
# Java

# *et les objets distribués*

Ce document présente les technologies de distribution d'objets Java. Il s'appuie sur l'expérience accumulée, dans ce domaine, par les équipes Java d'ima.

La synthèse réalisée décrit :

- Les modèles d'architectures distribuées Java
- L'API *Remote Method Invocation* (RMI)
- L'intégration CORBA/Java
- Les passerelles CORBA/DCOM

*ima* SOLERI -immeuble Lavoisier, La Défense 5, 92052 Paris La Défense Cedex 64
Téléphone : **01 41 26 62 61** . Télécopie : **01 41 99 91 23** e-mail : imacours@soleri.com

*Java par l'intermédiaire du mécanisme des applets s'est imposé dès ses débuts comme une plate-forme de choix pour l'implémentation d'architectures distribuées. Ce n'est cependant qu'avec la normalisation du mapping CORBA / Java et la sortie du JDK 1.1 que Java a acquis une stature suffisante pour s'attaquer au marché des applications à objets distribués.*
*Les technologies sont prêtes mais le marché est neuf, les acteurs sont nombreux et l'offre peu structurée.*
***Ima,** pôle formation conseil aux nouvelles technologies de* SOLERI *met à la disposition de ses clients son expérience dans le déploiement de ces différentes technologies objet pour leur offrir des repères et les aider à planifier leurs choix.*

## Pourquoi les objets distribués ?

### Valeur ajoutée d'une architecture

Longtemps les architectes informatiques ont concentré leurs efforts sur l'objectif d'assurer l'efficacité intrinsèque d'une application informatique en négligeant les possibilités d'évolution dans une architecture hétérogène. Sur le long terme, la capacité d'un système à s'intégrer à d'autres systèmes informatiques est aussi importante que la qualité de l'implémentation des services qu'il doit offrir.

L'initiative CORBA est née de ce transfert des préoccupations vers les bornes du système (amorcé avec l'apparition du Client/Serveur).

### Evolution et compétitivité

La réactivité est un élément fondamental de la compétitivité d'une entreprise. CORBA offre la possibilité de faire évoluer un système d'information à la demande et rapidement.

Java permet de s'affranchir de l'hétérogénéité des systèmes d'exploitations et les objets distribués apportent le moyen de faire évoluer les applications sans remettre en cause l'existant.

### Les modèles de distribution d'objets

On dénombre à l'heure actuelle 3 modèles de distribution d'objets :
– CORBA
– RMI
– DCOM
Chacun de ces modèles présente ses avantages spécifiques.

L'objectif d'***ima*** informatique est de fournir à ses clients l'information nécessaire à la réalisation d'un choix technologique (perspectives d'évolution à long terme, moyens d'intégration à l'existant, produits) et les ressources indispensables à sa mise en œuvre (formation, équipes de développement expérimentées).

## CORBA et Java

### Qu'est ce que CORBA ?

CORBA (*Common Object Request Broker Architecture*) est une norme de distribution d'objets définie par *l'OMG (Object Management Group*), un organisme à but non lucratif qui regroupe actuellement plus de 850 entreprises du secteur de l'informatique.
CORBA permet à des objets développés dans différents langages (C, C++, ADA, Smalltalk et Java) d'interagir au travers d'un réseau.

### Le middleware CORBA : l'ORB

Le middleware qui permet à un objet de détenir une référence virtuelle sur un objet distant est nommé ORB (*Object Request Broker*).
Du point de vue du client, utilisateur d'un objet CORBA instancié sur le serveur, tout se passe comme s'il détenait une référence sur un objet local.

### Java et CORBA

Java a été conçu pour réaliser des applications distribuées. L'intégration de Java à la norme CORBA s'est donc faite rapidement et naturellement. L'offre produit est aujourd'hui importante et est dominée par OrbixWeb de IONA Technologies.

### Langage IDL et interfaces

Le modèle CORBA est basé sur la définition d'interfaces publiques pour les objets que l'on souhaite rendre accessibles au travers du réseau.

```
interface compte {
      attribute string proprietaire;
      void credite(in float);
      void debite(in float);
      attribute float solde;
};
```

Fig 1 : Interface IDL décrivant un objet compte bancaire

Ces interfaces sont décrites au moyen d'un langage de définition d'interfaces : IDL (*Interface Definition Language*).
Une interface IDL ne fait que décrire les services offerts par un objet CORBA. Elle est totalement dissociée des caractéristiques de l'implémentation de l'objet.

### *Stub* et *Skeleton* : les *proxies* CORBA

Le compilateur IDL est lui spécifique à un langage. La compilation de l'interface IDL produit du code qui implémente deux *proxies* :
- le *proxy* client ou *stub* ;
- le *proxy* serveur ou *skeleton*.

Les *proxies* (mandataires) jouent le rôle d'intermédiaires dans le dialogue entre objets client et objets serveur.

Le code généré par le compilateur IDL est en partie normalisé dans le *mapping* CORBA propre au langage d'implémentation.
Actuellement, 5 *mappings* CORBA ont été définis : C, C++, ADA 95, Smalltalk, Java.
Les *mappings* Cobol et ObjectiveC sont en cours de normalisation.

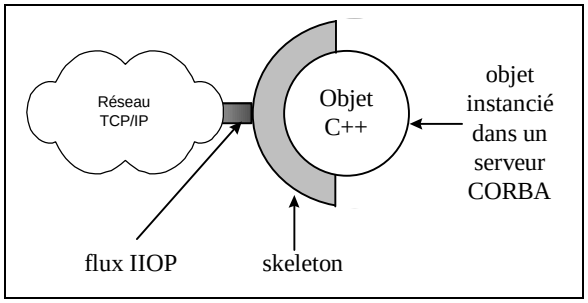


Fig 2 : Partie serveur d'une référence CORBA

Le *stub* et le *skeleton* assurent, en collaboration avec l'ORB, lors des appels aux méthodes de l'objet CORBA distant, la transmission des paramètres et des valeurs de retour (*marshaling* et *unmarshaling*). Le protocole utilisé est IIOP.

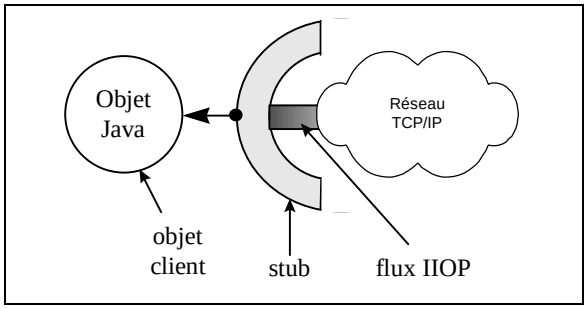


Fig 3 : Partie cliente d'une référence CORBA

Une fois les *proxies* générés, le développeur doit bien sûr implémenter l'objet. Il doit fournir la logique applicative qui viendra se greffer derrière l'interface publique de l'objet. Encore une fois, cette implémentation n'a pas à être connue de l'utilisateur final de l'objet CORBA.

### IIOP : l'espéranto des ORB

Le choix de CORBA comme technologie d'objets distribués n'impose pas le choix d'un éditeur grâce à IIOP (*Internet Inter-ORB Protocol*).
IIOP permet un dialogue entre ORB de différents éditeurs en définissant un protocole de communication standardisé au dessus de TCP/IP.

## *Remote Method Invocation* (RMI)

Intégré au *Java Developpement Kit* depuis la version 1.1, RMI est un protocole de distribution d'objets Java basé sur TCP/IP.

### Fonctionnement : un ORB RMI ?

L'ORB RMI n'est autre que la machine virtuelle Java. Pas d'IDL avec RMI, cette solution exploite directement les interfaces Java.
En effet, le développeur utilise une interface pour décrire les services fournis par un objet RMI serveur. La génération des *proxies* (*stub* et *skeleton*) est réalisée par un utilitaire (*rmic*) distribué avec le JDK.

### Une solution légère

La bonne intégration de RMI à Java et son faible coût d'acquisition en font une plate-forme de choix dans le cadre d'applications Intranet de moyenne ampleur.

Dans le cadre de projets de plus grande échelle, il n'est pas possible d'envisager cette solution sans développements complémentaires importants.
La prochaine adoption du standard IIOP comme protocole de communication pour RMI devrait contribuer à gommer la singularité de cette approche face à la technologie CORBA.

## DCOM

L'architecture COM (*Component Object Model*) et son extension distribuée DCOM (*Distributed COM*) sont soutenues par Microsoft. Etroitement dépendant des systèmes Windows, ce modèle de composants distribués constitue cependant une alternative envisageable dans le cas de systèmes "tout Windows".

### Produits et solutions

DCOM est intégré à Windows NT depuis la version 4.0 de cet OS. Il est assez aisé de développer des composants COM/ActiveX à l'aide de nombreux L4G (VisualBasic, PowerBuilder, Delphi, ...). En revanche cette technologie apparaît plus difficile d'accès (et plus *mono-éditeur*) depuis C++.
En outre la difficulté de configuration des solutions DCOM a longtemps rebuté les développeurs d'applications.

### Modèle de composants

L'essentiel de la force de DCOM réside dans son modèle de composants (*Component Object Model*) et dans son intégration à Windows.
Les objets COM interagissent facilement avec l'existant Windows 32 bits (avec la bureautique par exemple).

### Les passerelles DCOM/CORBA

Généralement le besoin d'utiliser COM provient d'une nécessité de s'intégrer à un existant Windows.

La plupart des principaux éditeurs d'ORB CORBA proposent des ponts qui permettent à des clients DCOM de référencer des objets CORBA (et inversement).

Une autre solution pour Intégrer COM et CORBA consiste à utiliser *le bridge ActiveX/JavaBeans* de SUN. Cet outil permet de générer une interface de type COM autour d'un composant JavaBeans et il est ainsi possible de faire interagir facilement COM et CORBA (par l'intermédiaire de Java).

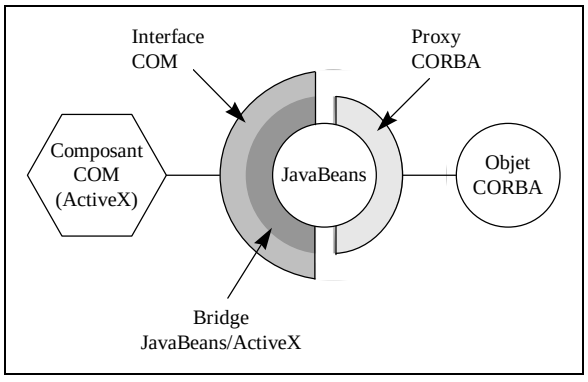


Fig 4 :Utilisation du bridge JavaBeans/ActiveX

## Choisir Java

La conception même de Java en faisait un langage « orienté architecture distribuée » : objet, interprété et portable.
Il ne manquait à Java qu'un modèle de composants et le mapping avec l'IDL CORBA. Le premier est apparu avec le JDK 1.1 et le second a été récemment ratifié par l'OMG.

Java peut s'intégrer aujourd'hui mieux que n'importe quel autre langage à tous les types d'architectures à objets distribués.

La mise sur le marché d'environnements de développement exploitant les composants JavaBeans offre la possibilité de réaliser des applications modulaires et relance le concept de client léger (concentré sur les tâches de gestion de l'interface utilisateur).

## Quelques adresses utiles

Object Management Group (OMG)
http://www.omg.org

IONA Technologies
http://www.iona.com

JavaSoft (filiale de SUN en charge des développements logiciels autour de Java)
http://www.javasoft.com

IBM et Java
http://www.ibm.com/java/

VisualAge pour Java
http://www.software.ibm.com/ad/vajava/

ActiveX
http://www.activex.com