SYSTÈMES D'OBJETS

RÉPARTIS

G. Florin

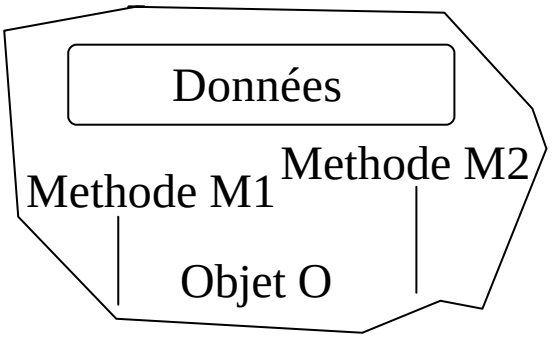
INTRODUCTION
Langages et systèmes d'objets répartis

• Issus de la convergence de deux approches

L'approche objet

Une structuration des applications qui place la notion d'objet au centre de la conception: performante et de plus en plus utilisée

Objet: **associe les traitements et les données** en rendant les codes moins coûteux à développer, à maintenir, à réutiliser.



Abstraction

Comportement communs => Typage.

Encapsulation

Association des actions aux données.

Héritage

Réutilisation du code.

Polymorphisme

Réutilisation des références.

La distribution

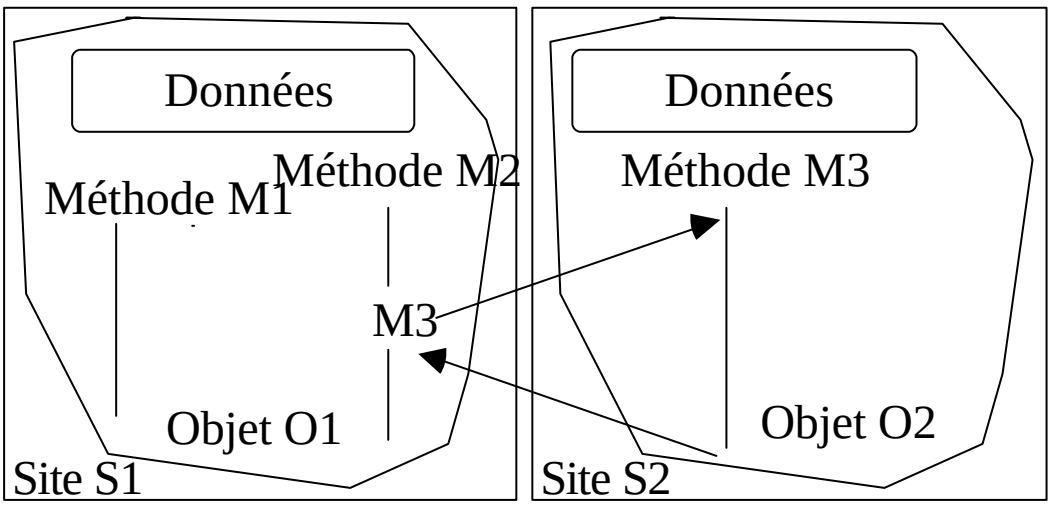
Intégration de la notion d'objet au système réparti

Les communications réseau offrent la possibilité d'implanter les objets de façon homogène sur les sites d'un système réparti.

L'interface du système d'objet réparti est un ensemble de primitives ou un langage objet concurrent et réparti

=> offrant la possibilité **d'implanter et d'exécuter à la demande** des objets sur des sites quelconques

- **Création à distance** d'instances
- **Exécution à distance** des méthodes.
- **Concurrence/synchronisation** .
- **Autres fonctions**: transactionnel, sécurité, optimisation du placement ...



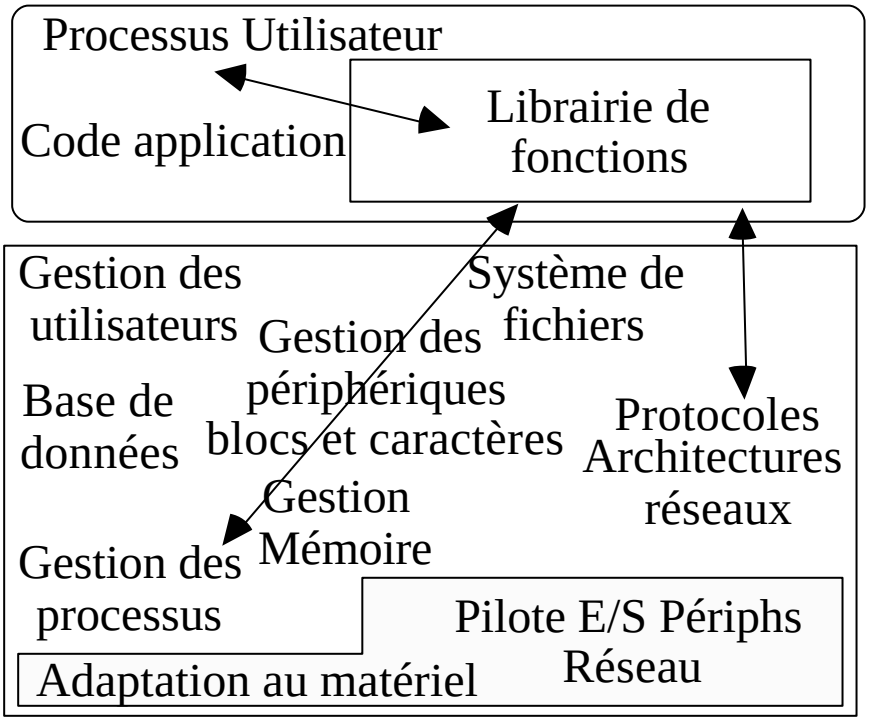
Un enjeu technologique majeur

Situation actuelle: l'hétérogénéité de désignation, de protection, d'interface

Objets manipulés par un utilisateur:

Fichiers, données en mémoires, processus, communications, objets spécifiques usagers ...

Les entités manipulées sont désignées, protégées, traitées de façon différente.

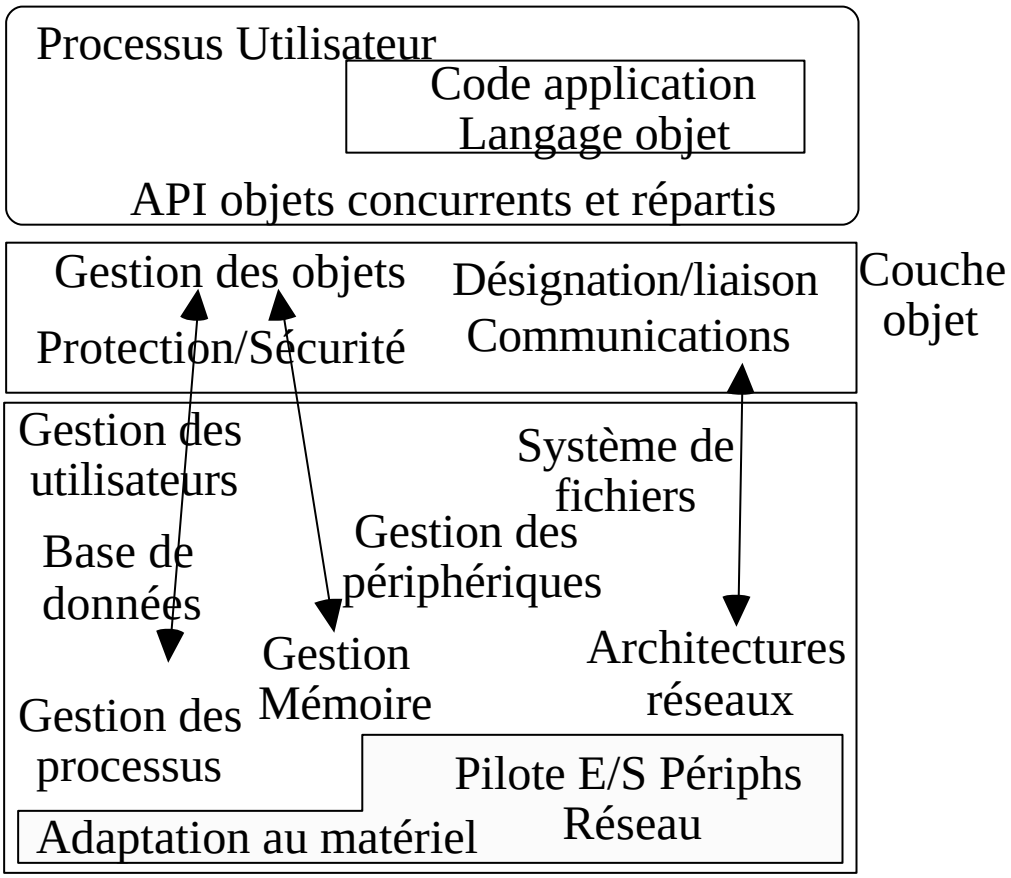


Situation actuelle: complexité de la programmation des applications particulièrement en univers réparti (hétérogène).

Projet : simplifier et uniformiser les accès et les communications offerts par les interfaces systèmes et les codes utilisateurs

Uniformisation des abstractions utilisées sous le concept objet

- Désignation universelle
- Interactions de communication simplifiées
- Protection et sécurité universelle



Réaliser effectivement le projet défini initialement pour les systèmes répartis

Nombreuses difficultés

- Performance des systèmes et applications développées en approche objets répartis.

Les solutions objets imposent le plus souvent un nombre de messages plus élevés et des délais d'interactions plus importants.

- Adaptation des développeurs aux techniques objets répartis.

Les habitudes changent difficilement.

- Difficulté d'une normalisation réelle permettant l'interopérabilité des applications.

Un enjeu pour la normalisation

Standardiser l'offre systèmes d'objets répartis pour intégrer des systèmes d'origine différentes dans une application:

La normalisation minimum habituelle par les protocoles (standardiser les interactions):

=> interopérabilité entre des implantations

- unifier les techniques d'appel distant
- accéder à des annuaires communs d'objets.

Puis standardiser les langages et techniques de développement ...

=> portabilité des applications

OMG - CORBA

"Object Management Group"

"Common Object Request Broker Architecture"

Microsoft OLE/DCOM

"Object Link Embedding"

"Distributed Component Object Model"

Beaucoup de travail reste à faire.

Des produits commerciaux

Un enjeu technologique pour tous les constructeurs et les consortiums.

Les plates-formes CORBA

IONA, **Visigenic**, **Chorus COOL**, **IBM DSOM** "Distributed System Object Model", ...

La plate-forme Microsoft OLE/DCOM/Active X

"Distributed Component Object Model"

Très nombreuses autres annonces (d'importance industrielle variable)

Plan

INTÉGRATION DE LA RÉPARTITION DANS L'APPROCHE OBJET: LE STANDARD CORBA

- Introduction: situation du standard
- La norme CORBA 2.0
- L'interopérabilité en CORBA 2.0

INTÉGRATION DE LA RÉPARTITION DANS L'APPROCHE OBJET: LE STANDARD CORBA

Introduction: situation du standard

L'OMG "Object Management Group"

Consortium d'entreprises, d'organismes
de systèmes/réseaux,
du logiciel,
d'utilisateurs finaux
Intéressés par les technologies objet.
13 membres en 1989 jusqu'à plus de 700 membres.

Objectif principal

Promouvoir la théorie, l'utilisation des objets dans les systèmes répartis (portabilité, réutilisation, interopérabilité dans les systèmes d'objets répartis ouverts).

Moyen principal

Définir des standards pour l'interopérabilité et la portabilité d'applications ‘objet réparti’
Définir les services dont elles ont besoin.

Fonctionnement

Émission de RFI et RFP

"Request For Information"

"Request For Proposals"

Expression d'un cahier des charges pour une fonction, un service par un groupe de travail de l'OMG

Réponses par différents organismes

Adoption d'une (ou plusieurs) propositions pour définir un standard.

Conséquence importante

L'OMG produit des normes (approche normative).

Les logiciels effectifs sont ensuite développés par qui le souhaite:

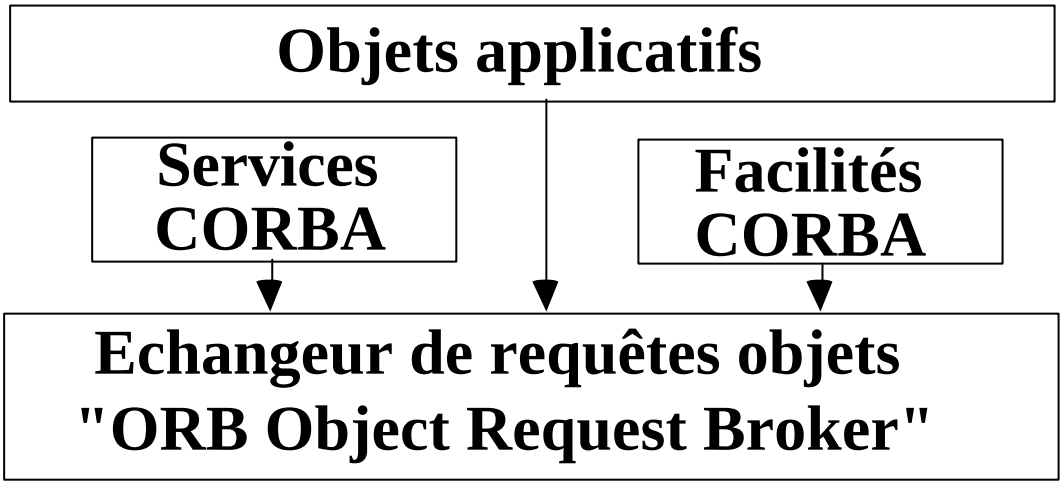
=> Problèmes d'interopérabilité

=> Problèmes de multiplicité des efforts de développement.

L'OMA
"Object Management Architecture"

Un modèle de référence pour une architecture dédiée aux objets répartis.

Environnement cooperatif de composants client-server



CORBA permet à des objets (des composants) d'interagir en univers ouvert

Découvrir la localisation, réaliser les invocations, pour de nombreux langages, environnements d'exécution, réseaux .

La norme définit en plus un ensemble de services

créer, supprimer, nommer, rendre persistants, gérer des transactions etc. entre objets

Les composants orientés systèmes

L'échangeur de requêtes objets "Object Request Broker" ORB

La partie centrale de l'ensemble qui réalise effectivement les interactions entre objets ("**bus à objets**").

Les services systèmes communs CORBA Services "Common Object Services"

Des objets développés en CORBA qui offrent des services systèmes complémentaires à l'ORB (utilisables par invocation ou par héritage).

Exemples: nommage, transactionnel,...

Les composants orientés applications

Les services objets communs
"CORBA facilities"
"Common facilities"

Boite à outils de services d'usage courant nécessaires aux objets d'applications.

Exemples: services d'impressions de documents composites, stockage de documents composites, messageries , ...

Les objets applicatifs
"Objets métiers"
"Applications objects"

Objets conçus pour des besoins particuliers.

Les objets applicatifs collaborent par échange de requêtes pour créer des applications "coopératives" client-serveur (applications réparties, multi-tiers, workflow ...)

L'échangeur de requêtes ORB "Object Request Broker"

Acheminement des requêtes et des réponses

L'ORB décharge une application objet de tout un ensemble de tâches pour l'exécution d'appels de méthodes à distance sur des objets en univers hétérogène:

- localisation des serveurs.
- transport des requêtes et des réponses.
- alignement, présentation des paramètres.
- activation à distance.

Réalisation d'invocations statiques et dynamiques à distance

Invocation statique: les objets appelants et appelés sont connus à la compilation (possibilité de vérification du typage fort).

Invocation dynamique: l'appelant et l'appelé peuvent être créés en exécution et néanmoins peuvent collaborer (possibilité de réaliser une liaison tardive).

Support d'une grande variété de langages

L'ORB permet à des applications en C, C++, JAVA, ADA, Cobol, Smalltalk, ... de coopérer en univers réparti. Les outils essentiels sont:

- Le langage de description d'interfaces IDL Corba ("Interface Definition Langage").

```
interface Personne              {
  readonly attribute unsigned short age;
  attribute string nom;     };
```

- La définition des correspondances ("mappings") entre les types des différents langages et les types IDL.

Exemple: IDL	Java
unsigned short	unsigned short
string	java.lang.string

- La définition d'un format commun de représentation des données CDR.

("**C**ommon **D**ata **R**epresentation")

Support d'une représentation des interfaces

L'ORB gère un référentiel d'interface: une base de données des définitions d'interfaces ("Interface Repository").

ORB et RPC

L'outil central de l'ORB est un protocole de RPC ("Remote Procedure Call").

L'ORB ajoute l'intégration du RPC à un univers d'objets répartis ouvert.

. Interconnexion d'univers objets répartis de types différents développés dans des langages différents.

. Détermination de la localisation d'un objet distant qui implante la méthode. Une requête d'exécution à distance comporte:

- Une référence de l'objet distant:
On s'adresse à un objet particulier.
- Un sélecteur de la méthode appelée
Les messages peuvent donc être "polymorphes" : pour un même message l'opération réalisée dépend de l'objet invoqué.
- Un contexte (optionnel)

Précisant des paramètres nécessaires au contexte de l'exécution (utilisateur pour la protection).

Les services systèmes communs "CORBA Services" "Common object services"

Toujours en cours de développements et d'amélioration.

Le service du cycle de vie "Life cycle service"

Création, déplacement, destruction d'objets (de composants) dans un système d'objets.

Le service de persistance "Persistance service"

Sauvegarde et rechargement d'objets sur des mémoires persistantes (bases de données).

Le service de nommage "Naming service

Politique de désignation et localisation des objets sur les sites d'un système d'objets CORBA.

Les services systèmes communs (suite)

Le service d'événements "Event service"

Service de génération et de distribution d'événements (asynchrones) à des objets qui s'abonnent ou se désabonnent.

Le service de contrôle de concurrence "Concurrency control service"

Gestionnaire de verrous pour réaliser un contrôle de concurrence entre activités ou transactions.

Le service transactionnel "Transaction service"

Service de validation à deux phases pour des unités transactionnelles.

Le service de relation "Relationship service"

Permet de créer des relations entre objets et d'utiliser les liens.

Les services systèmes communs (suite)

Le service d'externalisation "Externalization service"

Permet de transformer un objet en un format standard de type flot pour le sauvegarder ou le transmettre.

Le service d'interrogation "Query service"

Langage d'interrogation de données objets dérivé de SQL et OQL "Object Query Language"

Le service de licences "Licensing service"

Définition de fonctions de contrôle de l'usage d'objets (par création, par site, par session).

Le service de propriétés "Properties service"

Association de valeurs à des objets pour leur associer des propriétés (dates, attribut).

Les services systèmes communs (suite)

Le service de sécurité
"Security service"

Authentification d'accès, confidentialité, non répudiation, protection d'accès par listes de contrôle d'accès, journalisation.

Le service de temps
"Time service"

Obtenir l'heure d'un site distant, calculer un intervalle entre deux événements.

Le service de courtier"
"Trader service"

Service de pages jaunes d'annuaire.

Les utilitaires communs "CORBA Facilities" "Common facilities"

Ensemble d'outils définis sous forme d'objets.

Les utilitaires communs "verticaux"

Définis par métiers (segments de marchés):
domaines: électronique, santé, assurance, ...

Les utilitaires communs "horizontaux"

Définis par fonctions

1 Services d'interface utilisateur

2 Services d'archivage et de transfert de documents composites

Les services définis pour la manipulation des objets intégrés dans des documents composites: affichage, couper/coller, impression :**Opendoc**.

Services de stockage et de transmission.

3 Services d'administration système

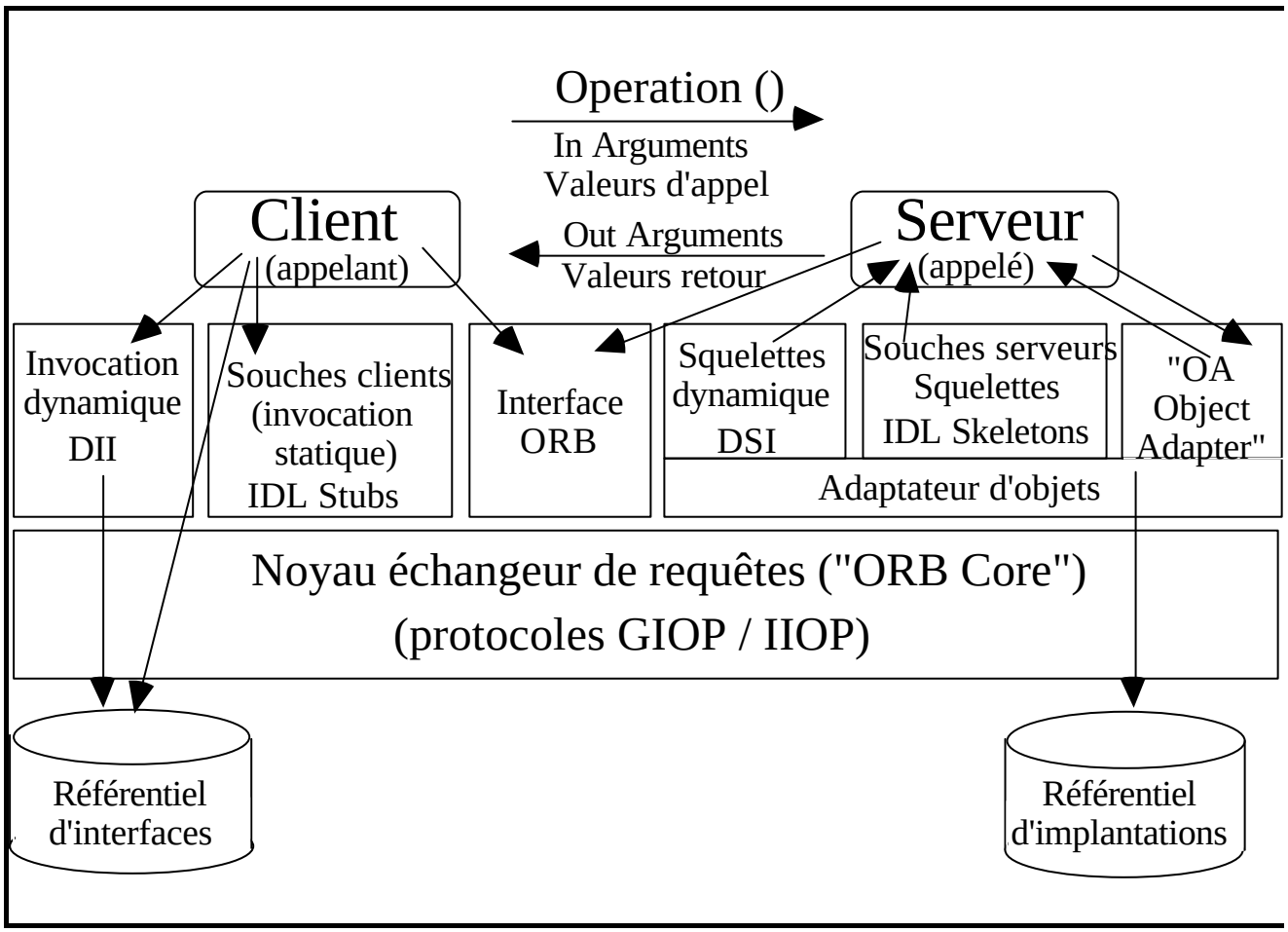
Fonctions habituelles d'administration: détection des pannes, installation, performances.

4 Services d'administration de tâches

Workflow, courrier électronique, ...

Les principaux éléments de la norme CORBA 2.0

La structure de CORBA 2.0



L'interface d'invocation statique
Les souches IDL coté client
"IDL stubs"

Lorsque le client connaît à la compilation toutes les procédures qu'il va utiliser il peut définir les interfaces d'appel et générer des souches clients. Idéalement l'appel distant peut ensuite être réalisé comme s'il s'agissait d'un appel

objet -> opération (arguments)

Rappel: utilisation de souches

Les souches clients sont des procédures d'interface statique pour accéder aux implantations distantes.

Ce sont les souches clients qui effectuent l'empaquetage puis le dépaquetage (marshalling unmarshalling) des paramètres de la méthode invoquée.

Langage l'IDL CORBA ("Interface Definition Language").

Les souches clients sont définies dans l'IDL CORBA puis traduites dans le langage du client par un compilateur.

Exemple de déclaration d'interface en langage IDL

```
module Voiture
{
    /* Définition d'une classe cabriolet */
    interface Cabriolet :
véhicule_thermique ;
    {
    attribute integer consommation ;
    exception Panne (string explication) ;
    void accélère ( in short durée)
        raises (Panne) ;
    void freine ( in short durée)
        raises (Panne) ;
    ....
    }
    interface Monospace :
véhicule_thermique;
    {
    attribute integer place ;
    void accélère ( in short durée)
    ....
    }
} /* Fin de module */
```

L'interface d'invocation dynamique "DII Dynamic Invocation Interface"

Une interface client-serveur générique capable d'envoyer tout type de requête à tout instant vers n'importe quel objet.

Permet en exécution la découverte d'un objet (non connu à la compilation)

- rechercher un objet,
- obtenir sa référence,
- générer un appel (paramètres),
- émettre l'appel
- récupérer les résultats.

Utilisation

En cas de liaison tardive. Exemple: création dynamique d'interface graphique (par clics) et utilisation à distance immédiate

Évaluation de performances

L'ensemble des opérations à réaliser rendent l'invocation dynamique assez coûteuse (plusieurs appels à l'API DII sont nécessaires impliquant d'éventuels échanges de messages). Dans la plupart des cas l'invocation statique est suffisante.

Quelques éléments de fonctionnement de l'interface d'invocation dynamique (1)

Liste des opérations à effectuer

Obtenir la référence de l'objet/méthode,
Créer une liste de paramètres,
Générer le premier paramètre,
...
Générer le dernier paramètre,
Envoyer l'invocation,
Récupérer les résultats.

Quelques requêtes

A) Connaître la méthode appelée get_interface, describe_interface

Pour obtenir la spécification de l'interface de l'objet. Pour obtenir des informations sur les opérations supportées par un objet du référentiel d'interface.

B) Créer une liste des arguments d'appel

create_list Créer la structure de donnée liste d'arguments.

add_arg Pour ajouter un à un chaque argument.

Quelques éléments de fonctionnement de l'interface d'invocation dynamique (2)

C) Créer une requête d'appel distant

create_request

Pour créer un objet requête pour l'opération considérée.
- référence de l'objet distant
- sélecteur de méthode
- liste des arguments

D) Effectuer l'invocation

invoke()

Pour effectuer la requête en RPC.

Le référentiel d'interface
IR "Interface Repository"

Un élément essentiel de l'ORB c'est la base de données d'interfaces des objets enregistrés.

Le référentiel d'interface est consulté pour connaître la description des opérations qu'une interface supporte et les liens d'héritage entre interfaces.

Usager : Découverte de composants réutilisables (en cas de liaison dynamique)

ORB : Vérification de conformité d'appel.

Des objets de type InterfaceDef (descripteurs d'interfaces IDL) sont stockés dans une librairie.

Quelques éléments de l'API

Object::get_interface

Retourne une référence sur un objet InterfaceDef qui décrit une interface d'objet.

InterfaceDef::describe_interface

Pour obtenir des informations sur les opérations supportées par un objet.

L'interface de l'ORB
"ORB Interface"

Différents styles d'implantations possibles des ORB:

- librairie de fonctions
- un ou plusieurs processus

CORBA spécifie l'interface de l'ORB de manière à être indépendante des détails d'implantations.

Exemple de types de fonctions:

Initialisation de l'ORB

Conversion des références d'objets en chaînes et vice versa.

Référentiel d'implantations "Implementation repository"

La base de données des classes (des implantations d'objets) supportées, des objets instanciés et de leurs références objets.

Autres informations gérées:
suivi, audit, sécurité, administration.

Squelette statique "IDL Skeleton"

En langage CORBA, c'est la procédure souche coté serveur lors d'une liaison statique.

Interface de liaison dynamique "DSI Dynamic Skeleton Interface"

C'est la partie souche coté serveur lors d'une invocation dynamique.

Il réalise la liaison avec l'objet et la méthode cible.

Les adaptateurs d'objets "OA Object Adapter"

Fonctions principales

Assister l'ORB dans la délivrance des requêtes d'accès aux objets coté serveur.

Permettre l'exécution d'une méthode d'un objet (constituer l'environnement) : instancier, assigner des références objets.

Enregistrer les objets dans le référentiel d'implantation.

Éviter la multiplication des variantes d'adaptateurs d'objets.

L'adaptateur de base BOA ("Basic Object Adapter")

La norme CORBA spécifie une version minimale d'adaptateur (le BOA).

a. Il gère le référentiel d'implantations (enregistrer et accéder aux implantations).

b. Il génère et interprète les références.

c. Il active et désactive les implantations.

d. Il reçoit les requêtes aux méthodes et les transmet via la souche serveur (‘skeleton’).

e. Il authentifie les clients.

Adaptateurs de base d'objets et gestion du parallélisme

1. Serveur partagé (BOA "Shared Server")

Un seul processus pour tous les objets.
Les requêtes sont traitées en séquence.

2. Serveur non partagé (BOA "Unshared Server")

Un processus créé par objet.
Possibilité de parallélisme entre objets.

3. Serveur par méthode (BOA "Server-Per-Method")

Un processus par requête.
Possibilité de parallélisme entre méthodes

4. Serveur persistant (BOA "Persistent Server")

Plusieurs processus sont créés au départ.
L'OA fonctionne en serveur partagé pour chaque processus.
Possibilité d'un niveau de parallélisme égal au nombre de processus serveurs.

Différents autres adaptateurs d'objets

Des adaptateurs d'objets spécialisés supportent certains styles d'implantation des objets.

Bases de données orientées objets

Utilisable pour la gestion des objets à grain fin et pour la persistance.

Disposent d'une interface spécifique de chaque base orienté objet.

Adaptateurs d'objets pour objets locaux

Pour éviter d'utiliser des techniques d'invocations distantes coûteuses pour des objets locaux.

Conclusion Objets Répartis
- CORBA -

Un outil très important dans les applications client serveur

Approche de plus en plus répandue de la programmation orientée objet: notion de programmation par "composants"

Idée de programmation indépendante des composants et de réutilisation.

Le système d'objets répartis est la clé de voûte de cette approche.

La normalisation des systèmes d'objets répartis: CORBA
Nombreuses implantations

Une bataille en cours entre l'approche CORBA et l'approche Microsoft

Bibliographie

Nombreuses pages WEB
http://www.omg.org/