Le standard CORBA

Janvier 1998

Virginie Amar
Centre Scientifique et Technique du Bâtiment (CSTB)
amar@cstb.fr

Sommaire

Introduction

Le problème : Intégration des applications

- ☞ Pas de consensus sur les langages de programmation
- ☞ Pas de consensus sur les plate-formes de développement
- ☞ Pas de consensus sur les systèmes d'exploitation
- ☞ Pas de consensus sur les protocoles réseau
- ☞ Pas de consensus sur les formats des données manipulées par les applications

➪ Consensus pour l'interopérabilité

1- OMG

Object Management Group (OMG)

- consortium international créé en 1989
- but non lucratif
- regroupement de plus de 800 membres
  - constructeurs (SUN, HP, DEC, IBM, ...)
  - environnements systèmes (Microsoft, OSF, Novell, ...)
  - outils et langages (Iona, Object Design, Borland, ...)
  - produits et BD (Lotus, Oracle, Informix, O2, ...)
  - industriels (Boeing, Alcatel, Thomson, ...)

Mission et objectif

Mission

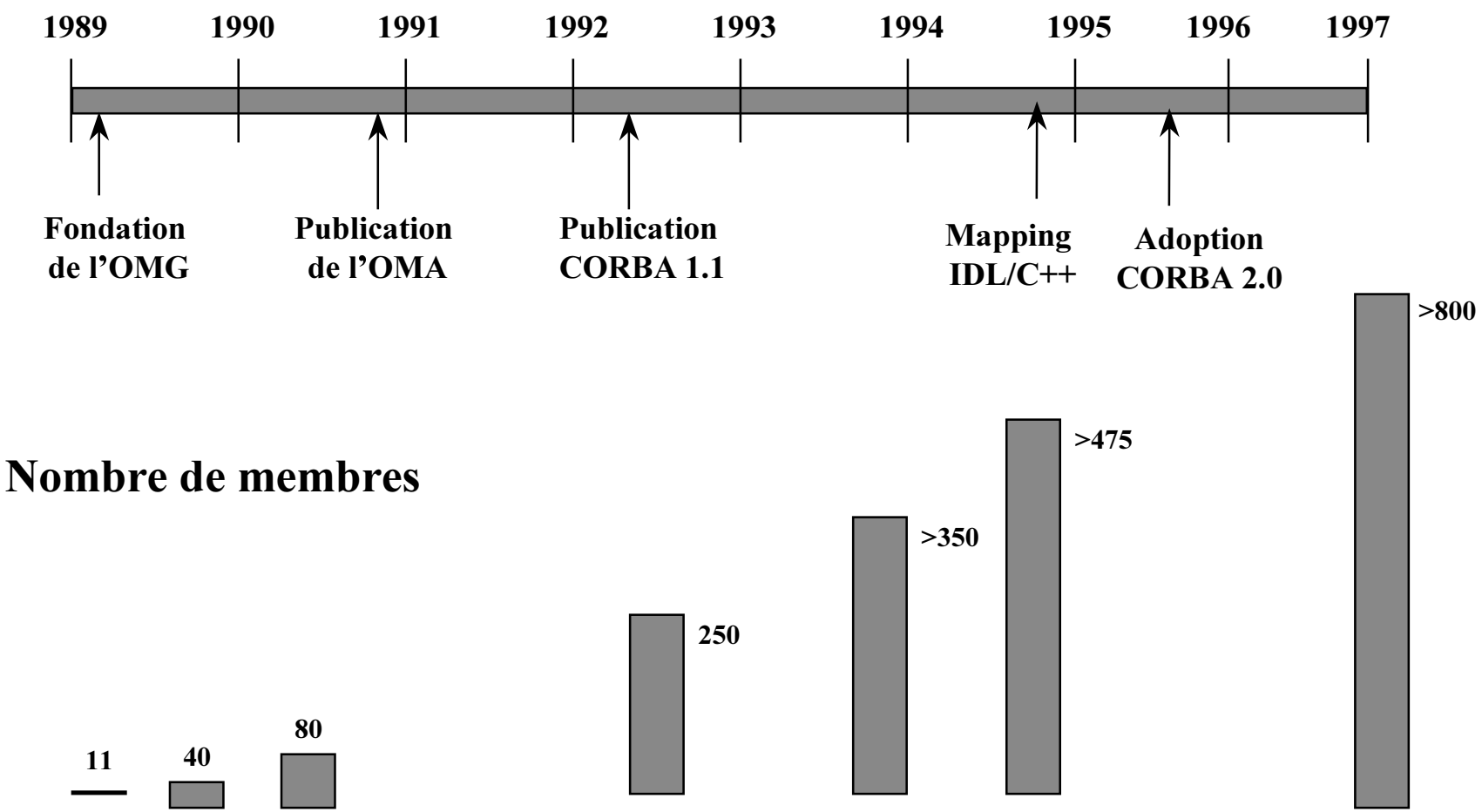
- ☞ Promouvoir la technologie orientée objet dans les systèmes informatiques distribués
- ☞ Fournir une architecture de base pour l'intégration d'applications distribuées tout en garantissant la réutilisabilité, l'intéropérabilité et la portabilité.

Objectif

Favoriser l'intéropérabilité et la portabilité d'applications réparties à travers :

- une terminologie unique dans le domaine de l'objet;
- un model de référence commun;
- des interfaces et des protocoles communes.

Historique



Organisation et procédures

Comité technique

- détermine la direction de l'architecture et des normes;
- émet des RFI (Request For Information) pour vérifier la disponibilité des technologies;
- émet des RFP (Request For Proposal) pour obtenir des propositions de spécifications;
- évalue les propositions et propose une sélection;
- fait des recommandations au comité directeur sur les choix des technologies.

Comité directeur

- prend les décisions finales pour l'adoption des spécifications.

2- Object Management Architecture

Object Management Architecture (OMA)

Description d'une architecture de Gestion Objet définissant les bases des futures spécifications incluant :

- une prise en compte des problèmes d'intégration dans des environnements distribués;
- les objectifs de l'OMG;
- **un modèle d'objets abstrait;**
- **l'architecture du modèle de référence;**
- un glossaire des termes utilisés.

Le modèle abstrait d'objets

- Modèle définissant la façon de décrire des objets distribués dans des environnements hétérogènes.
- Utilisé dans toutes les technologies conformes à l'OMG (ex : CORBA)
- Spécifie une sémantique commune définissant le comportement externe des objets d'une manière standard (indépendante des langages et des implémentations).

Définitions (1/2)

Client

Entité capable d'émettre des requêtes vers des objets qui fournissent des services.
Le client manipule des références vers des objets distants.

Référence Objet

Objet manipulé par le client pour invoquer des services sur un objet distant : objet implémentation.
Terme usité : proxy
Un proxy est un représentant local au client d'un objet distant.

Objet implémentation

Objet situé sur le serveur qui implémente le code des méthodes des opérations définies en IDL.

Définitions (2/2)

Requête

Emise par un client pour demander l'exécution d'une opération sur un objet cible.
La requête contient l'opération à exécuter, l'objet cible et les paramètres éventuels.

Interface

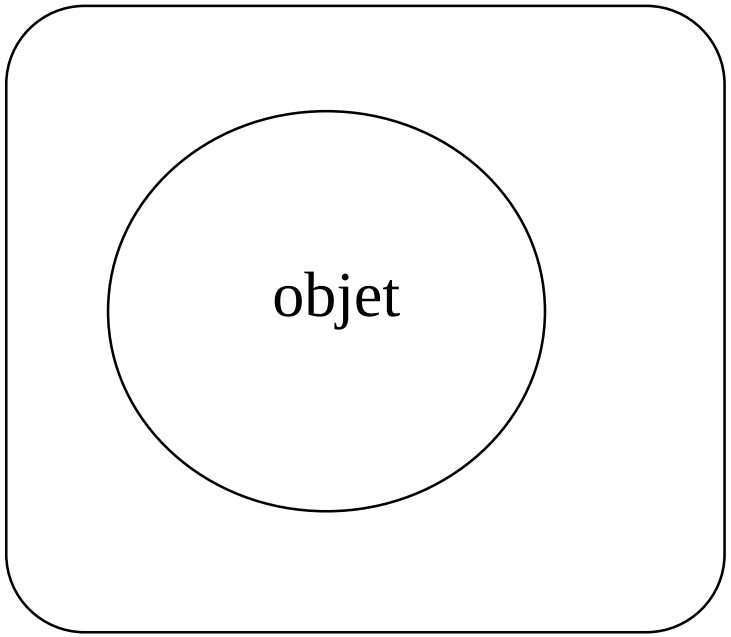
Description d'un ensemble d'opérations disponibles sur un objet.
Spécification des interfaces en IDL.

Opération

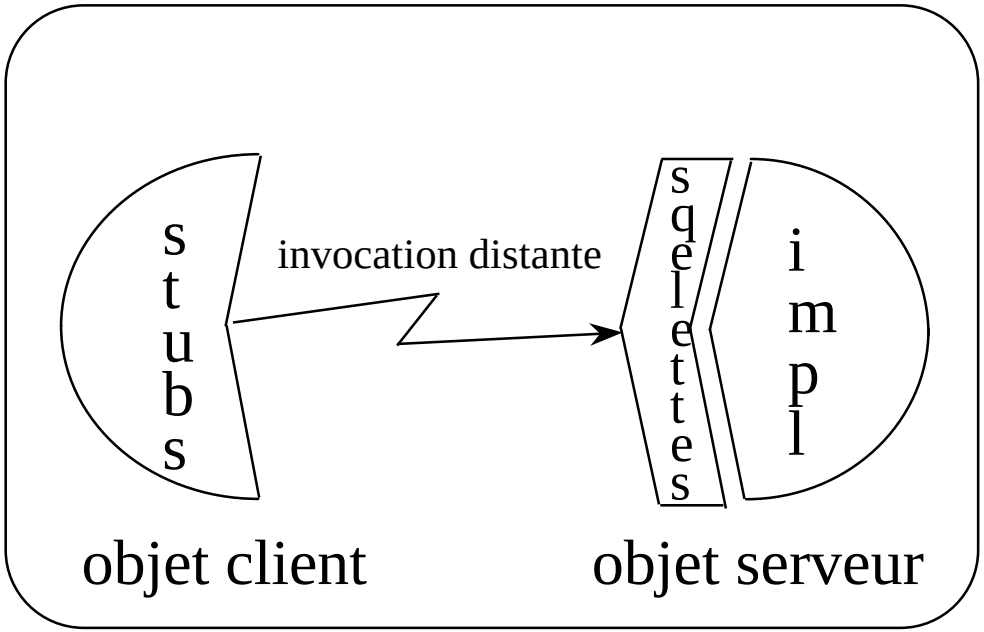
Entité identifiable caractérisée par une signature décrivant les paramètres de la requête et les valeurs de retour.

Concepts (1/2)

Vue d'un objet dans une application monolithique

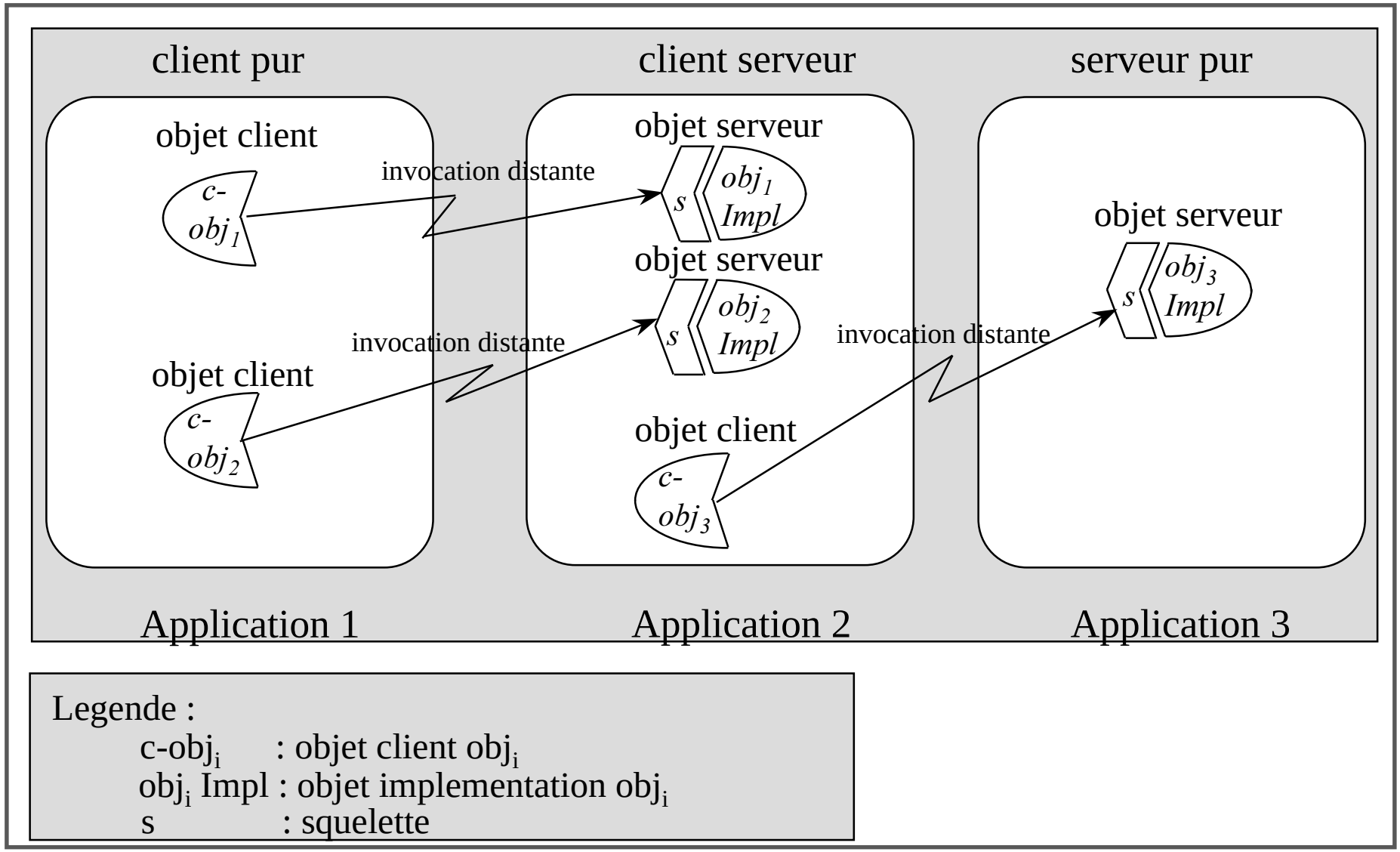


Vue d'un objet dans une application d'objets distribués

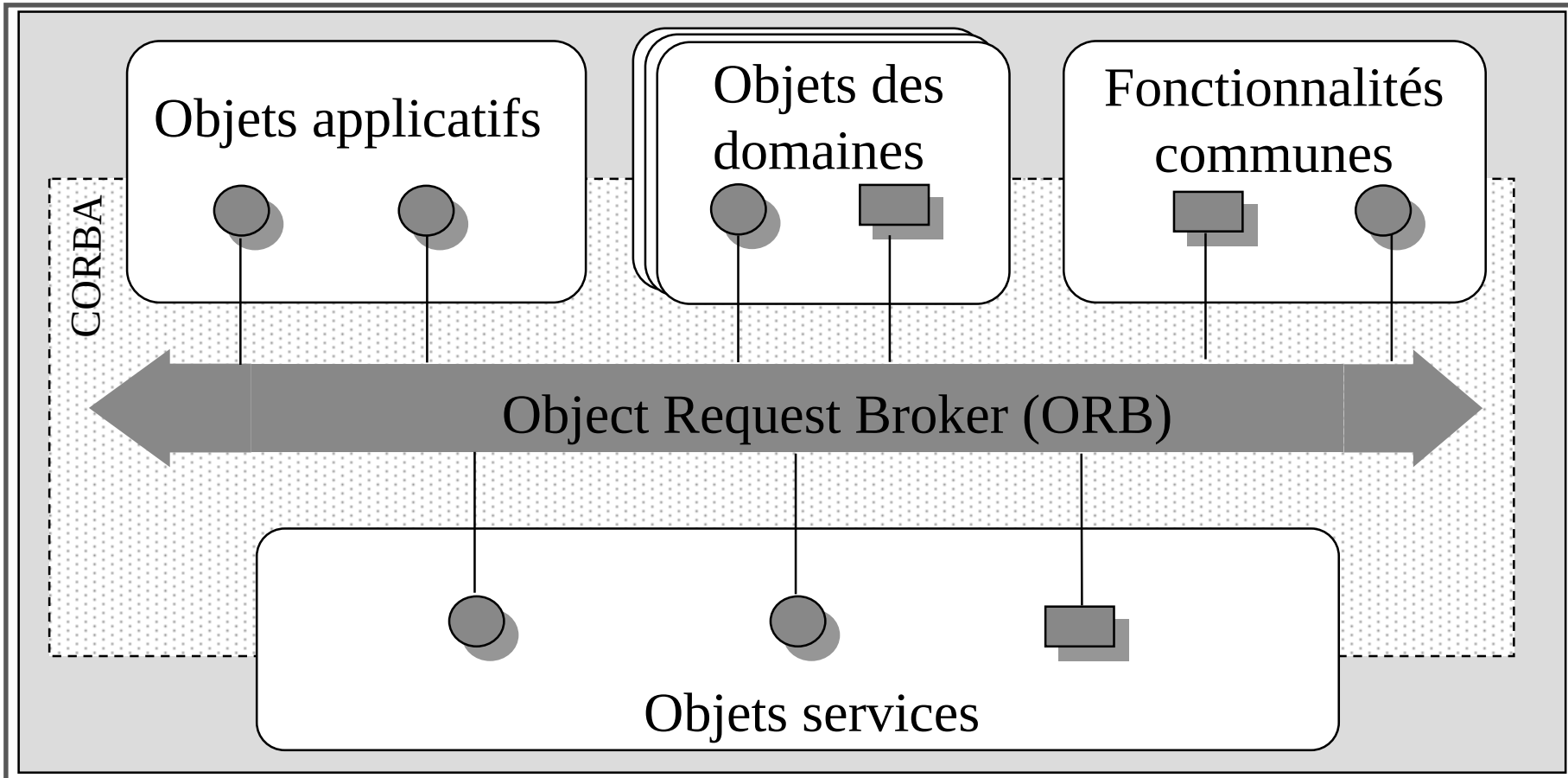


Legende
impl = objet implementation

Concepts (2/2)



Architecture du modèle de référence



Spécification des liens entre les objets

ORB (1/2)

ORB : Object Request Broker

Middleware qui gère les relations client/serveur entre les objets

Définition du concept de Middleware :
Courtier d'objets (en Français).

Ensemble des logiciels nécessaires pour permettre
et organiser la communication et l'échange de messages entre client et serveur.

ORB (2/2)

Composant central du standard CORBA qui gère :

- la location d'objet
- la désignation des objets
- l'empaquetage des paramètres (marshalling)
- le dépaquetage des paramètres (unmarshalling)
- l'invocation des méthodes
- la gestion des exceptions

Objets applicatifs

Les Objets applicatifs (CORBA Applications) :

- spécification d'interfaces IDL;
- définis par une application de l'utilisateur;
- hors du champ de standardisation de l'OMG;
- possibilité de standardisation pour des objets émergents.

Objets des domaines

Les Objets des domaines (CORBA Domains) :

- spécification d'interfaces IDL;
- standardisés par l'OMG;
- leurs fonctionnalités peuvent être étendues ou spécialisées par héritage;
- spécifiques à un domaine d'application (médical, financier, télécommunications...);
- interfaces "vertical-oriented ".

Fonctionnalités communes

Les Fonctionnalités communes (CORBA Facilities) :

- spécification d'interfaces IDL;
- leurs fonctionnalités peuvent être étendues ou spécialisées par héritage;
- ensemble de services de plus haut niveau fournissant des fonctionnalités utiles dans de nombreuses applications;
- indépendant des domaines d'application;
- interfaces «horizontally-oriented».

- Exemple : impression, aide en ligne, messages d'erreurs, etc.

Objets Services

Les Objets Services (CORBA Services) :

- spécification d'interfaces IDL;
- leurs fonctionnalités peuvent être étendues ou spécialisées par héritage;
- interfaces indépendantes des domaines d'application;
- interfaces «horizontally-oriented»;
- objectif : étendre les fonctions de l 'ORB.
- Spécification de nombreux services (cf. section 4).

3- Le langage IDL

Interface Definition Language

- langage de spécification d'interfaces, supportant l'héritage multiple;
- langage indépendant de tout langage de programmation ou compilateur;
- langage utilisé pour générer les stubs, les squelettes et pour définir les interfaces du Référentiel d'interface;
- la correspondance IDL langage de programmation est fournie pour les langages C, C++, Java, Smalltalk, Ada, Cobol.

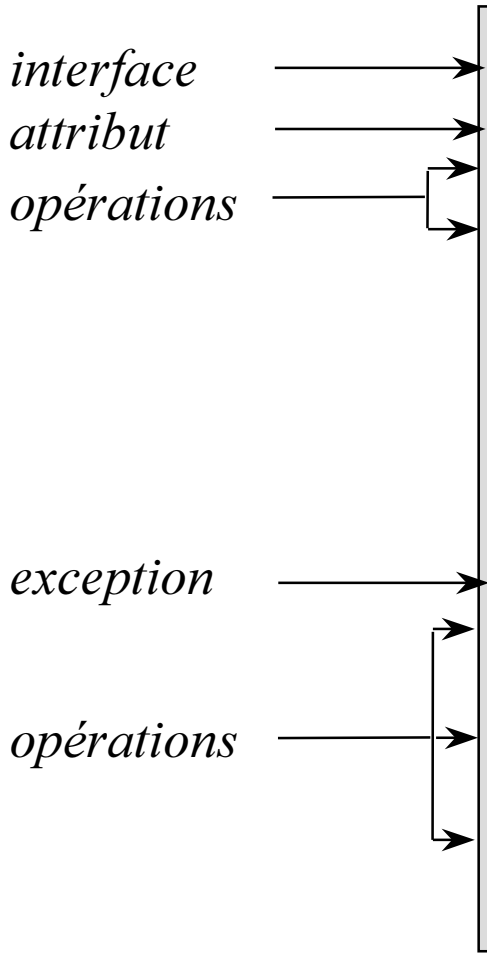
Interface IDL

- Une spécification IDL définit un ou plusieurs types, constantes, exceptions, interfaces, modules
- Un module permet de limiter la validité des identificateurs
- Interface : ensemble d'opérations et de types =>classe C++

Syntaxe :

Interface | [<héritage>] { <interface Body>};

Exemple




```
interface account {
   readonly attribute float balance;
   attribute string description;
   void makeLodgement (in float f);
   void makeWithdrawal (in float f);
};
interface currentAccount : account {
   readonly attribute float overdraftLimit;
};
interface bank {
   exception reject {string reason;};
   account newAccount (in string name)
            raises (reject);
   currentAccount newCurrentAccount (in string name, in float limit)
            raises (reject);
   void deleteAccount (in account a);
};
```

IDL vs C++

- Même règles lexicales que C++
- Grammaire IDL : sous ensemble de la grammaire ANSI C++ avec constructions supplémentaires
- Nouveaux mots clés :

attribute
interface
module
oneway
readonly
sequence
any

Types de base et autres types

Types de base
- short
- long
- unsigned short
- unsigned long
- float
- double
- char
- boolean
- octet
- any

Types construits
- struct
- union
- enum

Types génériques
- array
- sequence
- string

Attribut IDL

Définition d'attribut

```
interface account {
        readonly attribute float balance;
        attribute string description;
        ...
};
```

Equivaut à :

```
float _get_balance();
string _get_description();
void _set_description(in string s);
```

Operations (1/2)

- Paramètres nommés et associés à un mode
- Opérations bloquantes par défaut

```
void op1 (in long input,
          out long output,
          inout long both);


interface account;

interface bank {
  account newAccount (in string name);
  void deleteAccount (in account old);
};
```

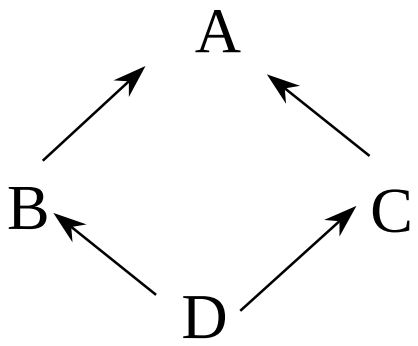
Opérations (2/2)

```
oneway void notify (in string message);
```

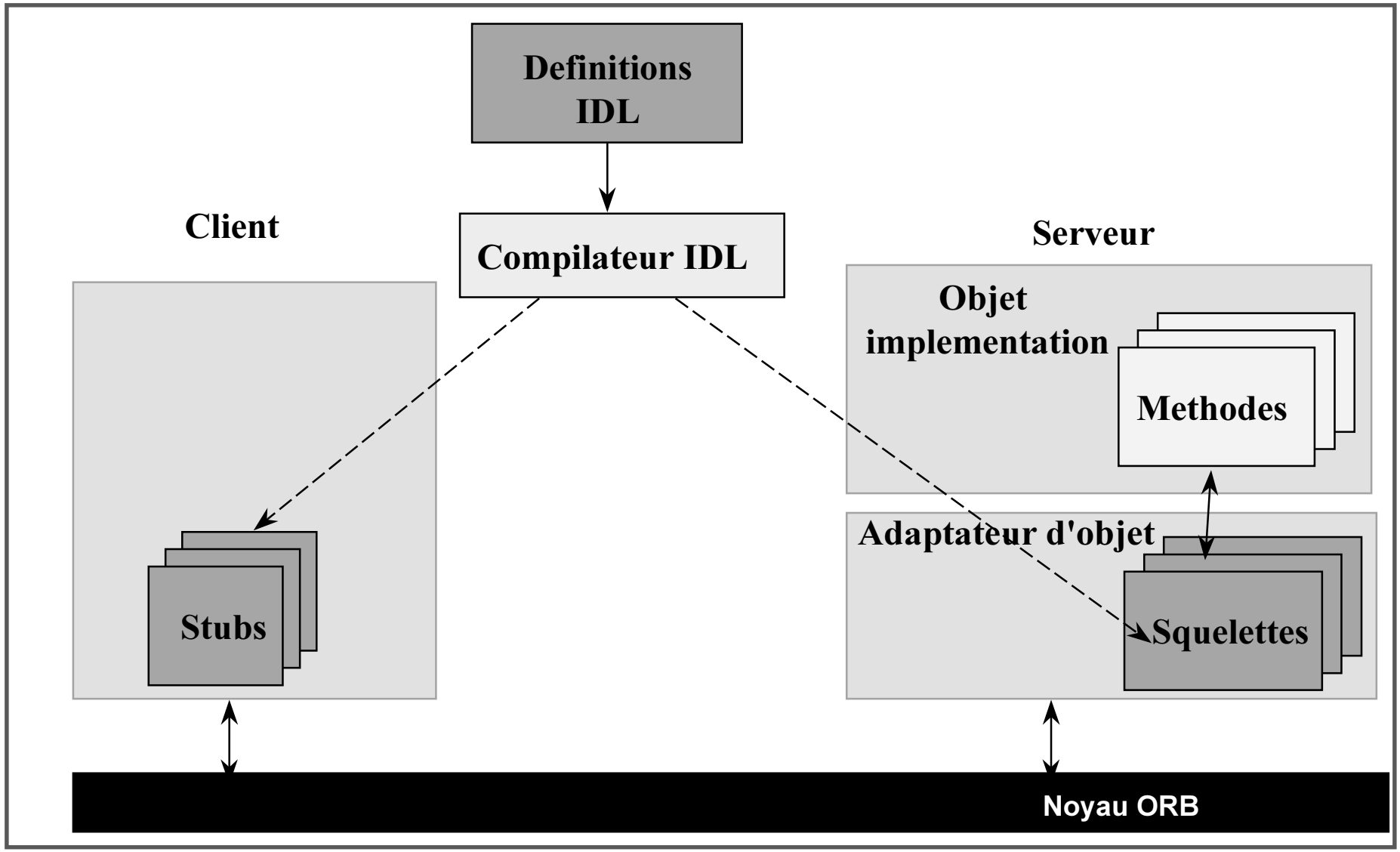
- ☞Opération non bloquante
- ☞ Pas de paramètre de type *out, inout* ou d'*exceptions*
- ☞ Valeur de retour : void

Héritage multiple

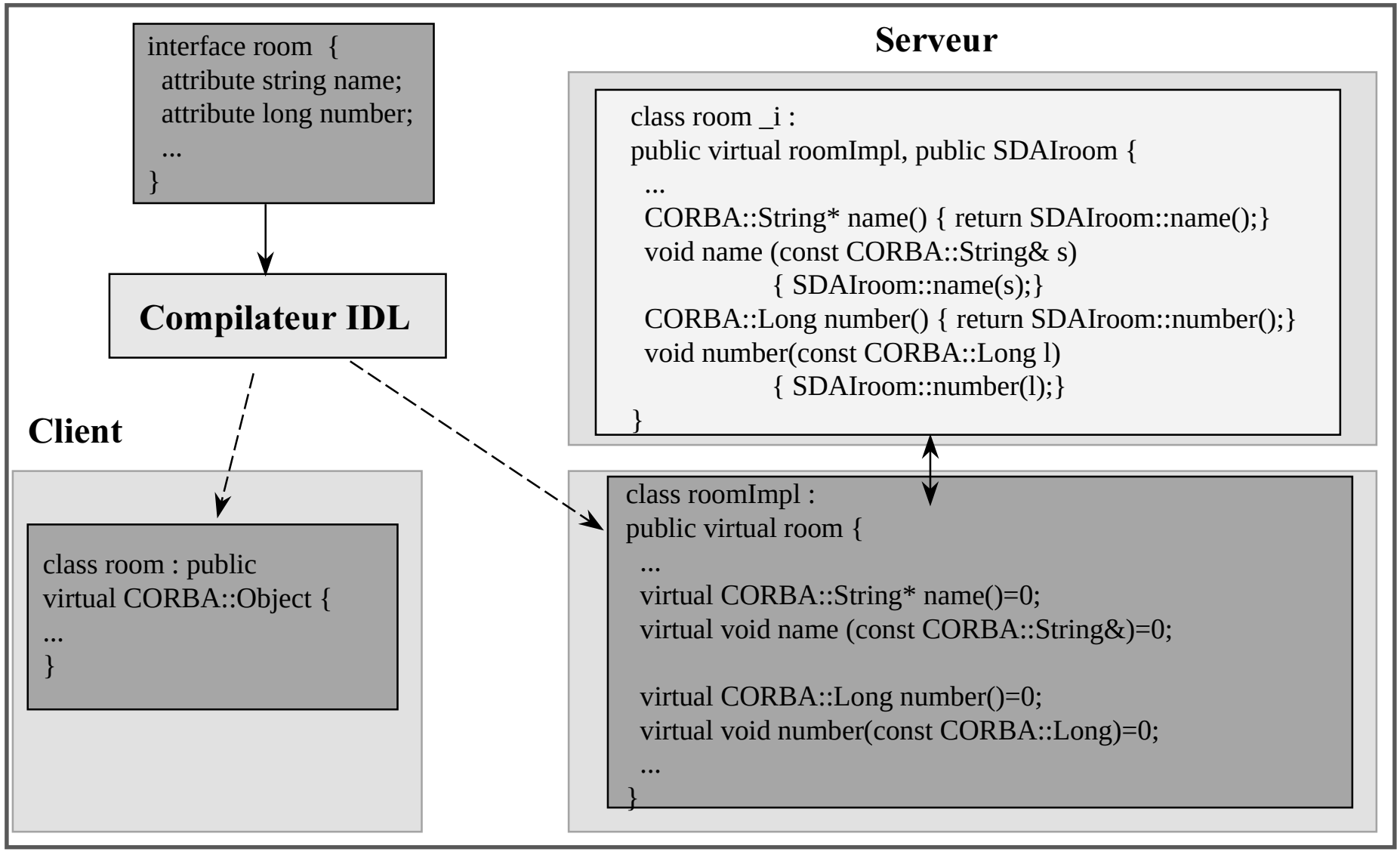
interface A { ... }
interface B : A { ... }
interface C : A { ... }
interface D : B, C { ...}



Développement



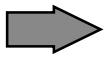
Mapping C++



Mapping C++ des modules

- traduit en namespace C++
- les modules CORBA contiennent des types prédéfinis des interfaces et des opérations

```
// IDL
module M
{
// définitions
};
```



```
// C++
namespace M
{
// définitions
};
```

Mapping C++ des références objet

Interface

- traduit en classes C++
- traduit en deux types C++
  - A_var

    Référence sur un objet (instance de classe A) avec gestion automatique de la mémoire.
  - A_ptr

    Similaire aux pointeurs C++.

```
interface A
{
  void op1 (in long param);
};
```

```
// C++-used
A_ptr aPtr;
A_var aVar;
aVar->op1(2);
aPtr->op1(5);
```

Mapping C++ des exceptions (1/2)

- Définition d'un ensemble d'exception système standardisées : UNKNOWN, BAD_PARAM, NO_MEMORY, etc ...;
- Possibilité de définir des exceptions propres à l'utilisateur;
- Exceptions propagées de l'objet implémentation vers le client;

Mapping C++ des exceptions (2/2)

Interface IDL

```
interface bank {
    exception reject {string reason;};
    account newAccount (in string name)
                                raises (reject);
...
}
```

Client

```
bank_var b;
...
try  {
   a = b->newAccount("Bill");
}
catch (const bank::reject& r) {
// actions
}
catch(...) {
...
}
```

Objet Implémentation

```
void bank_i::newaccount (const char* name,
                                CORBA::Environnement&) {
 ...
throw bank::reject ("Impossible to create a new account");
};
```

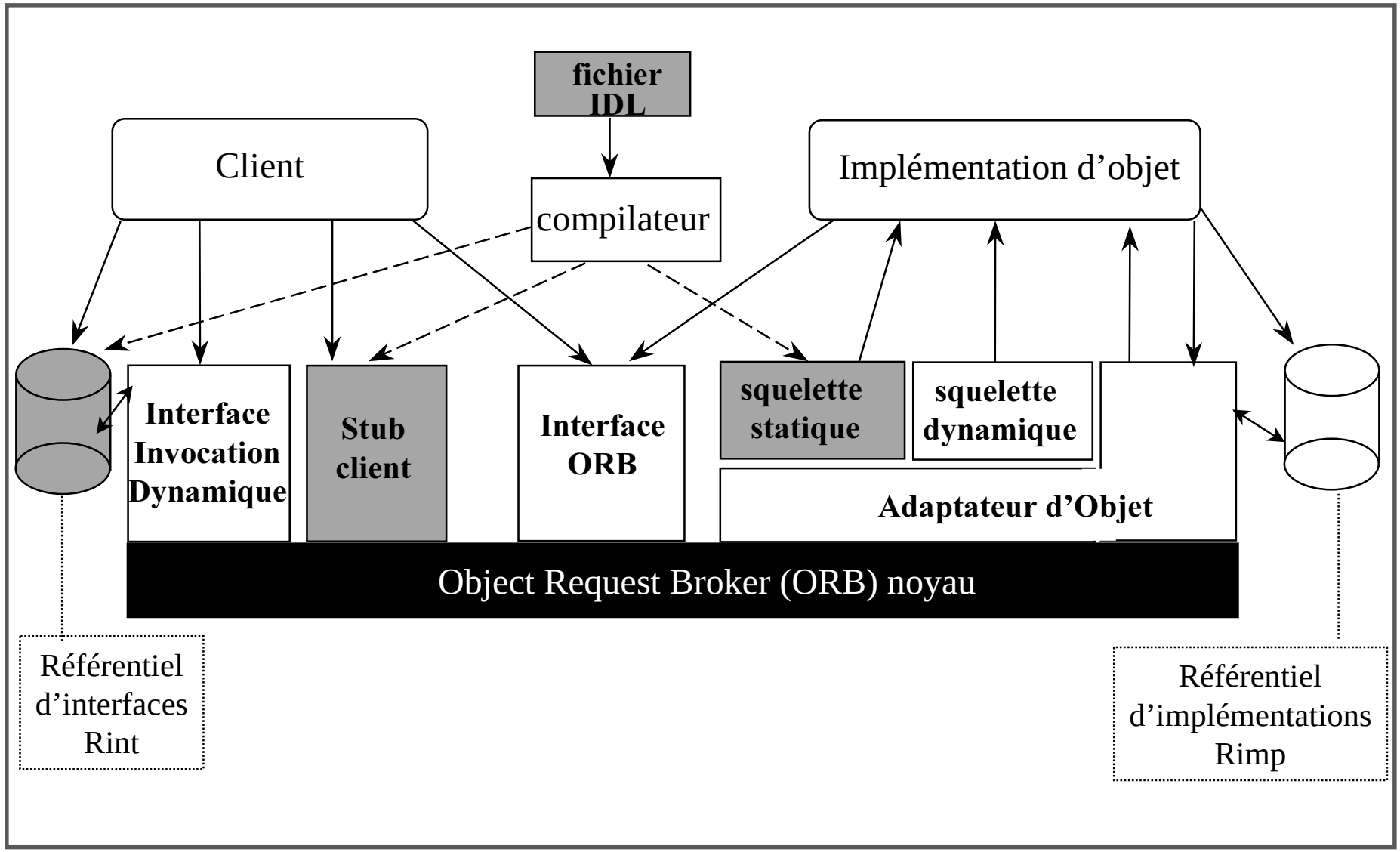
4- CORBA

Common Object Request Broker Architecture

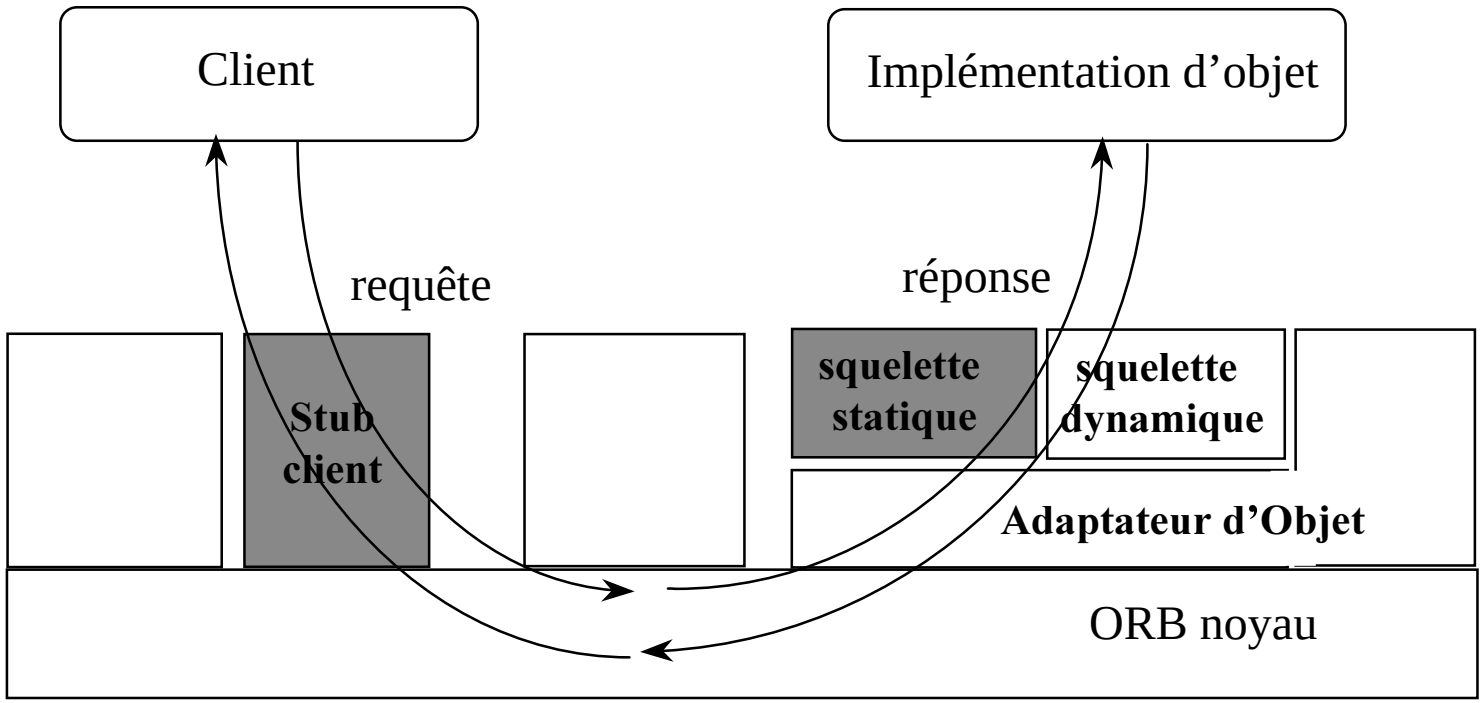
Architecture permettant de développer des applications distribuées :

- standardisées
- dans des environnements hétérogènes indépendant des langages de programmation et des systèmes d'exploitation;
- basées sur la technologie objet.

Architecture générale



Invocation statique



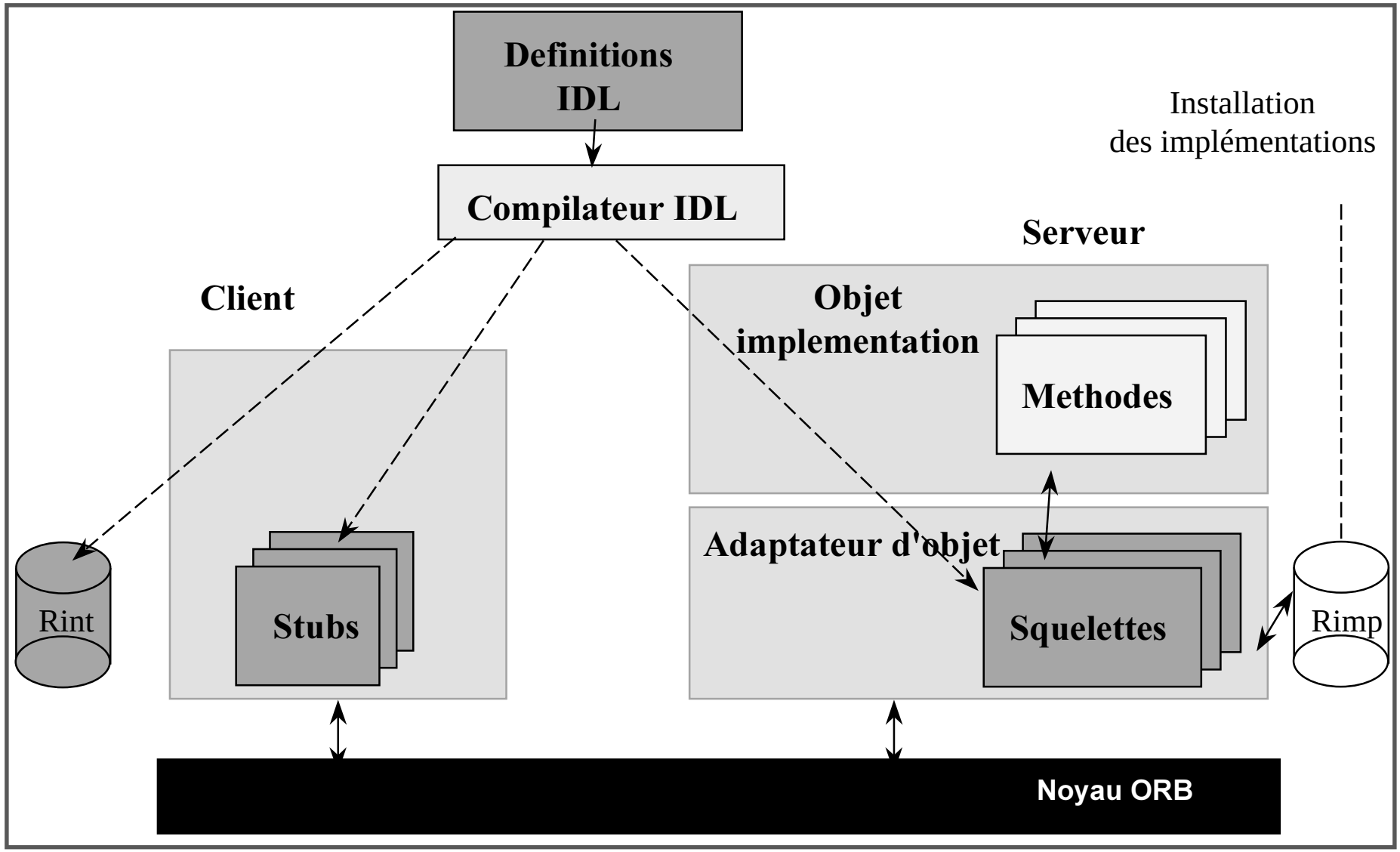
Stub

- Partie de code générée automatiquement par un compilateur IDL vers un langage de programmation cible.
- Code utilisé par le client lors des invocations statiques.
- Lien entre le client et l'ORB.
- Traduit les invocations du client en message transmissibles sur le réseau : opération "marshalling ».
- Traduit les messages qui proviennent de l'ORB en données C++ : opération "unmarshalling".

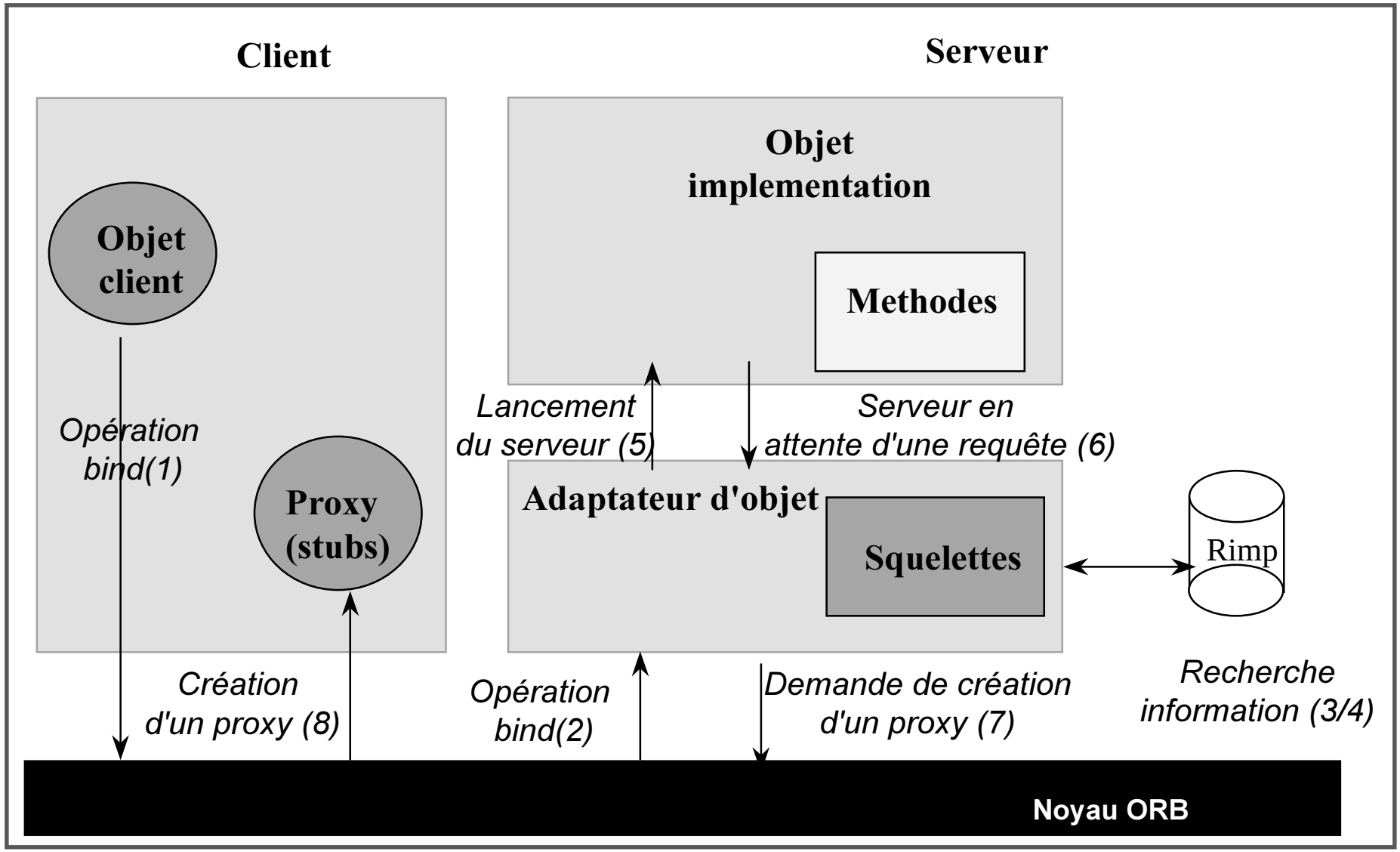
Squelette statique (skeleton)

- Partie de code générée automatiquement par un compilateur IDL vers un langage de programmation cible.
- Code utilisé par l'Adaptateur d'objet lors des invocations statiques.
- Lien entre l'ORB et l'objet d'implémentation.
- Reconstitue la requête du client de façon à invoquer la méthode C++ requise : opération «unmarshalling ».
- Traduit les paramètres de retour en messages transmissibles sur le réseau : opération «marshalling ».

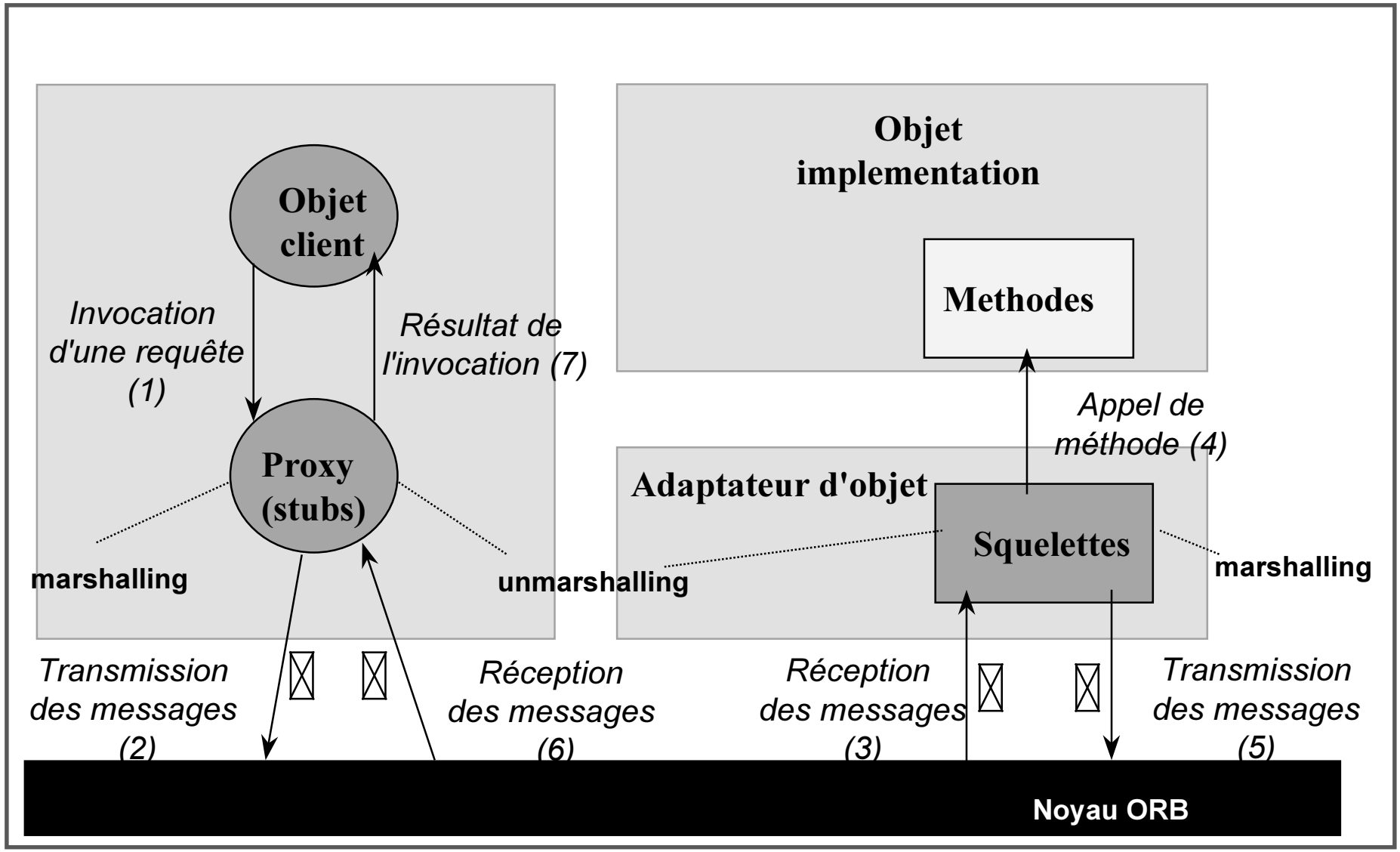
Initialisation des repositories



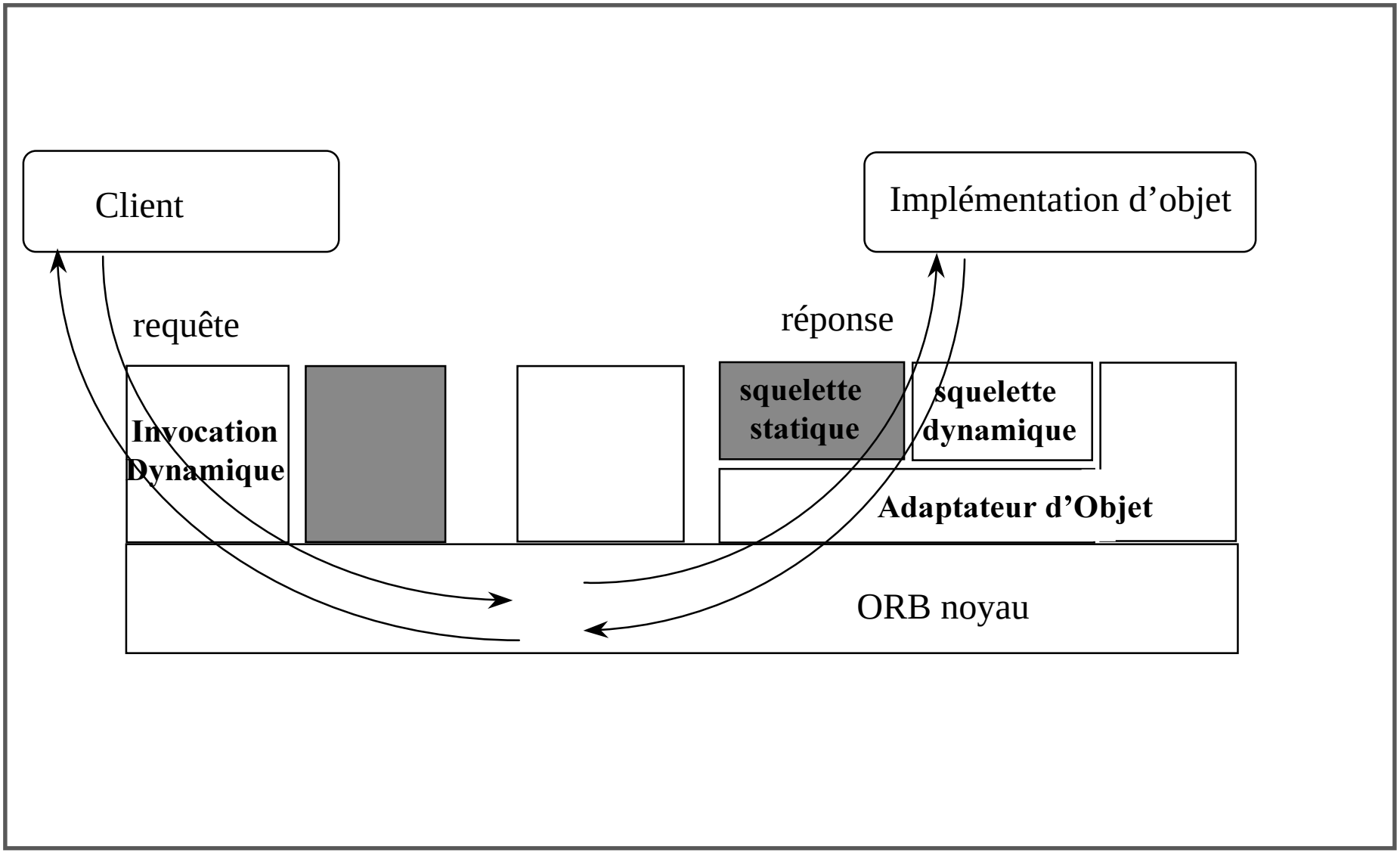
Connexion sur un objet distant



Mécanisme d'invocation



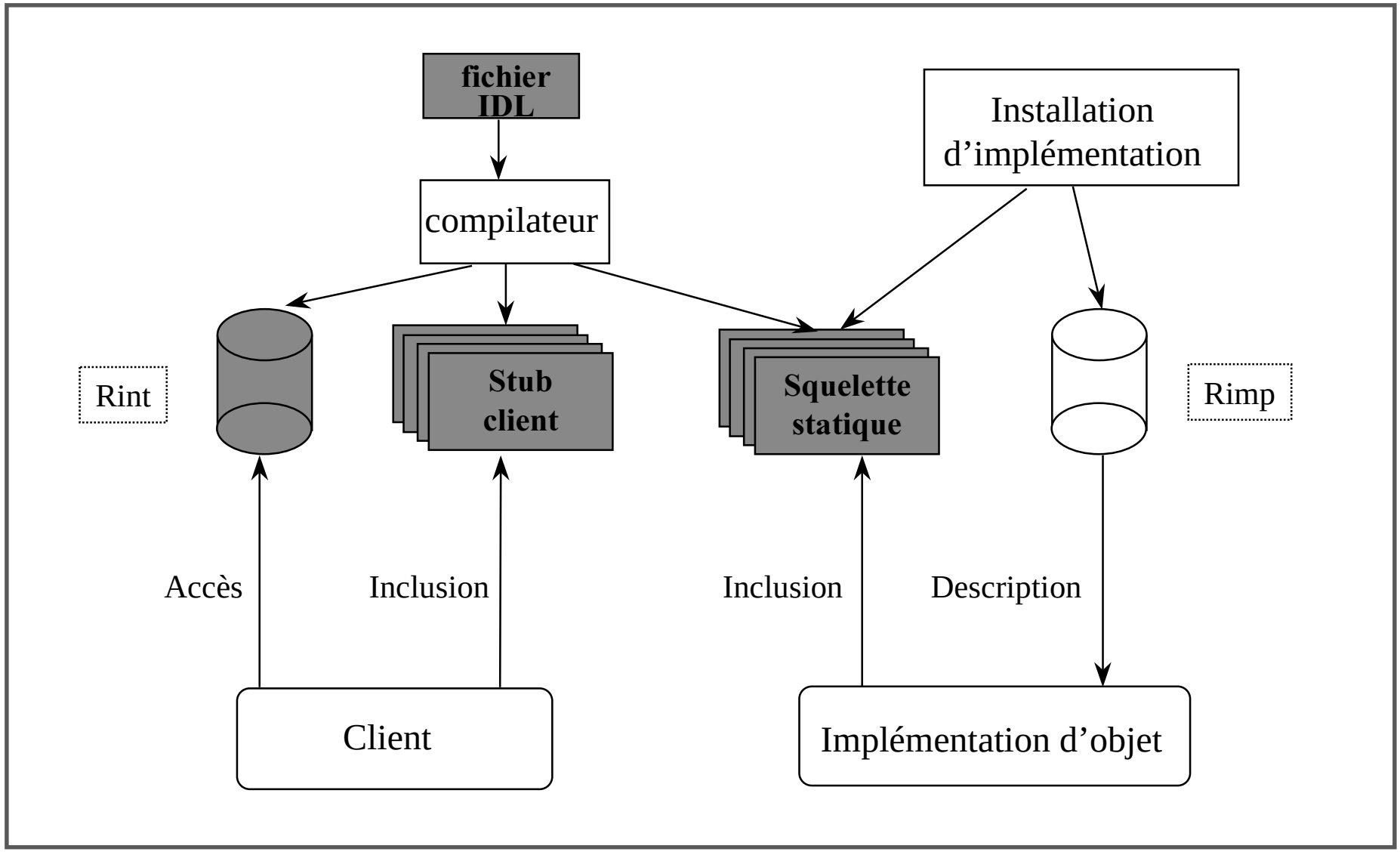
Invocation dynamique



Interface d'invocation dynamique

- Permet la création dynamique de requêtes;
- Utilisation du référentiel des interfaces pour récupérer les informations relatives aux interfaces IDL;
- Avantages :
  - les interfaces peuvent être découvertes dynamiquement;
  - code client générique indépendant d'une interface IDL;
- Etapes de création d'une requête :
  - recherche et interprétation de l'interface dans le référentiel des interfaces;
  - construction d'un objet requête;
  - spécification de l'objet cible et de l'opération;
  - ajout des paramètres;
  - invocation de la requête.

Référentiel d'interfaces et d'implémentations



Référentiel d'interfaces

- Maintient les informations sur les types, les interfaces etc...;
- Informations pour une interface :
  - son module
  - son nom
  - ses attributs
  - ses opérations (nom, nom et types des paramètres, exceptions, contexte)
  - ses héritages

Référentiel d'implémentations

- Responsable de l'enregistrement des serveurs dans le système (enregistrement de la commande exécutable).
- Spécifié par une interface IDL.

- Avec Orbix
  - Controlé par la commande *putit* dans les commandes associées *lsit*, *catit*, *rmit*, *chmodit*.
  - Implémenté par un processus démon.

Interface de squelette dynamique

- Permet de délivrer une requête à un objet implémentation qui est inconnu lors de la phase de compilation.
- Interprète une requête et ses paramètres.
- Analogue au DII mais du côté serveur.
- Utiliser pour créer des ponts entre des ORBs de vendeurs différents.
- Utiliser pour intégrer des applications existantes (legacy application). Les applications peuvent ne pas être conforme aux standard CORBA et peuvent également ne pas être OO.

Object Adapter : fonctions

Fonctions des Adaptateurs d'objets:

1- Enregistrement des objets implémentation.

2- Génération et interprètation des références d'objets.

3- Mapping des références objet vers leurs objets implémentation correspondant.

3- Activation et désactivation des objets implémentation;

4- Invocation des méthodes à travers les squelettes ou le DSI.

Object Adapter : offre

- **Basic Object Adapter (BOA)**
  Application cliente séparée de l'application serveur.

- **Library Object Adapter (LOA)**
  Le serveur est une librairie chargée dynamiquement chez le client.

- **Object-Oriented Database Adapter (ODA)**
  Le serveur est une BDOO.

Les ORB compatibles CORBA doivent implémenter par défaut le BOA.

Basic Object Adapter

Politiques d'activation des objets :

- ☞ serveur partagé : un processus pour tous les objets du serveur;
- ☞ serveur non partagé : un processus par objet;
- ☞ serveur par méthode : un processus par invocation d'opération;
- ☞ serveur persistent : équivalent au serveur partagé sans contrôle sur la création et la terminaison des processus;
  Exemple, TP moniteur et BD distribuées

5- Les services CORBA

- 1er RFP :
  - Nommage, Cycle de vie, Notification d'évènements.
- 2ème RFP :
  - Transactions, Concurrence, Externalisation, Relations.
- 3ème RFP :
  - Sécurité, Serveur de temps.
- 4ème RFP :
  - Propriétés, License, Serveur de requêtes.
- 5ème RFP :
  - Annuaire par fonctionnalités, Collection, Gestionnaire de versions.

Le service de Nommage

Le service de **Nommage** ou ***Namming service*** permet :

- ☞ d'associer un nom à une référence d'objet CORBA, relativement à un contexte de noms;
- ☞ de retrouver la référence d'objet ainsi que l'objet associé;
- ☞ de naviguer à l'intérieur d'un contexte de noms.

Le service Cycle de vie

Le service **Cycle de vie** ou ***Life cycle service*** permet :

- de gérer la création, la destruction, la copie et le déplacement d'objet;
- les objets sont crées par l'intermédiaire d'objets appelés Factories dont la référence est accessible au client;
- un objet est détruit, copié ou déplacé à l'aide d'opérations définient dans l'interface de base LifecycleObject;

Le service de Persistance

Le service de **Persistance** ou ***Persistent service*** permet de gérer la persistance de l'état interne des objets CORBA. Un objet peut :

- ☞ exporter la gestion de sa persistance qui est alors prise en charge par le client ;
- ☞ ou gérer lui-même sa persistance en faisant appel à un *Persistent Data Service (PDS)* qui fournit le mécanisme permettant de rendre les données persistantes et de les manipuler.

Le service de notification d'événements

Le service d'Evénements ou ***Event service*** permet de découpler la communication entre objets.

La plupart des requêtes CORBA se traduisent par l'exécution synchrone d'une opération (le client connaît la référence de l'objet auquel la requête s'adresse et le client ainsi que l'objet doivent être tous deux actifs.

Mode de communication découplé :

☞ lorsque le client et l'objet ne se connaissent pas;

☞ lorsque le client et l'objet ne sont pas actifs simultanément.

Le service de Transaction

Le service de **Transaction** ou ***Transaction service*** permet d'assurer la consistance des traitements en respectant les propriétés ACID (Atomicité, Consistance, Isolation et durabilité) des transactions.

Il permet :

- ☞ de commencer et de terminer une transaction;
- ☞ de contrôler la propagation d'une transaction;
- ☞ d'associer plusieurs objets répartis à une seule et même transaction;
- ☞ de coordonner la terminaison d'une transaction (2 phase commit).

Le service de Concurrence

Le service de **Concurrence** ou ***Concurrency control service*** permet de contrôler l'accès à un objet de manière à gérer la cohérence et la consistance des opérations entre cet objet et les objets qu'il accède ou bien qui l'accèdent.

Il permet de créer des verrous répartis et de spécifier le type de verrou crée (lecture, écriture, mise-à-jour etc...).

Le service d'Externalisation

Le service d'**Externalisation** ou ***Externalization service*** permet :

- ☞ l'externalisation d'un objet, c'est à dire la représentation de l'état de l'objet dans une séquence d'octets (en mémoire, sur disque, sur réseau) transportable, de durée de vie illimitée indépendante de l'environnement (ORB) d'origine.
- ☞ l'internalisation des données, impliquant la création d'un nouvel objet dans le même environnement ou dans un environnement différent.

Le service de Relations

Le service de **Relations** ou ***Relationships service*** permet d'établir dynamiquement des relations entre les objets distribués.
Une relation est définie par un rôle, un degré, une cardinalité et des attributs.

Trois niveaux de services :

- basique : service de base permettant de créer les relations et les objets et de naviguer à travers la relation, de détruire la relation;
- graphes d'objet en relation;
- relations spécifiques : Containment (1-n) et référence (n-m).

3ème RFP

Le **service de Sécurité** (***Security***) permet de gérer les fonctions suivantes :

- authentification
- autorisation
- sûreté et intégrité des communications
- encryptage
- administration de la sécurité

Le service de **Temps** (***Time***) permet la synchronisation périodique d'horloges dans tous les composants d'un système réparti.

4ème RFP

Le **service de Propriétés** (***Properties***) permet d'associer dynamiquement une liste de propriétés à un objet. Il est possible d'ajouter, de supprimer, de modifier et de retrouver toute propriété associée à un objet.

Le **service d'interrogations** (***Query***) permet d'exprimer des requêtes sur des ensembles d'objets CORBA.

Le **service de License** (***Licensing***) contrôle et gère les rémunérations associées à l'utilisation d'un service objet donné.

5ème RFP

Le service de **Gestion des versions** (***Change Management***) permet de gérer l'évolution des versions des interfaces des objets ainsi que de l'adéquation avec leurs implémentations.

Le service d'**Annuaire par fonctionnalités** (***Trader***) permet d'associer des fonctionnalités à des objets CORBA. Le client utilise ce service comme l'annuaire des pages jaunes.

Le service de **Collection** (***Collection***) permet de créer et de manipuler des collections d'objets.

Taxonomie des services

- Nommage + Annuaire par fonctionnalités
- Persitance + Externalisation
- Cycle de vie + Relation
- Serveur de requêtes + Collections + Properties
- Transaction + Concurrence
- Securité + License
- Gestionnaire des versions
- Time
- Event

6 - CORBA 2.0

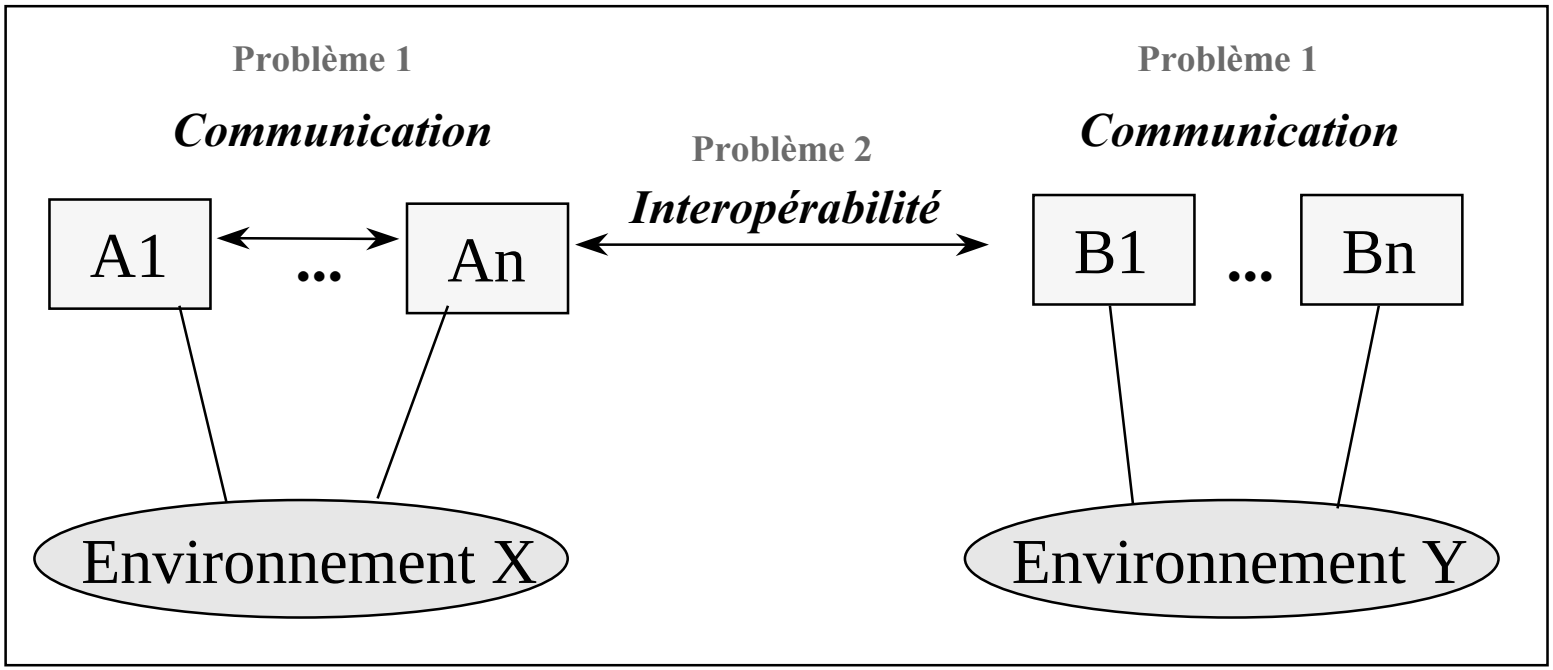
Problème d'interopérabilité entre ORBs.

Interopérabilité

Capacité pour un ORB A d'invoquer une opération définie
en IDL sur un objet résidant sur un ORB B.
L'ORB A et B étant des implémentations de CORBA différentes.

Interopérabilité d'applications avec CORBA

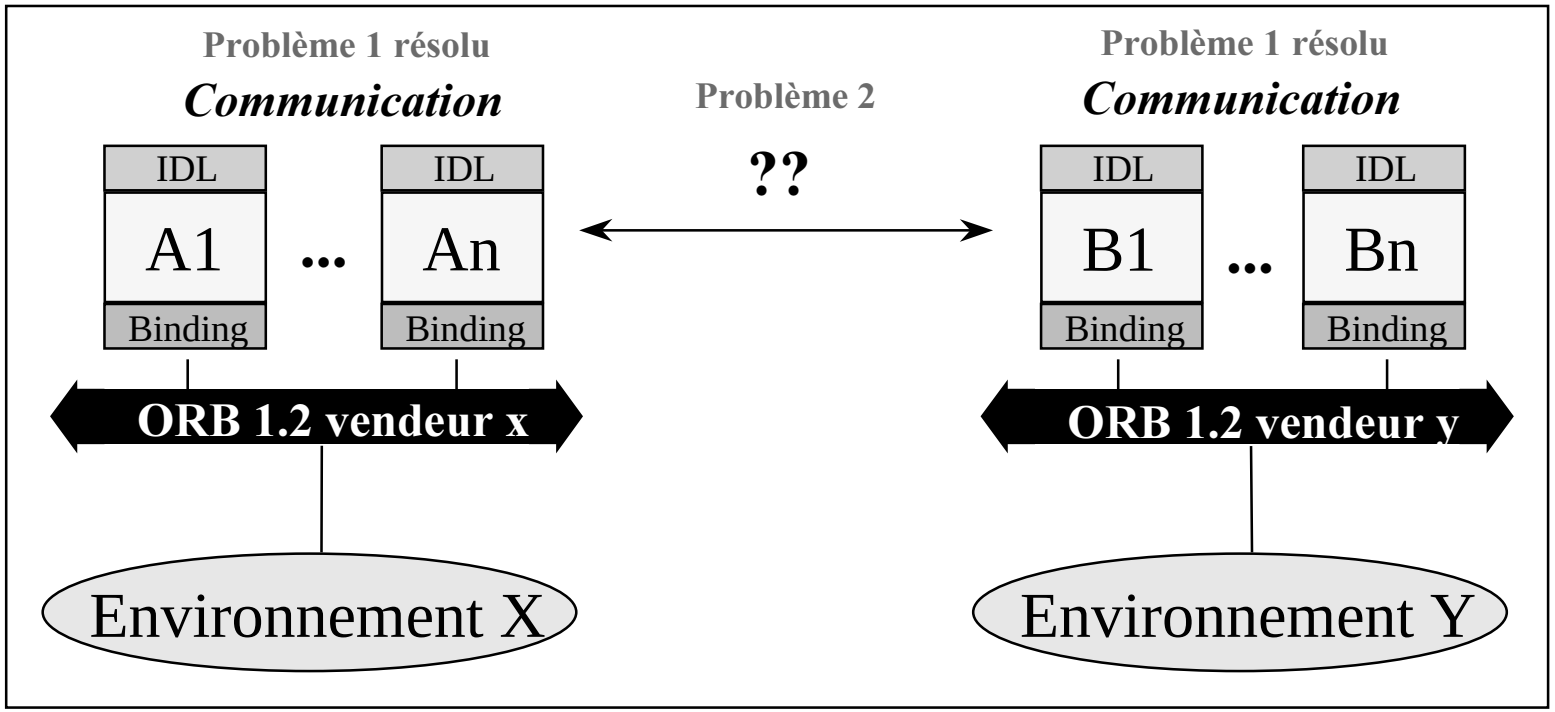
Deux problèmes :
1- communication d'applications distribuées au sein d'un même environnement;
2- interopérabilité d'applications distribuées entre environnements hétérogènes.



Portabilité d'applications avec CORBA 1.2

CORBA 1.2 permet de :

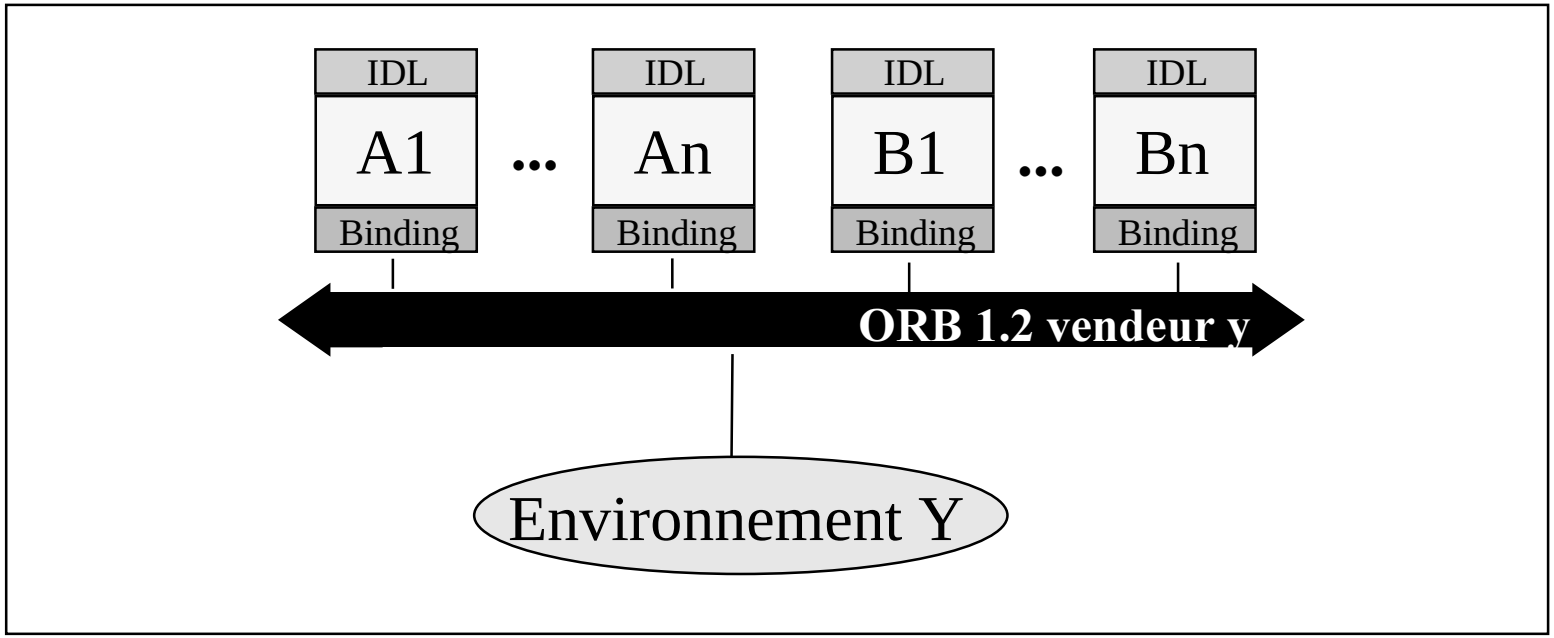
- résoudre le problème de communication d'applications distribuées au sein d'un même environnement;



Portabilité d'applications avec CORBA 1.2

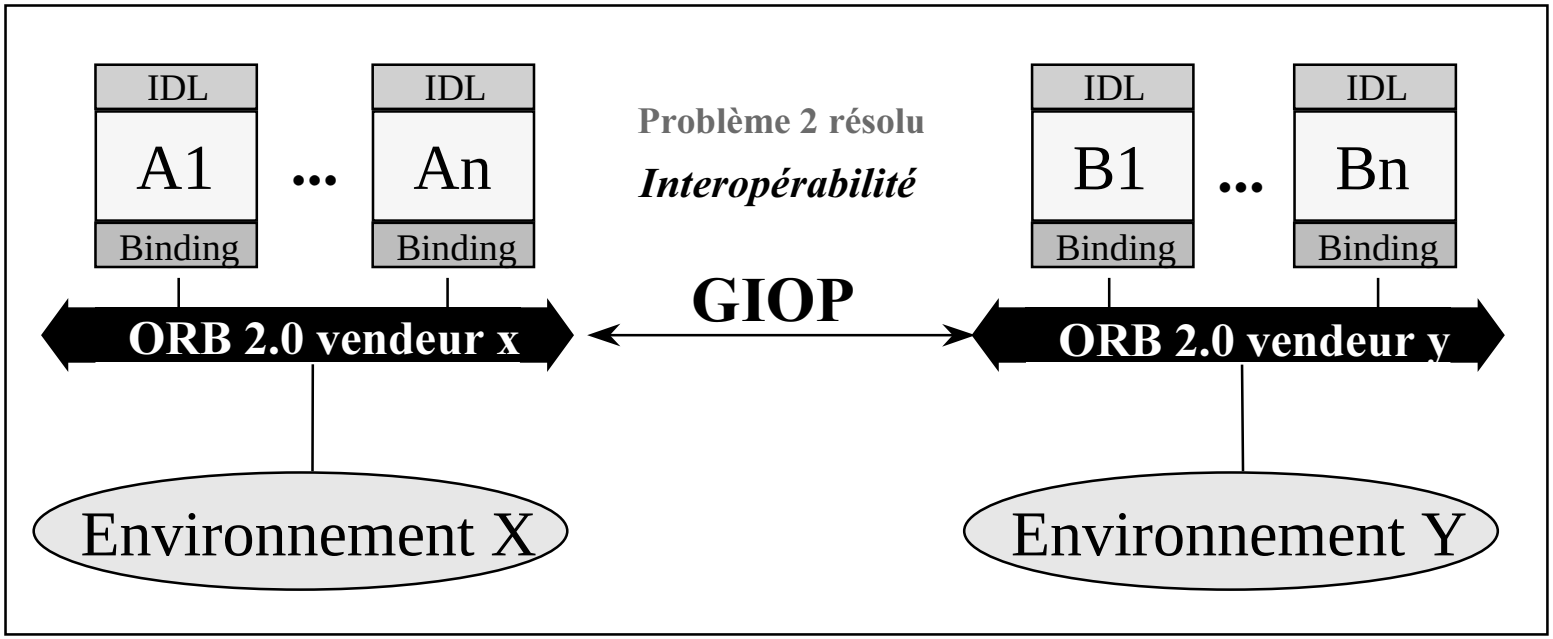
CORBA 1.2 permet de :

- simplifier le portage d'applications entre environnements hétérogènes grâce au langage IDL, aux standardisations des bindings.



Interopérabilité d'application avec CORBA 2.0

CORBA 2.0 permet de résoudre le problème d'interopérabilité d'applications distribuées entre des environnements hétérogènes grâce au protocole de communication commun GIOP (General Inter ORB Protocol).



Solution

La spécification CORBA 2.0 comporte 4 nouveaux éléments :

- le cadre architectural définissant l'interopérabilité entre différents ORBs;
- la définition de protocoles communs GIOP et IIOP;
- la définition de protocoles spécifiques à un environnement ESIOP et DCE/ESIOP;
- la définition de passerelles inter-ORB, permettant la communication entre différentes implémentations de CORBA (comprenant le DSI).

GIOP et IIOP

- **GIOP** (General Inter-ORB Protocol) spécifie un ensemble de formats pour les messages ainsi qu'une représentation commune des données échangées entre les ORBs.
  La représentation des données communes est basée sur la spécification CDR (Common Data Representation).

- **IIOP** (Internet Inter-ORB Protocol) est l'implémentation du protocole GIOP basé sur TCP/IP.

ESIOP et DCE-CIOP

- **ESIOP** (Environment-Specific Inter-ORB Protocol) spécifie un protocole particulier optimisé pour l'environnement OSF DCE (Distributed Computing Environment).

- **DCE-CIOP** (DCE Common Inter-ORB Protocol) est l'implémentation du protocole ESIOP basé sur RPC (Remote Procedure Call).

Passerelles

- **Passerelle complète :** interconnexion directe entre deux ORBs.
  Traduction des messages du 1er ORB au format du protocole de communication du second et réciproquement.
  Méthode performante mais très difficile à mettre en œuvre.

- **Demie-passerelle : connexion indirecte entre deux ORBs.**
  Traduction des messages d ’un ORB dans le format du protocole GIOP/IIOP.
  Solution simple a installer, nombre limitée de demie-passerelle .
  Solution moins performante.

Rm : Un ORB peut utiliser en interne le protocole IIOP

7- Produits CORBA

Market Share

Extract from "Le monde informatique" 14 nov 1997

Survey leaded by the Standish Group

Editors	Product	MS 96	MS 97	MS 98	MS 99	MS 2000
IONA	**Orbix**	**24%**	**26%**	**15%**	**13%**	**12%**
Borland-Visigenic	**Visibroker**	**9%**	**22%**	**55%**	**56%**	**53%**
BEA - DEC	ObjectBroker	27%	21%	12%	13%	14%
Sun	Neo/Joe	5%	4%	2%	2%	2%
IBM	Component Broker	4%	3%	2%	3%	6%
ICL	DAIS	4%	4%	2%	2%	2%
Autres		27%	20%	12%	11%	11%
Total		100%	100%	100%	100%	100%

MS : Market Share

Offres

Vendors	IDL	C++	C	SmallTalk	Ada	IIOP	DCE	DII	Java	OLE
Expersoft	Y	Y	-	Y	-	Y	-	Y	-	Y
Sun	Y	Y	Y	-	-	Y	-	?	Y	-
IONA (Orbix)	**Y**	**Y**	**Y**	**?**	**?**	**Y**	**-**	**Y**	**Y**	**Y**
Borland (Visibroker)	**Y**	**Y**	**-**	**Y**	**-**	**Y**	**-**	**Y**	**Y**	**?**
BEA (DEC-ObjectBroker)	Y	Y	Y	?	?	Y	Y	Y	Y	Y
HP	Y	Y	?	?	?	Y	Y	Y	?	Y
IBM	Y	Y	Y	Y	?	Y	?	Y	?	Y
Chorus	Y	Y	-	-	-	+	-	Y	?	?
Siemens					Cf. Sun					
Tandem					Cf. IBM					
ILOG					Cf. IONA					

? : unknown information
Y : feature supported but not necessarily conforming to CORBA standard
- : not supported
+ : being implemented

Implémentation des services

Vendeurs	Nm	Lf	Ev	Tr	Rl	Cc	Ex	Po	Tx	Qr	Tm	Pr	Cm	Sc	Li
Expersoft	Y		+												
Sun	Y	Y	Y		Y							Y			
IONA (Orbix)	**Y**		**Y**	**Y**					**Y**						
Borland (Visibroker)	**Y**		**Y**	**Y**											
BEA (DEC-ObjectBroker)	Y			**Y**					Y						
HP	Y	Y	Y												
IBM	Y	Y	Y			Y	Y	Y	Y						
Chorus	Y														
Siemens	Cf. Sun														
Tandem	Cf. IBM														
ILOG	Cf. IONA														

Nm : Naming
LF : Lifecycle
Ev : Event
Tr : Trading
Rl : Relationships
Cc : Concurrency
Ex : Externalisation
Po : Persistent Objects
Tx : Transactions
Qr : Query
Tm : Time
Pr : Properties
Cm : Configuration Management
Sc : Security
Li : licensing

Références

[T. J. Mowbray, R. Zahavi] "The Essential CORBA - System Integration Using Distributed Objects"
J. Wiley and Sons 1996
[R. Orfali, D. Harkey, J. Edward] "The Essential Distributed Objects Survival Guide"
J. Wiley and Sons 1996
[J. Siegel &al] "CORBA Fundamentals and Programming"
J. Wiley and Sons 1996
[OMG 95 - CORBA 1.2] "The Common Object Request Broker : Architecture and Specification"
Object Management Group
[OMG 95 - CORBA 2.0] "The Common Object Request Broker : Architecture and Specification"
Object Management Group
[OMG 94-1-1] "Common Object Services Specification"
Object Management Group
[OMG 94-11-12] "Objects Services Architecture"
Object Management Group

World Wide Web : http://www.omg.org
CORBA for beginners : http://www.omg.org/news/begin.html