OBJEKTNI MODEL PORAZDELJENEGA PROCESIRANJA

Matjaž B. Jurič, Marjan Heričko, Ivan Rozman

Povzetek:

Tehnologija porazdeljenih objektov omogoča izgradnjo programske opreme z integracijo komponent - objektov. Objekti se osvobodijo oklepa programskega jezika, v katerem so bili oblikovani in svobodno zaživijo v omrežju. Z zmožnostjo ponovne uporabe, prenosljivosti in povezljivosti v distribuiranem, heterogenem omrežju rešujejo večino težav, s katerimi se danes srečujejo razvijalci programske opreme in uporabniki računalnikov.

V prispevku opisujemo model *Common Object Request Broker Architecture* (CORBA). Temelji na posredniku zahtev objektov, ki ima vlogo neke vrste vodila in je zadolžen za zagotavljanje komunikacije med aplikacijami, objektnimi storitvami in skupnimi sredstvi. Predstavlja »de-facto« standard med distribuiranimi objektnimi modeli. Tehnologija porazdeljenih objektov bo sčasoma izpodrinila ostale tehnologije in nepovratno vplivala na način uporabe in izgradnje programske opreme.

Abstract:

Distributed object technology enables building software with integration of components - objects. Objects become independent of programming language they were designed in and can live in the network. With reuse, portability and interoperability in distributed heterogen network they address the majority of problems developers and end-users are dealing with today.

In this article the Common Object Request Broker Architecture (CORBA) is described. It is based upon an object request broker, which acts as a bus and is responsible for communication between applications, object services and common facilities. CORBA is "de-facto" standard for distributed object models. Over time distributed object technology will supersede other technologies and will have irreversible impact on the way software is used and built.

■■■

1. UVOD

Danes se srečujemo z množico različnih računalnikov, povezanih v omrežje. Najmočnejši super-računalniki izvajajo zapletene znanstvene izračune ali delujejo kot strežniki glomaznih podatkovnih baz. V sredini najdemo strežnike in namizne računalnike, ki nudijo kompleksno množico storitev. O najmanjših sistemih mnogokrat niti ne razmišljamo kot o računalnikih. To so televizijski aparati, termostati in alarmi, telefoni in elektronske beležnice. Vsi ti sistemi uporabljajo različne operacijske sisteme in programske jezike. Želimo si, da bi bili računalniki povezani ne glede na tip, velikost ali operacijski sistem.

Poglaviten problem današnjih računalniških omrežij, sestavljenih iz različnih računalnikov, operacijskih sistemov in aplikacij, je nezmožnost povezave. Le-ta preprečuje boljšo so-uporabo podatkov in večjo produktivnost uporabnikov. Vemo pa, da se je računalniška paradigma spremenila. Podjetja niso več zadovoljna z uporabo računalnikov zgolj za štetje denarja, ki so ga zaslužila. Računalnike želijo uporabiti kot orodje za služenje denarja.

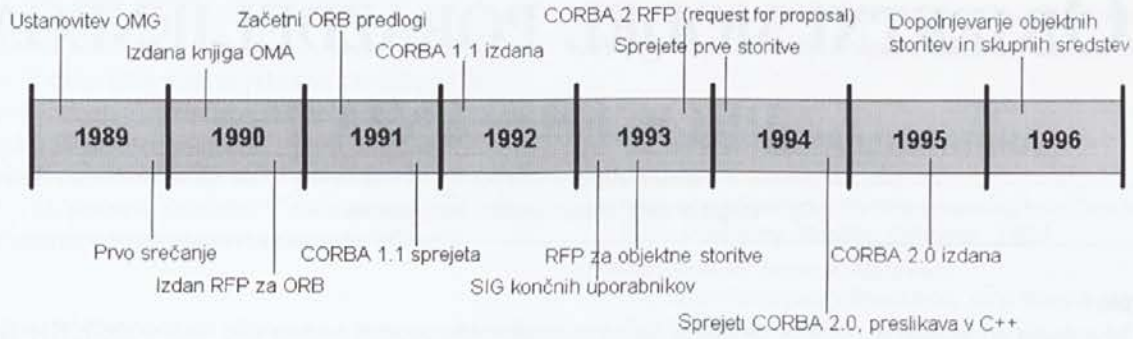
Poglejmo, kateri problemi izvirajo iz omenjenega okolja:

a) težko je zagotoviti integracijo računalniške strojne opreme,
b) še težje je zagotoviti integracijo računalniške programske opreme,
c) razvoj programske opreme traja predolgo in stane preveč.

Potrebujemo nov način pogleda na celoten problem, od definicije problema in analize, skozi potek načrtovanja in implementacije do uporabe, vzdrževanja in dopolnjevanja. *Objektna tehnologija* je tak nov način. V prispevku si seveda ne bomo mogli ogledati celotne tematike. Osredotočili se bomo na področje implementacije.

1.1. Porazdeljeni objekti

V preteklosti je računalništvo temeljilo na *podatkih*, namesto na *njihovi uporabi*, in na *računalniških* procesih, namesto na *poslovnih* procesih. Uporabniki so se morali prilagajati računalnikom. Strojna oprema je v tem času postala veliko zmogljivejša. To nam daje možnost, da računalnike prilagodimo svojim zahtevam in ne



Slika 1: **Kronologija dosežkov OMG-ja**

obratno. Uporabniki ne želijo več velikih, monolitnih aplikacij, ki znajo vse, ampak iščejo majhne komponente - objekte, ki jih lahko sami združujejo in tako oblikujejo orodja, prilagojena njihovim tipičnim poslovnim potrebam. Objekti pa delujejo skupaj le, če so načrtovani in oblikovani na osnovi standarda.

Objekti so enote kode in podatkov, ki komunicirajo s pošiljanjem in sprejemanjem sporočil. Če so zgrajeni korektno, so v sistemu visoko povezljivi in relativno enostavno je zamenjati lokalne objekte z oddaljenimi in tako razširiti komunikacijo skozi omrežje.

Narava objektov radikalno spreminja način razmišljanja o tem, *kje* so naši podatki. Podatki, ograjeni v objekte, lahko pridejo tja, kjer so najbolj potrebni. Trenutno razmišljamo, da so dokumenti enostavno shranjeni na določenem trdem disku. Distribuirani objektni sistemi zavračajo to tezo in nudijo moč in fleksibilnost shrambe, neodvisne od lokacije.

Naš ultimativni cilj je integracija objektov. Za dosego tega cilja pa moramo najprej rešiti problem povezljivosti objektov v distribuiranih, heterogenih okoljih. Rešitve obstajajo v obliki modelov porazdeljenega procesiranja. V prispevku se bomo osredotočili na najpomembnejši objektni model, model CORBA *(Common Object Request Broker Architecture)*.

Prehod na tehnologijo porazdeljenih objektov se ne bo zgodil čez noč; pot bo dolga in evolucijska. Pomembno pa je razumeti, kako tehnologije, ki so na voljo danes in tiste, ki bodo na voljo kmalu, pripravljajo uporabnike na življenje v svetu distribuiranih objektov.

1.2. Objektni modeli

Od leta 1989 obstaja konzorcij objektnih dobaviteljev. Imenuje se *Object Management Group*[1] (OMG). Ukvarja se s specifikacijo arhitekture odprtega programskega vodila, na osnovi katerega lahko povežemo objektne komponente različnih dobaviteljev prek omrežja. Organizacija OMG določa smernice in specifikacije upravljanja z objekti in določa okvir za razvoj aplikacij. Osnovni cilji so ponovna uporaba, prenosljivost in povezljivost objektno osnovane programske opreme v distribuiranih, heterogenih okoljih. Ujemanje s specifikacijami omogoča razvoj heterogenih aplikacij za vse pomembnejše računalniške platforme in operacijske sisteme.

Danes pomeni OMG-jeva specifikacija arhitekture upravljanja z objekti (*object management architecture* - OMA) »de-facto« standard med distribuiranimi objektnimi modeli [14]. OMA je konceptualna infrastruktura, na kateri temelji porazdeljeni objektni model CORBA *(Common Object Request Broker Architecture)*.

Slika 1 prikazuje kronološki razvoj skupine OMG in njene pomembnejše dosežke.

Omenimo naj še obstoj drugih porazdeljenih objektnih modelov. Microsoftov OLE 2.0/COM *(Object Linking and Embedding/Component Object Model)* je objektni model, ki pa ni distribuiran. Microsoftov DCOM *(Distributed Component Object Model)* je ugledal luč sveta v Windows NT 4.0. IBM-ov SOM *(System Object Model)*, bil je prvič uporabljen v OS/2, služi kot osnova *OpenDoc*-u. *OpenDoc* podpirajo Apple, Novell, Sun, Xerox, Oracle, IBM in Taligent. Zastavljen je bil neodvisno od operacijskega sistema, vendar je z razvojem daleč za konkurenco. Razširjeni model SOM omogoča dostop do distribuiranih objektov. DSOM (*Distributed* SOM) je skladen s specifikacijo CORBA. Vso funkcionalnost objektnih sistemov pa že nudi NextStep [15].

Od distribuiranih ne-objektnih modelov naj omenimo samo OSF DCE *(Open System Foundation Distributed Computing Environment)*. Le-ta služi kot podlaga številnim CORBA implementacijam in kot osnova Microsoftovega *Distributed Component Object Modela*.

2. Arhitektura upravljanja z objekti

Referenčni model identificira in določa značilnosti komponent, vmesnikov in protokolov, ki sestavljajo arhitek-

[1] *Od januarja 1996 je član OMG tudi Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Inštitut za informatiko.*

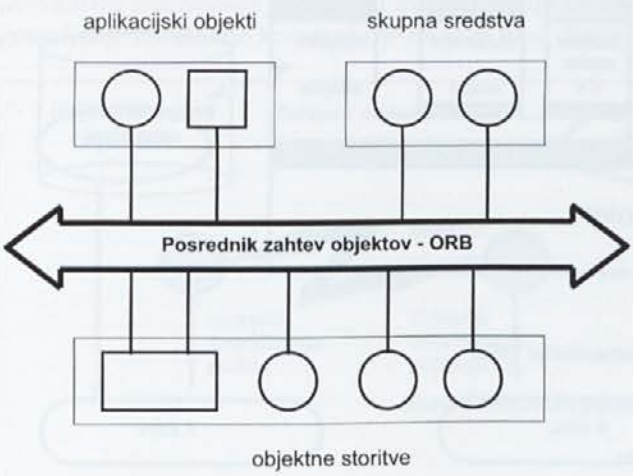
turo upravljanja z objekti, vendar jih natančno ne definira. Referenčni model [12] ima naslednje funkcije:

- Predstavlja ogrodje za pisanje specifikacij.
- Identificira komponente arhitekture upravljanja z objekti.
- Karakterizira funkcije, ki jih nudijo komponente.
- Razlaga relacije med komponentami in zunanjim operacijskim okoljem.
- Identificira protokole in vmesnike za dostop do komponent.

Aplikacija, skladna z arhitekturo upravljanja z objekti, je sestavljena iz množice povezljivih objektov, ki komunicirajo skozi posrednik zahtev objektov (*object request broker* - ORB). Referenčni model določa, kako objekti pošiljajo zahteve in sprejemajo odgovore, določa ključne operacije za vsak objekt in vmesnike objektov, ki nudijo skupna sredstva, uporabna v večini aplikacij.

Referenčni model je sestavljen iz naslednjih komponent (slika 2):

- *Posrednik zahtev objektov (object request broker)* je ključ do povezljivosti. Objektom omogoča transparentno oblikovanje in sprejemanje zahtev in odgovorov v distribuiranem okolju. Je osnova za izgradnjo aplikacij iz distribuiranih objektov in za povezljivost med aplikacijami v heterogenih in homogenih okoljih.
- *Objektne storitve (object services)* so zbirka storitev (vmesnikov in objektov), ki podpirajo osnovne funkcije za uporabo in oblikovanje objektov. Storitve so potrebne za konstrukcijo vsake distribuirane aplikacije in so vedno neodvisne od aplikacijske domene.
- *Skupna sredstva (common facilities)* so zbirka storitev, skupnih mnogim aplikacijam, ki pa niso fundamentalne, kakor so objektne storitve.



Slika 2: **Referenčni model arhitekture upravljanja z objekti**

- *Aplikacijski objekti* so izdelki posameznega dobavitelja. Aplikacijski objekti sovpadajo s tradicionalnim izrazom aplikacija in niso standardizirani s strani OMG. Aplikacijski objekti sestavljajo najvišji nivo referenčnega modela.

V splošnem so aplikacijski objekti in skupna sredstva aplikacijsko orientirani. Posrednik zahtev objektov in objektne storitve so orientirane na infrastrukturo sistema. Od štirih komponent referenčnega modela težijo tri k standardizaciji: posrednik zahtev objektov, objektne storitve in skupna sredstva. Četrta komponenta predstavlja aplikacijske objekte, ki so preveč specializirani za standardizacijo.

Pomembno je poudariti, da morajo aplikacije za sodelovanje v arhitekturi upravljanja z objekti zagotoviti samo vmesnike, skladne z OMA. Ni nujno, da je njihova konstrukcija objektno orientirana. Na sliki 2 so take aplikacije prikazane v obliki pravokotnika, medtem ko krogi ponazarjajo objektno orientirane aplikacije.

3. OBJEKTNI MODEL CORBA

Common Object Request Broker Architecture (CORBA) je konkreten objektni model. Izpeljan je bil iz referenčnega modela arhitekture upravljanja z objekti. Sestavljen je iz jedra in komponent [4].

Objektni model najprej opiše koncepte, razumljive odjemalcem, kot npr. oblikovanje objektov, zahteve in operacije, tipe in podpise. Nato opiše koncepte, povezane z implementacijo objektov strežnikov, med drugim metode in izvršna okolja. Objektni model je najbolj specifičen pri definiciji konceptov, razumljivih odjemalcem.

Ta objektni model je primer klasičnega objektnega modela, kjer odjemalec pošlje sporočilo objektu strežniku. Konceptualno gledano, objekt interpretira sporočilo in se odloči, katero storitev bo izvršil. V klasičnem modelu sporočilo identificira objekt in nič ali več dejanskih parametrov. Kot v večini klasičnih objektnih modelov, tudi tukaj zahtevamo enoličen prvi parameter, ki identificira operacijo, ki jo je potrebno izvesti. Objekt interpretira sporočilo z izbiro metode, bazirane na specifični operaciji.

Objekt je lahko odjemalec, strežnik, ali pa oboje. V določenem trenutku uporablja storitve drugih objektov, v določenem trenutku nudi storitve drugim objektom. Za konkretno akcijo pa so vloge točno definirane.

Obstajati mora način specifikacije storitev, ki jih nudi objekt strežnik. CORBA zato definira jezik za definicijo vmesnikov (OMG *Interface Definition Language* - IDL). Definicija vmesnika specificira operacije, ki jih je objekt pripravljen izvršiti, in vhodne ter izhodne parametre, ki jih potrebuje. Poleg tega so v definiciji navedene izjemne situacije, ki lahko nastopijo. Odjemalci lahko pristopajo do strežnikov samo prek vmesnika.

Arhitektura CORBA torej strogo loči vmesnik objekta od njegove implementacije. Implementacijo objekta napišemo v poljubnem programskem jeziku, za katerega obstaja ustrezna preslikava.

Specifikacija modela CORBA [7] obsega:
- jedro - posrednik zahtev objektov,
- povezljivost,
- preslikave jezika za definicijo vmesnikov v C, C++, Smalltalk in Ado[2],
- objektne storitve,
- skupna sredstva.

Minimalni zahtevi za sistem, združljiv s CORBA, sta upoštevanje specifikacij jedra in ene preslikave IDL-a v programski jezik. Vsaka nadaljnja preslikava v programski jezik je ločena, opcijska točka združljivosti.

3.1. Posrednik zahtev objektov

Odjemalec je entiteta, ki želi, da objekt izvrši operacijo. Implementacija objekta je kombinacija kode in podatkov, ki predstavljajo objekt. ORB je odgovoren za vse mehanizme, potrebne za iskanje implementacije objekta, za pripravo le-te za sprejem zahteve in za komunikacijo. Vmesnik, ki ga vidi odjemalec, je popolnoma neodvisen od tega, kje je objekt lociran, v katerem programskem jeziku je implementiran, ali od česarkoli drugega, kar se ne odraža v vmesniku objekta.

Slika 3 predstavlja strukturo posameznega ORB. Vmesniki k ORB so prikazani v obliki škatel, puščice pa nakazujejo, ali je klican ORB, ali pa ORB izvaja klic navzgor v vmesniku.

Za izvedbo zahteve lahko odjemalec uporabi vmesnik za dinamično proženje (vmesnik, neodvisen od vmesnika ciljnega objekta) ali pa kontrolni odrezek IDL (specifični kontrolni odrezek, odvisen od vmesnika ciljnega objekta). Za nekatere funkcije lahko odjemalec direktno komunicira z ORB skozi vmesnik ORB.

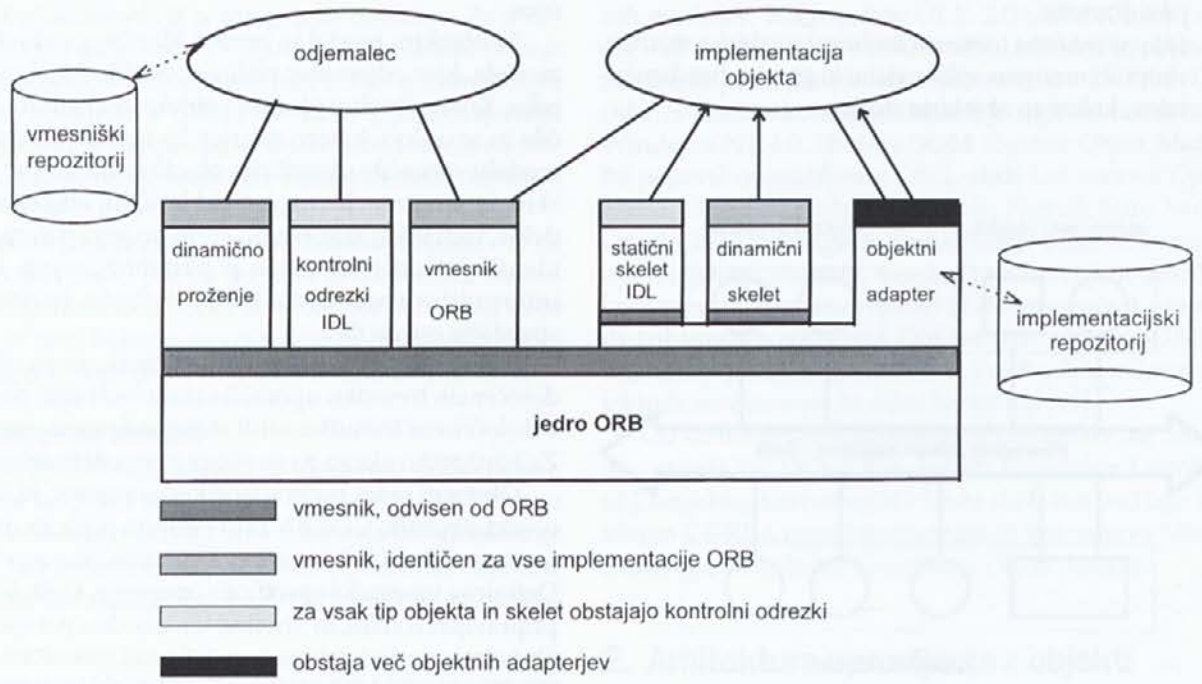
Implementacija objekta dobi zahtevo kot klic navzgor skozi skelet OMG IDL ali skozi dinamični skelet. Med procesiranjem zahteve ali drugače lahko kliče objektni adapter in ORB.

Vmesniki k objektom so lahko definirani na dva načina. Statično v IDL, kar imenujemo OMG IDL - jezik za definicijo vmesnikov. Ta jezik definira tipe objektov glede na operacije, ki jih lahko izvajamo nad njimi, in parametre teh operacij. Vmesniki so lahko dodani v vmesniški repozitorij *(interface repository)*. Ta predstavlja komponente vmesnika kot objekte, kar dovoljuje dostop do njih v času izvajanja. V vsaki implementaciji ORB imata IDL in vmesniški repozitorij enako izrazno moč.

Odjemalec izvrši zahtevo. Pri tem ima dostop do reference objekta in pozna tip in zaželeno operacijo za izvedbo. Odjemalec inicira zahtevo s klicem rutin kontrolnega odrezka, specifičnih objektu ali z dinamično konstrukcijo zahteve.

Dinamični vmesnik in vmesnik kontrolnega odrez-

[2] *V pripravi so preslikave za Javo in COBOL.*



Slika 3: **Struktura posrednika zahtev objektov**

ka za proženje zahteve zadovoljijo isto semantiko zahtev, zato sprejemnik sporočila ne more ugotoviti, kako je bila zahteva prožena.

ORB locira ustrezno implementacijsko kodo, pošlje parametre in prenese kontrolo na implementacijo objekta prek skeleta IDL ali dinamičnega skeleta. Skelet je specifičen vmesniku in objektnemu adapterju. Pri izvršitvi zahteve lahko implementacija objekta uporabi storitve ORB-ja skozi objektni adapter. Ko je zahteva popolna, se kontrolne in izhodne vrednosti vrnejo odjemalcu.

V implementaciji objekta se lahko izbere, kateri adapter bo uporabljen. Ta odločitev bazira na tipu storitev, ki jih implementacija objekta zahteva.

Informacije o implementaciji objekta se shranijo med njegovo instalacijo v implementacijski repozitorij za uporabo med izvajanjem zahtev. Vmesniške in implementacijske informacije, shranjene v ustreznih repozitorijih, so na voljo odjemalcu in implementaciji objekta. Vmesnik definira OMG IDL oziroma vmesniški repozitorij. Definicija se uporablja za oblikovanje odjemalčevih kontrolnih odrezkov in skeletov implementacije objekta.

3.2. Povezljivost posrednikov zahtev

Arhitektura povezljivosti med posredniki zahtev objektov je okvir za definicijo elementov povezljivosti. Opisuje nove mehanizme in specificira konvencije za dosego povezljivosti med neodvisno izdelanimi ORB-ji.

Arhitektura povezljivosti jasno definira vloge različnih tipov domen za specifične informacije. Kjer sta ORB-ja v isti domeni, lahko komunicirata direktno. Navadno je to najbolj zaželen način.

Ko mora informacija proženja zapustiti domeno, pravimo, da gre prek mosta (slika 4). Vloga mosta je zagotoviti preslikavo vsebine in semantike iz oblike enega ORB-ja v drugo tako, da vsak ORB vidi samo ustrezno vsebino in semantiko.

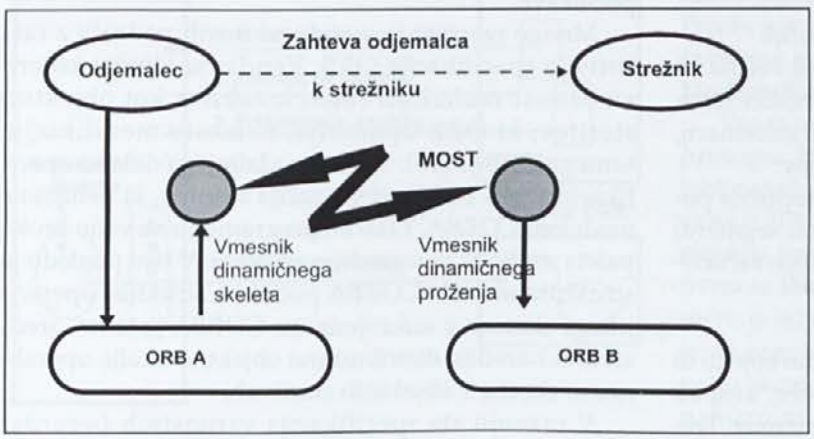
Podporo mostovom med ORB-ji specificira ORB API *(application programming interface)* in konvencije za zagotavljanje enostavne konstrukcije mostov med domenami ORB. Takšne mostove lahko razvije dobavitelj ORB-ja ali kdo drug. Slika 4 prikazuje splošen most med ORB-jema.

Razširitve, potrebne za podporo mostov med ORB-ji, so splošne in ne vplivajo na druge operacije. Uporabne so tudi v druge namene, kot npr. za čiščenje programov.

Most med ORB-ji je uporaben tudi za povezavo s sistemi, ki niso kompatibilni s CORBA, kot je Microsoftov *Component Object Model* - COM. Enostavnost takega početja je odvisna od teh sistemov in njihove podpore modela CORBA.

Protokoli za komunikacijo med ORB-ji so:

- *Splošen protokol med ORB-ji (General Inter-ORB Protocol - GIOP)* specificira sintakso prenosa na nizkem nivoju in množico oblik sporočil za komunikacijo med ORB-ji. Oblikovan je namensko za interakcije med ORB-ji in deluje preko vsakega transportnega protokola, ki ustreza minimalnim zahtevam. Protokol je enostaven, skalabilen in relativno enostaven za implementacijo. Omogoča prenosljive implementacije z majhno porabo pomnilnika in zglednimi zmogljivostmi, z minimalno odvisnostjo od podrejene programske opreme.
- *Internet protokol med ORB-ji (Internet Inter-ORB Protocol - IIOP)* specificira, kako se izmenjujejo GIOP sporočila z uporabo TCP/IP *(Transport Control Protocol/Internet Protocol)* povezav. IIOP specificira standardiziran protokol povezljivosti prek Interneta in omogoča povezljivost z drugimi kompatibilnimi ORB-ji. Uporaben je tudi kot protokol med polmostovi. Predstavlja osnovni protokol med ORB-ji za TCP/IP okolja.
- *Protokoli med ORBji, odvisni od okolja (Environment Specific Inter-ORB Protocol - ESIOP)* slonijo na sredstvih, ki jih nudijo določena okolja. Podpirajo lahko specialna področja, kot npr. varnost in administracijo. Lahko so optimizirani za določeno okolje. Skladati pa se morajo s splošno arhitekturo povezljivosti ORB-jev za omogočanje enostavne povezave z mostovi.



Slika 4: **Splošen most med ORB-jema**

3.3. Objektne storitve

Posrednik zahtev objektov ne more zagotavljati povezljivosti na nivoju semantike aplikacij. ORB lahko primerjamo s telefonsko centralo, ki nudi osnovne mehanizme komunikacije, ne zagotavlja pa pomenske komunikacije med naročniki. Zato so potrebni dodatni vmesniki in protokoli zunaj telefonskega sistema kot so telefonski aparati, modemi in telefonski imeniki. Le-ti so ekvivalentni vlogi objektnih storitev.

Specifikacija objektnih storitev je ponavadi sestavljena iz množice vmesnikov in opisa obnašanja storitev. Za opis vmesnika je uporabljen OMG IDL. Trenutno so standardizirane naslednje objektne storitve [6]:

- *storitve poimenovanja*, ki nudijo standardni pristop za oblikovanje imenskih kontekstov objektov,
- *dogodkovne storitve*, ki razgradijo komunikacijo med objekti,
- *storitve trajnih objektov* ponujajo skupen vmesnik za mehanizem, ki se uporablja za ohranjanje in upravljanje trajnega stanja objektov,
- *storitve življenjskega cikla* definirajo storitve in konvencije za oblikovanje, brisanje, kopiranje in premikanje objektov,
- *storitve kontrole hkratnosti* omogočajo koordiniranje dostopa več odjemalcev do deljenih virov,
- *storitve eksternalizacije* definiranjo protokole in konvencije za eksternalizacijo in internalizacijo objektov
- *relacijske storitve* omogočajo ekspliciten prikaz entitet in relacij za modeliranje entitet realnega sveta,
- *transakcijske storitve* združujejo transakcijski in objektni vzor in naslavljajo probleme komercialnega procesiranja transakcij,
- *storitve povpraševanja* omogočajo uporabnikom in objektom sprožiti povpraševanja na zbirki objektov,
- *licenčne storitve* nudijo mehanizme za nadzor uporabe intelektualne lastnine proizvajalcev.

V prihodnosti pričakujemo razvoj novih objektnih storitev, kot npr. storitev trgovanja in varnostnih storitev.

3.4. Skupna sredstva

Skupna sredstva obsegajo specifikacije višje nivojskih storitev in vertikalnih tržnih področij. Nekateri splošni primeri skupnih sredstev so elektronska pošta, tiskanje, ipd. Ta sredstva so potrebna v večini aplikacijskih domen. Skupna sredstva so primerno področje za standarde pri povezljivosti med neodvisnimi dobavitelji programske opreme [5].

Skupna sredstva so razdeljena v dve kategoriji:

- *Horizontalna skupna sredstva*, ki si jih deli večina ali vsi sistemi. Obstajajo štiri poglavitne množice sredstev: uporabniški vmesnik, upravljanje informacij, upravljanje sistema in upravljanje poslov.
- *Vertikalna tržna sredstva*, ki podpirajo specifična področja, povezana z vertikalnimi tržnimi segmenti (npr. informacijske avtoceste, proizvodnja, računovodstvo, idr.).

Ločnice med skupnimi sredstvi, aplikacijskimi objekti in objektnimi storitvami niso ostre in jasne, ampak odražajo evolucijo tehnologije objektnih sistemov. Trenutna postavitev ločnic predstavlja trenutne ukrepe standardizacije OMG. Z večanjem izkušenj in zrelostjo aplikacij bomo odkrili in definirali nova skupna sredstva. Prav tako se bodo deli skupnih sredstev preselili v objektne storitve.

3.5. Praktična uporaba

CORBA 2.0 je ugledala luč sveta 15. avgusta 1995 na Object World-u v San Franciscu. Takrat je petnajst podjetij predstavilo svoje neodvisne implementacije CORBA, na katerih je delovala ena distribuirana aplikacija. Ta demonstracija je prikazala zrelost in dostopnost tehnologije. Pričakujemo, da bo do konca tisočletja uporabljalo arhitekturo CORBA prek devet milijonov računalnikov, kar bo predstavljalo nekako 75% trga distribuiranih objektov [3].

Pomembnejši dobavitelji posrednikov zahtev objektov, ki so skladni s CORBA, so Hewlett Packard, DEC, Sun, Iona, Expersoft in Visigenic.

Danes uporabljajo posrednike zahtev objektov, ki so skladni s CORBA, številna podjetja [9]: banke *(Bank of Boston, Chemical Bank, IBM Credit Corp.)*, zavarovalnice *(Travelers)*, letalstvo *(Allied Signal Aerospace Technology, Space Telescope Science Institute - Hubblov teleskop)*, transport *(Narfolk Southern Railroad, Wheels)*, telekomunikacije *(Bell SYGMA Telecom Solutions, British Telecom Labs, Time Warner Communications)* in mnogi drugi. V večini primerov se je z novimi aplikacijami povečala produktivnost in stroški so se znižali. Prav tako je tehnologija porazdeljenih objektov omogočila prehod iz velikih računalnikov na omrežje osebnih računalnikov.

4. Nadaljnji razvoj

Kljub splošnem sprejemu specifikacije CORBA delo OMG-ja ni končano. Trenutno poteka razvoj na standardizaciji jezikovnih preslikav za COBOL in Javo. Čeprav lahko jezikovno preslikavo oblikuje vsak, je zelo pomembna standardizacija. Tako model CORBA že uporabljajo z Objective C, Eiffel, Common Lisp, Scheme in drugimi jeziki, za katere še ne obstajajo standardne preslikave.

Mnogo razprav je o podpori novih področij z razširitvijo specifikacije ORB. Vendar je ključni kriterij zmožnost realizirati funkcionalnost kot objektno storitev, ki ORB uporablja, namesto modifikacije samega ORB-ja [13]. Večina nadaljnjega dela na specifikaciji OMG bo standardizacija storitev, ki delujejo s modelom CORBA. Tako bo programerju na voljo široka paleta sredstev za izgradnjo aplikacij. V tem pogledu je struktura modela CORBA podobna strukturi operacijskega sistema z mikrojedrom: CORBA je jedro, sredstva, ki naredijo distribuirano objektno okolje uporabno, so zbrana v objektnih storitvah.

V razvoju sta specifikacija varnostnih *(security)* storitev [8] in storitev trgovanja *(trading)*.

4.1. Povezava z objektnimi podatkovnimi bazami

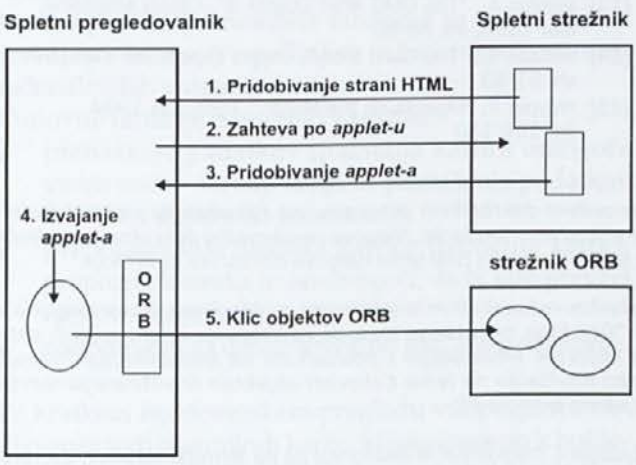
Upravljanje distribuiranih objektov in objektne podatkovne baze sta trenutno vodilni tehnologiji objektne revolucije. Objektne podatkovne baze so komercialno na voljo že skoraj pet let in so našle uporabo v številnih aplikacijskih domenah. V povezavi z upravljanjem distribuiranih objektov jih lahko uporabimo za implementacijo unificiranih distribuiranih rešitev.

Zelo poenostavljeno lahko rečemo, da je objektna podatkovna baza odgovorna za podaljšanje življenjske dobe objekta onstran procesa, ki je objekt oblikoval. Rečemo lahko, da objektna podatkovna baza upravlja z lokalnimi objekti [1].

Naslednji stavek najlepše oriše pomensko razliko med posredniki zahtev objektov (ORB) in objektnimi podatkovnimi bazami: posrednik zahtev objektov izroči zahtevo objektu, medtem ko objektna podatkovna baza izroči objekt zahtevi.

Slika 5: **ODBMS kot upravitelj objektov znotraj OMG arhitekture**

Obstaja več načinov povezave ORB-jev z objektnimi podatkovnimi bazami. Katero bomo izbrali, je odvisno od zahtevane funkcionalnosti. V najenostavnejšem primeru lahko objektna podatkovna baza nudi trajnost



Slika 6: **Integracija WWW z modelom CORBA in Javo**

objektom ORB[11]. Drugi način integracije je definicija objektnega adapterja za podatkovne baze (slika 5). Tretja možnost integracije je vmesnik IDL k API-ju objektne podatkovne baze ali h knjižnici razredov.

4.2. Porazdeljeni objekti in Internet

Danes sta pojma Internet in Svetovni splet (WWW) zelo popularna. Pomeni Svetovni splet smrt klasičnih arhitektur distribuiranih objektov, kot je CORBA? Odgovor je vsekakor *ne*. Pravzaprav je prav integracija spleta z arhitekturo CORBA idealno priložnost, da prevzame tehnologija distribuiranih objektov vodilno vlogo [2, 10].

CORBA je v osnovi strežniško orientirana arhitektura, ki veže stanje objekta in njegovo obnašanje s strežniškim procesom. Odjemalci pošiljajo zahteve objektom, ki so lahko kjerkoli. Odjemalci in strežniki lahko zamenjujejo vloge, kolikokrat želijo. Za določeno zahtevo pa so vloge odjemalca in strežnika strogo definirane.

Alternativa strežniško orientirani arhitekturi je takšna, kjer se objekti prosto premikajo med strežnikom in odjemalcem, s ciljem delovati lokalno. Odjemalno orientirani sistemi so uporabni pri aplikacijah, kjer samo beremo, ali je sprememb malo. Prednost je v tem, da lahko povezavo med strežnikom in odjemalcem prekinemo, kar je zelo ugodno za globalno distribucijo.

Največja pomanjkljivost pri implementaciji arhitekture CORBA je, da morajo biti odjemalcu metode objekta znane vnaprej. Z drugimi besedami, ko pridobivamo stanje objekta iz strežnika, mora biti nekaj kode na voljo lokalno, da lahko objekt uporabimo. Ravno tukaj pa vskoči WWW, ki zapolni praznino z Javo, Visual Basicom in drugimi jeziki, ki omogočajo, da iz strežnika naložimo kodo in podatke in jo izvedemo lokalno (glej sliko 6). Razvijemo lahko splošnega odjemalca z minimalnim vgrajenim znanjem. Znati mora samo pridobivati oddaljene objekte in jih interpretirati. Istočasno lahko naložene skripte znova pokličejo strežnik, kar daje svobodo mešanja arhitekturnih stilov.

Za možnost integracije distribuiranih objektov in svetovnega spleta se zavzema SunSoft, ki želi standardizirati jezikovno preslikavo med OMG IDL in Javo. Proces je v teku, medtem pa so številni dobavitelji izdali razvojne komplete, med drugimi: JavaSoft Joe, PostModern BlackWidow in Iona OrbixWeb.

Vsa ta orodja zagotavljajo integracijo na podobnem principu. Java program *(applet)*, ki se naloži iz spleta, lahko vsebuje odjemalčeve kontrolne odrezke, oblikovane iz IDL. Na ta način nudi vmesnik odjemalca nazaj objektu, implementiranemu na strežniku ORB. Alternativno se lahko uporabi tudi JavaScript. V vsakem primeru je odjemalčevi strani na voljo podlaga za izvajanje, ki zagotovi povezavo kode Java z objekti, ki so na voljo skozi ORB. To, skupaj z integracijo s splošnimi namiznimi orodji (npr. spletnimi pregledovalniki), napravi Javo idealnega odjemalca CORBA.

Kombinacija arhitekture CORBA in spleta torej obljublja veliko prilagodljivost za razvijalce, kakor tudi jasno začrtano pot k množični uporabi distribuiranih objektov.

5. Sklep

Tehnologija porazdeljenih objektov naslavlja ključne probleme razvoja programske opreme, kot so ponovna uporaba, prenosljivost in povezljivost programske opreme. Vodilo, v obliki posrednika zahtev (ORB), omogoča objektom življenje kjerkoli v omrežju in zagotavlja komunikacijo med njimi. Odjemalci dostopajo do njih s proženjem metod. Vmesnik in implementacija objekta sta ločeni, odjemalci so odvisni samo od vmesnika. Programski jezik in prevajalnik, uporabljena za oblikovanje porazdeljenih objektov, sta popolnoma transparentna za odjemalce. Odjemalcem ni potrebno vedeti, kje je distribuiran objekt, ali na kakšnem operacijskem sistemu se izvaja. Porazdeljeni objekti so »pametni« koščki programske opreme, ki lahko med seboj komunicirajo, ne glede na lokacijo.

Ko govorimo o porazdeljenih objektih, govorimo dejansko o neodvisnih programskih komponentah. Komponenta je torej objekt, ki ni vezan na določen program, programski jezik ali implementacijo. Komponenta je dovolj majhna, da jo lahko oblikujemo in vzdržujemo, dovolj velika, da jo uporabljamo in podpiramo ter ima standardni vmesnik za povezljivost. Objekti, zgrajeni kot komponente, so gradniki distribuiranih aplikacij naslednje generacije. Pomagajo nam zgraditi aplikacije, ki so boljše, hitrejše in uporabnejše od današnjih. Zgrajene so z manj napora in so zanesljivejše.

Objekti so del revolucije komponent, ki smo jo že doživeli pri strojni opremi. Predstavljajo programski ekvivalent mikro čipa in posledično osebnega računalnika [3]. Tehnologija komponent, kot nadaljevanje distribuiranih objektov, bo sčasoma izpodrinila vse ostale tehnologije, saj zna vse, kar znajo druge tehnologije, in to bolje.

Uporabljene kratice:

API	Application Programming Interface
COM	Component Object Model (Microsoft)
CORBA	Common Object Request Broker Architecture
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model (Microsoft)
DEC	Digital Equipment Corporation
DSOM	Distributed System Object Model (IBM)
ESIOP	Environment Specific Inter-ORB Protocol - protokol med ORB-ji, odvisen od okolja
GIOP	General Inter-ORB Protocol - splošen protokol med ORB-ji
HTML	Hyper Text Markup Language
IBM	International Business Machines
IDL	Interface Definition Language - jezik za definicijo vmesnikov
IIOP	Internet Inter-ORB Protocol - Internet protokol med ORB-ji.
ODBMS	Object Database Management System - sistem za upravljanje z objektno podatkovno bazo
OLE	Object Linking and Embedding (Microsoft)
OMA	Object Management Architecture - arhitektura upravljanja z objekti
OMG	Object Management Group
ORB	Object Request Broker - posrednik zahtev objektov
OSF	Open System Foundation
RFP	Request for Proposal
SOM	System Object Model (IBM)
TCP/IP	Transport Control Protocol/Internet Protocol
WWW	World Wide Web - svetovni splet

Literatura:

[1] Barry D., "ODBMSs and Distributed Systems", *Object Magazine*, August 1996, str. 28- 30

[2] Chauvet J. M., "ORB Spiders On The Web", *Object Expert*, Sept/Oct 96, str. 29-31

[3] Guttman M., Matthews J. R., *The Object Technology Revolution*, John Wiley & Sons, Inc., USA 1995

[4] Jurič M. B., *CORBA - objektna zasnova porazdeljenega procesiranja*, FERI Maribor, oktober 1996

[5] OMG, *CORBAfacilities: Common Facilities Architecture*, Revision 4.0, November 1995

[6] OMG, *CORBAservices: Common Object Services Specification*, Revised Edition March 31, 1995, Updated: March 28, 1996

[7] OMG, *The Common Object Request Broker: Architecture and Specification*, Revision 2.0, July 1995

[8] OMG: "CORBA Security", *First Class*, Jun-Jul 1996

[9] OMG: "Meeting the Challenge of Vertical Industries", *First Class*, Aug-Sep 1996

[10] OMG: "Surfin' the Net with CORBA", *First Class*, Jan-Feb 1996

[11] Reddy M., "ORBs & ODBMSs: Two complementary ways to distribute objects", *Object Magazine*, Jun 1995, str. 24-31

[12] Soley R. M., Ph.D., *Object Management Architecture Guide*, OMG, John Wiley & Sons, Inc., Revision 3.0, Third Edition, Jun 15, 1995

[13] Watson A., "The OMG After CORBA 2", *Object Magazine*, Mar 1996, str. 58-60

[14] Watson A., "The OMG Story", *Object Expert*, Jan-Feb 1996, str. 51-53

[15] Wayner P., "Objects on the March", *Byte*, Jan 1994, str. 139-150

◆

Matjaž B. Jurič je diplomiral leta 1996. Študij nadaljuje po enovitem doktorskem programu na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru kot štipendist Slovenske znanstvene fundacije. Njegovo raziskovalno delo obsega področje objektne tehnologije s poudarkom na porazdeljenih objektnih sistemih, kar bo tudi tema njegove doktorske disertacije.

Mag. Marjan Heričko je asistent stažist na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru. Naziv magistra je pridobil leta 1993 z uspešnim zagovorom magistrske naloge "Objektno orientirane metode načrtovanja informacijskih sistemov". Njegovo raziskovalno razvojno delo obsega vse vidike objektne tehnologije, s poudarkom na metodologijah razvoja, orodjih CASE, razvojnih okoljih in metrikah. Pripravlja doktorsko disertacijo na temo Kakovost objektno orientiranega razvoja informacijskih sistemov. Aktivno sodeluje pri delu Centra za objektno tehnologijo.

Dr. Ivan Rozman je diplomiral na Fakulteti za elektrotehniko v Ljubljani, magistriral in doktoriral pa na Tehniški fakulteti Univerze v Mariboru. Je redni profesor Univerze v Mariboru in ustanovitelj Laboratorija za informacijske sisteme, ki ga vodi še danes. Je avtor več kot 200 publikacij, vodil je številne znanstveno raziskovalne projekte.