PRAKTIČNE IZKUŠNJE S PORAZDELJENIMI OBJEKTI PRI RAZVOJU SISTEMA ZA ELEKTRONSKO BANČNIŠTVO

Matej Trampuš, Matjaž Trontelj

Povzetek

Članek podaja osnovni pregled komponentnega objektnega modela (COM) in navaja praktične izkušnje o uporabi CORBE in COMa za komunikacijo med porazdeljenimi objekti, do katerih sta avtorja prišla ob razvoju Sistema za Elektronsko Bančništvo (SEB). Hkrati nakazuje tudi dileme, s katerimi se soočajo razvijalci aplikacij v sodobnem tehnološkem okolju.

COM

COM je komponenten objektni model. Microsoft ga je začel razvijati leta 1988, prva verzija pa je bila izdana leta 1993. Nastal je kot odgovor na vse močnejše zahteve po ponovni uporabljivosti in medsebojnem sodelovanju objektov različnih proizvajalcev [1]. Prva tehnologija, ki je uporabljala COM, je bila OLE 2.0 (Object Linking and Embedding). COM so hitro prevzeli tudi drugi proizvajalci: leta 1997 je bil trg COM komponent vreden 410 milijonov dolarjev (vir: Giga), ocenjujejo pa, da bo do leta 2001 ta vrednost presegla 2,8 milijarde dolarjev. Te številke ne vključujejo Microsoftovih produktov. S četrto verzijo operacijskega sistema Windows NT je prišel DCOM, ki omogoča, da med seboj komunicirajo tudi objekti, ki se nahajajo na različnih računalnikih in operacijskih sistemih. S ciljem, da omogoči sodelovanje pri razvoju tudi drugim proizvajalcem, je Microsoft prenesel nadzor nad modelom DCOM na Open Group.

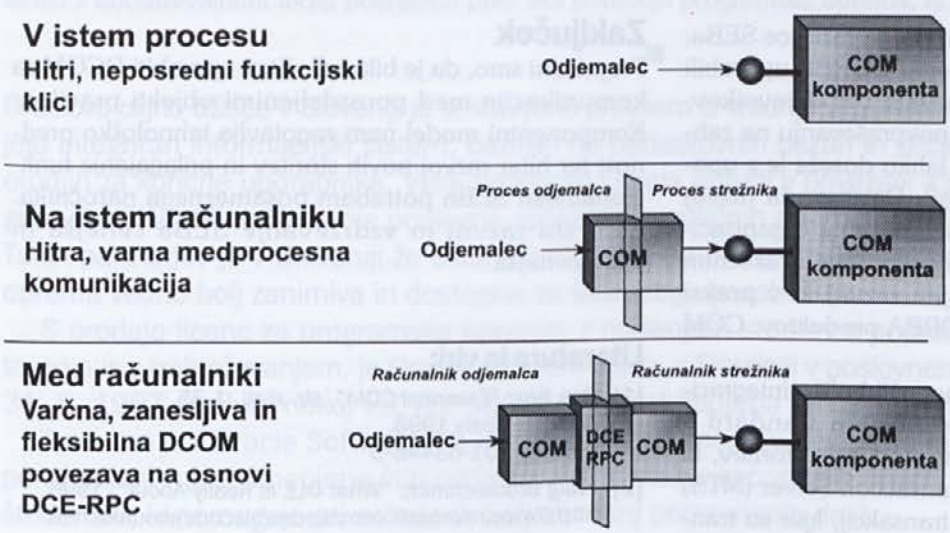
Komponentni model kot razširitev objektnega modela

Večina klasičnih tekstov o objektnem programiranju označuje jezik ali sistem kot objektno orientiran, če leta podpira enkapsulacijo, polimorfizem in dedovanje. Poglejmo si, kako COM zadošča tem zahtevam [1,2]. Odjemalci dostopajo do COM objektov preko natančno definiranih vmesnikov. O načinu implementacije ne vedo ničesar. Pravzaprav je COM na tem mestu še strožji od nekaterih drugih objektnih modelov: dostop do javnih članov je možen le preko metod, ki so natančno definirane v vmesnikih. COM objekti torej podpirajo enkapsulacijo. Polimorfizem je podprt na več načinov: prvi vmesnik, ki je z dedovanjem izpeljan iz drugega vmesnika, je polimorfičen le-temu, saj lahko odjemalec, ki potrebuje metode prvega vmesnika, uporablja tudi drugega. Razredi (in instance teh razredov), ki podpirajo isti vmesnik, so polimorfični v tem vmesniku - odjemalec lahko kliče metode določenega vmesnika na isti način ne glede na to, kateri objekt implementira ta vmesnik. Podoben polimorfizem velja tudi za skupine vmesnikov. Pri dedovanju moramo biti previdni: klasični teksti namreč obravnavajo dedovanje kot mehanizem za ponovno uporabljivost kode, ponavadi pa pozabijo povedati, da obstajata dve vrsti dedovanja. Dedovanje implementacije pomeni, da se deduje dejansko obnašanje objekta, dedovanje vmesnikov pa, da se deduje le specifikacija obnašanja. Dedovanje je le sredstvo, ki nam omogoča polimorfizem (skozi dedovanje vmesnikov), v nekaterih objektnih modelih pa tudi ponovno uporabljivost (skozi dedovanje implementacije). COM podpira le dedovanje vmesnikov. Ponovno uporabljivost COM objektov je mogoče doseči na dva načina. Prvi način, pri katerem zunanji objekt za implementacijo določene funkcionalnosti uporablja odjemalcu nevidne notranje objekte, je vsebovanost. Tu med zunanjim in notranjim objektom obstaja klasičen odnos odjemalec-strežnik. Drugi, redkeje uporabljeni način, je agregacija. Pri agregaciji zunanji objekt od notranjega zahteva kazalec na določen vmesnik in s tem implementacijo vmesnika. Zunanji objekt ta kazalec vrne odjemalcu. Tako se lahko ponovno uporabi celotna implementacija določenega vmesnika. Tudi tu odjemalec ne ve ničesar o notranjem objektu - z njegovega stališča vrnjeni vmesnik implementira zunanji objekt, čeprav dejansko neposredno kliče metode notranjega objekta.

COM je več kot samo objektni model. Je tudi binarni standard, ki med drugim predpisuje, kako različni objekti komunicirajo med sabo, standardne načine za lociranje in kreiranje novih objektov ter binarni format podatkov, v katerem so zapisani podatki o razredih, vmesnikih in podatkovnih tipih. Ker je COM binarni standard, omogoča, da nove komponente (tako imenujemo objekte, združljive na binarnem nivoju) dodajamo v sistem med tem, ko le-ta obratuje. Novo komponento lahko uporabimo takoj, ko je nameščena na sistemu, ne da bi bilo zaradi tega potrebno kakorkoli spreminjati ali ponovno prevajati odjemalce. Tako v hipu razširimo razvoj ali izboljšamo kvaliteto storitev, ki jih nudi aplikacija. Za distribuiranje komponent zadostujejo izvršilne datoteke. COM postavlja temelje, ki omogoča sodelovanje med objekti ne glede na to, v katerem programskem jeziku so implementirani.

Lokacijska neodvisnost objektov

Klic metod strežniškega objekta je za odjemalca enak ne glede na to, kje se strežniški objekt dejansko nahaja. Pri kreiranju nove instance strežniškega objekta lahko odjemalec pove, kje naj se novi objekt kreira. Strežnik se lahko nahaja v istem procesu kot odjemalec, v drugem procesu na istem računalniku ali na oddaljenem računalniku. V primeru, da se strežnik in odjemalec nahajata v istem procesu, so klici praktično enako hitri kot klici virtualnih funkcij v drugih programskih jezikih. Če strežnik (ali odjemalec) potrebuje večjo varnost (svoj naslovni prostor, drugačne varnostne nastavitve,...), se lahko strežnik izvaja v ločenem procesu. V primeru, da je strežnik vezan na vire, ki se nahajajo na drugem računalniku (močni procesorji, komunikacijske naprave, podatkovne zbirke,...), se strežnik lahko nahaja tudi na oddaljenem računalniku. V tem primeru se uporabi DCOM. DCOM je "COM z daljšo žico" in za komunikacijo med objekti uporablja ORPC - razširjeno verzijo DEC-ovega RPC (Remote Procedure Call). Podpira vse glavne mrežne protokole (TCP, UDP, IPX,SPX,...). Objekti se lahko nahajajo tudi v različnih operacijskih sistemih (Sun Solaris,Linux, IBM OS/390,...).

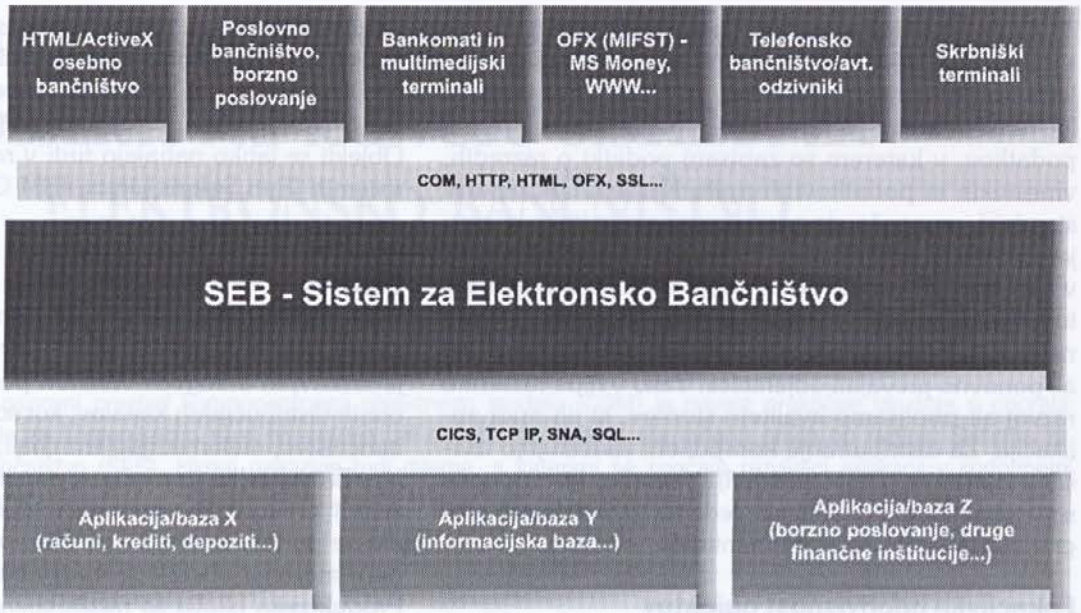


SEB

V ZASLONu uporabljamo COM med drugim tudi pri razvoju Sistema za Elektonsko Bančništvo (SEB). SEB je rešitev, ki omogoča bankam poslovanje preko sodobnih distribucijskih kanalov, kot so internet, telefonsko bančništvo, multimedijski terminali, avtomatski odzivniki... Banke želijo to možnost ponuditi svojim strankam zaradi konkurenčne prednosti, ki jo le-ta prinaša, vendar se pri tem soočajo s problemom, ki ga predstavlja neprimernost obstoječe informacijske infrastrukture. Produkcijska okolja so razdrobljena, sestavljena iz več ločenih aplikacij in podatkovnih baz, ki uporabljajo različne načine komunikacije. Oviro predstavlja tudi podatkovni model, v katerem so podatki vezani na aplikacijo in uporabnik ni predstavljen kot samostojna entiteta. Obstoječi distribucijski kanali so realizirani kot samostojne in neodvisne aplikacije, ki si ne delijo skupne baze podatkov, skrbniškega sistema in dostopa do aplikacij produkcijskega okolja. Razvoj in vzdrževanje takih storitev sta draga, zahtevna in časovno potratna.

Namen SEBa

SEB ponuja bankam učinkovito rešitev tega problema. SEB igra vlogo vmesnika, ki s pomočjo distribuirane objektne tehnologije omogoča dostop do obstoječih bančnih aplikacij na način, ki najbolj ustreza sodobnim oblikam elektronskega poslovanja. Vsebuje centralni varnostni modul, ki podpira sodobne varnostne mehanizme (kartice za generiranje enkratnih gesel, digitalni certifikati...) in bazo podatkov, prek katere se tradicionalni podatkovni model, v katerem so podatki organizirani po aplikacijah (računi, krediti...), preslika v obliko, kjer je stranka predstavljena kot celota. Poleg tega vsi distribucijski kanali vezani na SEB uporabljajo enoten skrbniški sistem, kar občutno poenostavi in poceni vzdrževanje in razvoj.



Izkušnje s CORBO

Za potrebe SKB banke smo začeli leta 96 razvijati prvo različico SEBa. Za komunikacijo med porazdeljenimi objekti smo uporabili Visigenicovo implementacijo CORBE. V več kot šestih mesecih delovanja v produkciji se je izkazala za solidno rešitev. Danes SKB NET uporablja že več kot 2500 uporabnikov. Pri uporabi CORBE pa smo naleteli na več pomankljivosti. Veliko časa in energije smo porabili za izbiro ustrezne implementacije CORBE. Približno tri mesece smo testirali različne produkte (Visigenic Visibroker, IONA Orbix, Digital ObjectBroker). Rešitve so bile v tistem času še precej nezrele in tudi sami smo aktivno sodelovali pri iskanju in odpravi programskih hroščev v Visibrokerju in Orbixu. Na koncu nas je čas prisilil, da smo se odločili za Visibroker, čeprav ni izpolnjeval vseh naših zahtev. Najbolj smo pogrešali varnostne storitve in upravljanje s transakcijami. Mehanizmi za dostop do virov podatkov niso bili integrirani v obstoječe implementacije CORBE. Zato smo podatke, ki smo jih preko ODBC knjižnice prebrali iz naše SQL baze, morali sami pretvarjati v CORBA podatkovne strukture.

Prehod na DCOM

Jeseni 97 smo pričeli z razvojem druge različice SEBa. Odločili smo se, da bomo namesto CORBE uporabili DCOM. K tej odločitvi je prispevalo več dejavnikov. SEB je produkt, ki mora slediti povpraševanju na zahtevnem bančnem tržišču, to pa lahko doseže le z uporabo najnaprednejših tehnologij. Dejstvo, da razvoj CORBE poteka pod okriljem mednarodne organizacije (OMG) in je implementacija prepuščena različnim proizvajalcem, upočasnjuje njen razvoj in v praksi otežuje povezljivost različnih CORBA produktov. COM teh težav nima.

Prehod na COM nam je omogočil boljšo integracijo z Windows NT okoljem, ki postaja standard v finančnih inštitucijah in uporabo dodatnih storitev, ki temeljijo na COMu. Microsoft Transaction Server (MTS) podpira distribuirano izvajanje transakcij, kjer so transakcije predstavljene kot COM komponente. COM Transaction Integrator prikaže obstoječe IBM CICS/IMS transakcije kot COM komponente v MTS. OLE DB omogoča univerzalne dostop do relacijskih, hierarhičnih in drugih virov podatkov. Naslednja verzija SEBa bo vsebovala OFX/Gold strežnik, ki omogoča komunikacijo preko OFX/Gold protokola, reliziranega s pomočjo Microsoft Internet Financial Server Toolkita. Pri razvoju naših aplikacij izkoriščamo močno podporo COMu v sodobnih razvojnih orodjih, kot so: Visual Java, Visual C++, Delphi, PowerBuilder... Podpora za COM je vključena v sam operacijski sistem, zato ni dodatnih stroškov povezanih z namestitvijo in razvojem COM aplikacij. Razvojne licence za CORBO stanejo od 2000 do 7500 dolarjev po razvijalcu.

Zaključek

Prepričani smo, da je bila odločitev uporabiti DCOM za komunikacijo med porazdeljenimi objekti pravilna. Komponentni model nam zagotavlja tehnološko prednost ter hiter razvoj novih storitev in prilagajanje funkcionalnosti SEBa potrebam posameznega naročnika. Zato sta razvoj in vzdrževanje SEBa cenejša in učinkovitejša.

Literatura in viri:

[1] Don Box: "Essential COM", str. xi-xii, 1-35, Addison-Wesly,1998, ISBN: 0-201-63446-5

[2] Kraig Brockschmidt: "What OLE is Really About", 1996, http://www.microsoft.com/oledev/olecom/aboutole.htm