CS2002 Software Development and Management

Service-Oriented Architecture

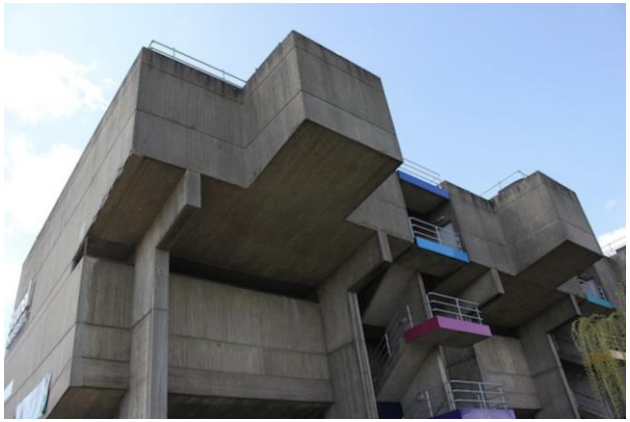
Rumyana Neykova
rumyana.neykova@brunel.ac.uk

Architectural style

Architecture vs Architectural style

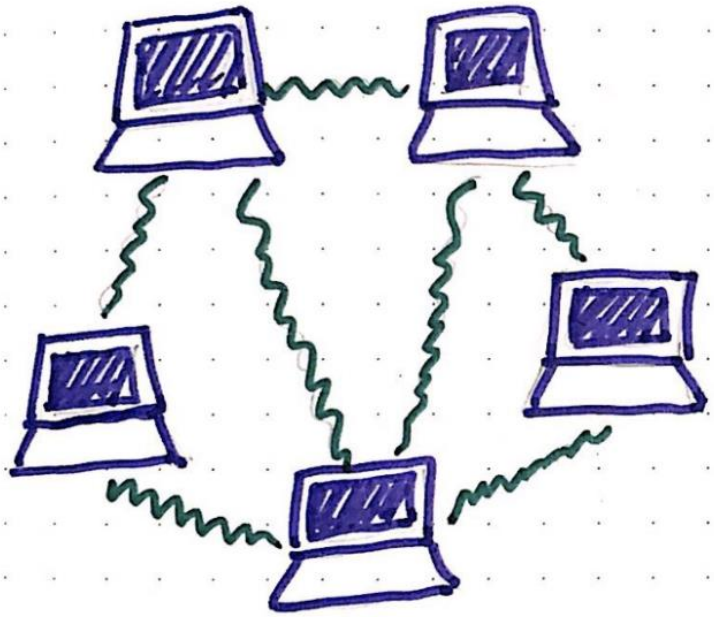
- Architectural Style: General principles informing the creation of an architecture
- Architecture: Designing a solution to a problem according to given constraints
- Architectural styles inform and guide the creation of architectures

- Architecture: Louvre [http://en.wikipedia.org/wiki/Louvre]
- Architectural Style: Baroque [http://en.wikipedia.org /wiki/Baroque_architecture]



?

What is a distributed system?



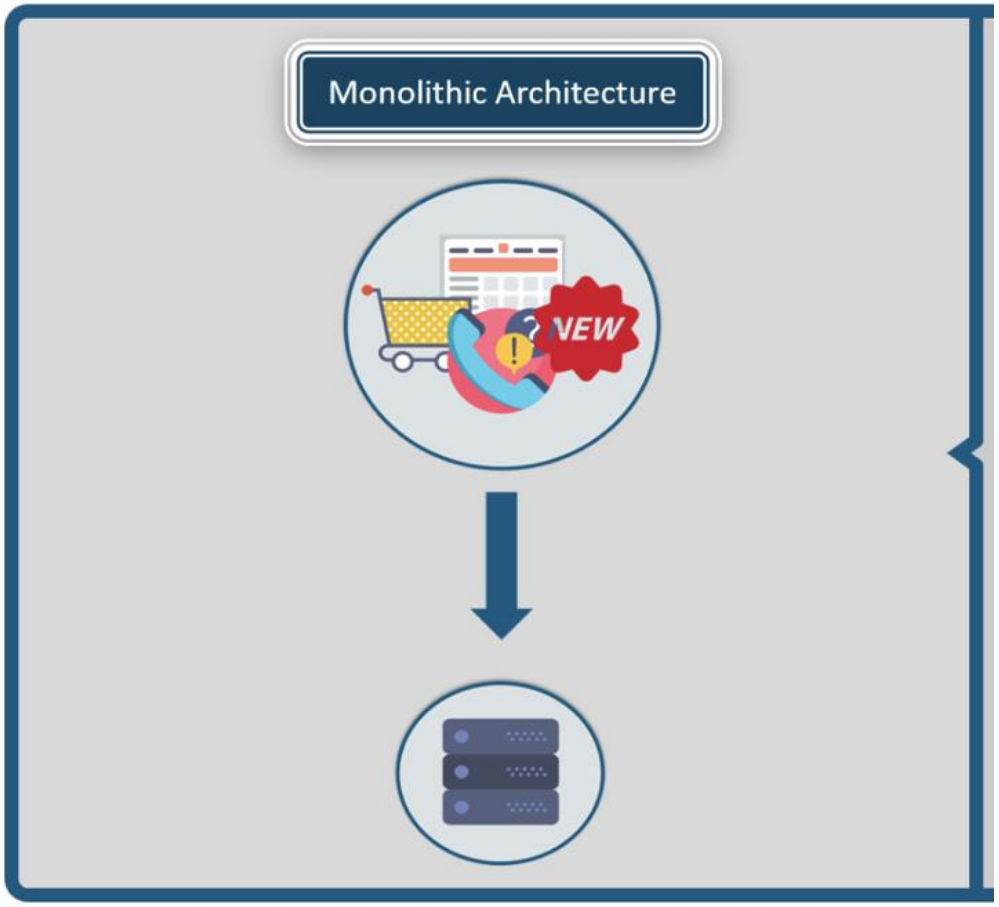
A **distributed system** involves multiple entities talking to one another in some way, while also performing their own operations.

 A brief introduction to distributed systems, Maarten van Steen & Andrew S. Tanenbaum

SOA on one slide

SOA is a way of developing distributed systems by combining stand-alone, reusable, loosely-coupled services

How did we use to write software



Many components but….

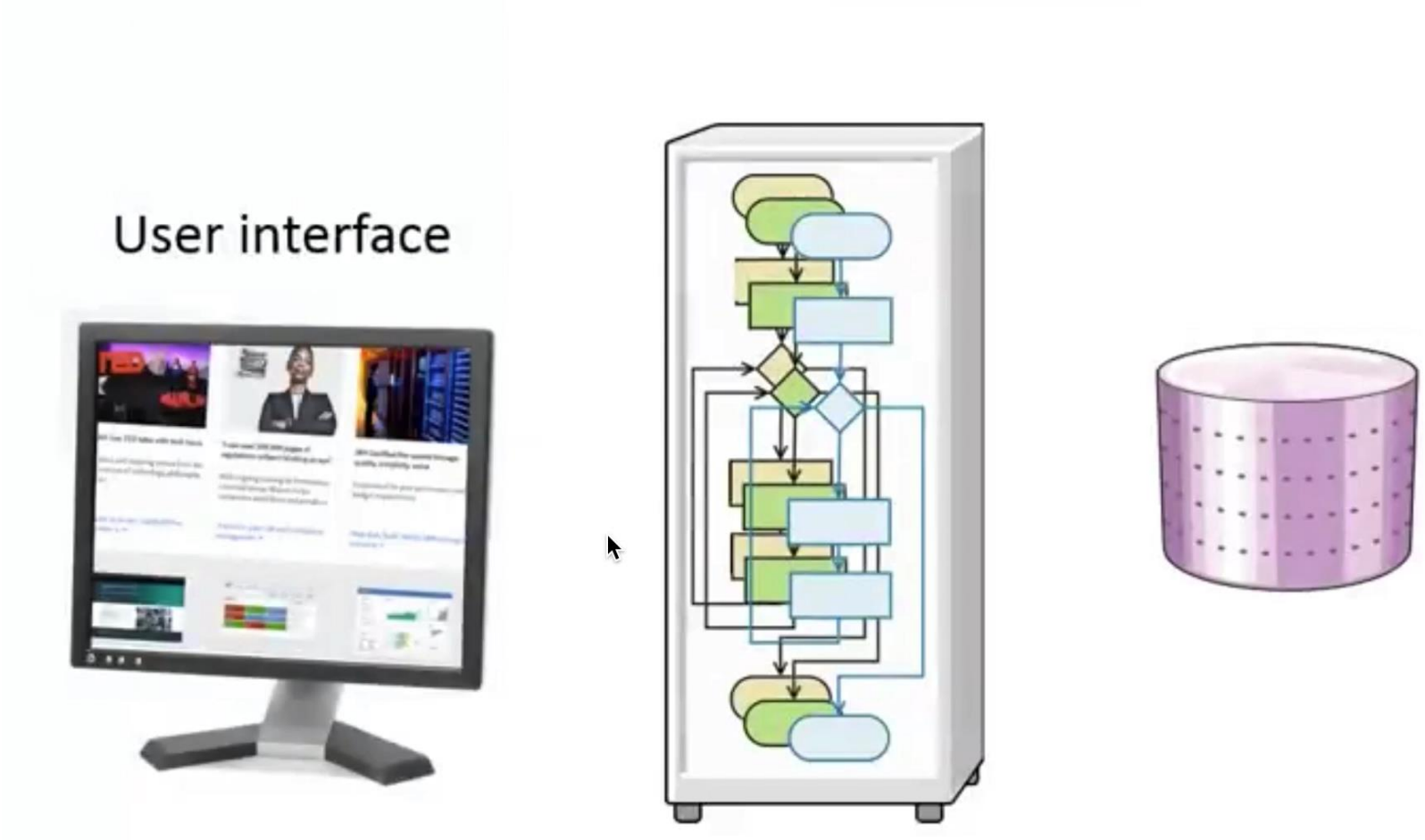
One database

A single deployment unit

A single language/technology

Do you see any problems with that?

From monolithic to SOA



https://www.coursera.org/learn/intro-ibm-microservices/lecture/r7uaa/characteristics-of-a-microservices-application

What is a service?

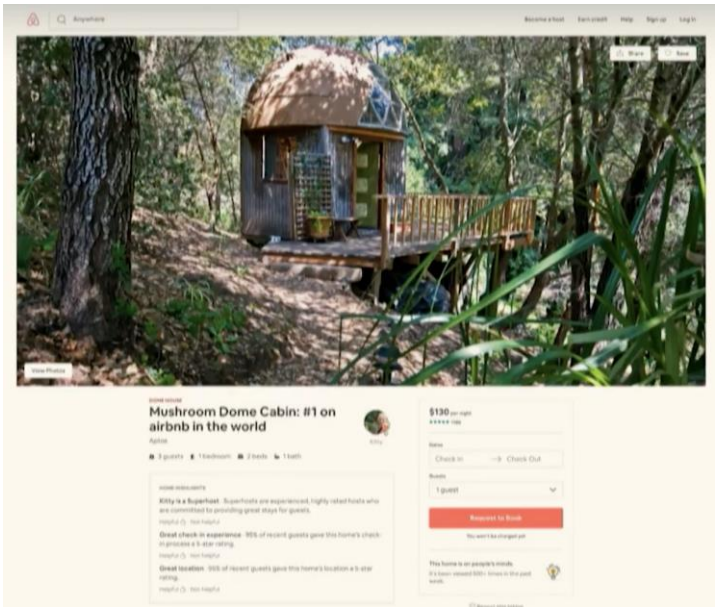
SOA centers around the concept of **decomposing business problems into services.**

Service = Business Capability

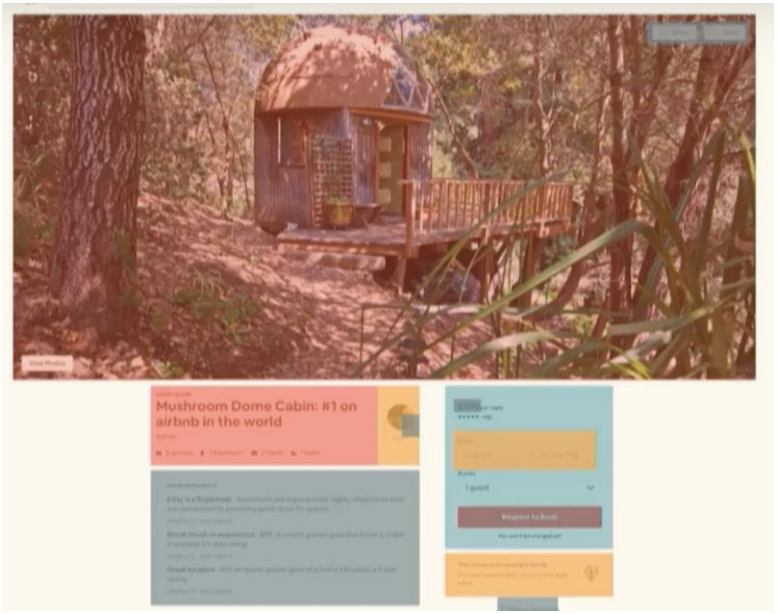
Examples


amazon.com
Spotify
UBER
NETFLIX
FT
FINANCIAL TIMES
GOV.UK
ebay
SOUNDCLOUD

AirBnB Example: From Monolith to SOA



16 teams needed to coordinate as to update this single page



Split into three services (products):

- image rendering
- price calculation
- host information

https://www.youtube.com/watch?v=_dFJOSR-aFs

The secret to Amazon success

amazon.com

Jeff Bezos's Mandate
2002

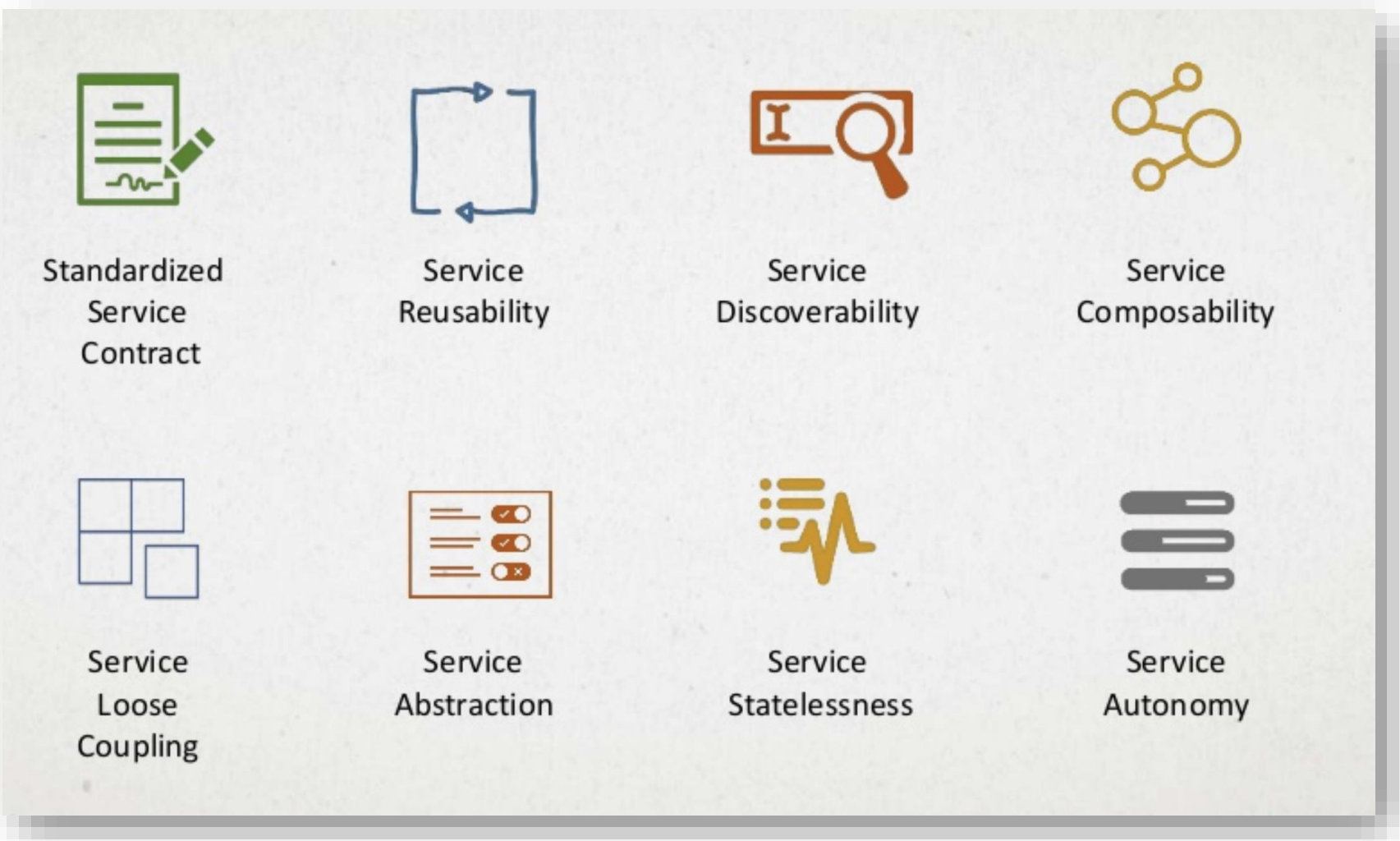
1. All teams will henceforth **expose their data and functionality through service interfaces**
2. Teams must **communicate with each other through these interfaces**
3. There will be **no other form of interprocess communication allowed**: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network
4. It **doesn't matter what technology you** use. HTTP, Corba, Pubsub, custom protocols -- doesn't matter
5. All service interfaces, without exception, must be designed from the ground up to be **externalizable**. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions

The mandate closed with:

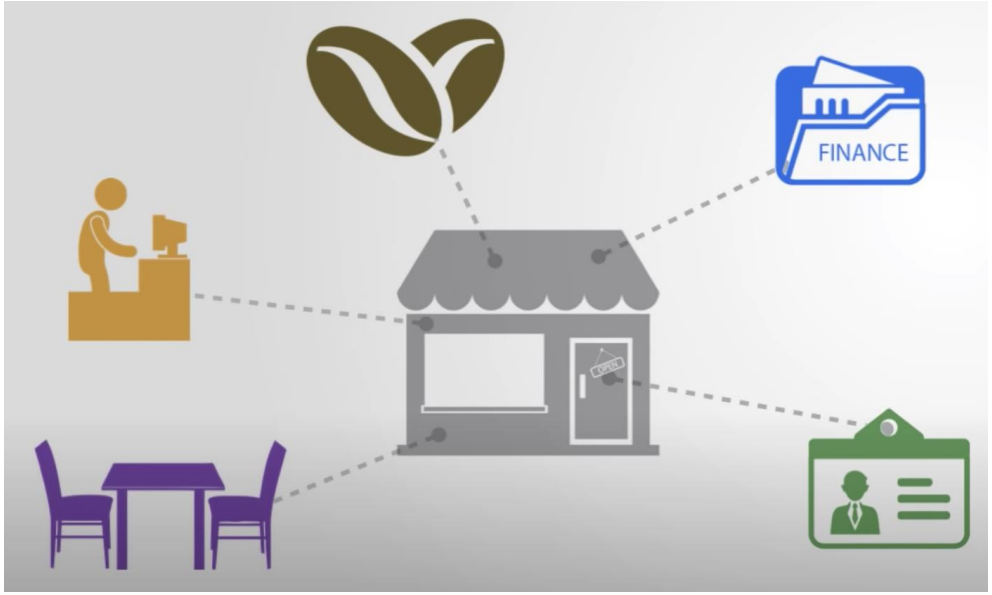
> ***Anyone who doesn't do this will be fired. Thank you; have a nice day!***

Jeff Bezos's Mandate (from Steve Yegge's Google platform rant) https://plus.google.com/+RipRowan/posts/eVeouesvaVX

Principles of SOA

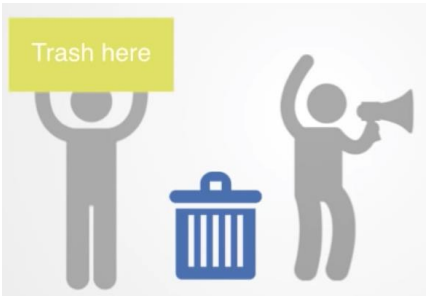


Let’s build a coffee shop



https://www.youtube.com/watch?v=_dFJOSR-aFs&t=5s

Let’s build a coffee shop



https://www.youtube.com/watch?v=_dFJOSR-aFs&t=5s

Principles of SOA

Standardized Service Contract
Service in same inventory are in compliance of same design service contract standards.

Service Reusability
Service contain agnostic logic that can be position as reusable enterprise resource.

Service Discoverability
Service meta data available for discoverability and interpreted.

Service Composition
Services are effective composition participants.

Service Loose Coupling
Contract decoupled from surrounding environment.

Service Abstraction
Contract contains only essential information , that is published to consumers.

Service Statelessness
Services minimize resource consumption , reduce state information.

Service Autonomy
Services exercise a high level of control over their underlying runtime execution environment.

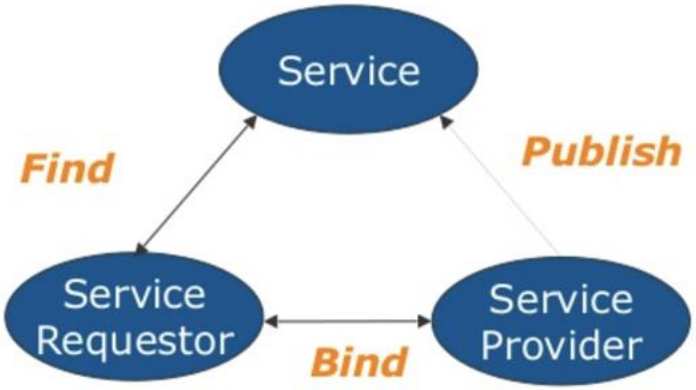
http://serviceorientation.com/static/pdf/SOA_Principles_Poster.pdf

Service-oriented architecture

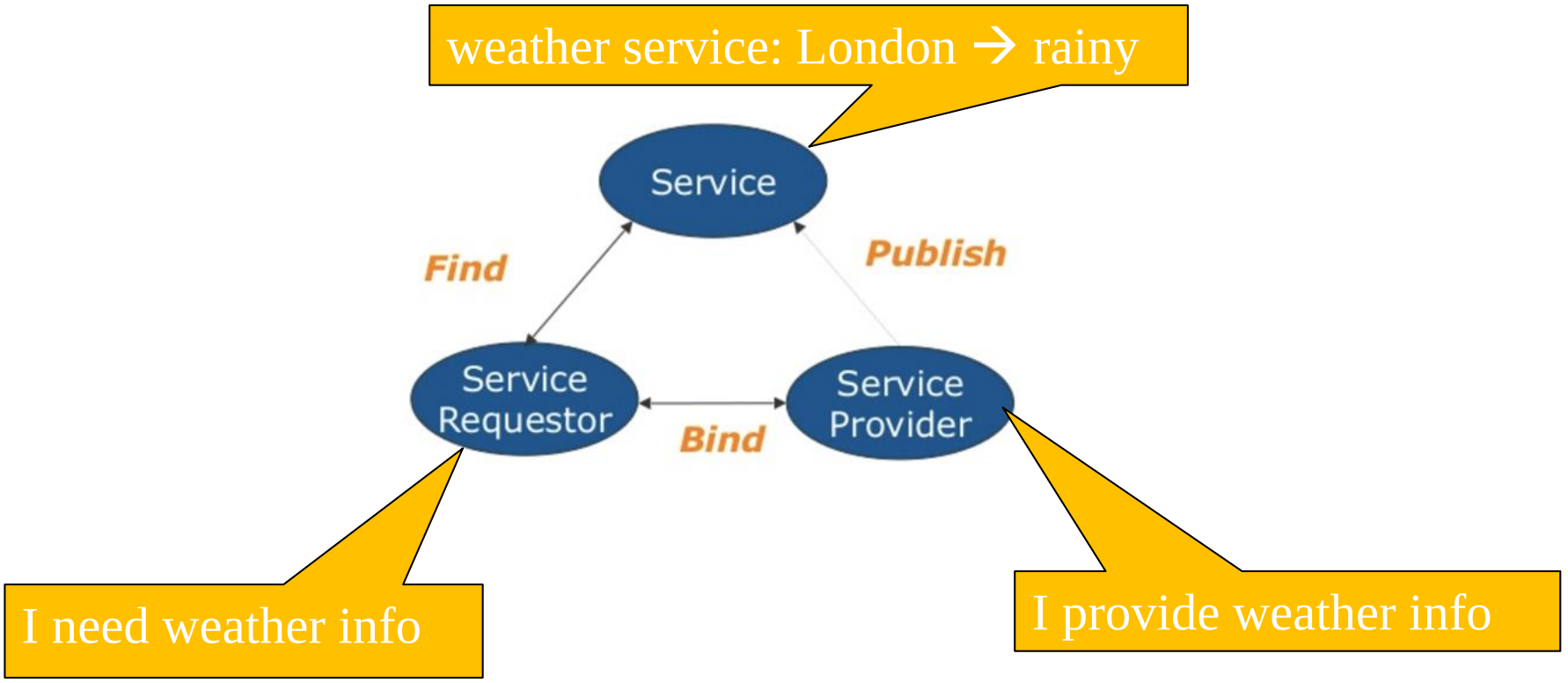
What?

Building blocks of SOA

SOA defines an architecture which usually consists of the following roles and the contracts between those roles



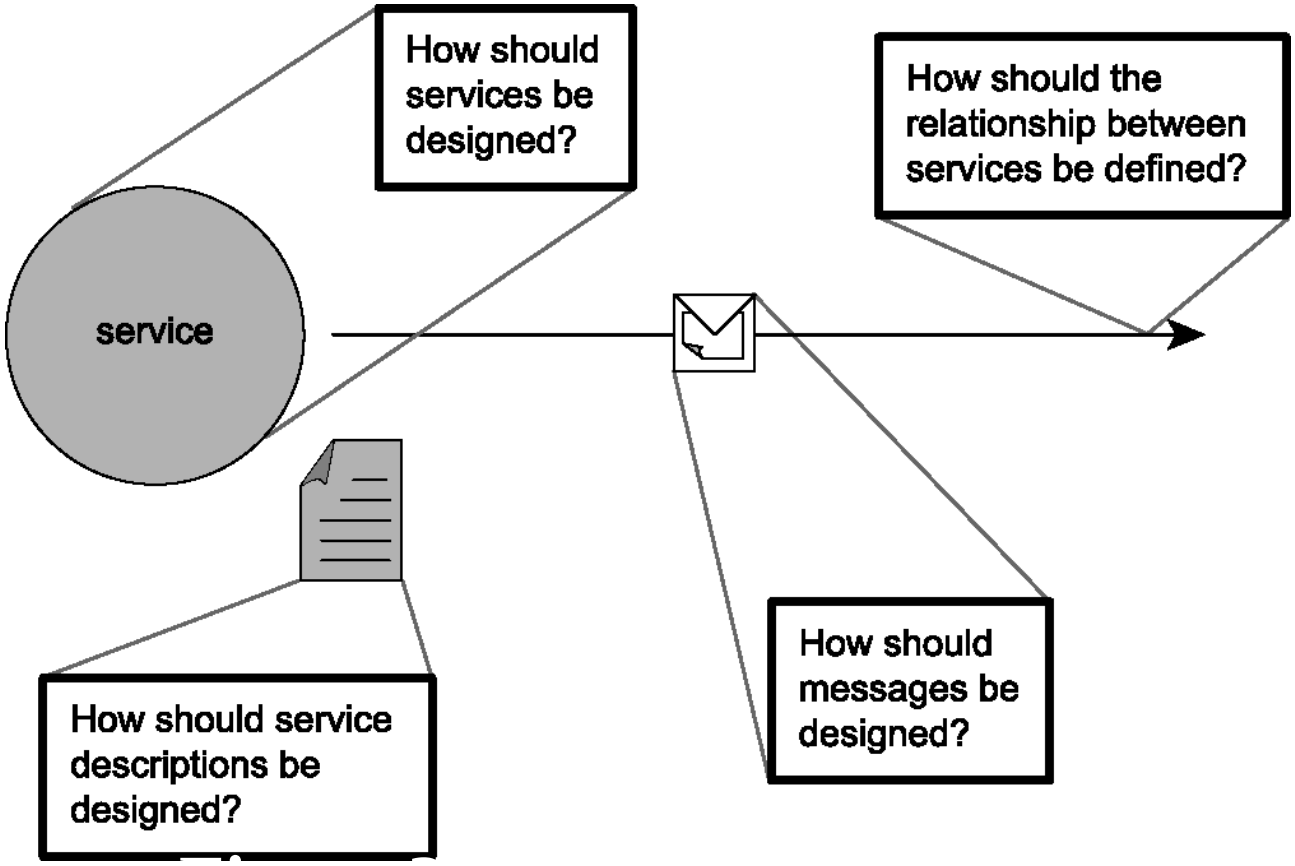
Example: A Weather Application



Service-oriented architecture

How?

Service Design: main issues



Building services

SOA

can be realised through a

variety of technologies

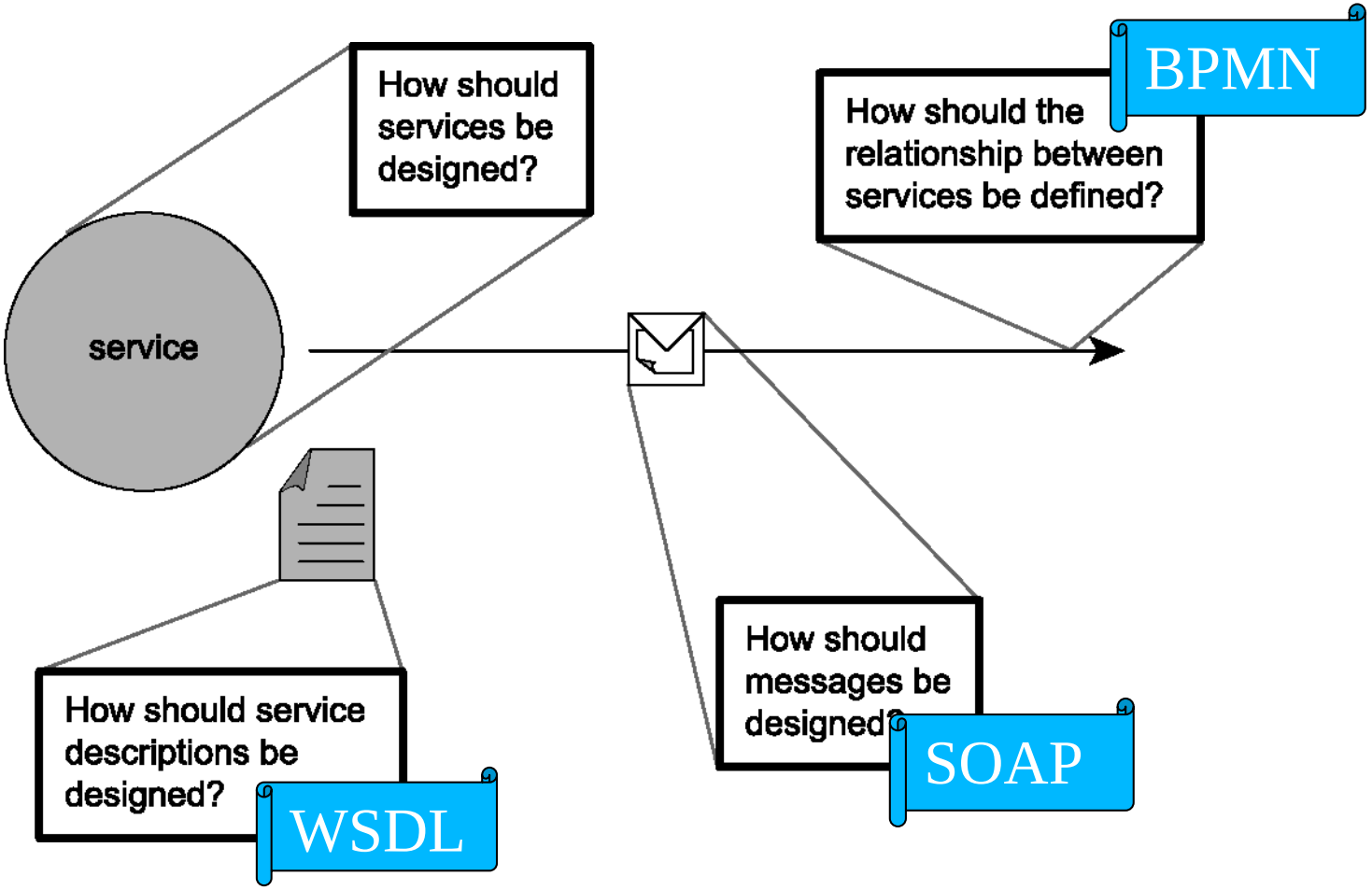
Evolution of SOA

1. SOAP-based web services
2. RESTful web services
3. Microservices

Types of Web Services

- *SOAP/WSDL based*
  - *Based on international standards*
  - *Service interfaces is exposed through WSDL documents*
  - *Message exchange using SOAP*
  - *Client code may be generated from WSDL description*
- *Representative State Transfer (REST)*
  - *Everything is a resource*
  - *Resources are identified by URIs and their state is manipulated through HTTP operations GET, POST, PUT, DELETE*
  - *Rather a set of architectural principals, than a standard*

The triad of WS-services



SOAP example

(http://www.w3.org/TR/2007/REC-soap12-part1-20070427/)

WSDL example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions targetNamespace="http://endpoint.myws/" name="MyWebServiceService"
   xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://endpoint.myws/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
   schemas.xmlsoap.org/wsdl/soap/">
  <types/>

  <message name="addIntRequest">
    <part name="n1" type="xsd:int"/>
    <part name="n2" type="xsd:int"/>
  </message>
  <message name="addIntResponse">
    <part name="sum" type="xsd:int"/>
  </message>
  <message name="greetRequest">
    <part name="arg0 " type="xsd:string"/>
  </message>
  <message name="greetResponse">
    <part name="return" type="xsd:string"/>
  </message>
```

```
public class MyWebService {

      public String greet( String name ) {
             return "Hello " + name + "!";
      }

      public int addInt(int n1, int n2 ) {
             return n1+n2;
      }
}
```

SOAP-based services: drawbacks

inefficient

Overly general

Heavyweight

2. RESTful Services

REST to the rescue!

The Enabler

- Roy Fielding' 2000 dissertation is what gave birth to the world of APIs as we know it.
- It laid the foundation for the protocol which is u… of APIs today
- https://www.ics…/~fielding/pubs/dissertation/fielding_dissertation.pdf

xignite

Clip slide

CHAPTER 5

Representational State Transfer (REST)

This chapter introduces and elaborates the Represe… architectural style for distributed hyper… principles guiding REST… …inciples, while … …styles. REST is a hybrid … architectural styles described in Chapter 3 … constraints that define a uniform connector interface. The …itecture framework of Chapter 1 is used to define the architectural elements of REST and examine sample process, connector, and data views of prototypical architectures.

5.1 Deriving REST

The design rationale behind the Web architecture can be described by an architectural style consisting of the set of constraints applied to elements within the architecture. By examining the impact of each constraint as it is added to the evolving style, we can identify the properties induced by the Web's constraints. Additional constraints can then be applied to form a new architectural style that better reflects the desired properties of a modern Web architecture. This section provides a general overview of REST by walking through the process of deriving it as an architectural style. Later sections will describe in more detail the specific constraints that compose the REST style.

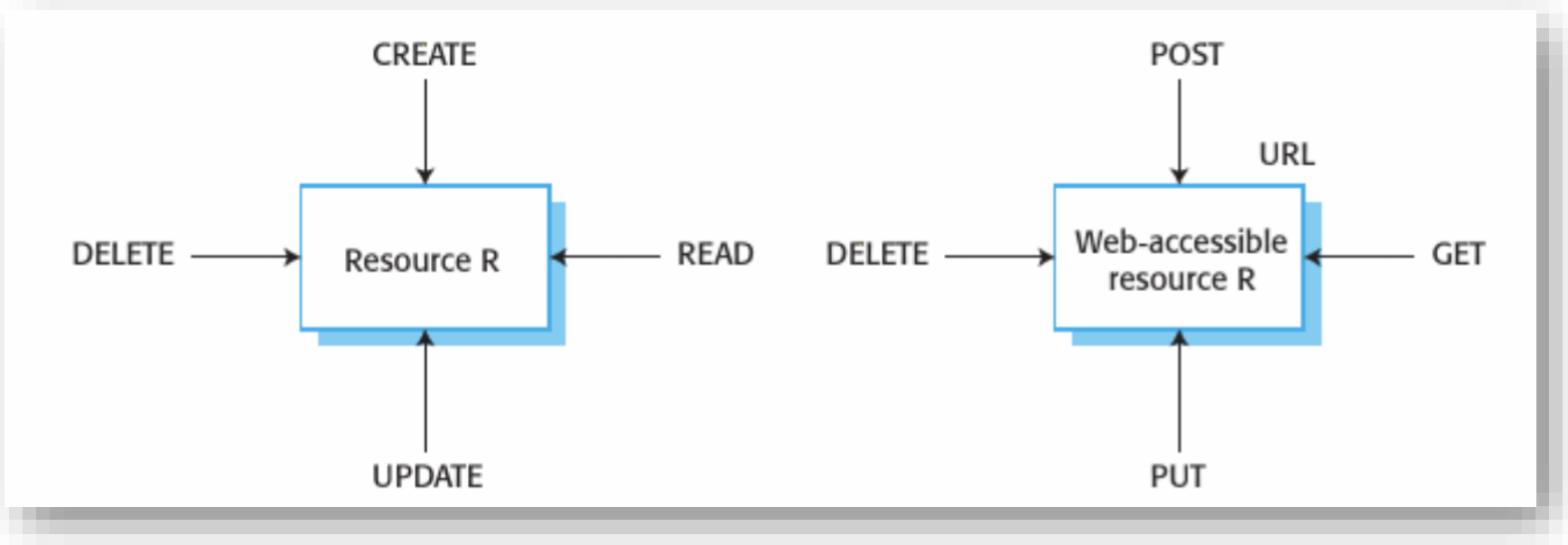
REST is an architectural style!

Example: The World Wide Web

The difference between a web service and a website is who accesses it

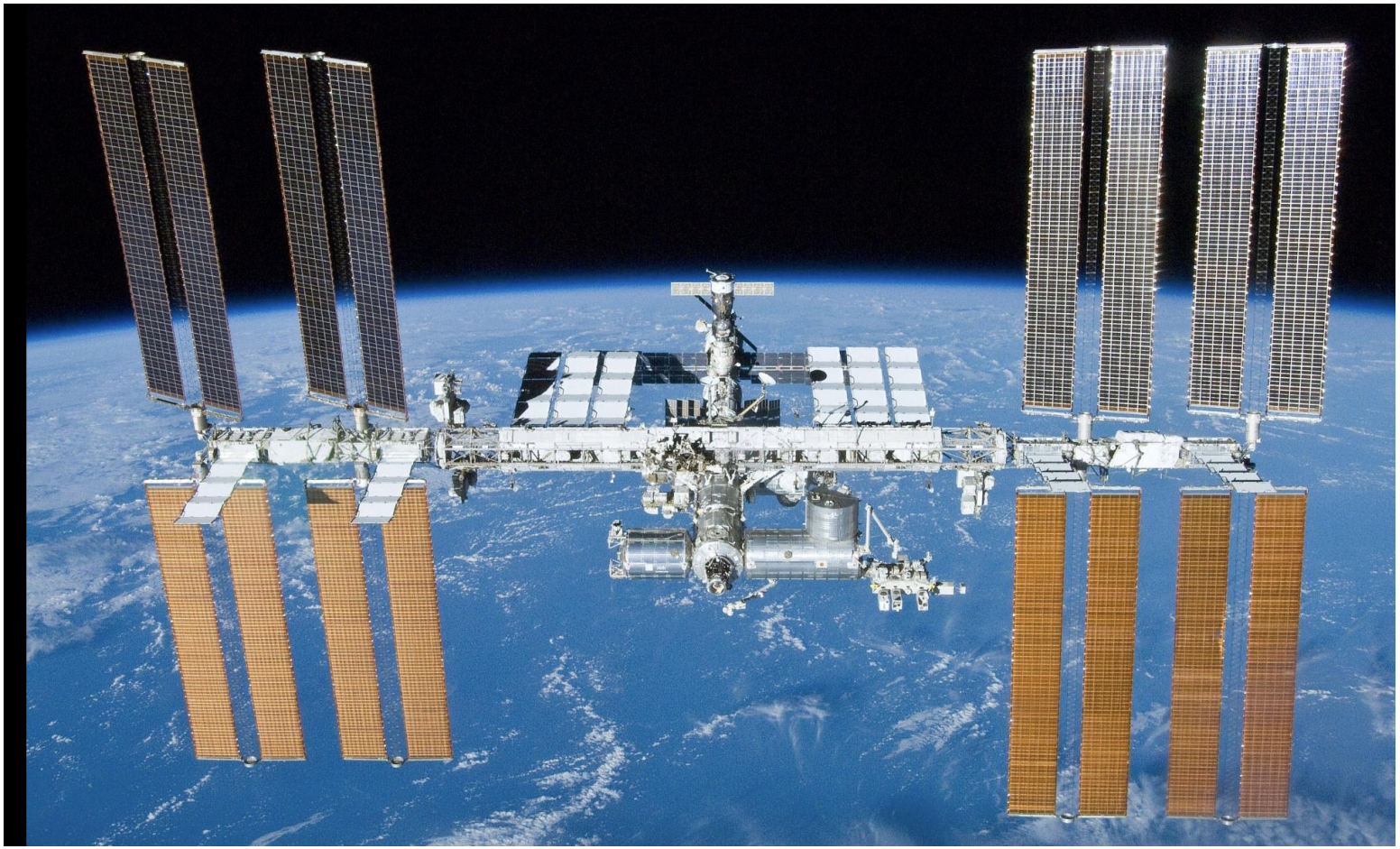
The latter is accessed by humans,
The former by programmed clients

Everything is a resource



Resource are manipulated over HTTP

In the lab:
Restful services by example:



- http://open-notify.org/Open-Notify-API/ISS-Location-Now/

SOAP vs REST

- **Independece** – *SOAP is language, and transport independent, while REST must use HTTP*
- **Efficiency** – *REST can use smaller messaging formats, while SOAP requires XML for all of its messages*
- **Environment** – *SOAP works well in a distributed environment, while REST assumes a direct point-to-point collaboration*
- **Dynamic** – *REST is easier to implement and learn to use, SOAP requires more bandwidth and resources to work efficiently*

Evolution of SOA

1. SOAP-based web services
2. RESTful web services
3. Microservices

3. Microservices

Microservices: evolution of SOA

Designed for Failure

- Designed for scale, if a service fails, the system should stay alive.
- Gave rise to chaos engineering – Netflix has a service that runs at production and kills services at randomly to test that the system can stay alive.

Deplopyment & Containerization

- Independent deployability over reuse.
- Microservices are often placed within a virtual container system such as Docker.

Automation

- Again, fast deployment, depends on infrastructure automation, DevOps, continuous integration

Monolith

How to improve?

SOA\Microservice architecture to the rescue

Summary

- *Not a single technology or architecture is a silver bullet*
- *Designing a system is always a trade off, simplicity vs scale vs availability vs agility vs …*
- *SOA is NOT a collection of standards and is not a single technology*
- *SOA is an abstract pattern, a collection of principles and practices.*
- *Web Service is* an implementation of SOA
- *Building blocks of web service* are service description (WSDL), service registry (UDDI), and service message format (SOAP)
- *REST is the architectural style of the web*
- *In REST* everything is a resource, the http verbs are used to manipulate the resource
- *Microservices are an evolution of SOA, an architectural style that promotes independence, fast deployment, fault-tolerance*

References

- Erl, T. (2007) Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall. ISBN 0131858580
- Chapters 18 of Sommerville
- Check the many resources on the slides
- Online curl command:
  - https://reqbin.com/curl
- A list of APIs:
  - https://www.programmableweb.com/

Next session …

Software Testing

Bonus slides

SOA Manifesto

Service orientation is a paradigm that frames what you do.
Service-oriented architecture (SOA) is a type of architecture
that results from applying service orientation.

We have been applying service orientation to help organizations
consistently deliver sustainable business value, with increased agility
and cost effectiveness, in line with changing business needs.

Through our work we have come to prioritize:

Business value over technical strategy

Strategic goals over project-specific benefits

Intrinsic interoperability over custom integration

Shared services over specific-purpose implementations

Flexibility over optimization

Evolutionary refinement over pursuit of initial perfection

That is, while we value the items on the right, we value the items on the left more.

---

http://www.soa-manifesto.org/

Bonus slides

Guiding Principles

We follow these principles:

Respect the social and power structure of the organization.

Recognize that SOA ultimately demands change on many levels.

The scope of SOA adoption can vary. Keep efforts manageable and within meaningful boundaries.

Products and standards alone will neither give you SOA nor apply the service orientation paradigm for you.

SOA can be realized through a variety of technologies and standards.

Establish a uniform set of enterprise standards and policies based on industry, de facto, and community standards.

Pursue uniformity on the outside while allowing diversity on the inside.

Identify services through collaboration with business and technology stakeholders.

Maximize service usage by considering the current and future scope of utilization.

Verify that services satisfy business requirements and goals.

Evolve services and their organization in response to real use.

Separate the different aspects of a system that change at different rates.

Reduce implicit dependencies and publish all external dependencies to increase robustness and reduce the impact of change.

At every level of abstraction, organize each service around a cohesive and manageable unit of functionality.

http://www.soa-manifesto.org/