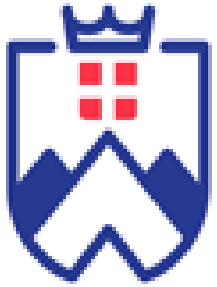


UNIVERSITÉ
SAVOIE
MONT BLANC

Architectures Orientées Services

David TELISSON

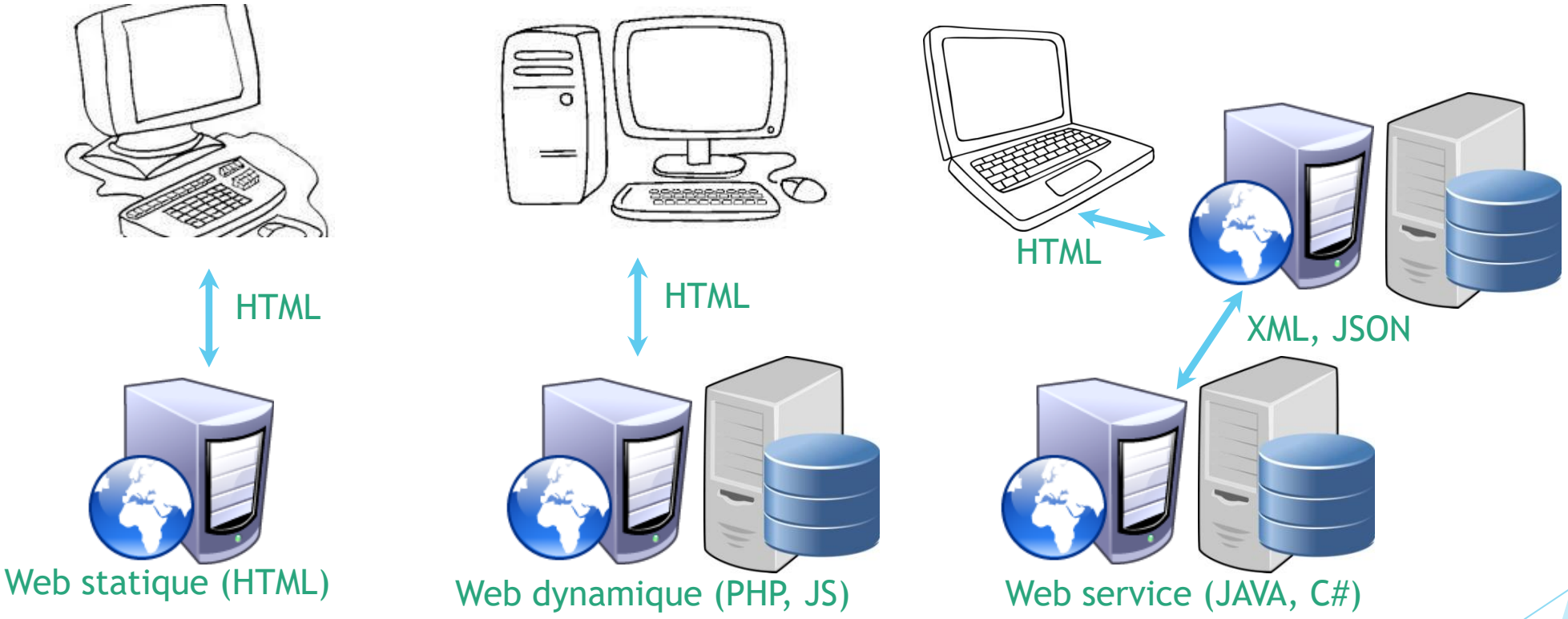
Un service Web, c'est quoi ?

- Un service Web est une unité logique applicative accessible en utilisant les protocoles standard d'Internet
- Une librairie fournissant des données et des services à d'autres applications
- Un objet métier qui peut être déployé sur Internet avec une faible dépendance vis-à-vis des technologies et des protocoles

Caractéristiques d'un WS

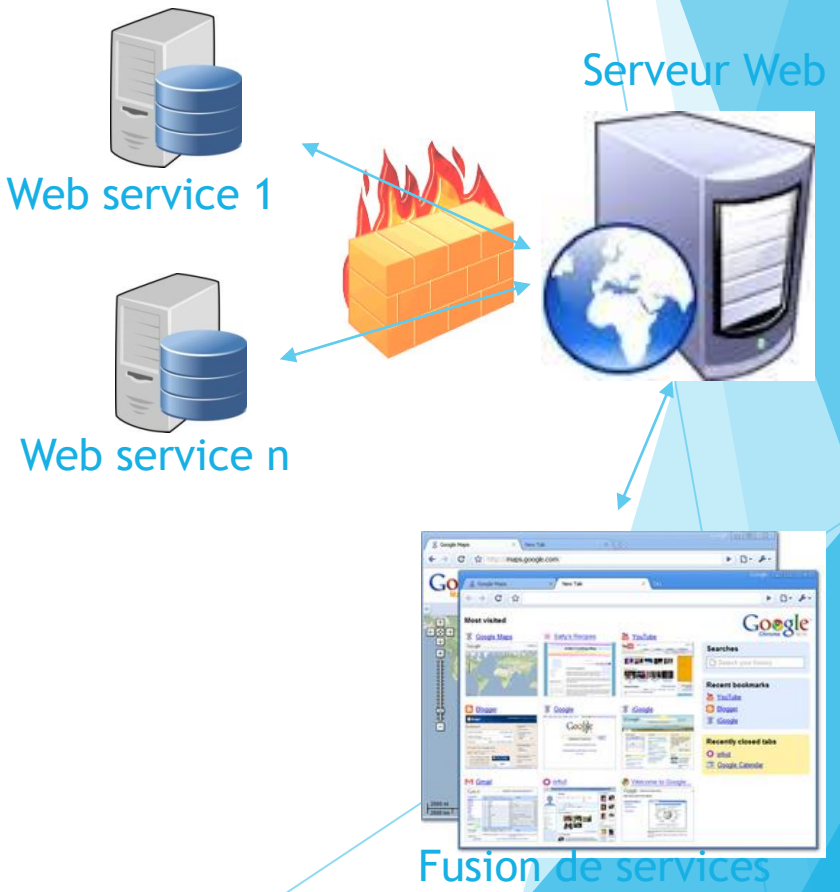
- Réutilisable
- Indépendant
  - De la plateforme sur laquelle il est déployé (Unix, Windows, etc.)
  - Du langage dans lequel il est implémenté (Java, PHP, C#, etc.)
  - De l'architecture sous-jacente (J2EE, .Net, Axis, etc.)

L’évolution du web



Le web orienté service

- Historiquement
  - Un site web fourni des pages HTML
  - Impossible à fusionner avec les pages d'un autre site
- Aujourd'hui
  - Un site web est un composant fournissant des services décrits en XML ou JSON
  - Il se construit par assemblage de services tiers



Des WS pour quoi faire ?

- Les services web permettent d'interconnecter
  - Différentes entreprises
  - Différents matériels
  - Plusieurs applications
  - Plusieurs clients (et pas seulement des navigateurs !)
- Distribuer et intégrer des logiques métiers
- Les Services Web sont faiblement couplés

Le Web et le client serveur

- Java RMI
  - Mono-langage : Java, multi-plateforme(JVM)
  - Nécessite d'ouvrir des ports spécifiques (performance, sécurité, …)
- CORBA / IIOP
  - Multilangage, multi-plateforme, Multi-vendeurs, OMG
  - Installation coûteuse (licences, ressources humaines)
- DCOM
  - Plateforme Win32 uniquement
  - Faible diffusion

Quels sont les objectifs ?

- Remplacer les protocoles historiques (RPC,DCOM,RMI) par une approche entièrement ouverte et interopérable, basée sur la généralisation des serveurs Web.
- Faire interagir des composants hétérogènes, distants, et indépendants avec des protocoles standard (HTTP, SOAP).
- Enrichir les applications B2B (Business to Business) et EAI (Enterprise Application Integration).

Solution

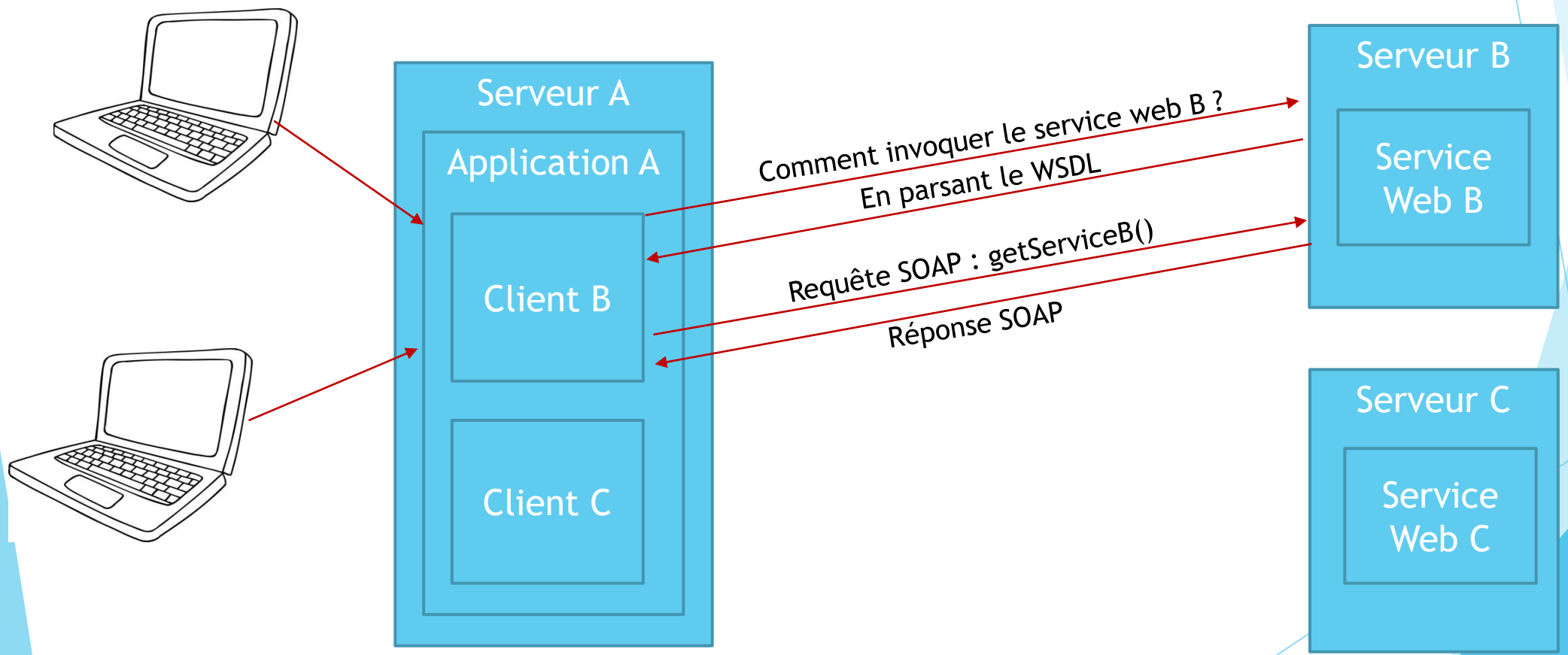
- Le Web avait besoin d'un protocole
  - Multi-langages, multi-plateformes
  - Respectant les formats d'échanges du Web
    - Réponses et requêtes en XML
  - Facile à implémenter sur différents protocoles de transport
    - RPC, HTTP ou autre MOM
  - Permettant de franchir les pare-feu
  - Avec une spécification non propriétaire garantie par un organisme
    - Le W3C
- La réponse :
  - SOAP (Simple Object Access Protocol)
  - WSDL (Web Service Description Language)

Cycle de vie d'un service Web

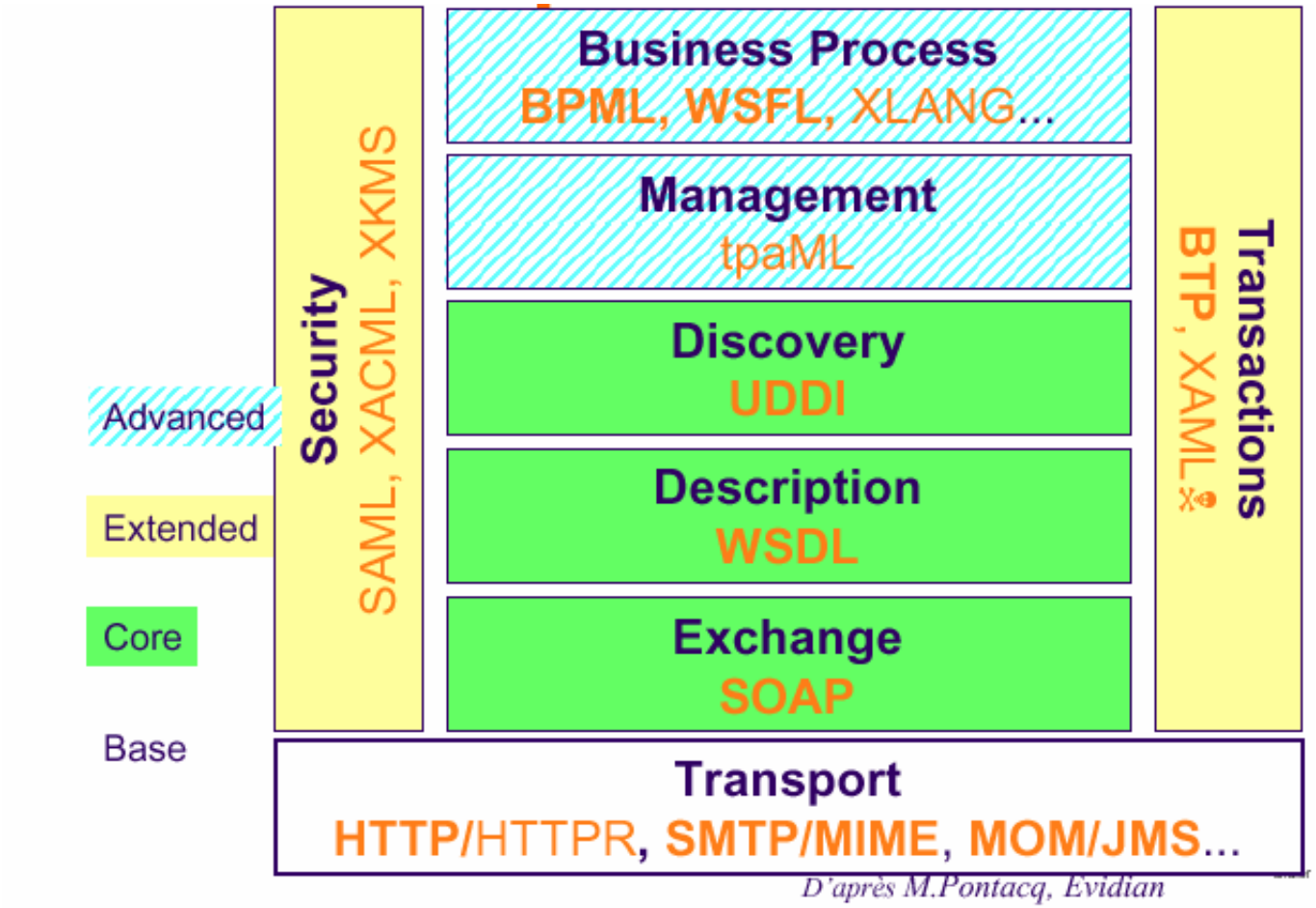
- Côté serveur
  - Implémentation du service (Java, C#, PHP, etc.)
  - Déploiement du service sur un serveur d'application (Jboss, TomCat, Glassfish, etc.)
  - Description du service Web = génération d'un fichier WSDL

- Coté client
  - Invocation du service Web = lecture du fichier WSDL
  - Interrogation du services = requêtes SOAP

Principe de fonctionnement



Vue en couches



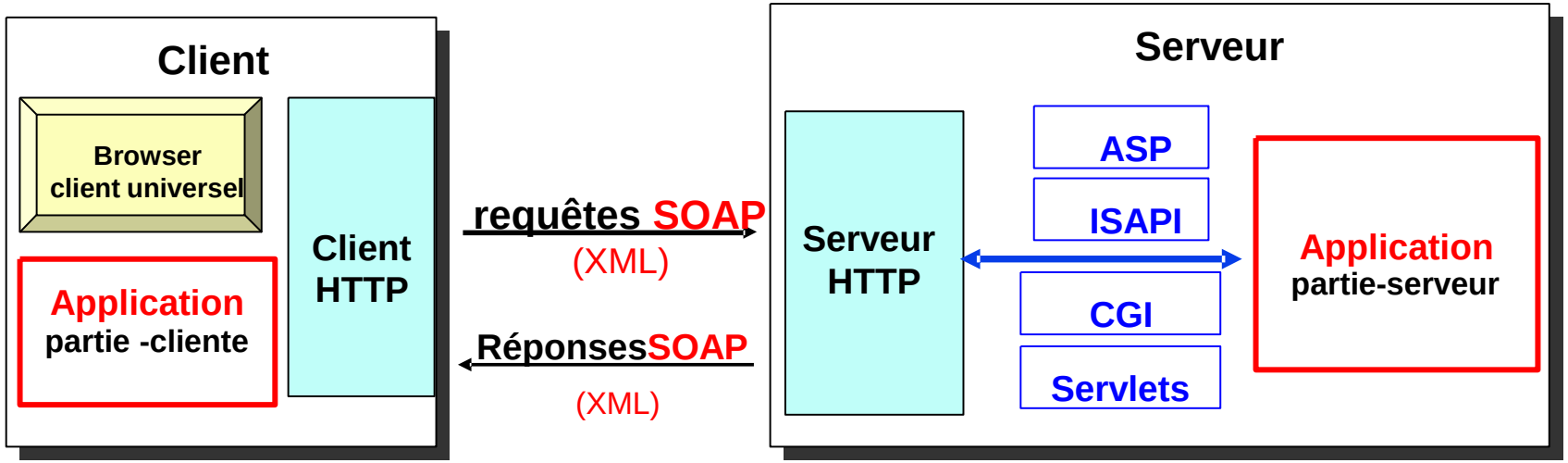
D'après M.Pontacq, Evidian

La philosophie SOAP

- Simple Object Access Protocol
- SOAP codifie l'utilisation conjointe de XML et HTTP
- SOAP est un protocole minimal pour appeler des méthodes sur des serveurs, services, composants, objets
  - Ne pas imposer une API
  - Ne pas imposer l'utilisation d'un ORB (CORBA, DCOM, …) ou d'un ou serveur web particulier (Apache, IIS, …)
  - Ne pas imposer un modèle de programmation
  - Ne pas réinventer une nouvelle technologie
- SOAP a été construit pour pouvoir être aisément porté sur toutes les plates-formes et les technologies
  - Vous pouvez écrire votre 1er appel SOAP en moins d'une heure !!

De manière simplifiée

- SOAP = HTTP + XML



Pourquoi utiliser HTTP ?

- Il est devenu de facto le protocole de communication d'Internet
- Il est disponible sur toutes les plates-formes
- C'est un protocole simple, qui ne requiert que peu de support pour fonctionner correctement
- HTTPS offre un niveau de sécurité simple et effectif
- C'est le seul protocole utilisable à travers des pare-feu

Rappel : requête/réponse en HTTP

HTTP Request

```
GET/bar/foo.txtHTTP/1.1
```

ou

```
POST/bar/foo.cgiHTTP/1.1
Content-Type: text/plain
Content-Length: 13

Goodbye, World
```

HTTP Response

```
200 OK
Content-Type: text/plain
Content-Length: 12

Hello, World
```

Pourquoi utiliser XML ?

- Texte interprétable
  - ATTENTION : le texte est globalement peu lisible et vite complexe pour un humain
- Construire correctement du texte XML est simple
  - Pas d'éléments qui se recouvrent (uniquement des imbrications)
  - Les attributs sont clairement identifiés
- XML permet d'ajouter du typage et de la structure à des informations
  - L'information peut être sauvegardée n'importe où sur le Net
  - Les données fournies par de multiples sources peuvent être agrégées en une seule unité
    - Chaque partie à sa propre structure XML
    - Chaque partie peut définir des types spécifiques

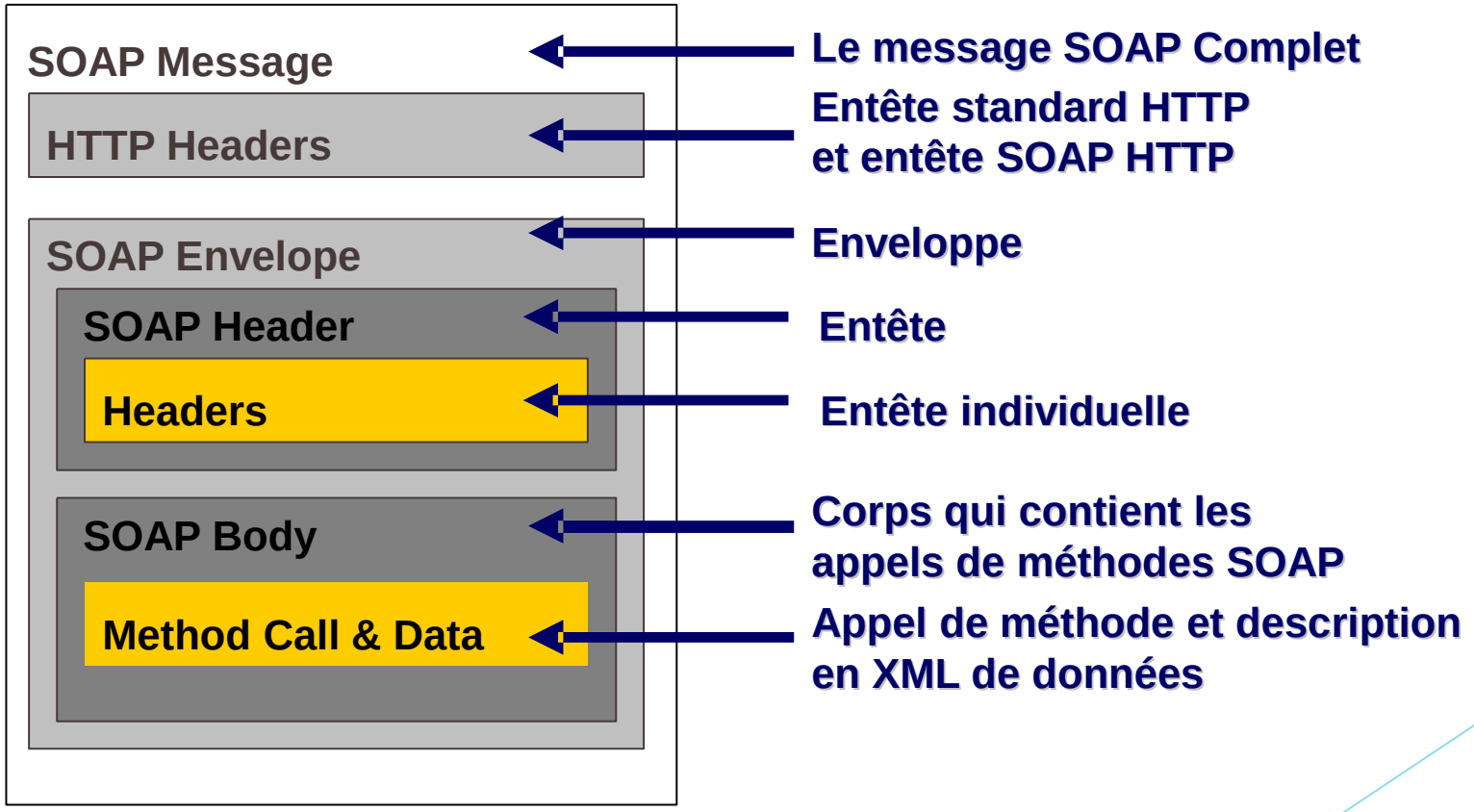
Exemple de requête HTTP/SOAP

```
POST /HTTP/1.1
Host: 192.168.2.1
Content-Type: text/xml; charset="utf-8"


<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:SayHelloWorldFrom>
            <arg>name</arg>
        </m:SayHelloWorldFrom >
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Exemple de réponse HTTP/SOAP

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"


<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:SayHelloWorldFrom>
            <arg>David</arg>
        </m:SayHelloWorldFrom >
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Structure d’un message SOAP



Règles d'encodage

Types primitifs

```
<elementname="price"type="float"/>
<elementname="greeting" type="xsd:string"/>
<price>15.57</price>
<greetingid="id1">Hello</greeting>
```

Structures

```
<elementname="Book"><complexType>
      <elementname="author"type="xsd:string"/>
      <elementname="title"type="xsd:string"/>
</complexType></element>
<e:Book>
<author>J.R.R Tolkien</author>
<title>A hobbitstory</title>
</e:Book>
```

Enumération

```
<elementname="color">
  <simpleTypebase="xsd:string">
    <enumerationvalue="Green"/>
    <enumerationvalue="Blue"/>
  </simpleType>
 </element>
 <color>Blue</color>
```

Tableaux

```
<SOAP-ENC:Arrayid="id3"SOAP-ENC:arrayType=xsd:string[2,2]>
     <item>r1c1</item>
     <item>r1c2</item>
     <item>r2c1</item>
     <item>r2c2</item>
</SOAP-ENC:Array>
```

Tableaux d'octet

```
<picturexsi:type="SOAP-ENC:base64">
     aG93IG5vDyBicm73biBjb3cNCg==
</picture>
```

Le retour d'erreurs (faults)

- 4 éléments
  - Faultcode(obligatoire)
    - Code d'erreur utilisé par le logiciel
  - Faultstring(obligatoire)
    - Explication lisible pour un humain
  - Faultactor(optionel)
    - Erreur en cours de cheminement du message (firewall, proxy, MOM)
  - Detail
    - Détail de l'erreur non lié au corps (body) du message

Exemple de retour d'erreur

```
HTTP/1.1 500 InternalServer Error
Content-Type: text/xml; charset="utf-8"


<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
       <SOAP-ENV:Fault>
           <faultcode>SOAP-ENV:Server</faultcode>
           <faultstring>Server Error</faultstring>
           <detail>
                <e:myfaultdetailsxmlns:e="Some-URI">
                 <message>Myapplication didn't
                          work</message>
                 <errorcode>1001</errorcode>
               </e:myfaultdetails>
            </detail>
       </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Portée de SOAP

- Ce que fait SOAP
  - Il permet de réaliser des appels de méthode sur le Web
  - Indépendant des OS, des modèles objets, des langages
  - Transport des messages par HTTP + XML
  - Fonctionne avec l'infrastructure Internet existante
- Ce que ne fait pas SOAP
  - Ce n'est pas un système réparti à objets
  - Pas de contrôle de types, pas de gestion de version
  - Pas de dialogue entre deux serveurs HTTP
  - Pas de passage d'objets par référence

Comparaison

	RMI	RPC	DCOM	CORBA	SOAP
Qui	SUN	SUN/OSF	MicroSoft	OMG	W3C
Plate-formes	Multi	Multi	Win32	Multi	Multi
Langages de Programmation	Java	C, C++, …	C++, VB, VJ OPascal, …	Multi	Multi
Langages de Définition de Service	Java	RPCGEN	ODL	IDL	XML
Réseau	TCP, HTTP, IIO customisable	TCP, UDP	IP/IPX	GIOP, IIOP, PluggableTransport Layer	RPC,HTTP SNMP
Firewall	Tunneling HTTP			HTTP Tunneling CORBA Firewall	HTTP
Nommage	RMI, JNDI,JINI	IP+Port	IP+Nom	COS Naming COS Trader	IP+Port, URL
Transaction	Non	Non	MTS	OTS, XA	Extension applicative dans le header
Extra	Chargement dynamique des classes			Services Commu Services Sectoriels	

WSDL – Web Service Description Language

- Spécification
  - Ariba, IBM, Microsoft (septembre 2000)
  - W3C v1.1 (mars 2001)
- Objectif
  - Décrire les services comme un ensemble d'opérations et de messages abstraits relié (*bind*) à des protocoles et des serveurs réseaux
- Grammaire XML (schemaXML)
  - Modulaire (import d'autres documents WSDL)

Éléments d'une définition WSDL

- <types>
  - Contient les définitions de types utilisant un système de typage
- <message>
  - Décrit les noms et types d'un ensemble de champs à transmettre, paramètres d'une invocation, valeur du retour, …
- <porttype>
  - Décrit un ensemble d'opérations. Chaque opération a zéro ou un message en entrée zéro ou plusieurs message de sortie ou de fautes
- <binding>
  - Spécifie une liaison d'un <porttype> à un protocole concret (SOAP1.1, HTTP1.1, …). Un <porttype> peut avoir plusieurs liaisons !
- <port>
  - Spécifie un point d'entrée (endpoint) comme la combinaison d'un <binding> et d'une adresse réseau
- <service>
  - Une collection de points d'entrée (endpoint) relatifs.

Élément <types>

- Contient les définition de types utilisant un système de typage

```
<!--type defs-->
<types>
  <xsd:schematargetNamespace="urn:xml-soap-address-demo"
                        xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <xsd:complexTypename="phone">
      <xsd:elementname="areaCode" type="xsd:int"/>
      <xsd:elementname="exchange" type="xsd:string"/>
      <xsd:elementname="number" type="xsd:string"/>
    </xsd:complexType>
    <xsd:complexTypename="address">
      <xsd:elementname="streetNum" type="xsd:int"/>
      <xsd:elementname="streetName" type="xsd:string"/>
      <xsd:elementname="city" type="xsd:string"/>
      <xsd:elementname="state" type="xsd:string"/>
      <xsd:elementname="zip" type="xsd:int"/>
      <xsd:elementname="phoneNumber" type="typens:phone"/>
    </xsd:complexType>
  </xsd:schema>
</types>
```

Élément <message>

- Décrit les noms et types d'un ensemble de champs à transmettre

```
<!--message declns-->
<message name="AddEntryRequest">
   <part name="name" type="xsd:string"/>
   <part name="address" type="typens:address"/>
</message>

<message name="GetAddressFromNameRequest">
   <part name="name" type="xsd:string"/>
</message>

<message name="GetAddressFromNameResponse">
   <part name="address" type="typens:address"/>
</message>
```

Références

- Services Web
  - Spécifications officielles : http://www.w3c.org
  - Documentations et exemples en ligne : http://www.xmlbus.com
- SOAP
  - Spécification SOAP 1.2 : http://www.w3c.org/TR/soap12
  - Exemplesen ligne : http://soapclient.com/soaptest.html
- UDDI
  - Spécification : http://www.uddi.org