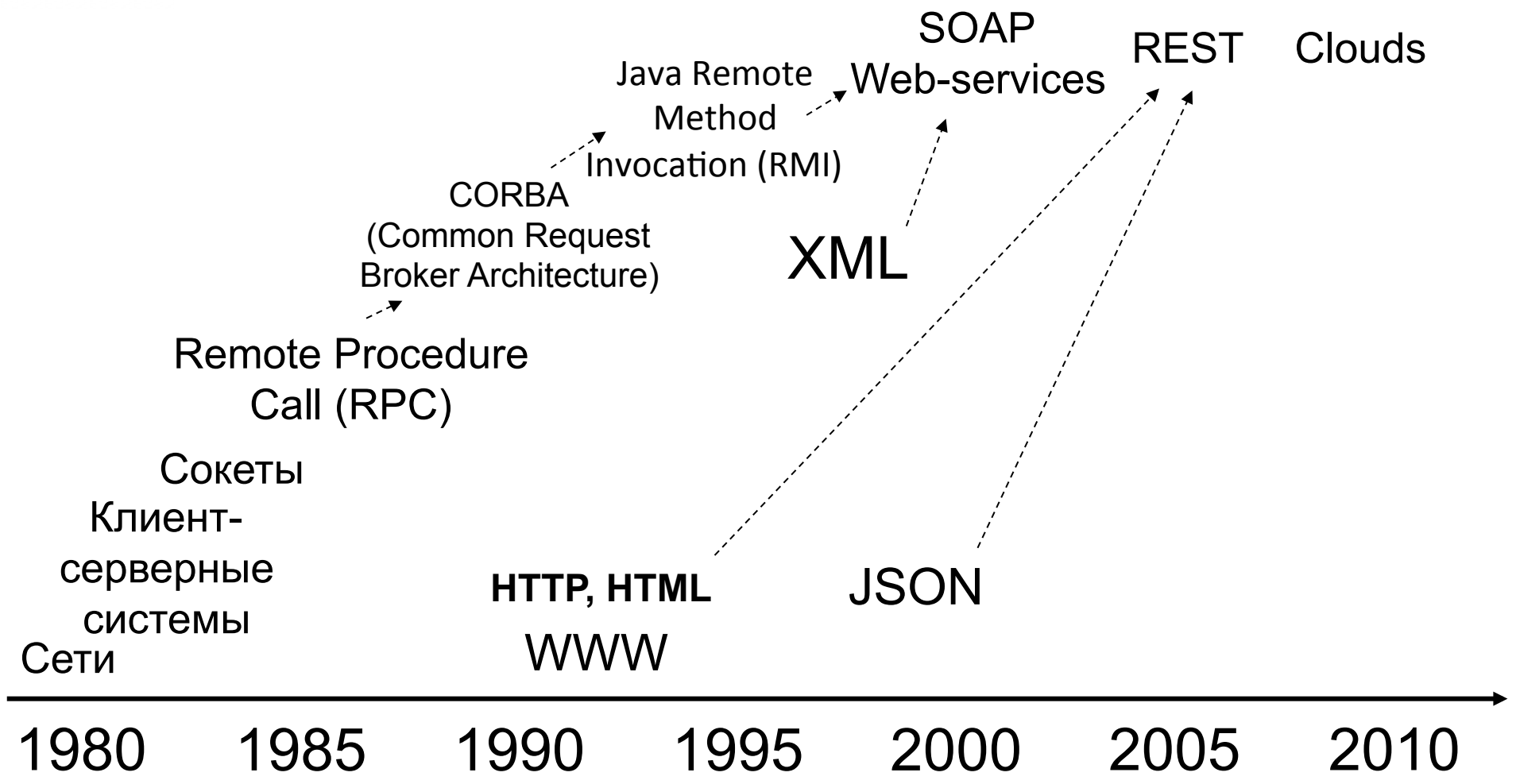
Распределенные вычислительные системы

Сервис-ориентированная архитектура

Стандарты, протоколы, архитектуры РВС



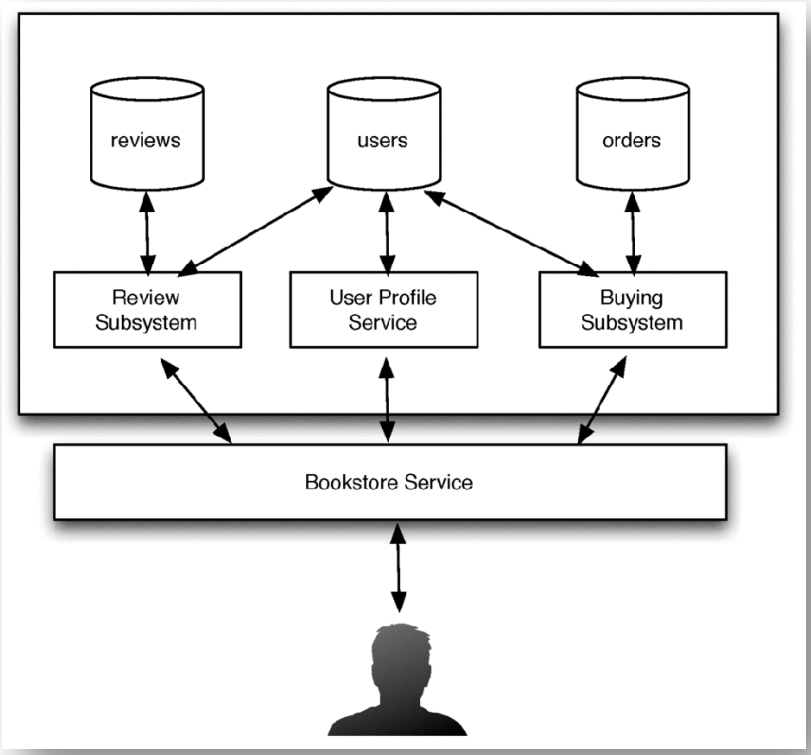
Введение: письмо Джеффа Безоса (2002)

- Все команды с настоящего момента обязаны предоставлять данные и функциональность через сервисные интерфейсы.
- Команды должны взаимодействовать друг с другом посредством этих интерфейсов
- Другие методы меж-процессной коммуникации запрещены: никакого прямого доступа к чужой оперативной памяти, никакого прямого доступа к чужим хранилищам данных и др. Всё взаимодействие – только через сервисные интерфейсы по сети.
- Не важно, какую технологию вы используете (HTTP, Corba…)
- Все интерфейсы сервисов должны быть спроектированы таким образом, чтобы быть расширяемыми. Команды должны разрабатывать интерфейсы так, чтобы сервисы можно было предоставить потребителям из внешнего мира. Без исключений.
- Любой, кто не следует этим правилам будет уволен.
- Спасибо, приятного дня!

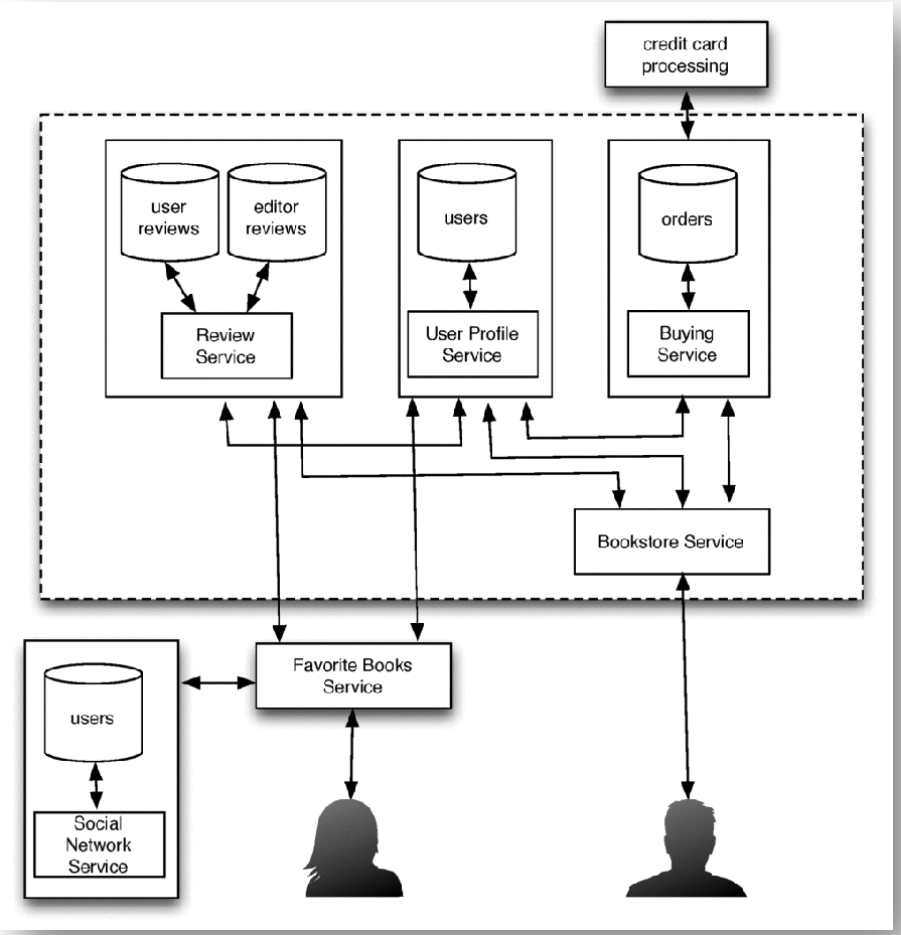
Facebook (2007) и Google+

- Аналогичный переход сделал Facebook в 2007 году
- Была запущена Facebook Platform, позволив сторонним программистам разрабатывать приложения, которые могли получать доступ к социальному графу и другим данным пользователей в Facebook.
- Google+ очень долго ругали за отсутствие API, но когда он был выпущен в сентябре 2011 года он оказался «монолитным» и абсолютно не «SOA-like», обеспечивая лишь простейший доступ к профилю пользователя и всему, что он видит.

Хранилище VS SOA



«Хранилище»



SOA

Сервис-ориентированная архитектура распределенных систем

Сервис-ориентированная архитектура (SOA)

- ***Сервис ориентированная архитектура*** – это парадигма организации и использования распределенных функциональных возможностей, которые могут предоставляться различными владельцами
  - Определение организации OASIS (Organization for the Advancement of Structured Information Standards)
- По существу, SOA означает, что *компоненты приложения* представляют собой взаимодействующие *сервисы*, которые могут быть использованы *независимо* друг от друга, или же вообще могут быть *отделены* для реализации других приложений.

Стили API Веб-сервисов

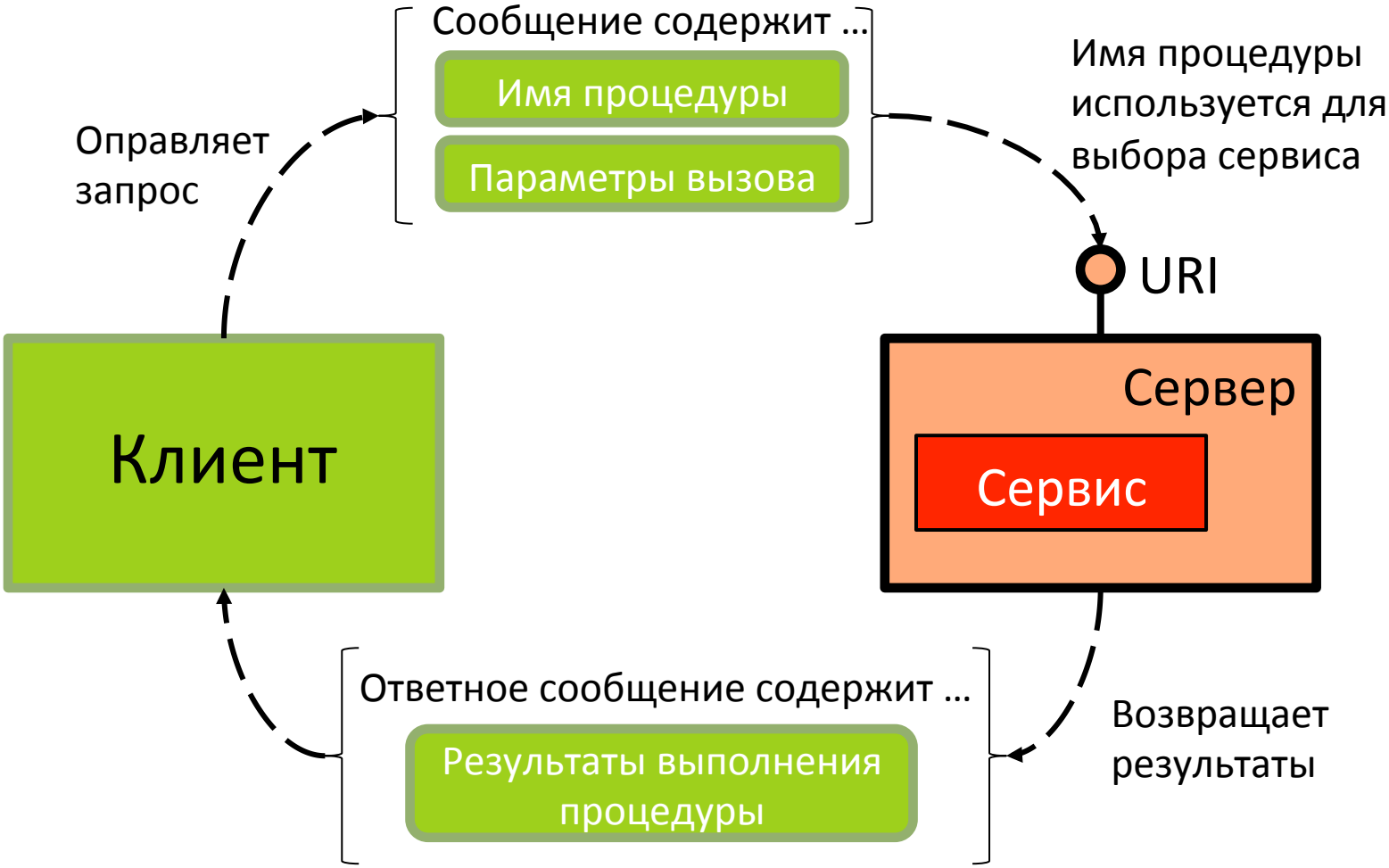
Стиль Веб-сервиса	Проблема
RPC API	Как клиенты могут выполнить удаленные процедуры посредством HTTP?
API сообщений	Как клиенты могут отправлять команды, нотификации или другую информацию удаленной системе по HTTP, но не привязывая себя к удаленным процедурам?
API ресурсов	Как клиент может манипулировать данными, предоставляемыми удаленной системой по HTTP, без связывания себя с удаленными процедурами, а также без необходимости создания специального проблемно-ориентированного API?

RPC API

Стили API Веб-сервисов: RPC API

- Необходимо определить сообщения, которые бы описывали:
  - *удаленные процедуры*,
  - а *также набор элементов, которые формируют набор параметров* для удаленной процедуры.
- Клиент должен отправить сообщение с этой информацией по указанному URI для выполнения процедуры.

Стили API Веб-сервисов: RPC API



Обмен сообщениями

- Использование низкоуровневого транспортного протокола для обмена сообщениями (BSD Socket API) приводило к необходимости сложной трансформации данных, чтобы типы данных, определенные у клиента можно было использовать на сервере.
  - *На разных платформах могут использоваться различные методы кодировки данных (ASCII , EBCDIC , UTF-8, UTF-16, Endianness) для представления и хранения данных.*
- Технологии CORBA и DCOM решили предложили решения для данной проблемы, посредством собственных абстракций над низкоуровневыми типами данных.
- Но появились проблемы при связи между различными инфраструктурами (CORBA и .NET) + обмен сообщениями через Интернет (т.к. часто на брандмауэре закрыты все порты, кроме 80-го).

RPC платформы

- Применение протокола HTTP решает многие из этих проблем, т.к. обеспечивает возможность взаимодействия клиентов и серверов, базирующихся на различных платформах через Интернет.
- Для организации RPC посредством HTTP существует множество реализаций на разных языках:
  - JAX-WS (Java)
  - WCF (Microsoft .NET)
- Они позволяют создавать простые веб-процедуры и клиентов к ним без необходимости понимания форматов данных (XML, JSON), методов кодирования (UTF-8 и др.) или описания структуры сервисов (WSDL).

Методы организации RPC

- Есть 2 основных подхода для описания контрактов таких сервисов:
  - *XML Веб-сервисы:* наиболее стандартный подход, использующий язык WSDL (Web Services Description Language) и спецификации WS-Policy и WS-Security для определения различных методов авторизации, шифрования и др.
  - Не-XML Веб-сервисы (RPC): это целый спектр различных протоколов RPC, начиная со стандарта JSON-RPC и заканчивая множеством пропиретарных или открытых стандартов от различных поставщиков.

JSON-RPC

- JSON-RPC (JavaScript Object Notation Remote Procedure Call — JSON-вызов удаленных процедур) — протокол удаленного вызова процедур, использующий JSON для кодирования сообщений.
- JSON-RPC работает отсылая запросы к серверу, реализующему протокол. Клиентом обычно является программа, которой нужно вызвать метод на удаленной системе. Все передаваемые данные — простые объекты, сериализованные в JSON

JSON-RPC – Запрос и ответ

- **Запрос** должен содержать три обязательных свойства:
  - *method* — Строка с именем вызываемого метода.
  - *params* — Массив объектов, которые должны быть переданы методу, как параметры.
  - *id* — Значение любого типа, которое используется для установки соответствия между запросом и ответом.

- **Ответ** должен содержать следующие свойства:
  - *result* — Данные, которые вернул метод. Если произошла ошибка во время выполнения метода, это свойство должно быть установлено в null.
  - *error* — Код ошибки, если произошла ошибка во время выполнения метода, иначе null.
  - *id* — То же значение, что и в запросе, к которому относится данный ответ.

JSON-RPC – пример

--> обозначает данные, отправленные серверу (запрос)

<-- обозначает ответ

```
...
--> {"method": "postMessage", "params": ["Hello all!"], "id": 99}
<-- {"result": 1, "error": null, "id": 99}
<-- {"method": "handleMessage", "params": ["user1", "Ok!"], "id": null}
<-- {"method": "handleMessage", "params": ["user3", "gotta go"], "id": null}
--> {"method": "postMessage", "params": ["I have a question:"], "id": 101}
<-- {"method": "userLeft", "params": ["user3"], "id": null}
<-- {"result": 1, "error": null, "id": 101}
...
```

Реализация JSON-RPC

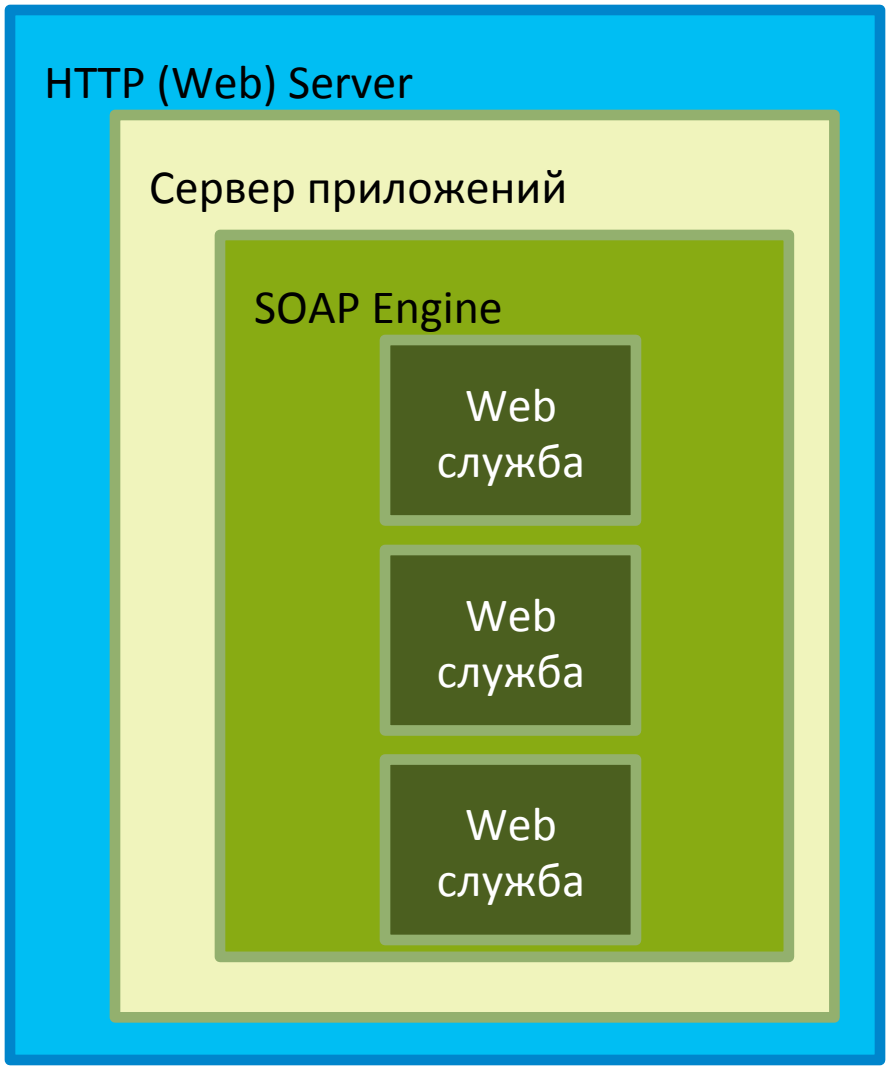
Название	версия JSON-RPC	Описание	Язык(и), Платформы
JSON-RPC.NET	2.0	Быстрый JSON-RPC сервер. Поддерживает сокеты, именованные сокеты и HTTP с помощью ASP.NET требует Mono или .NET Framework 4.0.	.NET
Jayrock	1.0	Серверная реализация JSON-RPC 1.0 для версий 1.1 и 2.0 Microsoft .NET Framework.	.NET
jsonrpc-c	2.0	Реализация JSON-RPC через TCP сокеты (только сервер).	C
libjson-rpc-cpp	2.0	C++ JSON-RPC фрэймворк, поддерживающий клиентскую и серверную стороны через HTTP.	C++
Phobos	2.0	Реализация для Qt/C++. Абстрагирует уровень передачи данных (готовые к использованию классы для TCP и HTTP).	C++
qjsonrpc	2.0	Реализация для Qt/C++. Поддерживает соединения между сообщениями и QObject слотами (как QDBus, qxtrpc) использует новые JSON классы, включенные в Qt5.	C++
JSON Toolkit	2.0	Реализация на Delphi	Delphi
go/net/rpc	?	Реализация JSON-RPC стандартной библиотеки Go	Go
jsonrpc4j	2.0	Java реализация JSON-RPC 2.0 поддерживает как сокеты, так и HTTP соединение.	Java
json-rpc	1.0	Базовая Java/JavaScript реализация, которая хорошо интегрируется в Android/Servlets/Standalone Java/JavaScript/App-Engine приложения.	Java / JavaScript
jpoxy	2.0	Простая Java реализация JSON-RPC созданная для упрощения реализации доступа к POJOs через сырой RPC фрэймворк.	Java
JSON Service	2.0	JSON-RPC серверная реализацияс поддержкой Service Mapping Description. Хорошо интегрируется с Dojo Toolkit и Spring Framework.	Java
JSON-RPC 2.0	2.0	Легкая Java library для разбора и сериализации JSON-RPC 2.0 сообщений (open source). Несколько реализация на сайте. (Base, Client, Shell, ...)	Java
java-json-rpc	2.0	Реализация для J2EE серверов.	Java
lib-json-rpc	2.0	Реализация servlet, client, JavaScript	Java
simplejsonrpc	2.0	Простой JSON-RPC 2.0 Servlet, обслуживающий методы класса.	Java
gson-rmi	2.0	Легковесный, независимый от способа передачи RMI фрэймворк разработанный для распределенных вычислений.	Java
jsonrpcjs	2.0	JavaScript клиентская библиотека для JSON-RPC 2.0, Не имеет зависимостей.	JavaScript
easyXDM	2.0	Библиотека для cross-domain соединений с поддержкой RPC. Поддерживает все браузеры postMessage, nix, frameElement, window.name, и FIM, очень проста в использовании.	JavaScript
Dojo Toolkit	1.0+	Предоставляет поддержку JSON-RPC	JavaScript
Pmrpc	2.0	JavaScript библиотека для использования в HTML5 браузерах. Реализация JSON-RPC, используя HTML5 postMessage API для передачи сообщений.	JavaScript
qooxdoo	2.0	Имеет JSON-RPC реализацию с опциональными бэк-эндами на Java, PHP, Perl и Python.	JavaScript, Java, PHP, PERL, & Python
JSON-RPC Реализация на JavaScript	2.0	Поддерживает JSON-RPC через HTTP и TCP/IP.	JavaScript
jabsorb	2.0	Легковесный Ajax/Web 2.0 JSON-RPC Java фрэймворк, расширяющий JSON-RPC протокол дополнительной ORB функциональностью, такой как поддержка циклических зависимостей.	JavaScript, Java
The Wakanda platform	2.0	Поддерживает JSON-RPC 2.0 клиент внутри Ajax Framework и JSON-RPC 2.0 сервис в серверном JavaScript	JavaScript
Deimos	1.0+2.0	Серверная реализация для Node.js/JavaScript.	JavaScript
Barracuda Web Server	2.0	Barracuda Web Server's интегрированный	Lua
DeferredKit	1.0	Поддерживает JSON-RPC 1.0 клиент.	Objective-C
Demiurgic	2.0	JSON-RPC 2.0 клиент для Objective-C	Objective-C
Oxen iPhone Commons JSON components	1.0	JSON-RPC 1.0 клиент для Objective-C	Objective-C
objc-JSONRpc	2.0	Objective-c JSON RPC клиент. Поддерживает уведомления, простые вызовы и множественные вызовы.	Objective-C
JSON::RPC	2.0	Реализация сервера JSON RPC 2.0	Perl
json-rpc-perl6	2.0	Клиент и сервер.	Perl 6
php-json-rpc	2.0	Простая PHP реализация JSON-RPC 2.0 через HTTP клиента.	PHP
JQuery JSON-RPC Server	2.0	JSON-RPC сервер, специально сделанный для работы с Zend Framework JSON RPC Server.	PHP, JavaScript
jsonrpc2php	2.0	PHP5 JSON-RPC 2.0 базовый класс и пример сервера	PHP
tivoka	1.0 + 2.0	Универсальный клиент/серверная JSON-RPC библиотека для PHP 5+.	PHP
junior	2.0	Client/server библиотека для JSON-RPC 2.0	PHP
json-rpc-php	2.0	Client/server библиотека для JSON-RPC 2.0	PHP
JSONRpc2	2.0	Реализация с «dot magic» для PHP (= поддержка группировки методов и разделения точками)	PHP
GetResponse jsonRPCClient	2.0	Объектно-ориентированная реализация клиента	PHP
zoServices	2.0	PHP, Node.js и JavaScript реализация JSON-RPC 2.0	PHP, JavaScript, Node.js
json-rpc2php	2.0	Серверная и клиентская реализация для PHP. Содержит JavaScript клиент, использующий jQuery	PHP, JavaScript
jsonrpc-php	2.0	JSON-RPC реализация для PHP	PHP
php-json-rpc	2.0	Реализация JSON-RPC 2.0.	PHP
Django JSON-RPC 2.0	2.0	JSON-RPC сервер для Django	Python
Pyjamas		JSON-RPC клиентская реализация.	Python
Zope 3	1.1	JSON RPC реализация для Zope 3	Python
jsonrpclib	2.0	JSON-RPC клиентский модуль для Python.	Python
tornadorpc	2.0	Поддерживает JSON-RPC требует Tornado web server.	Python
tinyrpc	2.0	Поддерживает JSON-RPC через TCP, WSGI, ZeroMQ и др. Разделяет передачу данных от обработки сообщений, может работать без пересылки сообщений.	Python
jsonrpc	2.0	JSON-RPC 2.0 для Python + Twisted.	Python
bjsonrpc	1.0+	Реализация через TCP/IP (асинхронная, двунаправленная)	Python
Barrister RPC	2.0	JSON-RPC реализация клиента и сервера	Python, Ruby, JavaScript (Node.js + web browser), PHP, Java
pyramid_rpc	2.0	Гибкая JSON-RPC реализация интегрированная в Pyramid web application. Работает с Pyramid's системой авторизации.	Python
rjr	2.0	JSON-RPC через TCP/UDP, HTTP, WebSockets, AMQP, и прочие.	Ruby (EventMachine) сервер с Ruby и JavaScript клиентами.
jimson	2.0	Клиент и сервер для Ruby	Ruby
JSON-RPC Objects	1.0+	Реализация только объектов (без клиента и сервера).	Ruby
JSON-RPC RT	2.0	Полная поддержка JSON-RPC 2.0 через TCP.	Windows Runtime (WinRT)
XINS	2.0	С Версии 2.0, поддерживает JSON и JSON-RPC.	XML

XML (SOAP) Веб-сервисы

XML Веб-сервисы

- XML Веб-сервисы – это основанная на языке XML платформо-независимая технология, поддерживающая разработку распределенных приложений.
- XML Веб-сервисы обеспечивает создание независимых, масштабируемых, слабосвязанных приложений.
- Они основаны на протоколах HTTP, XML, XSD, SOAP, WSDL.
- Реализации SOAP/ WSDL:
  - The Java API for XML Web Services (JAX-WS)
  - Apache CXF
  - Microsoft's Windows Communication Foundation (WCF)
  - …

Серверная часть Web служб

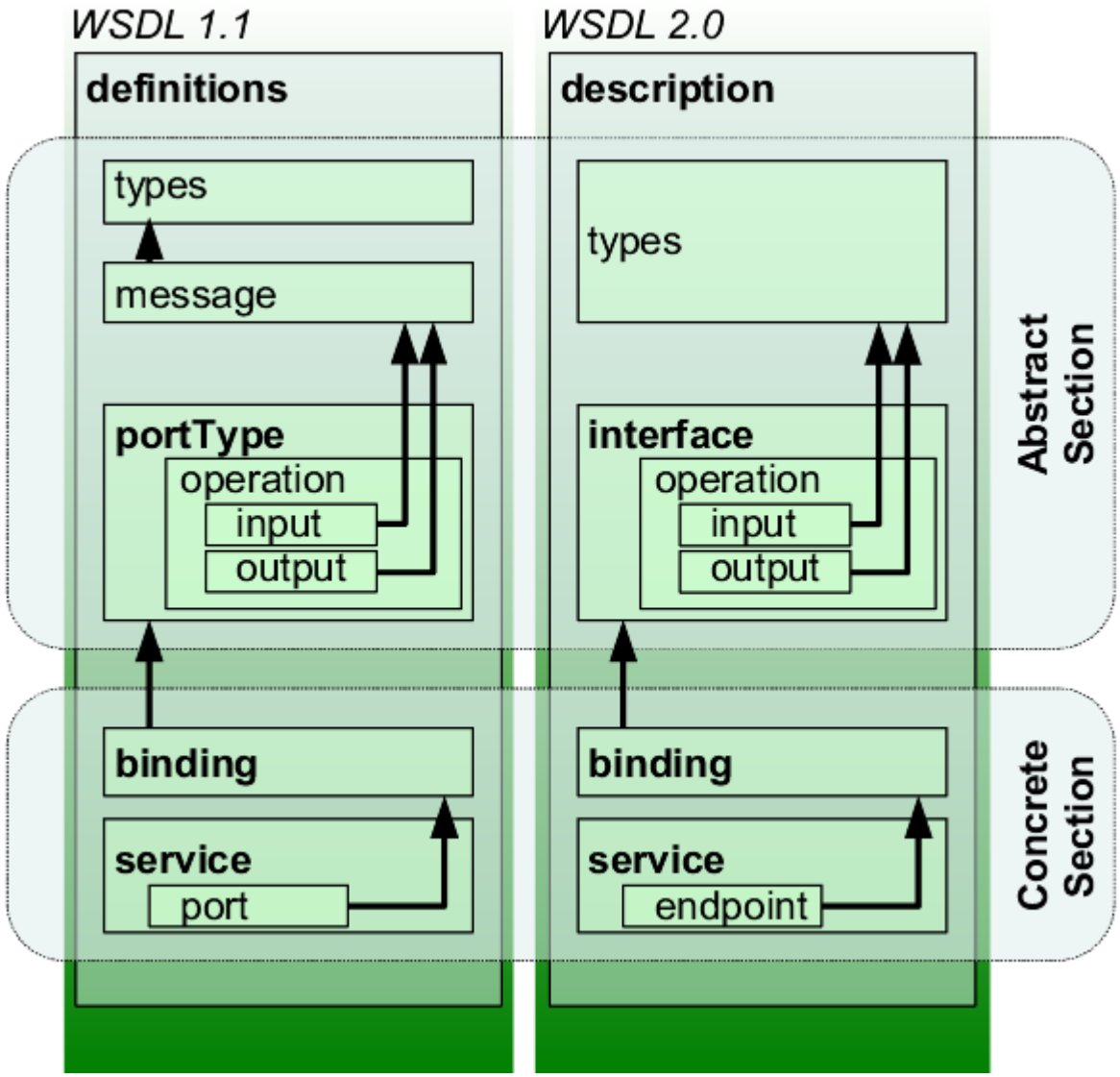


Borja Sotomayor . The Globus Toolkit 4 Programmer's Tutorial. University of Chicago Department of Computer Science

WSDL: Web Service Definition Language

- WSDL – это стандартный XML-документ, описывающий фундаментальные свойств Web службы, как то:
  - **Что это** – описание методов, предоставляемые Web службой;
  - **Как осуществляется доступ** – формат данных и протоколы;
  - **Где он расположен** – сетевой адрес службы (URI).

WSDL 1.1 VS WSDL 2.0



Пример Web службы

- Простой пример Web службы (Java)

```
@WebService
public class MyMath
     {
          @WebMethod
          public int squared(int x)
          {
               return x * x;
          }
     }
```

WSDL-документ

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://DefaultNamespace"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://DefaultNamespace"
  xmlns:intf="http://DefaultNamespace"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <wsdl:message name="squaredRequest">
       <wsdl:part name="in0" type="xsd:int"/>
    </wsdl:message>
    <wsdl:message name="squaredResponse">
       <wsdl:part name="squaredReturn" type="xsd:int"/>
    </wsdl:message>
    <wsdl:portType name="MyMath">
       <wsdl:operation name="squared" parameterOrder="in0">
          <wsdl:input message="impl:squaredRequest" name="squaredRequest"/>
          <wsdl:output message="impl:squaredResponse" name="squaredResponse"/>
       </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="MyMathSoapBinding" type="impl:MyMath">
       <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/ soap/http"/>
       <wsdl:operation name="squared">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="squaredRequest">
             <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/ encoding/"
                namespace="http://DefaultNamespace" use="encoded"/>
          </wsdl:input>
          <wsdl:output name="squaredResponse">
             <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/ encoding/"
                namespace="http://DefaultNamespace" use="encoded"/>
          </wsdl:output>
       </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="MyMathService">
       <wsdl:port binding="impl:MyMathSoapBinding" name="MyMath">
          <wsdlsoap:address location="http://localhost:8080/axis/testaccount/MyMath"/>
       </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Namespaces

messages - сообщения

portTypes - методы

binding - связи

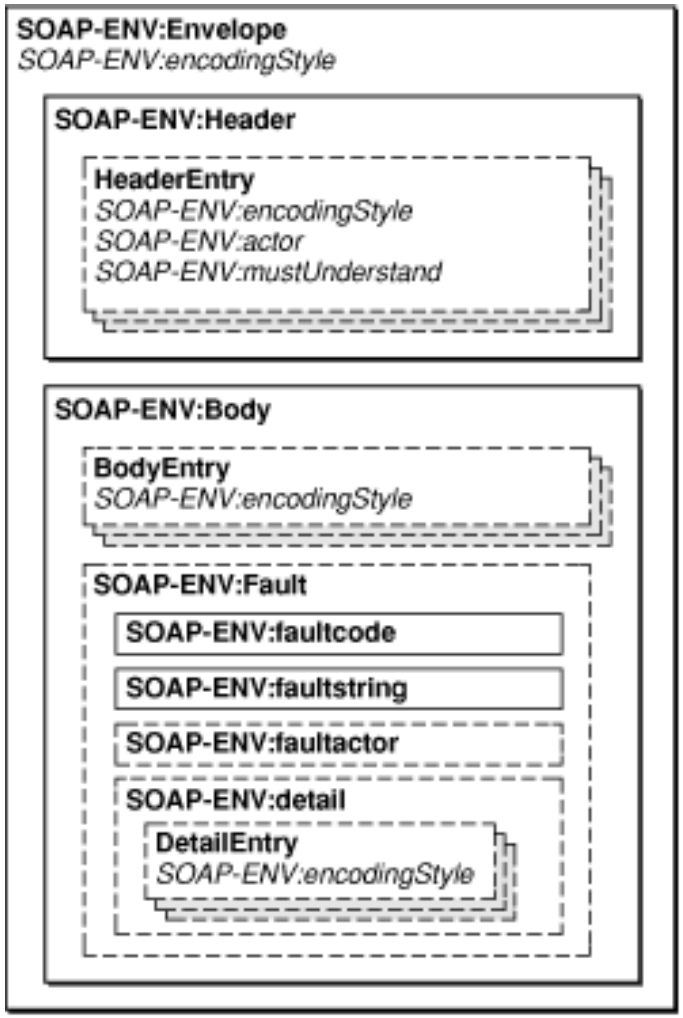
Определение службы

SOAP (уже не только Simple Object Access Protocol)

- SOAP – это протокол, основанный на обмене XML-документами.
- SOAP определяется следующим образом: «SOAP это основанный на XML протокол обмена информацией в децентрализованной распределенной среде.

Элементы сообщения SOAP

- Envelope (конверт) – корневой элемент сообщения.
- Header (заголовок) – не обязательный элемент сообщения. Может содержать дополнительную информацию для приложения, обрабатывающего запрос.
- Body (Тело) – обязательный элемент сообщения. Содержит вызовы необходимых методов и передаваемые параметры.



Шаблон сообщения SOAP

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap=http://www.w3.org/2001/12/soap-envelope
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
  ...
  ...
</soap:Header>
<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

W3C schools website (http://www.w3schools.com)

Пример заголовка SOAP

- В заголовке мы можем ввести новый элемент, не предусмотренный стандартом SOAP. Например, номер транзакции.
- Атрибуты:
  - mustUnderstand – получатель обязан обрабатывать этот элемент;
  - actor – указывает конкретное приложение-получатель при обработке сообщения в цепочке

```
<soap:Header>

   <trans:Transaction
   xmlns:trans="http: //www.host.com/
   namespaces/space/"
   soap:mustUnderstand="1">

   12

   </trans:Transaction>

</soap:Header>
```

Тело сообщения SOAP

Запрос

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetails xmlns="http://warehouse.example.com/ws">
       <productID>12345</productID>
     </getProductDetails>
   </soap:Body>
 </soap:Envelope>
```

Ответ

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
       <getProductDetailsResult>
         <productID>12345</productID>
         <productName>Стакан граненый</productName>
         <description>Стакан граненый. 200 мл.</description>
         <price>9.95</price>
         <inStock>true</inStock>
       </getProductDetailsResult>
     </getProductDetailsResponse>
   </soap:Body>
 </soap:Envelope>
```

Связывание SOAP и HTTP

- Передача SOAP сообщений происходит поверх HTTP – протокола посредством запроса POST (начиная со стандарта SOAP 1.2 возможно применение GET)
- Стандартный протокол связи:
  - Клиент посылает запрос
  - Сервер отвечает OK

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap
+xml;
charset=utf-8
Content-Length: nnn
…

HTTP/1.1 200 OK
Content-Type: application/
soap; charset=utf-8
Content-Length: nnn
…
```

Советы по проектированию RPC сервисов

RPC: «Плоский API»

- «Плоский API»: создание интерфейсов, которые выглядят как методы классов

```
@WebMethod(operationName = "ReserveRentalCar")
public RentalOptions ReserveRentalCar (
  @WebParam( name = "RentalCity") String RentalCity,
  @WebParam( name = "PickupMonth") int PickupMonth,
  @WebParam( name = "PickupDay") int PickupDay,
  @WebParam( name = "PickupYear") int PickupYear,
  @WebParam( name = "ReturnMonth") int ReturnMonth,
  @WebParam( name = "ReturnDay") int ReturnDay,
  @WebParam( name = "ReturnYear") int ReturnYear,
  @WebParam (name = "RentalType") String RentalType )
{ // implementation would appear here }
```

- Такой API очень не гибкий, и очень хрупкий, т.к. в нем невозможно даже изменить последовательность параметров без необходимости замены кода во всех клиенах.
- Возможное решение: использование одного аргумента-сообщения.

Daigneau, Robert (2011-10-25). Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Pearson Education. Kindle Edition.

RPC: Посредники

- Не смотря на то, что можно формировать RPC-сообщения вручную, на клиентской стороне чаще всего используются посредники (proxy), которые инкапсулируют процесс сетевого взаимодействия. В этом случае, с клиентской точки зрения, процесс вызова удаленного метода не отличается от вызова локального метода.
- Для генерации посредников используются специальные программные генераторы кода, которые на основе описания *клиентского интерфейса* (например, WSDL-документа) создают код вызова удаленной процедуры.
- При изменении интерфейса сервиса, все клиенты должны перегенерировать свои proxy-методы

Daigneau, Robert (2011-10-25). Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Pearson Education. Kindle Edition.

RPC: Адресация

- Для обеспечения прозрачной адресации, необходимо чтобы клиенты не имели информации о конкретном расположении сервиса.
- В этом случае его можно перенести или реплицировать (при необходимости)
- Для реализации такой возможности необходимо использовать промежуточный «Соединитель сервисов» (Service Connector), который обеспечивает управление соединениями клиентов с сервисом, перенаправляя запросы от клиентов на реальный адрес используемого сервиса.

Daigneau, Robert (2011-10-25). Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Pearson Education. Kindle Edition.