SOA

Arquitectures orientades a serveis

Antoni Oller Arcas

PID_00185402

Universitat Oberta de Catalunya

Índex

Introducció

Tots els mòduls explicats prèviament s'han basat a especificar i fer una descripció arquitectònica de sistemes distribuïts oberts mitjançant punts de vista independents amb un nivell d'abstracció respecte a la tecnologia en què s'implementaran. Successius mòduls presentaven diverses tecnologies per a la construcció de sistemes distribuïts.

En el mòdul "Java RMI" es descrivia el concepte d'*invocació a procediments remots* i s'explicaven les característiques i funcionament d'RMI (de l'anglès *remote method invocation*). En el mòdul "Java EE" es proposava una estratègia per a la construcció de sistemes distribuïts empresarials basada en la plataforma Java EE (de l'anglès *Java enterprise edition*). Aquesta plataforma ofereix una sèrie d'especificacions completa per a la construcció d'aplicacions, que habitualment s'ofereixen a l'usuari amb una interfície (Web). Aquest tipus d'aplicacions es denominen *B2C* (de l'anglès *business-to-consumer*), ja que es tracta de sistemes que relacionen l'empresa (i el seu negoci, per mitjà dels seus serveis) directament amb els clients (usuaris). Hi ha un altre tipus de sistemes, denominats *B2B* (de l'anglès *business-to-business*), que estableixen relació d'empresa a empresa. Aquest tipus de sistemes (B2B) estan basats en **serveis** als quals accedeixen terceres parts (i són reutilitzats per aquestes) i se solen implementar seguint una **SOA** (**arquitectura orientada a serveis**, de l'anglès *service oriented architecture*).

D'altra banda, apareix la necessitat de construir aplicacions més complexes, basades en **la composició de serveis** (ja existents). Les aplicacions es construeixen mitjançant l'**orquestració o la coreografia de serveis** que són oferts per terceres parts. Aquest model afavoreix la **reutilització** dels serveis i augmenta la productivitat en la construcció de les aplicacions.

Aquest mòdul tractarà d'explicar tots aquests conceptes i presentarà estratègies tecnològiques basades en **serveis web** (lleugers i pesants) per a la construcció d'aplicacions que segueixen un paradigma d'arquitectura orientada a serveis.

Finalment, es mostrarà una possible manera de fer un disseny d'aplicacions orientades a serveis amb UML, amb un exemple de perfil UML per a serveis web; i es donaran algunes recomanacions per a fer el pas de l'especificació a la implementació del sistema.

Objectius

L'objectiu bàsic del mòdul és presentar diverses tècniques per a la implementació d'**aplicacions orientades a serveis**. Els objectius més concrets són:

1. Analitzar l'evolució històrica d'Internet i entendre com s'han transformat els paradigmes de computació.

2. Entendre el concepte *SOA* (arquitectura orientada a serveis).

3. Conèixer el concepte de *servei* i *serveis web*.

4. Conèixer el concepte de *recurs* i *arquitectura orientada a recursos*.

5. Estudiar les diverses estratègies d'implementació de serveis web: serveis web pesants i serveis web lleugers.

6. Conèixer les tècniques de composició de serveis: **orquestració** i **coreografia**.

7. Veure una possible manera de dissenyar una aplicació orientada a serveis, a partir de l'especificació independent de la tecnologia obtinguda (tal com s'ha explicat en mòduls previs).

1. Introducció a les arquitectures orientades a serveis

En aquest apartat, es farà una descripció del concepte d'*arquitectura orientada a serveis* i les claus per al desenvolupament de serveis que utilitzin aquest estil arquitectònic. En primer lloc, es descriu l'evolució històrica dels serveis sobre Internet; posteriorment, es descriu el concepte *SOA* i alguns elements que poden participar en l'arquitectura orientada a serveis, les tecnologies de serveis web, l'orquestració i la coreografia.

1.1. Evolució històrica dels serveis a Internet

L'origen d'Internet es remunta als anys seixanta en un projecte d'investigació en comunicació de xarxes en una àrea militar. Aquesta investigació va produir una xarxa experimental de quatre nodes (ArpaNet) que va transmetre informació fiable en una xarxa en condicions hostils com fallades i caigudes de nodes. Apareix un nou paradigma de computació; fins llavors els sistemes es dissenyaven i implementaven de manera totalment **centralitzada**, i amb l'aparició de la xarxa es permet descentralitzar o distribuir les aplicacions en un esquema client-servidor.

ArpaNet

A més de l'equipament, ArpaNet va oferir serveis, com el correu (1972), el protocol TCP/IP (1974) i serveis com FTP, grups de notícies (*news*) i el World Wide Web (WWW), que ha estat el servei que, finalment, ha popularitzat Internet.

L'auge del Web marca la consolidació d'Internet sobretot per la seva evolució. En el **Web 1.0** les aplicacions estan limitades a un servidor que ofereix uns serveis i continguts; i un usuari amb un rol passiu de consumidor de serveis. En el **Web 2.0** els usuaris actuen com a generadors de continguts utilitzant aplicacions dinàmiques i es converteixen en els participants actius en el Web. La consolidació de protocols com HTTP facilita la creació d'aplicacions i serveis sobre el Web. Aquests serveis inicialment són construïts sense unes especificacions estàndard que defineixin les interfícies, regles de codificació de paràmetres i resultats; o el mecanisme d'invocació. Posteriorment, es crearan especificacions i noves propostes d'estils arquitectònics per a evolucionar la construcció de serveis sobre el Web.

HTTP

HTTP és un protocol d'aplicació basat en un paradigma de petició-resposta. Habitualment, funciona sobre el port 80 i sol tenir facilitat per a travessar entorns hostils (tallafocs, NAT).

En aquest context els serveis evolucionen envers una arquitectura orientada a serveis (SOA) i les aplicacions són construïdes com un núvol de serveis cooperats, en el qual hi ha interfícies i facilitat per a reutilitzar els serveis que ofereixen terceres parts.

1.2. Web 2.0

Un dels passos més grans en l'evolució del Web és en el moment en què es permet als usuaris participar activament a Internet (**democratització del Web**), i una de les maneres en què poden tenir un rol més actiu és millorant l'experiència de l'usuari. Per a millorar aquesta experiència de l'usuari, en el Web 2.0 s'utilitzen aplicacions RIA[1], que permeten la creació de noves aplica-

(1)De l'anglès *rich Internet applications*.

cions i un model que pretén portar les aplicacions d'escriptori al Web. Actualment, hi ha aplicacions implementades que integren components Adobe Flash en lloc d'una pàgina web i que interactuen amb els components de servidor quan sigui necessari. En una mateixa línia, HTML5 estableix nous elements i atributs que reflecteixen el típic ús modern dels llocs web.

Però el Web 2.0 no solament és un canvi de paradigma d'ús del Web, sinó que també apareixen aportacions tecnològiques significatives com la **transició i l'ús d'etiquetes semàntiques** (utilitzant XML). L'extensibilitat d'XML permet entendre un document creat per terceres parts i processar-lo. Apareixen abundants especificacions web 2.0 basades en XML com RSS, Atom, SOAP, XHTML, HTML5.

HTML5

És la cinquena revisió del llenguatge HTML. Proporciona nous elements (*canvas*, *audio*, *video*, *device*) per al llenguatge HTML5 i noves funcionalitats: WebGL, *sockets* web (*web sockets*), geolocalització, arrossegar i deixar anar (*drag&drop*)...

Hi ha una altra tecnologia anomenada *AJAX*[2], que consisteix que el client fa peticions i el servidor retorna documents XML de manera asíncrona. En el client, una peça addicional (*AJAX engine*) processa la informació dels documents XML i crea o complementa un arxiu HTML. Aquest motor (*AJAX engine*) determina si la petició és necessària per a cada acció de l'usuari. De vegades aquest comportament, basat en XML, pot ser substituït per unes estructures basades en JavaScript denominades *JSON*.

(2)De l'anglès *asynchronous Javascript and XML*.

Vegeu també

En l'apartat "Serveis web basats en REST" d'aquest mòdul es veurà amb més detall el mecanisme de codificació de dades JSON.

Els nous navegadors com l'IExplorer, Firefox, Safari, Opera o Google Chrome (entre d'altres) incorporen aquestes tecnologies que no solament processen HTML, sinó que també reprodueixen contingut audiovisual, com vídeos, arxius Flash, AVI/DIVX. Encara hi ha moltes funcionalitats que no són implementades en els diferents navegadors, fet que provoca incompatibilitats entre aquests, fruit del disseny específic de cada navegador per a solucionar aquestes incompatibilitats. Aquests problemes de compatibilitat entre navegadors dificulten la introducció de les tecnologies web 2.0.

La **participació dels usuaris** i la **intel·ligència col·lectiva** són conceptes web 2.0 que proporcionen aplicacions diferents i revolucionàries en termes de disseminació de la informació. Un exemple d'això és l'aparició dels **wikis** (el 1995), que permeten noves entrades d'informació que poden ser afegides per qualsevol usuari i corregit o modificat per altres. En altres aplicacions, com els **blogs** (1993) i la sindicació, que estan basats en usuaris, o aplicacions generadores de continguts (text pla, imatges o arxius multimèdia), qualsevol usuari es pot subscriure a aquests canals (*feed*) actuant com a seguidors (*follower*). RSS/Atom és un format/estàndard que defineix la gestió de la subscripció dels seguidors per a distribuir la informació actualitzada. Aquesta revolució afecta les notícies al món, i obre un camí fàcil i directe de disseminació de notícies. A més, el *podcast* s'uneix a aquest tipus d'aplicacions incloent-hi contingut multimèdia.

El 2005 apareix una altra forma de blogs: els **microblogs**. El contingut dels **microblogs** és de mida petita i consisteix en frases petites (en Twitter 140 caràcters), una imatge o un vídeo. Els microblogs tenen el potencial de convertir-se

en un mitjà informal de comunicació, especialment per al treball cooperatiu dins de les organitzacions usat per al màrqueting i les relacions socials (xarxa social). El nou moviment de les xarxes socials apareix permetent als usuaris que estiguin connectats entre ells i puguin compartir, en grups, continguts, les seves vides i sentiments (missatges, imatges, vídeos). Aquests efectes han estat magnificats en un context web 2.0, ja que les interconnexions entre usuaris han estat multiplicades en una línia en què les individualitats es converteixen en grups i societats fàcilment i ràpidament.

Altres aplicacions web 2.0 són:

- Les **remescles** (*mashup*) són aplicacions que permeten crear contingut mitjançant la combinació de continguts de terceres parts.

- **CMS**[3] són aplicacions que permeten la creació i gestió de continguts i el seu flux de treball (*workflow*).

(3)De l'anglès *content management system*.

Mashup

La traducció literal de *mashup* pot ser *farinetes*. Exemples: Amazon, eBay, Flickr, Yahoo, YouTube, Google Maps...

Twitter

Una de les funcionalitats més interessants de Twitter és l'anàlisi de les paraules més populars o emergents (*trending topic*) a la xarxa social de microblocs, i a més pot estructurar el contingut semànticament per mitjà d'etiquetes (*hashtags*).

1.3. Web semàntic

Les noves aplicacions i l'increment i la participació dels usuaris han augmentat la complexitat i els requisits de la Xarxa. Actualment, es genera gran volum d'informació i apareixen noves necessitats que han de ser cobertes, com nous sistemes de classificació i cerca intel·ligent de continguts que tinguin en compte nous paràmetres: sensibilitat del context, localització, nomadisme, les capacitats dels dispositius, ubiqüitat, mobilitat, etc.

Una de les iniciatives consisteix a tenir sistemes de classificació basats en **folksonomies**, és a dir, en la creació i la gestió col·laborativa de paraules clau o etiquetes per a categoritzar o identificar contingut. Les **etiquetes** (*tags*) són paraules clau o termes assignats a peces d'informació per part d'un usuari o un autor. La definició etimològica del terme defineix aquest concepte com a "classificació gestionada per la gent". Per aquest motiu, les folksonomies habitualment estan associades a entorns socials, en què els usuaris col·laboren en la descripció d'una mateixa peça d'informació. Hi ha dos tipus de folksonomies: en les primeres el creador del contingut no té cap influència sobre les etiquetes associades al contingut. En aquest cas, els usuaris són els únics que etiqueten el contingut. En el segon tipus de folksonomies, només el creador del contingut o un petit grup de persones està habilitat per a etiquetar el contingut.

No obstant això, la falta d'un control de la terminologia provoca problemes. Per exemple, diferents etiquetes (sinònims) amb el mateix significat. La mateixa etiqueta amb significats diferents (homònims) o la mateixa etiqueta amb diferents significats relacionats entre si (polisèmia). La idea d'afegir metainformació al World Wide Web ens condueix, mitjançant l'ús de llenguatges de descripció semàntica, al Web semàntic, i el nou concepte de futur Web.

Amb aquest propòsit, el W3C[4] ha fomentat el desenvolupament d'OWL[5], que consisteix en una família de llenguatges de representació del coneixement per a la creació d'ontologies. El concepte d'***ontologia*** correspon a una representació formal del coneixement com un conjunt de conceptes dins d'un domini. Les ontologies proveeixen d'un vocabulari compartit que pot descriure un domini i les interrelacions dins d'aquest.

(4)Sigla del World Wide Web Consortium.

(5)De l'anglès *ontology web language*.

2. El concepte *SOA*: arquitectures orientades a serveis

L'arquitectura orientada a serveis (SOA[6]) és un concepte d'arquitectura de programari que descriu l'ús de serveis per a proveir de solucions a requisits de negoci.

(6)De l'anglès *service-oriented architecture*.

Es defineix **SOA** com un paradigma per a l'organització i la utilització de capacitats distribuïdes que pot estar ofert sobre diferents dominis. Proveeix d'una manera normalitzada per a descobrir, oferir, interactuar amb un servei per a obtenir els efectes desitjats i definits per les seves precondicions/postcondicions.

SOA crea una arquitectura flexible i habilita la implementació de complexes arquitectures que reflecteixin l'organització. A més, proveeix d'un mapa ben definit per a la invocació de serveis que habilita la interacció entre diferents sistemes (per exemple, sistemes de propietat) i serveis externs (*third party services*).

Els beneficis de l'ús del paradigma de l'arquitectura orientada a serveis són els següents:

- Millora del temps per a fer canvis en processos.
- Capacitat per a desenvolupar lògiques de negoci basades en la subcontractació (*outsourcing*).
- Capacitat per a dirigir models de negoci amb altres entitats (socis, proveïdors).
- Capacitat per a reemplaçar elements d'una capa d'una aplicació SOA sense provocar trastorns en els processos de negoci.
- Simplicitat per a integrar tecnologies diverses.

Subcontractació (*outsourcing*)

La subcontractació és una tècnica que utilitzen les empreses i que es basa en l'externalització de certes tasques a empreses especialitzades per a reduir costos.

Quan es dissenya un sistema distribuït seguint un paradigma SOA, s'ha de tenir en compte la metodologia de disseny basada en SOA i els diferents perfils tecnològics disponibles.

2.1. Metodologia SOA

SOA proporciona una metodologia i un marc de treball (*framework*) per a descriure les lògiques de negoci i proporciona la integració i la consolidació de les activitats. S'ha de tenir en compte que:

- Les aplicacions són bàsiques i estan basades en sistemes implementats sota qualsevol arquitectura o tecnologia, situats geogràficament dispersos i en diferents dominis (i propietaris).

- Les capacitats són exposades com a serveis. Habitualment serveis web, però no exclusivament.

- S'habilita l'intercanvi d'informació entre els elements interns i les lògiques de negoci col·laboratives.

- Es defineixen els processos en termes de lògiques de negoci i les seves necessitats.

- Els serveis són desplegats per als usuaris finals en una determinada ubicació (punt d'entrada).

- El model està orientat a serveis com una funció sense estat, que accepta invocacions i retorna respostes per mitjà d'una interfície coneguda. Els serveis no depenen de l'estat de les seves funcions i processos, i la tecnologia específica usada per a proveir del servei no és una part de la seva definició. Es permet l'ús de serveis asíncrons.

- Un servei sense estat no depèn d'una condició preexistent. En la SOA, un servei no és dependent de cap altre. Els serveis obtenen la informació que necessiten per a respondre la petició. Els serveis sense estat poden ser seqüenciats en un esquema basat en bus o canonades (*pipeline*), o orquestrats, per a oferir una lògica de negoci complexa.

- Els proveïdors són els responsables de proporcionar un servei en resposta a una petició d'un client.

- El client és un component responsable de consumir el resultat d'un servei proporcionat per un proveïdor.

- L'orquestració coordina, organitza, arbitra serveis i proveeix de lògiques addicionals per a processar informació, sense incloure informació de presentació. La metodologia de modelització de SOA és denominada *anàlisi i disseny orientat a serveis*. L'arquitectura orientada a serveis és al mateix temps un *framework* (*framework*) per al desenvolupament de programari i un *framework* per a la fase d'implementació.

2.2. Implementació de serveis SOA

Quan es descriu l'arquitectura orientada a serveis, habitualment ens referim a un conjunt de serveis que es despleguen sobre Internet o intranet utilitzant serveis web.

Hi ha diferents estàndards associats als serveis web com XML, HTTP, SOAP, WSDL, UDDI o REST, XML, JSON, encara que SOA no es limita només a aquestes especificacions. Es pot dissenyar un sistema que segueixi un paradigma orientat a serveis i aplicar un perfil tecnològic diferent de l'habitual ús d'estàndards.

En un entorn SOA, els nodes d'una xarxa publiquen els seus recursos disponibles a altres participants com a serveis independents que són accessibles d'una manera estandarditzada. Moltes definicions de SOA identifiquen SOA amb l'ús de serveis web basats en SOAP (WS-*) i WSDL en la seva implementació, però és possible implementar SOA usant **qualsevol tecnologia** basada en serveis.

Protocols associats a serveis web

Hi ha multitud de protocols associats als **serveis web** basats en SOAP: *WS-addressing*, *WS-notification*, *WS-eventing*, *WS-policy*, *WS-discovery*, etc. S'abreujarà mitjançant el terme *WS-** quan es faci referència a tots aquests protocols.

Hi ha diversos estils per a implementar serveis, i en aquests materials es presentaran dues famílies de serveis web. Els **serveis web** basats en SOAP i WSDL (WS-*) i els **serveis web** basats en recursos (*RESTful web services*).

Vegeu també

En l'apartat següent veurem aquestes dues famílies de serveis web en detall.

2.2.1. Invocació a procediments remots

El mecanisme de comunicació bàsic en una arquitectura orientada a serveis és la **invocació a procediments remots (RPC)**. RPC[7] és un paradigma per a implementar un model client-servidor per a computació distribuïda. Una crida RPC és iniciada per un client que envia un missatge de petició a un servidor conegut amb la intenció d'executar un procediment especificat amb uns paràmetres. El desenvolupador d'aplicacions distribuïdes basades en RPC únicament ha d'implementar un client senzill que fa una invocació a un procediment com si estigués en local i el servidor fa la implementació del procediment en qüestió. El mecanisme d'RPC es responsabilitzarà de construir la invocació (crida i paràmetres), gestió de la capa de xarxa (tant en client com en servidor), invocació del procediment en qüestió i construcció de la resposta.

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

Vegeu també

En el mòdul "Introducció a les plataformes distribuïdes" hem vist la invocació de procediments remots RPC.

Aquest model d'RPC es proposa en la dècada de 1970; més endavant el grup de treball Object Management Group proposa un altre mecanisme basat en CORBA[8]. RMI és un mecanisme ofert pel llenguatge de programació Java i, amb l'auge d'Internet, apareixen diverses evolucions d'aquest paradigma com XML-RPC i els **serveis web**.

[8]De l'anglès *common object request broker architecture*.

CORBA

L'Object Management Group va proposar CORBA als anys noranta. Basat en el paradigma d'RPC, que funciona sobre objectes distribuïts, amb les característiques pròpies de l'orientació a objectes com encapsulació, polimorfisme i herència.

2.3. Exemple de SOA

Un exemple d'aplicació orientada a serveis podria ser una aplicació B2B d'una gran empresa de l'automoció (fabricant de vehicles) que ofereix els seus productes, amb la informació i les possibilitats de personalització de cadascun.

D'altra banda, un concessionari multimarca, que pot vendre vehicles de diferents fabricants o marques, pot disposar d'una aplicació web que mostri, amb la identitat corporativa del concessionari, la informació dels vehicles (de les diferents marques) amb les opcions de personalització, als seus clients (B2C). L'aplicació del concessionari accedirà als serveis que ofereix l'aplicació B2B del fabricant (o altres fabricants), per a obtenir la informació dels vehicles, que posteriorment serà presentada als usuaris amb el valor afegit que ofereixen els concessionaris. La figura 1 mostra aquest escenari.

Vegeu també

Consulteu el mòdul "Introducció a les plataformes distribuïdes" per a tenir més informació d'RMI i CORBA.

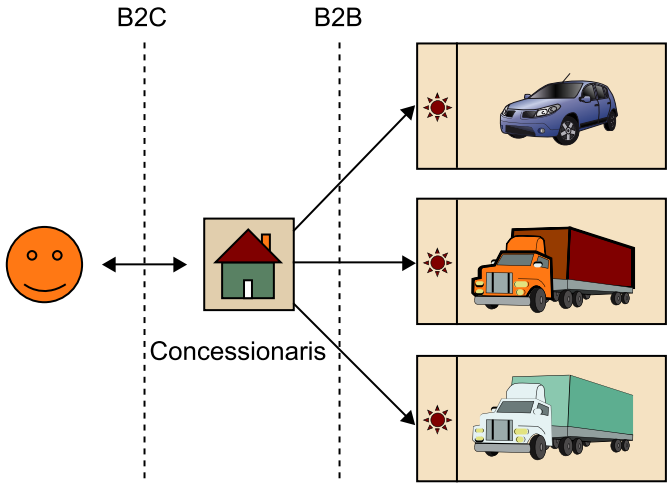

Figura 1. Cas d'ús de SOA

3. Serveis web

En aquest apartat es descriurà primer el concepte bàsic i l'arquitectura d'un servei web basat en SOAP i es descriuran les tècniques de composició de serveis, així com diverses estratègies per a implementar aquest tipus de serveis web. Posteriorment es presenten els serveis web basats en REST.

3.1. Serveis web basats en SOAP (WS-*)

Un **servei web** basat en SOAP és un sistema programari identificat per una URI, amb interfícies públiques i enllaços que es defineixen i descriuen usant XML. La definició pot ser descoberta per altres sistemes programari, i aquests sistemes solen interactuar amb el servei web de la manera prescrita per la definició, usant missatges basats en XML per mitjà de protocols estàndard d'Internet. En definitiva, un servei web exposa la seva funcionalitat a possibles consumidors en una URI i proporciona mecanismes per a invocar les seves operacions de manera remota (per mitjà d'Internet). El servei web es pot implementar en **qualsevol llenguatge** i en **qualsevol plataforma**. Els serveis web utilitzen una sèrie d'especificacions com SOAP, WSDL i UDDI, que es descriuran a continuació.

SOAP[9] és un protocol que permet la interacció amb serveis web basat en XML. L'especificació de SOAP inclou una sintaxi per a definir els missatges, regles de codificació i convencions per a representar els RPC.

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

Les especificacions de serveis es poden fer utilitzant WSDL.

WSDL[10] descriu on es localitza un servei, quines operacions proporciona, el format dels missatges que han de ser intercanviats i com és invocat el servei, a més de proporcionar informació addicional.

(10)De l'anglès *web services description language*.

La proliferació dels serveis web fa necessària l'existència de directoris públics que poden ser usats per al registre i la cerca de serveis.

UDDI[11] proporciona un mecanisme perquè els proveïdors de serveis puguin especificar els seus serveis d'una manera normalitzada, i els clients poden consultar informació d'aquests serveis.

(11)De l'anglès *universal description, discovery and integration*.

Les entrades dels UDDI consisteixen en pàgines blanques (adreça, informació de contacte, etc.), pàgines grogues (característiques basades en ontologies estàndard) o pàgines verdes (referències a especificacions de serveis).

Els serveis web WS-* utilitzen SOAP per a codificar els missatges XML que es transporten. Els missatges han de ser capaços d'interpretar-se de manera automatitzada, tal com es descriuen les operacions en WSDL. WSDL no és obligatori que estigui vinculat a un extrem SOAP, però sí que és un prerequisit per a automatitzar la generació de les classes auxiliars (*stubs* o *skeletons*) que s'utilitzen en diferents plataformes. Algunes organitzacions (com WS-I i OASIS) inclouen SOAP i WSDL en les definicions dels seus serveis web.

WS-I i OASIS

WS-I és una organització que proposa bones pràctiques en l'ús de serveis web.

OASIS és una organització/consorci que treballa sobre estàndards oberts en diferents àmbits, informàtica en el núvol (*cloud computing*), SOA, serveis web, etc.

3.1.1. Arquitectura dels serveis

L'esquema de funcionament dels serveis web basats en SOAP requereix diversos elements fonamentals, que constitueixen l'esquema normal de SOA. La figura 2 mostra el cicle de vida d'un servei web.

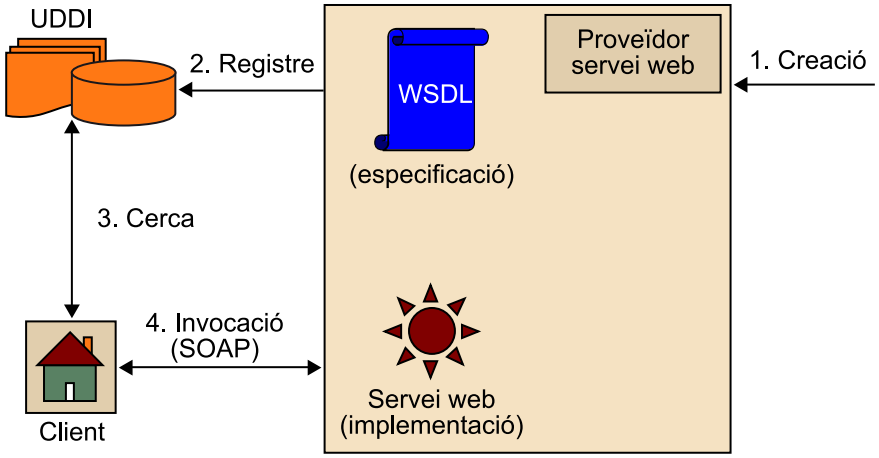

Figura 2. Cicle de vida d'un servei web

El model de funcionament es basa en **un proveïdor del servei web**, que és qui el dissenya, desenvolupa i implementa i el fa disponible per a usar-lo, ja sigui dins de la mateixa organització o al públic en general. Les operacions de publicació involucren l'anunci del servei com a tal, la qual cosa correspon a la ubicació del servei en un servidor específic i l'ús d'un servei de descripció (UDDI), perquè els clients puguin saber quines funcions té disponibles el servei web i quina informació s'ha de passar a aquestes funcions per a utilitzar-les.

D'altra banda, un **consumidor del servei**, que és qui accedeix al servei web per a utilitzar els serveis que aquest presta. Quan un consumidor vol accedir a un servei web, aquest pot fer ús d'un servidor de descobriment que permeti conèixer la ubicació exacta del servei, és a dir, s'ha de tenir un directori on hi hagi a punt les referències als serveis disponibles. Això s'aconsegueix gràcies a

Eines CASE

Hi ha eines CASE que simplifiquen mitjançant connectors (*plugins*) el procés de generació de clients a serveis web indicant únicament l'URL on se situa la descripció del servei (WSDL).

directoris UDDI. Una vegada el consumidor de servei disposa de l'especificació del servei (WSDL) ja pot fer el procés d'invocació mitjançant l'ús del protocol SOAP.

3.1.2. Composició de serveis

Les arquitectures orientades a serveis (SOA) estan basades en petites entitats (serveis) que poden ser coordinades per a desenvolupar entitats més grans o serveis complexos utilitzant protocols públics i serveis existents de manera similar a les peces d'un puzle.

La composició de serveis pot ser vista, tal com mostra la figura 3, com un model basat en capes en dos eixos, un de vertical i un d'horitzontal.

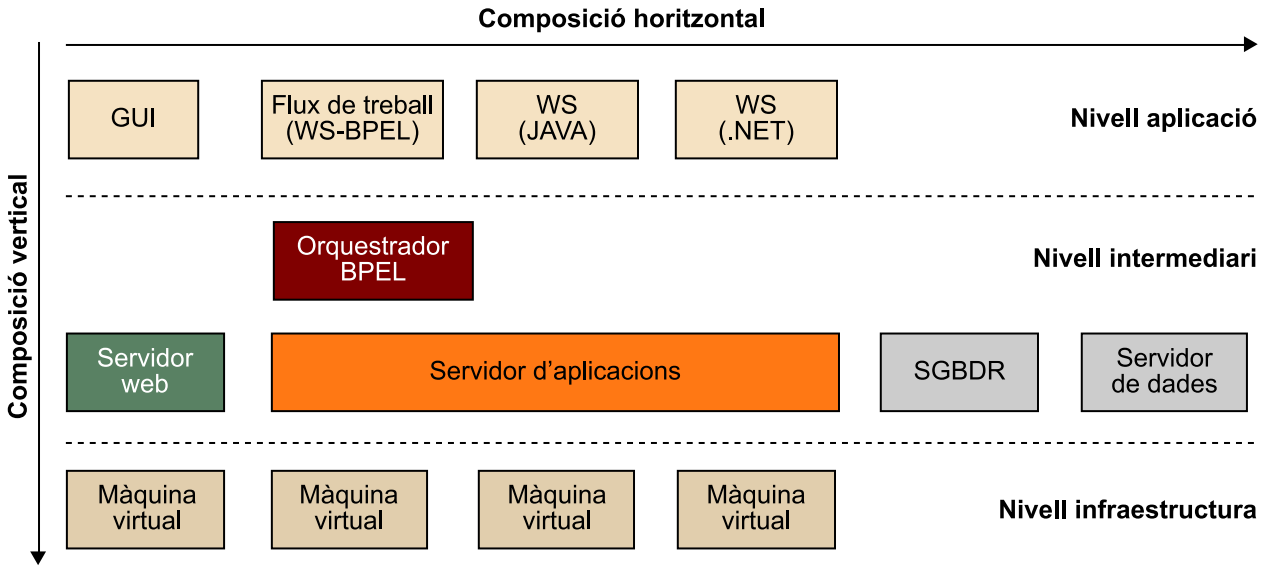

Figura 3. Composició de serveis

Composició de serveis vertical

L'eix de composició vertical disposa d'un nivell d'infraestructura, la capa intermediària o *middleware* (o *enterprise service bus*) i nivell d'aplicació.

El **nivell d'infraestructura** el formen tots els components de xarxa, computació, emmagatzematge de dades que es despleguen sobre els sistemes de xarxa, nodes i els dispositius dels usuaris. Aquests elements permetran la configuració i el desplegament de recursos sobre xarxes i serveis reals o virtuals.

Virtualització

La **virtualització** és un terme que es refereix a l'abstracció dels recursos, que pot ser aplicat sobre recursos (per exemple els nodes o equips), aplicacions i la xarxa, i tracta d'oferir una utilització millor dels recursos.

El **nivell intermediari** (*middleware*) és un nivell que pot estar basat en bus i resol problemes de connectivitat i interoperabilitat entre els possibles components que el formen: servidors d'aplicacions, sistemes de gestió de dades, motors d'orquestració, etc. Permetrà que qualsevol aplicació del nivell d'aplicació pugui ser utilitzada de manera transparent sobre la capa de recursos, i això facilita el desenvolupament i elimina detalls de baix nivell.

El **nivell d'aplicació** configura un núvol de components com interfícies d'usuari, fluxos de treball complexos (*workflow*), basats en composició i coordinació de serveis, serveis web i models de dades distribuïdes. Tots aquests components poden estar desenvolupats sota diferents perfils tecnològics i usen la capa intermediària en una composició vertical.

Composició de serveis horitzontal

En un nivell de composició horitzontal es poden construir aplicacions a partir d'un núvol de serveis que cooperen entre si, sobre la base de les interfícies i la possibilitat de reutilitzar els serveis de terceres parts (*third parties*). La figura 4 mostra un esquema de composició de serveis centralitzat basat en l'orquestració dirigida per un director d'orquestra. En la figura 5 es presenta un altre esquema de coordinació basada en la coreografia.

M2M

M2M fa referència al terme *Machine-to-Machine* i fa referència a l'intercanvi d'informació entre màquines (i sensors).

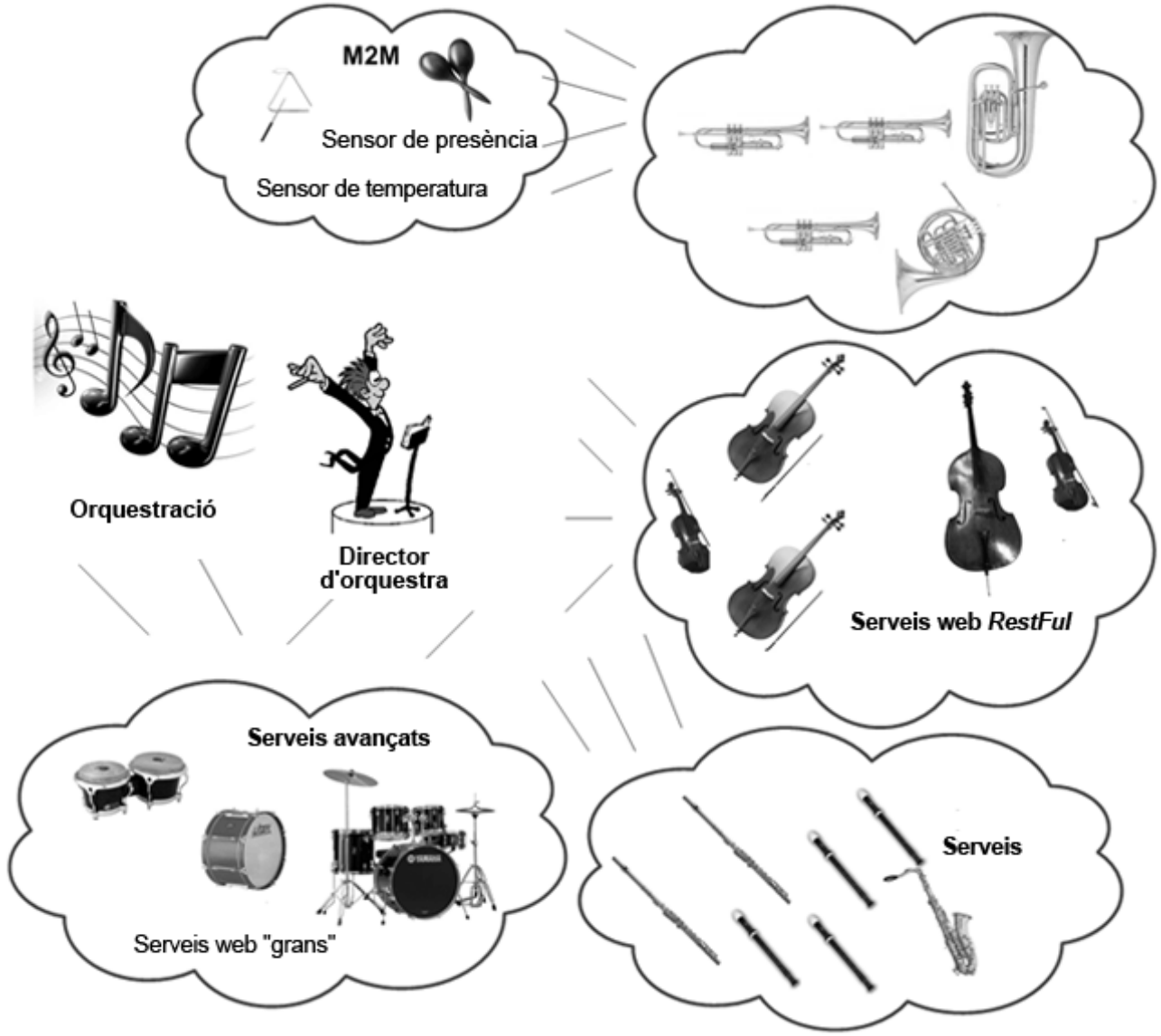

Figura 4. Orquestració

Per a fer possible la composició de serveis es disposa de protocols d'orquestració (WS-BPEL) i coreografia (*WS-Choreography*), que es descriuen a continuació.

WS-BPEL i *WS-Choreography*

WS-BPEL i *WS-Choreography* són protocols de la família WS-*.

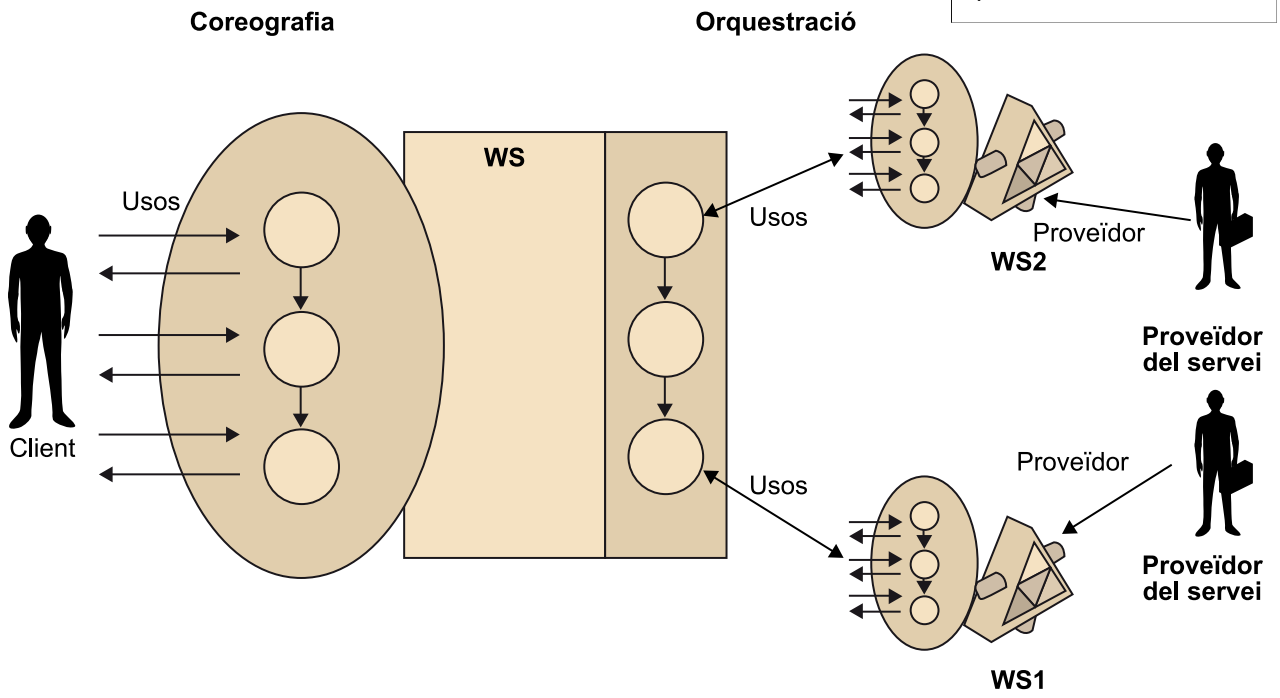

Figura 5. Coreografia/orquestració

La diferència entre l'orquestració i la coreografia es pot explicar utilitzant analogies. L'**orquestració** es refereix a un control central del comportament per a un sistema distribuït, l'orquestra amb els músics; mentre que la **coreografia** es refereix a un sistema distribuït (els ballarins) que opera d'acord amb unes regles, però sense un control centralitzat. D'aquesta manera, és important destacar que l'orquestració es controla d'una manera centralitzada, mentre que la coreografia es controla de manera distribuïda.

L'**orquestració** difereix de la **coreografia** en el fet que descriu un flux de processos entre serveis, controlats per un director d'orquestra (*single party*). De naturalesa més col·laborativa, la coreografia segueix la seqüència de missatges involucrats en múltiples participants, i totes dues tècniques poden ser usades en diferents casuístiques.

Hi ha diferents llenguatges de composició, encara que no hi ha una solució universal. En els subapartats següents es descriuran i compararan diferents llenguatges de composició.

Orquestració: WS-BPEL

BPEL ha estat dissenyat específicament com un llenguatge per a la definició de processos de negoci i està basat en un document XML i serveis web, incloent-hi SOAP, WSDL, UDDI, *WS-Reliable Messaging*, *WS-Addressing*, *WS-Coordi-*

nation i *WS-Transaction*. Un procés BPEL especifica els serveis web involucrats en la composició, l'ordre d'execució i les estructures de control de flux algorítmiques, entre les diferents activitats del servei compost.

Amb BPEL es pot especificar una seqüència d'operacions, o algun procés en paral·lel, i a més es pot expressar comportament condicional, com per exemple, quan una invocació a un servei web depèn del valor d'una invocació prèvia; definir iteracions, declarar variables, copiar i assignar valors, definir gestors associats a esdeveniments (per exemple, errors). A continuació es presenta un exemple d'una corredoria d'assegurances.

Un usuari accedeix a un portal d'una corredoria d'assegurances que treballa amb múltiples companyies i proporciona a l'usuari la millor oferta que s'ajusti a les condicions particulars de l'usuari.

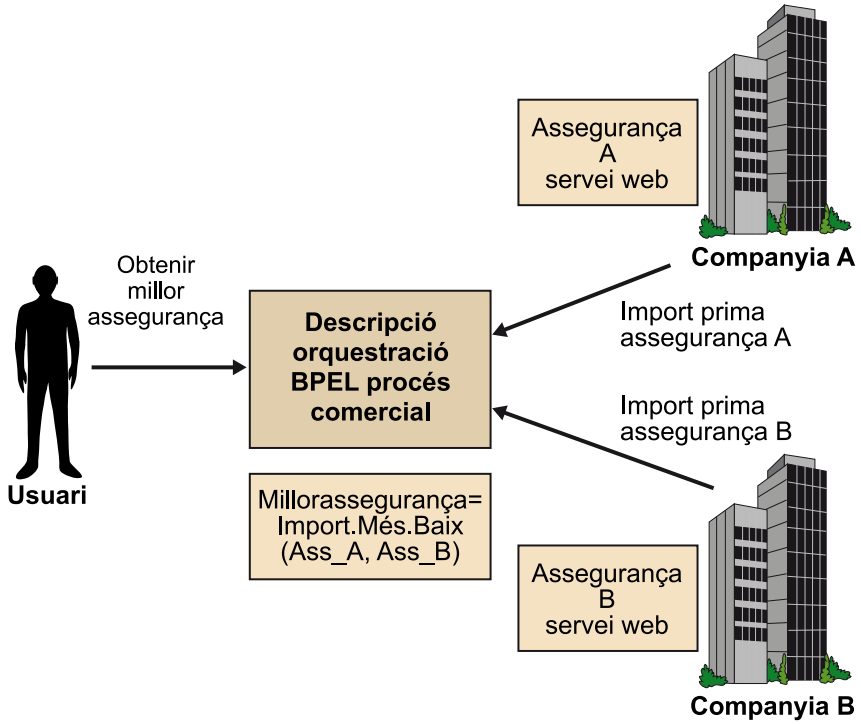

Figura 6. Exemple de corredoria d'assegurances

La combinació d'aquestes estructures pot definir nous serveis complexos compostos i, d'aquesta manera, BPEL és comparable a un llenguatge de propòsit general com Java, encara que no tan potent. D'altra banda, WS-BPEL és més simple i més adequat per a la definició de processos, encara que no pot ser reemplaçat, sinó que ha de ser utilitzat com a complement d'altres llenguatges moderns. A continuació, es presenta un exemple (*HelloWorld*) senzill de definició de procés de negoci amb BPEL. Habitualment es **modelitza** mitjançant una interfície gràfica la composició del servei i l'entorn genera el codi WS-BPEL.

```
<?xml version="1.0" encoding="UTF-8"?>
<bpws:process exitOnStandardFault="yes" name="HelloWorld"
    suppressJoinFailure="yes"
    targetNamespace="http://localhost:8080/ode/processes/HelloWorld"
    xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:tns="http://localhost:8080/ode/HelloWorld">
    <bpws:import importType="http://schemas.xmlsoap.org/wsdl/"
        location="HelloWorld.wsdl" namespace="http://localhost:8080/ode/HelloWorld"/>
    <bpws:partnerLinks>
        <bpws:partnerLink myRole="HelloWorldProvider" name="client"
            partnerLinkType="tns:HelloWorld"/>
    </bpws:partnerLinks>
    <bpws:variables>
        <bpws:variable messageType="tns:HelloWorldRequestMessage" name="input"/>
        <bpws:variable messageType="tns:HelloWorldResponseMessage" name="output"/>
    </bpws:variables>
    <bpws:sequence name="main">
        <bpws:receive createInstance="yes" name="receiveInput"
            partnerLink="client" portType="tns:HelloWorld"/>
        <bpws:assign name="Assign" validate="no">
          <bpws:copy>
            <bpws:from>
                <bpws:literal>
                  <tns:HelloWorldResponse xmlns:tns="http://www.ibm.com/wd2/ode/HelloWorld">
                      <tns:result/>
                  </tns:HelloWorldResponse>
                </bpws:literal>
            </bpws:from>
            <bpws:to part="payload" variable="output">
                <bpws:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                    <![CDATA[/tns:result]]>
                </bpws:query>
            </bpws:to>
          </bpws:copy>
          <bpws:copy>
            <bpws:from><![CDATA[concat("Hello ", $input.payload/tns:input)]]></bpws:from>
            <bpws:to/>
          </bpws:copy>
        </bpws:assign>
        <bpws:reply name="replyOutput" operation="process"
            partnerLink="client" portType="tns:HelloWorld" variable="output"/>
    </bpws:sequence>
</bpws:process>
```

Figura 7. Exemple WS-BPEL d'un "Hola, món" i d'orquestració en BPEL.

Coreografia: *WS-Choreography*

WSCI és un llenguatge de descripció de llenguatges que descriu un flux de missatges intercanviats per un servei web que interactua amb altres serveis web. WSCI descriu el comportament del comportament del servei web, que es descriu en termes de dependències temporals i lògiques, missatges intercanviats, regles, correlació, gestió d'excepcions i transaccions. WSCI, a més, descriu l'intercanvi col·lectiu de missatges, i proporciona una vista de les interaccions global.

WSCI

WSCI és un acrònim de *WS-choreography interface*; és una especificació del W3C basada en **la modelització** de processos amb XML. Els serveis web actuen com a iguals en un esquema de col·laboració.

WSCI no relaciona la definició i la implementació interna dels processos que finalment condueixen a l'intercanvi de missatges; l'objectiu de WSCI és descriure el comportament del servei web per mitjà d'una interfície basada en el flux de missatges. Aquesta descripció permet als desenvolupadors, als arquitectes i a les eines una vista general de l'intercanvi dinàmic de missatges i entendre les interaccions amb el servei web.

3.2. Serveis web basats en REST

Els **serveis web** basats en REST (*RESTful web services*) són serveis que estan dissenyats per a funcionar de manera més eficient sobre el Web. REST[12] és un estil arquitectònic que especifica restriccions, com ara una interfície uniforme

(12)De l'anglès *representational state transfer*.

(basada en identificadors per als recursos), que si s'aplica a un servei web infereix propietats com el rendiment o l'escalabilitat que fan possibles serveis més eficients, simples i que tinguin un alt rendiment. REST està basat en una arquitectura web denominada *arquitectura orientada a recursos* (ROA), que es basa en accions sobre **recursos**, per a permetre la interacció entre client i servidor, mitjançant l'ús dels mètodes *get*, *post*, *put* i *delete* inherents d'HTTP (i relacionats, com operacions **CRUD**) i implementat en navegadors, que no requereixen missatges SOAP/XML o descripcions WDSL. Això fa que aquesta estratègia sigui molt més lleugera.

REST

REST és un terme que fa referència a un estil arquitectònic orientat a recursos (ROA). Va ser introduït l'any 2000 per Roy Fielding en la seva tesi doctoral.

Vegeu també

En les assignatures de bases de dades s'expliquen amb detall les operacions CRUD (*create*, *read*, *update*, *delete*).

En REST, les dades i la funcionalitat són considerats com a **recursos**, i a aquests recursos s'accedeix mitjançant identificadors (URI), mitjançant un conjunt d'operacions senzilles i ben definides. L'estil arquitectònic REST està basat en una arquitectura client-servidor, i està dissenyat perquè clients i servidors intercanviïn informació mitjançant una interfície estandarditzada, simple i mitjançant l'ús d'un protocol de comunicació sense estat, en general, d'HTTP.

REST o els serveis web *RESTful*[13] s'estan utilitzant de manera regular, sobretot i, especialment, en dispositius d'escassos recursos, i en un context web 2.0 un desenvolupament basat en **serveis web** (anomenat *Web API*) també es pot fer amb un estil de comunicacions REST. *Web API* permet la combinació de múltiples serveis web en noves aplicacions anomenades remescles.

[13]Un **servei web *RestFul*** és un servei simple basat en HTTP i un estil REST.

Entity bean

En Java EE, un *entity bean* o un POJO té un comportament similar a **un recurs**, i ofereix un mecanisme d'accés a la informació.

Un *session bean* es pot veure com un controlador que proporciona unes operacions (**servei**).

A continuació, es descriurà el concepte de *recurs*, els estils i els formats de les dades (POX, JSON) i, finalment, un mecanisme d'especificació de serveis web basats en REST (WADL).

3.2.1. El concepte *recurs*

Un **recurs** pot ser definit com un artefacte de programari associat a informació específica (dades). Els recursos es defineixen mitjançant **noms** que descriuen la informació que proporciona el recurs i les operacions CRUD predefinides en una interfície semàntica.

Vegeu també

En el subapartat "Web 2.0" es pot veure amb més detall el concepte de *remescla*.

La semàntica en REST està basada en un conjunt d'operacions HTTP:

DBMS

Si WS-* es pot veure com el mecanisme d'RPC d'Internet, REST es pot veure com el DBMS (*database management system*) d'Internet.

- Crear recurs: crea un nou recurs i l'identificador únic (*PUT*).
- Obtenir la representació del recurs (*GET*).
- Esborrar un recurs (*DELETE*).
- Modificar un recurs (*POST*).
- Obtenir metainformació sobre un recurs (*HEAD*).

3.2.2. Formats per a representar els recursos

REST recomana l'ús d'estàndards establerts, com per exemple, URL per a l'adreçament, els mètodes HTTP per a la comunicació; i tipus *MIME* amb diferents tipus de dades: XML, JSON, XHTML, HTML, PNG i altres estils de formats d'informació. A continuació es descriuran els mètodes més usats per al procés de serialització (*marshalling*) de la informació: **POX** i **JSON**.

POX

POX[14] és un terme que s'ha inspirat en altres termes com *POJO*[15] i, conceptualment, segueix la mateixa filosofia en referència a la relació entre les dades i la manera abstracta d'organitzar-les. En el cas d'un POJO, el contenidor sobre el qual s'organitza la informació és una classe Java, i en el cas d'un POX, la informació s'estructura en una estructura jeràrquica, utilitzant un document XML.

(14)De l'anglès *plain old XML*.

(15)De l'anglès *plain old Java object*.

JSON

A més d'utilitzar XML (POX) per a serialitzar la informació que es transmet tant en la petició com en la resposta, hi ha altres estratègies com l'ús de JSON[16]. JSON s'empra quan la mida del flux de dades és de vital importància. La simplicitat de JSON ha donat lloc a la generalització del seu ús, per la facilitat de desenvolupament i rendiment. L'únic inconvenient és que penalitza la validació de la informació que envia el client.

(16)De l'anglès *Javascript object notation*.

JSON és un format lleuger d'intercanvi de dades. És fàcil de llegir i escriure per als éssers humans i, per a les màquines, d'analitzar i generar.

Es basa en un subconjunt del llenguatge de programació JavaScript. Està basat en dues estructures:

1) Una col·lecció de parells nom/valor. En diversos idiomes, això es fa com un diccionari.

2) Una llista ordenada de valors.

Aquestes són estructures de dades universals i, pràcticament, tots els llenguatges de programació moderns n'implementen d'una manera o una altra. En l'exemple que es presenta a continuació, es mostra una entitat *Persona* amb els atributs *nom*, *data de naixement*, *àlies* i una llista de números de telèfon: *mòbil*, *fix* i el de la feina.

Llenguatges de programació

Els llenguatges de programació ActionScript, C, C++, C#, ColdFusion, Common Lisp, Delphi, E, Eiffel, Java, JavaScript, Perl, PHP i Python implementen una API per a analitzar i generar JSON.

```
{
  "class": "Person",
  "name": "William Shakespeare",
  "birthday": -12802392000000,
  "nickname": "Bill"
  "phoneNumbers": [
```

```
      {
       "class": "Phone",
       "name": "cell";
       "number": "555-123-4567",
      },
      {
       "class": "Phone",
       "name": "home",
       "number": "555-987-6543"
      },
      {
       "class": "Phone",
       "name": "work",
       "number": "555-678-3542"
      }
    }
  }
```

WADL

Una manera d'especificar una interfície d'un servei basat en REST és utilitzar un llenguatge de descripció anomenat *WADL*[17]. **WADL** és un document XML que proporciona una descripció llegible d'aplicacions web basades en HTTP. El propòsit de WADL és permetre que els serveis a Internet es descriguin d'una manera processable per una màquina, perquè sigui més fàcil per a crear aplicacions web 2.0, i crear una manera dinàmica de creació i configuració de serveis. Sense una especificació del servei és necessari fer enginyeria inversa i construir l'aplicació d'una manera primitiva. WADL es pot considerar com l'equivalent del WSDL[18].

(17)De l'anglès *web application description language*.

(18)De l'anglès *web services description language*.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://research.sun.com/wadl/2006/10">
    <doc xmlns:jersey="http://jersey.dev.java.net/" jersey:generatedBy="Jersey: 0.10-ea-SNAPSHOT 08/27/2008 08:24 PM"/>
    <resources base="http://localhost:9998/">
        <resource path="/helloworld">
            <method name="GET" id="getClichedMessage">
                <response>
                    <representation mediaType="text/plain"/>
                </response>
            </method>
        </resource>
    </resources>
</application>
```

Figura 7. Exemple d'especificació basada en WADL

3.3. Diferències entre serveis web basats en REST i basats en SOAP (WS-*)

Una de les diferències fonamentals entre les dues estratègies d'implementació de serveis web estudiades (*RestFul WS* i basats en SOAP), és la filosofia intrínseca de totes dues opcions. Buscant una analogia, podem acordar que les aplicacions basades en REST s'organitzen en recursos (**noms**), mentre que les aplicacions basades en serveis web (WS-*) s'organitzen en accions (**verbs**).

La taula següent mostra una comparativa entre totes dues estratègies per a la construcció de serveis web i presenta els avantatges i els inconvenients sobre cadascuna de les estratègies.

Taula comparativa de serveis web basats en REST i WS-*

Serveis web	Característiques	Avantatges	Problemes
Model REST	• Arquitectura client-servidor. • Orientat a recursos. • Identificador únic de recursos basat en URI. • Protocol http: *get, post, put, delete.* • No té estat. • Missatges autodescriptius.	• Mecanisme lleuger. • Augment d'escalabilitat. • Augment de rendiment.	• Integració amb aplicacions B2B. • Falta d'estàndards.
WS-*	• Independent del llenguatge i plataforma. • No lligat a cap protocol. • Pot utilitzar un registre UDDI. • Basat en XML. • Especificació de serveis basada en WSDL.	• Especificacions W3C. • Permet utilitzar altres protocols de transport.	• Capacitat de computació per al processament dels XML. • Rendiment. • Amplada de banda alta.

La figura 9 mostra un exemple d'una petició i resposta en una invocació a un servei web que fa la suma de dos nombres. En aquest cas, la codificació de les dades en un esquema REST es fa mitjançant un document XML (POX).

POX

POX (*plain old XML*) és un concepte anàleg de POJO que pretén relacionar i serialitzar (*marshalling*) una entitat d'informació a XML.

El concepte POJO (*plain old Java object*) es pot veure amb més detall en el mòdul "Java EE" d'aquest material didàctic.

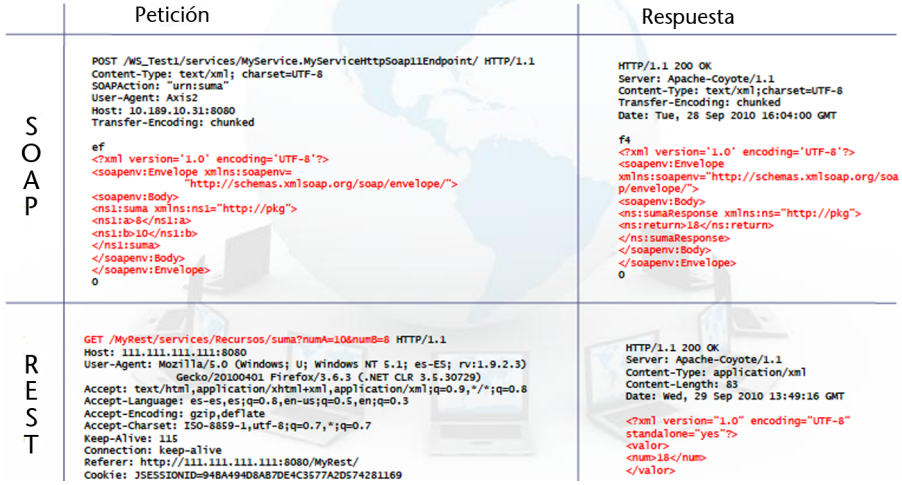

Figura 9. Exemple de petició SOAP - REST/POX

S'aprecia una diferència de pes (mida) en l'opció basada en serveis web pesants (SOAP), en comparació de l'opció basada en serveis web basats en REST/POX. En aquest últim cas (REST), tant les peticions com les respostes són relativament petites respecte a la mida dels missatges SOAP. El motiu fonamental és que SOAP requereix una codificació basada en XML per a les peticions i les respostes que incrementa la mida del missatge. En el cas de REST es pot minimitzar el pes dels missatges amb alternatives com JSON.

4. Disseny d'aplicacions orientades a serveis amb UML

En mòduls anteriors hem vist com s'utilitza RM-ODP per a definir l'arquitectura en termes de components arquitectònics i connectors entre aquests i com, posteriorment, es refina aquesta especificació en termes d'un conjunt de components de programari que els implementen tenint en compte que prèviament han estat especificats de manera independent de la tecnologia.

Vegeu també

Vegeu els mòduls "Disseny d'aplicacions distribuïdes", "Arquitectura del programari" i "Desenvolupament de programari basat en components" d'aquest material didàctic.

Una vegada s'ha fet l'elecció tecnològica (en aquest cas serveis web basats en SOAP i *RESTFul*) és necessari fer una fase prèvia de disseny per a evitar errors i disminuir la distància existent entre l'especificació i la implementació concreta, utilitzant diagrames UML per a representar aquestes decisions.

Aquesta fase partirà d'una representació dels components que formen l'aplicació independentment de la tecnologia i finalitzarà amb una representació dels mateixos components amb el perfil SOA.

Dins de l'ampli conjunt de diagrames que especifica UML 2.0, ens centrarem en com es representa el sistema concret amb els diagrames de components i desplegament. Per a mostrar-ho ens basarem en un exemple senzill que il·lustri els passos que s'han de fer i en algunes de les consideracions que cal tenir en compte.

4.1. Perfil UML per a aplicacions orientades a serveis

En el cas que ens ocupa, s'han triat els **serveis web** com a tecnologia d'implementació, que presenta algunes particularitats que fan necessària una extensió d'UML per a poder-ne representar tots els elements.

Algunes variacions o particularitats dels serveis web respecte als conceptes generals que defineix UML són les següents:

- Depenent del tipus de component: client SOAP, servei web, WDSL.
- Descriptors de desplegament.
- Formats estàndard d'empaquetament.

Mancant una definició estàndard per al perfil dels **serveis web**, per a redactar aquestes anotacions ens basarem en el perfil *web services*, que apareix com a exemple en el document de *Superestructura d'UML 2.0* i afegirem els elements que necessitarem per a modelitzar els aspectes més rellevants de les aplicacions basades en serveis web.

Web recomanat

El document de *Superestructura d'UML 2.x* està disponible a http://www.omg.org/spec/uml.

La taula següent mostra alguns dels estereotips que defineixen el document de *Superestructura d'UML 2.0* i altres que hem definit en aquestes anotacions per il·lustrar els serveis web.

Estereotip	Tipus d'element	Descripció
<<WebService>>	Component	El component que representa el servei web.
<<wdsdl>>	Interfície	Contracte o interfície del servei web.
<<asynchronousClient>>	Component	Component que actua com a client asíncron.
<<wsStaticClient >>	Component	Component que actua com a client i utilitza els *stubs* generats.
<<wsDynamicClient >>	Component	Component que actua com a client i que fa invocacions dinàmiques sobre el servei web.
<<WebServiceProvider>>	Component	Proveïdor del servei web.
<<WebMethod>>	Mètode	Mètode d'un servei web.
<<RequestWrapper>>	Component	Embolcall d'una petició.
<<ResponseWrapper>>	Component	Embolcall d'una resposta.
<<AsyncHandler>	Component	Gestor asíncron.
<<handlerChain>>	Component	Cadena de gestors.
<<uddi>>	Component	Component que actua com un servidor de directori.
<<soapmessage>>	Component	Missatge SOAP.
<aar>	Artefacte	Fitxer en format AAR (arxiu axis).

En el cas de serveis web basats en REST podem proposar una altra taula amb els elements següents:

Estereotip	Tipus d'element	Descripció
<<WebResource>>	Component	*RestFul web service.*
<<HttpMethod>>	Mètode	Mètode del servei web.
<<wadl>>	Interfície	Contracte o especificació del servei.
<<Produces>>	Component	Component que actua com a consumidor.
<<Consumes>>	Component	Component que actua com a consumidor.
<<Context>>	Component	Context del servei.
<<text_xml>>	MediaType	Tipus de dades.
<<application_json>>	MediaType	Tipus de dades.
<<application_xml>>	MediaType	Tipus de dades.
<<application_form_urlencoded>>	MediaType	Tipus de dades.

Estereotip	Tipus d'element	Descripció
<<ApplicationPath>>	Component	Arrel dels recursos REST.

L'objectiu d'aquest subapartat no és definir un perfil complet SOA, sinó mostrar la importància que té disposar d'un perfil per a la tecnologia concreta i com ens ajudarà per a especificar la fase de disseny.

4.2. Exemple

Per a veure com s'apliquen algunes de les decisions de disseny que hem comentat en l'apartat anterior i com es van refinant els diagrames de components fins a deduir-ne gairebé la implementació, anem a treballar sobre el component que permet als clients del banc en línia fer operacions amb els seus comptes.

Tal com es va presentar en el mòdul "Java EE" d'aquest material didàctic, aquest component es podria representar com ho mostra la figura 10, en el nivell més alt d'abstracció.

Vegeu també

Trobareu el disseny de l'aplicació en el mòdul "Java EE" d'aquesta assignatura.

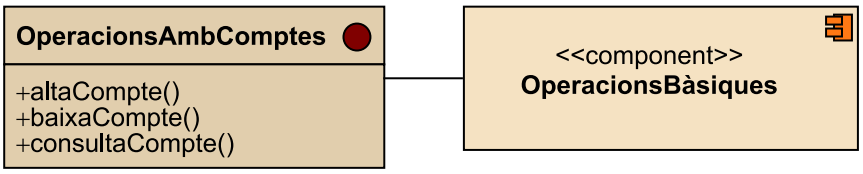

Figura 8. Component independent de la tecnologia

Vegeu també

El mòdul "Java EE" d'aquest material didàctic mostra tot el procés de disseny del component ***OperacionsBàsiques***.

Després de fer (en el mòdul "Java EE" d'aquest material didàctic) tot el procés de refinament es van analitzar un per un tots els components i es va prendre una decisió de desplegament basada a desplegar tota la part web en una màquina que actua com a contenidor web i la part de negoci i persistència en una altra màquina amb un contenidor EJB. El diagrama de desplegament va quedar com mostra la figura 11.

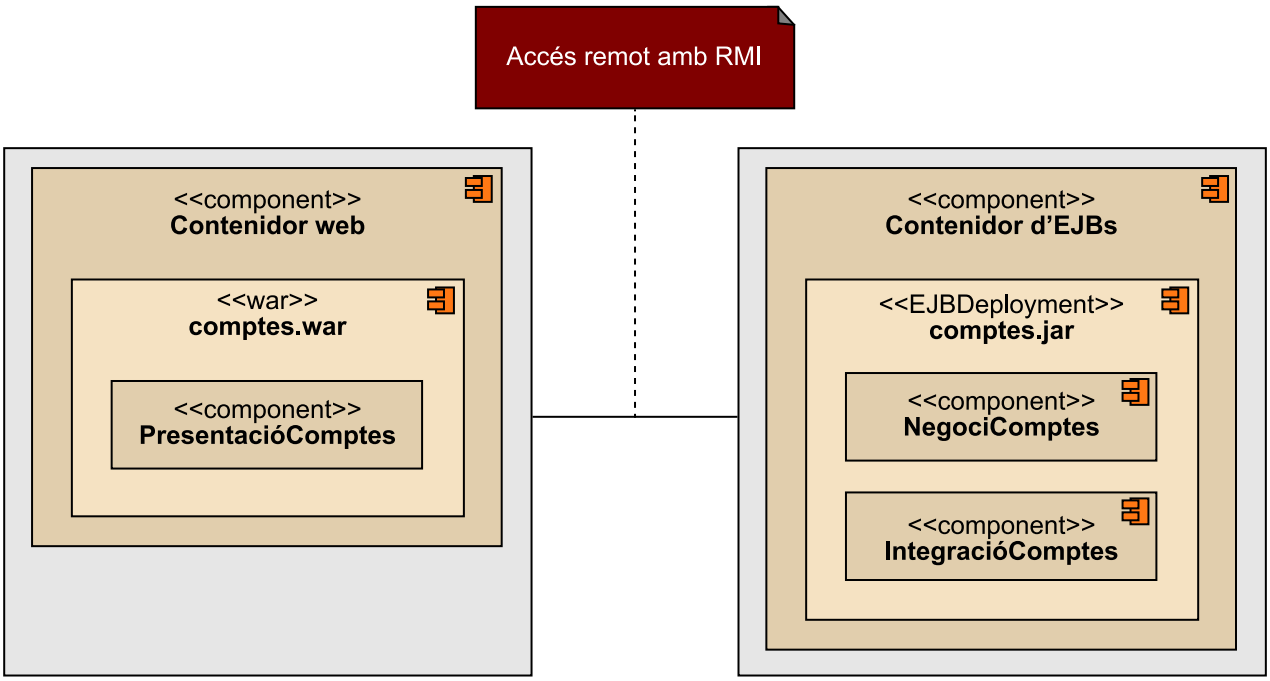

Figura 9. Diagrama de desplegament

Ara suposarem que el banc en línia és una part dins d'un conglomerat d'un gran grup bancari resultat d'una fusió de bancs i caixes; i tota la gestió de comptes bancaris es fa mitjançant la interacció entre els bancs mateixos que formen el grup fusionat.

El sistema ha de permetre que un usuari hi pugui accedir per mitjà de qualsevol oficina del grup fusionat, a pesar que la informació dels seus comptes corrents estigui en els sistemes d'un altre banc. L'estratègia d'integració de sistemes que es proposa en el grup fusionat consisteix en una solució basada en un EAI[19] i una arquitectura orientada a serveis. D'aquesta manera, quan un usuari interactua amb una oficina d'un banc, ja sigui amb el mòbil, el Web o presencialment, i no és el que posseeix el compte corrent, aquest ha d'accedir, mitjançant les API[20] que li proporcionen altres bancs, a tota la informació (productes i serveis) que s'ofereixen a l'usuari.

La figura 12 presenta un esquema que mostra tot l'escenari que s'ha descrit amb anterioritat.

(19) De l'anglès *enterprise application integration*.

(20) *API* és la sigla d'*application program interface*.

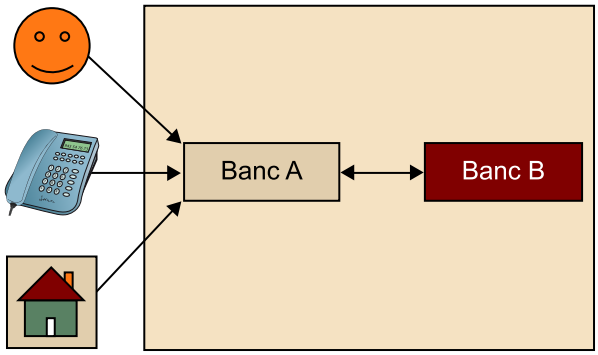

Figura 10. Grup bancari

Amb aquest escenari, el funcionament consisteix en usuaris que accedeixen per mitjà d'una aplicació *ad hoc* en el mòbil, un navegador o accedint presencialment a una oficina bancària als serveis contractats. Cada banc, quan ha d'oferir serveis a un usuari amb comptes corrents no gestionats directament per ell, interactua amb el banc corresponent en un model de negoci B2B.

Amb aquest escenari la situació quedaria com es presenta en la figura 11.

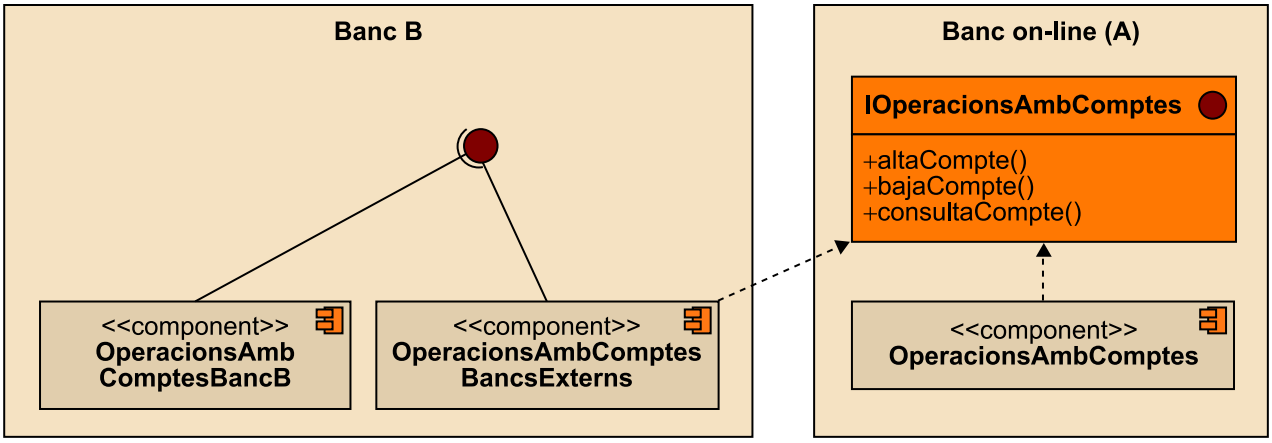

Figura 11. Exemple del banc en línia

En la figura 11 es poden apreciar 2 nodes. Un primer node, el banc en línia (banc A), que és el component original Java EE descrit en el mòdul "Java EE" d'aquests materials didàctics, en el qual **no s'ha implementat cap funcionalitat addicional** respecte al que s'ha descrit anteriorment. D'altra banda, tenim un segon banc (banc B) que ofereix operacions amb comptes que són gestionades pel banc mateix, i en aquest cas tindríem un comportament similar a les funcionalitats ofertes pel nostre banc en línia però que poden estar desenvolupades en plataformes i perfils tecnològics diferents. El banc en línia exporta i ofereix una API sobre les operacions amb comptes especificades i dissenyades anteriorment cap a altres bancs perquè puguin interactuar utilitzant, en aquest cas, serveis web WS-* basats en un esquema B2B.

Vegeu també

Aquest mòdul no té en consideració aspectes de seguretat que són descrits en l'assignatura *Seguretat en xarxes de computadors*.

D'aquesta manera, quan un usuari accedeix de manera presencial o mitjançant un dispositiu mòbil o navegador, a una oficina del grup per a fer una **operació bàsica** del seu compte corrent que aquest banc no gestiona directament, s'accedirà al banc originari mitjançant un **servei** que ofereix aquest banc i ofe-

rirà de manera transparent les operacions bàsiques del compte. La figura 12 mostra la interacció de components i el pas de missatges quan un usuari intenta accedir al saldo del seu compte corrent.

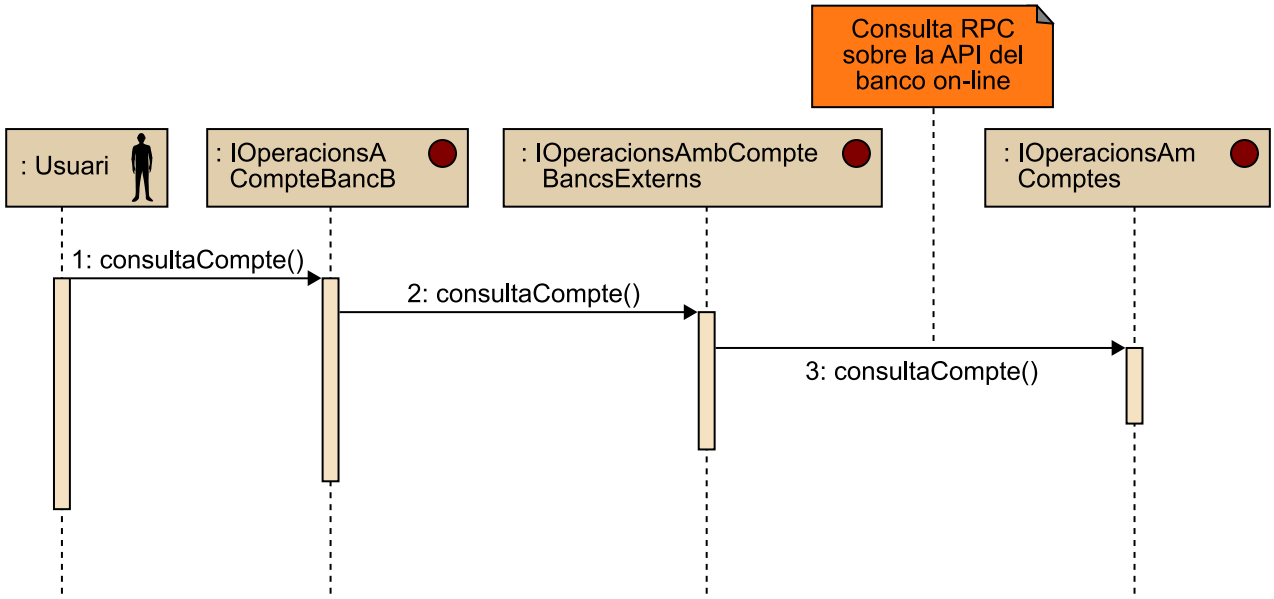

Figura 12. Diagrama de seqüència d'una consulta de compte corrent

Finalment, només queda determinar què ha estat necessari afegir en el mòdul Java EE original per a oferir en un servei web les operacions organitzades en forma d'API, que permetran que terceres parts, en aquest cas altres oficines, en puguin fer ús. El refinament de la capa de negoci queda tal com es mostra en la figura 13.

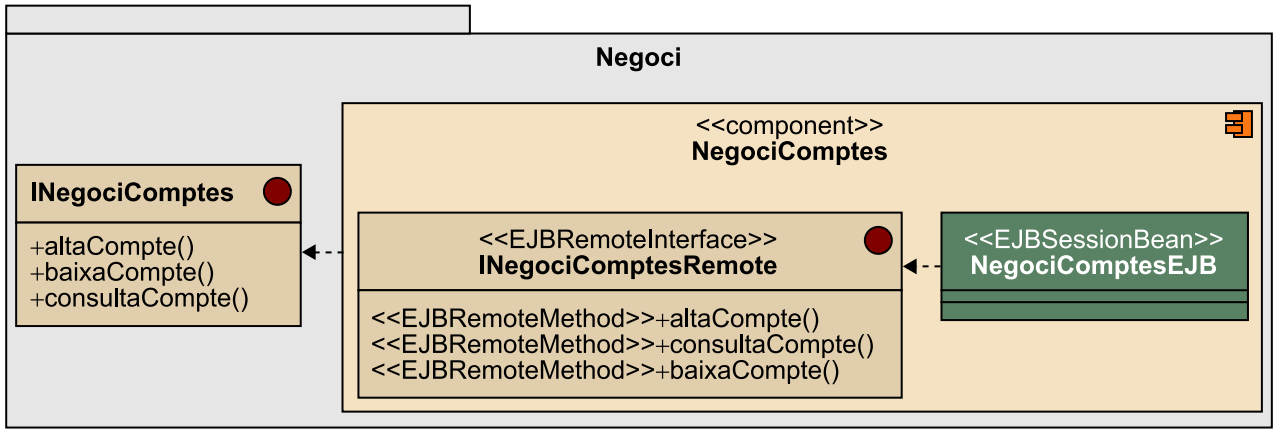

Figura 13. Refinament de la capa de negoci

Si apliquem el perfil tecnològic SOA per a serveis web basats en WS-* obtenim el diagrama de components que mostra la figura 14, en què es pot apreciar la interfície amb els mètodes del **servei web**, i la forma d'implementació del serveis exportant les operacions de l'EJB de *Session*.

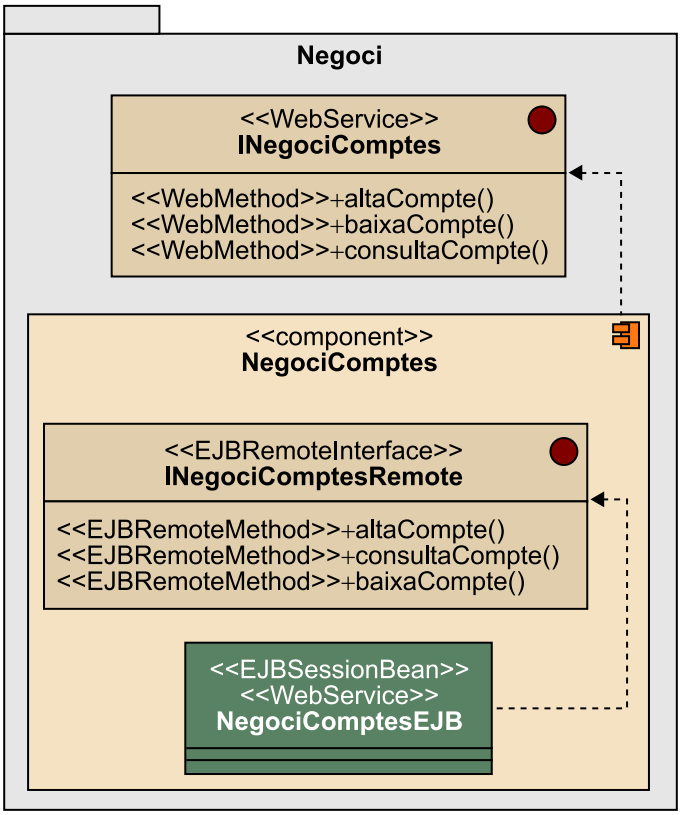

Figura 14. Aplicació del perfil SOA

Resum

Aquest mòdul ha donat una visió general de l'evolució dels serveis distribuïts, en paral·lel amb l'evolució d'Internet. A mesura que s'anaven ampliant les possibilitats del mitjà i els usos per part dels usuaris s'han produït canvis de paradigma i noves arquitectures. En aquest cas, aquest mòdul s'ha centrat en la definició de l'arquitectura **orientada a serveis** i les formes més comunes de la implementació de serveis distribuïts.

Al llarg del mòdul s'ha definit el concepte *SOA*, les característiques d'aquest model arquitectònic i s'han presentat esquemes de negoci B2C (negoci orientat a l'usuari) i B2B (orientat de negoci a negoci) que tenen sinergies entre aquests esquemes de negoci i l'**arquitectura orientada a serveis**. S'han explicat dues estratègies d'implementació: serveis web pesants (basats en HTTP/SOAP) i serveis web lleugers (*RESTFUL web services*); i finalment, es proposa com a forma de construcció de serveis complexos la **reutilització** i la **composició de serveis**, basades en l'orquestració o la coreografia.

L'últim apartat del mòdul ha servit per a presentar les bases del **disseny d'aplicacions SOA**, amb tot un seguit de recomanacions i aspectes que cal tenir en compte.

Activitats

1. Donats els tipus següents de programari per dissenyar, indiqueu en cada cas si és aconsellable l'ús d'una arquitectura orientada a serveis (SOA). En cas afirmatiu descriviu-ne breument l'escenari:

a) Una empresa que comercialitza mobles i treballa amb diferents fàbriques. Es vol oferir a l'usuari una plataforma única per a comercialitzar els productes.

b) Un sistema de vot electrònic, per a millorar la democràcia directa i facilitar la consulta i els referèndums als ciutadans que pugui funcionar a escala local, autonòmica, estatal i europea.

c) Un sistema de compartició d'arxius en el núvol, accessible per mitjà de dispositius mòbils i ordinadors personals.

d) Una xarxa social orientada a salut que vol oferir una API perquè terceres parts desenvolupin aplicacions.

e) Un sistema de vídeo sota demanda.

2. Implementeu la calculadora especificada en el mòdul "Java RMI", utilitzant serveis web basats en SOAP.

3. Implementeu la calculadora especificada en el mòdul "Java RMI" utilitzant serveis web basats en REST.

4. Utilitzant un monitor de SOAP o un analitzador de protocols, presenteu els missatges generats en la petició i resposta de l'exercici anterior (SOAP i REST).

5. Busqueu i analitzeu orquestradors (basats en WS-BPEL) comercials i no comercials que hi hagi en el mercat.

Exercicis d'autoavaluació

1. És cert que els serveis web solament poden funcionar per HTTP?

2. Expliqueu en què consisteix UDDI i quina és la seva funció.

3. Expliqueu la manera com els serveis en una arquitectura SOA estan **desacoblats**.

4. Descriviu la característica de l'**encapsulació de les dades** en un esquema SOA.

5. Quina és la tecnologia inherent a SOA?

6. Com s'implementen les operacions CRUD (*create*, *read*, *update*, *delete*) quan s'utilitza REST?

Solucionari

Exercicis d'autoavaluació

1. Fals, no hi ha tal limitació; els serveis web poden funcionar sobre multitud de protocols de comunicació. Per exemple, HTTP, TCP, UDP, etc.

2. UDDI és un registre de serveis SOA que té com a funció principal actuar com a índex dels serveis desplegats. Conté la informació bàsica (interfície) dels serveis: punt d'entrada (*endpoint*), operacions, format dels paràmetres, i informació addicional. Ha de tenir tot el necessari per a utilitzar un servei en un nou entorn.

3. L'única interrelació entre dos serveis o peces de programari és la interfície (definida per WDSL si utilitzem serveis web basats en SOAP) o contracte del servei.

4. L'encapsulació de les dades consisteix que les dades pertanyen al servei que les gestiona, és a dir, que **ningú** fora del servei no pot modificar o veure aquesta dada sense passar pel servei.

5. SOA és neutre respecte al punt de vista tecnològic. Ofereix únicament una arquitectura o paradigma de computació que permet que **qualsevol** tecnologia pugui executar components de manera remota (paradigma RPC) indicant un contracte del servei.

6. REST implementa les operacions CRUD, adaptant les operacions HTTP (*get*, *post*, *put* i *delete*).

Glossari

coreografia *f* Coordinació d'un grup de serveis sense un controlador central.

JSON (*JavaScript object notation*) *m* Format o estil de codificació de la informació inspirat en el llenguatge JavaScript.

orquestració *f* Coordinació per part d'un director d'orquestra de diversos serveis de manera centralitzada.

REST (*representational state transfer*) *m* Estil o arquitectura orientada a recursos que són accessibles per mitjà d'URL.

RESTFul web services *f* Tecnologia que implementa serveis lleugers basats en REST.

servei *m* Peça de programari que ofereix una interfície (contracte) basada en unes operacions amb uns paràmetres d'entrada/sortida especificats.

servei web (WS-*) *m* Tecnologia que implementa serveis basats en SOAP.

SOAP (*simple object access protocol*) *m* Protocol que especifica el mecanisme de serialització dels elements que intervenen en la petició o resposta.

UDDI (*universal description, discovery and integration*) *m* Directori o registre de serveis web basat en XML.

Bibliografia

Bibliografia bàsica

Snell, J.; Tidwell, D.; Kulchenko, P. (2001). *Programming Web Services with SOAP*. O'Reilly Media.

Woods, D.; Mattern, T. (2006). *Enterprise SOA*. O'Reilly Media.

Bibliografia complementària

Arkin, A.; Askary, S.; Fordin, S.; Jekeli, W.; Kawaguchi, K.; Orchard, D.; Pogliani, S.; Riemer, K.; Struble, S.; Takacsi-Nagy, P.; Trickovic, I.; Zimek, S. (2002, agost). *Web Service Choreography Interface (WSCI) 1.0. W3C Note 8*.

Besemer, D.; Butterworth, P.; Clément, L.; Green, J.; Ramachandra, H.; Schneider, J.; Vandervoort, H. (2008). "An Implementor's Guide to Service Oriented Architecture: Getting It Right". *Composite Software*.

Peltz, C. (2003, juny). "Web Services Orchestration and Choreography". *SOA World Magazine*.

Reuther, B.; Müller, P. (2008). "Future Internet Architecture - A Service Oriented Approach". *Information Technology Jahrgang* 50 Heft 6 (pàg. 383-389). Munic: Oldenbourg Verlag.

Richardson, L.; Ruby, S. (2007). *RESTful Web Services*. O'Reilly.

Van Than, D. (2003). *Web Service Orchestration: An open and standardized approach for creating advanced services*. Eurescom.

Referències bibliogràfiques

OASIS. *OASIS Commitees by Category: SOA*. [Data de consulta: juny de 2010] Disponible a http://www.oasis-open.org/committees/tc_cat.php?cat=soa.

Wollrath, A.; Riggs, R.; Waldo, J. (2009). *A Distributed Object Model for the Java System*. [Data de consulta: desembre de 2010] Disponible a http://pdos.csail.mit.edu/6.824/papers/waldo-rmi.pdf.