

УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные сведения, дайте ответы на контрольные вопросы

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com **до 06.03.2023.**

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: <https://vk.com/daykini>

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция 7

Тема: Базовые принципы и понятия технологии разработки объектно-ориентированных информационных систем

Цель: Изучить базовые принципы и понятия технологии разработки объектно-ориентированных информационных систем

1. Функциональное и визуальное моделирование деятельности

1.1. Структурный и объектно-ориентированный подходы к разработке ИС

В процессе создания ИС разработчиками используются два альтернативных подхода – *структурный и объектно-ориентированный*.

Структурный подход к разработке является классическим и предполагает последовательную реализацию следующих этапов разработки:

- анализа предметной области,
- проектирования,
- создания программных модулей,
- объединения модулей в единую систему,
- тестирования,
- внедрения.

Сущность структурного подхода к разработке ИС заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые, в свою очередь, делятся на подфункции, подразделяемые на задачи и т. д. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимосвязаны.

Однако эта технология не лишена некоторых недостатков. В соответствии с классическим подходом основное внимание должно уделяться *информации*, с которой работает система. Разработчик спрашивает пользователей, какая информация им нужна, проектирует БД для хранения этой информации, создает экранные формы для ее вывода, встраивает возможность распечатывать отчеты. Иначе говоря, разработчик «фокусируется» на *самой информации*, а тому, что с ней делать, т. е. *поведению* системы уделяется меньше внимания. Такой подход называется «*ориентированным на данные (data centric)*», он применялся при разработке тысяч различных систем много лет. Он неоптимален при проектировании БД и систем обработки информации, но при разработке бизнес-приложений возникают проблемы, главная из которых

состоит в том, что требования к системе могут меняться со временем. Система, ориентированная на данные легко приспосабливается к изменениям БД, однако изменить деловые правила или поведение такой системы значительно труднее.

Это обусловлено тем, что на основе информации о структуре БД генерируется клиентское приложение. Поскольку БД должна быть обязательно нормализована, данные хранятся в таблицах БД не всегда в той же форме, в которой они должны появляться на экранных формах. Другими словами, если код приложения генерируется не на основе описания предметной области, невозможно построить эффективное приложение со сложной бизнес-логикой.

Вторым недостатком структурного подхода является большая вероятность обнаружения ошибок на последнем этапе разработки, например этапе тестирования. В случае обнаружения ошибки необходимо вернуться на тот этап, где она допущена, и заново пройти все последующие этапы.

Для решения этой проблемы и были созданы **объектно-ориентированные** методы разработки ИС, при применении которых внимание уделяется как *информации*, так и *поведению*, что позволяет создавать *гибкие* системы, допускающие изменение их поведения и/или содержащейся в них информации. Эти методы основаны на использовании **объектно-ориентированного программирования (ООП)**.

Объектно-ориентированное программирование – это новая методология, созданная во второй половине 1980-х гг. Она позволяет решить проблему появления новых акцентов в программировании, обусловленных распространением персональных компьютеров в мелком и среднем бизнесе, в котором вычислительные и расчетно-алгоритмические задачи уступили первое место задачам обработки и манипулирования данными. ООП предполагает представление программы как набора взаимодействующих *объектов*, наделенных определенным *поведением* и *реакцией* на изменение внешних условий. В то время как классический процедурно-ориентированный стиль программирования направлен на представление программы в виде множества поочередно вызываемых процедур.

Объектно-ориентированные методы разработки основаны на *объектно-ориентированной концепции* или, как часто говорят, *парадигме*. *Парадигма* – теория (или модель, тип постановки проблемы), принятая в качестве образца решения исследовательских задач. *Объектно-ориентированная парадигма* – способ описания приложений, при котором приложение предварительно делится на множество маленьких кусочков или *объектов* относительно независимых друг от друга. Готовое приложение можно затем создать, сложив эти объекты вместе. Преимуществом такого решения является возможность разработки компонентов только один раз с последующим многократным их использованием для создания абсолютно различных по назначению систем.

Снижение риска в объектно-ориентированной технологии достигается за счет реализации технологии *итерационной разработки*, когда используется *спиральная модель* жизненного цикла разработки. Разработка состоит из ряда этапов (*итераций*), которые в дальнейшем приводят к созданию информационной системы. Каждая итерация может приводить к созданию фрагмента или новой версии и включает все необходимые этапы: этапы выработки требований, анализа, проектирования, реализации и тестирования. Поскольку тестирование проводится на каждой итерации, риск снижается уже на начальных этапах жизненного цикла.

Указанные преимущества могут быть реализованы только при правильном проектировании систем в соответствии с принципами *инкапсуляции*, *наследования* и *полиформизма*.

Инкапсуляция – это, во-первых, процесс объединения в один объект *данных* и *действий* (*операций*), осуществляемых над ними, или, иначе говоря, *поведения* системы в процессе обработки этих данных. Во-вторых, *инкапсуляция* предполагает введение *ограничений* на последствия изменений, вносимых в систему. Эти ограничения вводятся за счет скрытия внутренней информации, когда доступ к объекту возможен только опосредованно – через его *операции* (характеризующие взаимодействие объекта с внешней средой) и *свойства* (атрибуты). Операции и свойства объекта представляют собой **интерфейс** объекта.

Наследование – это механизм, позволяющий создавать новые объекты, основываясь на уже существующих объектах. Наследование позволяет выделить *свойства*, *операции* и *события* одного объекта и приписать их другому объекту, иногда с модификацией. Порождаемый

объект-потомок (child) наследует свойства порождающего, *родительского* объекта (parent). Например: если «Автомобиль» – некий родительский объект, то «Легковой автомобиль» – объект-потомок, которому присущи все свойства объекта «Автомобиль».

Полиформизм – это возможность порождаемых объектов, создаваемых на основе родительских объектов, *изменять свою реакцию* на одни и те же воздействия при различных внешних условиях. Иными словами, полиформизм – это способность объектов выбирать операцию на основе данных, принимаемых в сообщении из внешнего мира или от другого объекта.

Полиформизм можно также определить как свойство некоторых объектов принимать различные внешние формы в зависимости от обстоятельств, т. е. действия, выполняемые одноименными методами, могут отличаться в зависимости от того, к какому классу относится тот или иной метод. Например, операция «Выключить» имеет разный смысл и результат, будучи применена к разным классам – «Автомобилю», «Свету в комнате» или «Компьютеру».

Рассмотренные особенности альтернативных подходов к созданию информационных систем представлены в табл. 1.

Фундаментальным понятием ООП является *Объект*, представляющий собой некую сущность реального мира или концептуальную сущность с четко определенными границами и значением для системы. Объект может быть чем-то конкретным, например *студент Иванов* или *кассовый аппарат № 4*, или концептуальным, например *банковская операция*, *торговый заказ*, *зачетная сессия* или *ставка прибыли*.

Каждый объект имеет три характеристики: *состояние*, *поведение* и *индивидуальность*.

Состояние – одно из условий, в котором объект может находиться, меняется во времени и определяется атрибутами и отношениями между объектами. Например, имеем систему регистрации успеваемости. Объект *Студенты группы № 1* в системе регистрации успеваемости может находиться в одном из двух состояний: *сдавали экзамен по информатике* или *не сдавали экзамен по информатике*. Если сдавали информатику, могут сдавать другой экзамен или могут быть переведены на следующий семестр, а если не сдавали, необходимо организовать экзамен по информатике.

Таблица 1

	Структурный подход		Объектно-ориентированный подход	
	Особенности	Следствия	Особенности	Следствия
Подход к разработке	Декомпозиция работы системы на автоматизируемые функции (функциональные подсистемы)	Целостное представление, в котором все компоненты взаимоувязаны в пространстве и времени	Описание системы и ее приложений на основе отдельных независимых фрагментов объектов , наделенных определенным поведением и реакцией на изменение внешних условий	Однократная разработка всех объектов с последующим многократным использованием для создания различных по назначению систем
ЖЦ	Последовательная реализация этапов ЖЦ	Риск обнаружения ошибок на этапе тестирования, который является предпоследним	Спиральная модель ЖЦ	Снижение риска за счет наличия этапа тестирования на каждой итерации ЖЦ
Ориентация	Ориентация на данные (data)	- Удобно разрабатывать	Ориентация на объектно-	Объединение данных и

	centric), обработку информации. Перенос приложения отработку реакции системы запросы (поведение) решение задач	на на на и	БД и системы обработки информации и отслеживать изменение данных. - Проблемы при разработке бизнес- приложений, поскольку при изменении требований к системе изменить деловые правила и поведение системы оказывается сложно	ориентированную парадигму, использующую принципы инкапсуляции, наследования и полиформизма	поведения, обрабатывающего эти данные, позволяет легко отслеживать как изменения данных, так и изменения в функционировании системы.
--	--	---------------------	---	---	---

Поведение определяет, как объект реагирует на запросы других объектов и что может делать сам объект. Поведение реализуется с помощью наборов *операций* для объекта. В системе регистрации успеваемости объект *Студенты группы № 1* может иметь операции *перевести на следующий курс* или *направить на экзамен*.

Индивидуальность означает, что каждый объект уникален, даже если его состояние аналогично состоянию другого объекта. Например, *студент Иванов* и *студент Петров* – два объекта в системе регистрации успеваемости. Хотя они оба являются *студентами группы № 1*, каждый из них уникален.

Класс – фундаментальное понятие ООП, представляющее собой **описание группы объектов** с общими **свойствами** (*атрибутами*), **поведением** (*операциями*), определяющим взаимодействие объектов с внешней средой, а также **отношениями** с другими объектами и **семантикой**. Т. о. класс – шаблон для создания объекта. Каждый объект является *экземпляром* конкретного класса и не может быть экземпляром нескольких классов. Например, класс *Студенты группы № 1* может определяться следующими характеристиками:

- атрибуты – время экзамена, место проведения экзамена;
- операции – получить время экзамена, получить аудиторию для проведения экзамена, сдавать экзамен.

Студент Петров и *студент Иванов* – это объекты, принадлежащие классу *Студенты группы № 1*. Каждый из них имеет значения атрибутов и доступ к операциям, определенным классом *Студенты группы № 1*.

В соответствии с традиционным подходом данные располагаются в базе данных, а поведением занимается собственно приложение. Объектно-ориентированный подход предполагает объединение некоторого количества данных и поведения, обрабатывающего эти данные. Мы берем немного данных и немного поведения, а затем инкапсулируем все это в некоторую структуру, называемую *классом*.

1.2. Функциональные модели деятельности

Наиболее трудоемкими этапами разработки ИС являются этапы анализа и проектирования, в процессе которых необходимо обеспечить качество принимаемых решений и подготовку проектной документации. При этом большую роль играют используемые методы представления информации. Применяемые в настоящее время графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями. Они предполагают построение всевозможных диаграмм в реальном масштабе времени и использование многообразной цветовой палитры.

При **структурном подходе** к разработке системы разрабатывается **функциональная модель**, рассматривающая систему как набор *действий*, в котором каждое действие преобразует некоторый *объект* или *набор объектов*. *Функциональная модель* – это описание бизнес-процессов системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы, определяющие *назначение модели*. Она представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе. Функциональные модели выделяют действия посредством представления в виде специального элемента – *блока*. *Блок* – основной структурный элемент функциональной модели, графическим представлением которой является *диаграмма*

В процессе функционального моделирования деятельности используются следующие правила:

- Необходимо применять принцип *функциональной декомпозиции* сложных бизнес-процессов, которая показывает разбиение одного процесса на ряд более мелких функций до тех пор, пока каждую из них уже нельзя будет разбить без ущерба для смысла. Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой. Конечный продукт декомпозиции представляет собой иерархию функций, на самом нижнем уровне которой находятся атомарные с точки зрения смысловой нагрузки функции.

Приведем простой пример (рис. 6.1) такой декомпозиции.



Рис. 6.1.

- Необходимо применять *стандартный язык моделирования* с определенным *синтаксисом* и разработанными строгими формализованными *правилами* построения моделей бизнес-процессов.

В процессе функционального моделирования помимо функциональной модели разрабатывается *информационная модель*, отражающая существующие информационные структуры и взаимосвязи между ними.

Совокупность функциональной и информационной моделей представляет собой *концептуальную модель* деятельности, которая отражает:

- особенности предметной области,
- характер решаемых задач,
- информационные потребности,
- ресурсы,
- технологические ограничения и т. п.

1.3. Визуальные модели деятельности

Визуальное моделирование (visual modeling) – это процесс графического представления модели с помощью некоторого стандартного набора графических элементов. Наличие стандарта жизненно необходимо для реализации одного из преимуществ визуального моделирования – коммуникации. Общение между пользователями, разработчиками, аналитиками, тестировщиками, менеджерами и всеми остальными участниками проекта является основной целью визуального моделирования.

Поскольку люди – зрительно-ориентированные существа, они лучше воспринимают работу системы в виде модели, чем на вербальном уровне или в виде текстового описания. Создавая визуальную модель системы, можно показать ее работу на различных уровнях, взаимодействие между пользователями и системой, взаимодействие объектов внутри системы и даже взаимодействие между системами, если это необходимо.

Глядя на высокоуровневую 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 поймут, как взаимодействуют друг с другом системы в их организации. Таким образом, визуальные модели представляют собой мощный инструмент, позволяющий показать разрабатываемую систему всем заинтересованным сторонам.

Важным вопросом является выбор *графической нотации*, которая должна быть понятна всем заинтересованным сторонам, иначе она не будет полезна. *Нотация* – совокупность графических объектов, которые используются в моделях. Она является синтаксисом языка моделирования. *Графическая нотация* – система обозначений, предназначенная для представления информации об отдельных аспектах моделируемой предметной области. Было разработано несколько методов моделирования, использовавших разные *нотации* (представления элементов модели). В современных технологиях широко используется метод (нотация) американского ученого Г. Буча (Grady Booch), который разработал нотацию графических символов для описания различных аспектов модели. Так, *компоненты реального мира* (объекты) в этой модели представляются в виде облаков. Такое представление является иллюстрацией того, что эти компоненты могут быть почти чем угодно. *Отношения* между объектами представляются различного вида стрелками. Более простая графика использована в технологии объектного моделирования (OMT, Object Modeling Technology). Нотация *OMT* разработана Дж. Рамбо (James Rumbaugh). Эту нотацию поддерживают многие современные промышленные инструменты моделирования программного обеспечения, в частности Rational Rose.

2. Логические и физические модели структур данных

Современные технологии имеют два уровня представления модели данных – *логический* и *физический*. *Логический уровень* – это абстрактный взгляд на данные. На этом уровне данные представляются так, как они выглядят в реальном мире, и могут называться так же, как они называются в реальном мире. Это представление непосредственно связано с исследуемыми бизнес-процессами.

Логическая модель данных описывает *факты* и *объекты*, подлежащие регистрации в будущей БД. Основные компоненты модели – это:

- сущности;
- их атрибуты;
- связи между ними.

Логическая модель представляет собой *ER-диаграмму* и является основой для проектирования системы. Она не связана с конкретной системой управления базой данных (СУБД), не учитывает физические особенности конкретных платформ, применяемых при последующей физической реализации, и поэтому она понятна даже неспециалистам.

Физическая модель данных, напротив, связана с реализацией схемы данных в конкретной СУБД. Физическая модель отражает компонентный состав проектируемой системы с точки зрения ее реализации на некоторой технической базе и вычислительных платформах конкретных производителей. В процессе построения физической модели данных следует определить:

- наименования таблиц;
- типы данных для всех полей.

Физическая модель данных должна быть обязательно нормализована.

3. Современные технологии и CASE-средства

3.1. IDEF-технология разработки информационных систем

Современный комплекс **IDEF-технологий** основан на стандарте 1960-х гг. **SADT** (Structured Analysis and Designer Technique). Он содержит:

- IDEF0 – технологию функционального моделирования, позволяющую документировать процесс производства и отображать информацию об использовании ресурсов на каждом этапе проектирования системы;
- IDEF1X – технологию моделирования структуры данных;
- IDEF2 – технологию динамического моделирования поведения системы во времени (не была полностью реализована);
- IDEF3 – технологию моделирования бизнес-процессов в системе;
- IDEF4 – технологию построения объектно-ориентированных систем;
- IDEF5 – технологию онтологического (принципиального, структурного) исследования системы.

С помощью простого и наглядного языка **IDEF0** разрабатываемая система предстает в виде совокупности функциональных моделей, удобных для подробного и тщательного изучения бизнес-процессов. Как правило, моделирование средствами **IDEF0** – **первый шаг** на этапе системного анализа жизненного цикла системы. Методы и средства этого шага определяют IDEF0-технологию разработки информационной системы.

Продолжение исследований приводит к **IDEF3**-технологии, которая используется для анализа процессов, происходящих в изучаемой системе. С помощью IDEF3-технологии описываются сценарий и последовательность операций для каждого процесса. Обычно IDEF0-модели связаны с IDEF3-сценариями, т. к. каждый блок IDEF0-модели может быть представлен в виде одного или нескольких IDEF3-сценариев.

IDEF1X-технология связана с построением реляционных структур данных. С помощью логической модели данных разрабатывается БД информационной системы и решаются задачи нормализации БД. На основе логической модели данных осуществляется физическое проектирование. С помощью физической модели реализуется реляционная модель данных в рамках используемой СУБД.

При построении этих моделей производится нормализация и, если нужно, денормализация модели данных, после чего документируются результаты разработки БД. Это первые шаги **системного синтеза** жизненного цикла системы. Затем обычно создаются приложения, представляющие собой интерфейс пользователя и предназначенные для решения следующих задач:

- пополнение нормализованной БД;
- просмотр данных в каждой из таблиц;
- построение запросов для пополнения базы данных, для просмотра данных в таблицах и выполнения вычислений.

Остальные технологии комплекса (**IDEF2, IDEF4, IDEF5**) менее популярны и не нашли широкого распространения при разработке информационных систем.

3.2. CASE-средства, основанные на IDEF-технологии

Термин **CASE** используется в настоящее время в весьма широком смысле. Первоначальное значение этого термина было ограничено вопросами автоматизации разработки только лишь программного обеспечения. В настоящее время с помощью CASE-средств можно описывать предметную область в виде формализованной модели, на основе которой генерируется код приложений. Современные **CASE-средства** охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ИС.

В настоящее время под термином **CASE-средства** понимаются любые *программные средства*, автоматизирующие ту или иную совокупность процессов ЖЦ и поддерживающие процессы создания и сопровождения ИС, включая:

- анализ и формулировку требований;
- проектирование прикладного ПО (приложений);
- проектирование БД;
- генерацию кода;
- тестирование;
- документирование;
- обеспечение качества;
- конфигурационное управление и управление проектом и др.

Основой CASE-технологии служит БД, хранящая не только информационные объекты, но и правила работы с ними. Поэтому CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС. В настоящее время рынок программных средств насчитывает около 300 различных CASE-средств, наиболее мощные из которых так или иначе используются всеми ведущими западными фирмами.

CASE-технология представляет собой методологию *проектирования ИС*, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения и разрабатывать приложения в соответствии с информационными потребностями пользователей.

CASE-средства предусматривают автоматизацию перехода от одного этапа разработки к следующему. Для этого предусмотрены специальные утилиты (обслуживающие программы), с помощью которых по модели предметной области можно автоматически получать описание структуры БД и состава программных модулей. По описанию структуры БД и состава программных модулей после всех необходимых уточнений и дополнений можно автоматически генерировать готовые к выполнению программы.

Рассмотрим CASE-средства, которые находят широкое применение при разработке информационных систем для создания как *моделей бизнес-процессов и баз данных*, так и *приложений*. Эти средства разработаны фирмой **PLATINUM technology**.

3.2.1. PLATINUM BPwin

BPwin (Business Process на базе Win) – это CASE-средство высокого уровня (Upper CASE), предназначенное для анализа и построения моделей предметной области и позволяющее разрабатывать:

- функциональные модели бизнес-процессов (*Business Process, IDEF0*);
- диаграммы сценариев, отображающих взаимодействие процессов в системе (*Process Flow, IDEF3*);
- диаграммы потоков данных (*Data Flow Diagrams, DFD*).

Функциональные модели строятся как для существующих бизнес-процессов (модель **AS - IS**), так и для бизнес-процессов, которые удовлетворяют требованиям оптимизации и модернизации (модель **TO - BE**). *Функциональные модели* строятся на основе использования *принципа декомпозиции* в виде иерархических диаграмм, которые от верхнего уровня – *контекстной диаграммы* – доходят до декомпозиционных диаграмм нижнего уровня (рис. 6.2).

Диаграммы сценариев описывают действия и события, которые должны быть обработаны за заданный промежуток времени. Сценарий может создаваться как самостоятельная модель или как часть модели бизнес-процесса, последовательность выполнения которого известна.

Сценарий сопровождается описанием процессов и может быть использован для документирования каждой функции системы. Следовательно, сценарии являются частью системного анализа, т. к. дают возможность проанализировать ситуацию во времени и описать объекты, участвующие в одном процессе одновременно.

Сценарий использует операции, представляющие собой единицы работы (Unit Of Work, UOF), ссылки на данные и перекрестки (Junctions).



Рис. 6.2

Диаграммы потоков данных, создаваемые с помощью методологии **DFD**, описывают обработку данных в системе. С их помощью можно получить наглядное представление о функциях обработки данных, документах, участвующие в обработке, внешних ссылках и хранилищах данных. Совокупность построенных диаграмм потоков данных создает модель обработки информации в системе.

Таким образом, **BRwin** позволяет создать полезную документацию разрабатываемой системы, согласование которой с заказчиком существенным образом сокращает вероятность рисков.

На основе **BRwin** нельзя сгенерировать код приложений и программ, поскольку способы представления модели не предусматривают разработку БД, а являются *языком моделирования* и служат только для представления возможности общения различных специалистов.

3.2.2. PLATINUM ERwin

ERwin (Entity Relationship на базе Win) – это CASE-средство проектирования баз данных, обеспечивающее моделирование данных и генерацию схем баз данных (как правило на языке SQL) для наиболее распространенных СУБД.

Средствами **ERwin** строятся *логическая* и *физическая* модели данных ИС с учетом используемой системы управления базой данных (технология **IDEF1X**) На первом этапе построения модели строится *Диаграмма сущность-связь (Entity Relationship Diagram, ERD)*, которая представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована. Эта диаграмма может включать связи МНОГИЕ КО МНОГИМ и не включать описание ключей. Она используется для презентаций и обсуждения структуры данных с экспертами предметной области.

На следующих этапах строится *модель данных, основанная на ключах*, она включает описание всех сущностей и ключей.

На окончательном этапе строится *полная атрибутивная информационная модель* – наиболее детальное представление структуры данных: данные представлены в третьей нормальной форме, и в модель включены все сущности, атрибуты и связи.

На этапе физического моделирования строится *физическая* модель данных, ориентированная на конкретную СУБД, в которой зафиксированы таблицы, связи между ними и типы данных.

На основе созданной модели данных автоматически может быть сгенерирован код клиентского приложения, представляющий собой программу, отвечающую за интерфейс с пользователем (преобразует его запросы в команды запросов к серверной части, а также производит обратное преобразование: результаты выполнения команд преобразуются к виду, воспринимаемому пользователем).

Код приложения генерируется путем использования редактора схем БД и соответствующих программных средств, с которыми ERwin интегрирован (Power Builder, Visual Basic, Delphi).

3.3. RUP-технология разработки информационных систем

RUP-технология (**R**ational **U**nified **P**rocess) основана на использовании унифицированного процесса разработки информационной системы. Технология RUP структурирована в двух направлениях:

- время (разделение жизненного цикла системы на фазы и версии);
- компоненты процесса (набор средств для решения определенных задач).

Разработка системы с использованием RUP-технологии состоит из следующих временных этапов:

- *Задание* – определение общей задачи системы.
- *Проработка* – планирование необходимых работ и ресурсов.
- *Создание* – построение системы.
- *Переходный период* – поставка системы пользователю.

С точки зрения компонентов процесс разработки делится на следующие стадии:

- Построение бизнес-модели.
- Определение требований к системе.
- Анализ и проектирование.
- Реализация и внедрение

RUP-технология предполагает построение пяти представлений разрабатываемой системы:

1. **Представление использования** – основная часть модели описания системы, с помощью которой дается характеристика функций, выполняемых системой (аспектов использования) с точки зрения пользователей (субъектов).
2. **Логическое** представление – статическая часть модели описания системы, с помощью которой дается характеристика данных с точки зрения распределения их между связанными системными объектами.
3. **Компонентное** представление – описание структуры и взаимосвязей модулей реализации системы.
4. Представление **взаимодействия процессов** – описание согласованных действий, относящихся к взаимодействию и синхронизации отдельных компонентов системы.
5. Представление **распределения** – описание физической архитектуры системы, распределения ее компонентов в вычислительной сети.

Все вместе они позволяют легко и просто выполнить реализацию и внедрение системы. На основе этих представлений генерируется программный код и код приложения на языках программирования, которые при трансляции подвергаются процессу *компилирования* и преобразуются в исполняемые *exe-файлы*. В результате создаются приложения информационной системы для следующих программных сред: C ++, Visual Basic, Power Builder, Java и др.

В RUP-технологии активно используется универсальный язык моделирования **UML** (**U**nified **M**odeling **L**anguage). Язык прошел процесс стандартизации в рамках консорциума **OMG** (**O**bject **M**anagement **G**roup) и сейчас является международным стандартом.

3.4. CASE-средство Rational Rose и представления информационной системы на языке UML

RUP-технология используется в CASE-средстве **Rational Rose**. **Rational Rose** представляет собой программный пакет для визуального объектно-ориентированного моделирования систем на основе классов и их взаимодействия. Можно также сказать, что это визуальный редактор,

позволяющий создавать программные системы любой сложности на основе графических диаграмм языка UML.

Универсальный язык моделирования (UML) (создан в 1997 г.) – это результат совместных усилий разных разработчиков. В частности, для того, чтобы доступно и грамотно описать модель системы, И. Якобсоном был описан процесс выявления и фиксации требований к системе *в виде совокупности транзакций*, называемых *«use case»* (*case – случай, прецедент, вариант*). С помощью языка UML решается задача разработки некоторой предварительной модели программной системы, которая была бы понятна и заказчику и группе программистов. Он служит для определения, отображения и описания элементов объектно-ориентированных систем в процессе их создания. UML является стандартом в области объектно-ориентированного анализа и проектирования систем.

Язык UML представляет собой набор графических диаграмм, которые позволяют проектировать и создавать сложные программные системы. Он характеризуется следующими особенностями:

1. позволяет создавать несколько типов визуальных диаграмм;
2. позволяет моделировать не только программное обеспечение, но и более широкие классы систем с использованием объектно-ориентированных понятий;
3. обеспечивает взаимосвязь между базовыми понятиями для моделей концептуального и физического уровней;
4. обеспечивает масштабируемость моделей, что является важной особенностью сложных многоцелевых систем;
5. понятен аналитикам и программистам, а также поддерживается специальными инструментальными средствами, реализованными на различных компьютерных платформах.

UML позволяет создавать несколько типов визуальных диаграмм. Rational Rose поддерживает разработку большинства этих моделей, а именно:

- диаграмм вариантов использования (прецедентов) (use case diagrams), используемых для моделирования бизнес-процессов, и требований к создаваемой системе;
- диаграмм классов (class diagrams), используемых для моделирования статической структуры классов системы и связей между ними;
- диаграмм поведения системы (behavior diagrams):
 - диаграмм взаимодействия (interaction diagrams):
 - диаграмм последовательности (sequence diagrams), отражающих поток событий, происходящий в рамках прецедента;
 - кооперативных диаграмм (collaboration diagrams) – для моделирования процесса обмена сообщениями между объектами;
 - диаграмм состояний (statechart diagrams) – для моделирования поведения объектов системы при переходе из одного состояния в другое;
 - диаграмм деятельности (activity diagrams) – для моделирования поведения системы в рамках различных вариантов использования или моделирования деятельности;
- диаграмм реализации (implementation diagrams):
 - диаграмм компонентов (component diagrams) – для моделирования иерархии компонентов (подсистем) системы;
 - диаграмм размещения (deployment diagrams) – для моделирования физической архитектуры системы.

4. ATS-технология разработки информационных систем

Автоматизированный табличный сценарий (ATS) позволяет получить описание информационной системы в виде совокупности таблиц и диаграмм (частный случай – таблиц). В результате чего имеется полный набор моделей системы.

Применительно к ИС *сценарий* создания ИС – это *формализованный способ описания* сложной информационной системы (существующей или разрабатываемой). *Табличное описание сценария* – это метод описания сценария, позволяющий автоматизировать создание сценария на

основе системы управления БД как инструмента автоматизированных информационных технологий

Подход к описанию системы с помощью сценария предполагает необходимость проведения анализа действующей системы на загрузку и производительность, с предложением последующего ее улучшения. Предположим, в банке имеют место жалобы на нечеткое составление документации, медленное ее оформление, нечеткую работу подразделения, в результате чего из банка уходят клиенты. Банк обращается к специалистам и просит сделать что-нибудь во избежание этих проблем. Для этого с использованием ATS-технологии специалистами составляется *описание системы*, в результате которого автоматически создается ряд моделей системы, в том числе *Процедурная модель*. Анализируя процедурную модель, можно найти «узкие» места и порекомендовать, каким образом можно исправить положение. Поскольку все модели системы оформляются в виде таблиц, автоматизация разработки осуществляется с помощью систем управления БД.

Достоинством ATS-технологии является то, что в ранее рассмотренных технологиях функциональные и процедурные модели системы приходится разрабатывать и строить *вручную*, пользуясь графическими средствами описания, а с помощью ATS они создаются *автоматически* после преобразований таблиц методами реляционной алгебры.

К недостаткам табличных сценариев по сравнению с описанными ранее IDEF- и RUP-технологиями следует отнести их следующие особенности:

- потеря наглядности, т. к. отсутствуют графические образы,
- наличие трудностей с документированием моделей,
- разработка имеет характер интерпретации в среде выбранной системы управления БД,
- необходимость специальной подготовки разработчика в области БД.

На основании вышеизложенного можно провести сравнительный анализ CASE-средств, используемых при разработке ИС, исходя из их возможностей и степени автоматизации разработки приложений. Результаты такого анализа приведены в табл. 2.

Таблица 2

	BPwin	ERwin	Rational Rose	ATS
Технология	IDEF0, IDEF3, DFD	IDEF1X	RUP	Реляционная алгебра, СУБД-технология
Модель ИС	Функциональная модель деятельности (процессов)	Модель данных	Объектная модель	Модель деятельности
Способ создания моделей	Вручную разработчиком	Вручную разработчиком	Вручную разработчиком	Автоматически
Представления	<ul style="list-style-type: none"> • Диаграммы декомпозиции • Диаграммы дерева узлов • Стоимостной анализ • Диаграммы потоков данных • Сценарий • <i>диаграмма последовательности этапов,</i> • <i>диаграмма</i> 	<ul style="list-style-type: none"> • Логическая модель (ER-диаграмма) • Физическая модель (Реляционная БД) 	Представление (использования): <ul style="list-style-type: none"> • <i>диаграмма прецедентов</i> • <i>диаграмма последовательности</i> • Логическое представление: • <i>диаграмма классов</i> • Компонентное представление 	Табличный сценарий: <ul style="list-style-type: none"> • <i>функциональная модель,</i> • <i>структурная модель,</i> • <i>процедурная модель</i>

	<i>состояний</i>		<ul style="list-style-type: none"> • Представление взаимодействия процессов • Представление распределения 	
Язык	Язык текстографических представлений – SA-язык	Язык текстографических схем	Язык текстографических представлений – UML	Язык представления с помощью таблиц
Создание базы данных	Полуавтоматически (для экспорта в ERwin)	Автоматически	Автоматически	Вручную
Генерация кода приложения	–	Автоматически	Автоматически	Вручную

5. Выводы

1. В процессе создания ИС разработчиками используются структурный и объектно-ориентированный подходы.
2. Структурный подход к разработке ИС основан на использовании принципа функциональной декомпозиции.
3. При структурном подходе основное внимание уделяется информации, с которой работает система, поэтому он ориентирован на данные.
4. Если код приложения генерируется не на основе описания предметной области, а на основе БД, невозможно построить эффективное приложение со сложной бизнес-логикой.
5. При структурном подходе имеется большая вероятность обнаружения ошибок на последнем этапе разработки.
6. Объектно-ориентированные методы разработки ИС уделяют внимание не только информации, но и поведению системы.
7. Снижение риска в объектно-ориентированной технологии достигается за счет использования спиральной модели ЖЦ.
8. Объектно-ориентированные методы разработки ИС основаны на принципах инкапсуляции, наследования и полиморфизма.
9. Фундаментальными понятиями ООП являются объект и класс.
10. Каждый объект характеризуется состоянием, поведением и индивидуальностью.
11. Функциональное моделирование, основанное на структурном подходе к разработке ИС, использует IDEF-технологиию.
12. Функциональная модель – это модель бизнес-процессов, реализуемых в системе.
13. Информационная модель – это модель данных, обрабатываемых в системе.
14. Визуальное моделирование, основанное на объектно-ориентированном подходе к разработке ИС, использует RUP-технологиию.
15. IDEF-технология поддерживается CASE-средствами BPwin и ERwin.
16. RUP-технология поддерживается CASE-средством Rational Rose.
17. Основными диаграммами, разрабатываемыми в Rational Rose, являются диаграммы прецедентов, классов и взаимодействия и последовательности.
18. Технология ATS позволяет автоматизировать разработку моделей.

6. Контрольные вопросы

1. На чем основывается структурный подход к разработке ИС?
2. В чем заключаются достоинства и недостатки структурного подхода?
3. Назовите основное отличие объектно-ориентированного подхода от структурного подхода к разработке ИС.
4. Определите понятие «объектно-ориентированная парадигма», используемого в ООП.

5. Дайте определение:
 - объекта и класса, используемых в ООП.
 - понятия «наследование».
 - понятия «инкапсуляция».
 - понятия «полиформизм».
6. За счет чего наблюдается снижение риска в объектно-ориентированной технологии?
7. Какие характеристики имеют объекты в ООП?
8. Что означает операция декомпозиции модели?
9. Дайте определение функциональной и информационной моделям.
10. Что представляет собой концептуальная модель?
11. Назовите особенности визуального моделирования.
12. Что такое нотация?
13. Каковы различия между логической и физической моделями данных?
14. Какие требования предъявляются к физической модели данных?
15. Какая технология используется для моделирования бизнес-процессов?
16. Какая технология используется для моделирования данных ИС?
17. Какая технология используется для разработки диаграммы сценария в VPwin?
18. Какая методология используется для разработки диаграммы потоков данных?
19. Какие CASE-средства позволяют генерировать коды приложений?
20. Назовите представления ИС, разрабатываемые в RUP-технологии.
21. Назовите диаграммы, разрабатываемые с помощью языка UML?
22. Для чего создаются:
 - диаграмма прецедентов?
 - диаграмма классов?
 - диаграмма последовательности?