Chapitre 6 : Les ERP et l'EAI

1 Définition de l'EAI

EAI (Enterprise Application Integration ou intégration d'applications d'entreprise : IAE en français) **est une plateforme qui permet de réunir les différents programmes d'une organisation au sein d'un même espace, afin d'assurer la communication entre ces programmes.**

L'intégration d'applications d'entreprise (EAI) est le processus qui consiste à combler le fossé de communication entre les différentes applications au sein d'une entreprise, telles que les applications d'inventaire, de planification des ressources et de gestion de la relation client.

L'objectif de l'IAE est de fournir une forme normalisée d'accès à toutes les applications de l'entreprise, d'offrir une logique d'entreprise unifiée, de faciliter le flux de données entre les programmes sans modifier considérablement la configuration de la base de données, et de s'assurer que toute modification des données commerciales en un seul point est immédiatement reflétée dans toutes les bases de données correspondantes.

2 Types de communication d'applications en EAI

Il existe plusieurs architectures EAI communes parmi lesquelles une entreprise peut choisir. Vous pouvez soit sélectionner et appliquer une architecture unifiée dans toute l'entreprise, soit combiner plusieurs modèles EAI dans un seul projet.

2.1 Intégration point à point (P2P)

L'intégration P2P (Figure 1) repose sur un script pour extraire des données d'une application et les transmettre à une autre application. Il nécessite donc un programme de connecteur unique pour intégrer chaque paire d'applications.

Cette approche est efficace lorsque l'on n'intègre que quelques applications mais devient inefficace dans la gestion de systèmes plus grands à mesure que le nombre de scripts personnalisés augmente. Cette architecture n'est pas évolutive.

2.2 Intégration par pôle et rayon

Ce modèle utilise un concentrateur central, qui relie toutes les applications intégrées. Chaque solution envoie ses données au centre qui reformate et les transmet à l'application de destination (Figure 2).

Comme il n'est pas nécessaire d'établir une connexion directe entre chaque paire d'applications, l'architecture en toile est plus facile à mettre à l'échelle. Cependant, toutes les communications passent par le concentrateur, qui fonctionne sous une charge lourde et représente un point de défaillance unique. En outre, ce modèle nécessite encore un

entretien et une intervention humaine pour ajouter de nouvelles applications et soutenir les applications existantes.

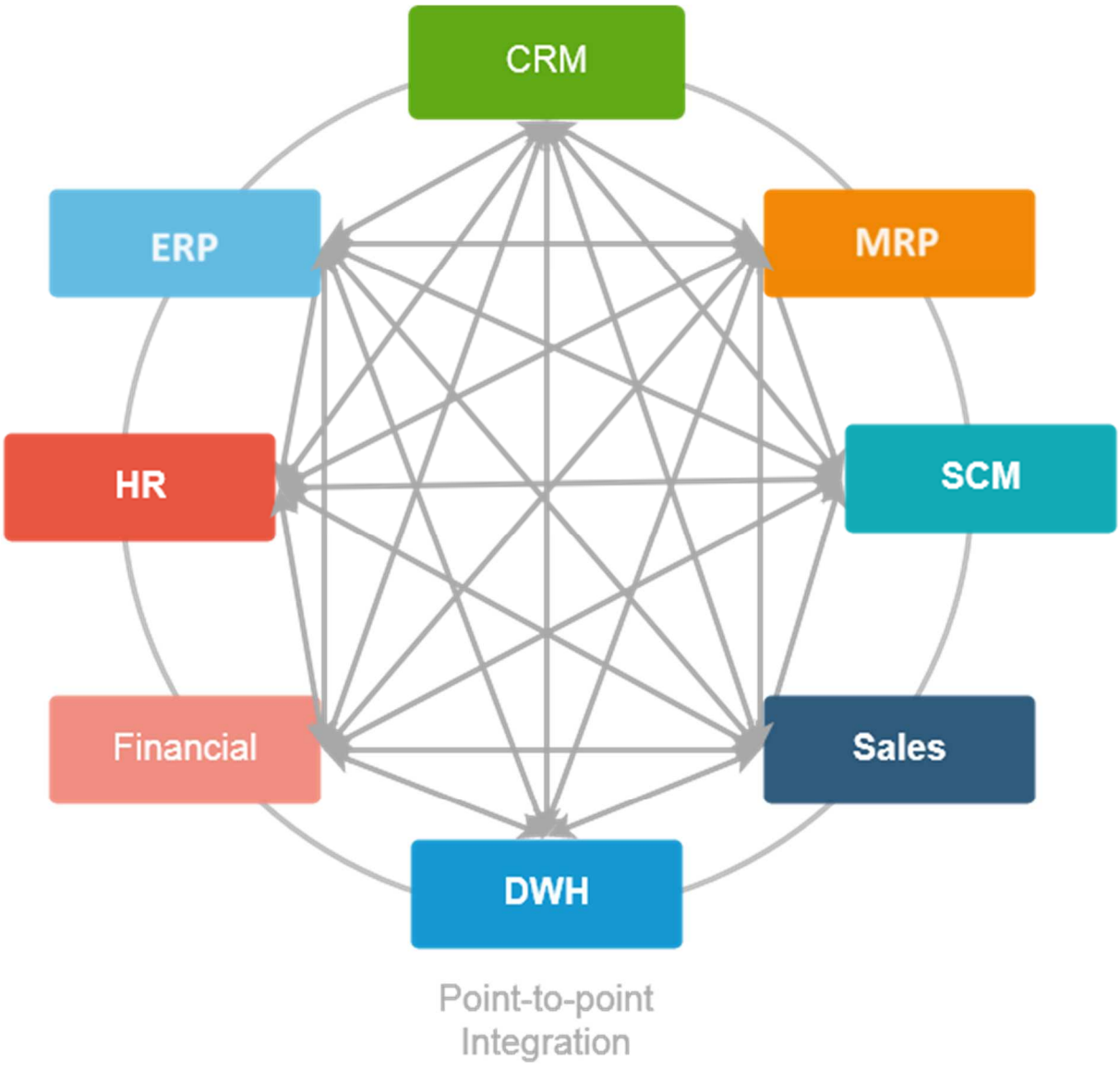

Figure 1: Intégration point à point (P2P)

2.3 Intégration en bus

Ce modèle établit une communication entre des systèmes utilisant une architecture orientée service (SOA). L'intégration des bus améliore le modèle précédent, car elle ne nécessite pas une intervention humaine constante pour acheminer les données reformatées, car tout cela est régi par des règles et des politiques prédéfinies.

Ce modèle est plutôt léger et adapté à l'intégration d'un grand nombre de systèmes. Et il est facile à mettre à l'échelle car les applications peuvent être ajoutées/supprimées avec un minimum de modifications du logiciel EAI. D'autre part, le modèle basé sur un bus est difficile à configurer et à maintenir, et la vitesse de communication est plus faible que dans les modèles précédents.

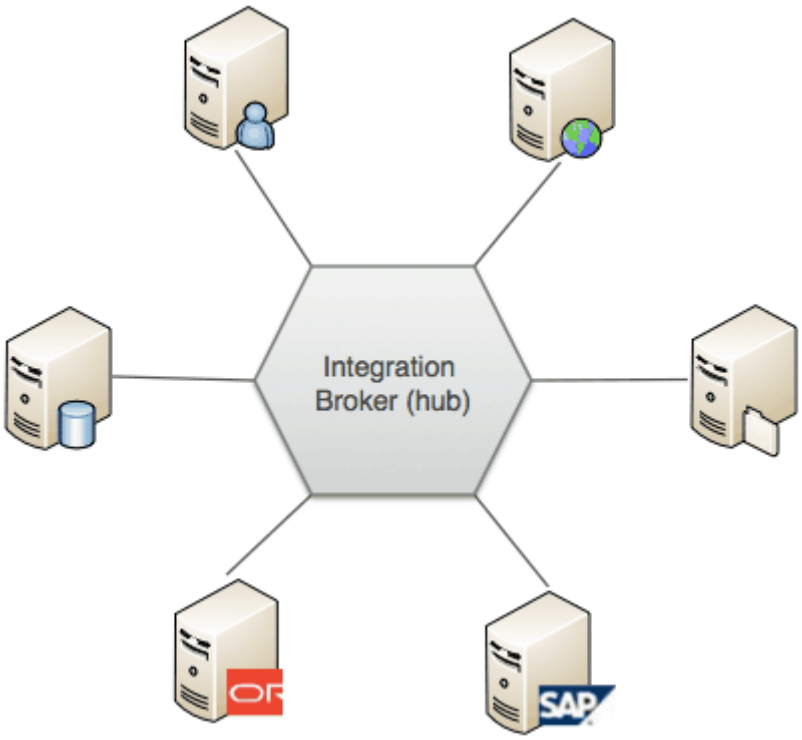

Figure 2: Intégration par le pôle et par rayon

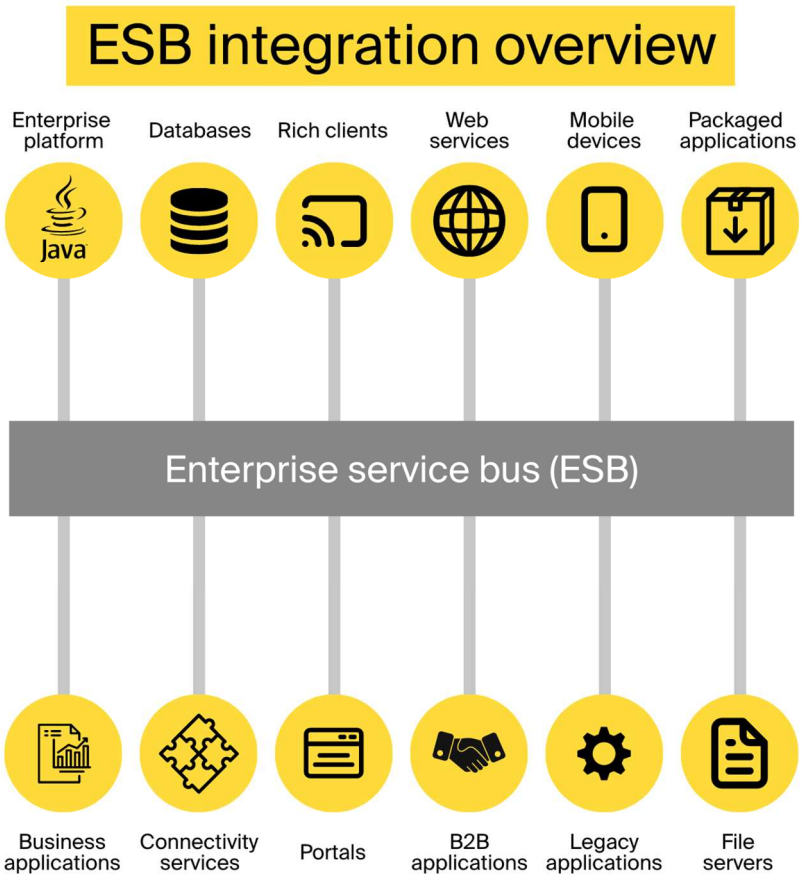

Figure 3: Intégration en bus

2.4 Intégration basée sur les middlewares

Middleware est un logiciel qui fonctionne entre l'interface utilisateur d'une application et un système d'exploitation d'un ordinateur. Dans le contexte de l'intégration d'applications d'entreprise, **le middleware c'est un intermédiaire qui facilite la traduction et l'échange de données entre les applications distribuées**. Les entreprises peuvent déployer différents types d'intergiciels, tels que serveur d'application ou intergiciel de base de données.

Comme dans les modèles précédents, l'intergiciel simplifie l'intégration en évitant une intervention humaine constante. Il est facile d'échelle et d'apporter des modifications aux applications impliquées. Cependant, si vous déployez un middleware prêt à l'emploi, cela pourrait limiter votre pile technologique aux options prises en charge par ce fournisseur. En outre, les intergiciels peuvent être assez coûteux, et il présente un point de défaillance unique.

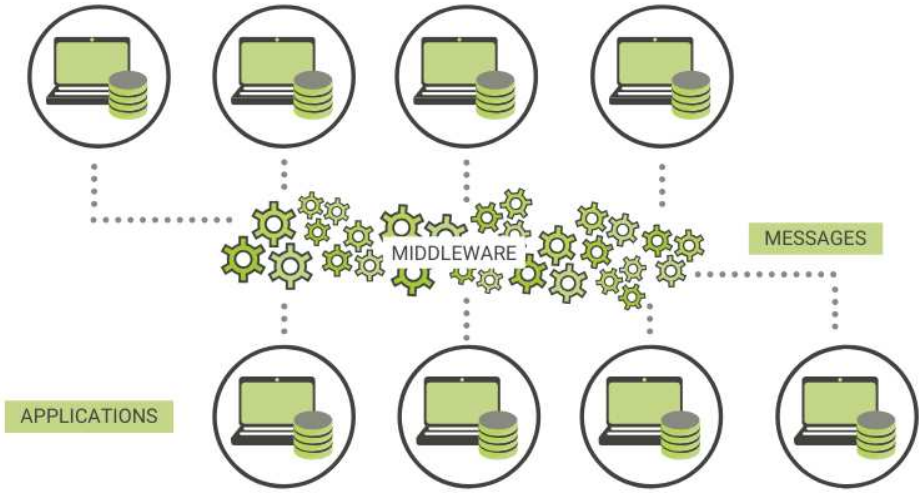

Figure 4: Intégration basée sur les middlewares

2.5 Intégration basée sur les micro-services

Les micro-services sont des petites applications légères qui servent un objectif spécifique et fournissent des services à d'autres applications (Figure 6). Il s'agit de la norme actuelle pour l'intégration d'applications d'entreprise basées sur le cloud.

Le déploiement de micro-services rend la solution EAI tolérante aux défaillances, car une défaillance d'un service ne peut pas faire tomber l'ensemble du système. Il est également plus facile de dépanner chaque service de manière indépendante que d'essayer de localiser un problème dans une solution de grand monolithe (Figure 5). Enfin, ce modèle nous permet de diversifier la pile technologique utilisée pour la mise en œuvre des micro-services. Cependant, si les micro-services sont mal organisés et optimisés, ils peuvent entraîner des frais généraux de communication et des retards de performance. De plus, un tel système est plus difficile à sécuriser, chaque micro-service ayant ses propres mécanismes d'authentification.

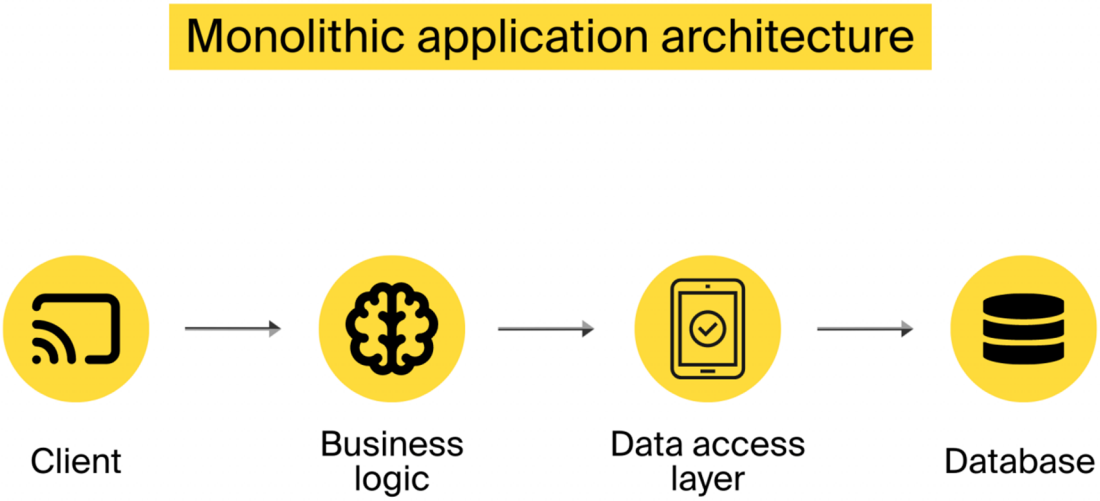

Figure 5: architecture d'une application monolithique (homogène)

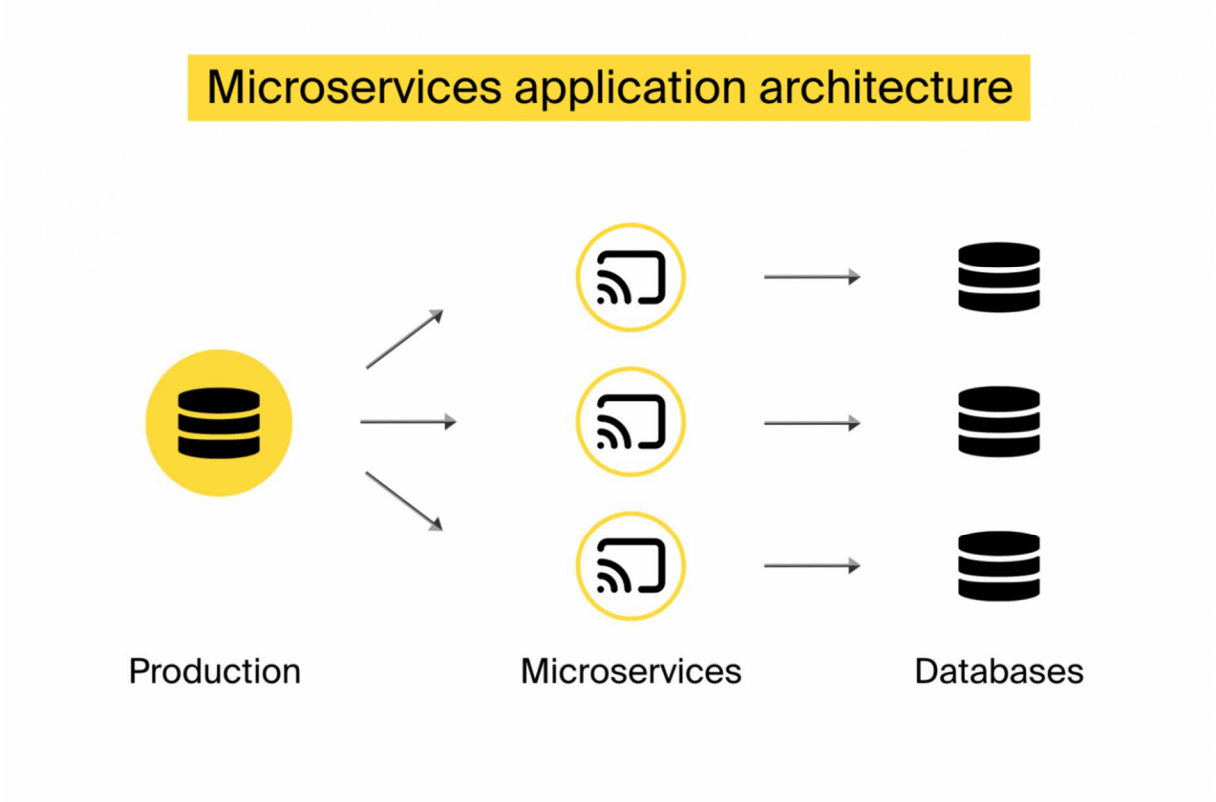

Figure 6: architecture d'une application micro-services

3 Les technologies d'intégration de systèmes

3.1 Les middlewares RMI et CORBA

3.1.1 RMI

Remote Method Invocation (RMI) **est une interface de programmation d'application (API) dans l'environnement de programmation Java**. Il permet aux objets sur un ordinateur ou à la machine virtuelle Java (JVM) d'interagir avec des objets fonctionnant sur une JVM différente dans un réseau distribué. Une autre façon de le dire est que RMI fournit un moyen de créer des applications Java distribuées via des appels de méthode simples.

RMI est la version Java de ce que l'on appelle un appel de procédure à distance (RPC), mais avec la possibilité supplémentaire de passer un ou plusieurs objets avec la demande**. Il permet une communication à distance entre les applications en utilisant deux objets -- souche et squelette – (figure 7)** et est fourni dans le cadre du Java Development Kit (JDK) de Sun Microsystems dans le paquet java.rmi.

Par exemple, lorsqu'un utilisateur sur un ordinateur distant, A, remplit un compte de dépenses, le programme Java interagissant avec l'utilisateur peut communiquer, en utilisant RMI, avec un programme Java dans un autre ordinateur, B. L'ordinateur B a toujours la dernière politique en matière de rapports sur les dépenses. En réponse, le programme de l'ordinateur B renverrait un objet et des informations de méthode associées qui permettraient au programme de l'ordinateur A de sélectionner les données du compte de dépenses de l'utilisateur d'une manière compatible avec la dernière politique. Si la politique change, un changement est nécessaire pour un programme dans un seul ordinateur (ordinateur B).

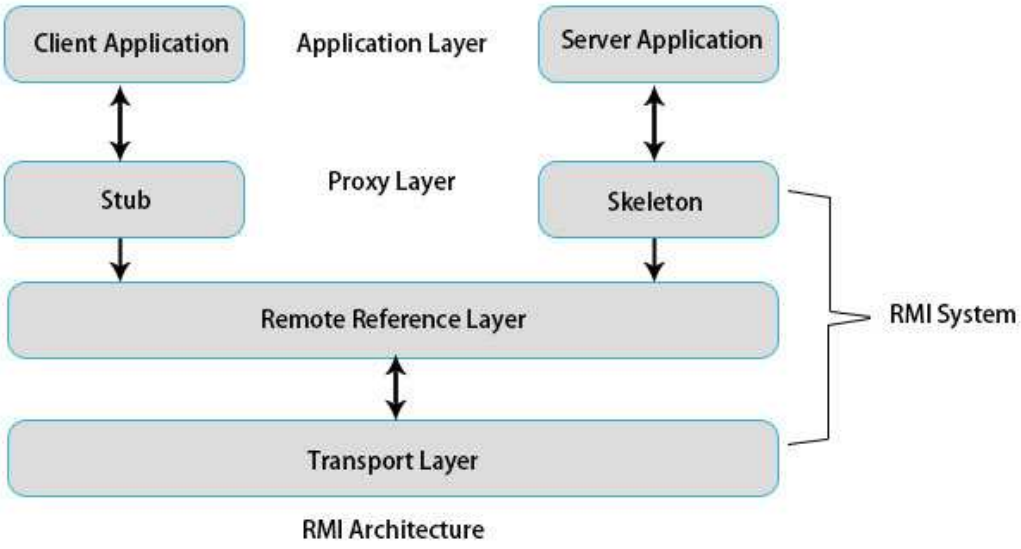

Figure 7: Architecture RMI

3.1.2 CORBA

CORBA (Common Object Request Broker Architecture) **est une norme rédigée par l'OMG. Elle définit une architecture distribuée fondée sur le concept d'objet.**

CORBA est une architecture distribuée dans laquelle des clients émettent des requêtes à destination d'objets, qui s'exécutent dans des processus serveurs. Clients et objets s'exécutent dans des environnements hétérogènes, sur des machines généralement distantes.

CORBA est une architecture logicielle pour le développement de composants et d'object ORB. Ces composants, qui sont assemblés afin de construire des applications complètes, peuvent être écrits dans des langages de programmation distincts, être exécutés dans des processus séparés, voire être déployés sur des machines distinctes.

3.1.2.1 Les objets ORB

L'Object Request Broker (ORB), **est un composant fondamental de l'architecture CORBA** ; **sa mission est de faciliter la communication entre objets** : il est chargé d'envoyer les requêtes aux objets et de retourner les réponses au client qui les a invoqués par un processus de sérialisation.

3.1.2.2 Le langage IDL

Le langage de définition des interfaces ou IDL (Interface Definition Language**), est utilisé pour définir les interfaces entre les composants** d'une application et permettre l'interopérabilité entre différentes technologies.

3.1.2.3 IIOP

Le protocole IIOP, pour Internet Inter-ORB Protocol, **est le protocole de communication utilisé par CORBA**. C'est une implémentation s'appuyant sur un transport TCP/IP du protocole de plus haut-niveau GIOP (General Inter-ORB Protocol).

3.1.2.4 L'architecture CORBA

Le cœur de la spécification CORBA est l'Object Request Broker (ORB) qui est responsable de la gestion des objets, de la transmission des messages entre les objets et de l'interface entre les objets et les services externes. Un objet est une entité identifiable et encapsulée qui fournit un ou plusieurs services demandés par les clients. Les demandes sont des événements qui ont des paramètres et un objet cible, et peuvent être générées par un objet client lorsqu'un service ou une information est requis. Les objets ont des références uniques dans tout le système ; ces références peuvent être utilisées par des applications sans savoir où résident les objets. Cela offre une « transparence du réseau » et permet aux programmeurs de développer des systèmes avec un espace d'adressage à l'échelle du réseau (Figure 8).

L'ORB permet de construire des applications à la fois au moment de la compilation et de l'exécution. Le langage de définition d'interface (IDL) définit des structures d'objets afin que des applications ayant une connaissance préalable des objets avec lesquels elles

interopéreront puissent être construites. L'interface d'invocation dynamique (DII) permet d'accéder aux structures d'objets au moment de l'exécution et permet ainsi aux applications d'être assemblées à partir d'objets qui n'ont aucune connaissance préalable les uns des autres.

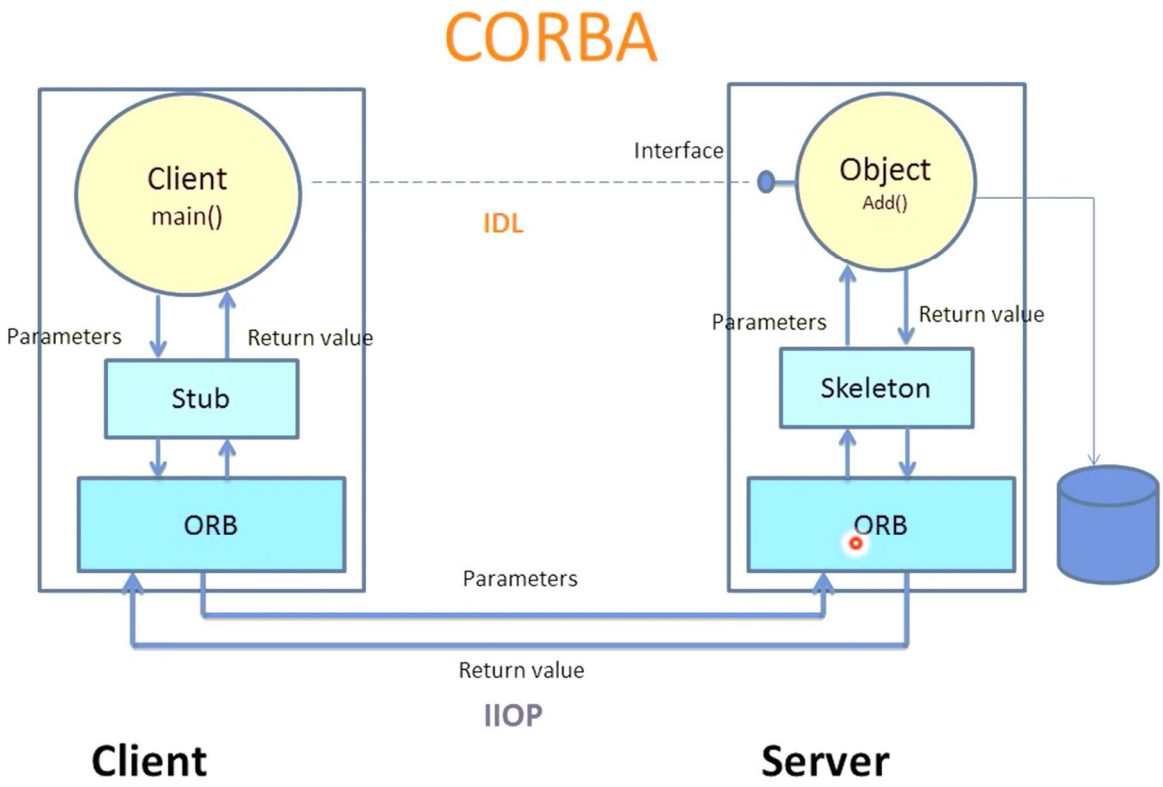

Figure 8: Architecture CORBA

3.2 Les services web

Un Web Service permet à des applications d'échanger des informations sans avoir forcément été construites dans le même langage de programmation. C'est le cas d'un ordinateur et d'un serveur distant.

Un Web Service est une application qui permet d'échanger des données avec d'autres applications web. Même si ces dernières sont construites dans des langages de programmation différents. Parmi les Web Services les plus connus on peut citer SOAP, REST ou HTTP. Elles sont utilisées généralement sur des infrastructure cloud, en cloud public, privé ou en cloud hybride.

3.2.1 Le fonctionnement d'un Web Service

Un Web Service fonctionne de la manière suivante (Figure 9) :

1. Le client (en général un utilisateur sur un ordinateur ou un support équivalent) effectue une requête dans un des langages suivants : XML, JSON ou HTTP.
2. Cette requête est transmise à un serveur distant via les protocoles SOAP, REST ou HTTP.

3. La réponse est ensuite délivrée sous le même format que sa demande : XML, JSON ou HTTP.

Figure 9: Le fonctionnement d'un service web

3.2.2 Modèle en couche

Et ce grâce à un modèle en couche (Figure 10). La 1re couche est l'invocation : elle décrit la structure des messages échangés par le client et le serveur. Elle s'effectue à l'aide des standards XML-RPC ou SOAP. La 2e est la découverte. C'est la phase de recherche et de localisation des données demandées par le client, le plus souvent via le protocole UDDI. La 3e est la description qui stipule les paramètres des fonctions et les types de données des services web utilisés. Son protocole standard est le WSDL qui repose sur la notation XML.

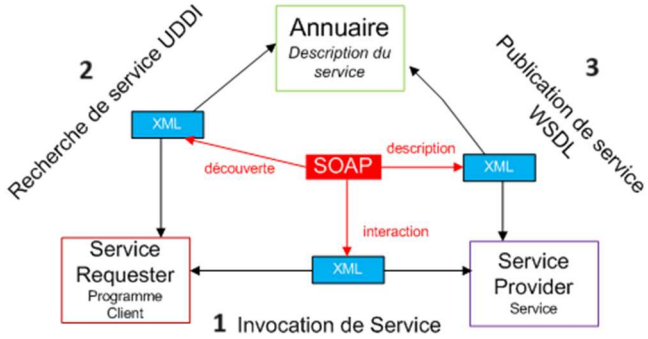

Figure 10: modèle en couche pour les services web