DCE ŞI CORBA: SINTEZĂ ŞI STUDIU COMPARATIV

dr. ing. Daniela Saru

Universitatea Politehnica, *Bucureşti*

Rezumat: Numărul mare de cerinţe şi complexitatea problemelor ce trebuie rezolvate în domeniul sistemelor distribuite eterogene actuale au condus, în ultimii ani, la apariţia unor tehnologii de elaborare a aplicaţiilor software, care să asigure atât programatorilor, cât şi utilizatorilor acestui tip de sisteme cât mai multe facilităţi, garantând performanţe ridicate şi uşurinţă în exploatare. O parte a acestor tehnologii sunt produse de firmă (proprietare) nestandardizate, în timp ce altele, mai puţine, dar din multe puncte de vedere mai valoroase, sunt specificaţii standardizate, elaborate în cadrul unor organisme specializate şi concretizate, deja, în implementări performante. Dintre acestea, în cadrul articolului, sunt prezentate separat (în sinteză) şi comparativ tehnologiile DCE şi CORBA, fiind evidenţiate elementele comune şi principalele deosebiri, valoarea şi avantajele oferite de către cele două tehnologii în funcţie de domeniul de utilizare, lipsurile existente şi câteva dintre posibilele remedii.

Cuvinte cheie: DCE, CORBA, sisteme distribuite eterogene, client, server, interfaţă, limbaj de definire a interfeţelor (IDL), UUID, RPC, ORB, servicii CORBA, facilităţi CORBA (orizontale şi verticale), *stub*, *skeleton*.

1. DCE (Distributed Computing Environment)

DCE (Distributed Computing Environment) este o tehnologie standard, elaborată de către *OSF (Open Software Foundation)* într-o primă variantă în anul 1992, urmată apoi de reactualizări periodice. Principalii producători de componente software ce se conformează acestui standard sunt, printre alţii: AT&T, Bull, DEC, Hitachi, IBM, SCO, Tandem, Transarc, Hewlett Packard.

DCE permite scrierea şi integrarea aplicaţiilor de tip client/server având la bază limbaje procedurale în cadrul mediilor distribuite eterogene. Pentru definirea interfeţelor implementate în server-e, DCE foloseşte un limbaj de tip IDL *(Interface Definition Language)* [1] şi asigură transparenţa între aplicaţii, preluând toate sarcinile legate de situarea în reţea, particularităţile platformei hardware şi ale sistemului de operare, limbajul folosit pentru implementare, formatul datelor, convenţiile de apelare, protocoluri în reţea etc.

Principalele componente ale unei arhitecturi de tip DCE sunt grupate în cadrul executivului şi al serviciilor extinse **(figura 1).**

Executivul DCE conţine următoarele componente:

- servicii de securitate *(Security Services)* care asigură:
  - autentificare (clientul şi server-ul pot demonstra cine sunt);
  - autorizarea accesului (pe baza unei liste de acces, server-ul poate stabili dacă clientul are dreptul de a obţine serviciul cerut);
  - integritate (se verifică dacă informaţia recepţionată coincide cu cea transmisă);
  - confidenţialitate (informaţiile pot fi transmise în formă criptată);
- servicii de administrare a directoarelor *(Directory Services)* care permit administrarea domeniilor DCE locale (numite *celule*) şi cuprind:
  - serviciul de administrare a celulelor *(Cell Directory Service)*;
  - serviciul de administrare global *(Global Directory Service)*;
  - serviciul de administrare a domeniilor *(Domain Name Service)*;
  - agentul de administrare global *(Global Directory Agent)*;
- serviciul de gestiune a timpului *(Distributed Time Service)* care sincronizează ceasurile tuturor maşinilor gazdă atât în cadrul unei celule DCE, cât şi între celule;
- mecanism de apelare la distanţă a procedurilor (RPC) [2] care permite clientului să invoce proceduri oferite de către server, fără a fi preocupat de detalii referitoare la localizarea server-ului în reţea, tipul de maşină sau de sistem de operare pe care se execută, reprezentarea datelor sau protocolul utilizat;
- set de facilităţi pentru lucrul cu mai multe fire de execuţie *(Threads Service)* atât pentru client, cât şi pentru server.

Serviciile DCE extinse cuprind:

- serviciul de fişiere distribuite *(DSF - Distributed File Service)* care implementează un sistem de fişiere logic simplu ce poate fi utilizat (prin *Directory Services*) atât în cadrul celulelor, cât şi între acestea, permiţând replicarea fişierelor şi refacerea acestora în urma defecţiunilor hardware;
- opţiunea de gestiune a reţelei *(Network Management Option)* care permite accesul aplicaţiilor la informaţii de gestiune prin intermediul protocoalelor de reţea specializate (CMIP şi SNMP);
- serviciul de evenimente *(Event Service)* care permite atât sistemului, cât şi aplicaţiilor utilizator efectuarea unor

operaţii cu evenimente: creare, filtrare, redirectare etc.

Deşi este destinat aplicaţiilor ce au ca bază programarea procedurală, DCE asigură totuşi posibilitatea utilizării anumitor mecanisme care se apropie de cele ale sistemelor orientate obiect:

- clientul poate stabili la momentul execuţiei server-ul de care are nevoie (către care va lansa RPC);
- server-ul poate stabili nume unice (UUID - *Universal Unique IDentifier*) pentru diferite obiecte pe care le gestionează; ca urmare, clientul poate folosi un astfel de nume pentru a preciza obiectul pe care doreşte să-l folosească (o anumită imprimantă, de exemplu);
- server-ul poate stabili şi nume unice cu tip, astfel încât clientul care specifică un obiect prin intermediul numelui unic să fie servit de o funcţie selectată în raport cu tipul obiectului respectiv (de exemplu, imprimantele pot fi obişnuite sau PostScript, operaţia de tipărire având caracteristici diferite);
- etc.

Deoarece suportul oferit pentru dezvoltarea aplicaţiilor orientate obiect constituie, în prezent, un criteriu important în alegerea unui sistem software, proiectanţii DCE urmăresc extinderea facilităţilor sale, folosind, în principal, două direcţii distincte [3].

Prima dintre cele două direcţii are în vedere faptul că, în momentul actual, DCE asigură o parte dintre caracteristicile necesare dezvoltării de aplicaţii orientate obiect: gruparea datelor şi a funcţiilor (metodelor) în cadrul server-ului, *încapsulare* (în interfaţa IDL sunt precizate funcţiile pe care clientul le poate invoca în relaţia cu server-ul; deoarece clientul poate accesa datele server-ului numai prin intermediul acestor funcţii, datele server-ului sunt efectiv încapsulate) şi *polimorfism* (server-ul permite accesarea mai multor implementări distincte prin intermediul aceleiaşi interfeţe; clientul utilizează în invocare identificatorul unic universal (UUID) pentru a preciza obiectul pe care doreşte să-l folosească). DCE nu asigură suport pentru *moştenire*, dar este evident că poate fi adaptat în mod corespunzător prin adăugare de componente (interfeţe C++ cu DCE). În prezent, există deja un produs de firmă (OODCE - Hewlett Packard) care extinde DCE cu ajutorul unor biblioteci de clase C++, destinat programatorilor ce doresc să folosească limbajul C++ într-un mediu DCE. Un alt produs, de data aceasta elaborat sub egida OSF (independent de o anumită firmă), îşi propune să includă o versiune îmbunătăţită de limbaj de specificare (IDL) cu orientare obiect, asemănătoare celor existente în CORBA sau OLE, şi o interfaţă de programare a aplicaţiilor (API) C++, asemănătoare celei deja existente în OODCE.

Cea de-a doua direcţie urmăreşte utilizarea DCE (nemodificat) din medii orientate obiect, care asigură facilităţi suplimentare pentru calcul distribuit de acest tip (CORBA, DCOM). De exemplu, o mare parte dintre firmele care comercializează produse ORB compatibile CORBA (DEC, HP, IBM etc.) implementează aceste produse folosind ca bază un produs DCE. Acest lucru este posibil deoarece standardul CORBA defineşte doar interfeţele, nu şi implementarea lor, astfel încât un produs CORBA poate asigura suportul necesar acestor interfeţe folosind, de exemplu, DCE.

2. CORBA (Common Object Request Broker Architecture)

CORBA(Common Object Request Broker Architecture) este o specificaţie standardizată elaborată într-o primă variantă în anul 1991 de către OMG *(Object Management Group)* pe baza OMA *(Object Management Arhitecture)* [1] şi urmată, până în prezent, de alte câteva versiuni îmbogăţite substanţial din punctul de vedere al facilităţilor oferite. Dintre principalii producători de componente software ce se conformează acestui standard pot fi menţionaţi: DEC, Expertsoft, HP, IBM, IONA Technologies, SunSoft, ICL etc.

CORBA permite scrierea şi integrarea aplicaţiilor de tip client/server având la bază limbaje orientate obiect în cadrul mediilor distribuite eterogene. La fel ca şi DCE, pentru definirea interfeţelor implementate în server-e, CORBA foloseşte un limbaj de tip IDL şi asigură transparenţa între aplicaţii, preluând toate sarcinile legate de situarea în reţea, particularităţile platformei hardware şi ale sistemului de operare, limbajul folosit pentru implementare, formatul datelor, convenţiile de apelare, protocoale în reţea etc.

OMG (Object Management Group) este un consorţiu nonprofit, format din peste 750 de membri (persoane fizice şi firme ce comercializează produse software), al cărui scop este promovarea tehnologiei orientate obiect în dezvoltarea sistemelor de calcul distribuite. Concret, OMG creează, prin intermediul unor specificaţii standardizate, un cadru arhitectural pentru aplicaţiile orientate obiect. *OMA (Object Management Architecture)* constituie arhitectura standard, elaborată de către OMG, care permite ca, pe baza unor interfeţe orientate obiect, deschise, standardizate, să se proiecteze şi să se implementeze componente software portabile, care să satisfacă cerinţe de interoperatibilitate şi de reutilizare.

Structura actuală a OMA provine dintr-o evoluţie cronologică a conceptelor şi a nivelurilor asociate, aşa cum au fost percepute de către cei

care au participat la elaborarea lor [1] **(figura 2).** Scopul OMA este ca funcţionalitatea de bază a aplicaţiilor să fie furnizată prin intermediul unei interfeţe standard, ceea ce face posibilă atât existenţa mai multor implementări ale aceleiaşi funcţionalităţi (care pot să difere din punctul de vedere al performanţei, preţului sau adaptării la diferite platforme specializate), cât şi crearea unor componente specializate, care să folosească această funcţionalitate prin intermediul interfeţei standard. Structura OMA, prezentată în figura 2, se conformează acestui scop: *serviciile CORBA* standardizează serviciile de bază, necesare oricărui obiect, în timp ce *facilităţile CORBA* folosesc aceste servicii, furnizând un alt nivel de servicii aplicaţiilor prin standardizarea gestionării informaţiilor utilizate în comun. Cel mai înalt nivel al arhitecturii, asociat obiectelor aplicaţie, nu va fi standardizat de către OMG, pentru a permite producătorilor software să îşi manifeste inventivitatea, elaborând cu un cost minim produse cât mai competitive.

Pentru fiecare dintre componentele OMA, OMG a publicat câte o specificaţie care cuprinde reguli de sintaxă (precizează modul de invocare a operaţiilor asupra obiectelor în limbaj OMG IDL) şi semantica asociată. Producătorii de pachete software, ce se conformează OMA, implementează şi vând servicii care pot fi accesate prin interfeţele IDL astfel specificate. Datorită acestui mod de standardizare, utilizatorul are posibilitatea de a folosi servicii ce provin de la producători diferiţi, nefiind nevoit să achiziţioneze un set complet de la acelaşi furnizor, ceea ce îi asigură o mai mare flexibilitate în configurarea propriului mediu sofware.

Componentele unei arhitecturi de tip CORBA sunt grupate în: ORB *(Object Request Broker)*, servicii CORBA şi facilităţi CORBA (orizontale şi verticale) (figura 3), ultima dintre aceste categorii nebeneficiind încă de specificaţii standard finalizate [6].

ORB (Object Request Broker) reprezintă componenta de bază a arhitecturii (CORBA), având ca sarcină principală asigurarea infrastructurii de comunicaţie, necesară accesului transparent la obiecte. ORB permite utilizatorului să invoce operaţii asociate obiectelor fără ca acesta să se preocupe de localizarea obiectelor în reţea, tipul hardware-ului pe care se execută, sistemul de operare, limbajele cu ajutorul cărora sunt implementate, deosebirile legate de modul de reprezentare a datelor sau protocoalelor de reţea utilizate. Standardul CORBA specifică atât funcţiile ce trebuie îndeplinite de ORB, cât si un set de interfeţe standard pentru aceste funcţii.

Deoarece într-o arhitectură OMA fiecare client şi fiecare obiect pot comunica numai prin intermediul unuia sau mai multor ORB-uri, fiecare serviciu CORBA este accesibil fiecărui obiect din sistem, OMA nefiind o arhitectură stratificată, chiar dacă în descrierea ei se foloseşte conceptul de "nivel" cu referire la componentele sale de bază **(figura 4).**

Serviciile CORBA pot fi grupate astfel [4]:

- servicii destinate sistemelor distribuite:
  - serviciu de gestiune a numelor *(Naming Service)* - asigură crearea şi desfiinţarea asocierii de legături între nume şi obiecte şi obţinerea referinţelor la obiecte prin intermediul acestor asocieri;
  - serviciu de gestiune a evenimentelor *(Event Service)* - asigură notificarea apariţiei evenimentelor definite în program către toate componentele interesate;
  - serviciu de securitate *(Security Service)* - asigură autentificarea, autorizarea accesului la obiecte sau grupuri de obiecte, integritatea acestora şi confidenţialitatea în timpul comunicaţiei;
  - *Trading Service* - permite obiectelor sau aplicaţiilor client aflate în căutarea unui anumit serviciu să găsească obiectele care oferă serviciul respectiv pe baza unui criteriu specificat;
  - servicii destinate bazelor de date:
  - *Concurrency Control Service* - serviciu destinat protejării datelor aparţinând obiectului în cazul în care există mai multe cereri concurente;
  - *Property Service* - permite asocierea de valori la numele *(named values)* unui obiect, valorile cu nume fiind echivalentele dinamice ale atributelor obiectului;
  - *Transaction Service* - asigură cadrul necesar pentru ca, un calcul format din una sau mai multe operaţii ce implică unul sau mai multe obiecte, să îndeplinească cerinţele de atomicitate (dacă tranzacţia este întreruptă accidental, toate rezultatele intermediare sunt ignorate), de izolare (chiar dacă tranzacţiile se execută concurent, se obţin aceleaşi rezultate ca în cazul în care se execută serial) şi de durabilitate (rezultatele unei tranzacţii încheiate cu succes nu se pierd decât în caz de "catastrofă"). Permite realizarea sau întreruperea tranzacţiilor care implică mai multe baze de date distincte, de acelaşi tip sau de tipuri diferite;
  - *Relationship Service* - asigură crearea, desfiinţarea, gestionarea şi itinerarea (parcurgerea) legăturilor dintre obiecte. Ca exemplu de

legătură se poate menţiona proprietatea de "a fi conţinut" a unui obiect "document" în cadrul unui obiect "dosar";

- *Query Service* - permite efectuarea de operaţii asupra unor seturi sau colecţii de obiecte, organizate pe baza unor anumite criterii; foloseşte ca limbaj specializat SQL (pentru baze de date relaţionale) şi OQL (pentru baze de date orientate obiect);
- *Persistent Object Service* - permite conservarea stării obiectelor în cadrul unei baze de date atunci când acestea nu sunt active (în memorie) sau între două execuţii consecutive ale aplicaţiei;
- *Externalization Service* - realizează conversia stării obiectului în/din o formă ce poate fi transmisă între sisteme prin alte mijloace decât cele asigurate de ORB (flux de octeţi);

- servicii generale:
  - *LifeCycle Service* - permite crearea, copierea, transferarea şi distrugerea obiectelor. Ca alternativă mai uşor de utilizat se preferă uneori interfeţele specializate de la nivelul aplicaţie;
  - *Licensing Service* - controlează existenţa licenţei software pe maşina gazdă; efectuează şi alte operaţii specifice;
  - *Time Service* - asigură ceasuri sincronizate tuturor obiectelor, indiferent de localizare. Poate fi folosit, de exemplu, pentru aflarea orei curente sau pentru generarea unui eveniment (sau a unui set de evenimente) după un interval de timp precizat;
  - *Collection Service* - asigură crearea şi manipularea colecţiilor de obiecte.

Facilităţile CORBA sunt utilizate în scopul creării de aplicaţii în domenii din cele mai diverse şi se clasifică în facilităţi *orizontale* (orientate utilizator) şi facilităţi *verticale* (specializate pe domenii).

Fiind o categorie de facilităţi de uz general (pot fi utilizate de aplicaţiile din orice domeniu), *facilităţile CORBA orizontale* asigură:

- interfaţa cu utilizatorul:
  - prezentarea obiectelor (afişare, tipărire etc.) pe ecran, imprimantă sau alte medii;
  - prezentarea documentelor compuse (un exemplu de facilitate CORBA deja acceptată ca standard pentru această categorie este sistemul OpenDoc);
  - asistarea utilizatorului pe durata execuţiei aplicaţiei (de exemplu, oferirea unor informaţii ajutătoare, verificarea corectitudinii gramaticale etc.);
  - gestiunea informaţiilor cuprinse pe ecran (*desktop management*);
  - crearea interactivă a *script*-urilor pentru automatizarea *task*-urilor şi a proceselor (*scripting*).
- gestionarea informaţiilor:
  - modelarea informaţiilor (crearea modelelor şi schemelor de informaţii) - în principal, se descriu tipurile de date, interfeţele obiectelor şi relaţiile dintre obiecte şi se ţine cont de organizarea şi integritatea informaţiilor, precum şi de modalităţile de accesare a acestora;
  - stocarea şi regăsirea informaţiilor în/din memorii permanente (documente, texte, grafice etc.) - adresându-se aplicaţiilor distribute, această categorie de facilităţi utilizează serviciul *Persistent Object* pentru operaţii de iniţializare, căutare, regăsire, control al accesului etc.;
  - asigurarea schimbului de date în cadrul documentelor compuse - permite înlănţuirea şi adnotarea obiectelor ce reprezintă date, conversia acestora la diferite tipuri, schimbul de obiecte *on-line* sau *off-line* etc.;
  - asigurarea schimbului (general de) date, inclusiv conversia formatului;
  - asigurarea schimbului de informaţii;
  - codificarea şi translatarea formatului datelor care sunt folosite în comun de către aplicaţii prin intermediul memoriilor permanente, al protocolurilor de reţea sau prin interfeţele de programare;
  - gestiunea informaţiilor de timp (referitoare la oră şi dată calendaristică) - permite reprezentarea unui grup de informaţii (dată/oră) ca eşantion de timp, durată sau fereastră cuprinsă între două eşantionări;
- gestionarea sistemului - asigură instrumente care să diminueze efortul de administrare a sistemelor distribuite, descrise cu ajutorul unor interfeţe specializate ce pot fi folosite în orice mediu de calcul, creând astfel premisa dezvoltării unor aplicaţii eterogene, interoperabile. Principalele componente ale acestui grup de facilităţi se referă la *instrumente de gestiune*, *gestiunea colecţiilor*, *control* şi se adresează următoarelor clase de utilizatori: administratorii sistemelor de calcul,

creatorii aplicaţiilor destinate gestiunii sistemelor de calcul, furnizorii serviciilor de sistem, planificatorii resurselor sistemului de calcul;

- gestionarea *task*-urilor (lucrărilor) - asigură infrastructura necesară aplicaţiilor pentru a modela şi a folosi *task*-urile utilizator, fiind grupate în patru categorii:
- gestiunea fluxului de lucru (obiecte care automatizează procesul cu care se lucrează);
- agenţi statici sau mobili;
- gestiunea regulilor (specificarea şi prelucrarea unor reguli declarative de tip eveniment-condiţie-acţiune; necesită un limbaj de specificare şi un sistem de calcul, capabile să lucreze cu reguli);
- automatizare (convenţii şi interfeţe care permit accesul la funcţionalitatea unui obiect din cadrul altui obiect prin intermediul *macro*-urilor şi al *script*-urilor).

Facilităţile CORBA verticale, destinate utilizării în aplicaţii din domenii cu cerinţe specifice, constituie nivelul superior al efortului de elaborare a specificaţiilor în cadrul OMG. Câteva dintre principalele domenii avute în vedere sunt:

- sănătate;
- telecomunicaţii;
- finanţe;
- CIM *(Computer Integrated Manufacturing)*;
- simulare distribuită;
- industria extractivă şi de prelucrare a petrolului şi gazelor naturale;
- dezvoltarea aplicaţiilor *(CAD - Computer Aided Design)*;
- Internet.

În prezent, interoperatibilitatea asigurată prin folosirea protocolului IIOP [11] şi setul cuprinzător de servicii oferite recomandă CORBA ca instrument deosebit de performant pentru programatorii experţi, care doresc să construiască sisteme distribuite complexe, multinivel. În perspectivă, se are în vedere extinderea domeniului de aplicare prin includerea unor facilităţi care să permită şi programatorilor mai puţin sofisticaţi să utilizeze arhitectura prin intermediul unor unelte vizuale de dezvoltare a aplicaţiilor şi a unor limbaje de tip *"scripting"* (limbaje de comandă etc.) [12][11]. De asemenea, deoarece o mare parte a aplicaţiilor avute în vedere se află în domeniul WWW, apare ca evidentă necesitatea de a putea numi obiectele cu ajutorul serviciilor *URL (Uniform Resource Locator)* şi *LDAP (Lightweight Directory Access Protocol)*, de a oferi suport de securitate a obiectelor conform SSL-3 folosind specificaţia SSL deja adoptată de către OMG, de a accesa cu uşurinţă obiectele distribuite (implementate în orice limbaj, rezidente pe orice gazdă) din Java şi de a le controla modul de funcţionare cu ajutorul unui limbaj de tip *"scripting"* ca JavaScript. Toate aceste cerinţe sunt luate în considerare de către OMG şi constituie, deja, obiectul unor grupuri de lucru ale acestui organism internaţional [6].

În următorii ani, se preconizează ca tehnologia obiectuală, standardizată de către OMG, să fie încorporată în sistemele de operare ce vor fi livrate o dată cu staţiile de lucru, ale celor mai importante firme specializate. Datorită interoperatibilităţii ORB-urilor (asigurată de standardul CORBA 2.0), a gamei largi de servicii şi a facilităţilor de programare orientată obiect oferite, este de aşteptat ca această platformă de calcul distribuit (CORBA) să devină un standard unanim acceptat, care să permită crearea unor aplicaţii deosebit de performante [5].

3. Studiu comparativ: DCE, CORBA

Cele două tehnologii, CORBA şi DCE, analizate în cadrul acestui articol, permit crearea şi integrarea aplicaţiilor în mediile distribuite eterogene, având multe asemănări atât în modul de operare, cât şi în privinţa facilităţilor pe care le oferă.

Un exemplu în acest sens este limbajul de definire a interfeţelor, IDL *(Interface Definition Language)* care foloseşte ca bază limbajul C++ pentru CORBA şi limbajul C pentru DCE. Ambele modele folosesc IDL pentru definirea interfeţelor implementate în server-e, adică a serviciilor ce vor fi utilizate de potenţialii clienţi. Compilarea sursei IDL creează un *stub* client şi un *stub* server (numit *skeleton* în CORBA) care sunt folosite în momentul în care aplicaţia client formulează o cerere către aplicaţia server (**figura 5**).

Un alt exemplu este modul în care se realizează transmiterea cererilor de servicii şi a răspunsurilor între clienţi si server-e. Întreaga responsabilitate revine CORBA sau DCE, creatorii sau utilizatorii aplicaţiilor nefiind preocupaţi de probleme ca situarea în reţea, platforma hardware, sistemul de operare, limbajul folosit pentru implementare (de exemplu, formatul datelor sau convenţiile de apelare), protocoluri în reţea etc.

Există, însă, şi deosebiri semnificative. CORBA este o arhitectură destinată, încă din faza de proiectare, aplicaţiilor orientate obiect, în timp ce DCE a fost creat iniţial pentru aplicaţiile ce folosesc limbaje procedurale. Specificaţiile CORBA definesc numai interfeţe, lăsând în seama producătorilor software alegerea unei variante de implementare, în timp ce DCE furnizează utilizatorilor atât specificaţiile standard, asociate interfeţelor, cât şi implementările referinţă ale

acestora, implementări care constituie fundamentul tuturor produselor DCE. (Acesta este şi motivul pentru care se pot crea produse compatibile CORBA, care să folosească pentru implementare DCE).

Pe lângă cele două deosebiri esenţiale dintre CORBA şi DCE, există însă şi anumite deosebiri de detaliu, care vor fi prezentate, pe scurt, în continuare.

Datorită momentului iniţierii procedurii de elaborare a specificaţiilor standard (pentru CORBA, după ce apăruseră deja produse destinate scopului propus iar pentru DCE înainte ca acest lucru să se întâmple), în implementările CORBA aparţinând unor producători diferiţi se regăsesc atât corespondentele celor mai importante interfeţe, cât şi o mare diversitate de caracteristici "moştenite" din versiunile anterioare standardizării. Produsele DCE, derivate virtual din acelaşi cod sursă de bază, au caracteristici asemănătoare, oricare ar fi producătorul lor [5].

În cadrul arhitecturii CORBA, o componentă de bază este ORB *(Object Request Broker)*, care asigură conectarea şi comunicarea grupurilor de obiecte. Şi în acest caz, există specificaţia standard a interfeţei asociate, implementarea fiind lăsată la latitudinea producătorului. Un ORB poate fi implementat, de exemplu, folosind RPC, un mecanism de transmitere a mesajelor etc., poate exista sub formă de proces sau bibliotecă "legată" cu obiectele client şi server, poate fi inclus în sistemul de operare sau poate să apară ca o combinaţie a variantelor prezentate anterior. Gama largă de posibilităţi provine din modul de adoptare a specificaţiilor [6]: deoarece o mare parte a membrilor comisiilor specializate din cadrul OMG reprezentau firme care implementaseră deja produse specializate, definirea abstractă a componentei ORB a permis fiecărei firme să păstreze rezultatele deja obţinute şi să adauge acestora anumite elemente care să realizeze alinierea la noile standarde. Din acest motiv, o parte a interfeţei ORB (cea asociată componentei server) nu a fost încă definitiv standardizată, neasigurând în momentul de faţă un suport corespunzător pentru portabilitatea aplicaţiilor. O soluţie a acestei probleme este documentul OMG, destinat standardizării unui adaptor de obiecte, care să înlocuiască *BOA (Basic Object Adapter)* [1][7] numit *POA (Portable Object Adapter)*. Documentul face parte din *ORB Portability Joint Submission (Final)* şi se numeşte *ORB Portability Enhancement/POA* [8].

CORBA oferă două modalităţi de realizare a invocării unui obiect server de către un obiect client [1][9][10]. Prima dintre ele, realizată prin intermediul interfeţei de invocare statică (SII), este asemănătoare apelurilor de proceduri la distanţă (RPC) din DCE. Cea de-a doua, realizată prin intermediul interfeţei de invocare dinamică (DII) şi implicând folosirea "depozitului" de interfeţe (IR), oferă aplicaţiilor un grad ridicat de flexibilitate, şi nu are corespondent în cadrul DCE.

Serviciile CORBA au fost standardizate astfel încât să poată fi folosite în contextul celor mai importante servicii similare. De exemplu, serviciul *Naming* poate fi utilizat împreună cu *Cell Directory Service, X.500 (OSI)* şi *NIS+ (Sun)*. Modalitatea aleasă în cadrul standardizării oferă avantajul asigurării flexibilităţii, dar şi dezavantajul păstrării ca funcţii în cadrul serviciului numai a celor rezultate ca intersecţie a funcţiilor oferite de cele trei servicii menţionate. De asemenea, există specificaţii în care, pentru anumite interfeţe sunt precizaţi parametrii care pot fi omişi sau care nu sunt definiţi, ceea ce generează probleme considerabile pentru cei care doresc să elaboreze seturi de teste pentru verificarea conformanţei produselor CORBA (în vederea garantării portabilităţii). Prin contrast, pentru produsele DCE (derivate virtual din acelaşi cod sursă de bază) acest aspect nu constituie o problemă, testele de conformanţă există şi sunt folosite.

Produsele compatibile CORBA conforme standardului CORBA 2.0 asigură comunicarea între ORB-uri furnizate de diferiţi producători prin intermediul IIOP *(Internet Inter-ORB Protocol)* [1][9]. În plus, ORB-urile pot folosi şi alte protocoluri de interoperatibilitate *(ESIOPs - Environment-Specific Inter-ORB Protocols)*, cum este, de exemplu, DCE CIOP *(DCE Common Inter-ORB Protocol)* care are la bază DCE RPC. Pentru protocolul folosit în comunicaţia dintre ORB-uri furnizate de acelaşi producător nu există însă restricţii, ceea ce permite apariţia unor produse CORBA mult mai diferite între ele decât în cazul standardului DCE, care impune folosirea unui protocol unic (DCE RPC) şi a aceloraşi interfeţe de programare a aplicaţiilor în toate produsele DCE.

Din punctul de vedere al tipurilor de date ce pot fi folosite în cadrul celor două arhitecturi, pot fi menţionate următoarele deosebiri:

- DCE permite folosirea unor tipuri de date care nu există în CORBA:
  - tablou variabil - tablou de dimensiune fixată, din care numai o parte a componentelor sunt transmise între client şi server (server-ul are alocat spaţiul maxim şi poate returna mai multe elemente decât i-au fost transmise). În CORBA, se poate obţine o comportare echivalentă prin folosirea unei secvenţe CORBA;
  - *pipes* - permit transmiterea unui volum mare de valori de parametri în serii de blocuri cu dimensiune redusă. În CORBA, se poate

implementa un mecanism asemănător (*pipelining*) doar prin fragmentarea conceptuală a unei operaţii într-o serie de operaţii;

- pointeri (ca parametri şi în cadrul parametrilor transmişi operaţiilor) - executivul DCE înglobează valoarea adresată de pointer în informaţiile ce vor fi transmise server-ului. Dacă se transmite ca parametru o structură complexă care include pointeri, executivul înglobează în informaţiile ce vor fi transmise toate valorile adresate, se realizează transmiterea către server şi se reconstituie în cadrul acestuia parametrul complex. Deoarece în CORBA nu există un tip de date echivalent, transmiterea ca parametru a unei structuri ce conţine pointeri se poate realiza prin una dintre următoarele metode:
  - scriere de cod suplimentar care să convertească la transmisie valorile adresate de pointeri în tipuri de date asociate în cadrul clientului şi care apoi să reconstituie parametrul complex cu pointeri în server;
  - redefinirea structurii complexe ca o colecţie de unul sau mai multe obiecte;
- CORBA permite folosirea tipului de date **any**, care poate reprezenta o valoare de orice tip care să fie transmisă între client şi server. Valoarea are asociat un cod care indică tipul corespunzător la un moment dat;
- DCE IDL nu permite folosirea mecanismului de moştenire şi defineşte un spaţiu de nume liniar. CORBA IDL permite folosirea moştenirii multiple şi defineşte un spaţiu de nume ierarhizat;
- CORBA defineşte un "depozit" de interfeţe (IR) [1] care conţine informaţii echivalente celor din fişierele IDL. Acestea pot fi inspectate şi folosite în momentul execuţiei;
- Server-ele DCE trebuie activate din exteriorul DCE. În CORBA, activarea server-elor se poate realiza şi automat (funcţia este îndeplinită de către ORB);
- Serviciile puse la dispoziţia aplicaţiilor de către DCE sunt mai puţin numeroase, dar în unele cazuri mai clar standardizate decât componentele arhitecturii CORBA care, însă, oferă utilizatorilor un suport vast şi flexibil pentru dezvoltarea aplicaţiilor.

4. Concluzii

Articolul prezintă trei ministudii de sinteză, două individuale şi unul comparativ, dedicate tehnologiilor DCE şi CORBA, alese ca reper dintre tehnologiile folosite, în prezent, pentru scrierea de aplicaţii software în sisteme distribuite eterogene. În articol, sunt descrise cele mai semnificative caracteristici, avantaje şi domenii de utilizare ale celor două tehnologii, fiind evidenţiate elementele comune şi principalele deosebiri, valoarea şi avantajele oferite în funcţie de domeniul de utilizare, lipsurile existente şi câteva dintre posibilele remedii. Pentru fiecare dintre tehnologiile analizate, se menţionează principalele obiective de perspectivă, care pot fi sintetizate astfel:

- extinderea suportului oferit de tehnologia DCE pentru dezvoltarea aplicaţiilor orientate obiect (prin adăugare de componente - interfeţe C++ cu DCE, respectiv prin folosirea DCE ca bază pentru medii orientate obiect, care asigură facilităţi suplimentare pentru aplicaţii distribuite, orientate obiect);
- extinderea domeniului de aplicare a tehnologiei CORBA prin includerea unor facilităţi care să o facă mai accesibilă şi mai atractivă pentru programatorii care nu fac parte din categoria experţilor: unelte vizuale de dezvoltare a aplicaţiilor, limbaje de tip *"scripting"* (limbaje de comandă etc.), pachete de componente, facilităţi WWW;
- includerea tehnologiei CORBA în sistemele de operare, livrate o dată cu staţiile de lucru, într-un mod asemănător DCOM/Windows NT şi elaborarea unor specificaţii standard pentru asigurarea interoperabilităţii aplicaţiilor scrise conform DCE, CORBA, DCOM, care să concretizeze scopul principal al tehnologiei CORBA: integrarea în sisteme distribuite, orientate obiect a componentelor/aplicaţiilor scrise în limbaje de programare diferite şi executate pe platforme hardware/software diferite.

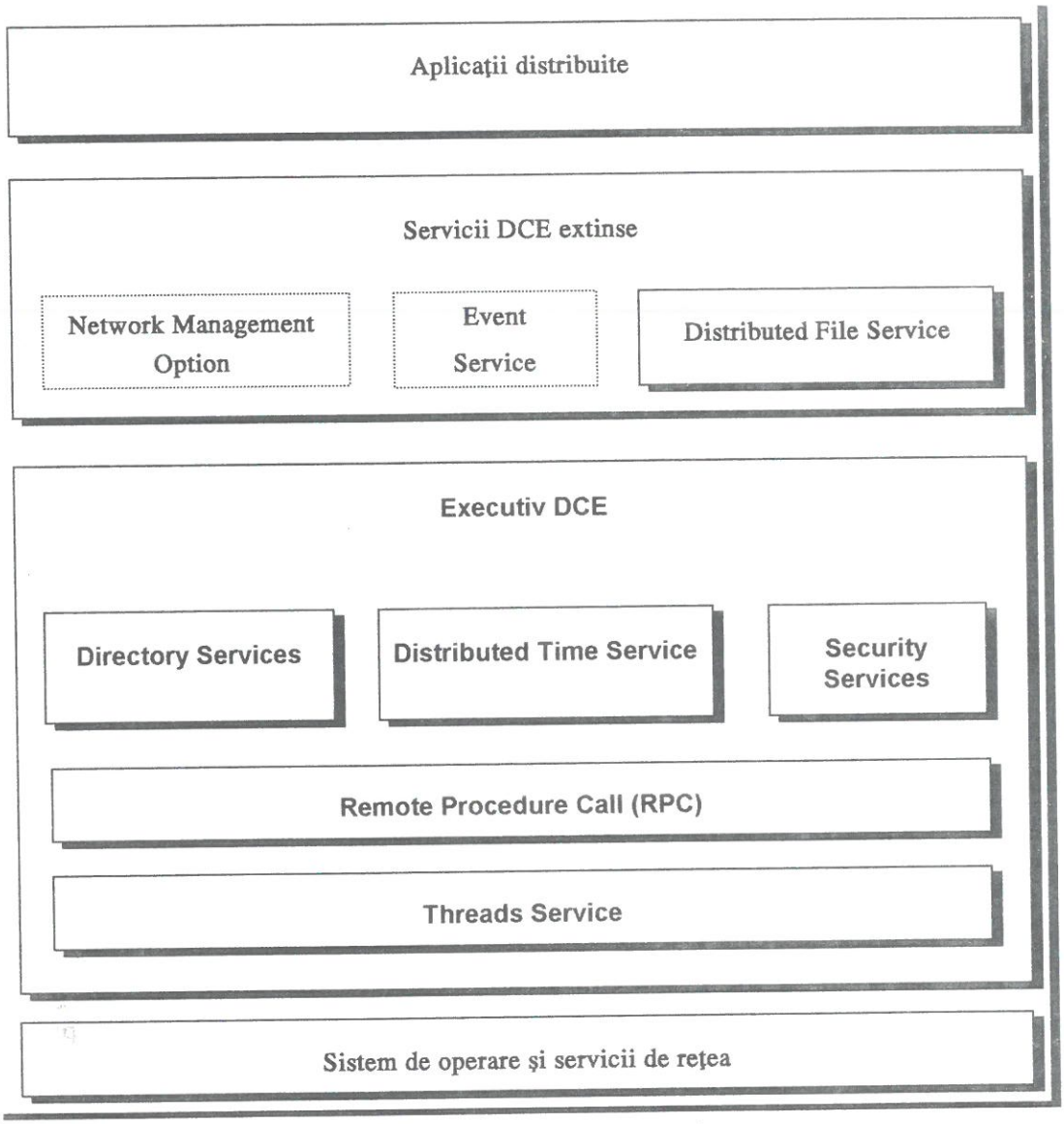

Figura 1. Arhitectura DCE OSF

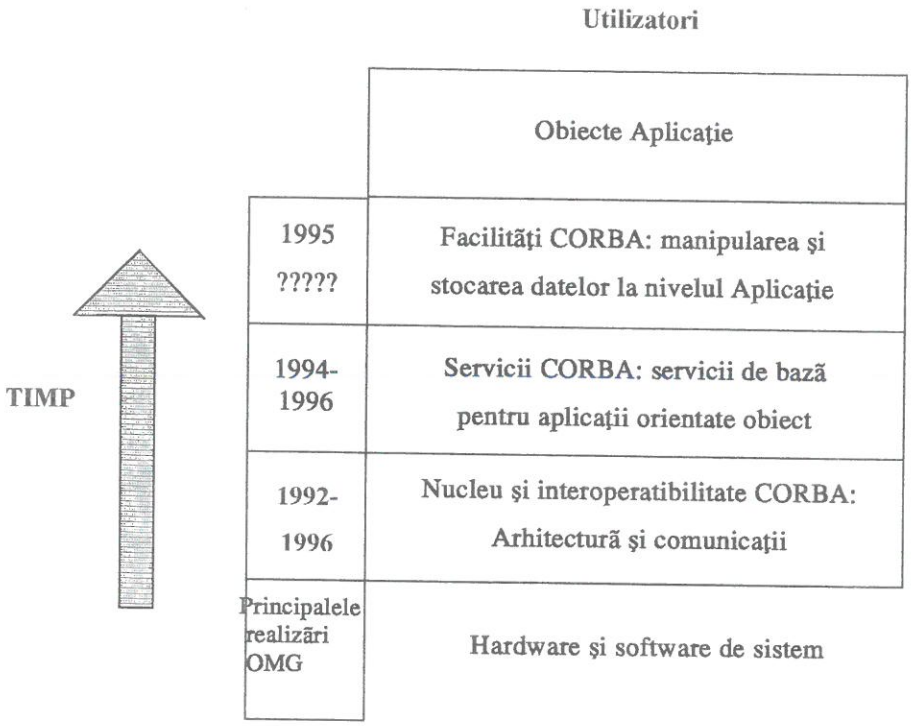

Figura 2. CORBA şi OMA. Evoluţie

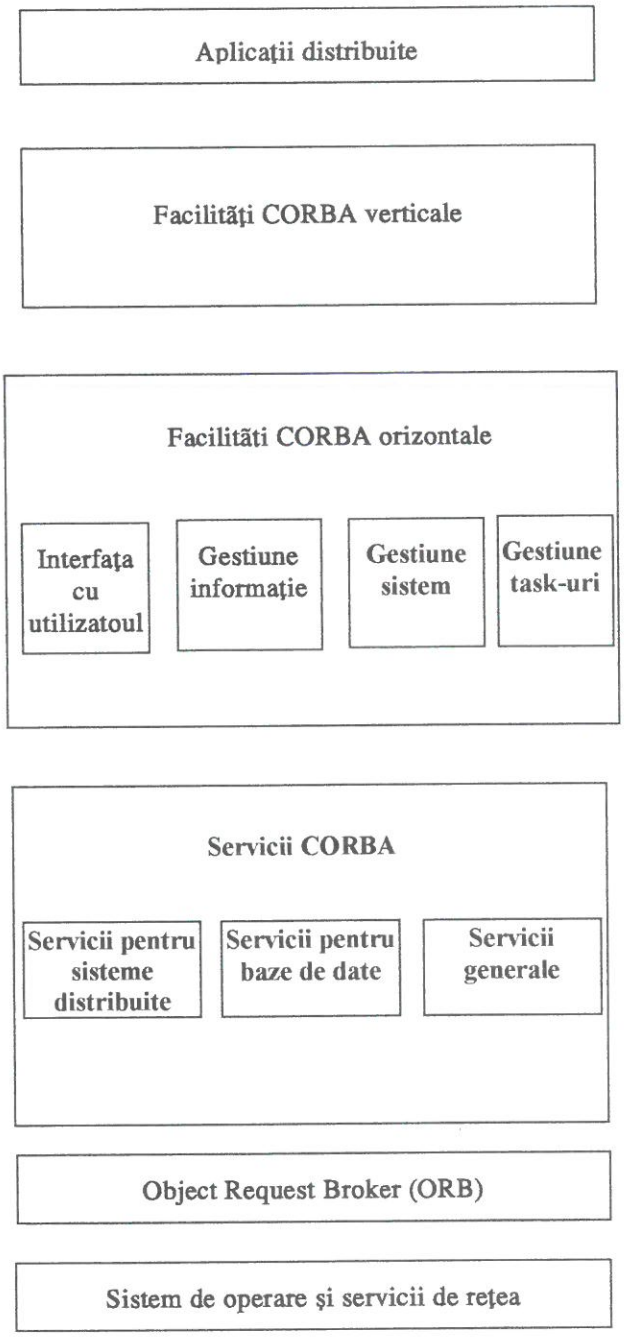

Figura 3. Arhitectura CORBA OMG

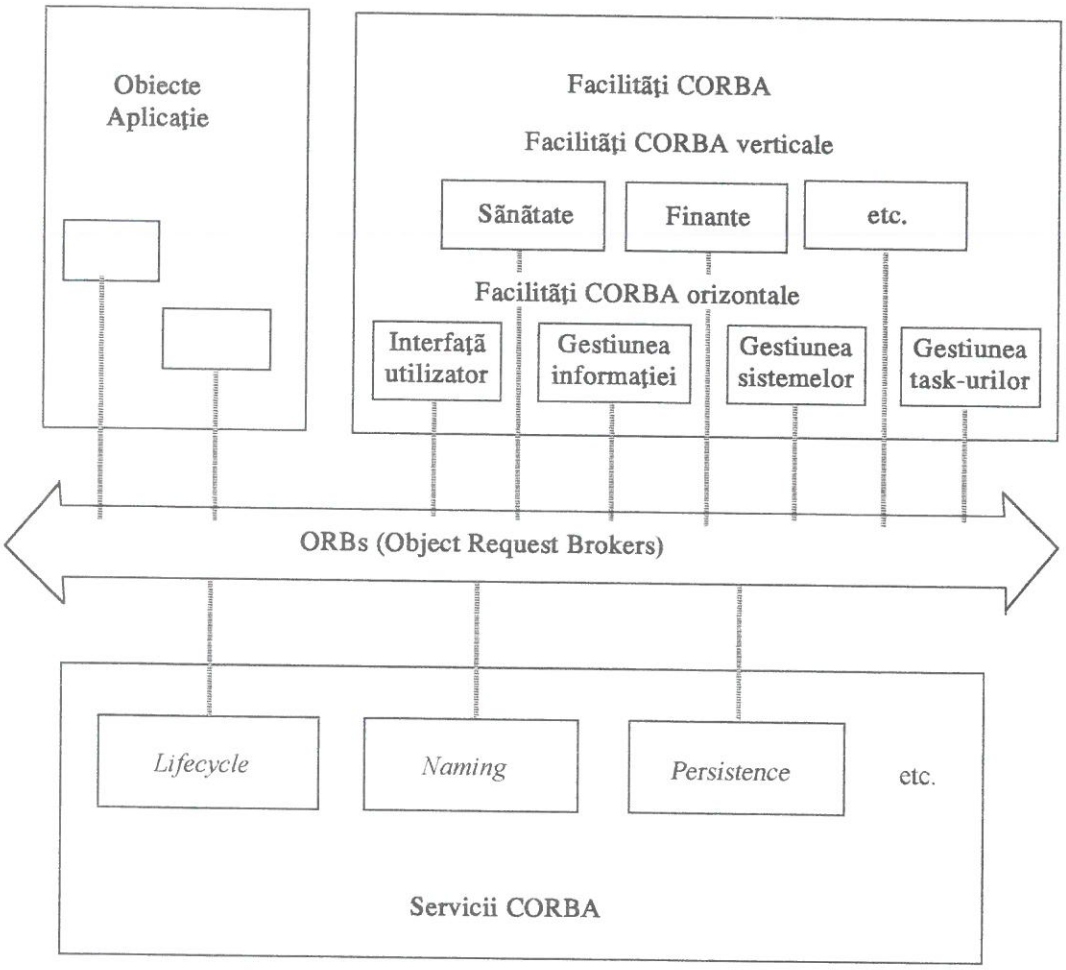

Figura 4. Structura OMA (Object Management Architecture)

Bibliografie

1. **SIEGEL, J.:** CORBA Fundamentals and Programming, Wiley Computer Publishing Group, USA,1996.
2. * * *: AES/Distributed Computing - Remote Procedure Call, Revision B - http://www.osf.org/dce
3. **CHAPPELL, D.:** DCE Today - http://www.opengroup.org/tech/dce/3rd-party/ChapRpt1.htm
4. **BAKER, S.:** CORBA Distributed Objects Using ORBIX, Addison-Wesley, USA,1997.
5. **BRANDO, TH., J.:** Comparing DCE and CORBA. MITRE Document MP 95B-93 - http://www.mitre.org/research/domis/reports/DCEvCORBA.html
6. * * *: OMG Technology Committee Work in Progress. OMG Technology Adoptions - http://www.omg.org/library/schedule.htm
7. * * *: OMG Recently Adopted Specifications- http://www.omg.org/library/schedule/Technology_Adoption.htm
8. * * *:ORB Portability Joint Submission (Final) Part1 of 2, ORBOS/97-05-15. ORB Portability Enhancement (POA - Portable Object Adapter) -ftp://ftp.omg.org/pub/docs/orbos/97-05-15.pdf
9. * * *:CORBA 2.0 - http://www.omg.org/corba/corbaiiop.htm
10. * * *:CORBA 2.1 - http://www.omg.org/corba/c2indx.htm
11. **CURTIS, D., STONE, CH., BRADLEY, M.:** IIOP: OMG's Internet Inter-ORB Protocol. A Brief Description - http://www.omg.org/news/news.htm#articles
12. * * *:CORBA Component Imperatives, ORBOS/97-05-25 - http://www.omg.org/news/610pos.htm