Nouvelles Technologies de la Répartition

Mikael Desertot

mikael.desertot@univ-valenciennes.fr

Détail du cours

- Volume horaire
  - 12h CM / 6h TP (+ 6h TD bonus FA)
- Plan du cours
  - Intro Web Services
  - XML / objets
  - Frameworks WS
  - JEE et WS
  - +…

Introduction

Besoin applicatifs

- Le Web classique
  - Conçu pour les applications à interaction humaine
  - Partage d'information
  - E-commerce de type B2C (Business to Consumer)
- Extensions nécessaires
  - Accès à l'information depuis des programmes. Contexte A2A (Application to Application)
  - Processus de e-commerce B2B (Business to Business) automatisés
  - Interopérabilité entre applications
  - Plate-forme pour le calcul distribué

Alternatives existantes

- RPC (Remote Procedure Call) 1980
- CORBA debut 1990
- COM puis DCOM mi 1990
- RMI mi 1990

- Pourquoi ne pas réutiliser les technologies existantes ?
  - Pas de standards

Evolution des intergiciels: RPC

- Principe: appel de procédure type client / serveur s'exécutant sur une machine distante, environnement d'applications distribuées.
  - Fonctionnement synchrone
  - Facile à comprendre et à coder
  - Complexe à administrer
  - Implantation spécifique à chaque vendeur
  - Purement impératif, pas d'abstraction objet

Evolution des intergiciels: CORBA

- Norme de communication pour l'échange entre objets logiciels hétérogènes
- IDL, Interface Definition Language, décrit les traitements effectués et les formats d'entrée / sortie
- Avantages:
  - Indépendant du système d'exploitation et du langage de programmation
  - Standardisé par l'OMG
  - Intègre les aspects métiers
- Inconvénients
  - Incompatibilité entre implantations
  - Complexe à appréhender et administrer
  - Problème de pare-feu
  - Nécessite des changements dans les applications

Evolution des intergiciels: RMI

- Permet l'appel, l'exécution et le renvoi du résultat d'une méthode exécutée dans une machine virtuelle différente de celle de l'objet appelant.
- Avantages
  - Plus simple que l'utilisation directe des sockets
  - POO
  - Transparence des communications
  - Gestion distribuée des ressources
- Inconvénients
  - Limité à Java
  - Pas de gestion de session

Evolution des intergiciels: COM/DCOM

- Modèle de Microsoft pour le développement de composants logiciels
- Réutilisable, orienté objet
- Protocole orienté connexion
- Avantages
  - Simple d'utilisation
- Inconvénients
  - Spécifique Microsoft
  - Gestion des sessions
  - Pas de portabilité du source

Evolution des intergiciels: Composants

- Brique pour la construction d'applications
- Offre des interfaces spécifiées

- Buts
  - Diminuer la complexité de la conception des applications
  - Améliorer la productivité
  - Améliorer la qualité logicielle

Caractéristiques des Web Services (WS)

- Applications
  - Auto descriptives
  - Modulaires
  - Faiblement couplées
- Modèle de programmation et de déploiement simple
- Repose sur des normes
- S'exécute sur une infrastructure Web

Positionnement des Web Services

- Interaction via les protocoles Internet et XML/JSON entre
  - Applications
  - Ordinateurs
  - Processus métier
- S'appuie sur des standards
  - HTTP + XML + JSON + SOAP + WSDL + UDDI +Composant Logiciels

Connexion de WS

- Une fois déployé, un WS peut être invoqué par une autre application
- Services répartis
  - Invocation par réseaux locaux ou étendus
- Un WS peut être
  - Une application autonome
  - Un ensemble d'application
  - Un ensemble de composant

Lien avec les composants

- Fonctions "boites noires" réutilisables
- Pas de protocole spécifique (DCOM, RMI, IIOP) mais standard XML / JSON

- Les infrastructures à base de composant constituent les principales plates-formes d'intégration de WS

Motivations

- WS = ouverture du SI vers d'autres usages, d'autres besoins ou des clients extérieurs
- Permettent de:
  - Automatiser facilement les processus métiers
  - Faciliter l'interopérabilité entre systèmes et plates-formes hétérogènes
  - Faciliter l'intégration d'applications et de services

Automatiser les processus métiers

- Assembler différent services (Orchestration, Chorégraphie)
- Intégrer différentes activités dans le SI

Faciliter l'interopérabilité

- Dialogue entre environnements hétérogènes
- Assurent le plus haut niveau d'inter-fonctionnement entre des applications d'entreprise

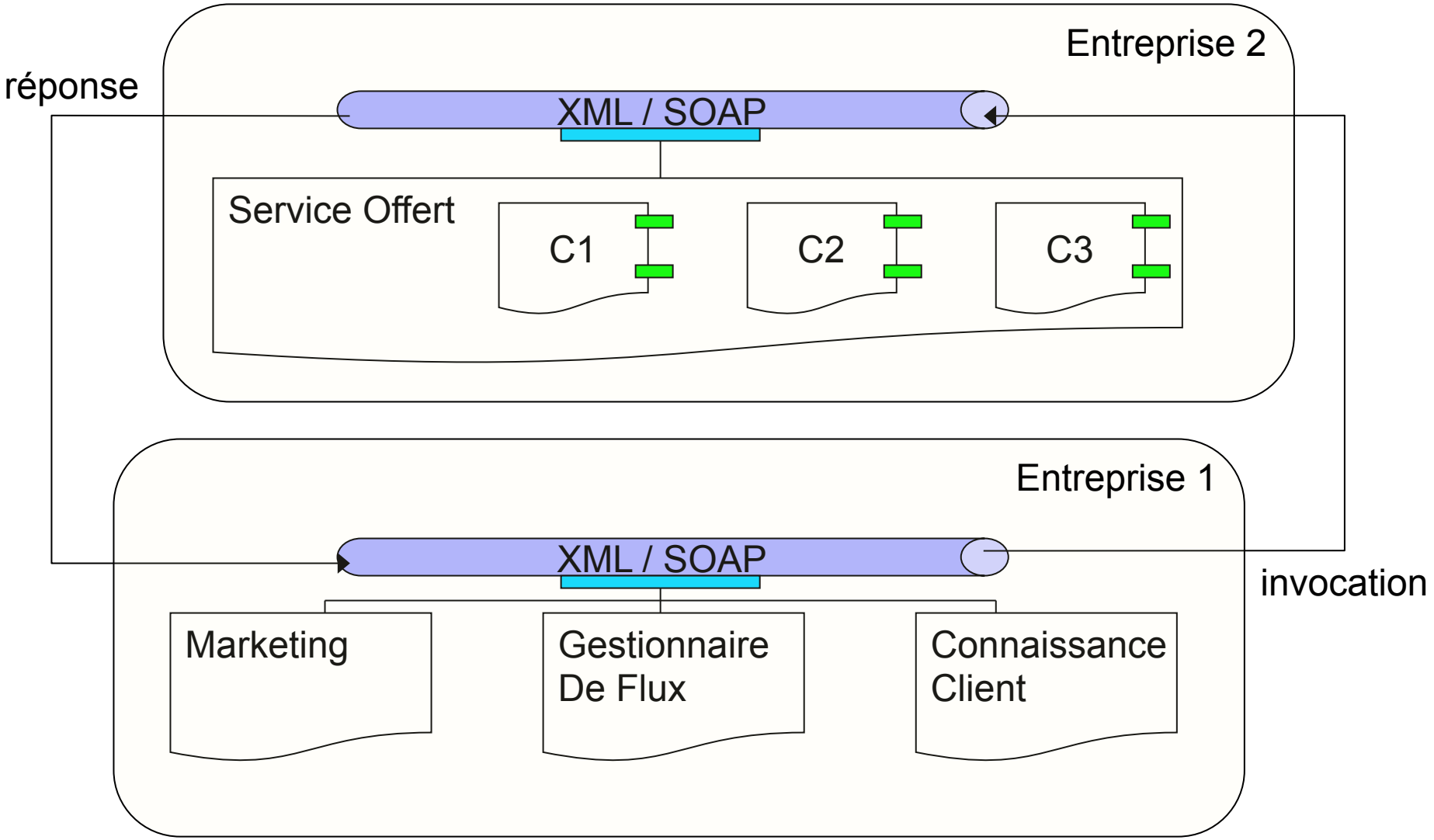
Mise en œuvre (1)

- Dans une logique d'EAI
  - Rapide, peu coûteuse
  - Suffit dans la plupart des cas

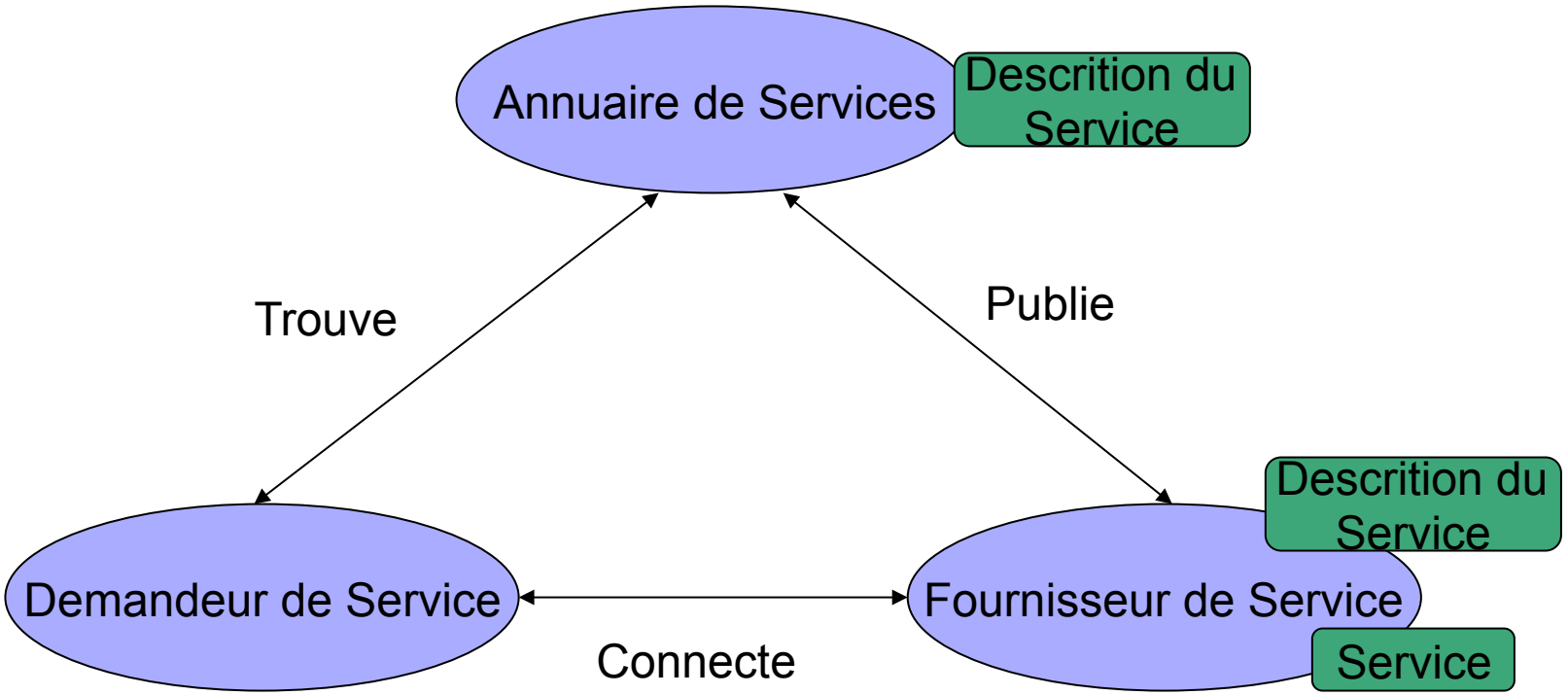
Mise en œuvre (2)

- Intégration en dehors de l'entreprise
  - Collaboration inter entreprise
  - Intégration de services métiers

Exemple


Entreprise 2
réponse
XML / SOAP
Service Offert
C1
C2
C3
Entreprise 1
XML / SOAP
invocation
Marketing
Gestionnaire
De Flux
Connaissance
Client

Modèle d'interaction



Apport de XML / JSON

- Décrit la structure logique des documents
- Pallie aux limites de HTML
  - Sépare contenu, structure et représentation
- Format universel

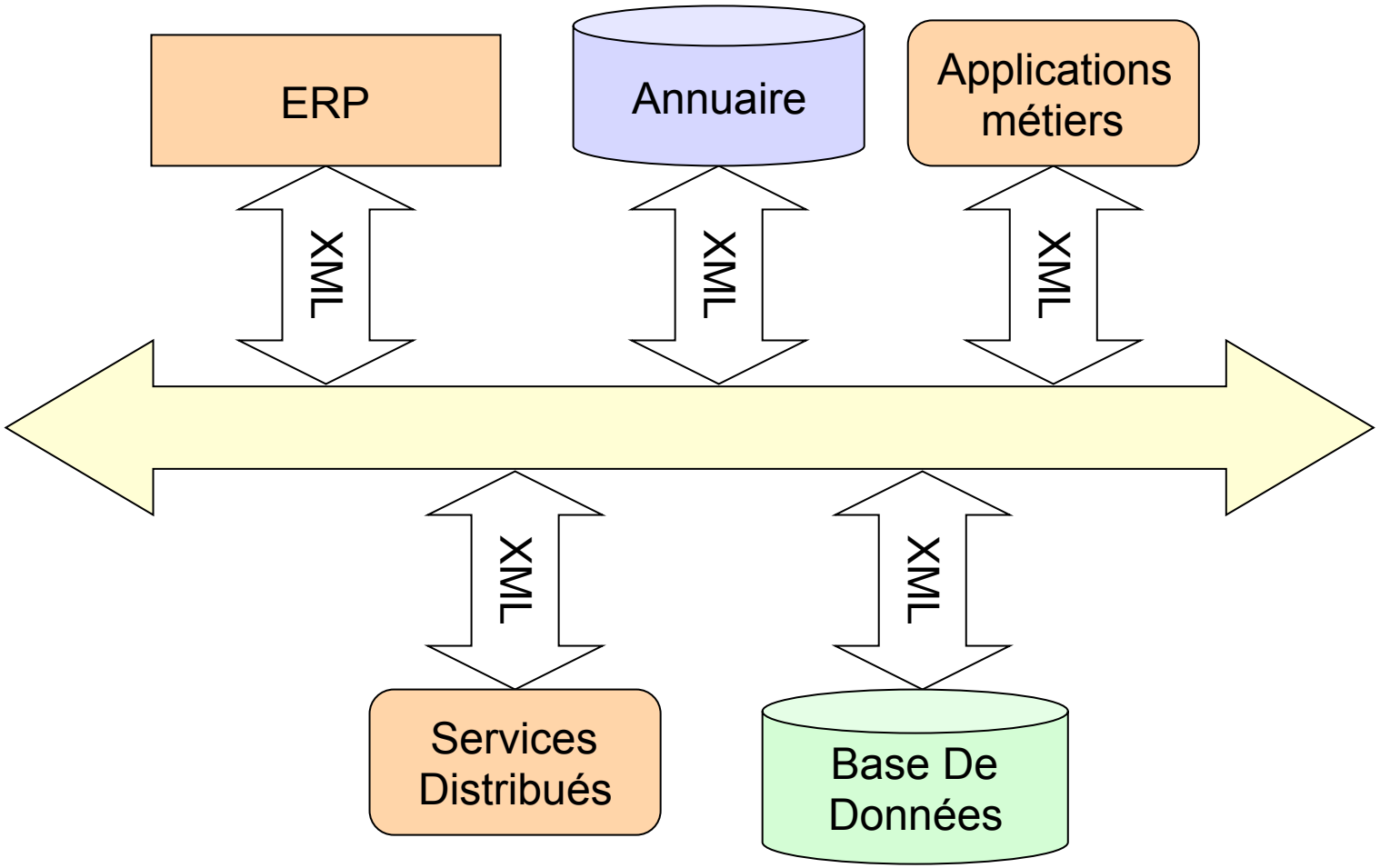
Technologies XML

- XML
  - Noyau du langage
- XML Schema (ex DTD)
  - Structure des documents XML
- XSL
  - Réorganisation et présentation de documents
- DOM, etc.
  - Api pour manipuler des fichiers XML
- Namespaces
  - Cataloguer des vocabulaires

XML et Services distribués

- Java EE, Corba et DCom sont les technologies majeures
- Le modèles des WS est proche de ces technologies
- Il les étend vers l'Internet en assurant l'extension de leur interopérabilité
- Avantage
  - Simplicité de XML
  - Indépendant du système d'exploitation

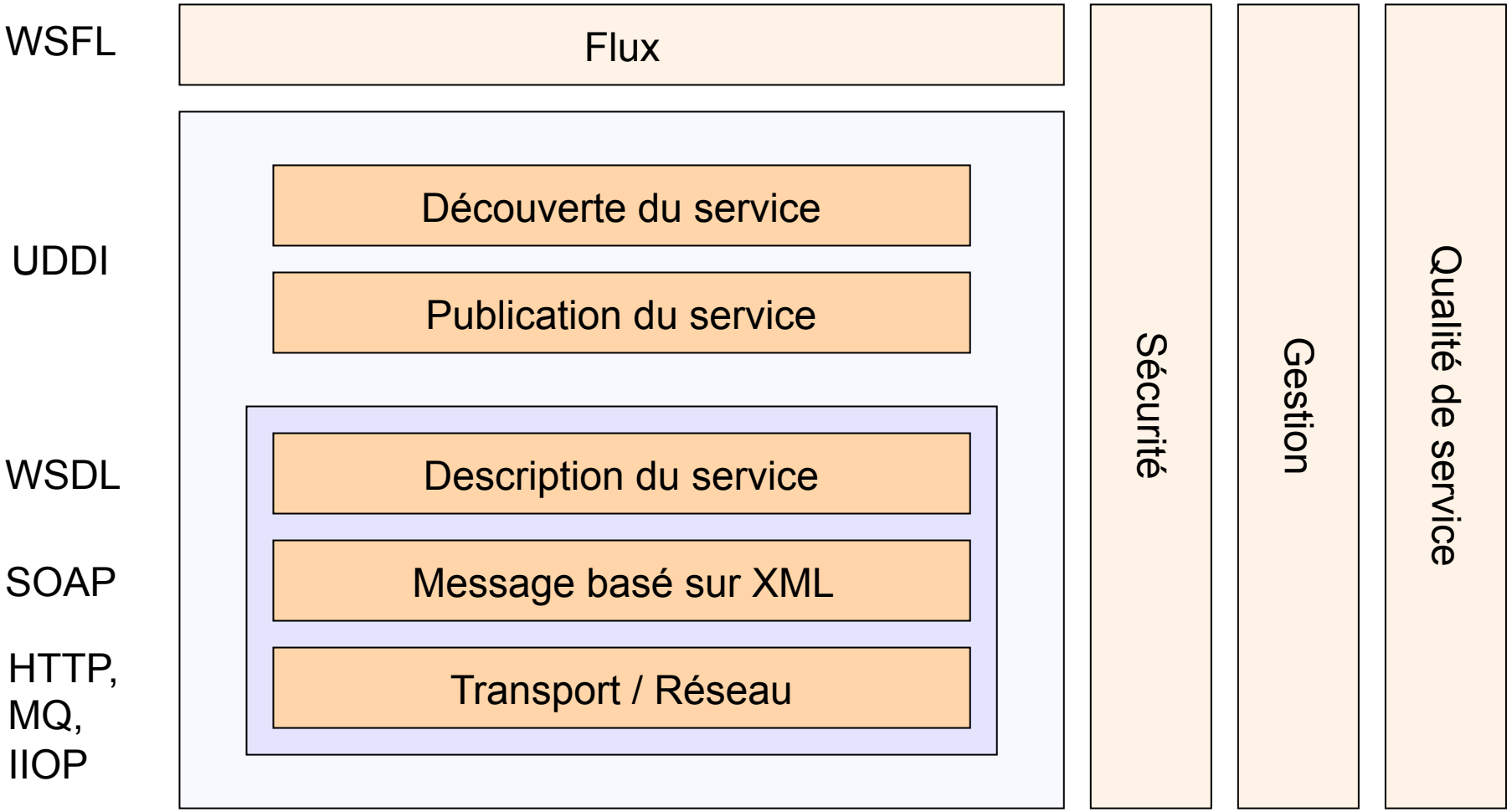
XML et système d'information



Niveaux d'intégration

Intégration de processus Modélisation du workflow ou de processus
Intégration d'applications Translation et transformation de données Routage basé sur des règles
Intégration de composants Serveurs d'applications
Intégration de données Outils d'extraction, transformation et chargement de données
Intégration de plates-formes Messagerie, ORB, RPC

Architecture des Web Services



XML

Un fichier XML

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Toi</to>
  <from>Moi</from>
  <heading>Debut</heading>
  <body>Un message</body>
</note>
```

Un XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.w3schools.com" xmlns="http://www.w3schools.com"
           elementFormDefault="qualified">

           <xs:element name="note">
                      <xs:complexType>
                                 <xs:sequence>
                                            <xs:element name="to" type="xs:string" />
                                            <xs:element name="from" type="xs:string" />
                                            <xs:element name="heading" type="xs:string" />
                                            <xs:element name="body" type="xs:string" />
                                 </xs:sequence>
                      </xs:complexType>
           </xs:element>

</xs:schema>
```

Faire référence au schéma

```
<note xmlns="http://www.w3schools.com" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com note.xsd">

      <to>Toi</to>
      <from>Moi</from>
      <heading>Debut</heading>
      <body>Un message</body>
</note>
```

Définition des éléments

```
<lastname>Nom</lastname>
<age>36</age>
<dateborn>1980-03-27</dateborn>
```

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

+ valeurs par défaut, fixées

Définition des attributs

```
<lastname lang="EN">Smith</lastname>
```

```
<xs:attribute name="lang" type="xs:string"/>
```

```
<xs:attribute name="lang" type="xs:string"
                          use="required"/>
```

Restrictions

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Types complexes

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Extensions

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Ordonnancement (1)

```
<xs:element name="person">
  <xs:complexType>
    <xs:all>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Ordonnancement (2)

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Ordonnancement (3)

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Nombre d'occurrences

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
                  maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Contraintes

```
<xs:key name="employeKey" id="employeKey">
      <xs:selector xpath="personnel/employe" />
      <xs:field xpath="@id" />
</xs:key>

<xs:keyref name="nomKey" refer="employeKey">
      <xs:selector xpath="personne/nom" />
      <xs:field xpath="@id" />
</xs:keyref>
```

Vers Java

Bijections Java - XML (1)

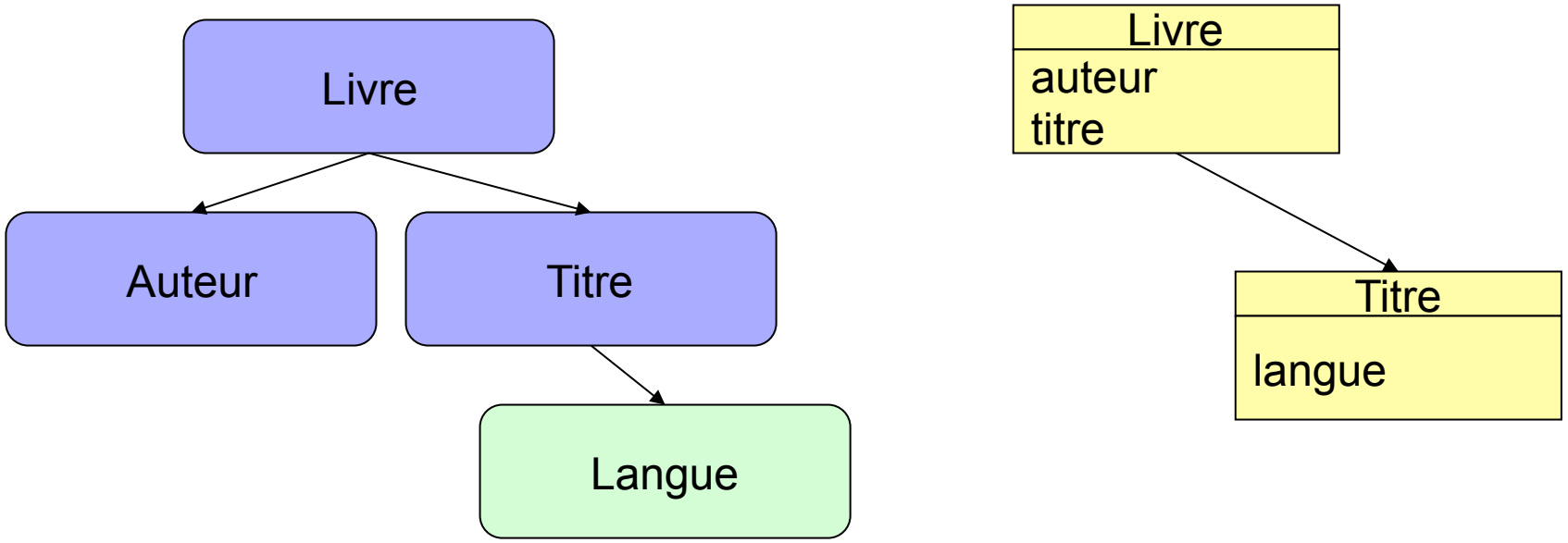
- Problème
  - Préserver le modèle Objet

    ou
  - Préserver le modèle de document
- Exemple

```
<livre>
        <auteur>Herge</auteur>
        <titre langue="francais">Tintin à l'ISTV</titre>
</livre>
```

Bijections Java - XML (2)

- On peut lui faire correspondre un objet de classe Livre



Bijections Java - XML (3)

- Doit-on créer une classe titre ?
- Un champs Java correspond-il à un élément XML ou à un attribut ?
- Comment une séquence se traduit-elle en Java ?
- Si la séquence comporte des éléments de balises différentes, faut-il créer une nouvelle classe pour cette séquence ?

Bibliothèques Java-XML

- JAXB
  - De Sun, intégré dans Java
- JIXB
  - L'équivalent dans le monde Apache
- ATOM
  - Mapping objet d'Apache
- Castor
  - Encore une solution en logiciel libre
- ADB
  - Axis Databinding Framework, alternative a JIXB chez Apache

Conversion en Java

- Classe Java

```
@XmlRootElement
class Point {
      int x;
      int y;
      Point(int _x,int _y) {x=_x;y=_y;}
}
```

Conversion en Java

- XML Généré

```
//Example: Code fragment corresponding to
  XML output
marshal( new Point(3,5), System.out);

<!-- Example: XML output -->
<point>
  <x> 3 </x>
  <y> 5 </y>
</point>
```

Conversion en Java

- XML Schéma

```
<!-- Example: XML schema definition -->
<xs:element name="point" type="point"/>
<xs:complexType name="point">
  <xs:sequence>
     <xs:element name="x" type="xs:int"/>
     <xs:element name="y" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
```

Conversion en Java

- Classe Java

```
@XmlRootElement
class Point3D extends Point {
          int z;
          Point3D(int _x,int _y,int _z) {
                super(_x,_y);z=_z;
           }
}
```

Conversion en Java

- XML Généré

```
//Example: Code fragment corresponding to
  XML output
marshal( new Point3D(3,5,0), System.out );

<!-- Example: XML output -->
<point3D>
  <x>3</x>
  <y>5</y>
  <z>0</z>
</point3D>
```

Conversion en Java

- XML Schéma

```
<!-- Example: XML schema definition -->
  <xs:element name="point3D" type="point3D"/>
  <xs:complexType name="point3D">
      <xs:complexContent>
            <xs:extension base="point">
                  <xs:sequence>
                     <xs:element name="z" type="xs:int"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
  </xs:complexType>
```

Conversion en Java

- Génération

```
@XmlRootElement
class Person {

private int _id;
private String _name;

public Person(){}

public Person(int id, String name){
_id = id;
_name = name;
} //fin construct Person

@XmlAttribute
public int getId(){ return _id; }
public void setId(int id){ _id = id; }

@XmlAttribute
public String getName(){ return _name; }
public void setName(String name){ _name = name; }
} //fin class Person
```

Conversion en Java

- Code Java

```
public class Main {

public static void main(String[] args) {
try {

// on crée un contexte JAXB pour la classe Person
JAXBContext context = JAXBContext.newInstance(Person.class);

// on crée un marshaller à partir du contexte
Marshaller m = context.createMarshaller();

// on veut un affichage formatté
m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

// on demande au marshaller de générer le XML de la personne "serge"
// et de l'afficher dans la console
m.marshal(new Person(1, "serge"), System.out);

} catch (JAXBException ex) {
ex.printStackTrace();
} //fin catch
} // fin main
} // fin class Main
```

Résultat

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<personne id="1" name="serge"/>

A vous de jouer :

- Définir un schéma pour la structure de donnée suivante :
  - Une bibliothèque est une liste d'ouvrage
  - Chaque ouvrage a un isbn d'attribué
  - Un ouvrage est défini par un titre, un ou plusieurs auteurs, un editeur, un prix et éventuellement une liste de références
  - Une liste de références contient des livres avec leurs isbn attribués <livre isbn="1234"/>

A vous de jouer

- Extraire les type références et ouvrage dans un autre XML shema à importer
- Gestion des namespaces

A vous de jouer

- Placer des contraintes pour que les références vers des ISBN de livres soient toujours valables

A vous de jouer

- Générer avec Eclipse les classes Java depuis votre xsd
- Regarder ce qui est généré.
- Faire un essai de lecture de fichier XML
- Faire un essai de génération de fichier XML

"Lire un XML"

```
JAXBContext jc = JAXBContext.newInstance("org.generated.biblio");
Unmarshaller unmarshaller = jc.createUnmarshaller();

SchemaFactory scfac =
      SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);

Schema sc = scfac.newSchema(new File("src/Cours3.1.xsd"));
unmarshaller.setSchema(sc);
Biblio bibliotheque =
       (Biblio) unmarshaller.unmarshal(new File("src/Cours3.xml"));

List<OuvrageType> listeO = bibliotheque.getOuvrage();
for (OuvrageType ot : listeO) {
       System.out.println(ot.getTitre());
}
```

"Ecrire un XML"

```
JAXBContext jaxbContext =
                JAXBContext.newInstance("<PACKAGE>");

Marshaller marshaller = jaxbContext.createMarshaller();

marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
                                                new Boolean(true));

marshaller.marshal(bibliotheque, new FileOutputStream(f));
```