Методология проектирования и управления информационными системами

Оглавление

Введение. О курсе

Реализуемые индикаторы компетенции

ОПК-7. Способен использовать методы научных исследований и математического моделирования в области проектирования и управления информационными системами	
ОПК-7.1 Использует методы научных исследований в области проектирования и управления информационными системами	ОПК-7.2 Проектирует и управляет информационными системами, в том числе с применением методов математического моделирования
Знания: • Актуальные проблемы разработки сложных программных систем. Эволюция моделей жизненного цикла информационных систем. • Онтологический подход концептуального моделирования предметной области. • Теория моделирования систем из объектов • Парадигмы программирования • Основы управления ИТ-инфраструктурой (информационный сервис, модели управления информационными системами (ITSM), библиотеки ITIL) Умения: • Сравнивать процессы проектирования и управления, принятые в различных парадигмах программирования Навыки: • Обоснования выбора модели жизненного цикла ИС	Знания: • Методологии ведения программных проектов: структурное, функциональное, объектно-ориентированное и унифицированное моделирование Умения: • Формировать и анализировать модели представления знаний • Применять методы многомерного анализа данных. Навыки: • Проектирования процессов и практик методологии Rapid Application Development, Unified Process. OpenUP

Цель дисциплины

- формирование у студентов теоретических знаний и практических навыков в области методологии процессов проектирования и управления информационными системами, адаптации информационных систем и технологий

Задачи дисциплины:

- Сформировать знания о государственных и международных стандартах в области создания, документирования, эксплуатации и сопровождения ИС; о процессах, стадиях и этапах жизненного цикла информационных систем и их содержании; о методах, технологиях и средствах создания и адаптации информационных систем; знания о методологии проектирования и управления информационными системами в парадигмах программирования, методологии моделирования информационных процессов, о развитии методологий проектирования и управления информационными системами с точки зрения различных научных подходов и технологических решений
- развивать и закрепить умения проводить анализ предметной области, выявлять информационные потребности и разрабатывать требования к ИС; применять современные методы и инструментальные средства прикладной информатики для автоматизации и информатизации решения прикладных задач различных классов и создания ИС; проводить маркетинговый анализ ИКТ и вычислительного оборудования для рационального выбора инструментария автоматизации и информатизации прикладных задач; организовывать работы по моделированию прикладных ИС и реинжинирингу прикладных и информационных процессов предприятия и организации; использовать информационные сервисы для автоматизации прикладных и информационных процессов; выбирать методологию и технологию проектирования ИС с учетом проектных рисков; разрабатывать программные приложения;
- сформировать профессиональные навыки работы с инструментальными средствами моделирования предметной области, прикладных и информационных процессов, навыки использования современных проектных систем для автоматизации разработки ИС;
- сформировать представление о перспективных направлениях развития методологии и технологий проектирования и управления информационных систем

О курсе

- 5 зет = 180 ч
- 18 ч = 9 лекций (6 тем)

- 16 ч = 8 практик 40
- 1 контрольная работа 10
- 1 индивидуальное задание 10
- Бонусные баллы 10
- Экзамен (комиссия) 40

Темы лекций

1. Коэволюция информационных технологий и информационных систем
2. Методологии моделирования информационных процессов: структурное, функциональное, объектно-ориентированное и унифицированное моделирование
3. Методы математического моделирования в области проектирования и управления информационными системами
4. Проектирование структур баз данных и баз знаний
5. Методологии проектирования и управления информационными системами в парадигмах программирования
6. Развитие методологий проектирования и управления информационными системами с точки зрения различных научных подходов и технологических решений

Тема 1. Коэволюция информационных технологий и информационных систем

Под термином «информационные технологии» обычно понимают цифровые компьютерные технологии, поэтому нередко утверждается, что их развитие началось в середине прошлого века. Но это не вполне верно. ИТ — это технологии, обеспечивающие сбор, хранение и обработку информации. Следовательно, к ИТ относятся даже устная речь, счет и письменность, а их история и история ИС насчитывает тысячи лет.

Однако Федеральный закон от 27 июля 2006 г. № 149-ФЗ «Об информации...» дает вполне конкретное определение ИС (см п. 1.1.2); такие понятия, как «информационная система», «компьютерная система», «вычислительная система>> и <<автоматизированная система» практически являются синонимами.

Актуальные проблемы разработки программного обеспечения

- Возрастающая сложность ПрО и ПП, как следствие – высокие требования к квалификации разработчиков и экспертов ПрО
- Необходимость наследования ранее созданного ПО и «старых» форматов данных
- Сокращение времени на разработку - «сейчас или никогда»
- Статистика оптимиста: 40% неудач

Развитие ИС неразрывно связано с развитием вычислительной техники. Первым этапом развития компьютерных ИС принято считать середину XX в. Они реализовывались на основе электромеханических счетных машин, позже — на ранних ЭВМ. Данные системы позволяли обрабатывать бухгалтерскую информацию и, отчасти, автоматизировать подготовку типовых документов (накладных, счетов, платежных ведомостей и т.д.). ИС первого поколения часто называли системами обработки данных.

В конце 1960-х гг. в результате объективной необходимости перехода от кустарных способов к индустриальным способам создания программного обеспечения появилась новая научная дисциплина - программная инженерия (software engineering)

В 1960-е гг., с развитием технических средств и возможностей программирования, совершенствовались и усложнялись ИС. В эти годы были созданы первые прототипы систем поддержки принятия решений: системы, по заранее подготовленным формам составляющие отчеты для управленцев высшего звена. Недостатками таких систем являлось то, что для обеспечения их функционирования было необходимо наличие квалифицированных посредников (программистов, техников) между конечным пользователем и информационной системой, а также их узкая специализация и большая трудоемкость любых модификаций.

В 1970-е гг. появились первые микропроцессорные ЭВМ, начали

использовать дисплеи, были разработаны визуальные интерфейсы, программное обеспечение стало более дружественным к пользователям.

В середине 1970-х гг. появились первые персональные ЭВМ. В этих условиях пользователи получили возможность работать с ИТ без участия посредников. В этот период были созданы первые полноценные СППР, помогающие управленческому персоналу на всех стадиях принятия решения.

1970-е и 1980-е гг. - систематизация и стандартизация создания программного обеспечения (на основе структурного подхода) и с начала 1990-х гг. - переход к сборочному, индустриальному способу создания программного обеспечения (на основе объектно-ориентированного подхода).

В процессе расширения сферы использования ПК и ИС на их основе ИТ-специалисты столкнулись с неожиданной проблемой. Несмотря на дружественный интерфейс, освоение пользователями персональных компьютеров шло с большим трудом, потому внедрение новых технологий шло медленно.

Ситуация изменилась только к концу 1980-х гг., когда начался массовый переход на ПК. Можно сказать, компьютеры стали модны, и потому специалисты, не имеющие ИТ-образования, охотно стали использовать в работе новые технологии. На протяжении 1980-1990-х гг. непрерывно упрощался интерфейс, происходило совершенствование средств представления информации, манипуляторов, а также развивались системы справочной информации по программам, встроенные помощники, облегчавшие работу с приложениями.

1990-е гг. ознаменовали первую реальную попытку превратить разработку программного обеспечения в инженерную дисциплину с помощью концепций CBSE (component-based software engineering - компонентная разработка программного обеспечения) и COTS (commercial off-the-shelf- готовые коммерчески доступные компоненты).

С распространением ПК усилилась децентрализация информационных систем, потребовалось создание новых технологий — сетевой обработки данных. Создавались первые корпоративные локальные сети, начали развиваться сетевые ресурсы — сервера файловые, телекоммуникационные, печати. Появились доступные системы управления базами данных, а также приложения. Активно развиваются технологии баз данных и систем управления ими. Разработка ИС и программ становится более доступной, появляются средства автоматизации разработки программ и БД (CASE1технологии.

CASE (анг л. Computer-Aided Software Engineering) — набор инструментов и методов программной инженерии для проектирования ПО, обеспечивающий высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов).

ИС стали использоваться на всех уровнях иерархии управления.

Растет производительность компьютеров и компьютерных сетей, оперативность получения и обработки информации, а соответственно и скорость бизнеспроцессов.

Распространение ПК, децентрализация, а также упрощение средств разработки программного обеспечения, привели к появлению так называемой лоскутной автоматизации, когда автоматизируются отдельные конкретные процессы на каждом конкретном рабочем месте, в том числе и самим пользователем. Это приводит к неоднородности информационной системы предприятия, и, по мере ее развития, приводит к проблемам совместимости отдельных элементов системы. Такая практика явилась препятствием в реализации одного из важнейших свойств ИС — интегрируемости.

ИС, создаваемые с конца 1990-х гг. и до настоящего времени характеризуются еще большей децентрализацией, повышением характеристик компьютеров, увеличением массивов хранимых и передаваемых данных, существенным повышением требований к производительности как ИС в целом, так и ее элементов. Развиваются новые системы БД, рассчитанные на хранение не только символов и чисел, но также на мультимедийное содержание, а также новые архитектуры, принципы организации. Получают распространение распределенные БД.

Кроме того, развиваются системы обработки данных, как, например, OLAP, обрабатывающие данные не по запросу и заданной форме, а по мере их поступления, по заложенным в систему алгоритмам, предусматривающим связи между данными. Таким образом, при запросе от пользователя представляется уже готовый вариант БД, в нужном преломлении. Такого рода системы требуют огромных вычислительных мощностей и значительного объема памяти, но позволяют повысить эффективность нахождения решений.

Жизненный цикл ПО

- Жизненный цикл (ЖЦ) программного обеспечения определяется как период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации.
- ISO/IEC 12207: 1995 «Information Technology - Software Life Cycle Processes» (ГОСТ Р ИСО/ МЭК 12207-99 введен в действие в июле 2000 г.)
- BS ISO/IEC 122007:2008 Systemand software engineering - Software lifecycle processes (ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия).
- Модель жизненного цикла ПО - это структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении ЖЦ. Модель ЖЦ ПО включает в себя стадии, результаты выполнения работ на каждой стадии,

ключевые события - точки завершения работ и принятия решений.

Тема 2. Методологии моделирования информационных процессов: структурное, функциональное, объектно-ориентированное и унифицированное моделирование

Методы разработки программного обеспечения делятся на четыре основные группы:

- метод структурного анализа и проектирования;
- метод потоков данных;
- объектно-ориентированные методы;
- методы унифицированного моделирования.

Структурный подход основан на алгоритмической декомпозиции. В методе потоков данных программная система рассматривается как преобразователь входных потоков в выходные. В основе объектно-ориентированного проектирования (ООП) лежит представление о том, что программную систему необходимо проектировать как совокупность взаимодействующих друг с другом объектов, рассматривая каждый объект как экземпляр определенного класса, причем классы образуют иерархию.

Все упомянутые методы основаны на идее создания моделей системы, которые можно представить графически, и на использовании этих моделей в качестве спецификации системы или ее структуры.

В *структурном подходе* (функционально-модульном) используется принцип функциональной декомпозиции, при которой структура системы описывается в терминах иерархии ее функций и передачи информации между отдельными функциональными элементами.

В период появления структурного подхода программные продукты отличались *процедурным* характером - запрограммированная процедура выполняла свою задачу последовательным и предсказуемым образом, после чего завершалась. *Структурный подход к разработке* успешно использовался для производства подобных систем.

Структурная методология предоставляла в распоряжение разработчиков строгие формализованные методы описания информационных систем и принимаемых технических решений. Она основывалась на наглядной графической технике: для описания проекта использовались различного рода диаграммы и схемы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако широкое применение этой методологии и следование ее рекомендациям при разработке конкретных проектов встречалось достаточно редко, поскольку ее практически невозможно реализовать на должном уровне ручным неавтоматизированным способом. Также очень трудно разработать вручную и графически представить строгие формальные спецификации системы, проверить их полноту и непротиворечивость и, тем более,

изменить. Если все же удается создать строгую систему проектных документов, то ее переработка при появлении серьезных изменений практически неосуществима.

Если участники проекта пытались прибегнуть к ручной разработке, то перед ними возникали следующие проблемы:

- неадекватная спецификация требований;
- неспособность обнаруживать ошибки в проектных решениях;
- низкое качество документации, снижающее эксплуатационные качества;
- затяжной цикл и неудовлетворительные результаты тестирования.

Отсутствие специализированных программно-технологических средств для разработки проектов, в частности, основанных на информатизации, затрудняло для проектировщиков информационных систем использование компьютерных технологий для повышения качества и производительности своей работы.

Структурный подход к анализу и проектированию отличается следующими особенностями, которые плохо увязываются с современными методами конструирования программного обеспечения.

- Этот подход скорее является последовательным и трансформационным, чем итеративным подходом с наращиванием возможностей, что осложняет реализацию непрерывного процесса разработки, осуществляемого посредством итеративной детализации и пошаговой поставки ПО с наращенными возможностями.
- Структурный подход предопределяет получение «негибких» решений, которые трудно масштабировать и расширять в дальнейшем.
- Подход предполагает разработку «с чистого листа» и не поддерживает повторное использование уже существующих компонент.

Трансформационный характер структурного подхода является источником повышенного риска неправильно истолковать исходные требования к системе в процессе разработки. Это связано с тем, что осуществляется постепенная замена декларативной семантики моделей анализа на процедурные решения для проектных моделей и программного кода.

Тем не менее, структурный подход по-прежнему сохраняет свою значимость и достаточно широко используется на практике, особенно в тех проектах сложной системы, в которых невозможно обойтись только одним способом декомпозиции.

Объектно-ориентированный подход. На основе структурного подхода сформировалась концепция объектно-ориентированного метода (программирования, анализа, проектирования, баз данных), - фактически целая философия разработки систем и представления

знаний на базе мощного подхода [5]. Основное отличие объектноориентированного анализа и проектирования от структурного состоит в декомпозиции проблемы на понятия (объекты), а не на функции.

Исторически развитие этой области началось с объектноориентированного программирования. В 1980-1990-х гг. появились объектно-ориентированные методы разработки программного обеспечения, предложенные Гради Бучем (Booch), Джеймсом Рамбо (OMT - *Object Modeling Technique)* и Айваром Якобсоном (OOSE - *Object Oriented Software Engineering).*

Несмотря на явное преимущество объектно-ориентированных технологий разработки программного обеспечения перед структурными, их распространение было ограниченным, поскольку ни один из методов не давал единой и цельной объектной модели системы. Кроме того, широкому распространению объектно-ориентированных методов при разработке программного обеспечения мешало отсутствие единого стандарта языка объектно-ориентированного моделирования.

В течение 1994-1996 гг. упомянутые разработчики объединили свои усилия под эгидой *Rational Software Corporation* для создания единого языка моделирования, который воплотил бы все существенные и успешные разработки в данной области и стал бы стандартом языка объектно-ориентированного моделирования.

Грандиозный труд, в котором наряду с *Rational* участвовали представители таких крупных компаний, как *Microsoft, IBM, Hewlett-Packard, Oracle, Platinum Technology* и нескольких сотен других, завершился созданием в январе 1997 года версии 1.0 унифицированного языка моделирования UML *(Unified Modeling Language)* [9].

Объектно-ориентированный метод использует объектную декомпозицию. При этом структура системы описывается в терминах объектов и связей между ними, а поведение системы - в терминах обмена сообщениями между объектами.

Гради Буч сформулировал главное достоинство объектно-ориентированного подхода (ООП) следующим образом: объектно-ориентированные системы более открыты и легче поддаются внесению изменений, поскольку их конструкция базируется на устойчивых формах. Это дает возможность системе развиваться постепенно и не приводит к полной ее переработке даже в случае существенных изменений исходных требований.

Буч отметил также ряд преимуществ ООП.

- Объектная декомпозиция дает возможность создавать программные системы меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование ООП существенно повышает уровень унификации разработки и пригодность для повторного использования не только программного обеспечения,

но и проектов, что, в конце концов, ведет к сборочному созданию. Системы зачастую получаются более компактными, чем их не объектно-ориентированные эквиваленты, что означает не только уменьшение объема программного кода, но и удешевление проекта за счет использования предыдущих разработок.

- Объектная декомпозиция уменьшает риск создания сложных программных систем, так как она предполагает эволюционный путь развития системы на базе относительно небольших подсистем. Процесс интеграции системы растягивается на все время разработки, а не превращается в единовременное событие.
- Объектная модель вполне естественна, поскольку в первую очередь ориентирована на человеческое восприятие мира, а не на компьютерную реализацию.
- Объектная модель позволяет в полной мере использовать выразительные возможности объектных и объектно-ориентированных языков программирования.

По сути, все современные приложения являются объектноориентированными, а некоторые языки программирования (например, *Java)* предполагают использование объектно-ориентированных структур.

В соответствии с объектно-ориентированной парадигмой все приложения делятся на элементы кода (объекты), относительно независимые друг от друга. Готовое приложение можно затем создать, сложив эти объекты вместе. При этом программные объекты выполняются случайным, непредсказуемым образом, и программа не завершается до тех пор, пока пользователь не прекратит ее выполнение.

Одной из причин популярности объектного подхода является возможность обеспечения функциональных требований для *приложений с элементами мультимедиа,* к которым относятся и компьютерные обучающие системы, путем реализации концепции «помещения объектов в оболочку».

Объектный подход к разработке систем следует *итеративному* процессу *с наращиванием возможностей.* Единая модель (и один проектный документ) конкретизируется на этапах анализа, проектирования и реализации - в результате успешных итераций добавляются новые детали, при необходимости вводятся изменения и усовершенствования, а выпуски программных модулей с наращенными возможностями поддерживают эволюцию требований к системе и обеспечивают обратную связь, необходимую для продолжения разработки модулей.

Разработка с помощью последовательной детализации становится возможной благодаря тому, что все создаваемые в ходе разработки модели (анализа, проектирования и реализации) обладают семантическим богатством и базируются на одном и том же языке, так

как базовый словарь этих моделей существенно не отличается (классы, атрибуты, методы, наследование, полиморфизм и т. д.).

Объектный подход устраняет большинство из наиболее значительных недостатков структурного подхода, однако служит источником некоторых новых нижеперечисленных проблем.

Этап анализа проводится на еще более высоком уровне абстракции, и если серверная часть решения по реализации предполагает использование реляционной базы данных, то семантический разрыв между концепцией и ее реализацией может быть значительным.

Хотя анализ и проектирование могут проводиться итеративно с наращиванием возможностей, в итоге разработка достигает этапа реализации, которая требует трансформации решения применительно к реляционной базе данных. Если в качестве платформы реализации используется объектная или объектно-реляционная база данных, трансформация проекта проходит значительно легче.

Управление проектом сложно осуществлять. Менеджеры измеряют степень продвижения разработки с помощью четко определенной декомпозиции работ, элементов комплекта поставки и ключевых этапов. При объектной разработке с помощью «детализации» не существует четких границ между этапами, а проектная документация непрерывно развивается.

Приемлемое решение в такой ситуации заключается в делении проекта на небольшие модули и управлении ходом разработки за счет частого выпуска выполняемых версий этих модулей (некоторые из этих выпусков могут быть для внутреннего применения, а другие - поставляться заказчику).

Проблема использования объектного подхода связана с возрастающей сложностью решения, что, в свою очередь, сказывается на таких характеристиках программного обеспечения, как приспособленность к сопровождению и масштабируемость.

Как отмечает известный специалист в области программной инженерии А. Вендров, объектно-ориентированный подход не дает немедленной отдачи. Эффект от его применения начинает сказываться после разработки двух-трех проектов и накопления повторно используемых компонентов, отражающих типовые проектные решения в данной области. Переход организации на объектно-ориентированную технологию - это смена мировоззрения, а не просто изучение новых CASEсредств и языков программирования.

Однако, сложности, связанные с объектным подходом, преодолимы и не должны повлиять на прерогативу их применения для разработки программных систем по сравнению с процедурным стилем пакетных приложений на языках программирования типа COBOL.

На сегодняшний день объектно-ориентированный подход - единственный известный метод, позволяющий осуществить разработку подобных систем - нового управляемого событиями программного

обеспечения, отличающегося высоким уровнем интерактивности.

Визуальное унифицированное моделирование систем

- Исходными принципами программной инженерии, лежащими в основе современных технологий разработки, являются итеративность, модульная архитектура системы и визуальное моделирование.
- В соответствии с принципами программной инженерии один из основных этапов проектирования программного обеспечения - визуальное моделирование систем.
- Модель программной системы - это формальное описание системы, в котором выделены основные объекты, составляющие систему, и отношения между этими объектами. В описаниях представляется замысел, или смысл системы. Модель всегда представляет собой некоторый уровень абстракции, охватывая лишь существенные черты.
- Графические языки моделирования уже продолжительное время широко используются в программной индустрии. Основная причина их появления состоит в том, что языки программирования не обеспечивают нужный уровень абстракции, способный облегчить процесс проектирования систем.

Принципы моделирования

Длительный опыт использования моделирования во всех инженерных дисциплинах позволил сформулировать основные принципы создания моделей программных систем.

Первый принцип: выбор модели оказывает определяющее влияние на подход к решению проблемы и на то, как будет выглядеть это решение.

Если программная система должна работать с большими базами данных или производить сложные математические расчеты, то основное внимание будет уделяться моделям «сущность-связь», где поведение инкапсулировано в триггерах и хранимых процедурах.

Структурный аналитик, как правило, создает модель, в центре которой находятся алгоритмы и передача данных от одного процесса к другому. Результатом труда разработчика, использующего объектно-ориентированную технологию, будет модель системы, архитектура которой основана на множестве классов и образцах взаимодействия, определяющих, как эти классы действуют совместно.

Второй принцип: каждая модель может быть воплощена с различной степенью абстракции.

Например, при моделировании компьютерных обучающих систем на этапе специфицирования требований целесообразно представить простую, быстро созданную и легко модифицируемую модель пользовательского интерфейса.

На этапе специфицирования межсистемных интерфейсов

необходимо создавать их модели с высоким уровнем детализации. Для аналитика или конечного пользователя наибольший интерес представляют модели, отвечающие на вопрос «что», а для разработчика на вопрос «как».

В общем случае лучшей моделью будет та, которая представляет уровень детализации, соответствующий категории исследователя (или пользователя) и цели исследования (или использования) модели.

Третий принцип: лучшие модели те, что ближе к реальному отражению системы.

Поскольку модель всегда упрощает реальность, то задача состоит в том, чтобы это упрощение не повлекло за собой существенные потери при реализации системы. Различие модели и реальной системы должно быть проанализировано и учтено в процессе разработки.

Рассматриваемый принцип легче реализуется при объектно-ориентированном подходе, чем при структурном. Это следует из того, что «ахиллесовой пятой» структурного анализа является несоответствие модели, принятой в нем, и модели проекта.

Если этот разрыв не будет устранен, то поведение созданной системы с течением времени начнет все больше отличаться от задуманного. При объектно-ориентированном под- ходе можно объединить все почти независимые представления системы в единое семантическое целое.

Принцип многомодельности

Важным принципом моделирования сложных программных систем является принцип многомодельности. Он формулируется следующим образом.

Нельзя ограничиваться созданием только одной модели. Наилучший подход при разработке любой нетривиальной системы использовать совокупность нескольких моделей, почти независимых друг от друга.

Определение «почти независимые» означает, что модели могут создаваться и изучаться по отдельности, но вместе с тем остаются взаимосвязанными.

Принцип многомодельности применительно к методологии объектно-ориентированного анализа и проектирования может быть реализован моделями, которые показаны на рис..

Эти модели считаются главными в объектно-ориентированном подходе. В совокупности они семантически достаточно информативны и универсальны, чтобы разработчик мог реализовать все стратегические и тактические решения, которые он должен принять при анализе системы и формировании ее архитектуры. Кроме того, эти модели достаточно полны, чтобы служить техническим проектом реализации практически на любом объектно-ориентированном языке.

Создавая визуальную модель системы, можно показать ее работу на различных уровнях: моделировать взаимодействие между

пользователями и системой, между объектами внутри системы и даже между различными системами.

Управление моделями облегчается средствами визуального моделирования, так как возможно предъявление модели с различной степенью полноты. Визуальное моделирование способствует поддержанию непротиворечивости артефактов системы: требований, проектов и реализаций, что является одной из сложнейших задач для команды разработчиков.

Модели используются практически каждым членом группы разработчиков, начиная с руководителя (координатора) проекта, работа которого связана с системой в целом, и заканчивая программистом, отвечающим за программную реализацию компонентов системы.

Созданные визуальные модели представляют всем заинтересованным сторонам, которые могут извлечь из них ценную информацию. Например, глядя на модель, пользователи визуализируют свое взаимодействие с системой. Аналитики увидят взаимодействие между объектами модели. Разработчики поймут, какие объекты нужно создать и что эти объекты должны делать. Тестировщики визуализируют взаимодействие между объектами, что позволит им построить тесты. Менеджеры увидят как всю систему в целом, так и взаимодействие ее частей. Руководители информационной службы, глядя на высокоуровневые модели, поймут, как взаимодействуют друг с другом системы в их организации.

Графические нотации моделирования

Система обозначений, или нотация (notation), представляющая собой совокупность графических объектов, используемых в моделях, является синтаксисом языка моделирования. Об обозначениях в разработке программного обеспечения имеется достаточно много литературы. И. Грэхем дал обзор ряда нотаций, специфичных для объектно-ориентированных методов.

По определению Гради Буча, нотация выполняет следующие функции:

- играет роль языка, используемого для описания неочевидных выводов, которые не проистекают непосредственно из кода как такового;
- сообщает семантику всех стратегических и тактических решений;
- обеспечивает форму представления, достаточно конкретную для восприятия человеком, и возможности манипуляции ею с помощью инструментальных средств.

Рассмотрим учебный пример использования графической нотации UML для моделирования абстрактной веб-системы

Диаграмма вариантов использования

Рисунок 1 представляет собой диаграмму вариантов использования. Данная диаграмма содержит трех актеров в лице

пользователя, клиентской и серверной части приложения.

Прецедентами для пользователей являются открытие и закрытие веб-приложения, а также проведение манипуляций над информацией на сервере. Вариант использования ManipulateDate расширяется такими стандартными для информационных систем действиями, как операции записи, обновления, и удаления, которые производятся исходя из намерения пользователя. Операция получения информации ассоциирована с актером клиентского приложения, поскольку именно оно получает и отображает данные, полученные с сервера.

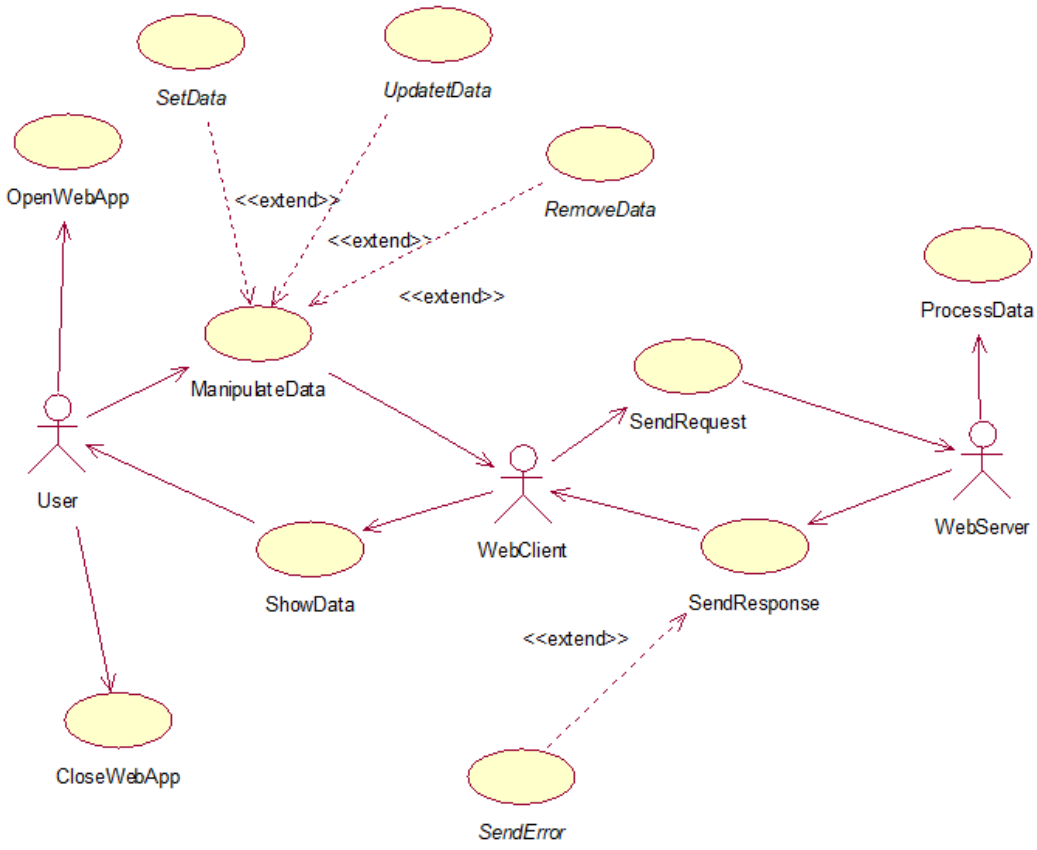


Рисунок 1 – Диаграмма вариантов использования

Помимо отображения информации, вариантом использования клиентского приложения является отправка запросов на сервер.

Актер сервера в свою очередь ассоциирован с прецендентами обработки хранящейся на нем информации и отправки ответа на клиентское приложения.

Вариант использования SendResponse расширяется с помощью прецендента SendError, поскольку в результате пользовательского запроса или внутренней работы сервера может возникнуть ошибка, о которой пользователь должен знать.

Прецеденты, которые расширяют другие прецеденты помечены как абстрактные потому, что они не могут быть самостоятельными и выполняться без варианта использования, в который они включены.

Диаграммы взаимодействия

Рисунок 2 представляет собой диаграмму последовательности, которая содержит поочередные действия над данными в информационной системе.

Все сообщения, между объектами, являются операциями соответствующих классов. Некоторые операции имеют списки аргументов, а также возвращаемые значения. Некоторые же производят какие-либо манипуляции и не возвращают ничего.

Пользователь взаимодействует с объектом браузера, который имеет соответствующий класс и методы открытия, закрытия и ввода адреса.

После ввода адреса веб-приложения веб-браузер открывает графический интерфейс, который имеет соответствующие методы открытия, закрытия, записи, обновления и удаления.

Веб-клиент приложения обращается к серверу через объект HTTP, который обладает методом создания запроса. Далее этот запрос отправляется объекту сервера, где он обрабатывается им.

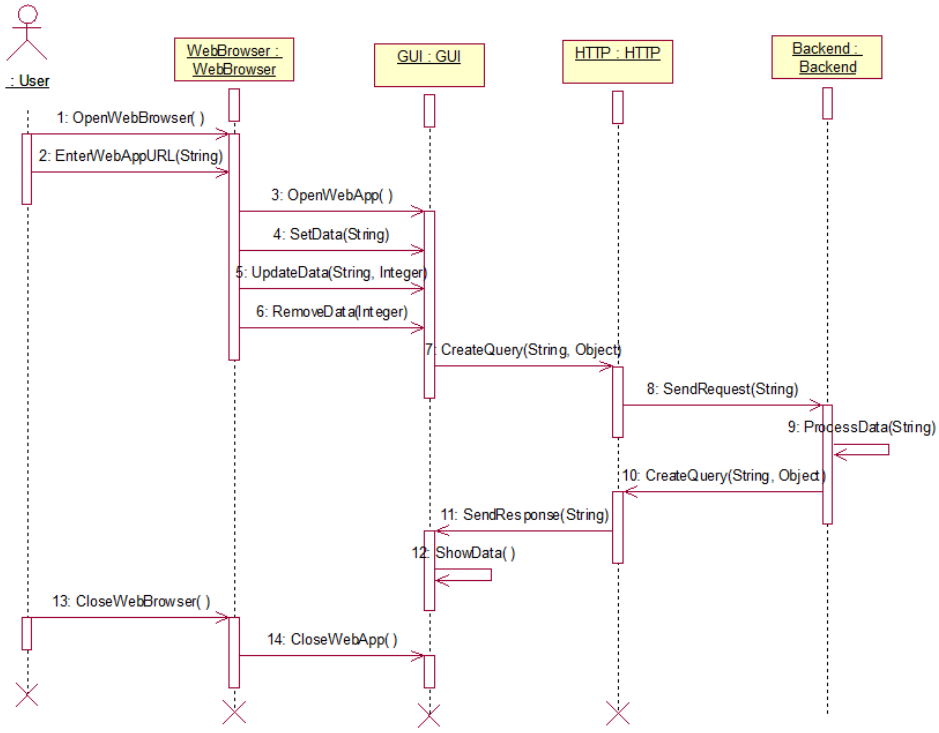


Рисунок 2 – Диаграмма последовательности

Сервер возвращает ответ клиентскому приложению через объект HTTP, затем графический интерфейс отображает полученную информацию.

Рисунок 3 представляет собой диаграмму кооперации, которая отображает не только последовательность действий во времени, но и отношения между объектами. Диаграммы последовательности и кооперации являются взаимозаменяемыми.

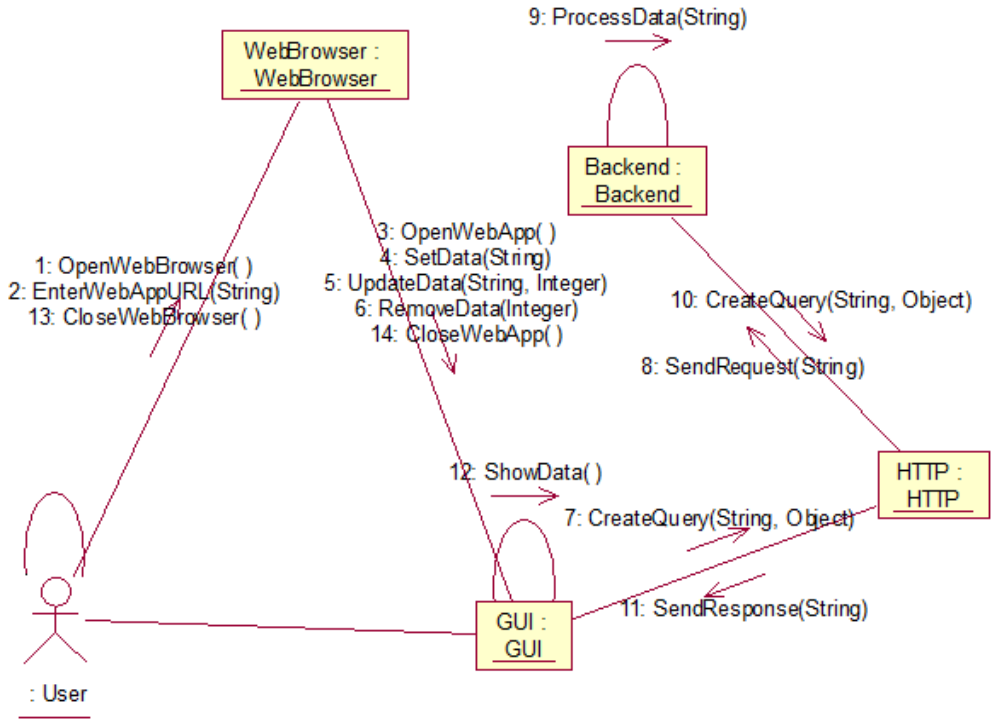


Рисунок 3 – Диаграмма кооперации

Диаграмма классов

Рисунок 4 представляет собой диаграмму классов, на которой изображены четыре класса системы, их стереотипы, атрибуты и методы, а также связи между ними.

Веб-браузер является граничным классом, ведь через него производится взаимодействие системы с внешней средой.

Сервер представляет собой сущность, которая должна хранить информацию после выключения системы. В качестве хранимой информации выступает атрибут с типом Vector.

Классы HTTP и GUI являются классами управления.

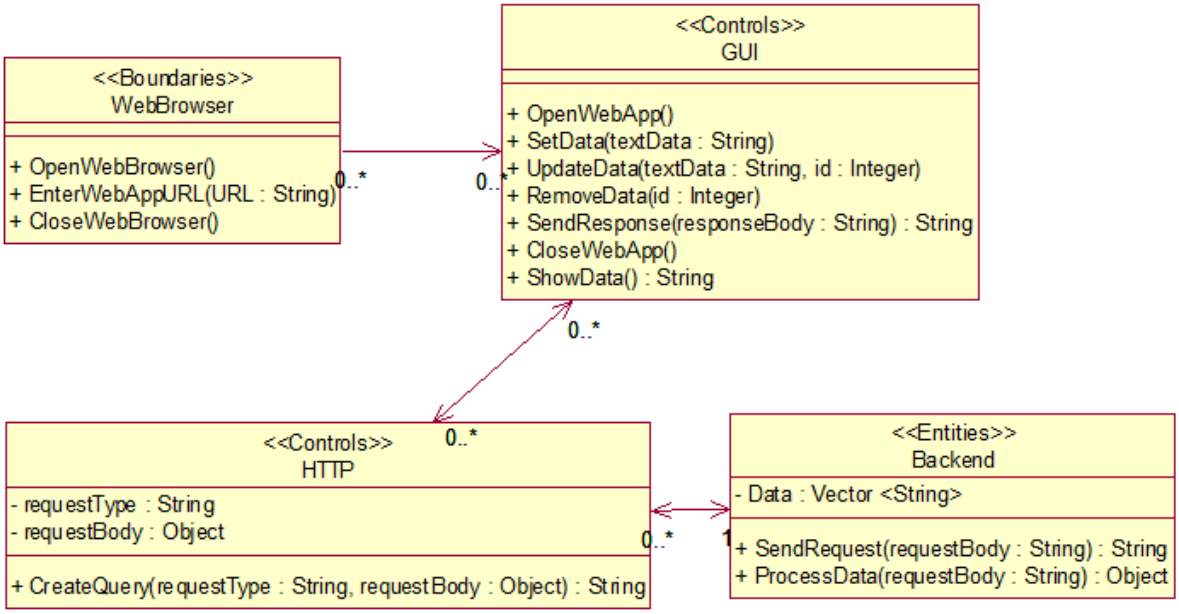


Рисунок 4 – Диаграмма классов

Классы связаны с собой последовательно, при этом классы GUI и Backend связываются двухсторонней связью через HTTP, ведь сообщения приходят как туда, так и обратно.

Значения множественности между браузером и интерфейсом установлено как 0 или много, ведь можно открыть много браузеров и в каждом из них можно открыть много экземпляров веб приложения.

Значение множественности между GUI и HTTP также является связью 0 или много. Много экземпляров интерфейса может посылать и принимать много запросов.

А вот отношение между HTTP и Backend установлено как 0 или много с одной стороны и 0 или 1 с другой. Запросов может быть много, а сервер в данном случае один.

Диаграммы состояний

Рисунок 5 содержит диаграмму состояний для класса WebBrowser.

Браузер может находиться в открытом и закрытом состояниях, чему предшествуют события открытия и закрытия. Помимо этого, после ввода ссылки на ресурс браузер переходит в состояние загрузки, во время которой он производит действия с получаемыми данными. После того, как страница загрузилась, браузер переходит в состояние отображения данных и слежения за обновлениями.

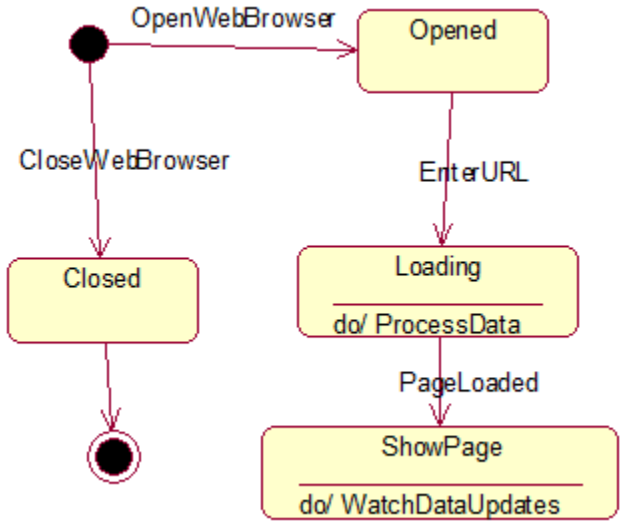


Рисунок 5 – Диаграмма состояний класса WebBrowser

Рисунок 6 представляет диаграмму состояний класса GUI.

Графический интерфейс может быть открытым и закрытым, чему предшествуют соответствующие события. По открытию страницы интерфейс получает начальные данные, а после того, как они получены, переходит в состояние их отображения.

После действия пользователя интерфейс переходит в состояние отправки запроса. После получения ответа клиентская часть приложения переходит в состояние обработки полученных данных. При появлении ответа на запрос GUI определяет его тип и обрабатывает данные, которые затем отображаются пользователю.

Рисунок 7 описывает состояния класса BackEnd.

Сервер может находиться в состоянии ожидания запроса. После получения запроса состояние ожидания меняется на состояние обработки.

В момент принятия запроса сервер определяет его тип и обрабатывает данные. После обработки сервер переходит в состояние отправки ответа. Когда событие DataProcessed поступает на вход состояния отправки ответа, производится определение типа полученных данных, на основе которых формируется ответ.

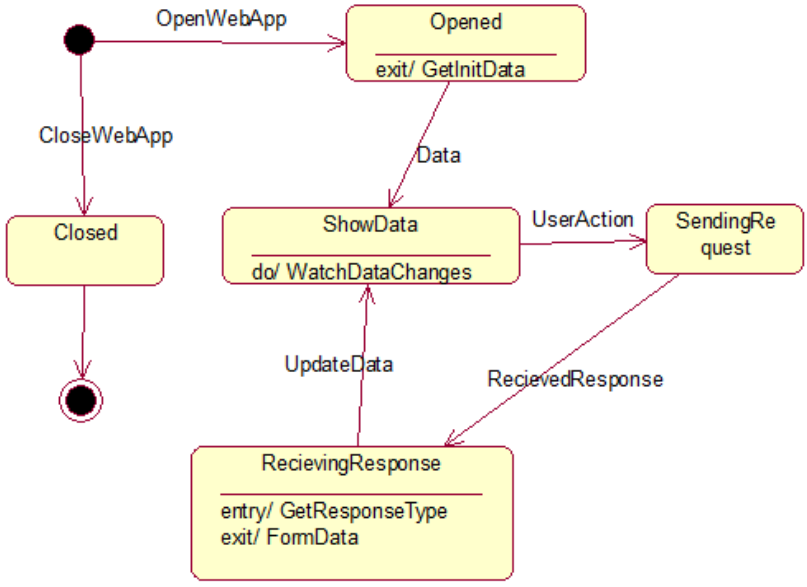


Рисунок 6 – Диаграмма состояний класса GUI

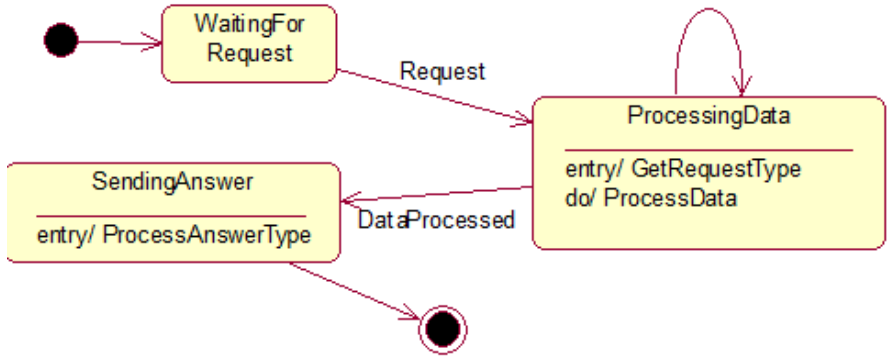


Рисунок 7 – Диаграмма состояний класса Backend

Диаграмма компонентов системы

Рисунок 8 представляет диаграмму компонентов веб-приложения. Для каждого класса были созданы спецификация и тело компонента. Также была произведена привязка компонентов к их классам.

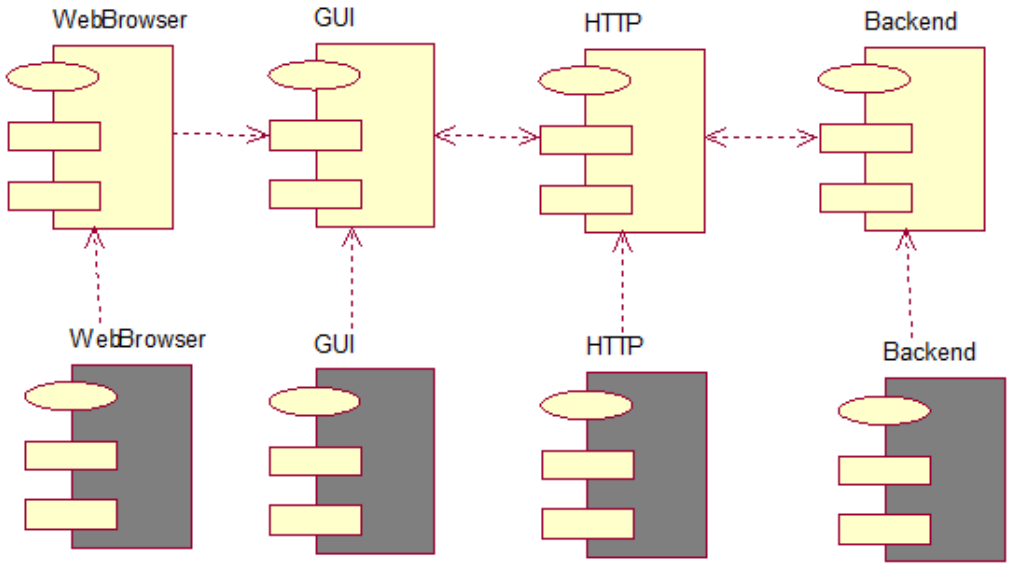


Рисунок 8 – Диаграмма компонентов

Диаграмма размещения

Рисунок 9 отображает физическое размещение веб-приложения.

Для того, чтобы манипулировать данными, пользователю необходимы устройства ввода и вывода, рабочая станция и подключение к серверу. Рабочая станция, роутер и сервер являются вычислительными устройствами, в то время как клавиатура и экран – устройствами периферии.

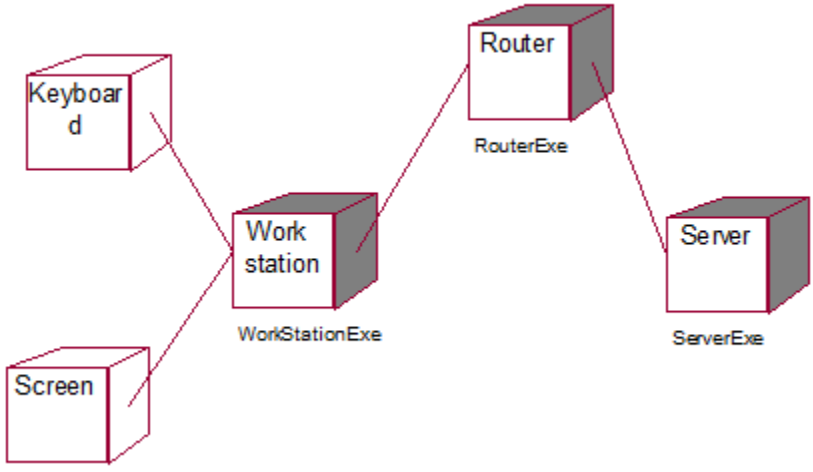


Рисунок 9 – Диаграмма размещения

В приведенном примере был описан процесс проектирование веб-приложения, предназначенного для проведения операций над информацией, хранящейся на сервере, через клиентский графический интерфейс. Были созданы диаграммы вариантов использования, взаимодействия, классов, состояний, компонентов и размещения.

Тема 3. Методы математического моделирования в области проектирования и управления информационными системами

Модель предметной области

Анализом предметной области занимаются системные аналитики или бизнес-аналитики

Результаты этого анализа представляются в виде модели предметной области – набора графических схем и текстовых документов

Предметная область (ПрО) – это совокупность понятий, концептов выделенных объектов и их характеристик

Под системой подразумевается совокупность взаимодействующих компонентов и взаимосвязей между ними

Моделью M некоторой системы S называется информационный объект, который может быть использован для получения ответов на некоторый круг вопросов относительно S

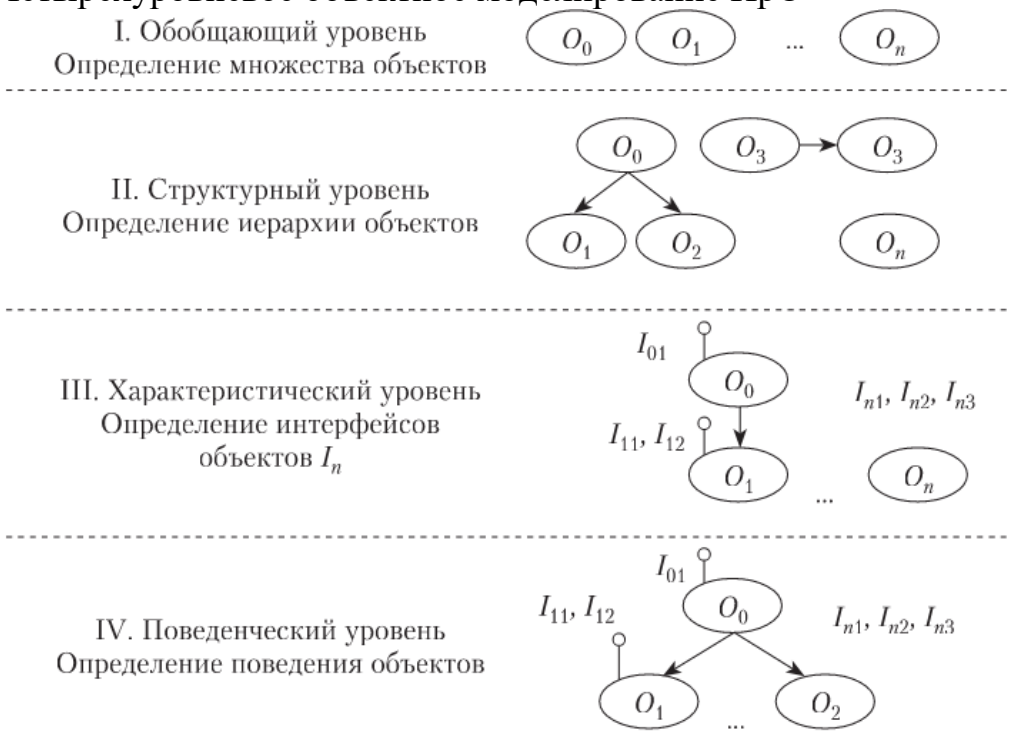
Принципы объектного анализа в моделировании:

- **Принцип всеобщности** означает, что на произвольном шаге объектного анализа все сущности – суть объекты.
  - Аксиома 1. Предметная область, которая моделируется из объектов, сама является объектом.
  - Аксиома 2. Моделируемая предметная область может быть отдельным объектом в составе другой предметной области.
- **Принцип существенности объектных различий**. На произвольном шаге объектного анализа каждый объект является уникальным элементом.
  - Аксиома 3. Каждый объект имеет по крайней мере одно свойство или характеристику, которая задает его уникальную идентификацию во множестве объектов.
- **Принцип объектной упорядоченности.** На произвольном шаге объектного анализа все объекты упорядочены в соответствии с отношениями между объектами.
  - Аксиома 4. Каждый объект имеет при необходимости одно отношение с другим объектом, которое обеспечивает его упорядоченность в рамках этой пары объектов.
- **Принцип целостности объектной модели**. На произвольном шаге объектного анализа совокупность объектов и отношений между ними однозначно определяют объектную модель предметной области для определенного уровня ее абстракции.
  - Аксиома 5. На произвольном шаге объектного анализа объектную модель можно представить в виде ориентированного связного графа, вершинами которого являются объекты, а дугам соответствуют отношения между объектами

Четырехуровневое объектное моделирование ПрО

- **I. Обобщающий уровень** определяет базовые понятия ПрО – объекты без учета их сущности и свойств. Объект задается в виде денотата в соответствии с теорией Фреге в виде <имя объекта> <концепт>.
- **II. Структурный уровень** определяет расположение объектов в структуре модели ПрО, устанавливает упорядочение объектов и представление их структуры в виде графа с операциями над объектами (объединения, пересечения, разности, приложения, симметричной разницы и др.)
- **III. Характеристический уровень** задает общие, специфические свойства и характеристики концептов объектов в логико-алгебраическом представлении объектов в виде графа с характеристиками и свойствами в модели.
- **IV. Поведенческий уровень** определяет поведение и изменение объектов в зависимости от событий, которые они создают при их выполнении.

Четырехуровневое объектное моделирование ПрО



Итоговая формализация модели ПрО

Модель ПрО имеет вид:

$$M_{\text{ПрО}} = (M_{\text{о}}, Mi_{\text{ис}}, M_{\text{ох}}, M_{\text{ПС}}, P, D),$$

где $M_{\text{о}}$ — множество объектов и отношений между ними, заданных в ОМ; $Mi_{\text{ис}}$ — модель интерфейса объектов ОМ; $M_{\text{ох}}$ — множество общих характеристик объектов и внутренних, присущих каждому объекту и обеспечивающих конфигурационную сборку объектов ПрО; $M_{\text{ПС}}$ — модель ПС, которая реализует задачи и функции ПрО; P — множество предикатов с порядком и условиями выполнения объектов, их функциональных и нефункциональных характеристик модели свойств FM и взаимосвязи объектов ПрО, методы которых обеспечивают их программную реализацию в ПС; D — множество данных ПрО, которые необходимы для выполнения отдельных компонентов и ПС и могут сохраняться в БД.

Методы моделирования ПрО

- Процесс построения Модели ПрО носит название **концептуального моделирования**
- Разновидности подходов концептуального моделирования:
  - GDM (Generative Domain Model)
  - FM (Feature Model)
  - MDD (Model Driven Development)
  - MDA (Model Driven Architecture)
  - VDM (Vienna Development Methods)
- Подробнее в курсе лекция будут рассмотрены
  - Унифицированный язык моделирования UML (Unified Modeling Language)
  - Онтологические модели OWL (Web Ontology Language) и ODM (Ontology Definition Metamodel)
  - SOA (Service-Oriented Model) и микросервисная архитектура

Определения онтологии

Слово "онтология" имеет два значения:

- Онтология 1 - философская дисциплина, которая изучает наиболее общие характеристики бытия и сущностей;
- Онтология 2 - это артефакт, структура, описывающая значения элементов некоторой системы.

Настоящий курс посвящен способам разработки и использования в приложениях онтологий как артефактов (Онтология 2).

Неформально онтология представляет собой некоторое описание взгляда на мир применительно к конкретной области интересов. Это описание состоит из терминов и правил использования этих терминов, ограничивающих их значения в рамках конкретной области.

На формальном уровне онтология - это система, состоящая из набора понятий и набора утверждений об этих понятиях, на основе которых можно описывать классы, отношения, функции и индивиды.

Одно из самых известных определений онтологии дал Том Грубер, звучит оно следующим образом: Онтология - это точная спецификация концептуализации.

Концептуализация - это структура реальности, рассматриваемая независимо от словаря предметной области и конкретной ситуации.

Например, если мы рассматриваем простую предметную область, описывающую кубики на столе, то концептуализацией является набор возможных положений кубиков, а не конкретное их расположение в текущий момент времени.

Более поздней модификацией определения Грубера является такое определение: Онтология - это формальная спецификация согласованной концептуализации. Под согласованной концептуализацией подразумевается, что данная концептуализация не есть частное мнение, а является общей для некоторой группы людей.

Сформулировано еще достаточно много разных определений онтологии. Например, Никола Гуарино определяет онтологию следующим образом: Онтология - это формальная теория, ограничивающая возможные концептуализации мира.

Некоторые определения отражают способы, которыми авторы строят и используют онтологии, например: Онтология - это иерархически структурированное множество терминов, описывающих предметную область, которое может быть использовано как исходная структура для базы знаний.

Содержание онтологии

Основными компонентами онтологии могут являться:

- классы (или понятия),
- отношения (или свойства, атрибуты),
- функции,
- аксиомы,
- экземпляры (или индивиды).

Классы или понятия используются в широком смысле. Понятием может быть любая сущность, о которой может быть дана какая-либо информация. Классы - это абстрактные группы, коллекции или наборы объектов. Они могут включать в себя экземпляры, другие классы, либо же сочетания и того, и другого. Классы в онтологиях обычно организованы в таксономию - иерархическую классификацию понятий по отношению включения. Например, классы Мужчина и Женщина являются подклассами класса Человек, который в свою очередь включен в класс Млекопитающие.

Отношения представляют тип взаимодействия между понятиями предметной области. Формально n -арные отношения определяются как подмножество произведения n множеств: $R \subseteq C_1 \times C_2 \times \ldots \times C_n$. Пример бинарного отношения - отношение ЧАСТЬ-ЦЕЛОЕ. Отношения тоже могут быть организованы в таксономию по включению; например, отношения быть_отцом_для и быть_матерью_для на множестве людей

содержатся в отношении быть_родителем_для, которое в свою очередь содержится в отношении быть_предком_для.

Функции - это специальный случай отношений, в которых n -й элемент отношения однозначно определяется n-1 предшествующими элементами. Формально функции определяются следующим образом: $F: C_1 \times C_2 \times ... \times C_{n-1} \rightarrow C_n$. Примерами функциональных отношений являются отношения быть_матерью_для на множестве людей, или цена_подержанного_автомобиля, которая вычисляется в зависимости от модели автомобиля, даты изготовления и пробега.

Аксиомы используются, чтобы записать высказывания, которые всегда истинны. Они могут быть включены в онтологию для разных целей, например, для определения комплексных ограничений на значения атрибутов, аргументы отношений, для проверки корректности информации, описанной в онтологии, или для вывода новой информации.

В качестве примера того, что в рамках онтологий понимается под аксиомами, можно привести следующее положение и его формальную запись на языке исчисления предикатов первого порядка:

Работник, являющийся руководителем проекта, работает в проекте.

Вводятся переменные E (работник) и P (руководитель проекта). Тогда аксиома записывается следующим образом:

Forall (E,P) Employee(E) and Head-Of-Project(E,P)
=> Works-At-Project(E,P)

Цели создания онтологий

В последние годы разработка онтологий - явное формальное описание терминов предметной области и отношений между ними - переходит из мира лабораторий по искусственному интеллекту на рабочие столы экспертов по предметным областям. Во всемирной паутине WWW онтологии стали обычным явлением. Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты (как на сайте Yahoo!), до категоризаций продаваемых товаров и их характеристик (как на сайте Amazon.com). Во многих дисциплинах сейчас разрабатываются стандартные онтологии, которые могут использоваться экспертами по предметным областям для совместного использования и аннотирования информации в своей области.

Например, в области медицины созданы большие стандартные, структурированные словари, такие как SNOMED и семантическая сеть Системы Унифицированного Медицинского Языка (Unified Medical Language System, UMLS). Также появляются обширные общецелевые онтологии. Например, Программа ООН по развитию (the United Nations Development Program) и компания Dun & Bradstreet объединили усилия для разработки онтологии UNSPSC, которая предоставляет терминологию товаров и услуг (unspsc.org).

Онтология определяет общий словарь для ученых, которым нужно совместно использовать информацию в предметной области. Она

включает машинно-интерпретируемые формулировки основных понятий предметной области и отношения между ними.

Почему возникает потребность в разработке онтологии? Вот некоторые причины, которые ниже будут рассмотрены подробнее:

- для совместного использования людьми или программными агентами общего понимания структуры информации;
- для возможности повторного использования знаний в предметной области;
- для того чтобы сделать допущения в предметной области явными;
- для отделения знаний в предметной области от оперативных знаний;
- для анализа знаний в предметной области.

Совместное использование людьми или программными агентами общего понимания структуры информации является одной из наиболее общих целей разработки онтологий. К примеру, пусть несколько различных веб-сайтов содержат информацию по медицине или предоставляют информацию о платных медицинских услугах, оплачиваемых через Интернет. Если эти веб-сайты совместно используют и публикуют одну и ту же базовую онтологию терминов, которыми они все пользуются, то компьютерные агенты могут извлекать информацию из этих различных сайтов и накапливать ее. Агенты могут использовать накопленную информацию для ответов на запросы пользователей или как входные данные для других приложений.

Обеспечение возможности использования знаний предметной области стало одной из движущих сил недавнего всплеска в изучении онтологий. Например, для моделей многих различных предметных областей необходимо сформулировать понятие времени. Это представление включает понятие временных интервалов, моментов времени, относительных мер времени и т.д. Если одна группа ученых детально разработает такую онтологию, то другие могут просто повторно использовать ее в своих предметных областях. Кроме того, если нам нужно создать большую онтологию, мы можем интегрировать несколько существующих онтологий, описывающих части большой предметной области. Мы также можем повторно использовать основную онтологию, такую как UNSPSC, и расширить ее для описания интересующей нас предметной области.

Создание явных допущений в предметной области, лежащих в основе реализации, дает возможность легко изменить эти допущения при изменении наших знаний о предметной области. Жесткое кодирование предположений о мире на языке программирования приводит к тому, что эти предположения не только сложно найти и понять, но и также сложно изменить, особенно непрограммисту. Кроме того, явные спецификации знаний в предметной области полезны для

новых пользователей, которые должны узнать значения терминов предметной области.

Отделение знаний предметной области от оперативных знаний - это еще один вариант общего применения онтологий. Мы можем описать задачу конфигурирования продукта из его компонентов в соответствии с требуемой спецификацией и внедрить программу, которая делает эту конфигурацию независимой от продукта и самих компонентов. После этого мы можем разработать онтологию компонентов и характеристик ЭВМ и применить этот алгоритм для конфигурирования нестандартных ЭВМ. Мы также можем использовать тот же алгоритм для конфигурирования лифтов, если мы предоставим ему онтологию компонентов лифта.

Анализ знаний в предметной области возможен, когда имеется декларативная спецификация терминов. Формальный анализ терминов чрезвычайно ценен как при попытке повторного использования существующих онтологий, так и при их расширении.

Часто онтология предметной области сама по себе не является целью. Разработка онтологии сродни определению набора данных и их структуры для использования другими программами. Методы решения задач, доменно-независимые приложения и программные агенты используют в качестве данных онтологии и базы знаний, построенные на основе этих онтологий.

Тема 4. Проектирование структур баз данных и баз знаний

Базы данных в проектировании и реализации информационных систем. Модели данных. Методология проектирования баз данных.

Предпосылки появления БД

Две основные предпосылки появления баз данных:

- Необходимость хранить и обрабатывать большое количество данных.
- Разработка методов совместного использования данных.

Массив данных общего пользования в системах, основанных на данных, называется ***базой данных***

Базы данных: термины

Информация – любые сведения о каком-либо событии, объекте или процессе, являющиеся объектом некоторых операций: восприятия, передачи, преобразования, хранения или использования.

Данные – это информация, зафиксированная в некоторой форме, пригодной для последующей обработки, передачи и хранения, например, находящаяся в памяти ЭВМ или подготовленная для ввода в ЭВМ.

Обработка данных – это совокупность задач, осуществляющих преобразование массивов данных.

Обработка данных включает в себя:

- ввод данных в ЭВМ,
- отбор данных по каким-либо критериям,
- преобразование структуры данных,
- перемещение данных на внешней памяти ЭВМ,
- вывод данных, являющихся результатом решения задач, в табличном или в каком-либо ином удобном для пользователя виде.

Система обработки данных (СОД) – это набор аппаратных и программных средств, осуществляющих выполнение задач по управлению данными.

Управление данными – совокупность функций обеспечения требуемого представления данных, их накопления и хранения, обновления, удаления, поиска по заданному критерию и выдачи данных. [ГОСТ 20886-85]

Предметная область – часть реального мира, подлежащая изучению с целью организации управления и, в конечном итоге, автоматизации.

База данных (БД) – совокупность связанных данных конкретной предметной области, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ.

Фактически база данных – это модель предметной области (ПО).

Данные БД могут быть использованы для решения многих задач многими пользователями

Ведение базы данных – деятельность по обновлению, восстановлению и изменению структуры базы данных с целью обеспечения её целостности, сохранности и эффективности использования [ГОСТ 20886-85].

Система управления базами данных (СУБД) – это совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия её с прикладными программами [ГОСТ 20886-85].

Требования к СУБД

- Эффективное выполнение функций ПО
- Минимизация избыточности
- Предоставление непротиворечивой информации
- Безопасность
- Простота в эксплуатации
- Простота в физической реорганизации
- Возможность централизованного управления
- Упрощение приложений

Автоматизированная информационная система (АИС) – совокупность программно-аппаратных средств, предназначенных для автоматизации деятельности, связанной с хранением, передачей и обработкой информации

Банк данных (БнД) – это автоматизированная информационная система, включающая в свой состав комплекс специальных методов и средств (математических, информационных, программных, языковых, организационных и технических) для поддержания динамической информационной модели предметной области с целью обеспечения информационных запросов пользователей.

Банк данных должен:

- Обеспечивать информационные потребности внешних пользователей.
- Обеспечивать возможность хранения и модификации больших объёмов многоаспектных данных.
- Обеспечивать заданный уровень достоверности хранимых данных и их непротиворечивость.
- Обеспечивать доступ к данным только пользователям с соответствующими полномочиями.
- Обеспечивать поиск данных по произвольной группе признаков.
- Удовлетворять заданным требованиям по производительности при обработке запросов.
- Иметь возможность реорганизации при изменении границ ПО.
- Обеспечивать выдачу пользователям данных в различной форме.
- Обеспечивать простоту и удобство обращения внешних пользователей к данным

Этапы проектирования БД

- Проектирование данных – процесс перевода общих

представлений о данных, выраженных концептуальной моделью, в конкретную логическую модель.

- На этапе инфологического проектирования осуществляется построение семантической модели, описывающей сведения из предметной области которые могут заинтересовать пользователей БД

Семантическая модель – представление совокупности сведений о ПО в виде графа, в вершинах которого расположены понятия, а дуги представляют отношения между понятиями

Подобные модели называются моделями «сущность-связь» (ER-model: Entity-Relationship)

Реальные процессы и объекты предметной области представлены в ER-моделях в виде сущностей.

Сущности: базовые и зависимые. Тип сущности. Экземпляр сущности.

Для каждого типа сущности необходимо определить имя.

Атрибуты: характеристики сущностей. Атрибуты бывают:

- Идентифицирующие и описательные.
- Составные и простые.
- Однозначные и многозначные.
- Основные и производные.
- Обязательные и необязательные.

Для каждого атрибута необходимо определить название, указать тип данных и описать ограничения целостности – множество значений, которые может принимать данный атрибут.

Между сущностями ПО могут существовать связи, имеющие различный содержательный смысл(семантику). Например, студент учится в группе

- Датологическое проектирование подразделяется на логическое и физическое проектирование
    - Логическое проектирование–преобразование требований к данным в структуры данных (в форматах, поддерживаемых выбранной СУБД).
    - Физическое проектирование – определение особенностей хранения данных, методов доступа и т. д.

Различие уровней представления данных на каждом этапе проектирования реляционной базы данных:

- КОНЦЕПТУАЛЬНЫЙ УРОВЕНЬ — Представление аналитика (используется модель «сущность-связь»). Термины: сущности, атрибуты, связи
- ЛОГИЧЕСКИЙ УРОВЕНЬ — Представление программиста. Термины: записи, элементы данных, связи между записями
- ФИЗИЧЕСКИЙ УРОВЕНЬ — Представление администраторов БД, СУБД, ЛВС. Термины: группирование данных, индексы, методы доступа, распределенная БД, репликация,

производительность, надежность, кластеры…

Независимость данных

Данные независимы, если существует возможность нормального функционирования БД при изменениях как со стороны концептуальной, так со стороны физической модели, т.е. обеспечивается логическая и физическая независимость

Модели данных

Модель данных – это формализованное описание, отражающее состав и типы данных, а также взаимосвязь между ними

Модель данных состоит из трех частей:

- Набора типов структур данных
- Множества допустимых операций, выполняемых в структуре данных
- Ограничения для контроля целостности данных

Типы структур

- Структурированная запись базируется на использовании концепций «агрегации» и «обобщения».
- Один из первых вариантов структуризации данных был предложен Ассоциацией по языкам обработки данных (Conference of Data Systems Languages, CODASYL)

Модели данных

- Модели первого поколения
- Иерархическая
- Сетевая
- Модели второго поколения
- Реляционная
- Модели третьего поколения
- Объектно-ориентированная
- Объектно-реляционная

Свойства иерархической модели

- Существует корень
- Узел содержит атрибуты
- Исходный и зависимый узлы находятся в отношении «непосредственный предок и потомок»
- Потомок соединен единственной связью с предком
- Предок может иметь несколько потомков
- Доступ к данным производится через предка
- Может существовать множество экземпляров узла
- При удалении узла удаляется все его поддерево

Операции над данными:

- ДОБАВИТЬ в БД новую запись. Для корневой записи обязательно формирование значения ключа
- ИЗМЕНИТЬ значение данных выбранной записи. Ключевые данные не должны подвергаться изменениям

- УДАЛИТЬ некоторую запись и все подчиненные ей записи
- ИЗВЛЕЧЬ
- Корневую запись по ключу
- Следующую запись (в порядке левостороннего обхода дерева)

Ограничения целостности

- Поддерживается только целостность связей между владельцами и членами группового отношения.
- Не обеспечивается автоматическое поддержание соответствия парных записей, входящих в разные иерархии

Достоинства:

- Простота в использовании и обслуживании
- Высокая скорость доступа

Недостатки

- Невозможность реализации отношения «многие ко многим» в рамках одной БД
- Возможны аномалии в работе с БД

Сетевая модель данных определена в тех же терминах, что и иерархическая модель данных

1. Отличия сетевой и иерархической моделей:

- Запись может быть членом более чем одного группового отношения
- Каждое групповое отношение именуется
- Тип группового отношения определяет свойства общие для всех экземпляров данного типа
- Экземпляр группового отношения представляется записью-владельцем и множеством (возможно пустым) подчиненных записей
- Экземпляр записи не может быть членом двух экземпляров групповых отношений одного типа

Операции над данными:

ДОБАВИТЬ в БД новую запись в зависимости от режима включения, либо включить ее в групповое отношение, где она объявлена подчиненного, либо не включать ни в какое групповое отношение.

ВКЛЮЧИТЬ В ГРУППОВОЕ ОТНОШЕНИЕ связать существующую подчиненную запись с записью-владельцем

ПЕРЕКЛЮЧИТЬ связать существующую подчиненную запись с другой записью-владельцем в том же групповом отношении.

ОБНОВИТЬ изменить значение элементов предварительно извлеченной записи

Достоинства:

- Реализуется отношение «многие ко многим»
- Высокая производительность

Недостатки

- Трудность реорганизации БД

- В процессе эксплуатации за счет некорректных удалений, сбоев и т.д. накапливается мусор – данные к которым нет доступа
- Слабая выразительность языка запросов. Обычно он позволяет манипулировать лишь одной запись одновременно, программист во время работы должен хорошо представлять пути доступа к данным

Реляционная модель данных

Основные понятия

- Предложена сотрудником компании IBM Е.Ф. Коддом в 1970 г.
- Кодд предложил использовать для обработки баз данных аппарат теории множеств и теории отношений.
- В реляционной модели данных достигается гораздо более высокий уровень абстракции данных, чем в иерархической или сетевой.

В основу реляционной модели Кодд положил три принципа (стремления):

- Независимость данных на логическом и физическом уровнях – стремление к независимости
- Создание структурно-простой модели – стремление к коммуникабельности
- Использование концепции языков высокого уровня для описания операций над порциями информации – стремление к обработке множеств.

В отличие от иерархической и сетевой моделей данных в реляционной отсутствует понятие группового отношения.

Для отражения ассоциаций между кортежами разных отношений используется дублирование их ключей.

Свойства отношений

- Нормализованные отношения представляются в виде табличной структуры
- Каждый кортеж отношения является уникальным, т.е. в отношении не может быть повторяющихся строк
- Упорядоченность кортежей теоретически несущественна
- Количество атрибутов и их порядок в отношении должно быть фиксированным (т.е. кортежи должны иметь одинаковую длину и формат)

Операции над данными:

Обрабатывающая часть модели определяется операциями реляционной алгебры:

- Выборка
- Проекция
- Объединение
- Пересечение
- Соединение
- и др.

Ограничения целостности

Существуют два ограничения, которые должны выполняться в любой реляционной базе данных. Это:

- Целостность сущностей.
- Целостность внешних ключей.

Тема 5. Методологии проектирования и управления информационными системами в парадигмах программирования

Одна из ключевых проблем современного программирования – повторное использование модулей и компонентов (КПИ). Ими могли быть программы, подпрограммы, алгоритмы, спецификации и т. п., пригодные для использования при разработке новых более сложных ПС. В них материализуется многолетний опыт компьютеризации разных знаний в деятельности человека, который может их применять непосредственно либо путем настройки и адаптации КПИ к новым условиям среды обработки. После модулей, используемых долгое время в практике программирования, появились КПИ, готовые к применению. Использование КПИ в программировании дало возможность усовершенствовать метод снизу– вверх проектирования сложных программ из простых программных элементов, находящихся в библиотеках. Программирование прошло длинный путь своего развития – от элементов библиотек стандартных машинных подпрограмм общего назначения на конкретной ЭВМ до современных библиотек программ и КПИ нового поколения в современных языках С, С++, Basic, JAVA и др.

Модуль – это логически законченная часть программы, выполняющая определенную функцию, обладающая свойствами завершенности, повторного использования и др.

Модуль возник как обобщение понятия стандартных подпрограмм и процедур, которые описывались в ЯП. В их заголовке задавались внешние входные данные, а в теле модуля – операторы вычислений и вызовов подпрограмм по их имени и списку фактических параметров. Последовательность и число формальных параметров соответствовало фактическим параметрам. Вызов заготовок на одном ЯП не является проблемным, так как типы данных параметров совпадают с типами данных ЯП.

Межмодульный интерфейс – это интерфейсный модуль-посредник между двумя взаимодействующими программными объектами, выполняя функции передачи и приема данных между ними. Впервые разработан язык определения интерфейсов (ЯОИ) для описания операторов вызова модулей, параметров и их типов, а также операций проверки правильности обмена данными.

Необходимыми условиями применения этого метода программирования является:

1) наличие большого количества разнообразных КПИ, как объектов сборки;
2) паспортизация объектов сборки;
3) наличие довольно полного набора стандартных правил сопряжения объектов, алгоритмов их реализации и средств автоматизации процесса сборки;

4) технологии линией с последовательностью операций постепенного изготовления и установления связей между КПИ при образовании системы или семейства ПП.

ПАРАДИГМА ОБЪЕКТНОГО ПРОГРАММИРОВАНИЯ

На рубеже 80-х XX столетия Г. Буч предложил объектный подход, который изменил сложившийся процесс структурного, функционального программирования. В то время индустрию программного обеспечения постиг *кризис сложности*.

Принципы объектного анализа в моделировании:

- Принцип всеобщности означает, что на произвольном шаге объектного анализа все сущности – суть объекты.
  - Аксиома 1. Предметная область, которая моделируется из объектов, сама является объектом.
  - Аксиома 2. Моделируемая предметная область может быть отдельным объектом в составе другой предметной области.
- Принцип существенности объектных различий. На произвольном шаге объектного анализа каждый объект является уникальным элементом.
  - Аксиома 3. Каждый объект имеет по крайней мере одно свойство или характеристику, которая задает его уникальную идентификацию во множестве объектов.
- Принцип объектной упорядоченности. На произвольном шаге объектного анализа все объекты упорядочены в соответствии с отношениями между объектами.
  - Аксиома 4. Каждый объект имеет при необходимости одно отношение с другим объектом, которое обеспечивает его упорядоченность в рамках этой пары объектов.
- Принцип целостности объектной модели. На произвольном шаге объектного анализа совокупность объектов и отношений между ними однозначно определяют объектную модель предметной области для определенного уровня ее абстракции.
  - Аксиома 5. На произвольном шаге объектного анализа объектную модель можно представить в виде ориентированного связного графа, вершинами которого являются объекты, а дугам соответствуют отношения между объектами

Четырехуровневое объектное моделирование ПрО

- I. Обобщающий уровень определяет базовые понятия ПрО – объекты без учета их сущности и свойств. Объект задается в виде денотата в соответствии с теорией Фреге в виде <имя объекта> <концепт>.
- II. Структурный уровень определяет расположение объектов в структуре модели ПрО, устанавливает упорядочение объектов и представление их структуры в виде графа с операциями над объектами (объединения, пересечения, разности, приложения,

симметричной разницы и др.)

- III. Характеристический уровень задает общие, специфические свойства и характеристики концептов объектов в логико-алгебраическом представлении объектов в виде графа с характеристиками и свойствами в модели.
- IV. Поведенческий уровень определяет поведение и изменение объектов в зависимости от событий, которые они создают при их выполнении.

ПАРАДИГМА КОМПОНЕНТНОГО ПРОГРАММИРОВАНИЯ

Переход к компонентам происходил эволюционно, начиная от подпрограмм, модулей и функций. При этом усовершенствовались сами элементы, методы их композиции и накопления для последующего использования [7].

К базовым элементам относятся: объект, компонент и сервис. Каждый из них включает: в себя описание имени (идентификатор): описание интерфейса в виде операторов вызова и параметров.

По сути компонент является самостоятельным продуктом, который реализует функцию предметной области и может взаимодействовать с другими компонентами через интерфейсы. Каждый компонент конструируется самостоятельно, как некоторая абстракция, которая содержит в себе информационную часть и артефакт.

Информационная часть имеет паспорт, который содержит назначение, дату изготовления, условия применения (ОС, среда, платформа и т. п.) и др.

Артефакт – это реализация (implementation), интерфейс (interface) и схема развертывания (deployment) компонента**.**

Реализация – это код, который будет выполняться при обращении к операциям, определенным в интерфейсах компонента. Компонент может иметь несколько реализаций в зависимости от операционной среды, модели данных, СКБД и др.

Интерфейс – это операции обращения к другим компонентам в языках IDL или APL для передачи аргументов и результатов при взаимодействии компонентов между собой. Каждый компонент может иметь множество интерфейсов.

Развертывание – это выполнение физического (кода) конфигурационного файла компонента путем запуска.

В общем случае сложились четыре возможных класса формальных композиций, элементами которых являются компоненты и каркасы, взаимодействующие между собой.

Сборка компонент–компонент обеспечивает непосредственное взаимодействие компонентов через интерфейс на уровне приложения.

Сборка каркас–компонент обеспечивает взаимодействие каркаса с компонентами, при котором каркас управляет ресурсами компонентов и их интерфейсов. Такое взаимодействие осуществляется на системном

уровне.

Сборка компонент–каркас обеспечивает взаимодействие компонента с каркасом типа "черного ящика", в видимой части которого находятся спецификации для его развертывания и выполнения определенной сервисной функции. Такое взаимодействие осуществляется на сервисном уровне.

Сборка каркас–каркас обеспечивает взаимодействие между каркасами, каждый из которых разворачивается в гетерогенной среде, компоненты которой взаимодействуют между собой через их интерфейсы на сетевом уровне.

Компоненты высокого уровня допускают агрегацию компонентов согласно рассмотренной композиции первого класса, а также взаимодействие каркасов и высокоуровневых компонентов.

Метод композиции – это средство или планомерный подход для обеспечения взаимосвязей компонентов в сложных структурах (комплексах, интегрированных, распределенных системах).

Главное в методе композиции компонентов – это понятие межкомпонентного (модульного) и межъязычного интерфейсов, которые обеспечивают взаимодействие в современных сетевых и гетерогенных средах.

ГЕНЕРИРУЮЩЕЕ ПРОГРАММИРОВАНИЕ. МОДЕЛИ И МЕТОДЫ

В начале 70-х годов XX ст. Г.Дейкстра ввел понятие семейства программ с общей "семейной" постановкой задач, по которой они порождаются, как программы. Некоторые из них менялись в связи с недостаточно заданными функциями или уточнением постановки отдельных задач семейства.

Цель генерирующего программирования – разработка разных программных ресурсов для их применения в сборке целевого семейства. На его основе сформировались направления:

1) прикладная инженерия – процесс производства конкретных ПС из КПИ и отдельных элементов процесса создания некоторой ПрО;
2) инженерия ПрО – построение семейства ПС путем сбора, классификации, фиксации КПИ и отдельных членов систем с четким выделением их задач.

Основные этапы инженерии ПрО: анализ ПрО и выявление общих и изменяемых характеристик в соответствующей МХ модели. Она устанавливает зависимость между различными членами семейства, а также создает базис для изготовленияч конкретных членов семейства с учетом механизмов изменчивости отдельных элементов в семействе. На основе модели МХ, повторных компонентов знания о конфигурациях генерируется доменная модель семейства.

СЕРВИСНОЕ ПРОГРАММИРОВАНИЕ

Эта парадигма программирования появилась как следствие рассмотрения программных компонентов, которые могут

использоваться в качестве сервиса. Для них определены интерфейсы взаимодействия с разными программами и распределенными системами (CORBA, DCOM и EJB, MS.Net, IBM и тому подобное) и с веб-сервисами Интернета. Сейчас действуют сервисноориентированная архитектура SOA (Service-Oriented Architecture), средства их поддержки (XML, SOAP, WSDL и др.) и механизмы взаимодействия обычных сервисов распределенных приложений и веб-сервисов Интернет.

Рассматривается три вида сервисов:

1) общие системные сервисы, которые есть в каждой общесистемной среде для поддержки процессов проектирования и реализации РПС на основе сформулированных моделей ПС, ПС и РПС;
2) объектные сервисы, которые поддерживают объекты и классы, операции ЖЦ, услуги необходимы для разработки РПС в объектно-ориентированной среде;
3) веб-сервисы, которые базируются на информационных ресурсах Интернет и обеспечивают создание элементов РПС путем композиции или интеграции КПИ и сервисов, способных к функционированию в вебе Всемерной паутины.

Сервисно-ориентированная архитектура (SOA) – это совокупность взаимодействующих между собой системных сервисов и веб-сервисов.

Любой из компонентов SOA создается сервисами безотносительно к конкретным технологиям, за которые можно принять готовые приложения типа "черный ящик". Интеграция компонентов и сервисов в архитектуру SOA включает в себя следующие виды:

1) *пользовательскую интеграцию* (user integration) для взаимодействия информационной системы с конкретным пользователем;
2) *связь приложений* (application connectivity) для задания взаимодействия;
3) *интеграцию процессов* (process integration) в бизнес-приложениях;
4) *информационную интеграцию* (information integration) для обеспечения доступа к интегрированной информации и данным.

Тема 6. Развитие методологий проектирования и управления информационными системами с точки зрения различных научных подходов и технологических решений

Рассмотрим следующие три проблемы

Проблема: 1) Сложность и разнородность

- Системы управления информационными технологиями (ИТ) предприятий и организаций являются достаточно **сложными**, поскольку требуется учет интересов множества участников, вовлеченных в создание и использование ИТ-ресурсов (спонсоров создания информационной системы, конечных пользователей и разработчиков)

Проблема: 2) Динамичность и инертность

- Системные изменения бизнеса (слияния, поглощения, смена собственника, пересмотр стратегических целей)
- Появление новых и развитие существующих информационных технологий, появление принципиально новых технических решений;
- Необходимость поддержания целостности хранимых данных.

Проблема: 3) Цифровая трансформация

- Цифровая трансформация - это не просто эволюция ИТ, а целостное изменение бизнеса, затрагивающее всю организацию

Понятие ИТ-менеджмента

ИТ-менеджмент охватывает управление всеми компьютерными и коммуникационными ресурсами предприятия.

Его основная задача состоит в создании и поддержании в работоспособном состоянии приложений и инфраструктуры, на которой они исполняются.

Уровни управления

- Стратегический, тактический и операционный.
- На стратегическом уровне обеспечивается установление соответствия между информационными функциями системы и ее контентом, что сводится к атрибуции задач на поле информационной политики, определению содержания информационных функций и ИТ-поддержке.
- На операционном и тактическом уровнях ИТ-менеджмента должны обеспечиваться заданные уровни работоспособности и надежности эксплуатации приложений информационной системы (ИС) на продолжении всего жизненного цикла системы
- Объекты управления ИТ-менеджмента
- Инфраструктура ИТ включает техническое и системное программное обеспечение. Техническое обеспечение ИТ состоит из серверов, персональных компьютеров, систем хранения данных, сети и коммуникационных приложений. Программное обеспечение характеризуется операционными системами, инструментальными средами разработки, программами

поддержки ИТ-менеджмента и средствами обеспечения информационной безопасности.

- Приложения обеспечивают поддержку бизнес-процессов предприятия и работоспособность отдельных автоматизированных рабочих мест.
- Организационная структура службы ИТ определяет состав подразделений, распределение между ними функций и задач. Служба ИТ должна обеспечивать разработку, ввод в действие и эксплуатацию информационной системы посредством координированных действий, которые обеспечивают непрерывность функционирования существующей системы в соответствии с согласованными правилами и процедурами на протяжении жизненного цикла ИТ.
- ИТ-проекты представляют собой проекты внедрения новых информационных систем, а также модернизацию существующих. При этом модернизация (изменения, дополнения) рассматривается как результат действий, выполненных по запросу и относящихся к функциональным или нефункциональным требованиям, которые не были специфицированы изначально, при разработке и внедрении системы.

ИТ-сервис

Основная роль ИТ на предприятии определяется как информационное обслуживание её подразделений с целью повышения эффективности бизнеса. Информационное обслуживание бизнеса состоит в предоставлении информационных сервисов (ИТ-сервисов) заданного качества подразделениям предприятия.

ИТ-сервис в корпоративной среде – это ИТ-услуга, которую ИТ-подразделение (департамент, отдел, служба) или внешний провайдер предоставляет бизнес-подразделениям предприятия для поддержки их бизнес-процессов.

Примерами корпоративных ИТ-сервисов могут быть электронная почта, сетевая инфраструктура, системы хранения данных, бизнес-приложения (начисление заработной платы, формирование счетов), бизнес-функции (списание/начисление денежных средств на счете клиента).

Три группы ИТ-сервисов. Набор ИТ-сервисов, необходимых организации, индивидуален и в значительной степени зависит от отрасли, размеров организации, уровня автоматизации, квалификации персонала, стратегии развития и т. п. Корпоративные ИТ-сервисы можно разбить на три большие группы:

- поддержка ИТ-инфраструктуры;
- поддержка бизнес-приложений;
- поддержка пользователей.

Параметры ИТ-сервисов. В общем случае ИТ-сервис

характеризуется рядом параметров:

- функциональность;
- время обслуживания;
- доступность;
- надежность;
- производительность;
- конфиденциальность;
- масштаб;
- затраты.

Деятельность службы ИС. Факторы, влияющие на выбор организационной структуры службы ИС:

- **масштаб службы ИС** - более крупные службы ИС обычно имеют более сложную и разветвленную организацией структуру;
- **отраслевую принадлежность**, с которой связано наличие или, напротив, отсутствие определенных структурных подразделений;
- **распределение организации по территории** - наличие территориально удаленных подразделений и филиалов существенно меняет организационную структуру службы ИС.

Функциональная модель управления ИС. Функциональная модель управления и основанная на ней организационная структура службы ИС длительное время представляли собой основной и единственный подход к управлению в этой области. Однако со временем выявился ряд ограничений функционального подхода, снижавших эффективность управления службой ИС

Противоречия между параметрами и функциональными направления деятельности служб.

Параметры ИТ-сервисов:

- функциональность;
- время обслуживания;
- доступность;
- надежность;
- производительность;
- конфиденциальность;
- масштаб;
- затраты.

Функциональные направления деятельности служб :

- планирование и организация;
- разработка, приобретение и внедрение;
- предоставление и сопровождение ИТ-сервиса;
- мониторинг.

1) Между функциями службы ИС и параметрами ИТ-сервиса нет прямого и однозначного соответствия: каждый параметр сервиса ИТ может определятся несколькими функциями ИТ и наоборот, одна и та же функция службы также может относиться к нескольким сервисам.

2) «Точка контакта» и ответственность за качество ИТ-сервисов требуют полномочий высокого уровня, вплоть до уровня директора ИТ. В результате руководители высокого уровня оказываются перегруженными большим потоком задач.

Шаги процессного подхода к управлению

- определение цели процесса и показателей достижения этой цели (количественных или качественных);
- назначение ответственного за процесс, задачей которого является достижение цели процесса;
- регламентация процесса в целом и составляющих его работ;
- при необходимости - автоматизация процесса посредством инструментальных средств, разработанных в самой организации либо закупленных извне.

Процессный подход к управлению

Управление процессами изменяет лишь управленческие функции службы ИС, не затрагивая функции собственно разработки и сопровождения ИТ-сервисов. Изменения состоят в систематическом целенаправленном решении задач координации функций в ходе выполнения процессов службы ИС. Для этого достаточно формализовать соответствующий процесс, т.е. назначить менеджера процесса, определить роли участников процесса и установить правила его выполнения, т.е. последовательность выполнения операций процесса, обязанности в рамках ролей, правила эскалации и т.д.

Переход к процессной модели управления обычно не требует ни дополнительного персонала, ни изменений в организационной структуре. Участники процесса выполняют свои должностные обязанности в рамках существующей организационной структуры; часть этих обязанностей, относящаяся к данному процессу, формализована в виде ролей процесса. Если все процессы службы ИС формализованы, то совокупность ролей совпадает с должностными обязанностями сотрудника

Преимущества типовых моделей

- Во-первых, типовая модель представляет в концентрированном виде опыт управления службой ИС в тысячах и даже десятках тысяч компаний. Соответственно, отказ от использования этого массива знаний по меньшей мере нецелесообразен.
- Во-вторых, переход к процессной модели управления для всех задач службы ИС одновременно, в рамках одного проекта маловероятен. В этом случае процессная модель дает менеджеру образ будущего, который становится ориентиром в ходе отдельных шагов внедрения.
- В-третьих, типовая модель процессов службы ИС всегда опирается на некую систему понятий, на некий язык. Использование этого языка значительно облегчает достижение взаимопонимания участников процесса.

- В-четвертых, типовая модель процессов поддержана разработчиками программного обеспечения автоматизации управления службой ИС и инфраструктурой ИТ. В результате программное обеспечение реализует именно эти процессы. Реализация собственных процессов потребует разработки собственного ПО.
- Наконец, стандартная модель процессов обычно внедряется во многих организациях. В результате образуется сообщество пользователей, которое является ценным источником информации по внедрению модели.

ITSM сосредоточен на процессах:

- Поддержка и доставка IT-услуг,
- Понимание текущего состояния IT-инфраструктуры,
- Поиск лучших практик управления IT посредством нахождения общего языка между пользователями и исполнителями,
- Создание технологического маршрута для бизнеса

Суть ITSM

IT-отделы концентрируются на технологических проблемах, но в то же время существуют по меньшей мере еще два аспекта, сказывающиеся на результативности деятельности: соответствие запросам конечных пользователей (сотрудников и клиентов, к которым поступит IT-продукт) и экономическая эффективность (оптимальная стоимость реализации продукта в соответствии с бюджетом).

ITSM фокусируется на этих аспектах и приносит лучшее понимание того, что нужно бизнесу и почему, то есть выступает фундаментом для повторяемых и масштабируемых процессов, сокращает дистанцию между конечными пользователями и IT-отделом. Благодаря этому у клиентов формируются реалистичные ожидания, а задержки между обнаружением проблем и их устранением минимизируются.

ITIL/ITSM - концептуальная основа процессов ИС-службы

ITIL (произносится как «а́йтил», ранее аббревиатура от IT Infrastructure Library — библиотека инфраструктуры информационных технологий. Сегодня это уже не аббревиатура, а отдельное название или бренд, используемый во всем мире.

По мере развития библиотеки инфраструктуры ИТ, акцент сместился на управление услугами и к подходу к жизненному циклу, а элемент инфраструктуры практически исчез

Самое распространенное в мире руководство по управлению ИТ-услугами (ITSM)

Вехи истории

1989 год Центральному агентству по компьютерам и телекоммуникациям (CCTA, Великобритания) было поручено создать набор стандартных методов, которые могли бы более согласованно объединить ИТ-системы как государственного, так и частного секторов.

Эта ранняя версия называлась GITIM, Government Information Technology Infrastructure Management, и хотя она сильно отличается от сегодняшней ITIL, они обе разделяют цель предложить улучшенную поддержку и поставку.

Вскоре библиотека ITIL выросла до каталога из 30 томов, в котором рекомендовались и предоставлялись лучшие практики в области информационных технологий, ориентированные на клиентов и бизнес-потребности и учитывающие их.

2000 год Выпущена ITIL V2. Новая версия была нацелена на то, чтобы сделать ITIL более доступной для широких масс, и разбила 30-томный фреймворк на девять взаимосвязанных категорий. Компания Microsoft приняла ITIL в качестве основы для развития своей Microsoft Operations Framework (MOF). В течение следующих нескольких лет ITIL стал стандартом передового опыта в области информационных технологий и наиболее широко используемым в мире инструментом управления ИТ-услугами.

2006 год Выпуск Глоссария ITIL, который стал еще одним подтверждением стремления ITIL к удобству использования.

2007 год ITIL V3. В новой версии больше внимания уделялось интеграции IT-бизнеса и которая была сосредоточена на концепции жизненного цикла услуги. ITIL v3 сжал 26 процессов и функций в 5 томов, а после запуска приобрел название ITIL Refresh Project (проект обновления ITIL).

2011 год Пересмотр V3, устранены ошибки и несоответствия. Обновлены 5 томов каталога услуг ITIL: Стратегия услуги ITIL, Проектирование услуги ITIL, Преобразование услуги ITIL, Эксплуатация услуги ITIL и Постоянное улучшение услуги ITIL. Этот комплект ITIL в настоящее время является основой для всех лучших практик ITIL по всему миру.

2013 год ITIL принадлежит компании AXELOS Ltd - совместному предприятию Capita Plc и Кабинета министров Великобритании.

2019 год ITIL 4. V4 имеет более практическое руководство по использованию ITIL, особенно в средах совместной работы (collaborative environments). Это облегчает организациям согласование ITIL с DevOps, Agile и Lean методами работы.

Ключевыми компонентами фреймворка ITIL 4 являются:

- Система создания ценности услуг ITIL (Service Value System)
- Модель четырех измерений (4 dimensions model)

ITIL и ITSM — история большого обмана. Есть ли польза? Сколько это стоит и кому точно НЕ стоит «внедрять ITIL»?

Решения Hewlett-Packard по управлению информационными системами

Для практического применения ITIL компания HP разработала собственный вариант методологии, получивший название "Типовой

модели HP ITSM" (IT Service Management Reference Model – ITSM Reference Model).

Ее первый вариант был опубликован в сентябре 1997 г., следующий - в январе 2000 г. Финальная версия HP ITSM 3.0 выпущена в июне 2003 г.

В настоящее время - часть портфеля решений Hewlett Packard Enterprise (HPE)

Ключевая идея методологии ITSM RM

Несмотря на разнообразие информационных систем, их работа на 80% может быть построена на базе стандартизованных процессов и регламентов. Поэтому адаптация методологии к конкретным, специфическим задачам предприятия требует настройки не более 20% системы ИТ-сервиса.

Пять основных групп процессов ITSM RM

- согласование задач бизнеса и ИТ (Business – IT Alignment);
- планирование и управление ИТ-сервисами (Service Design & Management);
- разработка и внедрение ИТ-сервисов (Service Development & Deployment);
- оперативное управление ИТ-сервисами (Service Operations);
- обеспечение ИТ-сервисами (Service Delivery Assurance).