Introduction à DCOM

S. Krakowiak

Université Joseph Fourier
Laboratoire Sirac (INPG-INRIA-UJF)

http://sirac.imag.fr/~krakowia

DCOM : solution Microsoft pour applications client-serveur réparties

- **Produit "propriétaire"**
  - **Intégré avec d'autres produits Microsoft (architecture DNA : Distributed Network Architecture)**
    - **COM (Component Object Model) : modèle d'objets, non réparti**
    - **ActiveX : applications sur le Web (accès via un navigateur)**
    - **MTS (Microsoft Transaction Server) : transactions**
    - **MQS (Message Queue Server) : messages**
  - **Fonctionne sur plusieurs plates-formes**
    - **Intégré à Windows NT (usage principal)**
    - **Windows 95**
    - **Unix, autres**

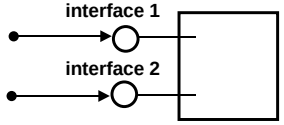
DCOM : Motivations

- **Développement et évolution d'applications réparties sur des machines hétérogènes**
  - **Mécanismes "transparents" d'appel à distance**
- **Réutilisation de code**
  - **Réutilisation de composants binaires exécutables**
    - **Pas d'accès au code source**
    - **Association d'interfaces multiples à un composant binaire**
  - **Mécanismes de composition**
    - **Construction de composants à partir de composants existants**
- **Évolution dynamique**
  - **Remplacement de composants (sans recompilation ou édition de liens)**
  - **Association de nouvelles interfaces**
  - **Capacité d'auto-description**

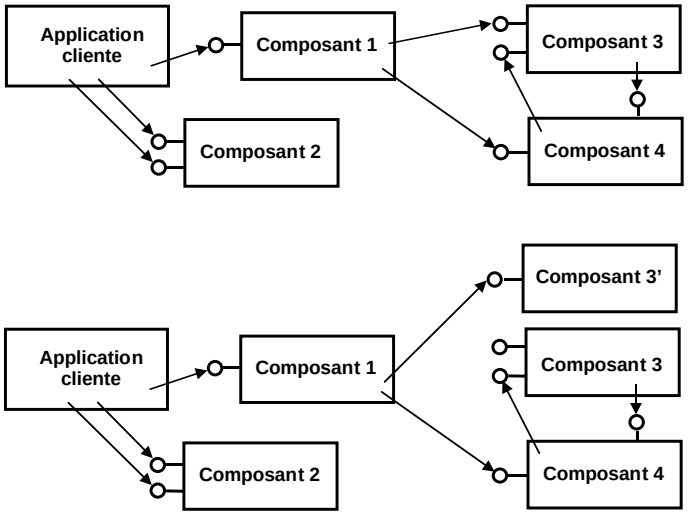
COM : notions de base

Objets et interfaces sont des notions séparées

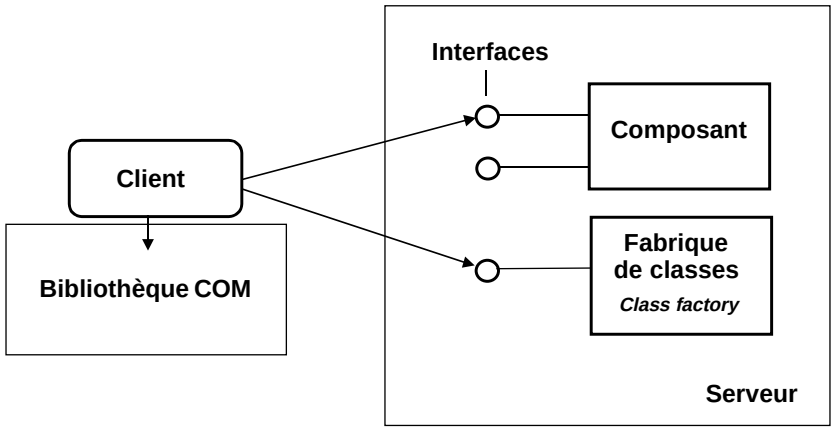
- **Objets (ou composants)**
  - **Un objet encapsule un binaire (DLL)**
  - **L'utilisation de l'objet est indépendante du langage initialement utilisé pour engendrer le binaire**
- **Interfaces**
  - **Pour accéder à un objet, il faut avoir accès à une interface associée à cet objet**
  - **Plusieurs interfaces peuvent être associées à un objet**



COM : modification dynamique d'une application



Entités visibles dans COM



Interfaces

- **Définition**
  - Interface = ensemble de fonctions permettant d'utiliser un composant
    - Ne comporte aucune implémentation
    - Description = ensemble de signatures
- **Propriétés**
  - Une interface est immuable
    - Identification globale unique
    - Pas de modifications, mais création d'une nouvelle version
  - Une interface peut hériter d'une autre interface
    - Héritage simple

Contraintes imposées à une interface

- **Format standard en mémoire**
  - Structure composée d'un tableau de pointeurs vers des fonctions
  - Conforme au standard de la table de fonctions virtuelles (*vtable*) de C++
- **Doit hériter d'une interface spécifiée : *IUnknown***
  - *IUnknown* comporte 3 fonctions
    - *Query Interface ()*
      - Permet de découvrir d'autres interfaces (auto-description)
    - *AddRef ()*
    - *Release ()*
      - *AddRef* et *Release* servent à la gestion de la mémoire (libération automatique des composants non utilisés)

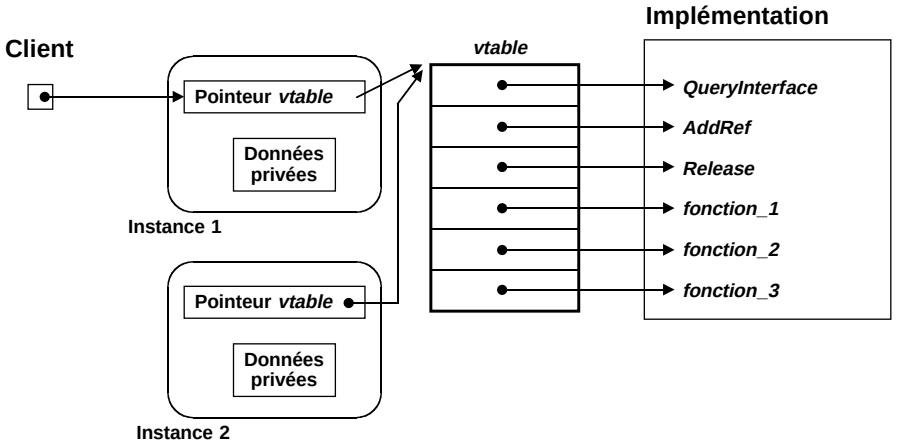
Classes et instances

- **Classe COM**
  - Mise en œuvre d'une ou de plusieurs interfaces
  - Réalisée concrètement par une liste de DLL contenant le code (*Dynamically Loadable Libraries*)
- **Objet COM (ou instance)**
  - Instancié à partir d'une classe
  - Utilise la mise en œuvre fournie par la classe
  - Définit un espace de variables propres
  - Les instances d'une même classe partagent l'implémentation

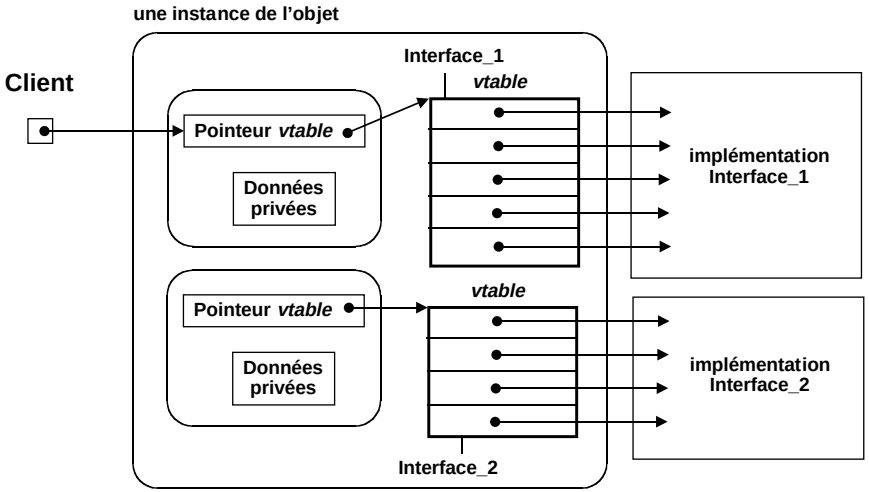
Différence avec langages à objets : l'intégration du code se fait au niveau du binaire, non du langage source

Interfaces, classes et instances

Toutes les instances d'une classe
- partagent la même interface et la même implémentation
- ont chacune ses propres données privées

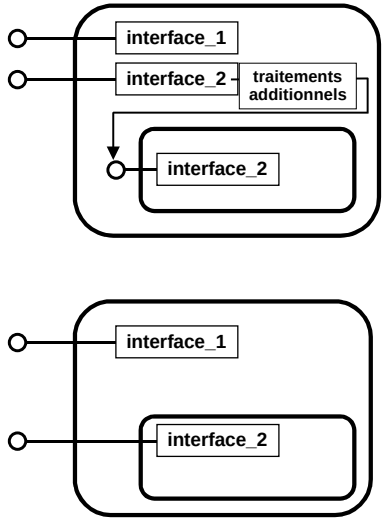


Objet comportant plusieurs interfaces



Composition d'interfaces

- **Délégation**
  - Un objet "externe" utilise les fonctions d'un objet "interne" pour mettre en œuvre ses propres opérations
- **Agrégation**
  - Un objet "externe" exporte les fonctions d'un objet "interne" pour les rendre accessibles en plus des siennes propres
- Ces mécanismes permettent de s'affranchir des limitations imposées par l'héritage simple des interfaces



L'interface *IUnknown*

- **Une fonction d'autodescription : *QueryInterface***
  - Permet de déterminer si une interface donnée (IID) est implémentée par un composant
  - Pdermet de naviguer entre les différentes interfaces du composant
- **Deux fonctions de gestion des références**
  - ***AddRef*** : appelée quand un nouveau pointeur est fourni vers une interface, ou quand un pointeur est copié ; incrémente compteur de références
  - ***Release*** : appelée quand on cesse d'utiliser un pointeur vers une interface ; décrémente compteur de références
  - La mémoire d'un composant est libérée quand le dernier compteur d'interface passe à 0

Désignation interne

- **GUID (*Global Unique Identifier*)**
  - Système de désignation par identificateur unique dans l'espace et le temps
    - 128 bits, construit à partir des caractéristiques de la machine et de l'heure absolue d'exécution (algorithme DCE)
    - Exemple : {11943941-36DE-11CF-953E00C0A84029E9}
  - Toute interface ou classe a un GUID
    - IID pour les interfaces
    - CLSID pour les classes (implémentations d'interfaces)
- **Pas de désignation directe des objets**
  - IID -> classe réalisant l'interface
  - CLSID -> création d'un objet à partir de la classe

La base de registre (*Windows Registry*)

- **Base de données permettant d'associers noms et GUID**
  - CLSID -> nom de fichier contenant le code de l'implémentation de la classe
    - {0003DF… 098} --> C:\ProgFiles\Internet\iexplore.exe
  - Nom familier (ProgId) --> CLSID
    - Explorer.Appli_2 --> {0003DF… 098}
- **Autres fonctions de désignation et recherche**
  - Informations sur configurations, usagers, etc.
  - Informations utiles pour la répartition (voir DCOM)

La bibliothèque COM

- **La bibliothèque COM fournit une API pour la gestion des composants**
  - Initialisation et terminaison
  - Allocation et libération de mémoire
  - Génération de GUID (identificateurs uniques)
  - Accès à la base de registre (GUID <--> noms)
  - Création de composants
  - Enregistrement et recherche de fabriques de classes
  - Bibliothèques pour emballage et déballage de paramètres

Les étapes de l'appel d'un composant

- **Créér un composant**
  - Composant identifié par son CLSID
- **Obtenir un pointeur d'interface**
  - interface identifiée par son IID
- **Appeler une méthode (fonction de l'interface)**
- **Terminer**
  - Gestion du compteur de références pour libération

Création d'un composant

- **Étapes**
  - Spécifier le CLSID du composant (obtenu par ailleurs)
  - Créer une fabrique pour la classe spécifiée par ce CLSID
  - Obtenir un pointeur vers l'interface *IClassFactory* de la fabrique
  - Appeler *IClassFactory::CreateInstance()*
  - Le composant est créé
- **Fonctions utilisées dans la bibliothèque COM**
  - *CoCreateInstance(CLSID, …, IID_IUnknown, …)*

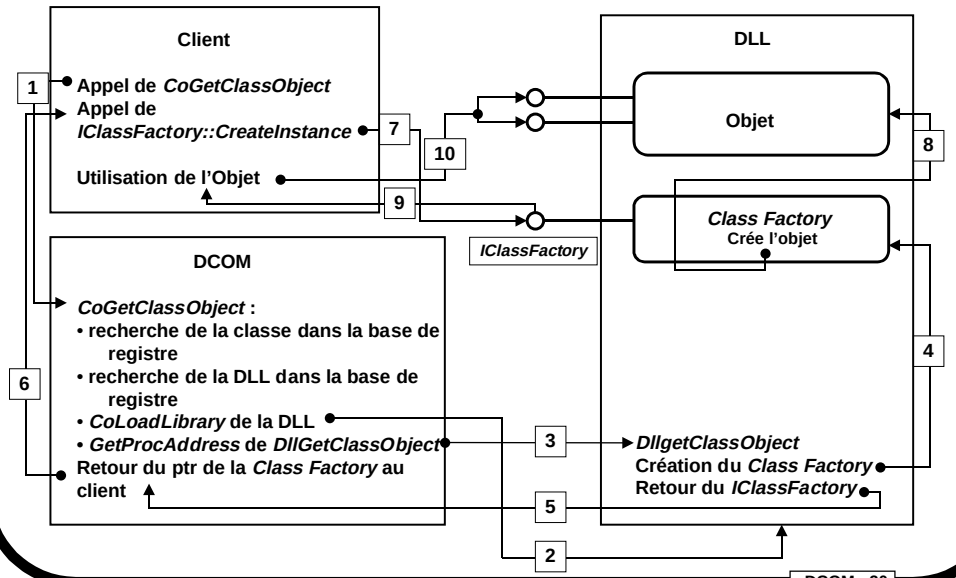
    crée un pointeur vers l'interface d'une instance (non initialisée) de la classe spécifiée
    - Étapes 1 à 5
  - *CoGetClassObject(CLSID, …, IID_IClassFactory, …)*

    crée un pointeur vers l'interface d'une fabrique de classes
    - Étapes 1 à 3

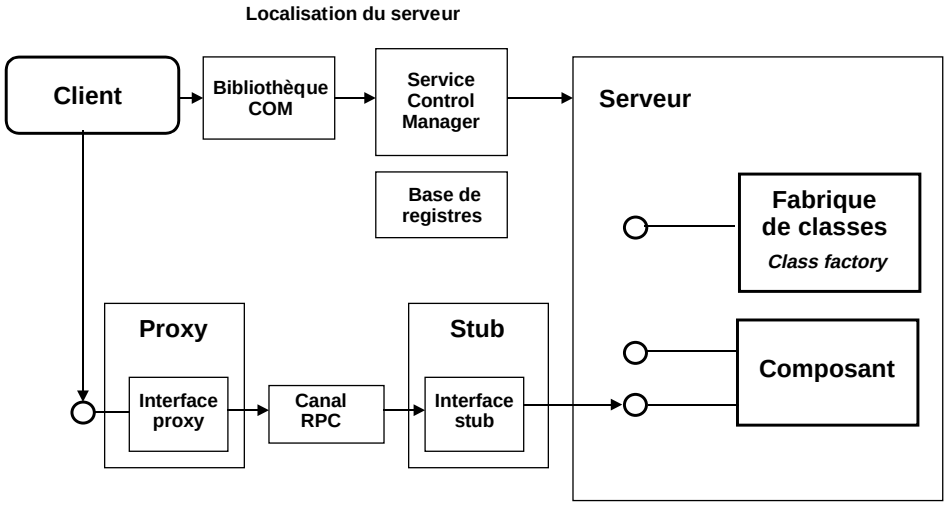
Obtenir un pointeur d'interface

- **Étapes**
  - Fournir l'identificateur de l'interface (IID)
  - Obtenir un pointeur vers cette interface
    - Comme résultat de la création
    - À partir de l'auto-description du composant choisi (interface *IUnknown*)
- **Utilisation de IUnknown**
  - *IUnknown::QueryInterface (iiD, &pointer)*
  - Si l'interface cherchée existe, fournit un pointeur vers cette interface et renvoie "succès"
    - Appelle automatiquement *AddRef* (nouveau pointeur)
  - Sinon renvoie "échec"
  - Pour une instance donnée, renvoie toujours le même pointeur (permet d'identifier l'instance)

Création d'un serveur "*in process*"



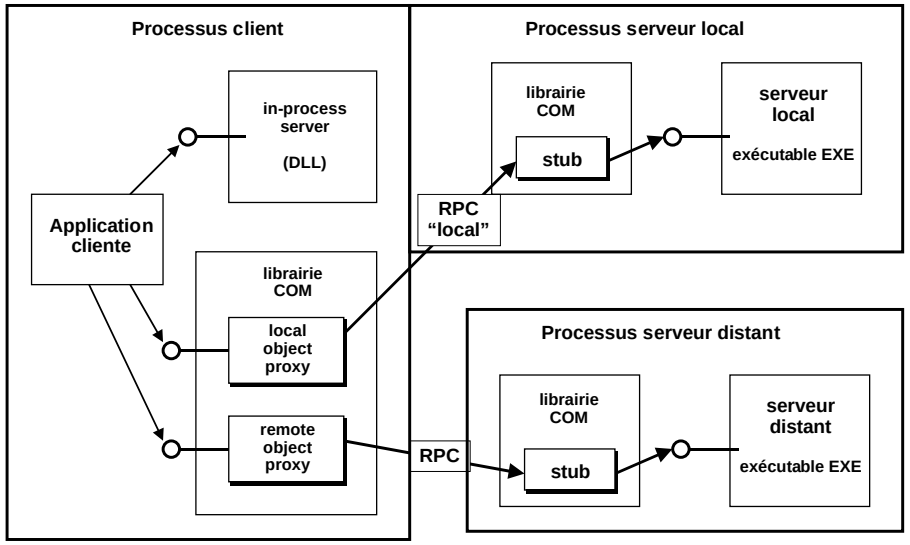
DCOM : réalisation répartie de COM



Relation client-serveur dans DCOM

- **Trois types de relations selon localisation du serveur par rapport au client**
  - **Même processus**
    - **DLL chargée dans l'espace d'adressage du client**
    - **Appel direct**
  - **Processus différent, même machine**
    - **EXE sur la même machine**
    - **Appel via schéma LPC ("Local Procedure Call")**
  - **Processus distant**
    - **EXE sur machine distante**
    - **Appel via schéma RPC ("Remote Procedure Call")**

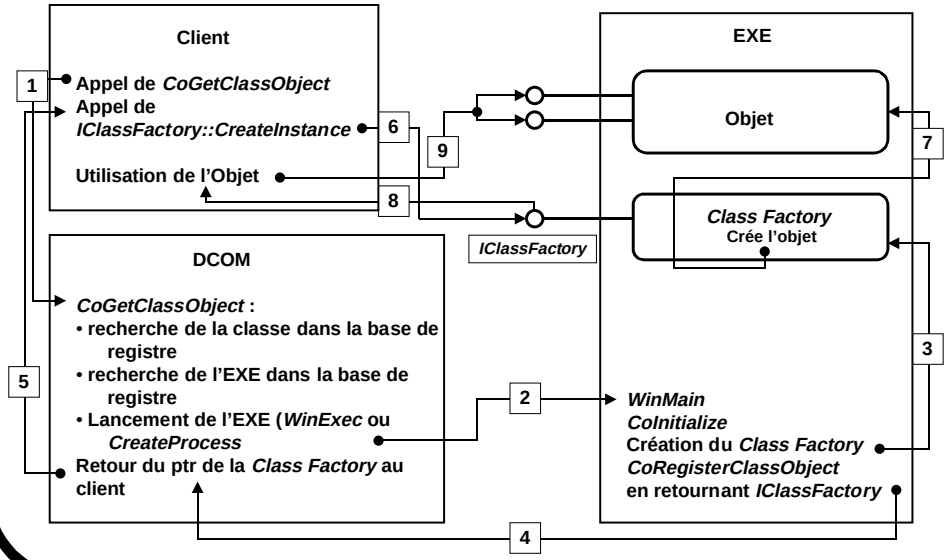
Relation client-serveur dans DCOM



Service Control Manager (SCM)

- **Fonction : localisation des serveurs**
  - **Existe sur chaque machine DCOM**
  - **Connaît les serveurs situés sur cette machine**
  - **Connaît pour chaque serveur les instructions nécessaires pour le lancer**
- **Recherche d'un composant**
  - **On cherche le composant dans la base de registre**
  - **S'il n'est pas disponible localement, la base contient l'identité du serveur où il réside**
  - **Le SCM local contacte le SCM de la machine distante**
  - **Le SCM distant trouve et lance le composant (via la fabrique)**

Création d'un serveur "*out process*"



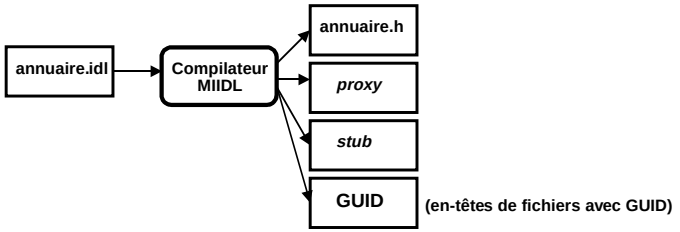
Exemple d'application

- **Étapes de la construction**
  - **Définition des interfaces des objets**
    - **Langage de définition d'interfaces MIDL**
    - **Génération des talons**
  - **Mise en œuvre du composant DCOM**
  - **Enregistrement des objets**
  - **Mise en œuvre du client**
- **Un exemple : l'application annuaire**

Définition des interfaces

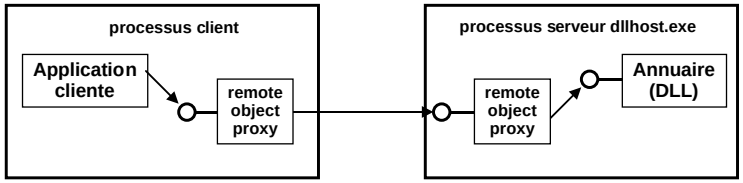
```
import "unknwn.idl";
        [ object, uuid(10000001-0000-0000-0000-000000000001)]
        // engendré par un générateur de GUID (COM library)
        interface ILookup : IUnknown{
            HRESULT Lookup([in, string] char* nom, [out, string] char** email);
        }
        [ uuid(10000003-0000-0000-0000-000000000001),
            helpstring("Annuaire Component Type Library"), version(1.0) ]

        // définit ce qui pourra être appelé dynamiquement par Automation
        library AnnuaireLib{
            importlib ("stdole32.tlb") ;
            interface ILookup ;
            [ uuid(10000002-0000-0000-0000-000000000001) ]
            coclass Annuaire {
                interface ILookup;
            }
};
```


Génération des talons



Mise en œuvre du composant

- Choix du type de composant (EXE ou DLL)
  - Intérêt du DLL : réutilisabilité
  - Mais si distant, doit être encapsulé dans un EXE (pour fonctionner comme serveur autonome)



Écriture du composant
Mise en œuvre des méthodes de *IUnknown*
QueryInterface, *AddRef*, *Release*
Méthodes de *ILookup* (méthodes spécifiques)

Méthode IUnknown::QueryInterface

```
HRESULT CAnnuaire::QueryInterface (REFIID riid, void** ppv) {
        if (riid == IID_IUnknown) {
           *ppv = (IUnknown*) this ;
        }
        else if (riid == IID_ILookup) {
           *ppv = (ILookup*) this ;
        }
        else {
           *ppv = NULL ;
           return E_NOINTERFACE ;
        }
        AddRef () ;
        return (S_OK) ;
}
```

Si l'IID fourni est celui d'une des interfaces réalisées par le composant, rendre un pointeur sur cette Interface (et incrémenter le compte de références), sinon signaler échec

Peut en fait être créée automatiquement (cf. plus loin)

Méthode IUnknown::AddRef / Release

```
ULONG CAnnuaire::AddRef () {

        return ++m_cRef ;     // incrémente compteur de références
}
```

```
ULONG CAnnuaire::Release () {

        if (--m_cRef != 0)
           return m_cRef ;     // décrémente compteur de références
        delete (this) ;        // et supprime le composant
        return (0) ;           // si le compteur passe à zéro

}
```


Interface ILookup

```
<définition des données internes>


HRESULT __stdcall Cannuaire::Lookup (wchar_t* pcle, wchar_t** presultat) {
        int i = 0 ;
        while annuaire [i] != NULL {
                if (wcscmp (pcle, annuaire{i}.cle == 0) {
                        *presultat = &annuaire{i}.email ;
                        return S_OK ;
                {
                i++ ;
        }
        return S_FAIL ;
}
```

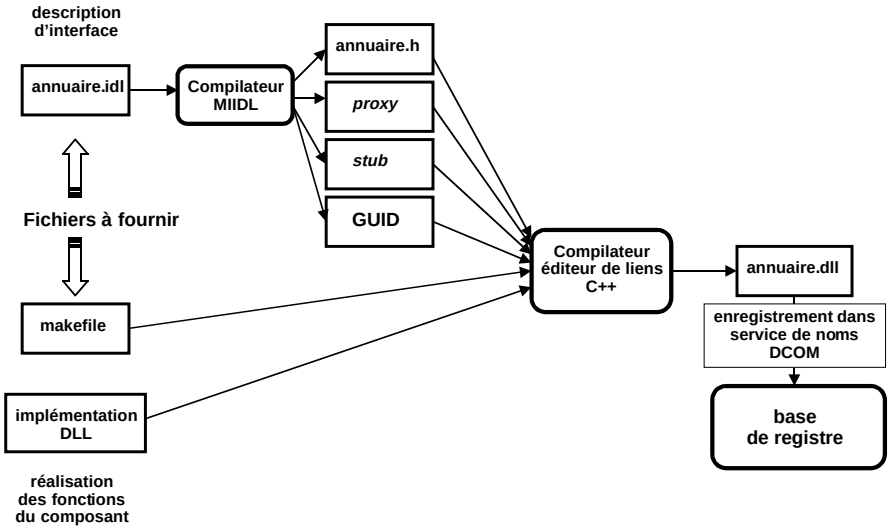
Compare la clé de recherche aux clés successives des entrées de l'annuaire (tableau) et renvoie le champ "email" en cas de succès.

Réalisation de la fabrique

◆ **Classe CFactory dédiée à la création de l'objet COM Annuaire**

```
interface IClassFactory : IUnknown {
   HRESULT CreateInstance ([in, unique] IUnknown *pUnkOuter,
          [in] REFIID riid,
          [out, iid_is(riid)] void **ppvObject) ; // iid_is convertit le pointeur vers le type correct

HRESULT LockServer ([in] BOOL fLock) ;


HRESULT CFactory::CreateInstance (IUnknown *pUnkOuter,
          REFIID riid, void **ppvObject)  {

   if (pUnknownOuter != NULL)
          return CLASS_E_NOAGGREGATION ;

   CAnnuaire *pAnnuaire = new CAnnuaire () ;
   if (pAnnuaire == NULL)
          return E_OUTOFMEMORY ;

    HRESULT hr = pAnnuaire->QueryInterface (riid, ppv) ;
   pAnnuaire->Release () ;
   return hr ;
}
```


Chaîne de production



Mise en œuvre du client

❖ **Étapes**

- ▲ Initialiser la bibliothèque COM (CoInitialize)
- ▲ Créer une instance de l'annuaire (CoCreateInstance) } appelle CoGetClassObject crée l'objet Factory et renvoie son pointeur au client appelle IClassFactory:CreateInstance
- ▲ Trouver l'interface (QueryInterface)
- ▲ Appeler la fonction désirée (Lookup)
- ▲ Fermer la bibliothèque (CoUnitialize)

```
void main () {
        HRESULT hr = CoInitialize(NULL);
        IUnknown* pUnknown;
        ILookup* pLookup;
        wchar_t * email;                                          non utilisé

        hr = CoCreateInstance (CLSID_Annuaire, NULL, CLSCTX_LOCAL_SERVER,
                IID_IUnknown, (void**) &pUnknown);

        hr = pUnknown->QueryInterface (IID_ILookup, (void**) &pLookup);
        if (FAILED(hr)) cout << "IID_ILookup not supported" << endl;
        hr = pUnknown->Release();

        hr = pLookup->Lookup("L. Bellissard", &email);
        if(SUCCEEDED(hr))
                cout << "email : " << email << endl;
        hr = pLookup->Release();
        CoUninitialize();
}
```


Utilisation pratique

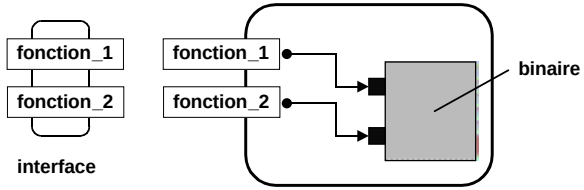
En réalité, la plupart des opérations précédentes peuvent être automatisées, et les programmes et structures de données sont invisibles à l'utilisateur

■ **Utilisation de l'outil ATL (*Active Template Library*)**

- ◆ L'outil ATL Wizard permet
  - ❖ de choisir le type de composant (DLL, EXE)
  - ❖ de définir les interfaces
- ◆ Puis, génération automatique
  - ❖ des CLSIDs
  - ❖ du fichier IDL
  - ❖ de la bibliothèque de types
  - ❖ de la classe de la fabrique
  - ❖ d'un cadre pour la classe du composant (Annuaire)
- ◆ Reste à écrire
  - ❖ la réalisation de la fonction *Lookup*

Réutilisation d'un binaire existant

- Créer un objet COM "enveloppe"
  - Définir une interface correspondant aux fonctions accessibles
  - Dans l'implémentation
    - Fournir les fonctions de l'interface
    - Pour chaque fonction, appeler le point d'entrée correspondant de l'objet binaire, avec les paramètres fournis
    - Récupérer les résultats éventuels
  - Mettre le composant sous forme EXE pour permettre son utilisation à distance



Autres services

- Nombreux autres services disponibles
  - Automation : interface *IDispatch*
    - permet à un client de fabriquer un appel à un objet COM ou DCOM (analogue au DII de CORBA), sans *proxy* sur le site client
    - utilise les bibliothèques de types pour autodescription
    - très souple, mais lent
  - ObjectViewer
    - permet la navigation parmi les objets, avec visualisation
  - Désignation
    - service de "surnoms" (*monikers*)
    - objet servant à nommer et activer d'autres objets
    - connaît la manière de lier le client appelant à l'objet désigné
    - réalisations standard pour fichiers, classes, URL, etc.

Conclusion sur DCOM

- Produit "propriétaire"
  - Lié à l'environnement Microsoft
    - Environnement de développement spécifique
    - Intégration aux autres outils
      - Office, Serveur Web IIS, Internet Explorer, etc.
  - Interconnexion possible avec CORBA
- Outil puissant, mais concepts difficiles à appréhender
  - Spécification opaque
  - Héritage de OLE
- Nombreux composants diffusés
  - La plupart des applications Windows sont des objets COM

Références

D. Rogerson, *Inside COM*, Microsoft Press, 1997

G. Eddon, H. Eddon, *Au cœur de Distributed COM*, Microsoft Press France, 1998

R. Grimes, *Professional DCOM Programming*, Wrox Press, Canada, 1997

Page de pointeurs
http://www.sente.ch/cetus/OO_ole.html

Comparaisons CORBA-DCOM
http://research.microsoft.com/~ymwang/papers/HTML/DCOMnCORBA/S.html

http://www.execpc.com/~gopalan/misc/compare.html