



Introduction à l'architecture SOA

Module SOA
A.U 2023-2024



Objectifs



- Définir la position de SOA par rapport aux autres architectures des systèmes de Web
- Décrire l'architecture orientée services basée sur le middleware
- Familiariser l'apprenant avec le style d'architecture SOA

Plan

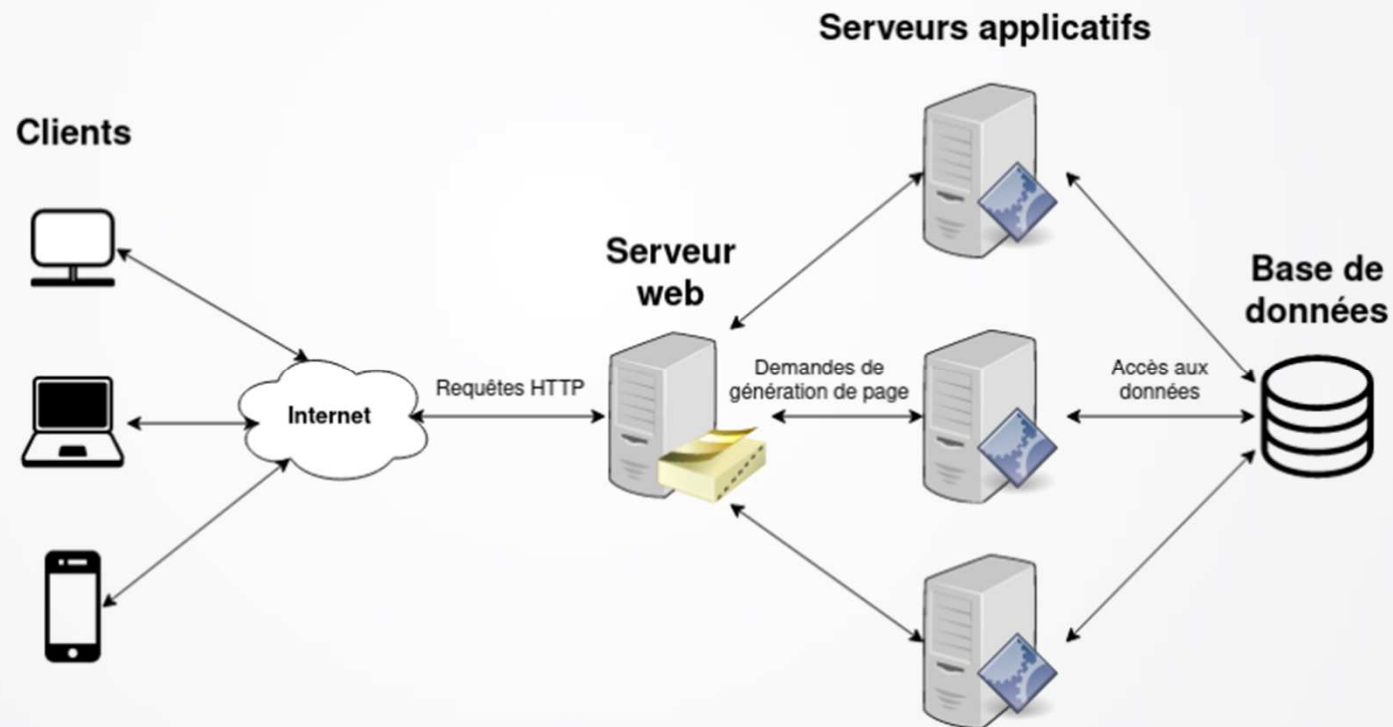
- Présenter les différents historiques des architectures des systèmes de Web
- Expliquer le principe de base des systèmes distribués
- Présenter le Middleware en relation avec l'architecture SOA
- Définir l'Architecture Orientée Services "SOA"



Historique des architectures des systèmes basés sur le Web

Systemes basés sur le Web

❑ Fonctionnement





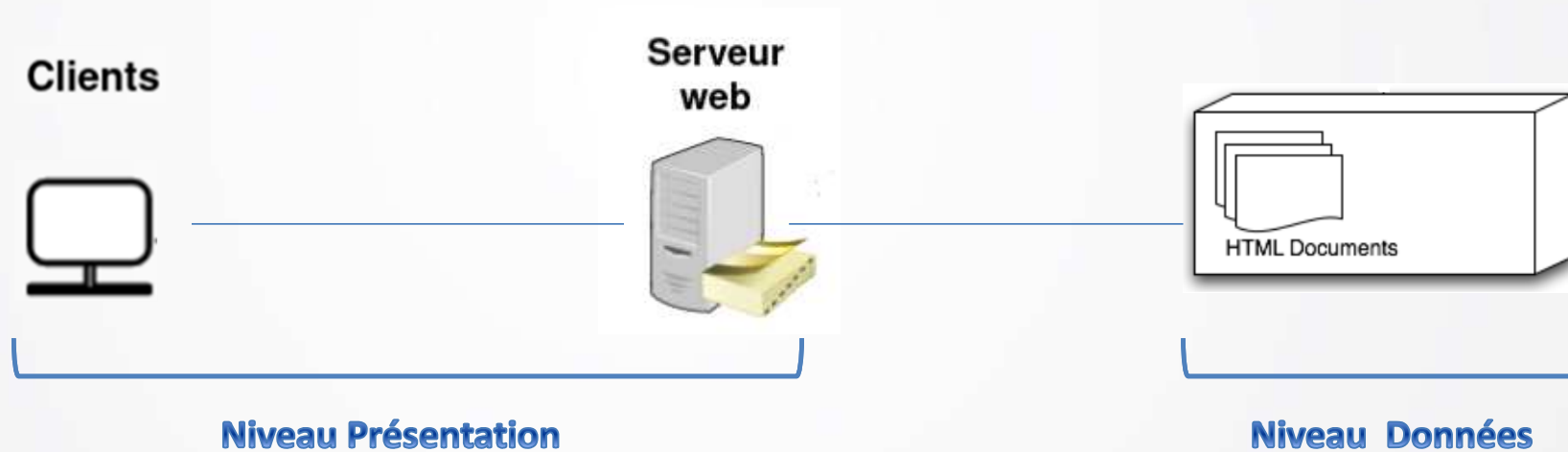
Systemes basés sur le Web



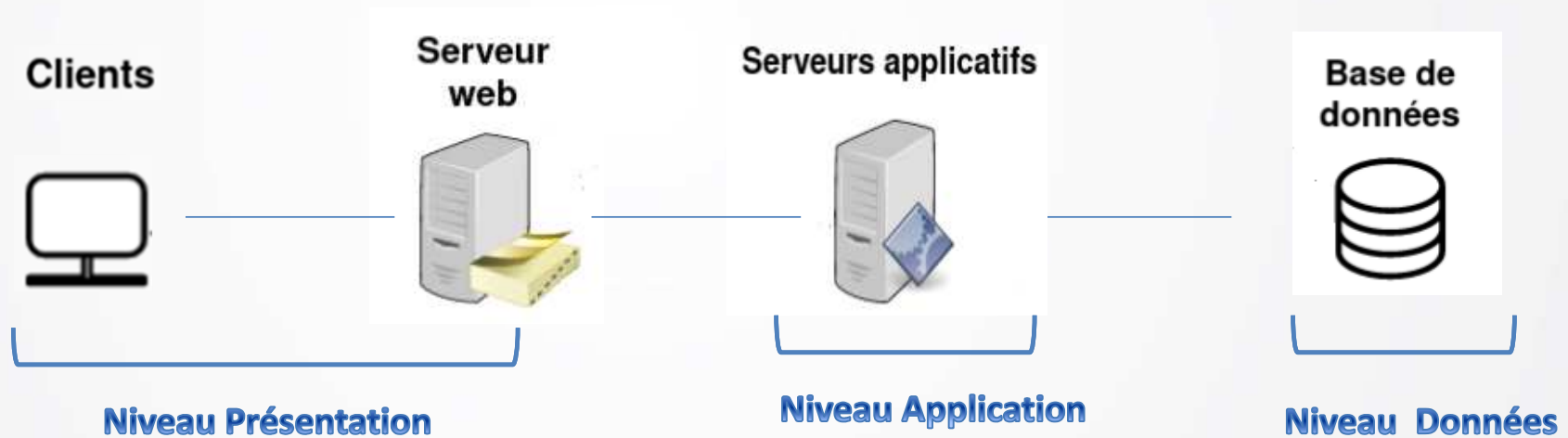
□ Couches

- **Client (navigateur):** affiche les informations à l'utilisateur
- **Serveur Web:** gère les requêtes HTTP entrantes en traitant les données statiques et retourne le résultat finale au Client
- **Application:** exécute des applications web dynamiques en traitant la logique métier et en communiquant avec d'autres services
- **Données:** stocke les données utilisées par le serveur applicatif

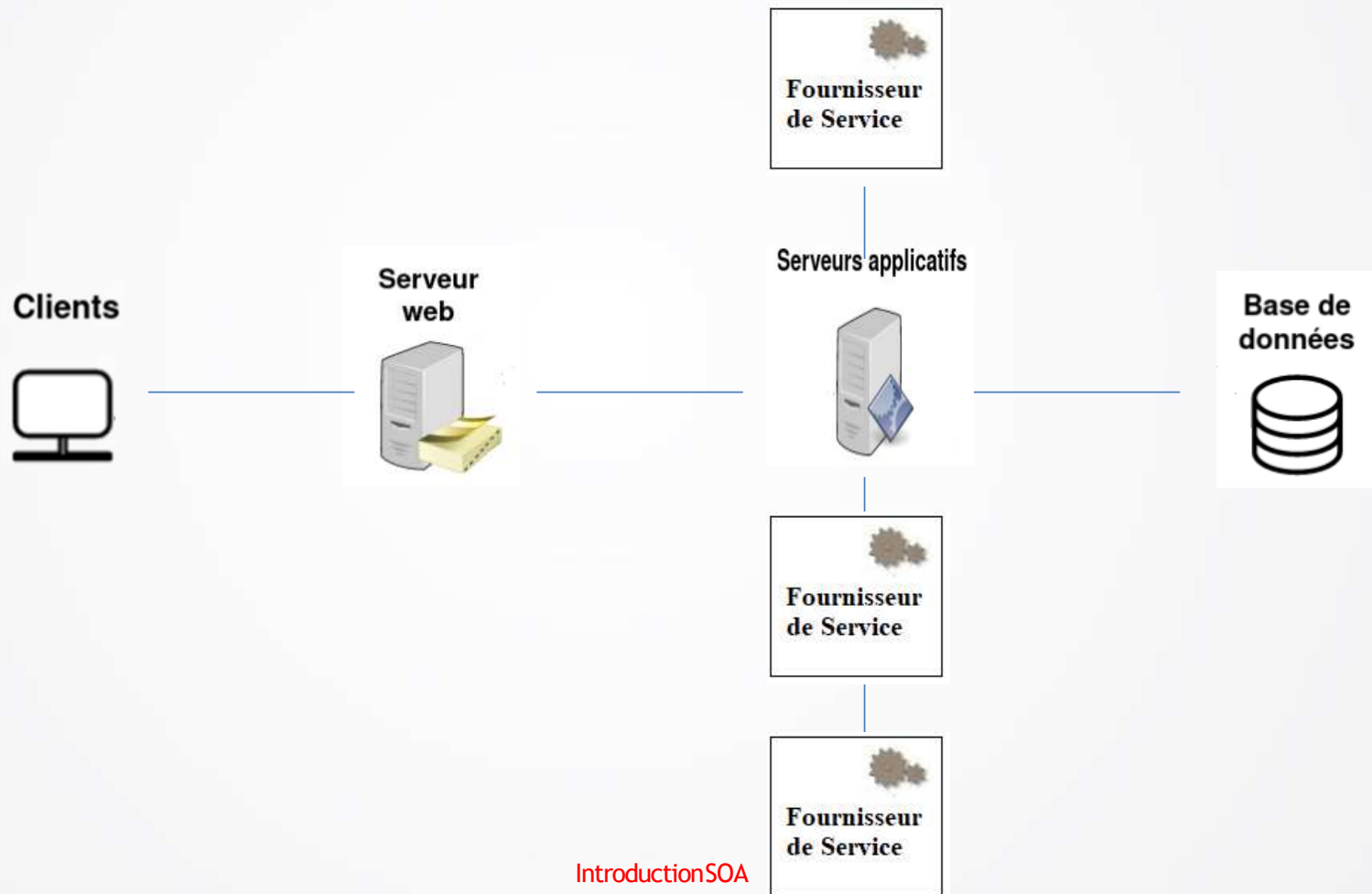
Couches pour le contenu Web statique



Couches pour le contenu Web dynamique



Couches de vue de Services





Middleware RPC



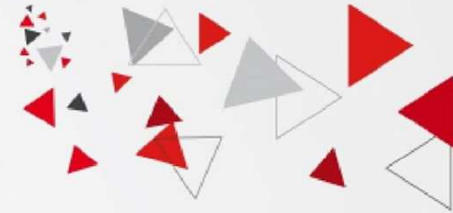
Systèmes distribués



- Ensemble d'ordinateurs **indépendants** répartis sur différents **environnements**:
- ils communiquent à l'aide d'un middleware.
 - ✓ Ils travaillent ensemble pour apparaître comme un seul système cohérent à l'utilisateur final.
 - ✓ Echangent des messages entre eux afin d'atteindre des objectifs communs.



Systèmes distribués



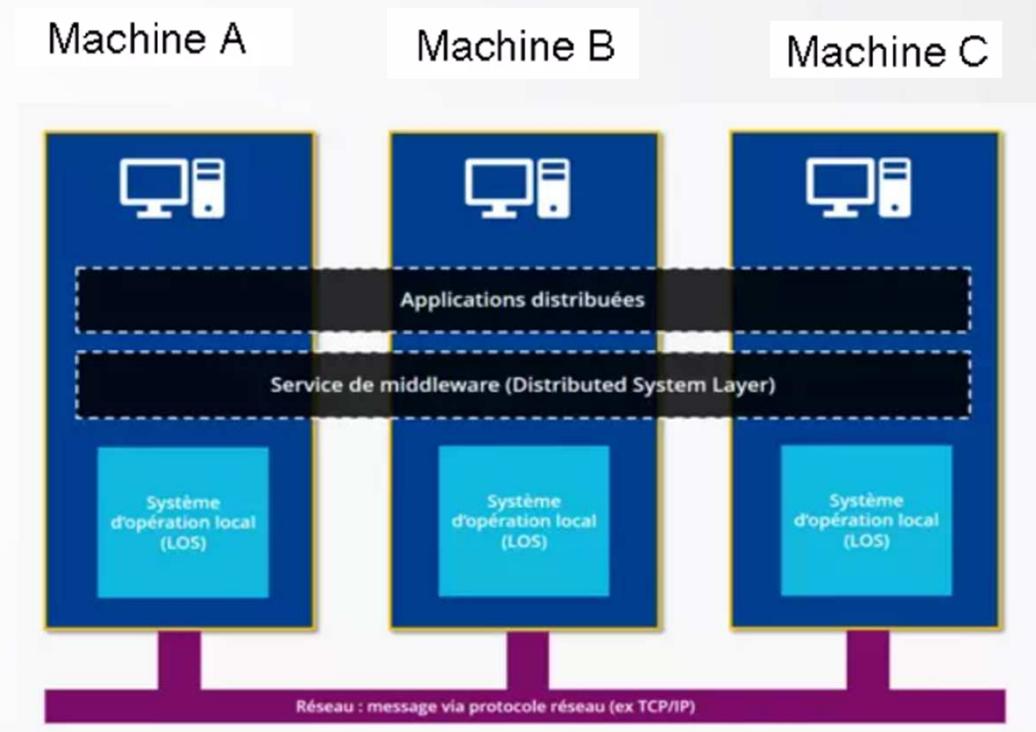
Caractéristiques:

- ☐ Evolutivité,
- ☐ La répartition des données et des services
- ☐ Disponibilité: tolérance aux interruptions de service
- ☐ Transparence: interagir avec un seul périphérique logique sans avoir à se préoccuper de l'architecture du système

Systemes distribués

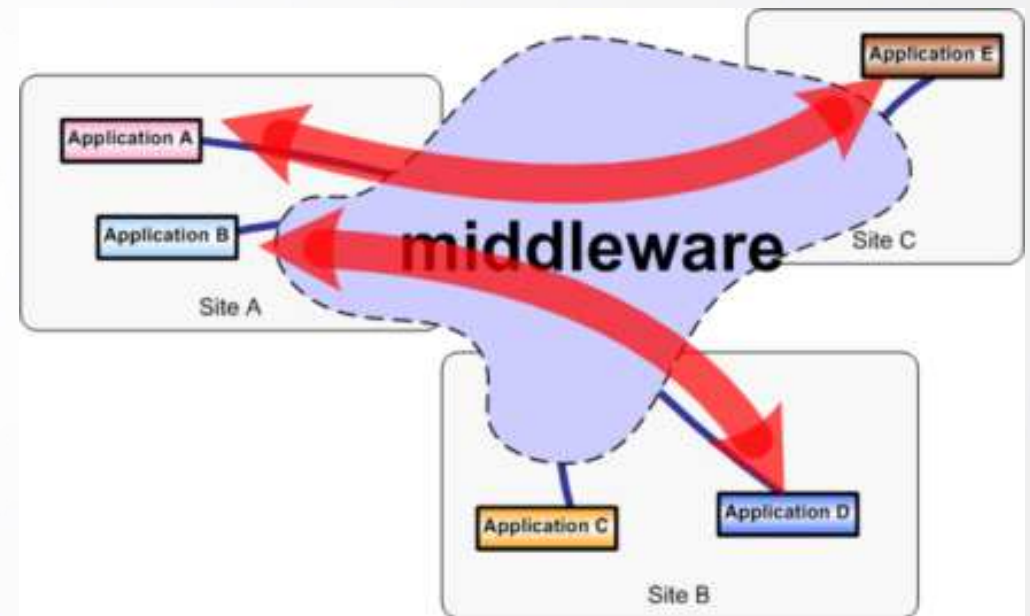
Communication:

Étant donné que les environnements de machine et d'exploitation sont différents entre les clients et les serveurs, ils communiquent à l'aide d'un **middleware**.



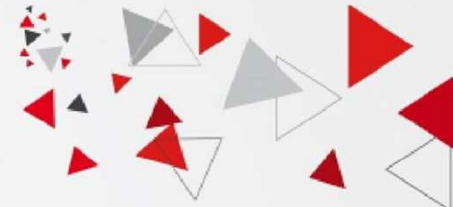
Middleware

- Type d'architecture utilisé pour faciliter la communication des services entre deux applications qui fonctionnent sur des systèmes dont l'environnement est différent.

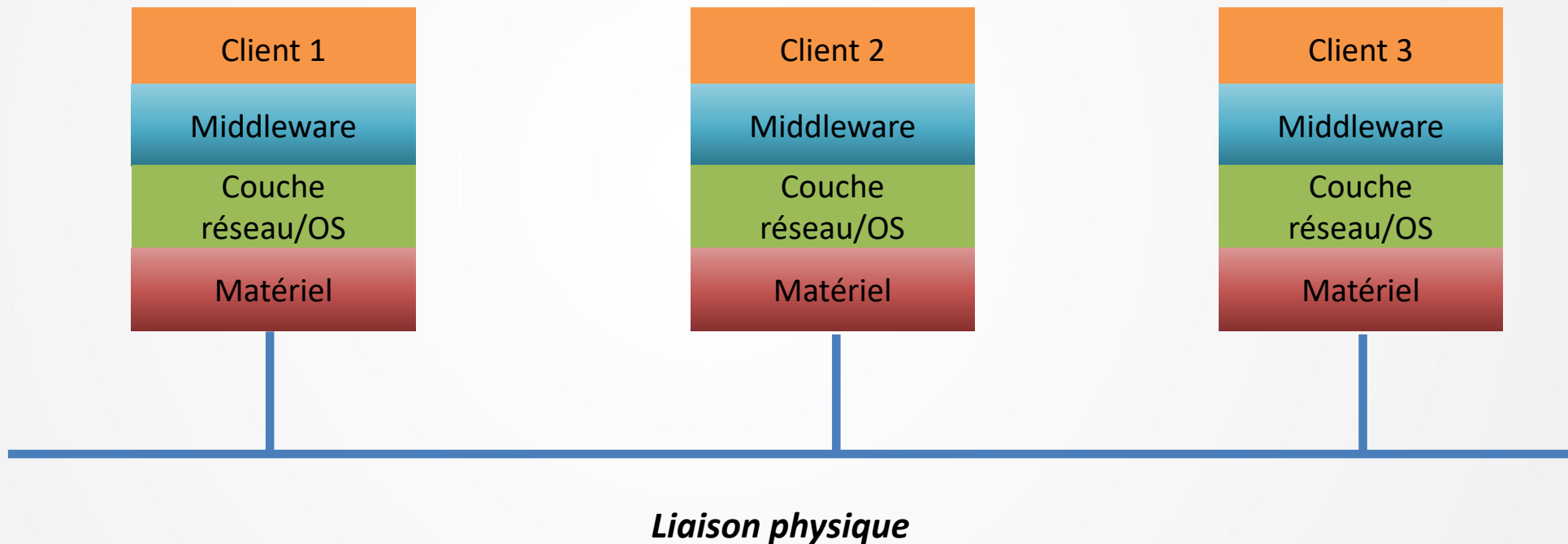




Middleware

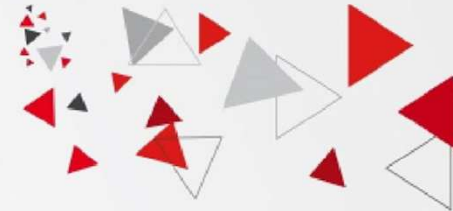


Fonctionnement:

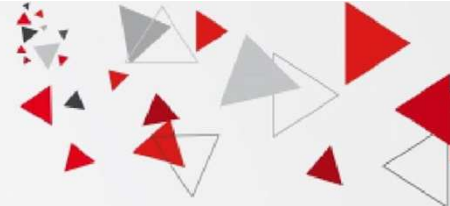




Middleware



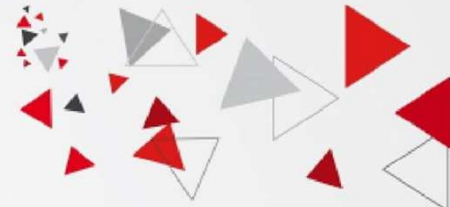
- Principaux familles de communication:
 - **RPC**
 - RMI
 - CORBA
 - etc



RPC (Remote Procedure Call)

Les RPC, ou Appels de Procédures à Distance, représentent un mécanisme qui permet à un programme de faire appel à des fonctions ou méthodes distantes.

Ils constituent la base des middleware modernes utilisant les technologies réseau.

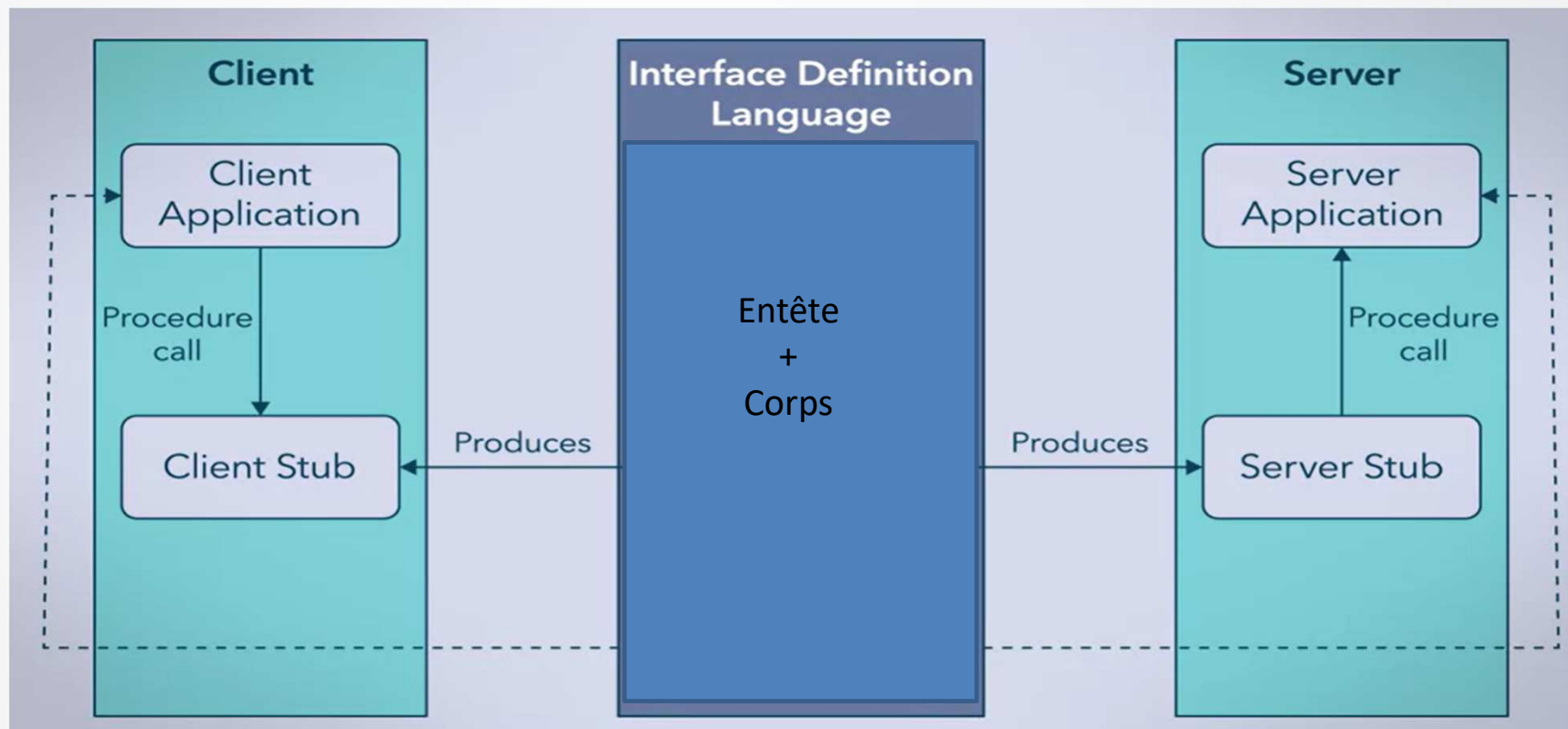


RPC (Remote Procedure Call)

Se composent de trois composants principaux :

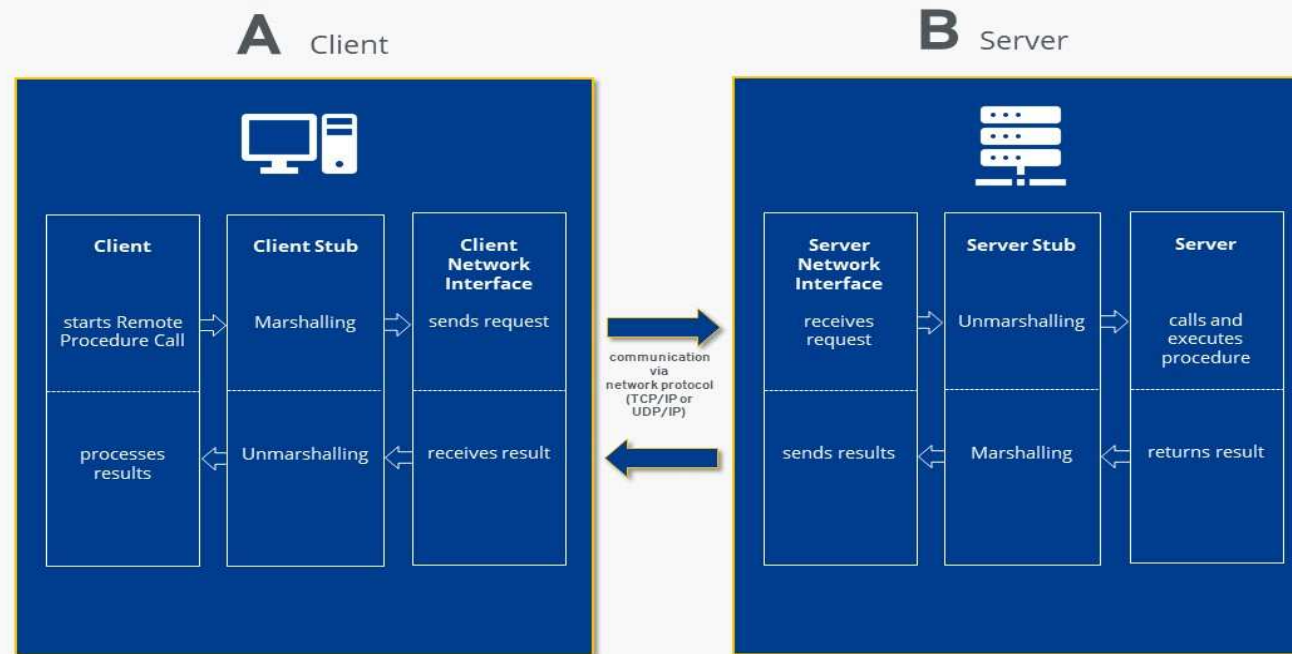
- Un client (l'appelant): le composant qui effectue l'appel à distance;
- Un serveur (l'appelé): le composant qui implémente la procédure invoquée;
- Un langage de définition d'interface (IDL): le langage par lequel le client et le serveur communiquent.

RPC (Remote Procedure Call)



RPC (Remote Procedure Call)

Remote
Procedure
Call (RPC)





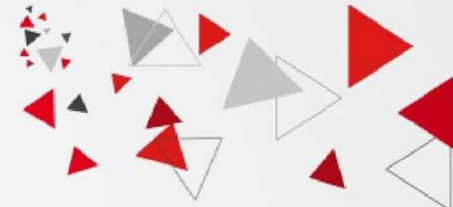
Le Stub Client

- Les stubs sont des classes placées dans la partie du client
- Lorsque un client fera l'appel à une méthode distante, cet appel sera transféré au stub
- Donc le stub joue le rôle d'un relais du côté de client.
- Il convertit les paramètres des méthodes distantes en un format de message standardisée telle que XML, et les envoie dans le flux des données.
- D'autre part, il déformate l'objet de retour des méthodes distantes



Le Stub Serveur (Skeleton)

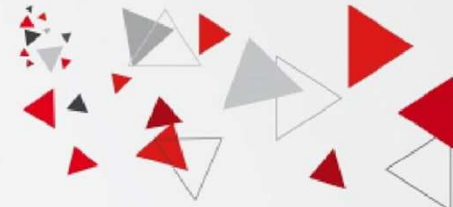
- Un relais mais coté serveur.
 - Il traduit le message reçu par le stub client, dans un format de données reconnu par le serveur;
 - Il transforme la valeur de retour dans un format standard;
 - Il renvoie la réponse du serveur au stub client
- ➔ Les Stubs sont des intermédiaires entre le client et le serveur qui gèrent le transfert distant des données.



IDL (Interface Definition Language)

IDL: le premier composant implémenté

- Utilisé pour définir les procédures du serveur qui sont disponibles pour le client.
- Décrit les paramètres d'entrée ainsi que la réponse renvoyée.
- Indique au client quels services distants sont disponibles,
- Comment ils sont accessibles et quelle sera la réponse du serveur.



IDL (Interface Definition Language)

Exemple:

file hello.idl

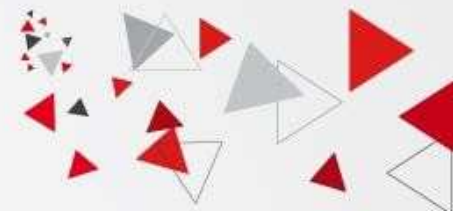
```
[
  uuid(7a98c250-6808-11cf-b73b-00aa00b677a7),
  version(1.0)
]
interface hello
{
  void HelloProc(
    [in, string] unsigned char * pszString
  );
  String HelloTo(
    [in, string] unsigned char * pszString,
    [out, string] unsigned char * pszString
  );
}
```



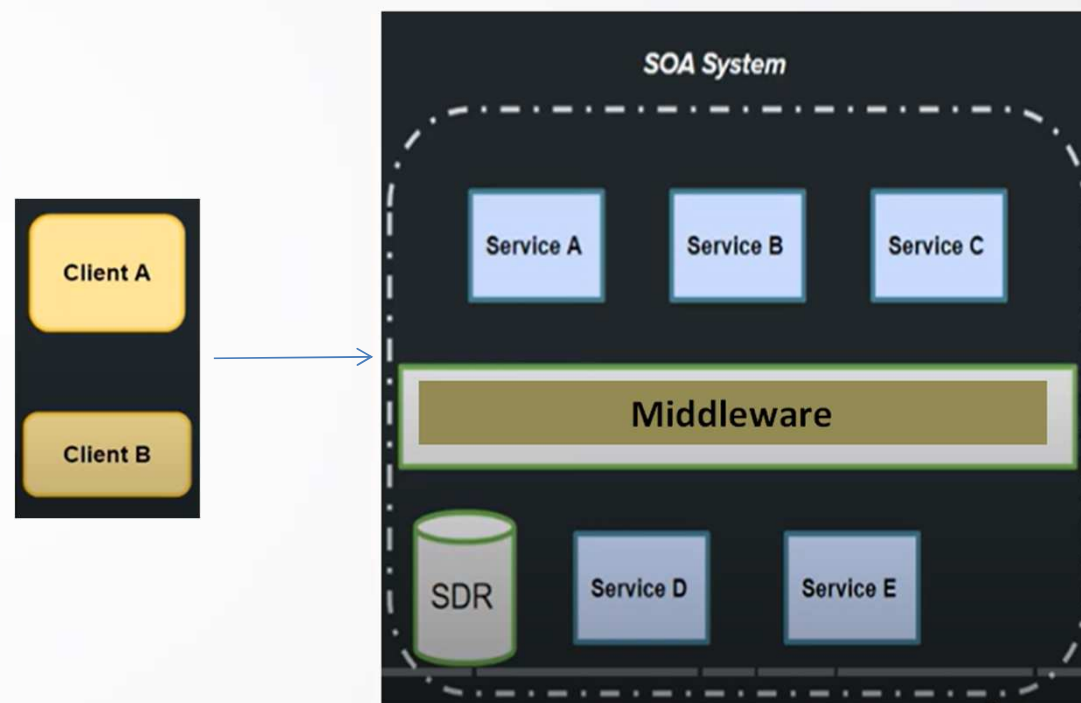

L'architecture SOA



Présentation SOA

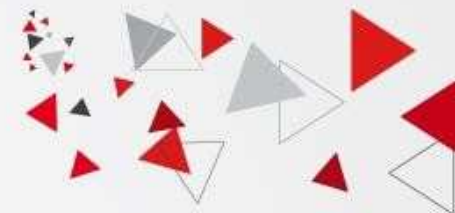


- L'architecture orientée services concerne la manière de créer, d'utiliser et de combiner des services.





Présentation SOA



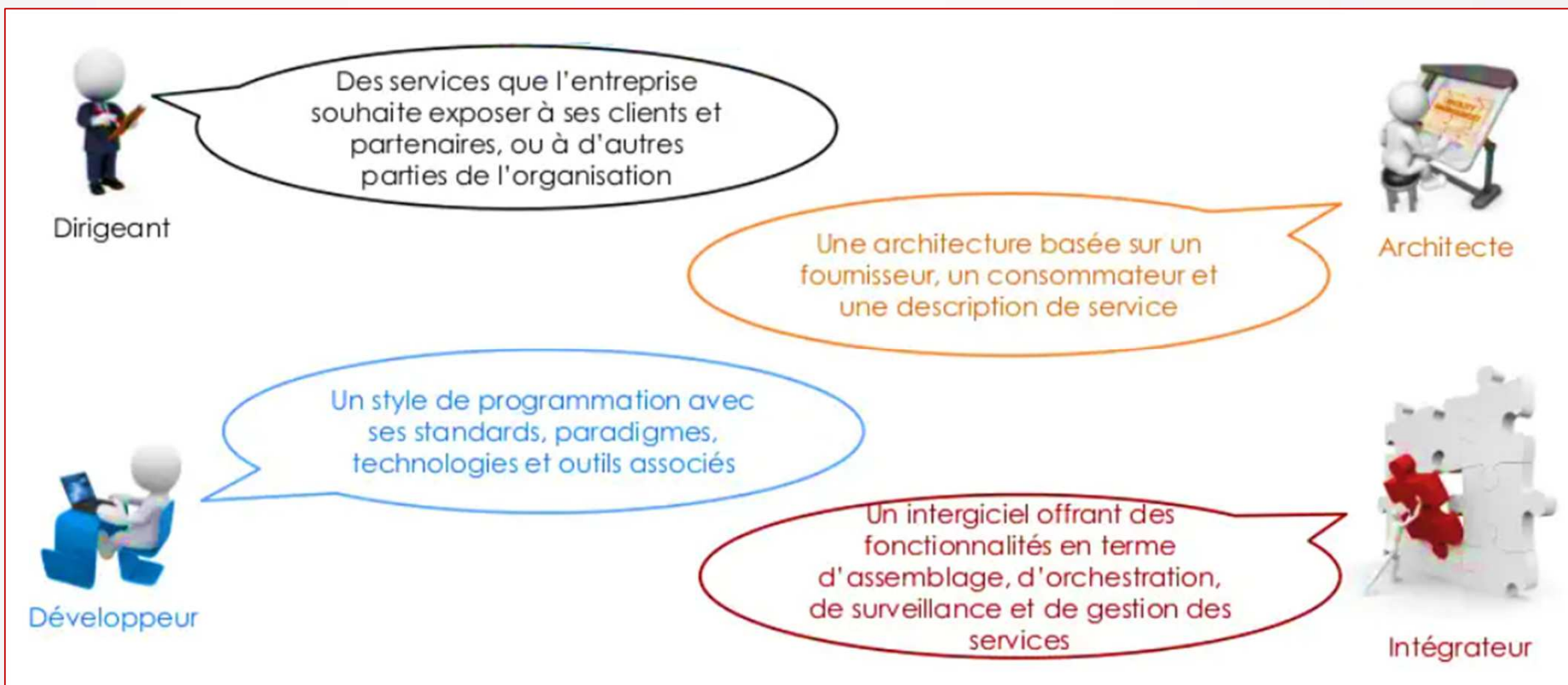
- “ L’architecture orientée service constitue un **style d’architecture** basée sur le principe de séparation de l’activité métier en une série **de services**”.
- “ Ces services peuvent **être assemblés et liés** entre eux selon le principe de couplage lâche pour exécuter l’application désirée. ”

Gartner - Septembre 2005

- Objectifs: Décomposer une fonctionnalité en un ensemble de fonctions basiques(services) fournies par des composants.
- Décrire finement le schéma d’interaction entre ces services.

Présentation SOA

Besoins...



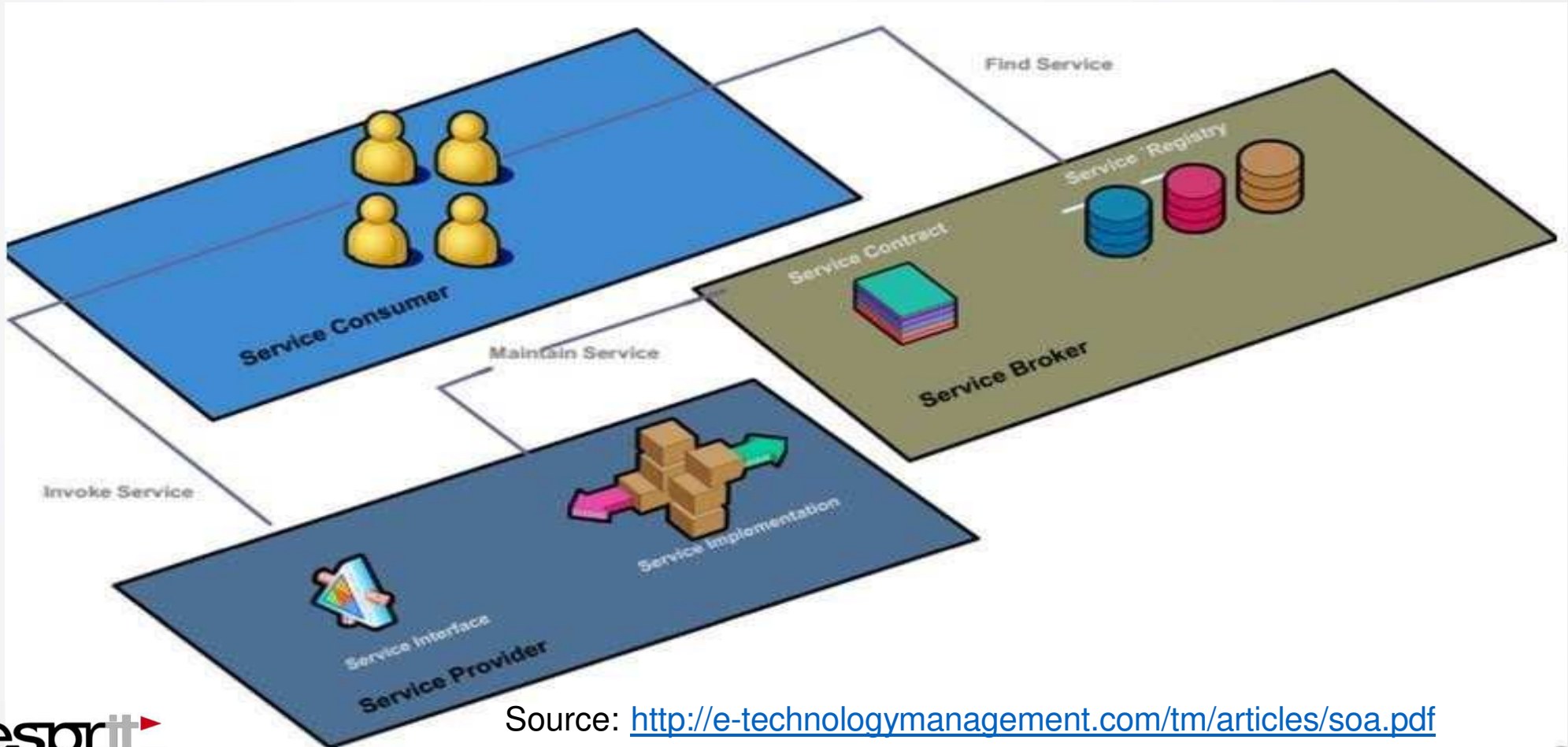


Éléments SOA



- **Le fournisseur de service** crée le service Web, puis publie son interface ainsi que les informations d'accès au service, dans un annuaire de services Web.
- **L' annuaire de service** rend disponible l'interface du service ainsi que ses informations d'accès, pour n'importe quel demandeur potentiel de service(peut être public ou privé).
- **Le consommateur de service** accède à l'annuaire de service pour effectuer une recherche afin de trouver les services désirés. Ensuite, il se lie au fournisseur pour invoquer le service.

Architecture SOA






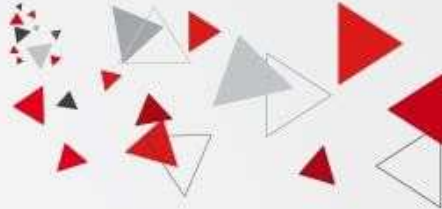
Apports de SOA



- Améliorer la rapidité ainsi que la productivité des développements.
- Une réutilisabilité possible des services.
- De meilleures possibilités d'évolution.
- Une maintenance facilitée.
- Couplage faible entre les services.
- Architecture basée sur des standards ouverts.
- L'indépendance par rapport aux aspects technologiques.
- Une modularité permettant de remplacer facilement un service par un autre.



Architecture SOA ou microservices ?



- L'architecture SOA est indépendante de la technologie
 - Permet une interopérabilité transparente entre les différents services.
 - Un service web est une application **monolithique** :
 - développée en un seul bloc;
 - déployée d'une manière unitaire dans un serveur d'application.
- L'architecture micro-services est un modèle d'architecture lié à une application,
 - Se compose d'un ensemble de petits services **indépendants** (langages, architecture, etc).
 - Chaque micro-service:
 - est gérée par une équipe;
 - est séparément déployé;
 - s'exécute dans un processus qui lui est propre.



Références



- 1 <http://fr.wikipedia.org/wiki/Paradigme>
- 2 <http://design-patterns.fr/introduction-a-la-programmation-orientee-objet>
- 3 <http://fr.wikipedia.org/wiki/Middleware>
- 4 <http://blog.xebia.fr/2009/04/29/soa-du-composant-au-service-lautonomie>