Ecole Nationale Supérieure de Tunis
Département Informatique

ARCHITECTURE ORIENTÉE SERVICES

Niveau: 2ème Année Génie Informatique

Enseignante: Inès Elouèdi

Email: ielouedi@gmail.com

Plan du cours

Chapitre1: Introduction aux web services

Chapitre2: WSDL, SOAP, UDDI

Chapitre3: Web Services RESTFUL

Travaux Pratiques:

1. Web Services SOAP
2. Web Services Restful

CHAPITRE1:INTRODUCTION AUX WEB SERVICES

Learning OutComes

- Quels sont les critères de performance d'une Application Web?
- Un web service? C'est Quoi?

- Pourquoi Le web Service?

Plan du chapitre1 :

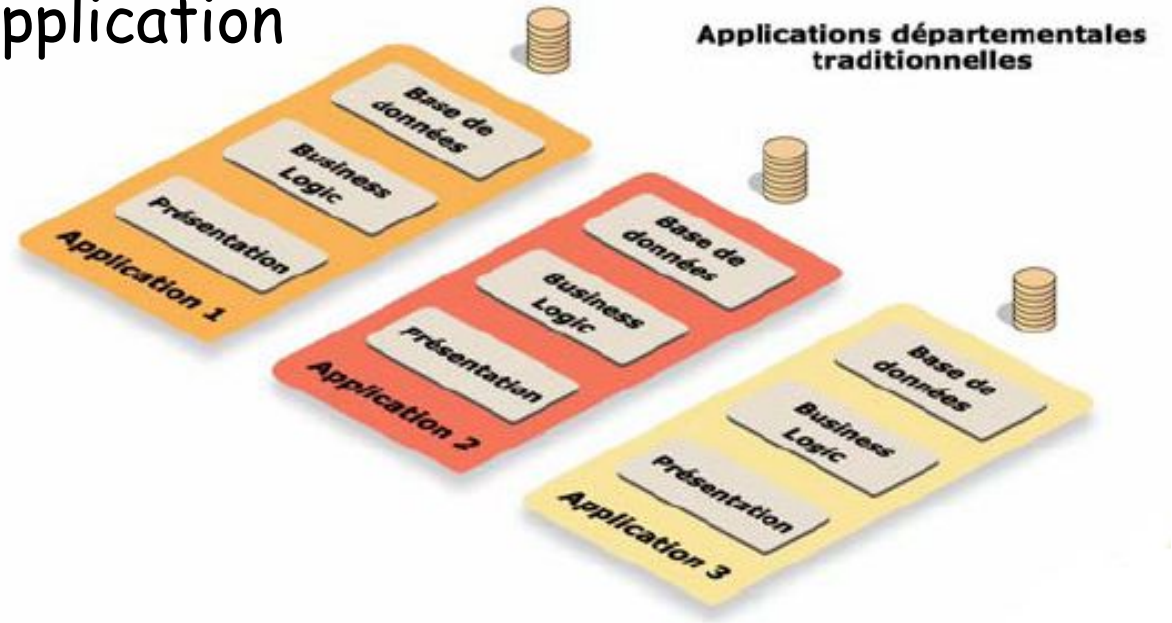
1. Exigences d'un projet informatique
2. Historique
3. Evolution des architectures
4. Les Services Web
    1. Les standards SOAP, WSDL, UDDI
    2. L'Architecture REST
    3. Scénario de service web
    4. Caractéristiques et fonctionnalités

Exigences d'un projet Informatique

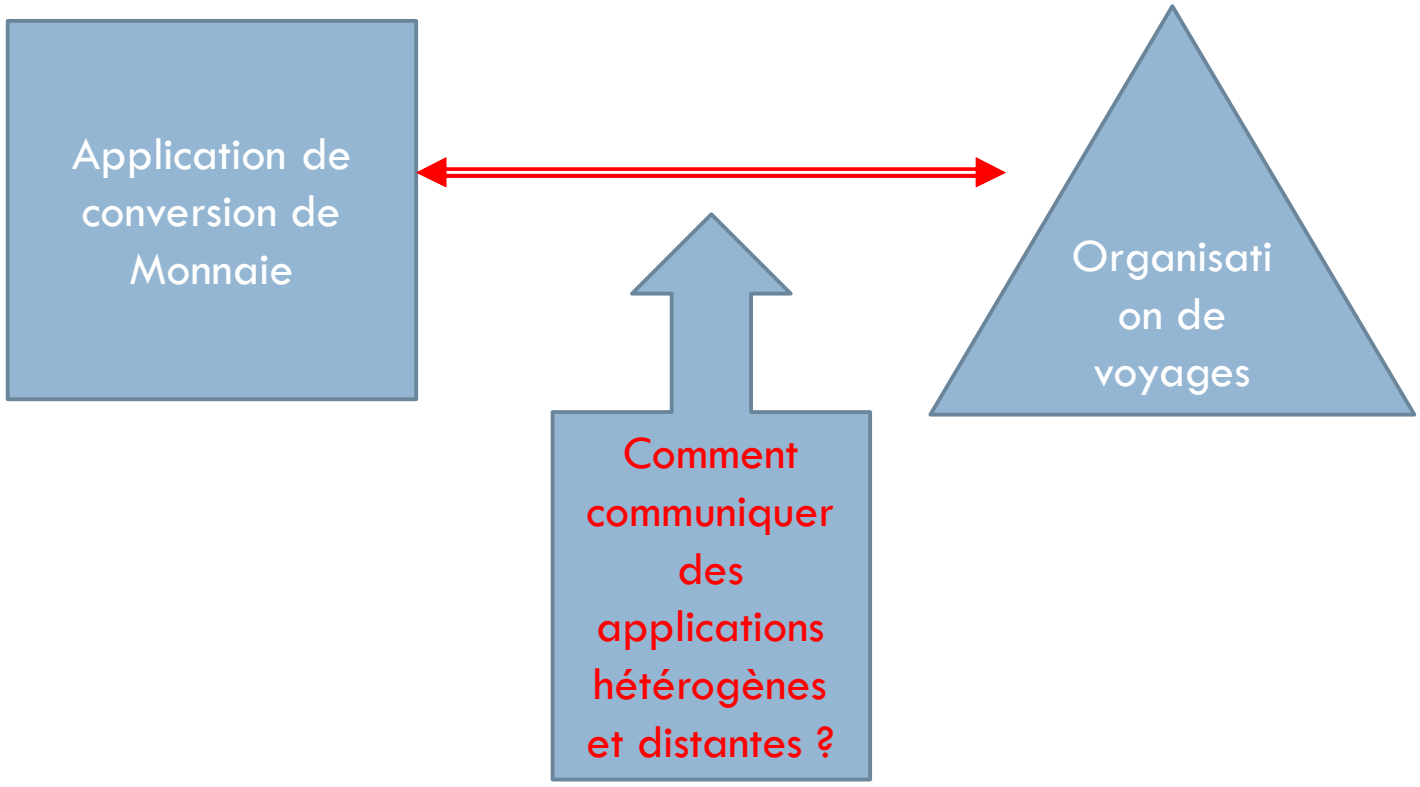
- Exigences fonctionnelles
  - Une application est tout d'abord créée pour répondre aux besoins fonctionnels de l'entreprise.
- Exigences techniques
  - Les performances
    - réduire le temps de réponse,
    - Haute disponibilité et tolérance aux pannes
    - Eviter le problème de montée en charge
    - Interfaces responsives, Réutilisation du code,,
  - Maintenance
    - Exprime la simplicité de correction et de modification du logiciel
    - Possibilité d'extension et de réutilisation
  - Sécurité
  - Portabilité
  - Distribution
  - Capacité de communication avec d'autres applications distantes
  - Coût

Problématique de l'intégration en entreprise

- Le découpage présentation/traitement/base de données de l'architecture 3-tiers favorise le cloisonnement en silos applicatifs indépendants (blocs monolithiques)
- Certaines fonctions sont redondantes : une version pour chaque application

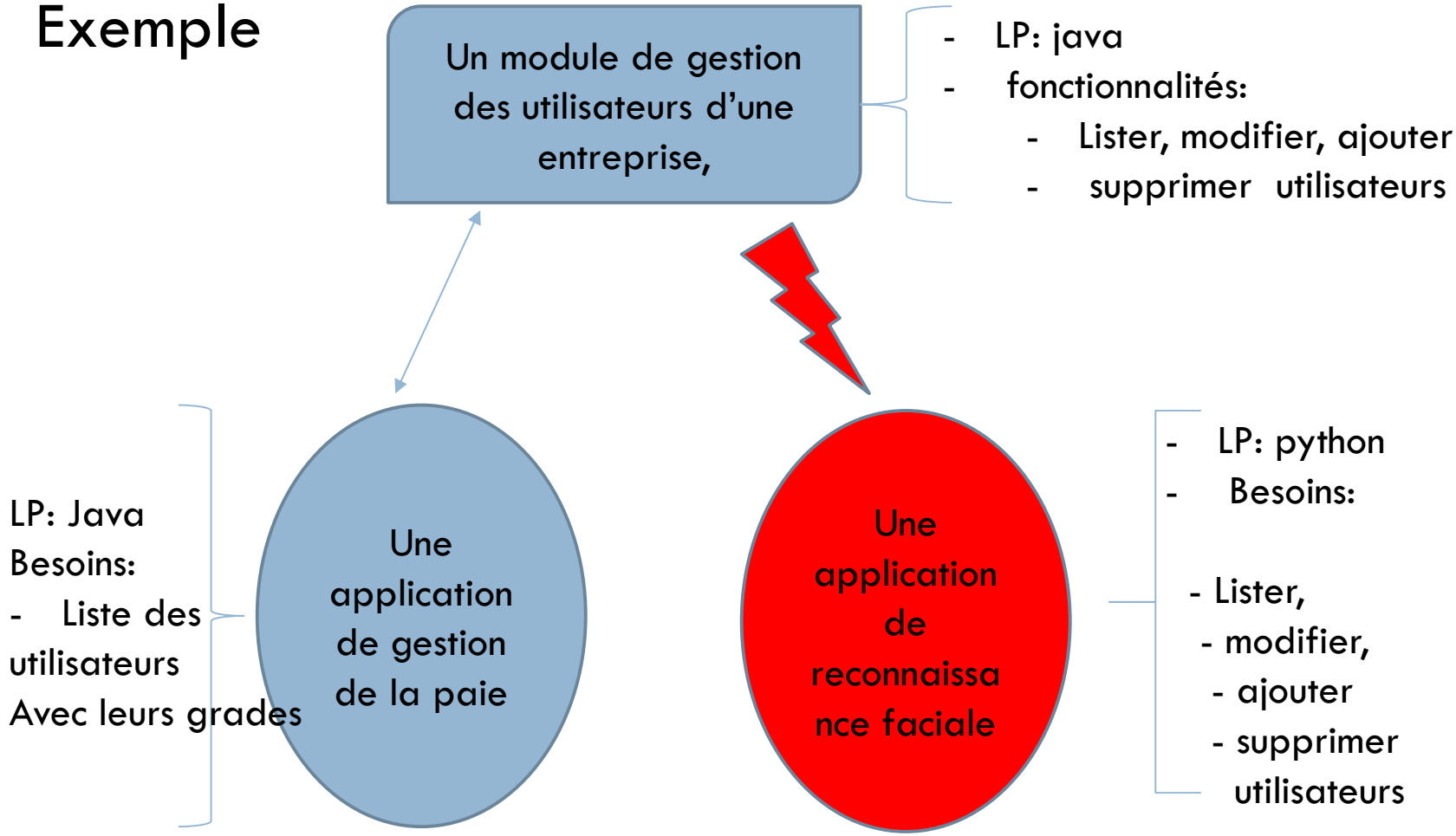


Problématique d'interopérabilité

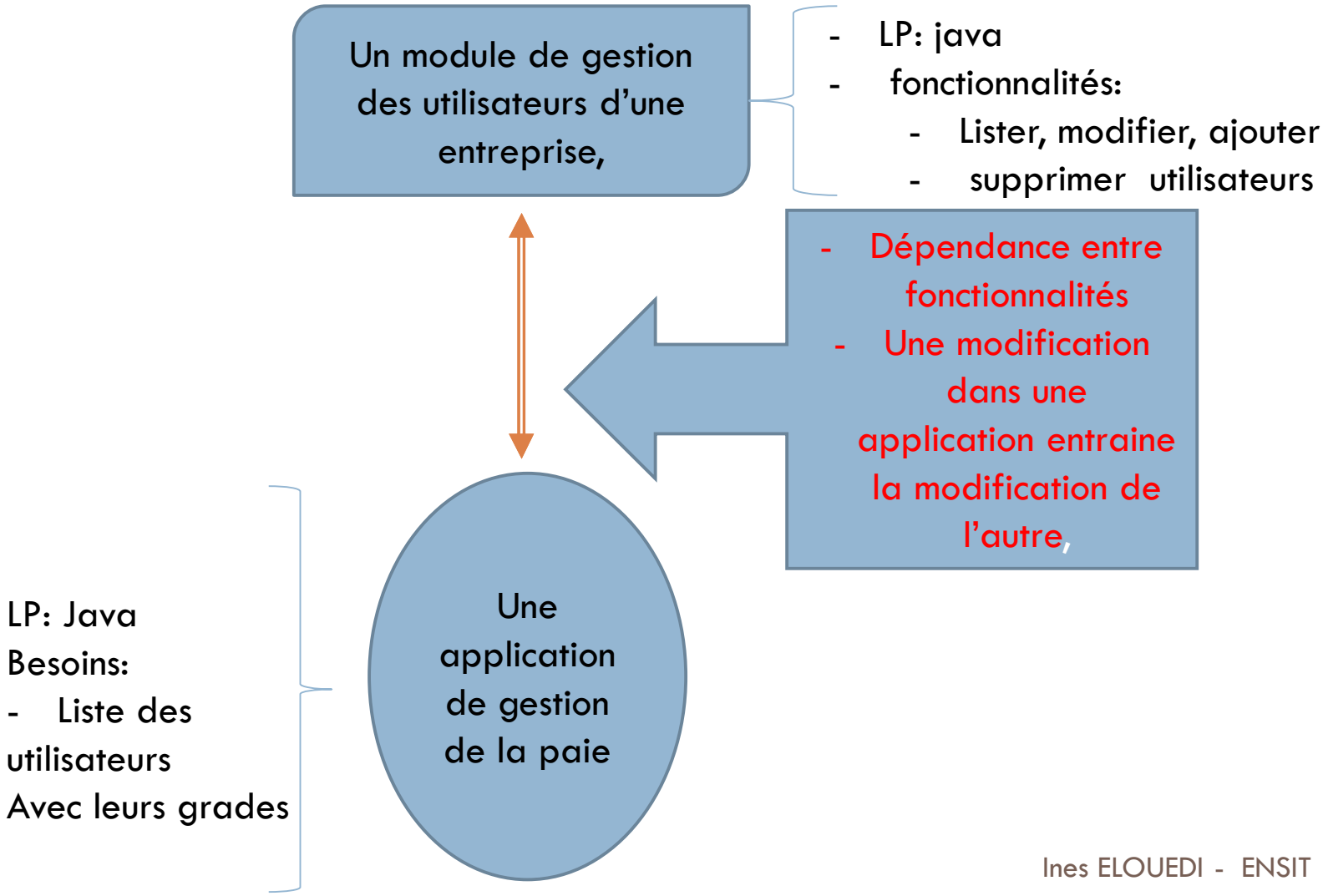


Problématique de Réutilisation du code

- Exemple

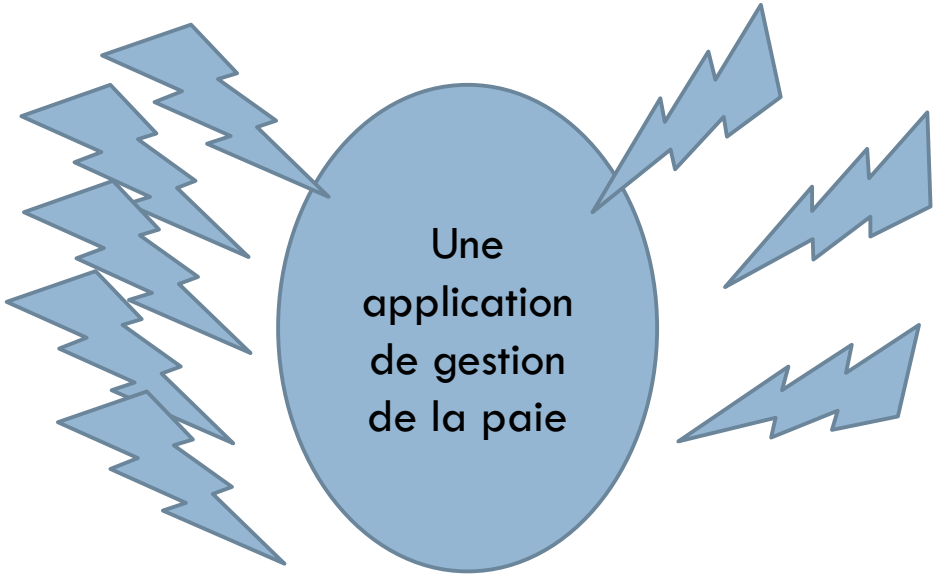


Problématique du couplage fort



Problématique de la montée en charge

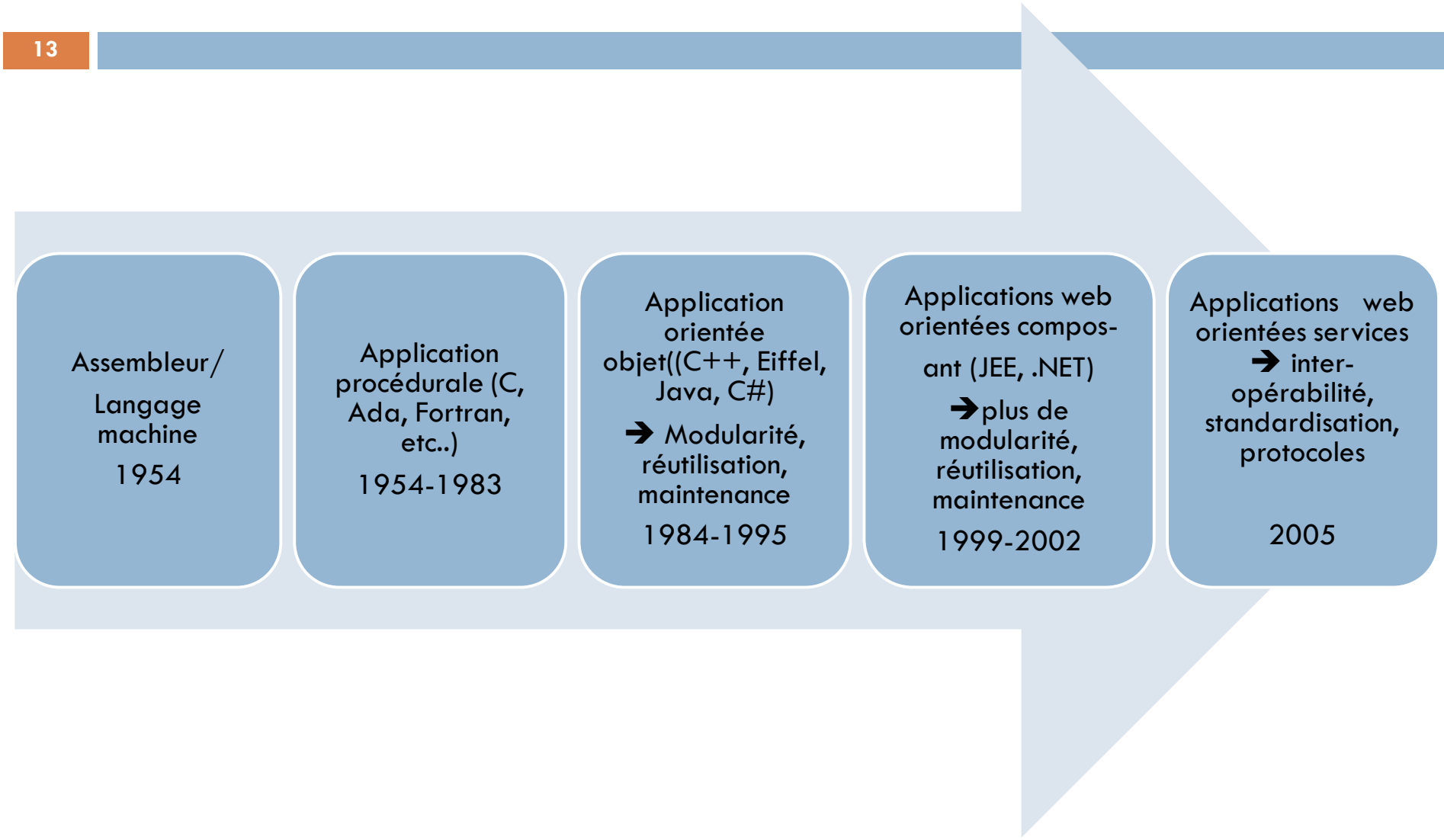
- comment faire en sorte que les applications proposées en ligne soient **toujours disponibles** et ne souffrent jamais de coupure ou de ralentissement, quelle que soit l'affluence des utilisateurs sur celles-ci ?



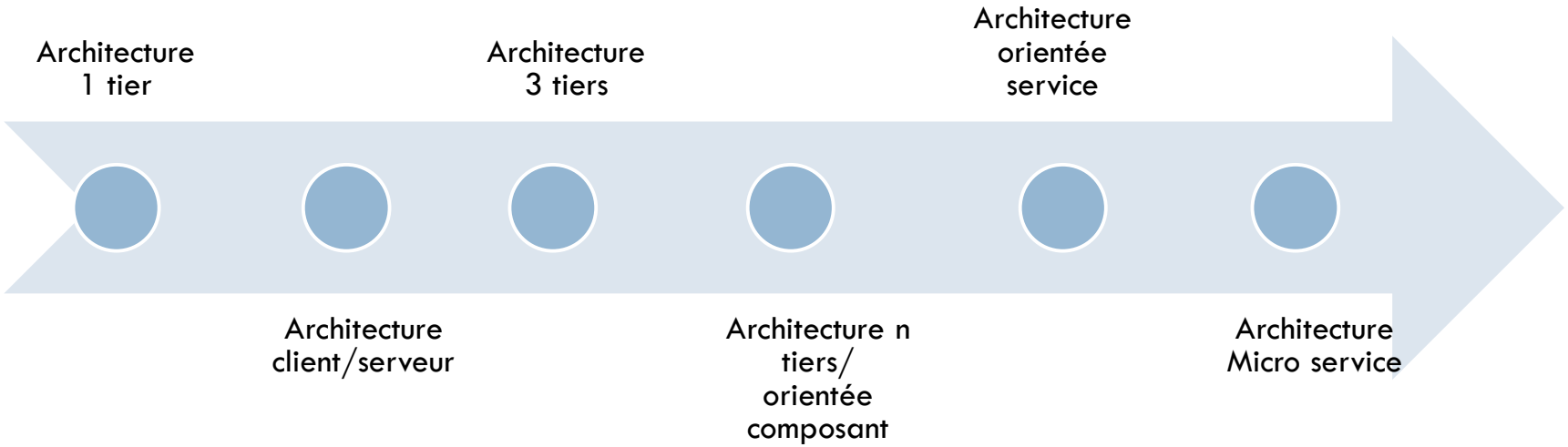
Solution?

- Modules indépendants, autonomes
- Faiblement couplés aux autres modules
- Sécurisé
- Facilement accessibles par la communication
- inter-opérable

Historique


Assembleur/
Langage machine
1954
Application procédurale (C, Ada, Fortran, etc..)
1954-1983
Application orientée objet((C++, Eiffel, Java, C#)
➔ Modularité, réutilisation, maintenance
1984-1995
Applications web orientées compos-ant (JEE, .NET)
➔plus de modularité, réutilisation, maintenance
1999-2002
Applications web orientées services
➔ inter-opérabilité, standardisation, protocoles
2005

Evolution des architectures



Architecture 1tier- Main Frame (1)

- Les utilisateurs se connectent aux applications exécutées par le serveur central (*mainframe) à l'aide de* ***terminaux passifs***
- le serveur central prend en charge la gestion des données et des traitements, y compris l'affichage qui est transmis sur des terminaux passifs.

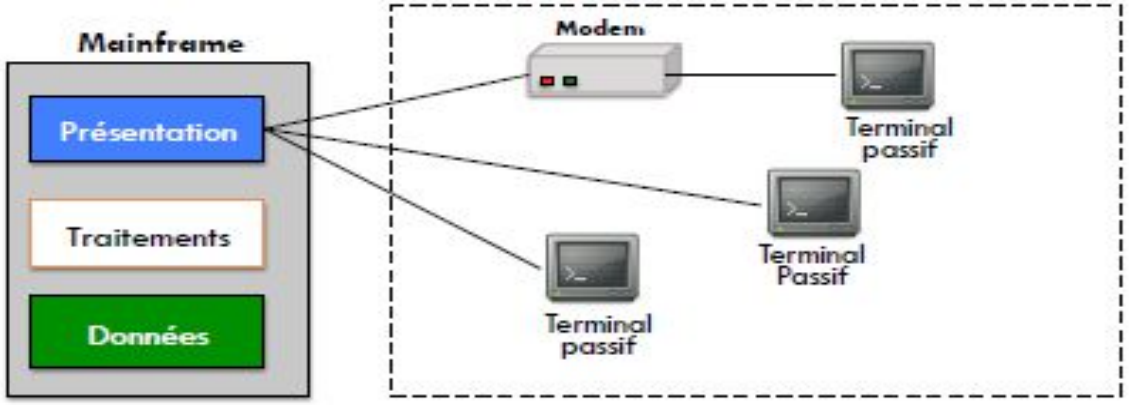


Figure tirée de [4]

Architecture 1tier- Main Frame (2)

Limites

- Interface utilisateur en mode caractères est obsolète
- utilisateur passif

Avantages

- fiabilité de l'application grâce à la gestion centralisée des données

Solution

répartir l'application en parties distinctes et coopérantes :

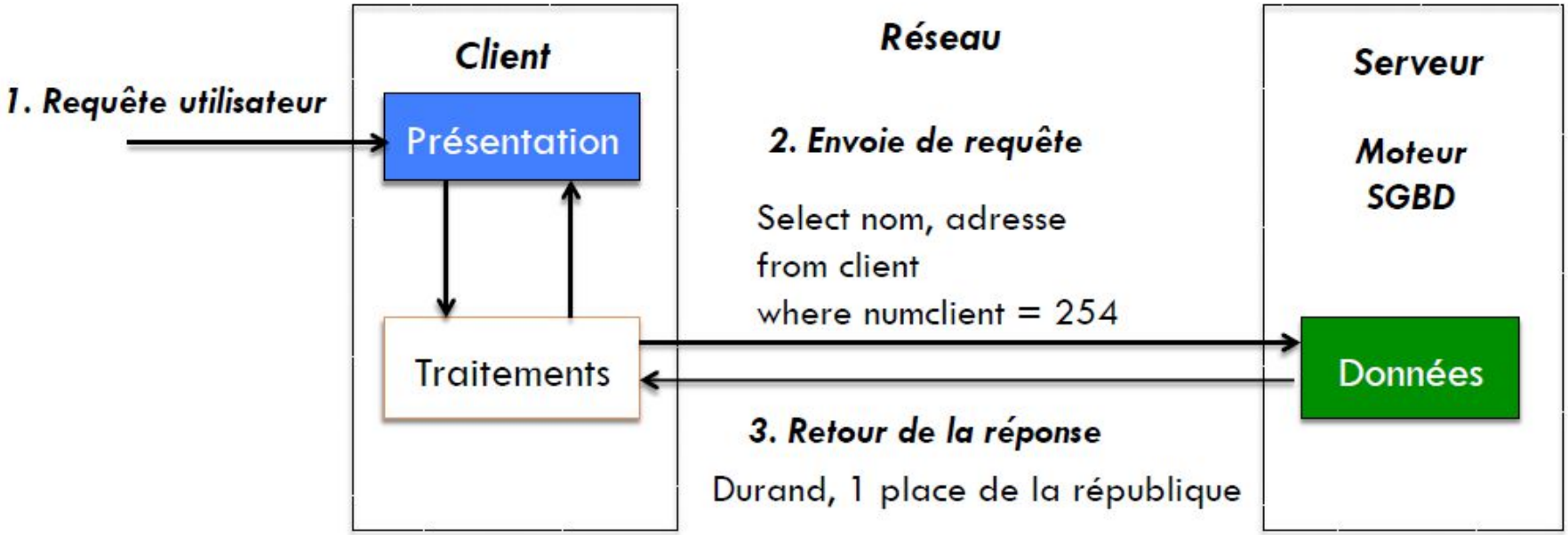
- gestion centralisée des données
- gestion locale de l'interface utilisateur

→ Architecture client/serveur

Architecture 2 tiers- Client/Serveur (1)

- Architecture 2-tiers ou C/S de première génération ou C/S de données
  - Le client s'occupe de la présentation et la logique applicative
  - Le serveur s'occupe de la gestion des données
- Exemple : Application de gestion de stock fonctionnant sur Windows et exploitant un SGBD (oracle) centralisé.

Architecture 2 tiers- Client/Serveur(2)



[4]

La gestion des données est prise en charge par un SGBD centralisé, s'exécutant le plus souvent sur un serveur dédié

➔ serveur de données

Le serveur de données est interrogé en utilisant un **langage de requête qui est, le plus souvent, SQL**

Architecture 2 tiers- Client/Serveur(3)

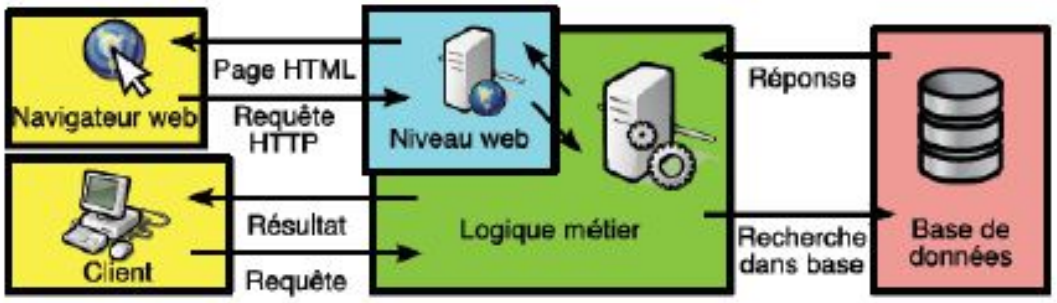
- Limites
  - Charge importante du poste client qui réalise l'ensemble des traitements applicatifs
  - Maintenance et mises à jour difficiles à gérer
  - Conversation entre client et serveur est assez bruyante
  - Ces limites proviennent de type du client : client lourd
  - Frontal complexe et non standard (Windows, Linux, Mac,…)
  - Middleware entre client et serveur n'est pas standard

Architecture 3 tiers (1)

- L'arch. 3-tiers, ou C/S de 2 ème génération, sépare l'application en 3 niveaux:
  - Niveau 1 : l'affichage et les traitements locaux (contrôles de saisie, mise en forme de données...) sont pris en charge par le poste client
  - Niveau 2 : les traitements applicatifs globaux sont pris en charge par le service applicatif : serveur d'application
  - Niveau 3 : les services de base de données sont pris en charge par le serveur de données

Architecture 3 tiers (2)

- Trois niveaux:
  - Client léger (un navigateur web)
    - présentation de l'application
    - traitements locaux de vérification de saisie et la mise en forme des données
  - Un serveur d'applications (JEE, .NET)
    - Le code applicatif, ou code **métier**, est stocké sur le **serveur** d'application, et est déployé et géré de manière centralisée
    - une **couche web, appelé serveur web, pour communiquer avec le navigateur**
  - Un serveur de bases de données (SGBD)



Architecture 3 tiers (4)

- Avantages:
- client léger et indépendant des modifications au niveau applicatif
- Limites:
  - Le **serveur d'application réalise la majorité des traitements**
  - Problème de gestion de la **montée en charge rappelant** l'époque des mainframes.
  - le **client est soulagé, mais le serveur est fortement sollicité**

➔ L'équilibre de la charge entre client et serveur semble atteint avec la génération suivante : Architecture n-tiers

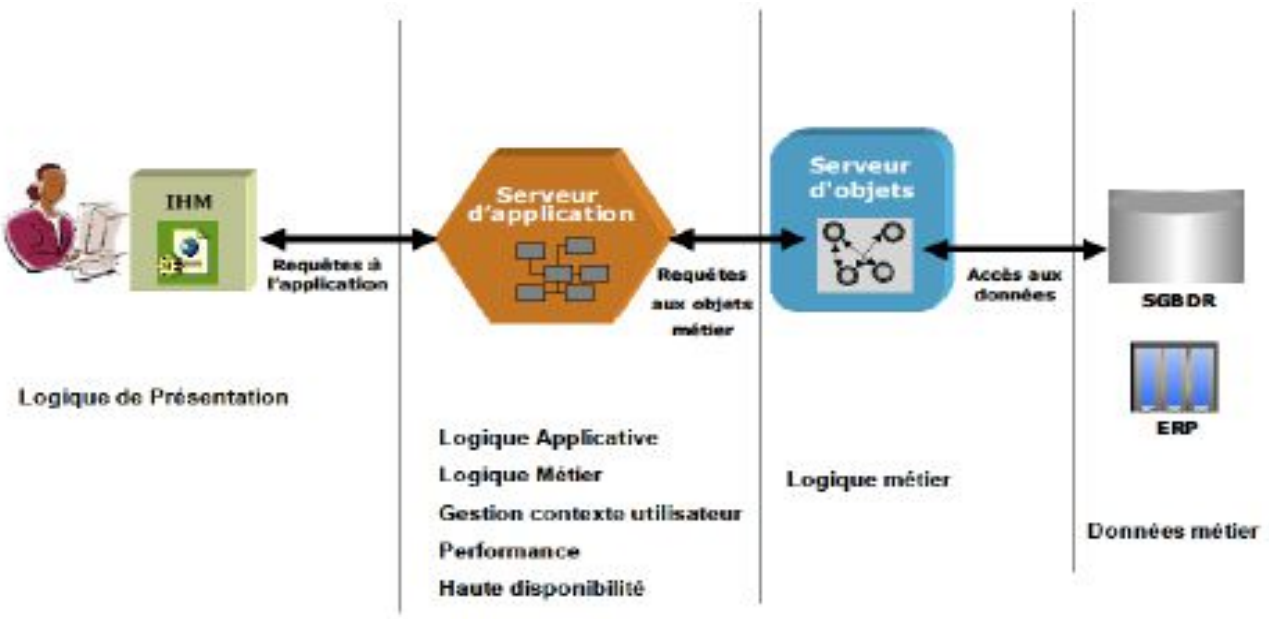
Architecture n-tiers (1)

- **Toujours 3 niveaux d'abstraction : n-tiers ne signifie** pas un nombre indéterminé de niveaux de service (Prés., Log. applicative et Données)

- L'architecture **n-tiers qualifie la distribution de la** logique applicative entre de multiples composants.

Architecture n-tiers (2)

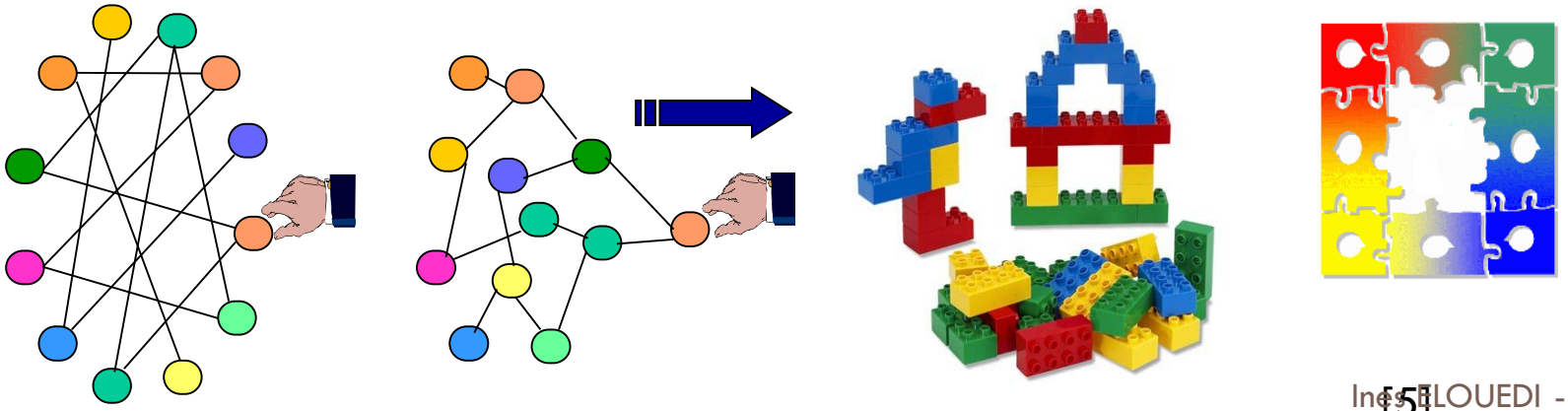
- Une architecture n-tiers comprend généralement une ***couche de présentation, une couche applicative, une couche objets métier et*** une ***couche d'accès aux données***



[5]

Architecture n-tiers (3)

- Qu'est ce qu'un composant?
  - Un composant est un module logiciel
    - autonome
    - rend un **service générique** et clairement **identifié.**
    - **réutilisable**
  - Les composants sont capables de communiquer entre eux et peuvent donc coopérer en étant implantés sur des machines distinctes.
  - granularité plus élevée par rapport à l'orienté objet:



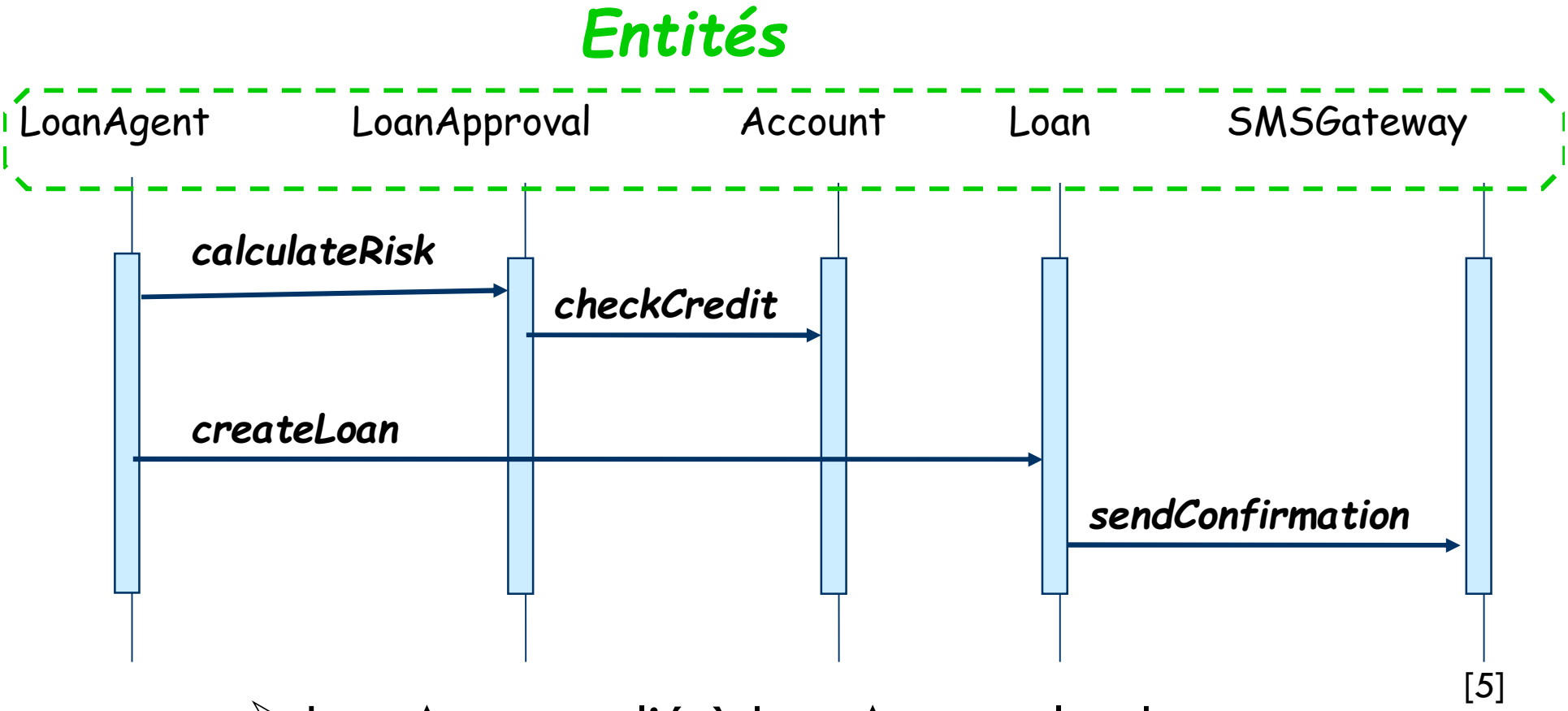
Architecture n-tiers (4)

- Avantages:
  - Modularité,
  - Réutilisation,
  - Plus de distribution,
  - Réduction du problème de montée en charge

Architecture n-tiers (5)

- Limites:
  - Couplage fort entre les composants
  - problème de Maintenance: Une modification d'un composant entraine la modification des autres.

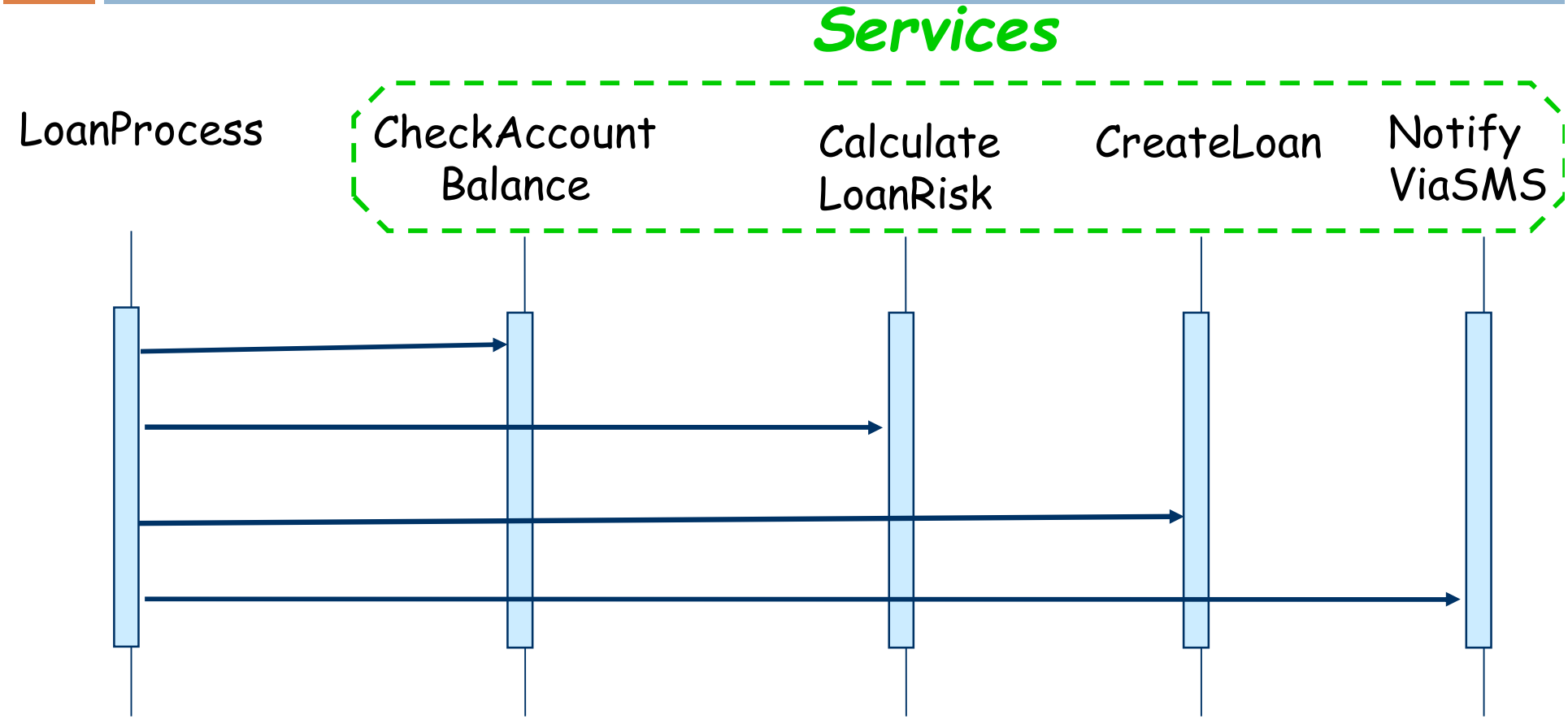
Exemple de couplage fort: Gestion de prêts



[5]

- LoanAgent est lié à LoanApproval et Loan
- LoanApproval est lié à Account
- Loan est lié à SMSGateway

Gestion de prêts en couplage faible



[5]

- Qu'est ce que LoanProcess ?
- Un **processus métier** !
  Il permet d'**orchestrer** les services => couplage lâche

Architecture n-tiers (6)

- Limites
  - Traffic d'échange de messages important entre des plateformes hétérogènes

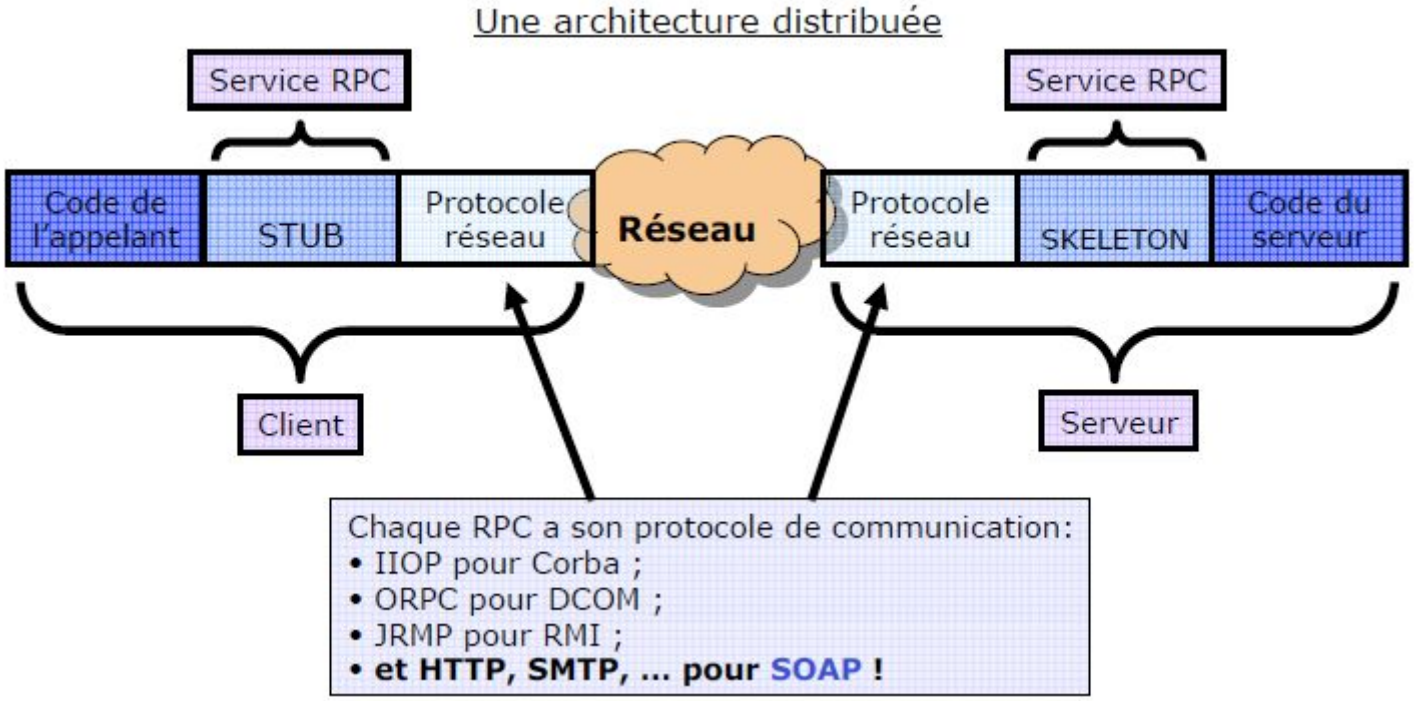
    ➔ Nécessité de l'utilisation des Middlewares pour communiquer

Architecture n-tiers (7)

- Middleware :Intergiciel permettant la communication entre des applications hétérogènes
- Un RPC (Remote Procedure Call), est un mécanisme permettant l'appel local d'une méthode distante.
- Différents langages de RPC existent, dont :
  - RMI *(Remote Method Invocation)*
  - SunRPC
  - DCE *(Distributed Computing Environment)*
  - Corba *(Common Object Request Broker Architecture)*
  - DCOM *(Distributed Component Object Model)*

Architecture n-tiers (8)

- La RPC permet de masquer la différence entre un appel local et un appel distant. Il n'y a donc plus à se soucier de la couche réseau, qui est gérée par le RPC.



[1]

Architecture n-tiers (Limites et solutions)

- Limites des middlewares "traditionnels"
  - Java RMI : Mono-langage
  - DCOM : Mono-plateforme (Windows)
  - CORBA : Multi-langage, Multiplateforme, complexe à mettre en œuvre.
- Solution:

  Adaptation des architectures réparties au contexte de l'Internet où le Web est considéré comme un nouveau middleware:
  - Multi-langage
  - Multiplateforme
  - Simple à mettre en œuvre

➔Mettre en œuvre des architectures N-tiers à base d'objets reposant sur les standards technologiques(HTTP, xml) ➔ Architectures Orientée Services

Définition d'un web service

- *Un service Web est une application logicielle identifiée par un URI dont les interfaces et les liaisons sont définies, décrites et découvertes en XML et supporte une interaction directe avec les autres applications logicielles en utilisant des messages XML via un protocole Internet (W3C definition).*

Analyse de la définition (1/4)

- **Application logicielle : programme**
- **URI : Uniform Resource Identifier**
  - 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)
- **Liaison (*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

Analyse de la définition (2/4)

- **XML : eXtensible Markup Langage**
  - pré-requis indispensable pour faire des services web
- **protocoles internet :**
  - bas niveau TCP/IP
  - haut niveau (applicatif) :
    - HTTP (web)
    - SMTP (mail)
    - FTP (file)
    - etc.

Analyse de la définition (3/4)

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.

Analyse de la définition (4/4)

- Une vision plus simple : un service web est
  - un programme accessible par internet
  - par l'intermédiaire de messages XML
  - transmis par HTTP

Exemples

- Service de traduction (Google)
- Conversions de monnaies
- Catalogues de prix
- Réservation de billets

http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL
http://www.gcomputer.net/webservices/dilbert.asmx
https://www.dataaccess.com/webservicesserver/NumberConversion.wso
https://www.dataaccess.com/webservicesserver/TextCasing.wso
http://www.dneonline.com/calculator.asmx?wsdl

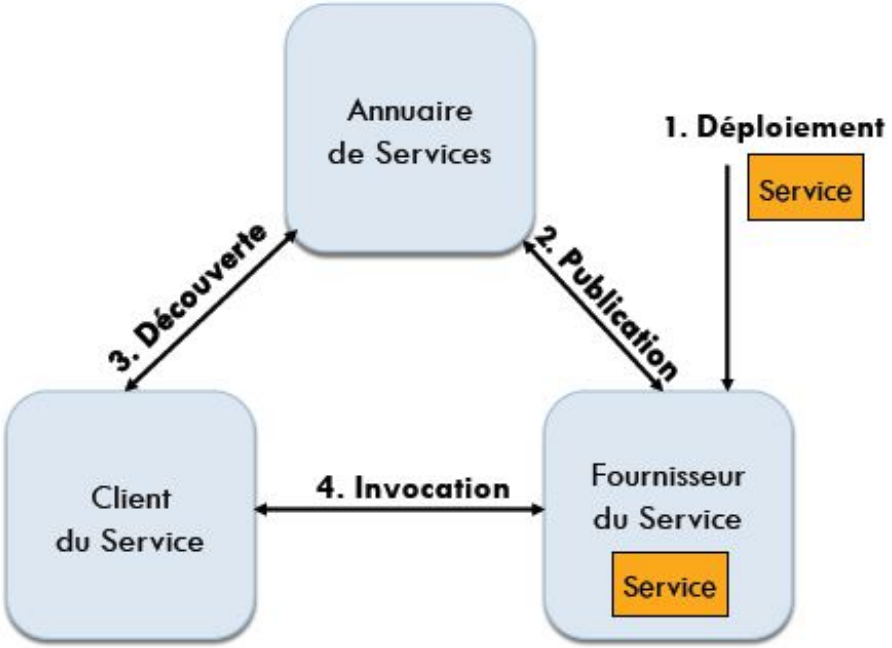
Caractéristiques des services web

- Modularité
  - Réutilisation et composition de services,
- Interopérabilité
  - Dialogue entre environnements et plate-formes hétérogènes,
  - Couplage faible (communications synchrones/asynchrones),
- Intégration
  - Intégration du système d'information au sein et en dehors de l'entreprise,
  - Masquage de la complexité,
- Indépendance de :
  - la plate-forme (UNIX, Windows,…)
  - leur implémentation (Java, C++, Visual Basic,…)
  - l'architecture sous-jacente (.NET, J2EE,…)

Les acteurs de services web

- Les principaux acteurs dans une architecture des services web : le client et le fournisseur.
  - Un fournisseur est représenté par un serveur d'application (par exemple un serveur J2EE).
  - Un fournisseur peut être le client d'un autre fournisseur (interopérabilité).
  - Une fois le service est publié, le client peut y accéder.

Infrastructure d'une AOS



[2]

Une infrastructure AOS se compose de

- Un fournisseur de web service
- Un annuaire où un fournisseur publie ses web services
- Un client qui consulte l'annuaire de services pour chercher et invoquer un web service
- Une AOS est déployée sur un réseau internet ou intranet

Standards de l'architecture SOAP

Les standards sont un élément clé d'une AOS, ils assurent l'interopérabilité entre les différents acteurs d'une AOS.

SOAP
W3C
Simple Object Access Protocol

Transporte

WSDL
W3C
Web Services Description Language

Décrit le contrat

UDDI
Microsoft, IBM, HP
Universal Description Discovery and Integration

Stocke les descriptions de contrats

Les trois piliers des Services Web [5]

Standards des Services Web: SOAP

- SOAP (Simple Object Access Protocol)
  - Assure les appels de procédure à distance au dessus d'un protocole de transport.
  - Fonctionnement côté client :
    - Ouverture d'une connexion HTTP
    - Requête SOAP est un document XML décrivant : les méthodes à invoquer sur une machine distante, et ses paramètres.
  - Fonctionnement côté serveur SOAP :
    - Récupère la requête
    - Exécute la méthode avec les paramètres.
    - Renvoie une réponse SOAP (document XML) au client.

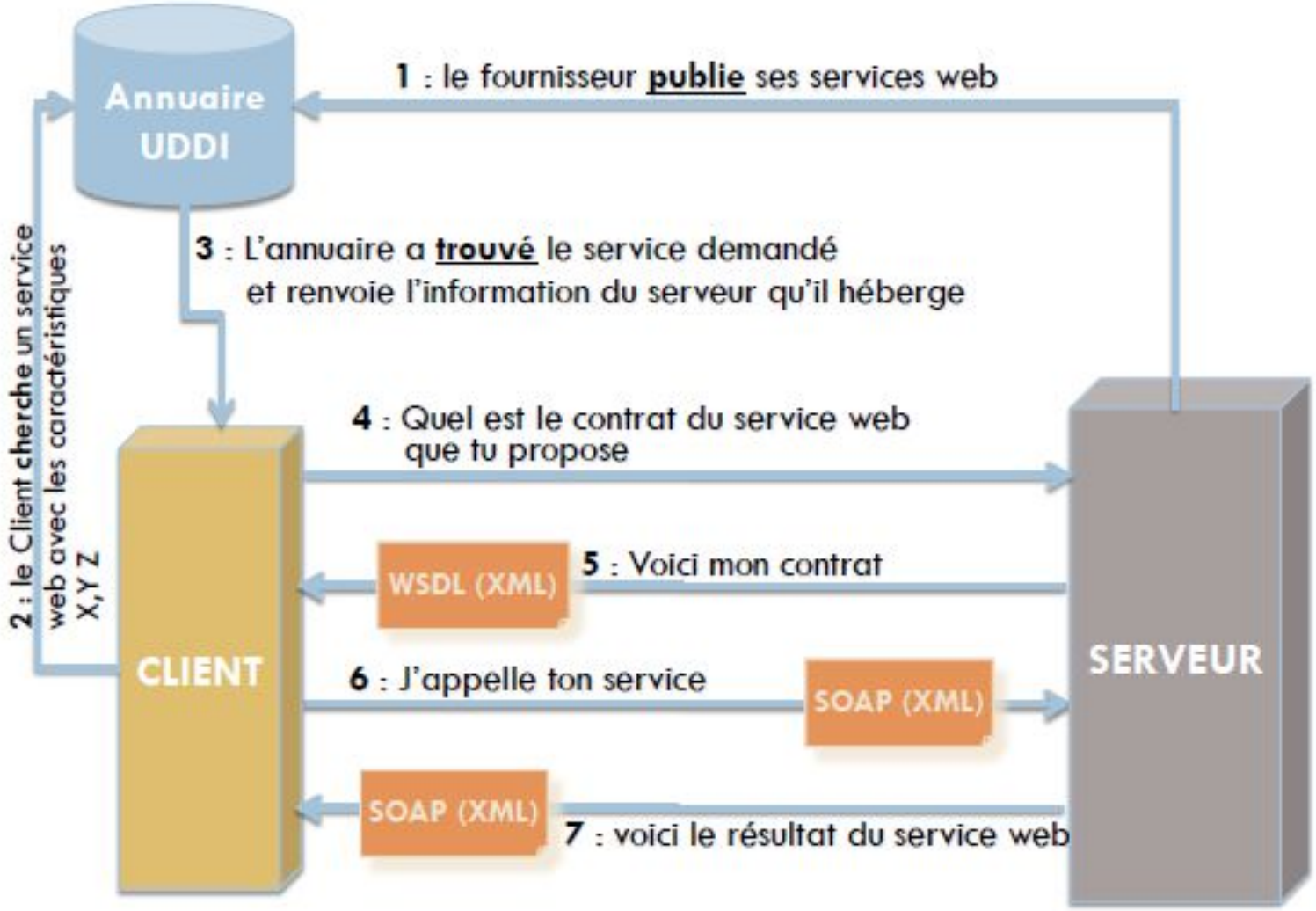
Standards des Services Web: WSDL

- WSDL (Web Services Description Language)
  - Une interface qui décrit les fonctionnalités offertes par le service sous forme d'un contrat
  - Le fichier WSDL est écrit en format XML. Il regroupe toutes les informations nécessaires pour interagir avec le service:
    - Les méthodes
    - Les paramètres et les valeurs retournées.
    - Les protocoles de transport
    - La localisation du service

Standards des Services Web: UDDI

- UDDI (Universal Description, Discovery, and Integration)
  - Spécification pour la définition d'un service de registre.
  - Fournisseur :
    - Déclaration du fournisseur
    - Enregistrement de ses services web disponibles
  - Client:
    - Requête de recherche de service web
    - Mise en relation avec le web service d'un fournisseur.

Fonctionnement



Le scénario complet d'invocation WS (1)

- Etape 1: Définition, description du service
  - La définition est faite en WSDL au sein du fournisseur de service.
- Etape 2: Publication du service
  - Le service peut être déclaré dans un annuaire. On parle donc de publication du service afin de le rendre accessible aux clients.
  - La publication est effectuée au sein d'un annuaire dédié UDDI.

Le scénario complet d'invocation WS (2)

- Etape 3: Recherche du service
  - Le client se connecte à un annuaire (UDDI) pour effectuer une recherche de service.

- Etape 4: Enregistrement du service web
  - Une fois le service est trouvé par le client, ce dernier doit s'enregistrer auprès du fournisseur associé au service.
- Etape 5: Mise en œuvre du service web
  - Le client peut invoquer le service web suivant les conditions inscrites au sein de l'annuaire lors de sa publication.

Exemple d'un code XML


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BIBLIO SUBJECT="XML">
    <BOOK ISBN="9782212090819" LANG="fr" SUBJECT="applications">
        <AUTHOR>
            <FIRSTNAME>Jean-Christophe</FIRSTNAME>
            <LASTNAME>Bernadac</LASTNAME>
        </AUTHOR>
        <AUTHOR>
            <FIRSTNAME>François</FIRSTNAME>
            <LASTNAME>Knab</LASTNAME>
        </AUTHOR>
        <TITLE>Construire une application XML</TITLE>
        <PUBLISHER>
            <NAME>Eyrolles</NAME>
            <PLACE>Paris</PLACE>
        </PUBLISHER>
        <DATEPUB>1999</DATEPUB>
    </BOOK>
```

Exemple de requête/réponse SOAP

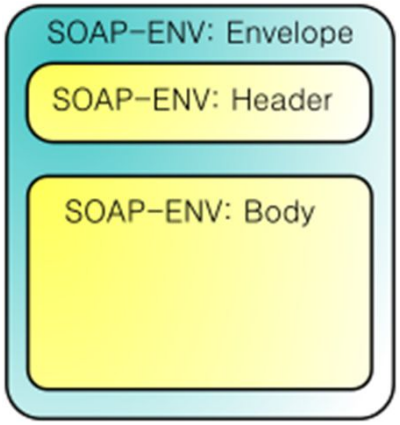
add Method invocation

Method parameter(s)

Type	Value
int	3
int	5

Method returned

int : "8"



SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:add xmlns:ns2="http://webservice/">
            <i>3</i>
            <j>5</j>
        </ns2:add>
    </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:addResponse xmlns:ns2="http://webservice/">
            <return>8</return>
        </ns2:addResponse>
    </S:Body>
</S:Envelope>
```

Exemple de WSDL


```
- <definitions targetNamespace="http://webservice/" name="CalculatriceWS">
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://webservice/" schemaLocation="http://localhost:8080/Calculatrice/CalculatriceWS?xsd=1"/>
      </xsd:schema>
    </types>
  - <message name="add">
      <part name="parameters" element="tns:add"/>
    </message>
  - <message name="addResponse">
      <part name="parameters" element="tns:addResponse"/>
    </message>
  - <portType name="CalculatriceWS">
    - <operation name="add">
        <input wsam:Action="http://webservice/CalculatriceWS/addRequest" message="tns:add"/>
        <output wsam:Action="http://webservice/CalculatriceWS/addResponse" message="tns:addResponse"/>
      </operation>
    </portType>
  - <binding name="CalculatriceWSPortBinding" type="tns:CalculatriceWS">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    - <operation name="add">
        <soap:operation soapAction=""/>
      - <input>
          <soap:body use="literal"/>
        </input>
      - <output>
          <soap:body use="literal"/>
```

Architecture REST

- REST est un style architectural de web service
- Toutes les ressources d'un web service REST sont identifiées et accessibles via un URI.
- REST n'est pas un protocole comme SOAP.
- Utilisation du protocole HTTP
- Plusieurs formats de données XML, HTML et JSON
- Simple à implémenter et à invoquer