Sistemas de Información

Tecnologías de objetos distribuidos: OMG-CORBA. Invocación dinámica

Agradecimientos: Juan Pavón Mestras (UCM), Jesus Villamor Lugo y Simon Pickin (UC3M), Juan José Gil Ríos (Terra)

CORBA

DII (*Dinamic Invocation Interface)*

- DII
  - Motivación
  - ¿Qué es?
  - ¿Para qué sirve?
  - ¿Cómo se usa?
  - ¿Cómo funciona?
- Repositorio de interfaces
  - ¿Qué es?
  - ¿Para qué sirve?
  - ¿Cómo se usa?
  - ¿Cómo funciona?
- Ejemplos

CORBA *Invocación dinámica*
Motivación

- El problema:
  - El modelo basado en stubs tiene algunas restricciones
    - Los stubs son generados en tiempo de compilación
    - Un cambio en el IDL implica rehacer todos los clientes
  - Hay algunas aplicaciones (navegadores, depuradores, intérpretes) que requieren flexibilidad mayor porque:
    - No saben de antemano que interfaz tendrán los objetos a los que accederán
    - Necesitan la existencia de mecanismos que les permitan:
      - Descubrir esas interfaces
      - Crear de forma dinámica la petición de operación
- La solución: CORBA ofrece una solución basada en:
  - El repositorio de interfaces: IR *(**I**nterface **R**epository)*
    - Guarda la descripción de las interfaces
  - La interfaz de invocación dinámica: DII *(**D**inamic **I**nvocation **I**nterface)*
    - Permite construir en tiempo de ejecución una petición de una determinada operación

CORBA

DII (*Dinamic Invocation Interface*)

- ¿Qué es?
  - Interfaz Corba que permite la construcción dinámica de invocaciones en tiempo de ejecución para un determinado objeto remoto
- ¿Para qué sirve?
  - Para dar más flexibilidad al cliente ya que no necesita conocer el interfaz de los objetos a los que va a invocar en tiempo de compilación puede:
    - descubrir el interfaz de los objetos
    - construir la invocación

    en tiempo de ejecución

CORBA

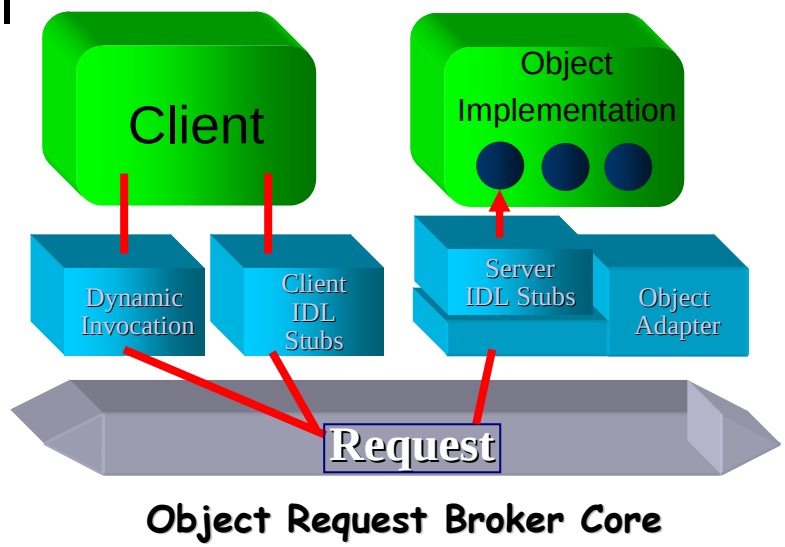
DII (*Dinamic Invocation Interface)*

- ¿Cómo se usa?
  - En vez de utilizar una llamada a una función determinada de un objeto concreto, que el cliente puede especificar a través de una llamada o conjunto de ellas:
    - el objeto
    - la invocación
    - los parámetros a pasar a la invocación
  - Una invocación dinámica se compone de:
    - Una referencia al objeto,
    - Una operación
    - Una lista de parámetros
  - Todos estos datos se obtienen del Repositorio de Interfaces (IR)
  - De cara al servidor la invocación es idéntica a una que llega a través de la interfaz estática, pero dentro del cliente se logra una flexibilidad fundamental en arquitecturas complejas y dinámicas.

CORBA *(Invocación dinámica)*

¿Cómo funciona?

- La interfaz del servidor se descubre en tiempo de ejecución, utilizando el repositorio de interfaces
- La petición se construye con la interfaz de invocación dinámica DII



- El Cliente no necesita *Stubs de compilación*
- El Servidor percibe la misma llamada

CORBA **IR (*Interface Repository*)**

- ¿Qué es?
  - Un **repositorio de información** que permite:
    - Localizar los objetos en tiempo de ejecución
    - Activar la implementación de los objetos (esta funcionalidad ahora la realiza el POA)
  - Es un **servicio** estándar CORBA disponible a través del ORB que ofrece objetos persistentes que representan la información IDL de los interfaces disponibles en CORBA, de una forma accesible en tiempo de ejecución (run-time)

CORBA **IR (*Interface Repository*)**

¿Para qué sirve?

- Proporciona información de los objetos en tiempo de ejecución puede ser utilizado:
  - Por el ORB para realizar peticiones.
  - Por el Programador de aplicaciones
    - Para acceder a objetos cuya interfaz no se conocía en tiempo de compilación
    - Para determinar que operaciones son válidas para un objeto.
- Otras funcionalidades de IR:
  - Instalación de implementaciones
  - Control de las políticas para la activación y ejecución de las implementaciones de los objetos. Por ejemplo, los permisos por usuario para acceder e invocar los objetos son especificados aquí.
  - Con la introducción de POA en CORBA 2.2, las políticas de activación y ejecución se localizan ahora dentro de POA, en el propio código del servidor en lugar de hacerlo en IR

CORBA **IR (*Interface Repository*)**

¿Cómo funciona?

- Hay que obtener una referencia al Servicio proporcionado por el ORB
  - El servicio IR proporciona una base de datos de definiciones de objetos
    - Cada objeto representa una definición IDL
    - Los objetos están anidados en una estructura que refleja las definiciones IDL
    - Es posible navegar y buscar definiciones en dicha estructura
  - Las definiciones pueden suministrarse al Repositorio de Interfaces:
    - Usando una utilidad (irfeed en ORBacus, putidl en OrbixWeb, idl2ir en Visibroker)
    - Desde cualquier programa usando las funciones de escritura de la interfaz del Repositorio de Interfaces
  - Se puede acceder al Repositorio de Interfaces como un servicio bien definido (se accede al objeto Repository raíz de la jerarquía):

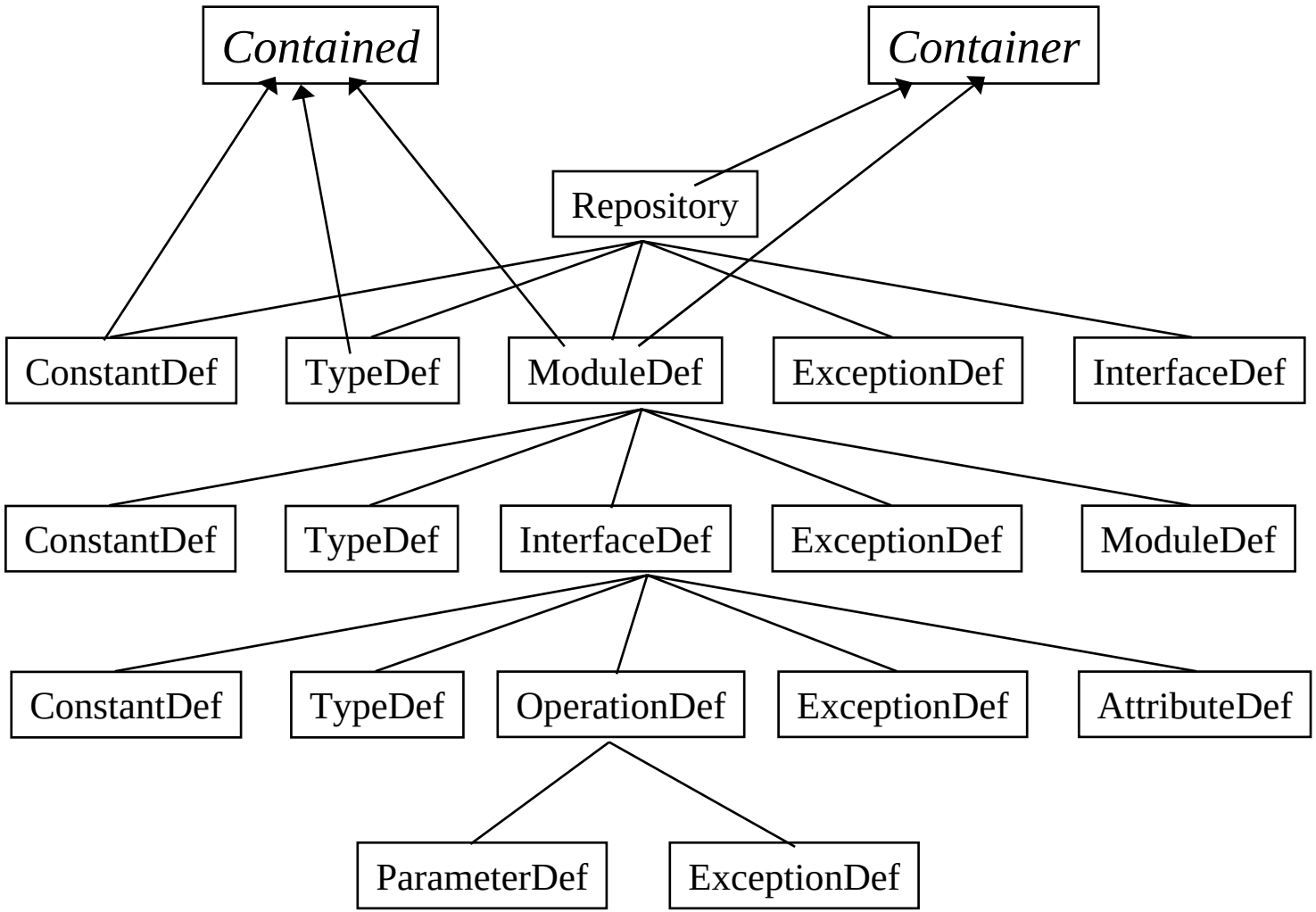
```
org.omg.CORBA.Object obj = orb.resolve_initial_references("InterfaceRepository");
org.omg.CORBA.Repository intfRepository = org.omg.CORBA.RepositoryHelper.narrow(obj);
```

CORBA **IR (*Interface Repository*)**

¿Cómo funciona?

- Está implementado como un conjunto de objetos persistentes (metadatos) que representan la información que contiene (un objeto por cada tipo de construcción IDL)
  - ModuleDef
  - InterfaceDef
  - OperationDef
  - ParameterDef
  - AttributeDef
  - ConstantDef
  - ExceptionDef
  - TypeDef
- además del objeto
  - Repository, que contiene a todos

CORBA **IR (*Interface Repository*)**

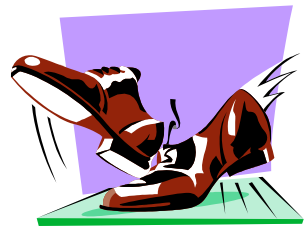


CORBA
Invocación Estática vs Dinámica

- Invocaciones Estáticas
  - Más fáciles de programar
  - Chequeo de tipos más robusto
  - Buen rendimiento
  - Autodocumentado
- Invocaciones Dinámicas
  - Sistema más extensible
  - Soporte de búsqueda de servicios en *“run-time”*
  - Permite escribir código más genérico

CORBA: *Invocación Dinámica*

Procedimiento



1. Partiendo del IR obtener:
   - la interfaz
   - la descripción del método (si no se utiliza create_operation_list)
2. Crear y configurar el objeto ***request***. (2 formas):
   - crear una lista de argumentos ***NVList*** y luego un objeto ***request***
   - crear un objeto ***request*** y luego añadirle los argumentos
4. Solicitar al ORB que envie la petición al objeto destino con ***invoke*** (3 formas de invocar la petición):
   - Síncrona
   - síncrona diferida
   - asíncrona
5. Recoger los resultados (con ***return_value*** del objeto ***request)***
6. Liberar los recursos (del objeto ***request*** y la ***NVList*** si existe)

CORBA: *Invocación Dinámica*

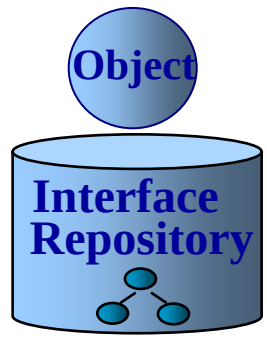
Procedimiento

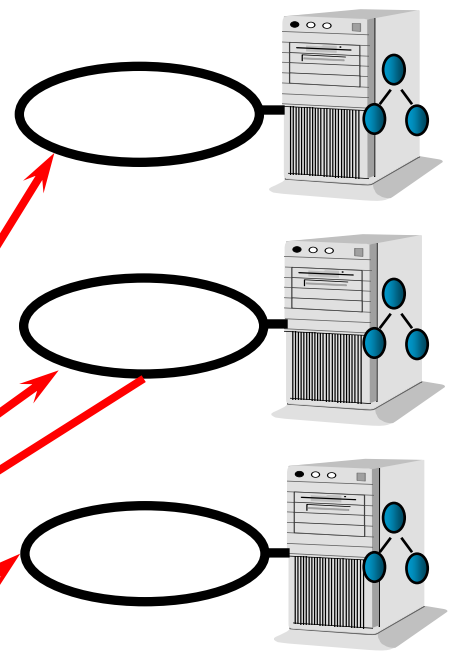
Client Object

1. **Obtener el nombre de la interfaz**
   - `get_interface()`
2. **Obtener la descripción del método**
   - `lookup_name ()`
   - `describe ()`
3. **Crear la lista de argumentos**
   - `create_list ()`
   - `add_item ()...add_item ()...add_item ()`
4. **Crear la petición**
   - `create_request ()`
5. **Invocar el método remoto**

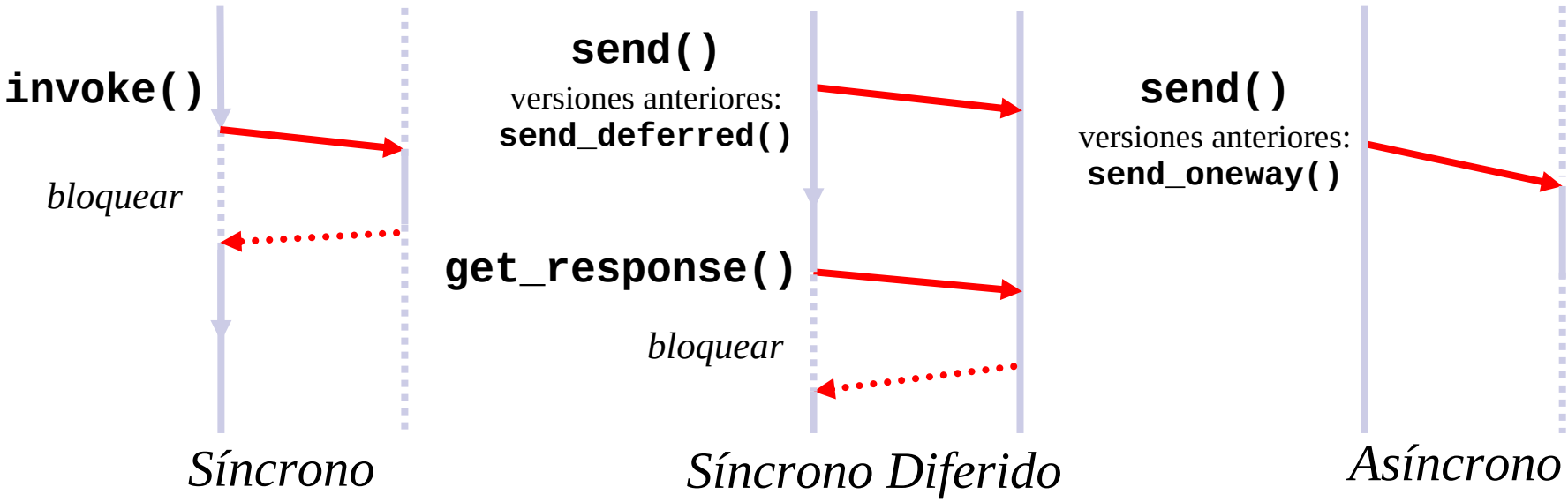
3 formas de hacer la invocación remota

- *invocación síncrona:* `invoke ()`
- *o bien*
- *síncrona diferida:* `send()` / `get_response ()`
- *o bien*
- *asíncrona (“oneway”):* `send ()`





Modelos de Interacción

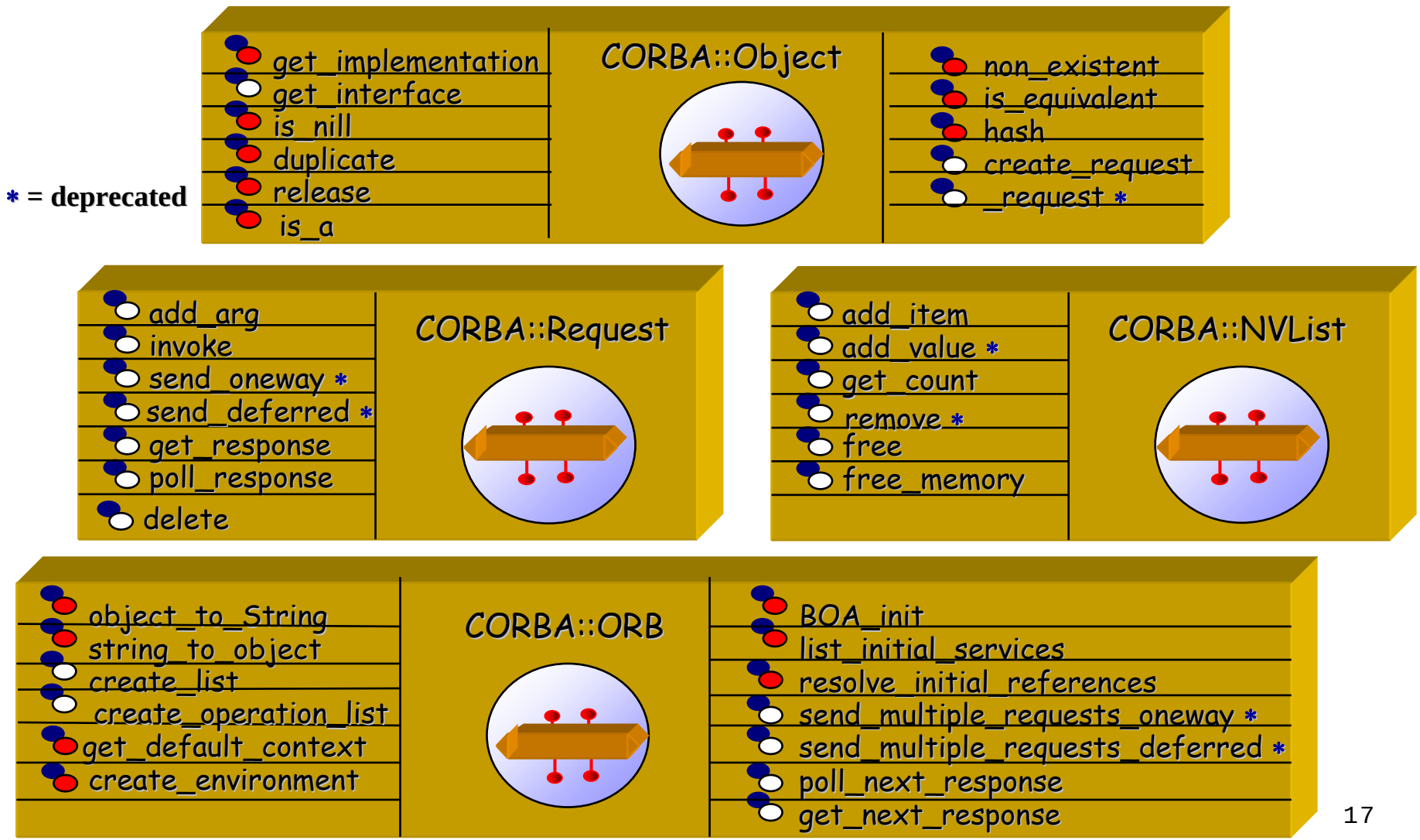


- **Invocación síncrona** (bloqueante): se envía la petición y se obtiene un resultado. (semántica de "*exactly-once*" y si hay una excepción "a*t-most-once*")
- **Invocación sincrónica diferida** (no bloqueante): disponible únicamente con invocación dinámica. Antes de invocar `get_response()`, se puede invocar `poll_response()` para saber si la respuesta está lista.
- **Invocación asíncrona o oneway** (semántica de "best-effort" ¡CUIDADO!)

La Interfaz de Invocación Dinámica (DII)

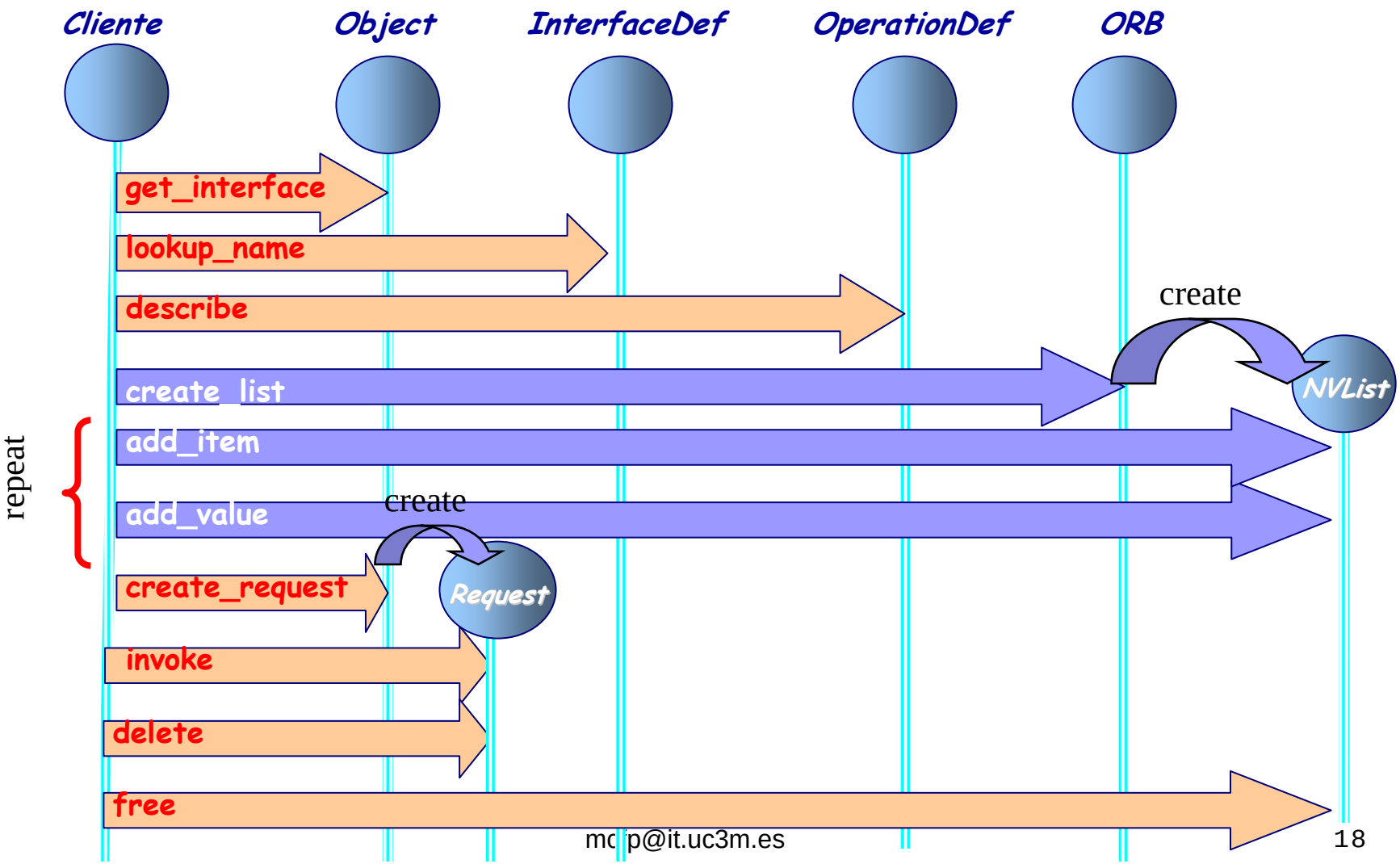
- Existen 4 interfaces de pseudo-objetos CORBA:
  - *CORBA::Object*
    - Operaciones que soportan todos los objetos CORBA. Para invocaciones dinámicas:
      - *get_interface()*
      - *create_request()*
      - *_request()*
  - *CORBA::Request*
    - Representa una petición de operación y ofrece métodos para darle parámetros, para invocarla, y leer los resultados
  - *CORBA::NVList*
    - Permite construir listas de parámetros
  - *CORBA::ORB*
    - Permite crear objetos NVList, y enviar y recibir múltiples peticiones

Interfaces de Invocación Dinámica



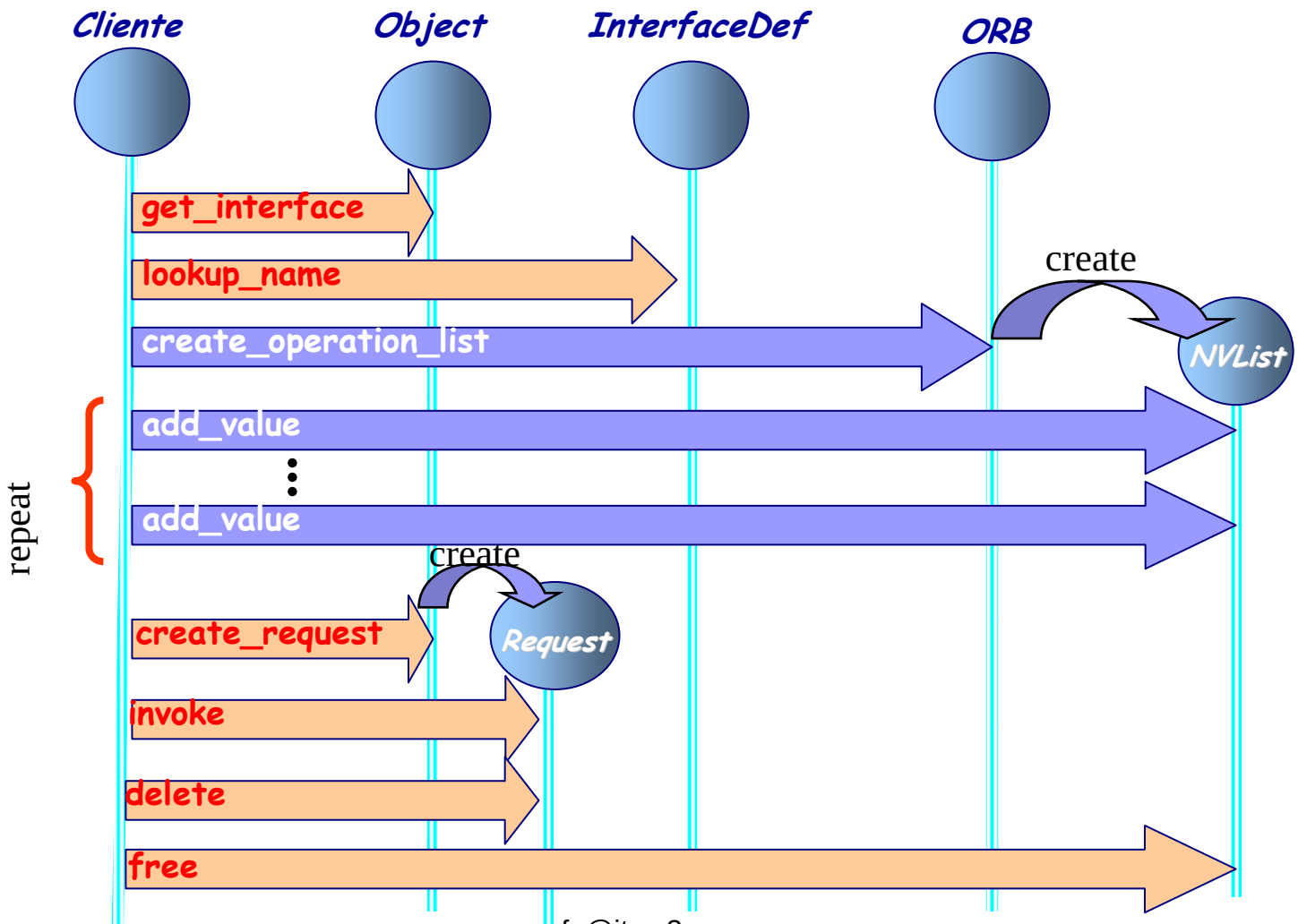
Escenarios (1/3)

1. Escenario “hágalo-usted-mismo”



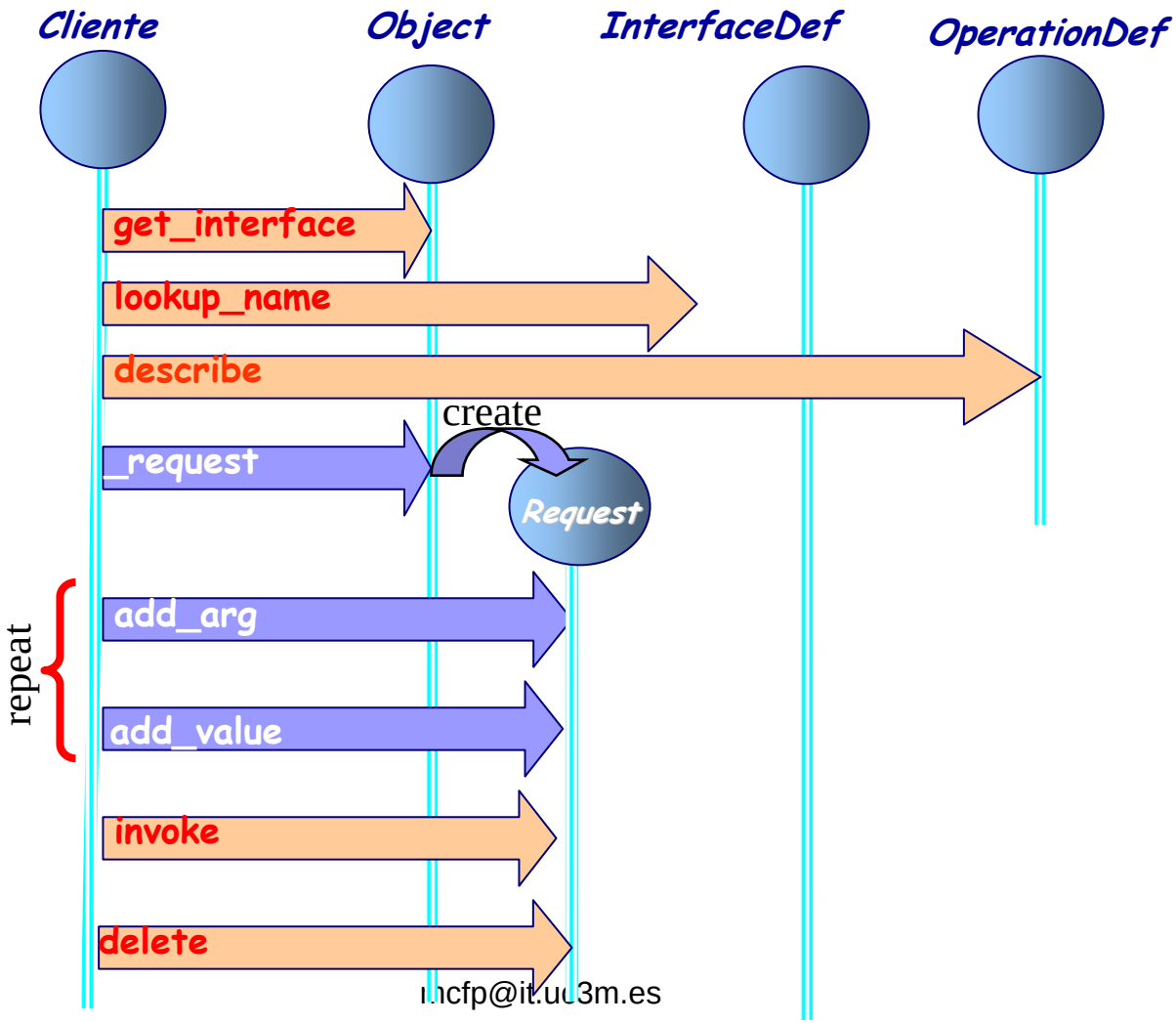
Escenarios (2/3)

2. Escenario “ORB-puede-ayudar”



Escenarios (3/3)

3. Escenario “sin NVList explicita”



Ejemplo en OrbixWeb (1/4) `Bank.idl`

```
// A bank account
interface Account {
readonly attribute float balance;
atttribute long accountNumber;
void makeDeposit(in float sum);
void makeWithdrawal( in float sum,
   out float newBalance);
};

// A factory for bank accounts
interface Bank {
exception Reject (string reason; );

// Create an account
Account newAccount (in string owner,
   inout float initialBalance) raises (Reject);

//Delete an account
void deleteAccount(in Account a);
};
```

Ejemplo en OrbixWeb (2/4)
1. “Haga-lo-usted-mismo”

```
import org.omg.CORBA.Object;
Import org.omg.CORBA.ORB;
import org.omg.CORBA.Request;
import org.omg.CORBA.FloatHolderHolder; //Para los parámetros inout (OrbixWeb)
  ...
// Crea una lista vacía de 2 valores
  NVList argList = ORB.init().create_list(2);
// Añade los valores a la lista argList
  NamedValue owner = argList.add(ARG_IN.value);
  Owner.value().insert_string(“Chris”);
  NamedValue initBal = argList.add(ARG_INOUT.value);
  initBal.value().insert_float(56.50);
// Crea la petición
  Request req = target._create_request(ctx,”newAccount”,argList,result);
// Invoca la petición
  req.invoke();
// Extrae el resultado
  Org.omg.CORBA.Object objRef = req.return_value().extract_Object();
  ...
```

ctx es el CORBA::Context en el que se pasa información del cliente que no es conveniente pasar por parámetros

Nota: El uso del método **add()** en vez de **add_item()** ya no es estándar.
El uso del método **return_value()** ya no es estándar

Ejemplo en OrbixWeb (3/4)
2. “ORB-puede-ayudar”

```
import org.omg.CORBA.Object;
Import org.omg.CORBA.ORB;
import org.omg.CORBA.Request;
import org.omg.CORBA.FloatHolderHolder; //Para los parámetros inout (OrbixWeb)
  ...
// Crea una lista vacía de 2 valores
  NVList argList = ORB.init().create_operation_list(...);
  ...
// Añade los valores a la lista argList
  NamedValue owner = argList.add(ARG_IN.value);
  Owner.value().insert_string(“Chris”);
  NamedValue initBal = argList.add(ARG_INOUT.value);
  initBal.value().insert_float(56.50);
// Crea la petición
  Request req = target._create_request(ctx,”newAccount”,argList,result);
// Invoca la petición
  req.invoke();
// Recoge el resultado
  Org.omg.CORBA.Object objRef = req.return_value().extract_Object();
  ...
```

Nota: El uso del método **add()** en vez de **add_item()** ya no es estándar.
El uso del método **return_value()** ya no es estándar

Ejemplo en OrbixWeb (4/4)
3. “Tercera-vía”

```
import org.omg.CORBA.Object;
Import org.omg.CORBA.ORB;
import org.omg.CORBA.Request;
import org.omg.CORBA.FloatHolderHolder; //Para los parámetros inout (OrbixWeb)
  ...
// Coge una referencia del objeto
  Object target = ORB.init().string_to_Object(refStr);
  ...
// Crea una petición vacía
  Request req = target.create_request(ctx,”newAccount”,null,result);
// Prepara la Petición
  FloatHolder f = new FloatHolder((float)1000.00);
  req.add_in_arg().insert_String(“Chris”)
  req.add_inout_arg().insert_Streameable(new FloatHolderHolder(f))
  req.set_return_type(ORB.init().create_interface_tc(“IDL:Account:1.0”,“Account”));
// Invoca la petición
  req.invoke();
// Recoge el resultado
  Org.omg.CORBA.Object objRef= req.return_value().extract_Object();
  ...
```

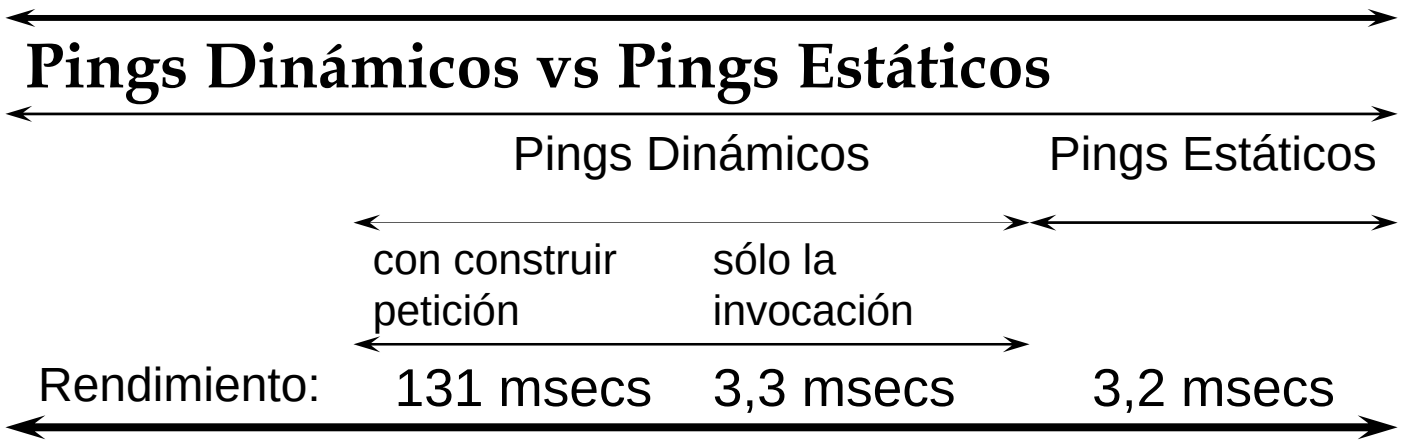
Nota: El uso del método **add_X_arg()** en vez de **add_arg()** ya no es estándar.
El uso de los métodos **set_return_type()** y **return_value()** ya no es estándar

Modelos de Interacción

- Invocaciones Síncronas
  - `r.invoke();`
- Invocaciones Síncronas diferidas
  - `r.send();` // antes: **`send_deferred()`**
  - `r.poll_response();` // determina si la operación ha terminado
  - `r.get_response();` // coge el resultado de la operación si está disponible
    // (**`r.poll_response()`** es cierta) y si no, bloquea
- Invocaciones Asíncronas
  - `r.send();` // antes: **`send_oneway()`**
- También existen: // sobre **`org.omg.CORBA.ORB`**
  - **`ORB.send_multiple_requests()`**
    // antes: **`..._deferred() y ..._oneway()`**
  - **`ORB.poll_next_response()`**
  - **`ORB.get_next_response()`**

El precio de la libertad

- Un “Benchmark” sobre el ORB “Visibroker”
  - Se realizaron 1.000 invocaciones remotas



- La Invocación Dinámica es 40 veces más lenta
  - El mayor gasto es debido a la preparación de la petición
  - Pregunta: ¿Cuál es el escenario más gravoso en tiempo?

Cuándo usar Invocación Dinámica

- En CORBA con Java existen 3 opciones
  - Invocación Estática
  - Invocación Dinámica
  - Stubs descargables (Java Applet)
- Depende del Uso del Cliente

Uso del Cliente	Técnica de invocación recomendada
El cliente invoca el servidor frecuentemente: el objeto servidor no cambia	Invocación Estática
El objeto cliente invoca al servidor pocas veces	Puede utilizar la invocación Dinámica
El cliente descubre el servidor en tiempo de ejecución	Invocación Dinámica
El cliente corre en un navegador y descubre un nuevo objeto	Utiliza Applet y stubs descargables. Luego, el Applet hace uso de la Invocación Estática

Resumen

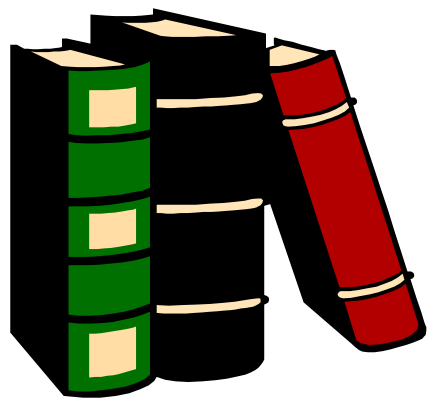
- La invocación dinámica sirve permite crear aplicaciones que descubren nuevas interfaces y sean capaces de utilizarlas, dotando así de mayor flexibilidad al sistema
- La información para poder construir dinámicamente un objeto Request se puede encontrar en el Repositorio de Interfaces

Algunas referencias

– ***Orfali, R., Harkey D. Y Edwards J. (1997). Instant CORBA.*** *New York: John Wiley &Sons inc. ISBN 0-471-18333-4.*

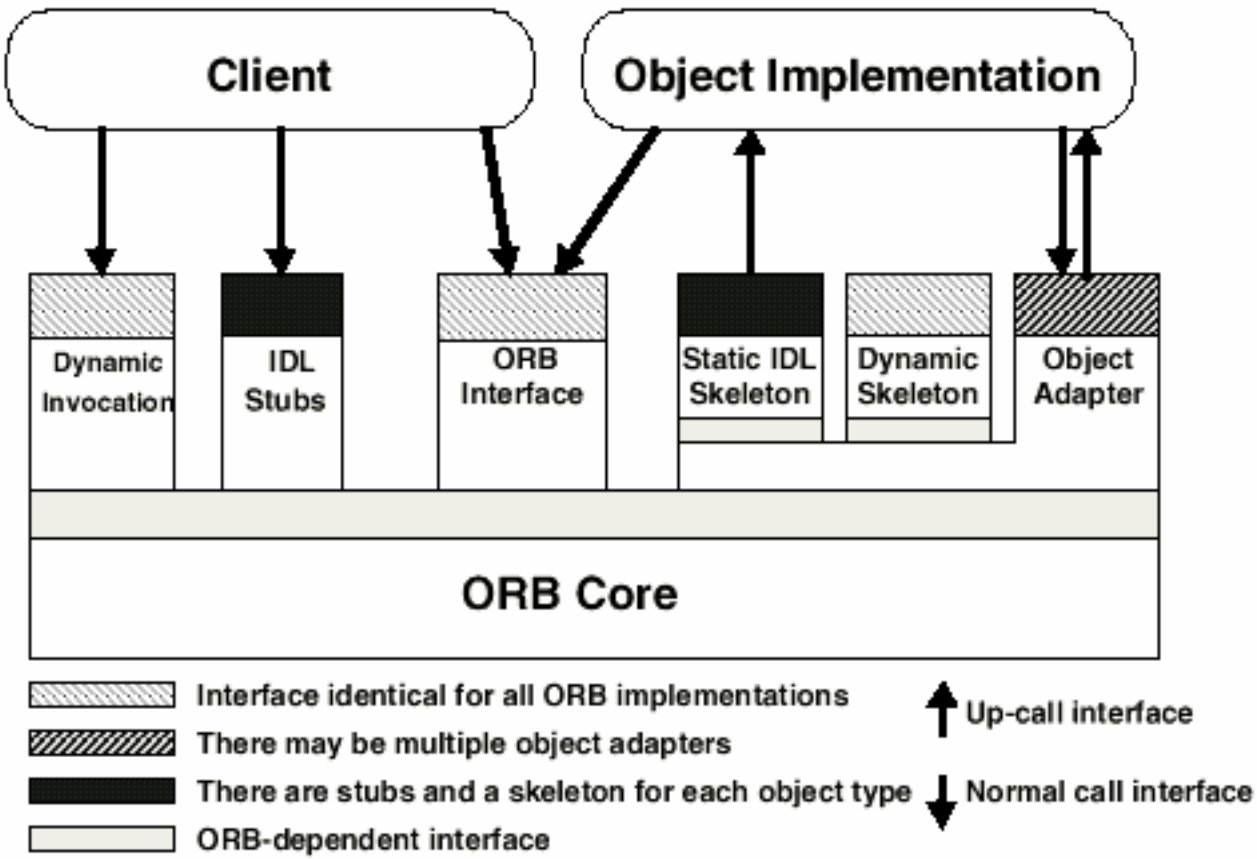
– ***IONA, OrbixWeb Programming Guide (1996).*** *Noviembre 1996.*

– ***Orfali, R. y Harkey D. (1997). Client/Server Programing with JAVA and CORBA****. New York: John Wiley &Sons inc. ISBN 0-471-16351-1.*



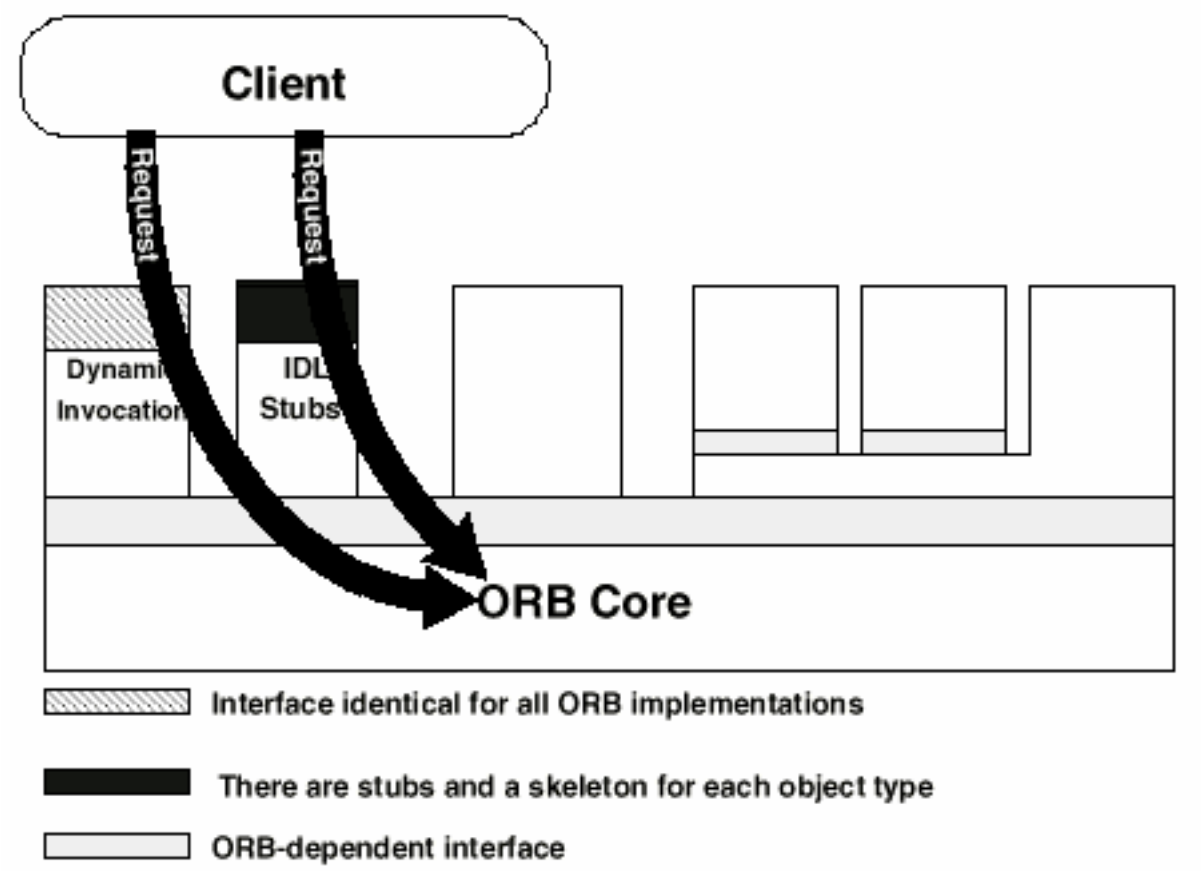
CORBA ¿Cómo funciona?

Elementos de la arquitectura



CORBA ¿Cómo funciona?

¿Cómo se realiza la invocación?



CORBA ¿Cómo funciona?

¿Cómo se recibe la petición?