Introducció a les plataformes distribuïdes

Jordi Ceballos Villach

PID_00185399

UOC
Universitat Oberta de Catalunya

Índex

Introducció

En els mòduls 1 a 3 s'ha abordat des d'un punt de vista genèric (sense entrar en els detalls de tecnologies concretes) el disseny d'aplicacions distribuïdes, i també la programació orientada a components. Posteriorment, en els mòduls següents, s'analitzen detalladament tecnologies molt utilitzades actualment, com la tecnologia Java EE, per al desenvolupament d'aplicacions de components distribuïts. També s'analitzen les tecnologies RMI (com a pas previ per a entendre Java EE) i els serveis web.

Per a completar la visió global de les plataformes distribuïdes creiem que és necessari, en aquest moment, abans de detallar les tecnologies distribuïdes actuals, fer una mica d'història i analitzar com s'ha arribat a la situació actual. Farem, breument, un repàs de les tecnologies que van formar part dels inicis de les plataformes distribuïdes i les que formen part de les plataformes actuals.

Aquest mòdul comença oferint una visió històrica de les diferents tecnologies de desenvolupament d'aplicacions distribuïdes. Com a plataforma més representativa en aquests inicis, a pesar que encara està en ús en certs entorns, analitzem amb cert detall l'OMG/CORBA. L'elecció d'aquesta tecnologia no és en va, ja que resulta molt didàctica per a entendre la base de les tecnologies distribuïdes actuals.

Tant en el cas concret de CORBA, com en el repàs breu de la resta de tecnologies, intentarem mantenir sempre un punt de vista arquitectònic (allunyat dels detalls concrets de programació) que ofereixi a l'estudiant la possibilitat de veure les principals similituds i diferències entre les diferents tecnologies. En definitiva, no s'ha d'oblidar que totes aquestes tecnologies no són sinó diferents propostes existents a l'abast del programador, entre les quals caldrà triar quan arribi el moment d'implementar una aplicació distribuïda.

Objectius

La lectura d'aquest mòdul permetrà assolir els objectius següents:

1. Obtenir una visió històrica de les diferents tecnologies de desenvolupament d'aplicacions distribuïdes, i també conèixer les principals similituds i diferències entre aquestes.

2. Conèixer amb més detall les tecnologies CORBA.

3. Conèixer que hi ha diferents alternatives per al desenvolupament d'una aplicació distribuïda, i adquirir el criteri per a triar entre una o una altra.

1. Una mica d'història: RPC, RMI, DCOM...

Des que va sorgir la possibilitat de connectar diversos ordinadors en xarxa, es va començar a buscar com comunicar processos (aplicacions) que s'executaven en els diferents nodes, creant el que es denominen ***aplicacions distribuïdes***. El primer avenç important va ser l'aparició d'RPC[1].

(1)De l'anglès *remote procedure call*.

1.1. RPC

Als anys vuitanta es va definir el protocol **RPC**, que permet crear aplicacions client/servidor basades en la crida a procediments remots. RPC permet que un programa client pugui invocar un procediment situat en un servidor remot sense haver de preocupar-se pel procés de comunicacions entre tots dos. L'intercanvi de missatges està definit independentment tant del llenguatge de programació com del maquinari i el sistema operatiu de les màquines client i servidor.

Vegeu també

Repasseu el procediment d'invocació remota RPC en l'assignatura de *Xarxes i aplicacions Internet*.

En RPC, una funció client crida una funció remota i rep la resposta d'igual manera que en fer una crida a una funció local. En aquest sentit, RPC va ser molt innovador, ja que per a aconseguir el mateix fins que va aparèixer era necessari programar tot el codi relatiu a les comunicacions.

Per a utilitzar RPC, el programador declara en un fitxer IDL[2] les funcions que es podran cridar remotament. A continuació, el compilador d'IDL genera el codi l'*stub* (estub, en català) per al client i també per al servidor. L'*stub* client serà l'encarregat de preparar els missatges RPC que s'enviaran a través de la xarxa i que contindran la informació necessària per a la crida remota. L'*stub* del servidor rep aquesta informació, executa en el servidor l'operació sol·licitada i retorna el resultat corresponent.

(2)De l'anglès, *interface definition language*.

Es pot resumir l'arquitectura d'RPC en la figura 1.

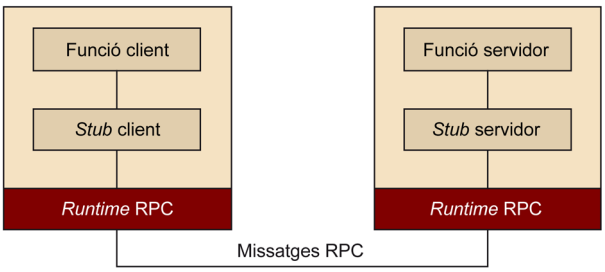

Figura 1. Arquitectura d'RPC

Quan es produeix una crida RPC, els passos són els següents:

1) El procés client crida una funció RPC remota, li passa els paràmetres necessaris i queda a l'espera de rebre la resposta.

2) L'*stub* del client crea un missatge RPC, on empaqueta els paràmetres en un format independent de la plataforma, i l'envia al servidor per mitjà de l'entorn d'execució d'RPC (anomenat *runtime RPC*).

3) L'*stub* del servidor rep el missatge, el desempaqueta i crida la funció sol·licitada, i li passa els paràmetres rebuts.

4) La funció remota s'executa i acaba retornant un resultat, que rep l'*stub* del servidor, l'empaqueta en un missatge i l'envia al client.

5) L'*stub* del client rep el missatge, el desempaqueta i envia el resultat a la funció client, que, en rebre'l, continua amb l'execució igual com si hagués fet una crida a una funció local.

Hi ha diferents implementacions del protocol RPC:

- **ONC RPC** (també anomenat *Sun RPC*): una de les primeres implementacions d'RPC, a càrrec de l'empresa Sun Microsystems. ONC RPC està molt estès en sistemes UNIX i és un estàndard de l'IETF.

- **DCE RPC**: desenvolupat als anys vuitanta pel consorci OSF (Open Software Foundation), anomenat actualment *The Open Group*. Pertany a l'estàndard de programari intermediari DCE per a aplicacions distribuïdes.

- **MSRPC** (Microsoft RPC): implementació derivada del DCE RPC, però amb algunes extensions específiques de Microsoft. MSRPC està integrat en els sistemes operatius de Microsoft des del Windows NT.

- **FreeDCE**: implementació de codi obert de DCE RPC per a sistemes Linux, duta a terme per SourceForge. De fet, el seu objectiu final és acabar implementant DCOM per a Linux.

1.2. Objectes distribuïts

Les **tecnologies d'objectes distribuïts** és una evolució d'RPC, però canviant el paradigma procedural per l'orientat a objectes. És a dir, mentre en RPC es fan crides a funcions remotes, amb objectes distribuïts es fan crides a mètodes d'objectes remots. El resultat obtingut és el mateix, ja que en definitiva estem executant codi d'una funció d'un sistema remot, però amb un nivell d'abstracció diferent.

Una aplicació d'aquest tipus està composta per diversos objectes distribuïts a través d'una xarxa heterogènia que interactuen sense importar en quins llenguatges de programació estiguin desenvolupats, ni les diferències en maquinari o sistemes operatius.

En cada sistema de la xarxa es disposa d'un **agent de petició d'objectes** (ORB[3]) encarregat de gestionar les comunicacions entre els clients (que fan peticions des de l'exterior) i cadascun dels objectes servidors.

(3)De l'anglès *object request broker*.

L'arquitectura d'un sistema d'objectes distribuïts és similar a l'arquitectura d'RPC, però substituint el concepte de *funció* pel d'*objecte*.

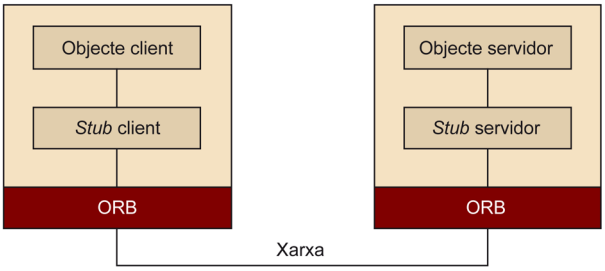

Figura 2. Arquitectura d'un sistema d'objectes distribuïts

Entre les tecnologies usades per a la invocació remota podem destacar: CORBA, RMI, DCOM, .NET Remoting i WCF[4].

(4)De l'anglès *windows comunication foundation*.

1.2.1. CORBA

El 1991 el consorci OMG va publicar l'especificació de **CORBA 1.0**, que defineix un estàndard per a la creació d'aplicacions orientades a objectes distribuïdes. Concretament, el protocol **IIOP**[5], definit també per OMG, permet fer crides entre objectes distribuïts amb independència del llenguatge de programació, les plataformes de maquinari i els sistemes operatius.

(5)*IIOP* és l'acrònim d'*Internet inter-ORB protocol*.

Vegeu també

CORBA es veurà amb més profunditat en l'apartat "CORBA" d'aquest mòdul.

Encara que les primeres versions de CORBA oferien pocs serveis de programari intermediari (*middleware*), amb el pas dels anys es van definir aquests serveis i, actualment, **CORBA 3** defineix un programari intermediari molt complet.

Utilitzant el llenguatge **CORBA IDL** es defineix la interfície IDL de cada objecte servidor, que conté la declaració dels mètodes que es poden cridar remotament. Posteriorment, amb el compilador de CORBA IDL es generen els *stubs* del client i servidor per a un cert llenguatge (per exemple, C++, Java[6] o SmallTalk). I, finalment, programem el codi del client i el del servidor fent les crides remotes corresponents.

(6)Es poden desenvolupar aplicacions CORBA en Java mitjançant el Java IDL.

L'avantatge principal de CORBA és que és un estàndard creat amb la intervenció d'un elevat nombre d'empreses del sector, amb la qual cosa gaudeix de gran popularitat. L'inconvenient principal és la seva elevada corba d'aprenentatge, sobretot perquè és un estàndard molt extens.

1.2.2. RMI

L'any 1997 Sun afegeix el servei ***remote method invocation*** **(RMI)** a Java 1.1, que permet fer crides remotes entre objectes situats en màquines virtuals de Java diferents, tant si aquestes es troben en la mateixa màquina física com si estan en màquines diferents. RMI disposa d'un protocol natiu anomenat ***JRMP***[7], que permet fer crides remotes entre objectes, però amb el requisit que tant el client com el servidor han d'estar implementats en llenguatge Java.

(7)De l'anglès *Java remote method protocol*.

Vegeu també

RMI es veurà amb detall en el mòdul "Java RMI" d'aquest material didàctic.

En RMI, en lloc de definir les interfícies remotes en un llenguatge IDL, s'utilitzen les interfícies de Java, amb la qual cosa no és necessari aprendre cap llenguatge IDL. RMI utilitza el mecanisme de serialització de Java per a enviar els missatges a través de la xarxa, la qual cosa fa que el pas de paràmetres sigui més potent que en altres tecnologies com, per exemple, CORBA o DCOM.

El 1998 s'afegeix **Java IDL** a Java 1.2, la qual cosa permet crear tant clients com servidors CORBA en llenguatge Java, basant-nos en les interfícies definides en CORBA IDL. D'aquesta manera, es poden crear aplicacions CORBA en Java, exactament igual com es faria, per exemple, en C++ o COBOL.

Web recomanat

Podeu ampliar la informació sobre el Java IDL en http://java.sun.com/products/jdk/idl.

L'any 2000 s'afegeix **RMI-IIOP** a Java 1.3, que ofereix la possibilitat de programar en el senzill estil d'RMI (sense necessitat d'escriure interfícies IDL), però utilitzant el protocol IIOP per a les comunicacions. D'aquesta manera, obtenim interoperabilitat entre objectes Java i objectes CORBA (encara que aquests últims no hagin estat desenvolupats en Java).

Web recomanat

Podeu ampliar la informació sobre RMI-IIOP en http://java.sun.com/products/rmi-iiop.

Algunes possibles utilitats d'RMI-IIOP són les següents:

- Permet accedir des d'un client RMI a un servidor CORBA. Cal tenir en compte que les interfícies RMI són un subconjunt del llenguatge CORBA IDL i, per tant, l'IDL d'un objecte CORBA existent no sempre es pot fer correspondre amb una interfície RMI.

- Permet accedir des d'un client CORBA a components d'un servidor Java EE. Aquests avantatges tenen com a contrapartida la complexitat de l'ús de CORBA des de Java, que el converteixen en una opció solament per a grans i costosos sistemes distribuïts en què la interoperabilitat entre diferents tecnologies és crítica.

Vegeu també

En el subapartat "Aplicacions distribuïdes en el món Java: Java EE i els Enterprise JavaBeans" de l'apartat "Programari intermediari d'aplicacions distribuïdes" es presenta la tecnologia EJB.

L'any 2004 es va revisar el model RMI a fons amb motiu del llançament de Java 5 i es van introduir importants millores que es van mantenir en les versions següents de Java, que encara persisteixen avui dia. Aquestes millores es refereixen a la consolidació del model simple RMI amb el seu protocol natiu de comunicació Java (**JRMP**), i s'allunya de la complexitat que representa la versatilitat amb altres llenguatges i plataformes mitjançant RMI-IIOP[8], el qual no va ser revisat en Java 5 i avui dia s'utilitza de manera marginal.

[8]El protocol IIOP es troba encara a disposició del programador RMI per a complexos sistemes distribuïts en què la interoperabilitat és crítica.

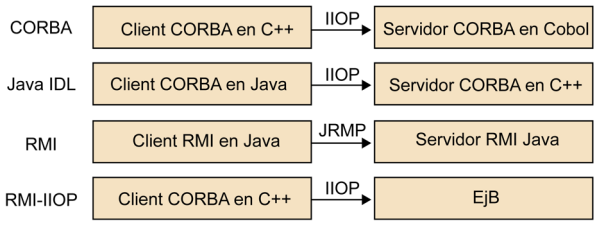

Figura 3. Exemples de desenvolupament

Java 1.5

A partir de la versió 1.5 Java va usar la notació Java X, amb X com a número de la versió.

1.2.3. COM/DCOM

Basant-se inicialment en OLE, Microsoft va llançar el 1993 el model de components **COM**[9], que estableix un estàndard binari d'interoperabilitat entre components, independent dels llenguatges i plataformes en què s'implementin. En altres paraules, COM permet crear i utilitzar components entre diferents processos que s'estiguin executant en una mateixa màquina, encara que estiguin escrits en diferents llenguatges de programació.

[9]De l'anglès *component object model*.

OLE

OLE (*object linking and embedding*) és la tecnologia de Microsoft per a la creació de documents compostos. Permet, per exemple, incrustar una gràfica Excel en un document Word.

Un component COM pot suportar múltiples interfícies, en què cadascuna consta d'un conjunt de funcions. Un client interactua amb un component COM adquirint un punter a una de les interfícies del component i cridant els mètodes per mitjà del punter, suposant que el servidor es trobi en el seu mateix espai d'adreces.

Terminologia

En els primers temps de la tecnologia COM, Microsoft va utilitzar el terme *OLE* per a referir-se a l'arquitectura basada en COM. Posteriorment, per motius de màrqueting, va establir igualment el terme *ActiveX*, ja que tots dos (OLE i ActiveX) es basen en COM.

Com a resultat d'aquesta confusió, originada des de Microsoft, sovint s'utilitza qualsevol combinació de les paraules *COM*, *OLE* o *ActiveX* seguides de *control*, *objecte* o *component* per a designar el mateix concepte (o tal vegada no). Entendre de què es parla requereix, de vegades, una anàlisi profunda del context.

Tota interfície COM té un identificador únic a escala global (denominat ***GIID***), alhora que un nom simbòlic per a identificar-la en temps de compilació. El dissenyador d'una interfície en defineix el nom, el GIID i les funcions que la componen utilitzant el llenguatge COM IDL, molt similar a CORBA IDL, però amb certes peculiaritats específiques de COM.

Podem classificar els components COM en tres tipus:

1) Servidor *in-process*: també denominat *control ActiveX*, és l'evolució dels antics controls OLE i s'executa dins de l'espai d'adreces del procés client. Està implementat en DLL i pot ser utilitzat tant en aplicacions Windows com en pàgines web. Actualment, hi ha un important mercat de controls ActiveX amb gran varietat de funcionalitats, com la creació de gràfics, visors 3D, calendaris, generació d'informes, etc.

2) Servidor local: el component s'executa en el mateix ordinador, però en un procés diferent.

3) Servidor remot: el component s'executa en un ordinador diferent del de l'aplicació client gràcies al protocol DCOM[10], introduït per Microsoft el 1996 que, per mitjà d'una infraestructura basada en RPC, permet fer crides a components remots.

[10]De l'anglès *distributed COM*.

DCOM ofereix algunes característiques molt interessants:

- **Possibilitat d'utilitzar objectes COM existents**. Ja que DCOM s'ha construït sobre COM, tots els objectes compatibles amb aquesta tecnologia són suportats per DCOM. Així, la reutilitzabilitat dels objectes COM està garantida per als seus usuaris antics, i es redueix el temps i cost de desenvolupament de noves aplicacions distribuïdes.

- **És neutral a la plataforma**. DCOM permet que components sobre diferents plataformes (sempre que aquestes implementin DCOM) puguin ser capaces d'interoperar entre si. De fet, encara que DCOM habitualment només s'associa amb plataformes Win32, també està disponible per a sistemes Unix i Macintosh.

Producte EntireX

El producte EntireX de Software AG ofereix suport DCOM per a sistemes Linux.

- **És neutral al llenguatge**. Microsoft clama que els components COM/DCOM poden ser escrits en qualsevol llenguatge de programació. No obstant això, aquesta afirmació és certa només en algunes excepcions. Per exemple, el llenguatge Java estàndard no pot treballar amb COM. Només la implementació de Java de Microsoft pot usar COM. En plataformes Win32, la família de Visual Studio (Visual C++, Visual Basic, Visual J++) i Delphi són els llenguatges més utilitzats per a desenvolupar objectes COM.

- **És neutral a la capa de transport**. DCOM treballa actualment amb TCP/IP, UDP, IPX/SPX, NetBIOS i AppleTalk. De fet, ja que està llest per a utilitzar TCP/IP, no resulta un problema fer ús d'Internet.

Seguretat

L'any 2003 un forat de seguretat en l'RPC de DCOM va permetre que el cuc Blaster infectés ràpidament milers de sistemes Windows. El cuc es colava pel port 135 de TCP/UDP i, enviant missatges RPC incorrectament creats, podia executar qualsevol codi maliciós en sistemes Windows 2000/XP, i prenia així el control total sobre les màquines.

Jeffrey Lee Parson, de 19 anys, va ser detingut i va confessar ser el creador de Blaster. Va ser condemnat a 18 mesos de presó.

La interfície de cada objecte servidor DCOM es defineix utilitzant el llenguatge DCOM IDL i, posteriorment, amb el compilador MIDL (Microsoft IDL) es generen els *stubs* corresponents.

Inicialment, CORBA i DCOM eren dues tecnologies sense possibilitat d'interconnexió possible, fins que van aparèixer alguns productes de propietat que ho permetien. Per estandarditzar la situació, l'any 1997 OMG va definir el ***COM/CORBA interworking specification***, un estàndard que defineix com ha de ser la comunicació DCOM-CORBA i, actualment, ja hi ha productes que el suporten com, per exemple, Orbix COMet de Iona.

1.3. Programari intermediari d'aplicacions distribuïdes

Els programaris intermediaris d'aplicacions distribuïdes estenen les tecnologies de distribució d'aplicacions (com RMI, CORBA i DCOM) amb l'objectiu de simplificar el treball dels programadors, i els ofereixen una sèrie de serveis bàsics (com suport per a transaccions, seguretat o bateries de connexions a base de dades). D'aquesta manera, el programador s'allibera d'escriure el codi corresponent a aquests serveis en les seves aplicacions.

1.3.1. DCE

El 1992 va aparèixer la primera versió pública de DCE[11], un estàndard de programari intermediari definit pel consorci OSF, actualment anomenat *The Open Group*.

L'objectiu de DCE és definir un entorn per a facilitar el desenvolupament d'aplicacions distribuïdes seguint el paradigma procedural i utilitza el protocol DCE RPC per a les comunicacions distribuïdes.

(11)De l'anglès *distributed computing environment*.

Web recomanat

Podeu ampliar informació sobre DCE a http://www.opengroup.org/dce.

L'inconvenient principal és que en lloc de ser orientat a objectes, DCE va ser definit seguint el paradigma procedural i, per això, es va guanyar moltes crítiques que afirmaven que només sortir la primera versió de DCE ja era obsolet. Només ha tingut èxit en entorns UNIX, i no ha aconseguit ser gaire popular.

1.3.2. Aplicacions distribuïdes en el món Java: Java EE i els Enterprise JavaBeans

El 1998 Sun va anunciar la versió **1.0 d'EJB**. Aquesta tecnologia de components distribuïts forma part de l'edició empresarial de Java, denominada *Java EE* (*Java EE* actualment). Els EJB s'executen dins del context d'un contenidor d'un servidor d'aplicacions (com, per exemple, BEA WebLogic, IBM Websphere o Sun ONE), que ofereix serveis com transaccionalitat, seguretat, persistència i suport multifil, i facilita així el treball dels programadors a l'hora de desenvolupar components EJB.

Vegeu també

La tecnologia EJB s'explica detalladament en el mòdul "Java EE" d'aquest material didàctic.

EJB utilitza internament el protocol RMI per a les comunicacions, i es pot configurar perquè usi JRMP o IIOP, segons les nostres necessitats.

1.3.3. Aplicacions distribuïdes en el món Microsoft: MTS, COM +, .NET Remoting i WCF

El 1996 Microsoft va anunciar **MTS**[12], disponible per Windows NT. MTS és un programari intermediari que facilita, en gran mesura, la creació d'aplicacions empresarials de gran escala, i s'encarrega de qüestions crítiques com la gestió de recursos compartits, seguretat, transaccions, etc.

(12)De l'anglès *Microsoft transaction server*.

El problema d'MTS és que estava desenvolupat sobre COM, però ni el sistema operatiu ni COM sabien de l'existència d'MTS. Per aquestes raons, MTS havia de fer molts ardits per a oferir els seus serveis, i no els podia oferir a tots els objectes COM (els multifils n'estaven exclosos).

El desenvolupament de **Windows 2000** va donar a Microsoft l'oportunitat de corregir les deficiències d'MTS, afegir nous serveis i integrar-lo dins del sistema operatiu. Així, el 1999 va aparèixer **MTS 3.0** que, per raons de màrqueting, es va anomenar ***COM+ 1.0***, la qual cosa va causar gran confusió en semblar que es tractava d'una nova versió de COM, però no era així, ja que es tractava d'una versió actualitzada del conjunt de serveis per a components.

Algunes de les característiques principals dels serveis COM+ són:

- Gestió de transaccions distribuïdes.
- Comunicació asíncrona entre aplicacions mitjançant cues de missatges.
- Balanceig dinàmic de càrrega entre els servidors de components COM.
- Millora del rendiment en els accessos a bases de dades.
- Notificació d'esdeveniments entre components.

El juliol de l'any 2000 Microsoft va anunciar la plataforma **.NET** en què, en gran part, Microsoft parteix de zero i crea un nou conjunt de tecnologies, pensant sobretot a facilitar el desenvolupament d'aplicacions (tant per a Windows com per al Web) i fer front a la forta competència de Java amb la plataforma empresarial Java EE.

Un dels canvis més radicals (pels seus efectes en l'ampli mercat de components COM existents) és el nou model de components .NET, que reemplaça COM. Un altre canvi és que **.NET Remoting** reemplaça DCOM com a tecnologia de components distribuïts i, pel que fa als serveis de components de COM+, es mantenen igual, encara que se'n canvia el nom i es passen a dir *NET Enterprise Services*.

L'evolució de la plataforma .NET ha estat contínua fins als nostres dies i a partir de la versió 3 s'aposta per WCF (*Windows comunication foundation*) com a tecnologia de comunicacions remotes entre components (tant entre components .NET com amb components desenvolupats en altres tecnologies).

Web recomanat

Trobareu una bona introducció a WCF a http://msdn.microsoft.com/ en-us/library/ ms731082%28v=VS.85%29.aspx.

1.4. Serveis web

Els **serveis web** són una de les tecnologies més consolidades actualment per al desenvolupament d'aplicacions distribuïdes a Internet, on són essencials la interoperabilitat i el suport de plataformes i xarxes heterogènies.

No hem catalogat els serveis web com una tecnologia d'objectes distribuïts per diferents raons:

- Encara que hi ha eines que permeten crear serveis web mitjançant tècniques orientades a objectes (i, per això, aparenten ser objectes distribuïts), no ens hem de confondre, ja que un servei web es pot implementar perfectament en un llenguatge no orientat a objectes.

- La tecnologia dels serveis web té limitacions en comparació dels sistemes d'objectes distribuïts. A manera d'exemple, els serveis web estan començant a buscar solucions per a oferir serveis transaccionals, mentre que ja fa anys que aquest aspecte està solucionat en els sistemes d'objectes distribuïts.

- Els sistemes distribuïts, sovint, ofereixen la possibilitat de mantenir un estat entre el client i el servidor, de manera que l'objecte remot pot contenir dades i un cert estat sobre el qual pot actuar el client durant el temps de vida de l'objecte. En canvi, els serveis web no tenen aquesta noció d'estat.

Vegeu també

La tecnologia dels serveis web s'explica detalladament en el mòdul "SOA" d'aquest material didàctic.

El 1998 l'empresa UserLand va definir un protocol molt simple, anomenat ***XML-RPC***, per a simular crides a procediments remots intercanviant missatges XML per mitjà d'HTTP. L'objectiu d'XML-RPC va ser crear un protocol independent del venedor i de la plataforma, fàcil d'implementar i que permetés fer crides a procediments per mitjà d'Internet, enviant missatges XML, que són només text, a diferència de la resta de tecnologies com DCOM i CORBA, que utilitzen protocols en format binari.

Webs recomanats

Podeu ampliar la informació sobre els serveis web en les adreces següents:

http://www.w3.org/standards/webofservices/

http://ws.apache.org/

Microsoft, IBM, UserLand i altres empreses, acollides pel W3C, han definit el protocol **SOAP**[13], molt més ambiciós que XML-RPC.

(13)De l'anglès *simple object access protocol*.

Algunes característiques addicionals de SOAP són:

- Els missatges SOAP normalment es transporten per HTTP, encara que també poden viatjar mitjançant altres protocols.
- SOAP permet crear tipus de dades definides per l'usuari.
- SOAP té un mecanisme de capçaleres que el fa extensible, de manera que es poden afegir signatures digitals, transaccions, encaminament, etc.
- SOAP suporta introspecció, gràcies a l'especificació WSDL, que permet descriure els serveis web.

L'avantatge de SOAP respecte a altres tecnologies d'aplicacions distribuïdes és que el podem utilitzar en qualsevol llenguatge i sistema operatiu, ja que l'únic requisit és poder processar documents XML, cosa relativament senzilla, ja que, en definitiva, és text. En canvi, comparat amb la resta de programaris intermediaris, hem de tenir en compte que SOAP és molt recent i les seves funcionalitats encara són una mica limitades.

1.5. Conclusions

L'apartat que hem treballat ens ofereix una visió general de les diferents tecnologies per a la creació d'aplicacions distribuïdes. En la figura 3 mostrem els anys en què han anat apareixent les diferents tecnologies, i també la relació entre aquestes.

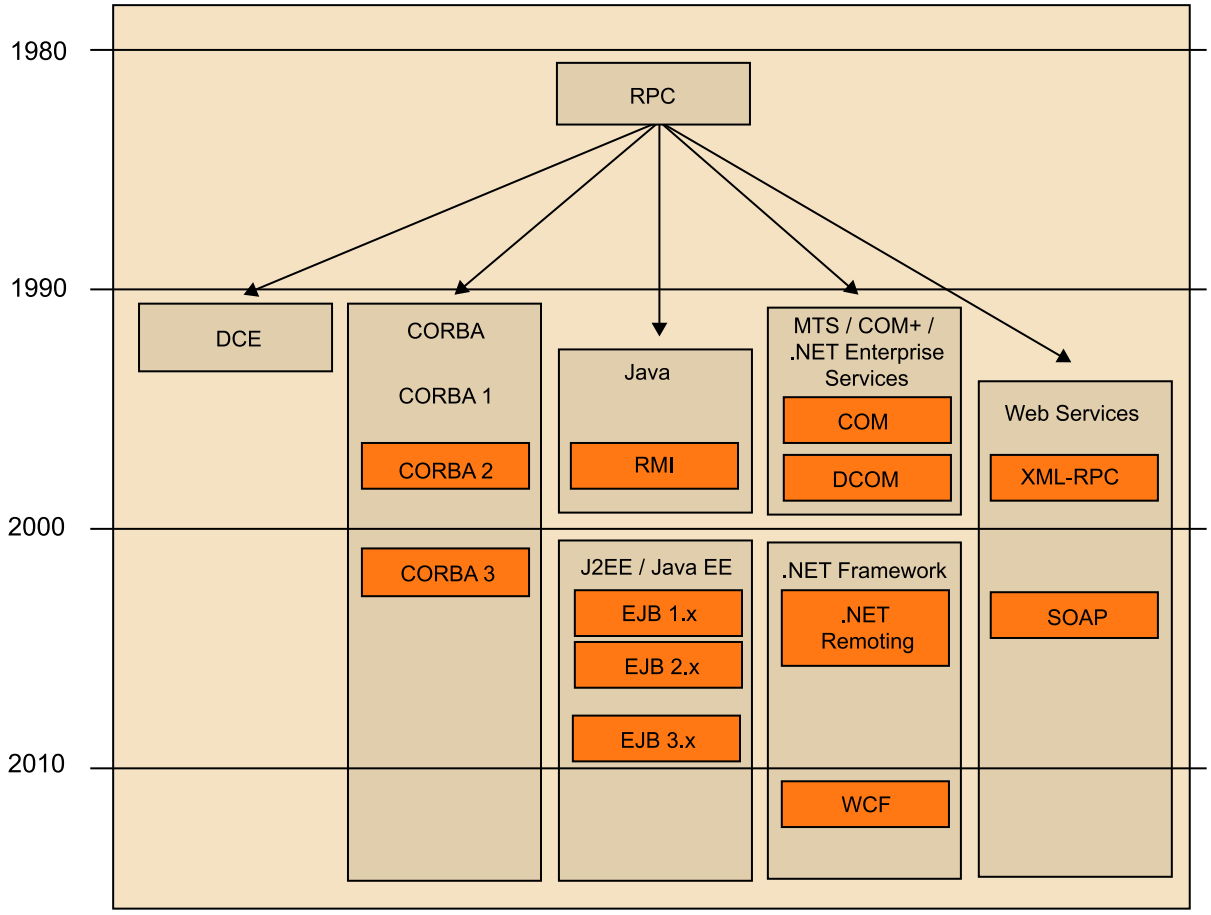

Figura 4. Tecnologies d'aplicacions distribuïdes

A continuació analitzarem amb més detall una de les primeres tecnologies que permetien la comunicació remota: CORBA.

Vegeu també

En els propers mòduls analitzarem detalladament les tecnologies RMI, Java EE i serveis web.

2. CORBA

El propòsit de **CORBA** és oferir interoperabilitat entre aplicacions en un entorn distribuït i heterogeni. Des de la seva aparició el 1991 ha anat evolucionant i, actualment, és un estàndard de gran acceptació.

2.1. Introducció

CORBA[14] és un *framework* (marc de treball, en català) proposat pel consorci OMG (en què participen més de 800 empreses) per a la creació d'aplicacions distribuïdes seguint el paradigma de l'orientació a objectes. De fet, l'especificació completa que ha creat l'OMG és l'*object management architecture* (OMA), de la qual CORBA és la part principal.

(14)De l'anglès *common object request broker architecture*.

OMG no produeix programari, sinó solament especificacions que es defineixen gràcies a la participació dels membres de l'OMG, entre els quals hi ha les principals empreses de programari. En haver-hi tantes empreses involucrades en la definició de CORBA, això afegeix una gran riquesa d'idees, i també uns estàndards àmpliament consensuats. L'inconvenient, en canvi, és que, en haver-hi tants participants, la creació d'aquestes especificacions és un procés lent (fins a l'aparició de CORBA 3 s'han necessitat uns 10 anys de treball).

Web recomanat

Trobareu les especificacions de CORBA en el web de l'OMG:
http://www.omg.org

CORBA ha anat evolucionant versió rere versió, adaptant-se a les necessitats que han anat sorgint amb els anys:

- **CORBA 1** (1991): l'objectiu principal va ser l'especificació de l'ORB i el llenguatge CORBA IDL.

- **CORBA 2** (1995): s'estandarditza la comunicació entre ORB de diferents proveïdors amb el protocol GIOP[15] i també l'especificació de la implantació sobre TCP/IP, que és el protocol IIOP[16].

- **CORBA 3** (2002): la principal novetat és el model de components de CORBA (CCM[17]), la qual cosa converteix CORBA en una plataforma de components distribuïts (fins al moment només podíem parlar de CORBA com una plataforma d'objectes distribuïts, ja que mancava de model de components). També s'afegeix, entre altres coses, suport per a crides a mètodes asíncrons i les especificacions de Minimum CORBA (per a sistemes encastats) i Real-Time CORBA (per a sistemes operatius de temps real).

(15)De l'anglès *general inter-ORB protocol*.

(16)De l'anglès *Internet inter-ORB protocol*.

(17)De l'anglès *CORBA component model*.

Web recomanat

Trobareu la història detallada de CORBA a http://www.omg.org/gettingstarted/history_of_corba.htm.

Alguns dels aspectes principals del **CCM** són:

- Els components CORBA s'executen dins d'un **contenidor** que ofereix serveis de transaccionalitat, seguretat i persistència.

- Estandarditza l'empaquetament (en arxius ZIP), configuració (en arxius XML) i desplegament dels components, amb l'objectiu de facilitar la creació d'un mercat de components.

- Integració amb **EJB**: un EJB desplegat en un contenidor CCM és vist com un component CORBA, que pot ser utilitzat per un client CORBA. D'igual manera, un client Java pot accedir a un component CORBA com si fos un EJB.

2.2. Arquitectura de CORBA

A continuació analitzem l'arquitectura de CORBA i l'ORB[18], que és una de les seves peces fonamentals.

[18] De l'anglès *object request broker*.

2.2.1. ORB

El cor de CORBA és l'ORB, que actua com un bus d'objectes, per mitjà del qual els objectes interactuen transparentment amb altres objectes situats en local o en un sistema remot.

Un objecte CORBA es representa com una interfície amb un conjunt de mètodes. Un client adquireix la referència de l'objecte CORBA que li interessa i la utilitza per a fer les crides a les funcions remotes, d'igual manera que si l'objecte estigués en el seu propi espai d'adreces. L'ORB és la peça encarregada d'oferir els mecanismes necessaris per a trobar el servent[19], preparar-lo per rebre una petició, enviar-li la petició i, finalment, retornar la resposta corresponent al client.

[19] La implementació de l'objecte servidor es denomina *servent* (*servant*) segons la terminologia CORBA.

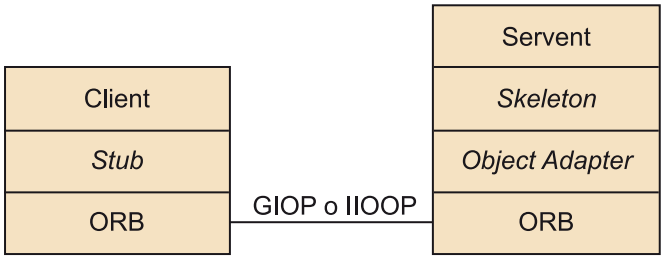

Figura 5. Arquitectura bàsica de CORBA

Normalment fem crides estàtiques a funcions d'objectes CORBA, de manera que el client coneix *a priori* la interfície del servent, i disposem de l'*stub* del client i l'*skeleton* (esquelet)[20] del servidor. Els passos s'enumeren a continuació:

[20] L'*stub* del servidor es denomina *skeleton* (esquelet) en terminologia CORBA.

1) El client invoca la funció remota per mitjà de l'*stub*, la petició arriba a l'ORB del client i, si es tracta realment d'una crida a un servidor remot, es passa la petició a l'ORB remot per mitjà de GIOP/IIOP (de fet, podríem parlar directament d'IIOP, ja que el més habitual és que les comunicacions entre ORB es facin per mitjà d'HTTP).

2) L'*object adapter* és el component de CORBA encarregat de fer d'intermediari entre les peticions que es reben i els servents. En rebre una petició, l'*object adapter* comprova si l'objecte servent està creat i, si no ho està, crea una instància, l'activa i li envia la crida corresponent.

3) El servent executa la funció sol·licitada, li passa el resultat (si n'hi ha) a l'esquelet i la resposta arriba a l'objecte client per mitjà del seu *stub*.

Object adapter

Inicialment, l'*object adapter* no era estàndard, amb la qual cosa variava d'una implementació de CORBA a una altra. Actualment, ja s'ha estandarditzat sota el nom de *POA* (*portable object adapter*), la qual cosa permet que les aplicacions CORBA siguin compatibles entre diferents implementacions de CORBA.

2.2.2. CORBA dinàmic

Encara que menys usual, també podem fer invocacions dinàmiques a mètodes d'objectes CORBA, és a dir, sense la necessitat de disposar d'*stubs* ni esquelets. En lloc seu utilitzarem la **interfície d'invocació dinàmica (DII)** i la **interfície d'esquelets dinàmics (DSI)**, tal com veurem a continuació.

(21)De l'anglès *dynamic invocation interface*.

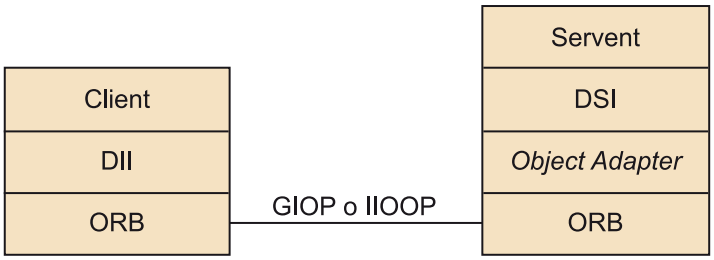

Figura 6. Arquitectura de CORBA dinàmic

1) Interfície d'invocació dinàmica (DII[21])

El client desconeix la interfície del servent en temps de compilació, ja que no té l'*stub*. En aquest cas, el client sol·licita l'operació dinàmicament amb una petició DII, en què especifica el nom de l'operació que vol, el tipus i valor dels seus paràmetres i el tipus del valor de retorn. Una vegada ha preparat la petició, crida l'operació remota amb una crida a `invoke()`.

2) Interfície d'esquelets dinàmics (DSI[22])

(22)De l'anglès *dynamic skeleton interface*.

És la contrapartida al DII, però en el servidor; és a dir, que de vegades ens podem trobar amb certs tipus d'aplicacions de servidor (com passarel·les o monitors) que no puguin saber *a priori* quins tipus d'objectes hauran de servir i, per tant, no puguin disposar dels esquelets dels servents en temps de compilació.

Un esquelet estàtic té la informació dels tipus dels paràmetres i tipus de retorn, la qual cosa permet al compilador d'IDL fer optimitzacions avançades. En canvi, per mitjà de DSI és necessari esbrinar la informació dels tipus en temps d'execució.

Per tant, l'avantatge de DSI és la seva flexibilitat, però té els inconvenients que és menys eficient i més complex de programar.

3) Repositori d'Interfícies (IFR[23])

(23)De l'anglès *interface repository*.

El repositori d'interfícies és un servei que permet emmagatzemar les definicions IDL dels servents; i en qualsevol moment els podem sol·licitar dinàmicament. En conjunció amb DII i DSI, ens permet construir clients totalment dinàmics, ja que podem sol·licitar a IFR les metadades d'una certa operació i invocar-la per mitjà de DII i DSI.

Gràcies al repositori d'interfícies, podem construir aplicacions "genèriques", el comportament de les quals es determina en temps d'execució. Per exemple, poden servir per a simplificar el desenvolupament de ponts entre diferents tecnologies de programari intermediari com, per exemple, COM o RMI.

2.3. El llenguatge IDL

CORBA IDL és el llenguatge utilitzat per a definir la interfície dels objectes CORBA, però no es pot usar per a l'especificació de la implementació d'aquestes interfícies, ja que CORBA IDL no és un llenguatge complet de programació. La interfície consta de la declaració de les operacions, els tipus de dades que l'objecte suporta i les possibles excepcions que es generen. Una vegada escrit el fitxer IDL, el compilador IDL és l'encarregat de generar l'*stub* i l'esquelet corresponents, que posteriorment s'enllacen amb les aplicacions client i servidor de CORBA.

Les interfícies IDL són similars a classes de C++ i a interfícies de Java. Un mòdul IDL defineix l'espai de noms per a un conjunt de definicions IDL. Considerem l'exemple d'una calculadora amb les operacions de suma, divisió i reinici:

```
module CalculadoraIDL {
   attribute float memoria;

   exception DivisioPerZero {
     float n1;
     float n2;
   };

   interface Calculadora {
```

```
        float suma (in float n1, in float n2);
        float div (in float n1, in float n2) raises ( DivisioPerZero );
        oneway void reset ();
      };
  };
```

Els atributs IDL corresponen a variables de l'objecte, i si estan precedits per `readonly`, són de només lectura. En l'exemple, es mostra l'atribut `memoria`, que permet que la calculadora emmagatzemi un valor en memòria, al qual podem accedir tant per a lectura com per a escriptura.

Les operacions IDL defineixen el format de les funcions remotes de l'objecte. Una operació IDL pot rebre uns paràmetres i retornar un valor, utilitzant els tipus de dades disponibles en IDL. Per a cada paràmetre hem d'especificar el mode de pas de paràmetres que cal utilitzar:

- `in`: el paràmetre es passa del client al servent.
- `out`: el paràmetre es passa del servent al client.
- `inout`: el paràmetre es passa en totes dues direccions.

Les operacions IDL poden generar excepcions, que poden ser de dos tipus:

1) **Excepcions de sistema**: conjunt d'excepcions estàndard predefinides.
2) **Excepcions definides per l'usuari**: el programador pot crear les seves classes d'excepció pròpies, com `DivisioPerZero` en l'exemple.

Totes les operacions IDL poden generar excepcions de sistema, per la qual cosa no és necessari indicar-ho. En canvi, si una operació genera excepcions definides per l'usuari, s'ha d'indicar amb la paraula clau `raises` seguida del nom de l'excepció. En el nostre exemple, l'operació `div` pot generar una excepció de tipus `DivisioPerZero`.

Per defecte, totes les operacions són síncrones, de manera que el client es bloqueja fins que el servent hagi processat la crida i retorni un valor. Un cas especial és quan utilitzem la paraula clau `oneway`. En aquest cas el client no es bloqueja esperant la resposta, ja que les operacions d'aquest tipus no retornen mai cap valor. Un exemple és l'operació `reset` de la calculadora.

Els tipus de dades en IDL es poden classificar en els tipus següents:

a) Tipus de dades bàsiques: són els tipus de dades bàsiques de qualsevol llenguatge. Els principals tipus de dades d'IDL són:

- `short, long, long long`: valors sencers de més o menys grandària.

- `float, double, long double`: valors reals de punt flotant[24].
- `char`: caràcter (8 bits).
- `boolean`: valor booleà (*true* o *false*).
- `any`: tipus de dades universal que permet treballar amb qualsevol tipus de dades, que es determinarà en temps d'execució.

(24)Els tipus de dades de punt flotant d'IDL compleixen les especificacions ANSI/IEEE 754-1985.

b) Tipus de dades complexes: podem crear els nostres propis tipus de dades, que poden ser:

- `enum`: enumeracions.
- `struct`: estructures.
- `union`: unions.
- `string`: cadenes de caràcters[25].
- `sequence`: una espècie de matrius d'una sola dimensió, que poden tenir una grandària fixa, o no (*unbounded*).
- `arrays`: matrius d'una o múltiples dimensions, que han de tenir sempre una grandària fixa.
- `fixed`: permet definir nombres, especificant-ne el nombre de dígits totals i quants en volem dedicar a la part decimal.

(25)Els tipus de dades `char` i `string` no treballen amb dades Unicode. El que permeten és emmagatzemar qualsevol caràcter d'ISO Latin-1 (ISO 8859-1), excepte el NUL.

Una novetat de **CORBA 3.0** és que afegeix la possibilitat de passar objectes per valor, i no solament per referència. Apareix una nova construcció sintàctica en l'OMG IDL, denominada ***valuetype***, un híbrid entre `struct` i `interface`, ja que permet contenir tant atributs com mètodes. Quan es passa un `valuetype` com a argument a una operació, se'n passa una còpia, i no la seva referència. Desgraciadament, aquesta nova funcionalitat no està exempta de nombrosos problemes i limitacions, ja que si bé el pas de dades no representa cap problema, el pas de mètodes entre diferents llenguatges de programació no resulta gens simple.

2.3.1. Mapatges de llenguatges

IDL no és un llenguatge de programació complet, per la qual cosa solament serveix per a declarar les interfícies dels objectes. Per a implementar-les es pot utilitzar qualsevol llenguatge pel qual hi hagi disponible un mapatge IDL, que especifica com una interfície IDL es correspon amb una implementació en un llenguatge de programació. CORBA ofereix mapatges amb els llenguatges següents: ADA, C, C++, COBOL, Java, LISP, PL/1, Python, Smalltalk i XML.

Un mapatge assigna a cada tipus de dades IDL un tipus de dades del llenguatge de programació, i tradueix el format de cada operació IDL al format propi de cada llenguatge.

A manera d'exemple, es mostra una taula amb algunes de les conversions especificades en el mapatge d'IDL per al llenguatge de programació Java:

IDL	JAVA
`module`	`package`
`interface`	`interface`
`operació`	`mètode`
`attribute`	`variable`
`exception`	`class`
`long`	`int`
`float`	`float`
`char`	`char`
`enum, struct, union`	`class`
`string`	`java.lang.String`
`fixed`	`java.math.BigDecimal`
`any`	`org.omg.CORBA.Any`

2.4. Serveis

En l'arquitectura OMA, de la qual CORBA forma part, es defineixen un conjunt de serveis útils per a totes les aplicacions en general, que podem classificar en les categories següents:

- **Serveis bàsics**[26]: ofereixen les funcionalitats bàsiques que qualsevol objecte necessita per a comunicar-se per mitjà d'una infraestructura CORBA, com servei de noms, de temps, de seguretat, transaccions, etc.

(26)En anglès, *CORBAservices*™.

- **Serveis comuns**[27]: serveis compartits per les aplicacions, però no tan bàsics com els serveis anteriors. Aquests serveis estan especificats mitjançant la definició de les seves interfícies en IDL, i la seva semàntica està descrita en llenguatge natural. N'hi ha de dos tipus:
  - **Serveis horitzontals**: utilitzables en la majoria d'aplicacions, com internacionalització, interfície d'usuari, gestió de tasques, etc.
  - **Serveis verticals o de domini**: específics d'un domini o sector empresarial, com la medicina, les finances, etc.

(27)En anglès, *CORBAfacilities*™.

- **Objectes d'aplicació**[28]: són els objectes CORBA creats a mida en cadascuna de les nostres aplicacions.

(28)En anglès, *application objects*.

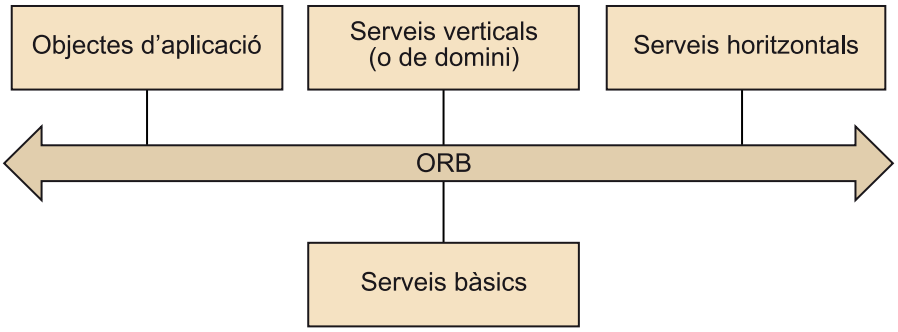

Figura 7. Arquitectura OMA

Arquitectura OMA

En l'arquitectura OMA es defineix la visió de l'OMG del desenvolupament d'aplicacions orientades a components `plug-and-play`.

Trobareu informació sobre l'arquitectura OMA a http://www.omg.org/oma.

(29)De l'anglès *request for proposals*.

2.4.1. Serveis bàsics

Els **serveis bàsics** de CORBA els va definir l'OMG entre 1993 i 1996, i es va fer en diferents etapes. En cadascuna es van sol·licitar propostes per a certs serveis en forma d'RFP[29]:

- **RFP1**: serveis de noms, notificació d'esdeveniments, cicle de vida i persistència.
- **RFP2**: relacions, externalització, concurrència i transaccions.
- **RFP3**: seguretat i temps.
- **RFP4**: llicències, propietats i consultes.

- **RFP5**: col·leccions i negociació.

ORB

Podem trobar ORB en el mercat que no incloguin els serveis bàsics de CORBA.

També podem trobar companyies que venen només els serveis, sense l'ORB.

En general, els serveis estan implementats en forma d'objectes CORBA, que poden ser activats localment o remotament.

Servei de cicle de vida

Gestiona el cicle de vida dels objectes distribuïts, quelcom bastant més complex que amb objectes locals, ja que cal tenir en compte, per exemple, la possibilitat que un objecte hagi de migrar a un altre servidor. El servei defineix un protocol per a la creació, còpia, moviment i eliminació d'objectes remots, basant-se en el concepte dels objectes factoria, que són els responsables de crear objectes de tipus específics.

Servei de col·leccions

Permet agrupar els objectes en col·leccions com llistes, piles, cues, etc., i recórrer-los amb iteradors.

Servei de concurrència

Permet gestionar els accessos concurrents de múltiples clients a un objecte remot. És similar al suport multifil dels llenguatges de programació, però en un context distribuït. El sistema està basat en bloquejos que adquireixen els clients per a determinats recursos. Aquests bloquejos poden ser de diferents tipus (lectura, escriptura, etc.) i s'hi poden aplicar diferents models de seguretat segons interessi.

Servei de consultes

El servei de consultes[30] permet fer consultes per a obtenir el conjunt d'objectes d'una col·lecció que compleixin uns criteris de cerca. El servei no imposa cap llenguatge de consultes en concret, i deixa llibertat per a utilitzar diferents llenguatges com SQL o OQL[31].

(30)En anglès, *query service*.

(31)De l'anglès *object query language*.

OQL

OQL és un llenguatge de consultes definit per l'ODMG, dissenyat per a fer consultes en SGBD orientats a objectes.

Trobareu més informació sobre el llenguatge OQL a http://www.odmg.org.

Servei de notificació d'esdeveniments

Ofereix la possibilitat de comunicació asíncrona entre objectes distribuïts. És útil en situacions en què es requereix notificar algun esdeveniment, en lloc d'una interacció síncrona entre dos objectes. Les notificacions circulen en forma d'esdeveniments a través de canals, en què els objectes interessats hauran informat prèviament del seu interès per rebre la informació.

El servei implementa un model productor/consumidor d'esdeveniments, en què es donen els passos següents:

a) Un canal emmagatzema l'interès de consumidors per certs esdeveniments.
b) Un productor envia un esdeveniment a un canal.

c) El canal reenvia l'esdeveniment a tots els consumidors interessats.

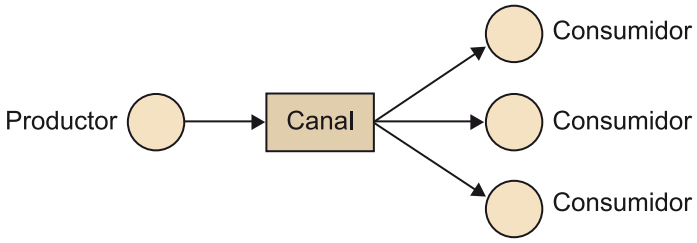

Figura 8. Notificació d'esdeveniments

Aquest model productor/consumidor pot ser al seu torn de tipus *push* o *pull*, segons si la comunicació és dirigida pel productor o pel consumidor, respectivament.

Servei d'externalització

Permet la transformació d'un objecte en un flux de bytes, per a enviar-lo per xarxa o emmagatzemar-lo en disc i, posteriorment, restaurar-lo de nou en un objecte, potencialment en un ORB o procés diferent.

Servei d'externalització

Per a poder utilitzar el servei, els objectes han d'implementar la interfície `streamable`, i implementar les operacions `externalize` i `internalize`.

Servei de llicències

Permet controlar l'accés als objectes sota un cert model de llicències, per a assegurar que solament es puguin utilitzar aquells serveis que s'hagin pagat. El client fa una petició d'un servei amb llicència, i mostra una prova de la seva llicència. El sistema verifica la llicència del servei i, si tot és correcte, en permet l'execució.

Servei de negociació

El servei de negociació[32] permet que els objectes servidors descriguin els serveis que ofereixen al sistema, i els clients descriguin les propietats de l'objecte que volen, i llavors el servei s'encarrega de buscar les correspondències entre serveis i clients.

(32)En anglès, *trader service*.

Servei de noms

El servei de noms[33] associa un nom a cada objecte servidor, per a poder-lo buscar posteriorment per mitjà del nom. Els noms estan en un cert context, que és un espai de noms en què no hi pot haver dos objectes amb el mateix nom. A més, un context pot contenir altres contextos, formant una estructura jeràrquica.

(33)En anglès, *naming service*.

Exemple de contextos i objectes

En l'exemple següent, tenim el context *Intranet*, que conté les referències als objectes *vacances* i *despeses*, i el context *Extranet* amb l'objecte *demanats*. Per a accedir a l'objecte *despeses*, sol·licitarem al servei de noms l'objecte que tingui assignada la cadena *Intranet/despeses*.

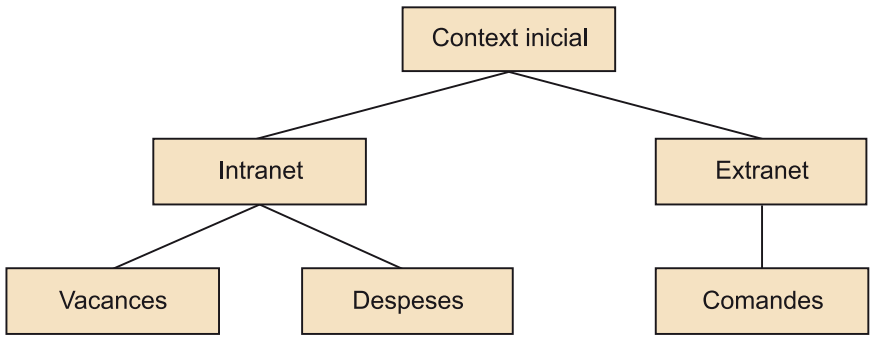

Figura 9

Servei de persistència

La persistència és la propietat d'un objecte de sobreviure al temps de vida dels processos en què és executat. Per a això, l'estat dels objectes es desa en un magatzem de dades (un SGBD o un sistema de fitxers). Per al client que accedeix a un objecte, li és transparent el fet que l'objecte ja estigui en memòria, o hagi de ser recuperat d'un SGBD.

Servei de propietats

Permet, en temps d'execució, assignar propietats als objectes. Cada propietat consta d'una parella nom/valor.

Servei de relacions

Permet crear relacions entre objectes distribuïts, sense haver-los de modificar, i sense que els objectes mateixos ho sàpiguen. Es defineix el tipus de relació, els rols que intervenen i la cardinalitat de cada rol.

Servei de seguretat

Permet afegir seguretat a una aplicació distribuïda. Suporta l'autenticació d'usuaris, el control d'accés, les funcions d'auditoria, l'habilitació de canals de comunicació segura i els esdeveniments no repudiables.

Servei de temps

Permet obtenir la data/hora actual en una aplicació distribuïda, i assegura una certa precisió entre els diferents sistemes. També permet crear esdeveniments disparats per temps, utilitzant temporitzadors i alarmes.

La representació de temps utilitzada és la UTC[34], que utilitza intervals de temps de 100 ns, amb origen en les zero hores (hora de Greenwich) del 15 d'octubre de 1582.

(34)De l'anglès *universal time coordinated*.

Servei de transaccions

Defineix les interfícies que permeten als objectes distribuïts establir interaccions transaccionals, formades per seqüències de crides a objectes remots. Si es produeix algun error, es desfà la transacció (*rollback*), i queda tot com si la transacció mai no hagués tingut lloc.

Transaccions distribuïdes

Les transaccions distribuïdes s'implementen mitjançant el protocol `two-phase commit`, que consta de dues etapes. En la primera etapa, es pregunta a tots els sistemes involucrats en la transacció si podem desar els canvis. Si tots responen afirmativament, en la segona etapa es comunica a tots els sistemes que desin els canvis. Però en canvi, si algun sistema respon negativament, s'avisa a tots els sistemes perquè desfacin els canvis fets.

2.4.2. Serveis horitzontals

Els **serveis horitzontals** són aquells que es poden utilitzar en la majoria d'aplicacions, però no són serveis tan bàsics com els comentats en l'apartat anterior. Els serveis actualment definits i acceptats són els següents:

- **Internacionalització**: permet representar les dates, hores i nombres de manera localitzada, segons les preferències culturals dels usuaris.

- **Agents mòbils**: suporta la interoperabilitat entre agents de diferents fabricants. Especifica aspectes com la gestió dels agents, la transferència d'agents entre sistemes, la identificació dels agents, etc.

Web recomanat

Podeu consultar la llista completa de serveis de domini a http://www.omg.org/ technology/ documents/domain_spec_catalog.htm.

2.4.3. Serveis verticals o de domini

Els **serveis verticals** són aquells específics d'un domini o sector empresarial. Alguns dels serveis ja definits per l'OMG són els següents:

- ***Air traffic control***: servei per a implementar sistemes de control de trànsit aeri que segueix el model MVC (model vista controlador).
- ***Audio / visual streams***: gestió d'intercanvis de reproducció en temps real d'àudio i vídeo que assegura una certa qualitat de servei.
- ***Bibliographic query service***: consultes a un repositori bibliogràfic.
- ***Biomolecular sequence analysis***: anàlisi, representació i manipulació de seqüències de dades biomoleculars.
- ***Computer aided design***: interoperabilitat entre eines CAD, CAM o CAU (disseny, fabricació i enginyeria assistida per ordinador).
- ***General ledger***: interoperabilitat entre sistemes de comptabilitat.
- ***Genomic maps***: representació de mapes de genoma i el seu contingut.
- ***Telecom service***: perquè els operadors de telecomunicacions puguin oferir serveis de control de crides, localització d'usuaris, etc.
- ***Workflow management facility***: gestió de fluxos de treball.

2.5. Implementacions de CORBA

Hi ha multitud d'implementacions de CORBA, entre les quals destaquem Orbix i Visibroker, com a comercials, i omniORB, com a gratuïta.

1) Orbix

Iona, fabricador d'Orbix, és l'empresa que ha desenvolupat les primeres implementacions de CORBA i les de més difusió, incloent-hi compiladors d'IDL per a C++, Smalltalk i Java. Orbix està implementat en C++ com un conjunt de biblioteques, una per als clients i una altra per als servents, un repositori d'implementacions on es registren els servents, i un dimoni que s'encarrega de l'activació de processos. Orbix incorpora els filtres i els *smart proxies*, figures exclusivament seves que permeten utilitzar tècniques reflexives per a afegir seguretat i adaptabilitat als objectes CORBA, construir memòries cau per a millorar l'eficiència o incloure depuradors en les aplicacions.

Web recomanat

Trobareu més informació sobre Iona a http://www.iona.com.

Iona també ha desenvolupat altres productes, com **OrbixDesktop**, que és una implementació per a Windows de CORBA, que permet integrar objectes COM en CORBA i suportar interfícies duals. Un altre producte és **OrbixWeb**, una implementació de CORBA escrita en Java que utilitza els mecanismes de Java (*applets*) per a comunicar objectes.

2) Visibroker

Visigenic va ser la companyia escollida per Netscape per a incloure la seva implementació de l'ORB de CORBA (Visibroker) en el seu navegador. Igual que OrbixWeb, **Visibroker** està desenvolupada completament en Java, i utilitza IIOP per a comunicar objectes CORBA. Una característica interessant de Visibroker és que suporta replicació d'objectes, la qual cosa pot resultar de molta utilitat, tant per a equilibrar la càrrega de les aplicacions distribuïdes com per a implementar mecanismes d'alta disponibilitat i tolerància a fallades.

Web recomanat

Trobareu més informació sobre Visibroker a http://www.borland.com/us/products/visibroker.

3) omniORB

Implementació de codi obert d'un ORB, que ha obtingut la certificació de The Open Group per a ser compatible amb la versió 2.1 de CORBA, i que a més suporta moltes de les característiques de CORBA 2.6. Ofereix suport per als llenguatges C++ i Python, i està disponible sota llicència LGPL. Actualment està reconegut per ser un dels ORB de C++ més ràpids.

Web recomanat

Trobareu més informació sobre omniORB a http://omniorb.sourceforge.net.

Resum

Aquest mòdul ha ofert una àmplia perspectiva de les diferents tecnologies per al desenvolupament d'aplicacions distribuïdes. Entre aquestes, s'ha estudiat amb especial atenció CORBA. En els propers mòduls s'estudiaran detalladament les tecnologies RMI, Java EE i els serveis web.

D'aquí a uns anys, potser una tecnologia distribuïda superarà clarament la resta, o les diferents tecnologies podran interoperar sense problemes, però actualment hi ha una forta competència motivada en gran mesura per interessos comercials, que en complica la interoperabilitat. Per aquesta raó, en aquests moments no hi ha cap altra opció que conèixer les diferents possibilitats tecnològiques existents i saber quins són els seus punts forts i febles, perquè, arribat el cas, puguem decidir utilitzar la que resulti més convenient.

Activitats

1. Els serveis comuns de CORBA estan actualment en fase de definició, amb la qual cosa constantment van apareixent nous serveis, i també noves versions dels ja existents. Consulteu a la pàgina web de l'OMG la llista actualitzada dels serveis comuns i compareu-la amb l'oferta en aquest mòdul.

A la pàgina web de l'OMG els serveis comuns apareixen dividits entre horitzontals i verticals, amb els noms següents:

- Serveis horitzontals: *CORBAfacilities specifications*
- Serveis verticals: *domain specifications*

2. Actualment .NET Remoting no permet cridar servidors CORBA, però hi ha diversos projectes com IIOP.NET o MiddCor.NET que permeten fer crides a objectes CORBA des d'una aplicació .NET. Informeu-vos sobre aquests productes i intenteu entendre com funcionen. Mireu també si sou capaços de trobar algun altre producte similar.

3. Informeu-vos sobre els serveis web que ofereixen Google i Amazon i intenteu respondre les qüestions següents:

a) Quines operacions permeten fer?
b) Són serveis gratuïts o de pagament?

4. En l'adreça http://www.programmableweb.com/apis es troba una important recopilació de serveis web.

a) Hi ha algun servei web que permeti enviar un fax?
b) I consultar la predicció del temps?

Nota

Si en el moment de fer l'exercici la pàgina http://www.programmableweb.com/apis no es troba disponible, cal buscar prèviament informació relativa a directoris públics de serveis web.

Exercicis d'autoavaluació

1. Quina diferència hi ha entre RMI tradicional i RMI-IIOP?

2. Definiu COM, DCOM i COM+.

3. Enumereu algunes de les principals novetats de CORBA 3 respecte a CORBA 2.

4. Què és un assemblament de .NET?

5. Quina relació hi ha entre COM+, .NET Remoting i WCF?

6. Per a què serveixen SOAP, WSDL i UDDI?

Solucionari

Exercicis d'autoavaluació

1. RMI tradicional utilitza el protocol JRMP per a les comunicacions, mentre que RMI-IIOP utilitza el protocol IIOP. L'avantatge d'IIOP és que permet tenir interoperabilitat entre objectes Java i CORBA.

2. COM és el model de components de Microsoft. **DCOM** és un protocol que permet fer crides remotes entre components COM. I, finalment, **COM+** és un programari intermediari de Microsoft, que facilita la creació d'aplicacions empresarials de gran escala.

3. Algunes de les principals novetats de CORBA 3 són:

- La introducció del model de components de CORBA (CCM), la qual cosa converteix CORBA en una plataforma de components distribuïts.
- Integració amb EJB.
- Suport per a crides a mètodes asíncrons.

Especificacions de Minimum CORBA i Real-Time CORBA.

4. Un assemblament és la unitat lògica d'empaquetament en .NET.

5. .NET Remoting és la tecnologia de .NET per a la comunicació entre components distribuïts (reemplaça DCOM), mentre que COM+ és un conjunt de serveis que poden ser utilitzats per qualsevol component de .NET. A partir de la versió 3 del *framework* .NET, Microsoft aposta per WCF com a tecnologia per a la comunicació entre components distribuïts, i substitueix .NET Remoting.

6. SOAP permet codificar les peticions i respostes dels serveis web en missatges XML. **WSDL** permet descriure els serveis web. I **UDDI** permet crear registres en què les empreses donin d'alta els serveis web que ofereixen, i en què els clients poden fer consultes i obtenir les referències als serveis web que compleixin certs criteris de cerca.

Glossari

COM (*component object model*) *m* Model de components de Microsoft.

CORBA *f* Especificació del consorci OMG que defineix un estàndard per a la creació d'aplicacions orientades a objectes distribuïdes.

DCOM (*distributed COM*) *m* Protocol de Microsoft que amb una infraestructura basada en RPC permet les crides a components remots.

EJB (*enterprise Java beans*) *f* Tecnologia de components distribuïts de l'empresa Sun que forma part de l'edició empresarial de Java.

IDL (*interface definition language*) *m* Llenguatge mitjançant el qual es defineix la interfície amb les funcions que es podran cridar remotament.

MSIL (*Microsoft intermediate language*) *m* Llenguatge intermedi definit per Microsoft que permet tenir independència del processador.

MTS (*Microsoft transaction server*) *m* Programari intermediari de Microsoft que facilita la creació d'aplicacions empresarials de gran escala.

RMI (*remote method invocation*) *f* Tecnologia que permet fer crides remotes entre objectes de diferents màquines virtuals de Java.

RPC (*remote procedure call*) *m* Protocol que permet crear aplicacions client/servidor basades en la crida a procediments remots.

SOAP (*simple object access protocol*) *m* Protocol basat en XML, per a l'intercanvi de missatges entre diferents sistemes.

***stub* (estub)** *m* Rutina encarregada de fer les crides remotes.

UDDI (*universal description, discovery and integration*) *m* Estàndard que defineix com crear registres de serveis web.

WCF (*Windows comunication foundation*) *f* Tecnologia de Microsoft per a comunicacions distribuïdes.

WDSL (*web services description language*) *m* Llenguatge XML que permet descriure les interfícies dels serveis web.

XML (*extensible markup language*) *m* Metallenguatge definit el 1998, que permet definir llenguatges de marques.

Bibliografia

Cerami, E. (2002). *Web services essentials*. O'Reilly.

Emmerich, W. (2000). *Engineering Distributed Objects*. Nova York: John Wiley & Sons, Inc.

Emmerich, W. (2002). "OMG/CORBA: An Object-Oriented Middleware". *Encyclopedia of Software Engineering* (pàg. 902-907). Nova York: John Wiley & Sons, Inc. Disponible a http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/Encyclopedia/corba.pdf.

Löwy, J. (2001). *COM & .NET Component Services*. O'Reilly.

MacDonald, M. (2003). *Microsoft® .NET Distributed Applications: Integrating XML Web Services and .NET Remoting*. Microsoft Press.

Myerson, J. (2002). *The Complete Book of Middleware*. Auerbach Publications.

Nash, S. (1999, desembre). "RMI over IIOP". *Java World*. Disponible a http://www.javaworld.com/javaworld/jw-12-1999/jw-12-iiop_p.html.

Orfali, C. J. (1999). *Client/Server Survival Guide* (3a. ed.). Nova York: John Wiley & Sons, Inc.

Pahl, S. (2002). *Understanding Enterprise Services (COM+) in .NET*. Microsoft Corporation. Disponible a http://msdn.microsoft.com/en-us/library/ms973847.aspx.

Suresh, G. (1998). A Detailed Comparison of CORBA, DCOM and Java/RMI. Disponible a (darrer accés: gener 2012) http://my.execpc.com/~gopalan/misc/compare.html.

Weerawaran, S. (2005). *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall.