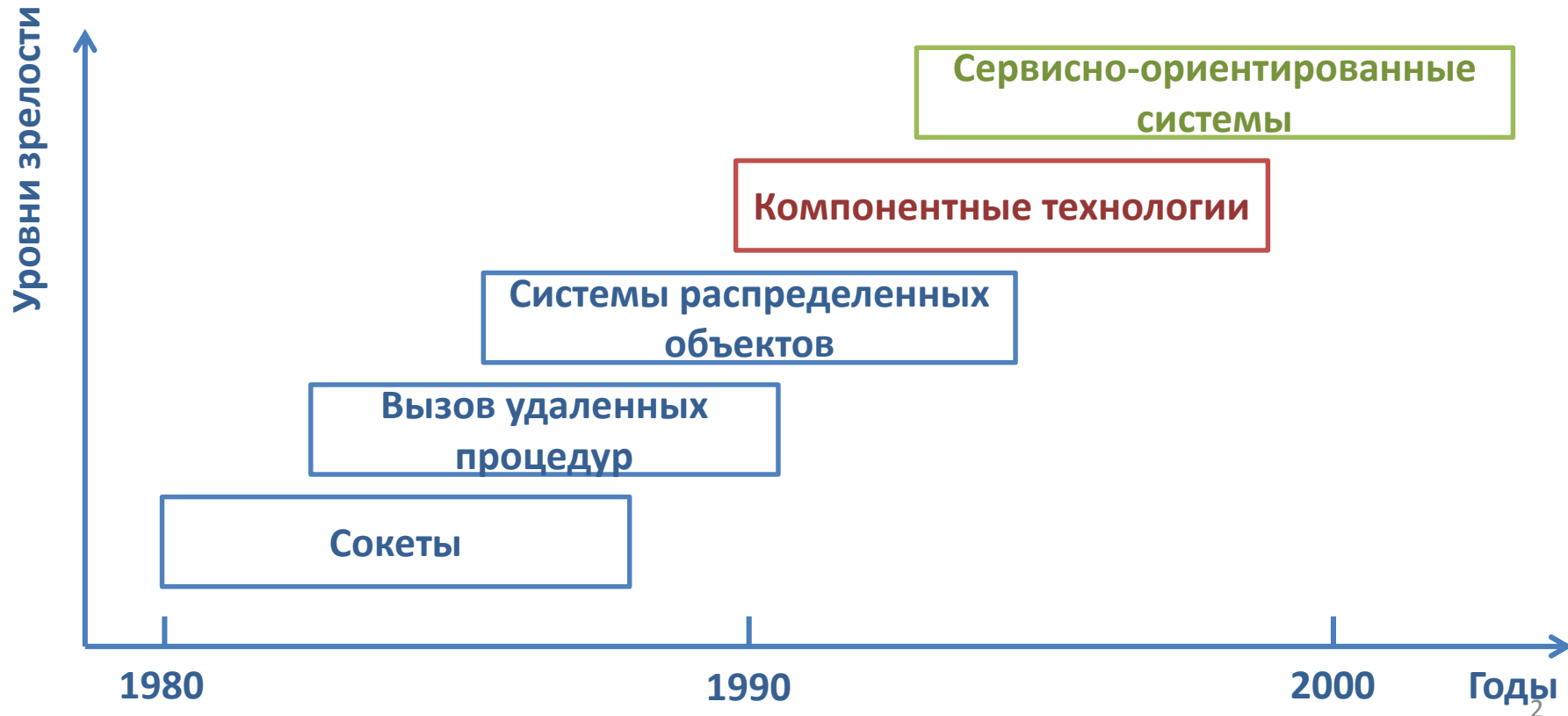


Лекция 7. Технологии разработки распределенных систем

Основные понятия разработки распределенных систем

1. Компонентные технологии
2. Сервис-ориентированные технологии
3. Интеграция приложений в проекте

Фазы развития технологий разработки распределенных систем



Программный компонент

- (программная архитектура) программный модуль, реализующий некоторую функцию или набор функций, который решает определенные подзадачи в рамках общих задач системы
- (компонентная технология) объект со специальными свойствами
- Структурная единица системы с четко определенным интерфейсом, который полностью описывает ее зависимости от контекста
- Откомпилированный автономный программный модуль
- Может быть простым (кнопка) или сложным (управление сетью)

Свойства программного компонента



- ☐ Фрагмент самодостаточного кода
- ☐ Самоустанавливаемый модуль
- ☐ Повторно используется в различных контекстах
- ☐ При работе с ПК используются механизмы динамического связывания
- ☐ Может быть объединен с другими ПК
- ☐ Используется по принципу «черного ящика»

Программный компонент

- Компонентная модель
 - Набор правил, определяющих интерфейсы ПК и их реализаций, а также правил, по которым ПК работают в системе и взаимодействуют друг с другом
- Компонентная среда
 - Набор базовых служб, необходимых для функционирования компонентов, вместе с поддерживаемой с их помощью компонентной моделью
 - Компонентный фреймворк

- Объект – повторное использование на низком уровне
- Объекты в рамках конкретного языка программирования
- Объекты связаны с конкретным языком программирования

Объект



- ПК – высокоуровневое повторное использование кода
- ПК – большая ориентация на интерфейсы
- ПК разрабатываются в рамках фреймворка
- ПК явно не связаны с языком, но связаны с платформой

Программный компонент



Квазикомпонентно-ориентированные технологии

Сокеты

- Программный интерфейс, предназначенный для передачи данных между приложениями
- Потокосовые (TCP) и датаграммные (UDP)

Вызов удаленных процедур

- Remote Procedure Call

Среда распределенных вычислений

- Distributed Computing Environment

Программный интерфейс вызова удаленных методов Java

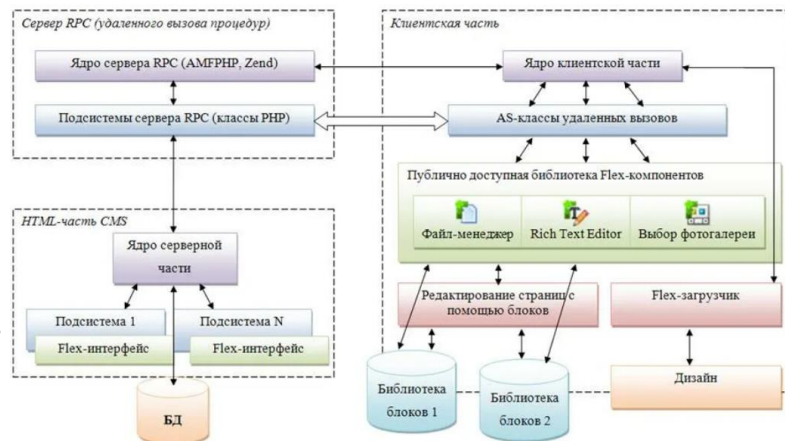
- Java Remote Method Invocation

Квазикомпонентно-ориентированные технологии.

Вызов удаленных процедур

Программа клиента выполняет локальный вызов процедуры посредством обращения к заглушке:

- Заглушка на клиентской стороне перекодирует запрос в стандартный формат (marshalling)
- Заглушка клиента связывается с заглушкой сервера
 - Нахождение удаленного хоста с сервером
 - Нахождение нужного серверного процесса на хосте.
- Заглушка сервера декодирует запрос (unmarshalling)
- Серверная заглушка вызывает нужную процедуру
- Процедура выполняется на сервере, результаты передаются серверной заглушке
- Серверная заглушка выполняет маршalling и отправляет результаты клиентской заглушке
- Получив результат, клиентская заглушка выполняет операцию дермаршallingа и передает результат основной программе



Особенности: параметры передаются по значению, облегчает работу программиста, ориентация на процедурный стиль программирования, реализация статических вызовов (Зглушки встраиваются в текст клиента и сервера на этапе разработки)

Квазикомпонентно-ориентированные технологии.

Среда распределенных вычислений

- Структура
 - Служба распределенных файлов
 - Служба каталогов
 - Служба безопасности
 - Служба распределенного времени
- Объекты
 - Язык определения интерфейсов IDL
 - Динамические распределенные объекты
 - Именованные распределенные объекты
- Недостатки
 - Близость к RPC

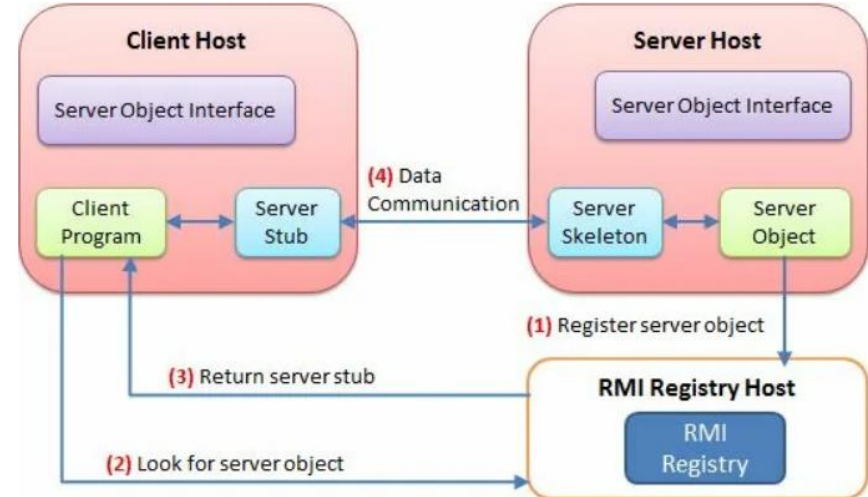


Квазикомпонентно-ориентированные технологии.

Программный интерфейс вызова удаленных методов Java



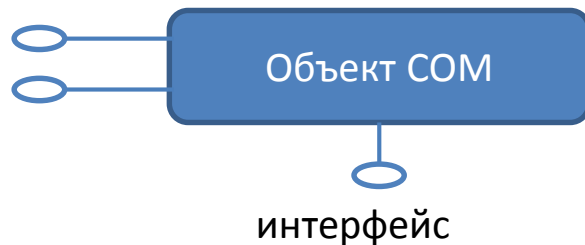
- Java RMI — Java Remote Method Invocation — программный интерфейс вызова удаленных методов в языке Java, представляющий собою распределенную объектную модель, специфицирующую, способы вызовов удаленных методов, работающих на другой виртуальной машине Java



Программный компонент.

Объектная модель компонентов

- COM (Component Object Model) - это технологический стандарт от компании Microsoft, предназначенный для создания программного обеспечения на основе взаимодействующих компонентов, каждый из которых может использоваться во многих программах одновременно
- Стандарт воплощает в себе идеи *полиморфизма* и *инкапсуляции* ООП
- Стандарт COM мог бы быть универсальным и платформо-независимым, но закрепился в основном на операционных системах семейства Windows
- На основе COM были реализованы технологии: Microsoft OLE Automation, ActiveX, DCOM, COM+, DirectX, а также XPCOM

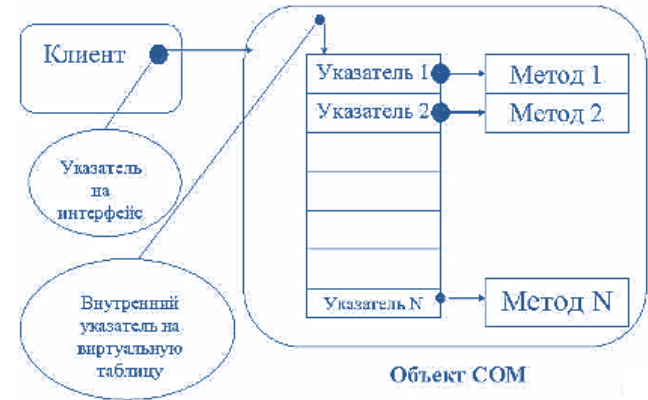


- ❖ Поддерживает более одного интерфейса
- ❖ Каждый интерфейс включает множество методов
 - ❖ Три стандартных
 - ❖ Пользовательские
- ❖ Определенный интерфейс нельзя менять и дополнять

Программный компонент.

Интерфейсы COM

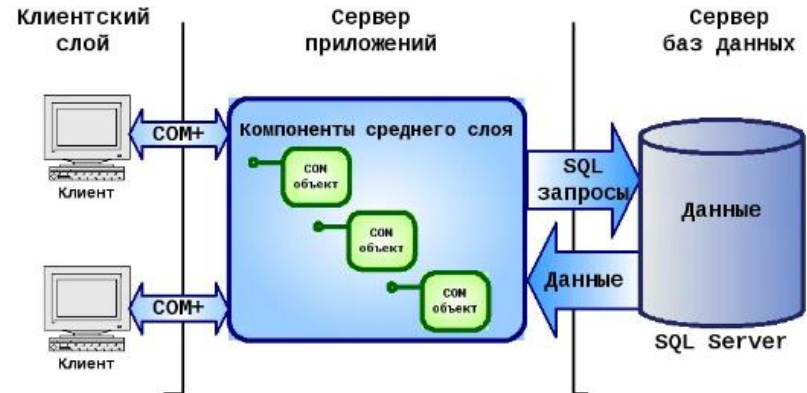
- Интерфейс – контракт между объектом и его клиентом
- Интерфейс имеет два имени
 - Имя для человека, типично начинается с I и понятно для человека
 - ISpeller, Ithesaur
 - Машинное имя – идентификатор интерфейса IID (128 бит длиной) или GUID (Globally Unique Identifier)
 - Первая часть (48 бит) – номер сетевой карты хоста
 - Вторая часть – текущее значение системных часов
- Объект COM должен поддерживать интерфейс IUnknown, который включает методы:
 - QueryInterface
 - AddRef
 - Release
- Все интерфейсы наследуются от Iunknown
- Доступ к методам осуществляется через указатель на метод
 - Указатель может быть передан другим клиентом
 - Указатель может быть получен от моникера



Программный компонент.

Серверы объектов COM и библиотека COM

- Сервер «в процессе»
 - в виде .dll
 - выполняется в одном адресном пространстве с клиентом
- Локальный сервер
 - в виде .exe
 - выполняется на том же хосте, что и клиент
- Удаленный сервер
 - или .dll или .exe
 - Расположен на удаленном по отношению к клиенту хосте



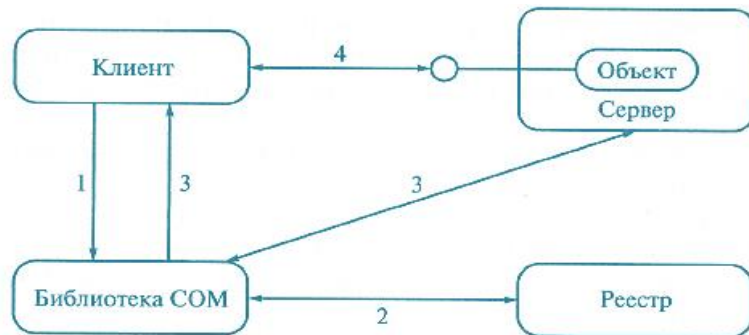
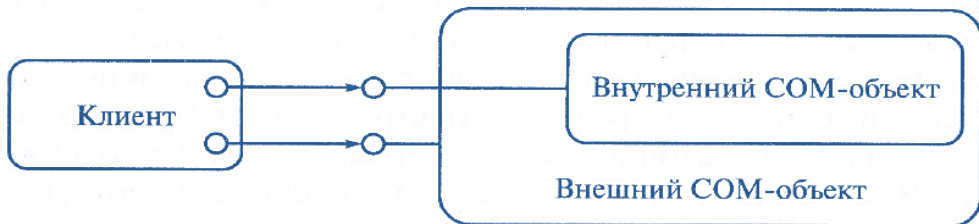
Библиотека COM предоставляет клиентам механизм запуска серверов объектов.

- Доступ к сервисам библиотеки COM осуществляется через вызов обычных функций, а не методов интерфейсов COM-объектов
- Имена функций библиотеки COM начинаются с "Co" - CoCreateInterface

Программный компонент.

Создание и применение объектов COM

- ❖ Вызов функции CoCreateInstance
- ❖ Поиск в системном реестре соответствующего класса
- ❖ Запуск сервера COM и возврат указателя на требуемый интерфейс
- ❖ Вызов требуемого метода



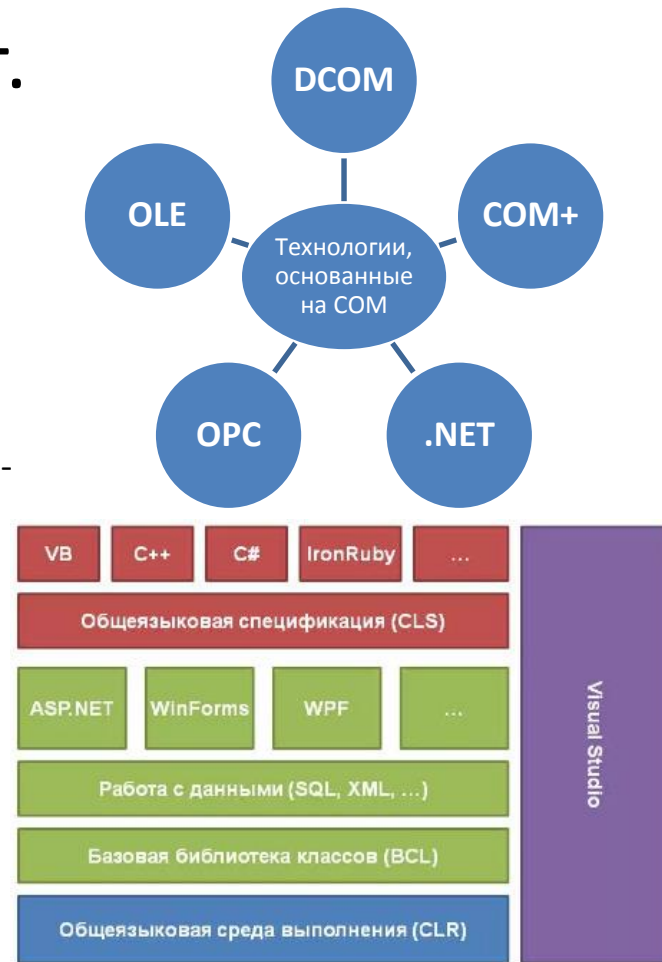
Программный компонент.

Технологии

- **DCOM** позволяет COM-компонентам взаимодействовать друг с другом по сети. Главным конкурент — CORBA.
- **OPC (OLE for Process Control)** — семейство программных технологий, предоставляющих единый интерфейс для управления объектами автоматизации и технологическими процессами. Многие из OPC-протоколов базируются на Windows-технологиях: OLE, ActiveX, COM/DCOM.
- **OLE (Object Linking and Embedding)** — технология связывания и внедрения объектов в другие документы и объекты, разработанные корпорацией Майкрософт.
- **.NET Framework** программная платформа, предназначенная для создания обычных и веб-приложений

Состав

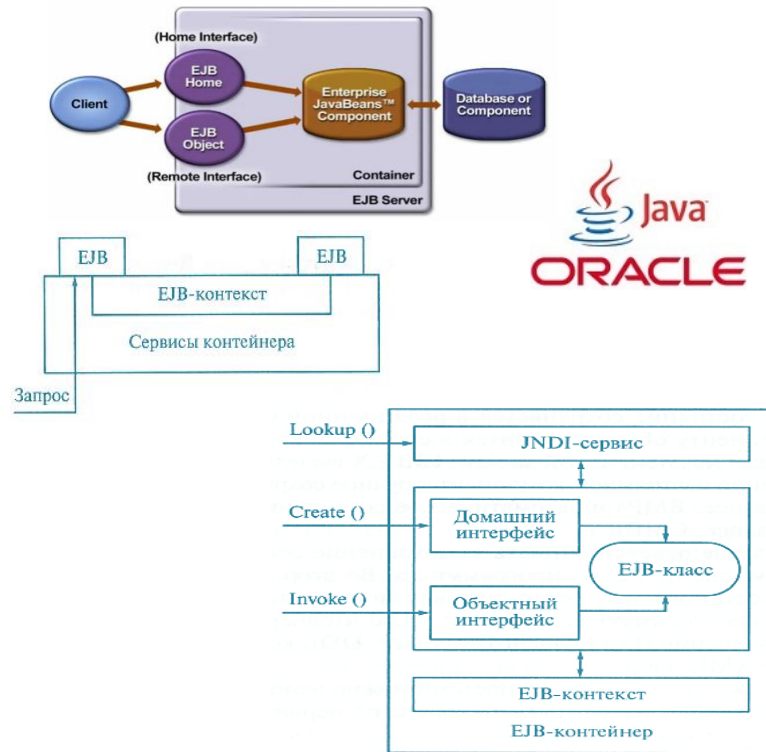
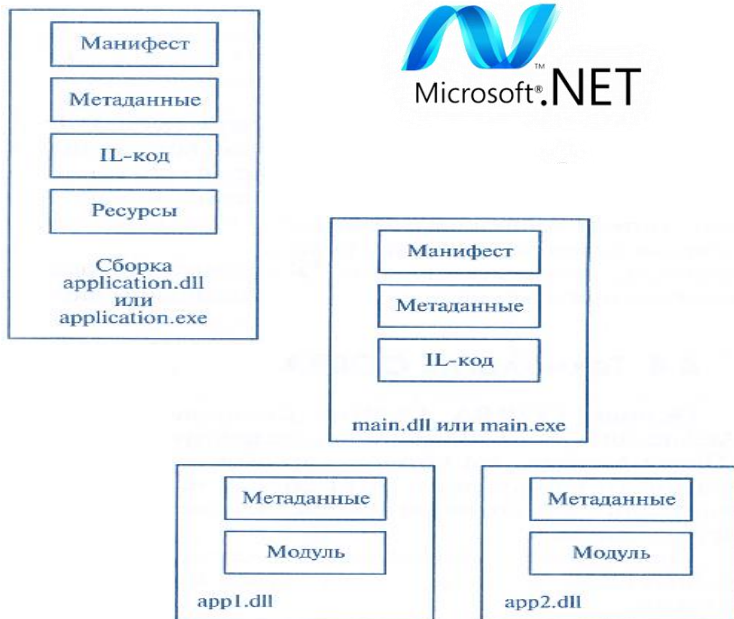
- CLR (common language runtime) - инвариантная к языку программирования среда исполнения
- FCL (framework class library) - библиотека классов Framework



Программный компонент.

Стек технологий .NET и Enterprise Java Beans

- Типовая структура сборки



Сервисно-ориентированная архитектура

- Service-oriented architecture, SOA – это подход к созданию ИС, основанный на использовании сервисов или служб (service)
- Подход к построению слабосвязанных систем, реализующих механизмы асинхронного взаимодействия
- Интеграционная архитектура, основанная на концепции сервисов



Сервис - любая дискретная функция, которая может быть предложена внешнему потребителю

- * отдельная бизнес-функция
- * набор функций, образующих бизнес-процесс

Преимущества SOA

- ☐ Снижение сроков освоения и внедрения новых ИТ-систем
- ☐ Быстрое создание новых ИТ-систем на базе уже существующих
- ☐ Уменьшение стоимости владения ИТ-продуктами и стоимости их интеграции
- ☐ Увеличение срока жизни ИТ-системы за счет оперативной модернизации
- ☐ Использование гибкой модели ценообразования за счет аутсорсинга
- ☐ Уменьшение стоимости работ по интеграции при слиянии компаний
- ☐ Реализация БП на уровне, не зависящем от приложений и платформ

Сервисно-ориентированная архитектура

Свойства

Представляет собой многократно используемую бизнес-функцию

Определяется с помощью формальных, не зависящих от реализации интерфейсов

Наличие протоколов связи, обеспечивающих прозрачность местонахождения и инвариантность по отношению к языку и платформе

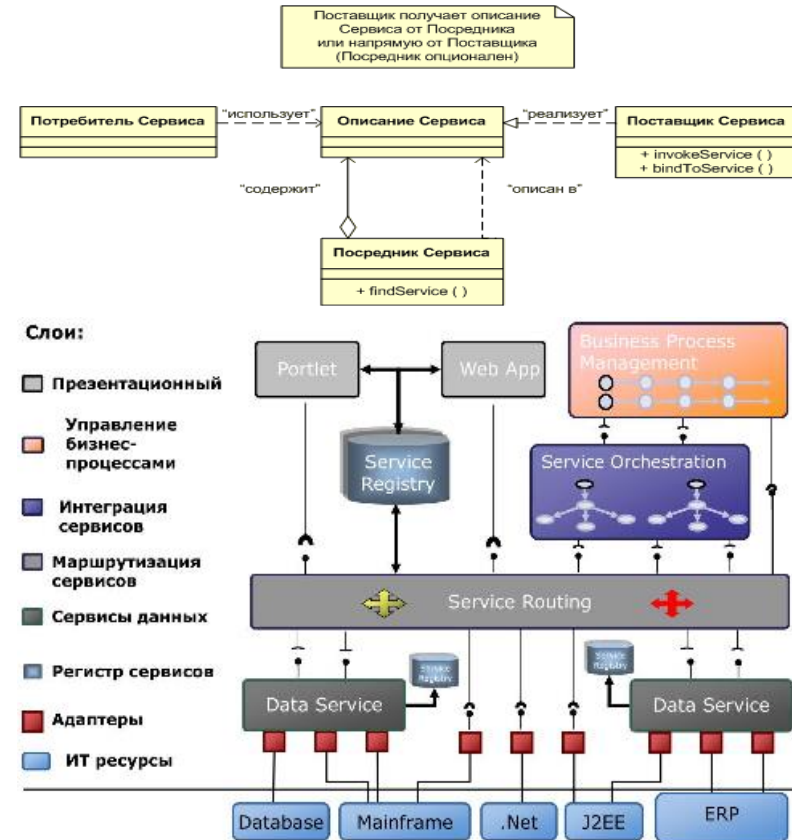
Атрибуты



Сервисно-ориентированная архитектура.

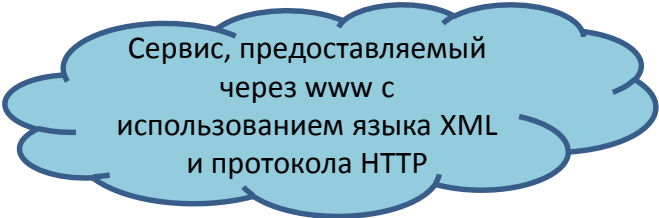
Концептуальная модель

- ✓ **Поставщик сервиса**
 - публикует описание сервиса и обеспечивает его реализацию
- ✓ **Потребитель сервиса**
 - для нахождения описания сервиса может напрямую использовать универсальный идентификатор ресурса (URI) или
 - может найти описание в реестре сервиса, с последующей привязкой и вызовом сервиса
- ✓ **Посредник сервиса**
 - обеспечивает и обслуживает реестр сервиса



Сервисно-ориентированная архитектура.

Web-сервис



Сервис, предоставляемый
через www с
использованием языка XML
и протокола HTTP

- Представляет собой приложение, которое идентифицируется строкой URI. Интерфейсы и привязки данного приложения описываются и обнаруживаются с использованием XML-средств. Приложения взаимодействуют посредством обмена сообщениями, которые пересылаются с использованием интернет-протокола

Свойства WEB сервисов

- Самодостаточность
- Самоописываемость
- Могут быть опубликованы, обнаружены и вызваны через Интернет
- Модульность
- Инвариантность к способу реализации
- Открыты и основаны на стандартах, техническая основа XML и HTTP
- Имеют свободные связи
- Являются динамическими
- Обеспечивают возможность интеграции унаследованных приложений

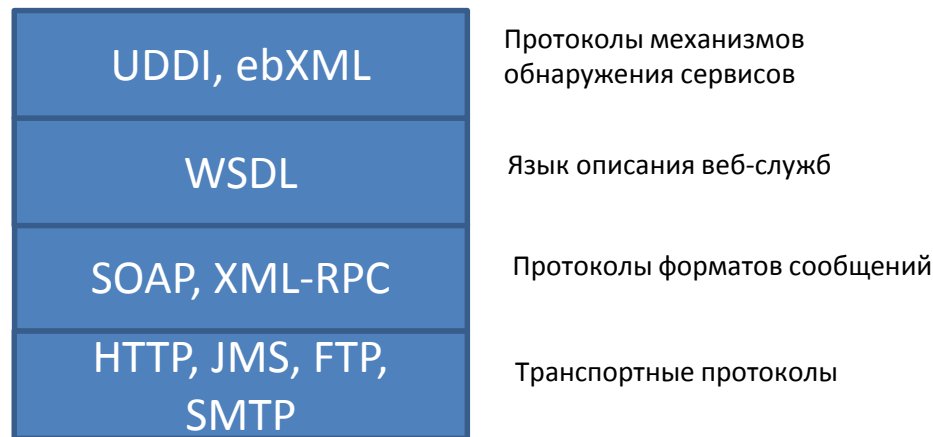
Сервисно-ориентированная архитектура.

Web-сервис

Компоненты архитектуры web-сервисов

- Провайдер сервиса (сервер)
 - Публикует, регистрирует сервис в репозитории
 - Например, как UDDI-реестр
- Пользователь сервиса (клиент)
 - Обращается к реестру с запросами и месте нахождения отдельных сервисов и способах обращения к ним
 - Может обращаться к сервису непосредственно, если известны его местонахождение и интерфейс

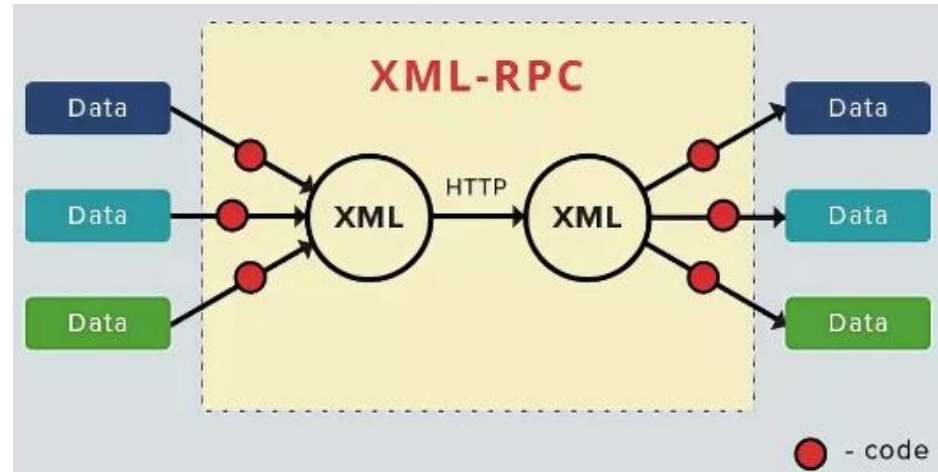
Стек протоколов



Сервисно-ориентированная архитектура.

Протокол XML-RPC

- Предшественник web-сервисов
- Вызов удаленных процедур осуществляется путем XML-сообщения
- Ответ также приходит в виде XML-сообщения



Минусы

Запросы являются самоописываемыми только частично

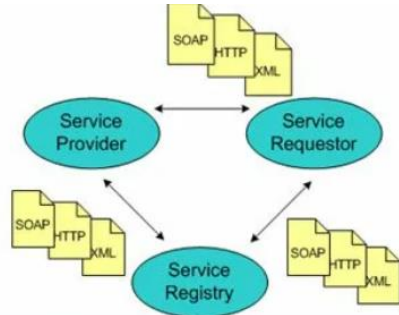
Не поддерживает работу с объектами

Разбор XML-сообщения требует специального ПО и времени

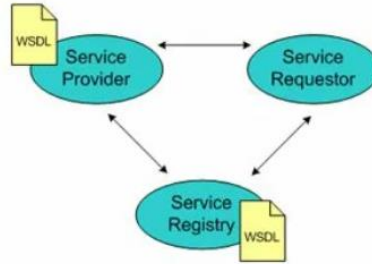
Сообщения в формате XML имеют большую избыточность

Сервисно-ориентированная архитектура.

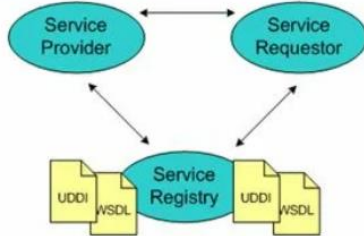
Протоколы web-сервиса



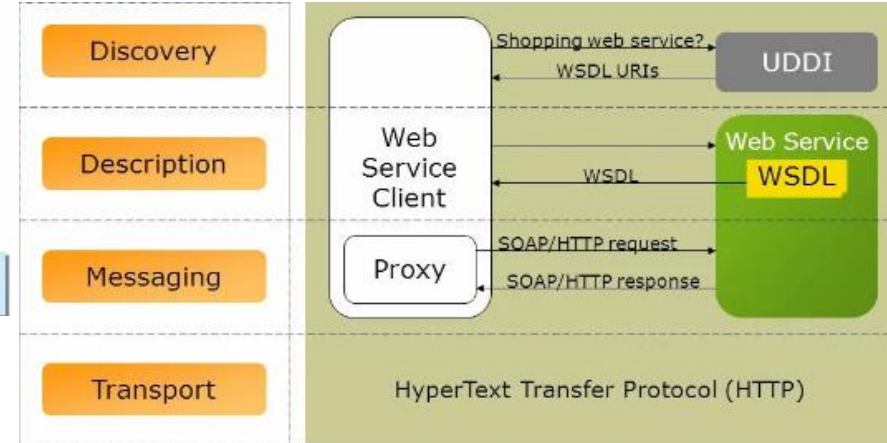
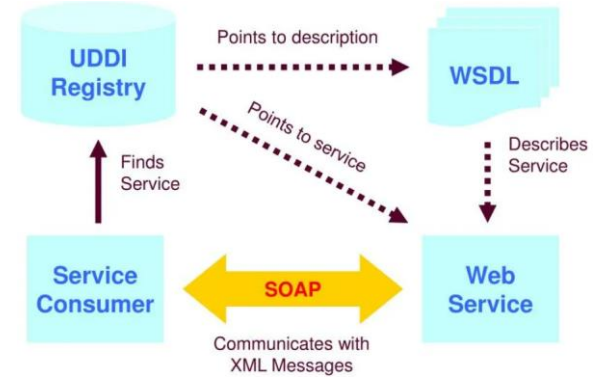
SOAP for communication



WSDL for contract & binding



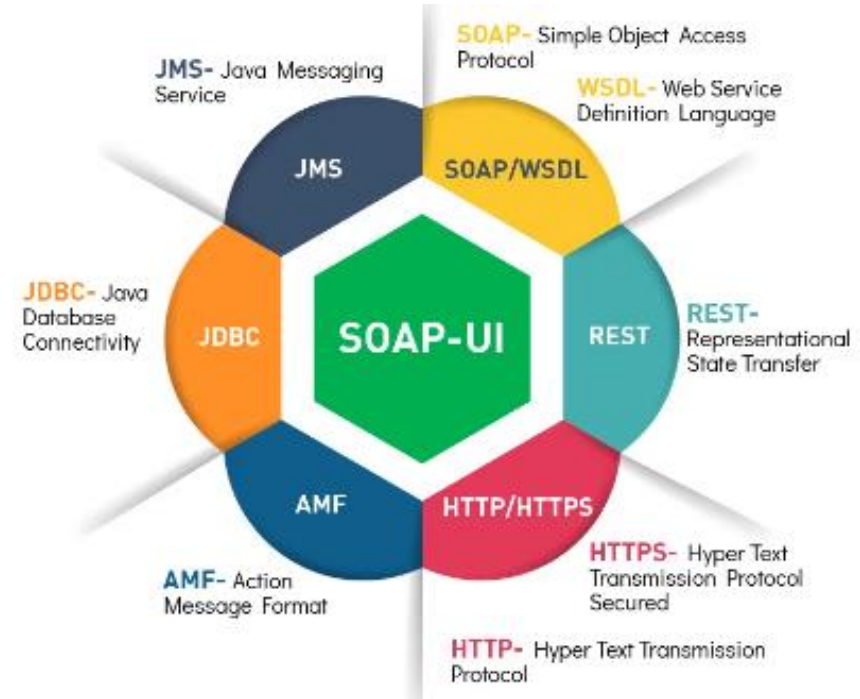
UDDI & WSDL for registration & discovery



Сервисно-ориентированная архитектура.

Протокол SOAP

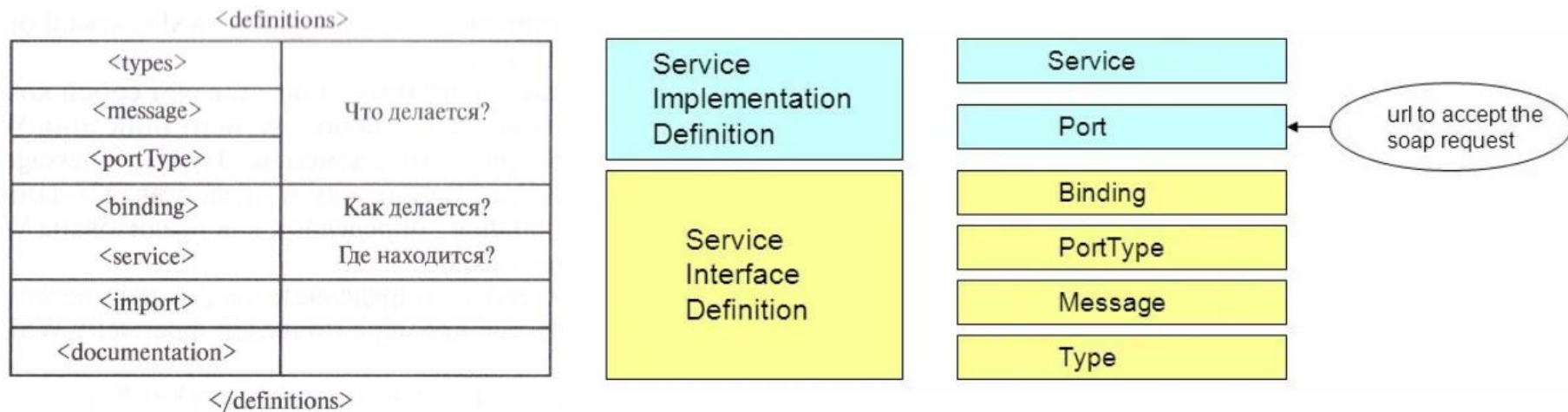
- SOAP (Simple Object Access Protocol) – это основанный на XML протокол, определяющий механизм обмена сообщениями в распределённой вычислительной среде
- Отличие от XML-RPC
 - Не различает вызов процедуры и ответ на него, но определяет формат послания
 - Самоописываемый документ
- Структура SOAP-послания
 - Заголовок (header) и тело (Body), которые помещаются в контейнер (Envelope)



Сервисно-ориентированная архитектура.

WSDL-описание

- WSDL (Web Services Description Language) - язык разметки, основанный на XML, предназначенный для описания веб сервисов.
- Используется для описания интерфейсов web-сервисов
- Представляет собой XML-документ.



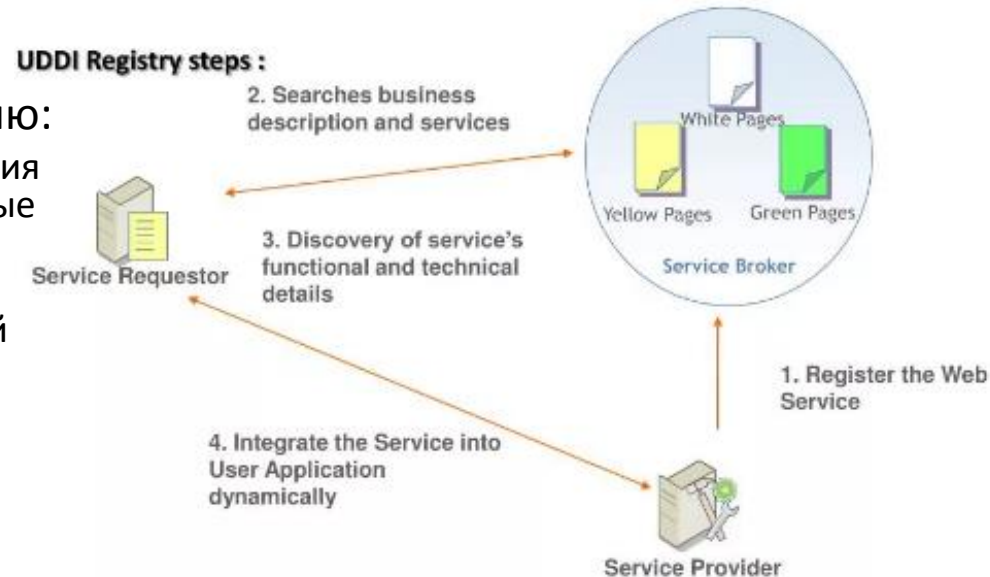
Сервисно-ориентированная архитектура.

UDDI-реестр

- Universal Description, Discovery, and Integration
- Техническая спецификация для построения распределенных репозиториях, которые позволяют отдельным организациями публиковать и находить требуемые сервисы

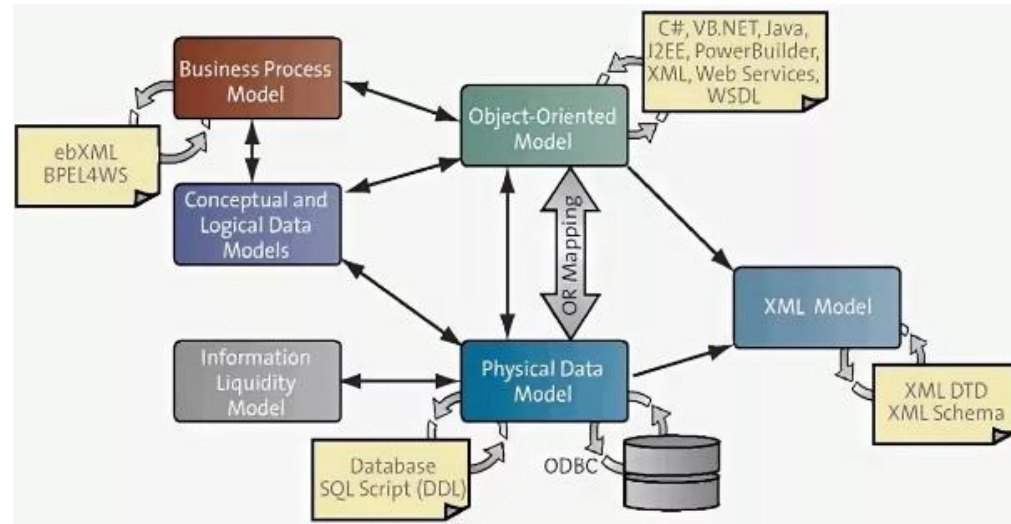
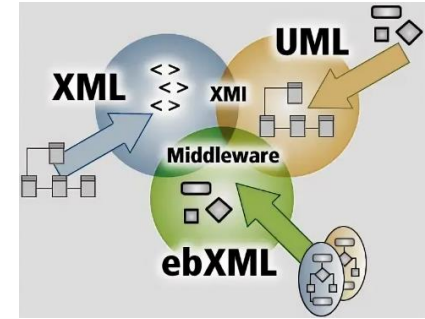
- Документ в формате XML и имеет трехуровневую организацию:

- **белые** страницы – базовая информация о бизнесе: адрес, контакты и известные идентификаторы;
- **жёлтые** страницы – промышленные категоризации на основе стандартной таксономии (классификация по типу бизнеса);
- **зеленые** страницы – техническая информация об услугах, доступных в бизнесе (способ доступа к сервисам)



Сервисно-ориентированная архитектура. Бизнес-реестр ebXML

- Электронный реестр для поддержки систем электронного бизнеса
- Протокол описания взаимодействия партнеров по типу Business to Business (B2B):
 - Parties – организации, осуществляющие взаимодействие
 - Business Collaboration – деловое сотрудничество, реализуемого сторонами
 - Collaboration Protocol Profile – заявление о сотрудничестве
 - Collaboration Protocol Agreement – соглашение о сотрудничестве



Интеграция приложений. Системы обмена данными

Асинхронная связь

- Системы, использующие асинхронные связи (Asynchronous communication)
- Отправитель после отправления сообщения немедленно продолжает работу
- Сообщение сохраняется в локальном буфере передающего хоста или ближайшего коммуникационного сервера

Синхронная связь

- Системы, использующие синхронные связи (Synchronous communication)
- Работа отправителя блокируется до того момента, когда сообщение будет доставлено получателю или сохранено в локальном буфере принимающего хоста
- Три степени «жесткости»
 - Отправитель может продолжать работу после того, как сообщение помещено во входной буфер получателя
 - Работа отправителя блокируется до момента получения сообщения непосредственно пользователем (получения подтверждения приема сообщения)
 - Работа отправителя блокируется до момента получения ответа

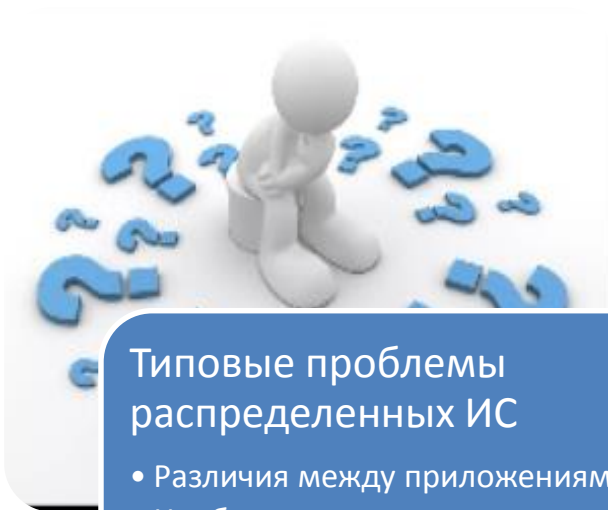
Сохранная связь

- Системы, работающие по принципу сохранной связи (Persistent communication)
- Сообщение, предназначенное для отсылки, хранится в коммуникационной системе до тех пор, пока его не удастся передать получателю
- Сообщения никогда не теряются и не пропадают

Несохранная связь

- Системы, работающие по принципу несохранной связи (Transient communication)
- Сообщение хранится в системе только в течение времени работы приложения, которые принимают и отправляют это сообщение.
- Если коммуникационный сервер не имеет физической возможности передать сообщение – оно уничтожается

Интеграция приложений



Типовые проблемы распределенных ИС

- Различия между приложениями
- Необходимость внесения изменений в код интегрирующих приложений
- Ограниченная скорость передачи данных
- Ненадежность сетевой инфраструктуры



Базовые механизмы интеграции приложений

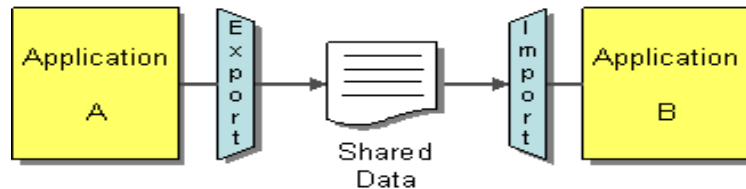
- Разделяемые файлы
- Разделяемая база данных
- Удаленный вызов процедуры и методов
- Обмен сообщениями



Интеграция приложений. Разделяемые файлы

Каким образом можно интегрировать несколько приложений так, чтобы они могли работать вместе и обмениваться информацией?

- Приложения можно интегрировать, используя файловую систему.
- Одно из приложений создает файлы, содержащие необходимую информацию, другое — считывает эти файлы.
- Формат файлов может быть любым — текст, XML, и т.п.
- Важно, чтобы формат для всех приложений был согласованный.



Достоинства

Простота реализации

Не требуется использования специальных технологий

Недостатки

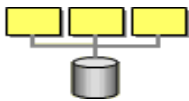
Сложность синхронизации процессов и разработки кода

Низкая скорость обмена данными

Рекомендация к использованию

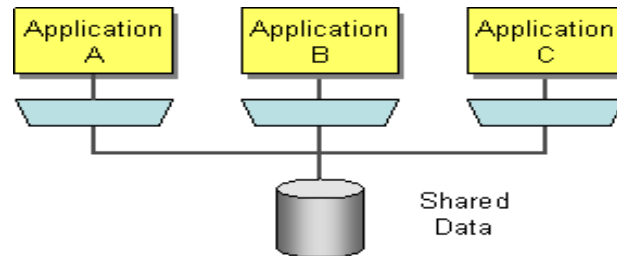
Приложения разработаны на различных платформах и с использованием различных ЯП

Обмен данными носит эпизодический характер



Интеграция приложений. Разделяемая база данных

- Каким образом можно интегрировать несколько приложений так, чтобы они могли работать вместе и обмениваться информацией?
 - Приложения можно интегрировать, используя общую базу данных.
 - В таком случае приложениям не нужно договариваться о формате данных, только о схеме данных.



Достоинства

Простота программирования

Недостатки

Сложность разработки структуры общей БД

Сложность с масштабируемостью системы

Рекомендация к использованию

Приложения разработаны на разных платформах и с использованием разных ЯП

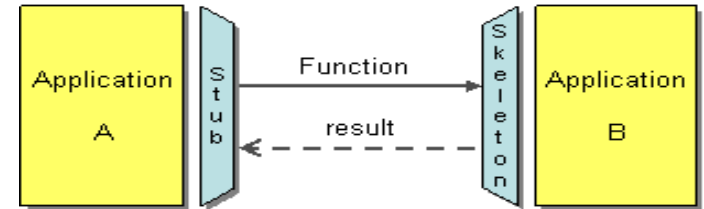
Жесткие требования к актуальности и целостности данных



Интеграция приложений.

Удаленный вызов процедуры и методов

- Каким образом можно интегрировать несколько приложений так, чтобы они могли работать вместе и обмениваться информацией?
 - Используйте RPI для обмена информацией с помощью вызова методов другого приложения удаленно.



Достоинства

Высокая скорость обмена

Инкапсуляция данных

Недостатки

Сложность организации
асинхронных взаимодействий

Рекомендации к использованию

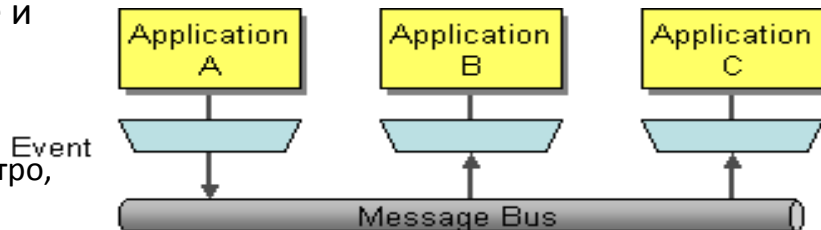
Требуется реализовать
распределенную
функциональность

Жесткие требования по
масштабируемости и скорости
взаимодействия



Интеграция приложений. Обмен сообщениями

- Каким образом можно интегрировать несколько приложений так, чтобы они могли работать вместе и обмениваться информацией?
 - Используйте механизм сообщений для отправки и получения данных.
 - Данные в сообщениях можно передавать часто, быстро, надежно и асинхронно.
 - Отправка сообщений не требует чтобы обе системы работали в одно и то же время — механизм отправки и получения позаботится о гарантированной доставке.



Достоинства

Поддержка асинхронных взаимодействий

Возможность создавать гибкие приложения

Недостатки

Относительно невысокая скорость обмена данными

Рекомендации к использованию

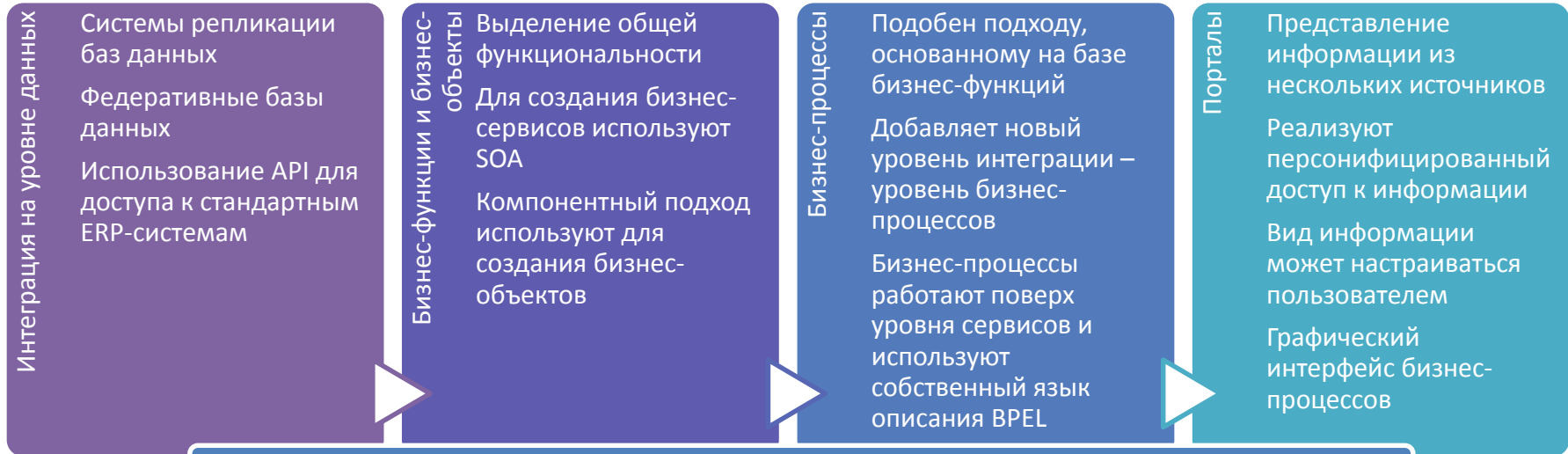
Требуется реализовать асинхронные взаимодействия

Высокие требования к модифицируемости приложений

Средние требования к скорости обмена данными

Интеграция приложений.

Задачи интеграции



Задачи интеграции корпоративных приложений

- Системы интеграции корпоративных приложений
- Enterprise Applications Integration
- Application-to-Application

Задачи интеграции приложений, функционирующих в составе разных КИС

- Системы интеграции между организациями
- Business-to-Business

Варианты интеграционных решений

• Достоинства

- Простота и возможность реализации каждой связи уникальным способом

• Недостатки

- Недостаточная гибкость и сложность поддержки многочисленных соединений
- Изменения сказываются на всех точках и нет общей модели безопасности, низкая надежность
- Сложно использовать фреймворки и поддерживать асинхронные взаимодействия

Точка-точка
(point-to-point)



• Достоинства

- Гибкость и поддержка многочисленных соединений
- Безопасность и допускают аутентификацию пользователя

• Недостатки

- Проблемы со временем передачи данных
- Трудности в устранении неполадок

Шлюз (hub-and-spoke)



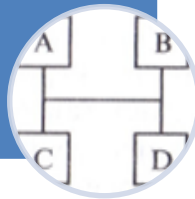
• Достоинство

- Уменьшение количества интерфейсов
- Независимость отправителя и получателя
- Относительная простота интеграции
- Маршрутизация данных
- Диагностика состояния

• Недостатки

- Требуется дополнительное ПО
- Сложные механизмы взаимодействия

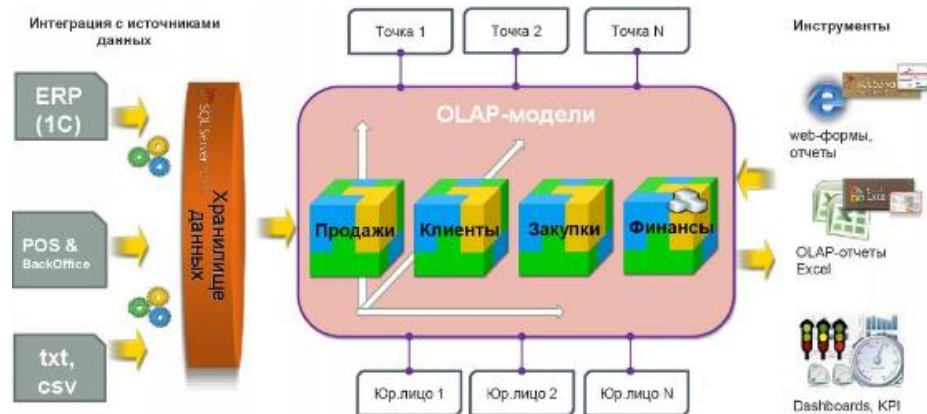
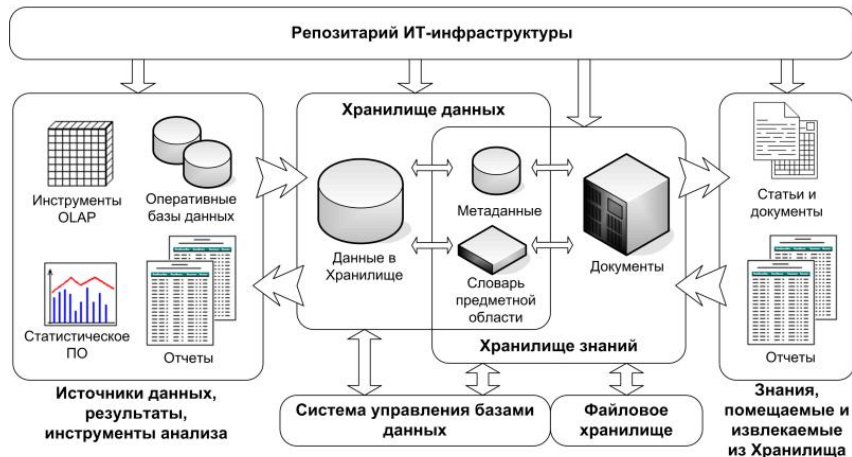
Шина (bus)



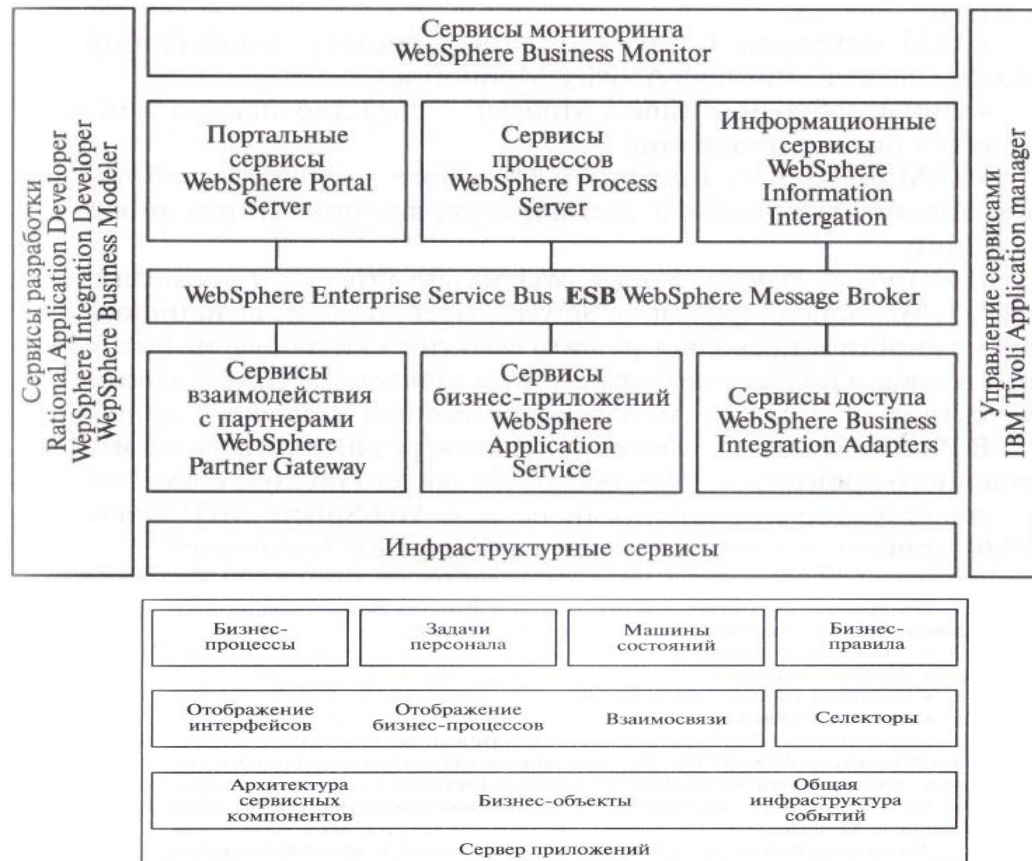
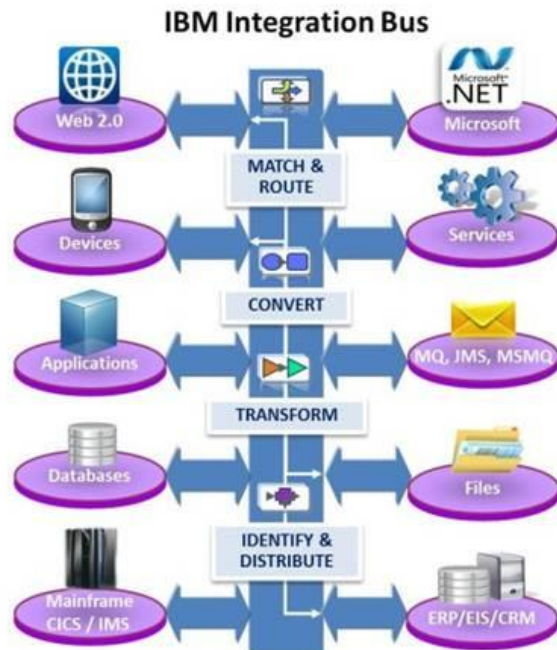
Интеграция приложений.

Архитектурная модель

- Уровень сопряжения (адаптеры и интерфейсы)
- Транспортная подсистема
- Уровень реализации бизнес-логики
- Уровень управления бизнес-процессами
- Уровень бизнес-управления (бизнес-правила, машины состояний)



Интеграционные системы на базе ESB IBM



Интеграционные системы на базе ESB 2го поколения

