Проектирование архитектур программного обеспечения

лекция 8

Зозуля А.В.

КСПТ

2014 г.

Предпосылки необходимости интеграции

- Использование нескольких приложений
- Приобретение ПО, созданного сторонними разработчиками
- Технологические отличия между приложениями
- Архитектурные отличия
- Отсутствие встроенных средств интеграции в приложениях

Типы интеграционных задач

- Информационные порталы: объединение информации из нескольких источников
- Репликация данных: доступ к одним и тем же данным их разных бизнес-компонентов
- Бизнес-функции совместного использования
- Распределенные архитектуры, ориентированные на службы
- Распределенные бизнес-процессы: различные системы в одной бизнес-транзакции
- B2B-интеграция («точка — точка»)

Проблемы интеграции приложений

- Ненадежность сети передачи данных
- Низкая скорость передачи данных
- Различия между приложениями
- Неизбежность изменений (*слабое связывание*)

Шаблоны интеграции

- **По структуре взаимодействия** — описывают основные компоненты единой интегрированной метасистемы
- **По методу интеграции** — описывают взаимодействие компонентов
- **По типу обмена данными** — описывают организацию обмена информацией между компонентами

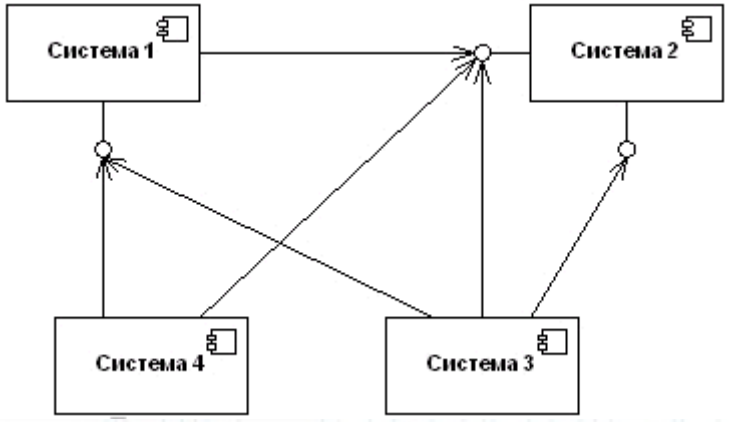
Дубина О. «Обзор паттернов проектирования»

Структурные шаблоны интеграции

- Взаимодействие «точка - точка»
- Взаимодействие «интегрирующая среда»
- Смешанный способ взаимодействия

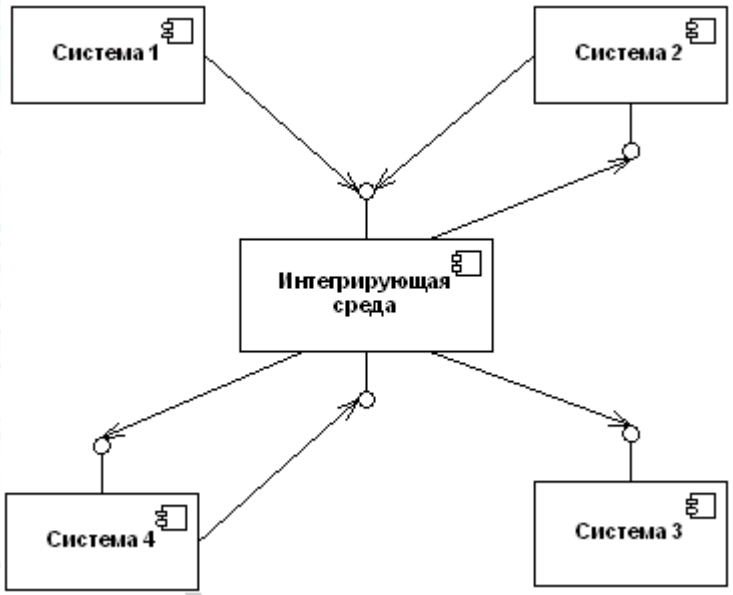
Взаимодействие «точка - точка»

- У одной из систем есть интерфейс для доступа к ней активной системы
- Применяется при стихийной интеграции систем
- Соответствует требованиям активной системы, но непригоден для использования другой системой в качестве активной



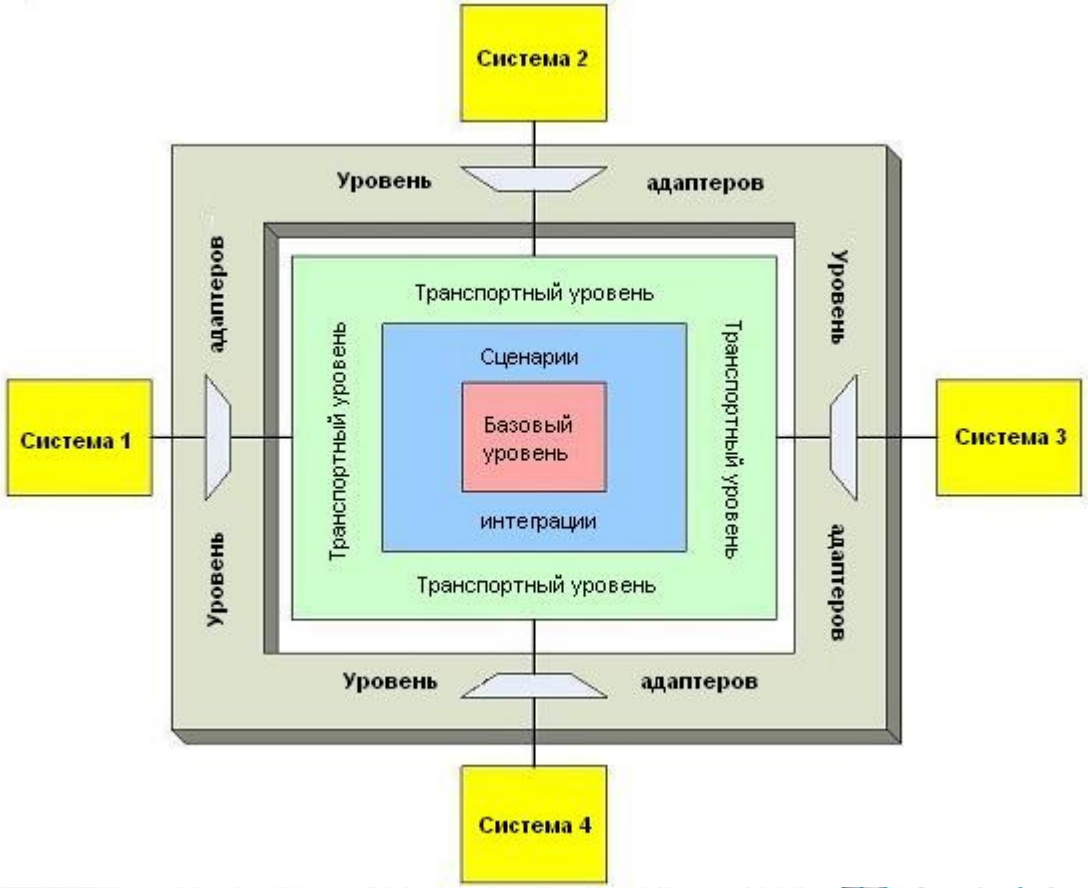
Взаимодействие «интегрирующая среда»

- Центральный компонент - интегрирующая среда, управляющая взаимодействием подсистем
- Интегрирующая среда имеет универсальный интерфейс для доступа активных систем
- Интегрирующая среда может использовать интерфейсы пассивных систем



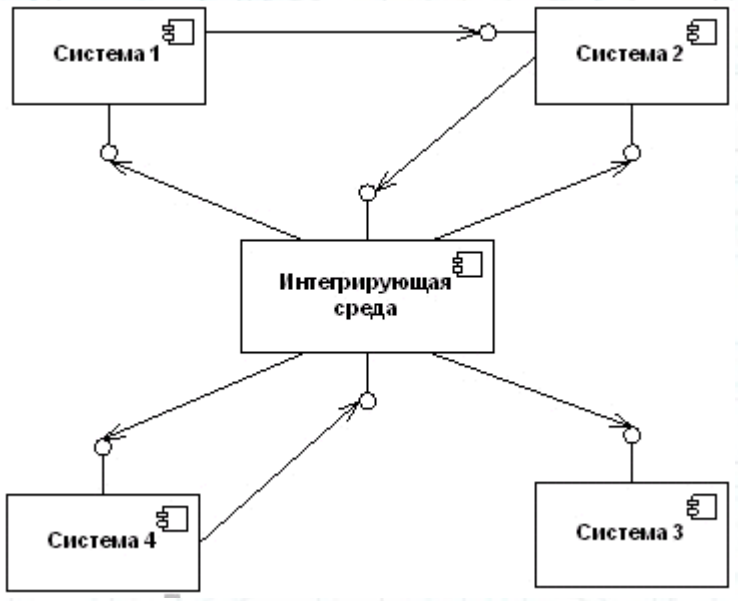
Взаимодействие «интегрирующая среда»

- **Базовый уровень** - платформа для исполнения сценариев транзакции, базовый функционал по взаимодействию приложений, службы протоколирования и мониторинга состояния
- **Уровень сценариев интеграции** - схема обмена сообщениями между системами, алгоритмы преобразования и маршрутизации
- **Транспортный уровень** - физическая доставка сообщений между компонентами
- **Уровень адаптеров** - взаимодействие с системой посредством API, генерация сообщений, передача сообщений базовому уровню посредством транспортного



Смешанный способ взаимодействия

- Интерфейсы частично могут использоваться непосредственно напрямую в обход интегрирующей среды



Шаблоны по методу интеграции

- Интеграция систем по данным (data-centric)
- Функционально-центрический (function-centric) подход
- Объектно-центрический (object-centric)
- Интеграция на основе единой понятийной модели предметной области (concept-centric)

Интеграция систем по данным

- Исторически первый в решении проблемы интеграции приложений
- Характерен для традиционных систем «**клиент-сервер**»
- Приложения объединяются в систему вокруг интегрированных данных под управлением СУБД
- Все функции прикладной обработки размещаются в клиентских программах
- Необходимость передачи больших объемов данных

Функционально-центрический подход

- Системообразующим фактором являются **сервисы** со стандартизованным интерфейсом, вокруг которых приложения объединяются в систему
- Реализуемые сервисами функции достоверны, непротиворечивы и общедоступны
- Интегрирующей средой является **сервер приложений** или монитор транзакций со стандартным API
- Общая архитектура системы является **трехзвенной**: 1) клиентское приложение; 2) функциональные сервисы; 3) СУБД
- Получил развитие в **Сервис-ориентированной архитектуре**

Сервис-ориентированная архитектура (Service-Oriented Architecture)

- Модульный подход + обнаружение компонента + согласование контракта взаимодействия
- Использование распределенных, слабо связанных заменяемых компонентов
- Компоненты имеют стандартизированные интерфейсы для взаимодействия по стандартизированным протоколам (SOAP, REST)
- Компоненты - набор веб-служб
- Интерфейсы компонентов инкапсулируют детали реализации

Объектно-центрический подход

- Основан на стандартах объектного взаимодействия (CORBA, COM/DCOM, .NET и пр.)
- Системы объединяются вокруг общедоступных **распределенных объектов** со стандартными интерфейсами
- Наличие унифицированного языка спецификации интерфейсов объектов
- Отделение реализации компонентов от спецификации их интерфейсов
- Интегрирующей средой является **брокер объектных запросов**
- Общая архитектура системы формируется на основе распределенных объектов и является **n-звенной**

Интеграция на основе единой понятийной модели предметной области

- Разрабатывается общесистемный язык взаимодействия компонентов, основанного на единой понятийной модели, описывающей объекты предметной области, их взаимосвязи и поведение
- Единая понятийная модель — база метаданных, хранящая описания интерфейсных бизнес-объектов каждого из компонентов и отношения между этими объектами
- Описания объектов и язык взаимодействия независимы от интегрирующего ПО
- Единицей информационного обмена являются сообщения
- Сообщения преобразовываются в вызовы функций дополнительной интегрирующей оболочкой с единым интерфейсом

Шаблоны по типу обмена данными

- Файловый обмен (File Transfer)
- Общая база данных (Shared Database)
- Удаленный вызов процедур (Remote Procedure Invocation)
- Обмен сообщениями (Messaging)

Файловый обмен

- Основывается на структурной концепции «точка — точка»
- Системы экспортируют общие данные в формате пригодном для импорта в другие системы (XML, JSON, YAML)
- Слабая связываемость систем, простота реализации
- «-»: Необходимость поддержки совместимости форматов обмена и определения периодичности обмена (рассинхронизация при редком обмене)

Форматы файлов

JSON

```
{
   "firstName": "John",
   "lastName": "Smith",
   "age": 25,
   "address": {
      "streetAddress": "21 2nd Street",
      "city": "New York",
      "state": "NY",
      "postalCode": "10021"
   },
   "phoneNumber": [
      {
         "type": "home",
         "number": "212 555-1234"
      },
      {
         "type": "fax",
         "number": "646 555-4567"
      }
   ]
}
```

YAML

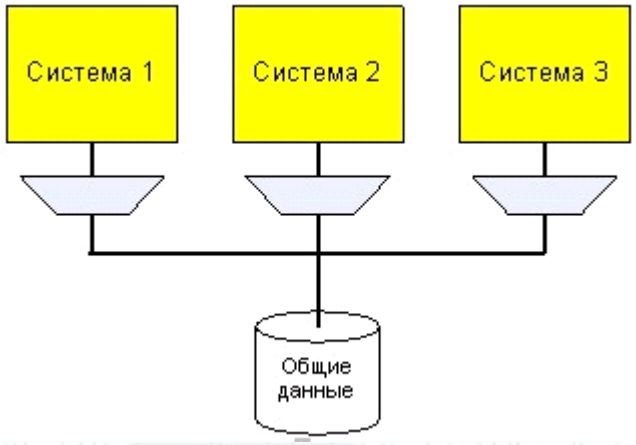
```
---
 firstName:  John
 lastName:  Smith
 age: 25
 address:
       streetAddress: 21 2nd Street
       city: New York
       state: NY
       postalCode: 10021

 phoneNumber:
       -
          type: home
          number: 212 555-1234
       -
          type: fax
          number: 646 555-4567
```

XML

```
<person>
 <firstName>John</firstName>
 <lastName>Smith</lastName>
 <age>25</age>
 <address>
  <streetAddress>21 2nd Street</streetAddress>
  <city>New York</city>
  <state>NY</state>
  <postalCode>10021</postalCode>
 </address>
 <phoneNumbers>
  <phoneNumber type="home">212 555-1234</phoneNumber>
  <phoneNumber type="fax">646 555-4567</phoneNumber>
 </phoneNumbers>
</person>
```

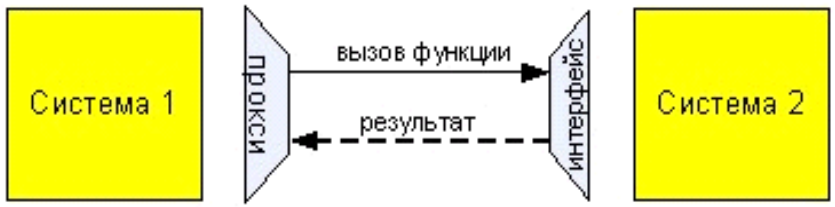
Общая база данных

- В основе метод интеграции систем по данным
- Интегрированная система приложений, работающая с едиными данными в любой момент времени
- Затруднительно интегрировать существующие системы, удобно использовать для вновь создаваемых
- «-»: Создание общей схемы данных; коммерческое ПО с собственной БД; БД становится узким местом; БД — неинкапсулированные данные



Удаленный вызов процедур

- Реализация объектно-центрического подхода
- Приложения представляются в виде объектов и интегрированы на уровне функций
- Изменение данных в другой системе происходит посредством вызова функций
- Удаленный вызов — принцип инкапсуляции, примененный к приложениям (CORBA, COM, .NET Remoting, Java RMI, SOAP)
- «-»: Каждая из систем самостоятельно заботится о поддержке данных в корректном состоянии; сильная связность компонентов

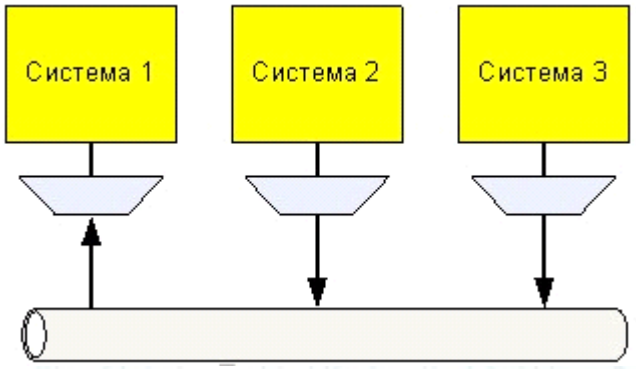


Типы обмена данными

	Передача файла	Общая БД	Удаленный вызов процедуры
Доступ к общим данным	+	+	--
Доступ к общей функциональности	--	--	+
Слабая связность	+	--	--
Скорость	--	+	+
Надежность	--	+	--

Обмен сообщениями

- Основан на асинхронном обмене сообщениям посредством шины данных
- Предназначен для интеграции независимых приложений с минимальными доработками существующих систем посредством вызова функций
- Является реализацией подхода интеграция на основе единой понятийной модели предметной области
- За логику интеграции отвечает интеграционная шина



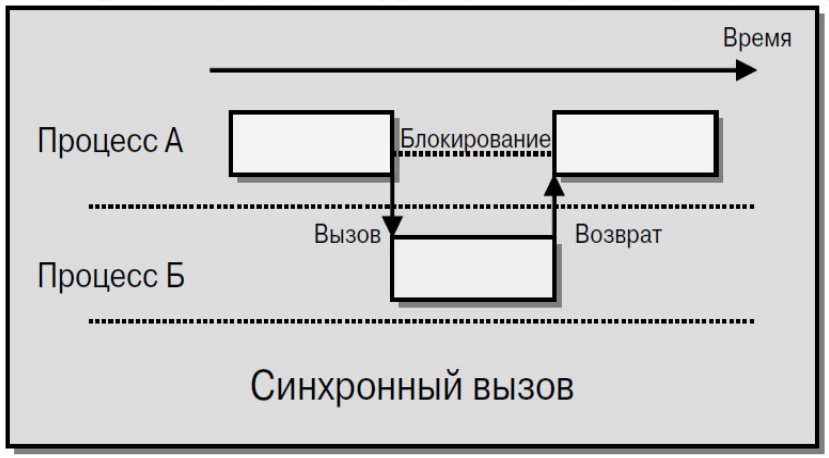
Обмен сообщениями

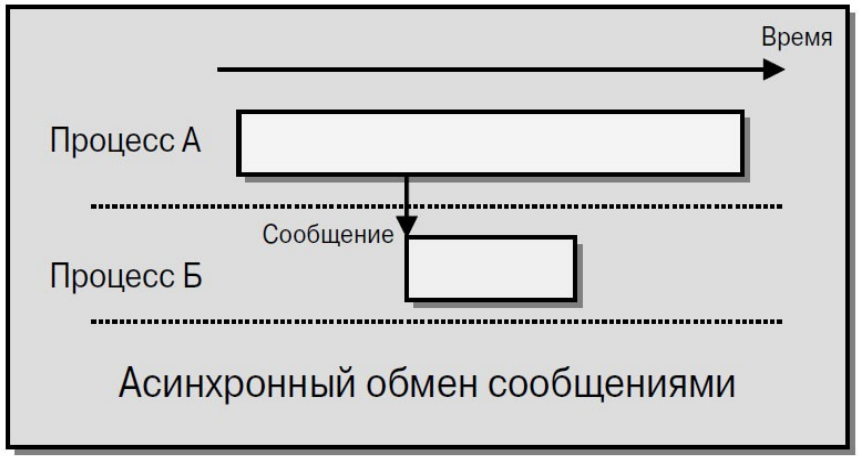
- Не требуется одновременной доступности отправителя и получателя
- Слабое связывание объединяемых приложений: преобразование при передаче
- Различные способы доставки: широковещательная рассылка, маршрутизация одному получателю, группе
- Отсутствие синхронных задержек
- «-»: требует «асинхронного» мышления, сложнее тестировать

Обмен сообщениями

- Технология высокоскоростного **асинхронного** взаимодействия между программами с гарантией доставки
- **Сообщения** — пакеты данных, которыми обмениваются программы. Сообщение состоит из заголовки и тела.
- **Канал** (очередь) — логический маршрут, объединяющий программы и использующийся для транспортировки сообщений
- **Отправитель** (поставщик) — программа, отправляющая сообщение путем размещения его в канале
- **Получатель** (потребитель) — программа, получающая сообщение путем его считывания из канала

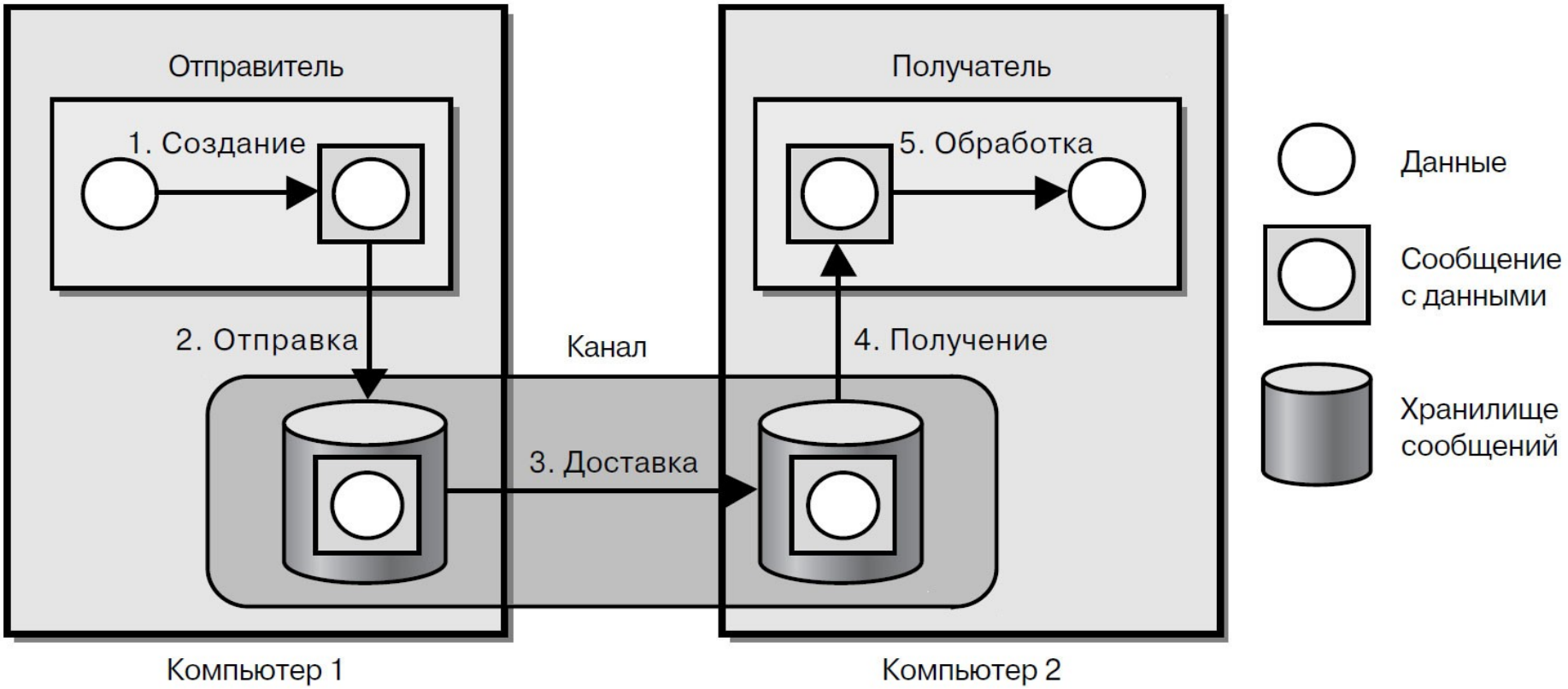
Асинхронное взаимодействие





- Несколько потоков выполнения: увеличение производительности, затруднение отладки
- Результат выполнения (если есть) возвращается обратными вызовом: увеличение производительности, вызывающий процесс должен быть готов к получению ответа в любой момент, а также помнить контекст
- Выполнение в любом порядке: обработка результатов с учетом источника и времени получения

Процедура передачи сообщения



Грегор Хоп, Бобби Вульф
«Шаблоны интеграции корпоративных приложений»

Система обмена сообщениями

- Программная система, обеспечивающая функциональную часть обмена сообщениями
- Связующее ПО, ориентированное на обмен сообщениями (Message-Oriented Middleware — MOM)
- Имеет много общего с БД
- Гарантирует доставку информации за счет повторной отправки сообщения
- Реализует важнейшие концепции обмена сообщениями:
  - «Отправить и забыть»
  - Передача с промежуточным хранением

Преимущества обмена сообщениями

- **Удаленное взаимодействие**: системы делегируют ответственность за сериализацию и передачу данных
- **Платформенная/языковая интеграция**: универсальное связующее звено между платформами, технологиями, окружениями, языками программирования
- **Асинхронное взаимодействие**: «отправил и забыл» (send-and-forget), подтверждение только о помещении сообщения в канал
- **Рассогласование во времени**: скорость размещения вызовов отправителем не ограничена скоростью их обработки получателем
- **Регулирование нагрузки**: получать контролирует скорость обработки сообщений из очереди

Преимущества обмена сообщениями

- **Надежное взаимодействие**: передача с промежуточным хранением (store-and-forward), посылка осуществляется до тех пор, пока сообщение не будет получено
- **Работа без подключения к сети**: накопление данных до тех пор, пока получатель не будет доступен
- **Посредничество**: приложению необходимо поддерживать связь только с системой обмена сообщениями
- **Управление потоками**: использование обратного вызова для доставки результата запроса позволяет минимизировать количество заблокированных потоков

Недостатки асинхронного обмена сообщениями

- Сложная модель программирования
- Порядок доставки сообщений
- Необходимость реализации синхронной модели
- Производительность
- Ограниченная поддержка программными платформами
- Зависимость от компании-разработчика

Основные понятия в обмене сообщениями

- Каналы (Message Channel)
- Сообщения (Message)
- Каналы и фильтры (Pipes and Filters)
- Маршрутизация (Message Router)
- Преобразование (Message Translator)
- Конечные точки (Message Endpoint)

Сервисная шина предприятия (Enterprise Service Bus)

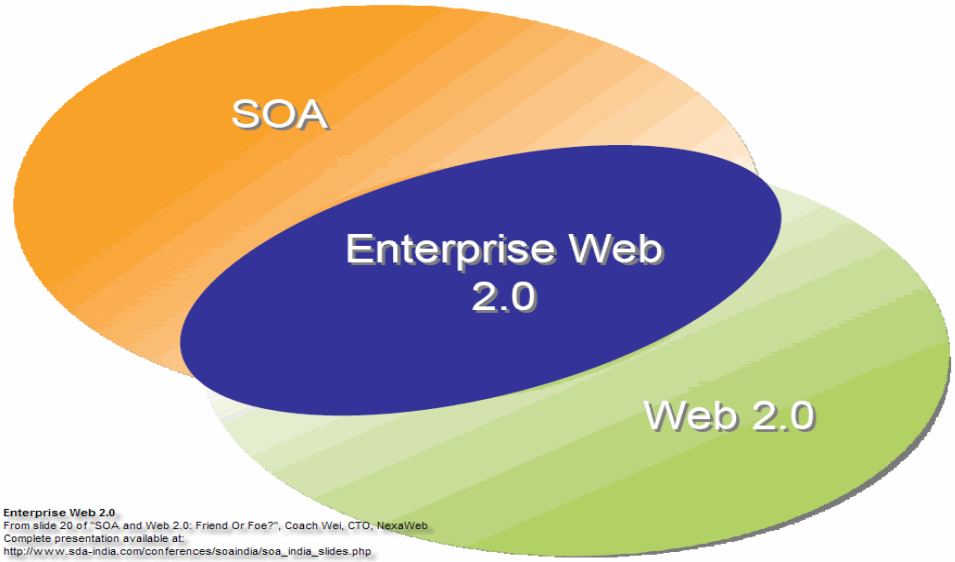
- ESB — связующее ПО, обеспечивающее централизованный и унифицированный событийно-ориентированный обмен сообщениями между различными информационными системами на принципах сервис-ориентированной архитектуры (SOA)
- ESB — единая точка обмена сообщениями между различными системами, обеспечивающая транзакционный контроль, преобразование данных, сохранность сообщений

Сервисная шина предприятия

- Поддержка синхронного и асинхронного способа вызова служб
- Использование защищённого транспорта, с гарантированной доставкой сообщений, поддерживающего транзакционную модель
- Статическая и алгоритмическая маршрутизация сообщений
- Доступ к данным из сторонних информационных систем с помощью готовых или специально разработанных адаптеров
- Обработка и преобразование сообщений
- Оркестровка и хореография служб
- Механизмы контроля и управления (аудиты, протоколирование)

Мэшап-сервис

- Веб-приложение, объединяющее данные из нескольких источников в один интегрированный инструмент
- Веб-реализация шаблона проектирования «Фасад»
- Обмен данными через открытый интерфейс или API
- Типы мешапов: пользовательские, мэшапы данных, бизнес-мэшапы, телеком-мэшапы и др.

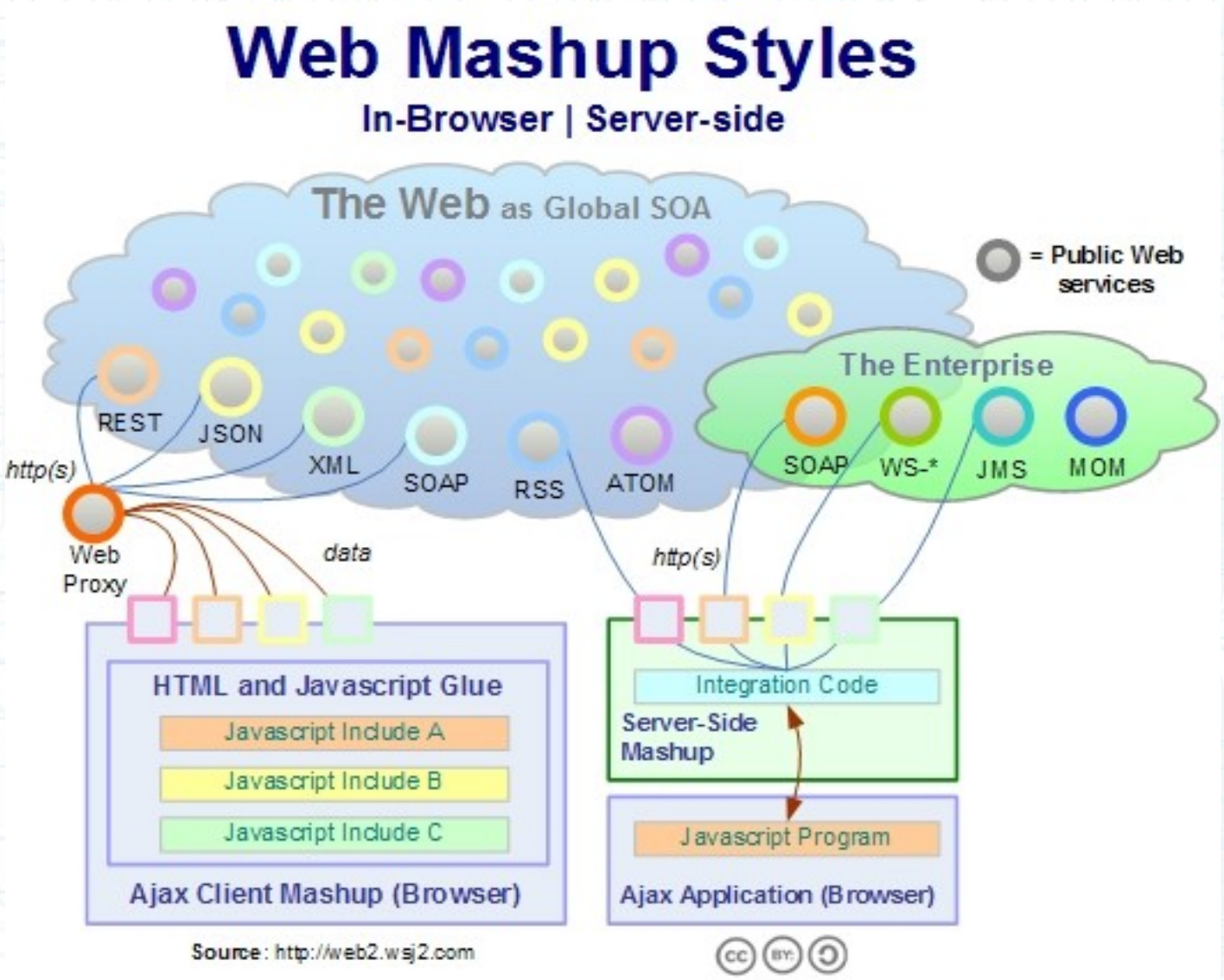


Enterprise Web 2.0
From slide 20 of "SOA and Web 2.0: Friend Or Foe?", Coach Wei, CTO, NexaWeb
Complete presentation available at:
http://www.sda-india.com/conferences/soaindia/soa_india_slides.php

Архитектура веб-мэшапов

- **Провайдер содержимого** — источник данных: данные доступны через API, RSS, REST и веб-сервисы
- **Мэшап-сайт** — веб-приложение, предлагающее новый сервис, использующий не принадлежащие ему источники данных
- **Браузер клиента** — пользовательский интерфейс мэшапа
- **Триггеры и действия**

Архитектура веб-мэшапов



Тест