Chapitre 3 : Middleware orienté objet

Partie II: Common Object Request Broker Architecture (CORBA)

1 Définitions

CORBA est une autre forme de modèle à objets Distribués. CORBA, acronyme de *Common Object Request Broker Architecture*, est une architecture logicielle, pour le développement de composants et d'*Object Request Broker* ou 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.

Un ORB est un ensemble de fonctions (classes Java, bibliothèques C++…) qui implémentent un « bus logiciel » par lequel des objets envoient et reçoivent des requêtes et des réponses, de manière transparente et portable : il s'agit de l'activation ou de l'invocation à distance par un objet, d'une méthode d'un autre objet distribué, en pratique les objets invoqués sont souvent des services. Pour qu'il y ait une communication entre un client et un serveur CORBA, il faut que le client et le serveur possèdent chacun un ORB : une application client-serveur possède au minimum deux ORB.

Échanges

- Entre objets liés par un même ORB
- Le protocole IIOP, pour ***I**nternet **I**nter-**O**RB **P**rotocol*, 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*).

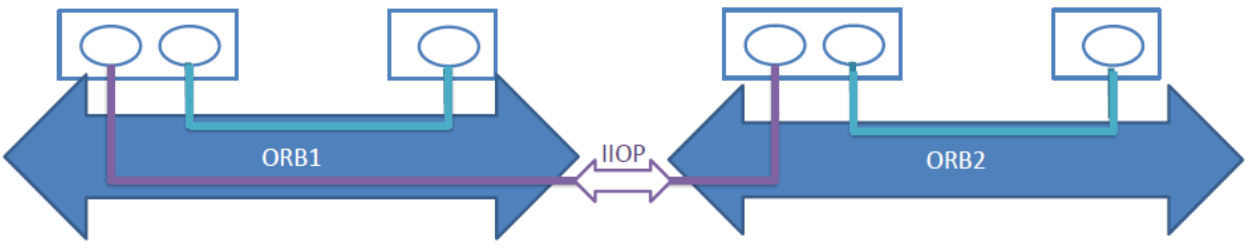


Figure 3.6- Protocole réseau utilisé par CORBA

2 Déroulement de requête sous CORBA

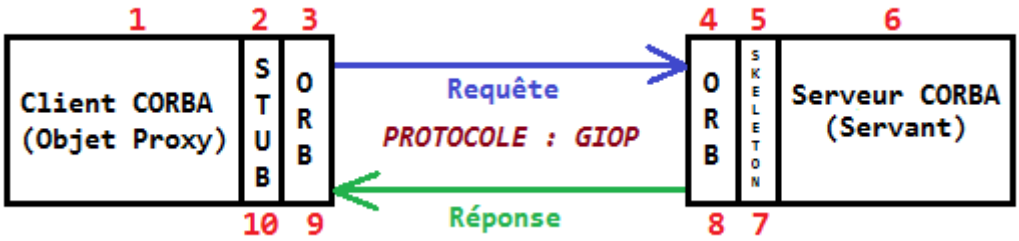


Figure 3.7- Déroulement de requête sous CORBA

1) Le client CORBA invoque une méthode distante.
2) Le Stub du client CORBA marshalise l'invocation de la méthode distante en requête CORBA.
3) La requête CORBA est envoyée à travers le réseau à partir l'ORB du client.
4) La requête CORBA est réceptionnée par l'ORB du serveur.
5) Le Skeleton du serveur CORBA démarshalise la requête CORBA.
6) Le serveur CORBA exécute le service lié à la méthode invoquée.
7) Le Skeleton du serveur CORBA marshalise la réponse CORBA.
8) La réponse CORBA est envoyée à travers le réseau à partir de l'ORB du serveur.
9) La réponse CORBA est réceptionnée par l'ORB du client.
10) Le Stub du client CORBA démarshalise la réponse CORBA.

3 Comment un client CORBA connaît-il le serveur CORBA?

La réponse est l'IOR. l'IOR est un objet contenant plusieurs informations permettant d'identifier un « servant » (la méthode distante).

Il existe deux moyens standards d'obtenir IOR :

- Le serveur stocke l'IOR dans un fichier et le client trouve un moyen de récupérer ce fichier pour lire son contenu. Il est possible de générer un objet proxy à partir de l'IOR grâce à une opération appelée « narrowing ».
- Le serveur possède un service appelé « NameService », c'est un programme qui tourne en daemon sur le serveur, il est plus connu sous le nom de « orbd » ou « tnameserv ». En utilisant l'host et port du serveur et un nom attribué au service, le client peut récupérer l'IOR et génère un objet proxy.

4 Interface Definition Language (IDL)

Un des avantages les plus importants de la spécification CORBA est de faciliter le développement des applications distribuées en évitant au développeur des difficultés concernant les mécanismes de communication, la location des applications, les clients et serveurs, le protocole de communication à utiliser ou le format de message à utiliser, et le plus important est que CORBA permet la communication entre différentes langues.

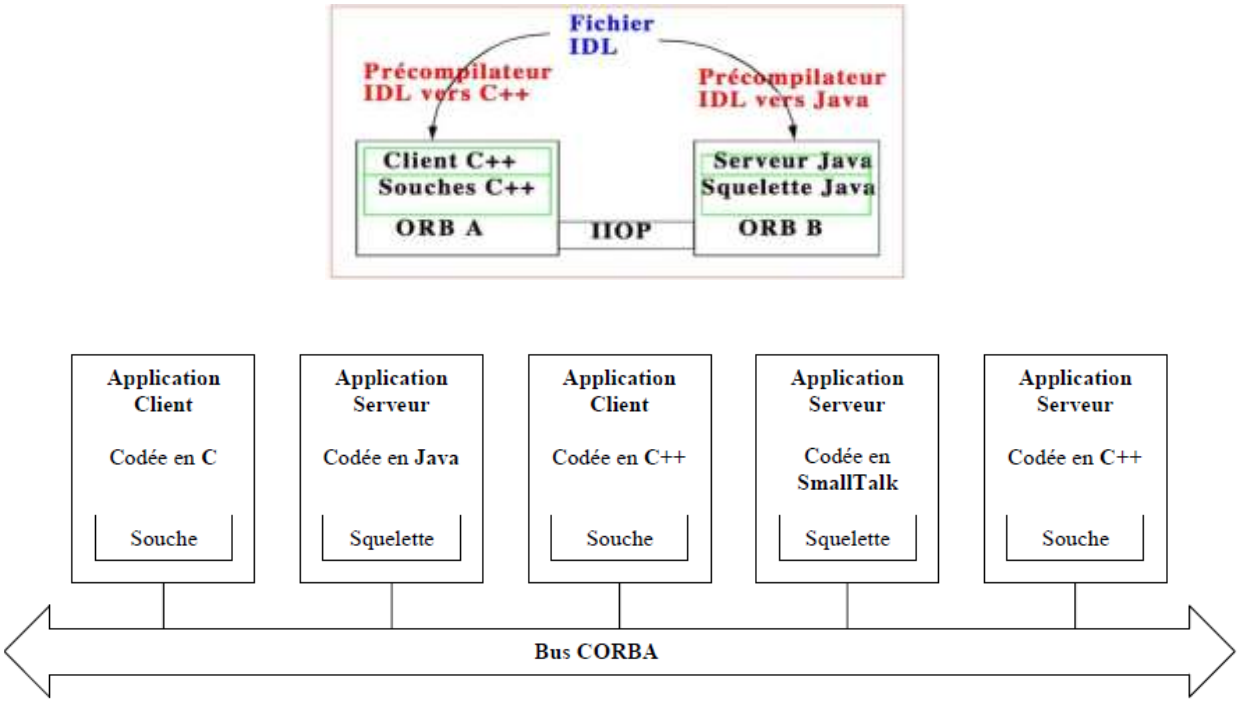


Figure 3.8- Hétérogénéité avec CORBA

Le standard IDL est défini par l'OMG et utilisé notamment dans le cadre d'applications ORB telles que CORBA. Le fichier IDL représente un contrat entre le serveur et ses clients : le serveur et ses clients doivent utiliser le même fichier de contrat pour générer leur code. IDL est un langage voué à la définition de l'interface de composants logiciels, laquelle permet de faire communiquer des modules

implémentés dans des langages différents, ou déployés à travers un réseau sur des systèmes hétérogènes (Windows, Linux, Mac OS X, VMS, etc.) dans la perspective d'architecture distribuée.

Exemple d'IDL:

IDL

Code IDL Sélectionnez

```
module fr {
  module ekinci {
    interface CalculationService {
      long factorial(in long num);
    };
  };
};
```

Extrait du code Java généré à partir de l'IDL

Code Java Sélectionnez

```
package fr.ekinci;

public interface CalculationServiceOperations {
  int factorial(int num);
}
```

Code Java Sélectionnez

```
package fr.ekinci;

public interface CalculationService
   extends CalculationServiceOperations,
           org.omg.CORBA.Object,
           org.omg.CORBA.portable.IDLEntity
{ }
```

Figure 3.9- Exemple d'IDL

5 Compilation du fichier IDL

La compilation du IDL générée les fichiers Java suivants :

- « Nom-Interface» : Interface de l'objet
- « Nom-Interface»Operations : Interface contenant les méthodes distantes.
- « Nom-Interface »POA : Skleton de l'objet distant coté serveur.
- _« Nom-Interface »Stub : Stub du client.
- « Nom-Interface »Helper :Contient des méthodes de manipulation/conversion des objet CORBA au java.
- « Nom-Interface»Holder : Implémente le passage de paramètres.

Example :

Voire figure 10 où :

- `_HorlogeStub.java` Un fichier souche (pour le client) qui est utilisé lors des appels de l'objet par le client.
- `_HorlogeImplBas.java` ou `HorlogePOA.java` Un fichier squelette (pour le serveur) qui se charge des requêtes envoyées par le client.
- `Horloge.java` Une Interface qui définit les services de l'objet Horloge.
- `HorlogeHelper.java` Une classe qui contient notamment la fonction narrow qui permet de convertir un objet en référence.
- `HorlogeHolder.java` Une classe utilisée lors de passage de paramètres d'appel de l'objet Horloge.
- Comme on programme en Java, l'implantation de l'objet est une classe (que nous appelons `HorlogeImpl`) qui implante la classe (que nous avons appelée `_HorlogeImplBase`) correspondant au squelette obtenu à l'étape précédente.

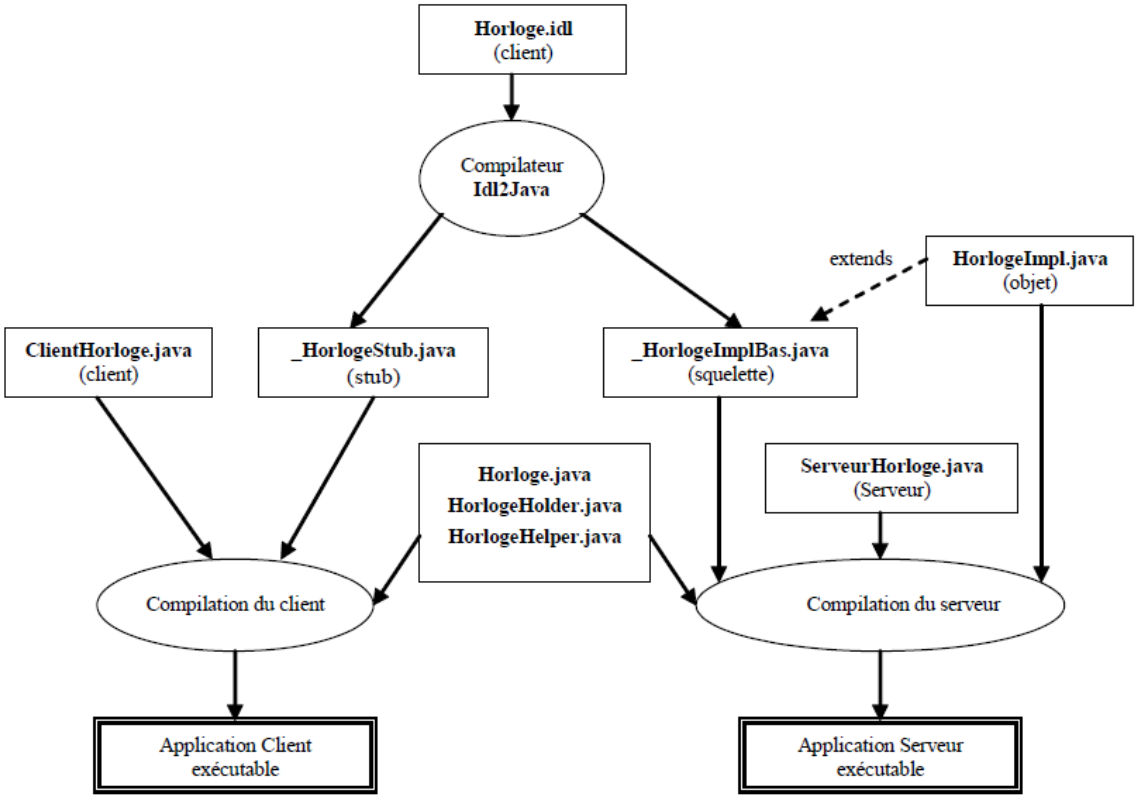
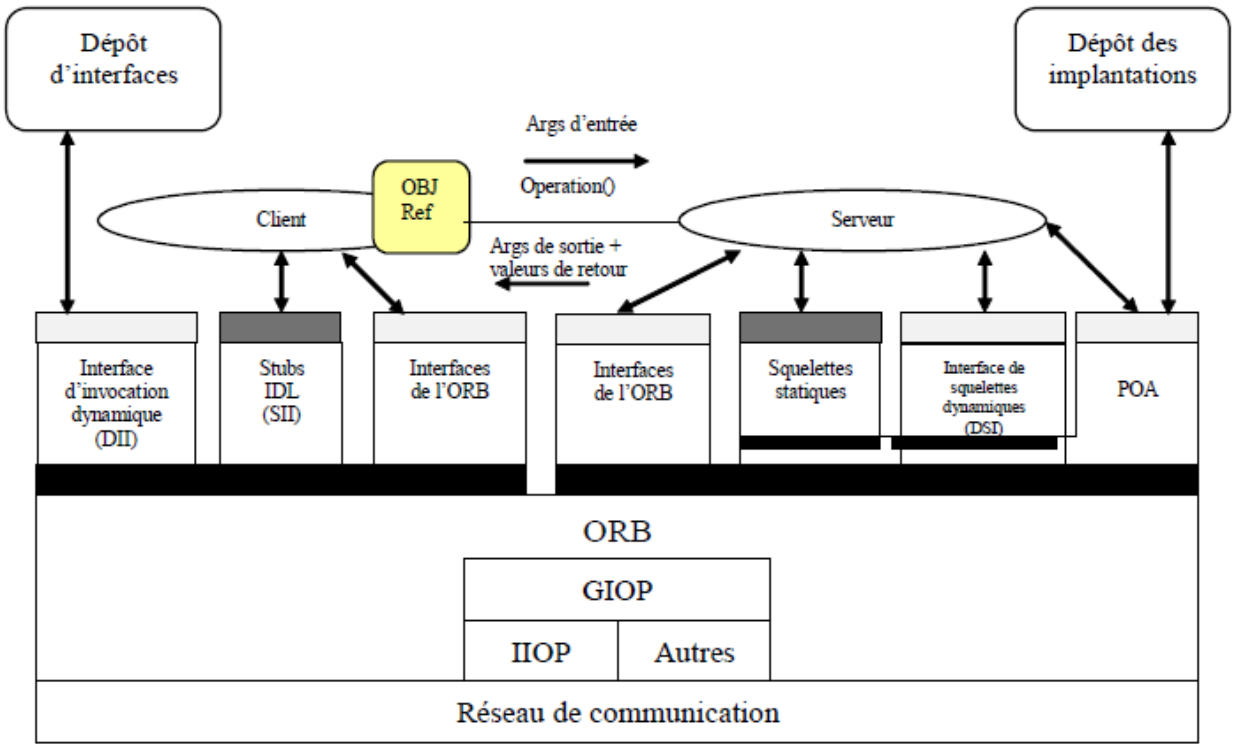


Figure 3.10- Fichiers générés pour l'application Horloge

6 Structure de CORBA



3.11 Structure de CORBA

Composant CORBA	Définition
Servant	Le Servant est une instance de la classe d'implémentation du service (**côté serveur**).
Objet Proxy	Un objet proxy est un objet qui est le représentant du servant (**côté client**).
IOR	IOR (*Interoperable Object Reference*) est une référence créée par le serveur, cela permet au client de localiser le servant.
POA	POA (*Portable Object Adapter*) c'est le composant qui s'occupe de créer les objets, de maintenir les associations entre objets, l'implantations et l'activation des objets, .gestion des IOR.
AOM	AOM (*Active Object Map*) est une table qui enregistre les couples IOR/Servant. L'AOM est géré par le POA.
ORB	Le bus logiciel CORBA (ORB) c'est le noyau de CORBA pour le transport des requêtes et des réponses.. Il Intègre, au minimum, les protocoles d'interopérabilité GIOP et
Stubs IDL et Squelettes statiques	Les stubs et les squelettes sont des interfaces d'invocation statique ; elles sont définies au moment de la compilation du script IDL. Les stubs, du côté du client, représentent le point d'accès aux opérations des objets définies par l'IDL. Du côté du serveur, une fois la compilation du script IDL faite, les méthodes spécifiées dans les interfaces IDL doivent être implémentées. L'implémentation d'objet va créer les routines, qui en accord avec l'interface ORB, seront appelées à travers les squelettes.

Interfaces d'invocation dynamique	Permettent de construire une requête dont la signature n'est pas connue jusqu'au moment d'exécution. Elles permettent de créer et d'invoquer les requêtes de façon dynamique, au lieu d'appeler un stub statique.
Squelettes dynamiques	Elles permettent d'accepter des invocations. Au lieu d'utiliser un squelette statique, l'implémentation de l'objet est atteinte par une interface qui fournit l'accès au nom de l'opération et les paramètres de façon dynamique. L'implémentation de l'objet doit fournir à l'ORB la description de tous les paramètres, et l'ORB doit fournir les valeurs de tous les paramètres d'entrée à utiliser pendant l'exécution
Dépôt d'interfaces	Contient la description des interfaces IDL accessibles aux applications à l'exécution.
Dépôt d'implantations	Contient la description les implantations d'objet.
Implantation de l'objet	Implantation de l'objet : entité codant l'objet CORBA à un instant donné et gérant un état de l'objet. Au cours du temps, un même objet peut avoir plusieurs implantations
Interopérabilité	L'interopérabilité représente la possibilité pour deux composants (des ORB dans le cas de CORBA) : • Développés avec des langages différents (C++, Java, etc.) • Utilisant des implémentations différentes (MICO, TAO, omniORB, etc.) • Exécutés dans des environnements différents (Windows, Linux, Mac, etc.)

7 Avantages

✓ Communication entre différents langages et interopérabilité complète.

✓ Obtenir un code très compact et efficace.

✓ Moins besoin de bibliothèques.

✓ Proposition des profils pour le temps réel et l'embarqué.

✓ Fiabilité (redondance).

✓ Intégration aux systèmes existants.

✓ Flexibilité du développement.

8 Inconvénients

- ✗ Complexité de développement.
- ✗ Problème de portabilité.
- ✗ Problème de maintenance
- ✗ Difficulté de mise en œuvre des Applications.
- ✗ Coût de développement trop cher.

9 Synthèse

	Multi langages	**Multi Systèmes d'exploitation**
RMI	✗	✓
DCOM	✓	✗
CORBA	✓	✓