УВАЖАЕМЫЕ СТУДЕНТЫ! Законспектируйте в своей рабочей тетради по дисциплине приведенную лекцию (объемом 4-5 страницы), ответьте *письменно* на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 23.01.2023.

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 30
Тема: Технологии программирования.

1 ТЕХНОЛОГИЯ DATARUN

Одной из наиболее распространенных в мире электронных технологий является технология DATARUN. В соответствии с этой технологией ЖЦ ПО разбивается на стадии, которые связываются с результатами выполнения основных процессов, определяемых стандартом ISO/IEC 12207..

Стадия формирования требований и планирования включает в себя действия по определению начальных оценок объема и стоимости проекта. Должны быть сформулированы требования и экономическое обоснование для разработки ЭИС, построены функциональные модели (модели деятельности организации) и исходная концептуальная модель данных, которые дают основу для оценки технической реализуемости проекта. Основными результатами этой стадии должны быть модели деятельности организации (исходные модели процессов и данных организации) и требования к системе, включая требования по сопряжению с существующими ЭИС. Каждую стадию должен завершать план работ на следующую стадию.

Стадия концептуального проектирования начинается с детального анализа первичных данных и уточнения концептуальной модели данных, после чего проектируется архитектура системы. Оценивается возможность использования существующего ПО и выбирается соответствующий метод его преобразования. После построения проекта уточняется исходный план. Результатами этой стадии являются концептуальная модель данных, модель архитектуры системы и уточненный план.

На *стадии спецификации приложений* продолжается процесс создания и детализации проекта. Концептуальная модель данных преобразуется в реляционную модель данных. Определяются структура приложения, необходимые интерфейсы приложения в виде экранов, отчетов и пакетных процессов вместе с логикой их вызова. Модель данных уточняется бизнес-правилами (ограничениями целостности) и методами для каждой таблицы. В

конце этой стадии принимается окончательное решение о способе реализации приложений. По результатам стадии должен быть построен проект ЭИС, включающий модели архитектуры ПО ЭИС, данных, функций, интерфейсов (с внешними системами и с пользователями), требований к разрабатываемым приложениям (модели данных, интерфейсов и функций), требований к доработкам существующего ПО, требований к интеграции приложений, а также сформирован окончательный план создания ЭИС.

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

Стадия внедрения охватывает действия по установке и внедрению баз данных и приложений. Основными результатами стадии должны быть готовая к эксплуатации и перенесенная на программно-аппаратную платформу заказчика версия системы, документация сопровождения и акт приемочных испытаний по результатам опытной эксплуатации.

Стадии сопровождения и развития включают процессы и операции, связанные с регистрацией, диагностикой и локализацией ошибок, внесением изменений и тестированием, проведением доработок, тиражированием и распространением новых версий ПО в места его эксплуатации, переносом приложений на новую платформу и масштабированием системы. Стадия развития фактически является повторной итерацией стадии разработки.

Технология DATARUN опирается на две модели или на два представления *(см.* главу 1):

• *модель деятельности организации;*

• *модель проектируемой ЭИС.*

Технология DATARUN базируется на системном подходе к описанию деятельности организации. Построение моделей начинается с описания процессов, из которых затем извлекаются первичные данные (стабильное подмножество данных, которые организация должна использовать для своей деятельности). Первичные данные описывают продукты или услуги организации, выполняемые операции (транзакции) и потребляемые ресурсы. К первичным относятся данные, которые описывают внешние и внутренние сущности, такие, как служащие, клиенты или агентства, а также данные, полученные в результате принятия решений, например графики работ, цены на продукты.

Основной принцип DATARUN заключается в том, что первичные данные, если они должным образом организованы в модель данных, становятся основой для проектирования архитектуры системы. Архитектура будет достаточно стабильной, если она основана на первичных данных, тесно связанных с основными бизнес-процессами.

Подход DATARUN преследует две цели:

• определить стабильную структуру, на основе которой будет строиться ЭИС. Такой структурой является модель данных, полученная на основе первичных данных, используемых фундаментальными процессами организации;

• спроектировать ЭИС на основе определенной структуры. Объекты, формируемые на основе модели данных, являются объектами базы данных, обычно размещаемыми на серверах в среде "клиент-сервер". Объекты интерфейса, определенные в архитектуре компьютерной системы, обычно размещаются на клиентской части.

Модель данных, являющаяся основой для спецификации совместно используемых объектов базы данных и различных объектов интерфейса, обеспечивает сопровождаемость ЭИС. На рис. 5.1 представлена последовательность шагов проектирования ЭИС.

Рис. 1. Последовательность шагов проектирования системы

На рис. 5.2 определен комплекс моделей, создаваемых в процессе разработки ЭИС. В его состав входят следующие модели:

• Business Process Model (BPM) — модель бизнес-процессов;

• Primary Data Structure (PDS) — структура первичных данных;

• Conceptual Data Model (CDM) — концептуальная модель данных;

• System Process Model (SPM) — модель процессов системы;

• Information System Architecture (ISA) — архитектура информационной

системы;

- Application Data Model (ADM) — модель данных приложения;
- Interface Presentation Model (IPM) - модель представления интерфейса;
- Interface Specification Model (ISM) — модель спецификации интерфейса.

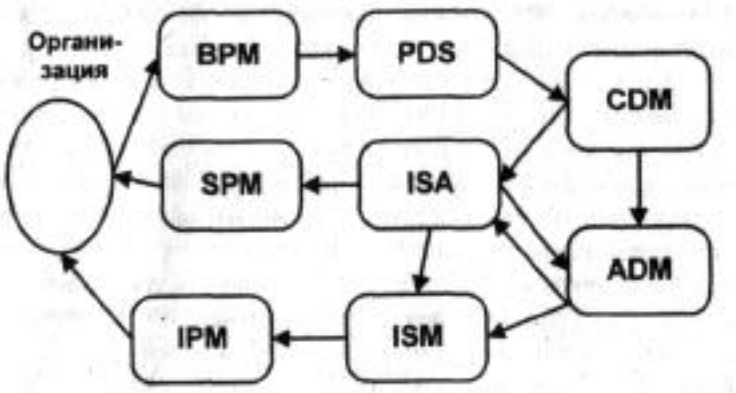

Рис. 2. Модели, создаваемые по технологии DATARUN

Для создания моделей используется CASE-средство Silverrun обеспечивает автоматизацию проведения проектных работ в соответствии с технологией DATARUN. Предоставляемая этим средством среда проектирования дает возможность руководителю проекта контролировать проведение работ. Каждый участник проекта, подключившись к этой среде, может выяснить содержание и сроки выполнения порученной ему работы, детально изучить технику ее реализации в гипертексте по технологиям и вызвать инструмент (модуль Silverrun) для реального выполнения работы.

Информационная система создается последовательным построением ряда моделей, начиная с модели бизнес-процессов и заканчивая моделью ПО, автоматизирующего эти процессы.

Создаваемая ЭИС должна основываться на функциях, выполняемых организацией. Поэтому первая создаваемая модель — это модель бизнес-процессов, построение которой осуществляется с помощью модуля Silverrun BPM. Для этой модели используется специальная нотация. В процессе анализа и спецификации бизнес-функций выявляются основные информационные объекты, которые документируются как структуры данных, связанные с потоками и хранилищами модели. Источниками для создания структур служат используемые в организации документы, должностные инструкции, описания производственных операций. Эти данные вводятся в том виде, в каком они существуют в организации. Нормализация и удаление избыточности производятся позже при построении концептуальной модели данных в модуле Silverrun ERX. После создания модели бизнес-процессов информация сохраняется в репозитории проекта.

В процессе обследования работы организации выявляются и документируются структуры первичных данных. Эти структуры заносятся в репозитории модуля BPM при описании циркулирующих в организации документов, сообщений, данных. В модели бизнес-процессов первичные структуры данных связаны с потоками и хранилищами информации.

На основе структур первичных данных в модуле Silverrun ERX создается

концептуальная модель данных. От структур первичных данных концептуальная модель отличается удалением избыточности, стандартизацией наименований понятий и нормализацией. Эти операции в модуле ERX выполняются с помощью встроенной экспертной системы. Цель концептуальной модели данных - описать используемую информацию без деталей возможной реализации в базе данных, но в хорошо структурированном нормализованном виде.

На основе модели бизнес-процессов и концептуальной модели данных проектируется архитектура ЭИС. Определяются входящие в систему приложения, для каждого приложения специфицируются используемые данные и реализуемые функции. Архитектура ЭИС создается в модуле Silverrun BPM с использованием специальной нотации ISA. Основное содержание этой модели — структурные компоненты системы и навигация между ними. Концептуальная модель данных разбивается на части, соответствующие входящим в состав системы приложениям.

Перед разработкой приложений должна быть спроектирована структура корпоративной базы данных. Технология DATARUN предполагает использование базы данных, основанной на реляционной модели. Концептуальная модель данных после нормализации переносится в модуль реляционного моделирования Silverrun RDM с помощью специального моста ERX-RDM. Преобразование модели из формата ERX в формат RDM происходит автоматически без вмешательства пользователя. После преобразования ERX-RDM получается модель реляционной базы данных. Эта модель детализируется в модуле Silverrun RDM определением физической реализации (типов данных СУБД, ключей, индексов, триггеров, ограничений ссылочной целостности). Правила обработки данных можно задавать как непосредственно на языке программирования СУБД, так и в декларативной форме, не привязанной к реализации. Мосты Silverrun к реляционным СУБД переводят эти декларативные правила на язык требуемой системы, что снижает трудоемкость программирования процедур сервера базы данных, а также позволяет из одной спецификации генерировать приложения для разных СУБД.

С помощью модели системных процессов детально документируется поведение каждого приложения. В модуле BPM создается модель системных процессов, определяющая, каким образом реализуются бизнес-процессы. Эта модель создается отдельно для каждого приложения и тесно связана с моделью данных приложения.

Приложение состоит из интерфейсных объектов (экранных форм, отчетов, процедур обработки данных). Каждый интерфейсный объект связан с некоторым подмножеством базы данных. В модели данных приложения формируется подсхема базы данных для каждого интерфейсного объекта этого приложения. Уточняются также правила обработки данных, специфичные для каждого интерфейса. Интерфейс работает с данными в ненормализованном виде, поэтому спецификация данных, как ее видит интерфейс, оформляется в виде отдельной подсхемы модели данных интерфейса.

Модель представления интерфейса — это описание внешнего вида интерфейса, как его видит конечный пользователь системы. Это может быть и документ, показывающий внешний вид экрана или

структуру отчета, и сам экран (отчет), созданный с помощью одного из

средств визуальной разработки приложений — так называемых языков четвертого поколения (4GL — Fourth Generation Languages). Так как большинство языков 4GL позволяют быстро проектировать работающие прототипы приложений, пользователь имеет возможность увидеть работающий прототип системы на ранних стадиях проектирования.

После создания подсхем реляционной модели для приложений проектируется детальная структура каждого приложения в виде схемы навигации экранов, отчетов, процедур пакетной обработки. Эта структура детализируется до указания конкретных столбцов и таблиц базы данных, правил их обработки, вида экранных форм и отчетов. Полученная модель детально документирует приложение и непосредственно используется для программирования специфицированных интерфейсов.

Далее средствами разработки приложений происходит физическое создание системы: приложения программируются и интегрируются в информационную систему.

Электронный вариант технологии DATARUN реализован с помощью инструментального средства SE (Software Engineering) Companion. Оно позволяет:

• создать гипертекстовое описание технологии в виде иерархии описания стадий, этапов и операций разработки;

• создать гипертекстовое описание всех методов и методик реализации процессов ЖЦ ПО;

• выделить из гипертекстового описания иерархию процессов ЖЦ ПО для планирования и управления процессом создания ПО (иерархию работ);

• изменять гипертекстовые описания ЖЦ и методов так, как это необходимо разработчику, иными словами, производить авторизацию технологии и отслеживать эти изменения в иерархии работ, предназначенной для управления проектом;

• привязать к процессам ЖЦ инструментальные средства поддержки этих процессов и обеспечить вызов инструментальных средств из соответствующих экранов гипертекстового справочника;

• обеспечить просмотр гипертекстовых экранов описания используемых методов с помощью инструментальных средств;

• обеспечить поддержку процесса управления разработкой, в частности, за счет взаимодействия со средством планирования работ Microsoft Project, оценивания трудоемкости проекта, отслеживания хода работ, рисования графиков работ и др. Особенно важными являются возможность авторизации технологии и интерактивный доступ любого разработчика к описанию любого метода или процесса в нужный ему момент времени. В условиях быстрого изменения как программных и аппаратных средств, так и задач бизнеса технология создания, сопровождения и развития ПО не должна быть неизменной; она должна иметь возможность изменяться и настраиваться на новые методы и инструментальные средства. Таким образом, разработчики ПО приобретают одну или несколько технологий поставщика, а затем создают на их основе собственные технологии, адаптированные к конкретным условиям. Гипертекстовое описание технологии создания ПО строится из описания процессов жизненного цикла, методов и методик и представляет собой единый гипертекстовый документ в формате Microsoft Help. Итоговое гипертекстовое описание получается в результате

трансляции исходного текстового документа. Все изменения и дополнения технологии производятся посредством корректировки исходного документа.

Описание технологии создания системы состоит из раздела описания процессов ЖЦ и разделов описания методов и методик. В свою очередь, раздел описаний процессов состоит из иерархии описаний стадий, этапов и операций жизненного цикла с обязательным описанием выходных компонентов каждого процесса. Компоненты ПО создаются с применением методов и методик, описываемых в соответствующих разделах.

2 ТЕХНОЛОГИЯ RUP

Технология RUP (Rational Unified Process) разработана компанией Rational Software. Она ориентирована на использование универсального языка объектно-ориентированного моделирования UML, являющегося фактическим стандартом в данной области *(см.* главу 3).

На рис. 5.3 показан общий вид процесса RUP в двух измерениях:

• горизонтальное измерение представляет время, отражает динамические аспекты процессов и оперирует такими понятиями, как циклы, фазы, итерации и контрольные точки;

• вертикальное измерение отражает статические аспекты процессов и оперирует такими понятиями, как действия, результаты деятельности, исполнители и рабочие процессы.

Динамический аспект

Согласно технологии RUP жизненный цикл ПО разбивается на отдельные циклы, в каждом из которых создается новое поколение продукта. Каждый цикл, в свою очередь, разбивается на четыре последовательные стадии:

• начальная стадия (inception);
• стадия уточнения (elaboration);
• стадия конструирования (construction);
• стадия ввода в действие (transition).

Каждая стадия завершается в четко определенной контрольной точке (milestone). В этот момент времени должны достигаться важные результаты и приниматься критически важные решения о дальнейшей разработке.

Первыми двумя стадиями являются начальная стадия проекта и уточнение.

Начальная стадия. Она может принимать множество разных форм. Для крупных проектов начальная стадия может вылиться во всестороннее изучение всех возможностей реализации проекта, которое займет месяцы. Во время начальной стадии вырабатывается бизнес-план проекта — определяется, сколько приблизительно он будет стоить и какой доход принесет. Устанавливаются также границы проекта, и выполняется некоторый начальный анализ для оценки размеров проекта.

Для того чтобы проделать такую работу, необходимо идентифицировать все внешние сущности (действующие лица), с которыми система будет взаимодействовать, и определить в самом общем виде природу этого взаимодействия. Это подразумевает идентификацию всех вариантов использования (use case, *см.* главу 3) и описание наиболее важных из них. Бизнес-план включает критерии успеха, оценку риска, оценку необходимых ресурсов и

общий план по стадиям, включающий даты основных контрольных точек.

Результатами начальной стадии являются:

• общее описание системы (основные требования к проекту, его характеристики и ограничения);

• начальная диаграмма вариантов использования (степень готовности - 10 - 20%);

• начальный проектный глоссарий (словарь терминов);

• начальный бизнес-план;

• план проекта, отражающий стадии и итерации;

• один прототип или несколько.

Рис. 3. Общий вид процесса RUP

Стадия уточнения. На этой стадии выявляются более детальные требования к системе, выполняются высокоуровневый анализ предметной области и проектирование для построения базовой архитектуры системы, создается план конструирования и устраняются наиболее рискованные элементы проекта.

Результатами стадии уточнения являются:

• диаграмма вариантов использования (завершенная по крайней мере на 80%), определяющих требования к системе;

• перечень дополнительных требований, включая требования нефункционального характера и требования, не связанные с конкретными вариантами использования;

• описание базовой архитектуры будущей системы;

• работающий прототип;

• уточненный бизнес-план;

• план разработки всего проекта, отражающий итерации и критерии оценки

для каждой итерации.

Самым важным результатом стадии уточнения является описание базовой архитектуры будущей системы. Эта архитектура включает:

• модель предметной области, которая отражает понимание бизнеса и служит отправным пунктом для формирования основных классов предметной области;

• технологическую платформу, определяющую основные элементы технологии реализации системы и их взаимодействие.

Базовая архитектура является основой всей дальнейшей разработки, она служит своего рода проектом для последующих стадий. В дальнейшем неизбежны незначительные изменения в деталях архитектуры, однако серьезные изменения маловероятны.

Стадия уточнения занимает около пятой части общей продолжительности проекта. Основными признаками завершения этой стадии являются два события:

• разработчики в состоянии оценить с достаточно высокой точностью, сколько времени потребуется на реализацию каждого варианта использования;

• идентифицированы все наиболее серьезные риски, и степень понимания наиболее важных из них такова, что известно, как справиться с ними.

Сущность планирования заключается в определении последовательности итераций конструирования и вариантов использования, реализуемых на каждой итерации.

Планирование завершается, когда определены место каждого варианта использования в некоторой итерации и дата начала каждой итерации. На данном этапе более детальное планирование не делается.

Стадия конструирования. RUP представляет собой итеративный и пошаговый процесс разработки, в котором программное обеспечение разрабатывается и реализуется по частям. На стадии конструирования построение системы выполняется путем серии итераций. Каждая итерация является своего рода мини-проектом. На каждой итерации для конкретных вариантов использования выполняются анализ, проектирование, кодирование, тестирование и интеграция. Итерация завершается демонстрацией результатов пользователям и выполнением системных тестов в целях контроля корректности реализации вариантов использования.

Назначение этого процесса состоит в снижении степени риска. Причиной появления риска зачастую является откладывание решения сложных проблем на самый конец проекта. Тестирование и интеграция - это достаточно крупные задачи, они всегда занимают больше времени, чем ожидается. Чем позже выполнять тестирование и интеграцию, тем более трудными задачами они становятся и тем более способны дезорганизовать весь проект. При итеративной разработке на каждой итерации выполняется весь процесс, что позволяет оперативно справляться со всеми возникающими проблемами.

Итерации на фазе конструирования являются одновременно инкрементными и повторяющимися. Итерации являются *инкрементными* в соответствии с той функцией, которую они выполняют. Каждая итерация добавляет очередные конструкции к вариантам использования, реализованным во время предыдущих итераций. Итерации являются *повторяющимися* по отношению к разрабатываемому коду. На каждой итерации некоторая часть существующего кода переписывается с целью сделать его более гибким.

Процесс интеграции должен быть непрерывным. В конце каждой итерации должна выполняться полная интеграция. С другой стороны, интеграция может и должна выполняться еще чаще. Приложения следует интегрировать после выполнения каждой сколько-нибудь значительной части работы. Во время каждой интеграции должен выполняться полный набор тестов.

Главная особенность итеративной разработки заключается в том, что она жестко ограничена временными рамками, и сдвигать сроки

недопустимо. Исключением может быть перенос реализации каких-либо вариантов использования на более позднюю итерацию по соглашению с заказчиком. Смысл таких ограничений — поддерживать строгую дисциплину разработки и не допускать переноса сроков.

При этом если на более поздний срок перенесено слишком много вариантов использования, то необходимо корректировать план, пересмотрев при этом оценку трудоемкости реализации вариантов использования. На данной стадии разработчики должны иметь более глубокое представление о такой оценке.

Помимо конструирования итерации могут присутствовать во всех стадиях, однако при этом конструирование является ключевой стадией.

Результатом стадии конструирования является продукт, готовый к передаче конечным пользователям. Как минимум, он содержит следующее:

• ПО, интегрированное на требуемых платформах;

• руководства пользователя;

• описание текущей реализации.

Стадия ввода в действие. Назначение этой стадии - передача готового продукта в распоряжение пользователей. Данная стадия включает:

• бета-тестирование, позволяющее убедиться, что новая система соответствует ожиданиям пользователей;

• параллельное функционирование с существующей (legacy) системой, которая подлежит постепенной замене;

• конвертирование баз данных;

• оптимизацию производительности;

• обучение пользователей и специалистов службы сопровождения.

Главная идея итеративной разработки — поставить весь процесс разработки на регулярную основу с тем, чтобы команда разработчиков смогла получить конечный продукт. Однако есть некоторые процессы, которые не следует выполнять слишком рано, например оптимизация.

Оптимизация снижает прозрачность и расширяемость системы, однако повышает ее производительность. В этой ситуации необходимо принятие компромиссного решения, поскольку система должна быть достаточно производительной, чтобы удовлетворять пользовательским требованиям. Слишком ранняя оптимизация затруднит последующую разработку, поэтому ее следует выполнять в последнюю очередь.

На стадии ввода в действие продукт не дополняется никакой новой функциональностью (кроме самой минимальной и абсолютно необходимой). Здесь только вылавливаются ошибки. Хорошим примером для стадии ввода в действие может служить период времени между выпуском бета-версии и окончательной версии продукта.

Статический аспект

Статический аспект RUP характеризуют четыре основных элемента:

• исполнители;

• действия;

• результаты деятельности;

• рабочие процессы.

Понятие *"исполнитель"* определяет поведение и ответственность личности или группы личностей, составляющих проектную команду. По существу, это понятие представляет собой роль, причем одна личность может играть в проекте много различных ролей.

Под *действием* конкретного исполнителя понимается единица выполняемой им работы. Действие имеет четко определенную цель, обычно выражаемую в терминах получения или модификации некоторых *результатов деятельности,* таких, как модель, элемент модели, документ, исходный код или план. Каждое действие связано с конкретным исполнителем. Продолжительность действия составляет от нескольких часов до нескольких дней. Оно обычно выполняется одним исполнителем и порождает только один результат или весьма небольшое их количество. Любое действие должно являться элементом процесса планирования. Примерами действий могут быть планирование итерации, определение вариантов использования и действующих лиц, выполнение теста на производительность.

Рабочий процесс (workflow) представляет собой последовательность действий, приводящую к получению значимого результата. В терминах UML рабочий процесс может быть описан с помощью диаграммы последовательности, сотрудничества или процессов. В рамках RUP определены шесть основных процессов:

• построение бизнес-моделей;

• определение требований;

• анализ и проектирование;

• реализация;

• тестирование;

• развертывание

и три вспомогательных процесса:

• управление конфигурацией;

• управление проектом;

• создание инфраструктуры (environment).

RUP как продукт входит в состав комплекса Rational Suite, причем каждый из перечисленных выше процессов поддерживается определенным инструментальным средством комплекса *(см.* подразд. 4.3.4). RUP состоит из базы знаний и руководства в твердой копии. База знаний включает следующие компоненты:

• руководства для всех участников проектной команды, охватывающие весь жизненный цикл ПО. Руководства представлены в двух видах — для осмысления процесса на верхнем уровне и в виде подробных наставлений по повседневной деятельности;

• наставления по использованию инструментальных средств, входящих в состав Rational Suite;

• примеры и шаблоны проектных решений для Rational Rose;

• шаблоны проектной документации для SoDa;

• шаблоны в формате Microsoft Word, предназначенные для поддержки документации по всем процессам и действиям жизненного цикла ПО;

• планы в формате Microsoft Project, отражающие итерационный характер разработки ПО.

Адаптация RUP к потребностям конкретной организации или проекта обеспечивается с помощью специального набора инструментов и шаблонов Development Kit. База знаний имеет формат гипертекста (HTML - HyperText Markup Language - стандартный язык для создания страниц Интернет). Доступ к ней может осуществляться с помощью Microsoft Internet Explorer или Netscape Navigator. Такой формат допускает как индивидуальное, так и коллективное использование базы знаний в сети Интранет.

3 МЕТОД Oracle

Метод Oracle (Oracle Method) — это комплекс методов фирмы Oracle, охватывающий все стадии ЖЦ ПО. В состав комплекса входят следующие основные методы:

• CDM (Custom Development Method) - метод разработки прикладного ПО;

• PJM (Project Management Method) - метод управления проектом;

• AIM (Application Implementation Method) - метод внедрения прикладного ПО;

• BPR (Business Process Reengineering) — реинжиниринг бизнес-процессов;

• DWM (Data Warehouse Method) - метод создания хранилищ данных.

Метод CDM

Метод CDM представляет собой развитие достаточно давно созданного Oracle CASE-Method, известного по использованию CASE-средств фирмы Oracle и книгам Р. Баркера. Этот метод полностью опирается на использование инструментальных средств Oracle, несмотря на утверждения о простой адаптации CDM к проектам, в которых используется другой инструментальный комплекс.

В соответствии с CDM ЖЦ ПО формируется из определенных этапов (фаз) проекта и процессов, каждый из которых выполняется в течение нескольких этапов (рис. 5.4).

Перечислим этапы CDM и их назначение:

• стратегия (определение требований);

• анализ (формулирование детальных требований к прикладной системе);

• проектирование (преобразование требований в детальные спецификации системы);

• реализация (написание и тестирование приложений);

• внедрение (установка новой прикладной системы, подготовка к началу эксплуатации);

• эксплуатация (поддержка и слежение за приложением, планирование будущих функциональных расширений).

В методе CDM предусмотрены следующие процессы:

• определение бизнес-требований, или постановка задачи (Business Requirements Definition);

• исследование существующих систем (Existing Systems Examination). Выполнение этого процесса должно обеспечить понимание состояния существующего технического и программного обеспечения для планирования необходимых изменений;

• определение технической архитектуры (Technical Architecture);

• проектирование и реализация базы данных (Database Design and Build). Процесс предусматривает проектирование и реализацию реляционной базы данных, включая создание индексов и других объектов БД;

• проектирование и реализация модулей (Module Design and Build). Этот процесс является основным в проекте. Он включает непосредственное проектирование приложения и создание кода прикладной программы;

• конвертирование данных (Data Conversion). Цель этого процесса — преобразовать, перенести и проверить согласованность и непротиворечивость данных, оставшихся в наследство от существующей системы и необходимых для работы в новой ИС;

• документирование (Documentation);

• тестирование (Testing);

• обучение (Training);

• внедрение, или переход к новой системе (Transition). Этот процесс включает решение задач установки, ввода новой системы в эксплуатацию, прекращения эксплуатации старых систем;

• поддержка и сопровождение (Post-System Support).

Процессы состоят из последовательностей задач. Задачи разных процессов взаимосвязаны явно указанными ссылками.

В соответствии с методом CDM на *этапе стратегии* определяются цели создания системы, приоритеты и ограничения, разрабатывается системная архитектура и составляется план разработки ЭИС.

На *этапе анализа* строятся модель информационных потребностей (диаграмма "сущность-связь"), диаграмма функциональной иерархии (на основе функциональной декомпозиции ЭИС), матрица перекрестных ссылок и диаграмма потоков данных.

На *этапе проектирования* разрабатывается подробная архитектура ЭИС, проектируются схема реляционной БД и программные модули, устанавливаются перекрестные ссылки между компонентами ЭИС для анализа их взаимного влияния и контроля за изменениями.

На эт*апе реализации* создается БД, строятся прикладные системы, производятся их тестирование, проверка качества и соответствия требованиям пользователей. Создаются системная документация, материалы для обучения и руководства пользователей.

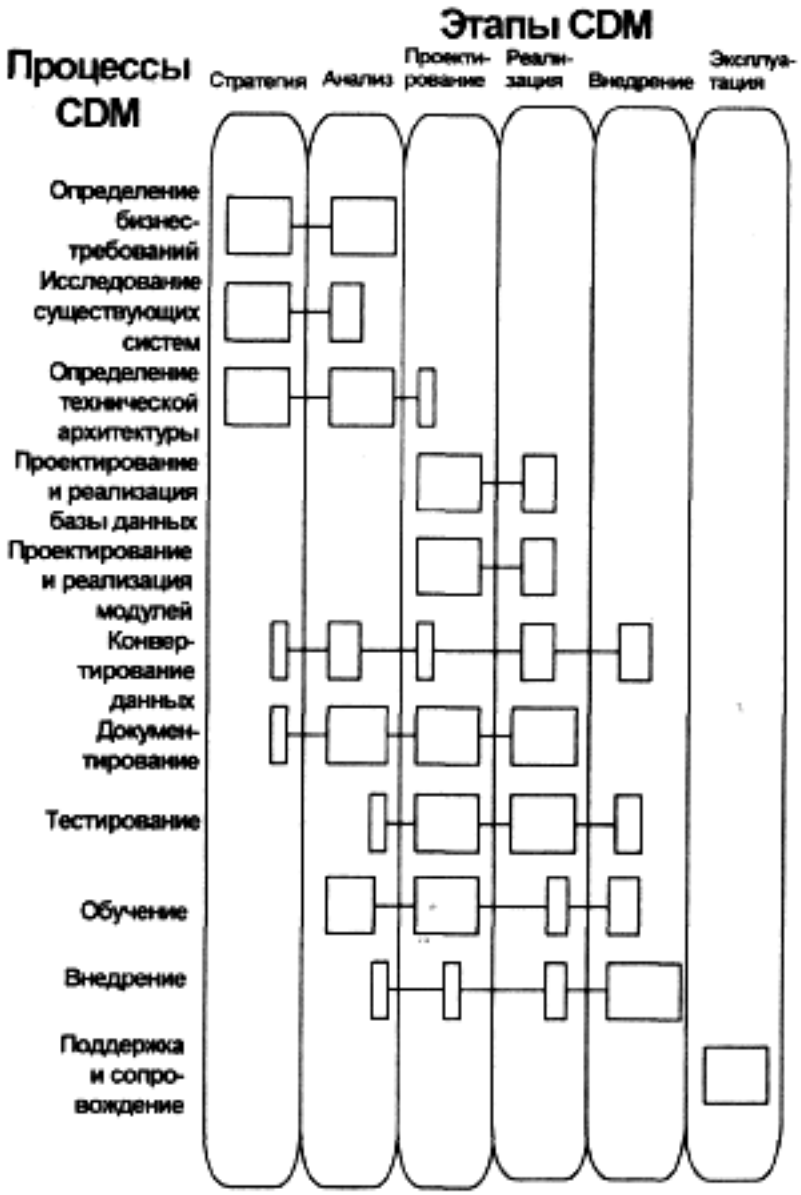

Рис. 4. Этапы и процессы CDM

На этапах внедрения и эксплуатации анализируются производительность и целостность системы, выполняются поддержка и, при необходимости, модификация ЭИС.

CDM предоставляет возможность выбрать требуемый подход к разработке. Это возможно, поскольку каждый процесс базируется на известных зависимостях между задачами одного типа и не зависит от того, на какие этапы будет разбит проект.

При определении подхода к разработке оцениваются масштаб, степень сложности и критичность будущей системы. При этом учитываются стабильность требований, сложность и количество бизнес-правил, количество автоматически выполняемых функций, квалификация и число пользователей, степень взаимодействия с другими системами, критичность приложения для основного бизнес-процесса компании и целый ряд других.

В соответствии с этими факторами в CDM выделяются три основных подхода к разработке:

• классический подход (Classic);

• подход быстрой разработки (Fast Track);

• подход облегченной разработки (Lite).

Классический подход. Этапы данного подхода представлены на рис. 4. Классический подход применяется для наиболее сложных и масштабных проектов. Для таких проектов характерны большое количество реализуемых бизнес-правил, распределенная архитектура, критичность приложения. Применение классического подхода также рекомендуется при нехватке опыта у разработчиков, неподготовленности пользователей, нечетко определенной задаче. Продолжительность таких проектов — от 8 до 36 мес.

Подход быстрой разработки. В этом подходе три этапа: моделирование требований, проектирование и генерация системы и внедрение в эксплуатацию. Подход используется для реализации небольших и средних проектов при условии простоты бизнес-правил. При этом основные функциональные возможности прикладной системы генерируются с использованием CASE-средства Oracle Designer. Для таких проектов также характерны невысокая сложность архитектуры системы, гибкие сроки и четкая постановка задачи. Продолжительность проекта от 4 до 16 мес.

Подход облегченной разработки. Здесь всего два этапа: реализация прототипа и внедрение в эксплуатацию. Подход применяется для реализации малых проектов. Подход Lite предназначен для разработки прототипов в сжатые сроки. Продолжительность проекта от 1 до 6 мес.

Все перечисленные подходы являются, по существу, каскадными. Даже облегченный подход, несмотря на итерационность выполнения действий по прототипированию, сохраняет общий последовательный и детерминированный порядок выполнения задач.

Большинство задач проектирования и разработки решается с использованием Oracle Designer — основного инструментального средства CDM. Для решения задач календарного планирования и управления разработкой можно воспользоваться готовым вариантом распределения работ по проекту, где уже составлен подробный график работ с исполнителями. Руководителю проекта остается только скорректировать сроки (предлагается это сделать либо в MS Project 4.0, либо в ABT Project Workbench 3.0). При этом руководитель проекта может в самом начале оценить трудозатраты по исполнителям и спланировать их работу по отдельным проектам. В справочной документации по CDM приводятся таблицы, в которых указаны оценки трудозатрат на выполнение отдельных процессов в процентах от трудозатрат по всему проекту или по отдельному его этапу. Можно оценить загруженность каждого исполнителя по проекту, по этапу и степень его участия при выполнении отдельной задачи.

В CDM отдельно решается задача документирования результатов проекта. Для каждого проектного результата имеется возможность с помощью макросов сгенерировать в MS Word шаблон документа, который может содержать примеры диаграмм в формате Visio 4.0.

Метод PJM

Метод PJM оформлен в виде коммерческого продукта и называется PJM Advantage. Цель реализованного в PJM подхода — обеспечить участников проекта технологией, в которой проекты разных типов могут быть спланированы, оценены по ресурсам, проконтролированы и нормально завершены.

Другими словами, PJM — это определенная дисциплина ведения проекта, позволяющая гарантировать, что цели проекта, четко определенные в его начале, остаются в центре внимания на протяжении всего проекта.

В основе PJM лежит метод, ориентированный на выполнение самостоятельных процессов (под *процессом* понимается набор связанных задач, выполнением которых достигается определенная цель проекта). Так же, как и CDM, метод руководства проектом представляется в виде четко определенной операционной схемы, в которой выделяются процессы, этапы, задачи, результаты решения задач и зависимости между задачами (рис. 5.5).

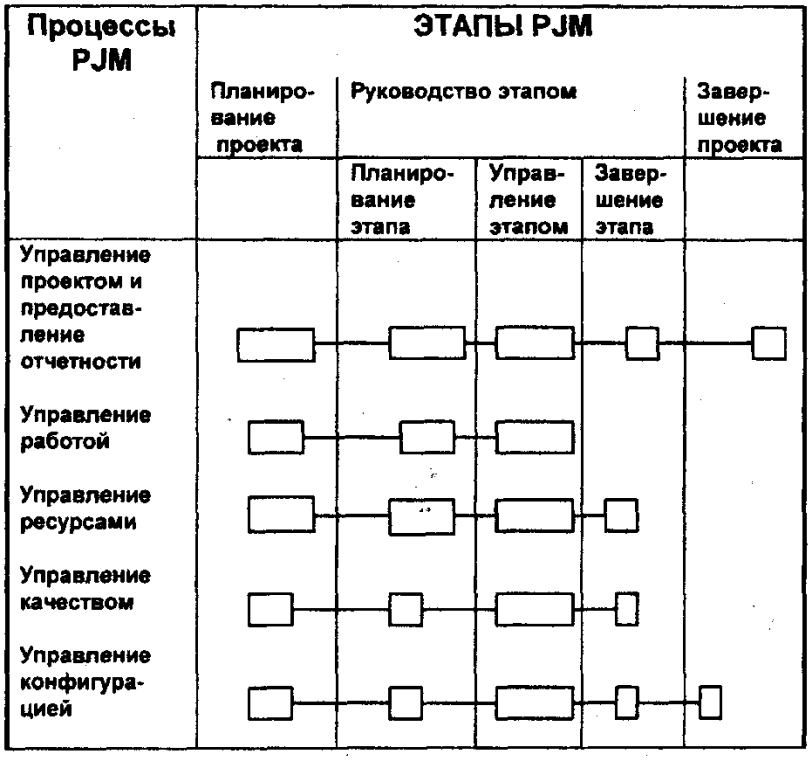

Рис. 5. Этапы и процессы PJM

Отдельный проект может включать большинство этих процессов, вне зависимости от того, кто отвечает за отдельный процесс - консалтинговая фирма, организация пользователя или другие лица.

Рассмотрим более подробно процессы, которые формируют полный набор решаемых в PJM задач:

• *Управление проектом и предоставление отчетности* (Control and Reporting) - содержит задачи, в результате решения которых определяются границы проекта и подход к разработке, происходит управление изменениями и контролируется возможный риск. Здесь же содержатся задачи, связанные с ведением планов и предоставлением отчетности по проекту.

• *Управление работой* (Work Management) - содержит задачи, помогающие руководить работами, выполняемыми по плану, и контролировать их. Предназначен также для поддержки финансового ведения проекта.

• *Управление ресурсами* (Resource Management) - включает задачи, связанные с обеспечением каждого этапа исполнителями, а также содержит указания о необходимых для выполнения работ по проекту умениях и навыках.

• *Управление качеством* (Quality Management) - гарантирует, что проект

отвечает требованиям пользователя в течение всего процесса разработки.

• *Управление конфигурацией* (Configuration Management) - содержит задачи, помогающие сохранить, организовать и проследить за всем тем, что получается в результате выполнения проекта. Цикл решения задач методом PJM состоит из отдельных этапов.

Количество этапов зависит от выбранного подхода к разработке. Задачи PJM можно распределить внутри каждого процесса по трем группам (задачи планирования, управления и завершения) и по уровням (отнести задачу на уровень проекта или на уровень отдельного этапа). В результате жизненный цикл PJM складывается из задач пяти категорий. Соотношение между процессами и этапами PJM представлено на рис. 5.

По аналогии с CDM в методе PJM предусмотрено широкое использование шаблонов разрабатываемых документов. Для составления календарного плана работ предлагается воспользоваться шаблонами формата MS Project 4.0 либо ABT Project Workbench 3.0, а для получения других документов - шаблонами MS Word.

Контрольные вопросы

1. Каковы основные характерные особенности технологии DATARUN?
2. В чем состоят основные характерные особенности RUP?
3. Каковы основные характерные особенности Oracle Method?
4. Что общего и какие различия имеются у перечисленных технологий?