Using the Web
as a distributed computing platform

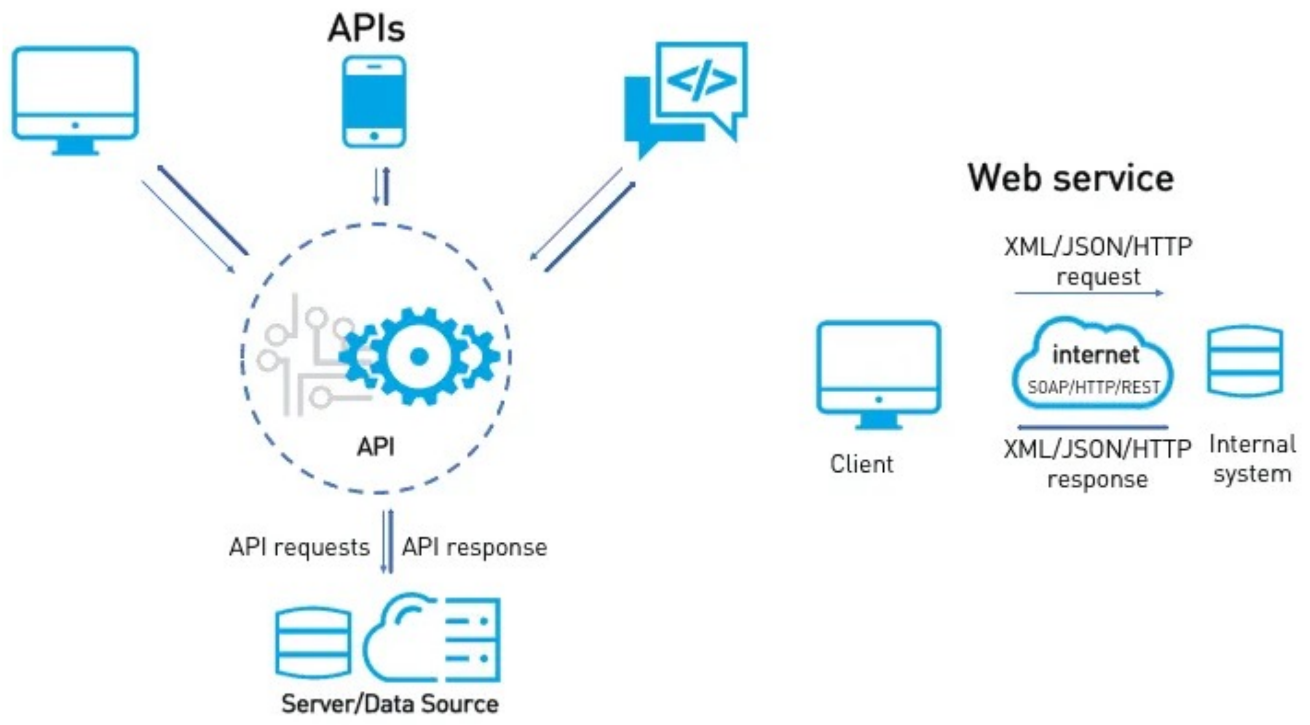
Distributed software systems
CdLM Informatica - Università di Bologna

Agenda

- Mashups
- Web Services
- Coordination of web services
- Applications of web services
    - Case study: Google Maps

Question

Can we exploit the Web as a middleware for building distributed applications?



https://rapidapi.com/blog/api-vs-web-service/

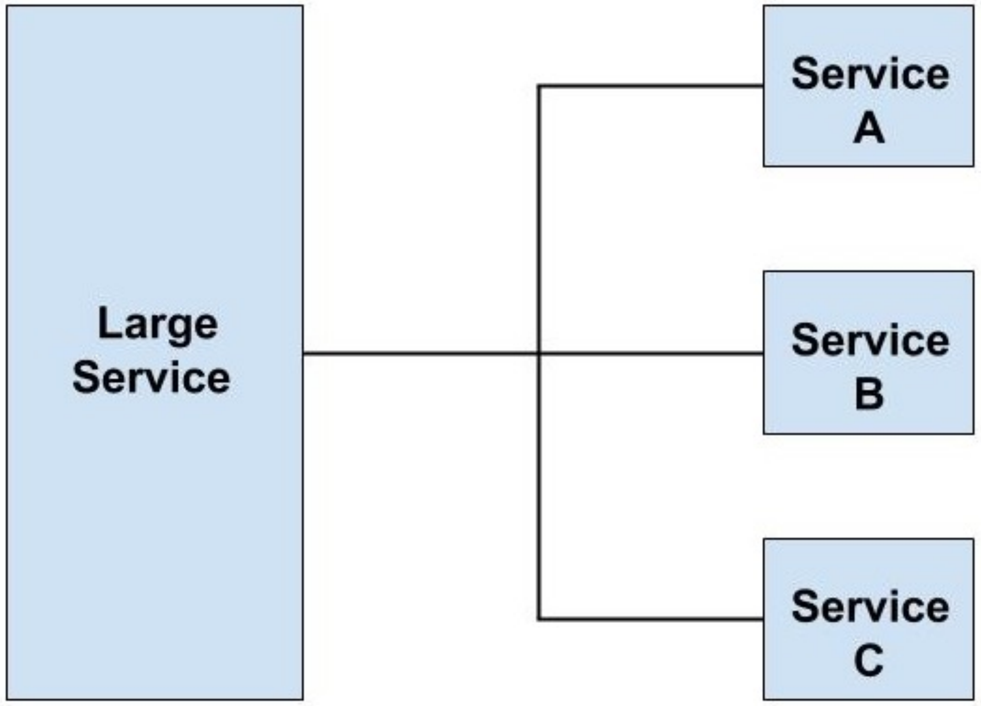
(software) Service

- In the contexts of software, the term service refers to a function that clients can reuse for different purposes, together with the policies that should control its usage (based on the identity of the client requesting the service, for example).
- OASIS defines a service as "a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description"

https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

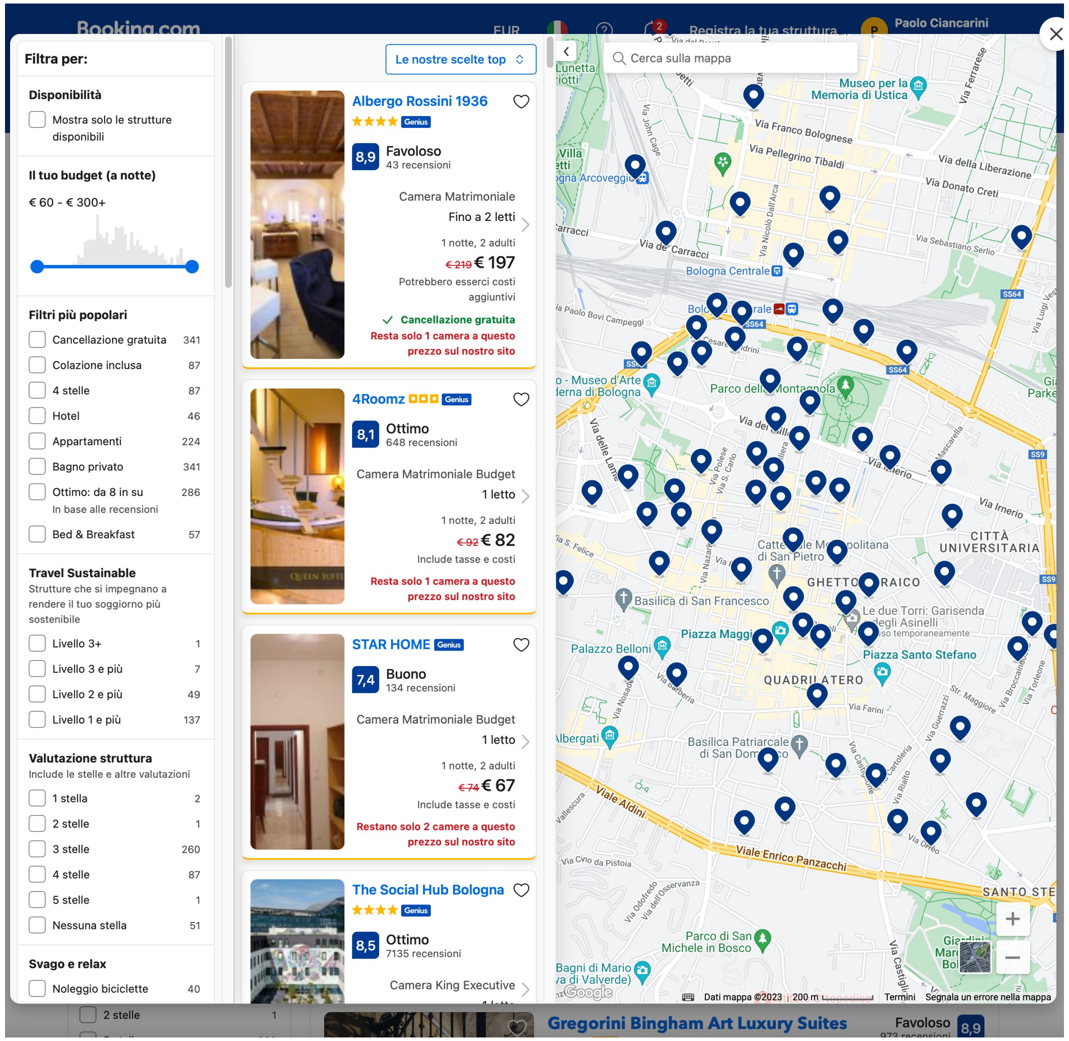
Service composition

- It is common to develop new applications by taking existing components and glueing them together
- This is the case for many Web-based applications, in particular those known as Web Services, a W3C standard
- Web-based middleware can help by standardizing the way services are accessed and providing the means to coordinate their functions in a specific order



Service composition: mashups

- An example of how service composition is deployed are mashups: Web pages or applications that aggregate data from different sources.
- Well-known mashups are those based on Google maps, in which maps are enhanced with extra information such as trip planners or real-time weather forecasts.



Filtra per:

Disponibilità
- Mostra solo le strutture disponibili

Il tuo budget (a notte)
€ 60 - € 300+

Filtri più popolari

Cancellazione gratuita	341
Colazione inclusa	87
4 stelle	87
Hotel	46
Appartamenti	224
Bagno privato	341
Ottimo: da 8 in su (In base alle recensioni)	286
Bed & Breakfast	57

Travel Sustainable
Strutture che si impegnano a rendere il tuo soggiorno più sostenibile

Livello 3+	1
Livello 3 e più	7
Livello 2 e più	49
Livello 1 e più	137

Valutazione struttura
Include le stelle e altre valutazioni

1 stella	2
2 stelle	1
3 stelle	260
4 stelle	87
5 stelle	1
Nessuna stella	51

Svago e relax

Noleggio biciclette	40

Albergo Rossini 1936 ★★★★ Genius
8,9 Favoloso — 43 recensioni
Camera Matrimoniale
Fino a 2 letti
1 notte, 2 adulti
~~€ 219~~ € 197
Potrebbero esserci costi aggiuntivi
✓ Cancellazione gratuita
Resta solo 1 camera a questo prezzo sul nostro sito

4Roomz Genius
8,1 Ottimo — 648 recensioni
Camera Matrimoniale Budget
1 letto
1 notte, 2 adulti
~~€ 92~~ € 82
Include tasse e costi
Resta solo 1 camera a questo prezzo sul nostro sito

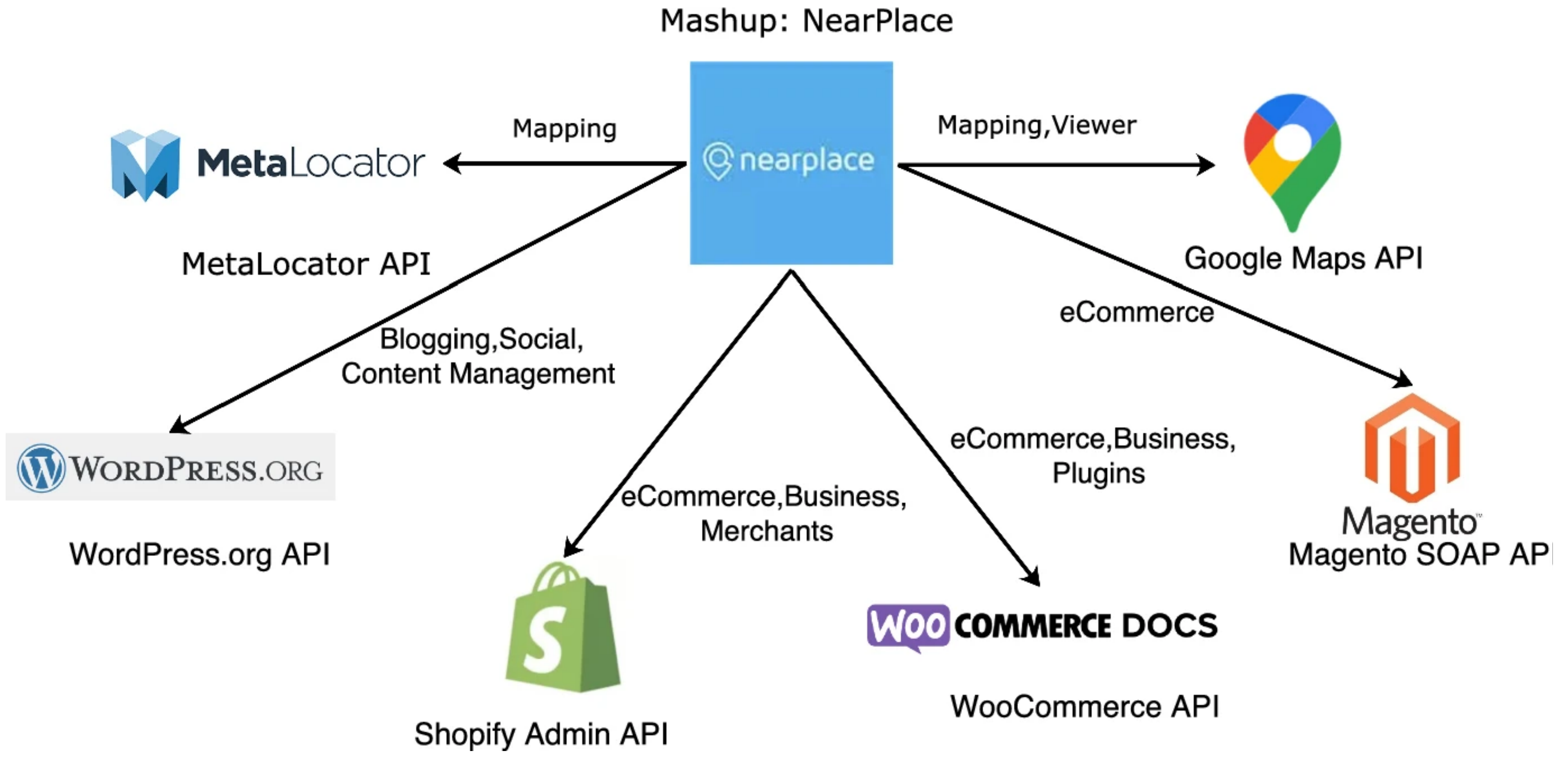
STAR HOME Genius
7,4 Buono — 134 recensioni
Camera Matrimoniale Budget
1 letto
1 notte, 2 adulti
~~€ 74~~ € 67
Include tasse e costi
Restano solo 2 camere a questo prezzo sul nostro sito

The Social Hub Bologna ★★★★ Genius
8,5 Ottimo — 7135 recensioni
Camera King Executive

Gregorini Bingham Art Luxury Suites — Favoloso 8,9

booking.com

Example: nearplace



https://datantify.com

(Web) Mashup

A mashup is a web application that uses content from more than one source to create a single new service displayed in a single interface

For example, a user could combine the addresses and photographs of their library branches with a Google map to create a map mashup.

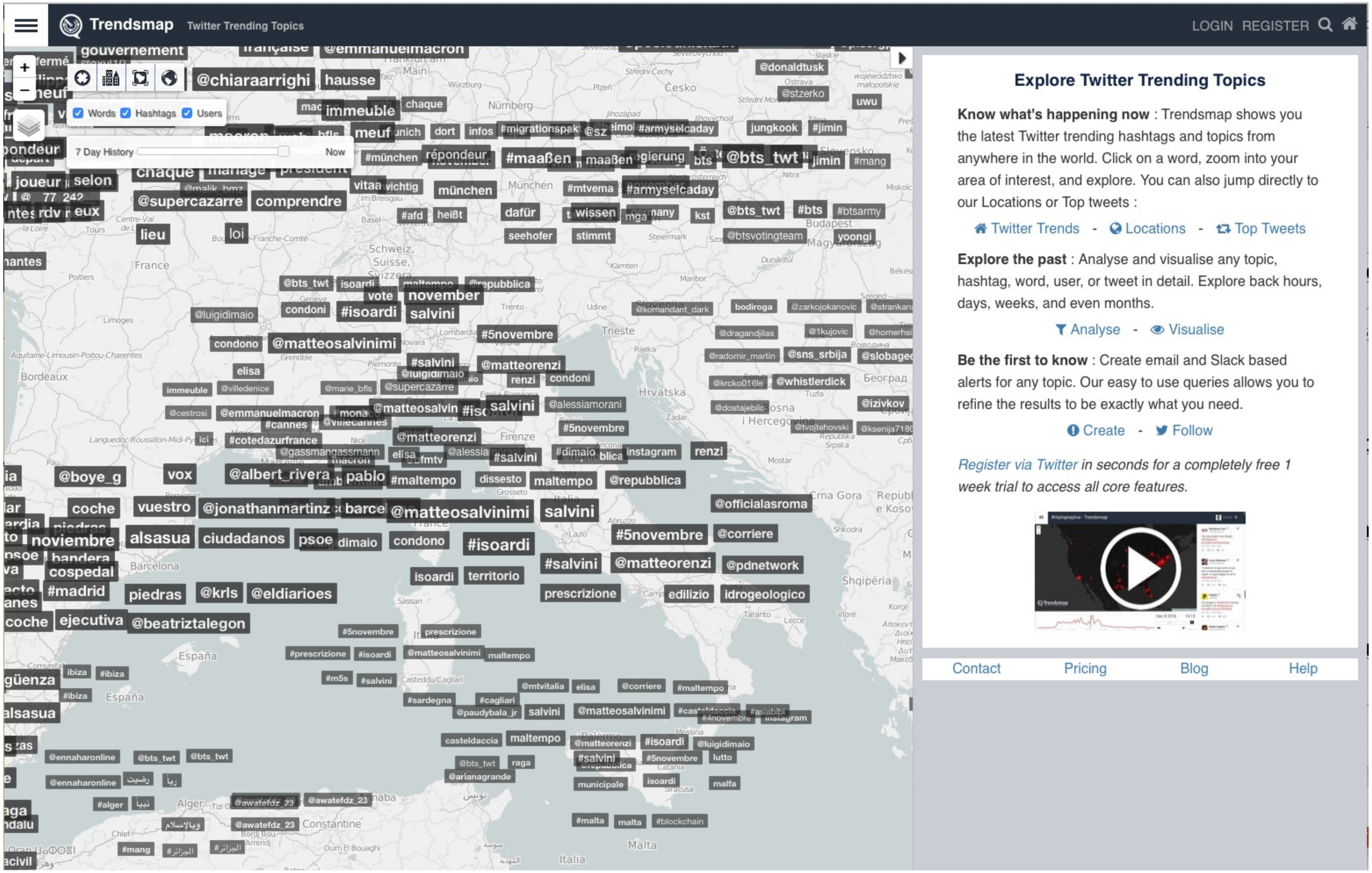
The term implies easy, fast integration, frequently using open application programming interfaces (open API) and data sources to produce enriched results that were not necessarily the original reason for producing the raw source data.

The main drivers of a mashup are data aggregation and visualization

A mashup makes data more useful, for personal or professional use.

To be able to permanently access the data of other services, mashups are generally client applications or hosted online.

Mashup example: trendsmap



Explore Twitter Trending Topics

Know what's happening now : Trendsmap shows you the latest Twitter trending hashtags and topics from anywhere in the world. Click on a word, zoom into your area of interest, and explore. You can also jump directly to our Locations or Top tweets :

Twitter Trends - Locations - Top Tweets

Explore the past : Analyse and visualise any topic, hashtag, word, user, or tweet in detail. Explore back hours, days, weeks, and even months.

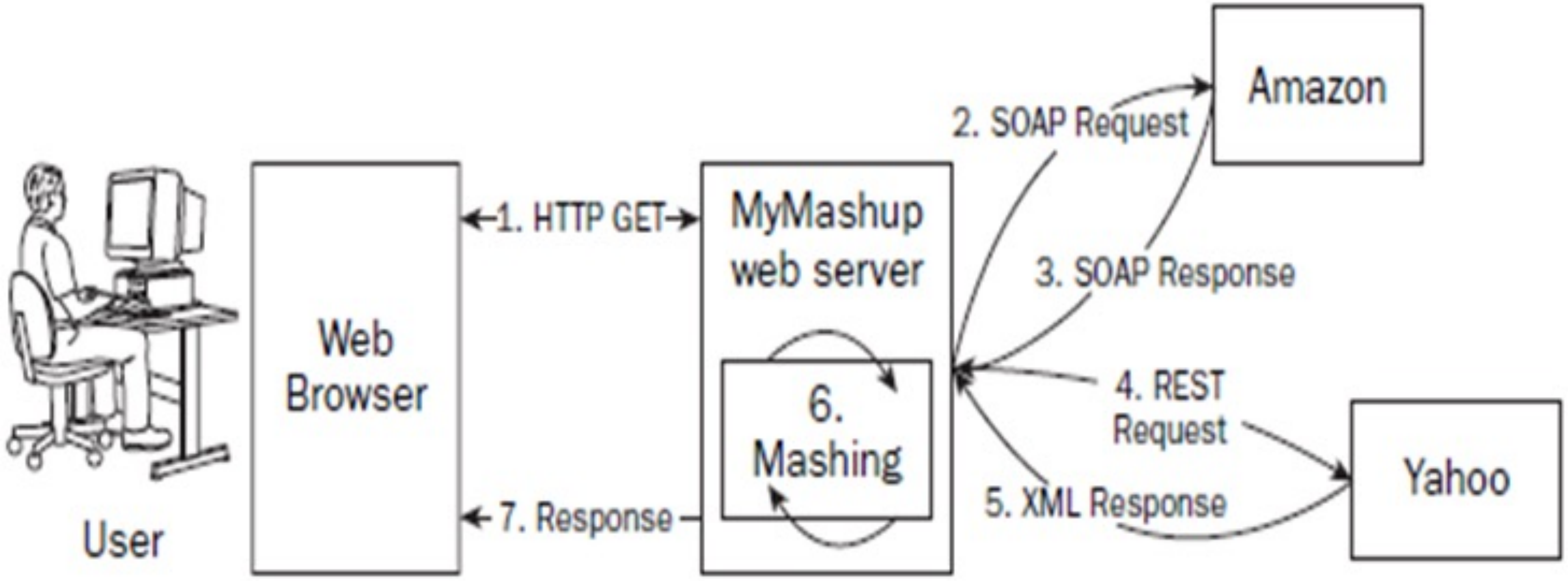
Analyse - Visualise

Be the first to know : Create email and Slack based alerts for any topic. Our easy to use queries allows you to refine the results to be exactly what you need.

Create - Follow

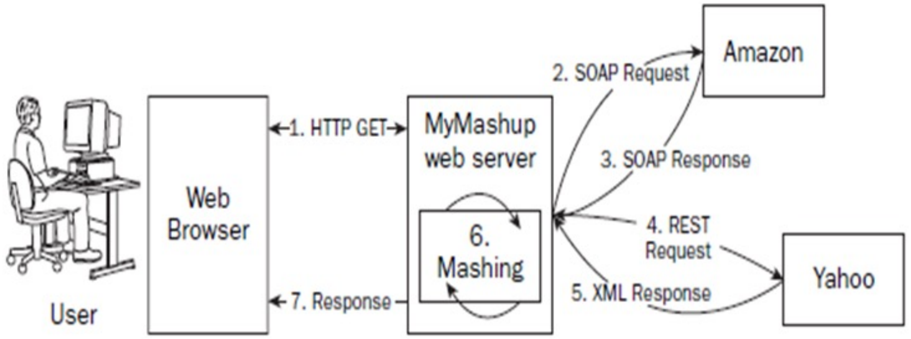
Register via Twitter in seconds for a completely free 1 week trial to access all core features.

Mashup architecture on web server: example



Mashup architecture on web server: example

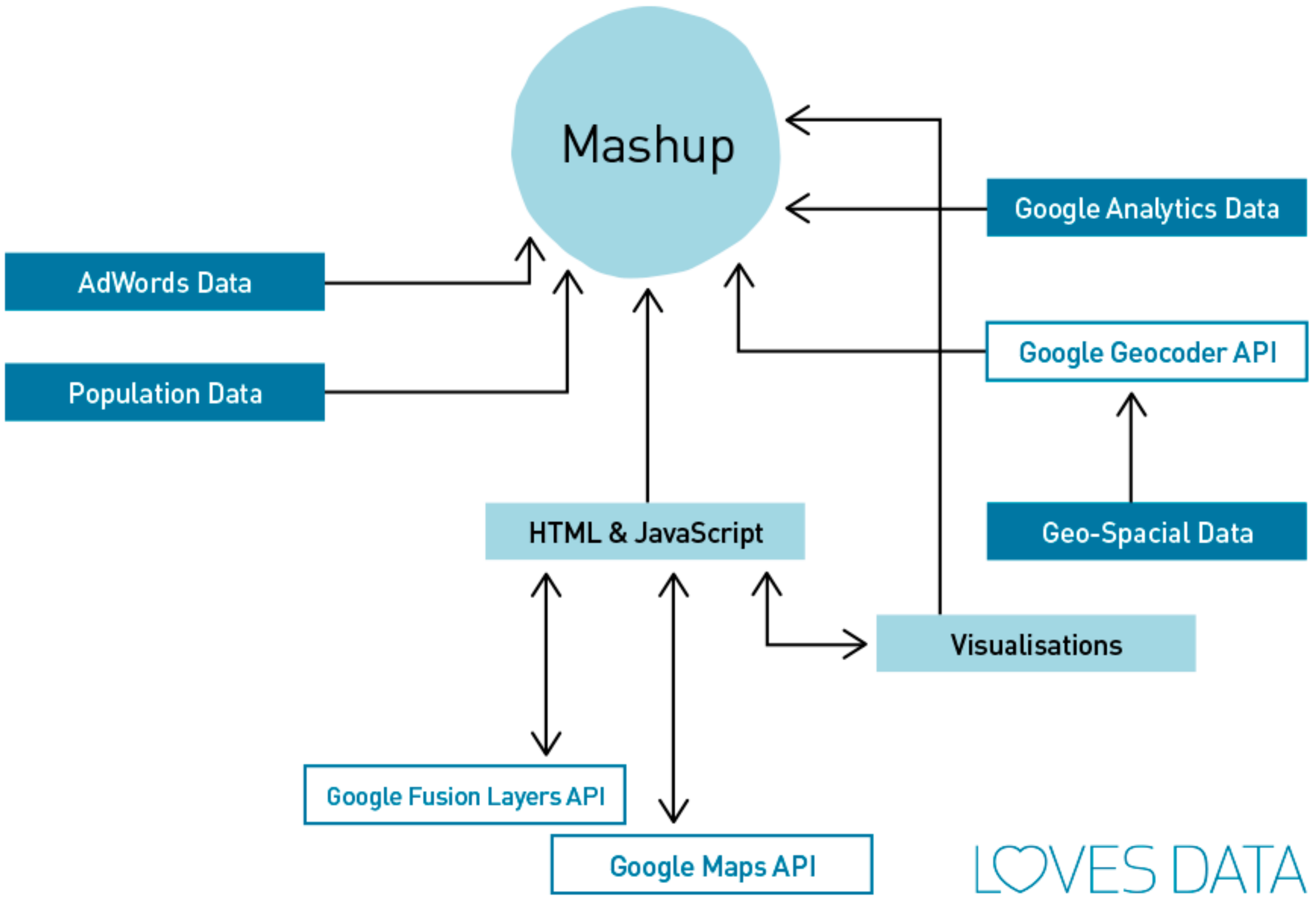
1. The web browser uses HTTP or HTTPS to send request for a Web page to the server.
2. The Web server constructs the page by connecting to the source or partner sites, which include Google, Amazon, Yahoo, etc. In our example, Amazon gets the first request from the browser using Simple Object Access Protocol (SOAP) over HTTP.
3. The web server gets the SOAP response back from Amazon.
4. Our example has the second request being sent to Yahoo through REST.
5. The web server receives plain old XML over HTTP from Yahoo.
6. The web server now combines and rationalizes the data in the most suitable manner an sends the response back.
7. The data resulting from the web server is combined in an HTML format and the response is sent back to the browser.



Kinds of mashups

- **Consumer mashups** combine data from multiple public sources in the browser and organize it through a simple browser user interface (e.g.: Wikimapia.org combines Google Map and a Wiki API)
- **Data mashups**, opposite to the consumer mashups, combine similar types of media and information from multiple sources into a single representation. The combination of all these resources create a new and distinct Web service that was not originally provided by either source.
- **Business (or enterprise) mashups** define applications that combine their own resources, application and data, with other external Web services. Enterprise mashups are visually rich Web applications that expose actionable information from diverse internal and external information sources.

Data Mashup with Google



https://www.lovesdata.com/blog/2014/mash-visualise-data

Web mashup vs web service

Web mashups	Web services
A Web Mashup is a web application that is aggregated from many other web applications and contains data from different sources merged into a tool.	Web Services allows different applications to share data and services among themselves.
Types of mashups include Server-side mashups and Client-side mashups.	Types of web services include SOAP and REST.
Web Mashups are used to share new business ideas, to increase agility, and to speed up development reducing costs.	Service descriptions are essential so that other applications can use the functionality of your program once it is exposed to the network.
Web mashups are combined information from many different websites integrated together into a new useful service.	web services are the key components in web mashups.
For example: TrendsMap, SongDNA, ThisWeKnow, etc.	**For example:** NET application communicates to java web services etc.

Web application vs web service

An application that the users access over the internet is called a web application. Any software that is accessed through a client web browser could be called a web application.

A Web Application is intended for humans, so it includes a Graphical User Interface (GUI)

A web service does not necessarily have a user interface since it is used as a component in an application offering services to other applications

A web service can be used to transfer data between Web Applications and can be accessed from any language or platform.

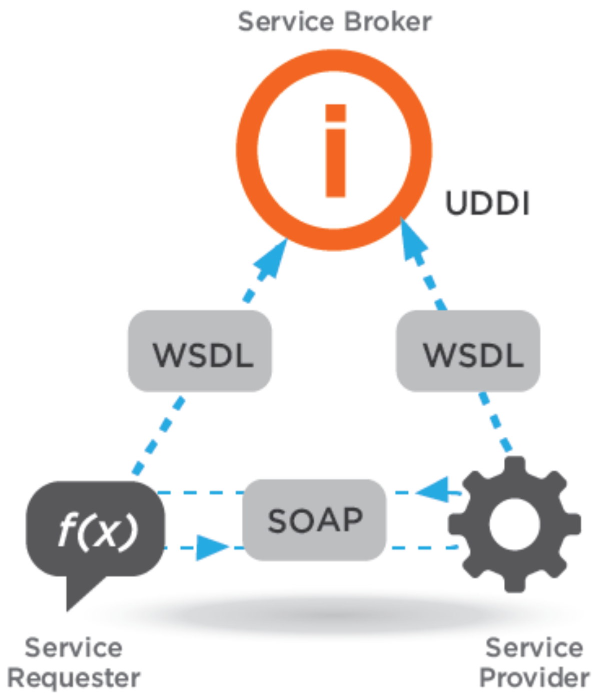
Web services

- Web services support interoperability across the Internet
- A web service provides an interface enabling clients to interact with servers in a more general way than browsers do
- Clients access the operations in the interface of a web service by means of requests and replies formatted in XML/JSON and usually transmitted over HTTP.
- Web services can be accessed in a more ad hoc manner than CORBA-based services, enabling them to be more flexible and suitable for Internet-wide applications

Definition: Web Service (according to the W3C https://www.w3.org/TR/ws-arch/)

- A web service is a software system identified by a URI and designed to support interoperable machine-to-machine interaction over a network.
- It has an *interface* described in a machine-processable format (specifically WSDL).
- Other systems interact with the web service using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.

Web services



DETAILED WEB SERVICES PROCESS

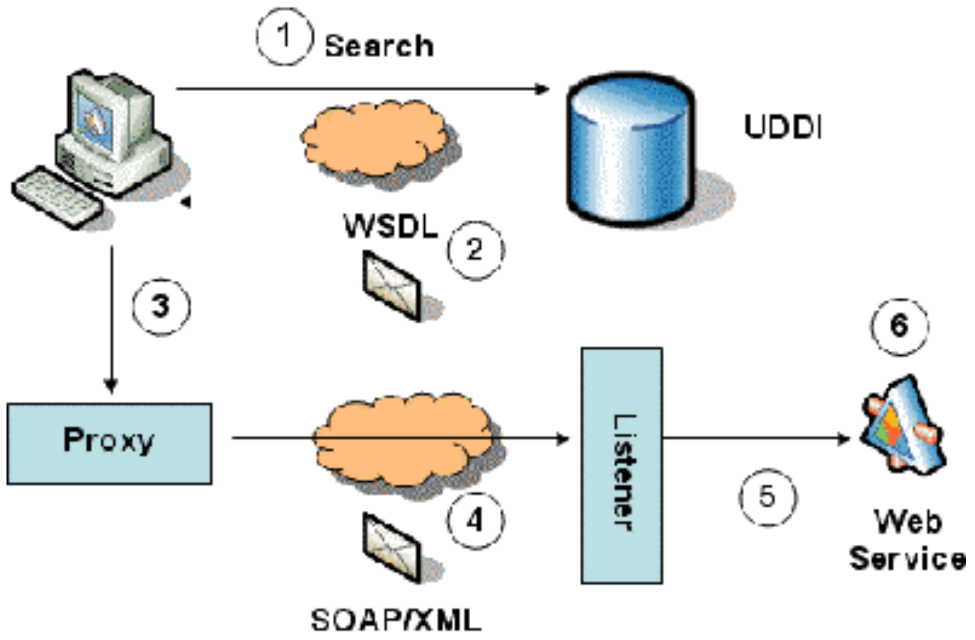
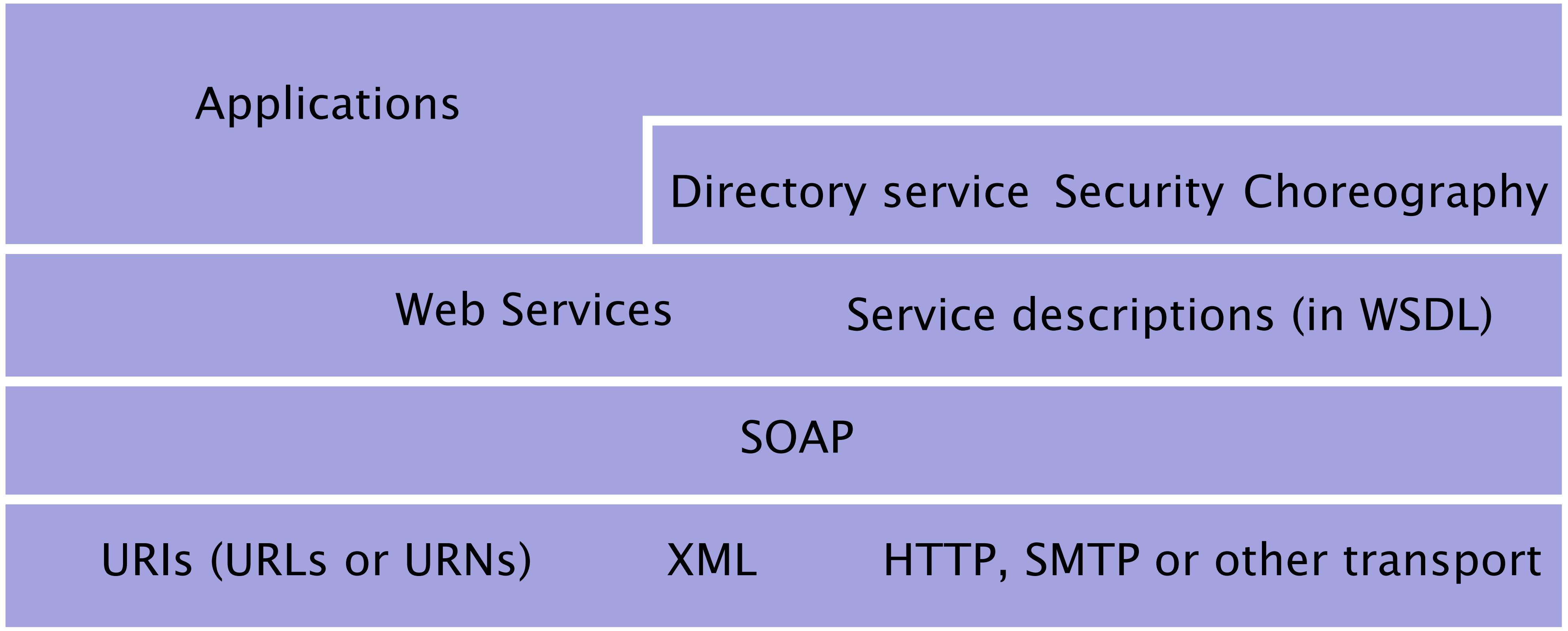


Figure 1: The process flow of a Web service

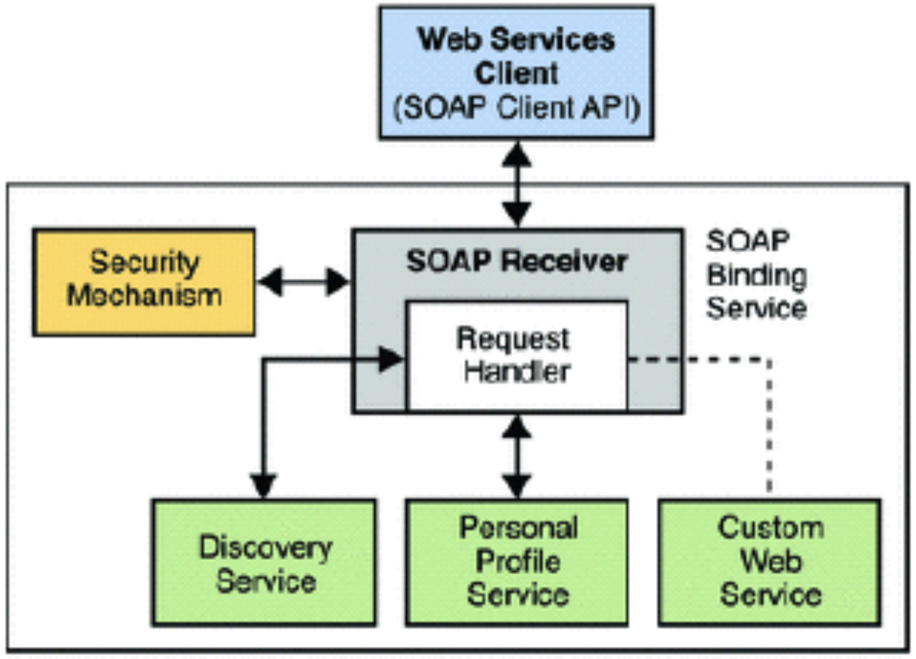
https://www.seguetech.com/web-services/

Web services infrastructure and components

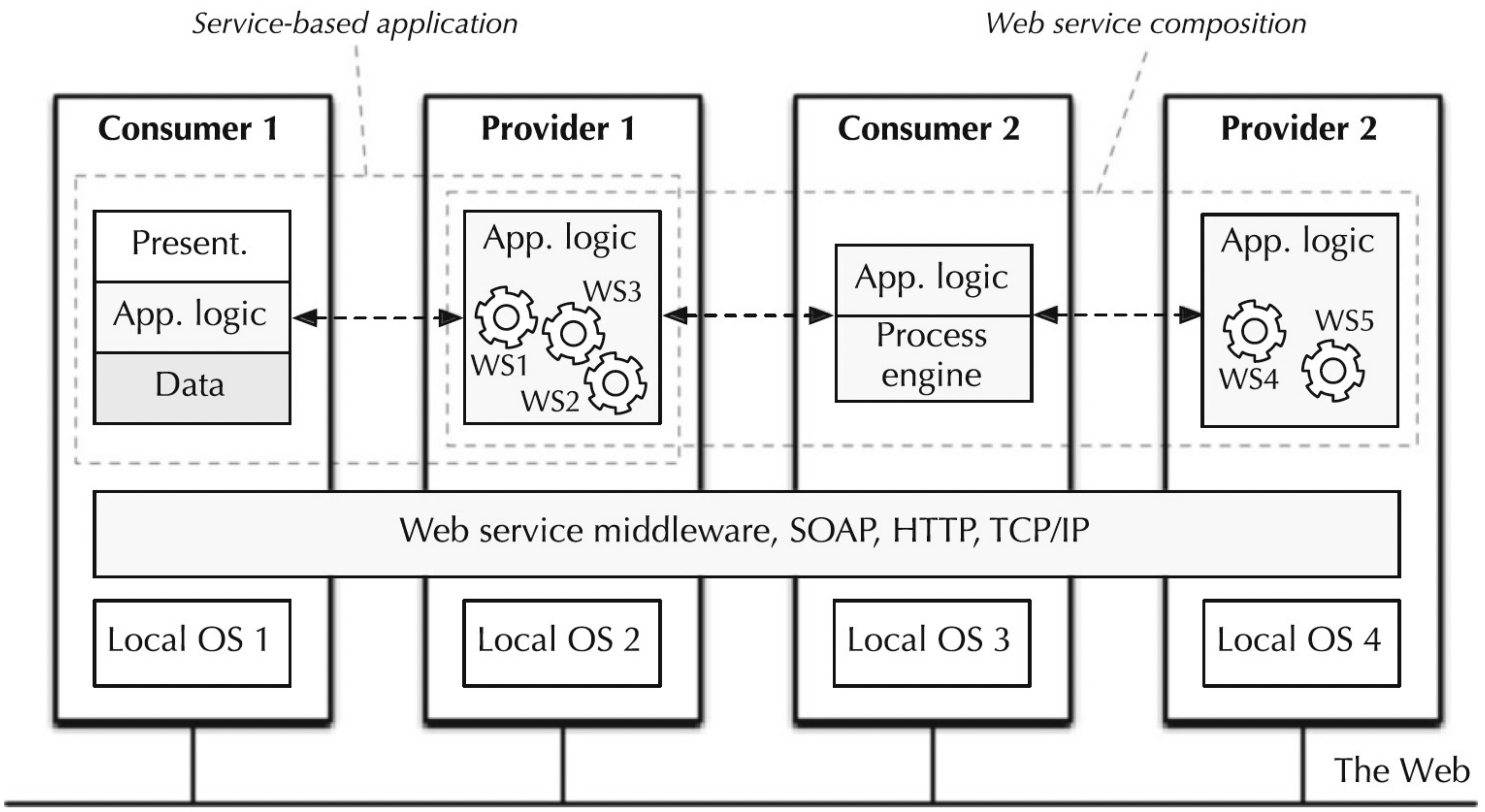


Web services are distributed applications developed using open technologies such as eXtensible Markup Language (XML), SOAP, and HyperText Transfer Protocol (HTTP).

Companies use these technologies as a mechanism for allowing their applications to cross network boundaries and communicate with those of their partners, customers and suppliers



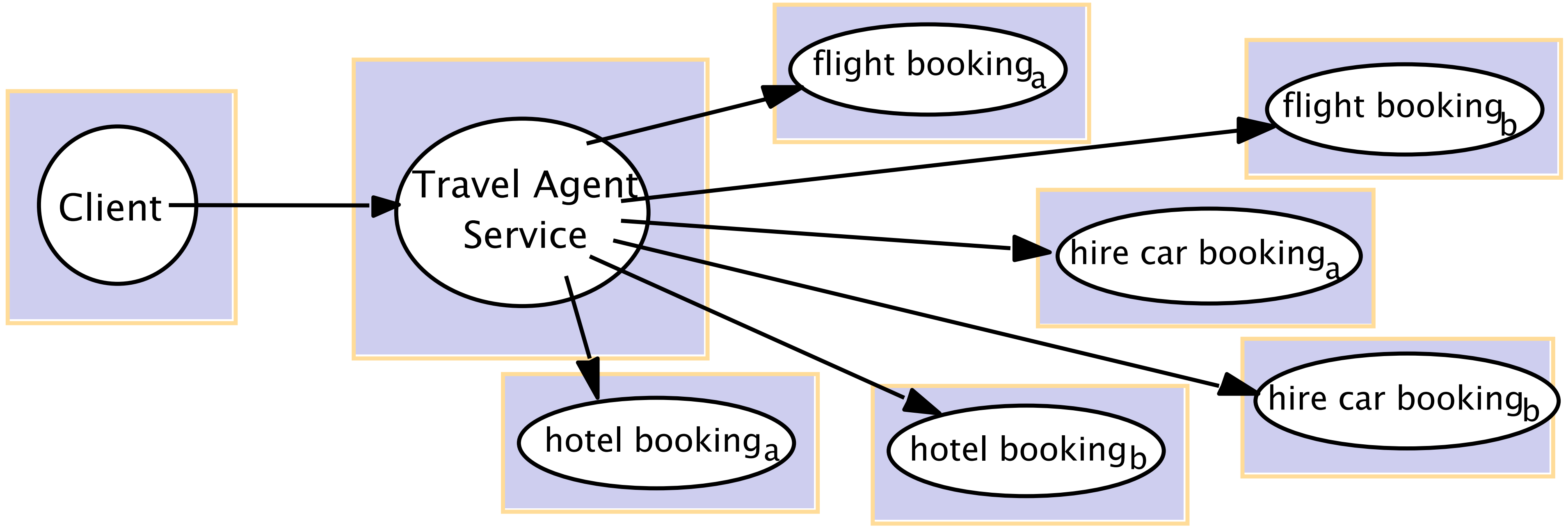
Software architecture with Web Services



Travel agent scenario

1. The client asks the travel agent service for information about a set of services; for example, flights, car hire and hotel bookings.
2. The travel agent service collects prices and availability and sends it to the client, which chooses one of the following on behalf of the user:
   (a) refine the query, possibly involving more providers to get more information, then repeat step 2;
   (b) make reservations;
   (c) quit.
3. The client requests a reservation and the travel agent service checks availability.
4. Either all are available;
   or for services that are not available;
   either alternatives are offered to the client who goes back to step 3;
   or the client goes back to step 1.
5. Take deposit.
6. Give the client a reservation number as a confirmation.
7. During the period until the final payment, the client may modify or cancel reservations

The ‘travel agent service’ combines other web services



Providing an interface for a web service allows its operations to be combined with those of other services to provide new functionality

SOAP

- SOAP is a protocol to enable both client-server and asynchronous interactions over the Internet
- It defines a scheme for using XML to represent the contents of request and reply as well as a scheme for the communication of documents
- Originally SOAP was based only on HTTP, but now it can use several transport protocols including SMTP, TCP or UDP
- Web Services realize interoperability through SOAP: it is the “language” that different Web Service implementations use to exchange messages.

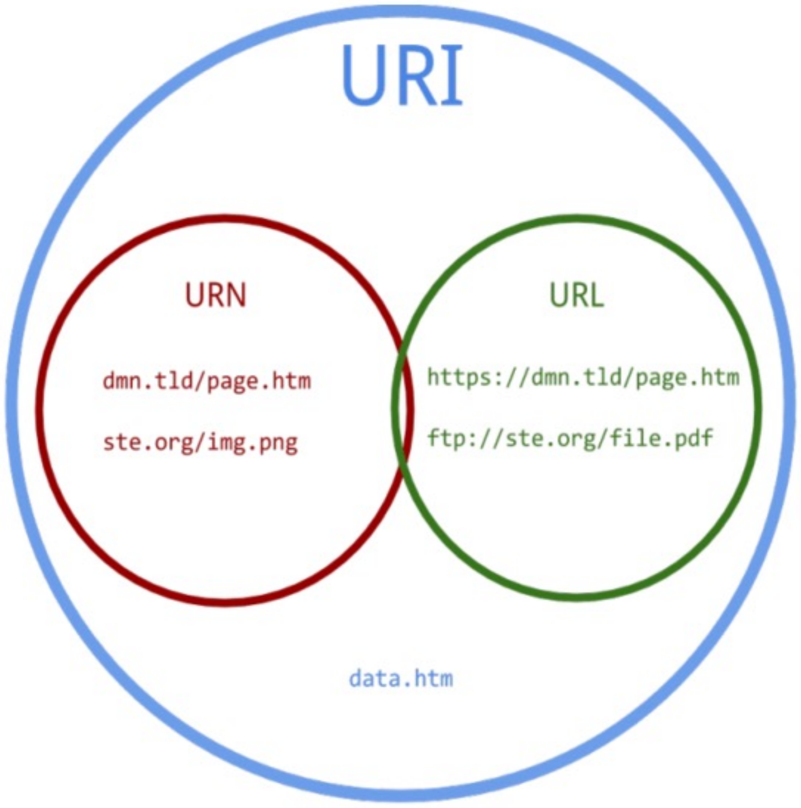
SOAP

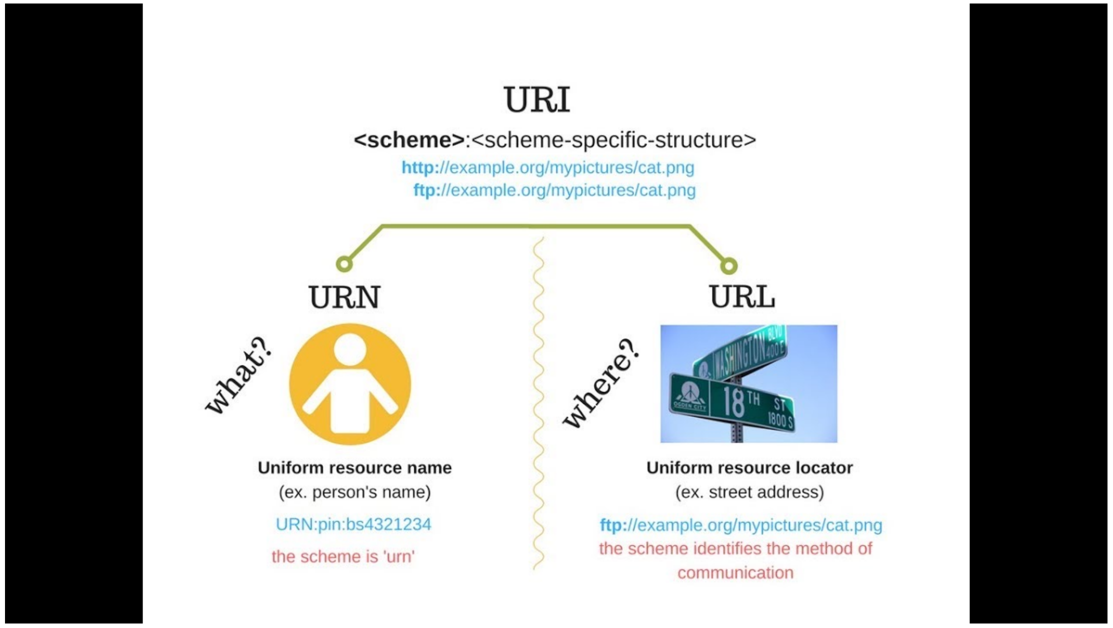
The SOAP specification states:

- how XML is to be used to represent the contents of individual messages;
- how a pair of single messages can be combined to produce a request-reply pattern;
- the rules as to how the recipients of messages should process the XML elements that they contain;
- how HTTP and other protocols should be used to communicate SOAP messages

URI, URL, URN

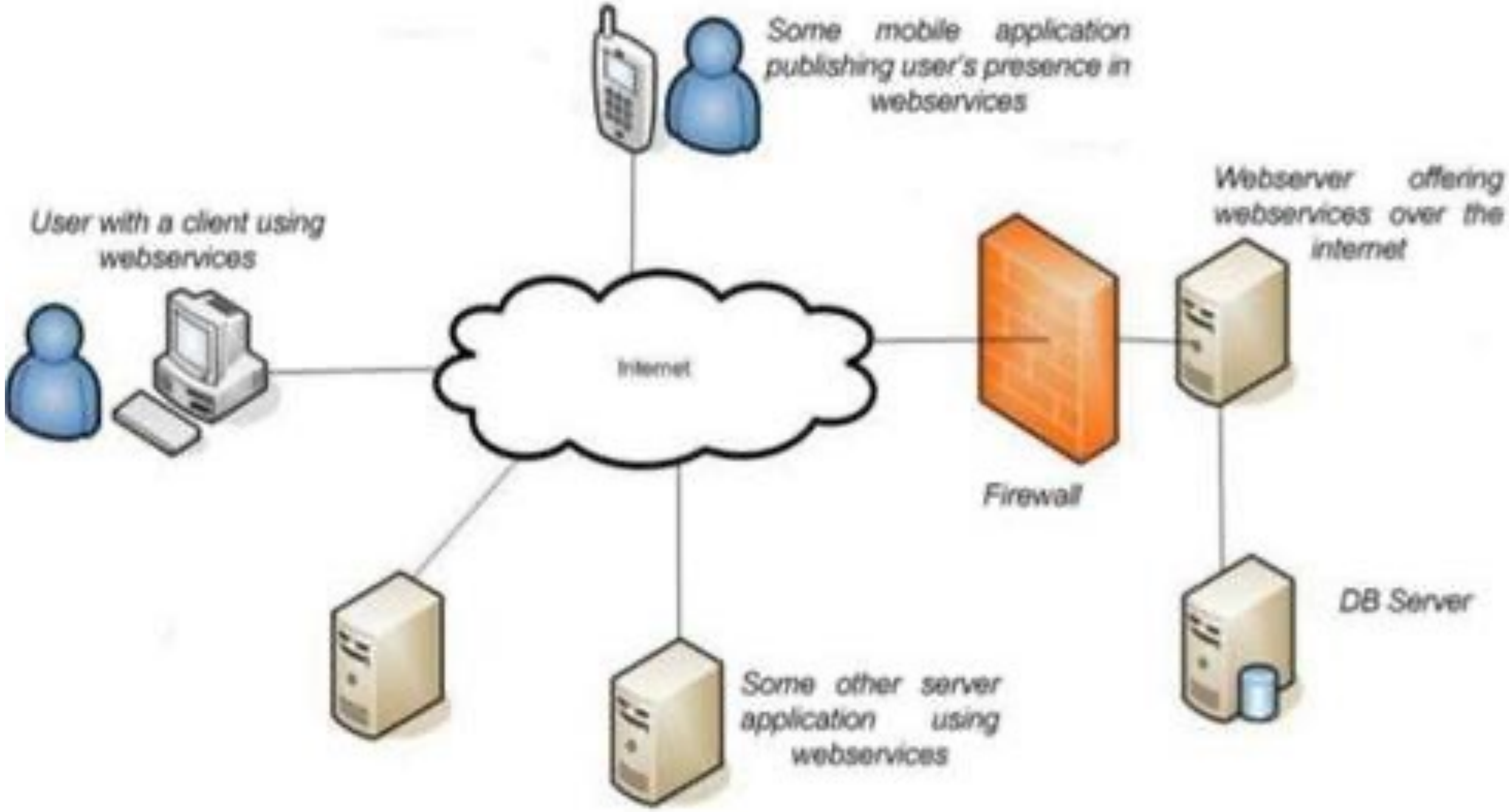
A Uniform Resource Identifier (URI) is a unique sequence of characters that identifies a logical or physical resource used by web technologies. URIs may be used to identify anything, including real-world objects, such as people and places, concepts, or information resources such as web pages and books.




URI
<scheme>:<scheme-specific-structure>
http://example.org/mypictures/cat.png
ftp://example.org/mypictures/cat.png
URN
what?
Uniform resource name
(ex. person's name)
URN:pin:bs4321234
the scheme is 'urn'
URL
where?
Uniform resource locator
(ex. street address)
ftp://example.org/mypictures/cat.png
the scheme identifies the method of communication

https://en.wikipedia.org/wiki/Uniform_Resource_Identifier

Need for web services


Some mobile application publishing user's presence in webservices
User with a client using webservices
Internet
Firewall
Webserver offering webservices over the internet
DB Server
Some other server application using webservices

WebServices vs RMI

- A web service has an interface that can provide operations for accessing and updating the data resources it manages.
- At a superficial level, the interaction between client and server is very similar to RMI, where a client uses a remote object reference to invoke an operation in a remote object.
- For a web service, the client uses a URI to invoke an operation in the resource named by that URI.

WS vs RMI

- The URI of a web service can be compared with the remote name (=object reference) of a single object.
- However, in the distributed object model, objects can create remote objects dynamically and return remote references to them
- Nothing like this can be done with web services, which cannot create instances of remote objects
- A web service consists of a single remote object and therefore both garbage collection and remote object referencing are irrelevant

WSDL

- The Web Services Description Language is an XML-based interface definition language that is used for describing the functionality offered by a web service.
- The acronym is also used for any specific WSDL description of a web service (also referred to as a WSDL file), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.
- Therefore, its purpose is roughly similar to that of a method signature in a programming language.
- The current version of WSDL is WSDL 2.0.

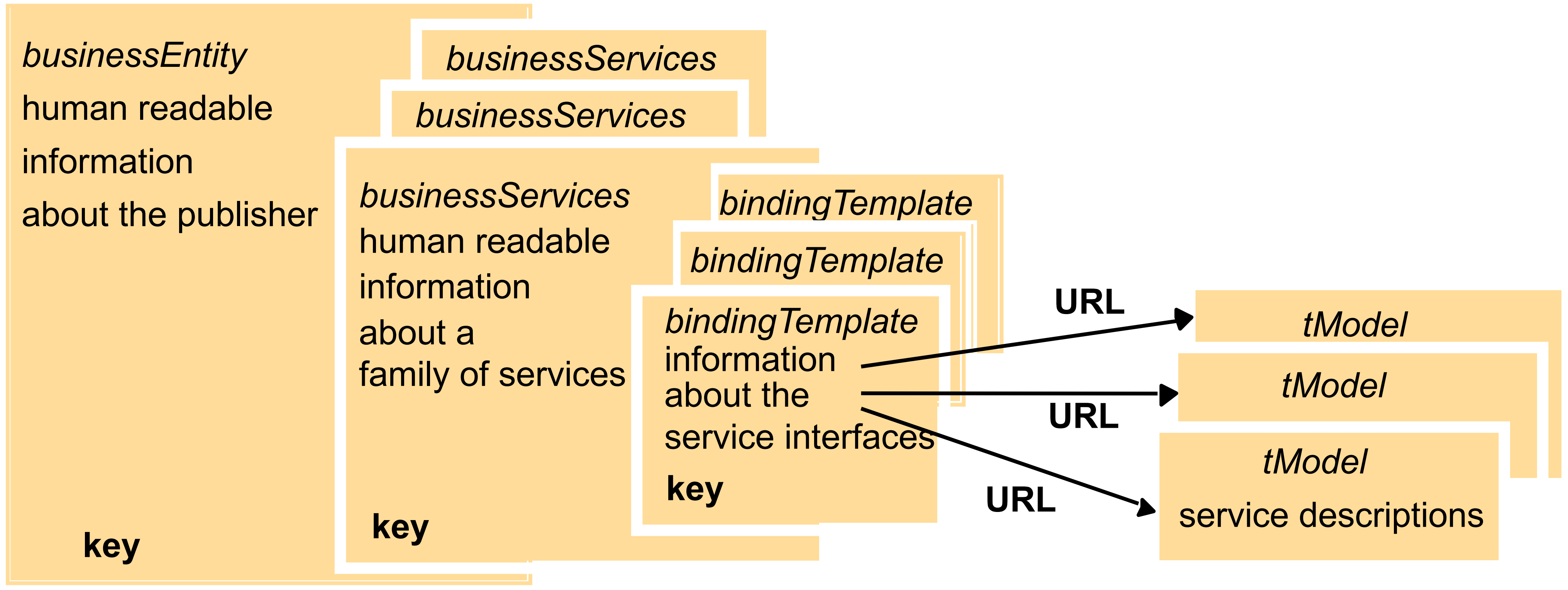
Directory service

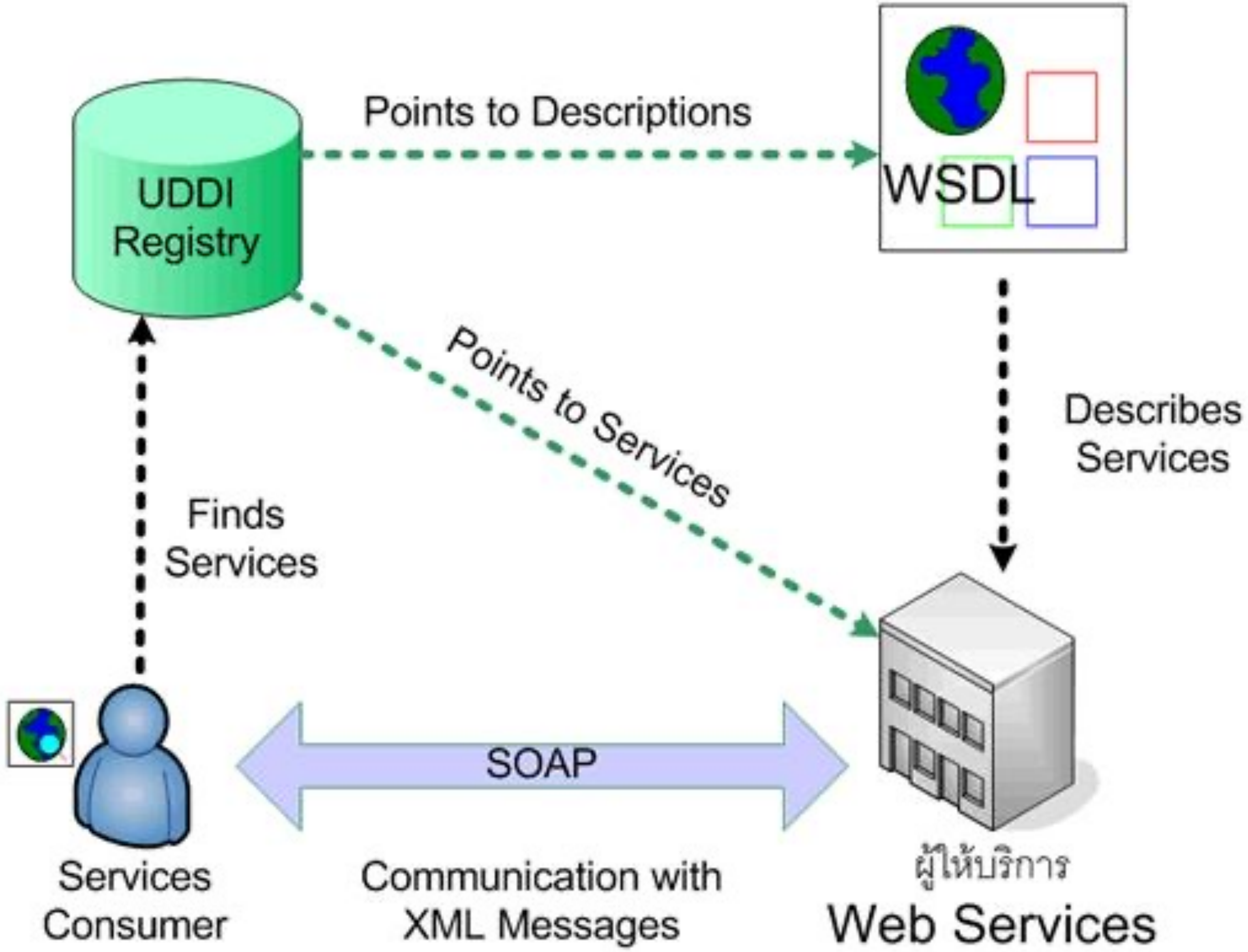
- Any organization that plans to base its applications on web services will find it more convenient to use a directory service to make these services available to clients.
- This is the purpose of the Universal Description, Discovery and Integration service (UDDI), which provides both a name service and a directory service
- WSDL service descriptions may be looked up by name (a white pages service) or by attribute (a yellow pages service).
- They may also be accessed directly via their URLs, which is convenient for developers who are designing client programs that use the service

UDDI

- Universal Description, Discovery and Integration (UDDI, pronounced /ˈjʊdiː/) is a platform-independent, Extensible Markup Language protocol that includes a (XML-based) registry by which businesses worldwide can list themselves on the Internet, and a mechanism to register and locate web service applications.
- UDDI is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), for enabling businesses to publish service listings and discover each other, and to define how the services or software applications interact over the Internet

The main UDDI data structures




UDDI Registry
Points to Descriptions
WSDL
Points to Services
Describes Services
Finds Services
SOAP
Services Consumer
Communication with XML Messages
ผู้ให้บริการ
Web Services

XML security

- «XML security» consists of a set of related W3C designs for signing, key management and encryption
- It is intended for use over the Internet involving documents whose contents may need to be authenticated or encrypted
- Example: a document containing a patient’s medical records

XML signature

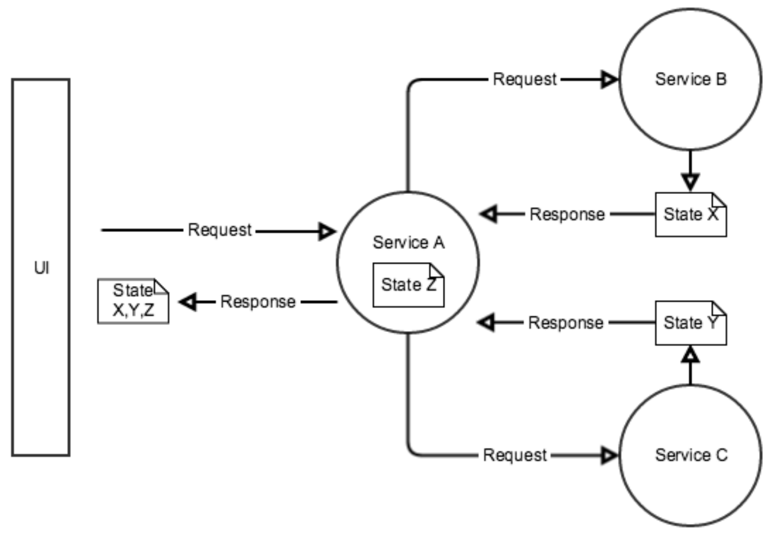
- «XML Signature» (also called XMLDSig, XML-DSig, XML-Sig) defines an XML syntax for digital signatures and is defined in the W3C recommendation XML Signature Syntax and Processing.
- Functionally, it has much in common with PKCS#7 but is more extensible and geared towards signing XML documents.
- It is used by various Web technologies such as SOAP, SAML, and others.

- XML signatures can be used to sign a resource of any type, typically XML documents, but anything that is accessible via a URL can be signed.
- An XML signature used to sign a resource outside its containing XML document is called a detached signature; if it is used to sign some part of its containing document, it is called an enveloped signature; if it contains the signed data within itself it is called an enveloping signature

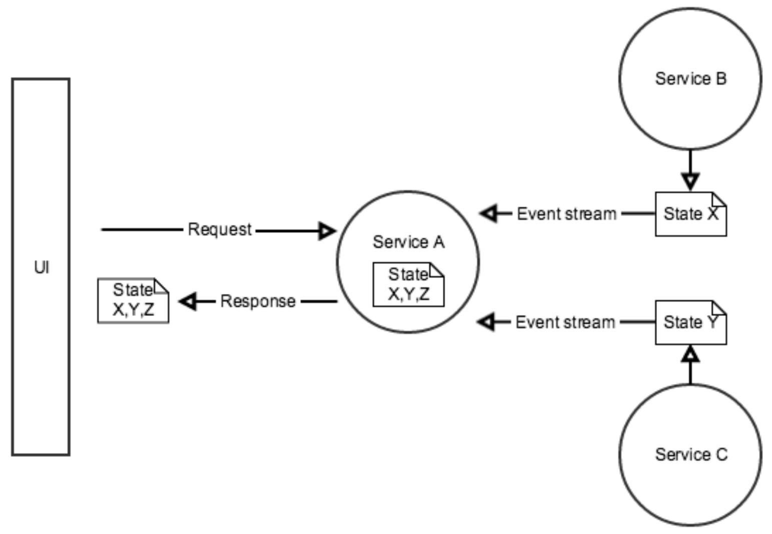
WS choreography («Dancers dance following a global scenario without a single point of control»)

- The W3C uses the term *choreography* to refer to a language based on WSDL for defining service coordination
- For example, the language might specify constraints on the order and the conditions in which messages are exchanged by participants.
- A choreography is intended to provide a global description of a set of interactions, showing the behaviour of each member of a set of participants, with a view to enhancing interoperability.

Orchestration and Choreography



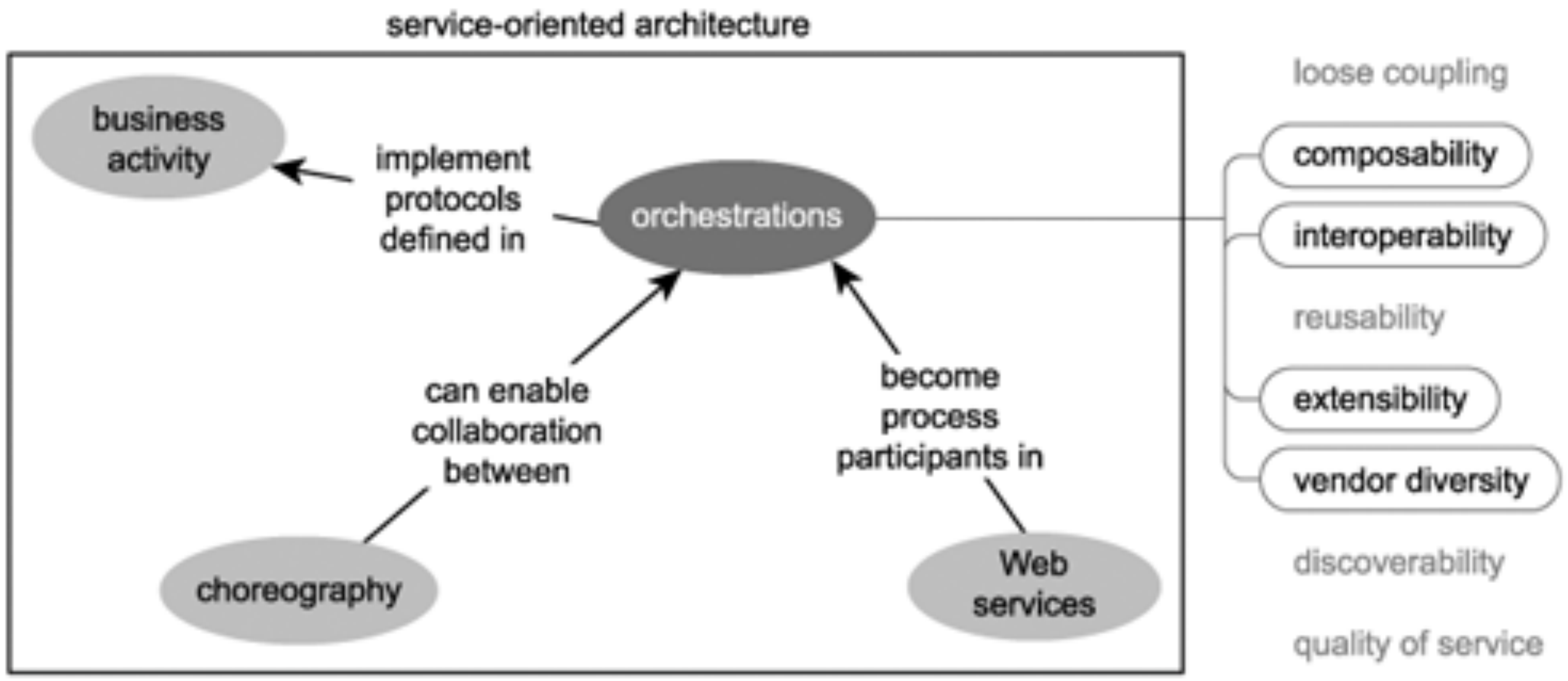
Orchestration: service A is dependent on both service B and service C when handling its own inbound HTTP calls. Service A fetches state X (from service B) state Y (from service C), aggregating them with some of its own state, Z, to finally yield an HTTP response to its callers. Failures in either B or C will prevent A from fully fulfilling its responsibilities



Choreography: When handling an incoming request, service A does not need to communicate with service B or service C

https://medium.com/@dreweaster/the-art-of-microservices-integration-using-service-choreography-69a4bbbf81c5

WS, orchestrations and choreographies



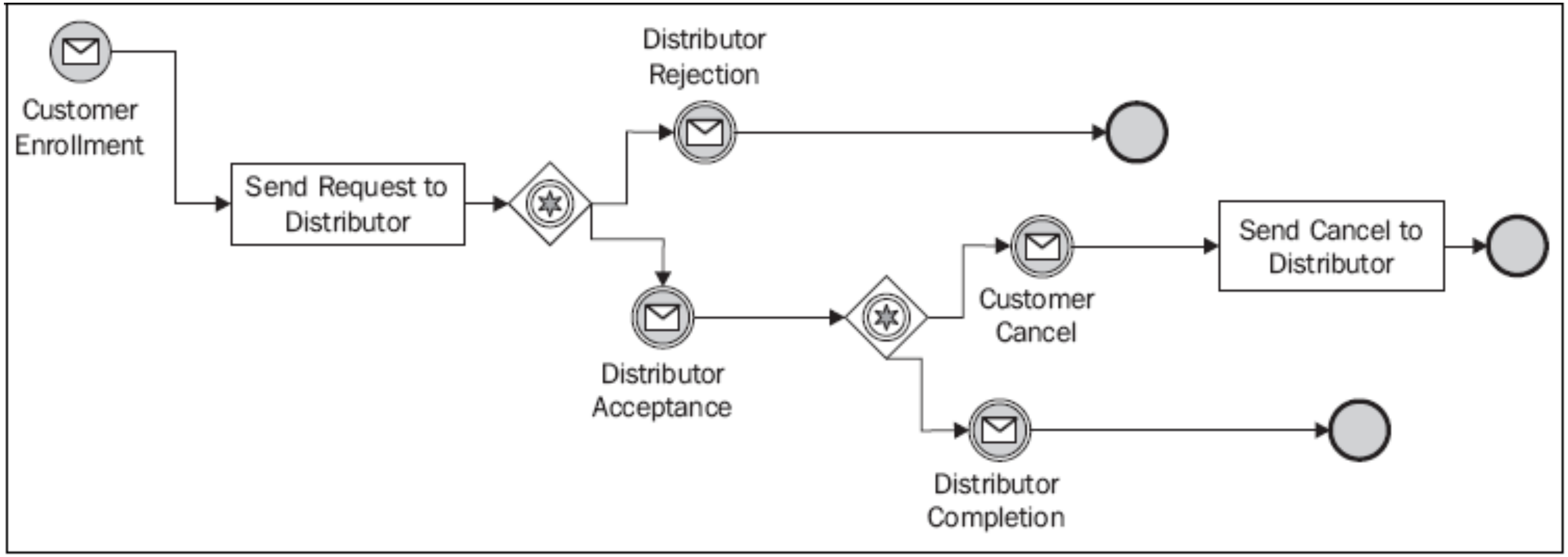
Orchestration and choreography

An orchestration process includes public and private activities

The public activities are those required by the choreography

Private activities are there to meet internal requirements, but are not visible to partners

The figure shows the public activities of the orchestration process for an energy retailer



www.packtpub.com/article/modeling-orchestration-and-choreography-in-service-oriented-architecture

Choreography languages: features

A choreography language includes the following features:

- hierarchical and recursive composition of choreographies;
- the ability to add new instances of an existing service and new services;
- concurrent paths, alternative paths and the ability to repeat a section of a choreography;
- variable timeouts – for example, different periods for holding reservations;
- exceptions, for example, to deal with messages arriving out of sequence and user actions such as cancellations;
- asynchronous interactions (callbacks);
- reference passing, for example, to allow a car hire company to consult a bank for a credit check on behalf of a user;
- marking of the boundaries of the separate transactions that take place, for example, to allow for recovery;
- the ability to include human-readable documentation

Choreography and orchestration languages

the following specifications focus on defining languages to model service choreographies:

- Web Service Choreography Description Language (WS-CDL) is a W3C XML-based specification for modelling choreographies using constructs inspired by Pi calculus
- Web Service Choreography Interface (WSCI) is an XML-based specification served as input to the Web Service Choreography Description Language (WS-CDL)
- The OMG specification BPMN version 2.0 includes diagrams to model service choreographies.

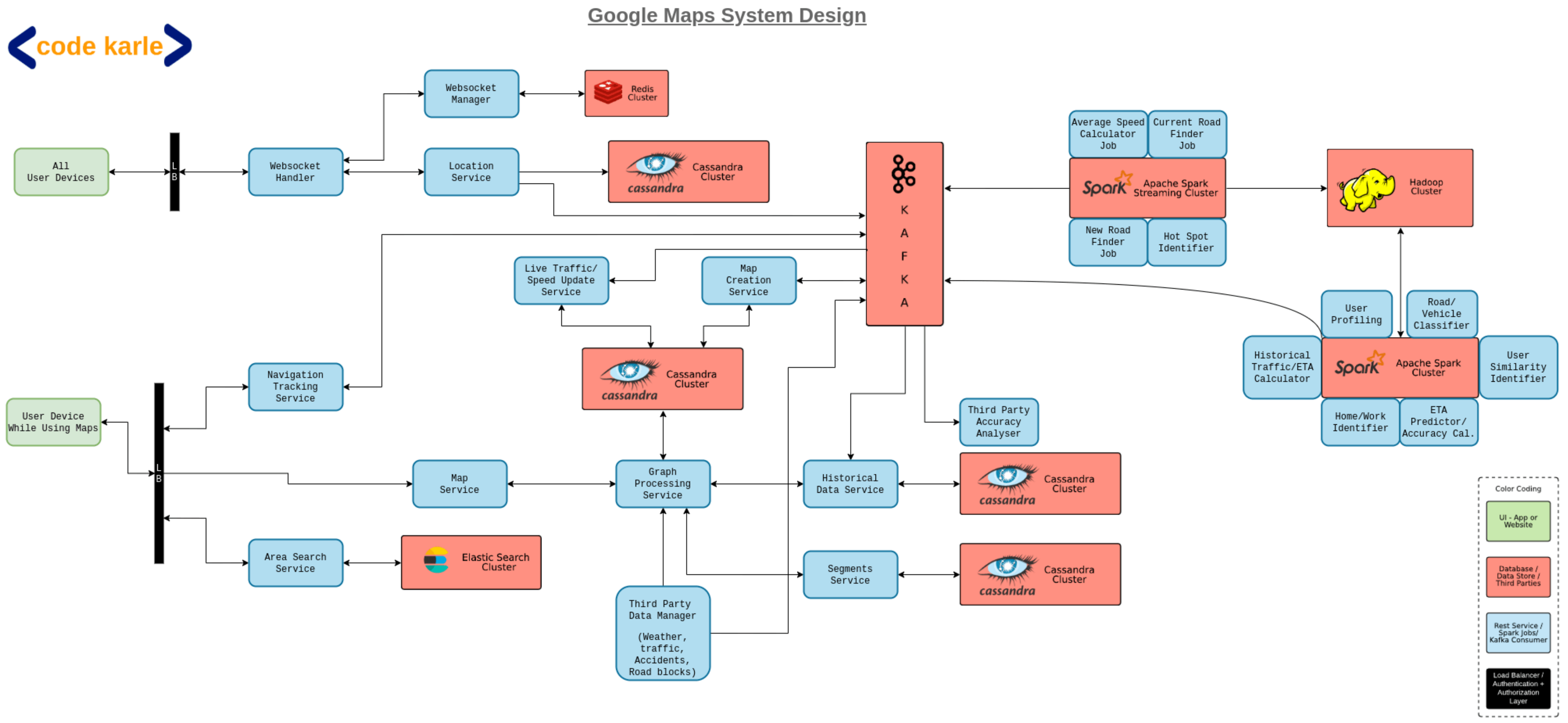
Proposals for service choreography languages include: Let's Dance, BPEL4Chor, Chor

A number of theoretic choreography formalisms have been proposed based on:

- Petri Nets and variants
- Finite State Machines
- Guarded Automata, Timed Automata
- Pi calculus and other process calculi

Case study: Google maps

The Google Maps Platform web services are a collection of HTTP interfaces to Google services providing geographic data for your maps applications.



code karle
Google Maps System Design
All User Devices
LB
Websocket Handler
Websocket Manager
Redis Cluster
Location Service
Cassandra Cluster
KAFKA
Average Speed Calculator Job
Current Road Finder Job
Apache Spark Streaming Cluster
New Road Finder Job
Hot Spot Identifier
Hadoop Cluster
Live Traffic/ Speed Update Service
Map Creation Service
Cassandra Cluster
Navigation Tracking Service
User Device While Using Maps
LB
Map Service
Graph Processing Service
Historical Data Service
Cassandra Cluster
Third Party Accuracy Analyser
Area Search Service
Elastic Search Cluster
Segments Service
Cassandra Cluster
Third Party Data Manager (Weather, traffic, Accidents, Road blocks)
User Profiling
Road/ Vehicle Classifier
Historical Traffic/ETA Calculator
Apache Spark Cluster
User Similarity Identifier
Home/Work Identifier
ETA Predictor/ Accuracy Cal.
Color Coding
UI - App or Website
Database / Data Store / Third Parties
Rest Service / Spark Jobs/ Kafka Consumer
Load Balancer / Authentication + Authorization Layer

https://www.codekarle.com/system-design/Google_Maps-system-design.html

Conclusions

- Web services have arisen from the need to provide a global infrastructure to support interoperability between different organizations.
- This infrastructure uses the widely used HTTP protocol to transport messages between clients and servers over the Internet and is based on the use of URIs to refer to resources.
- XML or JSON are used for data representation and marshalling.

Conclusions

Two separate goals led to the emergence of Web Services.

- One was the addition of **service interfaces** to web servers with a view to allowing the resource on a site to be accessed by client programs other than browsers and using a richer form of invocation
- The other was the desire to provide something like RPC over the Internet, based on the existing protocols.
- Web services provide interfaces with sets of operations that can be called remotely.
- Like any other form of service, a web service can be the client of another web service, thus allowing a web service to integrate or combine a set of other web services.
- SOAP is the communication protocol that is generally used by web services and their clients.

Conclusions: Mashups vs web services

Mashups:

1. **Integration of Content:** Mashups involve combining content and services from multiple sources to create a new, integrated web application. This could include data from various websites, like maps, social media, news, or other APIs.
2. **User-Focused:** Mashups are typically user-centric and are designed to provide a seamless and personalized experience for end-users. They often involve presenting data or services in a visually appealing and user-friendly manner.
3. **Client-Side:** Most mashups are implemented on the client side, meaning the integration and presentation of data occur in the user's browser. JavaScript and HTML are commonly used to create mashup applications.
4. **Examples:** A classic example of a mashup is a map that combines data from Google Maps and real-time traffic information.

Conclusions: Mashups vs web services

Web Services:

1. **Interoperable Data Exchange:** Web services are designed for interoperable data exchange between different software systems. They are not focused on user interfaces but rather on providing machine-to-machine communication.
2. **Standardized Protocols:** Web services often use standardized communication protocols like SOAP (Simple Object Access Protocol) or REST (Representational State Transfer) for data exchange.
3. **Server-Side:** Web services are typically implemented on the server side, exposing APIs that can be accessed by various clients, including web applications, mobile apps, and other software systems.
4. **Examples:** Examples of web services include weather data APIs, payment processing APIs, or social media APIs like Twitter's API, which allows developers to access and interact with Twitter's data.

Exercise part a: mashup

Topic: Weather Forecast Mashup

Description: Create a weather forecast mashup that combines weather data from multiple sources and presents a comprehensive weather forecast for a specific location. This mashup should include data like current conditions, hourly forecasts, and a 5-day outlook.

Hints to the Solution:

1. Choose two or more weather data sources or APIs (e.g., OpenWeatherMap, Weather.com, or a local weather service).
2. Design a user-friendly web interface using HTML, CSS, and JavaScript to input and display the weather data.
3. Write JavaScript code to fetch data from the selected weather APIs and organize it into a unified forecast.
4. Include features like current conditions, hourly forecasts, and a 5-day outlook.
5. Ensure the mashup provides accurate and up-to-date weather information for the specified location.

Exercise part b: web service

Topic: Weather Forecast Web Service

Description: Develop a weather forecast web service that allows users to retrieve weather forecasts for specific locations by sending requests to your service. The service should provide current weather conditions, hourly forecasts, and daily forecasts.

Hints to the Solution:

1. Choose a language (e.g., Python) to implement your weather forecast web service.
2. Utilize established weather data APIs (e.g., the OpenWeatherMap API) to fetch real-time weather data.
3. Create RESTful API endpoints for retrieving weather information based on parameters like location, time frame (hourly or daily), and forecast details.
4. Structure the API responses in a clear and standardized format (e.g., JSON) that includes current conditions, hourly forecasts, and daily forecasts.
5. Test your web service with different location and timeframe requests to ensure it provides accurate weather information.

Exercise part c. Discuss/compare strenghts and weaknesses of the two solutions