Les services web
Un condensé

NASSRALLAH Rabih
nassrallah@ensm-douai.fr
http ://nassrallah.free.fr

BOURAQADI Noury
bouraqadi@ensm-douai.fr
http ://csl.ensm-douai.fr/noury

Dépt. G.I.P
Ecole des Mines de Douai
941 rue Charles Bourseul - B.P. 838
59508 Douai Cedex - France

Technical Report 2003-5-5
Ecole des Mines de Douai
16-Mai-2003

ble des m tières

1 Introduction

Internet est devenu omniprésent. Nous l'utilisons de plus en plus pour passer des commandes, réserver une chambre à l'hôtel, acheter un livre etc. Ainsi Internet est devenu un moyen pour faire du B2C[1]. Des entreprises par milliers ont lancé leurs sites Internet qui s'adressent au grand public de l'Internet (l'utilisateur final du produit). Aujourd'hui, les entreprises sont de plus en plus ouvertes les unes, aux autres et elles ont besoin de communiquer ensemble, que ça soit entre plusieurs filiales d'une même entreprise ou entre cette dernière et ses partenaires. Pour comprendre ce besoin croissant de communication, prenons l'exemple classique de l'agence de voyage. Quand vous vous rendez dans une agence de voyage, vous choisissez un paquetage de services,qui comprend le transport (billets d'avion, de trains, de bus, voitures de location), l'hôtel, les visites quotidiennes ou d'autres activités. L'agence devra alors contacter les entreprises de transports, et les hôtels ainsi que d'autres entreprises comme les sociétés d'assurances voyage, pour obtenir les meilleurs prix, confirmer les réservations, passer les commandes et facturer le client. Dans cet exemple, l'agence de voyage joue le rôle d'un courtier. Elle doit pouvoir accéder aux systèmes des autres entreprises depuis son logiciel. Mais nous pouvons toujours imaginer que les entreprises n'utilisent pas forcément les mêmes plate-formes, leurs applications ne sont pas écrites dans le même langage, ce qui rend très difficile l'intégration de ces différentes applications. Alors comment faire ? Une réponse possible consiste à utiliser les services web.

Un service web [W3C03b] est un système logiciel identifié par son URI [2] et ses Interfaces publiques et liens sont décrites en XML. Sa définition peut être découverte par d'autres systèmes logiciels. Ces systèmes peuvent interagir avec ce service web en échangant des messages XML (en respectant sa description) transporté par des protocoles Internet.

L'agence de voyage offrira un service web, qui décrira l'ensemble des opérations (réservation hôtel, achat billet, paiement électronique, etc.). Ce service web communiquera avec les autres services web offerts par les sociétés de transport (achat billet, paiement électronique), les hôtels (réservation, paiement électronique), etc. en échangeant des messages SOAP[3] (message XML) via HTTP[4] ou autres protocoles d'échange sur Internet. Nous parlons donc d'intégration des applications hétérogènes et distribuées.

Dans ce rapport nous présentons les services web et les technologies XML associées. Dans la première partie, nous aborderons la norme XML, la norme XML-Schema, la norme XML-Namespaces et nous clôturons par une présentation des analyseurs DOM[5] et SAX[6]. Dans la deuxième partie, nous présentons les services web, leur architecture, et enfin les technologies WSDL[7], SOAP et UDDI[8].

[1] Business to Customer
[2] Unified Resource Identificator
[3] Simple Object Access Protocole
[4] Hyper Text Transfer Protocol
[5] Document Model Object
[6] Simple API for XM
Web Service Description anguage
[8] Universal Description, Discovery and Integration

2 XML : eXtensible M rkup L ngu ge

XML [W3C00b] est une recommandation du W3C[9] qui date de février 1998. XML est un langage de balisage qui ressemble à HTML[10]. La norme HTML [W3C99a] définit un ensemble fini de balises, qui permettent essentiellement de visualiser les données. Au contraire la norme XML laisse à l'utilisateur la liberté de choisir ses propres balises. Elle est donc plus générale que HTML, car elle permet à la fois la gestion (c'est-à-dire l'organisation, l'échange et la transformation) et la représentation des données. SGML [11] [W3C86], le prédécesseur de HTML, était conçu dans le but de normaliser l'archivage des documents. XML dérive de SGML et il a l'avantage d'être moins complexe et plus universel. XML est un langage souple, il définit un ensemble de règles pour organiser les données. Plusieurs outils ont été développés autour cette norme permettant la manipulation des données décrites dans le document XML.

2.1 Document XML

Un document XML est un fichier texte composé d'une en-tête optionnelle et d'un corps qui a une structure d'arbre. L'en-tête contient des informations sur la version de XML et le codage utilisés, elle peut aussi référencer une DTD [12] ou un XML-Schema. Une DTD est une grammaire qui définit la structure d'un document XML. Elle permet de valider que chaque composant est à la bonne place. XML-Schema, définit une grammaire plus évoluée que celle des DTD. Elle permet en plus de spécifier les types des données (chaîne de caractères, décimal, etc...). Nous reparlerons plus loin de ces deux grammaires.

2.1.1 Eléments de XML

Le corps d'un document XML contient des éléments. Il est représenté par un couple de balises ouvrante et fermante. Il commence toujours par la balise ouvrante de l'élément racine et se termine par la balise fermante du même élément. l'élément racine encapsule plusieurs sous-éléments, qui eux encapsulent d'autres sous-éléments. Un élément peut avoir des attributs. Un attribut représente la même chose que les sous-éléments. La norme XML 1.0 [W3C00b][Gir01] ne précise pas les cas d'utilisations des attributs et des éléments. Mais il est conseillé d'utiliser les éléments pour exprimer le contenue du document et les attributs pour décrire les relations entre les éléments ou les propriétés des éléments.

Le corps du document peut contenir des commentaires et des sections littérales. <!- - voici un exemple de commentaire - ->. Les sections littérales constituent un mécanisme qui permet d'insérer dans un document XML une chaîne de caractères qui doit être traitée par le processeur comme étant une chaîne littérale. Tout caractère servant de délimiteur sera alors traité comme simple caractère et non comme balisage.

<![CDATA[Voici le <contenu> de la section littérale]]>

Dans cet exemple, la chaîne de caractères <contenu> ne sera pas interprétée comme étant une balise d'ouverture mais comme du texte.
Attention : une section littérale ne peut pas contenir la chaîne]] puisqu'elle tient lieu de délimiteur de fermeture.

Un document XML est bien formé [W3C00d] s'il respecte les règles de syntaxe XML. à savoir :

- Il faut toujours utiliser des balises fermantes pour les éléments non vide
- Les balises ne doivent pas se chevaucher. Dans l'exemple suivant, les balises <element> et <subElement> se chevauche, donc le document est mal formé.

```
<root>
    <element>
       <subElement> blablabla
```

World Wide Web Consortium
[10] Hyper Text Markup anguage
[11] Standard General Markup angage
[12] Document Type Definition

```
        </element>
           </subElement>
    </root>
```

2.1.2 Exemple d'un document XML

Dans cette section, nous allons donner un exemple de document XML qui modélise un Agenda simple (SimpleCalendar). Un Agenda est la propriété d'une personne (owner), contient des événements simples. Un évènement simple possède une description ; il commence à un instant précis (startingDateAndTime) et il dure un certain temps (duration) qui peut être exprimé en années, mois, jours, heures et minutes.

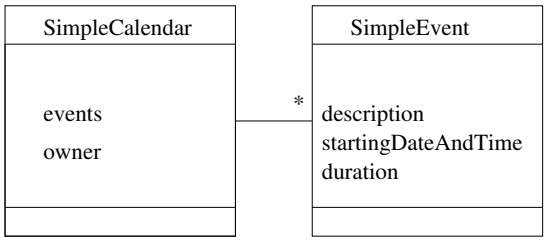

FIG. 1 – Modélisation de l'application agenda

Dans cet exemple, SimpleCalendar contient des événements SimpleEvent. L'élément SimpleCalendar sera alors la racine du document XML, il a comme attribut owner. Dans notre designSimpleCalendarpossède une variable d'instanceevents, qui est en fait un ensemble d'événements. l'élémentSimpleCalendaraura un sous–élément unique events. Ce dernier aura à son tour un ou plusieurs sous–élémentSimpleEvent.Chaque élémentSimpleEventaura trois sous–éléments : description, startingDateAndTimeetduration. L'élémentdurationpouvant être exprimé en : minutes, hours, days, months, years. Il est doté d'un attributUnits.

Généralement les balises correspondent au nom de l'élément comme <SimpleCalendar> </SimpleCalendar>. Un élément vide peut être représenter de deux façons : <elementName> </elementName> ou bien <elementName/>. Le document XML de la figure 2 (page 5) constitue la représentation d'un agenda qui contient trois événements.

2.2 La grammaire XML

Nous avons besoin d'une grammaire pour valider un document XML et pour définir sa structure et les types de données. Dans cette partie nous allons parler des deux types de grammaires qui sont DTD et XML-Schema.

2.2.1 DTD : Document Type Definition

Une DTD [W3C00a][Gir01] définit la structure d'un document XML. Elle décrit :
- les éléments types, i.e. elle donne les noms de tous les éléments et leur modèle de contenu.
- les attributs pour chaque élément (nom, type et valeur par défaut).

Une DTD peut être intégrée dans le document XML, elle est dans ce cas *interne*. A l'opposé une DTD peut être *externe* dans un fichier séparé. La DTD externe sera référencée dans l'en-tête du document XML comme suit :

```
<!DOCTYPE dtdName SYSTEM ‘‘/directory/subDirectory/myDTD.dtd’’>
ou bien
<!DOCTYPE dtdName PUBLIC ‘‘http://www.aSiteAddress.com/.../MyDTD.dtd’’>
```

La mention "SYSTEM"indique que la DTD est sur un disque local, alors que la mention "PUBLIC"indique que la DTD se trouve sur le web.

```
<!-- Voici un commentaire : Exemple d'un document XML -->
<?xml version=    .0'' encoding=   ISO-8859- ''?>
<SimpleCalendar owner=''Nassrallah Rabih''>
   <events>
      <SimpleEvent>
         <description> Meeting with Dr.Noury </description>
         <StartingDateAndTime>  4-April-2003 at 09:30 AM </startingDateAndTime>
         <duration Units=''minutes''> 30 </duration>
      </SimpleEvent>

      <SimpleEvent>
         <description> Camping </description>
         <StartingDateAndTime> 26-April-2003 at 02:00 PM </startingDateAndTime>
         <duration Units=''days''> 2 </duration>
      </SimpleEvent>

      <SimpleEvent>
         <description> Dinner with Lora </description>
         <StartingDateAndTime> 28-April-2003 at 09:00 PM </startingDateAndTime>
         <duration Units=''hours''> 2 </duration>
      </SimpleEvent>
   </events>
</Calendar>
```

FIG. 2 – Document XML d'un agenda contenant trois événements

La DTD externe est plus avantageuse que la DTD interne, car nous pouvons l'utiliser pour décrire la structure de plusieurs documents XML. Elle est donc réutilisable, au contraire de la DTD interne qui est spécifique aux documents qui la contiennent. Un document XML peut avoir plusieurs DTD, et sa DTD globale sera l'union de toutes les DTD internes et externes qu'il utilise. A titre d'exemple nous donnons la DTD (figure 3) qui décrit la structure d'un document XML représentant un agenda.

```
<!DOCTYPE SimpleCalendar[
       <!ELEMENT SimpleCalendar (events)>
       <!ATTLIST SimpleCalendar Owner CDATA  #REQUIRED>
       <!ELEMENT events (SimpleEvent+)>
       <!ELEMENT SimpleEvent (description, startingDateAndTime, duration)>
       <!ELEMENT description #PCDATA>
       <!ELEMENT startingDateAndTime #PCDATA>
       <!ELEMENT duration  #PCDATA>
       <!ATTLIST duration Units (''Years'',''Days'', ''Hours'', ''Minutes'') #REQUIRED>
       ]>
```

FIG. 3 – La DTD correspondante à un agenda

- Dans < !ELEMENT> on définit l'élément racine, les éléments nœuds (qui possèdent des sous éléments) et les éléments feuilles (qui ne possèdent pas de sous éléments). Pour la racine et les nœuds, < !ELEMENT> définit les constituants (sous éléments), et leurs nombre d'occurrences.
  - < !ELEMENT element (subElement*)> signifie que l'élément de type element peut contenir zéro ou plusieurs sous–éléments.

- < !ELEMENT element (subElement+)> signifie que l'élément de type element doit contenir au moins un sous–élément.
- < !ELEMENT element (subElement ?)> signifie que l'élément peut contenir zéro ou un sous–élément.
- < !ELEMENT element (subElement)> signifie que l'élément doit contenir un et un seul sous–éléments.
- < !ELEMENT element (ANY)> signifie que l'élément peut contenir n'importe quel sous–éléments.

La DTD ne permet pas vraiment de spécifier le type de l'élément (entier, chaîne de caractère etc..). Dans les feuilles < !ELEMENT> spécifie que l'élément contient une chaîne de caractères (PCDATA[13]).

- Dans < !ATTLIST element attribut Type defaultValue Constraint> on définit les attributs de chaque éléments et leurs valeurs par défaut.
  les contraintes sont :
  - REQUIRED :attribut obligatoire.
  - IMPLIED : attribut optionnel.
  - FIXED : attribut de valeur fixe.
- Les principaux types des attributs sont :
  - CDATA (pour Character DATA) : chaîne de caractère.
  - ("value1","value2","value3") : ensemble de chaîne de caractère possibles.
  - ID : l'attribut doit avoir un identifiant unique dans chaque élément. Il permet de référencer l'élément dans une autre partie du document en utilisant un attribut de type IDREF (voir l'exemple de la figure 4)

```
<?xml version='' .0'', encoding=''ISO-8859 - ''?>
<!DOCTYPE person
[
<!ELEMENT Person (#PCDATA)>
<!ELEMENT parent (son*)>
<!ELEMENT son EMPTY>
<!ATTLIST person identifier ID #REQUIRED>
<!ATTLIST parent identifier IDREF #REQUIRED>
<!ATTLIST son identifier IDREF REQUIRED>
]>
<person identifier=''0''>Fred</person>
<person identifier='' ''>Ted</person>
<parent identifier=''0''>
     <son identifier='' ''/>
</parent>
```

FIG. 4 – Exemple d'utilisation de ID et IDREF dans une DTD

  - IDREF : la valeur de cet attribut doit correspondre à la valeur d'un attribut de type ID. En d'autre terme cet attribut se réfère à l'identificateur unique d'un autre attribut.

2.2.2 XML-Schema

XML-schema [W3C01b][Gir01] est une recommandation du W3C. Elle permet de décrire d'une façon plus complète la structure d'un document XML. XML-Schema permet de :

- définir la structure de l'arbre XML (parent, fils, ordonnancement, nombre de fils)
- définir les types des éléments de document XML (entier, chaîne de caractères etc.)

[13]Parsed Character Data

- définir les attributs des éléments de document XML (types et contraintes *optionnels, obligatoires, nombre d'occurrences etc.*)

Un XML-Schema est plus avantageux que les DTD, mais leurs écriture est plus compliqués. Le grand inconvénient de XML-Schema est que pour le moment, il y a peu d'outils de validation stables. Ses avantages sont :

- Un document XML-Schema est écrit en XML, ce qui n'est pas le cas pour les DTD
- XML-Schema permet de spécifier les types des données (décimal, chaîne de caractère, structure etc.)
- XML-Schema s'appuie sur les espaces de nommages, qui permettent d'identifier de façon uniques les balises et leur sémantique dans un document XML. Nous présentons les espaces de nommage dans la prochaine section.
- XML-Schema est extensible, car le langage XML est en lui même extensible.

Reprenons l'exemple de l'agenda que nous avons vu plus haut (figure 2 page 5). Le schéma-XML correspondant est décrit dans la figure 5 (page 8).

```
<!-- dans cette première ligne on spécifie l'espace de nommage -->
<xsd:schema xmlns:xsd=''http://www.w3c.org/2000/10/XMLSCHEMA''>

<!-- définition d'un nouveau type ``timeEnumerate'' -->
<!-- que  nous utilisons pour définir le type de l'attribut Units de la balise duration -->
<xsd:simpleType  name=''timeEnumerate'' base=''xsd:string''>
   <xsd:enumeration value=''Years''>
   <xsd:enumeration value=''Days''>
   <xsd:enumeration value=''Hours''>
   <xsd:enumeration value=''Minutes''>
</xsd:simpleType>

<!-- definition de la structure du document et du type des éléments -->

<!-- définition de la racine SimpleCalendar -->
   <xsd:element name= ``SimpleCalendar''>
     <xsd:complexType>
        <xsd:sequence>
           <xsd:element name= ``events'' type= ``complexType''
                        minoccures= ``1'' maxoccures= ``1''/>
        </xsd:sequence>
        <xsd:attribute name= ``owner'' type= ``xsd:string''>
     </xsd:complexType>
   </xsd:element>

<!-- définition de l'élément events -->
    <xsd:element name= ``events''>
       <xsd:complexType>
          <xsd:sequence>
             <xsd:element name= ``SimpleEvents'' type= ``complexType''
                          minoccurence= ``0'' maxoccurence= ``unbounded''/>
          </xsd:sequence>
       </xsd:complexType>
    </xsd:element>

<!-- définition de l'élément SimpleEvent -->
    <xsd:element name= ``SimpleEvent''>
       <xsd:complexType>
          <xsd:sequence>
<!-- définition de l'élément feuille description -->
             <xsd:element name= ``description'' type= ``xsd:string''
                          minoccurence= ``1'' maxoccurence= ``1''/>

<!-- définition de l'élément feuille startingDateAndTime -->
             <xsd:element name= ``startingDAteAndTime'' type= ``xsd:string''
                          minoccurence= ``1'' maxoccurence= ``1''/>

<!-- définition de l'élément feuille duration -->
             <xsd:element name= ``duration'' type= ``xsd:integer''
                          minoccurence= ``1'' maxoccurence= ``1''>
                <xsd:attribute name= ``Units'' type= ``timeEnumeration'' uses= ``required''/>
             </xsd:element>
          </xsd:sequence>
       </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

FIG. 5 – Le schéma XML correspondant à un agenda

2.2.3 Espaces de nommage

Les espaces de nommage [W3C99b] (XML Namespaces) sont une recommandation W3C qui a vu le jour le 18 décembre 2002. Ils permettent d'identifier d'une façon unique les balises d'un document XML en utilisant des URI. Pour vous illustrer l'importance des espaces de nommage, prenons l'exemple suivant :
Pour fixer un rendez-vous entre la direction de l'entreprise A et celle de l'enteprise B, chacune va proposer un rendez-vous. Les deux entreprises disposent chacune d'une application de gestion d'agenda différente. Les représentations XML des rendez-vous générées par les deux applications sont présentées à la figure 6.

```
<!-- Rendez-vous produit par l'application de la société A -->

<?xml version= ''1.0'' encoding= ''ISO-8859-1''?>

<SimpleEvent>
   <description> RDV proposition  </description>
   <StartingDateAndTime> 11-April-2003 at 03:30 PM </startingDateAndTime>
   <duration Units= ''Hours''> 2 </duration>
</SimpleEvent>

<!-- Rendez-vous produit par l'application de la société B -->

<?xml version= ''1.0'' encoding= ''ISO-8859-1''?>

<EvenementSimple>
   <description> RDV proposition  </description>
   <dateEtHeure> 11-April-2003 at 03:30 PM </dateEtHeure>
   <duree En= ''Heures''> 2 </duree>
</EvenementSimple>
```

FIG. 6 – Exemple d'une même information (rendez-vous) codée avec des balises différentes

Comme nous le voyons bien, les deux entreprises n'utilisent pas les même balises pour exprimer la même chose. Il en résulte une impossibilité lors de l'interprétation des deux rendez-vous par les deux différentes applications. La solution est d'utiliser un espace de nommage commun aux deux entreprises. Cet espace définira les balises à utiliser et la sémantique associée à chacune d'elle. Un exemple d'utilisation des espaces de nommage est présenté figure 7.

```
<?xml version= ''1.0'' encoding= ''ISO-8859-1''?>

<!-- Référence à l'espace de nommage Calendars qui définit les balises à utiliser-->
<!-- (SimpleEvent et ses sous-éléments) -->

<Calendars:SimpleEvent xmlns:Events   = ''http://www.ourCommonCalendarsNameSpace.org''>
   <description> RDV proposition  </description>
   <StartingDateAndTime> 11-April-2003 at 03:30 PM </startingDateAndTime>
   <duration Units= ''Hours''> 2 </duration>
</Calendars:SimpleEvent>
```

FIG. 7 – Exemple d'utilisation des espaces de nommages

2.3 Les analyseurs XML

Nous venons de donner l'exemple des deux entreprises "A" et "B" qui s'échangent des documents XML, pour se mettre d'accord sur la date de leur prochaine réunion. Chaque entreprise possède une application de gestion d'agenda partagé. Comme les données internes aux applicatopns sont généralement représentées dans un format autre que XML (exemple : des objets en Smalltalk ou Java), il est nécessaire de disposer d'outils pour la convertion depuis et vers XML. Dans ce rapport nous nous limiterons à SAX et DOM.

2.3.1 SAX : Simple API for XML

SAX [Meg01] est un parseur de type "callback", créé par David Megginson (Megginson Technologies). SAX n'est pas normalisé par le W3C, comme le pense beaucoup de gens. Il se base sur un modèle d'envoie d'événements. Il parcourt le document XML sans considérer sa structure d'arbre et renvoie un événement à chaque fois qu'il rencontre une nouvelle balise. Le programmeur définit le traitement à effectuer en implémentant un gestionnaire d'événement *event handler*. SAX fournit des interfaces pour accéder aux différents composants du document XML. Elles sont décrites en Java, mais il existe d'autres travaux pour les décrire en d'autres langages. Les deux principales interfaces fournies par SAX sont :

1. ContentHandler : interface qui contient les méthodes de base pour analyser le contenu d'un document XML. Parmi ces méthodes on cite :
   - void startDocument() : elle reçoit un événement lorsque le parsseur passe par le début de document
   - void endDocument() : elle reçoit un événement lorsque le parsseur arrive à la fin de document
   - void startElement(String namespaceURI, String localName, String qName, Attributes attr) : elle reçoit un événement quand le parsseur passe par la balise ouvrante de l'élément.
   - void characters(char[] ch, int start, int length) : elle reçoit un événement quand le parsseur passe par une chaîne de caractère.
2. ErrorHandler : interface qui gère les événements déclenchés lorsque des erreurs se produisent (exemple : le document est mal formé).

La figure 8 de la page 11 illustre le code java qui permet d'afficher la structure d'arbre d'un document XML analysé par SAX. Le résultat de l'execution sur un document décrivant un agenda est illustré par la figure 9(page 12).

```
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import org.apache.xerces.parsers.SAXParser;



public class Trace extends DefaultHandler {
  int indent;


  void printIndent() {
    for (int i=0; i<indent; i++) System.out.print("-");
  }

  public void startDocument() {
    System.out.println("start document");
  }

  public void endDocument() {
    System.out.println("end document");
  }

  public void startElement(String uri, String localName,
                           String qName, Attributes attributes) {
    printIndent();
    System.out.println("starting element> " + qName);
    indent++;
  }

  public void endElement(String uri, String localName,
                         String qName) {
    indent--;
    printIndent();
    System.out.println("end element> " + qName);
  }



  public static void main(String[] args) {
    Trace t = new Trace();
    SAXParser p = new SAXParser();
    p.setContentHandler(t);
    try { p.parse(args[0]); }
    catch (Exception e) {e.printStackTrace();}
  }
}
```

FIG. 8 – Le code java qui permet de tracer l'arbre d'un document XML analysé par SAX

```
start document
starting element> Calendars:SimpleCalendar
-starting element> Calendars:events
--starting element> Events:SimpleEvent
---starting element> Events:description
---end element> Events:description
---starting element> Events:startingDateAndTime
---end element> Events:startingDateAndTime
---starting element> Events:duration
---end element> Events:duration
--end element> Events:SimpleEvent
-end element> Calendars:events
end element> Calendars:SimpleCalendar
end document
```

FIG. 9 – Exemple de résultat d'exécution de notre analyseur XML basé sur SAX

2.3.2 DOM : Document Object Model

DOM [W3C03a], est créé au sein du W3C, donc il est normalisé. DOM représente le document XML sous forme d'un arbre d'objets dans la mémoire et fournit des interfaces permettant de les manipuler. Les interfaces DOM sont décrites en IDL [OMG03][14] de l'OMG[15] ce qui les rends indépendantes des langages de programmation. Les principales interfaces que DOM fournies pour la manipulation d'un document XML sont les suivantes :

1. Node : cette interface contient toutes les méthodes d'ajout, de suppression, de consulation et de parcours des nœuds de l'arbre.
2. Document : cette interface très importante contient les méthodes nécessaires à la génération d'un document XML. Elles permettent de créer des éléments, des commentaires et des attributs. Elles permettent également une recherche sur les nœuds du document.
3. Attr : cette interface dérive de l'interface nœud. Elle représente les feuilles (les éléments qui ne possèdent pas de sous éléments) de l'arbre.
4. Text, CDATASection : cette interface représente les chaînes de caractères du document DOM.
5. Comment : cette dernière interface représente les commentaires du document DOM.

L'exemple de la figure 10 (page 13) permet de mieux apréhender DOM à l'aide d'une de ses implémentation en Java appelée JDOM[16]. Nous allons créer un document XML vide en mémoire et puis l'enrichire pour obtenir un document XML qui décrit un agenda.

2.3.3 Comparaison entre SAX et DOM

Nous venons de présenter deux types de parsseurs (analyseurs) XML, qui utilisent chacun une technique différente pour analyser un document XML. Nous ne recommendons pas l'utilisation de l'un ou l'autre puisque chacun à ses spécifités.

- Contrairement à DOM, SAX ne construit pas une image du document XML dans la mémoire, ce qui peut savèrer très utile si on manque de ressources (mémoire, CPU etc.).
- DOM permet non seulement d'analyser un document XML, mais aussi de le modifier, ce qui n'est pas le cas de SAX.
- Les intérfaces DOM sont décrites en IDL [OMG03] (langage de description normalisé et indépendant des langages de programmation). Ce qui permet d'avoir des implémentations dans différents langages.
- DOM est une norme du W3C alors que SAX ne l'est pas.

[14] Interface Definition anguage
[15] Object Management Group
[16] Java Document Object Model

```
import java.io.*;
import com.sun.xml.tree.*;
import org.w3c.dom.*;


public class sampleDOM {
  public static void main (String arg[])
         throws IOException, DOMException {


  /*                       création du document XML vide                                    */

  XmlDocument xmlDoc               = new XmlDocument();
  ElementNode SimpleEvent          = (ElementNode)xmlDoc.createElement(``SimpleEvent'');
  ElementNode description          = (ElementNode)xmlDoc.createElement(``description'');
  ElementNode startingDateAndTime  = (ElementNode)xmlDoc.createElement(``startingDateAndTime'');
  ElementNode duration             = (ElementNode)xmlDoc.createElement(``duration'');

  writer out = new OutputStreamWriter (system.out);
  out.write(``Empty XML document creation completed...\n'');

  /* Maintenant nous allons établire les liens entre les éléments et enrichire le document */


  xmlDoc.appendChild(SimpleEvent);
  SimpleEvent.appendChild(description);
  description.appendChild(xmlDoc.createTextNode(``RDV proposition'');
  SimpleEvent.appendChild(startingDateAndTime);
  startingDateAndTime.appendChild(xmlDoc.createTextNode(``11-April-2003 at 03:00 PM'');
  SimpleEvent.appendChild(``duration'');
  duration.appendChild(xmlDoc.createTextNode(``2'');
  duration.setAttribut(``Units'' , ``Hours'');

  out.write(``Document: \n'');
  xmlDoc.write(out);
  out.write(``\n'');
  out.flush();

  system.exit(0);
  }
}
```

FIG. 10 – Exemple de création d'un document XML avec JDOM

3 Services web

La définition des services web [W3C03b] donnée par le W3C est la suivante :

> "A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols." [17]

Un service web [Man01][Got02][Som02][Cur02][Cha02] possède les caractéristiques suivantes :

- Il est accessible via le réseau.
- Il dispose d'une interface publique (ensemble d'opérations) décrite en XML.
- Ses descriptions (fontionalités, comment l'invoquer et où le trouver) sont stockées dans un annuaire.
- Il communique en utilisant des messages XML. Ces messages sont transportés par des protocoles Internet (généralement HTTP, mais rien n'empêche d'utiliser d'autres protocoles de transfert).
- L'intégration d'applications en implémentant des services web, produit des systèmes faiblement couplés. Le demandeur du service ne connait pas forcement le fournisseur. Ce dernier peut disparaître sans perturber l'application cliente qui trouvera un autre fournisseur en cherchant dans l'annuaire.

Les services web peuvent être classés selon leur rôle, sous deux grandes catégories :

- Les services web métier : ce sont les services proposés par l'entreprise. Exemple : Le service de consultation dans un agenda distribué.
- Les services web techniques : ce sont des services indispensables au fonctionnement de l'assemblage de services web métier. Exemple : l'annuaire UDDI, qui permet la découverte des services.

Les services web peuvent être déployés sur Internet, Intranet et sur l'Extranet de l'entreprise aussi. Ils supportent des applications de type B2B[18], B2C ou D2D[19]. L'utilisateur d'un service web, peut être une personne, une application ou un autre service web. Cette technologie émergante n'est pas encore très mature, mais nous commencons à voir de plus en plus d'entreprises – même de taille moyenne – adopter cette solution.

3.1 Architecture

La figure 11 de la page 15 donne les acteurs et les opérations d'une architecture de service web [Kre01][Leb02]. Les acteurs sont :

- Le demandeur du service : c'est l'utilisateur, l'application ou le service web qui demande le service.
- Le fournisseur du service : c'est l'entreprise prpriétaire du service ou l'hébérgeur de service.
- le service d'enregistrement : c'est un service web, qui permet d'enregistrer les descriptions des services web – publiées par l'entreprise proporiètaire des services – dans un annuaire de service web.

Les opérations sont (dans l'ordre) :

- La publication du service : c'est le fait de diffuser la description du service web.
- Le découverte d'un service : c'est le fait de trouver sa description et son URI pour pouvoir l'invoquer.
- L'invoquation du service : une fois que le demandeur récupére l'URI et le descriptif du service, il les utilise pour l'invoquer en lui envoyant un message XML.

[1] Un service web est un système logiciel identifié par son URI (Unified Resource Identificator) et ses Interfaces publiques et liens sont décrites en XM . Sa définition peut être découverte par d'autres systèmes logiciels. Ces systèmes peuvent interagir avec ce service web en échangant des messages XM (en respectant sa description) transporté par des protocoles Internet.

[18] Business To Business

[1] Department To Department

La publication des services web peut se faire de deux façons :

1. publication dynamique : un fournisseur s'enregistre auprès d'un annuaire de services. Les clients trouveront ce fournisseur en invoquant l'opération "Find" du service annuaire (figure 11).
2. publication directe : Le fournisseur connait ses clients, il leurs envoie les descriptions et les URI des services directement sans passer par un annuaire (figure 12).

De manière symétrique, ill existe deux types de découverte de service.

1. la découverte dynamique : consiste à utiliser le service fournit par l'annuaire (UDDI) pour trouver la listes des services qui correspondent aux besoins du demandeur de service. C'est analogue à l'utilisation d'un annuaire sur Internet pour trouver le lien vers une ressource sur le web. Ce type de découverte n'est possible que si les fournisseurs de services publient les descriptions de leurs services dans l'annuaire en invoquant la méthode "Publish" du service annuaire (figure 11).
2. la découverte static : elle est le résultat d'une publication directe. Dans ce cas le client n'a pas besoin d'invoquer la méthode "Find" du service annuaire (figure 12).

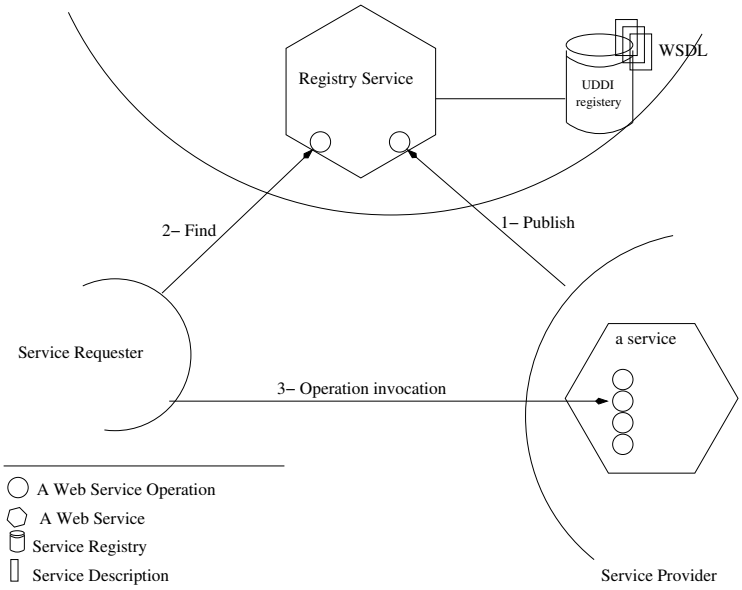

FIG. 11 – Architecture des services web avec publication/découverte dynamique

Les services web sont accessibles via le réseau, grace à une couche réseau (figure 13). Cette couche réseau, correspond à la couche applicative dans le modèle OSI [ISO70]. Nous pouvons utiliser les protocoles de transport, comme HTTP, SMTP[20], FTP[21], etc.
Le protocole HTTP, est généralement le plus utilisé, surtout quand nous déployons des services web sur Internet. En utilisant HTTP, nous évitons les problèmes de pare-feu ou de configuration de réseaux IP.

Pour invoquer les services web et guarantir l'intéropérabilité entre eux, Nous avons besoin d'une deuxième couche de communication. cette couche (figure 13) définit les potocoles d'échange de messages basé sur XML entre le demandeur du service et le fournisseur du service. C'est à ce niveau là que nous définissons le format du message XML, qui nous permettera de publier, chercher et invoquer un service web. SOAP [W3C00c][22] est le format standardisé et le plus recommendé. Nous reviendrons plus en détail sur ce format.

[20] Simple Mail Transfer Protocol
[21] File Transfer Protocol
[22] Simple Object Access Protocol

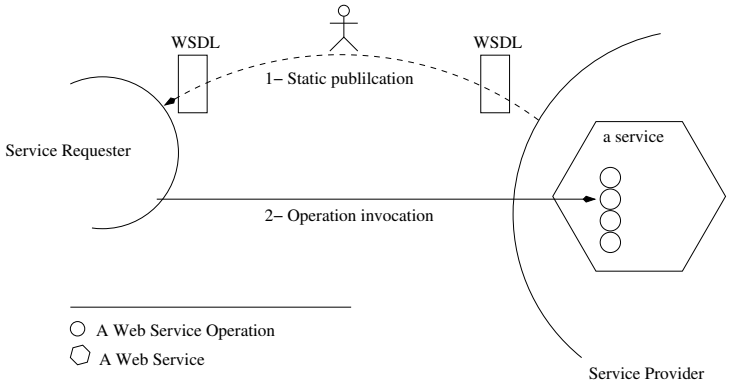

FIG. 12 – Architecture des services web avec publication/découverte statique

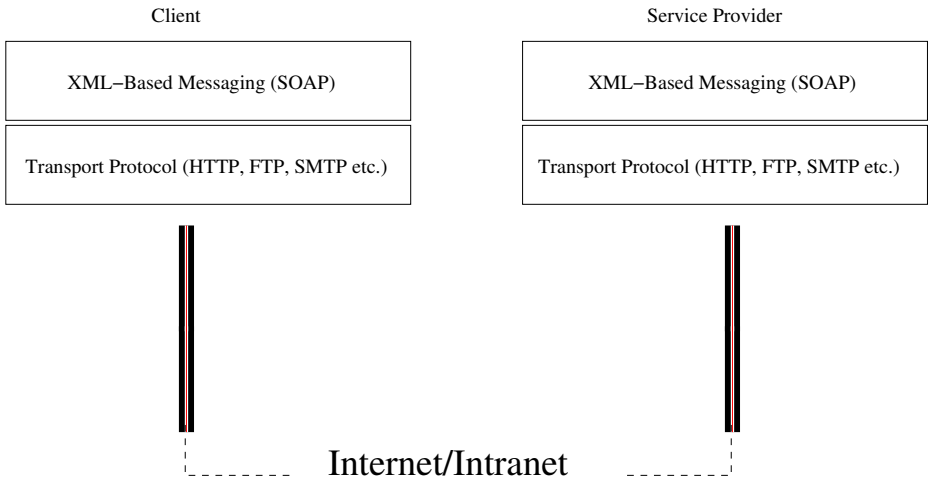

FIG. 13 – Les couches de bases de communications des services web

Pour pouvoir invoquer, un service, nous avons besoin de sa déscription. Cette description ressemble au guide d'utilisation du service web. Un service web est décrit généralement en WSDL [W3C01a] qui est un langage de description standard aussi. Nous parlerons plus en détail de ce langage.

3.2 WSDL : Web Service Description Language

WSDL [W3C01a][Got02][Cha02] est un langage de description des services web. Il est élaboré par un Consortium d'entreprise dont IBM, Intel et Microsoft. Un draft décrivant les spécifications de ce langage est soumis au W3C depuis le 14 mars 2001, et il n'est toujours pas normalisé.

Un document WSDL est un fichier text au format XML qui décrit un service web. Il définit, de manière abstraite et indépendante du langage, l'ensemble des fonctionalités offertes par le service web. Il décrit "comment" communiquer avec un service web et "ou" le trouver. Le langage WSDL, permet de décrire :

1. Les opérations qu'un service web peut exécuter.
2. Le type de messages XML (exemple : SOAP) qu'il peut traiter.
3. Les protocoles de communication qu'il supporte (exemple HTTP).

4. le point d'accès à une instance du service web (URL[23]).

La figure 14 explique mieux la structure d'un document WSDL.

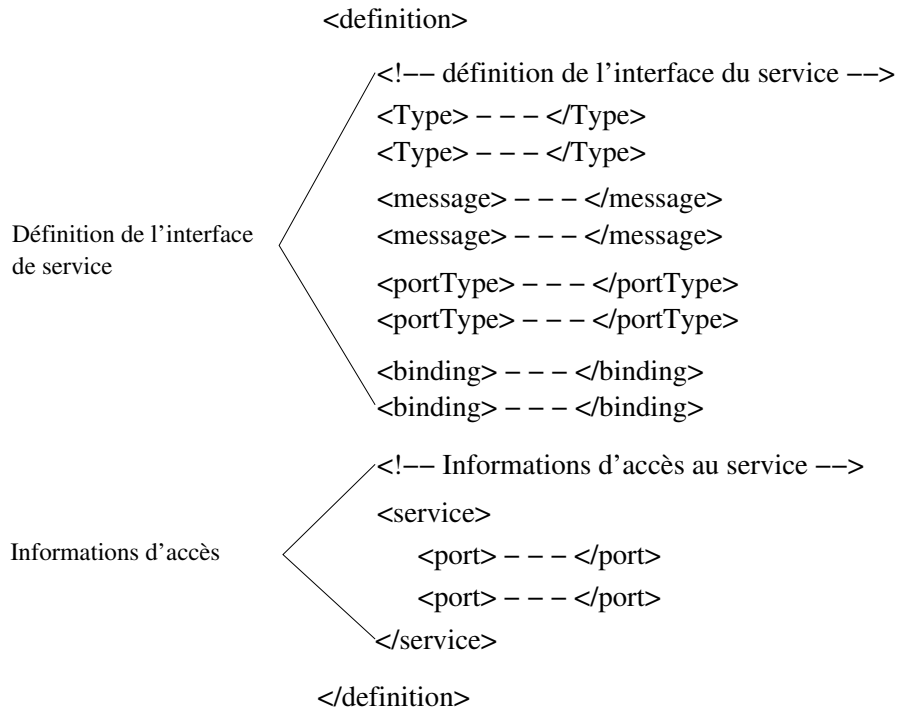

FIG. 14 – La structure d'un document WSDL

Comme nous pouvons le remarquer le document est composé de deux parties séparées :

1. La définition de l'interface du service ("comment" et "où" cette fonctionnalité est proposée) qui contient la définition des éléments suivants :
   - Type : décrit les types complexes.
   - Binding :décrit le protocole de transport à utiliser et le format des données.
   - PortType :décrit l'ensemble des opérations (méthodes, signatures) publiques
   - Message : décrit séparement les types des arguments et le type de retour de chaque méthode.
2. La partie "informations d'accès" regrouppe la définition des éléments suivant :
   - Service : il contient les informations nécessaire pour invoquer le service web. Il contient plusieurs éléments "port".
   - Port : correspond à une URL qui donne accès à certaines opérations du service.

Les programmeurs peuvent écrire le document WSDL en entier, mais il existe des outils qui permette de générer une partie du document WSDL à partir du code écrit dans un langage de programmation.

La figure 15 de la page 18 est un exemple d'un document WSDL qui décrit un service web CalendarService. Les clients peuvent invoquer l'opération displayEventsqui retourne la liste des événements d'un agenda donné. Cette méthode prend une chaîne de caractère en argument calendarIDet retourne une chaîne de caractères eventsDescriptions.

[23]Unified Resource ocator

```
<definition
            xmlns:xsd = ‘‘http://www.w3c.org/1999/xmlshema’’
            xmlns:soap = ‘‘http://schemas.xmlsoap.org/wsdl/soap’’>

   <message name = ‘‘displayEventsInput’’>
      <port name = ‘‘calendarID’’ type = ‘‘xsd:string’’/>
   </message>

   <message name = ‘‘displayEventsOutput’’>
      <port name = ‘‘eventsDescriptions’’ type = ‘‘xsd:string’’/>
   </message>

   <portType name = ‘‘displayEventsPortType’’>
      <operation name = ‘‘displayEvents’’>
         <input message = ‘‘displayEventsInput’’/>
         <output message = ‘‘displayEventsOutput’’/>
      </operation>
   </portType>

   <binding name = ‘‘displayEventsSoapBinding’’ type = ‘‘tns:displayEventsPortType’’>

      <soap:binding style = ‘‘document’’
                    transport = ‘‘http://schemas.xmlsoap.org/soap/http’’/>
      <operation name = ‘‘displayEvents’’>
         <soap:operation soapAction = ‘‘urn:CalendarService’’/>
         <input>
            <soap:body use = ‘‘encoded’’
                       encodingStyle = ‘‘http://schemas.xmlsoap.org/soap/encoding/’’/>
         </input>

         <output>
            <soap:body use = ‘‘encoded’’
                       encodingStyle = ‘‘http://schemas.xmlsoap.org/soap/encoding/’’/>
         </output>

      </operation>

   </binding>

   <service name = ‘‘CalendarService’’>

      <port name = ‘‘diplayEventsSoap’’ binding = ‘‘tns:displayEventsSoapBinding’’>
         <soap:address location = ‘‘http://csl.ensm-douai.fr/CalendarService/’’>
      </port>

   </service>

</definition>
```

FIG. 15 – Le document WSDL correspondant à un service web agenda

3.3 SOAP : Simple Object Access Protocol

SOAP [W3C00c][Cha02][Cur02] est définit comme un protocole d'échange de donnée dans un environnement distribué, décentralisé. Il est basé sur la norme XML et comporte trois parties : une enveloppe qui définit un cadre décrivant le contenu du message SOAP et comment l'analyser, un ensemble de règles de codage qui décrivant les types de données (types simples, types complexes) et une convention de représentation des requêtes/réponse. SOAP peut être transporté par différents protocoles de transport, pourtant seul l'assosiation de SOAP avec HTTP est décrite dans le draft de la norme SOAP.

SOAP était dévéloppé par un groupe d'entreprises dont HP, IBM et Microsoft. Le document de travail Le draft de la norme SOAP était soumis au W3C le 18 avril 2000, mais il n'est pas encore normalisé.

SOAP peut être résumé en huit points :

- il est un protocol de communication
- il est basé sur XML
- il définit un format de message
- il est utilisé pour faire communiquer les applications distribuées dans un réseau Intranet, Extranet ou Internet.
- il permet l'intéropérabilité des applications.
- il est indépendant des plate-formes.
- il est indépendant des langages de programmation.
- il est simple.

La figure 16 de la page 19 illustre la structure d'un message SOAP.

```
<!-- début de l'enveloppe SOAP-->

<SOAP-ENV:Envelope
           xmlns:SOAP-ENV = ‘‘http://schemas.xml.org/soap/envelope/’’
           SOAP-ENV: encodingStyle = ‘‘http://schemas.xml.org/soap/encoding’’/>

<!-- début de l'en-tête SOAP-->

<SOAP-ENV:Header>
    <!-- contenue de l'en-tête>
</SOAP-ENV:Header>

<!--fin de l'en-tête SOAP-->

<!-- début du corps SOAP-->

<SOAP-ENV:Body>
    <!-- contenue du corps>
</SOAP-ENV:Body>

<!--fin du corps SOAP-->


</SOPA-ENV:Envelope>
<!-- fin de l'enveloppe SOAP>
```

FIG. 16 – Structure d'un message SOAP

Pour illustrer les messages SOAP, supposons que le fournisseur du serviceCalendarService(la description WSDL est donné par la figure 15) vient d'être découvert par un client qui souhaite avoir la liste des événements de son agenda. Cet agenda est identifiée par son nomaClientCalendar. Nous présentons le contenu du message SOAP que le client envoie au fournisseur du service dans

la figure 17.

```
<!-- L'enveloppe contient l'espace de nommage définissant la version de SOAP et le codage utilisés -->

<SOAP-ENV:Envelope
          xmlns:SOAP-ENV = ''http://schemas.xml.org/soap/envelope/''
          SOAP-ENV: encodingStyle = ''http://schemas.xml.org/soap/encoding''/>

<!-- Dans Header nous précisons des informations sur le recepteur du message -->

<SOAP-ENV:Header
   SOAP-ENV:actor = ''http://schemas.xmlsoap.org/soap/actor/next''
   SOAP-ENV:mustUnderstand = ''1''>

<!-- l'attribut ''actor'' précise les informations concernant les serveurs
<!-- intermédiaires  entre la source et la destination -->

<!-- mustUnderstand est un attribut binaire. La valuer ''1'' indique indique au destinataire qu'il -->
<!-- doit reconnaître le contenue de l'en-tête.-->

   <!-- identification de l'émetteur et de la transaction -->
   <indentifier number = ''12345''>
   <transaction type = ''listing'' number = ''XYZ''>
</SOAP-ENV:Header>

<!-- Body contient les informations que nous vondrons transmettre à l'application -->
<SOAP-ENV:Body>
   <SOAP-ENV:Body>

      <!-- définition de l'espace de nom ''m'' commun -->
      <!-- à l'utilisateur et au fournisseur du service -->
      <m:displayEvents xmlns:m = ''CalendarService''>

         <!-- invocation du service -->
         <calendarID type = ''xsd:string''> aClientCalendar </calendarID>
         <!--  fin de l'invocation du service -->

      </m:displayEvents>
   </SOAP-ENV:Body>
</SOAP-ENV:Body>

</SOPA-ENV:Envelope>
```

FIG. 17 – Structure d'un message SOAP

3.4 DD : niversal Description, Discovery and ntegration

UDDI [OAS02][Cha02][Cur02] a été créé par IBM, Microsoft et Ariba. C'est une architecture répartie qui permet aux fournisseurs de services web, d'enrergistrer leur services et aux applications (demandeurs de service) de rechercher les services correspondant à leur besoins. UDDI est donc un annuaire distribué de services web et d'entreprises. UDDI se comporte en lui même comme un service web dont les méthodes (publish et find) sont appellées via SOAP.
UDDI est composé de deux parties :

- Annuaire d'entreprise
- Les interfaces d'accès à ces annuaires

L'annuaire des entreprises contient des informations sur l'entreprise (adresse, activité, secteur etc.), les services métiers offerts par l'entreprise ainsi que les descriptions techniques des services proposés. Les fournisseurs de services viennent s'enregistrer auprès de l'UDDI, en publiant leurs services dans l'annuaire des entreprises, et définissant les interfaces d'accès à ces services. Quand un demandeur de service lance une requête UDDI (requête SOAP, pour invoquer la méthode find), il obtient les informations sur l'entreprise et ses services offerts, ainsi un descriptif (le fichier WSDL) du service demandé. Il existe plusieurs utilisations d'annuaires UDDI [Kre01], nous citerons :

- L'UDDI des opérateurs : c'est un annuaire global distribué géré par un opérateur (propriétaire de l'annuaire) et accèssible depuis l'internet. Tous les fournisseurs de services peuvent enregistrer leur services dans cet annuaire.
- L'UDDI interne d'une entreprise : c'est l'annuaire des services web accessibles depuis l'intranet de l'entreprise.
- L'UDDI portail : cet annuaire se situe au niveau du pare-feu de l'entreprise, il est accessible depuis l'extérieur. C'est un sorte de serveur publique qui sert à publier les services métiers de l'entreprise.
- L'UDDI catalogue des partenaires : c'est un annuaire interne à l'entreprise mais qui n'est accessible que via ses applications. Dans cet annuaire on rajoute des informations sur les partenaires de l'entreprises et éventuellement sur les services qu'ils proposent.
- L'UDDI de la place du marché : c'est un annuaire qui regroupe des informations sur les entreprises d'un même métier.

Références

[Cha02] Jean-Marie Chauvet. *Services Web avec SOAP, WSDL, UDDI, ebXML... Solutions d'entreprise*. Eyrolles, 2002.

[Cur02] Francisco Curbera. Unraveling the web services web : An introduction to soap, wsdl, and uddi. *Computer Society website*, March/April 2002. http ://www.computer.org/-internet/v6n2/w2spot.htm.

[Gir01] Didier Girard. Xml pour l'entreprise. *Application Servers*, juin 2001. http ://-www.application-servers.com/pagePublications.jsp.

[Got02] K. Gottschalk. Introduction to web services architecture. *IBM System Journal, Volume 41, Number 2, 2002 New Developments in Web Services and E-commerce*, 2002. http :/-/www.research.ibm.com/journal/sj/412/gottschalk.html.

[ISO70] ISO. International organization for standardization website. *ISO website*, 1970. http :/-/www.iso.ch.

[Kre01] Heather Kreger. Web services conceptuale architecture. *IBM Website*, May 2001. http ://dwdemos.alphaworks.ibm.com/wstk/common/wstkdoc/doc/-WebServicesArchitecture.pdf.

[Leb02] Gérard Leblanc. *C Sharp et .Net deuxième édition d'après la version finale de visual studio .net*. Eyrolles, 2002.

[Man01] Anne Thomas Manes. enabling open, interoperable, and smart web services. *W3C workshop on Web services*, 11-12 April 2001. http ://www.w3.org/2001/03/WSWS-popa/paper29.

[Meg01] David Megginson. The official website for sax. *Sax project website*, November 2001. http ://www.saxproject.org/.

[OAS02] OASIS. Uddi : Universal description, discovery and integration. *OASIS website*, 2002. http ://www.uddi.org.

[OMG03] OMG. Omg idl : Details. *OMG website*, May 2003. http ://www.omg.org/-gettingstarted/omg_idl.html.

[Som02] Frank Sommers. A birds-eye view of web services. *Java World*, 25 January 2002. http ://www.javaworld.com/javaworld/jw-01-2002/jw-0125-webservices.html.

[W3C86] W3C. Overview of sgml resources. *W3C website*, 1986. http ://www.w3.org/MarkUp/-SGML/.

[W3C99a] W3C. Html 4.01 specification. *W3C website*, December 1999. http ://www.w3.org/-TR/html4/.

[W3C99b] W3C. Namespaces in xml. *W3C website*, January 1999. http ://www.w3.org/TR/-1999/REC-xml-names-19990114/.

[W3C00a] W3C. Dtd ; document type definition. *W3C website*, October 2000. http ://-www.w3.org/TR/2000/CR-SVG-20001102/svgdtd.

[W3C00b] W3C. Extensible markup language (xml) 1.0 (second edition). *W3C website*, October 2000. http ://www.w3.org/TR/REC-xml.

[W3C00c] W3C. Simple object access protocol (soap) 1.1 (w3c note). *W3C website*, May 2000. http ://www.w3.org/TR/SOAP/.

[W3C00d] W3C. Well-formed xml documents. *W3C website*, October 2000. http ://www.w3.org/-TR/REC-xml.

[W3C01a] W3C. Web services description language (wsdl) 1.1 (w3c note). *W3C website*, March 2001. http ://www.w3.org/TR/wsdl.

[W3C01b] W3C. Xml schema part 0 : Primer. *W3C website*, May 2001. http ://www.w3.org/-TR/2001/REC-xmlschema-0-20010502/.

[W3C03a] W3C. Document object model (dom). *W3c website*, January 2003. http ://-www.w3.org/DOM/.

[W3C03b] W3C. Web services glossary. *W3C website*, May 2003. http ://www.w3.org/TR/ws-gloss/.