Sistemas distribuidos de tiempo real

VIII.1: CORBA Estándar para objetos distribuidos

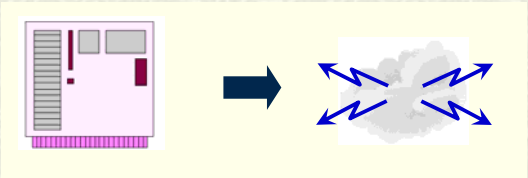
José M. Drake

Notas:

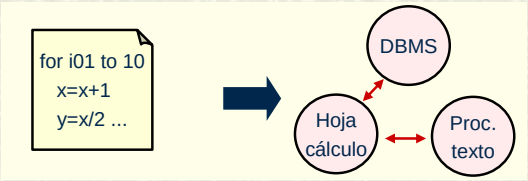
Posibilidades que ofrece Java para la comunicación en red: Socket,RMI y URL.

Evolución de los sistemas informáticos

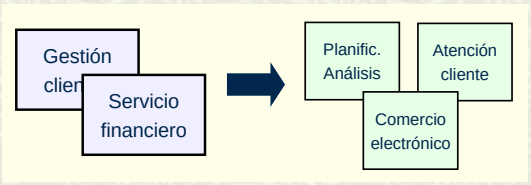
- Se pasa de unas plataformas basadas en un potente computador central a sistemas descentralizados basados en múltiples y minicomputadores heterogéneos.



- El desarrollo de la aplicaciones evoluciona de estar basadas en un diseño a propósito, a la integración de sistemas legados ya existentes.



- El manejo de las aplicaciones pasa de corresponder a servicios informáticos especializados, a ser realizado por todos las unidades de la empresa.



Notas:

Nuevo escenario de los sistemas de información

- A partir de los años 90 el escenarios típico de los sistemas de información es:
  - Se utilizan múltiples plataformas, lenguajes y sistemas.
  - Se hibridan aplicaciones distribuidas basadas en el paradigma cliente/servidor con aplicaciones centralizadas basadas en un gran computador (mainframe).
  - La arquitectura no está bien definida o simplemente no existe.
  - Entre los subsistema existen diferentes formatos de datos y definiciones semánticas.
  - La organizaciones necesitan llevar a cabo continuas integraciones de elementos que no estaban previstos en el diseño original.

Notas:

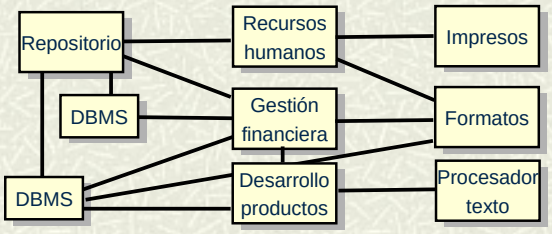
Llaves del desarrollo con éxito:

- Se requiere tener agilidad para responder a los rápidos cambios del entorno de negocios y de la tecnología:
  - Basar el desarrollo en la integración y no en el diseño del código específico.
  - Prever la evolución del futuro.
  - Hacer uso de la tecnologías que aparecen.

Notas:

Integración de los sistemas

- El aspecto central del desarrollo de un sistema ha pasado de la capacidad de diseño específico (programación) a la integración de sistemas ya disponibles con nuevos sistemas legados (Off-the-Shell).



- Aspectos relevantes de la integración, son:
  - Programación e integración son actividades diferentes, que requieren diferentes conocimientos y experiencias.
  - El diseño de productos específicos propios, compromete la facilidades de integración en el futuro.
  - Se requiere una tecnología apropiada, para que el esfuerzo que se necesita en el desarrollo de las interfaces de interconexión entre componentes legados propio del proceso de integración, no sea mayor que el de desarrollo del código de los propios componentes.

Notas:

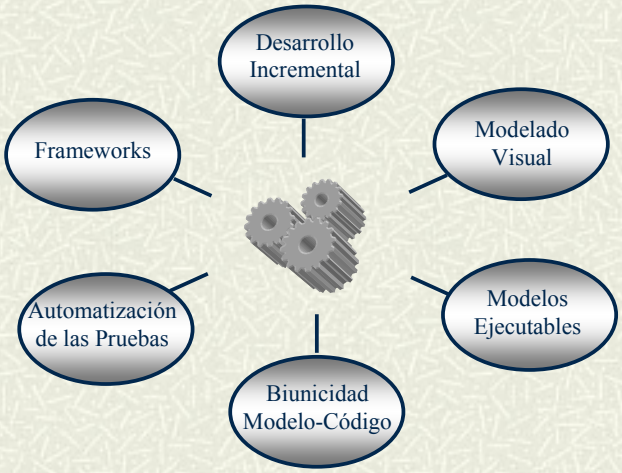
Gestión del futuro

- La clave para que una empresa sea competitiva, es tener capacidad de prever los cambios futuros.
  - Los rápidos cambios en hardware, redes, sistemas operativos, GUIs, etc. ha hecho dominar las arquitecturas cliente/servidor.
  - El incremento de la sofisticación de las aplicaciones ha incrementado los requerimientos de nuevos servicios, capacidad de respuesta, calidad de respuesta, cuya implementación específica no es rentable y requiere la integración de productos desarrollados por otros.
- El diseño de un nuevo sistema debe prever la incorporación de productos que aún no existen, y esto sólo es posible de conseguir con la estandarización de las infraestructuras, de las APIs y de los patrones de interacción.

Notas:

Disponibilidad de la tecnología

- Los procesos de desarrollo basados en plantillas, en librerías y extensiones, son de muy bajo nivel y raramente son aplicables al desarrollo de software para plataformas distribuidas heterogéneas.



Notas:

CORBA la solución propuesta por OMG

- El OMG (Object Management Group) se crea en 1989 como una asociación de las 1000 empresas que son líderes de la tecnología software, a fin definir especificaciones que puedan ser implementadas por todos ellas, y con ello, facilitar la interoperatividad de sus productos.
- CORBA (Common Object Request Brocker Architecture) es la tecnología que propone OMG para:
  - Para constituir la base de la tecnología emergente DOM (Distributed Object Management)
  - Para facilitar el diseño de aplicaciones basadas en el paradigma Cliente/Servidor.
  - Facilitar la integración de sistemas legados (Off-the-Shell)

Notas:

CORBA

- Define servidores estandarizados a través de un modelo de referencia, los patrones de interacción entre clientes y servidores y las especificaciones de las APIs.
- Con CORBA se facilita:
  - El diseño de middleware de distribución que facilita el diseño de aplicaciones en plataformas heterogéneas sin necesidad de conocer los detalles de los recursos y servicios que ofrece cada elemento de la plataforma.
  - La capacidad de diseñar aplicaciones desarrolladas en diferentes lenguajes de programación. Supliendo los recursos necesarios para implementar las interfaces entre ellas.
  - La insteroperatividad entre aplicaciones desarrolladas por diferentes fabricantes. Para que un componente sea interoperable sólo se requiere que ofrezcan las interfaces y los patrones de interacción basados en la especificación CORBA.

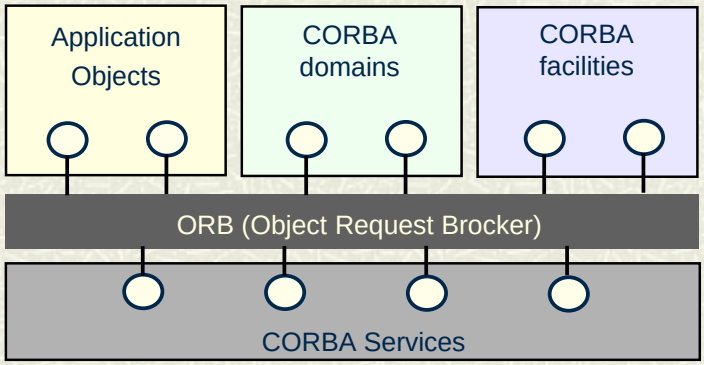
Notas:

Beneficios que ofrece CORBA.

- Capacidad para que los clientes invoquen métodos de objetos ubicados en cualquier nudo de la plataforma.
- Capacidad de invocar los métodos estáticamente (conocidos cuando se compila el cliente) y dinámicamente (desconocidos cuando se compiló el cliente).
- Facilita la heterogeneidad de los lenguajes de programación. Los clientes y servidores pueden ser desarrollados en lenguajes diferentes. CORBA proporciona los recursos necesarios para compatibilizarlos.
- Capacidad de incorporar información reflectiva que describe en tiempo de ejecución a los clientes las capacidades que ofrecen los servidores instalados.
- Transparencia de la ubicación en las invocaciones de los objetos que se invocan.
- Incorpora los mecanismos de seguridad en los acceso y de consistencia de las transacciones que se ejecutan.
- Polimorfismo en las invocaciones.
- Coexistencia con otras tecnologías (EJB, DCOM, etc.) a través de la especificación de los elementos puentes.

Notas:

ARQUITECTURA CORBA



- ORB (Object Request Brocker): Constituye la infraestructura de comunicación estandarizada, a través de las que se realizan las invocaciones de los métodos de los objetos y servicios. Se basa en el protocolo GIOP (General Inter-ORB Protocol) definido por la especificación CORBA, o las especializaciones del mismo para determinados medios de comunicación (como IIOP para Internet).

Notas:

Arquitectura CORBA (2)

- **CORBA services**: Provee los servicios básicos a nivel de sistema, tales como:
  - Servicios de nombres.
  - Servicios de notificación de eventos.
  - Ciclo de vida
  - Transacciones.
  - ....
  - Servicio de persistencia
  - Concurrencia
  - Externización
  - Seguridad.
- **CORBA Facilities**: Proporcionan un conjunto de funciones de alto nivel que facilitan que cubren aspectos generales como interfaces de usuario, gestion de información, etc.
  - User interface Management.
  - System management.
  - Information Management
  - Task Management
- **CORBA Domain**: Son especificaciones y definiciones semánticas que son comunes a ciertos dominios de aplicación en los que CORBA está implantada.
  - Financials.
  - Telecom
  - Healthcare
  - Internet
  - Bussines

Notas:

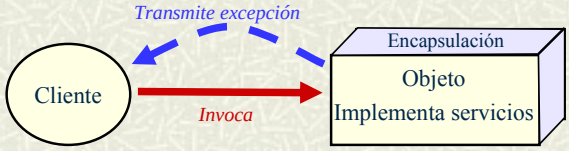
Conceptos CORBA

- CORBA constituye el principal middelware comercial abierto que sirve de base para insertar componentes software implementados por muy diferentes fabricantes: La compatibilidad se basa:
  - Acceso uniforme a los servicios.
  - APIs uniformes para descubrir los servicios y recursos disponibles
  - Gestión uniforme de errores y excepciones.
  - Políticas uniformes de seguridad.
- La especificación CORBA está basada en tres conceptos fundamentales:
  - Modelo orientado a objetos
  - Entorno de computación distribuido y abierto.
  - Integración y reutilización de componentes.

Notas:

Modelo de Objetos

- La especificación CORBA es altamente compatible con el paradigma orientado a objetos:
  - Objetos: Los servicios CORBA encapsulan atributos que describen el estado y métodos que implementan el comportamiento.
  - Clases: A nivel de diseño los objetos se describen mediante la clase de las que son implementación. Todos los objetos de una misma clases tienen el mismo comportamiento, aunque su propio estado.
  - Encapsulación: El estado y el comportamiento de un objeto se formulan como paquetes software de límites bien definidos. El comportamiento interno solo es accesible a través de las interfaces públicas que ofrece.
  - Herencia: Es la capacidad de una clase para transferir la naturaleza de sus estado y su comportamiento a las clases que desciendan de ella.
  - Polimorfismo: La capacidad de dos o mas clases para responder a una misma invocación, especializando su respuesta.
- El modelo de un objeto CORBA está compuesto por:
  - Modelo funcional estático: Define que respuesta dan sus servicios.
  - Modelo de ejecución dinámico: Define como se pueden invocar los servicios.



Notas:

Entorno de computación distribuida abierta

- CORBA se basa en un modelo de computación distribuida basada en el paradigma cliente/Servidor implementada a través de un broker.
- El Broker reduce la complejidad de la implementación jugando dos funciones:
  - Independiza la implementación de los clientes y servidores de la plataforma de ejecución (Procesadores, redes, sistemas operativos, servicios de comunicaciones, etc.)
  - Proporciona un conjunto de servicios comunes como intercambio de mensajes, servicio de directorio, acceso a metadata, transparencia de ubicación, seguridad, etc.
- Se basa en un modelo peer-to-peer de comunicación entre cliente y servicio de tipo síncrono, y con algunas opciones de invocación asíncrona.
- La comunicación remota se basa en un protocolo definido por la especificación denominado GIOP (General Inter-ORB Protocol), lo que proporciona interoperatividad entre ORB de diferentes empresas.

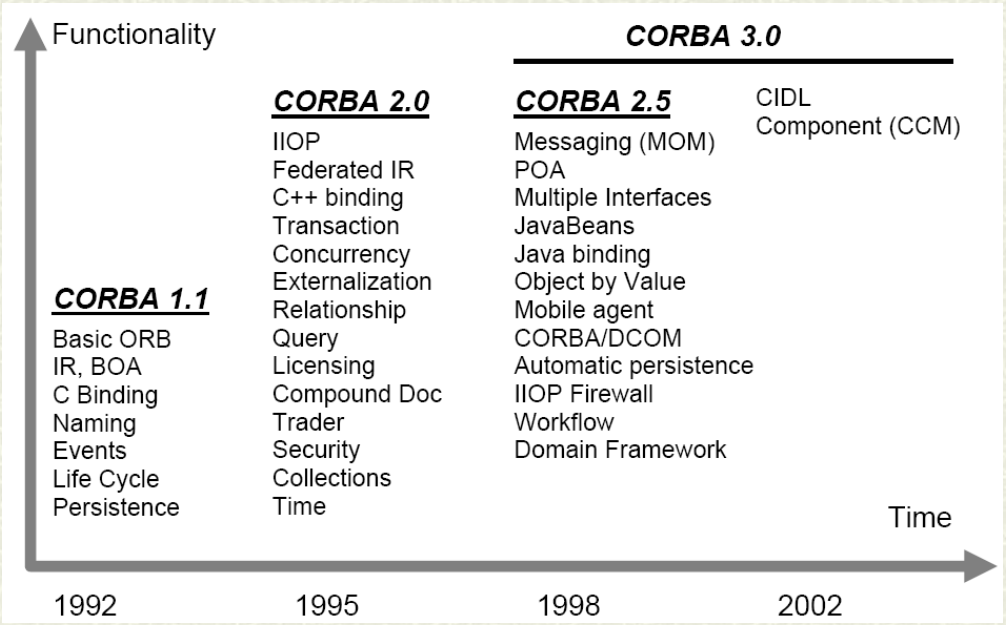
Notas:

Integración y reutilización de componentes

- Reutilización e integración de software son dos caras de una misma moneda;
  - Integración es la combinación de dos o mas componentes pre-existentes.
  - La reutilización no es posible sin un mecanismo estandarizado de integración.
- El broker CORBA proporciona un mecanismo de intercomunicación estándar entre cualquier par de componentes.
- A través de la especificación de la interfaces mediante el lenguaje neutro IDL (Interface Definition Language) las interfaces de los objetos se hace con independencia de la plataforma y de los lenguajes de implementación.
- Con los servicios CORBA como Object Query los clientes pueden descubrir y utilizar nuevos servicios, que pueden haber sido desarrollados incluso después del cliente.
- A través de las facilidades CORBA se facilitan frameworks de interoperación que estandariza las interacciones entre los objetos.
- Los dominios CORBA proporcionan las definiciones semánticas que simplifican el desarrollo de componentes reutilizables, dentro de un dominio de aplicación (Telecomunicaciones, Instrumentación médica, servicios financieros, etc.)

Notas:

Evolución de la especificación CORBA



Notas:

CORBA IDL (Interface Definition Language)

- Es un elemento esencial de la especificación CORBA, ya que hace posible especificar los componentes y servicios con independencia del lenguaje de implementación y de la plataforma de ejecución.
  - Es un estándar muy estable que prácticamente ha permanecido inalterado desde su definición.
  - En la especificación de IDL se incluyen los mapeos estándares a los principales lenguajes de programación (C, C++, Java, ADA95, SmallTalk, Phyton, etc,)
  - Garantiza la independencia de la plataforma. Los servicios especificados con IDL no tienen ninguna característica que sea propia de la plataforma o del ORB.
  - IDL es un lenguaje puro de especificación y no de implementación.
  - La calidad y el estilo de la especificación IDL de los servicios es un aspecto clave de la reusabilidad de los componentes CORBA.

Notas:

Elementos claves de IDL (1)

- Modulos IDL: Sirve para organizar las especificaciones y para definir un espacio de nombres.

```
module Assembly{
   typedef string Widget;
}
```

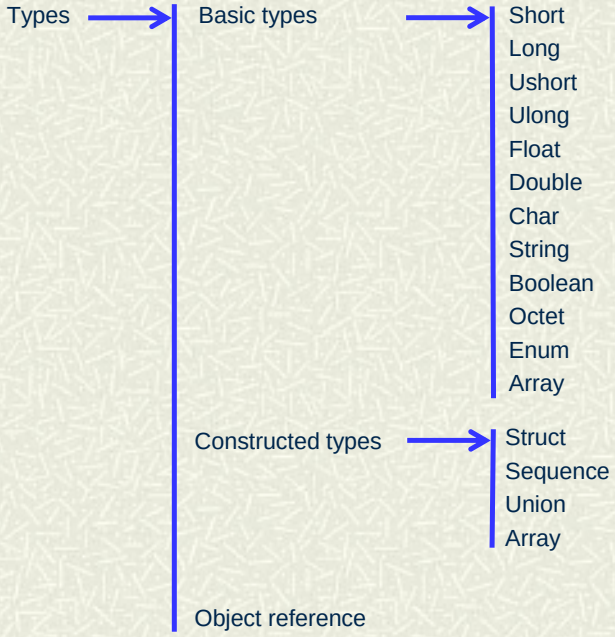
```
module Part{
  typedef ::Assembly::Widget PartWidget;
  module ComponentPart{
     typedef PartWidget ComponentWidget;
  };
}
```

- Interfaces IDL: Define un servicio externo ofrecido por un objeto como un conjunto de operaciones. Cada interfaz define un nuevo tipo. Las interfaces admiten relaciones de herencia.

```
module Counter{
   interface Count{
     attribute long sum;
     long increment();
   };
};
```


Notas:

Elementos claves de IDL (2)



Notas:

Elementos claves de IDL (3)

- Constantes IDL: Permite definir constantes de los tipos: integer, character, boolean, floating point, string y tipos definidos

```
const unsigned long MAX_NUM=12000;
const char SEPARATOR='/';
const double AVOGADRO= 6.02E25;
const string MI_REGION= "Cantabria";
const float DIAS_POR_MES= 364.5/12;
```

- Tipos IDL: permite definir nuevos tipos de elementos.

```
typedef unsigned long NumeroTelefonico;
typedef string Nombre, Direccion;
enum TarjetasCreditos {MasterCard, VISA, AmericanExpress};
struct Invitado{
  Nombre elNombre;
  Direccion laDireccion;
  TarjetasCredito laTarjeta;
  unsigned long numTarjeta;
};
const float DIAS_POR_MES= 364.5/12;
typedef sequence<Invitado> ListaInvitados;
typedef Invitado Invitados[100];
```


Notas:

Elementos claves de IDL (4)

- Atributos IDL: Los atributos representan valores internos de los objetos que pueden ser accedidos a través de operaciones accessor (*get*) y mutator (*set*)

```
interface CensusData{
   attribute unsigned short age;
   readonly attribute string birth_date;
   atribute string lastName;
   struct HouseHold{
      string address;
      unsigned short numOccupants;
   }
   attribute HouseHold house;
}
```

- Excepciones IDL: CORBA garantiza que cualquier invocación siempre recibe la respuesta o en caso de fallo una excepción. Las excepciones pueden estar definidas en la especificación y ser lanzada por el ORB, o estar definida por el usuario y es lanzada por las operaciones de los servicios.

```
exception CardExpired{
   string expirationDate;
}
```


Notas:

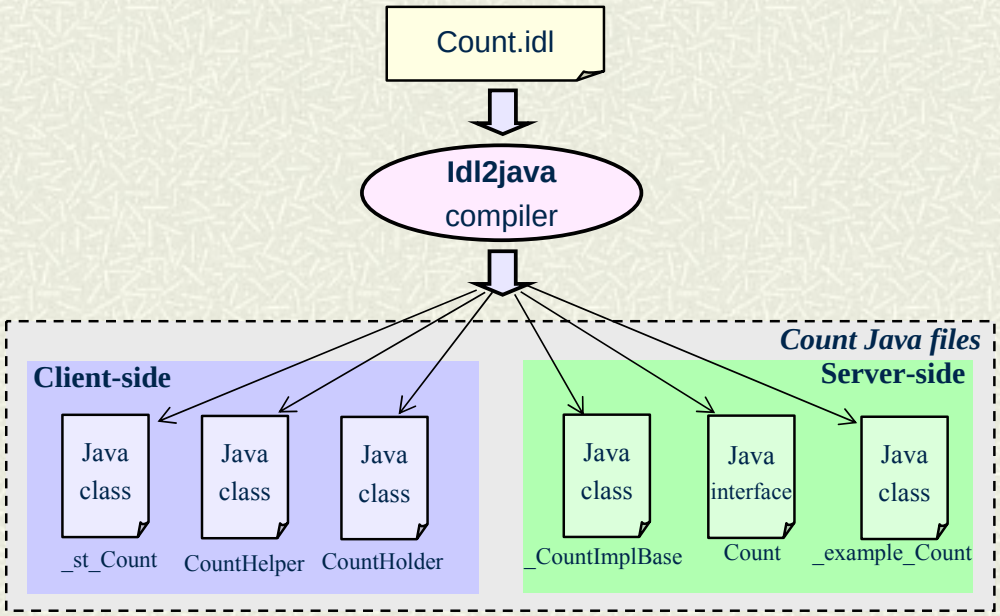
Elementos claves de IDL (5)

- Descripción de las operaciones IDL: Describe un método de un objeto que puede ser invocado por un cliente. Incluye su descriptor, el valor que retorna, los parámetros que intercambia y las excepciones que puede lanzar.

```
interface AirlineReservation{
   typedef unsigned long ConfirmationNumber;
   enum SeatKind{Windows,Aisle,Middle};
   exception BadFrequentFlayerNumber{};
   exception SeatNotAvailable{};
   exception BadConfirmationNumber{};
   ConfirmationNumber makeReservation(
     in string passagerName,
     in unsigned long frequentFlyerNumber,
     inout SeatKind kind;
     out string seatAssignement
   ) raises(BadFrequentFlayerNumber, SeatNotAvailable);
   oneway void cancelReservation(
     in ConfirmationNumber number,
   ) raises(BadFrequentFlayerNumber);
}
```


Notas:

Compilación de la especificación IDL



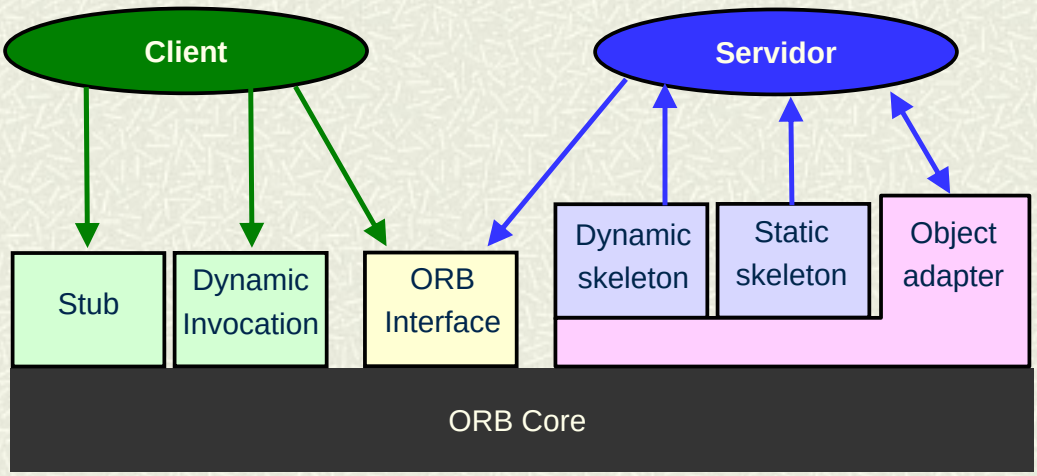
Notas:

Ficheros Java resultantes de compilar IDL

- **_st_Count**: Clase Java que implementa el stub del lado cliente del objeto Count. Ofrece una interfaz local Count al cliente y las funciones de marshalling y unmarshalling y de acceso al ORB.
- **CountHelper**: Ofrece funciones de utilidad para que los clientes transformen la vista CORBA object a Count Object.
- **CountHolder**: Ofrece funciones para que los clientes gestione los tipos y parámetros definidos en la especificación IDL.
- **_CountImplBase**: Implementa el esqueleto del servidor Count. Esto es ofrece hay el ORB la interfaz definida por la especificación y contiene los métodos Java que son invocado por las invocaciones del cliente y cuyos cuerpos deben ser conectados con el propio servidor. Habitualmente es una clase raiz que debe ser la base por extensión de cada servidor.
- **Count**: Interfaz Java que describe las operaciones definidas en la especificación IDL.
- **_example_Count**: Ejemplo de como debe ser construido el servidor. Se suele utilizar como base para generar el servidor mediante edición.

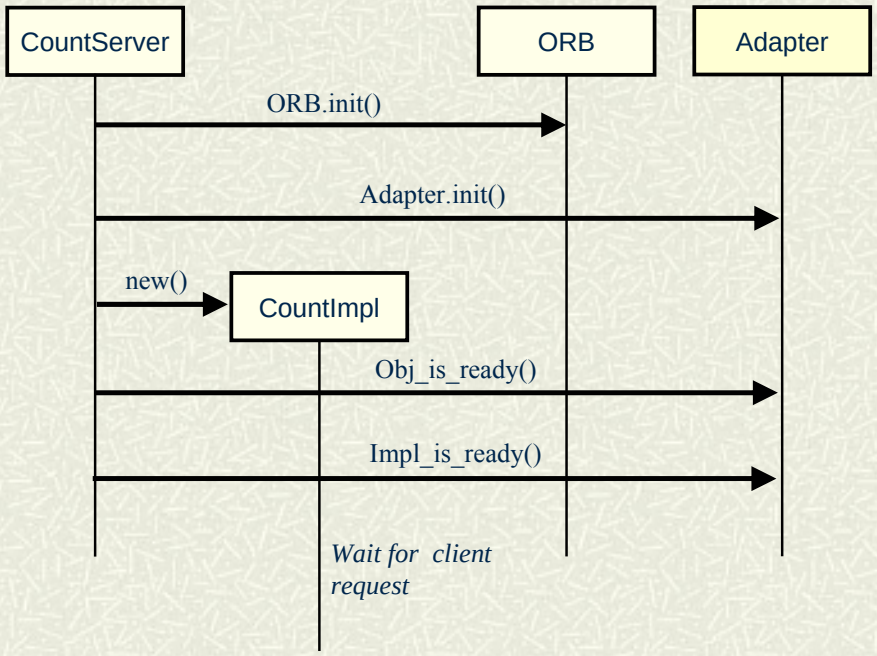
Notas:

Principales elementos de una aplicación CORBA



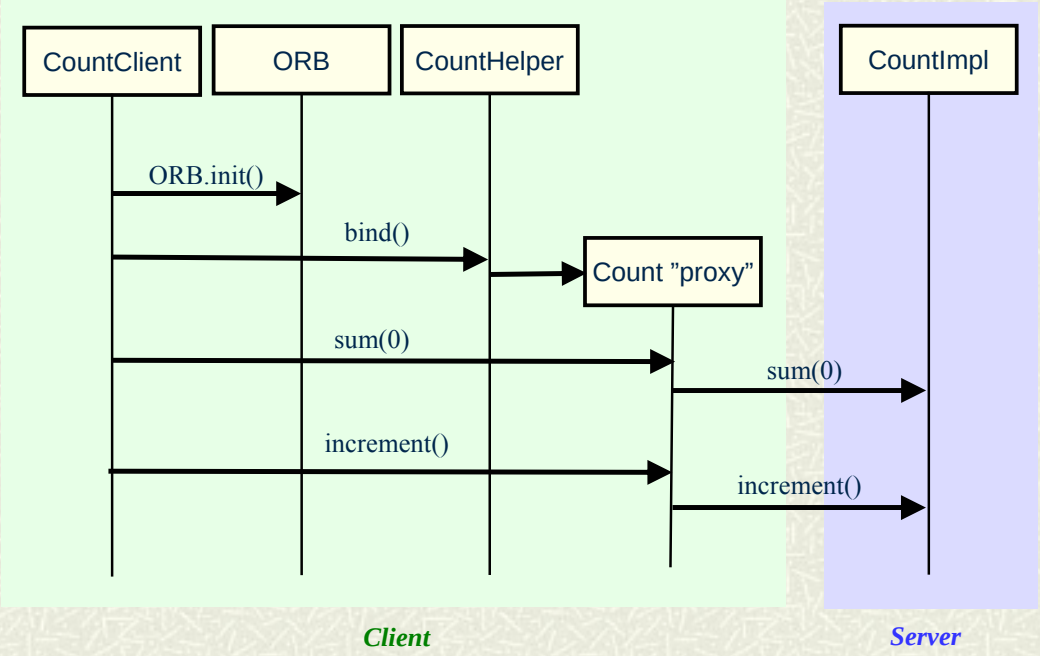
Notas:

Proceso de instanciación de un servidor

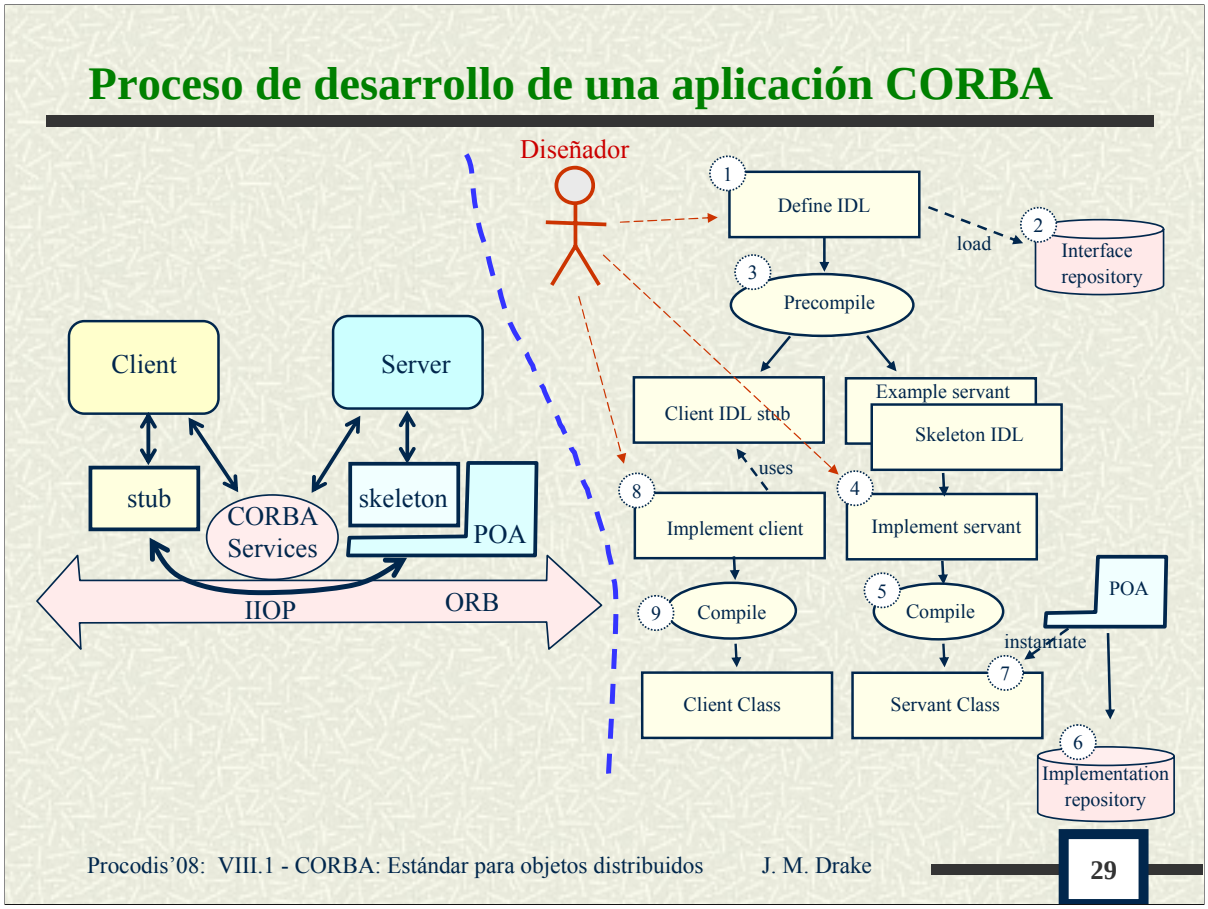


Notas:

Proceso de conexión de un cliente



Notas:


Proceso de desarrollo de una aplicación CORBA
Diseñador
1
Define IDL
load
2
Interface repository
3
Precompile
Client
Server
Client IDL stub
Example servant
Skeleton IDL
uses
stub
CORBA Services
skeleton
POA
8
Implement client
4
Implement servant
IIOP
ORB
9
Compile
5
Compile
POA
instantiate
7
Client Class
Servant Class
6
Implementation repository
Procodis'08: VIII.1 - CORBA: Estándar para objetos distribuidos
J. M. Drake
29

Notas:

Proceso de desarrollo de una aplicación CORBA

1. Define la interfaz del servidor: Define la interfaz del servidor utilizando IDL. Con ello se formaliza que operaciones ofrece el servidor y como pueden ser invocadas.
2. Registra la definición en el repositorio de interfaces: La definición de la interfaz se almacena en un repositorio de interfaces, a fin de que en tiempo de ejecución los clientes obtengan la información reflectiva que ofrecen.
3. Compila la especificación de la interfaz: Compila la especificación IDL y obtienes los ficheros de código Java que se requieren para desarrollar la aplicación: Stub del cliente, esqueleto del servidor y ficheros auxiliares con funciones.
4. Implementa el código del servant: Escribe el código Java que implementa la funcionalidad del servidor.
5. Compila el código Java del servidor: Compila el código Java utilizando un compilador Java ordinario.
6. Registra el servidor en el repositorio de objetos: Registra los objetos del servidor en el registro de nombres que ofrece el sistema para ser posteriormente localizados por los clientes.
7. Instancia los objetos del servidor: Crea un adaptador, y a través de él, el servidor queda a la espera de recibir invocaciones.
8. Implementa el código del cliente: Implementa el código Java del cliente. A través del registro de objetos, se obtiene el proxy al servidor, y se invocan las operaciones como si fuesen locales.
9. Compila el código Java del cliente: Se compila y ejecuta el cliente como cualquier aplicación Java ordinaria.

Notas:

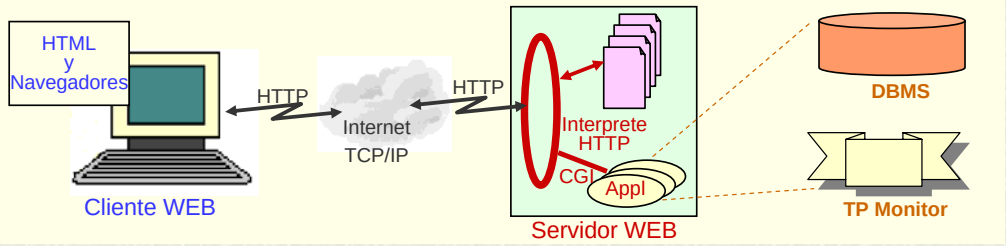
Declive de las plataformas OO generales

- Desde los años 90 las plataformas middleware han posibilitado la creación de sistemas distribuidos complejos que se ejecutan en plataforma heterogéneas.
- Hasta ahora ha habido tres grandes colosos:
  - DCOM, .net son los mas extendidos, pero solo son soportados por las plataformas Microsoft.
  - RMI, EJB son soportados por cualquier plataforma que soporte una JVM, pero sólo pueden ser utilizados por aplicaciones que utilicen Java.
  - CORBA es el que busca la estandarización y está abierto a cualquier empresa que desee implementarlo. Pero es tan complejo que casi ningún fabricante ha sido capaz de implementar de forma completa las especificación.
- El resultado es que ninguno se ha impuesto, han entrado en declive y se han buscado nuevas soluciones:
  - SOAP Service Oriented Application) y WEB Services
  - MDA (Model Driven Arquitectura y MDD (Model Driven Development)
  - Plataformas propietarias ligeras.

Notas:

SOAP y WEB Service

- Se basa en la disponibilidad en todos los sistemas interconectado por la infraestructura WWW (World Wide WEB) servidores WEB con capacidad de interpretar requerimientos formulados con el protocolo HTTP.

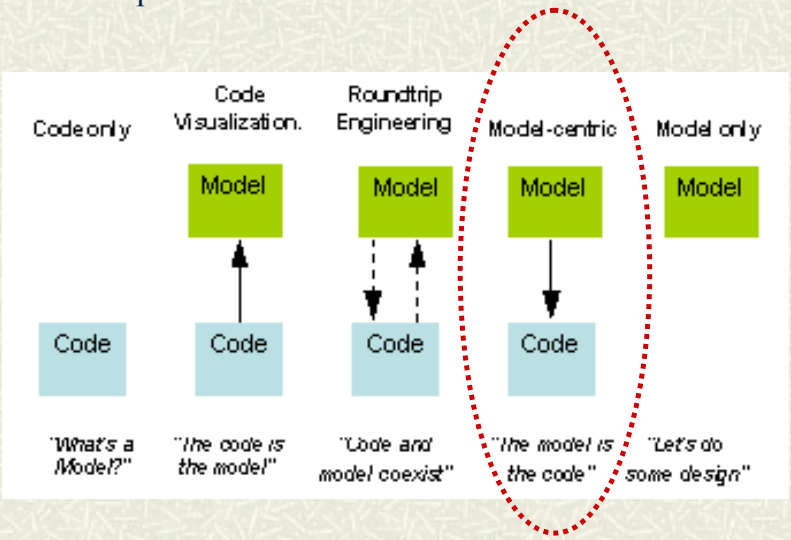


- Actualmente no se ha conseguido implantar esta tecnología como la solución general:
  - La tecnología se basa en los WEB Services que básicamente consisten en una aplicación que recibe los mensajes con protocolos http, los interpreta y los responde. El interprete HTTP constituye un cuello de botella que limita las prestaciones.
  - SOAP es una especificación basada en mensajes, su nivel de abstracción es muy bajo y limita la complejidad de las aplicaciones que se pueden desarrollar.
  - Los sistemas basados en HTTP representan un agujero importante para la seguridad de la plataforma.

Notas:

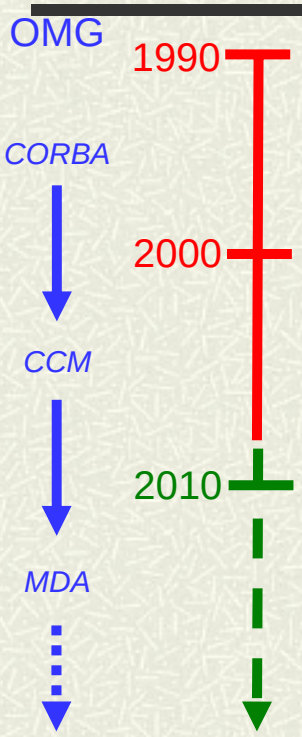
MDA (Model Driven Architecture)

- La nueva apuesta de OMG



Notas:

Soluciones de OMG (Object Management Group)



El software se hace reutilizable utilizando un middleware estándar (p.e. CORBA) que lo hace interoperable e independiente de la plataforma.
Problema: *La rápida evolución de la tecnologías de las plataformas no puede ser seguida por los middleware.*

Se utiliza tecnología de componentes: El software de negocio se diseña con independencia de la plataforma. Herramientas automáticas introducen el código que lo adaptan a la plataforma.
Problema: *La rápida evolución de las tecnologías de las plataformas no puede ser seguida por las herramientas.*

Se utilizan tecnología MDA: El software de negocio se diseña con independencia de la plataforma. Las herramientas se diseñan automáticamente a partir de modelos que describen las plataformas.
Problema: Actualmente en desarrollo.

Notas:

ICE: Internet Communications Engine

- Es una plataforma ligera desarrollada por la empresa ZeroC creada por algunos de los creadores de CORBA y por ahora, gratuita.
- Sus objetivos son:
  - Proporcionar una plataforma middelware abierta adecuada para sistemas heterogéneos de cualquier fabricante.
  - Proporcionar un soporte completo para el desarrollo de aplicaciones de cualquier tipo de dominio.
  - Implementar una plataforma muy ligera en la que se evite cualquier complejidad innecesaria.
  - Optimizar la implementación para que sea eficiente en uso de la anchura de banda de la red, uso de memoria y utilización de CPU.
  - Dotarla con los recursos mas avanzados para que se pueda construir con ella aplicaciones seguras aun cuando opere en la red pública.

Notas: