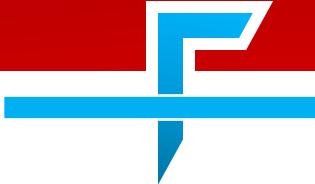




Introduction



✓ Middleware

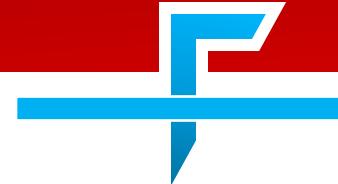
- En architecture informatique, un middleware (anglicisme) ou intericiel est un logiciel tiers qui crée un réseau d'échange d'informations entre différentes applications informatiques.
- En architecture Client – Serveur, le middleware s'intègre en-dessous de la couche Application, et au-dessus de la couche transport (TCP)

• Exemples de middleware:

- CORBA, RMI, Java Remote Method (RMI),
- TIBCO Rendezvous, SOAP, Microsoft COM, .NET, IBM MQ Series



Introduction



✓ Broker de messages

- A message broker (also known as an integration broker or interface engine [1]) is an intermediary computer [program module](#) that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver.
- Message brokers are elements in telecommunication or computer networks where software applications communicate by exchanging formally-defined messages. [1]
- Message brokers are a building block of [message-oriented middleware](#) (MOM) but are typically not a replacement for traditional middleware like MOM and [remote procedure call](#)(RPC) [2,3] .

[1] "[IB \(integration broker\)](#)". IT Glossary. Gartner, Inc. Retrieved 17 May 2018.

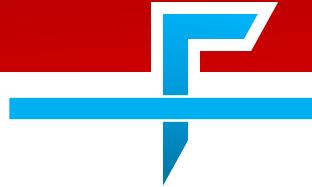
[2] Kale, V. (2014). "[Integration Technologies](#)". [Guide to Cloud Computing for Business and Technology Managers: From Distributed Computing to Cloudware Applications](#). CRC Press. pp. 107–134. [ISBN 9781482219227](#). Retrieved 17 May 2018.

[3] Samtani, G.; Sadhwani, D. (2013). "[Integration Brokers and Web Services](#)". In Clark, M.; Fletcher, P.; Hanson, J.J.; et al. [Web Services Business Strategies and Architectures](#). Apress. pp. 71–84. [ISBN 9781430253563](#). Retrieved 17 May 2018.

OMA

(Object Management
Architecture)

Object Management Group (OMG)



- Organisation internationale, ouverte et à but non-lucratif promouvant l'usage des technologies Orienté-Objet (OO)
- Crée en 1989 et composée aujourd'hui de plus de 750 membres dont :
 - Universités, fournisseurs de logiciels, Décideurs,...

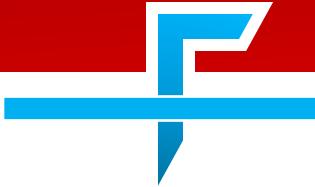


Objectifs de l'OMG:

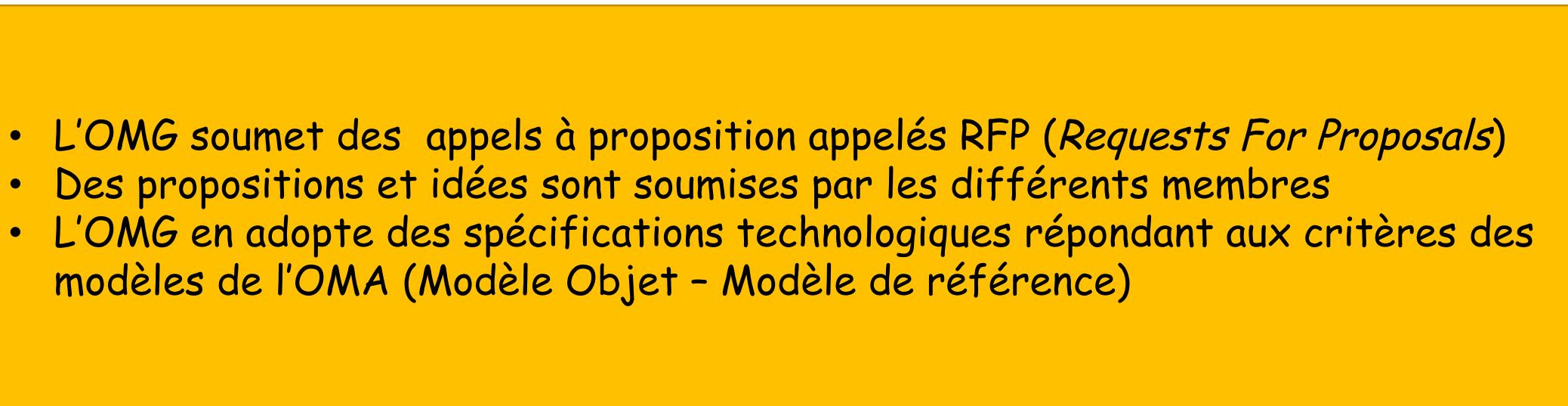
OBJECT MANAGEMENT GROUP

- Définition d'architectures et de spécifications standards pour les systèmes basés sur l'Orienté-Objet (OO)
- Exemples de standards définis par l'OMG
 - **OMA**, CORBA, IDL
 - UML

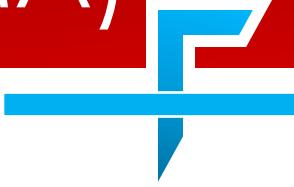
Object Management Architecture (OMA)



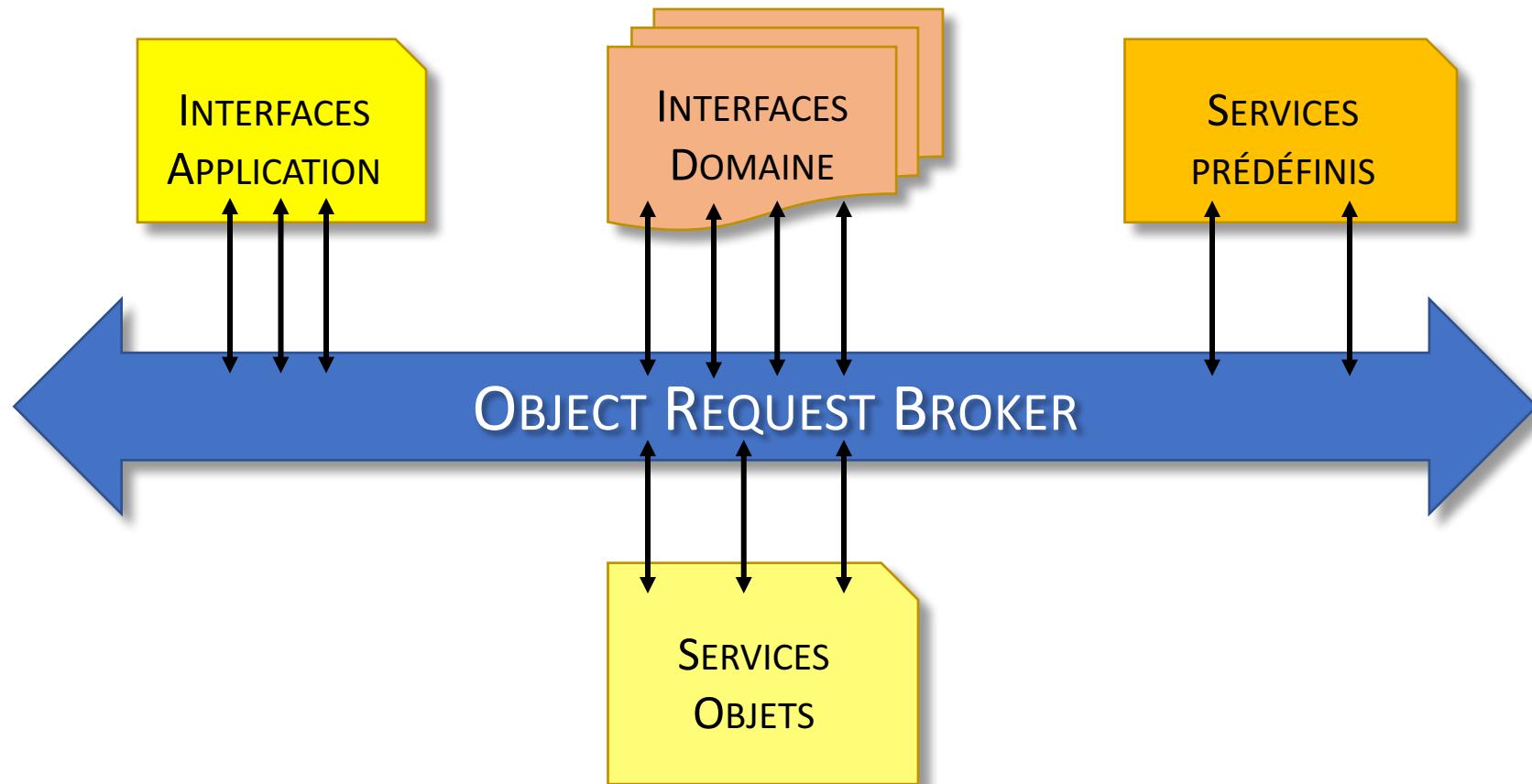
- **Architecture** composée de deux modèles
 - Modèle Objet
 - Modèle de référence
- Modèle Objet
 - Fournit une description des objets dans un système hétérogène
- Modèle de référence
 - Caractérise les interactions entre les objets



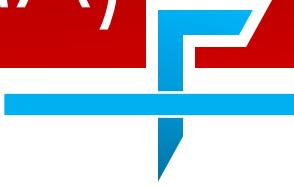
Object Management Architecture (OMA)



- Composants de l'OMA



Object Management Architecture (OMA)



- Services objets

- Services de nommage (Naming Service)
 - Permet aux clients de retrouver les objets par leurs noms
- Services de « trading »
 - Permet aux clients de retrouver les objets par leurs propriétés
- Services de notification, transaction, sécurité, etc.

- Services prédefinis

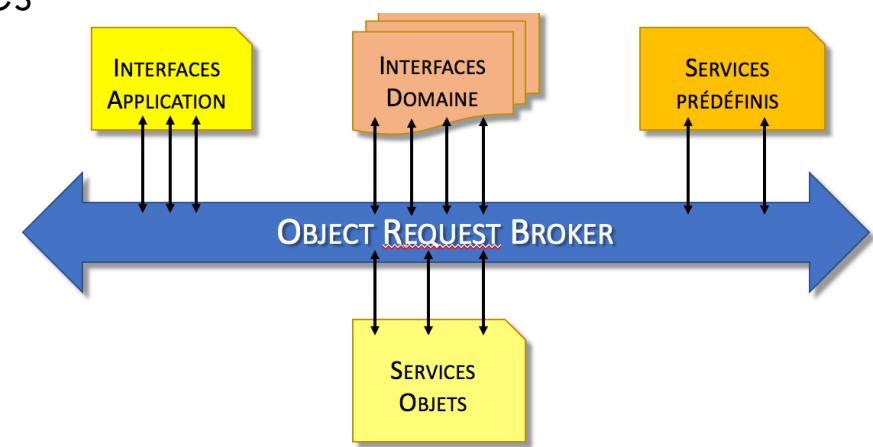
- Services orientés utilisateurs
- Exemple: Service de gestion de documents (DDCF)

- Interfaces domaines

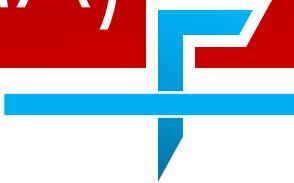
- Services orientés domaine d'application (télécoms, médical, finance, etc.)

- Interfaces application

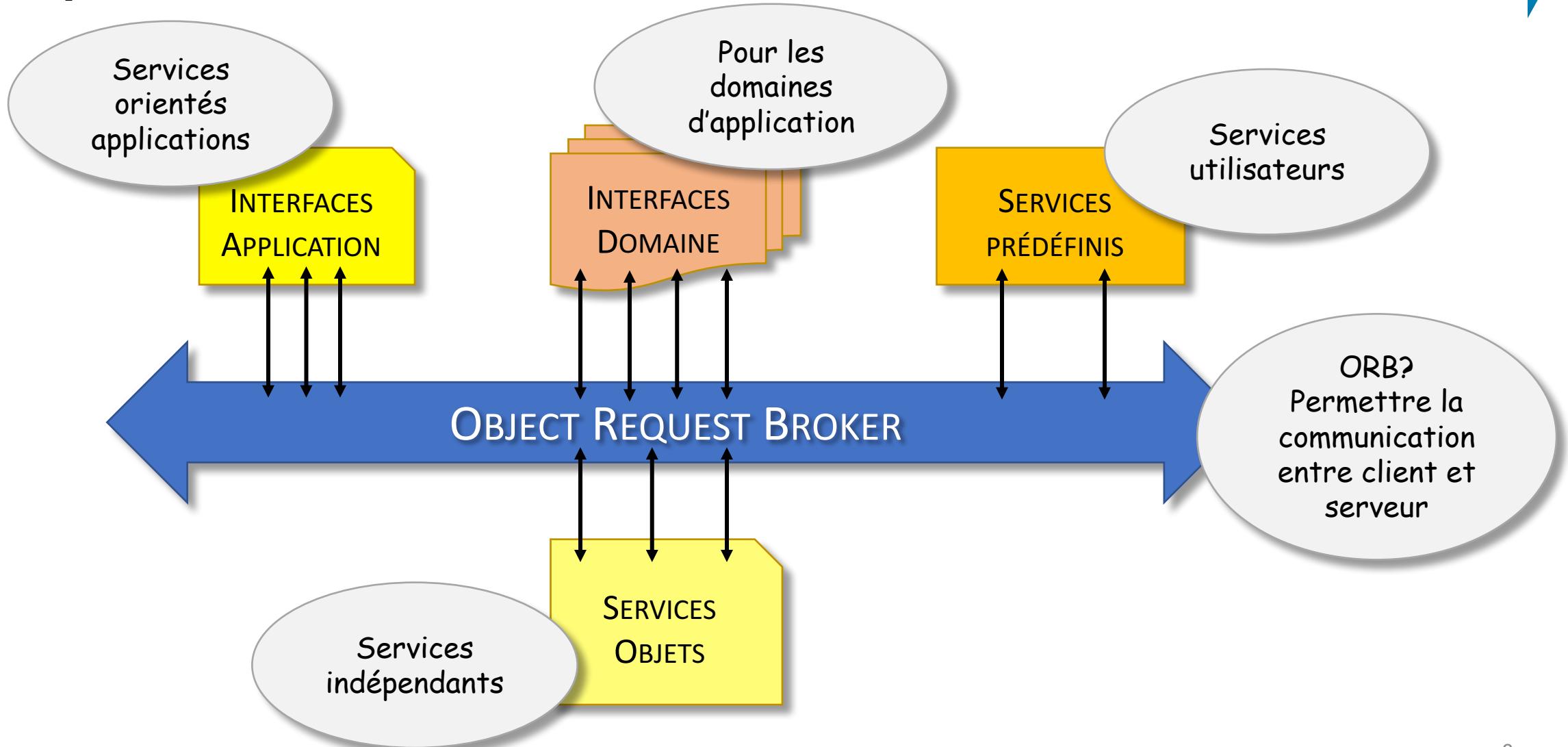
- Interfaces des applications



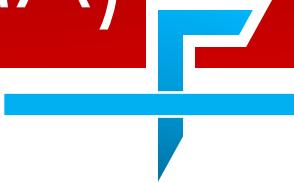
Object Management Architecture (OMA)



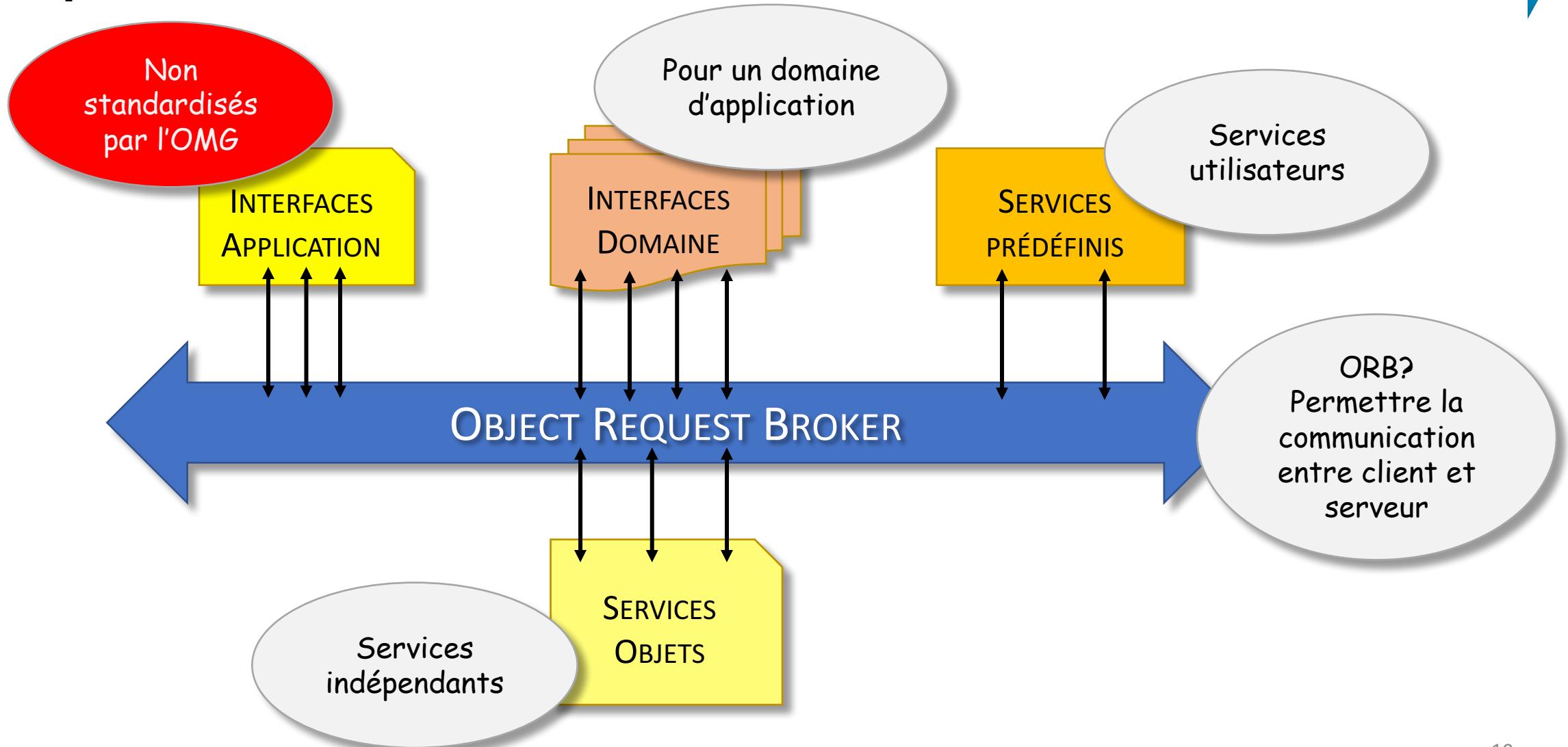
- Composants de l'OMA



Object Management Architecture (OMA)



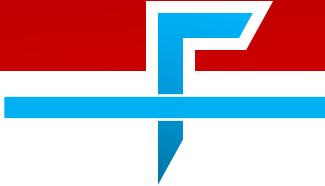
• Composants de l'OMA



ORB

(Object Request Broker)

Object Request Broker



Object Request Broker

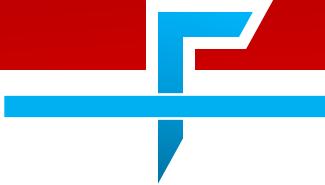
« Gestionnaire de requêtes sur des objets »

- Object Request Broker (ORB) :
 - L'ORB est basé sur l'Orienté-Objet
 - **RPC (Procédure d'appel à distance)** basé sur l'Orienté-Objet

ORB ?

L'ORB est un middleware utilisant les spécifications de l'OMG. L'ORB gère tous les détails concernant l'invocation d'un objet par un client, et achemine la réponse à sa destination. L'ORB est aussi le garant de l'Interface Repository (IR). L'IR est une base de données contenant les différentes interfaces et leurs définitions.

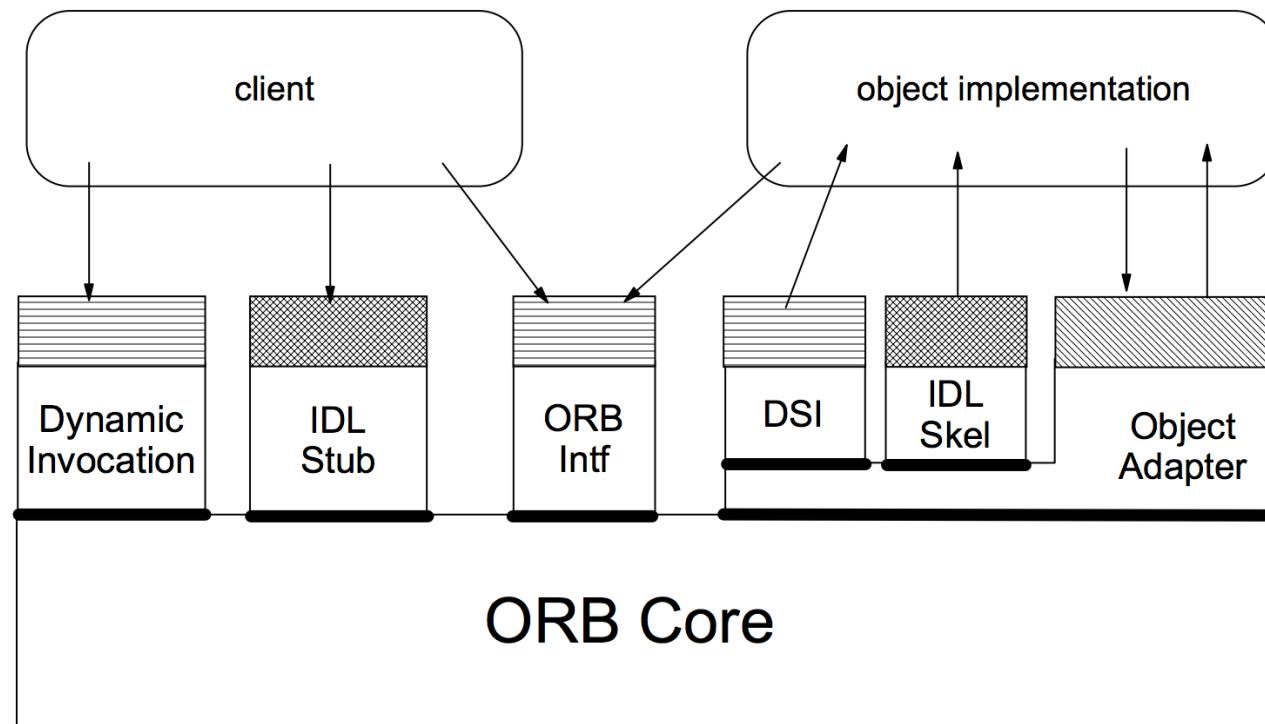
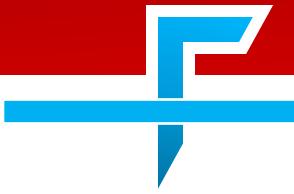
Architecture de l'ORB (1)



Rôle de l'ORB:

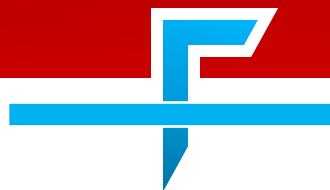
- Gère la transmission et la réception d'objets par le client
- Caractéristiques de l'ORB
 - Le client ne connaît pas où réside l'objet cible
 - Le client ne connaît pas comment l'objet est implémenté
 - Le client ne connaît pas le langage de programmation ou script avec lequel le code de l'objet est écrit
 - Le client ne connaît pas l'OS, ni la plate forme matérielle cible
 - Lors d'une requête sur un objet cible, le client ne connaît pas la procédure d'activation de celui-ci
 - Le client ne connaît pas le mécanisme de communication par lequel la requête est délivrée à l'objet,
 - Et par lequel le résultat est retourné

Architecture de l'ORB (2)



- Same for all ORBs
- Interface-specific stubs and skeletons
- There may be multiple object adapters
- ORB-private interface

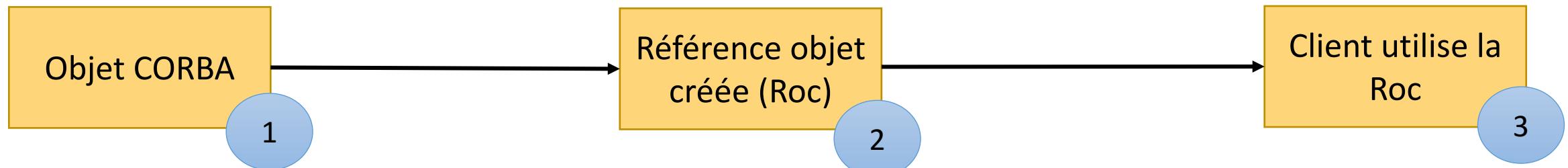
Références Objets Clients



Pour effectuer une requête, le client utilise une référence objet

Les références objets sont créées dès la création d'un objet CORBA

Une référence objet se réfère de manière unique à un seul objet CORBA



Le contenu du Roc est inconnu au client

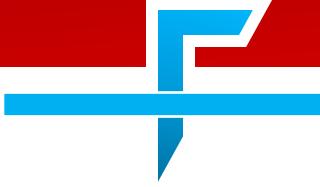
Seul l'ORB connaît et peut opérer sur le contenu du Roc

Le client peut obtenir des références objets

- par le service de nommage,
- par invocation d'un service de recherche,
- par création d'un objet par l'intermédiaire de l'Interface Factory (IF)
 - Une référence est créée dès la création d'un objet par le client

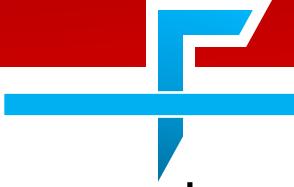
CORBA

CORBA



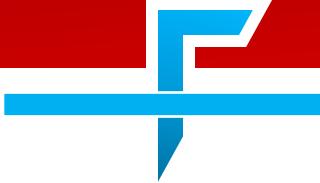
- Common Object Request Broker Architecture
- Architecture standardisée de ORB
- **Object Request Broker (ORB)** :
 - Procédure d'appel à distance basée sur l'Orienté-Objet
 - Un ORB est un middleware permettant des appels de procédure entre des machines distantes
 - Garantit la transparence dans la procédure de communication
 - Facilite l'interopérabilité entre différents SD Orienté-Objet

Quelques avantages de CORBA



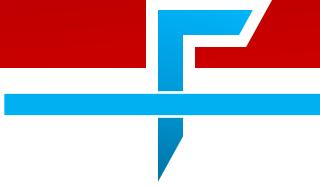
- CORBA permet l'interopérabilité entre différentes applications, et sous diverses contraintes :
 - Sur des machines différentes (connectées en réseau)
 - Sur des OS différents (Windows, Unix, Mac, Solaris, ...)
 - Sur différents types de processeurs (SPARC, Intel, PowerPC,)
 - Compatible avec plusieurs langages de programmation
 - Java, C, C++, Smalltalk, COBOL, Python, PHP, Lisp, Ada, ...

Architecture CORBA



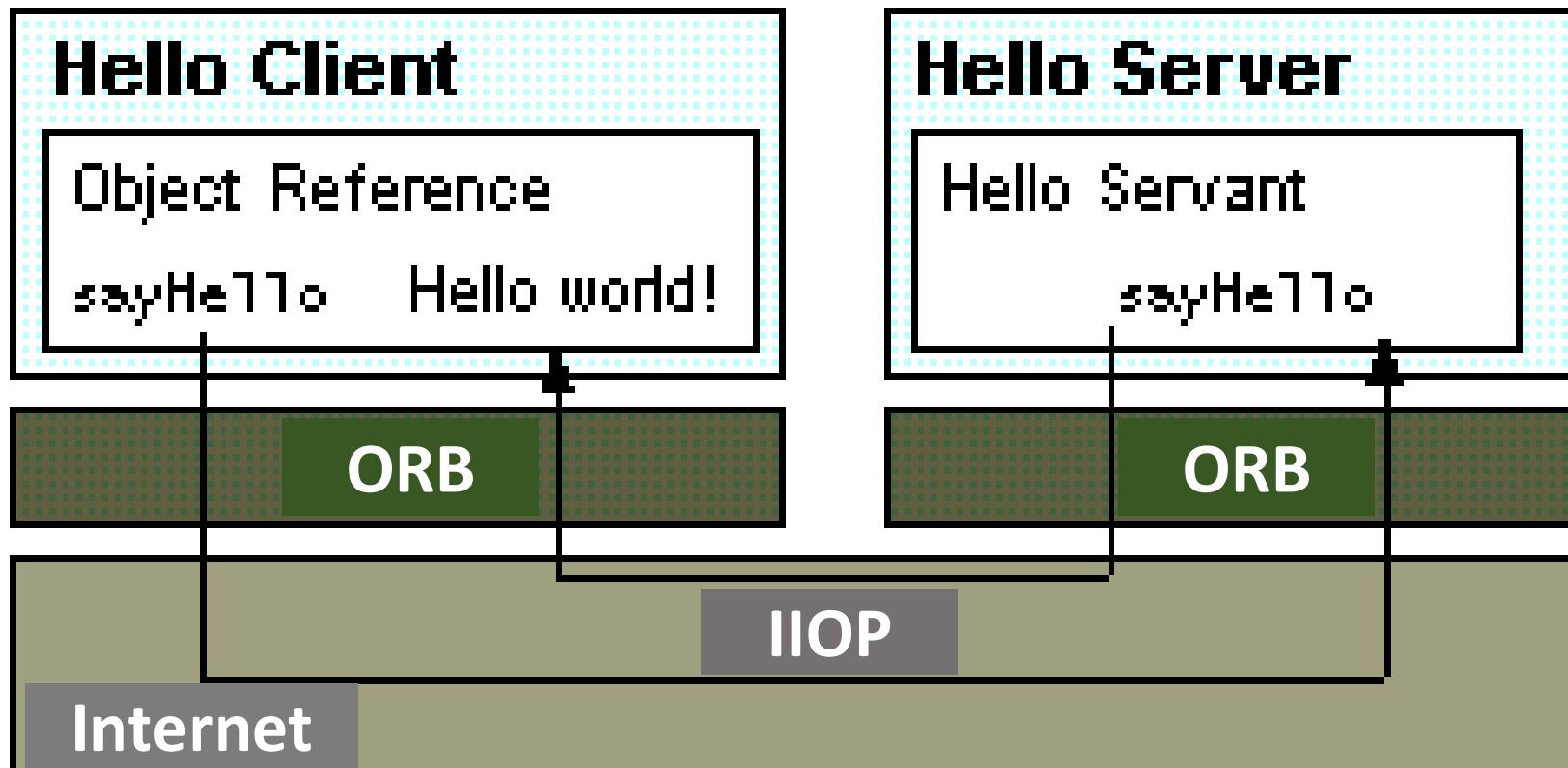
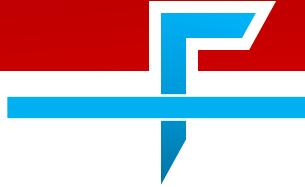
- Eléments d'architecture de CORBA
 - **IDL** (Interface Definition Language)
 - **ORB** (Object Request Broker) responsable des interactions entre différents objets, ou applications les utilisant
 - **POA** (Portable Object Adaptor) responsable de l'activation et de la désactivation des objets, ainsi que le mapping entre l'ID de l'objet appelé, et son implémentation
 - **Naming service** (Service de nommage) permet aux clients distants de retrouver les objets distants sur un réseau
 - **IOP** (Internet Inter ORB Protocol) définit les interactions entre objets communicants sur Internet

Architecture CORBA

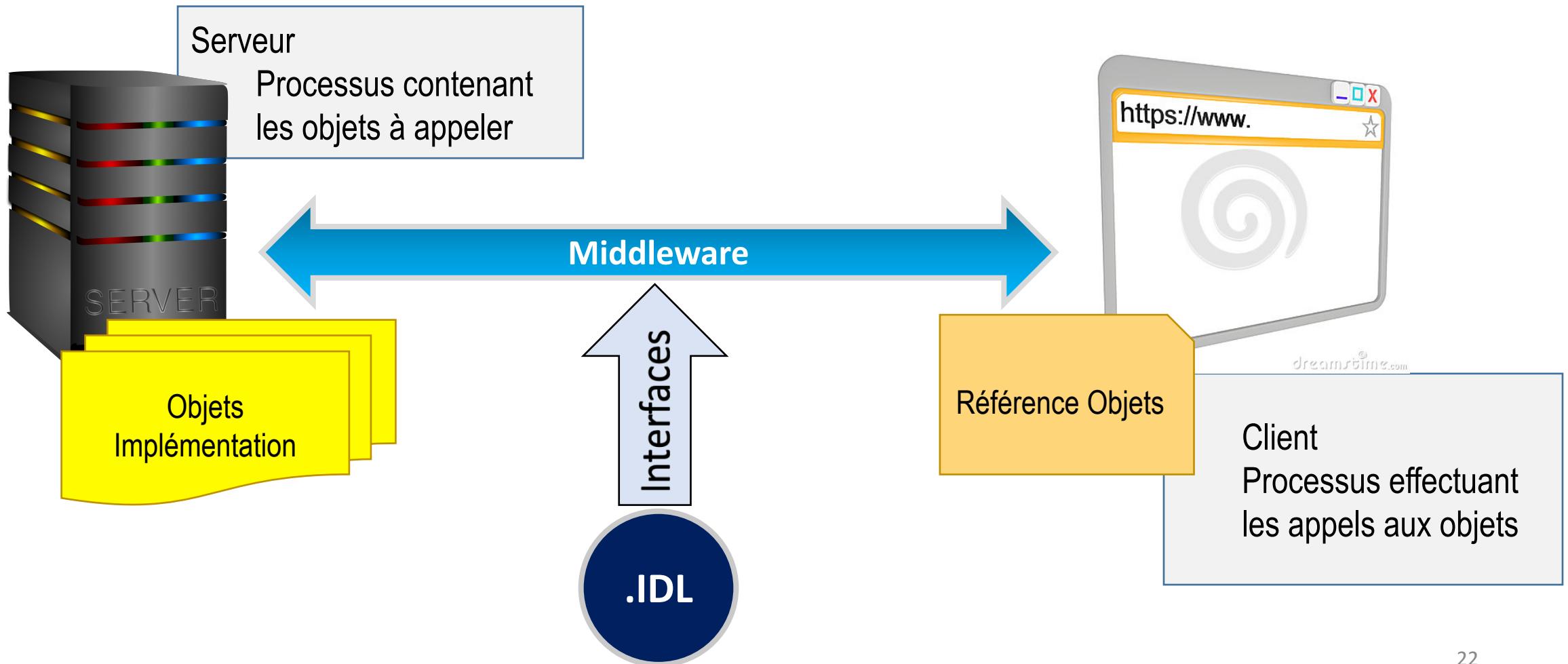
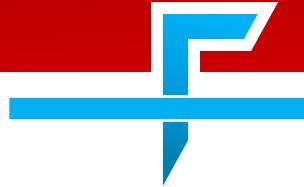


- La plupart des technologies RPC entrevoient Client-Serveur en terme d'applications
- Sous CORBA, le Client et le Serveur sont considérés par rapport à l'objet
 - Le Client détient une référence sur l'objet distant.
 - La référence est liée à une **souche (stub client)** qui est également liée à l'ORB.
 - Lorsque le client invoque un objet, la **souche** transmet l'appel à l'ORB qui se charge de router l'appel vers le serveur distant.
- Le serveur de son côté utilise un **squelette** pour traduire l'appel distant en une invocation de méthode sur un objet local.
- A la terminaison du processus, le **squelette** du serveur transmet le résultat (y compris les erreurs), et les renvoie au client via l'ORB.

Architecture CORBA - Exemple



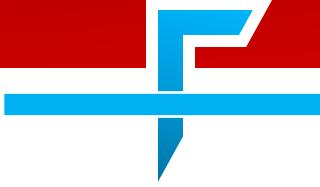
CLIENT-SERVEUR SOUS CORBA



IDL

(Interface Definition Language)

Langage IDL

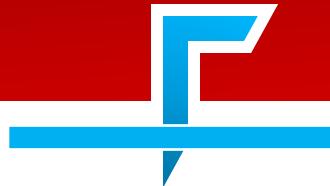


- Avant d'invoquer un objet, le client doit connaître les types d'opérations supportées par celui-ci.
- Sous CORBA, les types d'opérations faisables sur un objet sont définis sous forme d'Interface.
- Les interfaces sont définies avec un **Langage de Définition d'Interface (IDL)**

```
1. // Définition d'interface IDL
2. interface Factory {
3.     Object create();
4. };
```

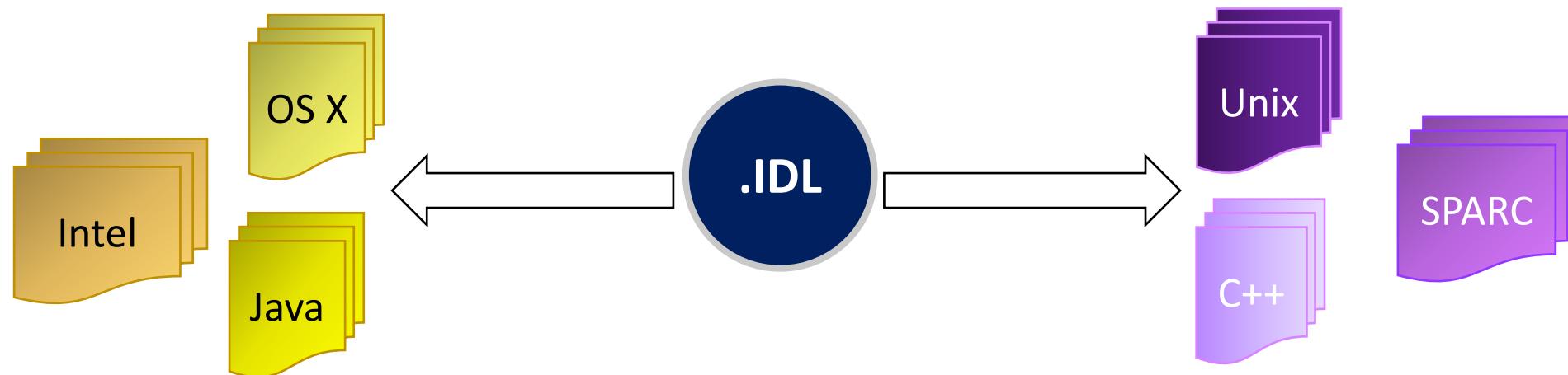
- L'interface **Factory** contient une méthode **create()** servant à créer un objet
- Le client invoque une référence objet de type **Factory** pour créer un nouvel objet

Langage IDL

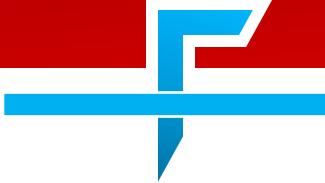


Interface Definition/Description Language

- IDL est un langage de description utilisé pour décrire la partie visible d'un objet
 - Attributs - Méthodes
- **Objectifs**
 - Définir les composants d'un API « indépendamment » du langage de programmation
 - **Exemple** : Pour permettre à deux applications (l'une en Java et l'autre en C++) d'interagir, IDL permet de définir dans ce cas les méthodes de l'API sous une interface, qui sera traduit des deux cotés à travers un mapping, afin de permettre la communication.
 - IDL introduit un **pont** entre différents systèmes

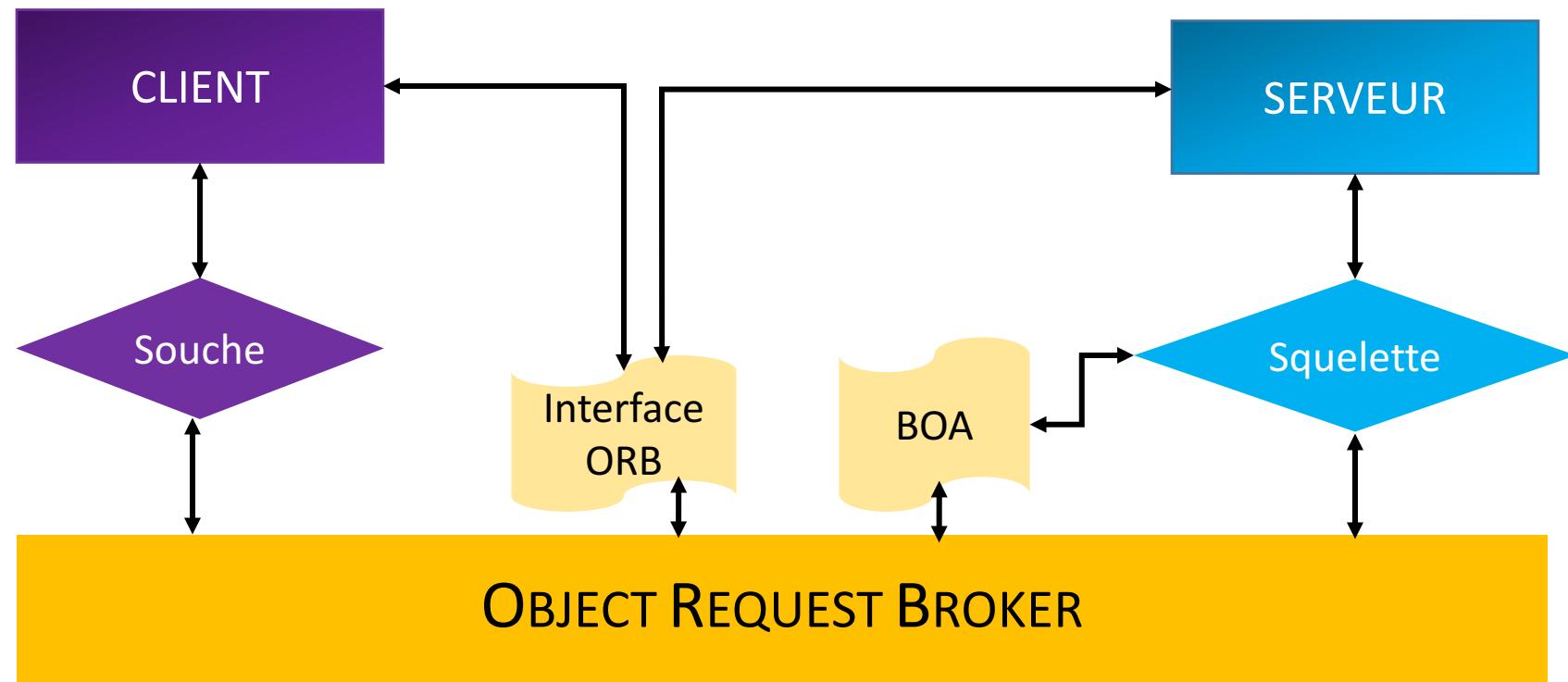


Langage IDL

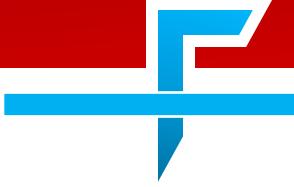


Interface Definition/Description Language

- IDL est un langage de description utilisé pour décrire la partie visible d'un objet
 - Attributs - Méthodes
- **Principe de l'IDL**
 - Séparer l'interface d'un objet et son implémentation



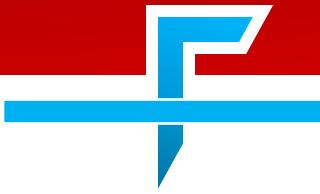
IDL – Exemple (1)



```
module Finance {  
    typedef sequence<string> StringSeq;  
    struct AccountDetails {  
        string      name;  
        StringSeq   address;  
        long        account_number;  
        double      current_balance;  
    };  
    exception insufficientFunds { };  
    interface Account {  
        void deposit(in double amount);  
        void withdraw(in double amount)  
            raises(insufficientFunds);  
        readonly attribute AccountDetails details;  
    };  
};
```

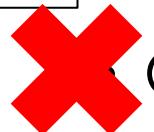
- Un interface IDL est composé de :
 - Attributs
 - Méthodes
- Les paramètres des méthodes sont spécifiés par :
 - « in » : Client → Serveur
 - « out » : Serveur → Client
 - « inout » : Client <--> Serveur
- L'interface Account fournit deux méthodes
 - deposit et withdraw
 - 4 attributs + l'attribut details en lecture seule

IDL – Exemple (2)



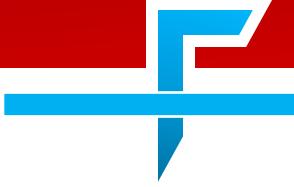
```
module Finance {
    typedef sequence<string> StringSeq;
    struct AccountDetails {
        string      name;
        StringSeq   address;
        long        account_number;
        double      current_balance;
    };
    exception insufficientFunds { };
    interface Account {
        void deposit(in double amount);
        void withdraw(in double amount)
            raises(insufficientFunds);
        readonly attribute AccountDetails details;
    };
};
```

- IDL expose plusieurs types d'exceptions prédéfinies qu'utilise le système CORBA
- Les exceptions prédéfinies sont utilisables par le programmeur
- D'autres exceptions peuvent être définies en addition à celles prédéfinies



Cependant, IDL ne permet pas la redéfinition d'exceptions par héritage sur les exceptions prédéfinies

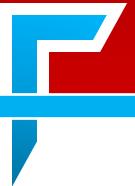
IDL – Exemple (3)



```
module Finance {
    typedef sequence<string> StringSeq;
    struct AccountDetails {
        string      name;
        StringSeq   address;
        long        account_number;
        double      current_balance;
    };
    exception insufficientFunds { };
    interface Account {
        void deposit(in double amount);
        void withdraw(in double amount)
            raises(insufficientFunds);
        readonly attribute AccountDetails details;
    };
};
```

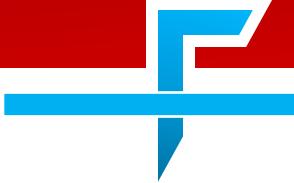
- Les paramètres de retour d'une méthode IDL peuvent être de différents types :
 - Types prédéfinis
 - String - Boolean - Long
 - Types définis
 - Structure
 - Sequence
 - Tableau
 - Type de variables (Typedef)
 - Union

IDL - Préprocesseur C++



- Un **compilateur IDL** utilise un préprocesseur C++ pour exécuter les différents directives de preprocessing inclus dans le fichier source .IDL
- Directives de preprocessing
 - **#include** (Inclut les contenus des fichiers spécifiés)
 - **#ifndef** (Vérifie si l'élément spécifié a déjà été défini)
 - **#define** (Définit les constantes, macros, variables globales, ...)
 - **#endif** (Fin de condition)
 - **#pragma** (Méthode spécifiée par le standard C pour fournir des instructions complémentaires au compilateur et externes au langage)

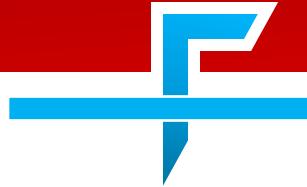
IDL – Interface « Factory »



- Un client IDL ne peut pas créer un objet placé sur un serveur distant
- Un nouvel objet est créé une fois que le client aura invoqué une des méthodes sur un objet existant du coté serveur.
- **Interface Factory**
 - Objet pouvant créer un autre objet
 - Méthode « create » pour créer un objet
 - L'Interface « Factory » permet d'instancier un **constructeur** sur un objet serveur
 - Exemple :

```
1. interface Foo {  
2.     void destroy();  
3.     ...  
4. };  
5. interface FooFactory {  
6.     Foo create(...);  
7.     ...  
8. };
```

IDL et Héritage



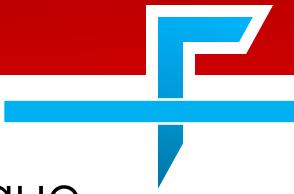
- IDL permet la redéfinition d'interfaces par héritage
- Toutes les interfaces IDL héritent de l'interface « Object » défini sous CORBA

```
1. // CORBA::Object is the base interface  
2. // for all interfaces  
3. interface Factory : Object { ... };
```

On peut omettre la
partie héritée
comme ceci !

```
1. // CORBA::Object is the base interface  
2. // for all interfaces  
3. interface Factory { ... };
```

IDL et Références objets



- IDL a défini un interface appelé FactoryFinder contenant un type séquence spécifique appelé FactorySeq
- FactorySeq : type séquence représentant une séquence illimitée de références aux objets

```
1. // OMG IDL
2. interface FactoryFinder {
3.     // define a sequence of Factory
4.     // object references
5.     typedef sequence <Factory> FactorySeq;
6.     FactorySeq find_factories(in string interface_name);
7. }
```

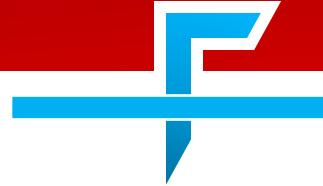
- La méthode « `find_factories` » prend en argument un `string` et retourne une liste de références objets

IDL Mapping (1)



- IDL est par définition un langage déclaratif
- IDL n'offre pas toutes les possibilités en programmation d'applications
- **D'où la nécessité d'élaborer des mappings**
- IDL Mapping est un ensemble d'éléments de correspondance entre IDL et un langage de programmation donné
- Exemples de mappings standards :
 - C, C++, Java, Python, PHP
 - Ada, Smalltalk, COBOL

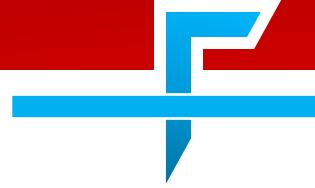
IDL Mapping (2)



Quelques règles de mapping

- Pour un langage de programmation Objet
 - Interface correspond à Classe
 - Opération correspond à Méthode
- Pour un langage de programmation structuré
 - Exemple (Langage C)
 - Interface correspond à struct
 - Opération correspond à fonction

IDL Mapping (C++)



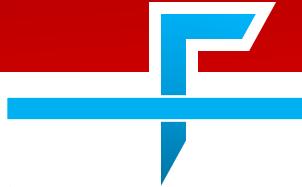
.IDL

- 1) Interfaces
- 2) Opérations
- 3) long, short, float, double, char, boolean
- 4) Enum
- 5) octet
- 6) Any
- 7) struct class
- 8) string
- 9) wstring
- 10) sequence
- 11) fixed
- 12) object reference

C++

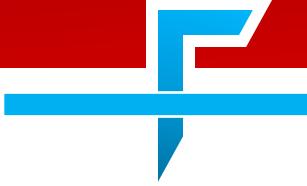
- 1) Classes
- 2) Méthodes
- 3) long, short, float, double, char, Bool
- 4) Enum
- 5) unsigned char
- 6) Any Class
- 7) struct union
- 8) char*
- 9) wchar t*
- 10) Class
- 11) Fixed template class
- 12) pointer or object

Invocation d'objets



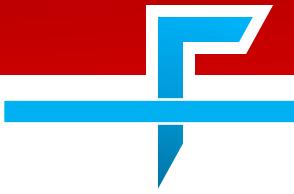
- Il existe deux types manière d'invoquer un objet sous CORBA:
 - Invocation statique
 - Le stub et le squelette se chargent de canaliser la requête, alors que le squelette expédie la requête à l'objet cible
 - Invocation dynamique
 - CORBA supporte deux interfaces pour l'invocation dynamique d'un objet :
 - Dynamic Invocation Interface (DII)
 - Dynamic Skeleton Interface (DSI)

Invocation statique



- Lors de l'invocation d'un objet par un client, le « stub/souche » et le « squelette » générés par l'ORB, respectivement coté Client et coté Serveur se chargent de l'acheminement et de l'expédition de la requête
- **Coté Client**
 - La souche permet de déclencher une requête coté Client.
 - La souche génère la requête en transformant l' invocation de sa forme originelle en une forme transmissible vers l'objet cible
- **Coté Serveur**
 - Le squelette réceptionne l'objet et expédie la requête à l'objet CORBA.
 - Le squelette traduit la requête sous le langage de programmation de la plate forme cible.

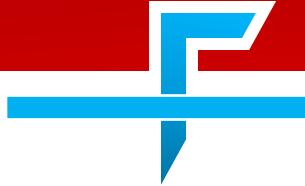
Invocation statique (Exemple)



- Dans cet exemple, un objet Factory_var est référencé (objref).
- Une requête est invoquée à travers l'objet créé en utilisant un appel de méthode de la classe Factory_var.
- Le stub côté client traduit la requête donnée dans cet exemple sous une forme transmissible tenant compte du protocole de transport utilisé.
- Le squelette reçoit la requête sous cette forme, et la convertit sous le langage de la plate forme cible et fait appel à l'objet concerné.

```
1. // C++
2. Factory_var factory_objref;
3. // Initialiser factory_objref en utilisant le service de nommage ou
4. // (omis pour cet exemple), et déclencher une requête
5. Object_var objref = factory_objref->create();
```

Invocation dynamique



L'invocation d'objets sous CORBA via le « stub » et le squelette est statique.

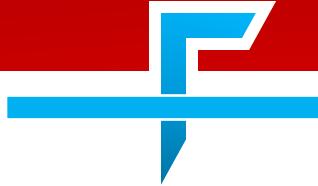
CORBA supporte une invocation dynamique à travers deux interfaces :

- Dynamic Invocation Interface - DII (invocation dynamique par un client)
- Dynamic Skeleton Interface - DSİ (accès dynamique aux objets)

DII et DSİ peuvent être vus respectivement comme un « stub » générique et un squelette générique

Les interfaces du DII et du DSİ sont fournies par l'ORB, indépendamment des interfaces des objets de l'IDL.

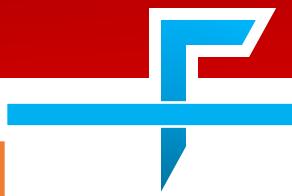
Invocation dynamique



Une invocation dynamique est effectuée en appelant la méthode **create_request** défini dans **CORBA::Object**.

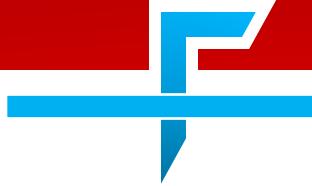
Etant donné que tous les objets CORBA héritent de l'interface **CORBA::Object**, tous les objets peuvent invoquer **create_request** pour une requête dynamique

Invocations statique vs dynamique



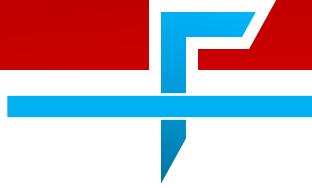
	Invocation statique	Invocation dynamique
Utilisation de stub et squelette ?	Oui	Non
Appels de méthodes ?	Méthode create (CORBA::Object)	Méthode create_request (CORBA::Object)
Type de compilation ?	Compilation statique	Compilation dynamique
Coût ?	La compilation génère tous les données requises pour les invocations	A chaque compilation, l'IR est interrogée par le système. Plus couteux
Flexibilité ?	Non	Oui

Quelques services CORBA



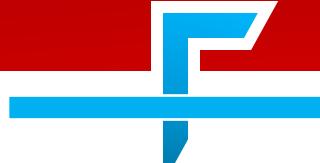
- CORBA propose une variété de **services**
 - **Object Life Cycle** : Service gérant la création, la destruction, et d'autres opérations sur les objets CORBA
 - **Naming Service** : Service proposant une structure de nommage pour les objets distants
 - **Persistence Object Service**: Service fournissant un modèle de communication (fournisseur – consommateur) créant un lien asynchrone entre objets communicants.
 - **Transactions** : Service coordonnant l'accès exclusif aux objets CORBA
 - **Contrôle de la concurrence** : Service proposant un mécanisme de blocage pour l'accès concurrent aux objets CORBA

Références URLs



1. <http://www.corba.org/.>
2. <http://www.corba.org/faq.htm>
3. http://www.service-architecture.com/articles/web-services/object_request_broker_orb.html
4. <http://docs.oracle.com/javase/7/docs/technotes/guides/idl/jidlExample.html>
5. [McHale, Ciaran. "Corba explained simply." URL http://www.ciaranmchale.com/corba-explained-simply/benefits-of-corba.html \(2007\).](http://www.ciaranmchale.com/corba-explained-simply/benefits-of-corba.html)
6. <http://corba.developpez.com/presentation/exemple/>
7. <http://www.sylbarth.com/corba/corba.html>
8. <https://www.unf.edu/~sahuja/cis6302/corbaexample.doc>

Autres références



1. Siegel, Jon, and Ph. D. CORBA 3 fundamentals and programming. Vol. 2. New York, NY, USA:: John Wiley & Sons, 2000.
2. Vinoski, Steve. "CORBA: integrating diverse applications within distributed heterogeneous environments." IEEE Communications magazine 35.2 (1997): 46-55.
3. Zinky, John A., David E. Bakken, and Richard E. Schantz. "Architectural support for quality of service for CORBA objects." Theory and Practice of Object Systems 3.1 (1997): 55-73.

Boîte de réception x CORBA : What is C x Yahoo - connexion x NeoGPS/src at ma x 83989TCPA0107- x tutoriel langage c x LANGAGE C - 3 - x Apprenez à progra x Babacar

Sécurisé | https://www.youtube.com/watch?v=PpbJq5OA66Y

Applications Y f



corba distributed system



SUMMARY

- CORBA is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together (i.e., it supports multiple platforms).
- CORBA uses an interface definition language (IDL) to specify the interfaces which objects present to the outer world.

Computer science academy by Dinesh Sir



9:50 / 10:11



CORBA : What is CORBA in Distributed System ? : Distributed System

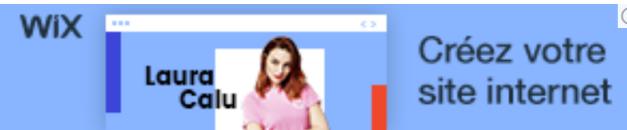
4 579 vues

1 37

20

PARTAGER

...



Créez un site

Annonce Wix.com

COMMENCEZ

À suivre

LECTURE AUTOMATIQUE

RMI

(REMOTE METHOD INVOCATION)
(Hindi)RMI remote method invocation
in hindiLast moment tuitions ✓
59 k vuesJoss Stone - I Put a Spell on
You (Luna Park, 2015)Jefferson Levi
Recommandée pour vous

What is API ?

Hitesh Choudhary ✓
310 k vuesNatural Mystic/Just a Little Bit
feat. Jack Johnson | Playing F...

exercicesGr5B (1).docx

...

GPSport.h

...

NMEAGPS.cp

...

NMEAGPS.h

...

Tout afficher

x

Page 1 sur 3

765 mots

Français

96 %

