Services Web

Fabrice Rossi

http://apiacoa.org/contact.html.

Université Paris-IX Dauphine

Plan du cours

1. Introduction
2. SOAP
3. WSDL
4. UDDI

Site du cours : `http://apiacoa.org/teaching/webservices/`

Plan de l'introduction

1. Qu'est-ce qu'un service web ?
2. Architecture des services web
3. Les évolutions futures

Un service web ?

Définition du W3 (`http://www.w3.org/TR/ws-gloss/`) :

> Un service web est un système logiciel identifié par un URI, dont les interfaces publiques et les incarnations sont définies et décrites en XML. Sa définition peut être découverte [dynamiquement] par d'autres systèmes logiciels. Ces autres systèmes peuvent ensuite interagir avec le service web d'une façon décrite par sa définition, en utilisant des messages XML transportés par des protocoles Internet.

incarnation : ma traduction de *binding*

Traduction (1)

- **système logiciel** : programme
- **URI** : Uniform Resource Identifier (RFC 2396, `http://www.ietf.org/rfc/rfc2396.txt?number=2396`). Exemples :
  - URL : `http://www.w3.org/`
  - mail : `mailto:getlost@nospam.org`
  - FTP : `ftp://ftp.ufrmd.dauphine.fr/pub/docs/`
- **interface** : une description des opérations proposées par le composant logiciel (même esprit que les interfaces Java)
- **incarnation** (*binding*) : spécification du protocole et du format des données utilisés pour échanger des messages en vue de l'utilisation d'une interface
- **découverte** (dynamique) : obtention de la description d'un service web

Traduction (2)

- **XML** : eXtensible Markup Langage (`http://www.w3.org/XML/`) :
  - ASCII du XXIème siècle !
  - **pré-requis indispensable** pour faire des services web
  - publicité : `http://apiacoa.org/teaching/xml/`
- **protocoles internet** :
  - bas niveau TCP/IP
  - haut niveau (applicatif) :
    - HTTP (web)
    - SMTP (mail)
    - FTP (file)
    - etc.

Traduction (3)

Un service web est donc :

- un programme
  - décrit en XML
  - identifié par un URI
- proposant diverses fonctionnalités que d'autres programmes peuvent
  - découvrir dynamiquement
  - et utiliser grâce à des protocoles
    - décrits en XML
    - basés sur l'échange de messages
      - écrits en XML
      - transmis par HTTP, FTP, SMTP, etc.

Une vision plus simple

Un service web est

- un programme accessible par internet
- par l'intermédiaire de messages XML
- transmis par HTTP

Un standard pour les systèmes répartis basés sur XML+HTTP : XML-RPC (cf `http://www.xmlrpc.org/`)

Lien avec l'existant

- Architectures pour les systèmes répartis (RPC, DCOM, RMI et CORBA) :
  - même idées générales (interface, découverte dynamique, protocoles, etc.)
  - problèmes d'inter-opérabilité :
    - DCOM spécifique Microsoft
    - RMI spécifique Java
  - problèmes techniques :
    - RPC technologie vieillissante
    - CORBA techonologie extrêmement complexe
  - publicité :

    `http://apiacoa.org/teaching/distributed/`
- Applications web (sites évolués) :
  - ça fonctionne mais...
  - problèmes de sécurité, fiabilité, etc.

But des services web

Fournir une architecture générale pour les applications réparties sur internet :

- inter-opérables :
  - basé sur des standards ouverts
  - sans composant spécifique à un langage ou un système d'exploitation
- faiblement couplées :
  - limiter au maximum les contraintes imposées sur le modèle de programmation des différents éléments de l'application
  - par exemple ne pas imposer un modèle objet
- supportant la montée en charge : par exemple en n'imposant pas un modèle de type RPC
- etc.

Exemples de services existants

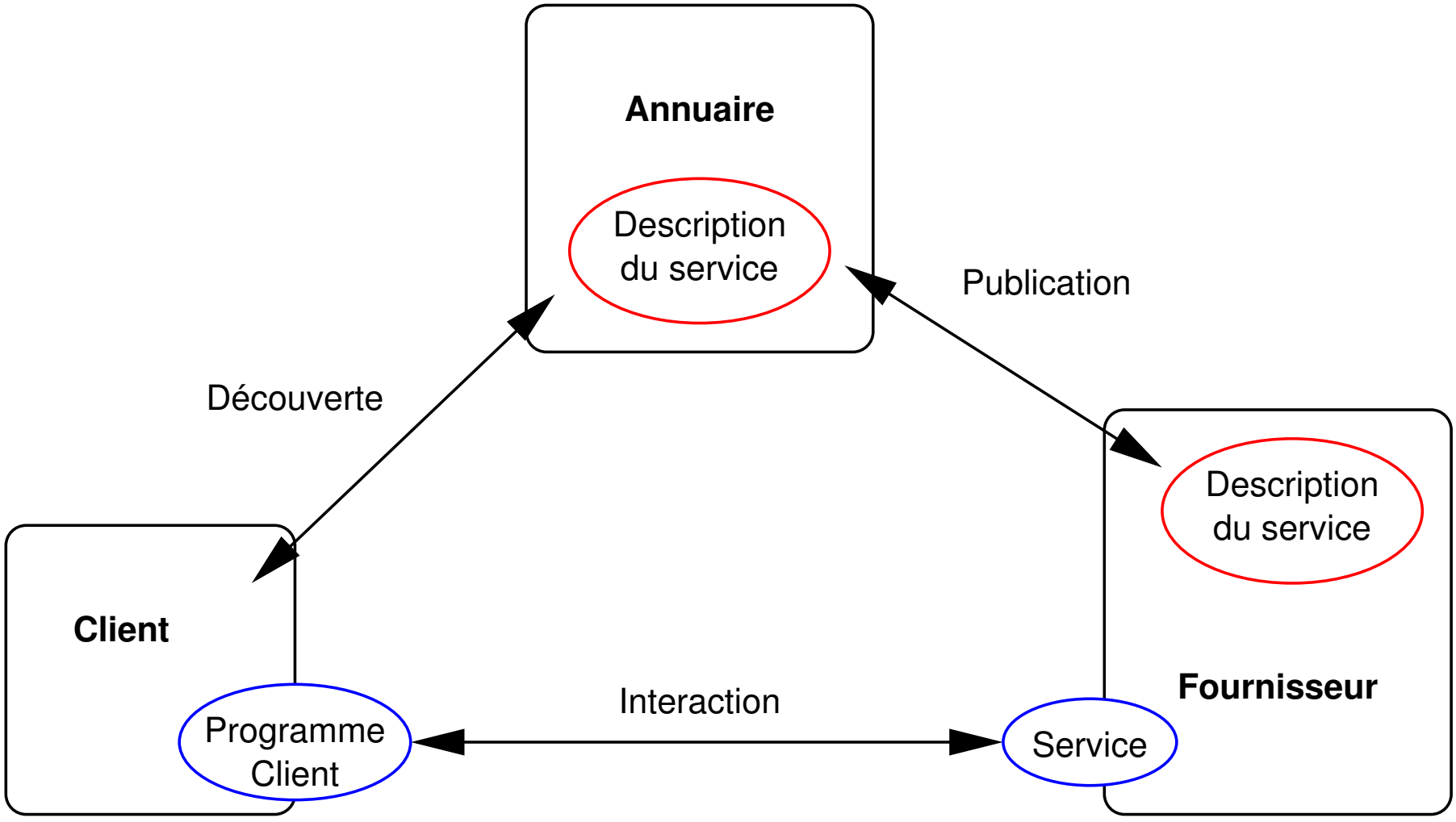
- Google (`http://www.google.com/apis/`) :
  - accès gratuit mais limité (1000 requêtes par jour après enregistrement)
  - propose trois opérations :
    - recherche
    - obtention d'une page depuis le cache
    - correction orthographique
- Amazon
  (`http://associates.amazon.com/exec/panama/associates/join/developer/resources.`
  - accès gratuit mais limité (1 requête par seconde après enregistrement)
  - propose recherche et gestion d'un panier d'achats
- bien d'autres ! (cf `http://www.xmethods.com/` par exemple)

Architecture de base

Trois acteurs :

- le fournisseur de service (*service provider*) :
  - définit le service
  - publie sa description dans l'annuaire
  - réalise les opérations
- l'annuaire (*discovery agency*) :
  - reçoit et enregistre les descriptions de services publiées par les fournisseurs
  - reçoit et répond aux recherches de services lancées par les clients
- le client (*service requestor*) :
  - obtient la description du service grâce à l'annuaire
  - utilise le service

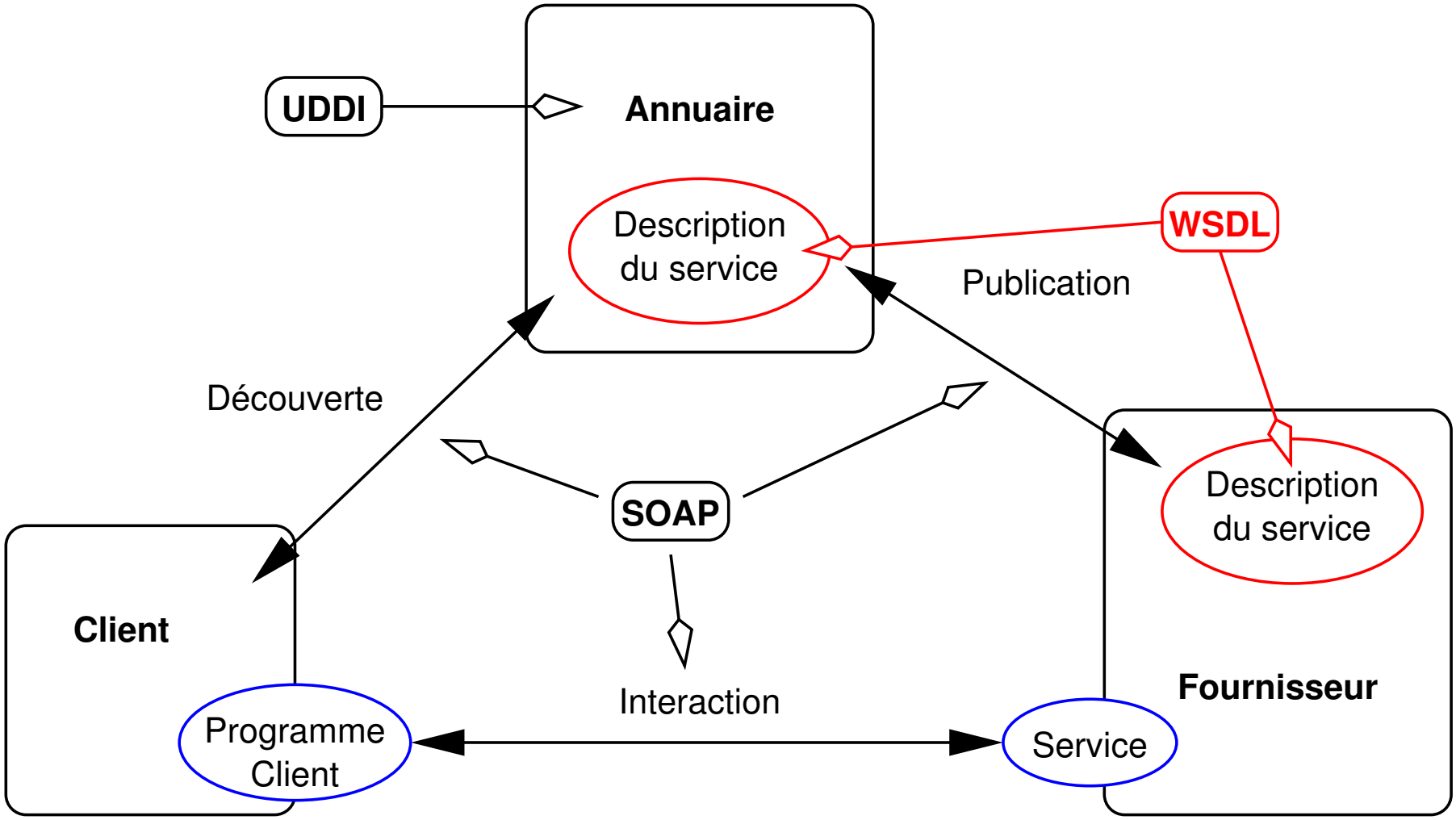
Architecture de base (2)



Briques de l'architecture de base

- **SOAP** (Simple Object Access Protocol) :
  - version 1.1 : mai 2000 (`http://www.w3c.org/TR/SOAP/`)
  - version 1.2 : en cours (CR en décembre 2002)
  - cadre général permettant l'échange de données structurées au format XML
  - protocole de transport de ces données basé sur HTTP
- **WSDL** (Web Services Description Language) :
  - version 1.1 : mars 2001 (`http://www.w3.org/TR/2001/NOTE-wsdl-20010315`)
  - version 1.2 : en cours (Working Draft)
  - dialecte XML permettant de décrire un service web
- **UDDI** (Universal Data Description Interface) :
  - version 3 : juillet 2002 (`http://uddi.org/pubs/uddi-v3.00-published-20020719.htm`)
  - annuaire permettant d'enregistrer et de rechercher des descriptions de services web

Architecture de base (3)



Exemple SOAP

doGoogleSearch.xml

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
3     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
4     xmlns:xsd="http://www.w3.org/1999/XMLSchema">
5    <env:Body>
6      <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch"
7          env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
8        <key xsi:type="xsd:string">00000000000000000000000000000000</key>
9        <q xsi:type="xsd:string">shrdlu winograd maclisp teletype</q>
10       <start xsi:type="xsd:int">0</start>
11       <maxResults xsi:type="xsd:int">10</maxResults>
12       <filter xsi:type="xsd:boolean">true</filter>
13       <restrict xsi:type="xsd:string"></restrict>
14       <safeSearch xsi:type="xsd:boolean">false</safeSearch>
15       <lr xsi:type="xsd:string"></lr>
16       <ie xsi:type="xsd:string">latin1</ie>
17       <oe xsi:type="xsd:string">latin1</oe>
18     </ns1:doGoogleSearch>
19   </env:Body>
20 </SOAP-ENV:Envelope>
```

Exemple WSDL

ITempConverter.xml

```
1  <?xml version="1.0"?>
2  <definitions
3     xmlns="http://schemas.xmlsoap.org/wsdl/"
4     xmlns:xs="http://www.w3.org/2001/XMLSchema"
5     name="ITempConverterservice"
6     targetNamespace="http://www.borland.com/soapServices/"
7     xmlns:tns="http://www.borland.com/soapServices/"
8     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
9     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
10    <message name="CtoFRequest">
11      <part name="temp" type="xs:int"/>
12    </message>
13    <message name="CtoFResponse">
14      <part name="return" type="xs:int"/>
15    </message>
16    <message name="FtoCRequest">
17      <part name="temp" type="xs:int"/>
18    </message>
```

Exemple WSDL (2)

ITempConverter.xml

```
19  <message name="FtoCResponse">
20    <part name="return" type="xs:int"/>
21  </message>
22  <portType name="ITempConverter">
23    <operation name="CtoF">
24      <input message="tns:CtoFRequest"/>
25      <output message="tns:CtoFResponse"/>
26    </operation>
27    <operation name="FtoC">
28      <input message="tns:FtoCRequest"/>
29      <output message="tns:FtoCResponse"/>
30    </operation>
31  </portType>
32  <binding name="ITempConverterbinding" type="tns:ITempConverter">
33    <soap:binding
34      style="rpc"
35      transport="http://schemas.xmlsoap.org/soap/http"/>
36    <operation name="CtoF">
```

Exemple WSDL (3)

ITempConverter.xml

```
37      <soap:operation soapAction="urn:TempConverterIntf-ITempConverter#CtoF"/>
38      <input>
39        <soap:body use="encoded"
40          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
41          namespace="urn:TempConverterIntf-ITempConverter"/>
42      </input>
43      <output>
44        <soap:body use="encoded"
45           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
46           namespace="urn:TempConverterIntf-ITempConverter"/>
47      </output>
48    </operation>
49    <operation name="FtoC">
50      <soap:operation soapAction="urn:TempConverterIntf-ITempConverter#FtoC"/>
51      <input>
52        <soap:body use="encoded"
53          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
54          namespace="urn:TempConverterIntf-ITempConverter"/>
55      </input>
```

Exemple WSDL (4)

ITempConverter.xml

```
56        <output>
57          <soap:body use="encoded"
58             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
59             namespace="urn:TempConverterIntf-ITempConverter"/>
60        </output>
61      </operation>
62    </binding>
63    <service name="ITempConverterservice">
64      <port name="ITempConverterPort" binding="tns:ITempConverterbinding">
65        <soap:address
66  location="http://developerdays.com/cgi-bin/tempconverter.exe/soap/ITempConverter"
67      </port>
68    </service>
69  </definitions>
```

Mise en œuvre de l'architecture

Attention : l'architecture précise

- des formats (dialecte XML)
- des sémantiques associées (sens de chaque élément des documents XML)
- des protocoles (e.g., utilisation de HTTP pour transmettre des messages SOAP)
- des services web standards (comme les API de UDDI)
- **et c'est tout !**

Questions ouvertes :

- comment programmer un service web ?
- comment accéder à un service web ?

Mise en œuvre de l'architecture (2)

Réponse : ça dépend !

- en Java, Sun standardise des API et des outils associés :
  - JAX-RPC (JA XML-based RPC) : utilisation de SOAP (mode RPC)
  - JAXR (JA XML Registries) : utilisation de UDDI
  - JAXM (JA XML Messaging) : utilisation de SOAP (mode message)
  - SAAJ (SOAP with Attachments API for Java) : utilisation de SOAP (pièces jointes)
  - outil de transformation d'une description WSDL en interface Java et vice-versa
  - etc.
- Microsoft fait pareil pour `.NET` :
  - API dans la bibliothèque de classes de `.NET`
  - utilisation en C# ou VB

Mise en œuvre de l'architecture (3)

Comment faire ça **bien** ?

- redémarrage automatique du serveur
- persistance des données
- interfaçage avec les systèmes d'information de l'entreprise (SGBD, ERP, CRM, etc.)
- etc.

Utiliser un serveur d'applications :

- serveur EJB (J2EE 1.4 va intégrer le support des services web)
- serveur `.NET`

Évolution de l'architecture

Il manque de nombreuses choses dans l'architecture de base :

- Sécurité :
  - identifier l'envoyeur
  - garantir l'intégrité du contenu
  - brique de base : WS-Security, OASIS (IBM, MS et Verisign)
  - construite à partir de XML-signature et XML-encryption (W3C)
  - autres briques :
    - SAML (Security Assertion Markup Language) : en gros permettre la signature unique avec identifiant transportable
    - XACML (eXtensible Access Control Markup Language) : définition d'une politique de sécurité en XML

Évolution de l'architecture (2)

- Fiabilité :
  - être sûr que les messages arrivent !
  - "standards" concurrents :
    - WS-R (Reliability), OASIS (Oracle, Sun et autres)
    - WS-RM (Reliable Messaging) BEA, IBM, MS et TIBCO
- Orchestration :
  - combiner simplement des services web
  - beaucoup de concurrence :
    - WSCL (Web Services Conversation Language), W3C (HP)
    - WSCI (Web Service Choreography Interface), W3C (BEA, Intalio, SAP et SUN)
    - BPEL4WS (Business Process Execution Language for Web Services), BEA, IBM et MS
    - etc.

Les enjeux

Remarques personnelles :

- les grands acteurs cherchent à se positionner pour dominer le marché
- mais ils veulent être “aimés” : ils proposent des standards à des organismes “indépendants” (le W3C, OASIS, etc.)
- tout en gardant le contrôle par des brevets logiciels
- avec une licence RAND (Reasonable and non-discriminatory) qui permet de faire payer les implémenteurs
- plutôt qu’une licence RF (Royalty-Free)

Futur :

- Pouvoir de WS-I (`http://www.ws-i.org/`), la Web Services Interoperability Organization ?
- Brevets logiciels sur SOAP 1.2 ?