A4M36AOS – Architektury orientované na služby

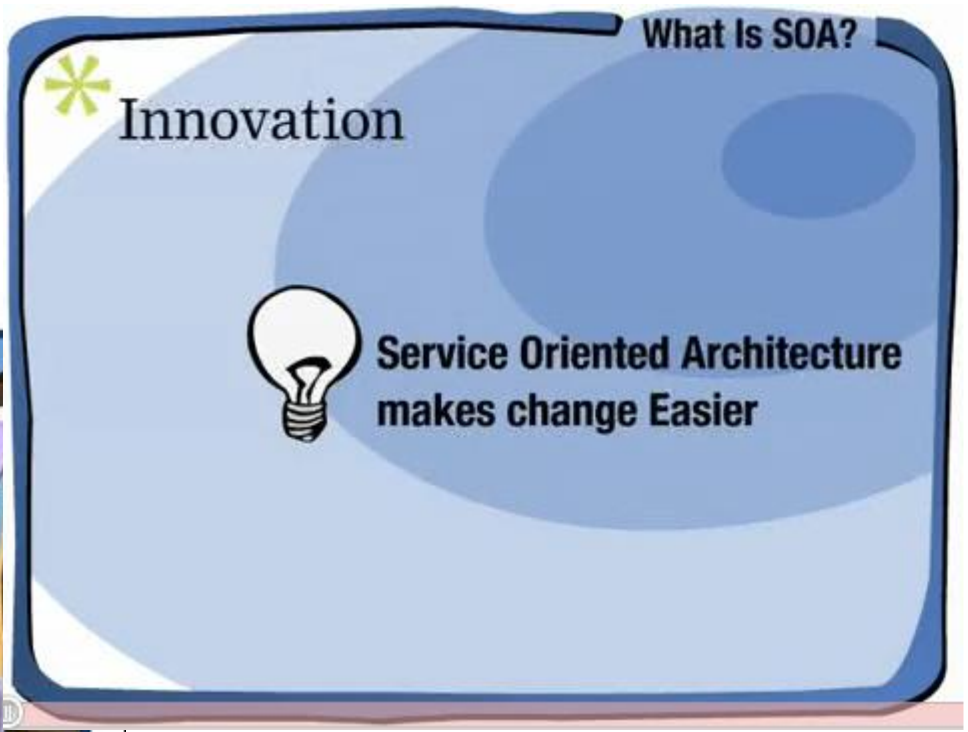
1. Introduction and Concepts

Jiří Vokřínek

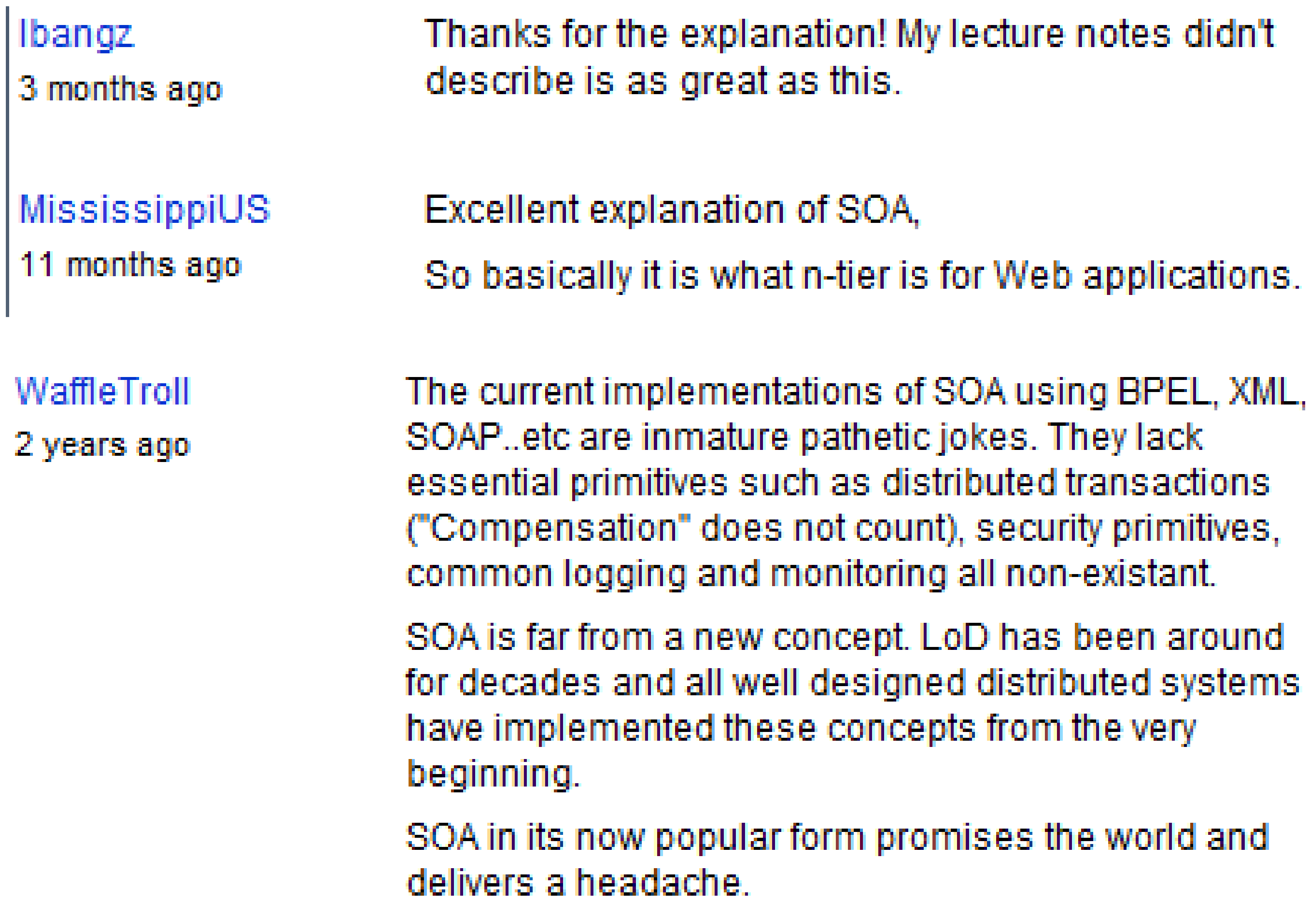
Agent Technology Center
Department of Computer Science and Engineering
Faculty of Electrical Engineering, Czech Technical University in Prague

jiri.vokrinek@fel.cvut.cz http://agents.fel.cvut.cz

What is Service Oriented Architecture SOA?



http://www.youtube.com/watch?v=sbd_1G8Kqjs&feature=channel
http://www.youtube.com/watch?v=dyHWAiG6c-Y&feature=channel



Ibangz
3 months ago

Thanks for the explanation! My lecture notes didn't describe is as great as this.

MississippiUS
11 months ago

Excellent explanation of SOA,

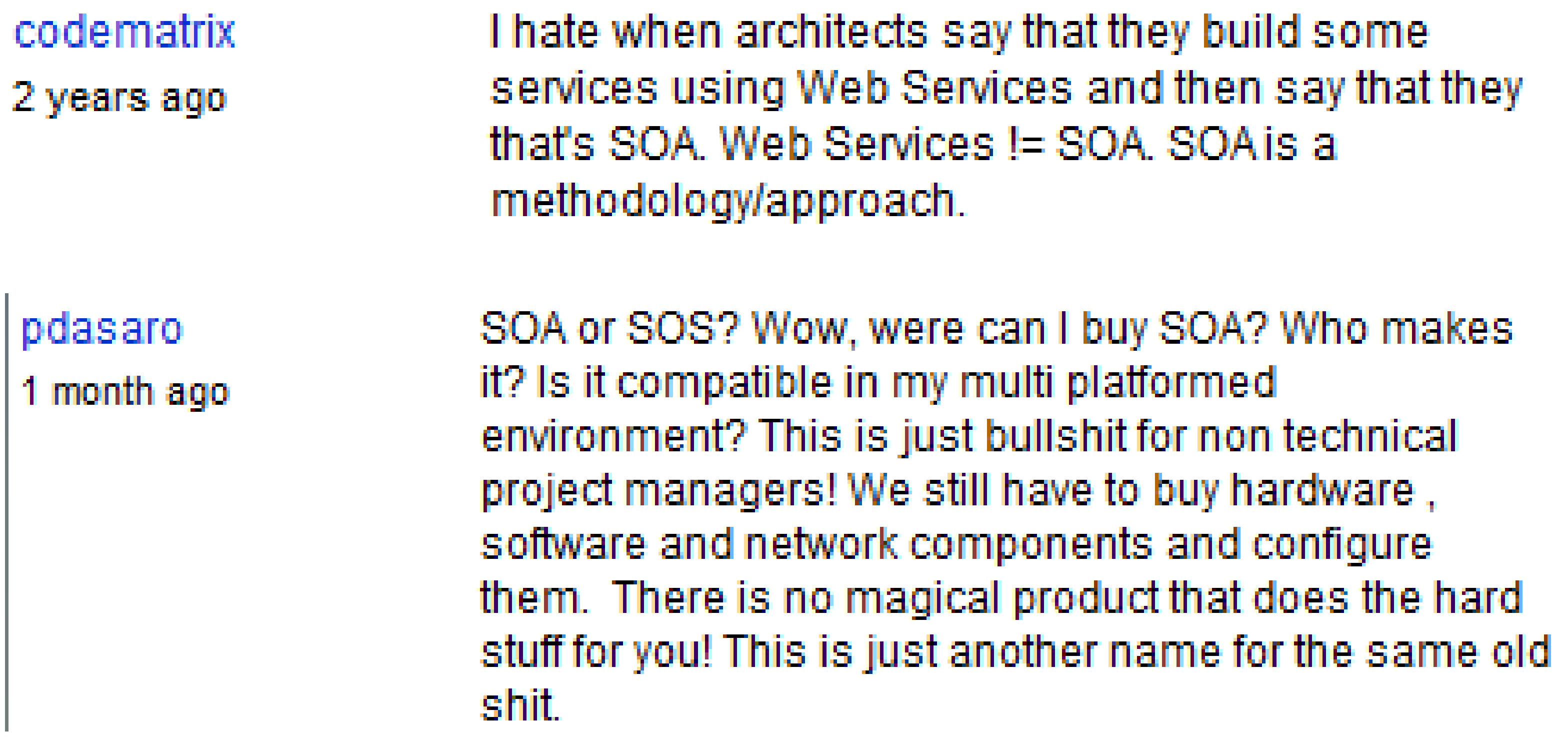
So basically it is what n-tier is for Web applications.

WaffleTroll
2 years ago

The current implementations of SOA using BPEL, XML, SOAP..etc are inmature pathetic jokes. They lack essential primitives such as distributed transactions ("Compensation" does not count), security primitives, common logging and monitoring all non-existant.

SOA is far from a new concept. LoD has been around for decades and all well designed distributed systems have implemented these concepts from the very beginning.

SOA in its now popular form promises the world and delivers a headache.



codematrix
2 years ago

I hate when architects say that they build some services using Web Services and then say that they that's SOA. Web Services != SOA. SOA is a methodology/approach.

pdasaro
1 month ago

SOA or SOS? Wow, were can I buy SOA? Who makes it? Is it compatible in my multi platformed environment? This is just bullshit for non technical project managers! We still have to buy hardware , software and network components and configure them. There is no magical product that does the hard stuff for you! This is just another name for the same old shit.

What is Service-oriented Architecture?

Service-oriented Architecture

- is a flexible **set of design principles** used during the phases of systems development and integration
- provide a **loosely-integrated** suite of services that can be used within multiple business domains
- defines how to integrate widely disparate applications for a world that is **distributed** and uses **multiple** implementation **platforms**

Service-oriented Architecture

- defines the **interface** in terms of protocols and functionality
- requires **loose coupling** of services with operating systems, and other technologies that underlie applications
- separates functions into **distinct** units, or **services**, which developers make accessible over a network

What is that Service?

Services

- services and their corresponding consumers **communicate** with each other by passing data in a **well-defined**, shared **format**, or by **coordinating an activity** between two or more services

- services comprise unassociated, loosely coupled **units of functionality** that have no calls to each other embedded in them

Services

- each service implements **one action**
- instead of services embedding calls to each other in their source code they use **defined protocols** that describe how services pass and parse messages, using **description metadata**
- allow users to **combine and reuse** them in the production of applications

Principles

- service **encapsulation** – many services are consolidated for use under the SOA; often such services were not planned to be under SOA
- service **loose coupling** – services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other
- service **contract** – services adhere to a communications agreement, as defined collectively by one or more service-description documents

Principles

- service **abstraction** – beyond descriptions in the service contract, services hide logic from the outside world
- service **reusability** – logic is divided into services with the intention of promoting reuse
- service **composability** – collections of services can be coordinated and assembled to form composite services
- service **autonomy** – services have control over the logic they encapsulate

Additional Constraints

- stateless service
- stateful service
- idempotent request

Additional Constraints

- stateless service
- stateful service
- idempotent request

Additional Constraints

- **stateless service**

  – each message that a consumer sends to a provider must **contain all necessary information** for the provider to process it. This is effectively "*service in mass production*" since each request can be treated as generic. There are **no intermediate states** to worry about, so recovery from partial failure is also relatively easy. This makes a service **more reliable**.

Additional Constraints

- stateless service
- stateful service
- idempotent request

Additional Constraints

- **stateful service**

  – holds a **session** between a consumer and a provider. Stateful services require both the consumer and the provider to **share the** same consumer-specific **context**. The drawback of this constraint is that it may **reduce the** overall **scalability** of the service provider because it may need to remember the shared context for each consumer.

Additional Constraints

- stateless service
- stateful service
- **idempotent request**

Additional Constraints

- **idempotent request**

  – **duplicate requests** received by a service have the **same effects** as a unique request. This constraint allows providers and consumers to improve the overall service **reliability by** simply **repeating** the request if **faults** are encountered.

RPC

CORBA

UDDI

WSDL

SOA Technologies

REST

DCOM

SOAP

JAX-RPC

JAX-WS

1976

- Apple Computer Company is formed by Steve Jobs and Steve Wozniak
- Peugeot becomes a part of Citroen
- The first 4.6 miles of the Washington Metro subway system opens
- The first laser printer is introduced by IBM
- **Remote Procedure Call** for ARPANET is introduced

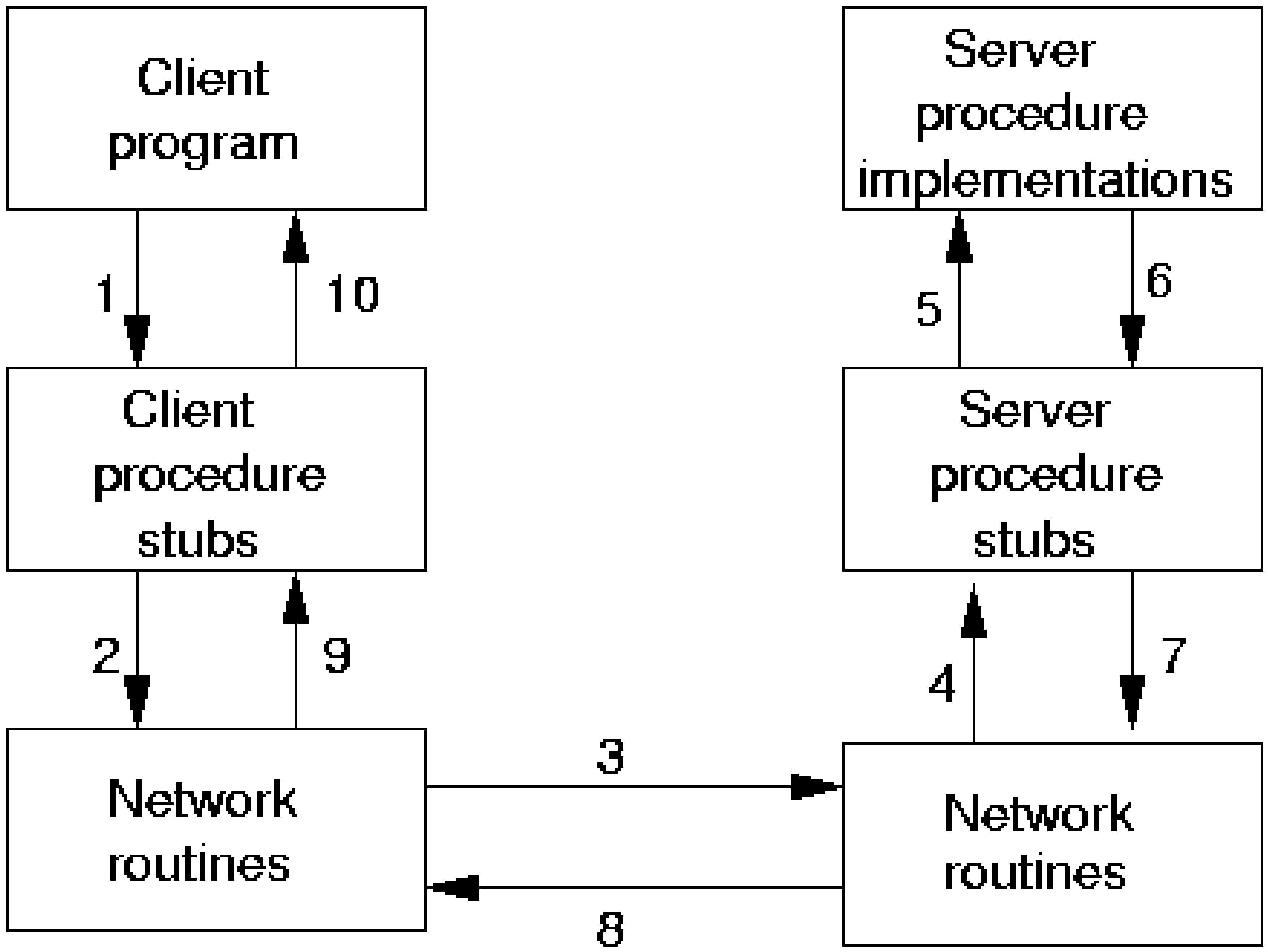
Remote Procedure Call (RPC)

- First business use by XEROX in 1981
- Popular UNIX implementation by Sun
- Base for later DCOM (Microsoft) and CORBA
- **Inter-process communication** allowing a program to execute subroutine in another address space **without** the need of **explicitly coding** this remote interaction
- Programmer writes essentially the **same code** whether the subroutine is local or remote

Remote Procedure Call (RPC)

- In object-oriented scope, RPC is called remote invocation or **remote method invocation**
- RPC is initiated by client that is **blocked** until server responses
- **Can fail** because of unpredictable network conditions
- Idempotent procedures easily handled

RPC Sequence



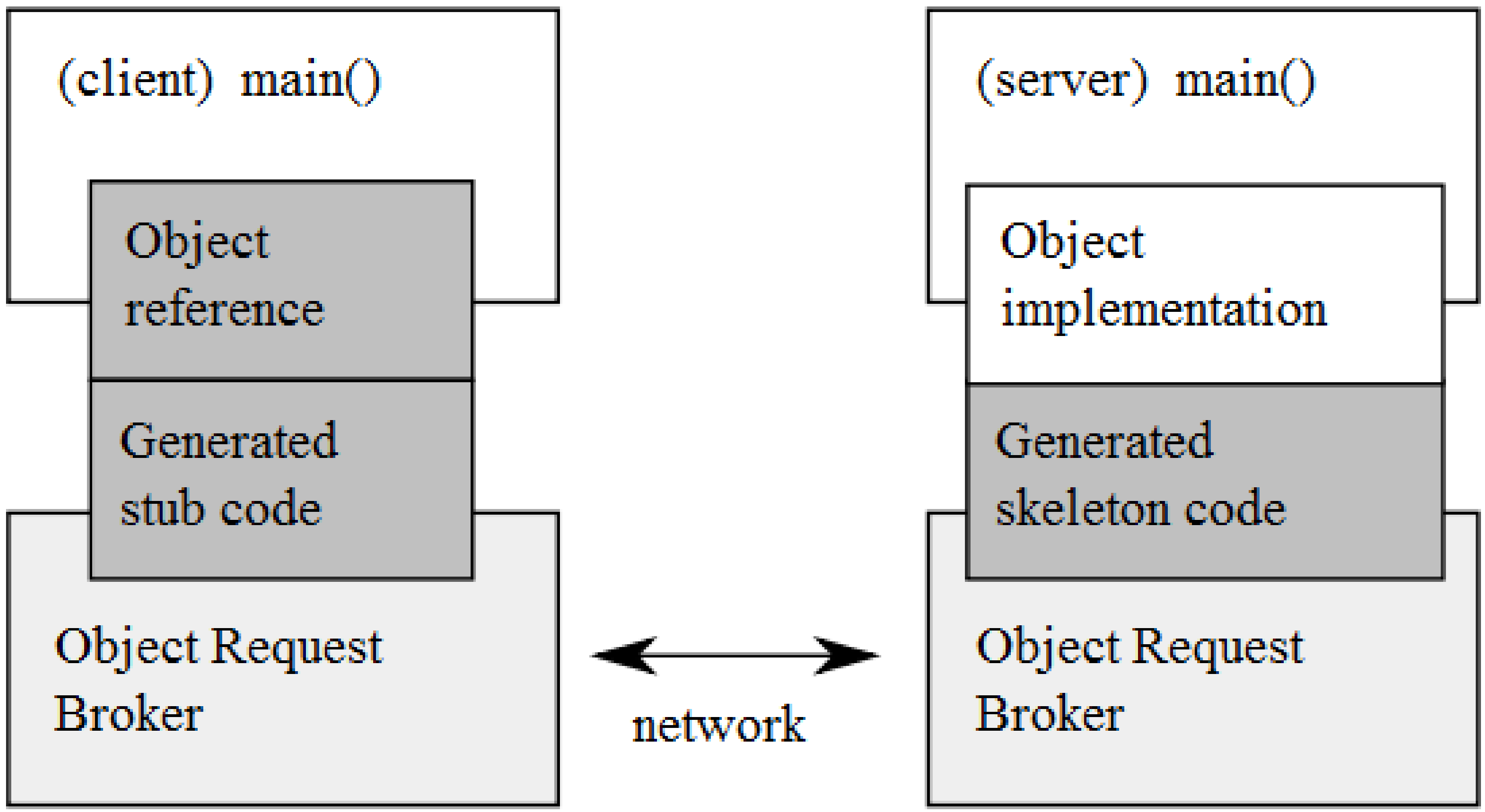
1991

- Soviet Union collapses and is formally dissolved
- Sweden wins the 36th Eurovision Song Contest
- First mobile operator starts (EuroTel)
- AMD starts to compete with Intel by Am386
- Internet connects more than 100 computers
- **Common Object Request Broker Architecture** is introduced

Common Object Request Broker Architecture (CORBA)

- Standard that enables software components written in **multiple computer languages** running on **multiple computers** (i.e. platforms) to work together
- Specification include: data typing, exceptions, network protocol, communication timeouts, etc.
- Does not address: object lifetimes, redundancy, naming/models semantics, memory management, load balancing, etc.

CORBA Call Structure



CORBA Features

- Language independence
- OS independence
- Freedom from technologies
- Strong data typing
- CORBA Interface Definition Language

CORBA Features

- Freedom from data transfer details
- Compression of marshaled binary data
- Transactions and security
- … etc

1996

- NASA launches the Mars Global Surveyor
- AMD's first in-house x86 processor K5
- CESNET is founded
- **HTTP** v1.0 is introduced
- Working Draft of an **XML** specification is published
- **Distributed Component Object Model** introduced in Windows NT

Hypertext Transfer Protocol (HTTP)

- Networking protocol for distributed, collaborative, **hypermedia information** systems
- Application layer **request-response protocol** in the **client-server** computing model
- Foundation of data communication for the World Wide Web

HTTP Request Methods

- HEAD – similar to GET but without response body
- GET – requests a representation of the specified resource
- POST – submits data to be processed to the identified resource
- PUT – uploads a representation of the specified resource
- DELETE – deletes the specified resource

HTTP Request Methods

- TRACE – echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers
- OPTIONS – returns the HTTP methods that the server supports for specified URL
- CONNECT – converts the request connection to a transparent TCP/IP tunnel
- PATCH – used to apply partial modifications to a resource

Extensible Markup Language (XML)

- Set of rules for **encoding documents** in machine-readable form
- Design goals emphasize simplicity, generality, and usability over the Internet
- **Textual data format** with focus on documents
- Widely used for **interface data definitions**

Extensible Markup Language (XML)

- Uses tags, elements, attributes
- **Defined by schemas** – Document Type Definition (DTD) or XML Schema Definition (XSD)

```
<?xml version="1.0" encoding="UTF-8" ?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
    <date>1511</date>–<date>1512</date>.
  </caption>
</painting>
```

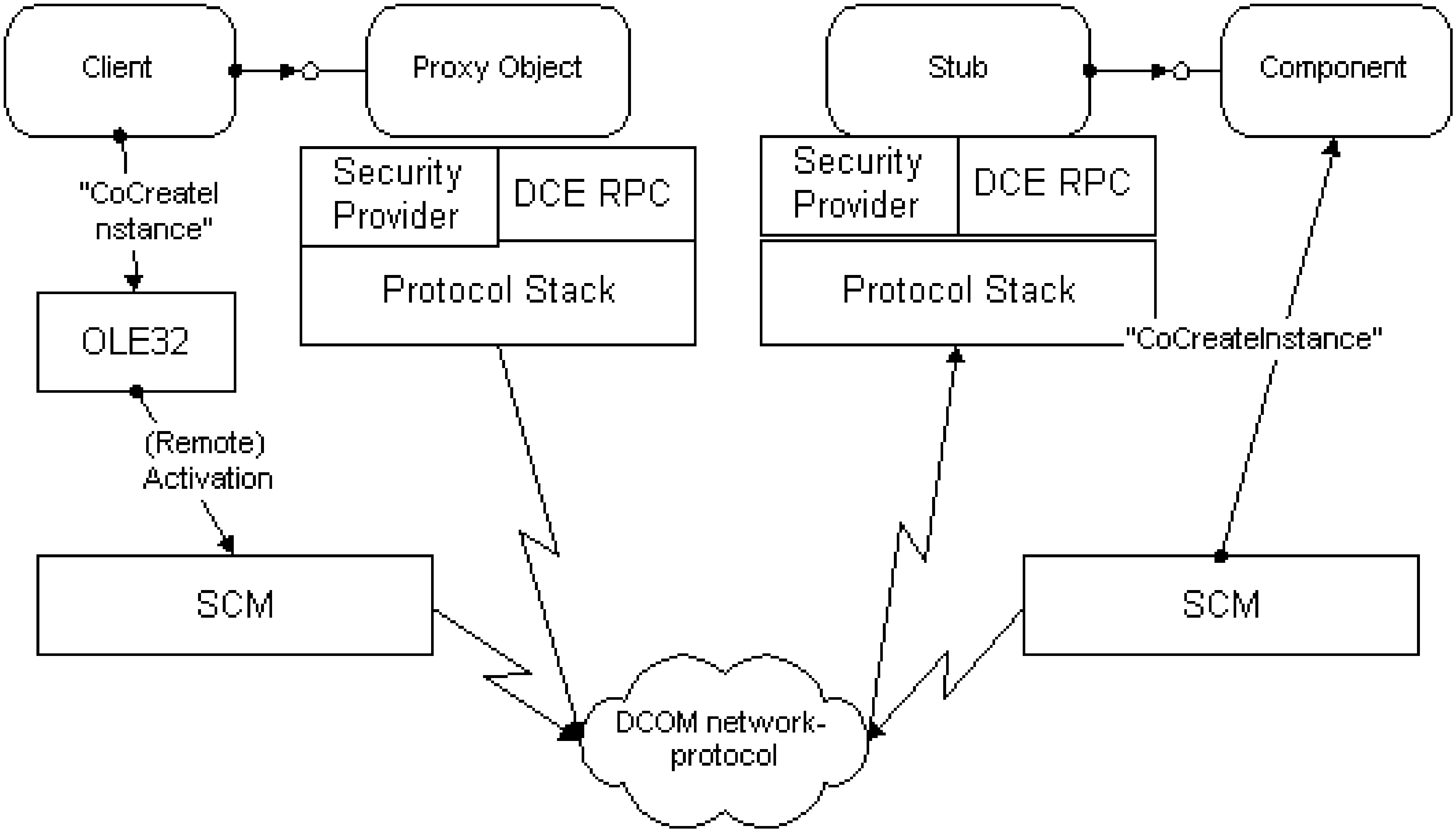
Distributed Component Object Model (DCOM)

- **Proprietary** Microsoft technology for communication among **software components distributed** across networked computers
- Has been **deprecated** in favor of the Microsoft .NET Framework
- COM based implementation of RPC
- Competitor to CORBA

DCOM Added Features

- **Marshalling** – serializing and deserializing the arguments and return values of method calls
- Distributed **garbage collection** – ensuring that references held by clients of interfaces are released when, for example, the client process crashed, or the network connection was lost

DCOM Call Structure



2000

- Nuclear power plant Temelin starts operation
- Russian submarine K-141 Kursk sinks in the Barents Sea
- IIHFWC final: Czech Republic beats Slovakia 5:3
- ATI launches Radeon series
- **UDDI** standard is defined
- **WSDL** is introduced
- **REST** is introduced and defined

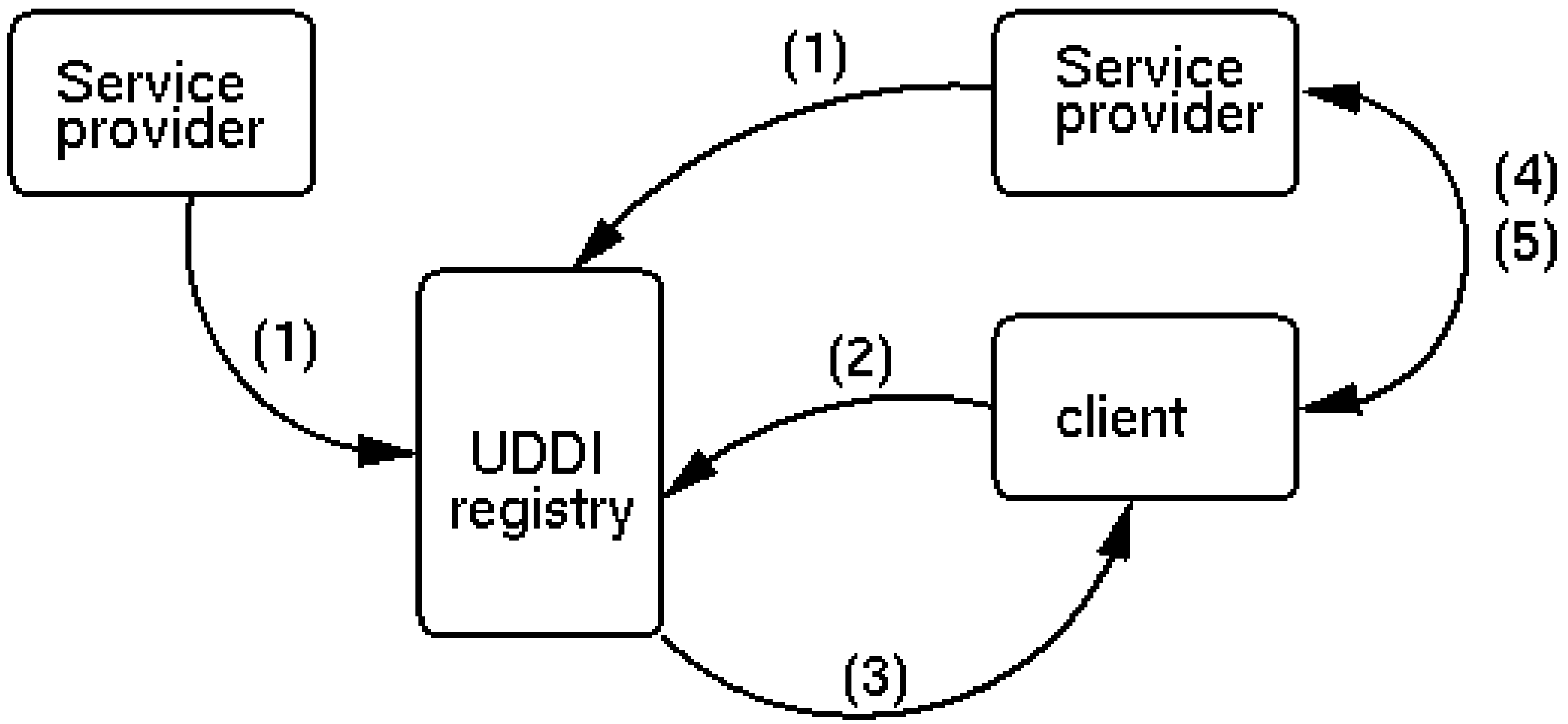
Universal Description, Discovery and Integration (UDDI)

- **Platform-independent** XML-based **registry** for businesses worldwide to list themselves on the Internet
- Open industry initiative
- Enables businesses to **publish service** listings and discover each other and define how the services or software applications interact over the Internet

UDDI Registry

- **White Pages** — address, contact, and known identifiers
- **Yellow Pages** — industrial categorizations based on standard taxonomies
- **Green Pages** — technical information about services exposed by the business

UDDI Search


Service provider
(1)
(1)
Service provider
(4)
(5)
UDDI registry
(2)
client
(3)

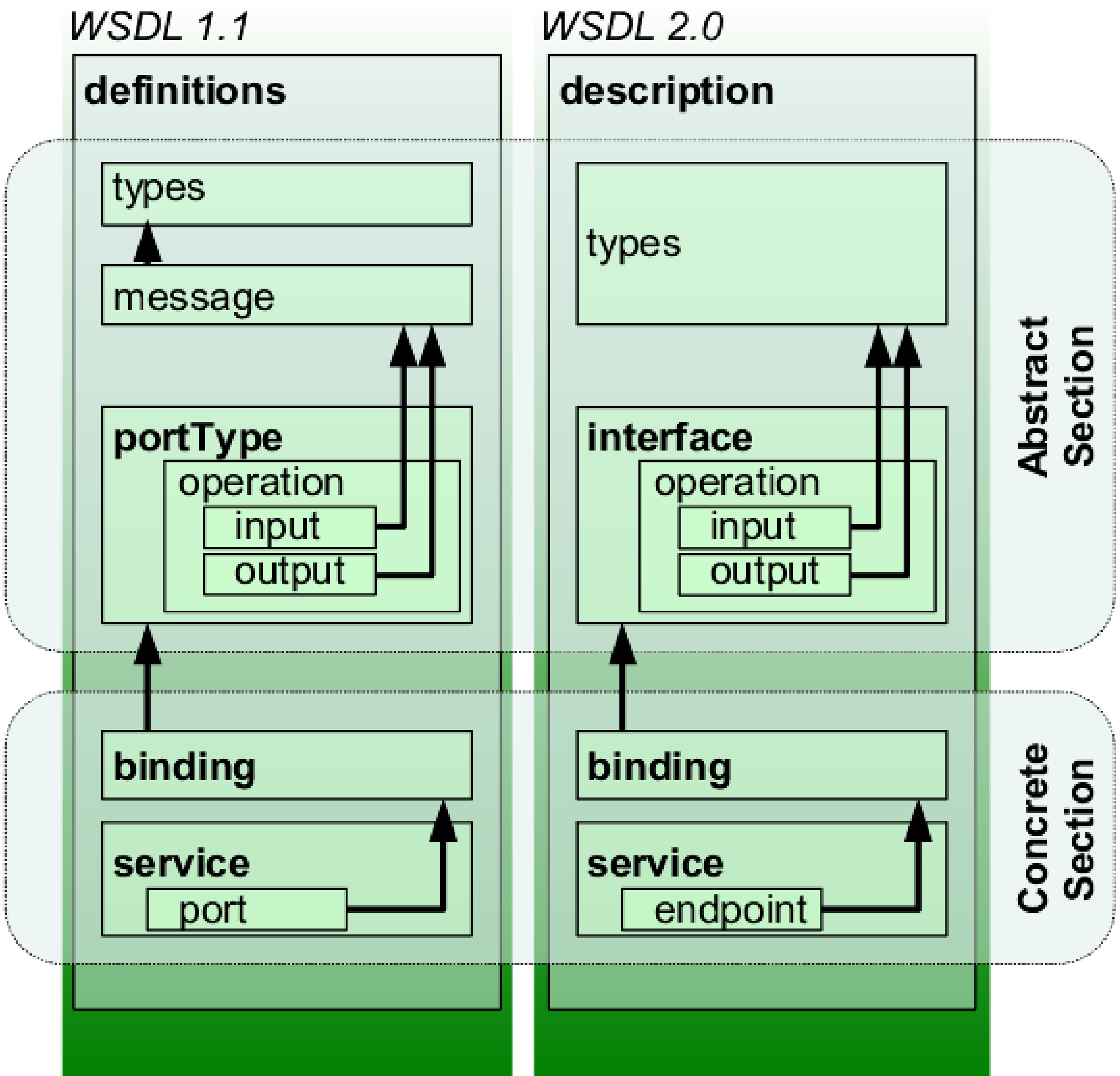
Web Services Description Language (WSDL)

- XML-based language that provides a model for describing Web services
- Defines services as collections of network endpoints, or ports associating a network address with a reusable bindings
- Describes the public interface to the web service
- Used in combination with SOAP and XML Schema to provide web services over the Internet

WSDL Versions

- WSDL 1.0 (2000), 1.1 (2001) – developed by IBM, Microsoft, and Ariba
- WSDL 1.2 (2003) – W3C draft; easier and more flexible for developers; attempts to remove non-interoperable features
- WSDL 2.0 (2007) – W3C recommendation

WSDL Representation



Representational State Transfer (REST)

- Style of software architecture for distributed hypermedia systems such as the World Wide Web
- REST was by Roy Fielding, principal author of the HTTP 1.0 and 1.1
- Conforming to the REST constraints is referred to as being ‘RESTful’

REST Architecture

- Clients and servers
- Clients initiate requests
- Servers process requests and return appropriate responses
- At any particular time, a client can either be in transition between application states or "at rest" (is able to interact with its user, but creates no load and consumes no per-client storage on the set of servers or on the network)

REST Architecture

- Initially described in the context of HTTP, but is not limited to
- RESTful architectures can be based on other Application Layer protocols
- Utilize preexisting defined interface and other built-in capabilities provided by the network protocol (minimize the addition of new application-specific features on top of it)

REST Constraints

- Client–server – uniform interface, separation of concepts
- Stateless – no client context being stored
- Cacheable – clients are able to cache responses
- Layered system – transparent connection
- Code on demand (optional)
- Uniform interface

RESTful web services

- Simple web service implemented using HTTP
- URI for the web service looks like `http://example.com/resources/`
- MIME type of the data supported by the web service (e.g. XML document)
- Operations supported by the web service using HTTP methods (e.g., POST, GET, PUT or DELETE).

2003

- A total solar eclipse is seen over Antarctica
- Space Shuttle Columbia is launched on what turns out to be its last flight
- The Human Genome Project is completed with 99% of the human genome
- Wikipedia is being distributed
- **Simple Object Access Protocol** became a W3C recommendation
- Java **API for XML-based RPC** is standardized

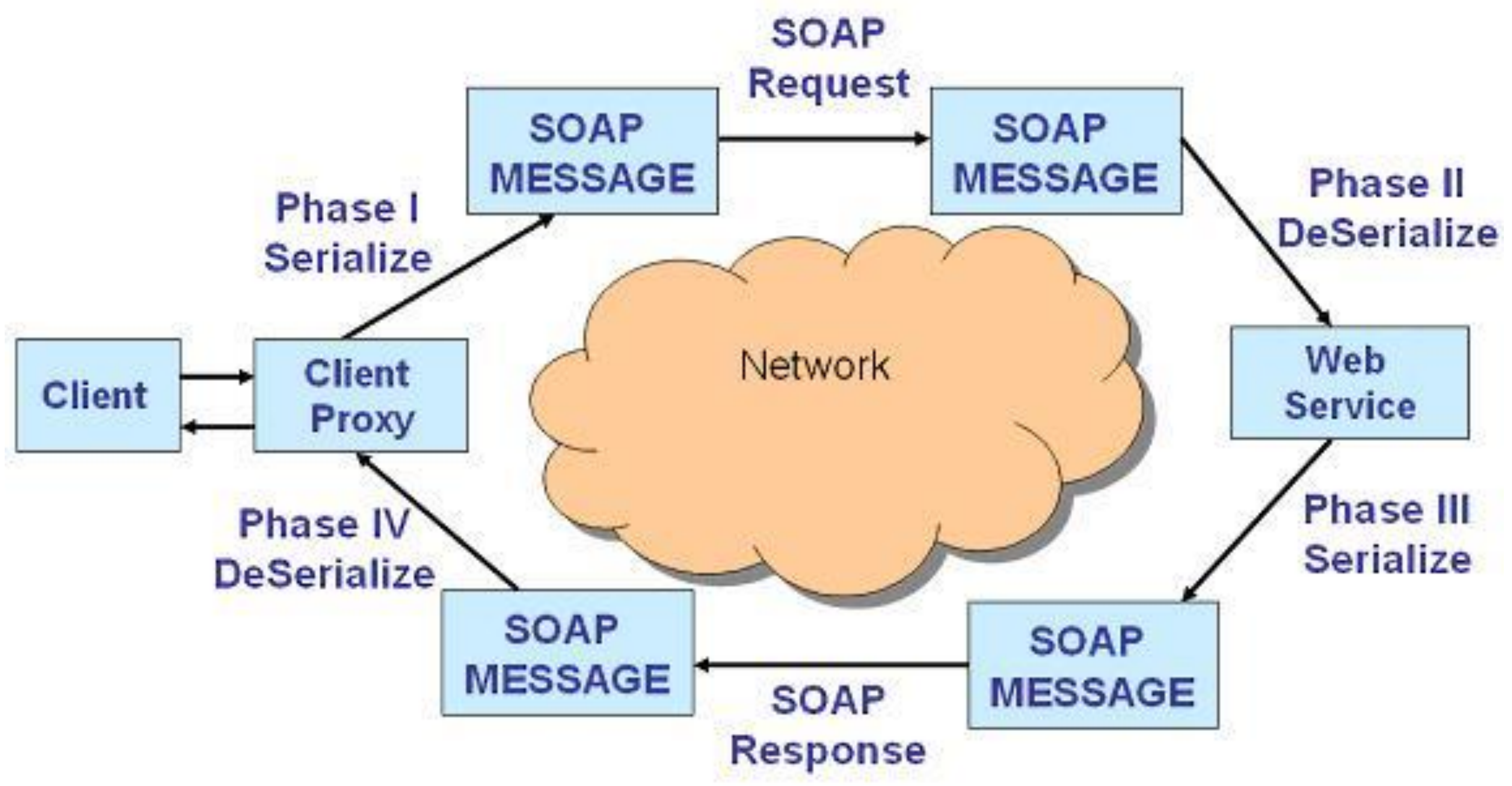
Simple Object Access Protocol (SOAP)

- Protocol specification for exchanging structured information for WS
- Relies on XML for its message format
- Uses other application layer protocols (RPC, HTTP)
- Originally designed as an object-access protocol
- Underlying layer for Web Services, based on WSDL and UDDI

SOAP Specification

- SOAP processing model – defines the rules for processing a SOAP message
- SOAP extensibility model – defines the concepts of SOAP features and SOAP modules
- SOAP underlying protocol binding – framework describing the rules for defining a binding to a protocol exchanging SOAP messages between nodes
- SOAP message construct – defines the structure of a SOAP message

SOAP Call Structure



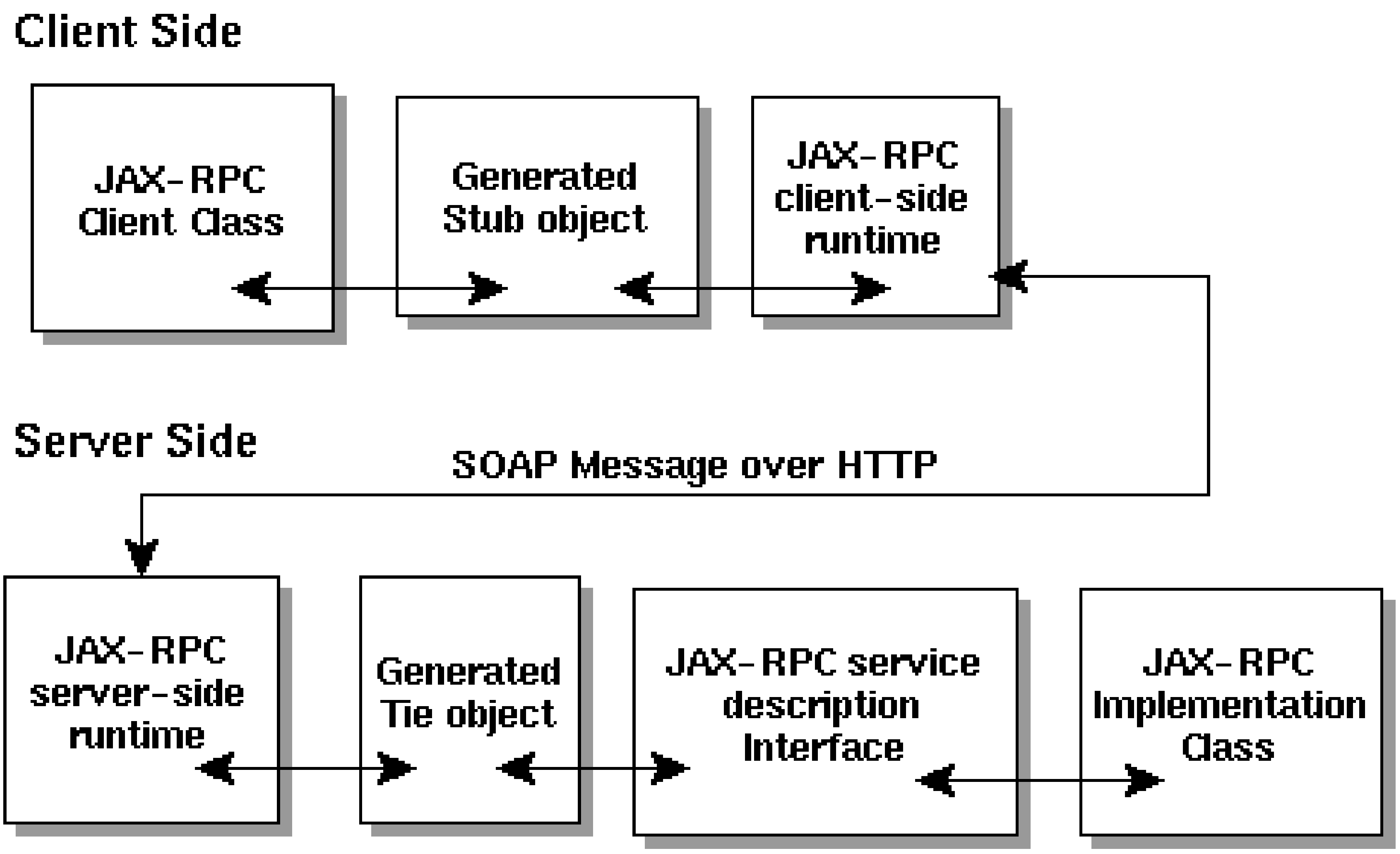
SOAP Message Example

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

<?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:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Java API for XML-based RPC

- JAX-RPC
- Allows Java application to invoke a Java-based Web Service
- Can be seen as Java RMI over Web Service
- Java program invokes methods on stub
- Runtime system converts calls to SOAP and uses HTTP requests

JAX-RPC Call Structure


Client Side
JAX-RPC Client Class
Generated Stub object
JAX-RPC client-side runtime
Server Side
SOAP Message over HTTP
JAX-RPC server-side runtime
Generated Tie object
JAX-RPC service description Interface
JAX-RPC Implementation Class

Java API for XML Web Services

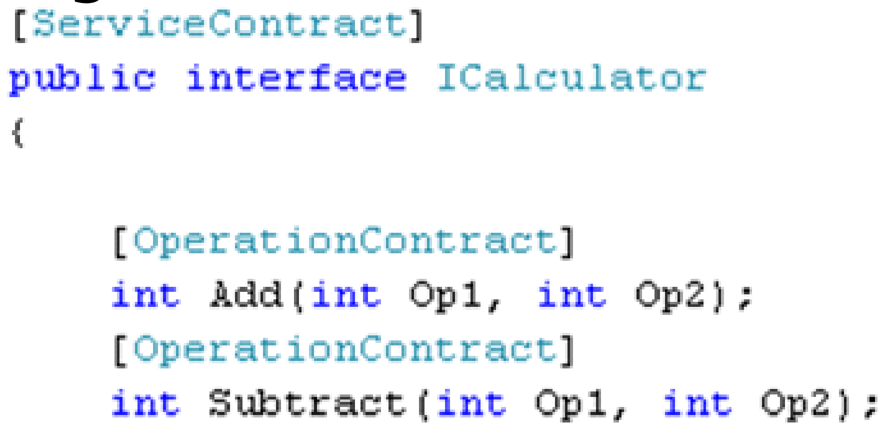
- JAX-WS (2006)
- Java API for creating web services
- Part of the Java EE platform
- Uses annotations to simplify the development and deployment of web service clients and endpoints
- Interface/data mapping model

2007

- MESSENGER spacecraft makes its second fly-by of Venus
- Czech Republic joins the Schengen border-free zone
- Windows Vista fails to substitute XP
- Desktop HDD meets 1TB capacity
- **Windows Communication Foundation is released within** .NET Framework 3.5

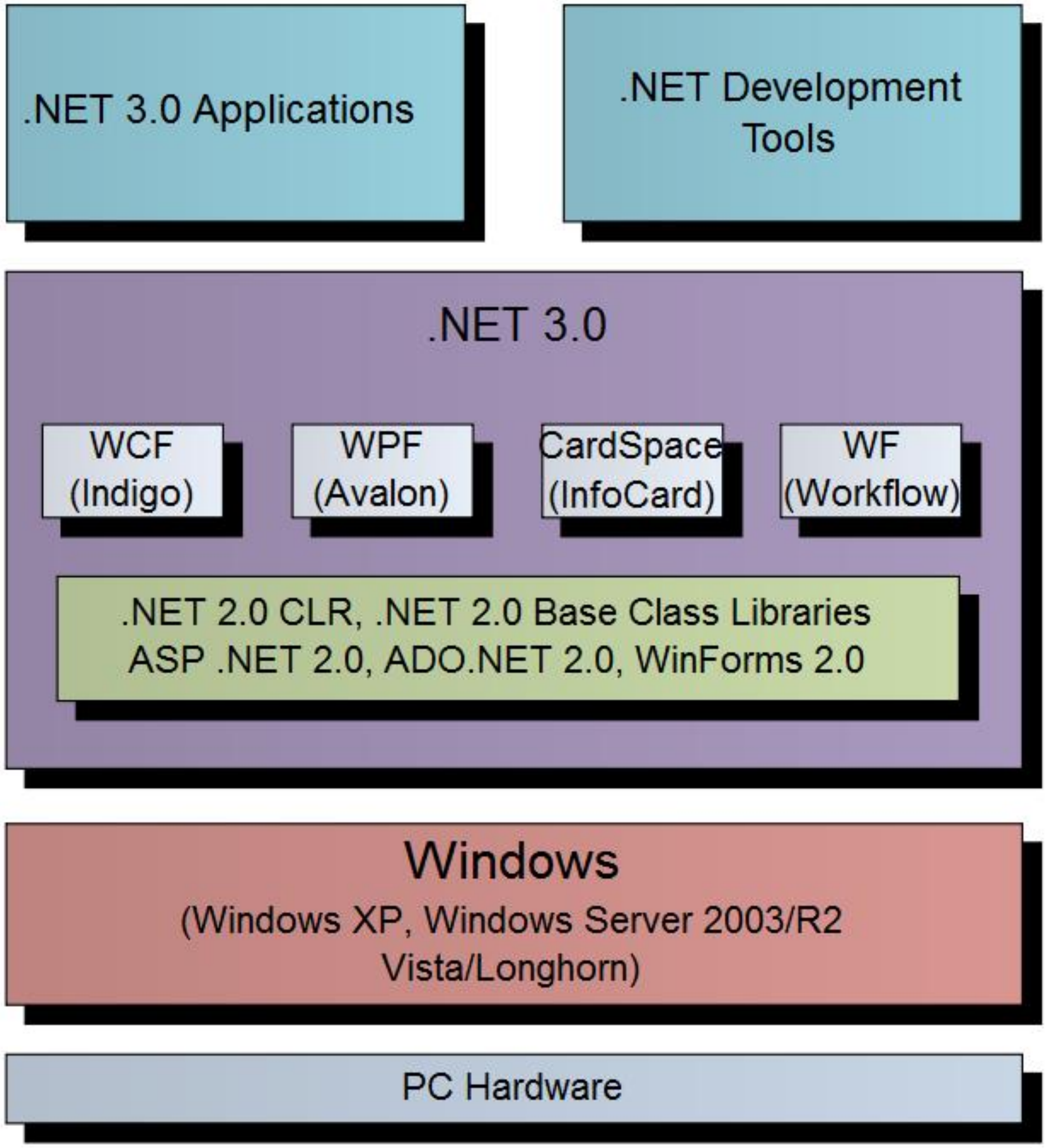
Windows Communication Foundation (WCF)

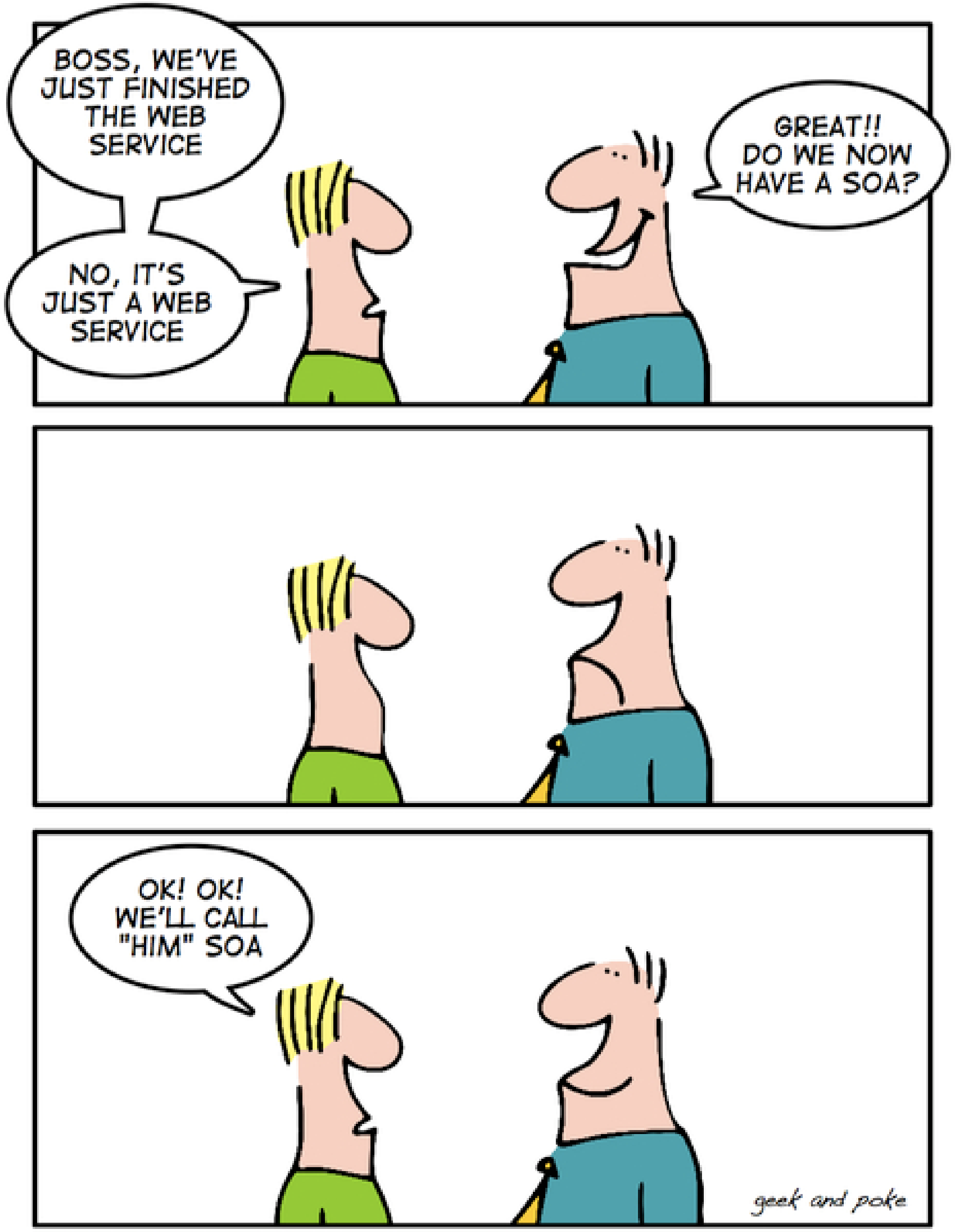
- API in the .NET Framework for building connected, service-oriented applications
- WS support – WSDL, SOAP, XML based
- Wide support for protocols bindings
- RESTful, Queued Messaging
- … etc.



```
[ServiceContract]
public interface ICalculator
{

    [OperationContract]
    int Add(int Op1, int Op2);
    [OperationContract]
    int Subtract(int Op1, int Op2);
```

WCF Structure




BOSS, WE'VE JUST FINISHED THE WEB SERVICE
GREAT!! DO WE NOW HAVE A SOA?
NO, IT'S JUST A WEB SERVICE
OK! OK! WE'LL CALL "HIM" SOA
geek and poke

HOW TO GET A SOA