

Introduction à CORBA

CORBA

- CORBA 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.

Architecture client/serveur traditionnelle

- L'application cliente (interface homme/machine et traitements) est située sur le poste client.
- Le lien entre le client et le serveur (serveur de BD par exemple) est direct.
- L'application cliente accède aux données de la base via des requêtes SQL.
- Peu de traitements sont localisés au niveau du serveur.

Architecture client/serveur à trois niveaux

- Le client n'accède jamais directement au serveur. Il émet des requêtes à un serveur d'application qui exécute les traitements et transmet les requêtes au serveur de données. On parle d'architecture à 3 niveaux (ou tiers) :
 - client : où est placée la couche présentation (interface homme/machine)
 - serveur d'application : on y trouve l'ensemble des traitements applicatifs séparés des données physiques
 - serveur de bases de données : stockage des données
- Cette architecture ne repose pas sur le concept objet. Les appels distants sont transparents sous forme RPC.

Architecture distribuée

- Dans cette architecture l'ensemble des dialogues entre machines est pris en charge par le "middleware".
- Il prend en charge tous les problèmes de communication liés à l'intégration des différentes plates-formes utilisées :
 - langages distincts
 - machines distantes
 - environnements hétérogènes

Les avantages de Corba

- **Interopérabilité** : Les objets Corba communiquent par l'intermédiaire du protocole IIOP (Internet Inter-ORB Protocol). Ce protocole permet la communication entre entités quelconques supportant le protocole TCP/IP.
- **Intégration aux systèmes existants** : Tout code existant peut être encapsulé dans un objet Corba (wrapping). Des passerelles existent pour des standards de distribution d'objets du marché (DCOM, DCE).
- **Flexibilité du développement** : Les services rendus par les objets sont définis par leur interface qui tient lieu de contrat entre l'utilisateur et le fournisseur du service. Les parties définition et implantation d'un objet sont totalement dissociées.

Objets distribués de CORBA

- Localisation indifférente dans le réseau
- Sur des systèmes hétérogènes
- Implantés dans des langages différents

Exemple

- réutiliser un module COBOL situé sur un site central et afficher les données à partir d'une application C++
- permettre à une feuille de calcul excel d'obtenir des données à partir d'une base Oracle

Corba repose sur la notion de bus logiciel (Object Request Broker).

En Corba, tout est objet. On bénéficie donc des mécanismes d'héritage, polymorphisme et encapsulation.

Composants logiciels

- Accédés par n'importe quel client
- Via des invocations de méthodes
- Le client ne connaît que l'interface publiée par le serveur
- L'interface est spécifiée dans un langage standard, l'IDL

Le modèle objet client-serveur CORBA

- Il est fondé sur une entité virtuelle, l'objet CORBA, gérée par le bus CORBA.
- Chaque application peut exporter ses services sous la forme d'objets CORBA.

Coopération client-serveur

- Le client détient une référence sur un objet CORBA qui permet de le localiser sur le bus.
- Le client dispose de l'interface de l'objet CORBA (type abstrait de l'objet CORBA) qui définit ses opérations et ses attributs (exprimés dans le langage IDL).
- Le client réalise une requête (invoque) une opération sur l'objet CORBA.
- Le bus CORBA achemine cette requête vers l'objet CORBA tout en masquant les problèmes d'hétérogénéité liés aux langages, systèmes d'exploitation, machines, réseaux.
- L'objet CORBA associe l'objet CORBA à un objet d'implantation
- Le serveur détient l'objet d'implantation qui code l'objet CORBA (cette implantation pouvant évoluer au cours du temps) et gère son état temporaire.

Le contrat IDL

- Le contrat IDL rend transparent aux fournisseurs et clients l'infrastructure logicielle et matérielle.
- Il met client et fournisseur en relation à travers un bus CORBA.

Compilation d'un contrat IDL

- La compilation d'un contrat IDL crée une souche (talon) IDL (ou SII, interface d'invocation statique) dans l'environnement de programmation du client et une souche (squelette) IDL (ou SSI, interface de squelette statique) dans l'environnement de programmation du fournisseur.
- Le client invoque localement la souche pour accéder à l'objet. La souche construit alors la requête qui est ensuite transportée par le bus logiciel pour être délivrée au squelette IDL (coté serveur donc) qui la délègue à l'objet.

IDL

- C'est un langage de définition des services proposés par un objet
- Une interface comprend un ensemble d'opérations nommées et des paramètres sur ces opérations
- L'IDL est le moyen par lequel un objet particulier indique à ses clients potentiels les opérations disponibles et la façon des les invoquer
- C'est un langage neutre et totalement déclaratif
- A partir des définitions d'IDL, il est possible de mettre en correspondance (mapping) les objets CORBA dans un langage de programmation particulier ou avec des objets systèmes.
- Il définit les types d'objet par spécification des interfaces

Structure syntaxique d'un module IDL

`<module> /* un contexte ou espace de nommage */`

`<déclaration de types>`

`<déclaration de constantes>`

`<déclaration d'exceptions>`

`<interface> /* une classe */`

`<déclaration de types>`

`<déclaration de constantes>`

`<déclaration d'exceptions>`

`<déclaration d'attributs> /* variables */`

`<déclaration d'opérations> /* méthodes */`

`<module>`

```

/* définition du contexte mes_animaux */
module mes-animaux;
{
    /* définition de l'interface Chien */
    interface Chien:Animal_de_compagnie,Animal /* héritage */
    {
        attribute long age;
        exception non_interesse{string explication};

        void aboie(in short combien_de_temps)
                                raises (non_interesse);
        void assis(in string ou)      raises (non_interesse);
        void grogne(in short apres_qui) raises (non_interesse);
    }

    /* interface Chat */
    interface Chat:Animal
    {
        void mange();
        void viens();
    }
}

```

Syntaxe IDL

La syntaxe IDL est très proche de la syntaxe C++.

Les **interfaces** définissent les classes de l'environnement. Elles peuvent hériter la structure d'une ou plusieurs autres interfaces.

Les **attributs** définissent les variables publiques. Les clients peuvent lire ou écrire ces attributs. Cependant, ils peuvent être marqués en lecture seule :

```
readonly attribute float position;
```

Les **opérations** sont les méthodes de l'objet. elles retournent une valeur (éventuellement vide `void`) et transmettent une liste de paramètres : `in` (en entrée), `out` (en sortie), `inout` (en entrée/sortie). Elles peuvent lever une exception.

Mappings

- Un mapping est une traduction des éléments fournis par l'IDL en éléments d'un langage de programmation.
- Les principaux mappings disponibles sont : C, C++, Java, Smalltalk, CLOS, Python, Ada95, Cobol Objet.
- Les compilateurs IDL ne traduisent que des squelettes. Les objets clients utilisent ces squelettes pour déterminer les opérations légales qu'ils peuvent invoquer sur un serveur. Les objets serveur fournissent une implantation pour ces squelettes.

Référentiels d'interface

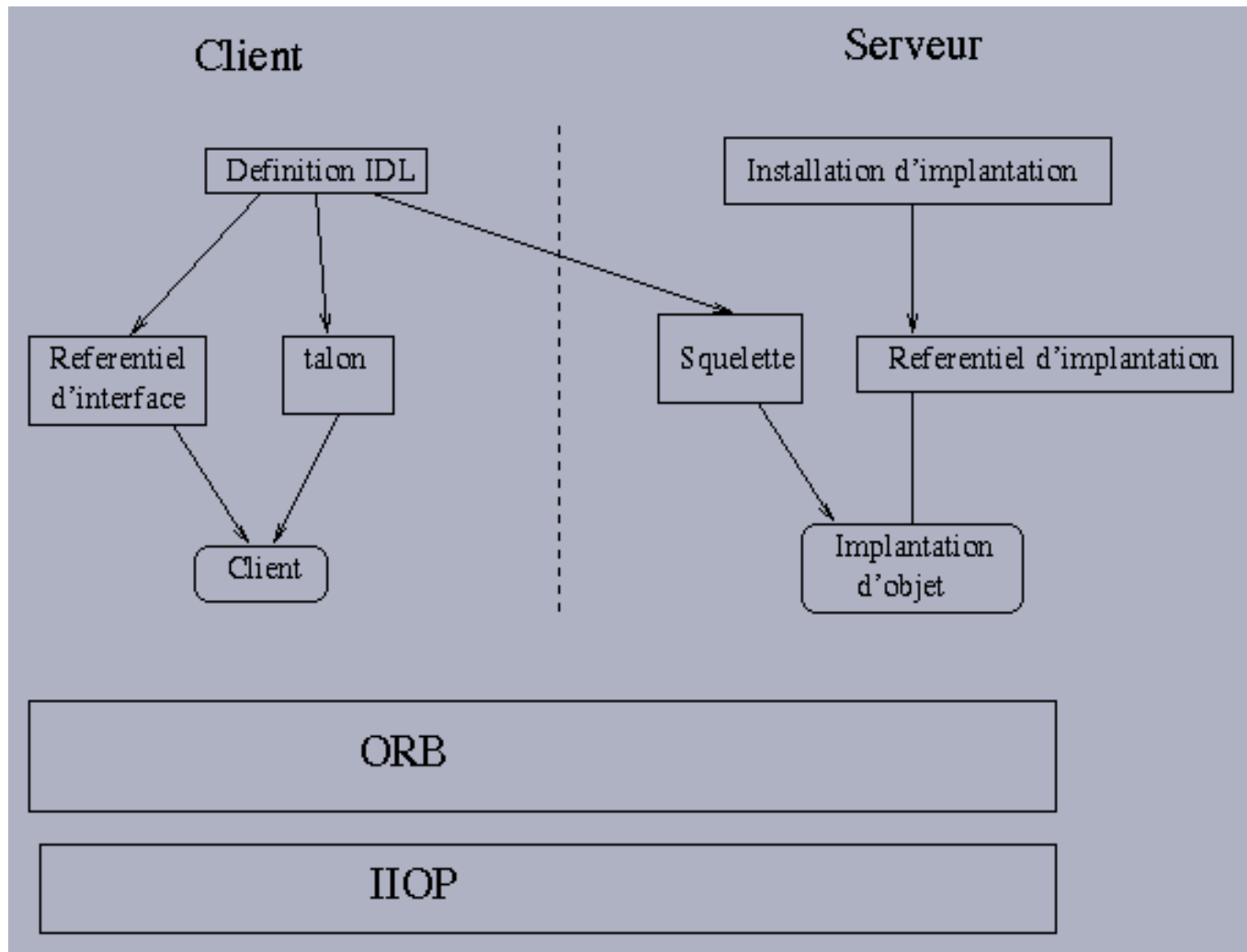
- C'est une base de données en ligne de définitions d'objets, d'interfaces, de modules. Les définitions d'objets sont fournies par les compilateurs d'IDL ou par des fonctions d'écriture du référentiel.
- Les référentiels d'interfaces peuvent se fédérer et coopérer à travers les ORBs.

ORB ou Bus CORBA

- C'est un bus à objets.
- Il permet aux objets de faire des requêtes vers des objets locaux ou distants et de recevoir des réponses en rendant transparents les mécanismes de communication à travers le réseau.

Quelques propriétés des ORBs

- Les invocations peuvent être statiques ou dynamiques
- Permettent des liaisons ("mappings") avec des langages de haut niveau. Le client n'a pas à se soucier du langage d'implantation des objets sur le serveur. Il peut donc utiliser son propre langage.
- Peuvent fonctionner seuls ou s'interconnecter avec d'autres ORBs
- Gèrent la sécurité
- Gèrent les transactions



Coté client

- Le client connaît l'interface d'un objet spécifique (il dispose de l'IDL).
- Le client possède une représentation de l'objet distant appelée **talon** ("stub") générée par l'IDL.
- Le **talon** transmet la requête à l'objet distant via le bus logiciel (**ORB**). Il s'agit d'une représentation de l'objet distant responsable de la transmission d'une invocation de méthode du talon vers l'objet distant. Il ne s'agit pas d'une copie de l'objet distant.
- A travers le talon le client voit l'interface de l'objet sans savoir où est l'objet et sans connaître le langage de programmation utilisé pour son implantation.

En pratique

le client :

- crée une instance locale d'ORB
- récupère la référence du serveur de noms
- récupère la référence de l'objet
- invoque la méthode sur l'objet distant

Coté serveur

- L'ORB utilise un **squelette** de code pour traduire l'invocation distante en un appel de méthode sur l'objet local.
- Le **squelette** traduit l'appel et les paramètres dans le format de l'implantation spécifique et appelle la méthode invoquée.
- Au retour de la méthode, les résultats (ou erreurs) sont traduits par le squelette et renvoyés au client via l'ORB.

Communication entre ORB, protocoles réseaux

- Tout bus à la norme CORBA 2.0 doit fournir les protocoles GIOP et IIOP.

GIOP est un protocole générique. Il fournit :

- une représentation commune des données : CDR (Common Data Representation)
- un format de référence d'objet interopérable : IOR (Interoperable Object Reference)
- un ensemble de messages de transport de requêtes aux objets (request, reply, ...)

- Les ORBs partagent un protocole commun **IIOP** ((Internet Inter ORB Protocol), implantation de GIOP sur TCP/IP et donc sur l'internet. Ce protocole définit comment les ORBs (répondant à la spécification CORBA) se transmettent de l'information.

L'Object Management Architecture (OMA)

Elle identifie les différents composants d'une architecture distribuée :

- L'ORB
- Les services objets (CORBAServices)
- Les utilitaires communs (CORBAFacilities)
- Les interfaces de domaines
- Les objets applicatifs

L'ORB

- Il se charge de localiser les objets, de les charger en mémoire, de transmettre au serveur les demandes du client.
- Il assure les opérations de maintenance, cycle de vie, gestion des erreurs.

Les services objets (CORBAServices)

Il s'agit d'un ensemble de services spécifiés par l'OMG que l'on peut retrouver dans la plupart des applications :

Service de nommage : tout objet est enregistré dans un catalogue sous la forme d'une couple (nom symbolique, référence). Tout utilisateur peut donc obtenir la référence d'un objet connaissant son nom symbolique.

Service d'évènements : produit des évènements asynchrones au niveau des serveurs

Service de cycle de vie : fournit un ensemble d'opérations pour créer, supprimer, déplacer, copier des objets

Service de persistance : stockage et restauration d'objets

Service d'externalisation : permet de placer des objets dans un flux

Service transactionnel : garantit l'intégrité des données lors d'un traitement

Service de sécurité : authentification des clients, cryptage des données, gestion des certificats

Service de courtage : rechercher un objet dont on ne connaît pas le nom à partir de critères.

etc ...

Les utilitaires communs (CORBAFacilities)

Gestionnaires d'impression, outils de gestion documentaire, diverses boîtes à outils.

Les interfaces de domaines

Il s'agit d'un ensemble d'interfaces d'objets utilitaires liés à un domaine d'activité spécifique.

Les objets applicatifs

Il s'agit des objets créés par les utilisateurs qui vont s'insérer comme un sous-ensemble de l'architecture OMA.

Le marché CORBA actuel

Chaque éditeur est libre de répondre à tout ou partie de la norme. Tous les services ne sont pas implantés .

INPRISE	Visibroker	nombreux services
IONA	Orbix	passerelle DCOM
IBM	SOMObjects	OS2 et Windows seulement
Objects Oriented Concepts	ORBacus	code source

Développement d'une application à objets distribués avec JavaIDL

- L'ORB fourni par JavaIDL ne supporte que la recherche d'objets par leur nom (service de nommage).
Ces différentes étapes sont décrites concrètement à travers l'exemple HelloWorld qui établit une relation client-serveur simple :
 1. Le client (applet ou appli) invoque la méthode `sayHello` du serveur `HelloServer`.
 2. L'ORB transfère l'invocation à l'objet distant correspondant à cette interface (dans le référentiel d'interfaces).
 3. La méthode `sayHello` du servant s'exécute et retourne la chaîne de caractères `"Hello"`.
 4. L'ORB transfère la chaîne vers le client
 5. Le client l'imprime
- L'invocation de la méthode est statique.

Travail demandé

- Etudier et faire tourner l'exemple HelloWorld
- Créer un objet calculette et un programme client qui utilise celle-ci à distance