UNIVERSITY OF JYVÄSKYLÄ

Lecture 1: SOA in a nutshell

TIES4560 SOA and Cloud Computing
Autumn 2020

University of Jyväskylä

Khriyenko Oleksiy

Cloud Computing

Cloud Computing *is a buzzword that means different things to different people:*

- *another way of describing IT (information technology) "outsourcing"*
- *any computing service provided over the Internet or a similar network*
- *any bought-in computer service you use that sits outside your firewall*
- *etc.*

Some definitions of Cloud Computing…

- *In Computer science, cloud computing describes a type of **outsourcing of computer services**, similar to the way in which electricity supply is outsourced. Users can simply use it. They do not need to worry where the electricity is from, how it is made, or transported. Every month, they pay for what they consumed. The idea behind cloud computing is similar: The user can simply use storage, computing power, or specially crafted development environments, without having to worry how these work internally. (Wikipedia)*
- *It is a style of computing in which **IT-related capabilities** are provided "**as a service**", allowing users to access technology-enabled services from the Internet ("in the cloud") without knowledge of, or control over the technologies behind these servers. (Wikipedia)*
- *Cloud Computing is a paradigm in which information is permanently stored in servers on the Internet and cached temporarily on clients that include computers, laptops, handhelds, sensors, etc. (according to a paper published by IEEE Internet Computing in 2008)*
- *…*

Cloud Computing

Why do you want to study cloud computing?

- *To make your boss happy ...*

- *It becomes too expensive to manage hardware yourself - especially with low load.*
  - *Most in house data center usage rates are very low.*
  - *But it becomes cheaper again at some point (if you get the utilization rate up and management costs down)*

- *More and more services are moved to the cloud*
  - *When developing services, you must know cloud computing*

Cloud Computing

Exhibit 1
Nine Challenges in Cloud Computing

1 Efficiency of **Service Provisioning**
- a. Usage of development tools and components
- b. Creation of scalable architectures
- c. Resource management and flexibility
- d. Availability of services

2 Effectiveness of **Service Usage and Control**
- a. Contracts including questions of liability
- b. Control of services by users
- c. Governance/escalation mechanisms

3 **Transparency** of Service Delivery and Billing
- a. Billing including license management
- b. Quality assurance and monitoring SLA
- c. Type and location of data processing

4 Information **Security**
- a. Identity and rights management
- b. Privacy and integrity
- c. Access control, logging, and attack prevention
- d. Verification and certification

5 **Data Privacy**

6 **Interoperability**
- a. Migration into/out of the cloud
- b. Ability to integrate into on-premise IT
- c. Cloud federation

7 **Portability** Between Providers
- a. Service portability
- b. Data portability

8 Ensuring Fair **Competition** in the Market

9 **Compliance** with Regulatory Requirements

Picture is taken from [1]

Cloud Computing

- *At the beginning **Cloud Computing** has started as a business term*
- *Cloud Computing is the use of computing resources (hardware and software) that are delivered as a service over a network. (Wikipedia)*

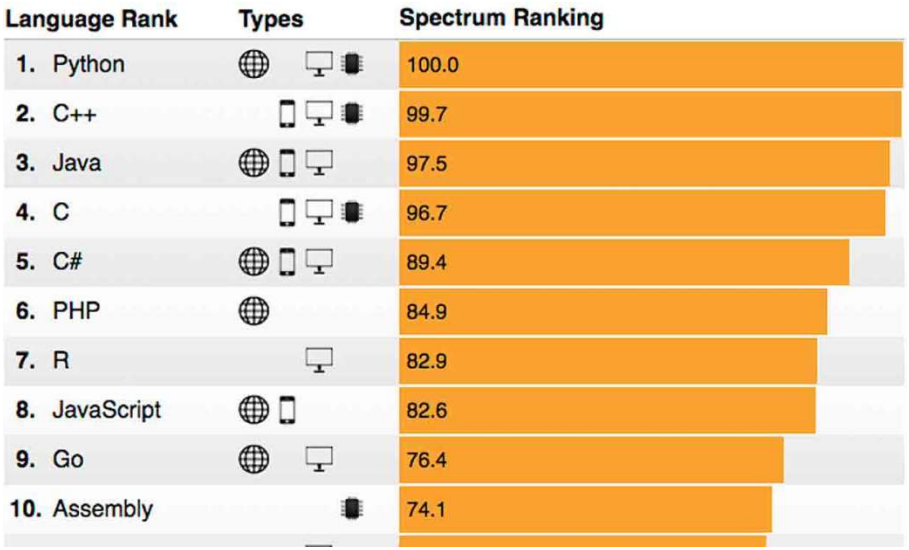
*Thus, **Services** come first!!!*

- *The business aspects have difficult consequences on the implementation side*

… students more interested in *management* and *planning* could consider
TJTSS70 – Cloud Computing
(please, check availability!!!)

Main course targets

- *Learn generalizable techniques and skills*



According to IEEE Spectrum ranking [2]

- *Basics of Services (which are just a part of SOA)*
- *Basics of Cloud Computing*

Course practicalities

- ***Couse consists of 6 lectures***

Lecture	Time	Title	Location
Lecture 1 (08.09.2020)	10:15 - 12:00	*SOA in a nutshell*	*Zoom meeting*
Lecture 2 (15.09.2020)	10:15 - 12:00	*from SOAP towards REST*	*Zoom meeting*
Lecture 3 (22.09.2020)	10:15 - 12:00	*REST Web Service (with Jersey)*	*Zoom meeting*
Lecture 4 (29.09.2020)	10:15 - 12:00	*Security and Access Control*	*Zoom meeting*
Lecture 5 (06.10.2020)	10:15 - 12:00	*Cloud Computing*	*Zoom meeting*
Lecture 6 (13.10.2020)	10:15 - 12:00	*Serverless Architecture*	*Zoom meeting*
(20.10.2020)	*10:15 - 12:00*		*Zoom meeting*

- ***Additionally we have 6 demo***

Demo	Time	Location		+extra	time	Location
Demo 1 (16.09.2020)	10:15 - 12:00	*Zoom meeting*	-	(16.09.2020)	12:15 - 14:00	*Zoom meeting*
Demo 2 (23.09.2020)	10:15 - 12:00	*Zoom meeting*	-	(23.09.2020)	12:15 - 14:00	*Zoom meeting*
Demo 3 (30.09.2020)	10:15 - 12:00	*Zoom meeting*	-	(30.09.2020)	12:15 - 14:00	*Zoom meeting*
Demo 4 (07.10.2020)	10:15 - 12:00	*Zoom meeting*	-	(07.10.2020)	12:15 - 14:00	*Zoom meeting*
Demo 5 (14.10.2020)	10:15 - 12:00	*Zoom meeting*	-	(14.10.2020)	12:15 - 14:00	*Zoom meeting*
Demo 6 (21.10.2020)	10:15 - 12:00	*Zoom meeting*	-	(21.10.2020)	12:15 - 14:00	*Zoom meeting*

Course information webpage: *http://users.jyu.fi/~olkhriye/ties4560*

Course practicalities

- ***Practical tasks:***

Practical tasks are planned to be done in groups of 4-5 people (depending on the total amount of students). If you wish to do it alone, you may do it as well (just take into account corresponding workload). Please, discuss this with your colleagues and inform me (via email) about the groups before morning 14.09.2020.

Course is practically oriented (students supposed to be able to program).

- Task 1(after Lecture 1): result presentation at *Demo 1*
- Task 2(after Lecture 2): result presentation at *Demo 2*
- Task 3(after Lecture 3): result presentation at *Demo 3*
- Task 4(after Lecture 4): result presentation at *Demo 4*
- Task 5(after Lecture 5): result presentation at *Demo 5*
- Task 6(after Lecture 6): result presentation at *Demo 6*

Results of the Tasks (PowerPoint presentation and software demonstration) will be presented by the groups during the Demos. Groups will have a possibility to discuss the results of others, provide comments and share opinions (active participation in the discussions will be taken into account during evaluation). Every team member should send individual self-evaluation report where mention contribution of each team member with respect to the task (simply distribute 100 points among your team members).

Related courses

- ***TIES4520 - Semantic Technologies for Developers (7 ECTS)*** ***(28.09.-31.12.2020)***
  - *covers relevant aspects of Automated Service Integration, Semantic Web Services, Ontology Alignment (data model mapping), etc.*

Service Oriented Architecture

VS

Manufactory ⟵ *Workmanship*

Services and SOA

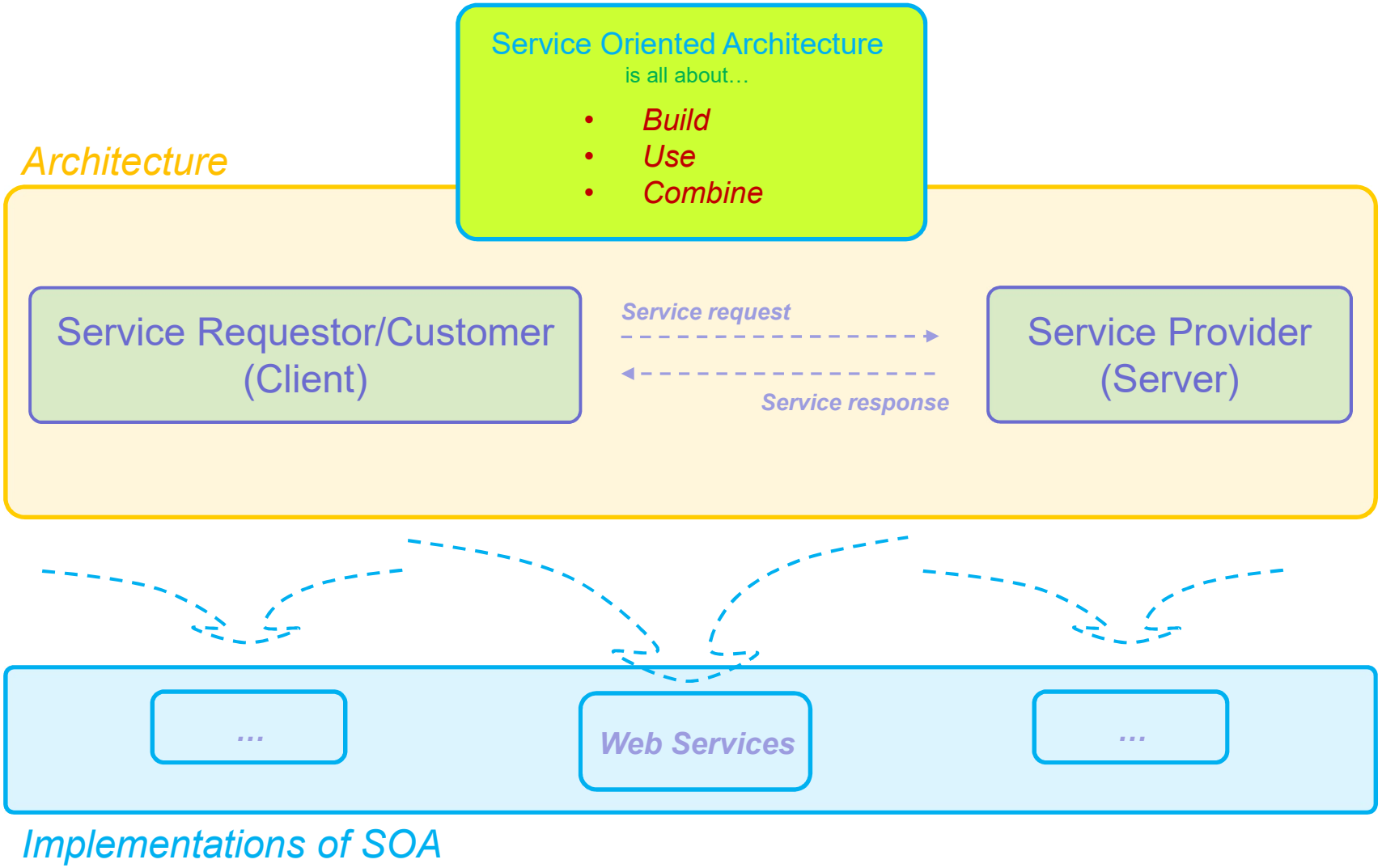
Some definitions of a Service Oriented Architecture (SOA)

- *A service-oriented architecture (SOA) is an **architectural pattern** in computer software design in which **application components provide services to other components via a communications protocol**, typically over a network. The principles of service-orientation are independent of any vendor, product or technology.[3]*
- *A service-oriented architecture is essentially a **collection of services**. These services **communicate with each other**. The communication **can involve** either **simple data passing** or it could involve **two or more services coordinating some activity**. Some means of connecting services to each other is needed.*
- *SOA is a **style of architecting applications** in such a way that they are **composed of discrete software agents** that have **simple**, **well defined interfaces** and are **orchestrated through a loose coupling** to perform a required function.*
- *A **paradigm for organizing and utilizing distributed capabilities** that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations. (OASIS group)*
- *Service-Oriented Architecture (SOA) is an architectural style that supports service-orientation. Service-orientation is a way of thinking in terms of services and service-based development and the outcomes of services. (Open Group)*
- *…*

Some definitions for a "Service" concept…

- *A service is a function (e.g. producing data, validating a customer, or providing simple analytical services, etc.) that is well-defined, self-contained, and does not depend on the context or state of other services.*
- *A service comprises a stand-alone unit of functionality available via a formally defined interface.*
- *A service is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports); is self-contained; may be composed of other services; is a "black box" to consumers of the service. (Open Group)*
- *A services requires three fundamental aspects: implementation; elementary access details; and a contract. (Ben Margolis with Joseph Sharpe: SOA for the Business Developer)*
- *…*

SOA and Web Services



Why services?

- *Services are ubiquitous and are the main building block for the Internet.*
- *All companies' IT infrastructure is based on services.*
- *Service orientation makes adapting to change easier.*
- *SOA is popular because it lets you reuse applications and it promises interoperability between heterogeneous applications and technologies.*
- *SOA allows to orchestrate existing services to build different business processes.*
- *Simple in concept, SOA is also a best practice to fix broken architectures. With the wide use of standards such as Web services, SOA is being promoted as the best way to bring architectural agility.*
- *Services uses similar concepts as modular programming and interface based design.*
- *Defining isolated services makes testing easier.*
- *…*

Be aware…

- *SOA is a valid approach to solve many of the architectural problems that enterprises face today. However, those who implement SOA typically look at it as something you buy, not something you do. Thus, many SOA projects are about purchasing some technology that is sold as 'SOA-in-a-box.' You get something-in-a-box, but not SOA, and that only adds to the problems.*
- *Web services and SOA are not synonymous. SOA is a design principle, whereas web services is an implementation technology. You can build a service-oriented application without using web services (for example, by using other traditional technologies such as Java RMI). Web services bring the platform-independent standards such as HTTP, XML, SOAP, and UDDI, thus allowing interoperability between heterogeneous technologies.*
- *In the past, loosely coupled architectures have relied upon other technologies like CORBA and DCOM or document-based approaches like EDI for B2B integration that were conceptual examples of SOA.*

Fundamental to the service model

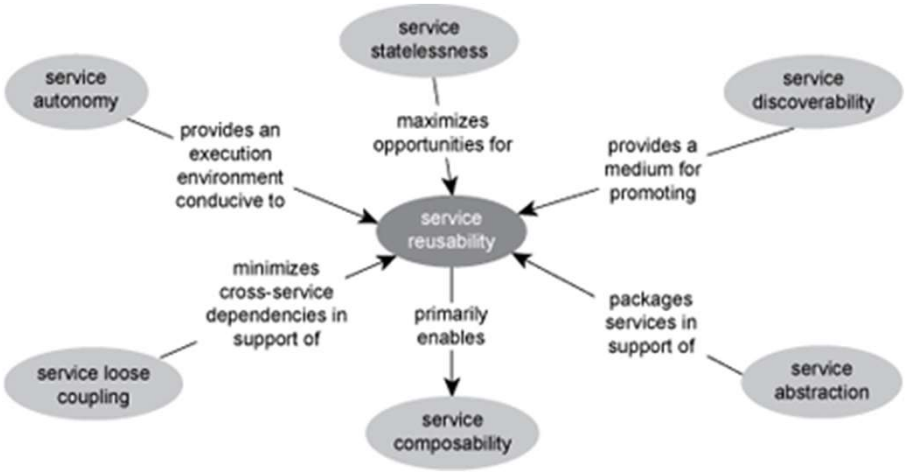
- *Separation between the interface and the implementation.*
- *Invoker of a service need/should only understand the interface.*
- *Implementation can evolve over time without disturbing the clients of the service.*
- *With abstraction of the capability from how the capability is delivered, the same interface can be offered by many implementations, or conversely, that implementations can change without affecting the aggregate application.*

- *Service interaction should be based solely upon a service's policies, schema, and contract-based behaviors. Schema defines the structure and content of the messages, while the service's contract defines the behavior of the service itself. A service's contract is generally defined using WSDL, while contracts for aggregations of services can be defined using BPEL (which, in turn, uses WSDL for each service aggregated).*
- *Contracts should be designed as explicitly as possible while taking advantage of the extensible nature of XML schema (xsd:any) and the SOAP processing model (optional headers).*
- *Interaction with the service occurs through the public interface that consists of public processes and public data representations. The public process is the entry point into the service while the public data representation represents the messages used by the process. If we use WSDL to represent a simple contract, the <message> represents the public data while the <portType> represents the public process(es).*
- *Services should be easy to consume. The service's interface (contract) should be designed to enable evolving the service without breaking contracts with existing consumers.*
- *Keep service surface area small. The more public interfaces that a service exposes the more difficult it becomes to consume and maintain it. Provide few well-defined public interfaces to your service. These interfaces should be relatively simple, designed to accept a well-defined input message and respond with an equally well-defined output message.*
- *Internal (private) implementation details should not be leaked outside of a service boundary, otherwise, it will most likely result in a tighter coupling between the service and the service's consumers.*

Main principles to be followed by services

- ***Formal contract:*** *Services adhere to a communications agreement, as defined collectively by one or more service-description documents. A service's contract consists of the following elements: service endpoint, message interchange formats, Message Exchange Patterns (MEPs), capabilities (service operations) and requirements, inputs and outputs, possible aggregation of multiple services (optional). Service consumers will rely upon a service's contract to invoke and interact with a service. Given this reliance, a service's contract must remain stable over time. Contracts should be designed as explicitly as possible.*
- ***Abstraction:*** *Beyond descriptions in the service contract, services hide logic from the outside world. The information published in a service contract is limited to what is required to effectively utilize the service, and does not contain any superfluous information that is not required for its invocation.*
- ***Loose coupling:*** *This principle is applied to the services in order to ensure that the service contract is not tightly coupled to the service consumers and to the underlying service logic and implementation. This results in service contracts that could be freely evolved without affecting either the service consumers or the service implementation.*
- ***Autonomy:*** *Services have control over the logic they encapsulate, from a Design-time and a run-time perspective. Autonomy provides services with improved independence from their execution environments. This results in greater reliability, since services can operate with less dependence on resources over which there is little or no control.*
- ***Reusability:*** *Service should has the potential to be reused across the enterprise. These reusable services are designed in a manner so that their solution logic is independent of any particular business process or technology. Logic is divided into services with the intention of promoting reuse. Service-orientation encourages reuse in all services, regardless of whether immediate requirements for reuse exist.*
- ***Discoverability:*** *Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted. Service discovery requires a common language to allow software agents to make use of one another's services without the need for continuous user intervention*
- ***Composability:*** *Services are effective composition participants, regardless of the size and complexity of the composition.*
- ***Statelessness:*** *This principle is applied to design scalable services by separating them from their state data whenever possible. This results in reduction of the resources consumed by a service as the actual state data management is delegated to an external component or to an architectural extension. By reducing resource consumption, the service can handle more requests in a reliable manner.*
- ***...***

Interrelation of the principles



- *Service **autonomy** establishes an execution environment that facilitates reuse because the service achieves increased independence and self-governance. The less dependencies a service has, the broader its reuse applicability.*
- *Service **statelessness** supports reuse because it maximizes the availability of a service and typically promotes a generic service design that defers state management and activity-specific processing outside of the service boundary.*
- *Service **abstraction** fosters reuse because it establishes the black box concept. Proprietary processing details are hidden and potential consumers are only made aware of an access point represented by a generic public interface.*
- *Service **discoverability** promotes reuse as it allows those that build consumers to search for, discover and assess services offering reusable functionality.*
- *Service **loose coupling** establishes an inherent independence that frees a service from immediate ties to others. This makes it a great deal easier to realize reuse.*
- *Service **composability** is primarily possible because of reuse. The ability for new automation requirements to be fulfilled through the composition of existing services is feasible when those services being composed are built for reuse. (It is technically possible to build a service so that its sole purpose is to be composed by another, but reuse is generally emphasized.)*

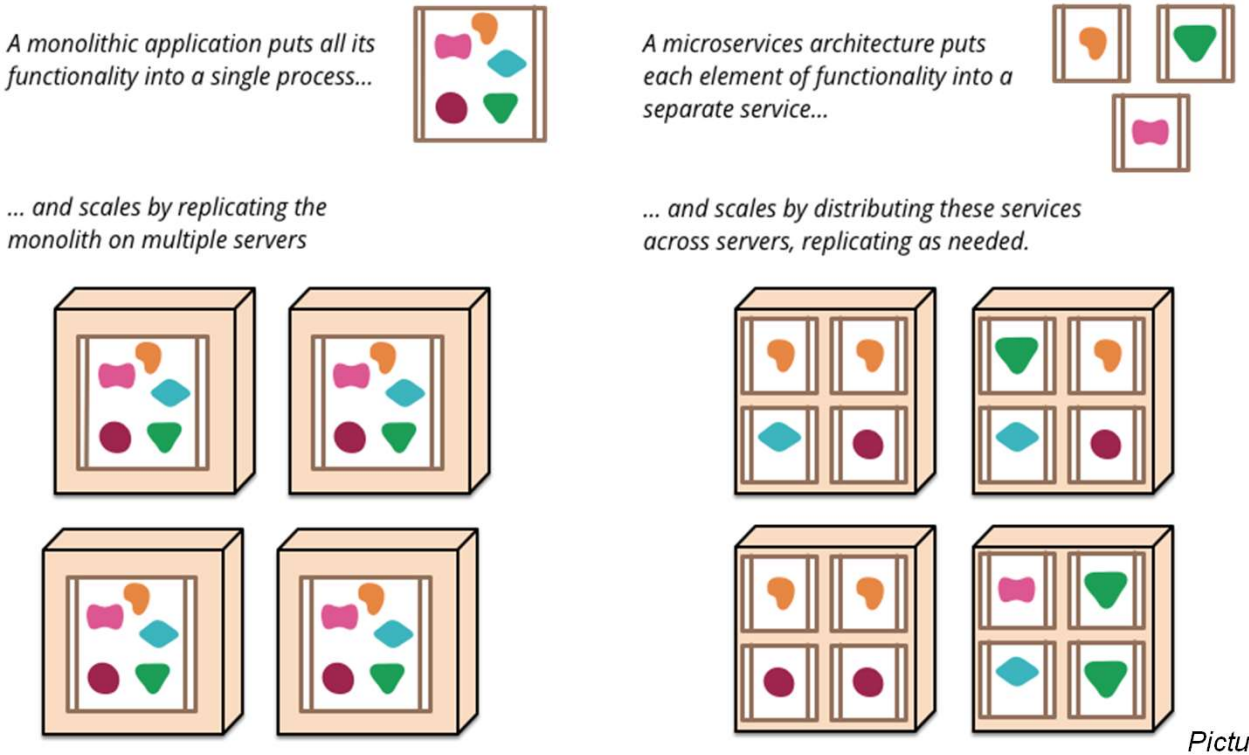
Picture is taken from [4]

Microservices

Microservice Architecture has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API [5,6,7,8,9].

- *minimum of centralized management of the services,*
- *different programming languages,*
- *different data storage technologies.*

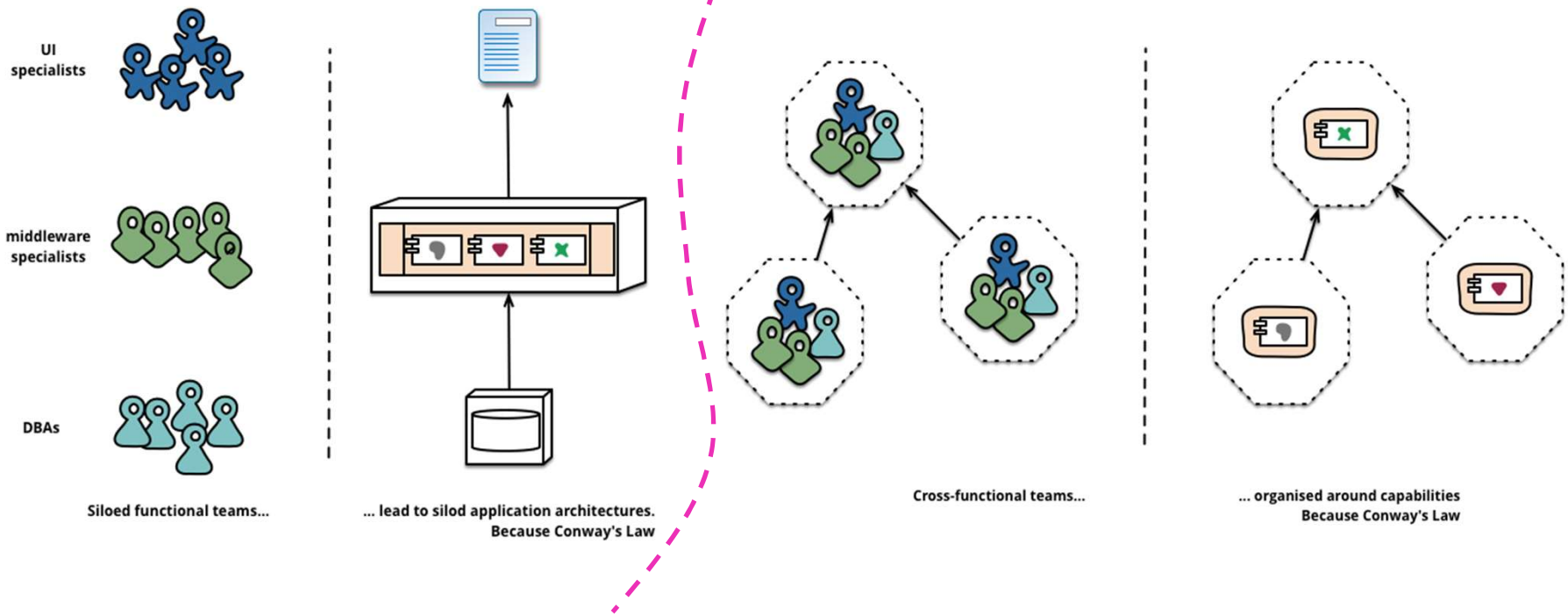
Libraries vs. Services



Picture is taken from [5]

Microservices

Organized around Business Capabilities



Decentralized Governance… Products not Projects

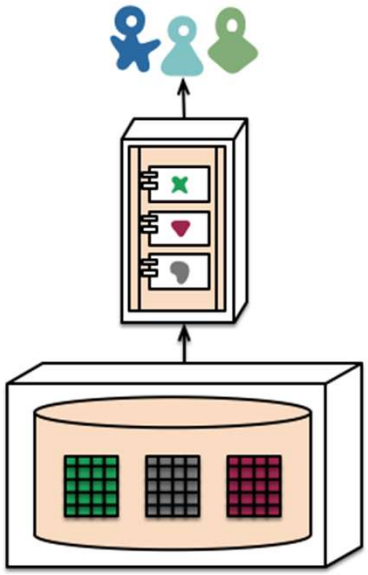
Smart endpoints instead of centralized Enterprise Service Bus

(HTTP request-response with resource API's and lightweight messaging)

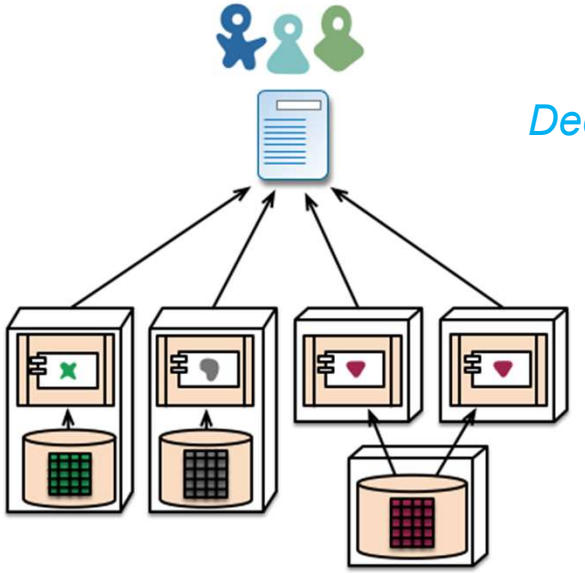
Picture is taken from [5]

Microservices

Decentralized Data Management

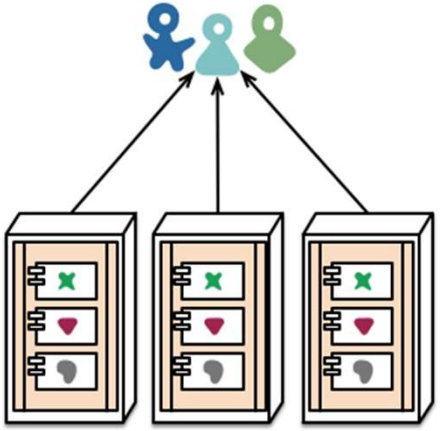


monolith - single database

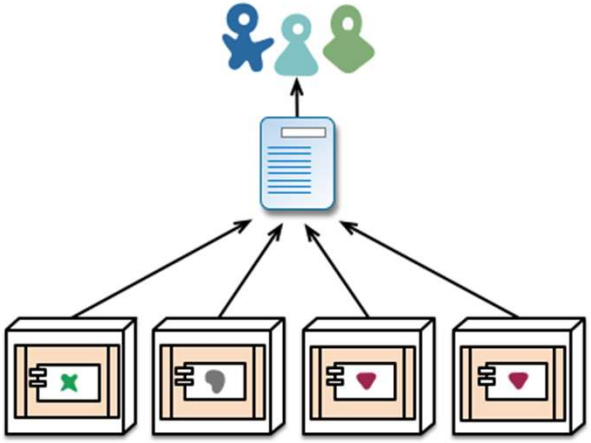


microservices - application databases

Deployment



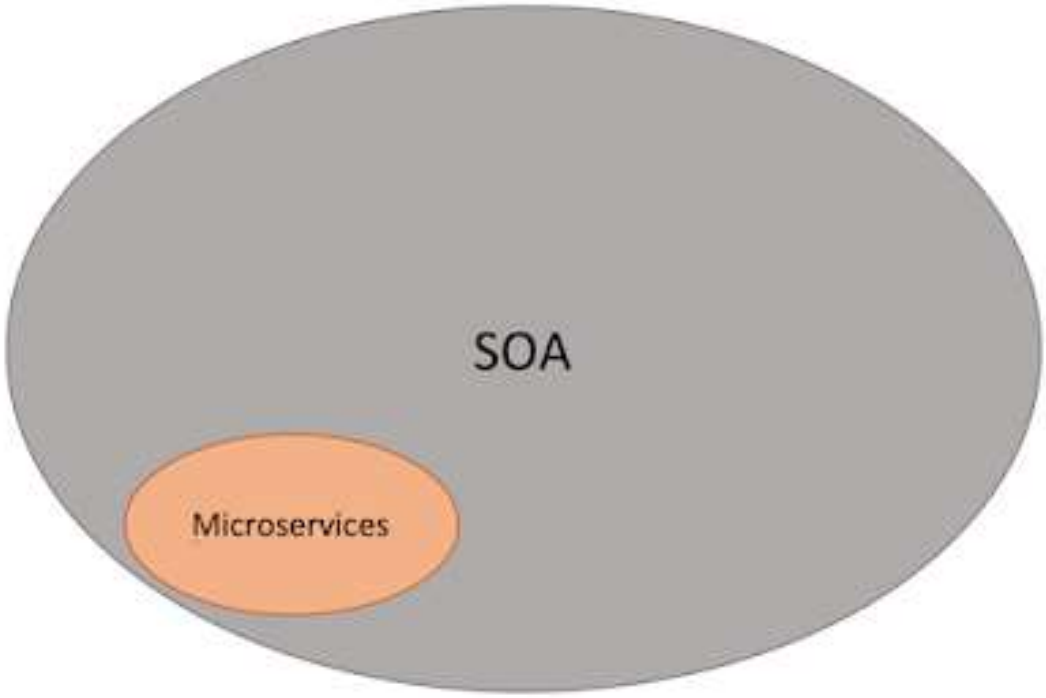
monolith - multiple modules in the same process



microservices - modules running in different processes

Picture is taken from [5]

Microservices



Picture is taken from [9]

SOAP Web Services

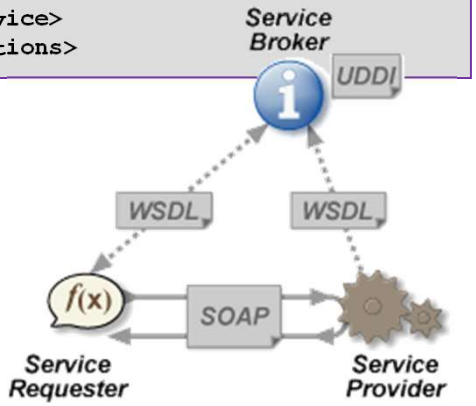
Web Services – are software systems offered over the Internet via platform- and programming-language independent interfaces defined on the basis of a set of open standards such as XML, SOAP, and WSDL.

(*http://www.tutorialspoint.com/webservices/webservices_tutorial.pdf*)

- *Web Services description/definition has been done with WSDL (http://www.tutorialspoint.com/wsdl/index.htm) that provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. Specification of only syntactic interoperability without semantic meaning of data requires programmers to reach specific agreements on the interaction of web services and makes automatic composition very difficult;*
- *Web Services interaction has been implemented via XML-based SOAP (Simple Object Access Protocol) messaging (http://www.tutorialspoint.com/soap/index.htm);*
- *Web Service composition has been supported by several languages in order to combine services in a process-oriented way (e.g., BPEL4WS, WS-BPEL);*
- *Web Service publication has been performed via UDDI (Universal Description, Discovery, and Integration)registries that were relatively complex and do not support expressive queries (http://www.tutorialspoint.com/uddi/index.htm).*

WSDL Document Structure

```
<definitions>
   <types>
    definition of types........
   </types>
   <message>
    definition of a message....
   </message>
   <portType>
      <operation>
       definition of a operation.......
      </operation>
   </portType>
   <binding>
    definition of a binding....
   </binding>
   <service>
    definition of a service....
   </service>
</definitions>
```

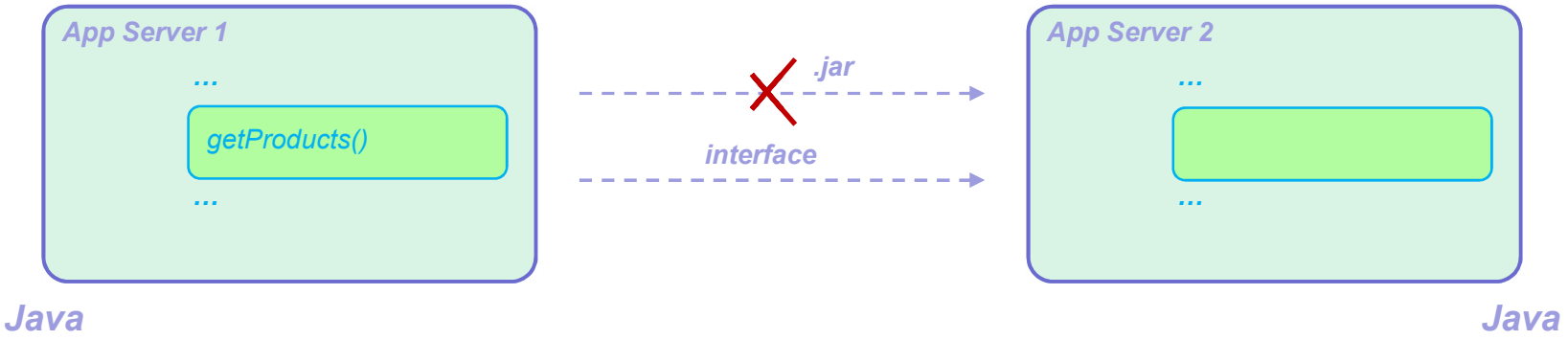


Web Service related practical tutorials:

- *http://www.softwareagility.gr/index.php?q=node/29*
- *http://www.javatpoint.com/web-services-tutorial*
- *http://docs.oracle.com/javaee/6/tutorial/doc/bnayl.html*
- *http://wiki.eclipse.org/Creating_a_top-down_Axis2_Web_service*
- *http://www.eclipse.org/webtools/community/education/web/t320/Implementing_a_Simple_Web_Service.pdf*
- *https://docs.oracle.com/cd/E17802_01/webservices/webservices/docs/1.6/tutorial/doc/JavaWSTutorial.pdf*

SOAP Web Services

Why Web Services?..



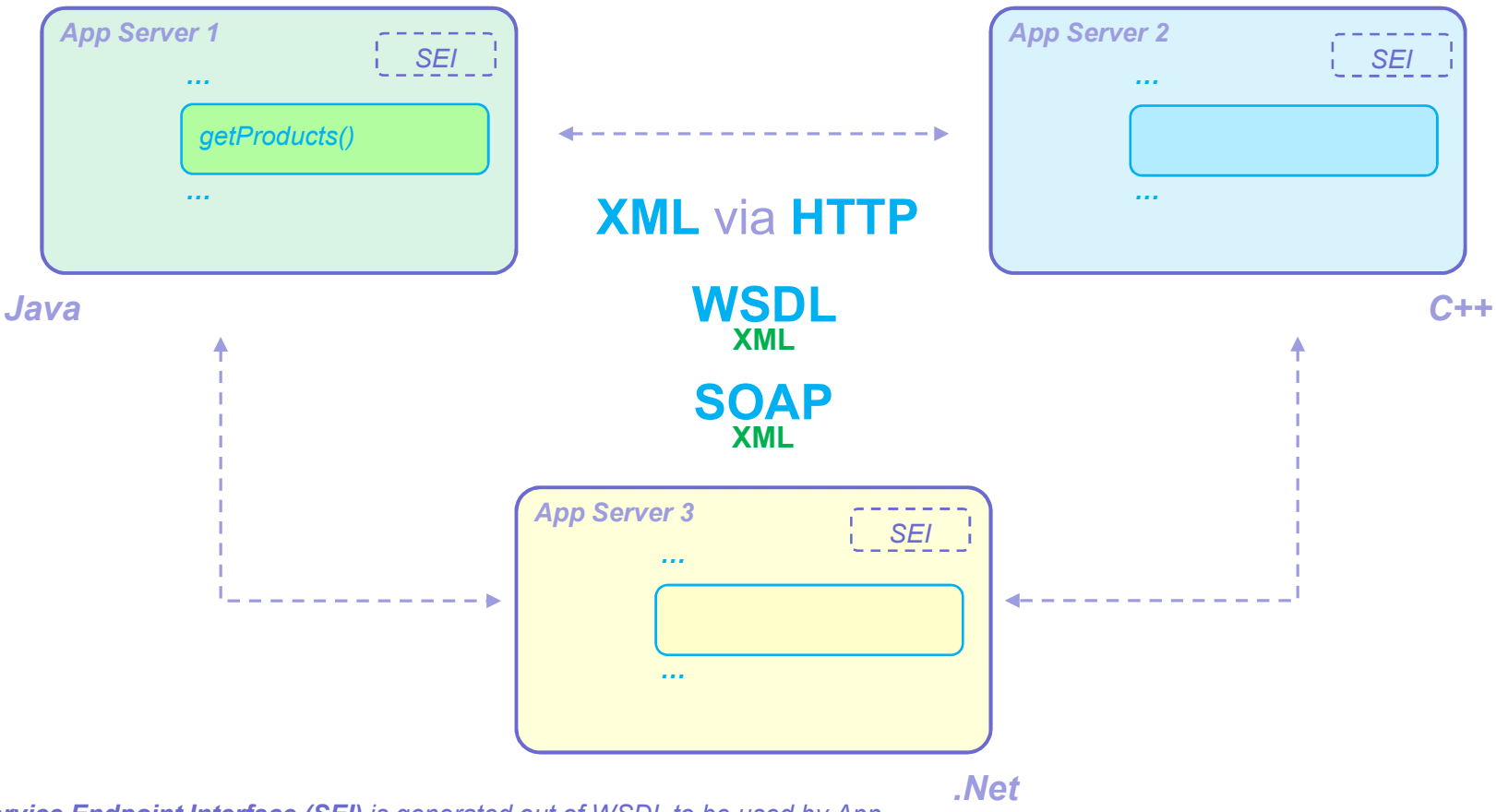
- *Simple packaging of business logic into **.jar** file and porting it to other server does not work in case of dependency of business logic on other resources (applications, database, etc.) from the mother server.*
- *To allow consumer to use an implementation of some business logic, provider should provide an **Interface** as a form of contract.*

… but, if we consider clients that are implemented with different technology (different programming language, platform, etc.),

we have to provide Technology Independent Interface!!!

SOAP Web Services

Why Web Services?..



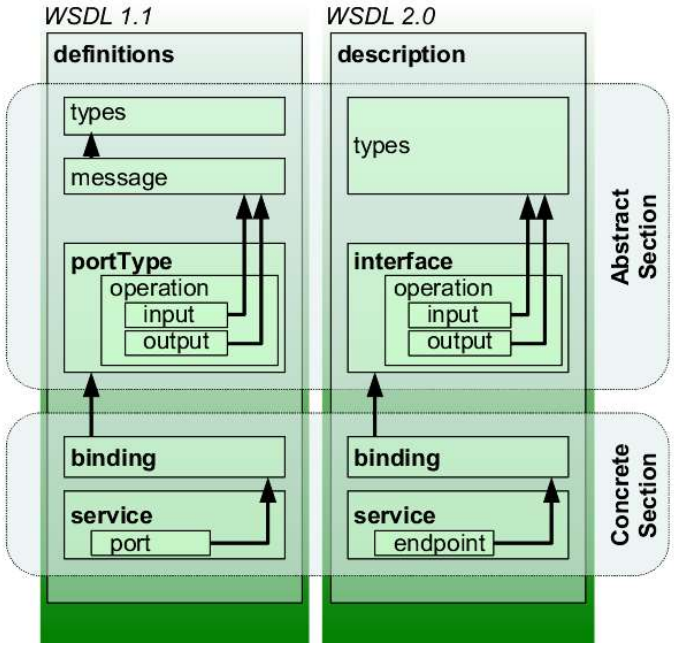
Service Endpoint Interface (SEI) *is generated out of WSDL to be used by App.*
It covers all the complexity of web service (converting objects and access to web service into SOAP message).

SOAP Web Services

Web Services Description Language (WSDL) – is an XML-based interface definition language that is used for describing the functionality offered by a web service. The current version of WSDL is *WSDL 2.0*.

WSDL file – is WSDL description of a web service, which provides a machine-readable description of:

- how the service can be called;
- what parameters it expects;
- what data structures it returns.



The WSDL describes services as collections of network *endpoints*, or *ports*. The abstract definitions of *ports* and *messages* are separated from their concrete use or instance, allowing the reuse of these definitions. A *port* is defined by associating a network address with a reusable *binding*, and a collection of ports defines a *service*. *Messages* are abstract descriptions of the data being exchanged, and *port types* are abstract collections of supported *operations*. The concrete protocol and data format specifications for a particular *port type* constitutes a reusable *binding*, where the *operations* and *messages* are then bound to a concrete network protocol and message format.

WSDL is often used in combination with SOAP and an XML Schema to provide Web services over the Internet. A client program connecting to a Web service can read the WSDL file to determine what operations are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the operations listed in the WSDL file using for example XML over HTTP.

WSDL 2.0 is a *W3C recommendation* (WSDL 1.1 is not). WSDL 2.0 specification offers better support for *RESTful web services* since accepts binding to all the HTTP request methods (not only GET and POST as in version 1.1), and is much simpler to implement. However support for this specification is still poor in software development kits, and is not supported by *Business Process Execution Language* (*BPEL*) version 2.0.

Web Application Description Language (*WADL*) is a machine-readable XML description of HTTP-based web services. WADL is the REST equivalent of SOAP's WSDL. But it is not standardized by W3C yet.

- WSDL: *https://en.wikipedia.org/wiki/Web_Services_Description_Language*
- WADL: *https://en.wikipedia.org/wiki/Web_Application_Description_Language*

WSDL Analyzer: *https://www.wsdl-analyzer.com/*

SOAP Web Services

Simple Object Access Protocol (SOAP) – is industry open-standard, XML-based messaging protocol for exchanging information among computers. It can be delivered via a variety of transport protocols, but the initial focus of SOAP is remote procedure calls transported via HTTP. SOAP messages are written entirely in XML and are therefore uniquely platform- and language-independent.

- SOAP is a communication protocol designed to communicate via Internet.
- SOAP can extend HTTP for XML messaging.
- SOAP provides data transport for Web services.
- SOAP can exchange complete documents or call a remote procedure.
- SOAP can be used for broadcasting a message.
- SOAP is platform- and language-independent.
- SOAP is the XML way of defining what information is sent and how.
- SOAP enables client applications to easily connect to remote services and invoke remote methods.

A SOAP message contains:

- *Envelope* : Defines the start and the end of the message. (mandatory element).
- *Header*: Contains any optional attributes of the message used in processing the message. (optional element).
- *Body*: Contains the XML data comprising the message being sent. (mandatory element).
- *Fault*: Provides information about errors that occur while processing the message. (optional element).

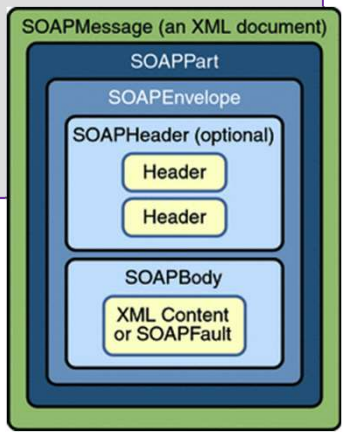
SOAP Message Structure

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

   <SOAP-ENV:Header>
      ...
      ...
   </SOAP-ENV:Header>

   <SOAP-ENV:Body>
      ...
      ...
      <SOAP-ENV:Fault>
         ...
         ...
      </SOAP-ENV:Fault>
      ...
   </SOAP-ENV:Body>

</SOAP_ENV:Envelope>
```



- SOAP: *http://www.tutorialspoint.com/soap/soap_tutorial.pdf*
  http://www.digilife.be/quickreferences/pt/xml%20messaging%20with%20soap.pdf

Demo

Client to a SOAP Web Service

Check a list of SOAP Web Services from the course webpage…

Web Service: Number Convertor (http://www.dataaccess.com/webservicesserver/numberconversion.wso)

WSDL: http://www.dataaccess.com/webservicesserver/numberconversion.wso?WSDL

wsimport tool is used to parse an existing Web Services Description Language (WSDL) file and generate required files (JAX-WS portable artifacts) for web service client to access the published web services.

(*http://www.mkyong.com/webservices/jax-ws/jax-ws-wsimport-tool-example/*)

If you use Java version later than 1.8.0_152, it might not include WS related Jars. So, add them to the project or to the server you use, or simply use Java version that has corresponding classes. There are some sources for jar download:

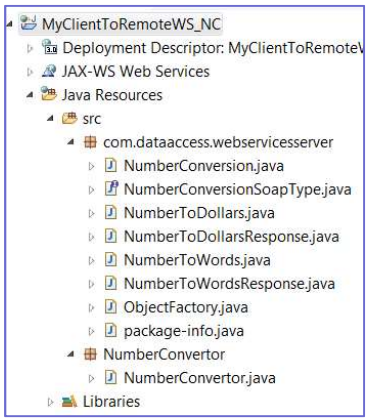
- *http://www.java2s.com/Code/Jar/j/Downloadjavaxxmlws30preludejar.htm*
- *https://mvnrepository.com/artifact/org.glassfish/javax.xml.ws/10.0-b28*

```
wsimport [options] <wsdl_uri>
```

Use *-keep* and *-s* options to generate and store *.class* and *.java* source files

```
wsimport -keep -s <folder> <wsdl_uri>
```

1) Import *.java* files to your project:



2) Check *wsdl:service* and *wsdl:port* elements from the service WSDL file:

```
      </output>
    </operation>
  </binding>
  <service name="NumberConversion">
    <documentation>
      The Number Conversion Web Service, implemented with DataFlex, provides functions that convert numbers into words or dollar amounts.
    </documentation>
    <port name="NumberConversionSoap" binding="tns:NumberConversionSoapBinding">
      <soap:address location="http://www.dataaccess.com/webservicesserver/numberconversion.wso"/>
    </port>
    <port name="NumberConversionSoap12" binding="tns:NumberConversionSoapBinding12">
      <soap12:address location="http://www.dataaccess.com/webservicesserver/numberconversion.wso"/>
    </port>
  </service>
</definitions>
```

3) Use corresponding java classes to get an access to the service functionality:

```
 9 public class NumberConvertor {
10     public static void main(String[] args) {
11         BigDecimal input_D = new BigDecimal("145.75");
12         BigInteger input_N = new BigInteger("23");
13         NumberConversion NC_service = new NumberConversion(); //created service object
14         NumberConversionSoapType NC_serviceSOAP = NC_service.getNumberConversionSoap(); //create SOAP object (a port of the service)
15         String result_D = NC_serviceSOAP.numberToDollars(input_D);
16         String result_N = NC_serviceSOAP.numberToWords(input_N);
17         System.out.println("Number is: "+result_N);
18         System.out.println("Price is: "+result_D);
19     }
20 }
```

Demo Project uses Java 1.8

Demo

Client to a SOAP Web Service

Build Maven Dynamic Web Project

- Build new Dynamic Web Project in Eclipse.
- Convert it into Maven project (configure -> convert to Maven project).
- Edit pom.xml file with following dependencies
- Define targeted runtime (e.g. Apache Tomcat v9.0). Right click on project (properties -> Targeted Runtimes).

```
<plugin>
   <groupId>org.codehaus.mojo</groupId>
   <artifactId>jaxws-maven-plugin</artifactId>
   <version>2.6</version>
   <executions>
      <execution>
         <id>wsimport-from-jdk</id>
         <goals>
            <goal>wsimport</goal>
         </goals>
      </execution>
   </executions>
   <configuration>
      <wsdlUrls>
         <wsdlUrl> url to WSDL file </wsdlUrl>
      </wsdlUrls>
      <keep>true</keep>
      <packageName>com.soap.ws.client</packageName>
      <sourceDestDir>src</sourceDestDir>
   </configuration>
</plugin>
```

```
<dependencies>
      <dependency>
         <groupId>javax.servlet</groupId>
         <artifactId>javax.servlet-api</artifactId>
         <version>3.1.0</version>
         <scope>provided</scope>
      </dependency>
      <dependency>
         <groupId>com.sun.xml.ws</groupId>
         <artifactId>jaxws-ri</artifactId>
         <version>2.3.1</version>
         <type>pom</type>
      </dependency>
</dependencies>
```

Generate JAX-WS portable artifacts for web service client to access the published web services

- Edit pom.xml file with following plugin, and specify corresponding WSDL location
- Apply changes (right click on pom.xml file, Maven -> Update Project)

Useful links:

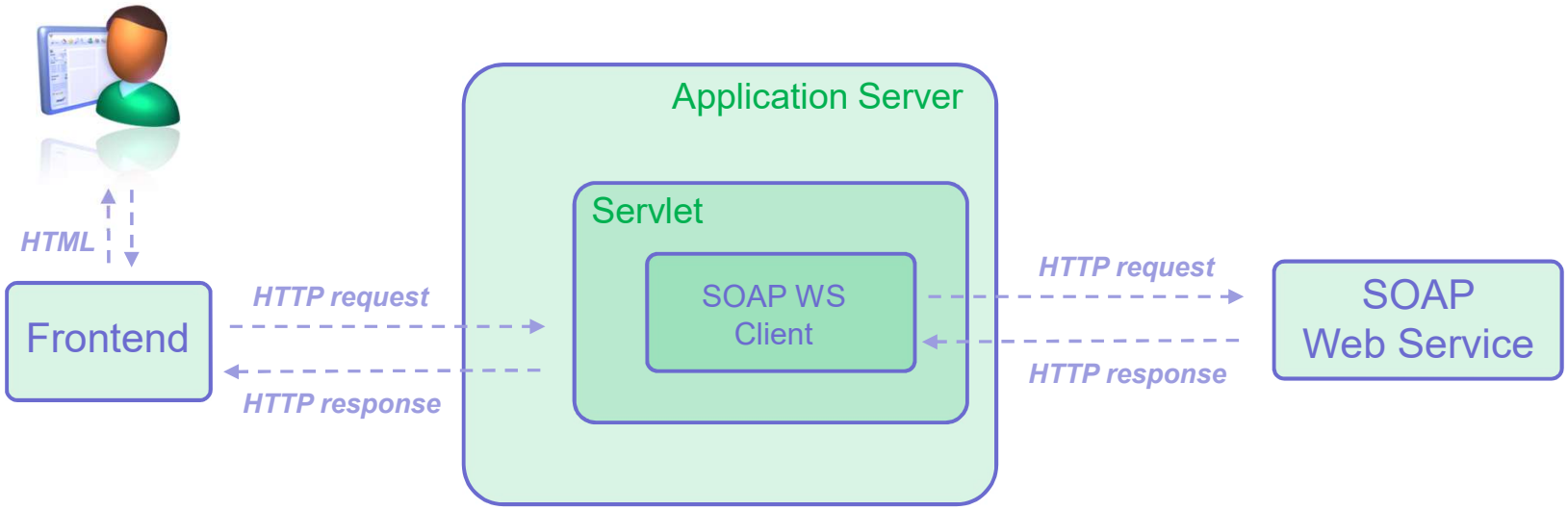
https://crunchify.com/how-to-create-dynamic-web-project-using-maven-in-eclipse/
https://www.baeldung.com/jax-ws
https://www.baeldung.com/java-soap-web-service

Demo

Java Servlets

Servlets provide a component-based, platform-independent method for building Web-based applications. A servlet is an extension to a server that enhances the server's functionality. The most common use for a servlet is to extend a web server by providing dynamic web content. A servlet is a normal Java class which extends from the *javax.servlet.http.HttpServlet* class.

- In the class you can override the *doPost* or *doGet* method depending on the type of requests this Servlet needs to answer to.
- In the method, you can call methods on the *request* and *response* objects to get the query parameters, set status codes and write the response itself.



- Servlets: *http://www.oracle.com/technetwork/java/servlet-142430.html*
  http://www.tutorialspoint.com/servlets/index.htm
- Handling HTML form data with Java Servlet: *http://www.codejava.net/java-ee/servlet/handling-html-form-data-with-java-servlet*
- JavaScript, servlet and JSP: *https://www.quora.com/What-is-the-difference-between-JavaScript-servlet-and-JSP*
- Ajax: *https://netbeans.org/kb/docs/web/ajax-quickstart.html*
  http://stackoverflow.com/questions/4112686/how-to-use-servlets-and-ajax

Task 1

References

[1] http://www.digitalistmag.com/technologies/cloud-computing/2012/07/26/top-9-challenges-in-cloud-computing-that-are-slowing-its-adoption-011918

[2] http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018

[3] https://msdn.microsoft.com/en-us/library/bb833022.aspx

[4] http://searchsoa.techtarget.com/tip/The-principles-of-service-orientation-part-6-of-6-Principle-interrelationships-and-service-layers

[5] https://www.martinfowler.com/articles/microservices.html

[6] https://smartbear.com/learn/api-design/what-are-microservices/

[7] https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/architect-microservice-container-applications/microservices-architecture

[8] https://dzone.com/articles/microservices-vs-soa-is-there-any-difference-at-al

[9] https://dzone.com/articles/microservices-vs-soa-2

If you use Java version later than 1.8_152, it might not include ws related Jars. So, add them to the project or to the server you use, or simply use Java version that has corresponding classes. Alternatively, build Maven project with corresponding dependencies. There are some sources for jar download:

- http://www.java2s.com/Code/Jar/j/Downloadjavaxxmlws30preludejar.htm
- https://mvnrepository.com/artifact/org.glassfish/javax.xml.ws/10.0-b28