованию интеллектуальных информационных систем можно рассматривать с разных точек зрения.

Применительно к каждой области и бизнес-процессу используются разные подходы и разные типы систем.

Одним из основных направлений развития интеллектуальных информационных систем является разработка систем, основанных на знаниях. Основной целью построения таких систем является выявление, изучение и применение знаний высококвалифицированных экспертов для решения сложных задач, которые возникаю при решении практических задач. Когда проектируются системы, основа которых-знания, необходимо использовать именно те знания, которые были накоплены экспертами. Эти знания необходимо привести к такому виду, чтобы их можно было использовать как шаблоны и правила для решения поставленных конкретных практических задач.

Такой подход симулирует работу человеческого мозга, когда он анализирует информацию с целью дальнейшей с ней работы. Причем здесь можно использовать данные любого вида. Структурированные, формализованные, неструктурированные, неформализованные.

Подобно работе человеческого мозга интеллектуальная информационная система выделяет из общей массы знаний, только те, которые необходимы для использования в конкретной задаче. Систематизирует эти знания, структурирует, формализует. Знания формируют базы данных. Так как знания накоплены экспертами и итоговое решение также принимается экспертом такие интеллектуальные системы получили название экспертных.

Экспертные системы помимо анализа знаний способны переводить знания с естественного человеческого языка на машинный и наоборот. А также на разные естественные языки. Таким образом достигается универсальность интеллектуальной системы. И удобство работы с ней любому пользователю. Информация легко и быстро извлекается. Рутинные виды работ перерабатываются в короткие сроки, без использования человеческого труда, и отбирается только необходимая информация для решения конкретной прикладной задачи. Причем с одинаковой легкостью анализируется и обрабатывается информация на естественном языке, информация с большим количеством научных и технических терминов. И результат выдается в том виде , который требуется пользователю.

Для того чтобы правильно отобрать нужную информацию интеллектуальная экспертная система должна обладать достаточным объемом знаний в разных предметных областях, уметь ориентироваться в сложных

моделях. На основе этих знаний экспертная система осуществляет анализ выделенной информации, ее синтез, поиск готовых типовых решений.

Переработанная информация должна быть предоставлена пользователю на естественном языке в формате диалога. С возможностью дополнительных уточнений от пользователя и преобразовании решения запроса с учетом этих уточнений.

Помимо работы с разными типами текстовой информации экспертная интеллектуальная система должна уметь распознавать и речь на любом естественном языке и генерировать ее в текст требуемого формата.

Необходимость навыка распознавания человеческой речи очевидна. Эта возможность значительно ускоряет процесс ввода информацию в базу знаний экспертной системы и следовательно снижение трудовой нагрузки на персонал и снижение затрат на количество персонала. Недостаток этого метода ввода информации также очевиден. Не всегда можно точно и правильно распознать человеческую речь, за счет нечеткой речи, удаленности пользователя, акцента и прочих моментах.

Следующий тип информацию, которую экспертная интеллектуальная система должна уметь распознавать и обрабатывать визуальной информации.

Визуальная информация также может быть обработана, проанализирована и преобразована в тот вид, который требуется.

Причем для обработки визуальная информация может быть преобразована в текст, например, для анализа, а затем результат снова представлен в графическом в виде, для предоставления пользователю в качестве результата выполнения задачи.

Кроме работы с разными видами информации, с разными способами ее обработки экспертная интеллектуальная система должна обладать способностью обучать саму себя. На основе уже полученных результатов. Система сама может создавать типовые ответы на схожие запросы, уметь находить дополнительную информацию для полного ответа на поставленную задачу.

В ходе работы накапливается огромное количество знаний, огромное количество закономерностей, алгоритмов, методов, обобщений, синтеза, объединений, классификаций и так далее.

В экспертных интеллектуальных системах используется методика добычи данных . В международной терминологии Data Mining. А также используется система поиска закономерностей. В международной терминологии Knowledge Discovery.

Одно из новых возможностей экспертных интеллектуальных систем, которое в настоящее время используется повсеместно это распознавание образов. Хотя активное использование такой возможности стало распространяться не так давно именно распознавание образов было в основе идеи создания интеллектуальных систем.

Распознавание образов происходит с использованием математических расчетов параметров образов и сравнения их с информацией, которая находится в багаже знаний интеллектуальной системы.

Активно использующейся и активно развивающейся возможность интеллектуальной системы являются игры.

Игры используются для обучения, развития, творческой реализации, например в виде создания музыкальных произведений. Компьютерные игры, которые используются для обучения , начиная с самого раннего возраста. В эпоху стремительного развития дистанционного образования и обучения это возможность интеллектуальной системы особенно актуальна.

Для того, чтобы все возможности интеллектуальных средств были реализованы используется различное программное обеспечение, которое должно включать те составляющие, которые необходимы для конкретной интеллектуальной системы.

Это могут быть языки программирования, ориентированные на обработку текстовой информации, логические языки программирования, языки представления знаний. Также необходима подходящая программная среда с набором инструментальных средств. Для экспертных систем необходима оболочка. В этой оболочке должна быть возможность разработки экспертных систем для прикладных задач по типу конструктора. Необходимо наличие достаточного набора инструментов для создания экспертной системы без навыков программирования

Очевидно, что для реализации интеллектуальной системы необходима мощная компьютерная техника. И даже не всегда так важна мощность, как архитектура компьютера. Здесь нужно соблюсти баланс архитектуры компьютера, совместимостью его с массово используемыми, стоимостью, соизмеримой со стоимость разработки и экономического эффекта интеллектуальной системы.

Ну и безусловно венцом интеллектуальных систем являются роботы. Робототехника развивается с начала создания компьютерной техники и не прекращалась ни на минуту. Роботов в нашей жизни становится все больше и больше. Интеллектуальные роботы прочно вошли не только в профессиональную деятельность людей , но и в бытовые нужды. При этот сам принцип робота, который управляется человеком с использованием разных систем манипулирования. Эта методика использовалась с первого поколения компьютерной и робототехники и остается такой и по сей день.

Перспективы развития робототехники это разработка интеллектуальных роботов на основе искусственного интеллекта, которые смогут выполнять свои функциональные обязанности без участия человека, который будет управлять каждым действием и ставит конкретную задачу.

Низкая скорость развития таких роботов связана с недостаточной изученностью и недостаточным опытом в возможных побочных эффектах, сбоях, утечке информации, неконтролируемости действий роботов в нестандартных ситуациях.

Этот вопрос решится со временем. Когда будет накоплен достаточный опыт и достаточное количество испытаний.

Интеллектуальные системы. Также как и любая другая структурированная информация требует классификации для более эффективного использования.

Так как работа интеллектуальных систем основана на работе с информацией. Информация эта анализируется и отбирается для использования конкретных прикладных задач. И для их решения используются различные алгоритмы. Отталкиваясь от этого можно выявить признаки интеллектуальных систем, которые можно применять для их классификации.

Признаки интеллектуальных систем:

1. Необходимо наличие развитых и постоянно совершенствующихся коммуникационных возможностей. Для ведения диалога системы с пользователем.
2. Большинство данных, которые используются в интеллектуальных системах плохо формализованы и плохо поддаются формализации в принципе.
3. Интеллектуальная система обязательно и эффективно должна уметь самообучаться.
4. Интеллектуальная система обязательно долдна уметь адаптироваться ко всем обстоятельствам. К типам техники, языку пользователя. Качеству графической информации. Разного уровня детализации ответа, требуемого на поставленную задачу и многому другому.

Учитывая указанные признаки можно распределить интеллектуальные системы по классам, соответствующим этим признакам.

Безусловно, любая интеллектуальная система должна обладать всеми признаками. Но обязательно какой то признак будет преобладать. Это не говорит о качестве интеллектуальной системе. Это зависит от назначения самой системы и задач, которые она призвана решать.

На основе указанной методологии классификации можно выделить системы с использованием искусственного интеллекта. Для таких систем обязательны все четыре признака.

Можно отдельно выделить класс систем с интеллектуальным интерфейсом.

Такие системы при работе с базами данных могут не просто выдавать информацию, извлеченную из базы данных, а могут добывать информацию которая получается путем анализа хранящейся в базе информации.

Для доступа к интеллектуальным базам данных необходим хорошо продуманный интерфейс на естественном языке. Обязательно наличие контекстного поиска информации, использование возможностей распознавания

голоса для ввода запроса и команд, адаптация системы к вводу текстовой или голосовой информации на различных языках. Для реализации этой задачи должен присутствовать в механизме интеллектуальной системы мощный синтаксический, семантический, морфологический аппарат, аппарат анализа в этом контексте. Очень важно правильно понять смысл поставленной задачи, правильность понимания смысла запроса, корректность выдачи ответа. То есть обеспечения коммуникации между интеллектуальной системой и пользователем может столкнуться с непониманием, аналогичным разговору двух людей, говорящих на разных языках.

Цифровая информация, которую интеллектуальная система сгенерировала в качестве ответа на запрос необходимо преобразовать в тот вид, чтобы пользователь получил именно то, что он ожидает получить и именно в том виде, в котором он ожидает получить ответ на запрос.

Для возможности удобного и эффективного поиска по базе данных целесообразно использовать гипертекстовый подход к поиску по ключевым словам. Для корректности работы необходимо очень тщательная проработка списков ключевых слов. Подборка вариантов произношения (в случае голосового ввода) и вариантов написания каждого ключевого слова.

Такие гипертекстовые системы также носят название интеллектуальных. Помимо текста такие системы должны обладать способностью искать информацию не только по текстовым ключевым словам, но и по графической информации, аудио информации, видео информации. То есть мы можем дать таким системам название мультимедийные интеллектуальные гипертекстовые системы.

Современная интеллектуальная система должна уметь понять запрос пользователя и затем дополнить запрос контекстной информации и уже именно по ней осуществлять поиск ответа на запрос или решение задачи.

Пользователь имеет возможность описать запрос в свободной форме в любом виде. А уже интеллектуальная система формирует из этой информации подробный запрос с дополненными уточнениями. Такие системы можно условно назвать системы контекстной помощи.

Иногда запрос от пользователя можно представить только в графическом виде. Интеллектуальная система должна обладать способностью работать с графическими образами аналогично работе с текстом. Сравнивать с имеющимися образами в базе, дополнять, анализировать и выдавать ответ также в виде графических образов. В итоге пользователь нет необходимости описывать ситуацию словами. Это безусловно повышает скорость получения результата. Экономит время и повышает эффективность работы. Такие системы особенно важны для запросов в системах оперативного управления. Где необходимо решение сиюминутной задачи. Например в системах обучения летучка на тренажере. Такие системы получили название систем когнитивной графики.

Отдельным классом выделяются экспертные интеллектуальные системы. Основная задача таких систем имитация работы специалиста в конкретной области. То есть эксперта. Такие системы в основном решают советующие услугу пользователю и оказывают помощь в принятии решения.

Экспертные системы так же как и прочие интеллектуальные системы работают с данными со сложной структурой. Не обладающие четкой структурой и неформализируемые.

Такая информация может быть необходима для решения следующих задач:

1. Задачи , которые нельзя представить в числовом виде. Даже путем преобразования
2. Задачи, которые требуют использования данных из той предметной области,, которая недостаточно изучена. Или информация которая является противоречивой, неточной или может быть интерпретирована неоднозначно.
3. Для решения задач с помощью эксперта необходима четко поставлена цель решения задачи или проблемы. Существуют задачи, где нет четкой цели. Где нет однозначного пути разрешения проблемы.
4. Задачи, для решения которых нет типового алгоритма. Нет методик в накопленной базе знаний , которые смогли бы быть решены на их основе.
5. Задачи, для решения которых существует типовой алгоритм, но существуют причины по которым типовой алгоритм нельзя использовать. Например. необходимо нестандартное решение проблемы, необходимо более быстрое решение проблемы, чем предполагаемое алгоритмом, необходим больший объем ресурсов для предлагаемого решения и множество прочих причин.

Чем же отличаются экспертные системы от систем с искусственным интеллектом?

Данные в экспертной системы представлены в символьном виде, тогда как в интеллектуальной системе данные представлены в числовом формате.

В отличие от интеллектуальной системы, в экспертных используется методика обработки запросов на основе процедур логического вывода информации и эвристического способа решения проблемы.

В экспертных системах решаются задачи самой различной этимологии, самого разного характера, но в основном это именно прикладные задачи. Задачи, охватывающие различные бизнес процессы.

В большинстве случаев экспертная система работает не автономно и не заменяет самого эксперта, а скорее является вспомогательным инструментом для человека эксперта в качестве информационно советующей помощи для решения поиска решения задачи.

Где же можно использовать экспертные системы?

Там где необходим опытный помощник, обладающий огромной базой знаний. Например, для работников стажеров, у которых опыт еще невелик и опасность принять неправильное решение особенно велика. А иногда и для придания большей уверенности в своей правоте.

Иногда нужен анализ проблемы с разных сторон, когда даже опытный сотрудник с огромным багажом знаний не может проанализировать ситуацию с различных точек зрения. И здесь также на помощь придет экспертная система.

В случае, если сотруднику эксперту необходима помощь из смежной области, в которой у этого эксперта недостаточной профессиональных компетенций. И снова возникает необходимость привлечения экспертной системы. Это проще, дешевле, быстрее и в конечном итоге эффективнее, чем привлекать человека эксперта из смежной области.

Для эффективного применения экспертных систем их также необходимо классифицировать.

Вот какие признаки мы можем использовать для классификации экспертных систем.

1. Как именно формируется ответ на запрос или решения задачи или проблему. каким способом. Решение экспертной системой может быть принято путем анализа информации, когда решение принимается на основе уже существующих алгоритмов, методов и способов. А также путем синтеза. Когда решение синтезируется путем сбора ответа на запрос из обрывочных фрагментов.
2. Каким образом учитывается вопрос времени формирования запроса и ответа на него. И имеет ли это значение для данной системы. Экспертные системы могут быть статическими. То есть решение ими принимается для задач, где данные не изменяются в процессе решения. А также динамическими. Данные в таких задачах могут в процессе решения задачи изменяться.
3. Какие данные и информацию можно использовать в конкретной экспертной системе. Как для хранения ,так и ввода и вывода запроса и ответа на него. Данные могут носить детерминированный характер или неопределенный. При неопределенном характере информации могут быть пробелы в логической связке информации, нечеткости, неопределенности.
4. Как много источников получения знаний используется в конкретной экспертной системе. Можно использовать как один источник так и бесконечное множество источников.

Таким образом можно выделить следующие классы экспертных систем:

1. Классифицирующий класс экспертных систем
2. Доопределяющий класс экспертных систем
3. Трансформирующий класс экспертных систем
4. Мультиагентный класс экспертных систем

Также выделяют наряду с экспертными самообучающиеся интеллектуальные системы.

Такие системы за свою основу берут накопление жизненных ситуаций из прикладных задач. Их структурирование, классификацию и анализ.

Именно накопление таких знаний составляет базу обучающей системы. Каждый элемент описывается определенным набором признаков.

Чем больше таких знаний накоплено, тем быстрее и эффективнее происходит самообучение интеллектуальных систем.

Есть различные методы обучения интеллектуальных систем. Один из таких методов это метод обучения с учителем. Здесь признаки каждому элементу и примеру прописывает специалист. То есть учитель. При использовании метода обучения системы без учителя интеллектуальная система сама назначает признаки.

В процессе того как система проходит самообучение строятся правила, как именно описывать ситуации, как назначать признаки, как интерпретировать различные ситуации, различные варианты интерпретации применительно к разным прикладным областям.

Самообучающиеся интеллектуальные систем проходящие обучение по методу без учителя имеет несколько существенных недостатков.

Недостатки вполне очевидны.

При недостатке или неполноты знаний и накопленной базы знаний интерпретация может быть неверной и признаки могут быть назначены неправильно. Классификация данных также может быть неверной. И соответственно неверно обучившаяся интеллектуальная система будет выдавать неверный или неоднозначный, нечеткий результат.

Полученный ответ на запрос такой интеллектуальной системы может быть сложен для объяснения и непригоден для дальнейшего использования. То есть ожидания пользователя не выполнены.

Также проблема может быть описана слишком обобщенно. Это может произойти из за малого количества признаков. Такие системы плохо применяемы для узко специализированных предметных областей.

Также для самообучающихся интеллектуальных систем используется метод индукции. Здесь частные примеры обобщаются и выводятся общие классификационные признаки. В процессе самообучения происходит выбор признака, по которому происходит классификация. Затем , на основе этого признака множество примеров разбивается на еще более мелкие множества. Путем сравнения и анализ проверяется к какому классу может принадлежать каждое из множеств. Процесс завершается для тех множеств которые нашли свой класс. Для прочих процесс продолжается с самого начала.

Отдельным типом интеллектуальных систем являются нейронные сети.

Этот тип интеллектуальных систем основан на примерах. Использует большое количество математических методик, алгоритмов. Также такие системы умеют самообучаться.

Нейронные сети могут использоваться для решения сложных задач. Таких, где необходимо анализировать изображения, распознавать образы, прогнозировать.

Нейронная сеть (как это можно предположить и из названия) имитирует модель нервной системы человека, состоящей из нейронов. ЕЕ математическую модель. Как эти нейроны будут соединяться между собой зависит от типа нейронной сети и задач, которые она призвана решать.

Процесс решения задачи или проблемы с помощью нейронной сети схож на работу человеческого мозга при решении поставленной проблемы.

На начальном этапе в систему поступает информация о проблеме, которую необходимо решить.

Далее эта информация сопоставляется и сравнивается с аналогичными ситуациями в базе знаний системы.

Выбирается наиболее подходящий случай.

Этот случай, который оказался наиболее подходящим адаптируется к текущей проблеме.

Полученный вариант решения проверяется насколько он корректен и отвечает поступившему запросу. Всем его требованиям.

Информация по этому решению подробно описывается и сохраняется в базе знания для дальнейшего использования.

Следующий тип интеллектуальных информационных систем это информационные хранилища.

Здесь хранится информация. Которая не изменяется. Эту информацию используют в различных системах поддержки принятия решений.

Данные хранятся уже разделенными на разные предметные области.

Технологии извлечения знаний из хранилищ данных основаны на методах статистического анализа и моделирования, ориентированных на поиск моделей и отношений, скрытых в совокупности данных. Эти модели могут в дальнейшем использоваться для оптимизации деятельности предприятия или фирмы.

Для извлечения значимой информации из хранилищ данных имеются специальные методы . Эти методы уже были описаны выше. OLAP-анализа, Data Mining или Knowledge Discovery. Эти методы основаны на формализованных методах, методам статистического анализа, построения деревьев решения, картах Кохонена, методом ассоциативных правил и т.д.

И еще один тип интеллектуальных систем это адаптивные интеллектуальные системы.

Эти системы решают специфические задачи.

К ним предъявляются определенные требования. Такие как:

- способность показывать знания в каждой предметной области в зависимости от момента времени.

- способность быстро, просто и качественно подстраиваться под изменения любого характера в предметной области.

В процессе разработки адаптивных информационных систем применяется оригинальное или типовое проектирование. Оригинальное проектирование предполагает разработку информационной системы с нуля.

При типовом проектировании осуществляется адаптация типовых разработок к особенностям проблемной области.

Вопросы по теме лекции:

- в чем особенность интеллектуальных информационных систем?
- каковы возможные сферы применения интеллектуальных информационных систем?
- в чем особенность развития интеллектуальных информационных систем на современном этапе?
- какие существуют стандарты, применимые для построения информационных систем?
- какие существуют графические нотации построения информационных систем, в том числе, интеллектуальных информационных систем?
- в чем особенность обучения с учителем? без учителя? обучения с подкреплением?
- метод кластерного анализа может быть отнесен к обучению с учителем или без учителя?
- метод логит-регрессии моет быть отнесен к обучению с учителем или без учителя?
- в чем особенность экспертных систем?
- каков состав экспертной системы?
- какие можно предложить современные направления применения экспертных систем?
- какова роль интеллектуальных информационных систем в развитии цифровой экономики?

Литература и материалы по теме:

Советов, Б.Я. Интеллектуальные системы и технологии: Учебник / Б.Я. Советов. - М.: Академия, 2017. - 192 с.

Перлова, О.Н. Проектирование и разработка информационных систем: Учебник / О.Н. Перлова, О.П. Ляпина, А.В. Гусева. - М.: Academia, 2017. - 416 с.

Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий / Н.В. Федоров. - М.: МГИУ, 2008. - 280 с.

1.1.1.2 Современные интеллектуальные системы: проектирование и развертывание ИС, поддержка жизненного цикла ПО.

Проектирование любой информационной системы, в том числе интеллектуальной невозможно без составления плана и визуализации этапов развития проектируемой системы.

В рамках технического сленга такой план развития принято называть жизненным циклом. А в рамках международного технического сленга - Software development lifecycle или просто аббревиатура SDLC.

Жизненный цикл имеет любой проект, любая система и любая интеллектуальная система в нашем случае. Для программного обеспечения в целом и для интеллектуальной системы в частности жизненный цикл будет иметь немного отличные этапы от каких либо других проектов.

Для визуализации этапов жизненного цикла интеллектуальной системы можно использовать различные модели. Выбор модели будет зависеть от типа системы, масштабов, технического задания и множества особенностей конкретной интеллектуальной системы.

Следует отметить, что выбор модели никак не влияет на этапы разработки и развития интеллектуальной системы и на результат проектирования. Модель лишь визуализирует.

Итак. 1. Жизненный цикл начинается не со сбора информации, как это принято считать, а с идеи. Именно с нее начинает свой путь интеллектуальная система, как впрочем, и любой другой программный продукт. Именно на этом этапе закладывается основа. И именно сейчас «рождается» интеллектуальная система и все последующие этапы уже и есть ее жизненный цикл. На каждом этапе, который будет обозначен, жизненный цикл может прерваться или поменять свое направление («судьбу»).

Этот этап может проходить формально и закреплен документально, а может неформально в рабочем порядке в процессе мозгового штурма. Все идеи необходимо проанализировать, возможно с применением специальных методик, и выбрать одну рабочую.

2. После того как идея четко сформулирована и понятна цель, для чего разрабатывается интеллектуальная система настает очередь сбора информации. Рациональнее и логичнее осуществлять его с помощью метода мозгового штурма и опроса экспертов, а также всех потенциальных категорий пользователей системы.

3. На следующем этапе целесообразно проанализировать всю собранную информацию. Убрать лишний информационный «шум», расширить недостающую информацию, сгруппировать данные. Для анализа информации можно применить метод экспертного анализа. Особенно если будущая

интеллектуальная система является узкоспециализированной и разработчик не является специалистом в этой конкретной области. Здесь очень важно отобрать только то, что касается цели и идее разработки.

4. Помимо сбора информации необходимо также осуществить сбор требований к системе. Условий работы и отклика системы при различных условиях, в том числе тех, которые нельзя предсказать. Какие необходимы требования к безопасности интеллектуальной системы, к методам аутентификации, какие могут быть риски (в международной терминологии Risk Analysis) и как реагировать на них, к обеспечению качества работы системы (в международной терминологии Software Quality Attributes). Здесь необходимо максимально согласовать с заказчиками ожидания и требования к системе, что именно пользователь получит от работы с ней. По возможности построить прототип , либо имитационную модель работы интеллектуальной системы. Необходимо максимально снизить риск возникновения недовольства со стороны заказчика и несоответствия его ожиданиям от результата.

Также четкое указание требований защитит и разработчиков в случае спорных вопросов.

Каким образом будет происходить коммуникации между разработчиками и заказчиком. Определить точки контроля и отчета.

По опыту практикующих разработчиков необходимо предусмотреть возможность внесения корректировок к требованиям. По согласованию всех сторон. Иногда только в процессе разработки бывает видно как именно нужно сделать тот или иной функционал или другие технические вопросы.

Не менее важным является дизайн проект будущей системы. Требования к визуалу должны быть также максимально четко прописаны.

На этом этапе необходимо привлечь бизнес-аналитиков для проработки всей информации и составления требований. В международной терминологии это Software Requirement Specification. Если предполагается эксплуатировать интеллектуальную систему на международном уровне и привлекать иностранных специалистов к разработке, имеет смысл придерживаться стандартов при описании технических требований к системе.

Также именно сейчас необходимо составить план валидации и верификации в соответствии с международными стандартами. А также критерии, по которым эксперты будут принимать готовое программное обеспечение от разработчиков. По международным стандартам Acceptance Criteria.

5. Составление бюджета. После того, как требования были максимально четко описаны и утверждены заказчиком необходимо спланировать расходы. Бюджет в первую очередь зависит от возможности заказчика. Также на него влияет Методы разработки, Величина команды разработчиков. Квалификация разработчиков (зависит от сложности интеллектуальной системы).

Необходимость привлечения сторонних специалистов, таких как эксперты, дизайнеры, фотографы, видеомонтажеры, копирайтеры, психологи и прочих не может не сказаться на увеличении бюджета.

В процессе разработки может потребоваться дополнительный бюджет. Это также необходимо прописать и согласовать.

После согласования бюджета и утверждения заработной платы команды разработчиков необходимо заключить договор с каждым из них. Таким образом, в жизненном цикле интеллектуальной системы появятся новые «элементы».

Для успешной реализации проекта желательно привлекать специалистов, имеющих опыт разработки именно схожих по тематики систем. Таким образом риски и внештатные ситуации сократятся до минимума.

6. Планирование графика работы. Тайминг - необходимый этап планирования разработки интеллектуальной системы. Также как и любого проекта.

Необходимо составить план работ со сроками окончания проекта в целом, а также сроки каждого этапа, с возможными отклонениями от графика в любую сторону без срыва общего срока сдачи готового проекта.

Необходимо обозначить контрольные даты для отчета по каждому этапу.

7. Разработка структуры интеллектуальной системы. Здесь необходимо по возможности предусмотреть, чтобы архитектура программного обеспечения давала возможность реализовать весь утвержденный функционал будущей системы. Также желательно предусмотреть подключение дополнительных модулей системы. В процессе разработки структуры подбираются и технологии, и инструменты, средствами которых будет реализована интеллектуальная система, структура баз данных, потоки данных.

Для документирования подходов к разработке архитектуры можно воспользоваться стандартизированной международной спецификацией Design Document Specification. Здесь четко определяются все архитектурные модули интеллектуальной системы и, по необходимости, ее связь с внешними модулями. Например, может потребоваться связь с бухгалтерским модулем, с юридическим справочником и прочими. Иногда разделяют высокоуровневую архитектуру (в международной терминологии High-Level Design) и низкоуровневую архитектуру (в международной терминологии Low-Level Design).

8. Разработка программного обеспечения интеллектуальной системы. На этом этапе происходит непосредственно программная реализация системы. По итогу этого этапа интеллектуальная система уже будет работающим продуктом. Чем тщательнее были описаны и согласованы предыдущие этапы, тем более правильно и эффективно будет работать система.

Качество и точность описания требований существенно влияет на стоимость и продолжительность разработки. Чем хуже требования прописаны, тем больше ошибок нужно будет исправить, следовательно, увеличиваются незапланированные расходы и время на их исправление вносит сбои в график.

Сейчас может возникнуть необходимость корректировки требований, которые были согласованы, ранее. Может возникнуть необходимость привлечения дополнительного бюджета и сторонних специалистов. Могут быть скорректированы сроки. Без ущерба сдачи готовой работы в оговоренные сроки.

На этапе разработки программного обеспечения необходимые тесты для проверки работоспособности системы проводятся самими разработчиками без привлечения пользователей и заказчика. Сейчас необходимо исключить технические и программные ошибки, сбои и неточности.

9. Тестирование программного обеспечения. Этому этапу необходимо также уделить внимание. И сейчас к тестированию привлекаются и пользователи, и разработчик, и специально обученные тестировщики. Необходимо выявить все ошибки или дефекты, найти причины их возникновения с целью предотвращения возникновения новых. После устранения ошибок и дефектов необходимо провести повторное тестирование. И таким образом цикл проверки продолжается до момента «чистой» работы системы.

Целесообразно, проводить тестирование каждого программного модуля. Это сэкономит время тестирования всей системы и риска исправлять ошибки, допущенные в критических местах, затрагивающие работу всей интеллектуальной системы.

10. Ввод в эксплуатацию. После успешного прохождения опытной эксплуатации и тестов выпускается релиз программного продукта. Т.е. рабочую версию интеллектуальной системы. Производится настройка на стороне заказчика или пользователя рабочей среды, установка, конфигурация и запуск интеллектуальной системы.

На этом этапе необходимо подключит маркетинговую службу для разработки стратегии продвижения.

Используется несколько вариантов выпуска релиза.

Например, выпуск готовой интеллектуальной системы частями. По этапам. Таким образом цена этапа будет ниже, а по итогу цена полномодульной интеллектуальной системы выше. Но требуется время. И есть риск, что покупатель ограничится покупкой только первого модуля.

Либо релиз включает полную версию интеллектуальной системы и в целях продвижения запускается кросс-промо на всех рекламных площадках.

Целесообразно осуществлять опрос пользователей по работе с системой. Эту информацию можно учитывать при разработке новых версий, устранении недостатков или ненужных модулей. А также для накопления опыта, которым будет использован при разработке новых интеллектуальных систем.

На рисунке этапы жизненного цикла интеллектуальной системы представлены схематично.

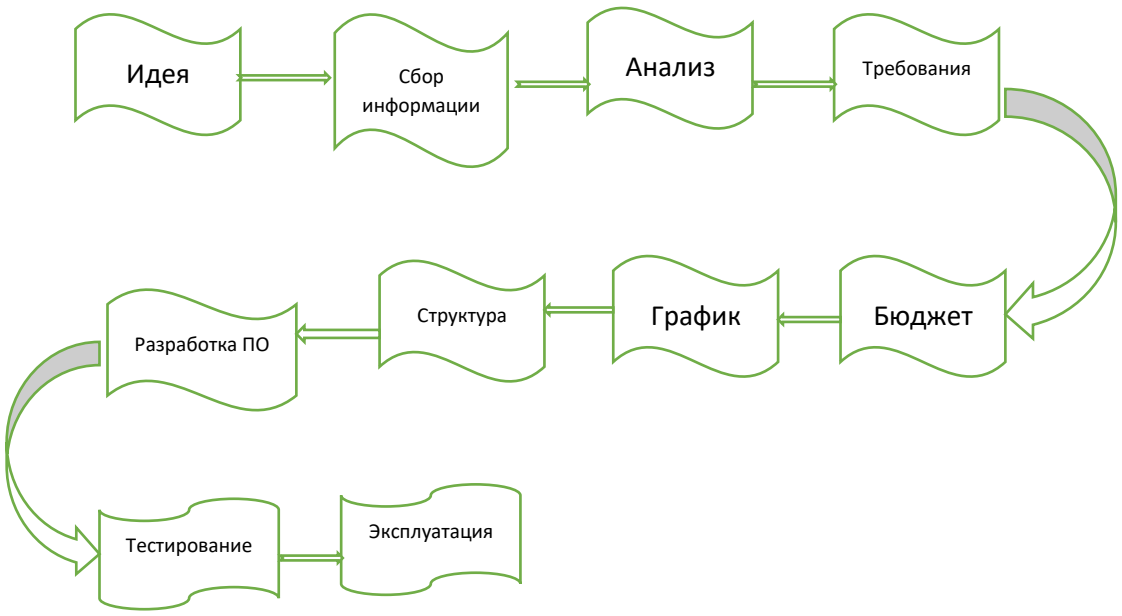


Рисунок - этапы жизненного цикла интеллектуальной системы

Описанные стадии жизненного цикла разработки интеллектуальной системы повторяются для каждого проекта. В зависимости от сложности разрабатываемой системы этапы могут быть более короткими или, наоборот, более подробными.

На всех этапах очень важно обеспечить надежную и оперативную коммуникацию команды разработчиков с заказчиком.

При разработке сложных систем, а интеллектуальные системы априори являются сложными, необходимо разработать прототипы нескольких вариантов системы, построить модели для визуального представления о работе будущей системы. Безусловно, это занимает очень много времени. Но по итогу сокращает риски.

С помощью прототипов и моделей уже на первых этапах можно предсказать и планировать рабочие процессы, изменения в бюджете (например, необходимо привлечение иностранных закупок, а курс валют – неустойчив.).

Также можно визуально и наглядно иметь представление о том, на каком этапе находится разработка и при возникновении форс-мажорных ситуаций или задержек в календарном графике можно увидеть причины их возникновения.

После запуска продукта он начинает развиваться, изменяться, дополняться.

11. И последний этап жизненного цикла интеллектуальной системы – это прекращение ее существования. Этот этап может не настать бесконечно долго, а может случиться еще на этапе запуска релиза.

На этом этапе система выводится из эксплуатации, либо она заменяется на более современный аналог, либо же новой версией той же интеллектуальной системы.

В среднем, по статистическим данным, активный жизненный цикл продолжается шесть-восемь лет.

Исходя из описания всех этапов можно увидеть, что львиную долю времени и усилий требует не разработка интеллектуальной системы, а составление технического задания и поддержка.

Для поддержки жизненного цикла интеллектуальной системы, как и любого другого программного продукта существуют специальные средства.

Современные методологии и реализующие их технологии включают в себя библиотеки процессов, шаблонов, методов, моделей и других компонент, предназначенных для построения программного обеспечения и интеллектуальных систем. Эти методологии включают также средства, которые должны обеспечивать их адаптацию для конкретных пользователей и развитие методологии по результатам выполнения конкретных проектов.

Одной из наиболее распространенных в мире электронных методологий является методология DATARUN. В соответствии с которой жизненный цикл программного обеспечения разбивается на стадии, которые связываются с результатами выполнения основных процессов, которые определяются международными стандартами. Каждую стадию должен завершать план работ на следующую стадию.

Стадия формирования требований и планирования включает в себя действия по определению начальных оценок объема и стоимости проекта. Должны быть сформулированы требования и экономическое обоснование для разработки интеллектуальной системы, функциональные модели и исходная концептуальная модель данных, которые дают основу для оценки технической реализуемости проекта. Основными результатами этой стадии должны быть

модели деятельности организации, требования к системе, включая требования по сопряжению с существующими ИС, исходный бизнес-план.

Стадия концептуального проектирования начинается с детального анализа первичных данных и уточнения концептуальной модели данных, после чего проектируется архитектура системы. Архитектура включает в себя разделение концептуальной модели на обозримые подмодели. Оценивается возможность использования существующих интеллектуальных систем и выбирается соответствующий метод их преобразования. После построения проекта уточняется исходный бизнес-план. Выходными компонентами этой стадии являются концептуальная модель данных, модель архитектуры системы и уточненный бизнес-план.

На стадии спецификации приложений продолжается процесс создания и детализации проекта. Концептуальная модель данных преобразуется в реляционную модель данных. Определяется структура приложения, необходимые интерфейсы приложения в виде экранов, отчетов и пакетных процессов вместе с логикой их вызова. Модель данных уточняется бизнес-правилами и методами для каждой таблицы. В конце этой стадии принимается окончательное решение о способе реализации приложений. По результатам стадии должен быть построен проект ИС, включающий модели архитектуры ИС, данных, функций, интерфейсов (с внешними системами и с пользователями), требований к разрабатываемым приложениям (модели данных, интерфейсов и функций), требований к доработкам существующих ИС, требований к интеграции приложений, а также сформирован окончательный план создания ИС.

На стадии разработки, и тестирования должна быть создана тестовая база данных, частные и комплексные тесты. Проводится разработка, прототипирование и тестирование баз данных и приложений в соответствии с проектом. Отлаживаются интерфейсы с существующими системами. Описывается конфигурация текущей версии ПО. На основе результатов тестирования проводится оптимизация базы данных и приложений. Приложения интегрируются в систему, проводится тестирование приложений в составе системы и испытания системы. Основными результатами стадии являются готовые приложения, проверенные в составе системы на комплексных тестах, текущее описание конфигурации ПО, скорректированная по результатам испытаний версия системы и эксплуатационная документация на систему.

Методология реализована с помощью инструментальной среды SE Companion

Оно позволяет:

- создать гипертекстовое описание методологии в виде иерархии описания стадий, этапов и операций разработки;
- создать гипертекстовое описание всех методов и методик реализации процессов жизненного цикла программного обеспечения;
- выделить из гипертекстового описания иерархию процессов жизненного цикла программного обеспечения для планирования и управления процессом создания ПО;
- изменять гипертекстовые описания жизненного цикла и методов так, как это необходимо разработчику, иными словами, производить авторизацию методологии и отслеживать эти изменения в иерархии работ, предназначенной для управления проектом;
- привязать к процессам жизненного цикла инструментальные средства поддержки этих процессов и обеспечить вызов инструментальных средств из соответствующих экранов гипертекстового справочника;
- обеспечить просмотр гипертекстовых экранов описания используемых методов из инструментальных средств;
- обеспечить поддержку процесса управления разработкой, в частности, за счет взаимодействия со средством планирования работ MS Project, оценивания трудоемкости проекта, отслеживания выполнения работ, создания графиков работ, и др.

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

Комплекс средств, поддерживающих полный жизненный цикл программного обеспечения (интегрированное CASE средства) содержит следующие компоненты;

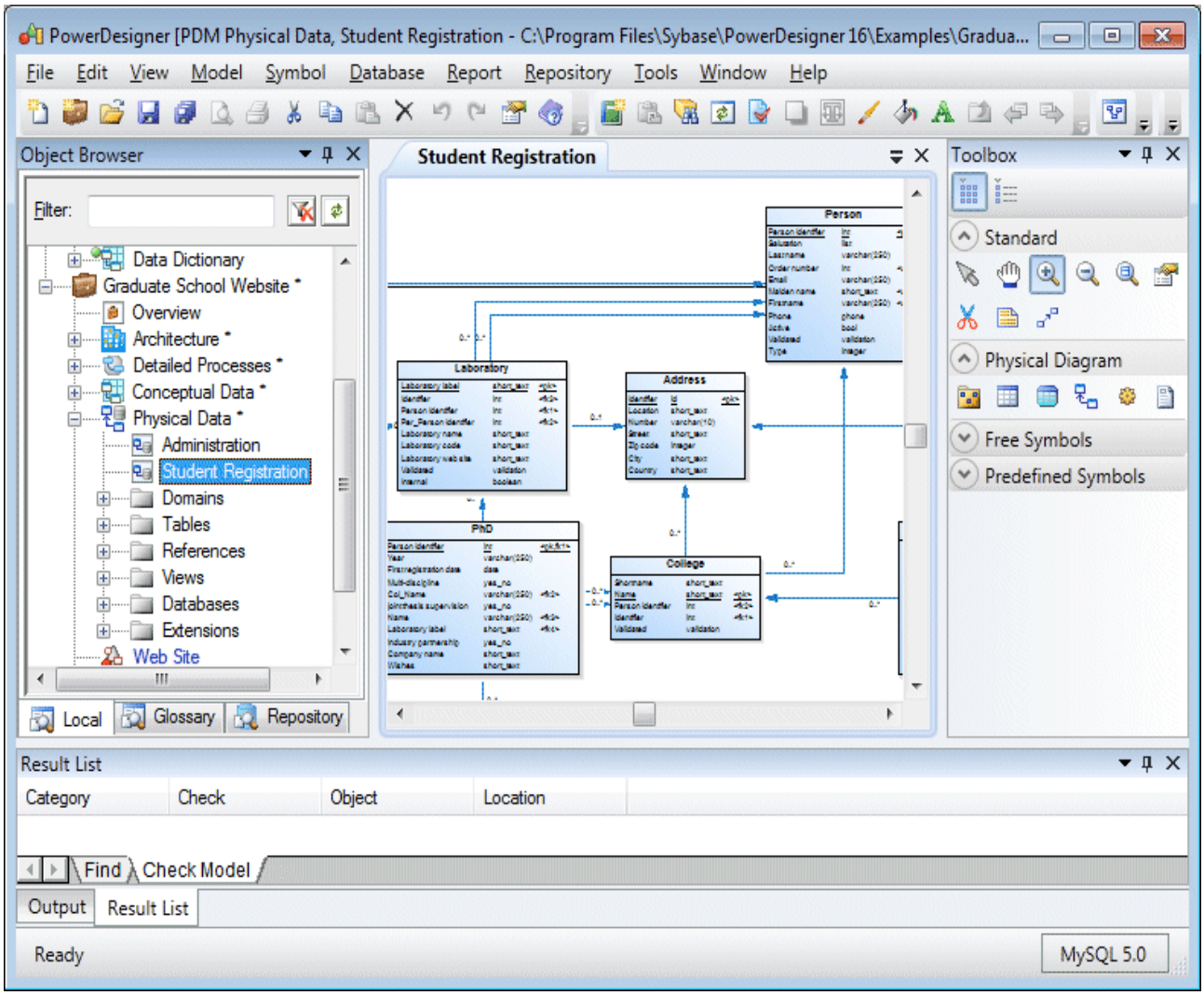
- репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;
- графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (DFD, ERD и др.), образующих модели ИС;
- средства разработки приложений, включая языки 4GL и генераторы кодов;
- средства конфигурационного управления;
- средства документирования;

- средства тестирования;
- средства управления проектом;
- средства реинжиниринга.

На сегодняшний день Российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами:

- Vantage Team Builder (Westmount I-CASE);
- Designer/2000;
- Silverrun;
- ERwin+BPwin;
- S-Designor;
- CASE.Аналитик.

Пример работы с одним из современных CASE-средств представлен на рисунке.ниже.



Вопросы по теме лекции:

- что такое жизненный цикл информационной системы?
- каковы стадии жизненного цикла информационных систем?
- в чем особенность водопадной или каскадной модели жизненного цикла информационных систем?
- какие существуют на сегодняшний день модели жизненного цикла информационных систем?
- в чем особенности и преимущества спиральной модели жизненного цикла информационных систем?
- в каких случая целесообразно применения какой модели жизненного цикла?
- что такое CASE-средство?
- приведите примеры CASE-средств
- в какой степени существующие CASE-средства ускоряют и упрощают разработку интеллектуальных информационных систем?

Литература и материалы по теме:

Гвоздева, Т.В. Проектирование информационных систем. Стандартизация: Учебное пособие / Т.В. Гвоздева, Б.А. Баллод. - СПб.: Лань, 2019. - 252 с.

Скрипкин К.Г. Экономическая эффективность информационных систем. ─ Проспект,2018. — 160 с.

1.1.1.3 Методы и средства проектирования ИС: технологии проектирования ИС и их классификация.

Проектирование интеллектуальных систем это сложный процесс. Во время проектирования определяется, как и с использованием каких методов и средств будет создаваться интеллектуальная система. А также, какие будут задействованы инструменты.

Во время проектирования решается какова будет структура и как будут обрабатываться объекты будущей базы данных, какие потребуются отчеты и экранные формы и как они будут визуализированы. В каких запросах возникнет необходимость во время работы с интеллектуальной системой и, каким образом они будут осуществляться. А так же другие особенности функционала.

Проектирование целесообразно начать после этапа анализа. Все этапы были рассмотрены в теме о жизненном цикле интеллектуальной системы. Но иногда проектирование может продолжаться даже после этапа выпуска релиза. Так как могут возникнуть новые требования и особенности при использовании интеллектуальной системы.

Цель создания любой интеллектуальной системы это быстрая, оперативная, гибкая ко всем изменениям обработка информации. А также подстраиваемость к постоянно нарастающим объемам информации. Таким образом цель сводится к обеспечению возможности эту информацию получать, обрабатывать и использовать путём создания функциональной безотказной системы. При этом необходимо учитывать достаточный уровень адаптации к постоянно меняющимся условиям, к постоянно растущему объему информации, к скорости обработки запросов и , безусловно, обеспечение высокого уровня безопасности.

Несомненно, учитывая разный уровень компьютерной подготовки будущих пользователей, необходимым условием является и простота в эксплуатации, простая и понятная навигация.

Структура интеллектуальной системы включает в себя:

- информацию, содержащуюся в базе данных;

- технологии и инструменты, обеспечивающие обработку информации;

- методы поиска информации;

- методы сбора информации;

- методы обработки информации;

- методы хранения информации;

- методы распространения информации;

- способы реализации указанных методов;

- контроль за процессом планирования разработки;

- контроль за процессом разработки структуры интеллектуальной системы;

- контроль за процессом программной реализации интеллектуальной системы;

- контроль за вводом в эксплуатацию;

- контроль за анализом возможных сбоев, ошибок и недоработок, которые могут возникнуть после выпуск релиза.

Проектирование интеллектуальной системы происходит в два этапа:

1. Проектирование, при котором учитываются особенности проектируемой системы и особенности используемых технологий.

2. Проектирование, при котором возможно многократная реализация с помощью любых технологий и для любой области.

Первый этап отражает ручную технологию проектирования. Предполагает использование универсальных компьютерных инструментов. Осуществляют реализацию этого этапа непосредственно исполнители.

Применяется, для локальных и относительно небольших интеллектуальных систем с минимальным использованием типовых решений. Адаптация проектных решений происходит только посредством перепрограммирования программных модулей.

Организовать этот этап возможно с помощью разделения процесса проектирования на этапы:

1. Предпроектная стадия. Производится предпроектный анализ и составляется техническое задание. То есть, формируются требования к интеллектуальной системе, разрабатывается её концепция, составляется технико-экономическое обоснование и пишется техническое задание на основе всех требований.

2. Проектная стадия предусматривает составление прототипного и технического проектов, разработку рабочей документации.

3. Послепроектная стадия предполагает выпуск релиза и все мероприятия с этим связанные, обучению персонала, анализу результатов испытания. Частью этого этапа становится сопровождение интеллектуальной системы и устранение выявленных недостатков.

Второй этап проектирования дает возможность декомпозиции проектируемой интеллектуальной системы с разделением на компоненты, в число которых входят программные модули, подсистемы, комплексы задач и др. Для реализации проекта интеллектуальной системы можно воспользоваться стандартными решениями, которые уже существуют на рынке, и настроить их под нужны конкретной организации. Такое проектирование предполагает обязательное наличие документации, в которой описаны все детали, особенности работы и настройки.

Декомпозиция может иметь несколько уровней, что позволяет выделить классы проектных решений:

- по отдельной задаче или элементу. Их называют элементарными,
- по отдельным подсистемам. Их называют подсистемными.
- тематические типовые проектные решения, содержащие весь набор подсистем, необходимый для конкретной задачи Их называют объектные.

При проектировании интеллектуальных систем, которые относятся к отдельной задаче или элементу имеется возможность модульного подхода реализации. Но нужно учитывать опасность, что отдельные элементы даже в рамках одной задачи могут быть несовместимы. Для устранения этой проблемы понадобится время, привлечение дополнительного бюджета и рабочей силы.

При проектировании интеллектуальных систем, которые предполагается разрабатывать отдельными подсистемами, также присущ модульный подход. И здесь есть возможность настроить параметры под разные уровни управления. При таком проектировании тоже может возникнуть проблема с несовместимостью, например, при использовании программного обеспечения разных производителей для разработки системы.

При проектировании интеллектуальных систем, которые относятся к классу типовых и содержат весь набор подсистем, по сравнению с предыдущими вариантами проблемы с совместимостью устраняются легче или отсутствуют вовсе.

Такой способ проектирования обладает следующими преимуществами:

- масштабируемость, при которой возможно проектировать конфигурацию интеллектуальной системы для разного числа рабочих мест, разный уровень доступа, подключение дополнительных модулей.

- компоненты объединены единой методологией.

- компоненты интеллектуальной системы изначально совместимы друг с другом.

- архитектура интеллектуальной системы открыта и дает возможность реализовывать проект на различных платформах.

- используя конфигурирование можно добавлять неограниченное количество подмножества компонентов интеллектуальных системы.

При проектировании используются различные стандартизированные на международном уровне методологии.

1. Методология функционального моделирования работ SADT. Методология основана на структурном анализе и графическом представлении организации как системы функций. Выделяется функциональная, информационная и динамическая модели. В настоящее время методология известна как нотация (стандарт) IDEF0. Анализируемый процесс графически представляется в виде прямоугольника, где сверху изображаются регламентирующие и управляющие воздействия, снизу – объекты управления, слева – входные данные, а справа – выходные.

2. Методология быстрой разработки приложений RAD. Это быстрая разработка приложений, которая представляется возможной благодаря применению компонентно-ориентированного конструирования. Методология применяется на проектах с ограниченным бюджетом, нечетко описанными требованиями к интеллектуальной системе, ограниченных сроках реализации. Методологию используют если пользовательский интерфейс можно продемонстрировать в прототипе, а проект разделить на функциональные элементы.

3. Методология RUP. В этой методологии реализуются итерационный и наращиваемый подходы. Построение системы происходит на основе архитектуры интеллектуальной системы, а планирование и проектное управление – на основе функциональных требований к интеллектуальной системе. Разработка интеллектуальной системы происходит итерациями, как комплекс отдельных небольших проектов со своими планами и задачами. Для итерационного цикла характерна периодическая обратная связь и адаптация к идее интеллектуальной системы.

Методы проектирования классифицируют по различным методам. Например:

- по использованию типовых проектных решений.

- по применяемым инструментам реализации

- по степени адаптивности

Информация в настоящее время является одним из самых ценных ресурсов. Для упорядочивания и управления информацией необходимы интеллектуальные информационные системы. Для каждой отдельной области и даже отдельной задачи требуется своя специализированная система.

Интеллектуальные системы являются разнотипными и отличаются не только принципами построения, требованиями, но и правилами обработки информации внутри конкретной системы.

С целью классификации интеллектуальных систем целесообразно выделить наиболее существенные их признаки, которые определяют функциональные возможности и особенности построения.

Способы классификации.

1. По типу хранимых данных. Проектируемые интеллектуальные системы принято разделять на:

- фактографические (предназначены для хранения и работы с данными, которые представлены в виде текста и цифр. Над ними удобно осуществлять необходимые манипуляции)

- документальные (предназначены для работы с информацией, которая представлена в виде документов. Здесь можно осуществлять ограниченный круг операций, а обработка данных в таких системах, по большому счету, не производится).

2. По степени автоматизации. Проектируемые интеллектуальные системы можно разделить на:

- ручные (здесь отсутствует использование автоматизированных средств для обработки информации),

- автоматические (здесь, наоборот, отсутствует использование человеческого труда)

- автоматизированные (здесь используется и ручной человеческий труд и технические средства.)

3. По характеру обработки данных. Проектируемые интеллектуальные системы можно разделить на:

- информационно-поисковые (здесь производится ввод, систематизацию, хранение, выдачу информации по запросу пользователя без сложных операций над данными)

- информационно-решающие (здесь существует алгоритм, по которому происходит обработка информации).

4. По характеру использования выходной информации. Проектируемые интеллектуальные системы можно разделить на:

- управляющие (здесь решаются задачи, где необходимы расчеты и работа с большими объемами информации)

- советующие (здесь имитируются процессы обработки знаний. Конкретные действия не производятся. Эта роль в итоге отводится пользователю.)

5. По сфере применения. Проектируемые интеллектуальные системы можно разделить на:

- системы организационного управления (здесь автоматизируются функции управленческого персонала. Для таких функций необходим оперативный контроль, быстрое решение возникающих проблем, учет и анализ таких проблем, планирование, бухгалтерский, финансовый и управленческий учет, организационные и специфические задачи),

- системы управления технологическими процессами (здесь автоматизируются функции производственного персонала. Для таких функций необходима возможность реализации специфических производственных задач и подключения различных приборов, а также контроль за соблюдением параметров по результатам измерений этих приборов.),

- системы автоматизированного проектирования (здесь автоматизируются функции инженерных, конструкторских, архитектурных, дизайнерских рабочих задач. При проектировании таких систем необходимо предусмотреть возможность создания различных чертежей и графических отчетов, а также моделирования),

- корпоративные системы (здесь автоматизируются все функции организации, охватываются все уровни персонала и все уровни рабочих задач. Как правило, такая система состоит из модулей, которые работают в единой информационной среде и поддерживает функции всех видов деятельности. Таких как, модуль маркетинга, модуль производства, модуль финансов модуль кадров, модуль руководства и другие. Наиболее популярные корпоративные системы: БЭСТ, Инотек, Инфософт, Microsoft-Business Solutions - Navision,

Axapta, 1С-Предприятие, БОСС-Корпорация, Галактика, Парус, SAP/R3 (SAP AG)Baan (Baan))

6. По уровню управления. Проектируемые интеллектуальные системы можно разделить на:

- системы оперативного уровня (здесь обрабатываются информация о сделках и событиях. Такие системы связывают организацию с внешним миром. Задачи, цели, источники информации и алгоритмы обработки на оперативном уровне заранее определены и в высокой степени структурированы)

- системы специалистов (здесь поддерживается работа с информацией и знаниями, повышают продуктивность и производительность работы конкретных специалистов. Такие системы внедряют в организацию новых спецификаций и помогают персоналу с рутинной бумажной работой)

- системы уровня менеджмента (здесь решаются задачи мониторинга, контроля, принятия решений и администрирования, сравниваются текущие показатели с прошлыми, составляются отчеты по заданным периодам)

- стратегические системы. (здесь обеспечивается поддержка принятия решений по реализации стратегических перспективных целей развития организации. Такие системы помогают руководству решать неструктурированные задачи, осуществлять долгосрочное планирование, сравнивать происходящих во вне изменений с существующим потенциалом организации. Они призваны создать общую среду компьютерной телекоммуникационной поддержки решений в возникающих ситуациях. системы способны оперативно предоставить информацию из различных источников и иногда обладают аналитическими возможностями)

7. по программно - аппаратной реализации. Проектируемые интеллектуальные системы можно разделить на :

- системы с традиционными архитектурами. (здесь используются архитектуры основанные на технологии Интернет, на использовании выделенных файл-серверов, на использовании серверов баз данных).

- системы, основанные на архитектуре DataWarehouse

- системы, основанные на архитектуре интеграции информационно-вычислительных компонентов на основе объектно-ориентированного подхода.

При выборе неправильного метода проектирования под угрозой находится судьба всего проекта. Создается множество систем, а работают и используется лишь определенный процент. По статистическим данных это около 70%.

С развитием техники, технологий, с все увеличивающимся объемом информации и ее существенным усложнением к разрабатываемой интеллектуальной системе предъявляются все больше требований и они становятся все сложнее.

И следовательно, это приводит к необходимости формирования новой методологии проектирования интеллектуальных систем.

Целью всех методологий является регламентирование процесса проектирования, обеспечении управления и контроля за процессом проектирования, гарантирования выполнения требований как к проектируемой системе, так и к процессам ее разработки.

Методология должна решать ряд задач, которые будут:

1. обеспечивать создание корпоративных интеллектуальных систем, которые соответствуют целям и задачам организации, а также предъявляемым требованиям по автоматизации бизнес процессов

2. гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта

3. поддерживать регламент сопровождения, модификации и увеличения системы

4. обеспечивать командный подход разработки,

Новая методология должна приводить к устранению недостатков существующих, т.е. снижение трудоемкости разработки, за счет максимально детального и точного описания требований и структуры разрабатываемой системы, использование самых современных методик и технологий.

Методология проектирования должна охватывать процессы проектирования объектов данных, проектирование отчетов, экранных форм и визуала, учитывать какая топология сети в организации, какая конфигурация аппаратных средств и прочие подобные вопросы.

Вопросы по теме лекции:

- что понимается под качеством информационной системы?
- какова роль управления требованиями при разработке информационных систем?
- что такое Agile?
- что понимается под командной разработкой?

Литература и материалы по теме:

Гвоздева, Т.В. Проектирование информационных систем. Стандартизация: Учебное пособие / Т.В. Гвоздева, Б.А. Баллод. - СПб.: Лань, 2019. - 252 с.

Скрипкин К.Г. Экономическая эффективность информационных систем. ─ Проспект,2018. — 160 с.

1.1.1.4 Методы и средства проектирования ИС: модели и средства описания бизнес-процессов.

В любой компании и в любой человеческой деятельности(вне зависимости от её величины) есть бизнес-процессы, которые подробно описывают все тонкости и варианты протекания какой либо деятельности и производственного процесса и определяют временные затраты на работу, могут влиять на качество продукции. Если какой то бизнес процесс перестал приносить прибыль, обеспечивать эффективную работу, стал отбирать слишком много ресурсов, прежде всего необходимо обратить внимание на описание бизнес-процессов с целью их оптимизации, улучшения.

Бизнес-процессом называют регламентированную, регулярно повторяющуюся последовательность действий одного или ряда сотрудников, благодаря которой достигается нужный результат.

Стали создавать текстовые инструкции, которые описывали и работу людей, и их взаимодействие с информационными системами. Поскольку со временем выяснилось, что работать с нотациями, составленными в произвольной форме, неудобно, было принято решение об их стандартизации.

Понятие описания бизнес-процесса

Описание бизнес-процесса — это пошаговое описание действий работников при выполнении той или иной операции, включая ответственность, порядок принятия решений, порядок взаимодействия с другими сотрудниками. Проще всего рассмотреть описание бизнес-процессов в компании на примере продаж. Так, один продавец в разных случаях (в зависимости от продаваемого им товара, стоящего перед ним покупателя, настроения, наконец) будет продавать по-разному. Соответственно, компания в разных случаях будет получать разный результат.

Но если чётко прописать процесс продажи, то, независимо от внутренних и внешних факторов, продавец будет действовать по регламенту, что с большей вероятностью обеспечит продажи.

Составляющие бизнес-процесса

Цель описания бизнес-процесса — разработать последовательные мероприятия, которые обеспечат прибыль.

Задачи бизнес процесса: контроль последовательности операций, обеспечение максимально возможной скорости их выполнения, помощь в поиске дублирующихся или лишних операций и т. д.

Любой бизнес-процесссостоит из:

1. вход — стартовый ресурс, который нужен для выполнения операций (заявка, сырьё, поставка);
2. выход — непосредственно результат, это готовый продукт, услуга, информация, документ.
3. Границы бизнес-процесса — это событие или время, которое служит началом и окончанием БИЗНЕС-ПРОЦЕСС.

Технологический процесс и бизнес-процесс

Не стоит путать бизнес-процесс технологическим процессом. В последнем случае всё происходит без участия человека, например, с привлечением программы или автоматической системы. Более того, основное отличие заключается в том, что в технологическом процессе на выходе всегда будет конкретный результат, то есть если это производство, то на выходе получается продукт с конкретными параметрами. Брак продукции не является исключением из правил, поскольку является нарушением технологического процесса.

В бизнес-процессе результат на выходе может отличаться даже без нарушения самого процесса. Это объясняется тем, что в самом алгоритме закладываются разные условия, при которых нужно выполнять те или иные действия.

Например:

если покупатель приходит за определённым товаром, продавец просто совершает продажу;

если покупатель не может определиться и выбирает товары, либо у него есть возражения, продавец по ситуации подсказывает, какой товар лучше решит проблему покупателя, или обрабатывает его возражения и после этого также совершает продажу.

Зачем моделировать (описывать) бизнес-процессы

Описание бизнес-процессов позволяет решать сразу 3 задачи.

Упрощает сложности за счёт схематического пошагового изображения всех этапов бизнес-процессов.

Позволяет изучить работу изнутри благодаря графическому изображению бизнес-процессов компании в виде схем. Это особенно важно на этапе оптимизации, масштабирования, так как в описаниях сразу видны проблемные места.

Например, есть компания, у которой насчитывается 7 отделений: построения, распространения, финансовое, техническое, отделение квалификации, отделение по работе с публикой, административное отделение. И есть владелец или руководитель бизнеса, который должен хорошо понимать их работу, чтобы эффективно ими управлять. Без описания основных бизнес-процессов человек физически не в состоянии этого сделать, а с ними он сразу видит особенности работы на каждом из этапов и понимает, что можно модернизировать.

Преимущества процессного подхода перед функциональным.

В рамках функционального подхода учитывается организационная структура управления, деятельность компании сводится к набору функций, которые выполняются теми или иными отделениями. Сложности заключаются в том, что при выполнении функций отделения концентрируют внимание на выполнении своих целей. С одной стороны, между целями разных отделений могут быть противоречия, с другой — в таких условиях могут неправильно пониматься главные операционные функции, а это повлечёт за собой снижение результативности работы. К тому же при таком подходе не будет нацеленности на итоговый результат, а это приводит к увеличению расходов и может стать причиной потери клиентов и т. д.

Процессный подход учитывает не организационную структуру и функции отделений, а деятельность руководства и команды, которую можно представить как последовательность операций, обеспечивающих нужный результат. То есть сама компания в таком случае — это комплекс взаимосвязанных бизнес-процессов, включающих функции разных отделений.

Иными словами, функциональный подход демонстрирует возможности организации, определяя, что надо делать, а процессный подход описывает технологию достижения нужных целей, давая ответ на вопрос: «Как надо делать?»

Основное преимущество процессного подхода в более высокой результативности работы компании, которая достигается за счёт:

1. клиентоориентированности;

2. передачи оперативного управления в руки наёмных менеджеров, благодаря чему руководство может заняться продумыванием стратегии развития;
3. постановки понятных целей и задач (всем ясно, что требуемые показатели нужны для получения конечного результата);
4. разработки описаний бизнес-процессов с чёткой последовательностью действий для сотрудников, что исключает риск ошибок;
5. обоснованности ресурсов (видно, для чего расходуются средства).

К тому же процессный подход даёт возможность бизнесу расширяться за счёт открытия новых площадок согласно схеме: если в одной организации бизнес-процессы чётко налажены, можно создать филиал с такими же подразделениями, обязанностями сотрудников и аналогичным образом выстроить в нём бизнес процесс. Неудивительно, что в таких условиях можно постоянно совершенствовать свою работу.

Методологии и инструментарий описания бизнес-процесса.

При выборе методологии описания бизнес-процессов и инструментария необходимо опираться на цели бизнеса:

ARIS — комплект программных обеспечений, который, прежде всего, создавался для описания алгоритмов и последовательности. Объединяет более 80 моделей.

CA ERwin Data Modeler — программа с хорошо реализованной возможностью описания взаимосвязанных моделей. Дополнительные задачи (построение дерева свойств, например) усложнены либо отсутствуют.

BPMN 2.0 — одна из лучших систем для описания бизнес-процессов, она гибкая, функциональная и простая, к тому же позволяет увидеть все взаимодействия сотрудников.

Виды бизнес-процессов.

1. Основные бизнес процессы — это процессы, направленные на производство продукции и оказание услуг. Ради них компания и создавалась, они обеспечивают ей доход. Например для ателье, основным процессом бдет пошив одежды
2. Сопутствующие бизнес-процессы также направлены на производство продукции либо оказание услуг, но в результате сопутствующей работы. Для ателье сопутствущим бизнес процессов будет перешив или ремонт одежды.

3. Вспомогательные бизнес-процессы способствуют выполнению основных. Для ателье такой процесс это ремонт швейного оборудования.
4. Обеспечивающие налаживают, делают возможным выполнение всех остальных процессов.
5. Управляющие бизнес-процессы позволяют осуществлять функции управления как в рамках отдельного бизнес-процесса, так и в комплексе.
6. Бизнес-процессы развития направлены на улучшение товаров и услуг, а также технологий.

Иногда используют более простую классификацию, где выделяются основные процессы, вспомогательные и управляющие.

Участники

Участниками бизнес-процесса называются лица или целые отделения, организации, которые выполняют определённые функции в рамках того или иного процесса. Внутренние — сотрудники и отделения предприятия, которые отвечают за ту или иную задачу. Внешние находятся вне организации, но используют результаты бизнес-процесса.

Также выделяют владельца бизнес-процесса. Он планирует, организовывает, контролирует выполнение, может вносить изменения, чтобы улучшить показатели, отвечает за результат. Реже выделяют ещё и менеджера, который обеспечивает оперативное управление. Дополнительно может создаваться экспертная группа вместе с начальниками отделов, бизнес-аналитиками, которые описывают, анализируют и улучшают бизнес-процессы.

Последовательность описания бизнес-процессов:

Сначала отмечают, как бизнес-процессаработает в текущий момент, чтобы понять его принцип, а затем делают описание методом последовательных приближений.

В описании бизнес-процесса выделяют следующие разделы:

1. стандартные формы бизнес-процесса: рекомендуется использовать шаблон для создания общего подхода;
2. карта бизнес процесса иллюстрирует процесс в виде схемы, где фиксируются действия исполнителей;
3. маршруты бизнес-процесса иллюстрируют движения сырья, людей, денег, демонстрируя логику операций;
4. матрицы бизнес процесса показывают наиболее важные бизнес-процессы и связь между ними;

5. блок-схемы бизнес процесса иллюстрируют взаимоотношения между участниками бизнес-процесса;
6. описание стыков бизнес-процесса: они всегда являются проблемными моментами, поскольку действия и зоны ответственности собственников процессов не всегда согласованы; чтобы решить задачу, описывают выходы, выявляют показатели результативности с процессом их измерения и интересующими значениями (к чему надо стремиться), а затем описывают входы;
7. вспомогательные описания бизнес-процесса с помощью диаграмм Гантта, сетевых графиков и т. д.;
8. развернутое описание бизнес-процесса: здесь важно не столько то, как описать бизнес-проект, в какой форме, сколько то, что он должен содержать; в нём должно быть название бизнес процесса, его код, определение, цель, владелец, руководитель, нормативы, входы с источниками, выходы, ресурсы, параметры, которые можно измерить, показатели результативности, бизнес процесс потребителей;
9. документирование бизнес процесса, лучше выбрать единый вид для описания всех бизнес процессов;
10. определение показателей и индикаторов бизнес процессов: чтобы измерять и анализировать результативность бизнес процессов, используют 4 группы показателей (качество, срок выполнения, количество, затраты), группа индикаторов бизнес процесса иллюстрирует степень достижения цели;
11. регламент выполнения бизнес процесса, объёмные бизнес процессы оформляют в отдельные документы, к которым в форме положений прикрепляются остальные бизнес процессы.

Правила описания бизнес процессов:

Мало знать, как правильно составить описание бизнес-процесса, важно также помнить о критериях, которые подтверждают, что перечень операций является описанием бизнес процесс, а именно:

1.законченность (если это продажа, то бизнес процесс должен описывать действия, ведущие к ней);

2. лаконичность (бизнес процесс должен быть информативным и в то же время максимально простым для восприятия);

3.применение общепризнанных нотаций (устоявшиеся правила, общепризнанные обозначения, с одной стороны, уберегут от ошибок, с другой — не вызовут вопросов, путаницы);

4. указание всех участников бизнес процесса.

5.описание должно быть понятным потребителю.

Этапы внедрения бизнес процесса. Выделяют 5 основных этапов внедрения:

1.выявление и документирование бизнес процесса: необходимо изучить ситуацию, чтобы найти проблемные места;

2.анализ бизнес процесса: на этом этапе продумывают изменения, которые нужно внести, а также подбирают необходимые инструменты;

3.непосредственно описание бизнес-процесса, составление задокументированного плана;

4.реализация принятых решений;

5.контроль, анализ бизнес процесса.

6. Сравнение нотаций

Есть 2 популярные нотации: ARIS eEPS и BPMN.

ARIS eEPS позволяет отображать поток документов со статусами. Также её отличительная черта заключается в использовании событий до и после операции, присутствии логических операторов. Моделирование в ней занимает больше времени. Диаграммы занимают больше места. В то же время семантика ограничена: если надо проиллюстрировать определённые аспекты на диаграмме, приходится обходиться тем, что есть. Дополнительное преимущество — в возможности корректной имитации процессов.

BPMN имеет наиболее развитую семантику, благодаря чему можно описывать бизнес процесс с учётом их специфики. На схеме можно применять события, логические операторы. Другое преимущество — имитация бизнес-процесса (можно имитировать и прерывание операции).

Описывая процесс для целей последующей автоматизации, стоит отметить, что использование именно нотации BPMN 2.0 может быть более удобным. В ней шире палитра. Она позволяет моделировать как изолированный поток работ, так и ряд взаимодействующих процессов. В ней есть правила сочетания значков друг с другом. Им важно следовать, чтобы информация была понятна ещё и программному обеспечению. Методы и средства проектирования ИС: модели и средства описания бизнес-процессов.

В любой компании и в любой человеческой деятельности(вне зависимости от её величины) есть бизнес-процессы, которые подробно описывают все тонкости и варианты протекания какой либо деятельности и производственного процесса и определяют временные затраты на работу, могут влиять на качество продукции. Если какой то бизнес процесс перестал приносить прибыль, обеспечивать эффективную работу, стал отбирать слишком много ресурсов, прежде всего необходимо обратить внимание на описание бизнес-процессов с целью их оптимизации, улучшения.

Бизнес-процессом называют регламентированную, регулярно повторяющуюся последовательность действий одного или ряда сотрудников, благодаря которой достигается нужный результат.

Стали создавать текстовые инструкции, которые описывали и работу людей, и их взаимодействие с информационными системами. Поскольку со временем выяснилось, что работать с нотациями, составленными в произвольной форме, неудобно, было принято решение об их стандартизации.

Понятие описания бизнес-процесса

Описание бизнес-процесса — это пошаговое описание действий работников при выполнении той или иной операции, включая ответственность, порядок принятия решений, порядок взаимодействия с другими сотрудниками. Проще всего рассмотреть описание бизнес-процессов в компании на примере продаж. Так, один продавец в разных случаях (в зависимости от продаваемого им товара, стоящего перед ним покупателя, настроения, наконец) будет продавать по-разному. Соответственно, компания в разных случаях будет получать разный результат.

Но если чётко прописать процесс продажи, то, независимо от внутренних и внешних факторов, продавец будет действовать по регламенту, что с большей вероятностью обеспечит продажи.

Составляющие бизнес-процесса

Цель описания бизнес-процесса — разработать последовательные мероприятия, которые обеспечат прибыль.

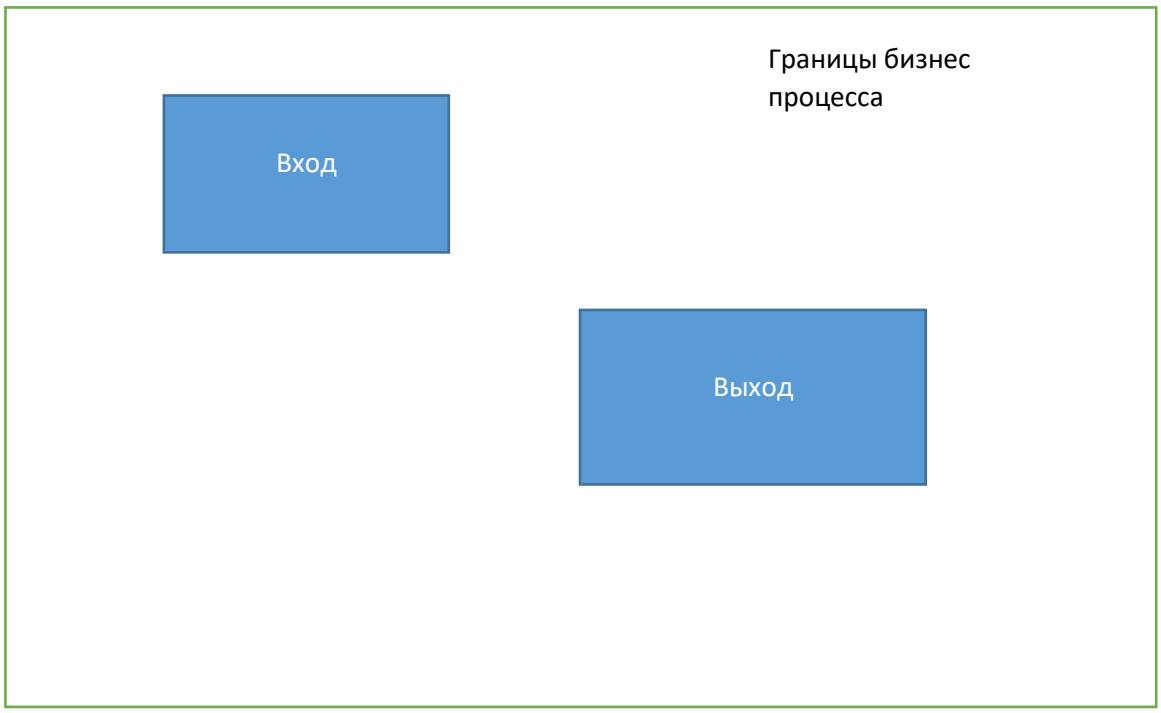
Задачи бизнес процесса: контроль последовательности операций, обеспечение максимально возможной скорости их выполнения, помощь в поиске дублирующихся или лишних операций и т. д.

Любой бизнес-процесс состоит из:

входа — какой либо стартовый ресурс, который нужен для выполнения операций (заявка, сырьё, поставка);

выхода — непосредственно результат, это готовый продукт, услуга, информация, документ.

Границы бизнес-процесса — это событие или время, которое служит началом и окончанием БИЗНЕС-ПРОЦЕСС.



Технологический процесс и бизнес-процесс

Не стоит путать бизнес-процесс технологическим процессом. В последнем случае всё происходит без участия человека, например, с привлечением программы или автоматической системы. Более того, основное отличие заключается в том, что в технологическом процессе на выходе всегда будет конкретный результат, то есть если это производство, то на выходе получается продукт с конкретными параметрами. Брак продукции не является исключением из правил, поскольку является нарушением технологического процесса.

В бизнес-процессе результат на выходе может отличаться даже без нарушения самого процесса. Это объясняется тем, что в самом алгоритме закладываются разные условия, при которых нужно выполнять те или иные действия.

Например:

если покупатель приходит за определённым товаром, продавец просто совершает продажу;

если покупатель не может определиться и выбирает товары, либо у него есть возражения, продавец по ситуации подсказывает, какой товар лучше решит проблему покупателя, или обрабатывает его возражения и после этого также совершает продажу.

Зачем моделировать (описывать) бизнес-процессы

Описание бизнес-процессов позволяет решать сразу 3 задачи.

Упрощает сложности за счёт схематического пошагового изображения всех этапов бизнес-процессов.

Позволяет изучить работу изнутри благодаря графическому изображению бизнес-процессов компании в виде схем. Это особенно важно на этапе оптимизации, масштабирования, так как в описаниях сразу видны проблемные места.

Например, есть компания, у которой насчитывается 7 отделений: построения, распространения, финансовое, техническое, отделение квалификации, отделение по работе с публикой, административное отделение. И есть владелец или руководитель бизнеса, который должен хорошо понимать их работу, чтобы эффективно ими управлять. Без описания основных бизнес-процессов человек физически не в состоянии этого сделать, а с ними он сразу видит особенности работы на каждом из этапов и понимает, что можно модернизировать.

Преимущества процессного подхода перед функциональным.

В рамках функционального подхода учитывается организационная структура управления, деятельность компании сводится к набору функций, которые выполняются теми или иными отделениями. Сложности заключаются в том, что при выполнении функций отделения концентрируют внимание на выполнении своих целей. С одной стороны, между целями разных отделений могут быть противоречия, с другой — в таких условиях могут неправильно пониматься главные операционные функции, а это повлечёт за собой снижение результативности работы. К тому же при таком подходе не будет нацеленности на итоговый результат, а это приводит к увеличению расходов и может стать причиной потери клиентов и т. д.

Процессный подход учитывает не организационную структуру и функции отделений, а деятельность руководства и команды, которую можно представить как последовательность операций, обеспечивающих нужный результат. То есть

сама компания в таком случае — это комплекс взаимосвязанных бизнес-процессов, включающих функции разных отделений.

Иными словами, функциональный подход демонстрирует возможности организации, определяя, что надо делать, а процессный подход описывает технологию достижения нужных целей, давая ответ на вопрос: «Как надо делать?»

Основное преимущество процессного подхода в более высокой результативности работы компании, которая достигается за счёт:

клиентоориентированности;

передачи оперативного управления в руки наёмных менеджеров, благодаря чему руководство может заняться продумыванием стратегии развития;

постановки понятных целей и задач (всем ясно, что требуемые показатели нужны для получения конечного результата);

разработки описаний бизнес-процессов с чёткой последовательностью действий для сотрудников, что исключает риск ошибок;

обоснованности ресурсов (видно, для чего расходуются средства).

К тому же процессный подход даёт возможность бизнесу расширяться за счёт открытия новых площадок согласно схеме: если в одной организации бизнес-процессы чётко налажены, можно создать филиал с такими же подразделениями, обязанностями сотрудников и аналогичным образом выстроить в нём бизнес процесс. Неудивительно, что в таких условиях можно постоянно совершенствовать свою работу.

Методологии и инструментарий описания бизнес-процесса.

При выборе методологии описания бизнес-процессов и инструментария необходимо опираться на цели бизнеса:

ARIS — комплект программных обеспечений, который, прежде всего, создавался для описания алгоритмов и последовательности. Объединяет более 80 моделей.

CA ERwin Data Modeler — программа с хорошо реализованной возможностью описания взаимосвязанных моделей. Дополнительные задачи (построение дерева свойств, например) усложнены либо отсутствуют.

BPMN 2.0 — одна из лучших систем для описания бизнес-процессов, она гибкая, функциональная и простая, к тому же позволяет увидеть все взаимодействия сотрудников.

Виды бизнес-процессов.

Основные бизнес процессы — это процессы, направленные на производство продукции и оказание услуг. Ради них компания и создавалась, они обеспечивают ей доход. Например для ателье, основным процессом бдет пошив одежды

Сопутствующие бизнес-процессы также направлены на производство продукции либо оказание услуг, но в результате сопутствующей работы. Для ателье сопутствущим бизнес процессов будет перешив или ремонт одежды.

Вспомогательные бизнес-процессы способствуют выполнению основных. Для ателье такой процесс это ремонт швейного оборудования.

Обеспечивающие налаживают, делают возможным выполнение всех остальных процессов.

Управляющие бизнес-процессы позволяют осуществлять функции управления как в рамках отдельного бизнес-процесса, так и в комплексе.

Бизнес-процессы развития направлены на улучшение товаров и услуг, а также технологий.

Иногда используют более простую классификацию, где выделяются основные процессы, вспомогательные и управляющие.

Участники

Участниками бизнес-процесса называются лица или целые отделения, организации, которые выполняют определённые функции в рамках того или иного процесса. Внутренние — сотрудники и отделения предприятия, которые отвечают за ту или иную задачу. Внешние находятся вне организации, но используют результаты бизнес-процесса.

Также выделяют владельца бизнес-процесса. Он планирует, организовывает, контролирует выполнение, может вносить изменения, чтобы улучшить показатели, отвечает за результат. Реже выделяют ещё и менеджера, который обеспечивает оперативное управление. Дополнительно может создаваться экспертная группа вместе с начальниками отделов, бизнес-аналитиками, которые описывают, анализируют и улучшают бизнес-процессы.

Последовательность описания бизнес-процессов:

Сначала отмечают, как бизнес-процесса работает в текущий момент, чтобы понять его принцип, а затем делают описание методом последовательных приближений.

В описании бизнес-процесса выделяют следующие разделы:

стандартные формы бизнес-процесса: рекомендуется использовать шаблон для создания общего подхода;

карта бизнес процесса иллюстрирует процесс в виде схемы, где фиксируются действия исполнителей;

маршруты бизнес-процесса иллюстрируют движения сырья, людей, денег, демонстрируя логику операций;

матрицы бизнес процесса показывают наиболее важные бизнес-процессы и связь между ними;

блок-схемы бизнес процесса иллюстрируют взаимоотношения между участниками бизнес-процесса;

описание стыков бизнес-процесса: они всегда являются проблемными моментами, поскольку действия и зоны ответственности собственников процессов не всегда согласованы; чтобы решить задачу, описывают выходы, выявляют показатели результативности с процессом их измерения и интересующими значениями (к чему надо стремиться), а затем описывают входы;

вспомогательные описания бизнес-процесса с помощью диаграмм Гантта, сетевых графиков и т. д.;

развернутое описание бизнес-процесса: здесь важно не столько то, как описать бизнес-проект, в какой форме, сколько то, что он должен содержать; в нём должно быть название бизнес процесса, его код, определение, цель, владелец, руководитель, нормативы, входы с источниками, выходы, ресурсы, параметры, которые можно измерить, показатели результативности, бизнес процесс потребителей;

документирование бизнес процесса, лучше выбрать единый вид для описания всех бизнес процессов;

определение показателей и индикаторов бизнес процессов: чтобы измерять и анализировать результативность бизнес процессов, используют 4 группы показателей (качество, срок выполнения, количество, затраты), группа индикаторов бизнес процесса иллюстрирует степень достижения цели;

регламент выполнения бизнес процесса, объёмные бизнес процессы оформляют в отдельные документы, к которым в форме положений прикрепляются остальные бизнес процессы.

Правила описания бизнес процессов:

Мало знать, как правильно составить описание бизнес-процесса, важно также помнить о критериях, которые подтверждают, что перечень операций является описанием бизнес процесс, а именно:

1.законченность (если это продажа, то бизнес процесс должен описывать действия, ведущие к ней);

2. лаконичность (бизнес процесс должен быть информативным и в то же время максимально простым для восприятия);

3.применение общепризнанных нотаций (устоявшиеся правила, общепризнанные обозначения, с одной стороны, уберегут от ошибок, с другой — не вызовут вопросов, путаницы);

4. указание всех участников бизнес процесса.

5.описание должно быть понятным потребителю.

Любой бизнес процесс внедряется в несколько этапов. Традиционно выделяют шесть этапов:

1.выявление проблемы и документальное оформления бизнес процесса: необходимо изучить ситуацию, выявляется проблема или ряд проблем которые существуют в настоящий момент;

2.анализ конкретного бизнес процесса: на этом этапе продумывают изменения, которые нужно внести, а также подбирают необходимые инструменты для реализации этих изменений;

3.непосредственно описание бизнес-процесса, составление документации по реализации решения проблем;

4.реализация принятых решений, используя различные инструменты;

5.контроль за внедрением, анализ бизнес процесса в процессе внедрения и после внедрения.

6. Сравнение нотаций бизнес процессов в предметной области

Есть 2 популярные нотации: ARIS eEPS и BPMN.

Нотация ARIS eEPS позволяет отображать поток документов со статусами. Также его отличительной чертой является использование событий до и после операции, наличие логических операторов. Моделирование в нем занимает больше времени. Диаграммы занимают больше места. В то же время семантика ограничена: если вам нужно проиллюстрировать определенные аспекты на диаграмме, вам придется обойтись тем, что у вас есть. Дополнительным преимуществом является возможность корректного моделирования процессов.

BPMN обладает наиболее развитой семантикой, благодаря которой можно описать бизнес-процесс с учетом их специфики. Вы можете использовать события, логические операторы на диаграмме. Еще одним

преимуществом является имитация бизнес-процесса (также можно имитировать прерывание операции).

Описывая процесс с целью последующей автоматизации, стоит отметить, что использование нотации BPMN 2.0 может оказаться более удобным. Имеет более широкую палитру. Он позволяет моделировать как изолированный рабочий процесс, так и серию взаимодействующих процессов. В нем есть правила сочетания иконок друг с другом. Важно следовать им, чтобы информация была понятна и программному обеспечению.

Вопросы по теме лекции:

- что такое бизнес-процесс?
- какие существуют средства описания бизнес-процессов?
- что дает графическое представление бизнес-процессов?
- какова связь моделирования и автоматизации бизнес-процессов?
- что такое модель бизнес-процесса?
- в чем значение бизнес-процессов для предприятия?
- какие интеллектуальные информационные технологии можно использовать для цифровизации бизнес-процессов предприятия?

Литература и материалы по теме:

Нелис, Й. Управление бизнес-процессами: Практическое руководство по успешной реализации проектов / Й. Нелис, Д. Джестон. - СПб.: Символ-плюс, 2015. - 512 с.

Тельнов, Ю.Ф. Инжиниринг предприятия и управление бизнес-процессами. Методология и технология: Учебное пособие / Ю.Ф. Тельнов, И.Г. Федоров. - М.: Юнити, 2017. - 304 с.

1.1.1.5 Автоматизированное проектирование ИС с использованием CASE-технологий: структурный подход к проектированию ИС.

Автоматизированное проектирование с помощью CASE технологий используется для интеллектуальных систем, в которых широко используются графические методы.

Аббревиатура CASE (Computer-Aided Software/System Engineering) используется в довольно широком смысле. Первоначально использование CASE было ограничено вопросами автоматизации программного обеспечения, а в настоящее время охватывает весь процесс разработки сложных интеллектуальных систем.

Большинство существующих CASE систем ориентировано на автоматизацию проектирования и основано на методологиях структурного или объектно-ориентированного проектирования и программирования, использующих спецификации в виде диаграмм или текстов для описания системных требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

Основой CASE-методологии является моделирование. CASE-технология - это модельный метод автоматизации проектирования системы.

CASE-технология основана на парадигме: методология – метод – нотации – средства.

Методология определяет общие подходы к оценке и выбору варианта системы, последовательность стадий и этапов проектирования, подходы к выбору методов.

Метод конкретизирует порядок проектирования отдельных компонентов (например, методы проектирования потоков данных в системе, задания описаний процессов, представления структур данных в хранилище и т.д.).

Нотации – графические средства обозначения и правила для описания структуры системы, этапов обработки информации, структуры данных и т.д. Включают графы, диаграммы, таблицы, блок-схемы, формальные и естественные языки.

Средства – инструментарий, средства автоматизации проектирования в виде программных продуктов, предназначенных для обеспечения

интерактивного режима проектирования (создание и редактирование графического проекта информационной системы) и кодогенерации программ (автоматического создания кодов программ системы).

Процесс моделирования может быть реализован в рамках различных методик, отличающихся прежде всего своим подходом к тому, что представляет собой моделирующая организация. В соответствии с различными представлениями об организации методики принято делить на объектно-ориентированные и функционально-ориентированные.

Объектные рассматривают моделируемую организацию как набор взаимодействующих объектов - производственных единиц. Объект определяется как осязаемая реальность - предмет или явление, имеющие четкое определяемое поведение. Целью применения данной методики является выделение объектов, составляющих организацию, и распределение между ними ответственностей за выполняемые действия.

Функциональные методики, наиболее известной из которых является IDEF, рассматривают организацию как набор функций, преобразующий поступающий поток информации в выходной поток. Процесс преобразования информации потребляет определенные ресурсы. Основное отличие от объектной методики заключается в четком отделении функций (методов обработки данных) от самих данных.

Каждый из представленных подходов имеет свои преимущества. Объектный подход позволяет построить более устойчивую к изменениям систему, а функциональное моделирование хорошо показывает себя в тех случаях, когда организационная структура находится в процессе изменения или вообще слабо оформлена.

Функциональная методика IDEF0. Целью методики является построение функциональной схемы исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы. В методологии используется четыре основных понятия: функциональный блок, интерфейсная дуга, декомпозиция и глоссарий.

Декомпозиция предполагает разбиение сложного процесса на составные части. Уровень детализации определяется разработчиком модели. Общая модель процесса представляется в виде иерархической структуры отдельных диаграмм, что делает ее более обозримой. IDEF0 всегда начинается с представления процесса как единого функционального блока с интерфейсными дугами, выходящими за пределы рассматриваемой области. Такая диаграмма называется контекстной. В пояснительном тексте к контекстной диаграмме должно быть указано краткое описание цели построения диаграммы и определена точка зрения.

Цель определяет те области деятельности предприятия, на которые необходимо обратить внимание в первую очередь. Точка зрения определяет направленность и уровень детализации разрабатываемой модели.

Глоссарием называется набор определений, ключевых слов, повествовательных изложений, характеризующих объекты изображенные на диаграмме. Также глоссарий обеспечивает включение в диаграммы IDEF0 необходимой дополнительной информации.

Также к функциональным методикам относится методика потоков данных. Целью методики является построение модели в виде диаграммы, обеспечивающей правильное описание выходов при заданном воздействии на вход системы. Диаграммы потоков данных (в международной терминологии DFD) являются основным средством моделирования функциональных требований к проектируемой системе.

Структурный подход к проектированию интеллектуальных информационных систем основан на следующих понятиях:

1. Сложные проблемы предлагается решать, предварительно разбив одну задачу на множество мелких подзадач независимых друг от друга. При этом подзадачи должны быть простыми для восприятия, понимания и их решения.
2. Составные части задачи структурируются в иерархическую структуру. На каждом уровне иерархии добавляются новые задачи и подзадачи.
3. Выделяются только основные задачи. И именно им предлагается уделять основное внимание. Остальные задачи считаются несущественными и убираются из поля внимания.
4. Задача должна быть максимально формализована и используется методический подход к ее решению.
5. Все подзадачи и элементы этих подзадач должны быть обоснованы и согласованы.
6. Так как подход имеет название структурного – все данные задач должны быть также структурированы и организованы в иерархические структуры.

Для структурного подхода используют следующие средства и виды моделей (модели иллюстрируются при помощи диаграмм):

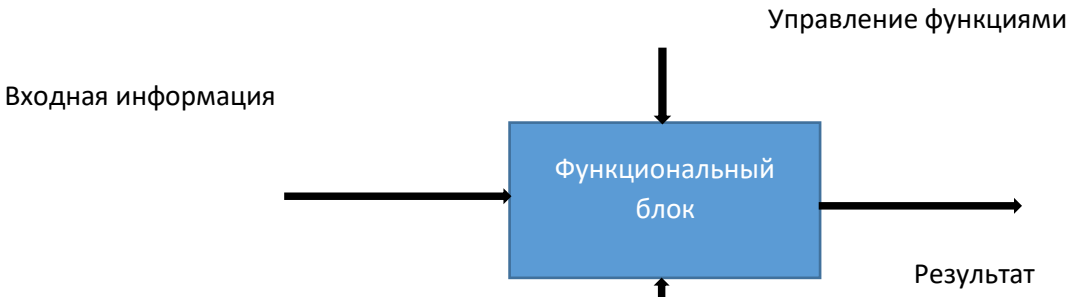
1. Методология структурного анализа и проектирования. В международной терминологии Structured Analysis and Design Technique (SADT). Здесь используются модели и иллюстрирующие их диаграммы. На основе этой методологии разработана методология IDEF0, которая рассмотрена выше.

Методология представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо конкретной прикладной предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями. Методология реализуется при помощи следующих понятий:

- Моделирование при помощи блоков. Каждая функция представлена в виде блока, а ее входы и выходы стрелками (дугами). Дуги описывают как блоки взаимодействуют друг с другом, как именно и в какой последовательности выполняются функциональные операции. А так же каким образом происходит управление функциями.
- Правила в данной методологии требуют точного и строгого выполнения. Но при этом строгость не должна слишком ограничивать работу аналитика. Необходимо соблюдать ограничение по количеству блоков на каждом уровне. Чаще всего это от трех до шести блоков. Номера блоков должны быть логичны и последовательны. И не повторяться. Имена блоков должны быть уникальными. Общепринятые правила синтаксиса для графики и текстов должны быть соблюдены. Организационная структура модели не должна влиять на функции.

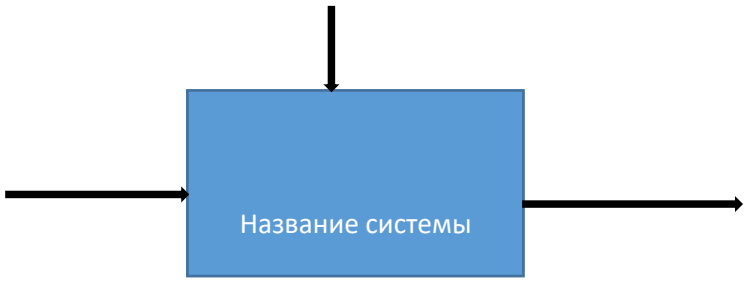
Методология структурного анализа и проектирования может использоваться для моделирования различных интеллектуальных систем и определения требований и функций, а затем для разработки системы, которая удовлетворяет этим требованиям и реализует эти функции. Для уже существующих систем методология может быть использована для мониторинга эффективности работы функциональных операций, выполняемых системой. С помощью SADT также можно дать рекомендации по устранению узких мест в системе, которые не позволяют работать максимально эффективно.

Результатом применения этой методологии является модель, которая состоит из диаграмм, фрагментов текстов и глоссария. Диаграммы - главные компоненты модели, все функции интеллектуальной системы и интерфейсы на них представлены как блоки и дуги. Место соединения дуги с блоком определяет тип интерфейса. Информация, которая управляет функциями системы входит в блок сверху, а та информация, которая будет использоваться для работы функций интеллектуальной системы, входит в блок слева, а результаты выхода - с правой стороны. Механизмы управления интеллектуальной системой (это может быть эксперт, либо машина), представляется дугой, входящей в блок снизу.



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

Построение SADT-модели начинается с того, что необходимо представить всю систему в целом и как ее можно разбить на компоненты, как их связать, как управлять отдельными функциями и системой в целом, какая необходма входящая информация и что будет в результате. Имя, которое будет указано на центральном функциональном блоке будет общим для всей системы.



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.

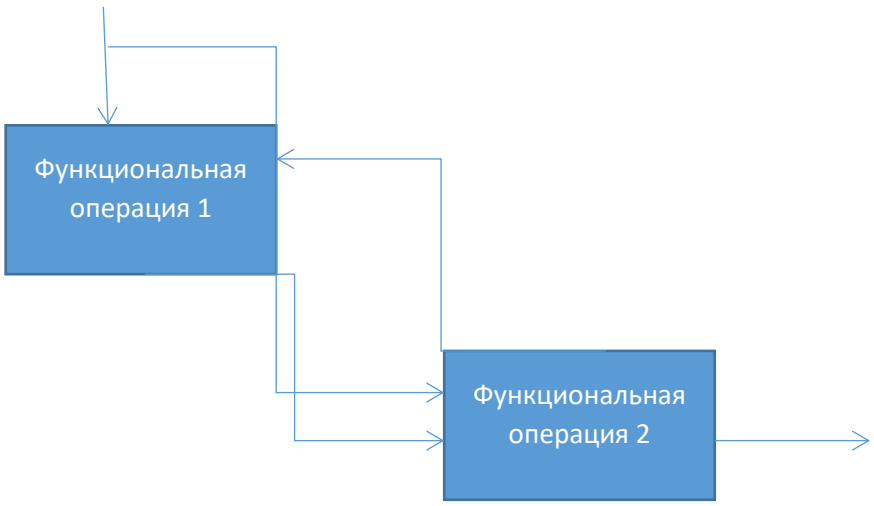
Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию. Кроме того, модель не может опустить какие-либо элементы, т.е., как уже отмечалось,

родительский блок и его интерфейсы обеспечивают контекст. К нему нельзя ничего добавить, и из него не может быть ничего удалено.

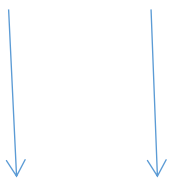
Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других диаграммах. Каждая детальная диаграмма является декомпозицией блока из 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 структурных диаграммах не указаны явно ни последовательность, ни время. Обратные связи, итерации, продолжающиеся процессы и перекрывающиеся функции могут быть изображены с помощью дуг. Обратные связи могут выступать в виде комментариев, замечаний, исправлений и т.д.

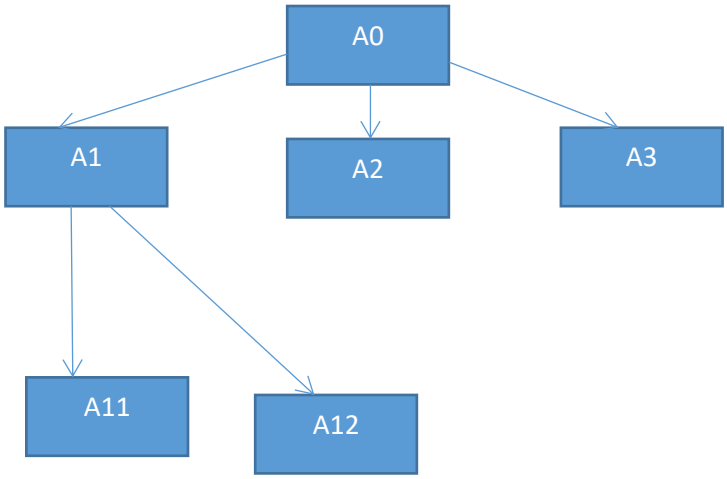


Механизмы показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию



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.

Для того, чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм. Например, A21 является диаграммой, которая детализирует блок 1 на диаграмме A2. Аналогично, A2 детализирует блок 2 на диаграмме A0, которая является самой верхней диаграммой модели.



2. Методология потоков данных. В международной терминологии Data Flow Diagrams (DFD).

Эта методология основана на том, что строится модель интеллектуальной системы, которая требует анализа. При этом система может быть только в проекте или уже быть в эксплуатации.

Диаграммы верхних уровней иерархии определяют основные процессы или подсистемы интеллектуальной системы с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. Такая декомпозиция продолжается, создавая многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процесс становятся элементарными и детализировать их далее невозможно.

Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям - потребителям информации. Таким образом, основными компонентами диаграмм потоков данных являются:

- внешние сущности;
- системы/подсистемы;
- процессы;
- накопители данных;
- потоки данных.

Цель моделирования данных состоит в обеспечении разработчика интеллектуальных систем концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

3. Модели Сущность – связь. В международной терминологии Entity-Relationship Diagrams (ERD).

Этот вид модели наиболее распространен. Здесь определяется наиболее важные объекты задачи, их свойства и взаимосвязи.

Первоначально, для построения модели извлекается информация из данных опроса экспертов и выделение наиболее важных объектов. Т.е. сущностей. В международной терминологии Entity. Информация об этом объекте должна иметь необходимость хранения.

Графически сущность изображается в виде прямоугольника с закругленными углами.

<имя объекта>

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и

отличаться от всех других экземпляров данного типа сущности. Каждая сущность должна обладать некоторыми свойствами:

- каждая сущность должна иметь уникальное имя, и к одному и тому же имени должна всегда применяться одна и та же интерпретация. Одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- сущность обладает одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;
- сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности;
- каждая сущность может обладать любым количеством связей с другими сущностями модели.

Следующим шагом моделирования является идентификация связей.

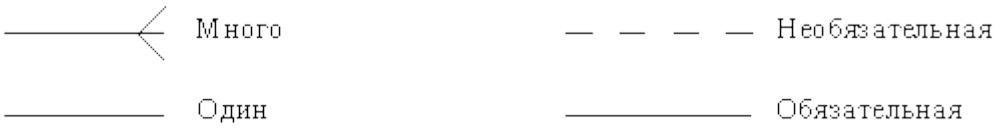
Связь (Relationship) - поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь - это ассоциация между сущностями, при которой, как правило, каждый экземпляр одной сущности, называемой родительской сущностью, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой сущностью-потомком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности родителя.

Связи может даваться имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи. Имя каждой связи между двумя данными сущностями должно быть уникальным, но имена связей в модели не обязаны быть уникальными. Имя связи всегда формируется с точки зрения родителя, так что предложение может быть образовано соединением имени сущности-родителя, имени связи, выражения степени и имени сущности-потомка.

Например, связь продавца с контрактом может быть выражена следующим образом:

- продавец может получить вознаграждение за 1 или более контрактов;
- контракт должен быть инициирован ровно одним продавцом.

Степень связи и обязательность графически изображаются следующим образом (рисунок 2.20).



Последним шагом моделирования является идентификация атрибутов.

Атрибут - любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Экземпляр атрибута - это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. В ER-модели атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Атрибут может быть либо обязательным, либо необязательным Обязательность означает, что атрибут не может принимать неопределенных значений (null values). Атрибут может быть либо описательным (т.е. обычным дескриптором сущности), либо входить в состав уникального идентификатора (первичного ключа).

Уникальный идентификатор - это атрибут или совокупность атрибутов и/или связей, предназначенная для уникальной идентификации каждого экземпляра данного типа сущности. В случае полной идентификации каждый экземпляр данного типа сущности полностью идентифицируется своими собственными ключевыми атрибутами, в противном случае в его идентификации участвуют также атрибуты другой сущности-родителя Атрибуты, определяющие первичный ключ, размещаются наверху списка и выделяются знаком "#".

Каждая сущность должна обладать хотя бы одним возможным ключом. Возможный ключ сущности - это один или несколько атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа, а остальные - как альтернативные ключи.

Подтипы и супертипы: одна сущность является обобщающим понятием для группы подобных сущностей

Взаимно исключающие связи: каждый экземпляр сущности участвует только в одной связи из группы взаимно исключающих связей

Рекурсивная связь: сущность может быть связана сама с собой
Неперемещаемые (non-transferrable) связи: экземпляр сущности не может быть перенесен из одного экземпляра связи в другой

Все эти типы моделей дают полное описание интеллектуальной системы независимо от того, является ли она существующей или проектируемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

Вопросы по теме лекции:

- что такое SADT?
- в чем особенности и преимущества структурного подхода к моделирования бизнес-процессов?
- что дает графическое представление бизнес-процессов?
- какова связь моделирования и автоматизации бизнес-процессов?
- что такое модель бизнес-процесса?
- что такое IDEFx?
- в чем разница IDEF0 и IDEF3 ?
- какую роль играют стороны каждого блока в модели IDEF 0 ?

Литература и материалы по теме:

Тельнов, Ю.Ф. Инжиниринг предприятия и управление бизнес-процессами. Методология и технология: Учебное пособие / Ю.Ф. Тельнов, И.Г. Федоров. - М.: Юнити, 2017. - 304 с.

1.1.1.6 Автоматизированное проектирование ИС с использованием CASE-технологий: объектно-ориентированный подход к проектированию ИС.

Объектно-ориентированная методика. Отличие между функциональным и объектным подходом заключается в способе декомпозиции системы. Объектный подход используется объектную декомпозицию, при этом статическая структура описывается в терминах объектов и связей между ними, а поведение системы в терминах обмена сообщениями между объектами.

Концептуальной основой объектного подхода является объектная модель, строящаяся по принципам:

- абстрагирование,
- инкапсуляция,
- модульность,
- иерархия,
- типизация,
- параллелизм,
- устойчивость.

Основными понятиями являются объект и класс. Объект – предмет или явление, имеющее четкое определенное поведение и обладающее состоянием, поведением и индивидуальностью. Класс – это множество объектов, связанных общностью структуры и поведения. Полиморфизм – способность класса принадлежать более чем одному типу. Наследование – построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов.

Методы объектного подхода включают язык моделирования и описание процесса моделирования. Процесс – описание шагов, которые необходимо выполнить при разработке проекта. В качестве языка используется унифицированный язык моделирования UML, который содержит стандартный набор диаграмм для моделирования. Диаграмма – графическое представление множества элементов. Чаще всего изображается в виде связного графа с вершинами (сущностями) и ребрами (отношениями) и представляет собой некоторую проекцию системы.

Преимущества объектно-ориентированного подхода:

- объектная декомпозиция дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих некоторую экономию выразительных средств. Повышается уровень унификации

разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей.

- объектная декомпозиция позволяет избежать создания сложных моделей, т.к. предполагает эволюционный путь развития модели на базе относительно небольших систем.

- объектная модель естественна, поскольку ориентирована на человеческое восприятие мира.

К недостаткам относятся высокие начальные затраты. Этот подход не дает немедленной отдачи. Диаграммы отражающие специфику объектного подхода менее наглядны.

Сравнение методик. В функциональных моделях (DFD-диаграммах потоков данных, SADT-диаграммах) главными структурными компонентами являются функции (операции, действия, работы), которые на диаграммах связываются между собой потоками объектов. Их достоинством является реализация структурного подхода к проектированию ИС по принципу сверху-вниз, когда каждый функциональный блок может быть декомпозирован на множество подфункций, выполняя модульное проектирование ИС. Для функциональных моделей характерны процедурная строгость декомпозиции ИС и наглядность представления. При функциональном подходе модели данных разрабатываются отдельно. Для проверки корректности моделирования предметной области между функциональными и объектными моделями устанавливаются взаимно однозначные связи. Главный недостаток в том, что процессы и данные существуют отдельно друг от друга – помимо функциональной декомпозиции существует структура данных, находящаяся на втором плане. Кроме того, не ясны условия выполнения процессов обработки информации, которые динамически могут изменяться.

Перечисленные недостатки функциональных моделей снимаются в объектно-ориентированных моделях, где главным структурирующим компонентом выступает класс объектов с набором функций, которые могут обращаться к атрибутам этого класса. Для классов объектов характерна иерархия обобщения, позволяющая осуществлять наследование не только атрибутов объектов от вышестоящего класса, но и функций (методов). При объектно-ор. подходе изменяется и принцип проектирования ИС. Сначала выделяются классы, а далее в зависимости от возможных состояний объектов определяются методы обработки, что обеспечивает наилучшую реализацию динамического поведения интеллектуальной системы.

Функциональные модели больше подходят для более регламентированных задач, а объектно-ориентированные для более адаптивных процессов. Однако в рамках одной и той же ИС для различных классов задач могут требоваться различные виды моделей, описывающих одну и ту же

предметную область. В таком случае должны использоваться комбинированные модели предметной области.

Объекты в рамках объектно-ориентированного подхода при проектировании интеллектуальных систем обладают такими свойствами как:

- инкапсуляция. Информация об объекте носит скрытый характер. Состав и свойства его атрибутов не зависит от какой либо информации, которая поступает из внешней среды.

- наследование. Все объекты структурируются такие образом, чтобы была видна иерархия и можно было увидеть родительские и дочерние объекты. Логически следуя из принципов иерархии приходим к выводу, что все свойства родительского объекта переходят к дочернему. При этом дочерние объекты могут иметь и общие и частные методы.

- полиморфизм. При получении какого либо сообщения объект может выбирать какой метод выбирать для ответа на это сообщение. Этот выбор зависит от свойств, характера и прочих факторов конкретного сообщения.

Все эти свойства способствуют возможности параллельно разрабатывать компоненты и модули проектируемой интеллектуальной системы таким образом, чтобы эти модули проектировались, создавались и работали автономно. Появляется возможность создания множества прототипов отдельных модулей и затем можно собирать интеллектуальную систему как конструктор из тех прототипов , которые лучше всего отвечают требованиям. Здесь реализуется итерационный метод проектирования интеллектуальной системы.

Несомненный плюс такого подхода состоит также и в том, что накапливается целая база прототипов модулей интеллектуальных систем. Многие их которых можно использовать как типовые в дальнейшем. Это сэкономит время и средства при разработке. Эта особенность связана с тем, что классы объектов повторяются в определенной мере при переходе от одной интеллектуальной системы к другой, а для повторяющихся классов уже запрограммированы методы, разработаны и описаны структуры объектов данных

Модель проектирования интеллектуальной системы при использовании объектно-ориентированного подхода формируется в несколько стадий.

1. Изначально проект проходит стадию анализа . Здесь определяются объекты, классы, осуществляется декомпозиция интеллектуальной системы.

2. Следующая стадия - проектирование. Здесь разрабатываются структуры данных, методы реагирования объектов, отношения между классами и сценарии того, как объекты будут взаимодействовать друг с другом.
3. Программирование. Здесь осуществляется непосредственная разработка программного обеспечения. Отдельных его компонентов. Затем тестирование, отладка и сборка интеллектуальной системы из разработанных компонент.
4. Модификация. Здесь нет необходимости полной переделки всей системы при неполадках или возникновении новых требования. Модифицируются лишь отдельные компоненты, классы и объекты. Это так же заслуга объектно-ориентированного подхода.

Стадии на рисунке намеренно не пронумерованы и не соединены стрелками. Отличительной чертой модели объектно-ориентированного проектирования является отсутствие строгой последовательности в выполнении стадий как в прямом, так и в обратном направлениях процесса проектирования по отдельным компонентам.

Основное преимущество объектно-ориентированного подхода состоит в упрощении проектирования интеллектуальной системы при наличии типовых проектных решений по отдельным компонентам, а также легкости модификации, поскольку модификация касается лишь отдельных компонент.

Следует отметить, что объектно-ориентированный подход трудно воспринимается пользователями и заказчиками и прежде всего предназначен для программистов. Пользователям понятнее функционально-ориентированный подход. Не специалисту проще видеть проект интеллектуальной системы с функциональными блоками и связями. Экономическая эффективность применения объектно-ориентированного подхода возрастает по мере приобретения опыта у разработчиков в большей мере, чем при функционально-ориентированном подходе. Также значительно снижается время и стоимость разработки.

Важное место в моделировании информационных систем занимает методология и системы, использующие UML — унифицированный язык моделирования (Unified Modeling Language).

UML — язык для спецификации, визуализации, конструирования и документирования сложных информационно-насыщенных объектных

систем. В настоящее время зарегистрирован как международный стандарт ISO/IEC 19501:2005 «Information technology - Open Distributed Processing - Unified Modeling Language (UML)».

UML-модель может включать в себя следующие аспекты:

1. Структурный аспект: Use-Case-диаграммы, идентифицирующие бизнес-процессы и бизнес-транзакции, их взаимосвязь, соподчиненность и взаимодействие; Package-диаграммы, описывающие структуру предметной области и иерархическую структуру организации.
2. Динамический аспект: Behavior-диаграммы (Activity, Statechart, Collaboration, Sequence), описывающие поведение (жизненный цикл) бизнес-процессов в их взаимодействии во времени и в пространстве с привязкой к используемым ресурсам и получаемым результатам.
3. Статический аспект: Class-диаграммы, отражающие совокупность взаимосвязанных объектов, т. е. рассматривающие логическую структуру предметной области, ее внутренние концепции, иерархию объектов и статические связи между ними, структуры данных и объектов; Deployment-диаграммы, отражающие технологические ресурсы организации.

Словарь языка UML включает три вида строительных блоков:

- сущности;
- отношения;
- диаграммы.

Сущности в UML — это абстракции, являющиеся основными элементами модели. Отношения связывают различные сущности; диаграммы группируют представляющие интерес совокупности сущностей.

В UML имеется четыре типа сущностей:

- структурные;
- поведенческие;
- группирующие;
- аннотационные.

Структурные сущности — это имена существительные в моделях на языке UML. Как правило, они представляют собой статические части модели, соответствующие концептуальным или физическим элементам системы. Существует семь разновидностей структурных сущностей: Класс, Интерфейс, Кооперация, Прецедент, Активный класс, Компонент, Узел.

Поведенческие сущности являются динамическими составляющими модели UML. Это глаголы языка: они описывают поведение модели во времени и пространстве. Существует всего два основных типа поведенческих сущностей: Взаимодействие и Автомат.

Группирующие сущности являются организующими частями модели UML. Это блоки, на которые можно разложить модель. Есть только одна первичная группирующая сущность, а именно — пакет.

Аннотационные сущности — пояснительные части модели UML. Это комментарии для дополнительного описания, разъяснения или замечания к любому элементу модели. Имеется только один базовый тип аннотационных элементов — примечание.

Все разновидности сущностей UML в диаграммах имеют свой способ графического изображения.

В языке UML определены четыре типа отношений:

- зависимость;
- ассоциация;
- обобщение;
- реализация.

Диаграмма в UML — это графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации системы с разных точек зрения. Теоретически диаграммы могут содержать любые комбинации сущностей и отношений. На практике, однако, применяется сравнительно небольшое количество типовых комбинаций, соответствующих пяти наиболее употребительным видам, которые составляют архитектуру ИС. Таким образом, в UML выделяют девять типов диаграмм:

- диаграммы классов (Class Diagrams);
- диаграммы объектов (Objects Diagrams);
- диаграммы прецедентов (Use Cases Diagrams);
- диаграммы последовательностей (Sequence Diagrams);
- диаграммы кооперации (Collaboration Diagrams);
- диаграммы состояний (State Diagrams);
- диаграммы действий (Activity Diagrams);
- диаграммы компонентов (Component Diagrams);
- диаграммы развертывания (Deployment Diagram).

К инструментам, поддерживающим методологию UML, относятся Rational Rose (Rational Software), Paradigm Plus (CA/Platinum), ARIS (IDS Sheer AG), Together Designer (Borland) и т. д.

Система Rational Rose позволяет создавать восемь типов диаграмм UML: диаграммы вариантов использования, диаграммы классов, диаграммы последовательности, диаграммы сотрудничества, диаграммы состояний, диаграммы действий, диаграммы компонентов, диаграммы развертывания. Основным типом диаграмм, своего рода ядром моделирования в UML, являются диаграммы классов. Кроме UML предусмотрено использование и других методов (Booch, OMT).

Система Paradigm Plus ориентирована на методологию OOCL (объектно-ориентированное изменение и обучение) и технологию проектирования и разработки компонентов. Он поддерживает различные диаграммы методов (UML, CLIPP, TeamFusion, OMT, Booch, OOCL, Martin/Odell, Shlaer/Mellor, Coad/Yourdon).

Система ARIS предоставляет четыре различных «представления» для моделирования и анализа: процессы, функции (с целями), данные, организация. Для каждого «представления» поддерживаются три уровня анализа (требования, спецификации, реализация). Каждый из уровней анализа состоит из своего набора моделей различного типа, в том числе диаграмм UML, диаграмм SAP/R3 и т. д. Каждый объект модели ARIS имеет множество атрибутов, позволяющих контролировать процесс разработки модели, определять условия выполнения функциональных анализ затрат, имитационное моделирование, взаимодействие с системами документооборота и др.

Together Designer Community Edition — это инструмент для построения диаграмм UML 2.0. Позволяет строить диаграммы вариантов использования (диаграммы сценариев взаимодействия пользователя с продуктом с точки зрения пользователя); диаграммы последовательности, описывающие порядок, в котором сообщения передаются от одного объекта к другому; диаграммы кооперации, описывающие взаимодействие объектов друг с другом, диаграммы деятельности, описывающие рабочие процессы и изменения состояний объектов, диаграммы развертывания. При необходимости может быть создана логическая модель данных, содержащая диаграммы сущность-связь; на его основе формируется физическая модель данных для конкретной СУБД, выбранной для реализации проекта.

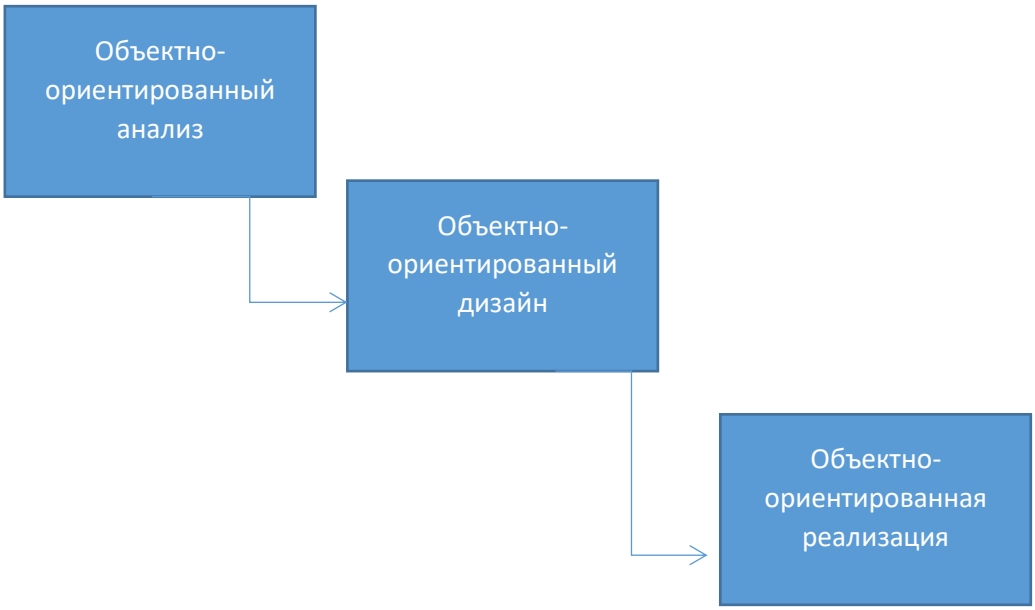
На этапе создания клиентского и серверного кода все современные средства UML-моделирования могут генерировать код на различных языках программирования.

Использование UML

UML используется и особенно эффективен в следующих задачах:

- Моделирование бизнес-процесса
- Описание архитектуры системы
- Отображение структуры приложения
- Захват поведения системы
- Моделирование структуры данных
- Построение подробных спецификаций системы
- Зарисовка идей
- Генерация программного кода

Жизненный цикл разработки объектно-ориентированной системы отличается от жизненного цикла систем при других подходах. Выделяют следующие этапы жизненного цикла:



1. Объектно ориентированный анализ.

   На этой стадии определяются системные требования. Так как понимание этих требований необходимо для построения диаграммы вариантов использования. Т.е. сценария, который описывает взаимодействие между интеллектуальной системой и человеком-пользователем. На этой модели можно составить представление о том, какие потребности у пользователей и как пользователь представляет себе будущую интеллектуальную системы.

   Также здесь определяются классы и их коммуникацию друг с другом. Именно их этих классов предметной области будет состоять интеллектуальная система.

2. Объектно-ориентированный дизайн

   На этой стадии осуществляется разработка и уточнение классов, атрибутов, методов и структур, идентифицированных на этапе анализа, пользовательского интерфейса и доступа к данным. Здесь также определяются дополнительные классы или объекты, необходимы для реализации системы требования.

3. Объектно ориентированная реализация.

   Эта стадия включает в себя несколько процессов. Прототипирование. Позволяет полностью понять, насколько легко или сложно будет реализовать некоторые функции системы. А также может дать

пользователям возможность оставить свои отзывы и предложения по удобству использования и полезности проектируемой интеллектуальной системы. Это может дополнительно определить вариант использования и значительно упростить его моделирование.

Выполнение. Здесь применяется либо разработку на основе компонентов (CBD), либо быструю разработку приложений (RAD).Компонентная разработка (CBD) CODD — это промышленный подход к процессу разработки программного обеспечения с использованием различных технологий, таких как инструменты CASE. Разработка приложений переходит от индивидуальной разработки к сборке готовых, предварительно протестированных, многоразовых программных компонентов, которые работают друг с другом. Разработчик CBD может собрать компоненты для создания полной программной системы. Быстрая разработка приложений (RAD). RAD — это набор инструментов и методов, которые можно использовать для создания приложения быстрее, чем это обычно возможно с помощью традиционных методов. Он не заменяет SDLC, а дополняет его, поскольку он более ориентирован на процесс и может идеально сочетаться с объектно-ориентированным подходом. Его задача заключается в быстрой и поэтапной реализации приложения для реализации дизайна требований пользователя с помощью таких инструментов, как Visual Basic, Power Builder и т. д.

Инкрементное тестирование. Разработка программного обеспечения и все его действия, включая тестирование, представляют собой повторяющийся процесс. Поэтому может быть дорого, если мы будем ждать, пока продукт не будет полностью разработан, чтобы протестировать продукт. Здесь в дело вступает инкрементальное тестирование, при котором продукт тестируется на разных этапах его разработки.

Вопросы по теме лекции:

- что такое ООП?
- в чем особенности и преимущества объектно-ориентированного подхода к моделирования и разработке?
- что дает графическое представление объектов?
- какова связь моделирования и автоматизации?
- что такое класс?
- как связаны класс и объект?
- какие могут быть отношения между классами?
- что такое принципы SOLID?
- в чем разница агрегации и ассоциации?
- что такое UML?
- какие диаграммы включает UML?

- можно ли применять объектно-ориентированный подход на различных стадиях анализа, проектирования и разработки интеллектуальных информационных систем?

Литература и материалы по теме:

Лафоре, Р. Объектно-ориентированное программирование в C++ / Р. Лафоре. - СПб.: Питер, 2019. - 928 с.

Гради Буч и др. Объектно-ориентированный анализ и проектирование с примерами приложений (UML 2). Третье издание. — М.: «Вильямс», 2008. — 720 с.

1.1.1.7 Методы и средства прототипного проектирования ИС: технология быстрого проектирования ИС.

При любом проектировании, а при проектировании интеллектуальных информационных систем целесообразно использовать методику, при которой создаются прототипы будущих систем.

Таким образом и разработчику , и заказчику и конечному пользователю будет виден результат, будет легче увидеть дополнительные или наоборот убрать ненужные требования к системе. И в конечном итоге выбрать лучший результат, который наиболее полно отвечает поставленной задаче.

В современных условиях заказчики ожидают быстрого результата. Да и разработчики заинтересованы в быстрой разработке и отдаче проектов. Более того иногда это является основным условием заказчика. А может даже и единственным. Этапы жизненного цикла интеллектуальной системы, которые были рассмотрены ранее, становятся все более короткими. Требования и условия работы настолько быстро меняются , что уже невозможно модифицировать готовую систему. Гораздо быстрее и эффективнее разработать новую интеллектуальную систему.

Создание прототипов становится дорогим и долгим методом проектирования.

Перед началом проектирования интеллектуальной системы необходимо собрать максимально возможный и полный объем информации, которая будет использована для построения интеллектуальной системы. А также информации, которая будет обрабатываться системой.

Например:
- информационный поиск;
- сложные расчеты;
- графика;
- обработка текстов.

Решение этих задач заблаговременно, во-первых, подготавливает высококвалифицированных специалистов по проектированию и разработке интеллектуальной системы, во-вторых, позволяет отделить от важные задачи.

Проектирование интеллектуальной информационной системы по технологии быстрого проектирования проходит в несколько этапов.

1. Выбор подходящей проблемы

На этом этапе необходимо определить предметную область, задачи которые нужно решить в этой конкретной области, произвести опрос экспертов.

Если таковых нет, то необходимо опросить экспертов со стороны. Определяется круг задач, масштабы работы и сколько потребуется разработчиков. Определяется какой подход будет использован для решения проблемы, какой бюджет потребует вся разработка и какую планируется получить прибыль от продажи либо использования разрабатываемой интеллектуальной системы.

Также на этом этапе необходимо составить план разработки во всех деталях.

Правильный выбор проблемы. Правильная ее формулировка является одним из самых важных моментов в процессе разработки системы. Если неправильно осуществить постановку заданной проблемы, проектирования задачи может длиться бесконечно долго и не приводить к результату. Неправильно поставленная проблема может также привести к созданию интеллектуальной системы, которая стоит намного больше, чем экономит или чем это было запланировано. Также может быть разработана система, которая работает, но не приемлема для пользователей. Даже если разработка выполняется самой организацией для собственных целей, эта фаза является подходящим моментом для получения рекомендаций извне, чтобы гарантировать удачно выбранный и осуществимый с технической точки зрения первоначальный проект.

При выборе области применения следует учитывать, что если знание, необходимое для решения задач, постоянное, четко формулируемое и связано с вычислительной обработкой, то обычные алгоритмические программы, по всей вероятности, будут самым целесообразным способом решения проблем в этой области.

Интеллектуальная система не устранит потребность в реляционных базах данных, статистическом программном обеспечении, электронных таблицах и системах текстовой обработки. Но если результативность задачи зависит от знания, которое является субъективным, изменяющимся, символьным, тогда область может обоснованно выступать претендентом на интеллектуальную систему.

Обоснование необходимости разработки и внедрения интеллектуальных систем:

- нехватка специалистов, расходующих значительное время для оказания помощи другим;
- потребность в многочисленном коллективе специалистов, поскольку ни один из них не обладает достаточным знанием;
- сниженная производительность, поскольку задача требует полного анализа сложное набора условий, а обычный специалист не в состоянии просмотреть (за отведенное время) все эти условия;
- большое расхождение между решениями самых хороших и самых плохих исполнителей;
- наличие конкурентов, имеющих преимущество в том, что они лучше справляются с поставленной задачей.

Подходящие задачи имеют следующие характеристики:

- являются узкоспециализированными;

- не зависят в значительной степени от общечеловеческих знаний или соображении здравого смысла;

- не являются для эксперта ни слишком легкими, ни слишком сложными (время, необходимое эксперту для решения проблемы, может составлять от трех часов до трех недель);

- условия исполнения задачи определяются самим пользователем системы;

- имеет результаты, которые можно оценить.

Обычно интеллектуальные системы разрабатываются путем получения специфических знаний от эксперта и ввода их в систему. Некоторые системы могут содержать стратегии одного индивида. Следовательно, найти подходящего эксперта - это ключевой шаг в создании интеллектуальных систем.

В процессе разработки и последующего расширения системы инженер по знаниям и эксперт обычно работают вместе. Инженер по знаниям помогает эксперту структурировал знания, определять и формализовать понятия и правила, необходимые для решения проблемы.

Во время первоначальных бесед они решают, будет ли их сотрудничество успешным. Это немаловажно, поскольку обе стороны будут работать вместе по меньшей мере в течение одного года. Кроме них в коллектив разработчиков целесообразно включить потенциальных пользователей и профессиональных программистов.

Предварительный подход к программной реализации задачи определяется исходя из характеристик задачи и ресурсов, выделенных на ее решение. Инженер по знаниям выдвигает обычно несколько вариантов, связанных с использованием имеющихся на рынке программных средств.

Окончательный выбор возможен лишь на этапе разработки прототипа. После того как задача определена, необходимо подсчитать расходы и прибыли от разработки интеллектуальной системы. В расходы включаются затраты на оплату труда коллектива разработчиков. В дополнительные расходы приобретаемого программного инструментария, с помощью которого разрабатывается интеллектуальная система.

Прибыль возможна за счет снижения цены продукции, повышения производительности труда, расширения номенклатуры продукции или услуг или даже разработки новых видов продукции или услуг в этой области.

Соответствующие расходы и прибыли от системы определяются относительно времени, в течение которого возвращаются средства, вложенные в разработку. На современном этапе большая часть фирм, развивающих крупные интеллектуальные системы, предпочли разрабатывать дорогостоящие проекты, приносящие значительные прибыли.

Наметились тенденции разработки менее дорогостоящих систем, хотя и с более длительным сроком возвращаемости вложенных в них средств, так как программные средства разработки экспертных систем непрерывно совершенствуются.

После того как инженер по знаниям убедился, что:

- данная задача может быть решена с помощью интеллектуальной системы;
- интеллектуальную систему можно создать предлагаемыми на рынке средствами;
- имеется подходящий эксперт;
- предложенные критерии производительности являются разумными;
- затраты и срок их возвращаемости приемлемы для заказчика;

Далее составляется детальный план разработки. План определяет шаги процесса разработки и необходимые затраты, а также ожидаемые результаты.

2. Разработка прототипной системы

Как уже было рассмотрено прототип является образом будущей интеллектуальной системы. Такой какой она представляется всем участникам ее разработки. И разумеется пользователям.

Именно на этом этапе необходимо участие эксперта. Приглашенного или штатного. Именно эксперт может сразу увидеть неточности в выполнении требований, некорректности работы, выбрать тот прототип, который наиболее полно соответствует всем требованиям.

Прототип не является полноценной системой. Для его оценки достаточно некоторого количества основных правил, примеров и задач, которые будет решать интеллектуальная система.

На этом этапе происходит наглядное выявление проблемы, которую необходимо решить при помощи интеллектуальной системы.

Уточняется задача, планируется ход разработки прототипа интеллектуальной системы, определяются:

- необходимые ресурсы. Это и временной ресурс, и человеческий, и финансовый, и технический;
- источники знаний. Может возникнуть необходимость привлечения дополнительных экспертов из смежных областей. Могут потребоваться книги или документы, без которых невозможно получить требуемую информацию.
- имеющиеся аналогичные интеллектуальные или экспертные системы, для анализа опыта их использования и учета возможных недостатков.;
- цели (распространение опыта, автоматизация рутинных действий и др.);
- классы решаемых задач и т.д.

Идентификация проблемы - знакомство и обучение коллектива разработчиков, а также создание неформальной формулировки проблемы.

Средняя продолжительность 1 - 2 недели.

Также на этом этапе происходит добыча знаний из всех истоников, включая экспертов.

Происходит перенос компетентности экспертов на инженеров по знаниям с использованием различных методов:

- анализ текстов;
- диалоги;
- экспертные игры;
- лекции;
- дискуссии;
- интервью;
- наблюдение и другие.

Извлечение знаний - получение инженером по знаниям наиболее полного представления о предметной области и способах принятия решения в ней.

Средняя продолжительность 1 -3 месяца.

Затем необходимо структурировать добытые знания.

Выявляется структура полученных знаний о предметной области, т.е. определяются:

- терминология;
- список основных понятий и их атрибутов;
- отношения между понятиями;
- структура входной и выходной информации;
- стратегия принятия решений;
- ограничения стратегий и т.д.

Концептуализация знаний - разработка неформального описания знаний о предметной области в виде графа, таблицы, диаграммы или текста, которое отражает основные концепции и взаимосвязи между понятиями предметной области.

Такое описание называется полем знаний. Средняя продолжительность этапа 2 - 4 недели.

После структурирования происходит формализация тех данных , которые этого требуют.

Традиционно на этом этапе используются:

- логические методы ;
- продукционные модели;
- семантические сети;
- фреймы;
- объектно-ориентированные языки, основанные на иерархии классов, объектов и др.

Формализация знаний - разработка базы знаний на языке, который, с одной стороны, соответствует структуре поля знаний, а с другой – позволяет реализовать прототип системы на следующей стадии программной реализации.

Средняя продолжительность 1 - 2 месяца.

И, наконец, реализация разработки.

Создается прототип интеллектуальной системы, включающий базу знаний и остальные блоки, при помощи одного из следующих способов:

- программирование на традиционных языках типа Паскаль, Си и др.;

- программирование на специализированных языках, применяемых в задачах искусственного интеллекта;

- использование инструментальных средств разработки интеллектуальных систем;

- использование пустых экспертных систем .

Реализация - разработка программного комплекса, демонстрирующего жизнеспособность подхода в целом. Чаще всего первый прототип отбрасывается на этапе реализации действующей интеллектуальной системы.

Средняя продолжительность 1 - 2 месяца.

На этом этапе происходит и тестирование. Здесь оценивается и проверяется работа программ прототипа с целью приведения в соответствие с реальными запросами пользователей. Прототип проверяется на: удобство и адекватность интерфейсов ввода-вывода (характер вопросов в диалоге, связность выводимого текста результата и др.)

- эффективность стратегии управления (порядок перебора, использование нечеткого вывода и др.);

- качество проверочных примеров;

- корректность базы знаний (полнота инепротиворечивость правил).

Тестирование - выявление ошибок в подходе и реализации прототипа и выработка рекомендаций по доводке системы до промышленного варианта.

Средняя продолжительность 1 - 2 недели.

3. Развитие прототипа до промышленной интеллектуальной системы

При неудовлетворительном функционировании прототипа эксперт и инженер по знаниям имеют возможность оценить, что именно будет включено в разработку окончательного варианта системы.

Если первоначально выбранные объекты или свойства оказываются неподходящими, их необходимо изменить. Можно сделать оценку общего числа эвристических правил, необходимых для создания окончательного варианта интеллектуальной системы. Иногда при разработке промышленной системы выделяют дополнительные этапы для перехода: демонстрационный прототип - исследовательский прототип - действующий прототип - промышленная система.

Однако чаще реализуется плавный переход от демонстрационного прототипа к промышленной системе, при этом, если программный инструментарий выбран удачно, необязательна перепись другими программными средствами.

Понятие же коммерческой системы в нашей стране входит в понятие промышленный программный продукт, или промышленной интеллектуальной системы.

Таблица. Переход от прототипа к промышленной интеллектуальной системе

Демонстрационный прототип интеллектуальной системы

Система решает часть задач, демонстрируя ж из неспособность полхода (несколько десятков правил или понятий)

Исследовательский прототип интеллектуальной системы

Система решает большинство задач, но не устойчива в работе и не полностью проверена.

Действующий прототип интеллектуальной системы

Система надежно решает все задачи на реальных примерах, но для сложной задачи требует много времени и памяти

промышленная система

Система обеспечивает высокое качество решений при минимальных затратах времени и памяти: переписана с использованием более эффективных средств представления знаний.

Коммерческая система

Промышленная система, подходящая для продажи, т. е. хорошо задокументированная и обслуживаемая

Главное на третьем этапе добавить большое количество дополнительных эвристик. Эти эвристики обычно увеличивают глубину системы, предоставляя больше правил для тонких аспектов отдельных случаев. В то же время эксперт по знаниям и инженер могут расширить область действия системы, включив в нее правила, управляющие дополнительными подзадачами или дополнительными аспектами интеллектуальной задачи (метазнания).

После создания базовой структуры интеллектуальной системы инженер по знаниям приступает к разработке и адаптации интерфейсов, через которые система будет общаться с пользователем и экспертом. Необходимо обратить особое внимание на языковые возможности интерфейсов, их простоту и удобство для управления работой интеллектуальной системы. Система должна предоставлять пользователю возможность легко и непринужденно задавать непонятные вопросы, приостанавливать работу и т. д. В частности, могут быть полезны графические представления.

На этом этапе разработки большинство экспертов будет достаточно знать о вводе правил и сможет самостоятельно вводить новые правила в систему. Таким образом, начинается процесс, в ходе которого инженер по знаниям передает право собственности и управление системой эксперту для доработки, детальной разработки и сопровождения.

4. Оценка системы

На этапе оценки разработанной интеллектуальной системы нужно просчитать экономический эффект от внедрения ее на предприятии. Для этого может потребоваться привлечение экспертов их смежных областей, работа которых непосредственно не связана с интеллектуальной системой, но затрагивает их профессиональную деятельность и влияет на ее эффективность.

Интеллектуальные системы оцениваются главным образом для того, чтобы проверить точность работы программы и ее полезность. Оценку можно

проводить, исходя из различных критериев, которые сгруппируем следующим образом:

- критерии пользователей (понятность и простота работы системы, удобство интерфейсов и др.);
- критерии приглашенных экспертов (оценка решений, предлагаемых системой, сравнение ее с собственными решениями, оценка системы объяснений выданных результатов и др.);
- критерии коллектива разработчиков (эффективность реализации, производительность, время отклика, дизайн, широта охвата предметной области, непротиворечивость, количество тупиковых ситуаций, когда система не может принять решение, анализ чувствительности программы к незначительным изменениям в представлении знаний, весовых коэффициентах, применяемых в механизмах логического вывода, данных и т.п.).

5.Стыковка системы

На этом этапе осуществляется стыковка разработанной интеллектуальной системы с элементами программной среды. Также необходимо определить должностные обязанности сотрудников, которые будут работать с этой системой и произвести их обучение.

В некоторых случаях при внедрении новой интеллектуальной системы в информационную среду предприятия может возникнуть необходимость изменить часть или даже всю информационную среду.

Для этого могут потребоваться услуги дополнительных специалистов и дополнительный бюджет.

Также во время этого этапа прорабатываются связи новой интеллектуальной системы с остальными элементами , а также и с внешней средой.

Когда экспертная система уже готова, инженер по знаниям должен убедиться в том, что эксперты, пользователи и персонал знают, как эксплуатировать и обслуживать ее. После передачи им своего опыта в области информационной технологии инженер по знаниям может полностью предоставить ее в распоряжение пользователей.

Для подтверждения полезности системы важно предоставить каждому из пользователей возможность поставить перед интеллектуальной системой реальные задачи, а затем проследить, как она выполняет эти задачи. Чтобы система была одобрена, необходимо представить ее как помощника, освобождающего пользователей от обременительных задач, а не как средство их замещения.

Стыковка включает обеспечение связи интеллектуальной системы с существующими базами данных и другими системами на предприятии, а также улучшение системных факторов, зависящих от времени, чтобы можно было обеспечить ее более эффективную работу и улучшить характеристики ее технических средств, если система работает в необычной среде (например, связь с измерительными устройствами).

6. Поддержка системы

При планировании поддержки интеллектуальной системы необходимо определить насколько изменяющейся является предметная область.

Если данные в системе будут меняться необходимо предусмотреть, что инструмент разработки должен оставаться единым и перекодировка невозможна. Это снижает адаптивность системы к различным условиям, но в некоторых случаях неизбежно.

Вопросы по теме лекции:

- что такое прототип?
- в чем разница прототипа и готовой информационной системы?
- в чем опасность применения прототипов в цикле разработки информационных систем?
- что дает применение прототипов?
- каковы средства прототипирования?
- можно ли применять прототипирование на различных стадиях проектирования и разработки интеллектуальных информационных систем?

Литература и материалы по теме:

Лафоре, Р. Объектно-ориентированное программирование в C++ / Р. Лафоре. - СПб.: Питер, 2019. - 928 с.

Гради Буч и др. Объектно-ориентированный анализ и проектирование с примерами приложений (UML 2). Третье издание. — М.: «Вильямс», 2008. — 720 с.

1.1.1.8 Методы и средства прототипного проектирования ИС: методы и средства организации метаинформации проекта ИС.

Создание и управление сложной системой, в том числе программной системой предполагает принятие решений в условиях действия большого числа внешних факторов, наличия множества взаимодействующих элементов управляемой системы и ориентировано на достижение комплекса различных целей. Средством обеспечения эффективности принимаемых решений служит моделирование.

Модель - искусственный объект, представляющий собой отображение (образ) системы и ее компонентов. Считается, что М моделирует А, если М отвечает на вопросы относительно А. Здесь М - модель, А - моделируемый объект (оригинал).

Определение основано на принятом стандарте для структурного моделирования (Рекомендации по стандартизации. – М.: ГОССТАНДАРТ РОССИИ, 2001.). При этом понятие модели далеко выходит за пределы задач проектирования информационных систем.

Разработка сколько-нибудь сложных программных систем превышает возможности интуитивного конструирования программного кода и требует привлечения методов, обеспечивающих его качество.

Необходимость создания защищенного программного кода обусловлена следующими комплексами факторов, влияющих на его эксплуатацию:

- ошибки программного кода;
- действия природных факторов и ненамеренных действий пользователя;
- действия злоумышленника по атаке на программные компоненты;
- изменения требований к программному обеспечению.

Типовая ИС в специальной базе метаинформации - репозитории - содержит модель объекта автоматизации, на основе которой осуществляется конфигурирование программного обеспечения.

Таким образом, модельно-ориентированное проектирование ИС предполагает, прежде всего, построение модели объекта автоматизации с использованием специального программного инструментария (например, SAP Business Engineering Workbench (BEW), BAAN Enterprise Modeler).

Возможно также создание системы на базе типовой модели ИС из репозитория, который поставляется вместе с программным продуктом и расширяется по мере накопления опыта проектирования информационных систем для различных отраслей и типов производства.

Репозиторий содержит базовую (ссылочную) модель ИС, типовые (референтные) модели определенных классов ИС, модели конкретных ИС предприятий.

Базовая модель ИС в репозитории содержит описание бизнес-функций, бизнес-процессов, бизнес-объектов, бизнес-правил, организационной структуры, которые поддерживаются программными модулями типовой ИС.

Типовые модели описывают конфигурации информационной системы для определенных отраслей или типов производства.

Модель конкретного предприятия строится либо путем выбора фрагментов основной или типовой модели в соответствии со специфическими особенностями предприятия (BAAN Enterprise Modeler), либо путем автоматизированной адаптации этих моделей в результате экспертного опроса (SAP Business Engineering Workbench).

Таким образом, описываемый подход предполагает работу не столько с кодом информационных систем, сколько с ее моделями на разных этапах развития для дальнейшего реализации и отражения в виде программного кода и иных артефактов.

Главный принцип: все необходимая информация фиксируется в репозитории системы в виде артефактов, все изменения фиксируются и могут быть обращены.

Модели могут быть преобразованы друг в друга, например, концептуальная модель порождает логическую, а та в свою очередь, физическую, непосредственно привязанную к выбранному инструментарию реализации, например, к системе управления базами данных MS SQL или Oracle.

Визуальные модели позволяют:

- осмысливать разрабатываемые программные решения;
- обеспечивать коммуникацию разработчиков, проектировщиков, заказчиков;
- документировать программную систему;
- автоматически генерировать программный код;
- проводить реверс-инжиниринг программного кода.

Объектно-ориентированное программное обеспечение обладает определенными преимуществами с точки зрения преодоления сложности решаемых задач.

Объектная декомпозиция позволяет разбить системы на небольшие, относительно независимые компоненты, более устойчивые к происходящим изменениям.

Принципы объектно-ориентированного программирования (абстракция, инкапсуляция, наследование, полиморфизм) позволяют более детально моделировать в рамках программной системы предметную область решаемой задачи, а значит, и снизить «семантический разрыв» между предметной областью и средствами ее выражения в программе.

Проектирование информационных систем возможно с привлечением типовых решений, принятых для тех или иных классов задач «паттернов».

Также можно отметить подход, основанный на использовании сторожевых начальных и конечных условий каждого метода. Предполагается, что начальные условия вызова метода полностью выполняются, что проверяется перед исполнением метода. После исполнения метода должны соблюдаться конечные условия, что также проверяется в программе. Инвариант класса – это условия, которое должно соблюдаться всегда, вне зависимости от текущего состояния класса. Таким образом, гарантируется теоретическая корректность программного кода.

Обеспечение работоспособности отдельных элементов программного кода соблюдается с помощью модульного тестирования (unit-тесты). Предполагается, что весь программный код покрыт совокупностью тестов. В любой момент их прогон показывает работоспособность элементов программного кода. Более того, предлагается создания множества тестов еще до написания кода, что обеспечивает его корректное исполнение поставленных задач. Покрытие тестами обеспечивает возможность модификации программного кода без опасений его разрушения.

Процессы управления моделями и их преобразования рассмотрим на примере известной модели ER – модели сущность-связь.

Рассмотрим учебный пример проектирования базы данных для информационной системы, содержащей сведения об автомобилях, их владельцах, водителях и фирмах производителях, включая концерны, которые контролируют другие фирмы. Такое текстовое описание сохраняется в репозитоарии проекта.

В первую очередь необходимо построить концептуальную модель. Эта может в форме модели сущность-связь показывает, как устроена предметная область решаемой задачи (предметная область – часть окружающей реальности, имеющая отношение к решаемой задаче). Обращаем внимание, пока вообще нет разговора о реализации. Мы хотим только разобраться в том, какие сущности имеются, каковы их характеристики и каковы взаимосвязи между этими сущностями?

Первым этапом является «мозговой штурм» для получения списка кандидатов в сущности. На этом этапе не допускается критика решений, их нужно просто фиксировать.

Напомним, что сущность – это «Объект или явление окружающего мира, который может быть мыслим отдельно».

Очевидно, что изучение пользовательской истории даст возможность выделить такие сущности, как:

- автомобиль
- водитель
- владелец
- фирма
- концерн

и, возможно, другие.

Далее следует оценить выбор и составить модель.

Список сущностей также является артефактом, сохраняемом для дальнейшего использования и ревизии.

Есть и альтернативный метод: изложить на русском языке пользовательскую историю и подчеркнуть существительные одной чертой (это кандидаты в сущности) и глаголы двумя чертами (это возможные связи). Отметим недостаток такого метода. Русский язык гораздо богаче языка модели сущность-связь. Одну и ту же мысль можно изложить по разному, например, «пассажир едет» или «пассажир совершает поездку». Поэтому такой «костыль» плохая замена полноценному анализу.

Итак, мы выделил сущности и изображаем их и их взаимосвязи с помощью графической модели.

Сам метод сущность-связь разработан в конце 70-ых годов прошлого века Ченом, и им же была предложена нотация, где атрибуты обозначались кружочками вокруг сущности.

Такая нотация является громоздкой и ее заменили другие, например, нотация Баркера. Здесь атрибуты помещаются внутрь сущности, а связи задаются значком «воронья лапа».

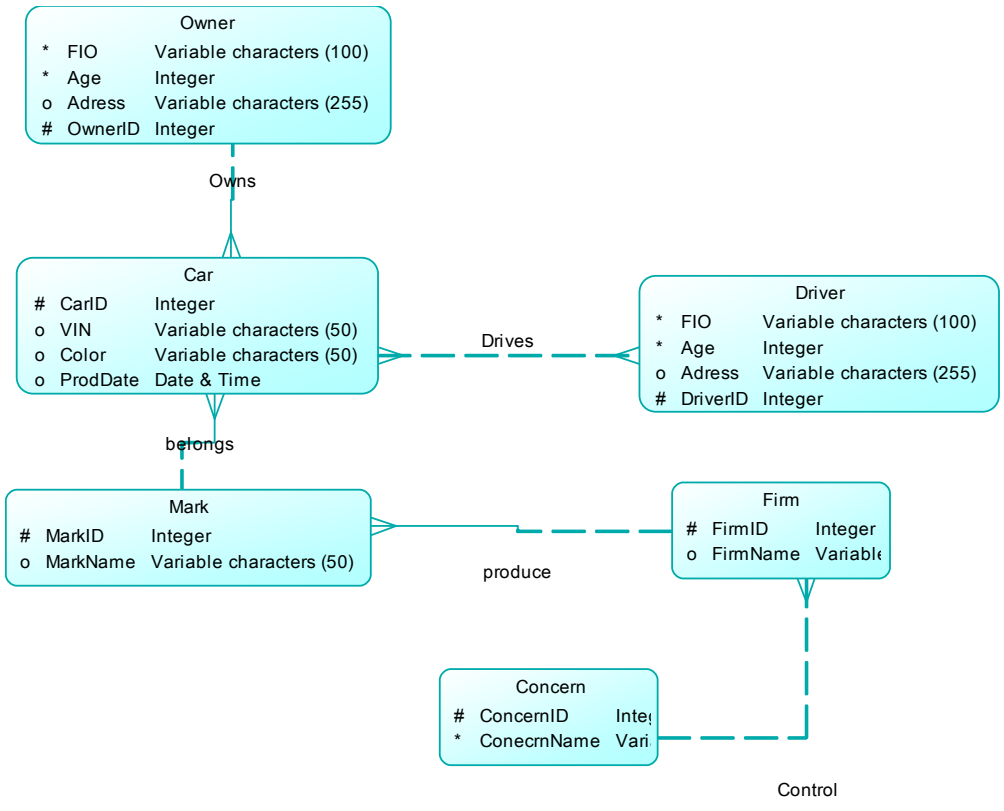
Для формирования связей задаем вопросы по следующей схеме:

- может ли автомобиль иметь несколько фирм производителей? Обязательно ли у него есть фирма –производитель?

- может ли одна фирма-производитель выпускать много автомобилей? Обязательно ли фирма имеет выпущенные автомобили?

То, что модель сущность-связь (а также ее способность организовывать взаимодействие с носителями знаний предметной области за счет понятности и наглядности самой идеи) поощряет задавать вопросы и стало причиной, по которой метод сущность-связь по прежнему активно используется спустя десятилетия после своего рождения. Разработчик не в коем случае не должен сам отвечать на возникающие вопросы. Во-первых, он не владеет предметной областью в должной мере, во-вторых, зачастую построение больших систем, которыми и являются современные предприятия может быть контр интуитивным. За ответами о предметной области нужно идти к эксперту в соответствующей сфере.

На рисунке ниже приведен первый вариант концептуальной модели для нашего примера.



Обратим внимание, что в первом варианте модели мы потеряли сущность Марка, которая не столь очевидна. Действительно, когда мы говорим автомобиль, мы можем подразумевать как конкретный автомобиль с конкретным vin номером, так и все множество таких автомобилей – то есть марка.

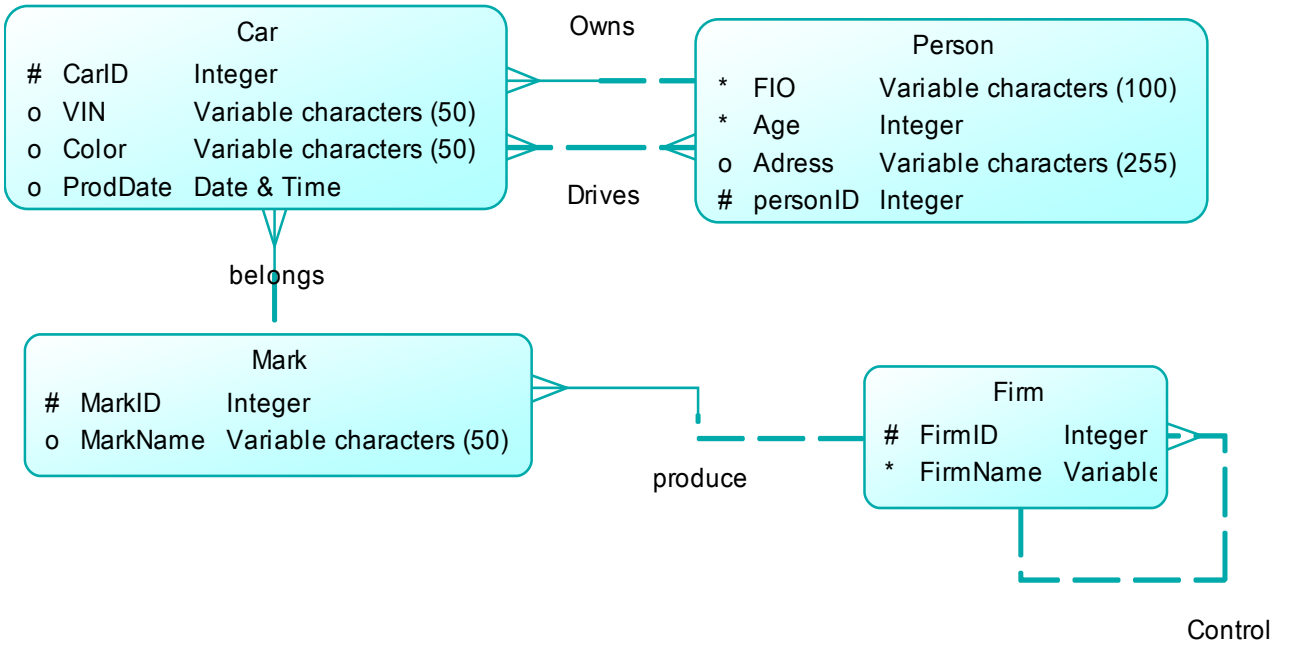
Существующий итеративный поход позволяет обогащать модель и другие артефакты по мере уточнения требований и наших знаний предметной области решаемой задачи.

Далее, на следующий итерации проведем ревизию предложенной концептуальной модели. Очевидно, ее главный недостаток – дублирование. Как пересечение по атрибутам, так и смысловое.

В самом деле – что такое концерн? Прежде всего это фирма.

А понятия водитель и владелец можно объединить понятием человек, персона.

Обновленная модель представлена на следующем рисунке.



Теперь введено понятие Персона, а факт владения автомобилем и фак права управления им отражены с помощью связей между сущностями автомобиль и персона.

Обратим внимание на интересный факт, если мы решили, что у человека может быть много автомобилей, которыми он может управлять, но и у автомобиля может быть много водителей, но владелец у автомобиля только один. И эта связь является обязательной, владелец есть всегда.

Такое решение возможно является правильным, но только в случае, если мы нуждаемся в «мгновенной фотографии». В случае же если нам нужна история значений, то связь была бы многие ко многим, чтобы отразить всех прошлых владельцев автомобиля.

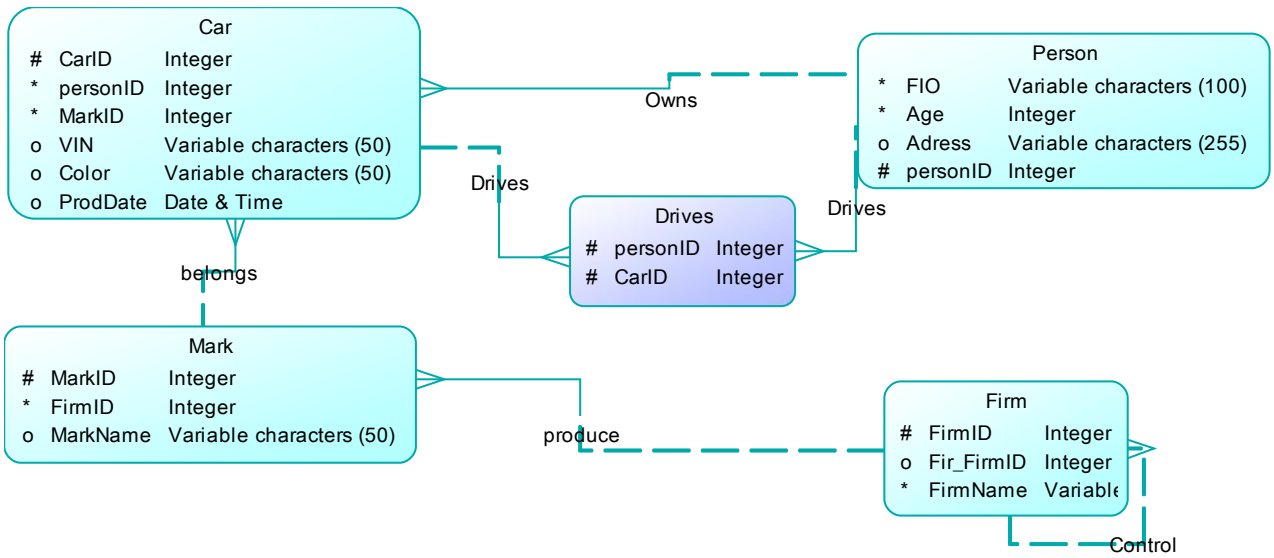
Как решить, достаточно ли детей мы отразили? Хорошим ориентиром является правило Парето 80/20. Нужен компромисс между возможностями системы и ее расширяемостью, с одной стороны и ее простотой наглядностью и временем на моделирование с другой.

Если вовремя не остановиться, можно впасть в аналитический паралич. моделировать не только систему, но и ее окружение, повышать степень детализации, но так и не приступить к разработке.

Для реализации факта владения концерном фирмы (например, концерн Фольксваген контролирует СЕАТ, Шкода, АУДИ, Порш и некоторые другие фирмы) использована связь сущности самой с собой. Така связь получила неофициальное название свиное ухо.

Следующим этапом является построение логической модели. Это тоже артефакт процесса разработки. Здесь речь впервые заходит о реализации. Если концептуальная модель может быть реализована любым способом, например, совокупностью классов Java или структурой реляционной базы данных, то логическая модель описывает систему в терминах реляционных баз данных, но не конкретной СУБД.

На рисунке представлена логическая модель для нашего примера с автомобилями.



Откуда появилась логическая модель? Одним вариантом является ее ручная реализация на основе концептуальной.

Но современные CASE-средства позволяют автоматически строить одни модели на основе других. В данном случае на основе концептуальной модели строится логическая.

Проанализируем, как реализуются связи в модели.

- связи «один к одному» реализуются в виде пары первичный ключ – внешний ключ. Внешний ключ добавляется на той стороне, где «многие». Например, ключ фирмы добавлен в сущность Марка, показывая какая фирма его выпускает.

- связи «многие-ко-многим» реализуются в форме промежуточной таблицы. В нее добавляются первичные ключи обоих связываемых сущностей.

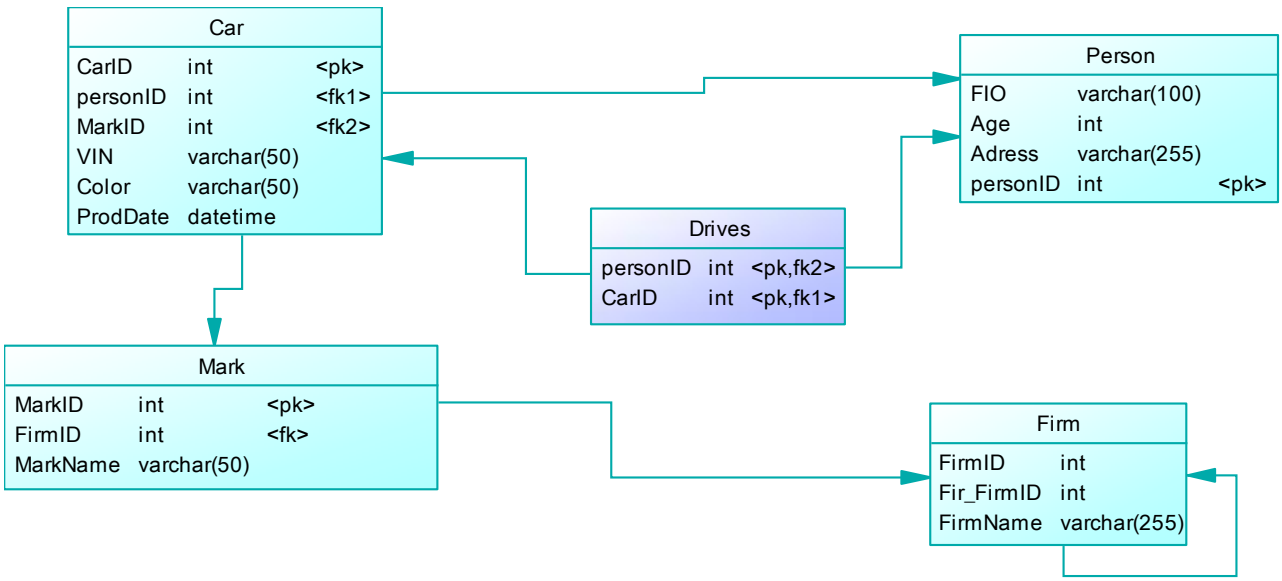
Их комбинация может выступать первичным ключом. Обратим внимание, что реляционные базы данных в принципе не работают со связями многие-ко-многим.

Со временем случается, что введенная промежуточная сущность обрастает атрибутами и занимает свое важное место в модели предметной области.

Иногда говорят о таком понятии (близком к рассматриваемому), как слабая сущность – ее особенность в том, что он не может существовать вне связи с другими сущностями, но как правило имеет свои атрибуты. Например, есть сущности Товар, Клиент, Заказ. А также есть сущность Позиция Заказа. Ее смысл в объединении сущностей заказа и товар, чтобы показать, сколько именно единиц определенного товара заказал клиент. Если экземпляр сущности Заказ уничтожается, то следует уничтожить и все его строки. Именно поэтому говорят о слабой сущности.

После завершения работы над логической моделью на ее основе может быть сформирован следующий артефакт – физическая модель структуры базы данных.

На рисунке ниже приведена физическая модель рассматриваемого примера.



Главной особенностью является привязка к конкретной СУБД, например, MS SQL Server.

Данная модель была получена автоматически из логической модели в CASE-средстве.

В ходе работы произошли следующие преобразования:

- сущности были заменены таблицами
- атрибуты полями
- ключи реализованы соответствующими ограничениями
- появилась возможность контроля ссылочной целостности для связей между таблицами

Теперь появилась возможность дополнить модель специфическими для конкретной СУБД значениями. Например: представления, индексы, хранимые процедуры, триггеры и т.д. и т.п.

После модификации модели имеется возможность формирования нового артефакта проекта: собственно, структуры базы данных в виде SQL-скрипта. Фрагмент скрипта приведен ниже.

```
/*==============================================================*/
/* Table: Car                                                   */
/*==============================================================*/
create table Car (
   CarID                int                  identity,
   personID             int                  not null,
   MarkID               int                  not null,
   VIN                  varchar(50)          null,
   Color                varchar(50)          null,
   ProdDate             datetime             null,
   constraint PK_CAR primary key nonclustered (CarID)
)

/*==============================================================*/
/* Table: Drives                                                */
/*==============================================================*/
create table Drives (
   personID             int                  not null,
   CarID                int                  not null,
   constraint PK_DRIVES primary key nonclustered (personID, CarID)
)
go

/*==============================================================*/
/* Table: Firm                                                  */
/*==============================================================*/
create table Firm (
   FirmID               int                  identity,
   Fir_FirmID           int                  null,
   FirmName             varchar(255)         not null,
   constraint PK_FIRM primary key nonclustered (FirmID)
)
go


alter table Car
   add constraint FK_CAR_OWNS_PERSON foreign key (personID)
      references Person (personID)
go

alter table Car
   add constraint FK_CAR_BELONGS_MARK foreign key (MarkID)
      references Mark (MarkID)
go

alter table Drives
   add constraint FK_DRIVES_DRIVES_PERSON foreign key (personID)
      references Person (personID)
go

alter table Drives
   add constraint FK_DRIVES_DRIVES2_CAR foreign key (CarID)
      references Car (CarID)
go
```

Этот скрипт сохраняется в репозитарии проекта интеллектуальной информационной системе и на определенной итерации служит для развертывания базы данных.

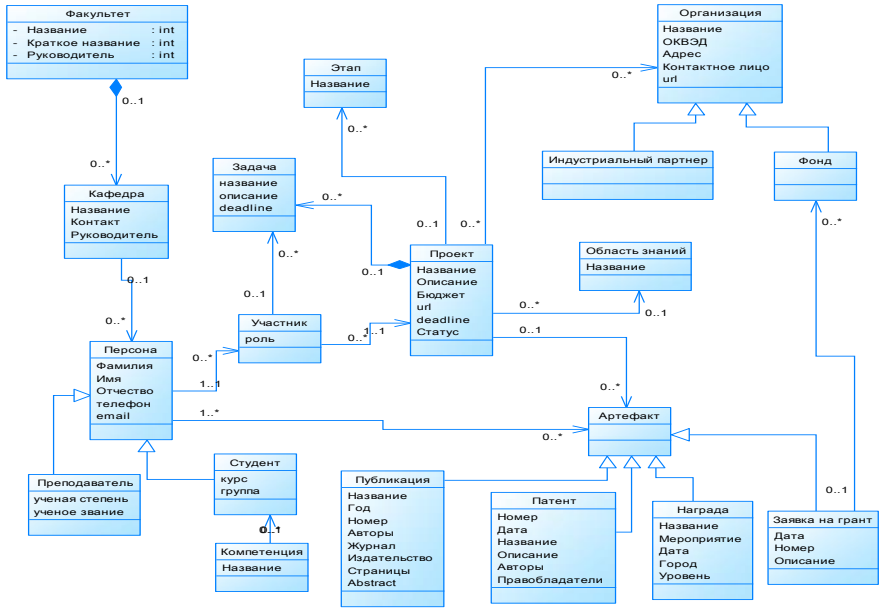
Бывают ситуации с возвратом к ранее созданным артефактам, например, если реализация модели привела к какому-то расхождению с задачей. В этом случае генерация может проводиться заново. Также возможна ситуация обновления моделей верхнего уровня на основе моделей нижнего уровня, например, на основе физической модели можно воссоздать обновленную логическую модель.

На такой техничке основано и понятие риверс-инжиниррринга, когда имеется некоторая база данных и на ее основе строятся модели, которые уже затем подвергаются модификации с целью повышения качества информационной системы.

Использование в нашем примере CASE-средств выглядит впечатляющим и позволяет экономить время разработчика. Однако CASE-средства не являются панацеей и скорее играют вспомогательную роль. Все дело в том, что совокупность интеллектуальных усилий для того, чтобы разобраться в предметной области, реализовать это знание в терминах моделей, определить эффективные проектные решения, все эти усилия значительно превосходят затраты труда, например, на разработку программного кода.

Все это может быть выражено словами Ф. Брукса «нет серебряной пули», что обозначает, что разрабатываемые информационные системы обладают имманентной сложностью, т.е. сложность присуща им как таковым и она не является следствием недостатка синтаксических средств или нехватки автоматизации для формирования кода.

Развитием идей моделей сущность-связь является диаграмма классов языка UML.



Такая модель позволяет:

- эффективно описывать наследование
- показывать разницу между разными видами связей: ассоциацией, агрегацией
- показывать видимость атрибутов
- отражать методы, т.е. возможное поведение объектов

Вопросы по теме лекции:

- что такое модель?
- что такое метаиинформация?
- в чем разница между анализом и проектированием?
- что такое артефакт?
- что содержит репозитарий проекта?
- в чем особенность спиральных процессов проектирования информационных систем?
- что такое Agile?
- для чего нужна модель ER?
- в чем разница сущности и экземпляра?
- приведите пример сущностей?
- какие бывают связи?
- может ли сущность иметь связь с собой?
- чем плох метод построения модели путем изучения грамматики «пользовательской истории»?
- может ли из концептуальной модели быть порождены разные варианты реализации, например, в реляционной СУБД или NoSQL?
- каким образом реализуется связь между сущностями типа «один-ко многим» при реализации концептуальной модели в виде реляционной базы данных?
- каким образом реализуется связь между сущностями типа «многие-ко многим» при реализации концептуальной модели в виде реляционной базы данных?
- что является развитием модели «сущность-связь» в языке UML (диаграмма)?
- почему метаданные информационной системы имеют более длительный жизненный период по сравнению с данными?

Литература и материалы по теме:

Гради Буч и др. Объектно-ориентированный анализ и проектирование с примерами приложений (UML 2). Третье издание. — М.: «Вильямс», 2008. — 720 с.

Арлоу Д., Нейштадт И. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2е издание. – Пер. с англ. – СПб: Символ Плюс, 2007. – 624 с.

Фаулер, Мартин UML. Основы. Краткое руководство по стандартному языку объектного моделирования / Мартин Фаулер. - Москва: СИНТЕГ, 2011. - 192 с.

Перлова, О.Н. Проектирование и разработка информационных систем: Учебник / О.Н. Перлова, О.П. Ляпина, А.В. Гусева. - М.: Academia, 2017. - 416 с.

Мейер Бертран Объектно-ориентированное конструирование программных систем. – М.: Издательско-торговый дом «Русская Редакция», «Интернет-университет информационных технологий», 2005. –1232 с.

.

1.1.1.9 Методы и средства прототипного проектирования ИС: репозиторий проекта и паттерны проектирования.

Современные методы проектирования интеллектуальных информационных систем основаны на использовании опыта, накопленного в индустрии.

Ключевой является идея паттерна или шаблона или образца проектирования.

Первоначальна идея была позаимствована из области архитектуры, строительства и дизайна. Известный архитектор и теоретик архитектуры К. Александер сформировал каталог типовых распространенных решений начиная от градостроительства, заканчивая дизайном помещений. Разработанный им язык Паттернов позволил описать ряд удачных типовых решений распространенных проблем.

В дальнейшем идея была перенесена в область разработки объектно-ориентированного программного обеспечения. Известный каталог паттернов был составлен Бандой Четырех в середине девяностых годов прошлого века и применяется до сих пор.

Паттерн следует понимать как некоторое типовое решение вписанное в контекст некоторой проблемы. Необходимо отличать паттерн просто от типового решения конкретной задачи, паттерн – это скорее более абстрактная идея решения, позволяющая добиться определенных преимуществ. Конкретная реализация может отличаться.

Банда Четырёх выделили три вида паттернов:

- порождающие;
- структурные;
- поведенческие.

Порождающие паттерны решают важную задачу повышения независимости различных компонентов программного обеспечения. В самом деле, огромным преимуществом объектно-ориентированного подхода являются наследование и полиморфизм. Можно создать, например, абстрактный класс Транспортное средство, с абстрактными методами для управления, которые реализовывать многочисленными наследниками (автобус, мотоцикл, гужевая повозка), при этом каждый класс-наследник будет выполнять команды управления, например, старт и стоп по своему. Остальным частям системы не нужно даже знать о существовании этих классов, они обращаются с их объектами как к транспортными средствами. Слабым местом, однако, является создание объектов. Здесь необходимо явно вызвать конструктор дочернего класса, что заставляет знать о нем и снижает гибкость и расширяемость системы. Набор порождающих паттернов позволяет вынести порождения объекта или взаимосвязанного набора объектов в отдельную часть системы, что повышает гибкость. Примеры порождающих паттернов: строитель, фабричный метод, абстрактная фабрика и др.

Структурные паттерны предназначены для гибкого формирования совокупности взаимосвязанных объектов, обеспечивающих решение поставленной задачи. Примеры структурных паттернов: мост, адаптер, компоновщик, фасад.

Поведенческие паттерны обеспечивают взаимодействие объектов различных классов для решения задач с тем, чтобы повысить гибкость и снизить зависимость между классами. Примеры поведенческих паттернов: наблюдатель, состояние, стратегия, команда и др.

Некоторый паттерны проектирования будут рассмотрены далее в ходе текущей лекции.

Каждый паттерн описывается по следующей схеме (теперь подобная схема является общепринятой):

- название
- альтернативные названия
- описание, суть, цель
- решаемая проблема
- структура (обычно в виде UML-модели)
- примеры реализации с кодом
- рекомендации по реализации
- связь с другими паттернами.

Далее возник термин Антипаттерн, обозначающей распространенное неудачное решение известной проблемы. У каждого антипаттерна имеются недостатки, делающие полученный код хрупким, снижающие гибкость системы.

Тем не менее, следует отличать антипаттерн от ошибки. Антипаттерн позволит создать работающую систему, однако при дальнейшей ее эксплуатации и развитии скажутся недостатки решения, например, создание дополнительного функцинала потребует коренной перестройки всей структуры классов.

Обратим внимание, что как правило, имеется веская причина применения неудачного решения, а также такое решение часто возникает в качестве хот-фикса, временного варианта.

Примеры антипаттернов из области разработки объектно-ориентированного программного обеспечения: спагетти код, божественный класс, ленивый класс и другие.

Со временем, идеи паттерна и антипаттерна вышли за пределы объектно-ориентированного проектирования информационных систем. Появились наборы паттернов и антипаттернов в таких областях сферы информационных технологий, как:

- управления процессом разработки;
- управления командами;
- проектном менеджменте;
- машинном обучении;
- проектирования пользовательского интерфейса;
- организации сетевого взаимодействия;

- проектирования баз данных.

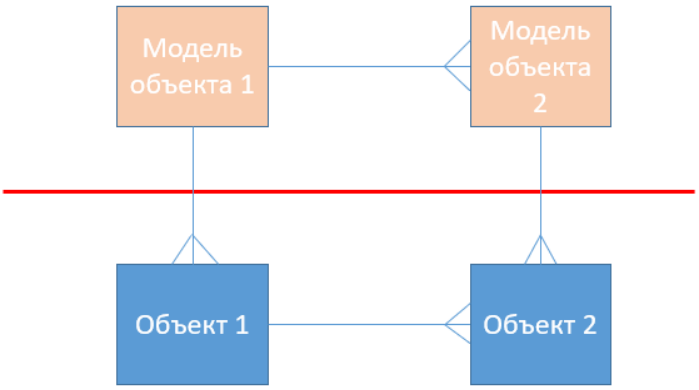
И др.

Рассмотрим примеры паттернов и антипаттернов из области реляционных баз данных. Их источником является работа М. Блаха по моделированию данных и паттернам баз данных.

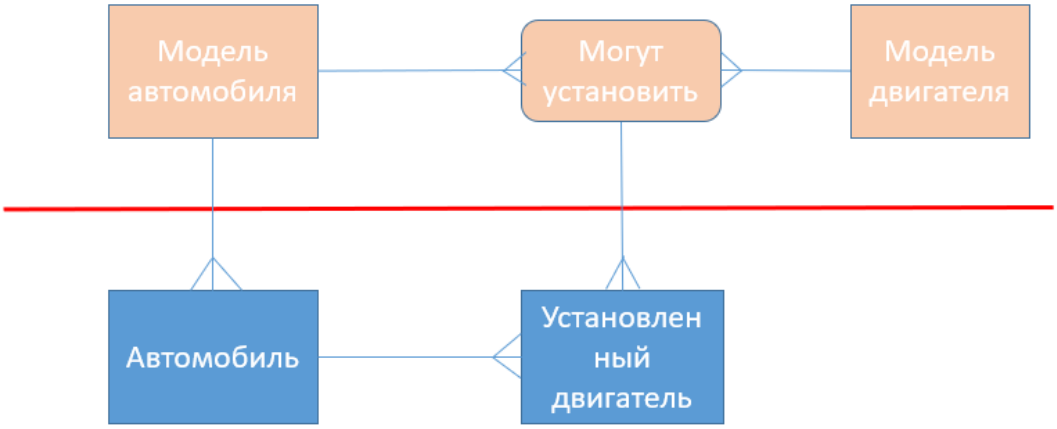
Пример паттерна «Модель-реализация».

Цель: Гомоморфное отражение потенциальных и реальных связей в базе данных.

Общая схема паттерна приведена на рисунке.



Решаемая проблема заключается в необходимости отражения как возможной связи между сущностями: двигатель определенной марки может быть установлен на определенную модель автомобиля, так и тот факт, что конкретный двигатель установлен в конкретный автомобиль. Как изображено на рисунке.

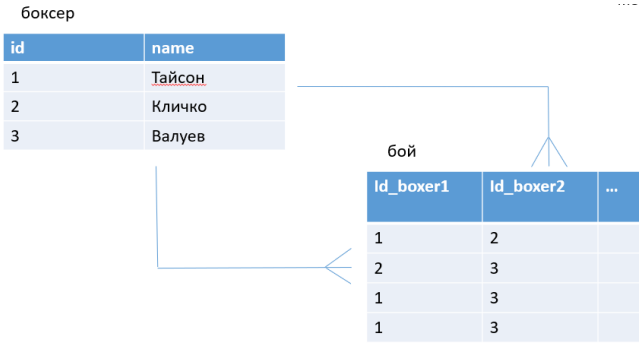


Обратим внимание, что связь автомобиля проведена через промежуточную сущность непосредственно к сущности, отражающей возможность установки – здесь фиксируется и конкретная модель двигателя в том числе.

Еще одним примером можно считать связи графа.

Цель: Гибкий и симметричный учет связей между объектами

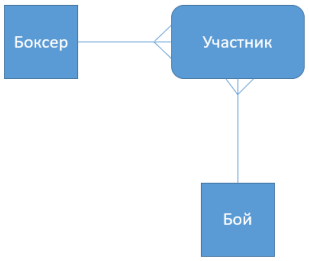
Предположим, нам необходимо фиксировать информацию о боксерских поединках. Очевидное решение приведено на рисунке ниже, оно, однако, характеризуется рядом недостатков.



Почему в каждой строке один боксер является первым, а другой вторым? Как нам найти все бои определенного боксера (понадобится сложный запрос)? Как нам обеспечить соблюдение определенных бизнес-правил?

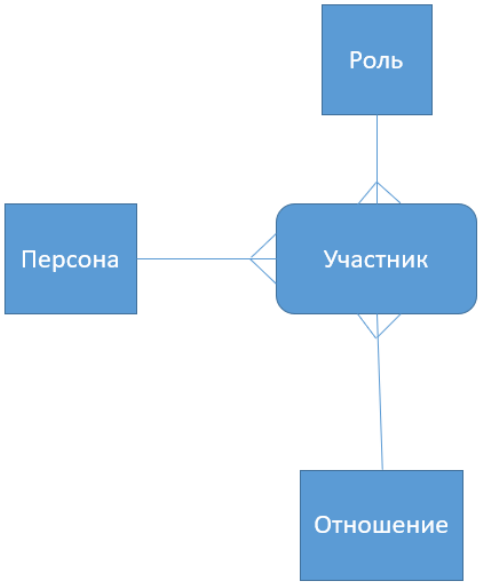
Таким образом, такая схема не очень удачна, особенно имея в виду дальнейшее расширение системы.

Более удачное решение показано на рисунке ниже:



Теперь имеется возможность легкого выбора участников одним SQL-запросом, при этом можно также легко вводить дополнительные жлементы в модель.

Ниже показана обобщенная схема.



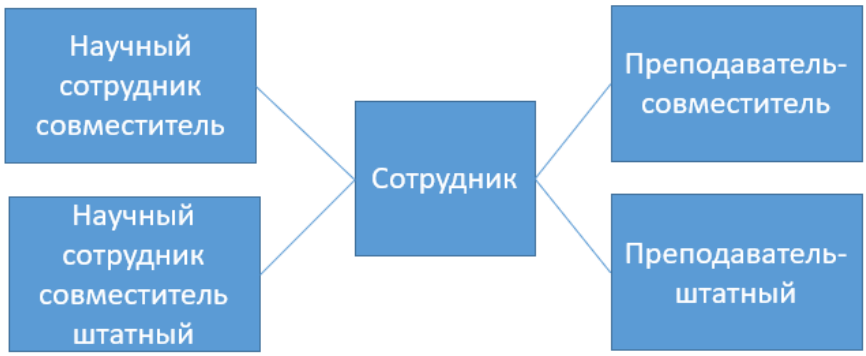
Такое решение позволит легко модифицировать модель, добавляя новые роли для участников отношений.

Продолжая тему проектирования баз данных рассмотрим теперь пример антипаттерна.

В качестве хорошего источника информации можно рекомендовать книгу Билла Карвина SQL Antipatterns.

Рассмотрим следующий пример.

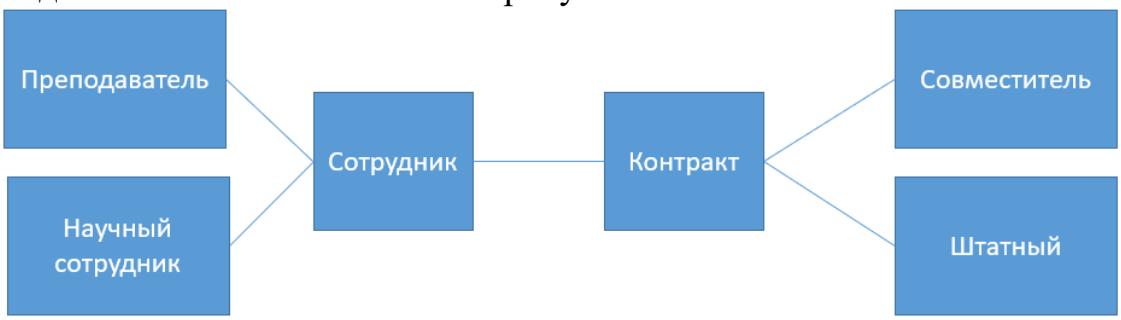
Мозаичное наследование.



Предположим, что сущности научный сотрудник/преподаватель и совместитель/штатный настолько отличаются друг от друга, что это отличие оправдывает существования их в виде отдельных сущностей (иначе разгницу можно было бы покрыть атрибутами)

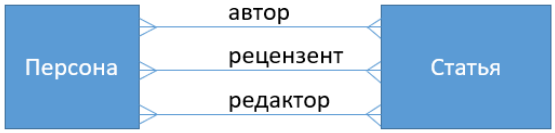
Тем не менее, нет никакого смысла строить столь сложную модель. Предположим, понадобится запрос на выборку всех преподавателей. В существующей схеме понадобится обращать к двум отельным таблицам. А также нет возможности обеспечивать целостность данных. А что если понадобится привлечь еще один вариант контракта, например, хозяйственно правовой договор?

Очевидно, что следует провести абстрактную границу между способом найма сотрудника и его обязанностями. Эти вещи не зависят друг от друга. Более адекватная схема показана на рисунке.



Несмотря на добавление лишней сущности обе части стали независмы друг от друга. Теперь стали легче запросы, обеспечение логической целостности, реализация бизнес-правил. Стало возможно менее болезненное расширение и модификация.

Еще один анти паттерн – параллельные связи. Пример приведен на рисунке.



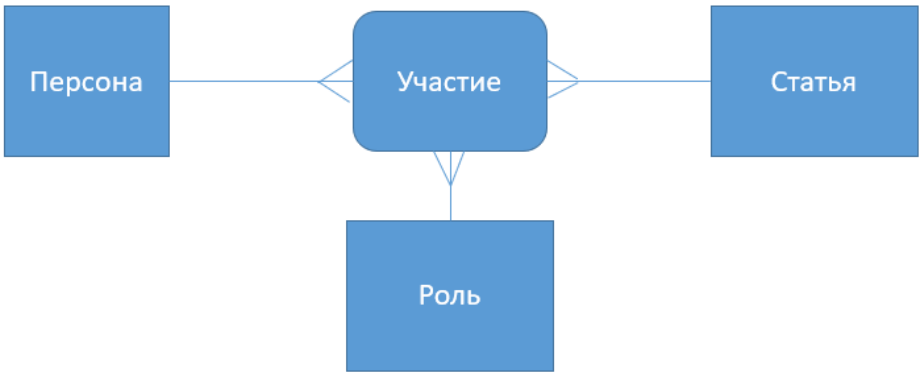
Все это потребует многочисленных промежуточных таблиц (обратим внимание, связи многие-ко-многим).

Недостатки:

-Сложные запросы

- Снижение гибкости

Как бороться с этим антипаттерном? Ниже представлен вариант решения «Участник-Роль-Отношение».



Здесь стоит обратит внимание, что мы пришли к паттерну, который рассматривался выше. Это не случайно, ведь паттерны и создавались как ответ на решения подобных распространенных проблем. И переход к паттерну и будет часто преодолением некоторого антипаттерна.

Рассмотрим еще один, крайне простой антипаттерн.

Столбцы-дубли.

person	language	language2
Иванов И.И.	испанский	Англ
Петров П.П.	Английский	

Это «детский антипаттерн» несмотря на свою очевидную несостоятельность регулярно возникает в некоторых системах. Причина заключается в получении хот-фикса, быстрого ответа на возникшую проблему. Добавление столбца позволяет, скажем, получить нужный немедленно отчет, но при отсутствии дисциплины повлечет к негативным последствиям в дальнейшем.

Недостатки очевидны:

- Сложные запросы (OR .. OR..)
- Невозможность
 контроля
- Низкая гибкость

Решение – нормализация.

Рассмотрим еще один антипаттерн. «Бессодержательные имена и отсутствие системы именования таблиц и атрибутов».

Недостатки:

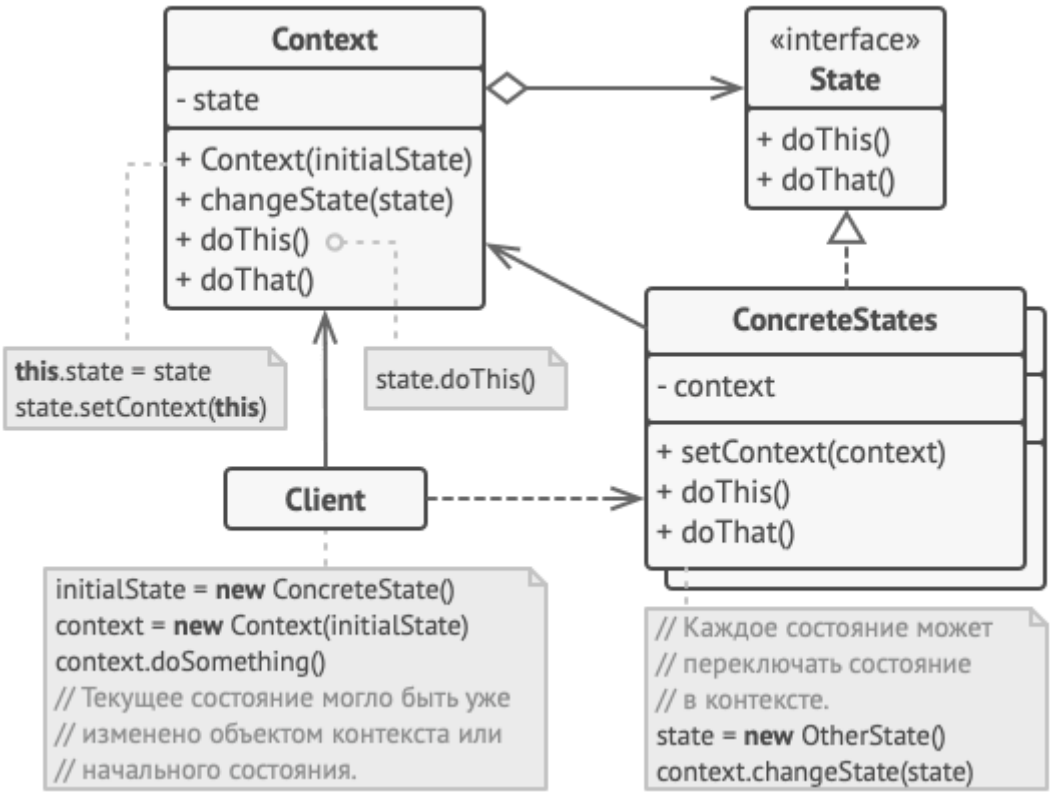
- Потеря времени
- Сложность сопровождения

Проблема создания проектного решения заключается не только в обеспечении технической стороны, но и в сохранении возможности поддержки и расширения. Понятность, наглядность кода не менее важны, чем его вычислительная эффективность или другие качества. В современных условиях, когда труд программистов стоит невероятно дорого, наглядный, понятный код позволит экономить время и убережет от ошибок.

Пример одного из известных паттернов проектирования – паттерна Состояние рассмотрим вкратце.

Состояние — это поведенческий паттерн проектирования, который позволяет объектам менять поведение в зависимости от своего состояния. Извне создаётся впечатление, что изменился класс объекта.

Схема паттерна на языке uml приведена на рисунке ниже.



Класс контекста не реализует теперь все необходимые реакции, делегировав их классу состояние. Это абстрактный класс, он описывает свои возможности, но их реализация осуществлена в конкретных состояниях – его наследниках.

Таким образом, если объект на экране захвачен мышкой и перемещается, то его реакций на щелчок левой кнопкой будет иная, чем если бы он находился в покое.

Далее рассмотрим концепцию рефакторинга программного обеспечения и ее связь с паттернами проектирования.

Идея рефакторинга была предложена М Фаулером и он же составил каталог рефакторингов, т.е. изменений кода небольшого размера, обеспечивающих повышение его гибкости и читаемости.

Рефакторинг – это изменение программного кода информационной системы для повышения качества без изменения ее функционала.

Рефакторинг является неотъемлемой частью современных схем проектирования и разработки программного обеспечения, в том числе методологии Agile.

Схема проведения рефакторинга приведена на рисунке.

Фаулер видит необходимым условием рефакторинга полное покрытие кода модульными тестами.

Итак, первым шагом является обнаружение признака некачественного кода. Примеры таких признаков:

- дублирование кода
- магические числа
- ленивый класс
- длинный метод
- класс, больше заинтересованный в членах другого класса, чем в своих методах и свойствах
- использование временных переменных
- операторы switch

И др

После обнаружения такого признака осуществляется применение шага рефакторинга.

Например:

- извлечь метод\извлечь класс
- поднять метод
- опустить метод
- внедрить шаблонный метод
- заменить переключатель полиморфизмом

и т.д.

Далее необходимо обеспечить выполнение всех тестов, после чего процесс можно повторить.

Рассмотрим пример применения рефакторинга заменить временную переменную вызовом метода (replace temp with query). Пример взят из работы Фаулера.

```
const basePrice = this._quantity * this._itemPrice;
if (basePrice > 1000)
  return basePrice * 0.95;
else
  return basePrice * 0.98;
```

сначала выполняется Extract Method (выделение метода)

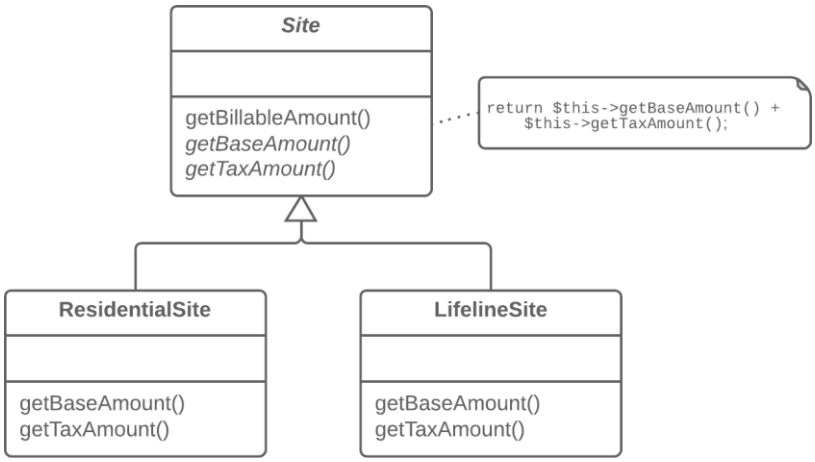
```
float basePrice() {
        this._quantity * this._itemPrice;
}
```

А далее модифицируется исходный метод.

```
if (this.basePrice() > 1000)
  return this.basePrice() * 0.95;
else
  return this.basePrice() * 0.98;
```

Обратим внимание, что код потерял в эффективности, один и тот же расчет теперь выполняется дважды, но выиграл в наглядности и гибкости.

Еще один пример рефакторинга – переход к шаблонному методу (вариант применения показан на рисунке).



Метод поднят в родительский класс и реализован за счет вынесения двух абстрактных методов. Они вызываются в шаблонном методе. Реализация двух шагов, двух абстрактных методов поручается дочерним классам.

Теперь код избавлен от дублирования, стал более наглядным.

Как видим, применение рефакторинга зачастую также выводит разработчика на один из паттернов проектирования.

Вопросы по теме лекции:

- что такое паттерн проектирования?
- из какой области человеческой деятельности пришло понятие паттерна проектирования?
- используется ли понятие паттерна за пределами объектно-ориентированного подхода к проектированию интеллектуальных информационных систем?
- какие группы паттернов как правило выделяют?
- в чем разница между паттернами Состояние и Стратегия?
- какой паттерн позволяет хранить историю команд и осуществлять при необходимости их отмену?
- какой паттерн позволяет порождать альтернативные группы взаимосвязанных объектов?
- какой паттерн обеспечивает работу с группой объектов так же как и с одним объектом?
- возможна ли альтернативная реализация одного и того же паттерна проектирования?
- в чем преимущество использования паттерна Декоратор?
- в чем смысл применения паттернов проектирования?
- чем паттерн проектирования отличается от готового типового решения задачи?
- к какому классу относится паттерн Шаблонный метод?
- что такое антипаттерн?
- в чем недостаток использования антипаттерна Спагетти код?
- какова роль языка UML при использовании паттернов проектирования?
- в чем заключается идея рефакторинга кода программного обеспечения информационных систем?
- какова связь паттернов с рефакторингом?
- существуют ли паттерны и антипаттерны в задачах проектирования структуры базы данных при построении интеллектуальных информационных систем?

Литература и материалы по теме:

Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Паттерны объектно-ориентированного проектирования / Пер. с англ.: А. Слинкин. — СПб.: Питер, 2021. — 448 с.

Фаулер М., Бек К., Брант Д., Робертс Д., Апдайк У. Рефакторинг: улучшение существующего кода.— СПб.: Символ-Плюс, 2009. — 432 с.

Гради Буч и др. Объектно-ориентированный анализ и проектирование с примерами приложений (UML 2). Третье издание. — М.: «Вильямс», 2008. — 720 с.