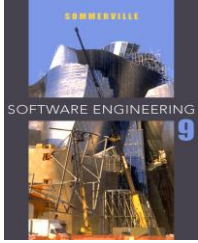
Service-oriented Architecture



Topics covered

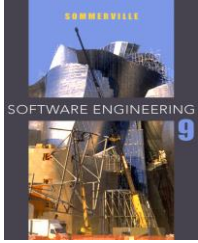
Services as reusable components

REST services

Software development with services

Service engineering

Cloud computing



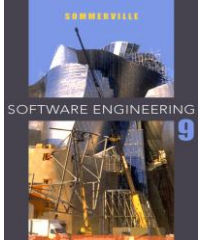
Web services

A web service is an instance of a more general notion of a service:

> *"an act or performance offered by one party to another. Although the process may be tied to a physical product, the performance is essentially intangible and does not normally result in ownership of any of the factors of production".*

The essence of a service, therefore, is that the provision of the service is independent of the application using the service.

Service providers can develop specialized services and offer these to a range of service users from different organizations.

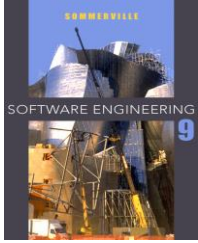


Service-oriented architectures

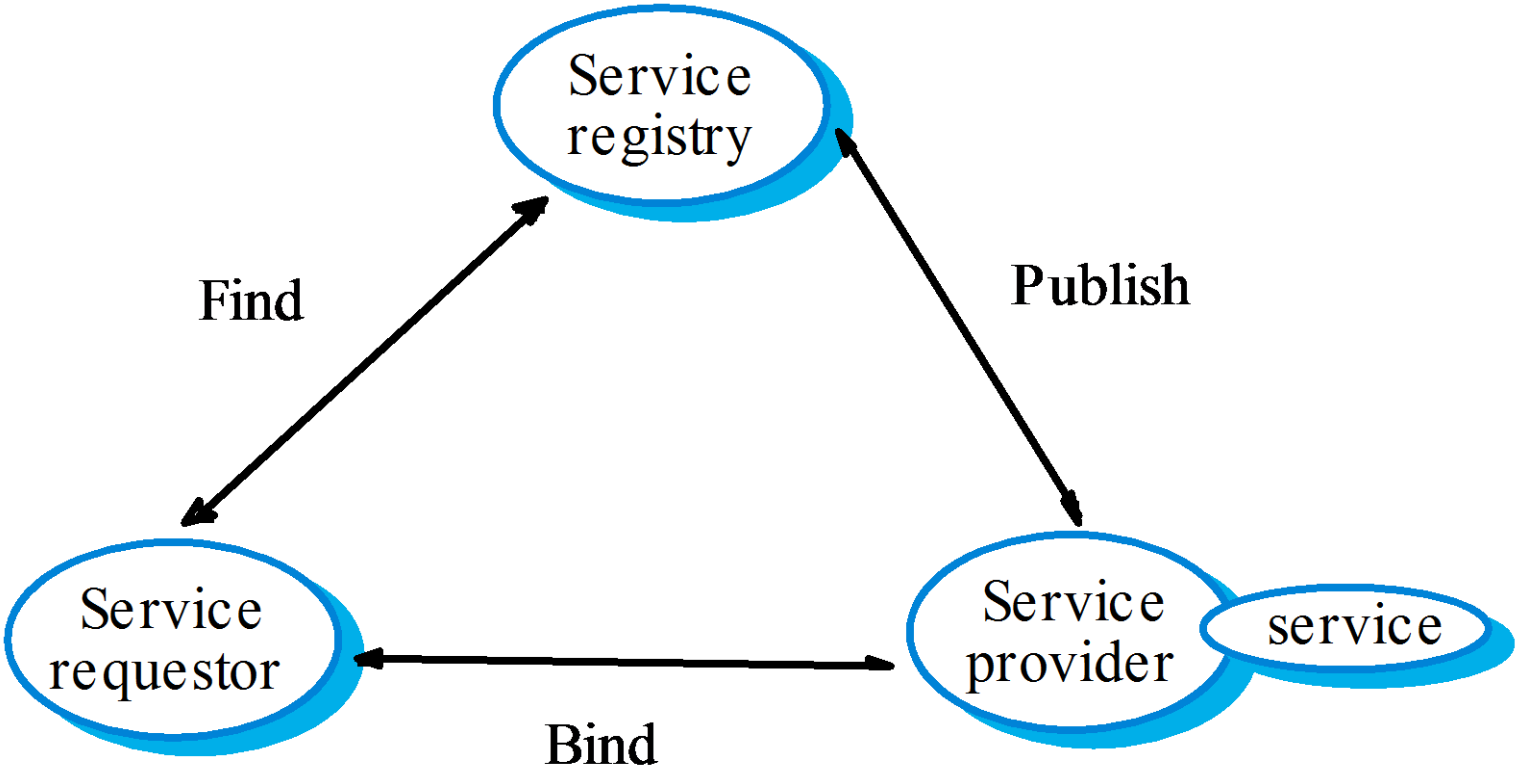
A means of developing distributed systems where the components are stand-alone services

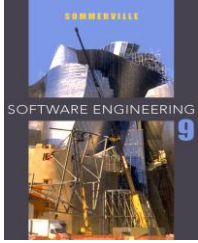
Services may execute on different computers from different service providers

Standard protocols have been developed to support service communication and information exchange



Service-oriented architecture





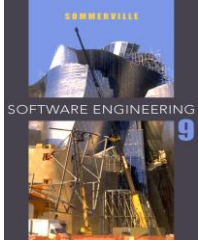
Benefits of SOA

Services can be provided locally or outsourced to external providers

Services are language-independent

Investment in legacy systems can be preserved

Inter-organisational computing is facilitated through simplified information exchange



Key standards

SOAP

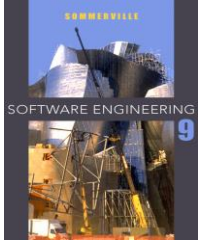
- A message exchange standard that supports service communication

WSDL (Web Service Definition Language)

- This standard allows a service interface and its bindings to be defined

WS-BPEL

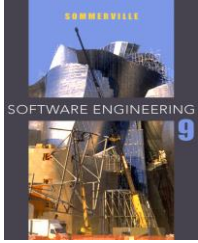
- A standard for workflow languages used to define service composition



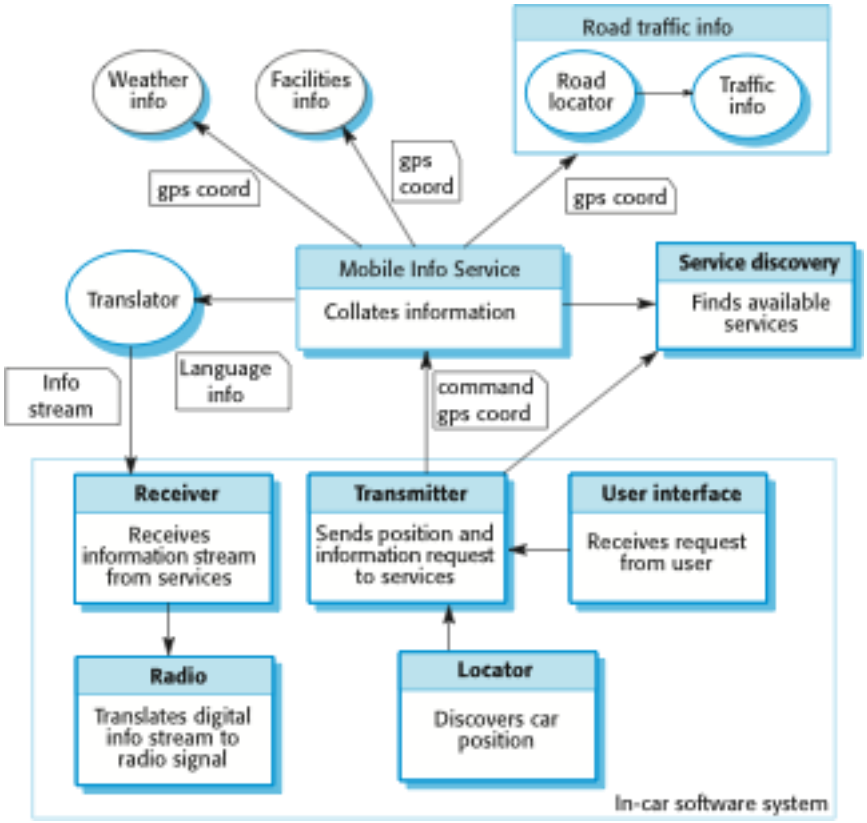
Services scenario

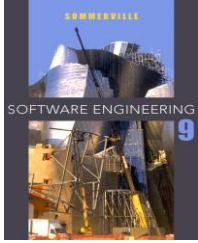
An in-car information system provides drivers with information on weather, road traffic conditions, local information etc. This is linked to car radio so that information is delivered as a signal on a specific radio channel.

The car is equipped with GPS receiver to discover its position and, based on that position, the system accesses a range of information services. Information may be delivered in the driver's specified language.



A service-based, in-car information system





Advantage of SOA for this application

It is not necessary to decide when the system is programmed or deployed what service provider should be used or what specific services should be accessed.

- As the car moves around, the in-car software uses the service discovery service to find the most appropriate information service and binds to that.
- Because of the use of a translation service, it can move across borders and therefore make local information available to people who don’t speak the local language.

Web service description language

The service interface is defined in a service description expressed in WSDL (Web Service Description Language).

The WSDL specification defines

- What operations the service supports and the format of the messages that are sent and received by the service
- How the service is accessed - that is, the binding maps the abstract interface onto a concrete set of protocols
- Where the service is located. This is usually expressed as a URI (Universal Resource Identifier)

XML (eXtensible Markup Language)

```
<note>
   <to>Toni</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this
   weekend!</body>
</note>
```

XML (eXtensible Markup Language)

```
<breakfast_menu>
    <food>
        <name>Belgian Waffles</name>
        <price>$5.95</price>
        <description>
                two of our famous Belgian Waffles with
    plenty of real maple syrup
        </description>
        <calories>650</calories>
    </food>
    <food>
    …..
    </food>

</breakfast_menu>
```

XML

Using XML formats as your machine-processable representations for resources allows applying new tools to old data

It also simplifies interconnection with remote systems

XML has plenty of tools, as we all know

Какво не е XML?

Заместител на HTML
(Може да се каже, че HTML е частен случай на XML)

Ориентиран към визуализацията
(По-скоро е ориентиран към съдържанието)

Език за програмиране
(може да се използва в комбинация с почти всеки друг език)

Протокол за комуникация
(XML се предава по мрежата и може да се използва за дефиниция на протокол)

База данни
(XML може да се съхранява в база-данни)

Какво е XML?

XML е мета-език за текстови документи/данни

XML позволява да се дефинира конкретния език за представяне на текстови документи/данни

Why not just use plain HTML?

Web pages are designed to be understood by people, who care about layout and styling, not just raw data

Every URI could have a human-readable and a machine-processable representation:

- Web Services clients ask for the machine-readable one
- Browsers ask for the human-readable one

Ползи от употребата на XML

Най-вече зависят от програмиста

Преносимост на данните

Оперативна съвместимост

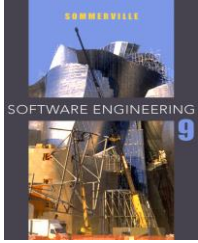
Лесно се използва от програми (machine readble)

(В повечето случаи) Лесно се чете или е разбираем от хора

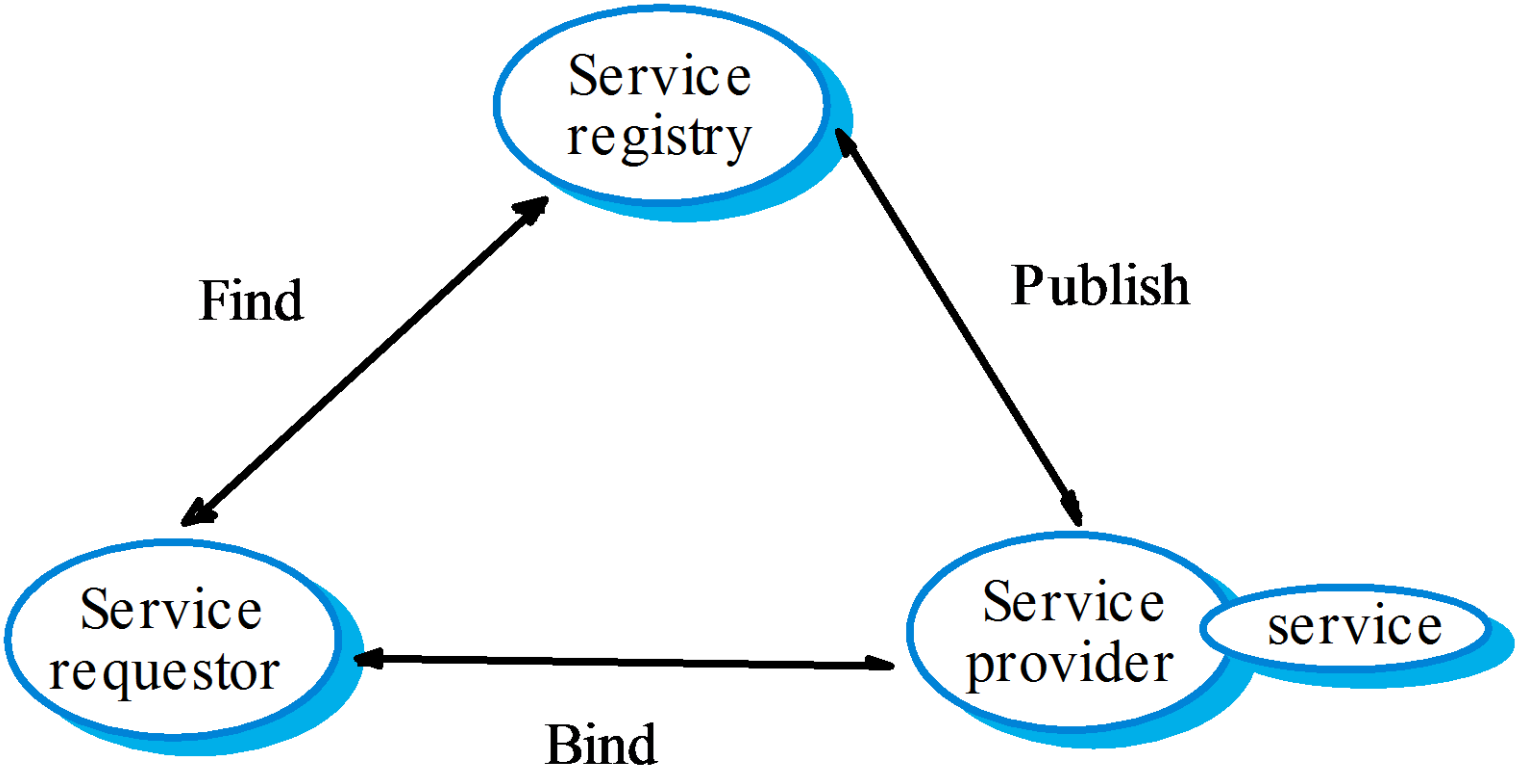
Гъвкав и лесно се променя

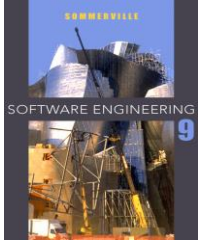
Голямо количество стандарти го използват и е широко разпространен
- Какво е “docx”, “pptx”, “xslx”?

Много инструменти го поддържат



Service-oriented architecture





Key standards

SOAP

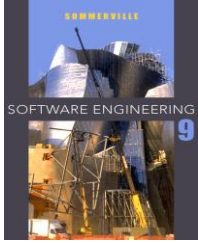
- A message exchange standard that supports service communication

WSDL (Web Service Definition Language)

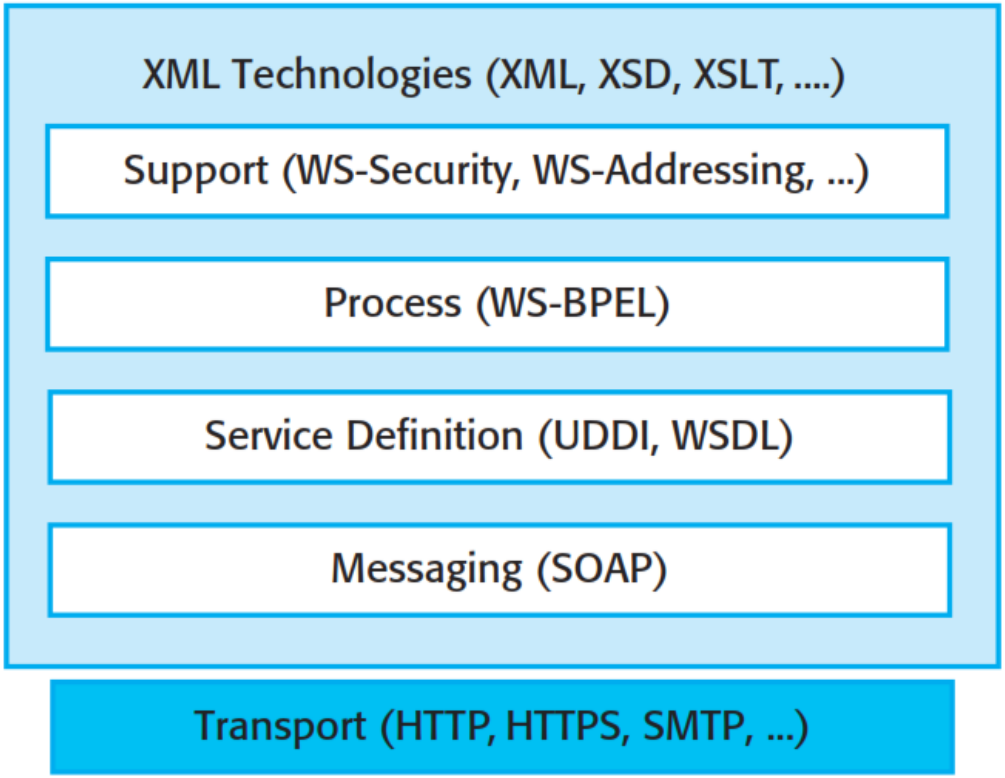
- This standard allows a service interface and its bindings to be defined

WS-BPEL

- A standard for workflow languages used to define service composition



Web service standards



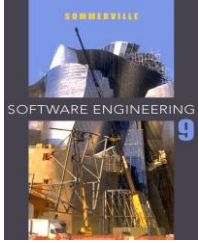
SOAP request

Which professional snowboarder endorses the K2 FatBob?

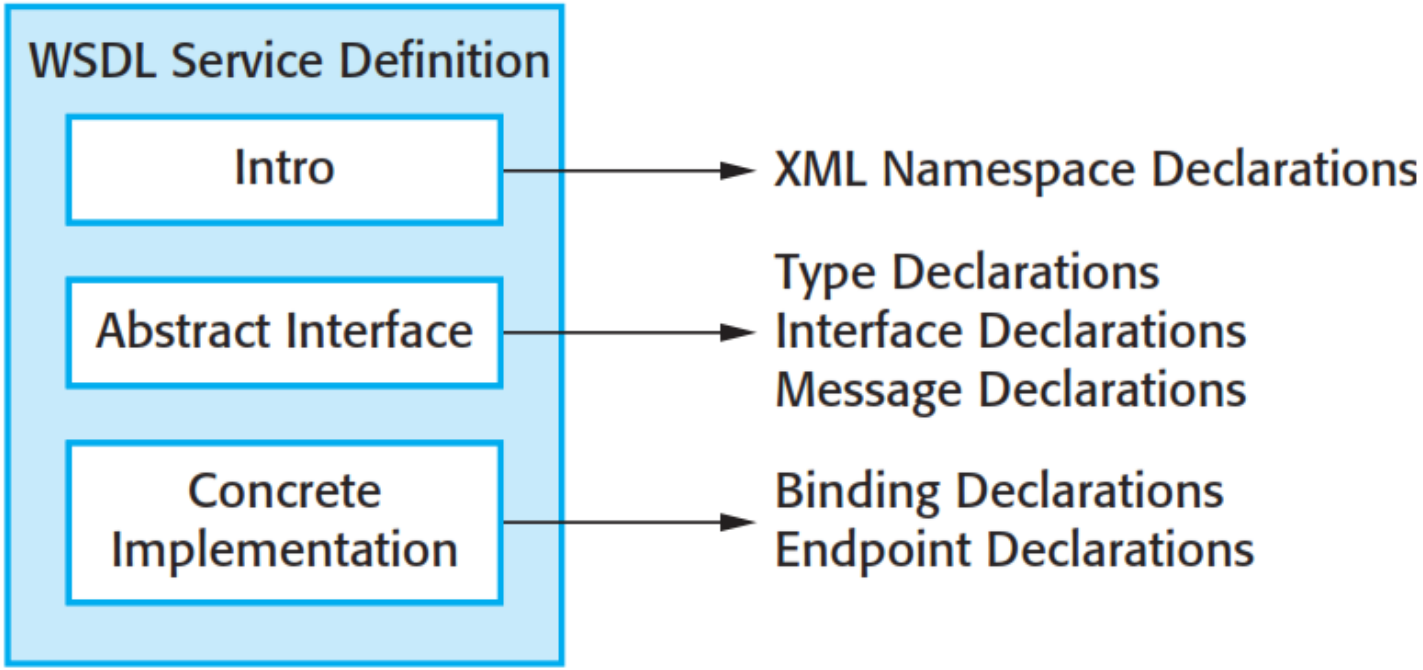
```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <SOAP-ENV:Body>
       <m:GetEndorsingBoarder xmlns:m="http://namespaces.snowboard-info.com">
<manufacturer>K2</manufacturer>
            <model>Fatbob</model>
      </m:GetEndorsingBoarder>
 </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

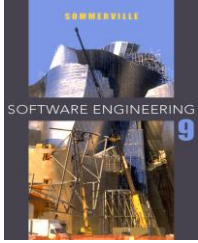
SOAP response

```
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Body>
                <m:GetEndorsingBoarderResponse
                        xmlns:m="http://namespaces.snowboard-
info.com">
                <endorsingBoarder>Chris Englesmann</endorsingBoarder>
                </m:GetEndorsingBoarderResponse>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



Organization of a WSDL specification



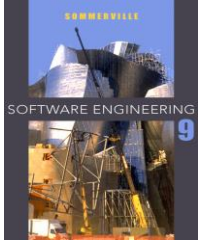


WSDL specification components

The ‘what’ part of a WSDL document, called an interface, specifies what operations the service supports, and defines the format of the messages that are sent and received by the service.

The ‘how’ part of a WSDL document, called a binding, maps the abstract interface to a concrete set of protocols. The binding specifies the technical details of how to communicate with a Web service.

The ‘where’ part of a WSDL document describes the location of a specific Web service implementation (its endpoint).



RESTful web services

Current web services standards have been criticized as ‘heavyweight’ standards that are over-general and inefficient.

REST (REpresentational State Transfer) is an architectural style based on transferring representations of resources from a server to a client.

This style underlies the web as a whole and is simpler than SOAP/WSDL for implementing web services.

RESTFul services involve a lower overhead than so-called ‘big web services’ and are used by many organizations implementing service-based systems that do not rely on externally-provided services.

REST services

REST

REpresentational State Transfer

Олекотен вариант на стандартните уеб-услуги

На мястото на сравнително тежкия класически SOAP се използват HTTP заявки

REST fundamentals

The basic API is C.R.U.D

- Create (HTTP POST)
- Retrieve (HTTP GET) *[no side effects]*
- Update (HTTP PUT)
- Delete (HTTP DELETE)

HTTP-REST Request Basics

The HTTP request is sent from the client.

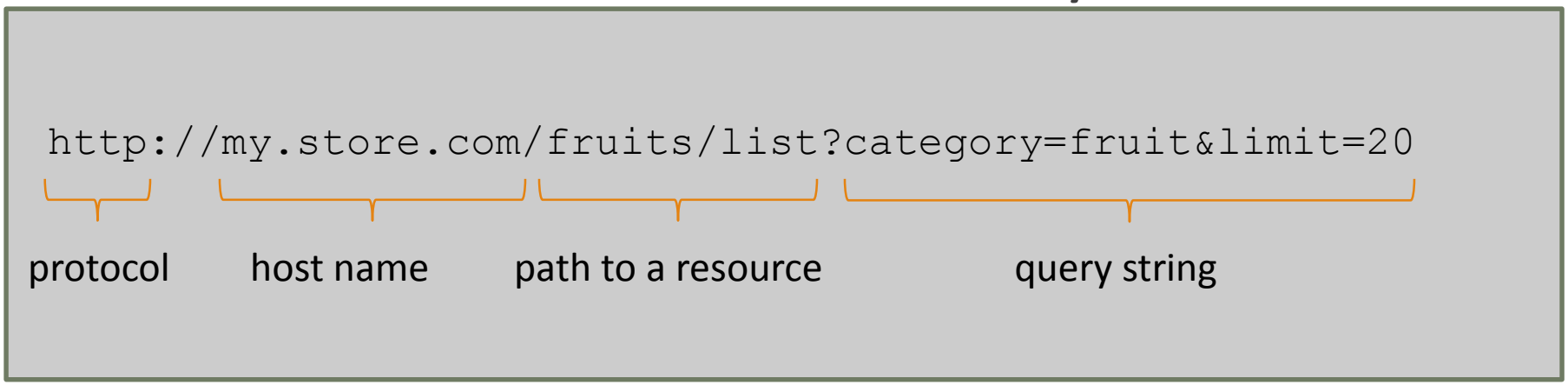
- Identifies the location of a resource.
- Specifies the verb, or HTTP method to use when accessing the resource.
- Supplies optional request headers (name-value pairs) that provide additional information the server may need when processing the request.
- Supplies an optional request body that identifies additional data to be uploaded to the server (e.g. form parameters, attachments, etc.)

HTTP-REST Response Basics

The HTTP response is sent from the server.

- Gives the status of the processed request.
- Supplies response headers (name-value pairs) that provide additional information about the response.
- Supplies an optional response body that identifies additional data to be downloaded to the client (html, xml, binary data, etc.)

HTTP-REST Vocabulary

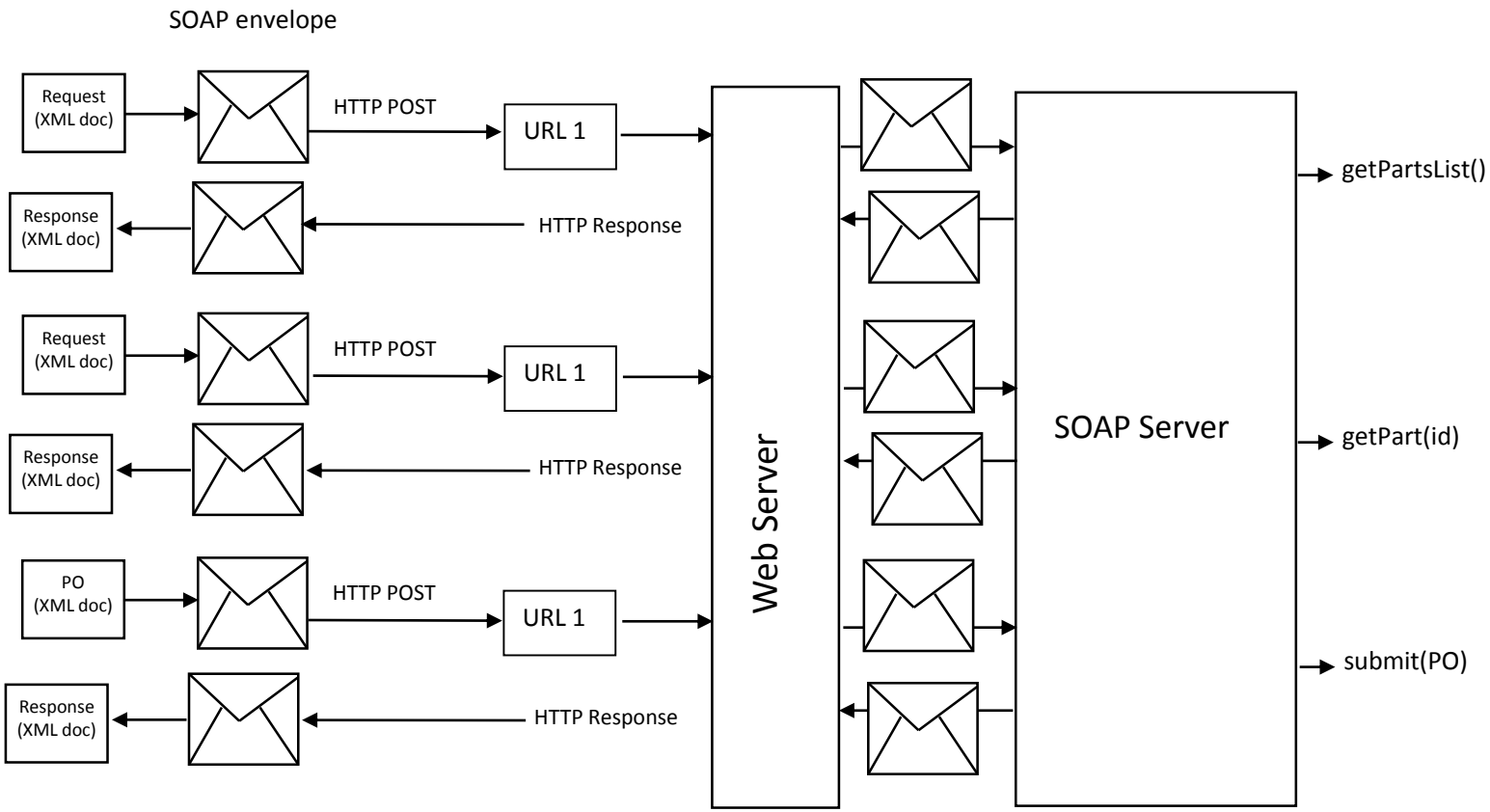


The protocol identifies the transport scheme that will be used to process and respond to the request.

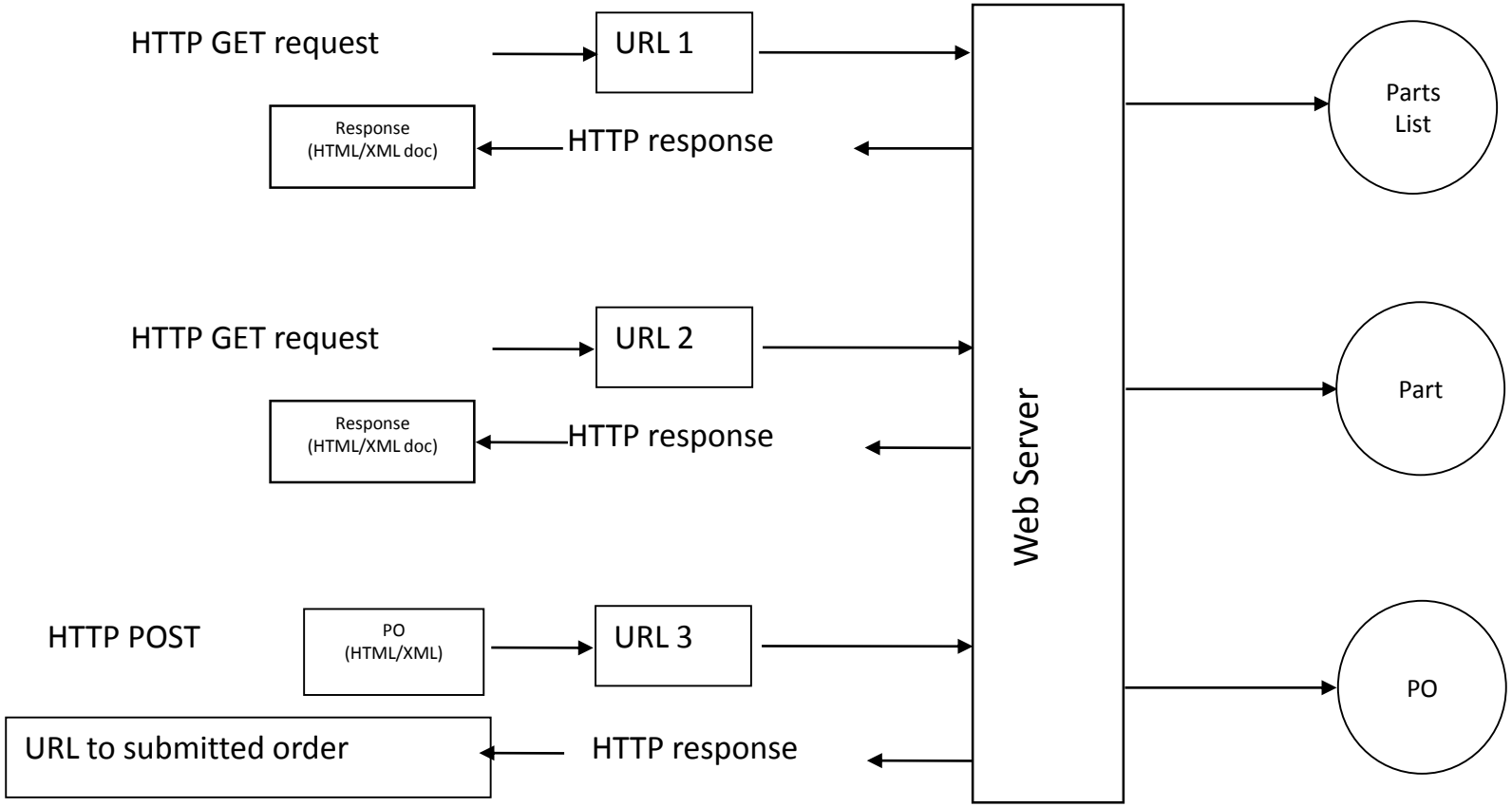
The host name identifies the server address of the resource.

The path and query string can be used to identify and customize the accessed resource.

Уеб услуги със SOAP



Уеб услуги със REST



Пример

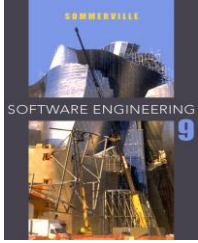
Съвсем базов

Липсват try/catch клаузи

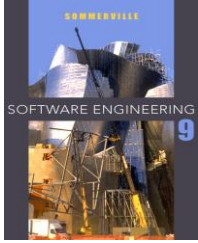
Не може да се използва в реални програми, целта е да се разбере идеята

```
public static String httpGet(String urlStr) throws IOException {
  URL url = new URL(urlStr);
  HttpURLConnection conn =
      (HttpURLConnection) url.openConnection();

  if (conn.getResponseCode() != 200) {
    throw new IOException(conn.getResponseMessage());
  }

  // Buffer the result into a string
  BufferedReader rd = new BufferedReader(
      new InputStreamReader(conn.getInputStream()));
  StringBuilder sb = new StringBuilder();
  String line;
  while ((line = rd.readLine()) != null) {
    sb.append(line);  }
  rd.close();

  conn.disconnect();
  return sb.toString();
}
```

```
public static String httpPost(String urlStr, String[] paramName,
String[] paramVal) throws Exception {
  URL url = new URL(urlStr);
  HttpURLConnection conn =
      (HttpURLConnection) url.openConnection();
  conn.setRequestMethod("POST");
  conn.setDoOutput(true);
  conn.setDoInput(true);
  conn.setUseCaches(false);
  conn.setAllowUserInteraction(false);
  conn.setRequestProperty("Content-Type",
      "application/x-www-form-urlencoded");

  // Create the form content
  OutputStream out = conn.getOutputStream();
  Writer writer = new OutputStreamWriter(out, "UTF-8");
  for (int i = 0; i < paramName.length; i++) {
    writer.write(paramName[i]);
    writer.write("=");
    writer.write(URLEncoder.encode(paramVal[i], "UTF-8"));
    writer.write("&");  }
  writer.close();
  out.close();

  if (conn.getResponseCode() != 200) {
    throw new IOException(conn.getResponseMessage());  }

  // Buffer the result into a string
  BufferedReader rd = new BufferedReader(
      new InputStreamReader(conn.getInputStream()));
  StringBuilder sb = new StringBuilder();
  String line;
  while ((line = rd.readLine()) != null) {
    sb.append(line);  }
  rd.close();

  conn.disconnect();
  return sb.toString();}
```



Software development with services

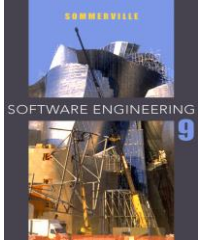


Software development with services

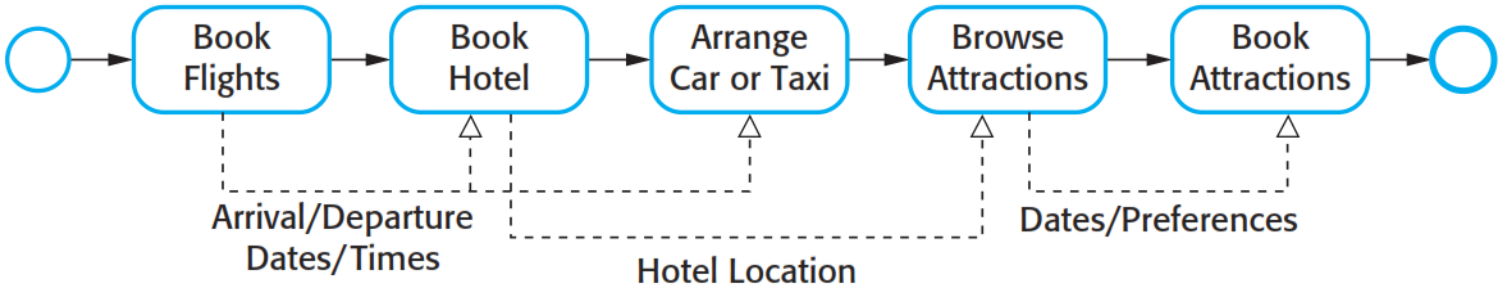
Existing services are composed and configured to create new composite services and applications

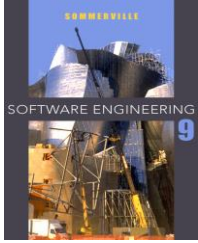
The basis for service composition is often a workflow

- Workflows are logical sequences of activities that, together, model a coherent business process
- For example, provide a travel reservation services which allows flights, car hire and hotel bookings to be coordinated

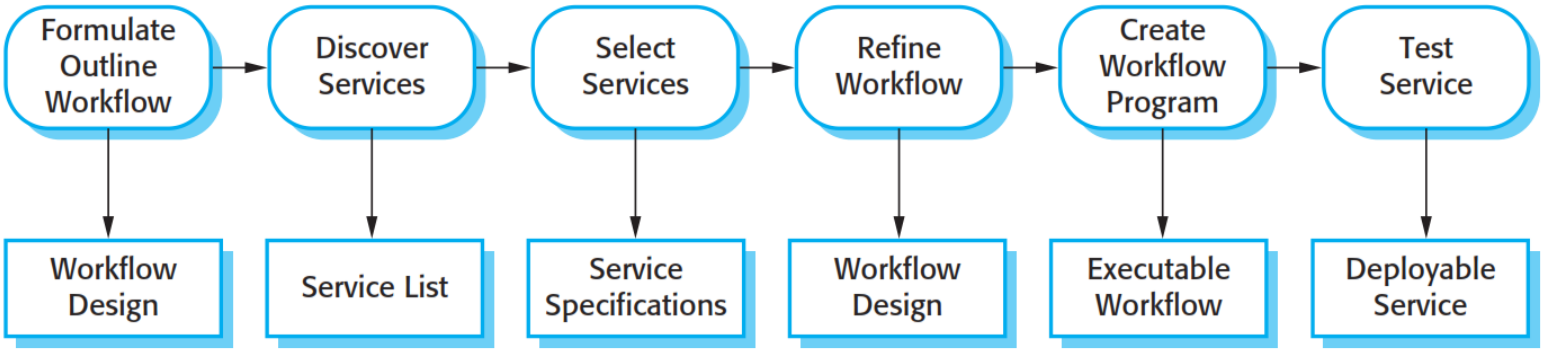


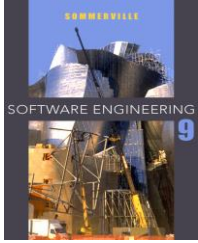
Vacation package workflow





Service construction by composition





Construction by composition

Formulate outline workflow

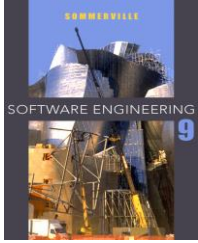
- In this initial stage of service design, you use the requirements for the composite service as a basis for creating an 'ideal' service design.

Discover services

- During this stage of the process, you search service registries or catalogs to discover what services exist, who provides these services and the details of the service provision.

Select possible services

- Your selection criteria will obviously include the functionality of the services offered. They may also include the cost of the services and the quality of service (responsiveness, availability, etc.) offered.



Construction by composition

Refine workflow.

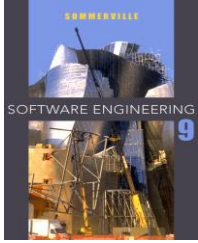
- This involves adding detail to the abstract description and perhaps adding or removing workflow activities.

Create workflow program

- During this stage, the abstract workflow design is transformed to an executable program and the service interface is defined. You can use a conventional programming language, such as Java or a workflow language, such as WS-BPEL.

Test completed service or application

- The process of testing the completed, composite service is more complex than component testing in situations where external services are used.



Workflow design and implementation

WS-BPEL is an XML-standard for workflow specification. However, WS-BPEL descriptions are long and unreadable

Graphical workflow notations, such as BPMN, are more readable and WS-BPEL can be generated from them

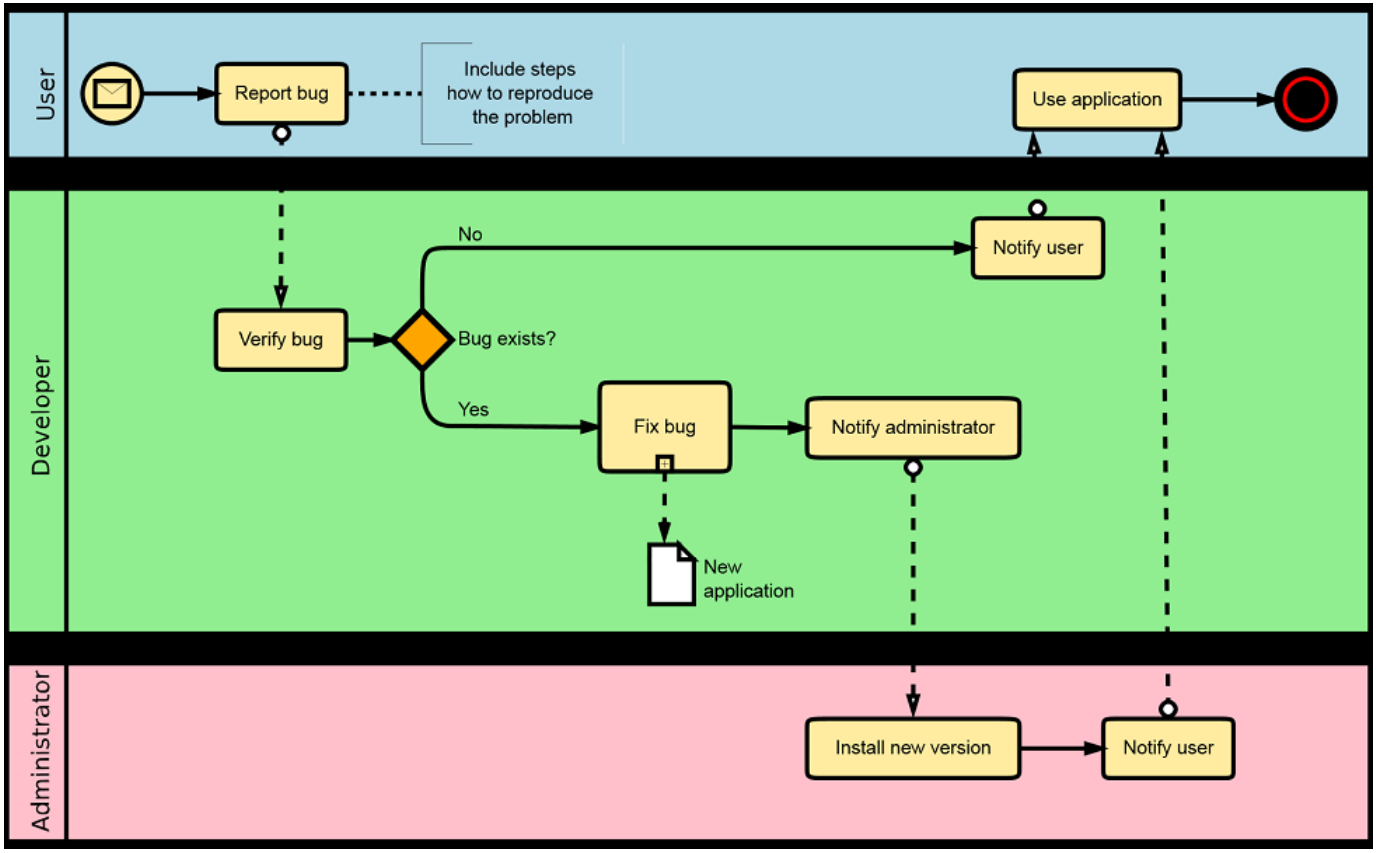
In inter-organisational systems, separate workflows are created for each organisation and linked through message exchange

Hello world BPEL example

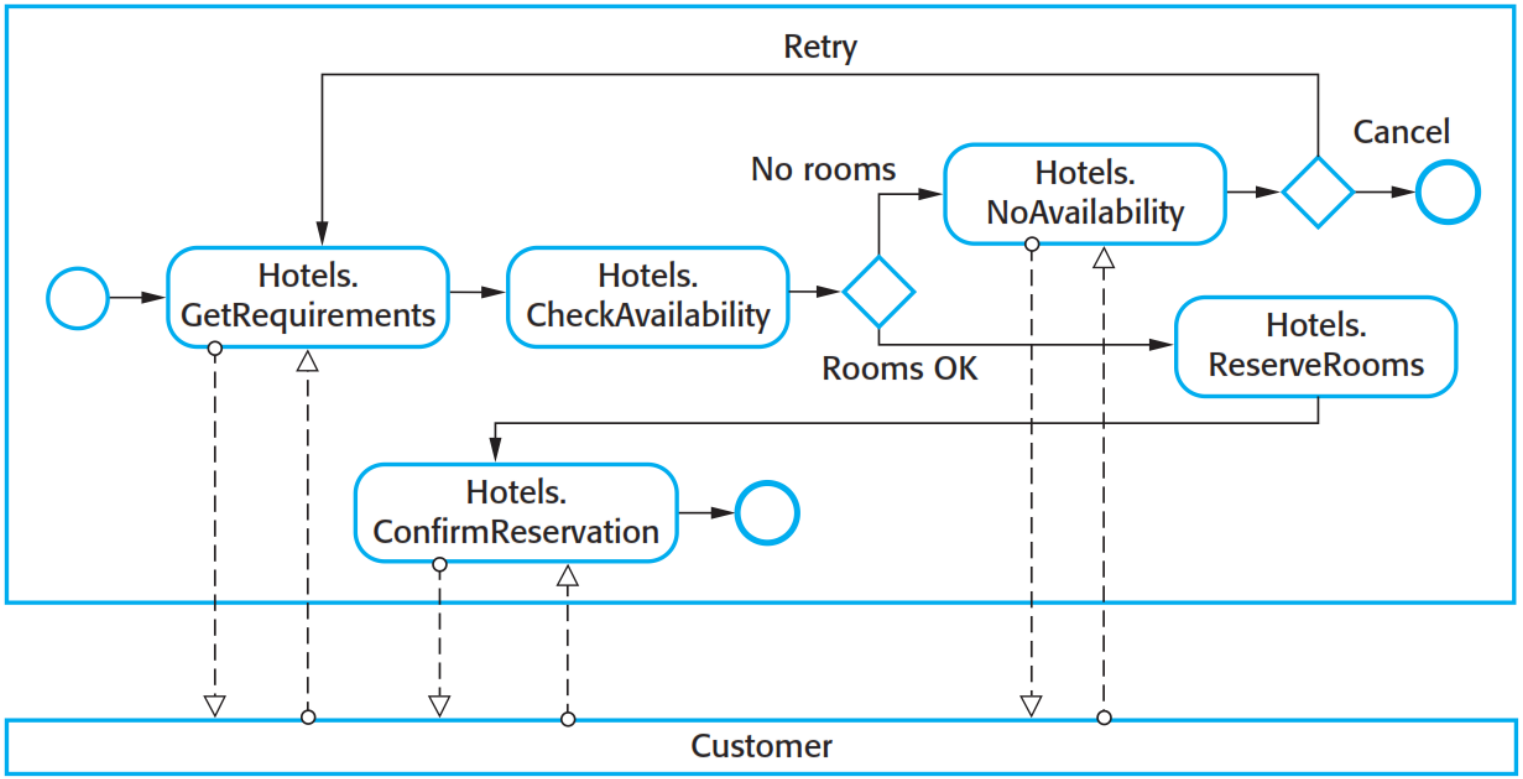
http://www.eclipse.org/tptp/platform/documents/design/choreography_html/tutorials/wsbpel_tut.html

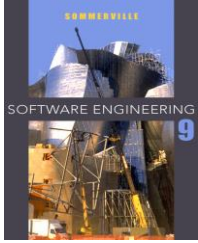
```
<?xml version="1.0" encoding="UTF-8"?>
<process
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:print="http://www.eclipse.org/tptp/choreography/2004/engine/Print">

        <!--Hello World - my first ever BPEL program -->
        <import importType="http://schemas.xmlsoap.org/wsdl/"
                location="../../test_bucket/service_libraries/tptp_EnginePrinterPort.wsdl"
                namespace="http://www.eclipse.org/tptp/choreography/2004/engine/Print" />

        <partnerLinks>
                <partnerLink          name="printService"
                                partnerLinkType="print:printLink"
                                partnerRole="printService"/>
        </partnerLinks>

        <variables>
                <variable     name="hello_world"
                                messageType="print:PrintMessage" />
        </variables>

        <assign>
                <copy>
                        <from><literal>Hello World</literal></from>
                        <to>$hello_world.value</to>
                </copy>
        </assign>

        <invoke partnerLink="printService" operation="print" inputVariable="hello_world" /> </process>
```

BPMN examples

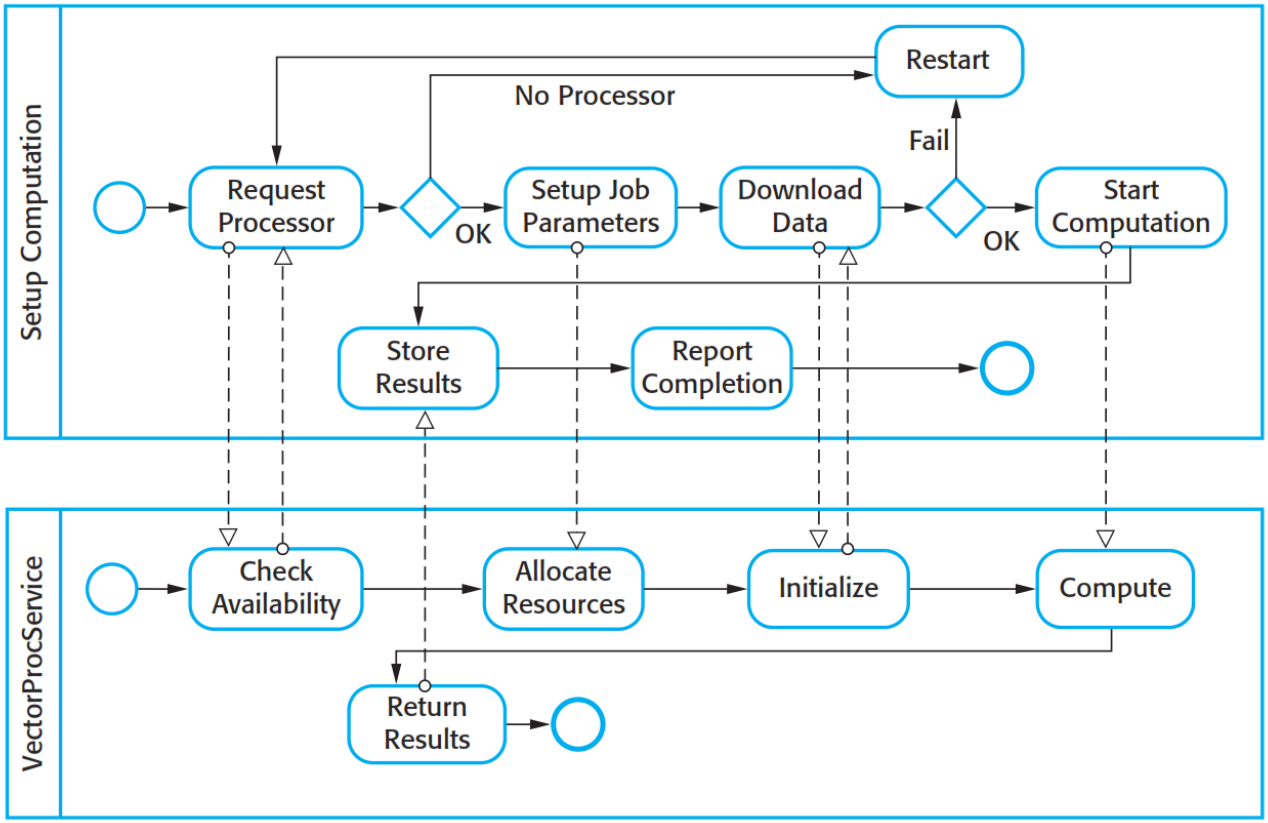


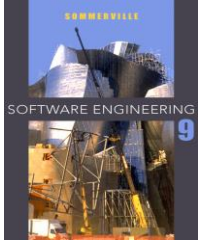
A fragment of a hotel booking workflow





Interacting workflows

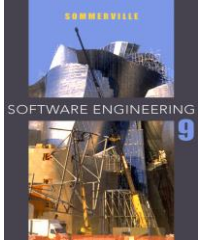




Service testing

Testing is intended to find defects and demonstrate that a system meets its functional and non-functional requirements.

Service testing is difficult as (external) services are ‘black-boxes’. Testing techniques that rely on the program source code cannot be used.



Service testing problems

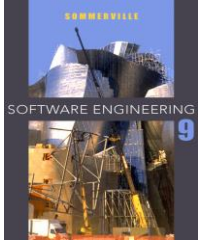
External services may be modified by the service provider thus invalidating tests which have been completed.

Dynamic binding means that the service used in an application may vary - the application tests are not, therefore, reliable.

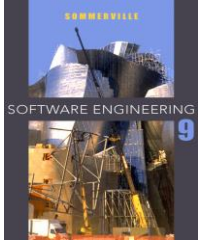
The non-functional behaviour of the service is unpredictable because it depends on load.

If services have to be paid for as used, testing a service may be expensive.

It may be difficult to invoke compensating actions in external services as these may rely on the failure of other services which cannot be simulated.



Service development



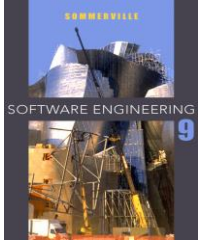
Service engineering

The process of developing services for reuse in service-oriented applications

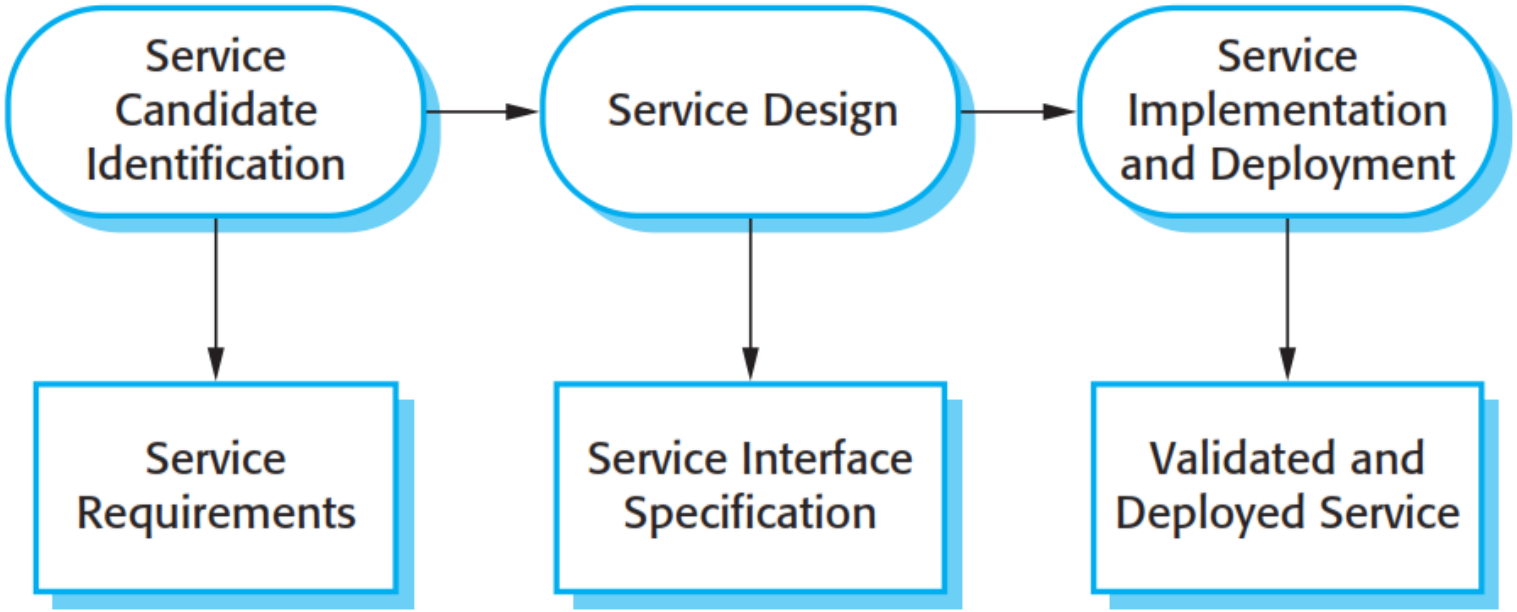
The service has to be designed as a reusable abstraction that can be used in different systems.

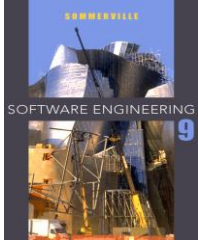
Generally useful functionality associated with that abstraction must be designed and the service must be robust and reliable.

The service must be documented so that it can be discovered and understood by potential users.



The service engineering process



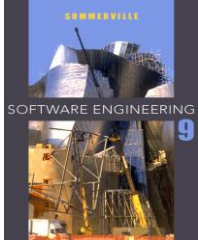


Stages of service engineering

Service candidate identification, where you identify possible services that might be implemented and define the service requirements.

Service design, where you design the logical and WSDL service interfaces.

Service implementation and deployment, where you implement and test the service and make it available for use.



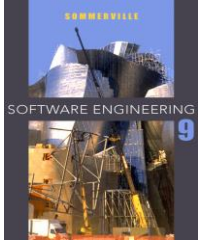
Service candidate identification

Services should support business processes.

Service candidate identification involves understanding an organization's business processes to decide which reusable services could support these processes.

Three fundamental types of service

- Utility services that implement general functionality used by different business processes.
- Business services that are associated with a specific business function e.g., in a university, student registration.
- Coordination services that support composite processes such as ordering.

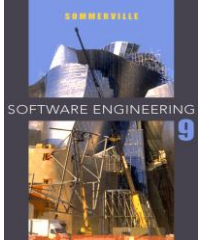


Task and entity-oriented services

Task-oriented services are those associated with some activity.

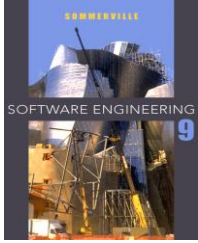
Entity-oriented services are like objects. They are associated with a business entity such as a job application form.

Utility or business services may be entity- or task-oriented, coordination services are always task-oriented.



Service classification

	Utility	Business	Coordination
Task	Currency converter Employee locator	Validate claim form Check credit rating	Process expense claim Pay external supplier
Entity	Document style checker Web form to XML converter	Expenses form Student application form	

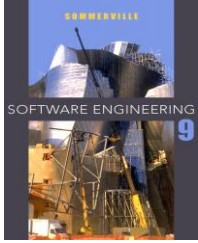


Key points

Service-oriented software engineering is based on the notion that programs can be constructed by composing independent services which encapsulate reusable functionality.

Service interfaces are defined in WSDL. A WSDL specification includes a definition of the interface types and operations, the binding protocol used by the service and the service location.

Services may be classified as utility services, business services or coordination services.



Service identification

Is the service associated with a single logical entity used in different business processes?

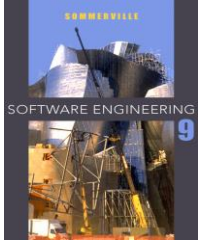
Is the task one that is carried out by different people in the organisation?

Is the service independent?

Does the service have to maintain state? Is a database required?

Could the service be used by clients outside the organisation?

Are different users of the service likely to have different non-functional requirements?

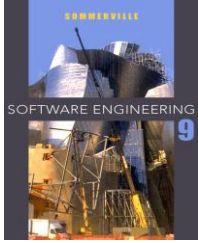


Service interface design

Involves thinking about the operations associated with the service and the messages exchanged

The number of messages exchanged to complete a service request should normally be minimised.

Service state information may have to be included in messages



Interface design stages

Logical interface design

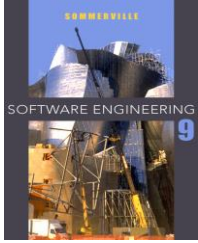
- Starts with the service requirements and defines the operation names and parameters associated with the service. Exceptions should also be defined

Message design

- Design the structure and organisation of the input and output messages. Notations such as the UML are a more abstract representation than XML

WSDL description

- The logical specification is converted to a WSDL description

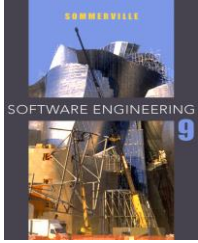


Service implementation and deployment

Programming services using a standard programming language or a workflow language

Services then have to be tested by creating input messages and checking that the output messages produced are as expected

Deployment involves publicising the service and installing it on a web server. Current servers provide support for service installation



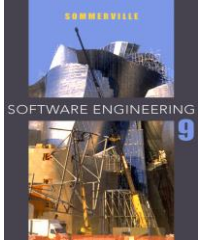
Service descriptions

Information about your business, contact details, etc. This is important for trust reasons. Users of a service have to be confident that it will not behave maliciously.

An informal description of the functionality provided by the service. This helps potential users to decide if the service is what they want.

A detailed description of the interface types and semantics.

Subscription information that allows users to register for information about updates to the service.



Legacy system services

An important application of services is to provide access to functionality embedded in legacy systems

Legacy systems offer extensive functionality and this can reduce the cost of service implementation

External applications can access this functionality through the service interfaces

Cloud computing

What is Cloud Computing?

Cloud Computing is a general term used to describe a new class of network based computing that takes place over the Internet,

- a collection/group of integrated and networked hardware, software and Internet infrastructure (called a platform).
- Using the Internet for communication and transport provides hardware, software and networking services to clients

These platforms hide the complexity and details of the underlying infrastructure from users and applications by providing very simple graphical interface or API (Applications Programming Interface).

What is Cloud Computing?

In addition, the platform provides on demand services, that are always on, anywhere, anytime and any place.

Pay for use and as needed, elastic

- scale up and down in capacity and functionalities

The hardware and software services are available to

- general public, enterprises, corporations and businesses markets

Cloud Computing – Simple Definition

Cloud Computing = Software as a Service
+ Platform as a Service
+ Infrastructure as a Service
+ Data as a Service

Cloud Computing – Simple Definition

Cloud Computing = Software as a Service
+ Platform as a Service

Software as a Service (SaaS)

- From end user's point of view
- Apps are located in the cloud
- Software experiences are delivered through the Internet

Cloud Computing – Simple Definition

Cloud Computing = Software as a Service
+ Platform as a Service

Platform as a Service (PaaS)

- From developer's point of view (i.e. cloud users)
- Cloud providers offer an Internet-based platform to developers who want to create services but don't want to build their own cloud

Cloud Computing – Simple Definition

Cloud Computing = Software as a Service
+ Platform as a Service
Infrastructure as a Service

Infrastructure as a Service (IaaS)

- Cloud providers build datacenters
  - Power, scale, hardware, networking, storage, distributed systems, etc
- Datacenter as a service
- Cloud users rent storage, computation, and maintenance from cloud providers (pay-as-you-go; like utility)

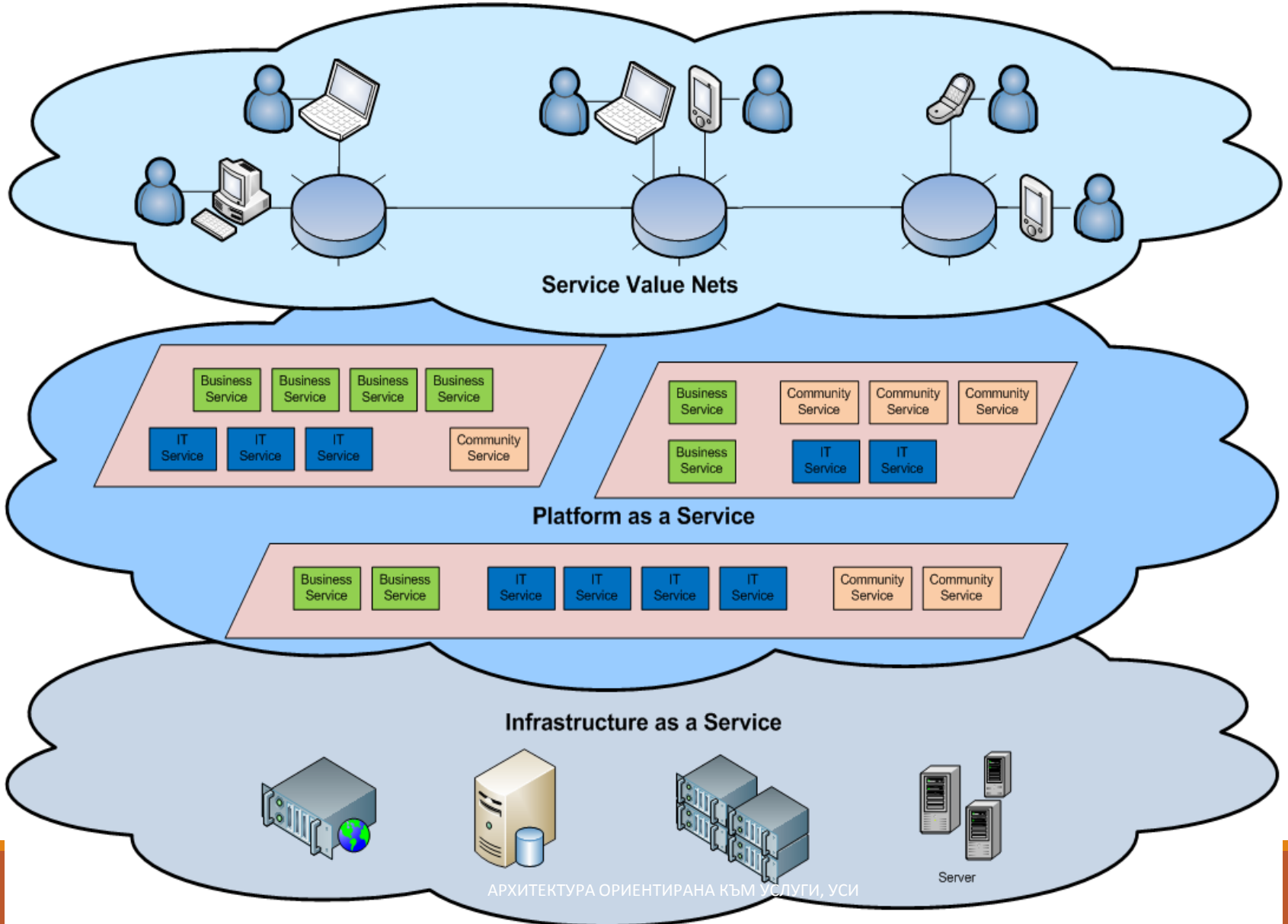
Cloud Summary

Cloud computing is an umbrella term used to refer to Internet based development and services

A number of characteristics define cloud data, applications services and infrastructure:

- Remotely hosted: Services or data are hosted on remote infrastructure.
- Ubiquitous: Services or data are available from anywhere.
- Commodified: The result is a utility computing model similar to traditional that of traditional utilities, like gas and electricity - you pay for what you would want!

Cloud Architecture



Cloud Computing Characteristics

Common Characteristics:

- Massive Scale
- Resilient Computing
- Homogeneity
- Geographic Distribution
- Virtualization
- Service Orientation
- Low Cost Software
- Advanced Security

Essential Characteristics:

- On Demand Self-Service
- Broad Network Access
- Rapid Elasticity
- Resource Pooling
- Measured Service

Basic Cloud Characteristics

The “**no-need-to-know**” in terms of the underlying details of infrastructure, applications interface with the infrastructure via the APIs.

The “**flexibility and elasticity**” allows these systems to scale up and down at will

- utilising the resources of all kinds
  - CPU, storage, server capacity, load balancing, and databases

The “**pay as much as used and needed**” type of utility computing and the “**always on!, anywhere and any place**” type of network-based computing.

Basic Cloud Characteristics

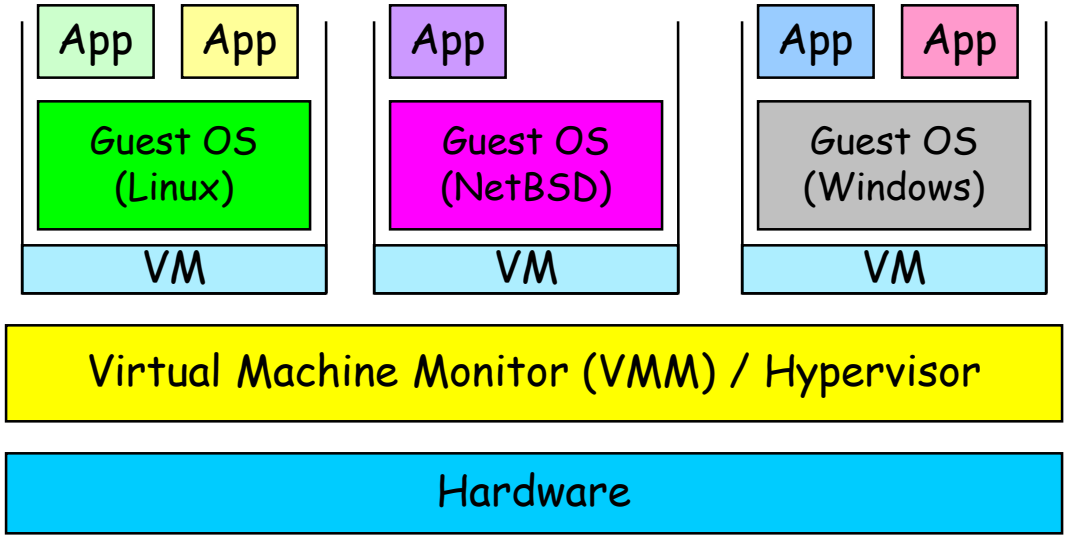
Cloud are transparent to users and applications, they can be built in multiple ways

- branded products, proprietary open source, hardware or software, or just off-the-shelf PCs.

In general, they are built on clusters of PC servers and off-the-shelf components plus Open Source software combined with in-house applications and/or system software.

Virtual Machines

VM technology allows multiple virtual machines to run on a single physical machine.



Performance: Para-virtualization (e.g. Xen) is very close to raw physical performance!