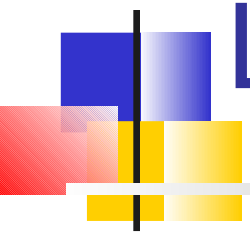
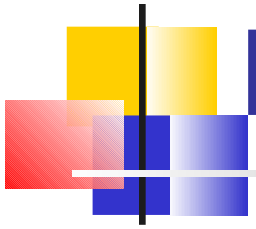


Les services Web

Jeremy Fierstone
Email : fierston@essi.fr

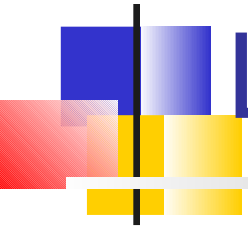
SAR5 – Novembre 2002

Merci à Mireille Blay-Fornarino, Didier Donsez
Michel Riveill, Microsoft, Sun … pour leurs slides



Les services Web

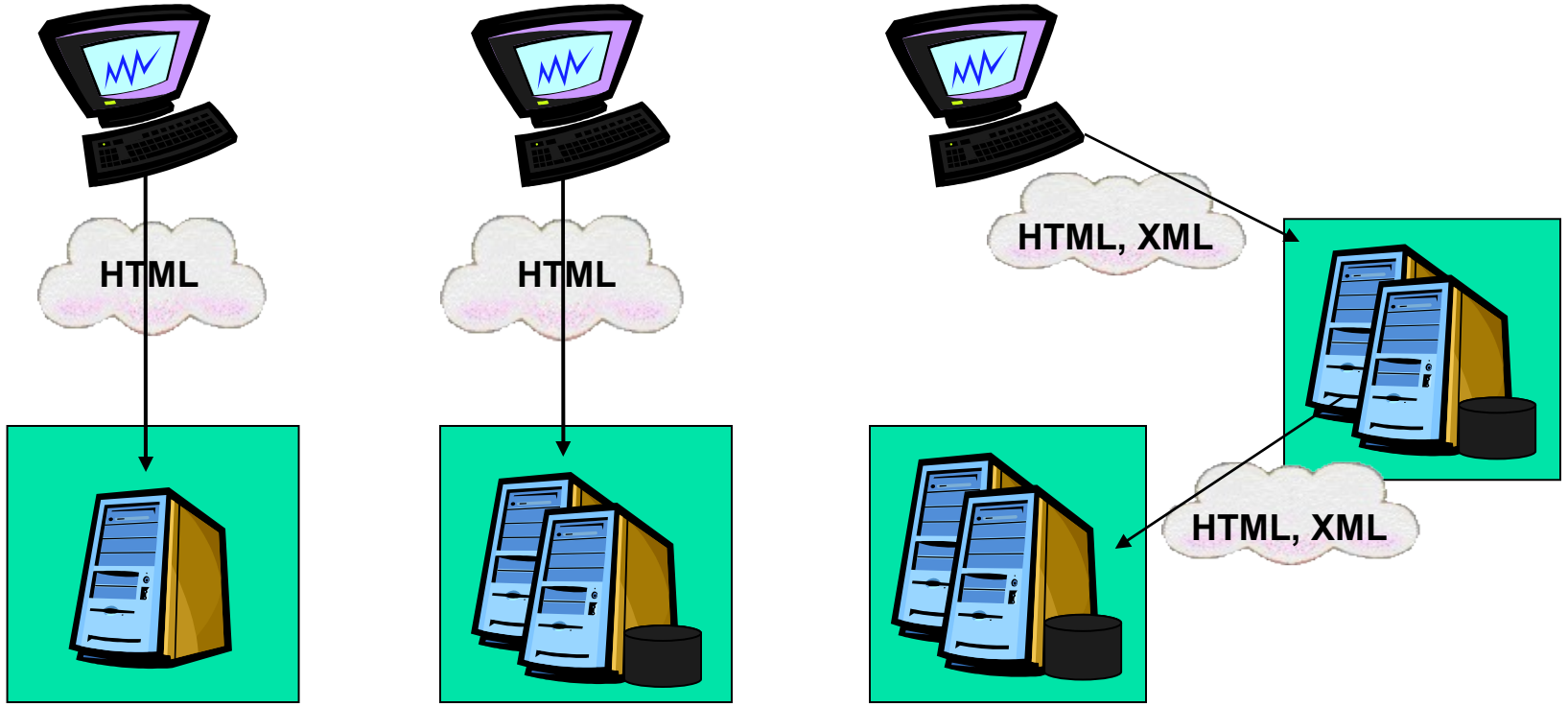
- Généralités
- Architecture
- SOAP
- WSDL
- UDDI
- Implémentations
  - Les APIs Java (JAXP, JAX-RPC, JAXM, JAXR, JAXB)
  - Implémentation avec JAX-RPC
  - Apache SOAP, Apache Axis
- Conclusion



Les services Web

Généralités

Evolution du Web



Generation 1
Static HTML

Generation 2
Web Applications

Generation 3
Web Services

Le Web 3ème génération

Aujourd'hui

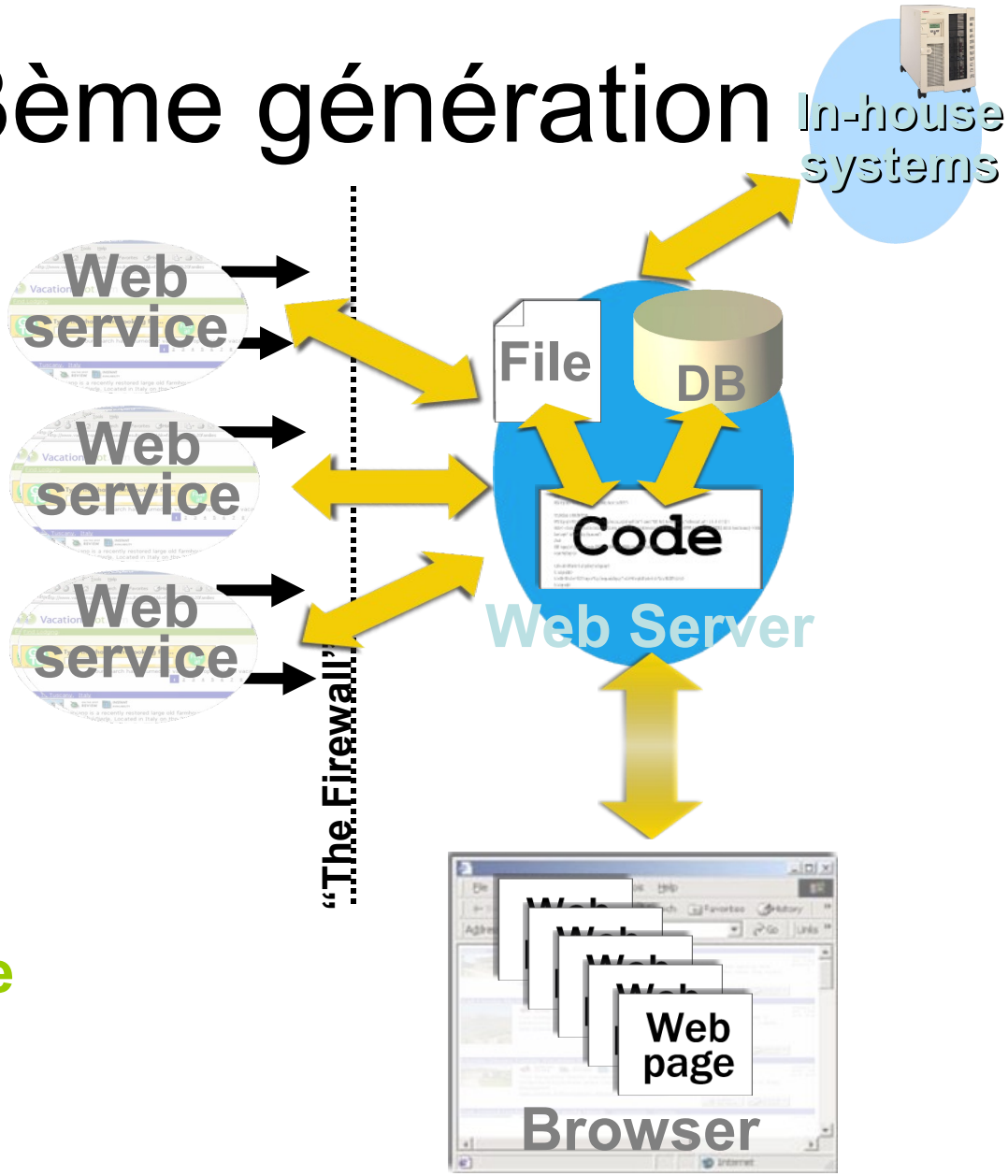
Un site Web fournie des pages HTML

- pas de structure
- impossible à fusionner avec d'autres pages

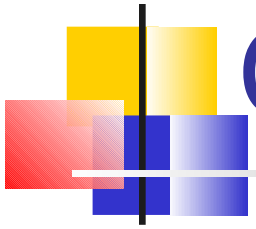
Demain

Un site Web est un composant fournissant des services en XML

- structure / sémantique
- fusion possible



"Dynamic Pages"

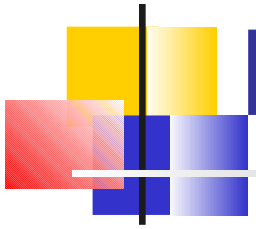


Quels objectifs ?

- Remplacer les protocoles actuels (RPC,DCOM,RMI) par une approche entièrement ouverte et interopérable, basée sur la généralisation des serveurs Web avec scripts CGI.

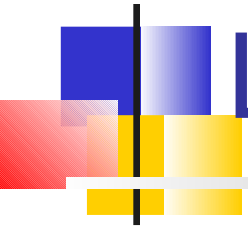
- Faire interagir des composants hétérogènes, distants, et indépendants avec un protocole standard (SOAP).

- Dédiés aux applications B2B (Business to Business), EAI (Enterprise Application Integration), P2P (Peer to Peer).



Et plus concrètement ?

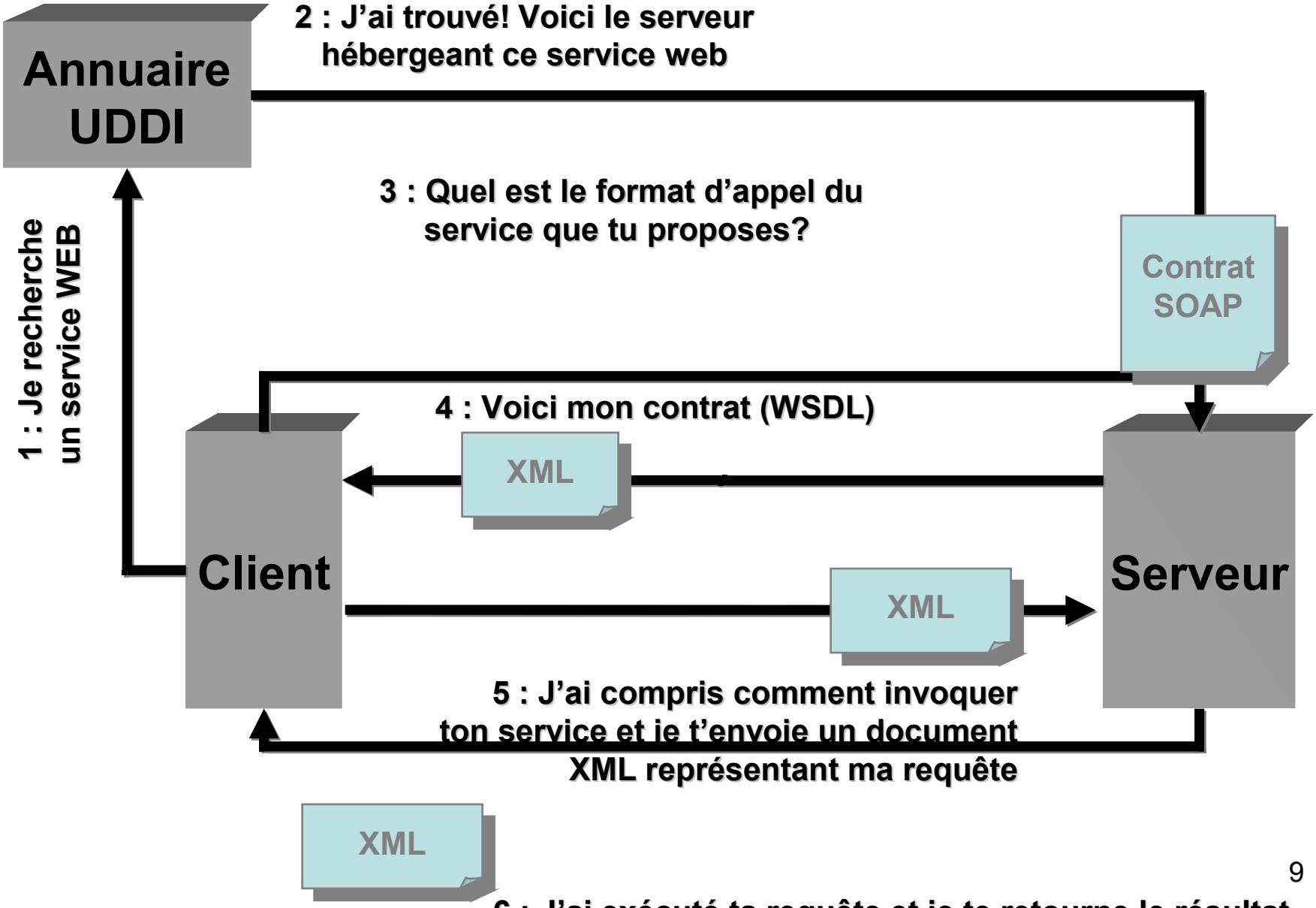
- Une nouvelle technologie des objets distribués ?
  - Invocation distante des services Web : **SOAP** (~IIOP)
  - Description des services Web : **WSDL** (~IDL)
  - Enregistrement et découverte de services Web : **UDDI** (~NameService)
- Basés sur des standards XML
  - Standards du W3C : XML, SOAP, WSDL
  - Standards industriels : UDDI, ebXML
  - Propriétaires : DISCO, WSDD, WSFL, ASMX, …
- Implémentations actuelles :
  - Microsoft .Net
  - Sun JavaONE : J2EE + Web services (WSDP = JAXP,JAX-RPC,JAXM…)
  - Apache SOAP / Axis, IBM WSTK
  - Oracle, Bea, Iona, Enhydra …

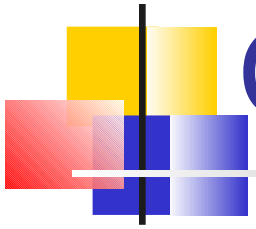


Les services Web

Architecture

Cycle de vie d'utilisation

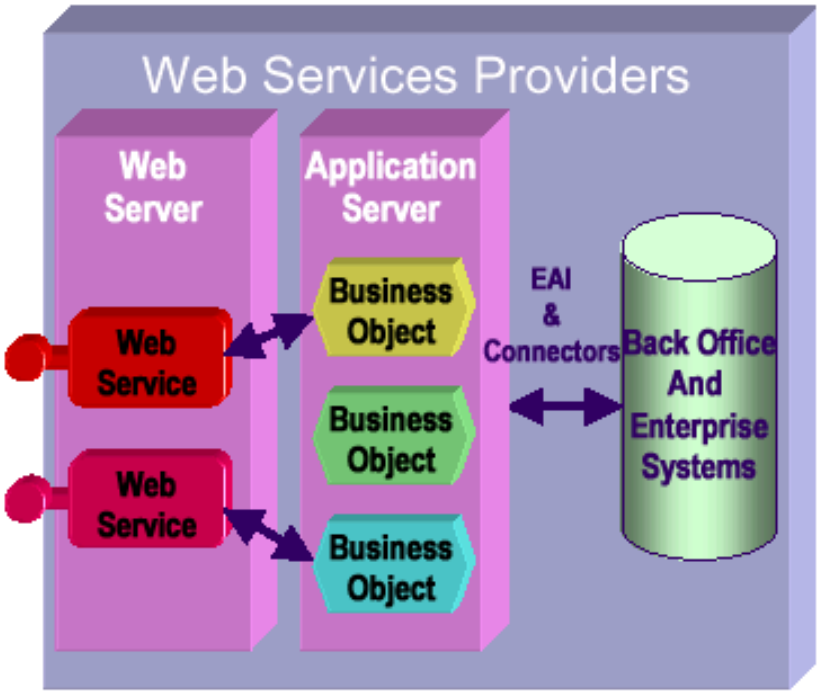




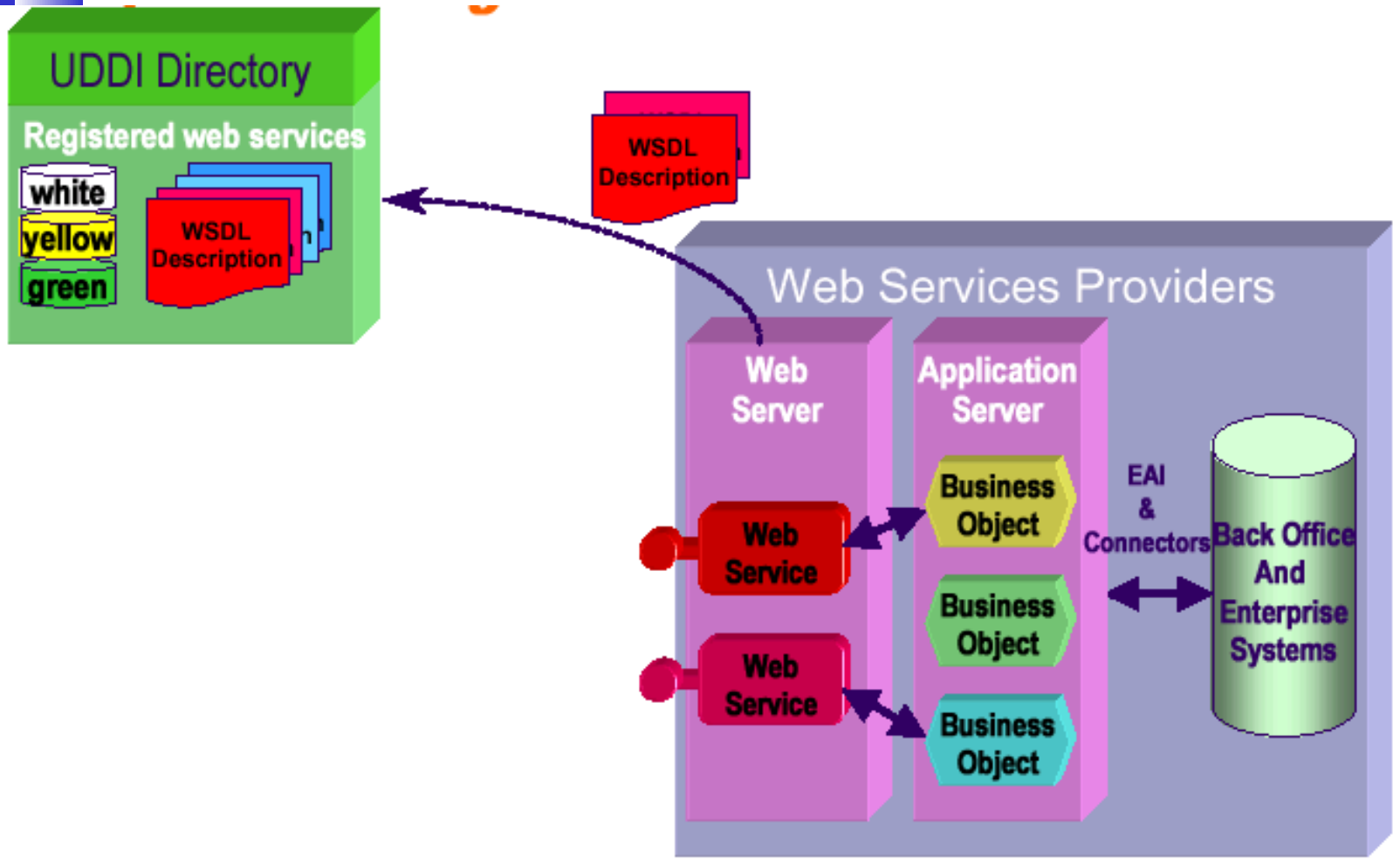
Cycle de vie complet

- Etape 1 : Déploiement du service Web
  - Dépendant de la plate-forme (Apache : WSDD)
- Etape 2 : Enregistrement du service Web
  - WSDL : description du service
  - Référentiels : DISCO (local), UDDI (global)
- Etape 3 : Découverte du service Web
- Etape 4 : Invocation du service Web par le client

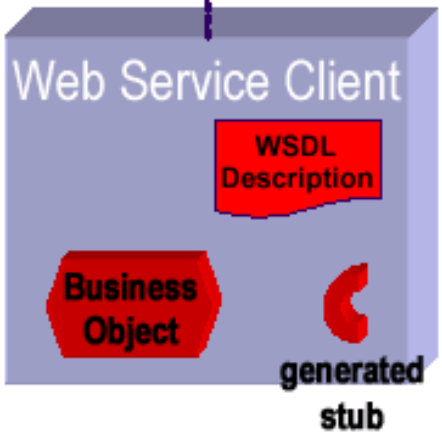
1: Déploiement du WS

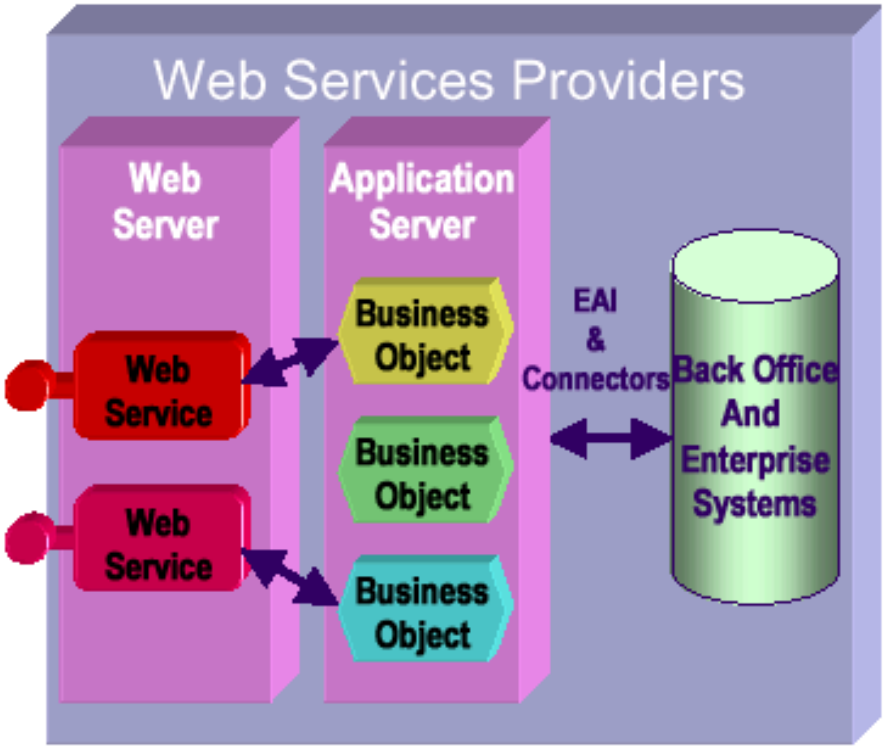


2: Enregistrement du WS

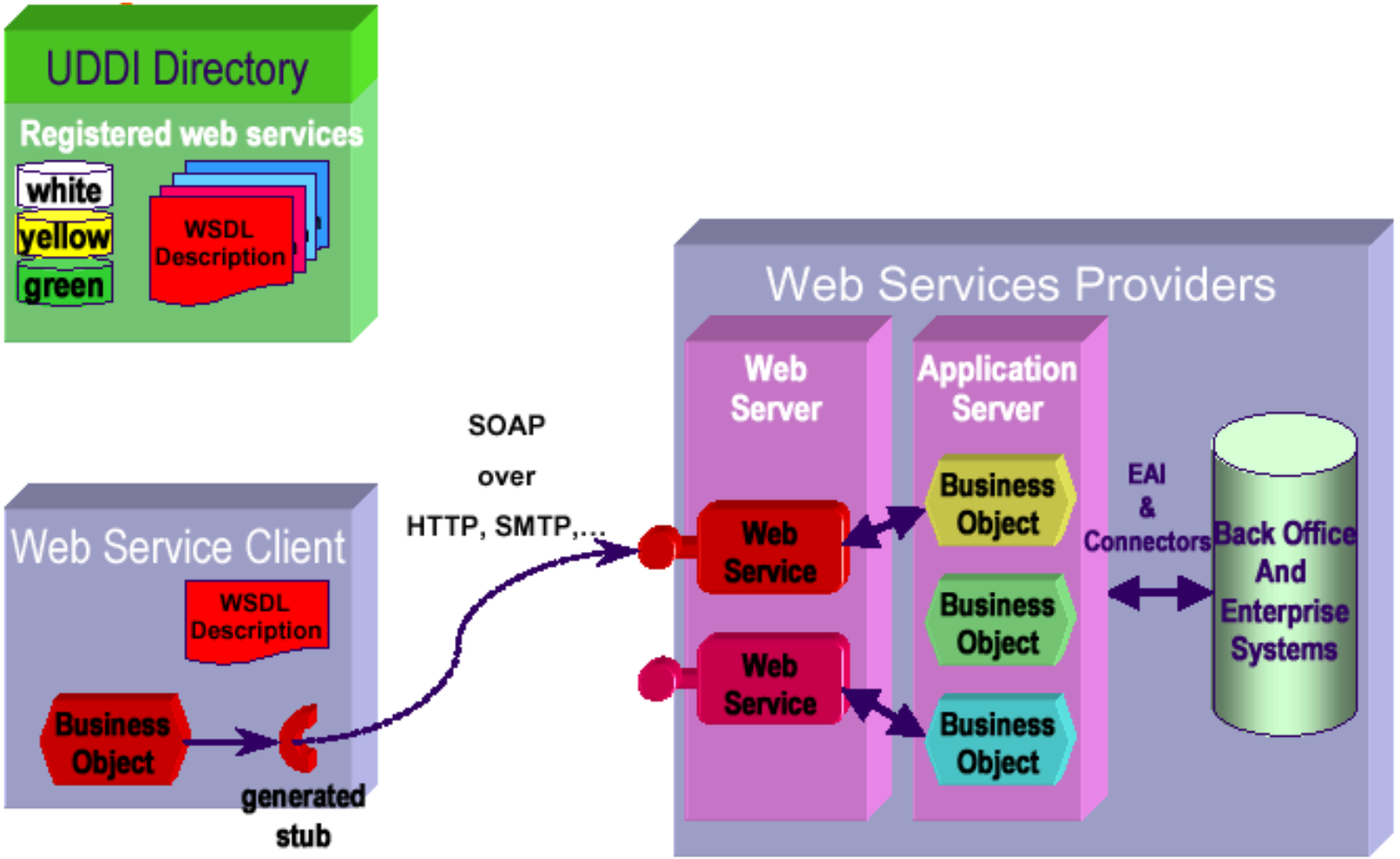


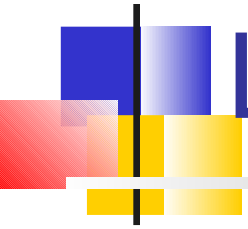
3: Découverte du WS





4: Invocation du WS

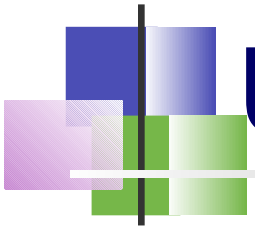




Les services Web

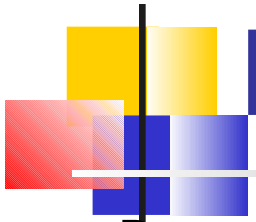
SOAP : Simple Object Access Protocol

Merci à Michel Riveill et Didier Donsez



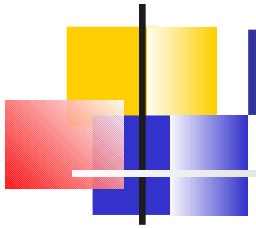
Un peu d'historique

- **Septembre 1999 : SOAP 0.9**
  - Spécifications par Microsoft et DevelopMentor
- **Décembre 1999 : SOAP 1.0**
  - Soumission des spécifications à l'IETF
  - Association de UserLand
- **Mai 2000 : SOAP 1.1 – Soumission au W3C**
  - Nombreuses associations : IBM, HP, Lotus, Compaq, Intel …
  - XIDL : rapprochement de Corba
- **Septembre 2000**
  - Groupe de travail W3C pour la standardisation de SOAP
  - Corba/Soap Interworking RFP => *SCOAP*



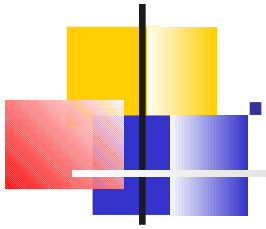
Le Web et le client serveur

- Proposition Web actuelle insuffisante
- Autres plates-formes client / serveur
  - Java RMI
    - Java, multi-plateforme (JVM)
  - CORBA / IIOP
    - Multilangage, multi-plateforme, Multi-vendeurs, OMG
    - Installation « coûteuse » si on doit acheter un ORB
      - Mais les open-sources sont gratuits et souvent plus complet
        - www.objectweb.org
  - DCOM
    - multi-langages, plateforme Win32, Propriétaire Microsoft
    - protocole orienté connexion
      - Échange de nombreux paquets pour créer/maintenir une session
    - Faible diffusion
      - Pas disponible sur MacOS, NT3.51, Win95, WinCE2
      - Coûteux sur UNIX, MVS, VMS ou NT



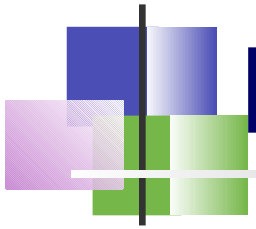
Le bilan…

- Approche insatisfaisante :
  - Protocoles sophistiqués
    - Coût d'installation (faite par un administrateur, consomme des ressources : machines, personnels, …)
    - Difficile à porter sur d'autres plates-formes
  - Règles de fonctionnement strictes en environnement ouvert (le Net)
    - Environnement sécurisé (intérieur d'un intranet)
    - Incapacité à fonctionner en présence de pare-feu (utilisation impossible sur Internet)
      - Les nouvelles version de CORBA peuvent ouvrir un port sur un pare-feu comme le port 80 d'HTTP



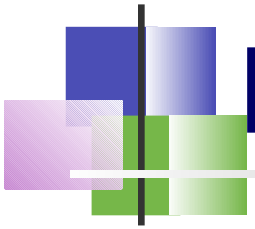
… et ses conséquences

- Le Web a besoin d'un nouveau 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 « firewalls »
  - Avec une spécification non propriétaire garantie par un organisme indépendant
    - W3C
- La réponse : SOAP (Simple Object Access Protocol)



La philosophie S.O.A.P

- SOAP codifie simplement une pratique existante
  - 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 ou un runtime
  - Ne pas imposer l'utilisation d'un ORB (CORBA, DCOM, …) ou d'un serveur web particulier (Apache, IIS, …)
  - Ne pas imposer un modèle de programmation
    - Plusieurs modèles peuvent être utilisés conjointement
  - Et "ne pas réinventer une nouvelle technologie"
- SOAP a été construit pour pouvoir être aisément porté sur toutes les plates-formes et les technologies

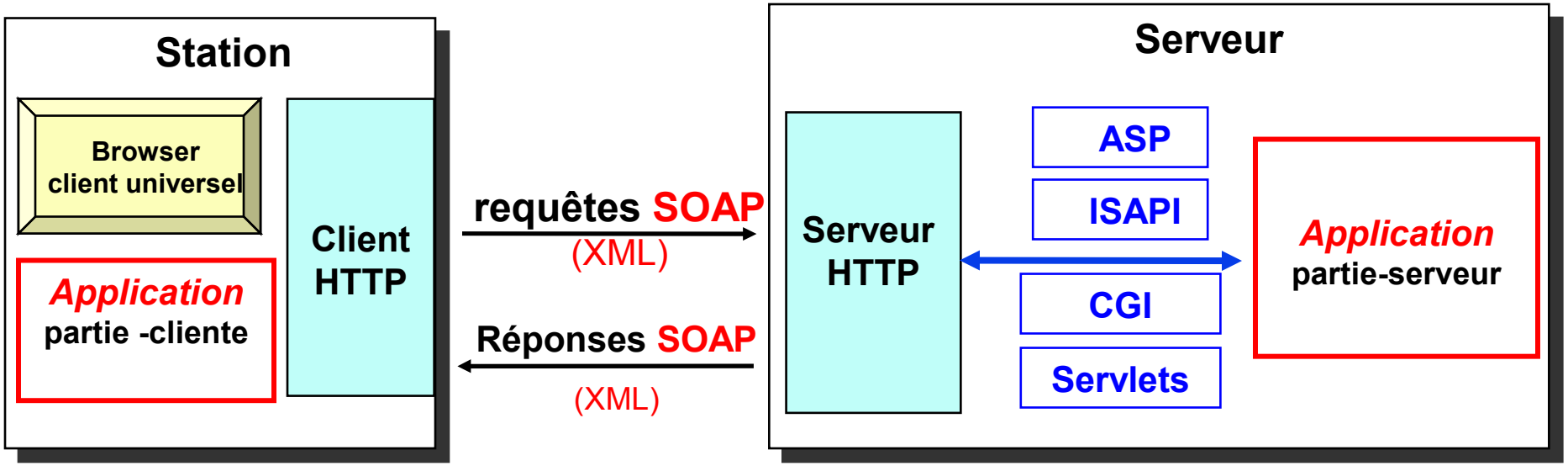


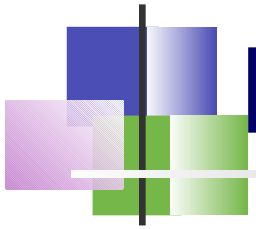
Les 3 aspects d'un appel SOAP

- SOAP peut être vu comme un autre RPC Objets
  - Les requêtes contiennent les paramètres IN et INOUT
  - Les réponses contiennent les paramètres INOUT et OUT
- SOAP peut être vu comme un protocole d'échange de "message"
  - La requête contient un seul message (appel sérialisé d'une méthode sur un objet)
  - La réponse contient un seul message (retour sérialisé d'un appel de méthode sur un objet)
- SOAP peut être vu comme un format d'échange de documents
  - La requête contient un document XML
  - Le serveur retourne une version transformée

En résumé

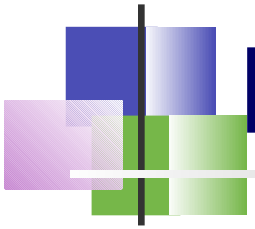
- SOAP = HTTP + XML





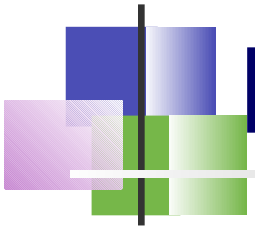
Pourquoi utiliser HTTP ?

- HTTP (HyperText Transfer Protocol) est devenu de facto le protocole de communication de l'Internet
- HTTP est disponible sur ***toutes*** les plates-formes – très rapidement
- HTTP est un protocole simple, qui ne requière que peu de support pour fonctionner correctement
- HTTP est un protocole sans connexion
  - Peu de paquets sont nécessaires pour échanger des informations
- HTTP est le seul protocole utilisable à travers des pare-feu



Fonctionnement d'HTTP

- HTTP utilise un protocole requête/réponse basé sur du texte
- La première ligne de la requête contient 3 éléments
  - Verbe : POST/GET/HEAD
  - URI : /default.htm
  - Protocole : HTTP/1.0 - HTTP/1.1
- La première ligne de la réponse contient 2 éléments
  - État : 200, 402
  - Phrase : OK, Unauthorized
- Les lignes suivantes contiennent un nombre arbitraire d'entête
- Le "contenu" suit une ligne d'entête vide
  - Utilisé essentiellement pour les réponses et pour les requêtes POST



Fonctionnement d'HTTP

HTTP Request

```
GET /bar/foo.txt HTTP/1.1
```

ou

```
POST /bar/foo.cgi HTTP/1.1
Content-Type: text/plain
Content-Length: 14

Goodbye, World
```

HTTP Response

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

Hello, World
```

Pourquoi utiliser XML ?

- Utilise du texte (peut être lu et écrit directement)
- Construire correctement du texte XML est simple
  - Pas d'éléments qui se recouvrent (uniquement des imbrications)
  - Les attributs sont clairement identifiés (dir="in")
  - Les caractères "<", ">", "&" doivent être précédés d'un caractère d'échappement (ou il faut utiliser CDATA)
- XML est aujourd'hui adopté par tous les acteurs de l'Internet : plates-formes, éditeurs, …
- XML permet une extensibilité aisée par l'utilisation d'espaces de nommage (namespaces et URIs)
- 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
- W3C n'impose pas un API mais en recommande un (DOM)
  - D'autres sont utilisés : SAX, strcat

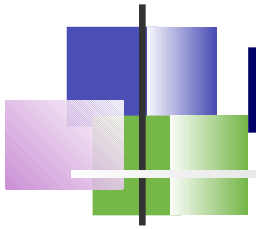
Exemple de requête utilisant HTTP

- Demande de cotation à un serveur

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<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:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Exemple de réponse utilisant HTTP

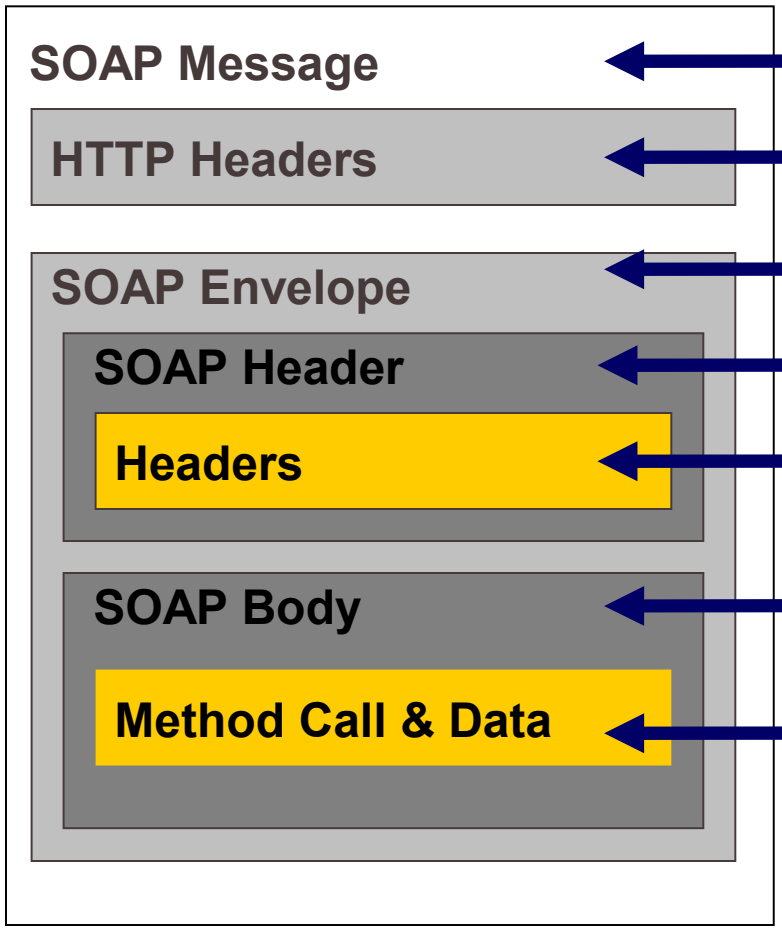
```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<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:GetLastTradePriceResponse
                 xmlns:m="Some-URI">
           <Price>34.5</Price>
       </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



Eléments de SOAP

- L'enveloppe (enveloppe)
  - Définit la structure du message
- Les règles d'encodage (encoding rules)
  - Définit le mécanisme de sérialisation permettant de construire le message pour chacun des types de données pouvant être échangés
- Fonctionnement en modèle client / serveur (RPC representation)
  - Définit comment sont représentés les appels de procédure et les réponses
- Proposer une mise en œuvre sur HTTP (HTTP Extension Framework)
  - RFC 2774
  - Définir l'échange de message SOAP sur HTTP

SOAP Message Structure



Le message SOAP Complet

Entête standard HTTP
et entête SOAP HTTP

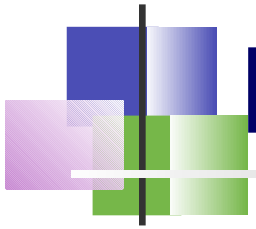
Enveloppe

Entête

Entête individuelle

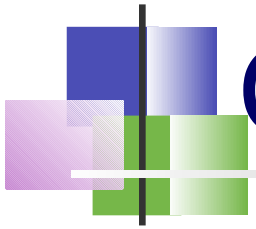
Corps qui contient les
appels de méthodes SOAP

Appel de méthode et description
en XML de données



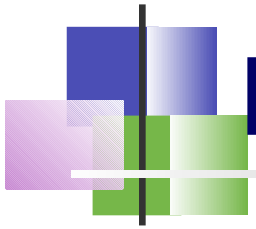
Modèle de message

- SOAP permet une communication par message
  - d'un expéditeur vers un récepteur
- Structure d'un message
  - Enveloppe / Envelope
    - Élément racine
    - Namespace :
      `SOAP-ENV` **`http://schemas.xmlsoap.org/soap/envelope/`**
  - Entête / Header
    - Élément optionnel
    - Contient des entrées non applicatives
      - Transactions, sessions, …
  - Corps / Body
    - Contient les entrées du message
      - Nom d'une procédure, valeurs des paramètres, valeur de retour
    - Peut contenir les éléments « fault » (erreurs)



Corps d'un Message

- Contient des entrées applicatives
- Encodage des entrées
- Namespace pour l'encodage
  - **`SOAP-ENC http://schemas.xmlsoap.org/soap/encoding/`**
  - xsd : XML Schema



Principes des règles d'encodage

- Les règles d'encodage définissent un système de type
  - Les types SOAP peuvent être décrit en utilisant XSD
  - SOAP utilise les conventions XSD pour associer les instances aux types
  - Les tableaux et les références sont typés de manière spécifique en utilisant XSD

Règles d'encodage

- **Types primitifs**

```
<element name="price"
type="float"/>
<element name="greeting"

type="xsd:string"/>
```

```
<price>15.57</price>
<greeting id="id1">Hello</greeting>
```

- **Structures**

```
<element name="Book"><complexType>
     <element name="author"
type="xsd:string"/>
     <element name="title"
type="xsd:string"/>
</complexType></element>
```

```
<e:Book>
  <author>J.R.R Tolkien</author>
  <title>A hobbit story</title>
</e:Book>
```

- **Enumération**

```
<element name="color">
  <simpleType base="xsd:string">
    <enumeration value="Green"/>
    <enumeration value="Blue"/>
  </simpleType>
 </element>
```

```
<color>Blue</color>
```

Règles d'encodage

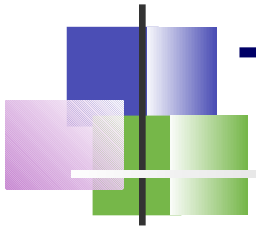
- **Références**

```
<element name="salutation" type="xsd:string"/>
<salutation href="#id1"/>

<e:Book>
   <title>My Life and Work</title>
   <firstauthor href="#Person-1"/>
   <secondauthor href="#Person-2"/>
</e:Book>
<e:Person id="Person-1">
   <name>Henry Ford</name>
   <address xsi:type="m:Electronic-address">
       <email>mailto:henryford@hotmail.com</email>
       <web>http://www.henryford.com</web>
   </address>
</e:Person>
<e:Person id="Person-2">
   <name>Samuel Crowther</name>
   <address xsi:type="n:Street-address">
       <street>Martin Luther King Rd</street>
       <city>Raleigh</city>
       <state>North Carolina</state>
   </address>
</e:Person>
```

Un exemple d'échange

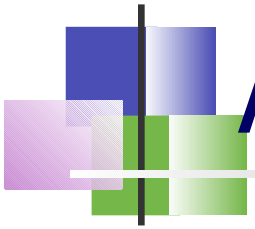
```
POST /path/foo.pl HTTP/1.1
Content-Type: text/xml
SOAPAction: interfaceURI#Add
Content-Length: nnnn

<soap:Envelope xmlns:soap='uri for soap'>
  <soap:Body>
    <Add xmlns='interfaceURI'>
      <arg1>24</arg1>
      <arg2>53.2</arg2>
    </Add>
  </soap:Body>
</soap:Envelope>
```

```
200 OK
Content-Type: text/xml
Content-Length: nnnn

<soap:Envelope
    xmlns:soap='uri for soap'>
  <soap:Body>
    <AddResponse xmlns='interfaceURI' >
      <sum>77.2</sum>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```



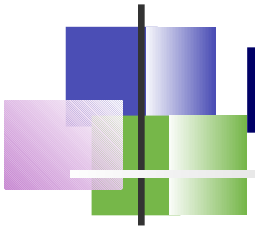
Types de message SOAP

- SOAP définit trois types de message
  - Appel (Call) - obligatoire
  - Réponse (Response) - optionnel
  - Erreur (Fault) - optionnel



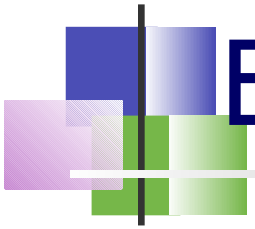
Appel simple

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml
Content-Length: nnnn
SOAPMethodName: Some-Namespace-URI#GetLastTradePrice

<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1”>
    <SOAP:Body>
        <m:GetLastTradePrice
          xmlns:m="Some-Namespace-URI”>
            <symbol>DIS</symbol>
        </m:GetLastTradePrice>
    </SOAP:Body>
</SOAP:Envelope>
```



Réponse

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1”>
    <SOAP:Body>
        <m:GetLastTradePriceResponse
          xmlns:m="Some-Namespace-URI”>
            <return>34.5</return>
        </m:GetLastTradePriceResponse>
    </SOAP:Body>
</SOAP:Envelope>
```

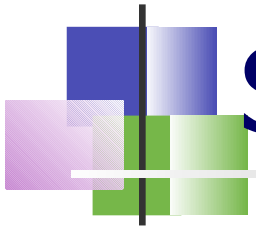


Erreur

```
<SOAP:Envelope
  xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1>
  <SOAP:Body>
    <SOAP:Fault>
      <faultcode>200</faultcode>
      <faultstring>
        SOAP Must Understand Error
      </faultstring>
      <runcode>1</runcode>
    </SOAP:Fault>
   <SOAP:Body>
</SOAP:Envelope>
```

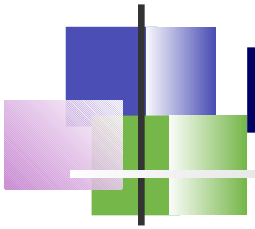
Autres éléments de SOAP sur HTTP

- Le type MIME d'une requête SOAP est text/xml
- Toutes les requêtes SOAP doivent pouvoir être reconnues comme telles par un serveur HTTP
  - Utilisation d'un entête HTTP spécifique
  - SOAPAction: interfaceURI#methodname
- Les erreurs HTTP utilisent l'infrastructure HTTP
- Les erreurs SOAP/app utilisent les éléments SOAP PDU
  - Modèle standard pour toutes les erreurs
  - Extensible pour prendre en compte les exceptions



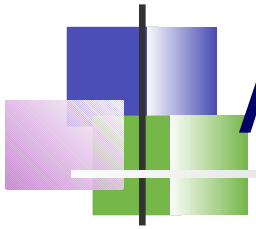
Sécurité

- Basé sur la sécurité dans http
  - HTTPS
  - Certificats X.509
- Les Firewalls peuvent filtrer les messages facilement
- Pas de transfert de code applicatif
  - Uniquement des données
- Les paramètres sont typés lors du transport



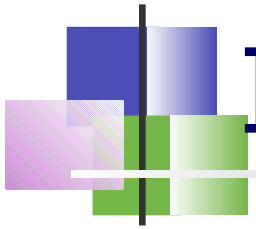
Portée de SOAP

- SOAP est simple et extensible
  - 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 on the wire
  - Fonctionne avec l'infrastructure Internet existante
  - Permet l'interopérabilité entre OS, langages et modèles objets
- Ce n'est pas un système réparti à objets
  Il ne couvre donc pas les fonctions suivantes :
  - Pas de ramassage des miettes
  - Pas de contrôle de types, pas de gestion de version
  - Pas de dialogue entre deux serveurs HTTP
  - Pas de passage d'objets par référence
    - Nécessite ramassage des miettes en réparti et HTTP bi-directionnel
  - Pas d'activation
    - Nécessite passage d'objets par référence



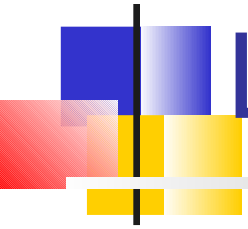
Autres Extensions

- Transport
  - SOAP sur SMTP/FTP/POP3/IMAP4/RMI-IIOP
    - Voir implémentation IBM/Apache
  - SOAP sur MOM (JMS)
- Encodage
  - XMI (UML)
    - Voir implémentation IBM/Apache
  - Litteral XML
    - DOM org.w3c.dom.Element sérialisé
    - Voir implémentation IBM/Apache



Implémentation de SOAP

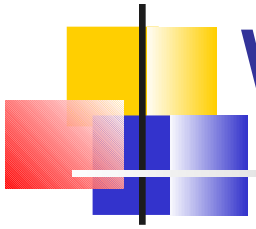
- On peut installer SOAP dans un ORB
  - Nouveau, Orbix 2000, Voyager, COM
- On peut installer SOAP dans un serveur Web
  - Apache, ASP/ISAPI, JSP/Servlets/WebSphere



Les services Web

WSDL : Web Services Description Language

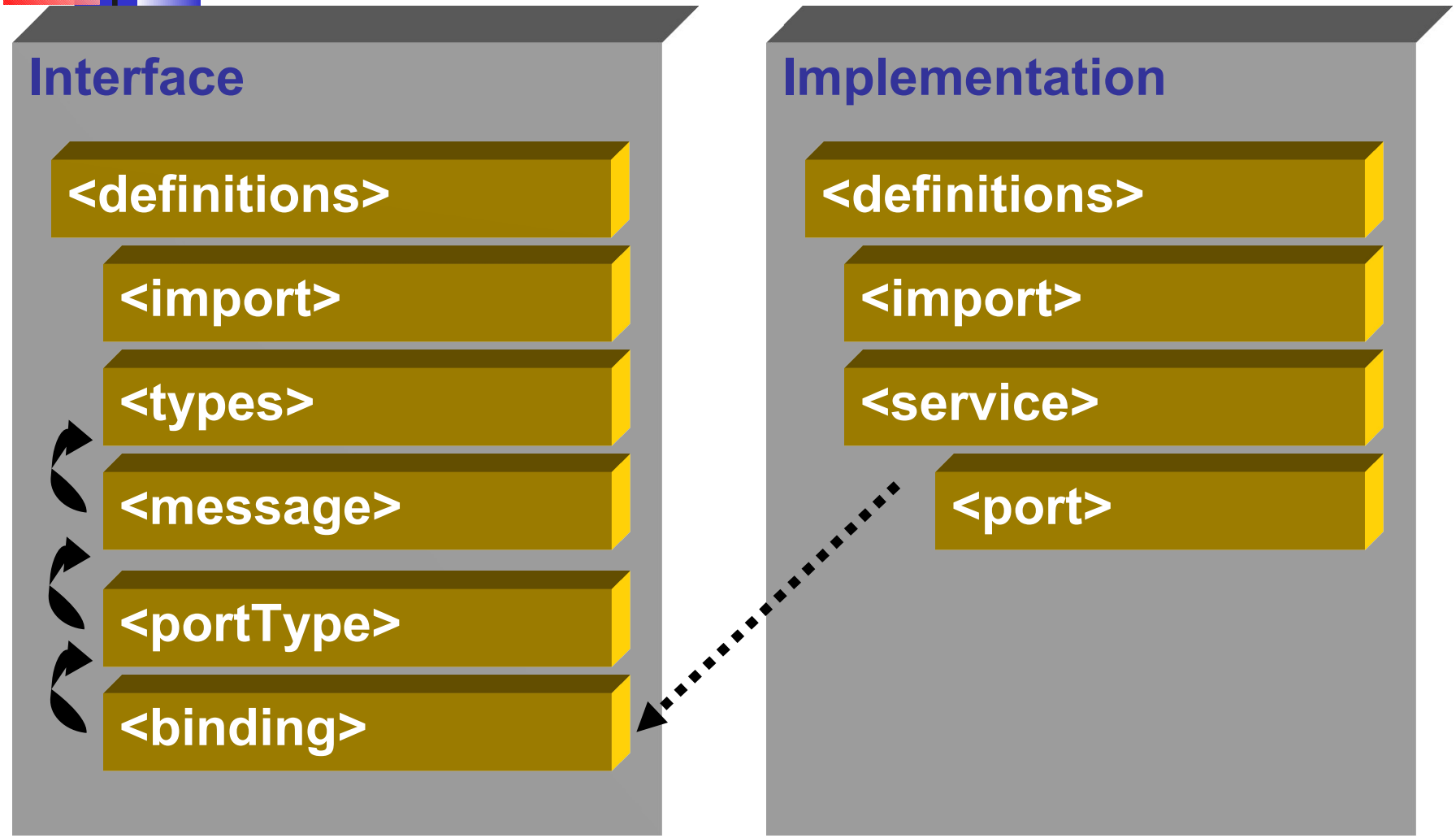
Merci à Didier Donsez

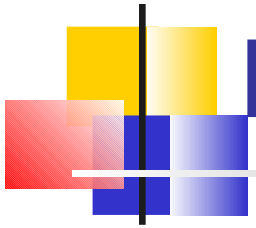


WSDL

- Spécification (09/2000)
  - Ariba, IBM, Microsoft
  - TR W3C v1.1 (25/03/2001)
- Objectif
  - Décrire les services comme un ensemble d'opérations et de messages abstraits relié (*bind*) à des protocoles et des serveurs réseaux
- Grammaire XML (schema XML)
  - Modulaire (*import* d'autres documents WSDL et XSD)
- Séparation entre la partie abstraite et concrète

WSDL


Interface
<definitions>
<import>
<types>
<message>
<portType>
<binding>
Implementation
<definitions>
<import>
<service>
<port>



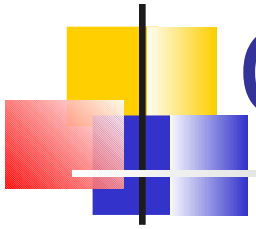
Éléments d'une définition WSDL

- <types>
  - Contient les définitions de types utilisant un système de typage (comme XSD).
- <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, MIME, …). 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>

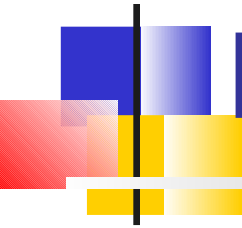
- Contient les définition de types utilisant un système de typage (comme XSD).
- **Exemple**

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



Outils

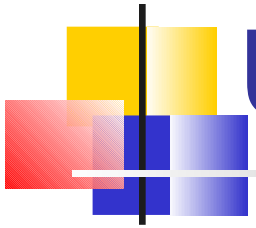
- Générateur WSDL à partir de déploiement SOAP ou EJB, …
- Générateur de proxy SOAP à partir de WSDL
- Toolkits (Wsdl2Java / Java2Wsdl, …)
  - Propriétaires (non normalisés)



Les services Web

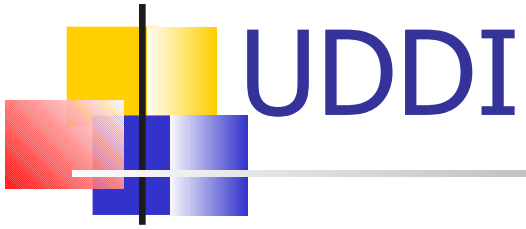
UDDI :

Universal Description, Discovery and Integration



UDDI

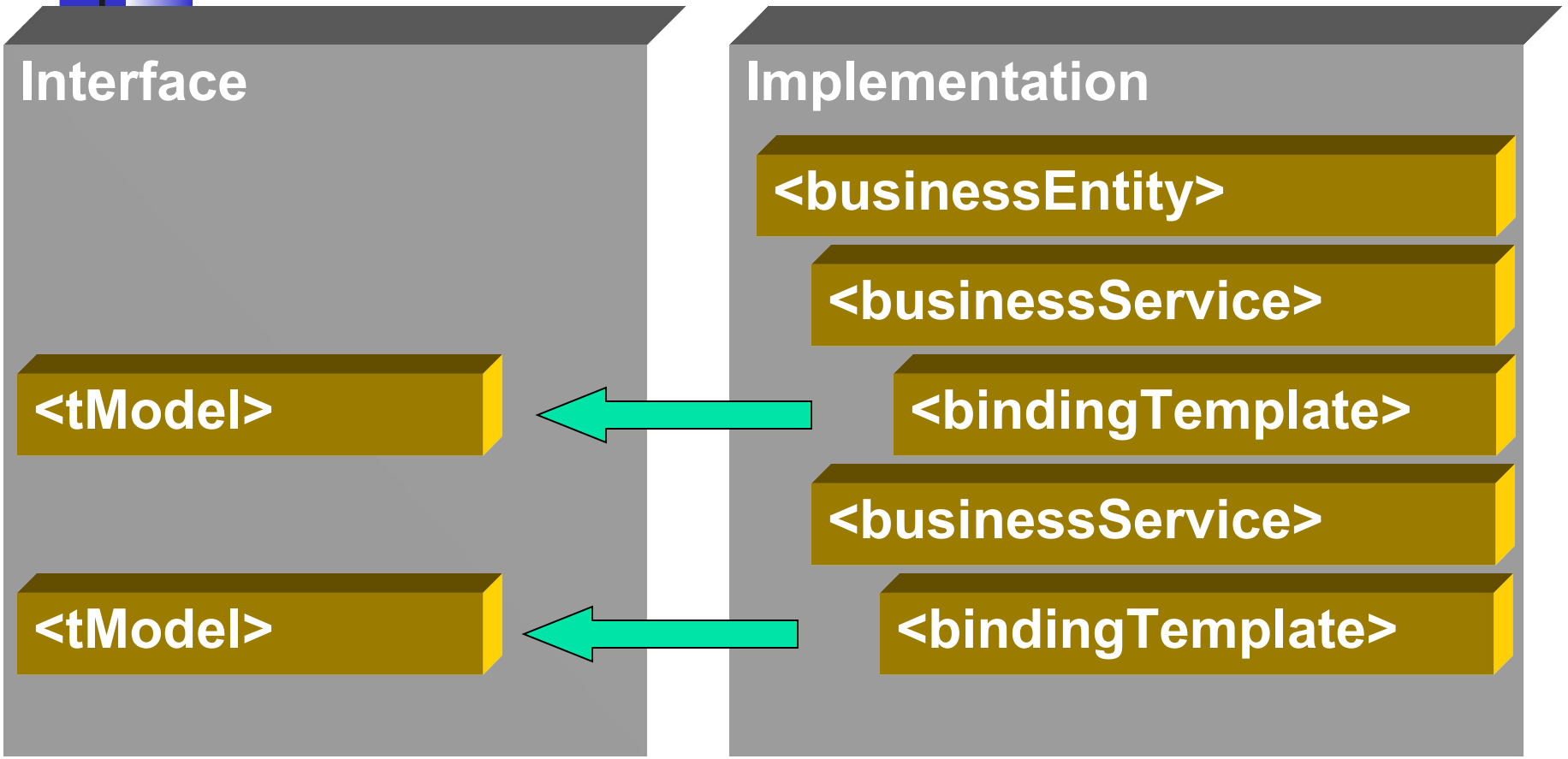
- Spécification (09/2000)
  - Ariba, IBM, Microsoft +260 autres sociétés
- Objectifs
  - annuaire mondial d'entreprises pour permettre d'automatiser les communications entre prestataires, clients, etc.
  - plusieurs entrées indexées : nom, carte d'identité des sociétés, description des produits, services applicatifs invocables à distance (références des connexions)
    - Indexation des catalogues propriétaires (ebXML, RosettaNet, Ariba, Commerce One, etc.)
- Grammaire XML (schéma XML)
  - Soumission/interrogation basées sur SOAP et WSDL



UDDI

- **Worldwide directory of companies, services, products...**
  - White pages,Yellow pages,Green pages
- **Green pages**
  - Namespace to describe how to use the service, etc...
  - Identifier of who published the service
  - Unique identifier (tModelKey) of this service for registration
- **Accessing web services**
  - Bindings declared in directory entries:
    - for example, (tModelKey, URL) associations
- **UDDI directories, search engines**
  - xmethods.net, soapware.org, salcentral.com, soap-wrc.com, …

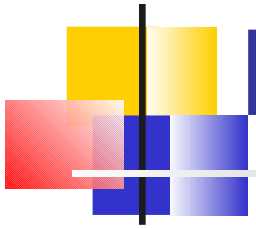
UDDI Schema


Interface
Implementation
<businessEntity>
<businessService>
<tModel>
<bindingTemplate>
<businessService>
<tModel>
<bindingTemplate>

UDDI : <tModel>

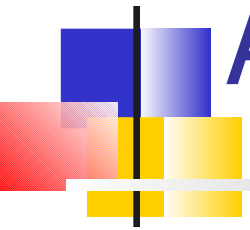
- <tModel> represents meta-data and interfaces

```
<tModel xmlns="urn:uddi-org:api" tModelKey="UUID:AAAAAAAA-AAAA-
   AAAA-AAAA-AAAAAAAAAAAA">
  <name>microsoft-com:creditcheck</name>
  <description xml:lang="en">Check credit limits</description>
  <overviewDoc>
    <overviewURL>http://schema.com/creditcheck.wsdl
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
      tModelKey="UUID:CD153257-086A-4237-B336-6BDCBDCC6634"
      keyName="Consumer credit gathering or reporting services"
      keyValue="84.14.16.01.00"/>
    <keyedReference
      tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62AB4"
      keyName="types"  keyValue="wsdlSpec"/>
  </categoryBag>
</tModel>
```



Références

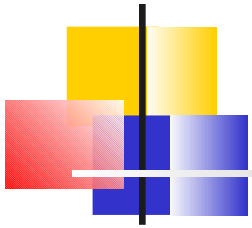
- Services Web :
  - Spécifications officielles : http://www.w3c.org
  - Documentations et exemples en ligne : http://www.xmlbus.com
- SOAP :
  - Spécification SOAP 1.1 : http://www.w3c.org/TR/SOAP
  - Spécification SOAP 1.2 : http://www.w3c.org/TR/soap12
  - Implémentations : http://www.soapware.org
  - Exemples en ligne : http://soapclient.com/soaptest.html
- UDDI :
  - Spécification : http://www.uddi.org
  - Serveur UDDI : http://uddi.microsoft.com et http://uddi.ibm.com
- Cours :
  - Didier Donsez, Web Services
  - Michel Riveill, SOAP



Un environnement pour WS : Axis

Jean Marc Farinone
Email : farinone@cnam.fr

Un environnement pour des WS : Axis



- Page d'accueil : `http://ws.apache.org/axis/`
- Cliquer sur les premiers liens pour charger une version d'Axis (1.4 le 28 mars 2008). On récupère un `.zip` à extraire.
- Axis est un environnement de développement pour WS (classes pour faire des services et clients WS) mais aussi une application Web.
- Voir tutorial d'axis à `http://ws.apache.org/axis/java/user-guide.html`

Développement d'un client et serveur WS

- Le serveur sera encapsulé dans l'appli web Axis.
- Le client est un programme Java
- Un serveur complet : `Reponse.jws`

```
public class Reponse {
  public String reponds(String st) {
    return "bonjour " + st;
  }

  public String disCoucou() {
    return "Coucou ";
  }
}
```

- Ben oui c'est une classe Java. Le fichier doit avoir pour extension `.jws`

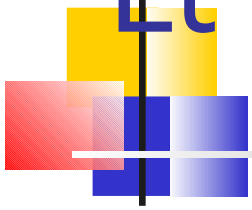
Le serveur WS `Reponse.jws`

- Il est placé dans l'application web Axis (sous `<racineDeWebappAxis>/axis/Reponse.jws`)
- Qui, à la première utilisation, va :
  - compiler ce fichier Java,
  - exécuter la méthode appelée et
  - retourner le résultat
- Une manière d'accéder à ce Web Service et, dans un client web, de demander la page d'URL : `http://localhost:8080/axis/Reponse.jws`

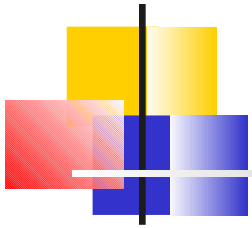
Le client WS `HelloClient.java`

```
package hello;

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;
import javax.xml.rpc.ParameterMode;

public class HelloClient
{
   public static void main(String [] args) throws Exception {
        Options options = new Options(args);
        String endpoint = "http://localhost:" + options.getPort() +
                          "/axis/Reponse.jws";
        args = options.getRemainingArgs();
        if (args == null || args.length != 1) {
            System.err.println("Usage: java HelloClient [nom]");
            return;
        }
        String nom = args[0];
        Service  service = new Service();
        Call     call    = (Call) service.createCall();
        call.setTargetEndpointAddress( new java.net.URL(endpoint) );
        call.setOperationName( "reponds" );
        call.addParameter( "st", XMLType.XSD_STRING, ParameterMode.IN );
        call.setReturnType( XMLType.XSD_STRING );
        String ret = (String) call.invoke( new Object [] { nom });

        System.out.println("resultat retourné  : \n" + ret);
           }
}
```

Et XML (SOAP) dans tout çà ?



- Nous n'avons fait que (!) de la programmation Java.
- Quid de SOAP ?
- En fait les messages échangés entre le client et le serveur WS sont en SOAP.
- Si, si.



TCPMonitor : `tcpmon`

- Axis propose un outil qui permet de voir les messages TCP échangés entre un client et un serveur : TCPMonitor
- Il suffit d'indiquer que le client envoit les messages à ... TCPMonitor, que TCPMonitor envoit les messages au serveur, et tout échange entre client et serveur est tracé dans cet outil.
- On lance cet outil par :

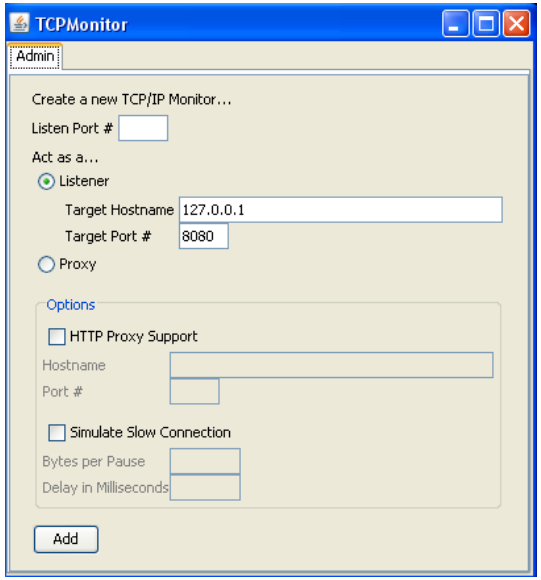
```
java org.apache.axis.utils.tcpmon
```

Utilisation de TCPMonitor 1/4

- Le client est programmé par :

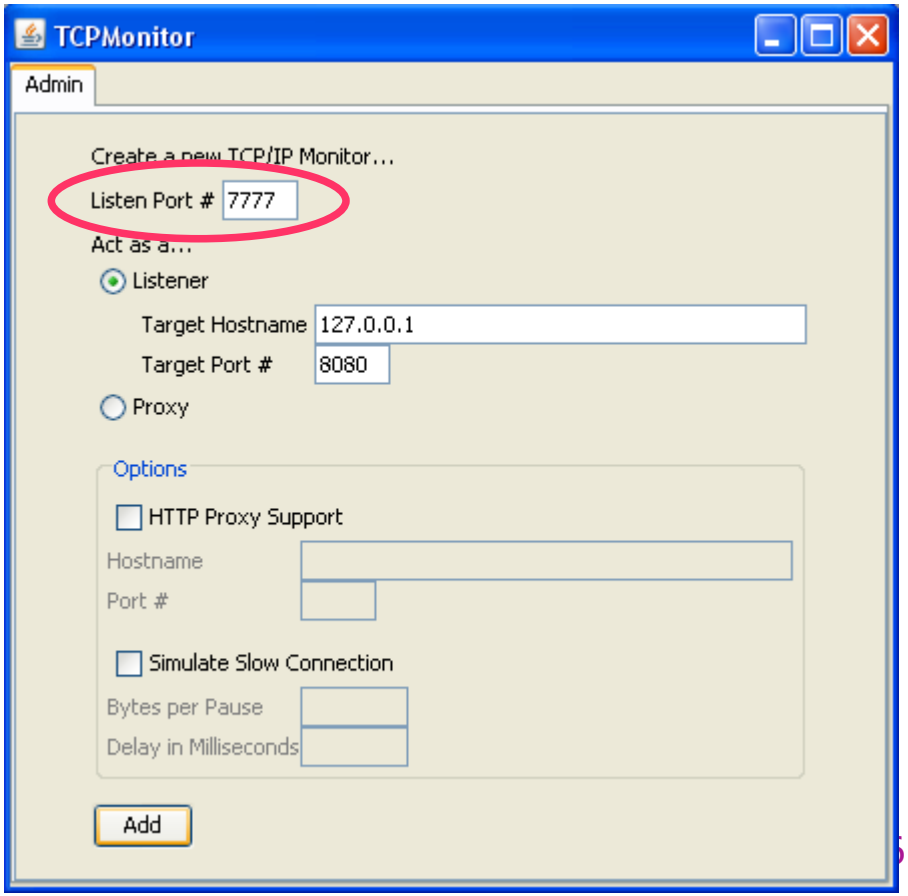
```
String endpoint = "http://localhost:7777" + "/axis/Reponse.jws";
...
Service  service2 = new Service();
Call call2    = (Call) service2.createCall();
call2.setTargetEndpointAddress( new java.net.URL(endpoint) );
....
```

- Lorsque TCPMonitor est lancé on obtient le premier écran :



Utilisation de TCPMonitor 2/4

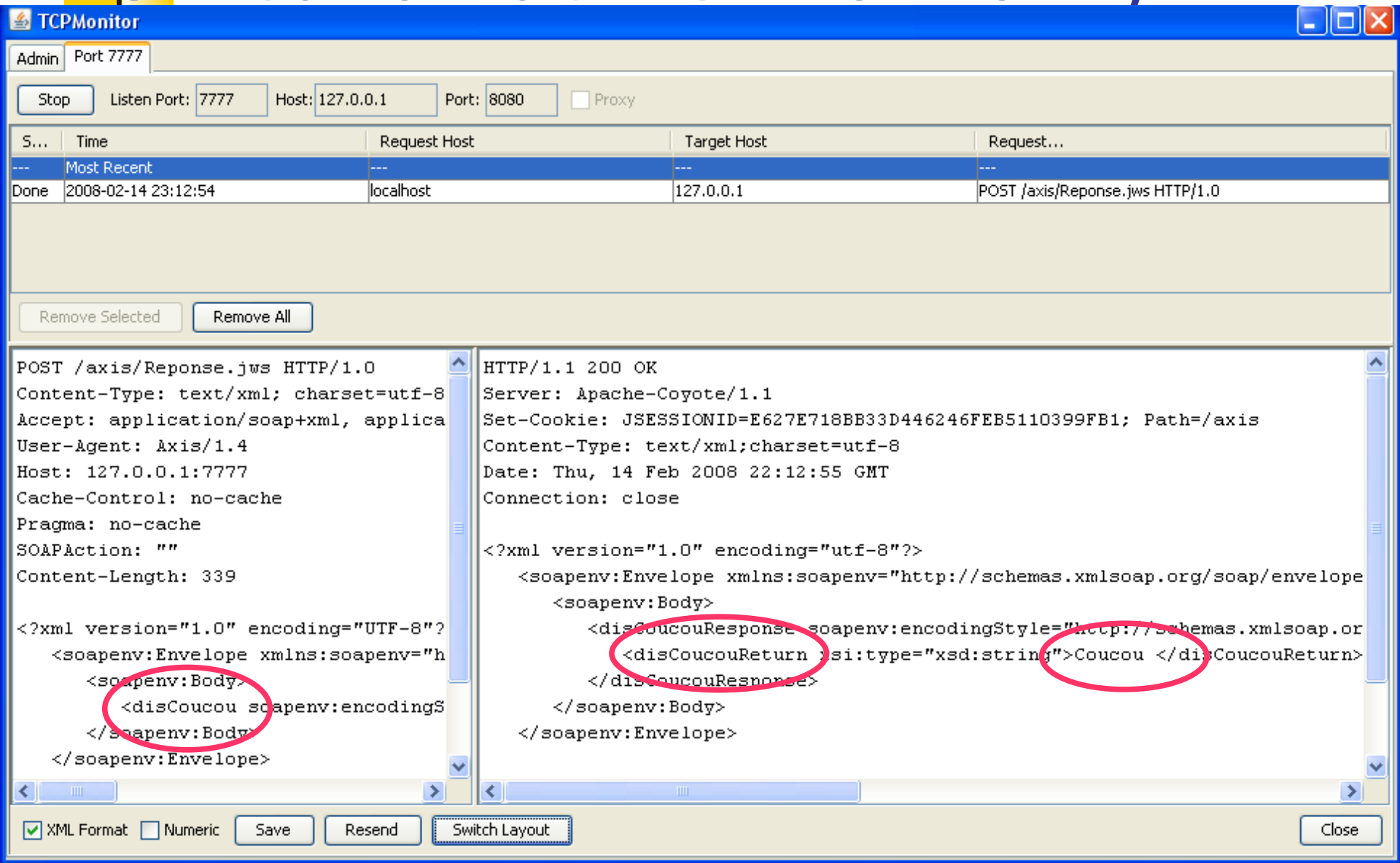
- Indiqué le port écouté dans le champ de texte qui suit dans "Listen Port #". Ici 7777.
- Et cliquer le bouton Add



Utilisation de TCPMonitor 3/4

- Dans le nouvel onglet obtenu, cocher "XML Format" et cliquer "Switch Layout".
- Puis lancer le client.
- Les messages véhiculés entre client et serveur WS seront affichés.

Utilisation de TCPMonitor 4/4



Bibliographie

- http://fr.wikipedia.org/wiki/Service_Oriented_Architecture
- http://java.sun.com/javaee/5/docs/tutorial/doc/bnayk.html : la partie du Java EE 5 tutorial consacrée aux services web
- http://www.w3.org/2002/ws/desc/ : page d'accueil du "Web Services Description Working Group"
- http://ws.apache.org/axis/ : implémentation open source Axis pour construire des services web