ТЕМА 15

1. Технология Rational Unified Process.
2. Технология Oracle.
3. Технология Borland.

Рассмотрим несколько современных технологий создания ПС, использующих CASE-средства, объектно-ориентированный подход и нотацию UML.

15.1 Технология Rational Unified Process

Согласно RUP ЖЦ 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;
- стадия разработки;
- стадия конструирования;
- стадия ввода в действие.

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

Начальная стадия может принимать множество разных форм. Для крупных проектов она связана с всесторонним изучением всех возможностей реализации проекта. В это же время вырабатывается бизнес-план проекта: определяется, сколько приблизительно он будет стоить и какой доход принесет. Кроме того, выполняется начальный анализ для оценки размеров проекта. Результатами начальной стадии являются:

- общее описание системы: основные требования к проекту, его характеристики и ограничения;
- начальная модель вариантов использования;
- начальный проектный глоссарий;
- начальный бизнес-план;
- план проекта, отражающий стадии и итерации;
- один или несколько прототипов.

На стадии разработки выявляются более детальные требования к системе, выполняется высокоуровневый анализ предметной области и проектирование для построения базовой архитектуры системы, создается план конструирования и устраняются наиболее рискованные элементы проекта. Результатами стадии разработки являются:

- завершенная модель вариантов использования, определяющая функциональные 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 разработки является описание базовой архитектуры будущей системы. Эта архитектура включает в себя:

- 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.

Созданная архитектура является основой всей дальнейшей разработки. В будущем неизбежны незначительные изменения в деталях архитектуры, однако, серьезные изменения маловероятны. Стадия разработки занимает около пятой части общей продолжительности проекта. Основными признаками ее завершения являются следующие:

- разработчики в состоянии оценить, 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.

Стадия конструирования напрямую связана с проработкой итераций. Они на стадии конструирования являются одновременно инкрементными и повторяющимися. Инкрементность связана с добавлением новых конструкций к вариантам использования, реализованным во время предыдущих итераций. Повторяемость относится к разрабатываемому коду: на каждой итерации некоторая часть существующего кода переписывается с целью сделать его более гибким. Результатом стадии конструирования является продукт, готовый к передаче конечным пользователям и содержащий следующее:

- ПО, интегрированное на требуемых платформах;
- руководства пользователя;
- описание текущей реализации.

Стадия ввода в действие связана с передачей готового продукта в распоряжение пользователей. Она включает в себя:

- бета-тестирование, позволяющее убедиться, что новая система соответствует ожиданиям пользователей;
- параллельное функционирование с существующей системой, которая подлежит постепенной замене;
- конвертирование баз данных;
- оптимизацию производительности;
- обучение пользователей и специалистов службы сопровождения.

Статический аспект RUP представлен четырьмя основными элементами: роли; виды деятельности; рабочие продукты; дисциплины.

Роль определяет поведение и ответственность личности или группы личностей, составляющих проектную команду. Одна личность может играть в проекте много различных ролей.

Под видом деятельности конкретного исполнителя понимается единица выполняемой им работы. Вид деятельности соответствует понятию технологической операции. Он имеет четко определенную цель, обычно выражаемую в терминах получения или модификации некоторых рабочих продуктов, таких, как модель, элемент модели, документ, исходный код или план. Каждый вид деятельности связан с конкретной ролью. Продолжительность вида деятельности составляет от нескольких часов до нескольких дней, он обычно выполняется одним исполнителем и порождает только один или весьма небольшое количество рабочих продуктов. Любой вид деятельности должен являться элементом процесса планирования. Примерами видов деятельности могут быть планирование итерации, определение вариантов использования и действующих лиц, выполнение теста на производительность. Каждый вид деятельности сопровождается набором руководств, представляющих собой методики выполнения технологических операций.

Дисциплина соответствует понятию технологического процесса и представляет собой последовательность действий, приводящую к получению значимого результата.

В рамках RUP определены шесть основных дисциплин: построение бизнес-моделей; определение требований; анализ и проектирование; реализация;
тестирование; развертывание.

Имеется три вспомогательные дисциплины: управление конфигурацией и изменениями; управление проектом; создание инфраструктуры.

15.2 Технология Oracle

Методическую основу технологии создания ПО корпорации Oracle составляет метод Oracle, представляющий собой комплекс методов, охватывающий большинство процессов ЖЦ ПО. В состав комплекса входят компоненты:

- CDM (Custom Development Method) - разработка прикладного ПО;
- PJM (Project Management Method) - управление проектом;
- AIM (Application Implementation Method) - внедрение прикладного ПО;
- BPR (Business Process Reengineering) - реинжиниринг бизнес-процессов;
- OCM (Organizational Change Management) - управление изменениями.

Метод CDM включает в себя PJM и оформлен в виде консалтингового продукта CDM Advantage. Это библиотека стандартов и руководств, представляющая собой развитие CASE-Method, ранее созданного Oracle. Фактически CDM является методическим руководством по разработке прикладного ПО с использованием инструментального комплекса Oracle

Developer Suite, а сам процесс проектирования и разработки тесно связан с Oracle Designer и Oracle Forms.

В соответствии с CDM ЖЦ ПО формируется из определенных этапов (фаз) проекта и процессов, каждый из которых выполняется в течение нескольких этапов (Рис. 15.1).

На этапе стратегии определяются цели создания системы, приоритеты и ограничения, разрабатывается системная архитектура и составляется план разработки.

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

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

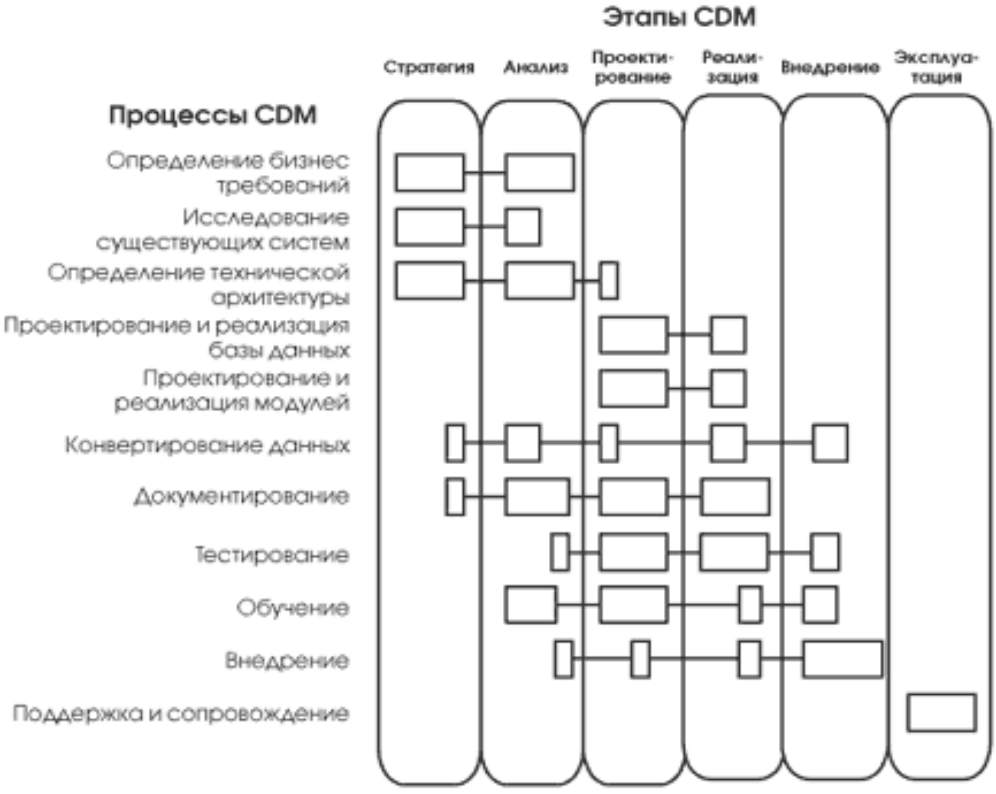

Рисунок 15.1 – Этапы и процессы CDM

На этапе реализации создается БД, строятся прикладные системы, производится их тестирование, проверка качества и соответствия требованиям пользователей. Создается системная документация, материалы для обучения и руководства пользователей.

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

Рассмотрим более подробно процессы CDM.

Определение бизнес-требований подразумевает постановку задачи проектирования.

Исследование существующих систем должно обеспечить понимание состояния, созданного технического и программного обеспечения для планирования необходимых изменений.

Определение технической архитектуры связано с выбором конкретной архитектуры разрабатываемой системы.

Проектирование и реализация базы данных это создание реляционной базы данных, включая индексы и другие объекты БД.

Процесс проектирования и реализации модулей является основным в проекте. Он включает в себя непосредственное проектирование приложения и создание кода прикладной программы.

Конвертирование данных связано с преобразованием, переносом и проверкой согласованности и непротиворечивости данных, оставшихся от "старой" системы и необходимых для работы в новой системе.

Документирование, тестирование и обучение – это процессы подготовки системы к внедрению.

Процесс внедрения связан с решением задач установки, ввода новой системы в эксплуатацию, прекращением эксплуатации старых систем.

Поддержка и сопровождение обеспечивают эксплуатацию созданной системы.

CDM предоставляет возможность выбрать требуемый подход к разработке. Для этого оценивается масштаб, степень сложности будущей системы, учитываются стабильность требований, сложность и количество бизнес-правил, количество автоматически выполняемых функций, разнообразие и количество пользователей, степень взаимодействия с другими системами, и целый ряд других параметров. В соответствии с перечисленными факторами в CDM выделяются два основных подхода к разработке:

Классический подход. Применяется для наиболее сложных и масштабных проектов, предусматривая последовательный и детерминированный порядок выполнения задач. Для таких проектов характерно 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 месяцев.

Подход быстрой разработки. В отличие от каскадного классического, он является итерационным. В нем четыре этапа: стратегия, моделирование требований, проектирование и генерация системы, внедрение в эксплуатацию. Подход используется для реализации небольших и средних проектов с несложной архитектурой системы, гибкими сроками и четкой постановкой задач. Продолжительность проекта от 4 до 16 месяцев.

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. Рассмотрим подробнее процессы PJM.

Управление проектом и предоставление отчетности. Этот процесс содержит задачи, в результате решения которых определяются границы проекта и подход к разработке, происходит управление изменениями и контролируется возможный риск.

Управление работой. Процесс содержит задачи, помогающие контролировать работы, выполняемые в проекте.

Управление ресурсами. Здесь решаются задачи, связанные с обеспечением каждого этапа исполнителями.

Управление качеством. Процесс гарантирует, что проект отвечает требованиям пользователя в течение всего процесса разработки.

Цикл решения задач PJM состоит из отдельных этапов. Их количество зависит от выбранного подхода к разработке. Задачи PJM можно распределить внутри каждого процесса по трем группам: планирования, управления и завершения. В зависимости от уровня задачу можно отнести на уровень проекта или на уровень отдельного этапа.

По аналогии с CDM в PJM предусмотрено широкое использование шаблонов разрабатываемых документов.

Комплекс Oracle Developer Suite содержит набор интегрированных средств разработки для быстрого создания приложений. В него входят средства моделирования, программирования на Java, разработки компонентов, бизнес-анализа и составления отчетов. Все эти средства используют общие ресурсы, что позволяет совместно работать над одним проектом группе разработчиков. Oracle Developer Suite интегрирован с Oracle Database и Oracle Application Server, образуя единую платформу для создания и установки приложений.

В Oracle Developer Suite встроена поддержка языка UML для разработки приложений на основе моделей. Модели хранятся в общем репозитории Oracle, который предназначен для поддержки больших коллективов разработчиков.

15.2 Технология Borland

Компания Borland в результате развития собственных разработок и приобретения целого ряда компаний представила интегрированный комплекс инструментальных средств, реализующих управление полным жизненным

циклом приложений (Application Life Cycle Management, ALM). В соответствии с технологией Borland процесс создания ПО включает в себя пять основных этапов: определение требований; анализ и проектирование; разработка;
тестирование и профилирование; развертывание.

Выполнение всех этапов координируется процессом управления конфигурацией и изменениями.

Определение требований реализуется с помощью системы управления требованиями CaliberRM, которая стала частью семейства продуктов Borland. CaliberRM сохраняет требования в базе данных, документы с их описанием создаются с помощью встроенного механизма генерации документов MS Word на базе заданных шаблонов. Система обеспечивает экспорт данных в таблицы MS Access и импорт из MS Word. CaliberRM поддерживает различные методы визуализации зависимостей между требованиями, с помощью которых пользователь может ограничить область анализа, необходимого в случае изменения того или иного требования. Имеется модуль, который использует данные требования для оценки трудозатрат, рисков и расходов, связанных с реализацией требований.

Анализ и проектирование выполняются с помощью ПС Together ControlCenter. Это интегрированная среда проектирования и разработки, поддерживающая визуальное моделирование на UML с последующим 7еалии7я7ем приложений для платформ J2EE (Java) и .Net (C#, C++ и Visual Basic). Кроме базовой версии, имеется уменьшенный вариант системы для индивидуальных разработчиков и небольших групп (Together Solo), а также редакции для платформы IBM WebSphere и среды разработки Jbuilder.

Разработка выполняется по технологии LiveSource, которая обеспечивает синхронизацию между проектом приложения и изменениями в системе. При внесении изменений в исходные тексты меняется модель, а при ее изменении соответствующим образом корректируется текст на языке программирования. Это исключает необходимость вручную модифицировать модель или переписывать код. Контроль версий осуществляется благодаря функциональной интеграции Together и системы StarTeam. Поддерживается также интеграция с системой управления конфигурацией Rational ClearCase.

Тестирование обеспечивается с помощью инструментальных средств Optimizeit Suite 5, Optimizeit Profiler for .NET и Optimizeit ServerTrace. Первые две системы позволяют выявить потенциальные проблемы использования аппаратных ресурсов: памяти и процессорных мощностей на платформах J2EE и .Net соответственно. Интеграция Optimizeit Suite 5 в среду разработки Jbuilder, а Optimizeit Profiler – в C#Builder и Visual Basic .Net реализует проводить контрольные испытания приложений по мере разработки и ликвидировать узкие места производительности. Система Optimizeit ServerTrace предназначена для управления производительностью серверных J2EE-приложений с точки зрения достижения заданного уровня обслуживания и сбора контрольных данных по виртуальным Java-машинам.

Сущность концепции ALM сосредоточена в системе управления конфигурацией и изменениями: именно она объединяет основные фазы ЖЦ ПО. Такой системой является StarTeam. Она выполняет функции контроля версий, управления изменениями, отслеживания дефектов, управления требованиями (в интеграции с CaliberRM), управления потоком задач и управления проектом. StarTeam совместима с интерфейсом Microsoft Source Code Control и интегрируется с любой системой разработки, которая поддерживает этот интерфейс. Кроме того, в системе реализованы средства интеграции со средствами разработки и моделирования Together, Jbuilder, Delphi, C++Builder и C#Builder.

В технологии Borland выделяется три уровня интеграции. *Функциональная* интеграция позволяет обратиться из одной системы к функциям другой, выбрав соответствующий пункт меню. Например, интерфейс управления изменениями StarTeam непосредственно отображается в системах Together, C#Builder и Visual Studio .Net. Такая интеграция дает возможность разделять информацию между системами, но не обеспечивает единого рабочего пространства, вынуждает пользователя переключать окна и приводит к дублированию процессов управления структурой проекта. *Встроенная* интеграция обеспечивает работу с одной системой непосредственно в среде другой. Например, не выходя из среды разработки Jbuilder, можно просматривать графики производительности, которые создает система Optimizeit. Самый высокий уровень интеграции – синергетический, позволяющий сочетать функции двух различных продуктов незаметно для разработчиков. В настоящее время для большинства продуктов Borland и других поставщиков начинают реалиизовываться принципы синергетической интеграции.