Образовательная программа
«Машинное обучение и технологии больших данных»

Технологии проектирования интеллектуальных систем

Тема 1.

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

29.07.2022

- Современные интеллектуальные информационные системы применяются во всех сферах и бизнес процессах человеческой деятельности.
- Следовательно, и исследования применения интеллектуальных информационных систем можно рассматривать с различных точек зрения.
- Применительно к каждой сфере и бизнес процессу используют различные подходы и различные типы систем.

Системы, основанные на знаниях

- Основной целью построения таких систем являются выявление, исследование и применение знаний высококвалифицированных экспертов для решения сложных задач, возникающих на практике.
- При построении систем, основанных на знаниях, используются знания, накопленные экспертами в виде конкретных правил решения тех или иных задач.
- Это направление преследует цель имитации человеческого искусства анализа неструктурированных и слабоструктурированных проблем.
- В данной области исследований осуществляется разработка моделей представления, извлечения и структурирования знаний, а также изучаются проблемы создания баз знаний, образующих ядро интеллектуальных систем, основанных на знаниях.
- Одним из видов таких систем можно выделить экспертные системы.

Системы машинного перевода

- Варианты построения интеллектуальных систем являются системы машинного перевода с одного естественного языка на другой обеспечивают быстроту и систематичность доступа к информации, оперативность и единообразие перевода больших потоков, как правило, научно-технических текстов.
- В их основе лежат знания в определенной предметной области и сложные модели, обеспечивающие дополнительную трансляцию "исходный язык оригинала.
- Они базируются на структурно-логическом подходе, включающем последовательный анализ и синтез естественно- языковых сообщений.
- Кроме того, в них осуществляется ассоциативный поиск аналогичных фрагментов текста и их переводов в специальных базах данных.
- Данное направление охватывает также исследования методов и разработку систем, обеспечивающих реализацию процесса общения человека с компьютером на естественном языке (так называемые системы ЕЯ-общения).

Генерация и распознавание речи

- Системы речевого общения создаются в целях повышения скорости ввода информации в ЭВМ, разгрузки зрения и рук, а также для реализации речевого общения на значительном расстоянии.
- В таких системах под текстом понимают фонемный текст (как слышится).

Обработка визуальной информации

- В этом научном направлении решаются задачи обработки, анализа и синтеза изображений.
- Задача обработки изображений связана с трансформированием графических образов, результатом которого являются новые изображения.
- В задаче анализа исходные изображения преобразуются в данные другого типа, например в текстовые описания.
- При синтезе изображений на вход системы поступает алгоритм построения изображения, а выходными данными являются графические объекты (системы машинной графики).

Обучение и самообучение

- Эта актуальная область включает модели, методы и алгоритмы, ориентированные на автоматическое накопление и формирование знаний с использованием процедур анализа и обобщения данных.
- К данному направлению относятся не так давно появившиеся системы добычи данных (Data-mining) и системы поиска закономерностей в компьютерных базах данных (Knowledge Discovery).

Распознавание образов

Это одно из самых ранних направлений построения интеллектуальных систем, в котором распознавание объектов осуществляется на основании применения специального математического аппарата, обеспечивающего отнесение объектов к классам, а классы описываются совокупностями определенных значений признаков.

Игры и машинное творчество

- Машинное творчество охватывает сочинение компьютерной музыки, стихов, интеллектуальные системы для изобретения новых объектов.
- Создание интеллектуальных компьютерных игр является одним из самых развитых коммерческих направлений в сфере разработки программного обеспечения.
- Кроме того, компьютерные игры предоставляют мощный арсенал разнообразных средств, используемых для обучения.

Программное обеспечение интеллектуальных систем:

- специальные языки программирования, ориентированные на обработку символьной информации (LISP, SMALLTALK, РЕФАЛ),
- языки логического программирования (PROLOG),
- языки представления знаний (OPS 5, KRL, FRL),
- интегрированные программные среды, содержащие арсенал инструментальных средств для создания интеллектуальных систем (KE, ARTS, GURU, G2),
- оболочки экспертных систем (BUILD, EMYCIN, EXSYS Professional, ЭКСПЕРТ), которые позволяют создавать прикладные ЭС, не прибегая к программированию.

Новые архитектуры компьютеров

- Это направление связано с созданием компьютеров не фон-неймановской архитектуры, ориентированных на обработку символьной информации.
- Известны удачные промышленные решения параллельных и векторных компьютеров, однако в настоящее время они имеют весьма высокую стоимость, а также недостаточную совместимость с существующими вычислительными средствами.

Интеллектуальные роботы

- Создание интеллектуальных роботов составляет конечную цель робототехники.
- В настоящее время в основном используются программируемые манипуляторы с жесткой схемой управления, названные роботами первого поколения.
- Несмотря на очевидные успехи отдельных разработок, эра интеллектуальных автономных роботов пока не наступила.
- Основными сдерживающими факторами в разработке автономных роботов являются нерешенные проблемы в области интерпретации знаний, машинного зрения, адекватного хранения и обработки трехмерной визуальной информации.

Признаки интеллектуальных систем:

- развитые коммуникативные способности;
- умение решать сложные плохо формализуемые задачи;
- способность к самообучению;
- адаптивность.

Интеллектуальные функции интеллектуальных систем:

- коммуникативные способности – способ взаимодействия конечного пользователя с системой;
- решение сложных плохо формализуемых задач, которые требуют построения оригинального алгоритма решения в зависимости от конкретной ситуации, характеризующейся неопределенностью и динамичностью исходных данных и знаний;
- способность к самообучению – умение системы автоматически извлекать знания из накопленного опыта и применять их для решения задач;
- адаптивность – способность системы к развитию в соответствии с объективными изменениями области знаний.

Системы с интеллектуальным интерфейсом:

- *Интеллектуальные базы данных*. Позволяют в отличие от традиционных БД обеспечивать выборку необходимой информации, не присутствующей в явном виде, а выводимой из совокупности хранимых данных.

- *Естественно-языковой интерфейс*. Применяется для доступа к интеллектуальным базам данных, контекстного поиска документальной текстовой информации, голосового ввода команд в системах управления, машинного перевода с иностранных языков. Для реализации ЕЯ-интерфейса необходимо решить проблемы морфологического, синтаксического и семантического анализа, а также задачу синтеза высказываний на естественном языке. При морфологическом анализе осуществляются распознавание и проверка правильности написания слов в словаре. Синтаксический контроль предполагает разложение входных сообщений на отдельные компоненты, проверку соответствия грамматическим правилам внутреннего представления знаний и выявление недостающих частей. Семантический анализ обеспечивает установление смысловой правильности синтаксических конструкций. В отличие от анализа синтез высказываний заключается в преобразовании цифрового представления информации в представление на естественном языке.

- *Гипертекстовые системы.* Используются для реализации поиска по ключевым словам в базах данных с текстовой информацией. Для более полного отражения различных смысловых отношений терминов требуется сложная семантическая организация ключевых слов. Решение этих задач осуществляется с помощью интеллектуальных гипертекстовых систем, в которых механизм поиска сначала работает с базой знаний ключевых слов, а затем – с самим текстом. Аналогичным образом проводится поиск мультимедийной информации, включающей кроме текста графическую информацию, аудио- и видеообразы.

- *Системы контекстной помощи.* Относятся к классу систем распространения знаний. Такие системы являются, как правило, приложениями к документации. Системы контекстной помощи – частный случай гипертекстовых и ЕЯ-систем. В них пользователь описывает проблему, а система на основе дополнительного диалога конкретизирует ее и выполняет поиск относящихся к ситуации рекомендаций. В обычных гипертекстовых системах, наоборот, компьютерные приложения навязывают пользователю схему поиска требуемой информации.

- *Системы когнитивной графики*. Ориентированы на общение с пользователем ИИС посредством графических образов, которые генерируются в соответствии с изменениями параметров моделируемых или наблюдаемых процессов. Когнитивная графика позволяет в наглядном и выразительном виде представить множество параметров, характеризующих изучаемое явление, освобождает пользователя от анализа тривиальных ситуаций, способствует быстрому освоению программных средств и повышению конкурентоспособности разрабатываемых ИИС. Применение когнитивной графики особенно актуально в системах мониторинга и оперативного управления, в обучающих и тренажерных системах, в оперативных системах принятия решений, работающих в режиме реального времени.

Экспертные системы

- Экспертные системы как самостоятельное направление в искусственном интеллекте сформировалось в конце 1970-х годов.
- История ЭС началась с сообщения японского комитета по разработке ЭВМ пятого поколения, в котором основное внимание уделялось развитию "интеллектуальных способностей" компьютеров с тем, чтобы они могли оперировать не только данными, но и знаниями, как это делают специалисты (эксперты) при выработке умозаключений.
- Область исследования ЭС называют "инженерией знаний".
- Этот термин был введен Е. Фейгенбаумом и в его трактовке означает "привнесение принципов и инструментария из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов".

Характеристики задач, решаемых с использованием ЭС:

- задачи не могут быть представлены в числовой форме;
- исходные данные и знания о предметной области обладают неоднозначностью, неточностью, противоречивостью;
- цели нельзя выразить с помощью четко определенной целевой функции;
- не существует однозначного алгоритмического решения задачи;
- алгоритмическое решение существует, но его нельзя использовать по причине большой размерности пространства решений и ограничений на ресурсы (времени, памяти).

Самообучающиеся системы

- Самообучающиеся интеллектуальные системы основаны на методах автоматической классификации ситуаций из реальной практики, или на методах обучения на примерах.
- Примеры реальных ситуаций составляют так называемую обучающую выборку, которая формируется в течение определенного исторического периода. Элементы обучающей выборки описываются множеством классификационных признаков.
- Стратегия "обучения с учителем" предполагает задание специалистом для каждого примера значений признаков, показывающих его принадлежность к определенному классу ситуаций.
- При обучении "без учителя" система должна самостоятельно выделять классы ситуаций по степени близости значений классификационных признаков.
- В процессе обучения проводится автоматическое построение обобщающих правил или функций, описывающих принадлежность ситуаций к классам, которыми система впоследствии будет пользоваться при интерпретации незнакомых ситуаций.
- Из обобщающих правил, в свою очередь, автоматически формируется база знаний, которая периодически корректируется по мере накопления информации об анализируемых ситуациях.

Нейронные сети

- Представляют собой классический пример технологии, основанной на примерах.
- Нейронные сети – обобщенное название группы математических алгоритмов, обладающих способностью обучаться на примерах, "узнавая" впоследствии черты встреченных образцов и ситуаций.
- Благодаря этой способности нейронные сети используются при решении задач обработки сигналов и изображений, распознавания образов, а также для прогнозирования.
- Нейронная сеть – это кибернетическая модель нервной системы, которая представляет собой совокупность большого числа сравнительно простых элементов – нейронов, топология соединения которых зависит от типа сети.
- Чтобы создать нейронную сеть для решения какой-либо конкретной задачи, следует выбрать способ соединения нейронов друг с другом и подобрать значения параметров межнейронных соединений.

Информационные хранилища

- Информационные хранилища отличаются от интеллектуальных баз данных, тем, что представляют собой хранилища значимой информации, регулярно извлекаемой из оперативных баз данных.
- Хранилище данных – это предметно-ориентированное, интегрированное, привязанное ко времени, неизменяемое собрание данных, применяемых для поддержки процессов принятия управленческих решений.
- Предметная ориентация означает, что данные объединены в категории и хранятся в соответствии с теми областями, которые они описывают, а не с приложениями, которые их используют.
- В хранилище данные интегрируются в целях удовлетворения требований предприятия в целом, а не отдельной функции бизнеса.
- Привязанность данных ко времени выражает их "историчность", т.е. атрибут времени всегда явно присутствует в структурах хранилища данных.
- Неизменяемость означает, что, попав однажды в хранилище, данные уже не изменяются в отличие от оперативных систем, где данные присутствуют только в последней версии, поэтому постоянно меняются.
- 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.

Адаптивные информационные системы

- Потребность в адаптивных информационных системах возникает в тех случаях, когда поддерживаемые ими проблемные области постоянно развиваются.
- В связи с этим адаптивные системы должны удовлетворять ряду специфических требований, а именно: адекватно отражать знания проблемной области в каждый момент времени; быть пригодными для легкой и быстрой реконструкции при изменении проблемной среды.
- Адаптивные свойства информационных систем обеспечиваются за счет интеллектуализации их архитектуры.
- Ядром таких систем является постоянно развиваемая модель проблемной области, поддерживаемая в специальной базе знаний – репозитории.
- Ядро системы управляет процессами генерации или переконфигурирования программного обеспечения.

Тема окончена.

- Вопросы?
- Комментарии?
- Замечания?

Образовательная программа
«Машинное обучение и технологии больших данных»

Технологии проектирования интеллектуальных систем

Тема 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.

ЭТАПЫ ЖИЗНЕННОГО ЦИКЛА

Этап 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.

Этап 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.

Этап 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 идеи разработки.

Этап 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.

Этап 4

- Необходимо максимально снизить риск возникновения недовольства со стороны заказчика и несоответствия его ожиданиям от результата.
- Четкое указание требований защитит и разработчиков в случае спорных вопросов.
- Необходимо решить, как будет происходить коммуникации между разработчиками и заказчиком, и 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.

Этап 4

- 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.

Этап 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 влияют методы разработки, состав 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.

Этап 5

- 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 тематике систем. Таким образом риски и внештатные ситуации сократятся до минимума.

Этап 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.

Этап 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.

Этап 7

- 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).

Этап 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.

Этап 8

- Здесь 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.
- Необходимо исключить технические и программные ошибки, сбои и неточности.

Этап 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.

Этап 10

- Ввод в эксплуатацию.
- После успешного прохождения опытной эксплуатации и тестов выпускается релиз программного продукта, т.е. рабочая версия 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.

Этап 10

- 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.

Этап 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.

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

Тема окончена.

- Вопросы?
- Комментарии?
- Замечания?

Образовательная программа
«Машинное обучение и технологии больших данных»

Технологии проектирования интеллектуальных систем

Тема 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.

Структура интеллектуальной системы:

- информация, содержащаяся 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 выпуска 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.

Подэтапы 1-го этапа проектирования:

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

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.

Преимущества типового проектирования:

- 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.

Методологии проектирования:

- 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.
- 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.
- 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.

Классификация методов проектирования:

- 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.

Классификация информационных систем:

- 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);

Классификация информационных систем:

- 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);

Классификация информационных систем:

- По характеру обработки данных. Проектируемые интеллектуальные системы можно разделить на:
  - информационно-поисковые (здесь производится ввод, систематизация, хранение, выдача информации по запросу пользователя без сложных операций над данными);
  - информационно-решающие (здесь существует алгоритм, по которому происходит обработка информации);

Классификация информационных систем:

- 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.);

Классификация информационных систем:

- 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;

Классификация информационных систем:

- 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);

Классификация информационных систем:

- 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 данным 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:

- обеспечивать создание корпоративных интеллектуальных систем, которые соответствуют целям и задачам организации, а также предъявляемым требованиям по автоматизации бизнес-процессов;
- гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта;
- поддерживать регламент сопровождения, модификации и увеличения системы;
- 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.

Тема окончена.

- Вопросы?
- Комментарии?
- Замечания?

Образовательная программа
«Машинное обучение и технологии больших данных»

Технологии проектирования интеллектуальных систем

Тема 4.

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

29.07.2022

- 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:

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

Технологический процесс и бизнес-процесс

- Не стоит путать бизнес-процесс с 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.

Описание бизнес-процессов:

- 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.

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.

Преимущества процессного подхода перед функциональным

- 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

- 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 будет пошив одежды
- Сопутствующие бизнес-процессы также направлены на производство продукции либо оказание услуг, но в результате сопутствующей работы. Для ателье сопутствующим бизнес-процессом будет перешив или ремонт одежды.
- Вспомогательные бизнес-процессы способствуют выполнению основных. Для ателье такой процесс это ремонт швейного оборудования.

Виды бизнес-процессов:

- 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.

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.

Участники

- 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.

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 Ганта, 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:

- законченность (если это продажа, то бизнес-процесс должен описывать действия, ведущие к ней);
- лаконичность (бизнес-процесс должен быть информативным и в то же время максимально простым для восприятия);
- применение общепризнанных нотаций (устоявшиеся правила, общепризнанные обозначения, с одной стороны, уберегут от ошибок, с другой – не вызовут вопросов, путаницы);
- указание всех участников бизнес-процесса.
- описание должно быть понятным потребителю.

Этапы внедрения бизнес-процесса:

- выявление и документирование бизнес процесса: необходимо изучить ситуацию, чтобы найти проблемные места;
- анализ бизнес процесса: на этом этапе продумывают изменения, которые нужно внести, а также подбирают необходимые инструменты;
- непосредственно описание бизнес-процесса, составление задокументированного плана;
- реализация принятых решений;
- контроль, анализ бизнес процесса.

ARIS eEPS

- 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

- 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.

Тема окончена.

- Вопросы?
- Комментарии?
- Замечания?

Образовательная программа
«Машинное обучение и технологии больших данных»

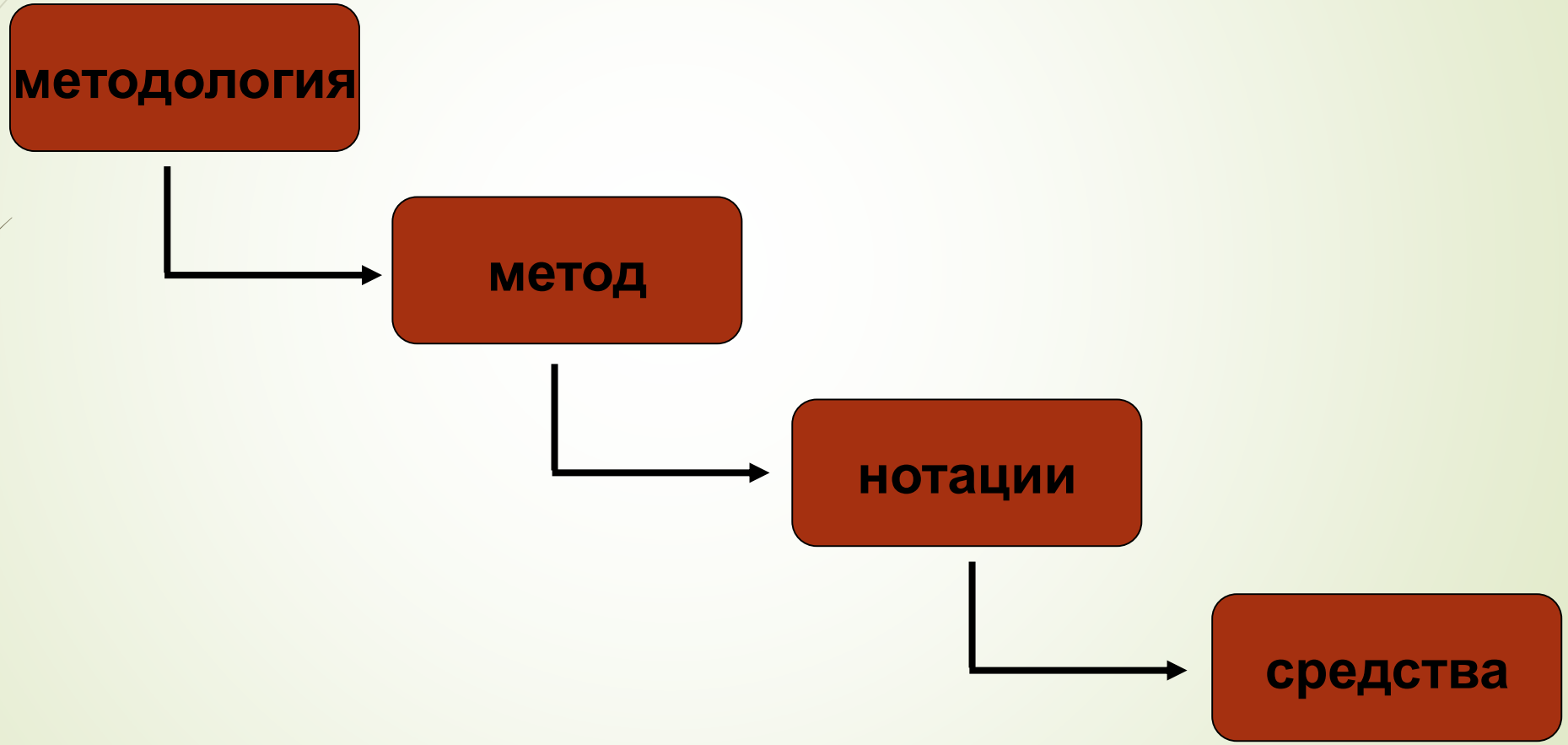
Технологии проектирования интеллектуальных систем

Тема 5.
Автоматизированное проектирование ИС с использованием CASE-технологий: структурный подход к проектированию ИС

29.07.2022

- ✓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.

CASE-технология основана на парадигме:



- ✓*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).

Методики:

- ***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.
- ✓***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.
- ***Функциональный блок*** обозначает определенную функцию в рамках рассматриваемой системы и в графическом виде обозначается прямоугольником.

Функциональная методика IDEF0

- ***Интерфейсная дуга*** обозначает элемент системы, который обрабатывается функциональным блоком или оказывает некоторое влияние на выполнение блоком своей функции.
- Графически интерфейсная дуга отображается в виде однонаправленной стрелки.
- Началом или концом могут быть только функциональные блоки, началом может быть только входная сторона, а концом любые другие.
- Входящими и исходящими дугами обозначаются финансовые и материальные потоки, потоки информации и ресурсы.
- Управляющими дугами обозначаются объекты потоков информации, а дугами механизмов только ресурсы.

Функциональная методика IDEF0

- ***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.

Функциональная методика IDEF0

- 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.

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

- 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.
- Составные части задачи структурируются в иерархическую структуру. На каждом уровне иерархии добавляются новые задачи и подзадачи.
- Выделяются только основные задачи. И именно им предлагается уделять основное внимание. Остальные задачи считаются несущественными и убираются из поля внимания.
- Задача должна быть максимально формализована и используется методический подход к ее решению.
- Все подзадачи и элементы этих подзадач должны быть обоснованы и согласованы.
- Так как подход имеет название структурного – все данные задач должны быть также структурированы и организованы в иерархические структуры.

Средства и виды моделей для структурного подхода (модели иллюстрируются при помощи диаграмм):

- Методология структурного анализа и проектирования. В международной терминологии Structured Analysis and Design Technique (SADT);
- Методология потоков данных. В международной терминологии Data Flow Diagrams (DFD);
- Модели Сущность – связь. В международной терминологии Entity-Relationship Diagrams (ERD).

Методология структурного анализа и проектирования (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 структурного анализа и проектирования (SADT). *Понятия методологии:*

- 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.

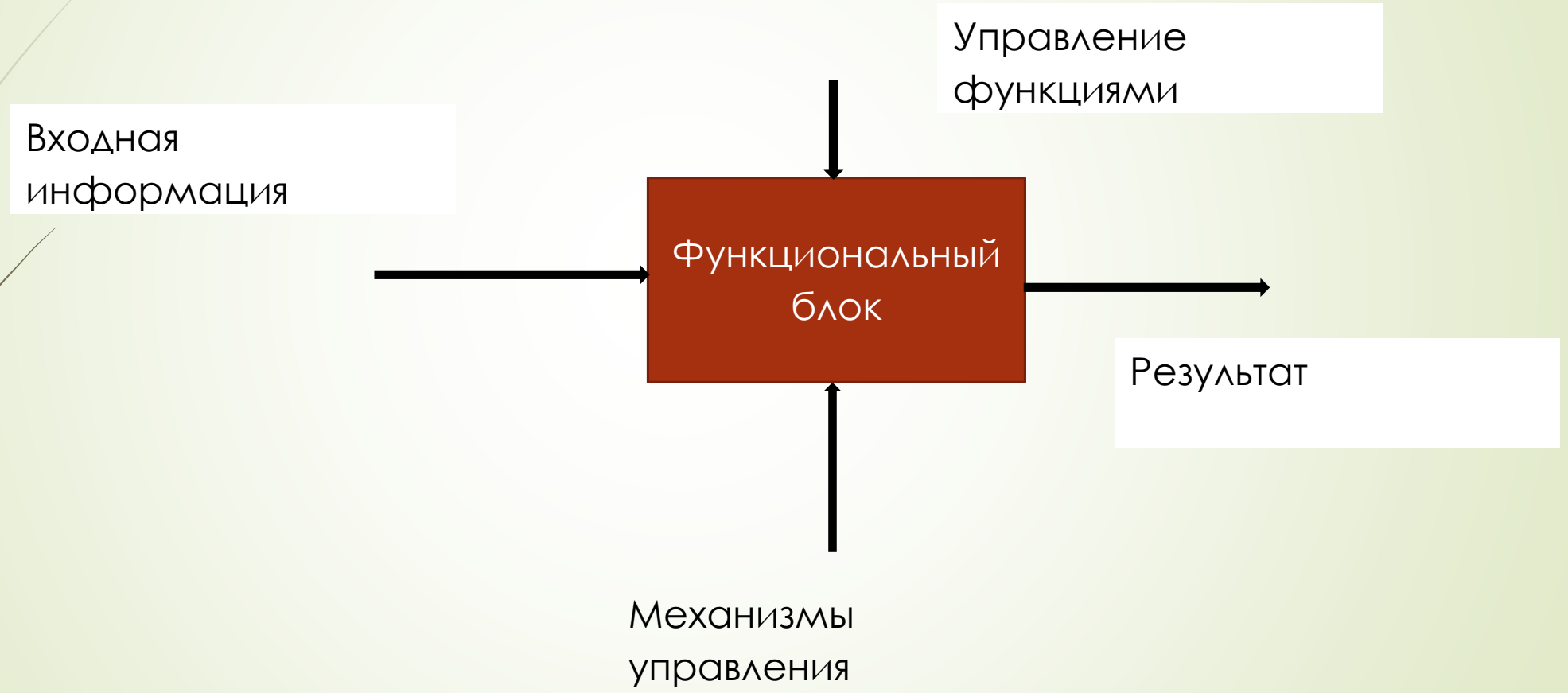
Методология структурного анализа и проектирования (SADT)

- 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.

Методология структурного анализа и проектирования (SADT)

- 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.

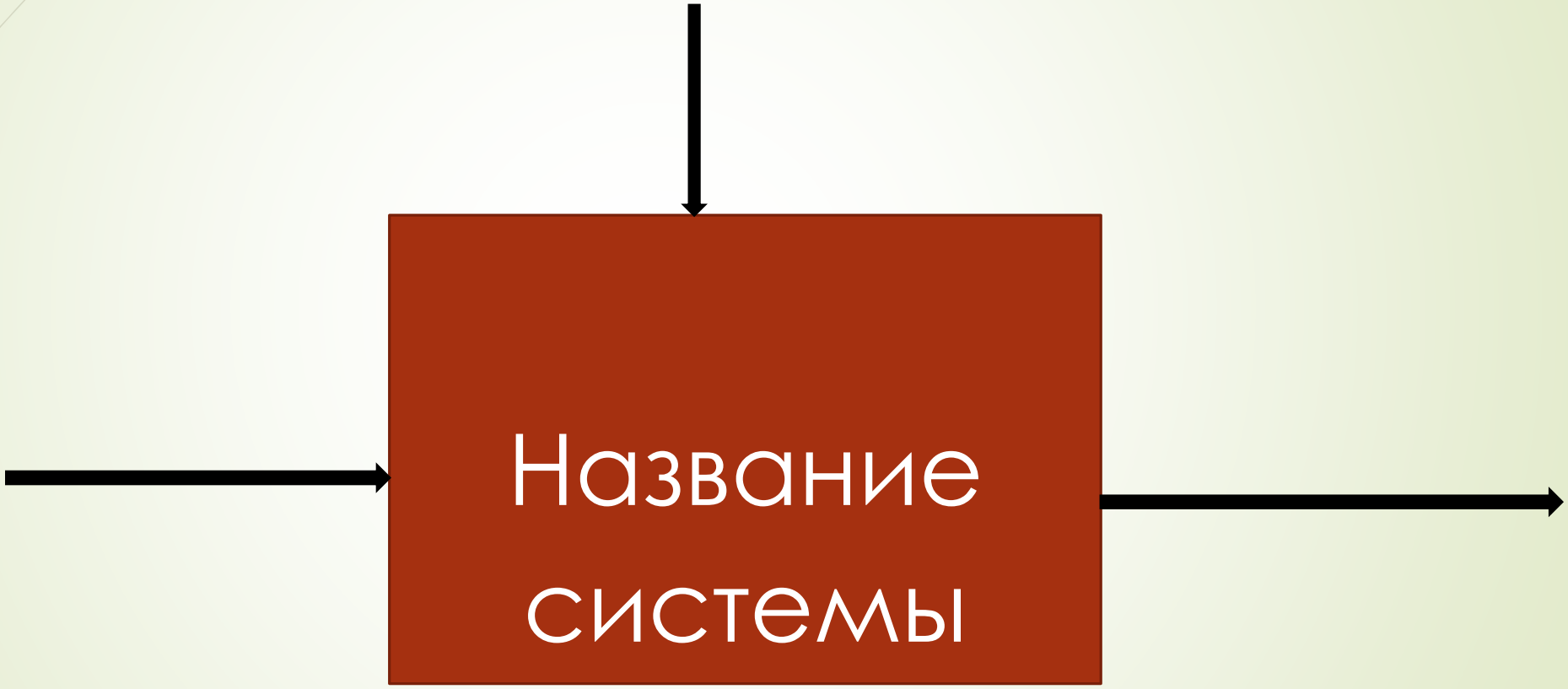
Методология структурного анализа и проектирования (SADT)



Методология структурного анализа и проектирования (SADT)

- 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 необходима входящая информация и что будет в результате.
- Имя, которое будет указано на центральном функциональном блоке, будет общим для всей системы.

Методология структурного анализа и проектирования (SADT)



Методология структурного анализа и проектирования (SADT)

- 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.

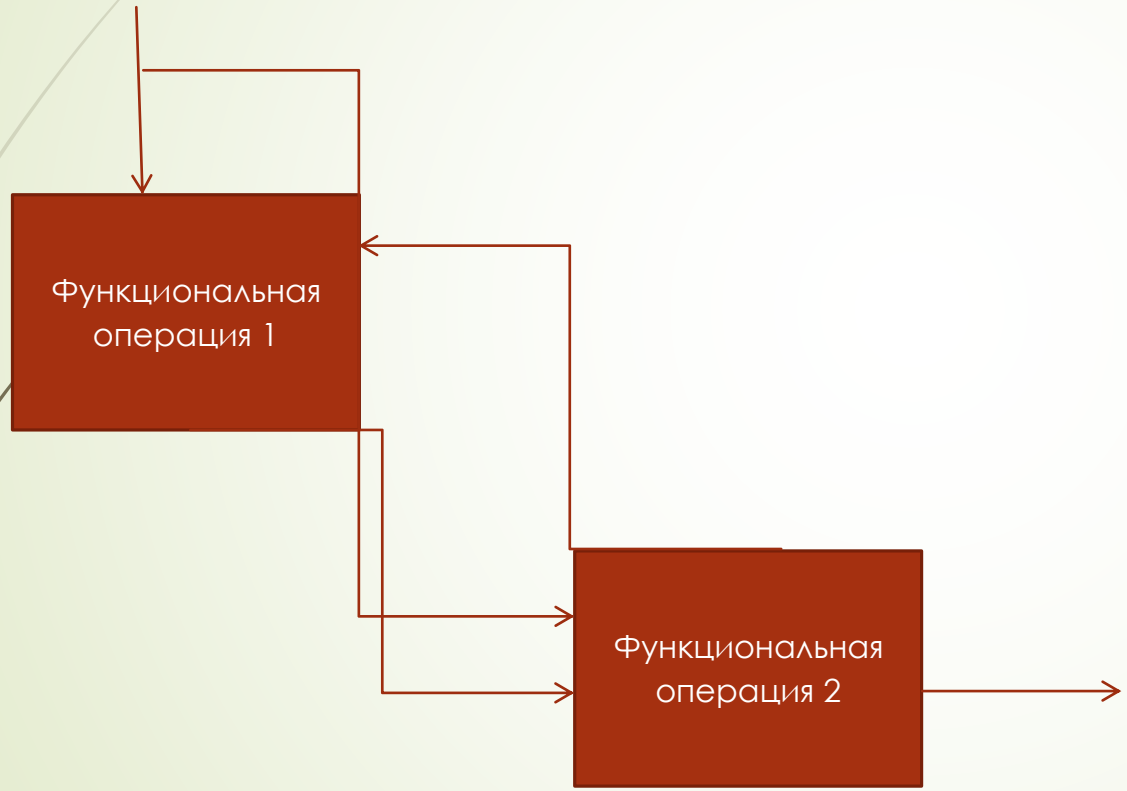
Методология структурного анализа и проектирования (SADT)

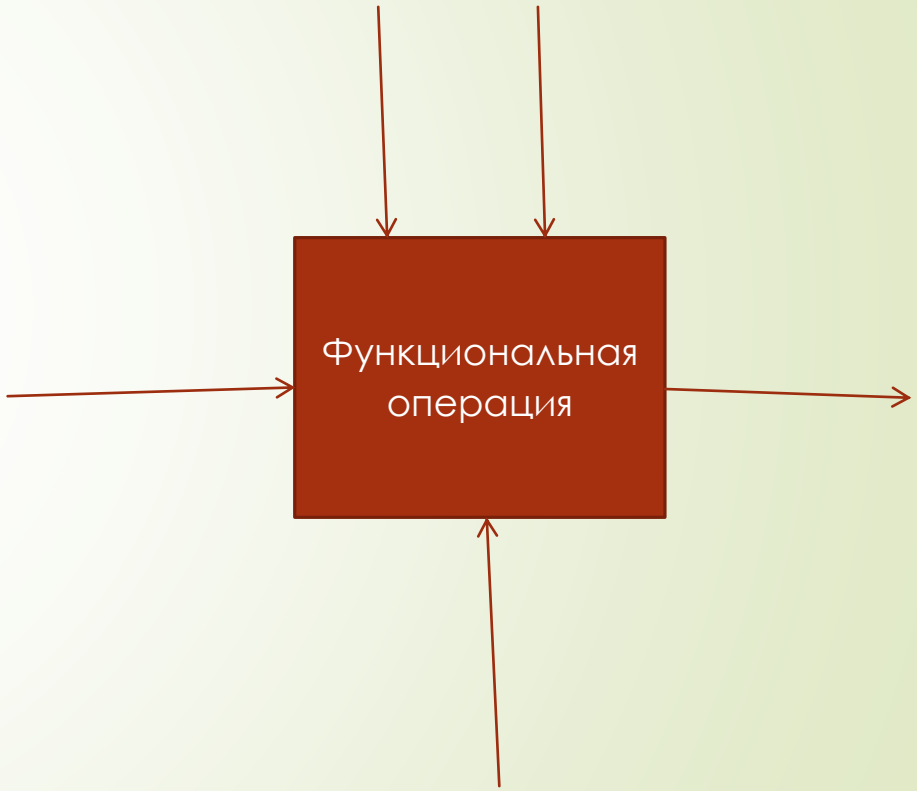
- Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию.
- Кроме того, модель не может опустить какие-либо элементы, т.е., как уже отмечалось, родительский блок и его интерфейсы обеспечивают контекст.
- К нему нельзя ничего добавить, и из него не может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

Методология структурного анализа и проектирования (SADT)

- 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.

Методология структурного анализа и проектирования (SADT)

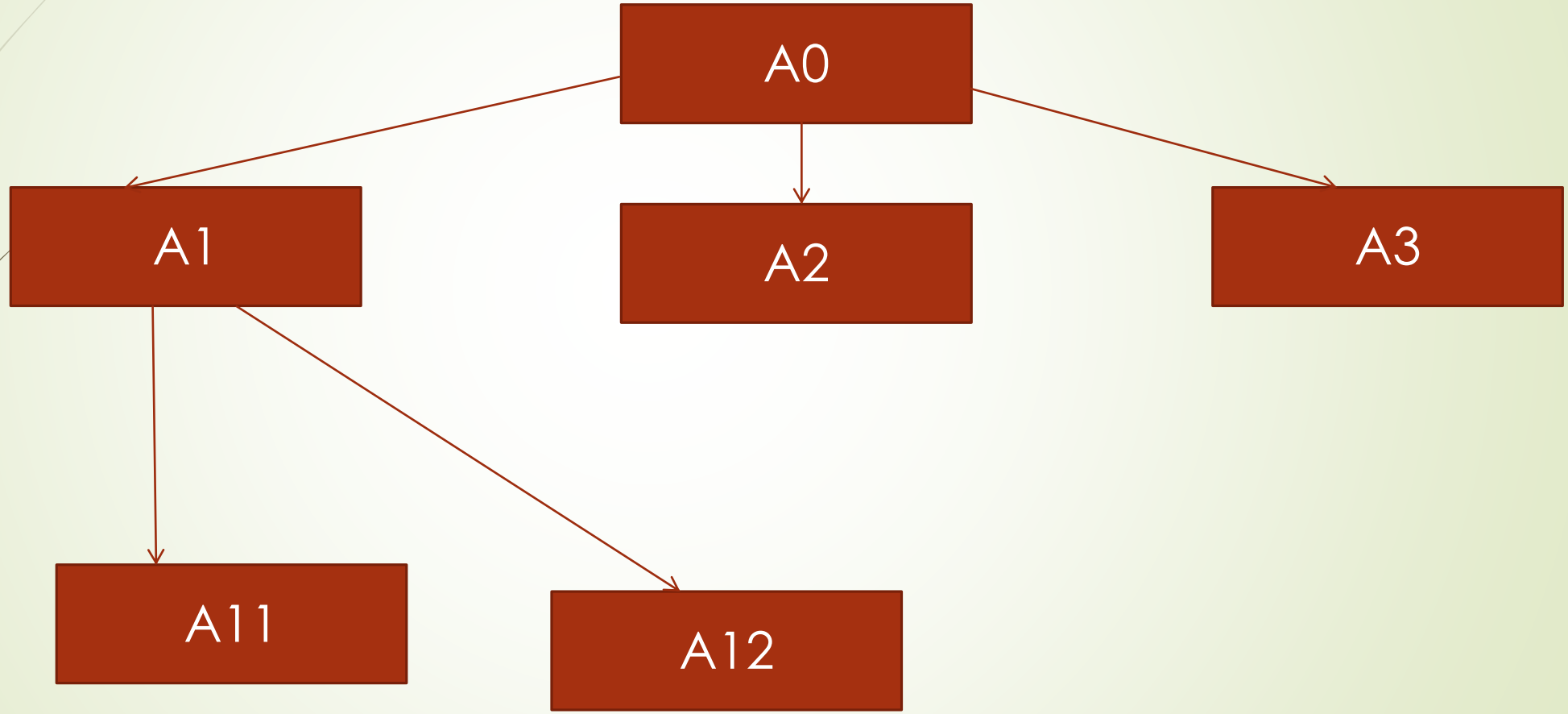




Методология структурного анализа и проектирования (SADT)

- 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.

Методология структурного анализа и проектирования (SADT)



Методология потоков данных (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.

Методология потоков данных (DFD)

- 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.

Модели Сущность – связь (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.
- Каждая сущность должна обладать уникальным идентификатором.
- Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности.

Модели Сущность – связь (ERD). Свойства сущностей:

- 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.

Модели Сущность – связь (ERD)

- 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.

Модели Сущность – связь (ERD)

- 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).

Модели Сущность – связь (ERD)

- **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.

Тема окончена.

- Вопросы?
- Комментарии?
- Замечания?

Образовательная программа
«Машинное обучение и технологии больших данных»

Технологии проектирования интеллектуальных систем

Тема 6.
Автоматизированное проектирование ИС с использованием CASE-технологий: объектно-ориентированный подход к проектированию ИС

- Отличие между функциональным и объектным подходом заключается в способе декомпозиции системы.
- Объектный подход использует 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.

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.

Сравнение методик

- 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-ориентированном 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.

Свойства объектов в рамках объектно-ориентированного подхода при проектировании интеллектуальных систем:

- *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.

Стадии формирования модели проектирования интеллектуальной системы при использовании объектно-ориентированного подхода:

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.

- 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.

UML

- 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)».

Аспекты UML-модели:

- 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.
- 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.
- 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

- 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.

Сущности в UML

- *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.

Отношения в языке UML:

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

Диаграмма в UML

- 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.

Типы диаграмм в UML:

- 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).

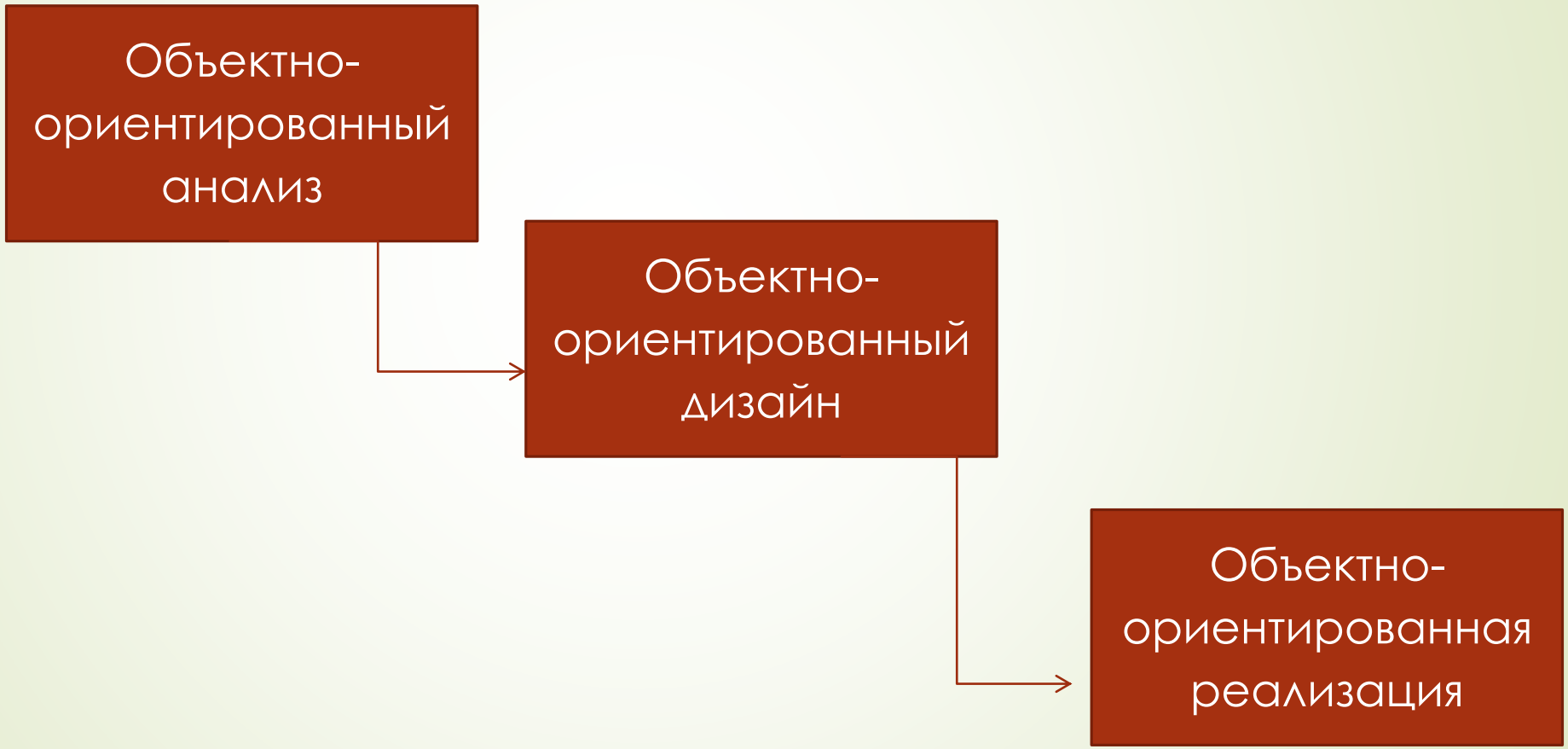
К инструментам, поддерживающим методологию UML, относятся:

- Rational Rose (Rational Software),
- Paradigm Plus (CA/Platinum),
- ARIS (IDS Sheer AG),
- Together Designer (Borland)
- и др.

Использование UML

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

Этапы жизненного цикла разработки объектно-ориентированной системы



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.

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.

Объектно-ориентированный анализ

- *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 применяют 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.
- *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.

Тема окончена.

- Вопросы?
- Комментарии?
- Замечания?

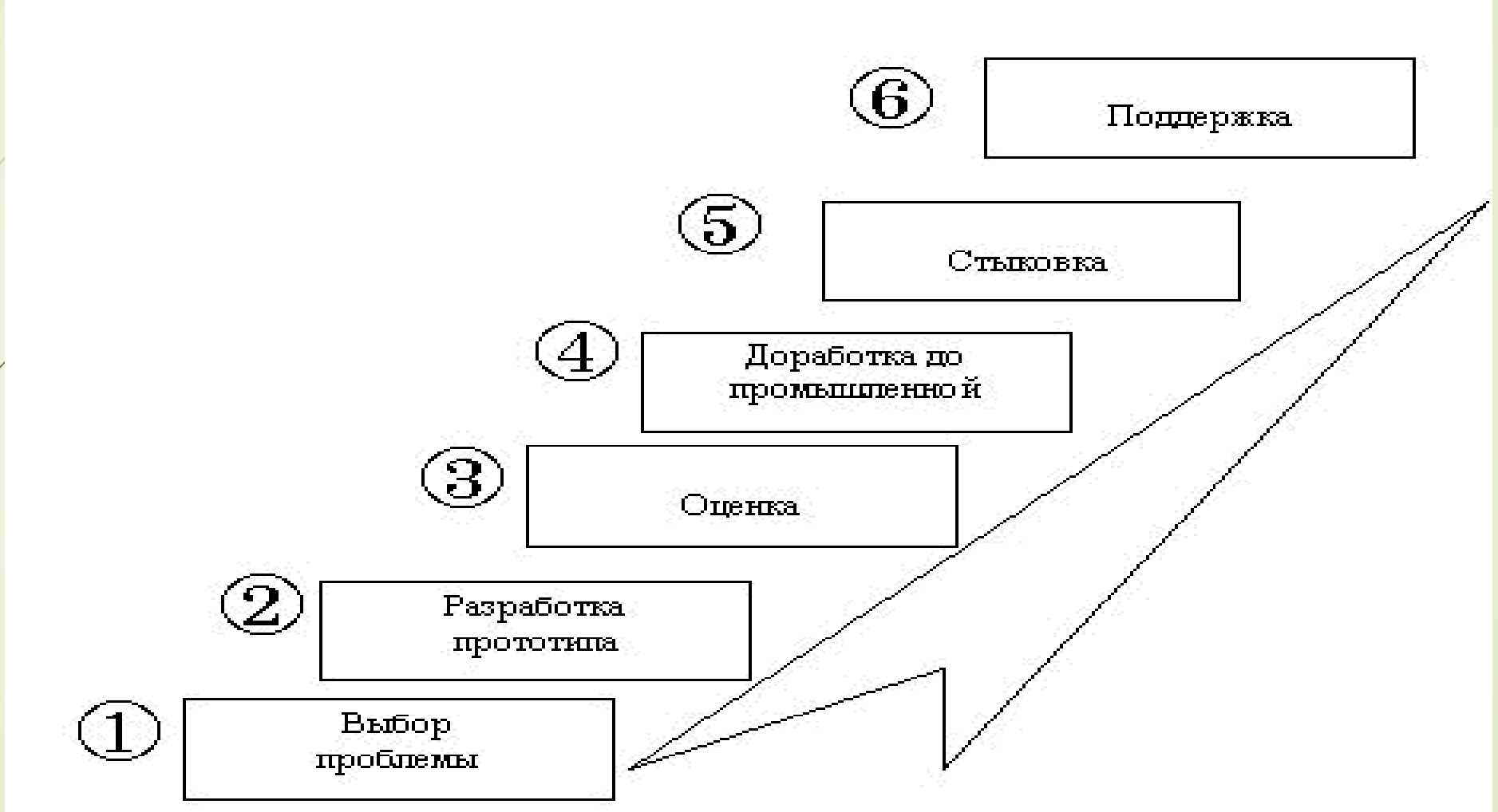
Образовательная программа
«Машинное обучение и технологии больших данных»

Технологии проектирования интеллектуальных систем

Тема 7.

Методы и средства прототипного проектирования ИС: технология быстрого проектирования ИС

Этапы разработки интеллектуальной системы



Этап 1: Выбор подходящей проблемы

- определение проблемной области и задачи;
- нахождение эксперта, желающего сотрудничать при решении проблемы, и назначение коллектива разработчиков;
- определение предварительного подхода к решению проблемы;
- анализ расходов и прибыли от разработки;
- подготовка подробного плана разработки.

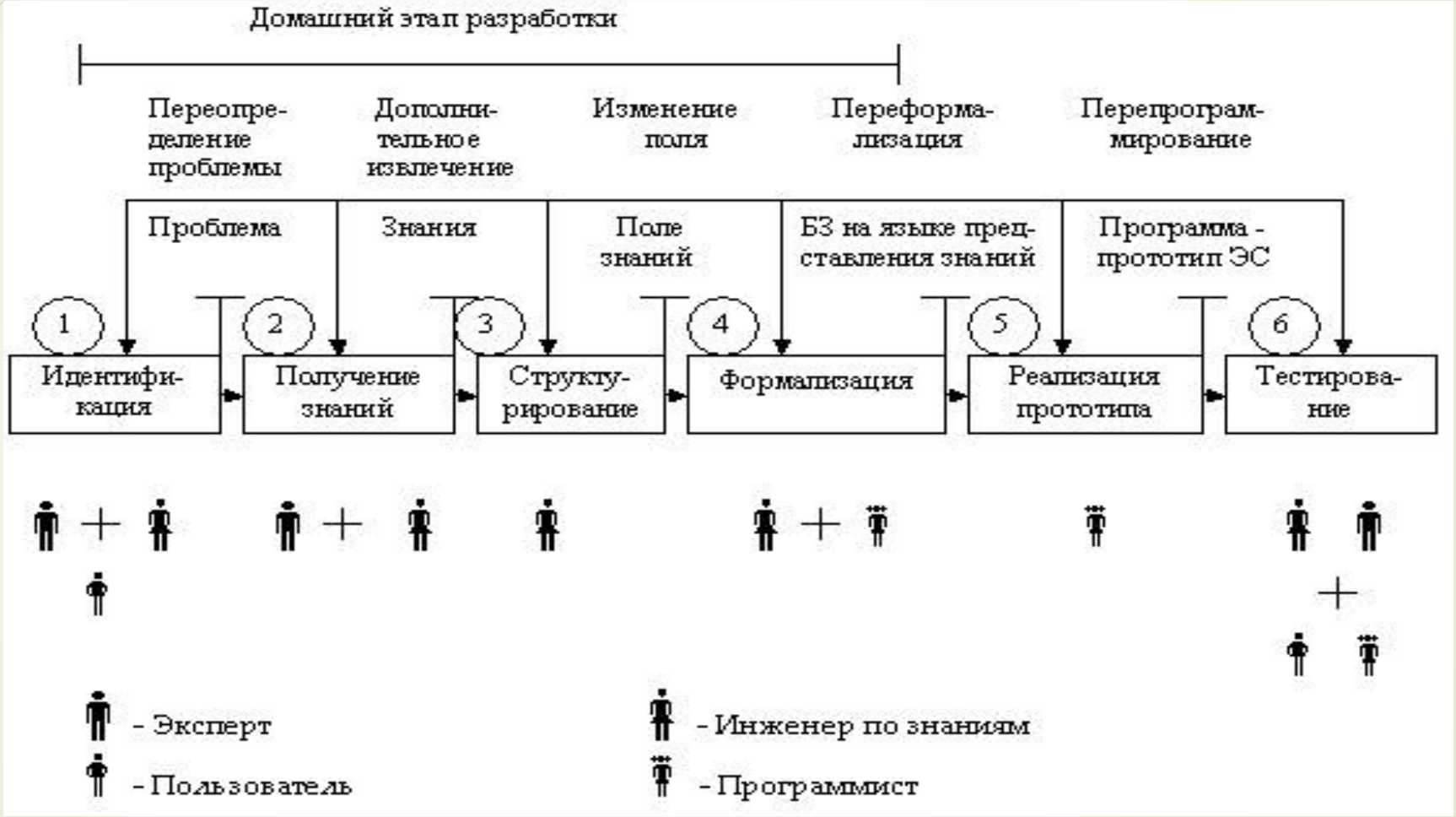
Обоснование необходимости разработки и внедрения интеллектуальных систем:

- 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.

Этап 2: Разработка прототипной системы

- Прототипная система является усеченной версией интеллектуальной системы, спроектированной для проверки правильности кодирования фактов, связей и стратегий рассуждения эксперта.
- Она также дает возможность инженеру по знаниям привлечь эксперта к активному участию в разработке интеллектуальной системы и, следовательно, к принятию им обязательства приложить все усилия для создания системы в полном объеме.
- Объем прототипа – несколько десятков правил, фреймов или примеров.

Стадии разработки прототипа интеллектуальной системы



Идентификация проблемы

- Уточняется задача, планируется ход разработки прототипа интеллектуальной системы, определяются:
  - необходимые ресурсы (время, люди, ЭВМ и т.д.);
  - источники знаний (книги, дополнительные эксперты, методики);
  - имеющиеся аналогичные экспертные системы;
  - 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.

Извлечение знаний

- 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.

Структурирование или концептуализация знаний

- 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.

Формализация

- Строится формализованное представление концепций предметной области на основе выбранного языка представления знаний.
- Традиционно на этом этапе используются:
  - логические методы (исчисления предикатов I порядка и др.);
  - продукционные модели (с прямым и обратным выводом);
  - семантические сети;
  - фреймы;
  - объектно-ориентированные языки, основанные на иерархии классов, объектов и др.

Формализация

- 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.
- Все чаще на этой стадии используется симбиоз языков представления знаний, например, в системе ОМЕГА-фреймы + семантические сети + полный набор возможностей языка исчисления предикатов.
- Средняя продолжительность 1-2 месяца.

Реализация

- 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: LISP, FRL, SmallTalk и др.;
  - использование инструментальных средств разработки интеллектуальных систем типа СПЭИС, ПИЭС;
  - использование "пустых" ЭС или "оболочек" типа ЭКСПЕРТ, ФИАКР и др.
- Реализация – разработка программного комплекса, демонстрирующего жизнеспособность подхода в целом. Чаще всего первый прототип отбрасывается на этапе реализации действующей ЭС.
- Средняя продолжительность 1-2 месяца.

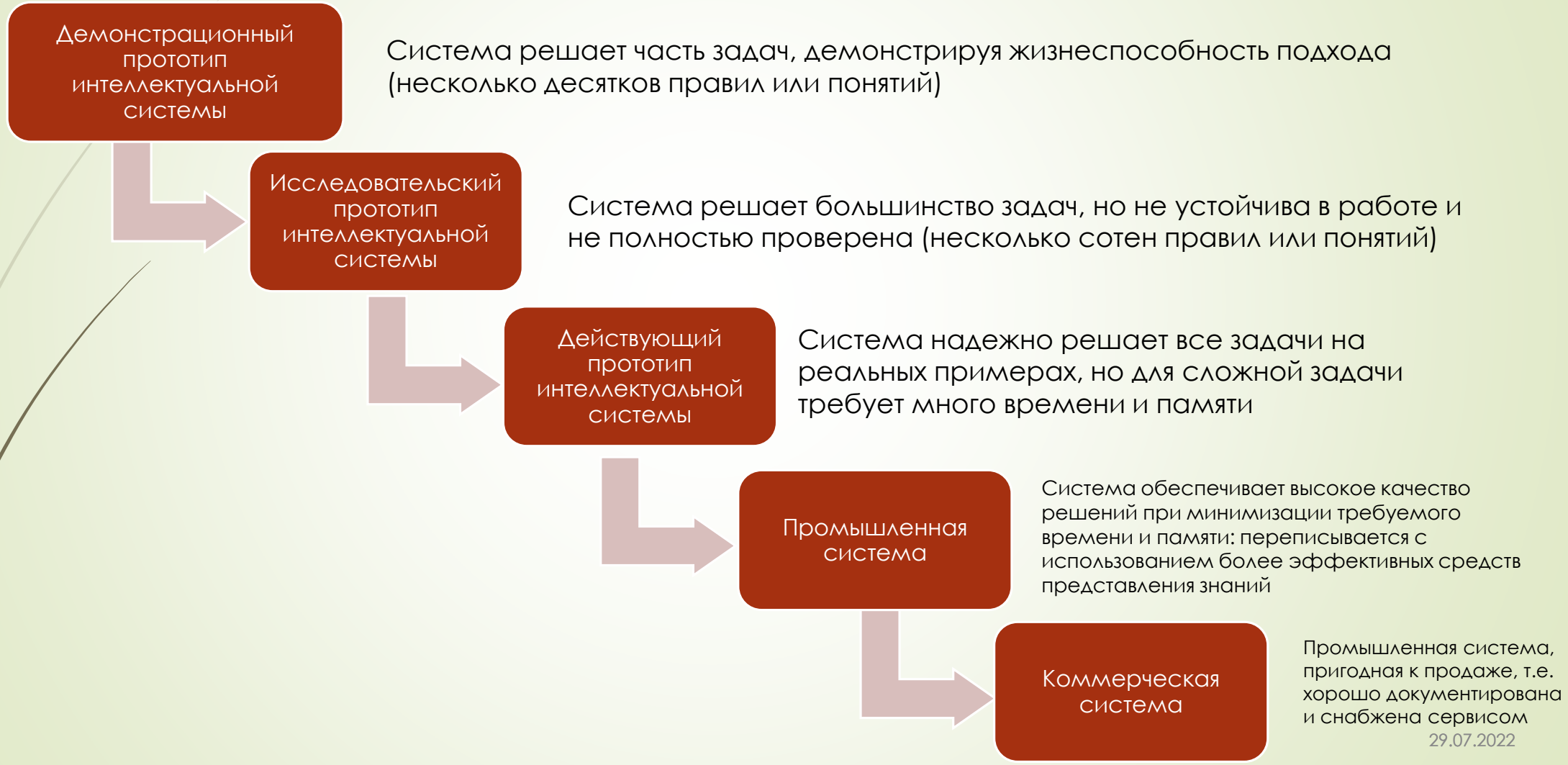
Тестирование

- 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.

Этап 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.

Переход от прототипа к промышленной интеллектуальной системе



Этап 3: Развитие прототипа до промышленной интеллектуальной системы

- Основное на третьем этапе заключается в добавлении большого числа дополнительных эвристик. Эти эвристики обычно увеличивают глубину системы, обеспечивая большее число правил для трудноуловимых аспектов отдельных случаев.
- В то же время эксперт и инженер по знаниям могут расширить охват системы, включая правила, управляющие дополнительными подзадачами или дополнительными аспектами интеллектуальной задачи (метазнания).
- После установления основной структуры интеллектуальной системы инженер по знаниям приступает к разработке и адаптации интерфейсов, с помощью которых система будет общаться с пользователем и экспертом.

Этап 3: Развитие прототипа до промышленной интеллектуальной системы

- Необходимо обратить особое внимание на языковые возможности интерфейсов, их простоту и удобство для управления работой интеллектуальной системы.
- Система должна обеспечивать пользователю возможность легким и естественным образом спрашивать непонятное, приостанавливать работу и т.д. В частности, могут оказаться полезными графические представления.
- На этом этапе разработки большинство экспертов узнают достаточно о вводе правил и могут сами вводить в систему новые правила. Таким образом, начинается процесс, во время которого инженер по знаниям передает право собственности и контроля за системой эксперту для уточнения, детальной разработки и обслуживания.

Этап 4: Оценка системы

- После завершения этапа разработки промышленной интеллектуальной системы необходимо провести ее тестирование в отношении критериев эффективности. К тестированию широко привлекаются другие эксперты с целью апробирования работоспособности системы на различных примерах.
- Экспертные 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 "прозрачность" работы системы, удобство интерфейсов и др.);
  - критерии приглашенных экспертов (оценка советов-решений, предлагаемых системой, сравнение ее с собственными решениями, оценка подсистемы объяснений и др.);
  - критерии коллектива разработчиков (эффективность реализации, производительность, время отклика, дизайн, широта охвата предметной области, непротиворечивость БЗ, 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.).

Этап 5: Стыковка системы

- На этом этапе осуществляется стыковка интеллектуальной системы с другими программными средствами в среде, в которой она будет работать, и обучение людей, которых она будет обслуживать.
- Иногда это означает внесение существенных изменений. Такое изменение требует непременного вмешательства инженера по знаниям или какого-либо другого специалиста, который сможет модифицировать систему.
- Под стыковкой подразумевается также разработка связей между интеллектуальной системой и средой, в которой она действует.
- Когда экспертная система уже готова, инженер по знаниям должен убедиться в том, что эксперты, пользователи и персонал знают, как эксплуатировать и обслуживать ее.

Этап 5: Стыковка системы

- 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).

Пример 1

- Успешно состыкована со своим окружением система PUFF – экспертная система для диагностики заболеваний легких.
- После того, как PUFF была закончена и все были удовлетворены ее работой, систему перекодировали с LISPa на Бейсик.
- Затем систему перенесли на ПК, которая уже работала и больнице.
- В свою очередь, эта ПК была связана с измерительными приборами.
- Данные с измерительных приборов сразу поступают в ПК.
- PUFF обрабатывает эти данные и печатает рекомендации для врача. Врач в принципе не взаимодействует с PUFF.
- Система полностью интегрирована со своим окружением – она представляет собой интеллектуальное расширение аппарата исследования легких, который врачи давно используют.

Пример 2

- Другая система, которая хорошо функционирует в своем окружении, - CAT-1 - экспертная система для диагностики неисправностей дизелей локомотивов.
- Эта система была разработана также на LISPe, а затем переведена на FORTH, чтобы ее можно было более эффективно использовать в различных локомотивных цехах.
- Мастер по ремонту запрашивает систему: определить возможные причины неисправности дизеля.
- Система связана с видеодиском, с помощью которого мастеру дают визуальные объяснения и подсказки относительно более подробных проверок, которые ему нужно сделать.
- Кроме того, если оператор не уверен в том, как устранить неисправность, система предоставляет ему обучающие материалы, которые фирма подготовила предварительно, и показывает ему на видеотерминале.
- Таким образом, мастер по ремонту может с помощью интеллектуальной системы диагностировать проблему, найти тестовую процедуру, которую он должен использовать, получить на дисплее объяснение, как провести тест, или получит инструкции о том, как справиться с возникшей проблемой.

Этап 6: Поддержка системы

- При перекодировании системы на язык, подобный Си, повышается ее быстродействие и увеличивается переносимость, однако гибкость при этом уменьшается.
- Это приемлемо лишь в том случае, если система сохраняет все знания проблемной области, и это знание не будет изменяться в ближайшем будущем.
- Однако, если экспертная система создана именно из-за того, что проблемная область изменяется, то необходимо поддерживать систему в инструментальной среде разработки.

Пример

- Удачным примером ЭС, внедренной таким образом, является XCON (R1) – ЭС, которую фирма DEC использует для комплектации ЭВМ семейства VAX.
- Одна из ключевых проблем, с которой столкнулась фирма DEC, – необходимость постоянного внесения изменений для новых версий оборудования, новых спецификаций и т.д.

Тема окончена.

- Вопросы?
- Комментарии?
- Замечания?

Образовательная программа
«Машинное обучение и технологии больших данных»

Технологии проектирования интеллектуальных систем

Тема 8.
Методы и средства прототипного проектирования ИС: методы и средства организации метаинформации проекта ИС

29.07.2022

- 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.

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.

Учебный пример

- 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 модель 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?

Учебный пример

- 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.

Учебный пример

- 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.

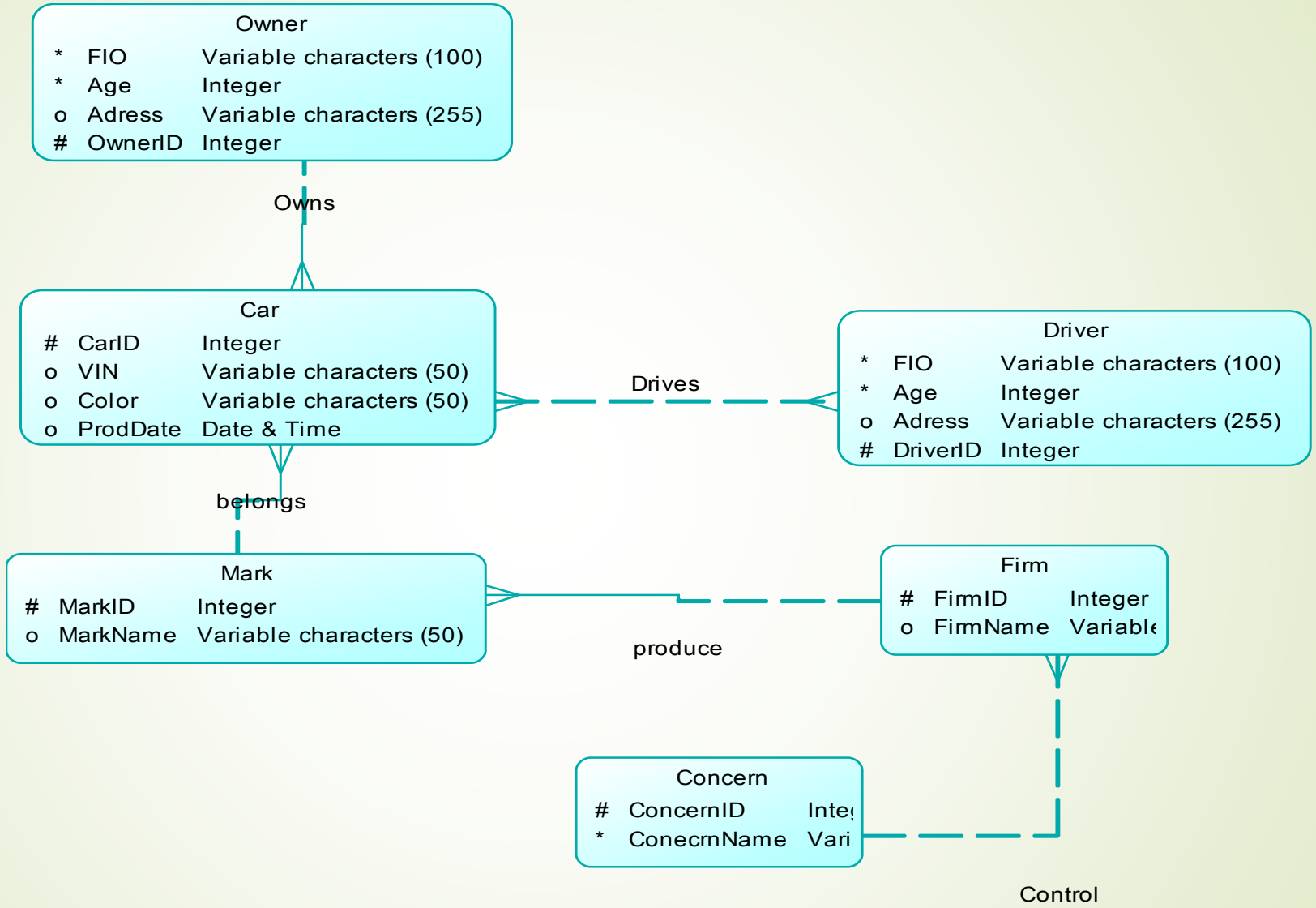
Учебный пример

- Итак, мы выделили 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?

Учебный пример

- 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.

Учебный пример

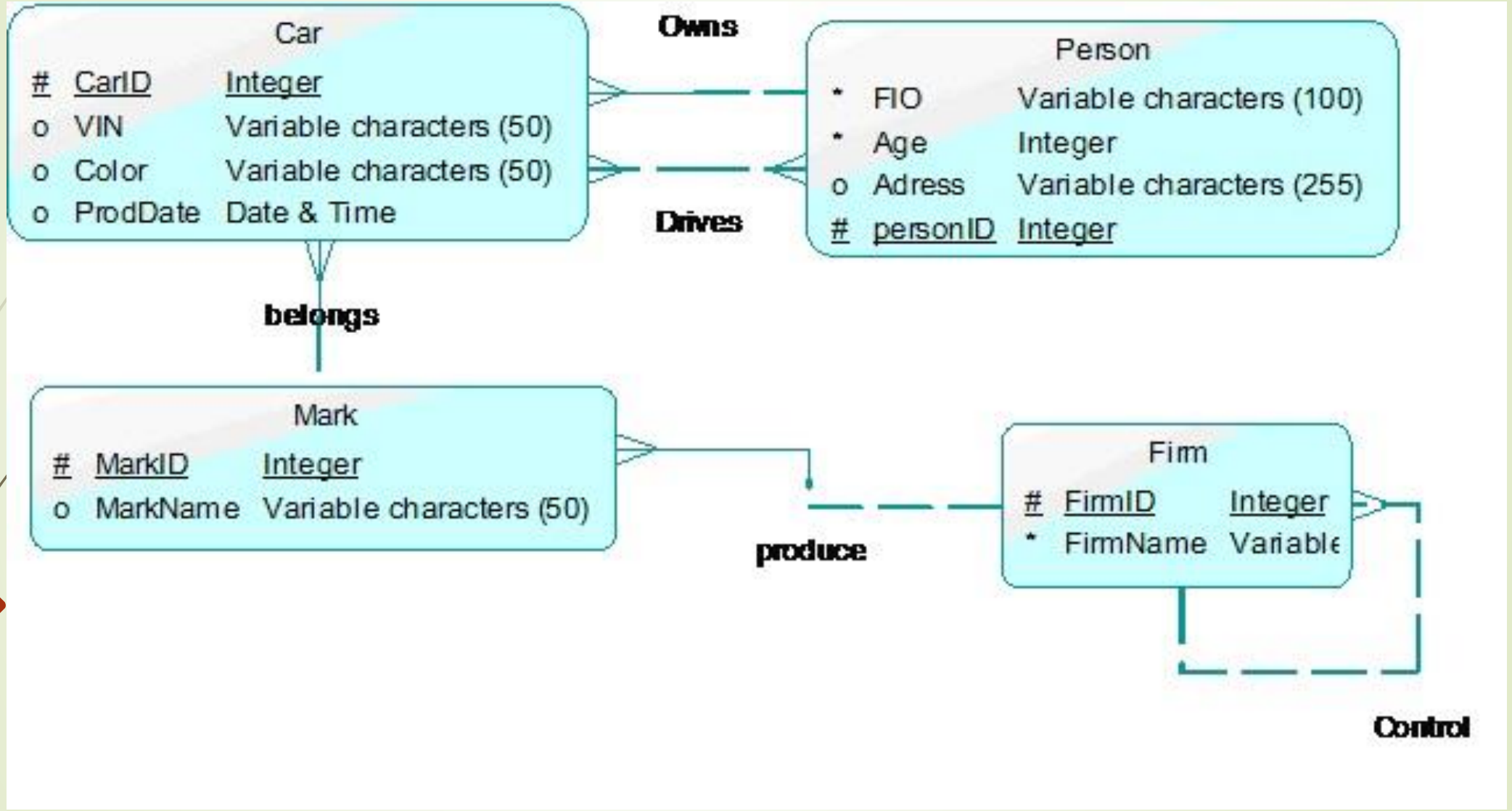


Первый вариант концептуальной модели для нашего примера

Учебный пример

- 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.

Учебный пример



Обновленная модель

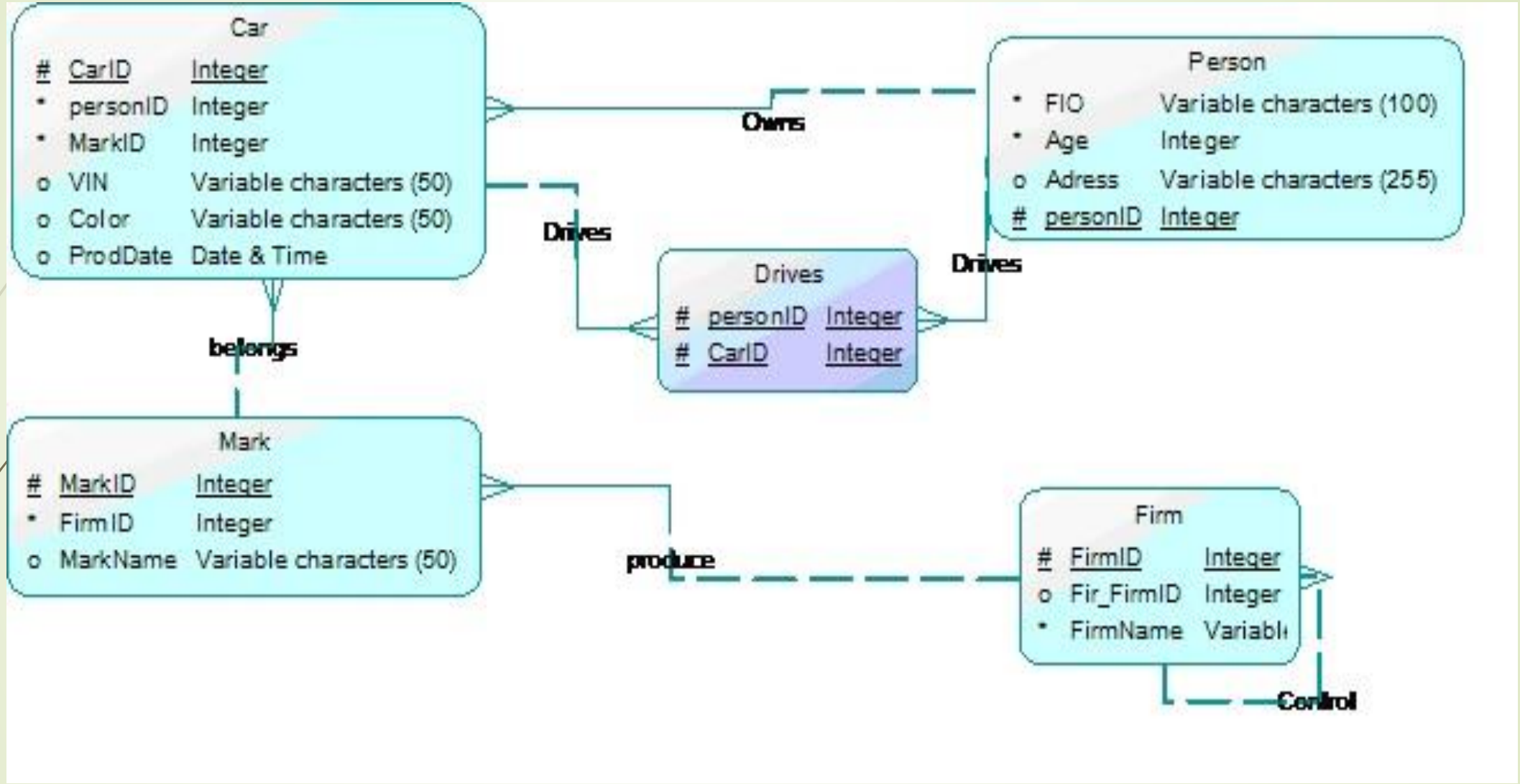
Учебный пример

- Теперь введено понятие Персона, а факт владения автомобилем и факт 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.

Учебный пример

- 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.
- 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.

Учебный пример

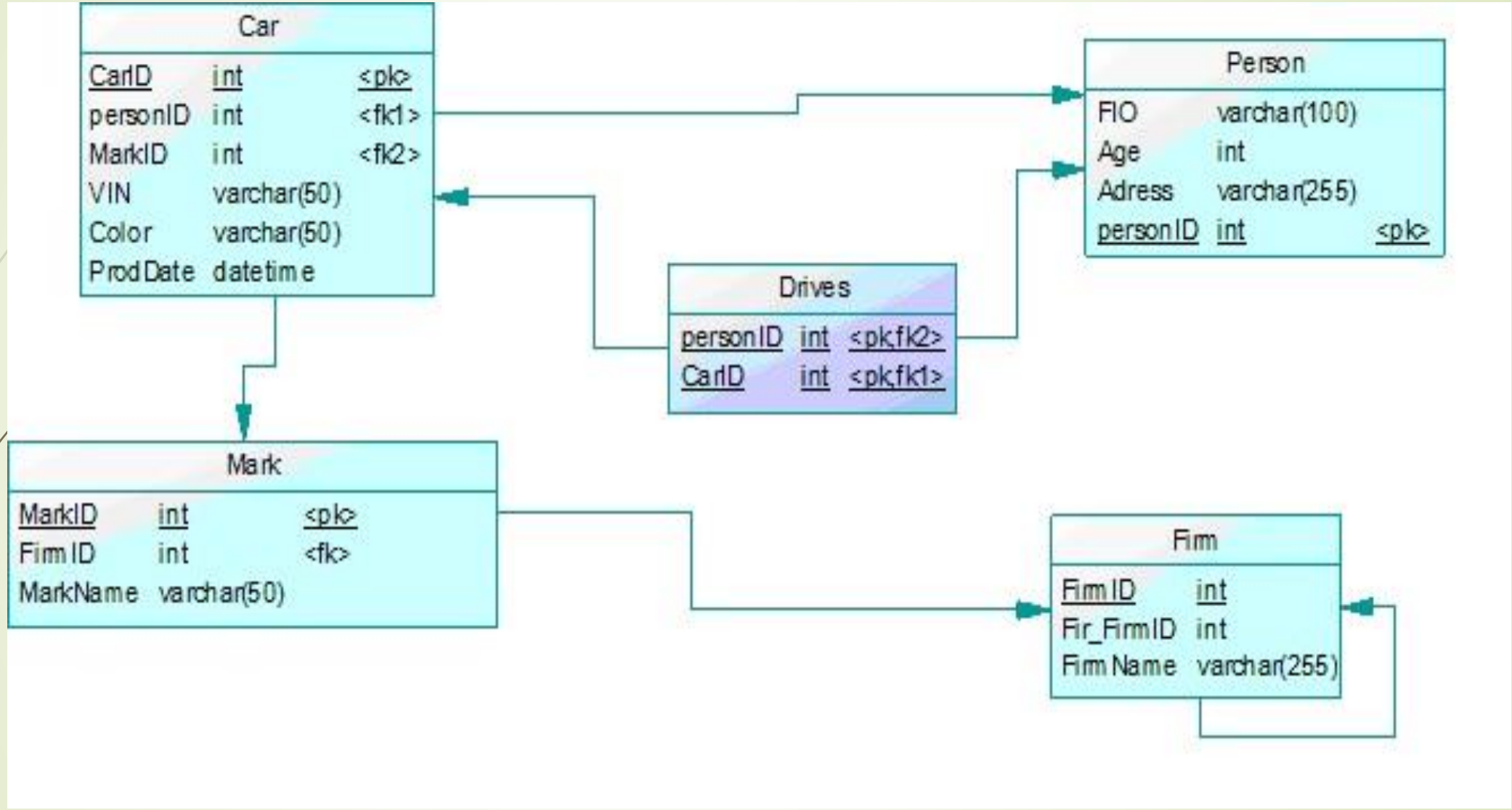


Логическая модель для нашего примера

Реализация связей 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.

Учебный пример



Физическая модель для нашего примера

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.

Тема окончена.

- Вопросы?
- Комментарии?
- Замечания?

Образовательная программа
«Машинное обучение и технологии больших данных»

Технологии проектирования интеллектуальных систем

Тема 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

- Порождающие паттерны решают важную задачу повышения независимости различных компонентов программного обеспечения.
- В самом деле, огромным преимуществом объектно-ориентированного подхода являются наследование и полиморфизм.
- 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

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

Поведенческие паттерны

- 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.

Каждый паттерн описывается по следующей схеме:

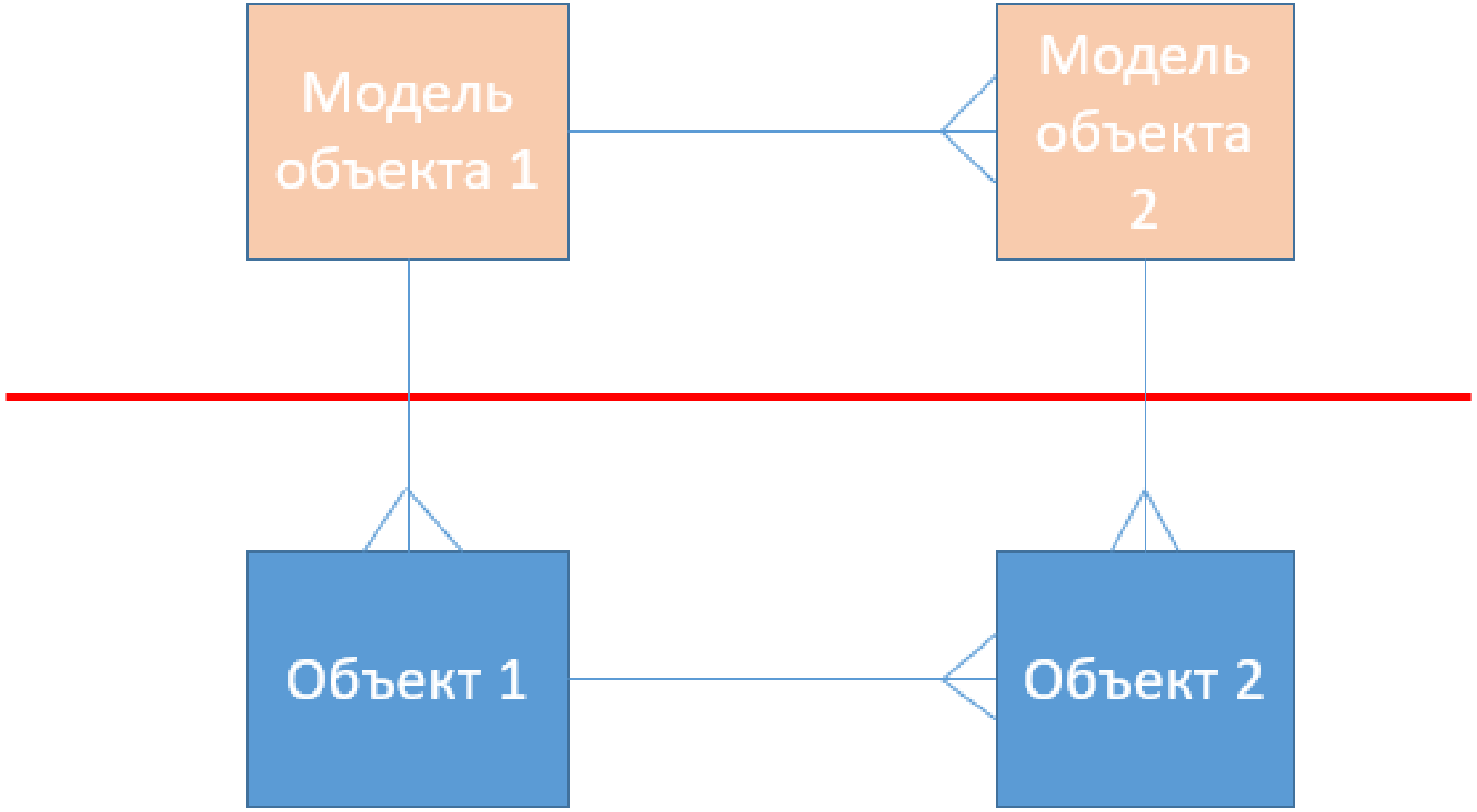
- название;
- альтернативные названия;
- описание, суть, цель;
- решаемая проблема;
- структура (обычно в виде UML-модели);
- примеры реализации с кодом;
- рекомендации по реализации;
- связь с другими паттернами.

- 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.

Наборы паттернов и антипаттернов в сфере информационных технологий:

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

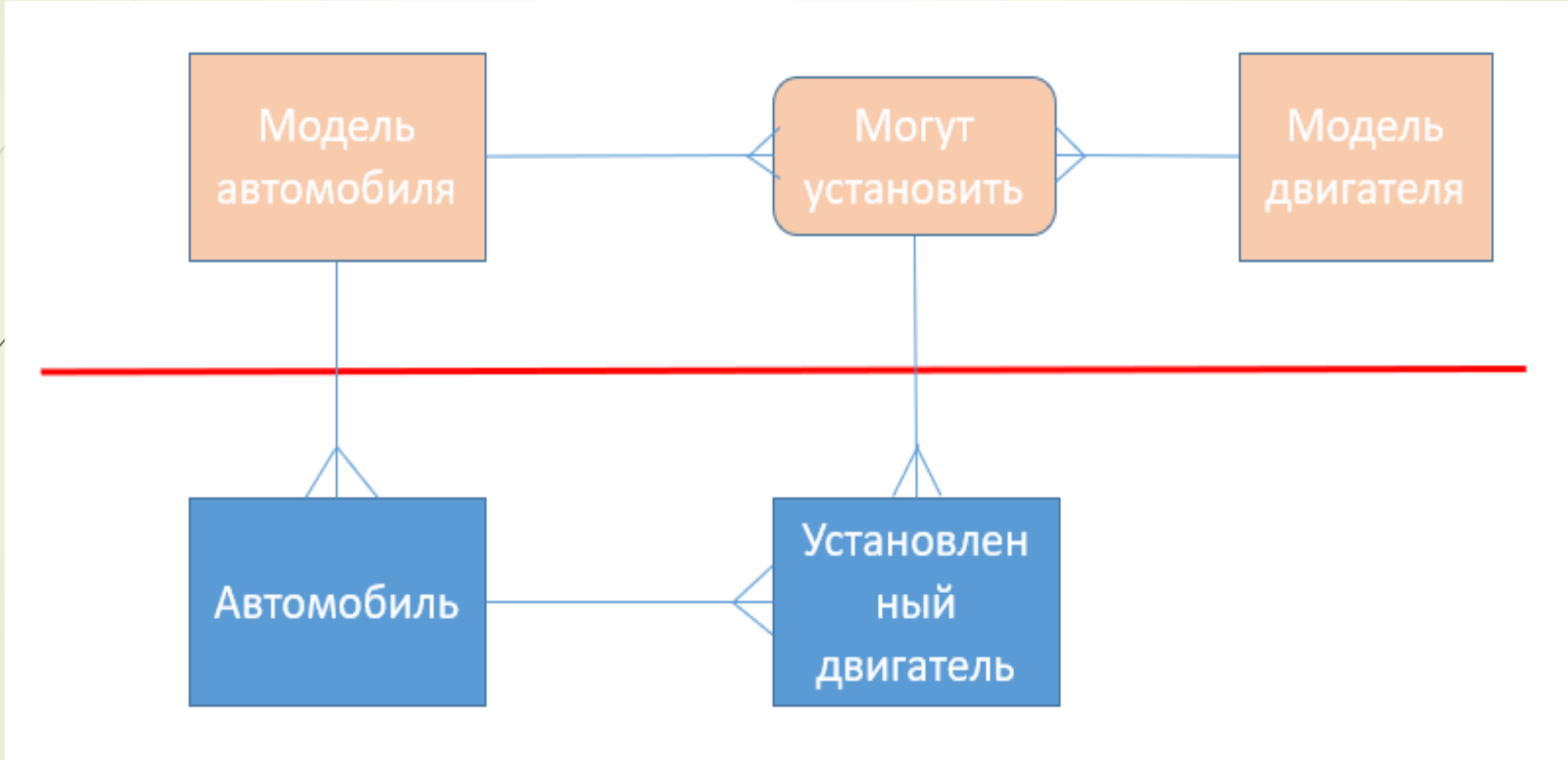
Пример паттерна «Модель-реализация»



Пример паттерна «Модель-реализация»

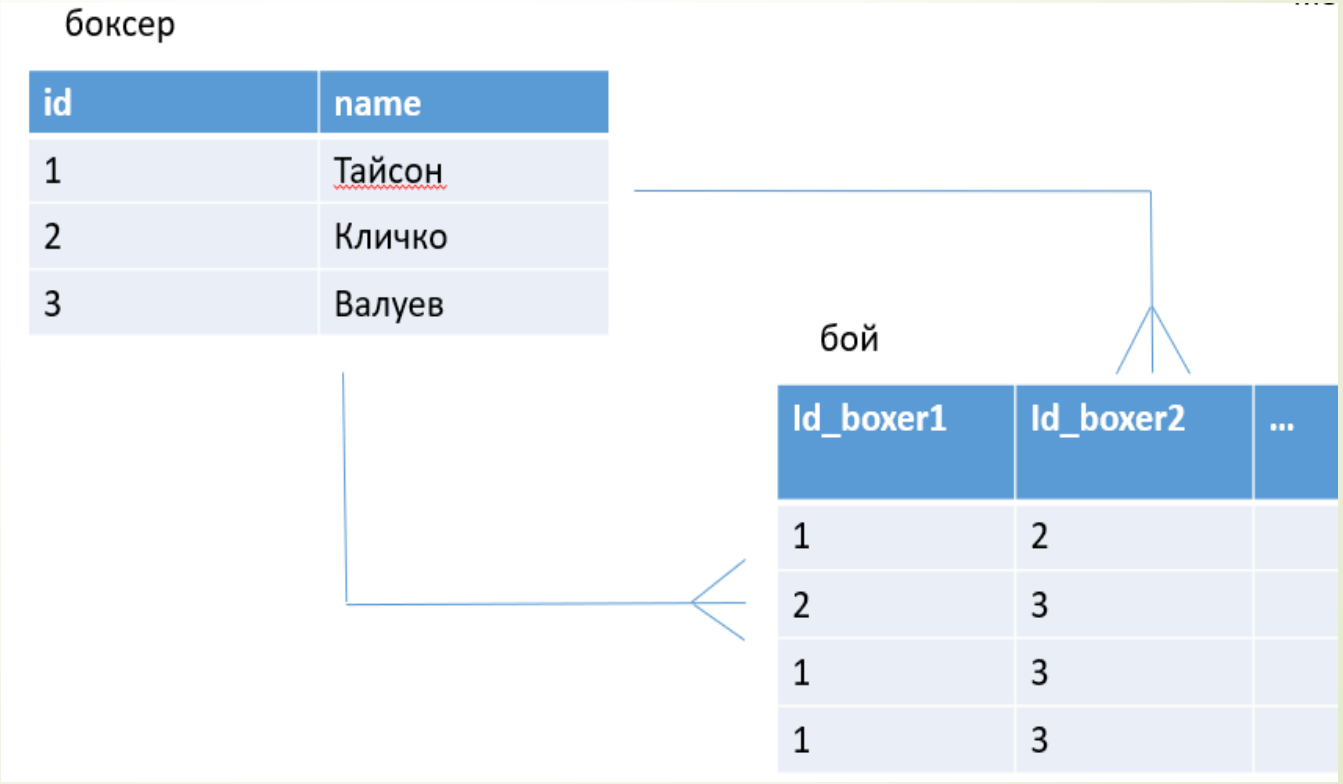
- Цель: Гомоморфное отражение потенциальных и реальных связей в базе данных.
- Решаемая проблема заключается в необходимости отражения как возможной связи между сущностями: двигатель определенной марки может быть установлен на определенную модель автомобиля, так и тот факт, что конкретный двигатель установлен в конкретный автомобиль.
- Обратим внимание, что связь автомобиля проведена через промежуточную сущность непосредственно к сущности, отражающей возможность установки – здесь фиксируется и конкретная модель двигателя в том числе.

Пример паттерна «Модель-реализация»



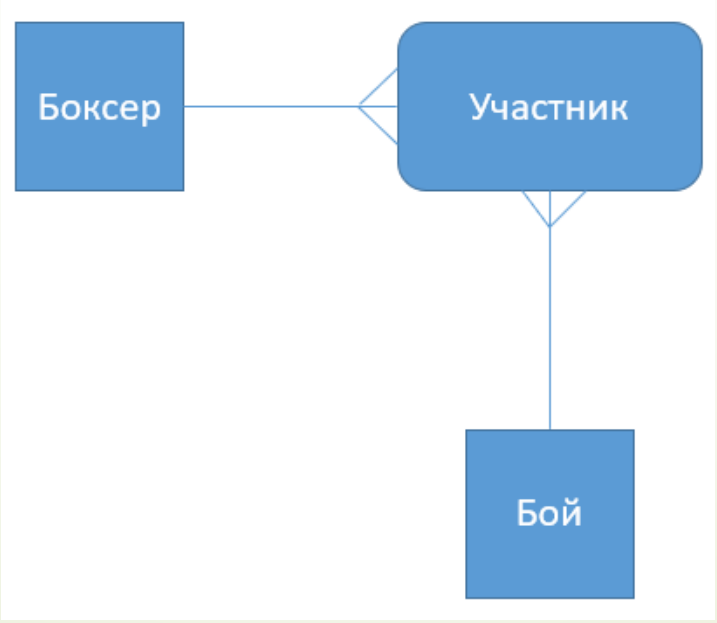
Пример связи графа

- 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.



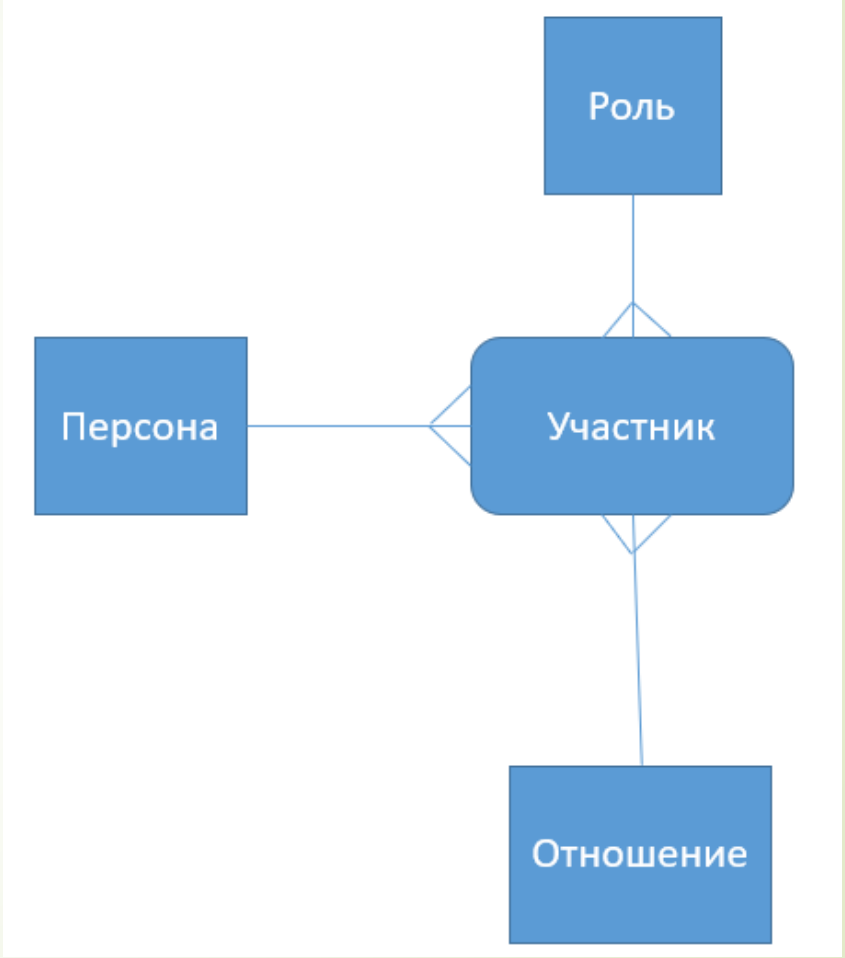
Пример связи графа

- 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:



Пример связи графа

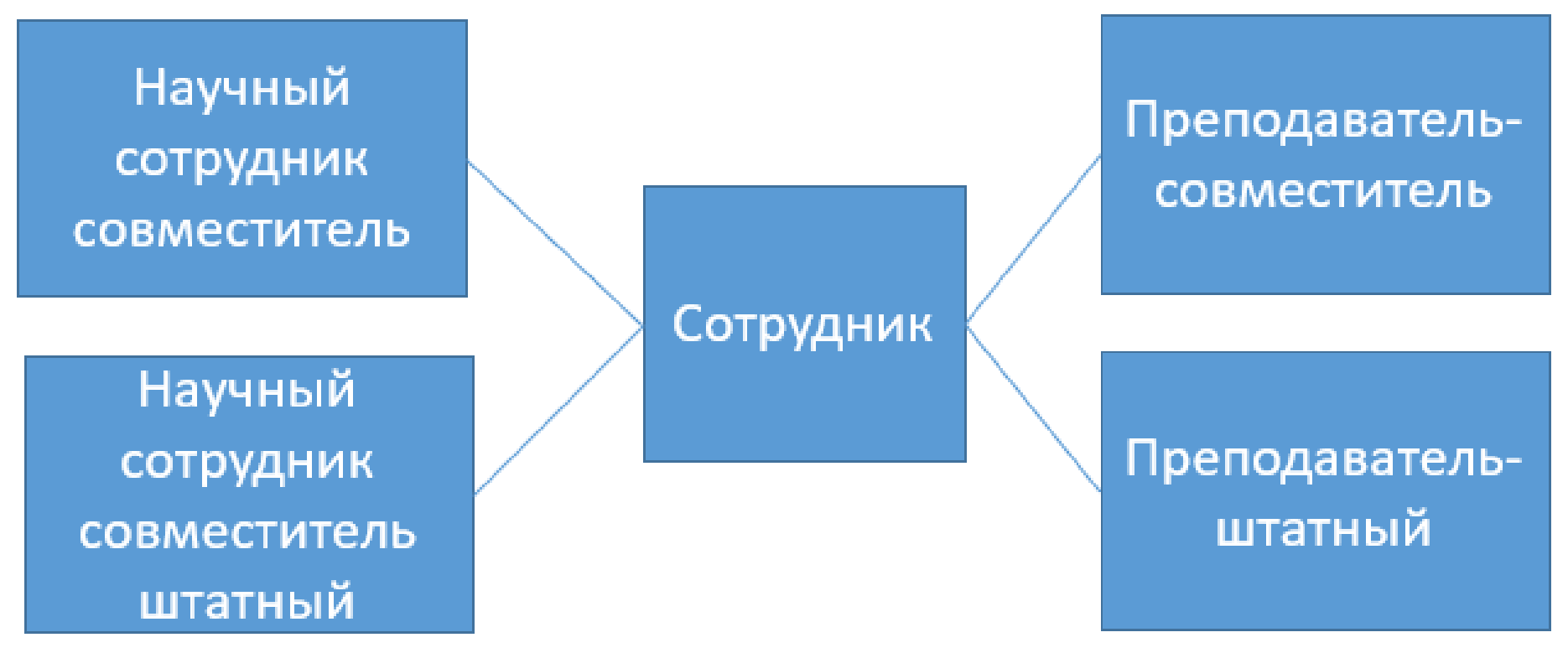
- Теперь имеется возможность легкого выбора участников одним SQL-запросом, при этом можно также легко вводить дополнительные элементы в модель.
- На рисунке показана обобщенная схема.
- Такое решение позволит легко модифицировать модель, добавляя новые роли для участников отношений.



Пример антипаттерна. 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 разницу 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.

Пример антипаттерна. Мозаичное наследование

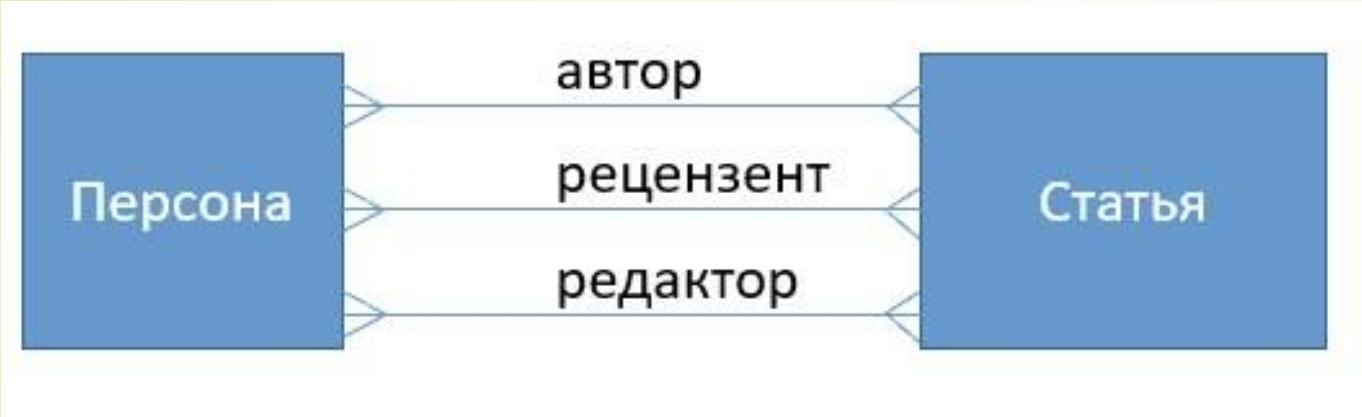


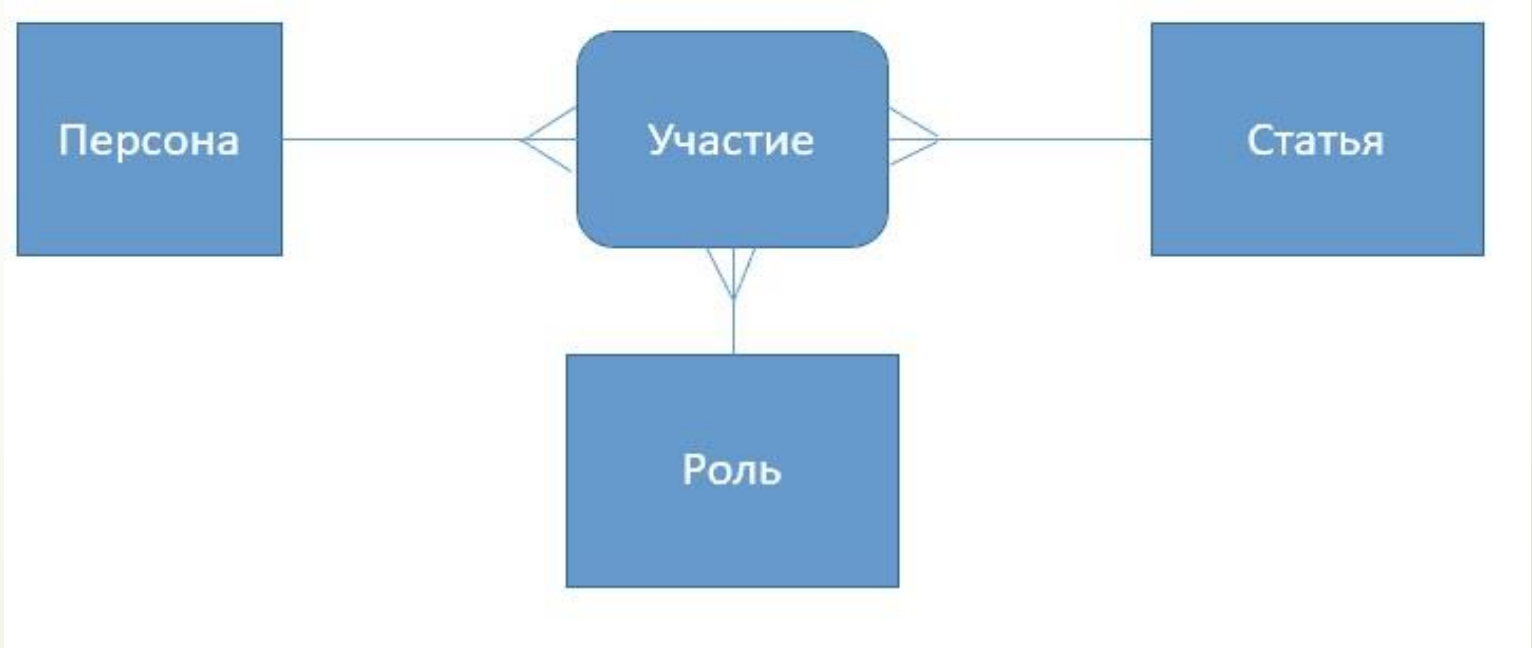
Исходная схема

Пример антипаттерна. Мозаичное наследование

Адекватная схема

Антипаттерн – параллельные связи





Концепция 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

- 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.

- 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.

Тема окончена.

- Вопросы?
- Комментарии?
- Замечания?