CORBA

Olivier Aubert

Liens

- `http://www.infosys.tuwien.ac.at/Research/Corba/OMG/arch2.htm`
- `http://hegel.ittc.ukans.edu/projects/corba/presentation/intro1.html`
- `http://perso-info.enst-bretagne.fr/~beugnard/cours/Corba.pdf`

Programmation Orientée Objet

- Un objet est une entité identifiable, encapsulée, qui fournit un ou des services pouvant être demandés par un client.
- Points importants :
  - Encapsulation : les données sont cachées
  - Polymorphisme : la possibilité de changer de comportement
  - Héritage : réutilisation
  - Composants

Calcul distribué

Une des manières de répondre aux demandes croissantes en puissance de calcul.

- Partage des ressources (optimisation de leur utilisation)
- Accélération des calculs (traitement parallèle)
- Robustesse (par la réplication)
- Portabilité / Passage à l'échelle

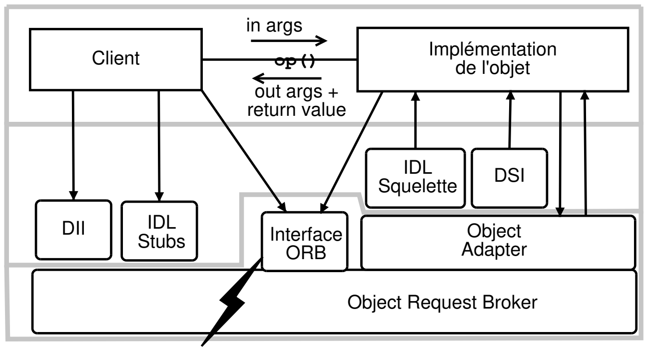
OMG

- Object Management Group
- `http://www.omg.org/`
- Créé en 1989 par Sun, HP, Dec, NCR, etc.
- Consortium à objectif non lucratif, consacré à la promotion des théories et des pratiques autour des objets distribués
- Regroupe de nombreux industriels, développeurs et utilisateurs
- Définition du standard CORBA

CORBA

- Common Object Request Broker Architecture
- Spécifié par l'OMG
- CORBA est un standard, pas un produit
- Mécanismes par lesquels des objets peuvent envoyer des requêtes et recevoir des réponses de manière transparente
- Intéropérabilité entre différentes applications sur différents environnements dans différents langages.
- Approche orientée objet

Composants principaux



- Object Request Broker (ORB)
- Interface Definition Language (IDL)
- Dynamic Invocation Interface (DII)
- Dynamic Skeleton Invocation (DSI)
- Interface Repository (IFR)
- Object Adapters (OA)

ORB

- *Objet Request Broker*
- Utilise des *Object References* uniques pour identifier et localiser des objets (les clients manipulent des *Object References*)
- CORBA 2.0 spécifie des *Interoperable Object References* (IOR)
- Fait suivre les requêtes jusqu'aux objets
- Renvoie les valeurs de retour aux clients
- Les services mis en œuvre sont transparents aux clients
- Interface ORB : contient les services CORBA de base (`object_to_string`, `string_to_object`)
- L'ORB met en œuvre pour une référence
  - `get_interface` retourne les méta-données de l'objet
  - `get_implementation`
  - `is_nil`

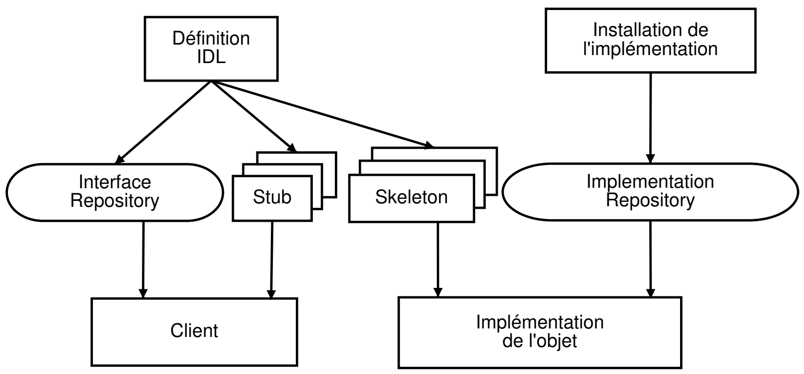
ORB vs. RPC

- RPC a besoin d'un accès réseau (pas de lancement local)
- RPC doit spécifier un serveur prêt à exécuter le service

IDL

- *Interface Definition Language*
- Découple l'implémentation de l'objet de son interface
- Langage déclaratif, syntaxe proche de C++
- Définit des types d'objets en spécifiant leurs interfaces, le nom de leurs opérations et des paramètres.
- Moyen par lequel les clients peuvent obtenir les opérations disponibles sur un objet.
- Traduit vers le langage de mise en œuvre désiré (C, C++, java, perl, python, etc)
- La compilation d'IDL produit des talons et des squelettes :
  - Talon (*stub*) : fonction locale que le client peut appeler
  - Squelette (*skeleton*) : fonction située sur le serveur effectuant l'appel local à l'objet.

IDL (II)



IDL - exemple

```
module POS {
  typedef string Barcode;
  interface InputMedia {
     typedef string OperatorCmd;
     void barcode_input (in Barcode item);
     void keypad_input(in OperatorCmd cmd);
  };
  interface OutputMedia {
     boolean output_text(in string string_to_print);
  };
};
```

Modules et portée

- Définissent des paquets de définitions
  - types
  - constantes
  - exceptions
  - interfaces
- Les définitions ne sont visibles que dans la portée des `{...}` englobantes
- Spécification du module par l'opérateur `::`
  `POS::Barcode`

Types IDL

- Définition de nouveaux types par `typedef`
- long, short, unsigned long, unsigned short
- float, double
- char (ISO-Latin1), string
- boolean
- octet
- any
- array
- struct
- union
- enum
- sequence

Interfaces et exceptions

- Interfaces : définissent des opérations naturellement liées
  - type de l'opération
  - nom de l'opération
  - types et noms des paramètres (in, out, inout)
- Exceptions : retournées au client via l'ORB pour traitement

```
exception input_out_of_range { long dummy };
void op (in long arg) raises
(input_out_of_range);
```


Sémantique des invocations

- Les opérations retournent au moins un objet
  - type
  - out
  - inout
- Sauf celles spécifiées *oneway*
  - doivent retourner `void`
  - ne pas avoir de paramètres `out` ou `inout`
  - ne pas lever d'exception
  - Exemple: `oneway void output (in string message);`

Attributs

- Définissent l'état des objets

```
interface exemple {
  attribute float height;
  readonly attribute float width;
}
```

- Et implicitement des fonctions d'accès

```
interface exemple {
  void _set_height (in float h);
  float _get_width ();
}
```


Héritage

- Relation entre les interfaces introduite par `:`
- Redéfinition des éléments interdite
- `interface voiture : vehicule { ... };`

DII

- *Dynamic Invocation Interface*
- Permet la découverte et la construction dynamique des requêtes aux objets
- Possibilité d'utiliser des objets dont l'interface n'est pas connue au moment de la compilation
- Plus flexible que l'approche statique, mais plus complexe et moins de vérifications de type
- Principe :
  - `Object::get_interface`
  - `InterfaceDef::describe_interface`
  - `Objet::create_request`
  - `Request::add_argument`
  - `Request::invoke`

Interface Repository

- *Interface Repository* (IR)
- Service fournissant des objets persistants permettant d'accéder aux différentes IDL au moment de l'exécution
- Enregistre les IDL
- Fournit les informations de type nécessaire pour la DII
- Stocke des informations supplémentaire (debug, bibliothèques de stubs ou skeletons, etc)
- garantit la correction du graphe d'héritage

Object Adapter

- *Object Adapter* (OA)
- Fournit une interface unifiée aux différentes implémentations et contrôle le cycle de vie des objets
- Autorise plusieurs méthodes de mise en œuvre et facilite l'intégration des applications existantes
- Les implémentations doivent s'enregistrer auprès de l'OA
- Quand un client demande un service à un objet, l'OA fait suivre la requête à l'implémentation adéquate (en créant l'objet si besoin)

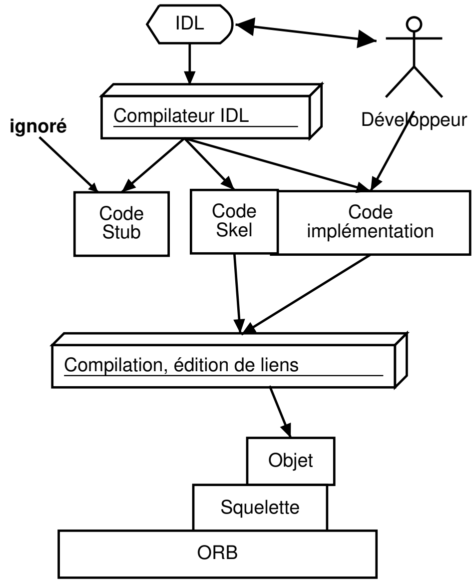
COSS

- *Common Object Services Specification*
- Ensemble de services fournis par l'ORB pour faciliter le développement d'applications
- *Naming Service* : fait correspondre des noms aux références
- *Event Service* : interfaces pour envoyer et recevoir des événements
- *Lifecycle Service* : définit les conventions pour créer/supprimer/copier/déplacer les objets
- *Persistent Object Service* : interface commune aux mécanismes de persistence
- *Transactions Service* : support de différents modèles de transaction
- *Query Service* : permet d'envoyer des requêtes à des groupes d'objets
- *Concurrency Control Service* : coordonne l'accès de multiples clients à une ressource partagée
- *Externalization Service* : transformer l'objet en un flux de données (externalisation) ou l'inverse (internalisation)

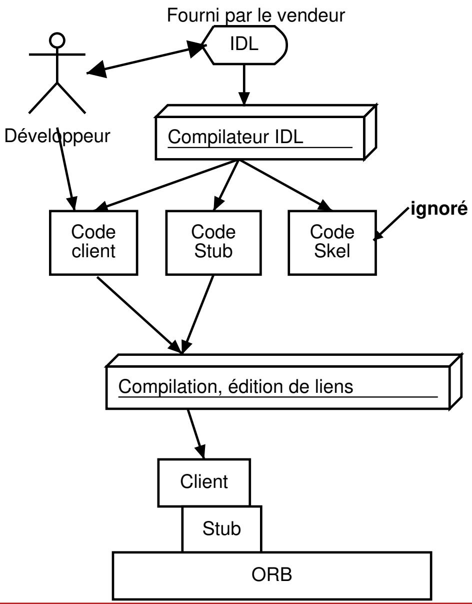
Communication client-serveur

- La compilation de l'IDL produit des stubs (côté client) et des squelettes (côté serveur)
- Le client obtient une référence sur un objet, et lui envoie ses requêtes
- L'ORB transforme la requête dans un format transmissible
- L'ORB demande à l'OA l'implémentation à utiliser
- L'IR assiste l'ORB pour la vérification de types
- L'ORB décode la requête et la passe au squelette
- Le squelette transmet la requête à l'objet réel et renvoie au talon la réponse (avec les exceptions éventuelles)

Fournir un service



Exploiter un service



Trouver le destinataire

- Côté client
  - On connait son *Object Reference*
    - via la réponse à une requête précédente
    - via le service de l'ORB `string_to_object`
  - On s'adresse à un service de nommage
- Côté serveur
  - C'est le rôle de l'*Object Adapter*
  - Recherche dans l'*Implementation Repository* `association (ObjetReference objet)`

Communication inter-ORB

- Spécification des moyens de communiquer entre ORB différents.
- GIOP (General Inter-ORB Protocol)
  - Common Data Representation (CDR)
  - GIOP Message Formats
  - GIOP Transport Assumptions
- IIOP (Internet Inter-ORB Protocol), spécialisation de GIOP dans le cas d'Internet
  - Internet IOP Message Transport

Le service de nommage

- Effectue le lien entre un nom et un objet
- Service standardisé
- Partage des services de nommage entre ORB

Définition du nommage

```
module CosNaming {
  typedef string Istring;
  struct NameComponent {
    Istring id;
    Istring kind;
  };
  typedef sequence<NameComponent> Name;
  interface NamingContext {
    void bind(in Name n, in Object obj)
       raises(NotFound, CannotProceed, InvalidName, Al
    void bind_context(in Name n, in NamingContext nc)
       raises(NotFound, CannotProceed, InvalidName, Al
    Object resolve(in Name n)
       raises(NotFound, CannotProceed, InvalidName);
  };
```