SISTEME INFORMATICE CLIENT/SERVER

Alexandru Bărbulescu

Academia de Studii Economice Bucureşti

Rezumat: Dezvoltarea sistemelor informatice, trecerea de la sistemele mainframe la sistemele client/server au făcut posibilă livrarea mult mai rapidă a informaţiei. Este bine ştiut că, într-o economie sau într-o companie, informaţia este un element de bază. Un al doilea element important, realizat prin introducerea sistemelor client/server, este reprezentat de calculul distribuit şi distribuirea bazei de date. *Calculul distribuit* se referă la împărţirea unei aplicaţii în două sau mai multe părţi şi distribuirea şi procesarea acestor aplicaţii distribuite pe două sau mai multe calculatoare fie clienţi, fie server. Într-un mediu distribuit, datele trebuie să poată fi stocate în multiple locaţii fizice, pe diferite tipuri de servere, într-o manieră transparentă, utilizatorului final. Aceste servere multiple de baze de date trebuie să poată comunica unele cu altele chiar dacă ele nu rulează sub acelaşi DBMS. Faţă de DBMS-urile clasice, acestea trebuie într-un fel adaptate şi pentru medii distribuite.

Cuvinte cheie: downsizing, upsizing, smartsizing, rightsizing, modelul PAD.

1. Aspectele economice ale sistemelor client/server

Pe măsură ce companiile căutau un avantaj competitiv în recesiunea economică de la începutul anilor '90, bunul cel mai de preţ folosit ca pârghie s-a dovedit a fi informaţia. Mai precis, furnizarea oportună a informaţiei, în formatul potrivit, către decidentul potrivit, la locul şi timpul potrivit poate fi factorul de diferenţiere între succes şi eşec, în viaţa economică a zilelor noastre. De o mare importanţă s-a dovedit a fi încă un criteriu: informaţia trebuie furnizată la costul potrivit. *Mainframe-urile* cu reţeaua ierarhică şi bugetul de întreţinere/dezvoltare asociate au fost considerate, poate pe nedrept, ca punctul cu cel mai mare buget din cadrul departamentelor sistemelor informatice.

Este important de înţeles că „reducerea dimensiunii sistemelor informatice" şi „ajustarea optimă a dimensiunii aplicaţiilor" nu se referă numai la economisirea de bani. Apariţia unei noi arhitecturi distribuite pentru procesare şi livrare a sistemelor informatice a fost posibilă prin combinarea următoarelor evenimente:

- introducerea de staţii de lucru personale puternice, la preţuri rezonabile;
- apariţia unor sisteme de operare server multitasking, accesibile şi puternice;
- cereri în continuă schimbare a sistemelor informatice, rezultate din climatul economic din zilele noastre.

1.1. Avantaje

Sistemele informatice, construite în concordanţă cu această paradigmă a arhitecturii distribuite, sunt, deseori, menţionate ca sisteme informatice client/server deoarece toate îndatoririle lor sunt împărţite între computerele client şi server. Aceste sisteme informatice distribuite sau ajustate la dimensiunea optimă nu numai că economisesc bani, dar şi furnizează informaţii mai bune, într-un mod mai flexibil, permiţând luarea unor decizii economice mai rapide, în situaţii competitive.

Cele mai importante avantaje ale implementării cu succes a sistemelor client/server sunt redate în tabelul următor:

Avantaj	Exemplu/Explicaţie
Costuri reduse în comparaţie cu sistemele bazate pe mainframe-uri	Sunt posibile reduceri ale bugetului de 40%-80%, cu reduceri importante de personal.
Flexibilitate	Citată ca avantaj în aproape toate studiile de caz. Informaţia este mai accesibilă şi formatele de afişare sunt mai flexibile datorită arhitecturii client/server.
Suport/răspuns la schimbările din mediul economic	Mai presus de flexibilitate. Demonstrează potenţial pentru folosirea proactivă a sistemelor informatice în afaceri. Timpul semnificativ redus de dezvoltare de noi aplicaţii este factor util.

Acces sporit la informaţie	Informaţia stocată în LAN-uri distribuite este mai uşor accesibilă unui grup mai larg de utilizatori, decât informaţia stocată într-o reţea ierarhică bazată pe mainframe.
Informaţii mai rapide	Procesarea distribuită permite puterii combinate a PC-urilor client şi server să crească viteza de procesare până la de 10 ori mai mult decât la soluţiile bazate pe mainframe.
Informaţii mai bune	Informaţia „corectă". Produsă ca rezultat al combinării unei procesări mai rapide, al unei formatări mai flexibile şi al unui acces îmbunătăţit la o sferă mai largă de informaţii.
Arhitectura deschisă	Arhitectura tipică client/server, bazată pe PC, oferă un potenţial nelimitat pentru soluţii multivânzător, pentru oportunităţile sistemelor informatice. Sistemele bazate pe mainframe-uri furnizate de un singur producător blochează competiţia şi, în multe cazuri , soluţiile inovative.
Utilizatori împuterniciţi	Poate cel mai semnificativ, utilizatorii vor juca un rol mai important în proiectarea şi dezvoltarea sistemelor informatice pentru că în arhitectura client/server sunt implicate instrumente de dezvoltare a sistemelor, instrumente ce sunt bazate pe client şi uşor de folosit.

Deşi câteodată umbrit de beneficii mai concrete, cum ar fi îmbunătăţirea performanţelor sau scăderea costurilor, modelul flexibil, datorat naturii distribuite a arhitecturii client/server, rămâne cel mai important avantaj al sistemelor client/server.

1.2. Dezavantaje

Eforturile de optimizare a sistemelor informatice au şi dezavantaje şi potenţiale capcane. Unele dintre aceste efecte negative sunt specifice eforturilor de trecere la tehnologia *client/server*, în timp ce altele sunt adevărate pentru orice schimbare în proiectarea sau arhitectura sistemelor informatice.

Principalul atu invocat la implementarea arhitecturii *client/server* este reducerea costurilor. De multe ori, însă, costurile de tranziţie depăşesc aceste economii. Dacă noul sistem *client/server* înlocuieşte un sistem existent, ambele sisteme trebuie întreţinute şi susţinute o perioadă de timp. În plus, costul scrierii unui nou software sau al convertirii vechiului software ar trebui luat în calcul, la fel şi costul dezvoltării unui nou utilizator sau a unor instrumente software de conducere.

Alte costuri ascunse au apărut numai după ce s-a încheiat tranziţia către sistemul *client/server*. Datorită naturii distribuită, multivendor a sistemelor client/server acestea sunt, adesea, mai dificil şi mai costisitor de susţinut şi de întreţinut decât platforma unitară, furnizată de un singur vânzător, pe care o înlocuieşte. Problema este că nu există aproape deloc o similaritate sau consistenţă în proiectarea sistemelor de management, asigurate de diferiţii furnizori ai echipamentelor, nici chiar pentru elementele cu funcţiuni similare. Integrarea lor într-un sistem de management global este cu atât mai dificilă.

Soluţia ar fi dezvoltarea de standarde şi integrarea lor în sisteme de management pentru reţele de firmă. Cele mai populare sunt: HP Openview, IBM NetView şi Sun SunNet Manager.

Lipsa instrumentelor de conducere pentru arhitectura de calcul distribuită poate fi un impediment serios pentru dezvoltarea sistemelor *client/sever*. Astfel de instrumente trebuie, în mod automat şi transparent, să realizeze backup-uri, recuperare în urma dezastrelor şi probleme de securitate cum ar fi login pentru utilizatori şi accesul utilizatorilor la întreaga reţea distribuită. Utilizatorii aşteaptă de la platforma distribuită a arhitecturii *client/server* toată funcţionalitatea sistemelor de operare sofisticate ale *maniframe-urilor*.

Datorită noilor staţii personale de lucru, mai sofisticate şi mai puternice, accesibile acum utilizatorului final, pregătirea mai bună a acestuia este o realitate şi un cost în plus, pe care echipele de implementare a arhitecturii *client/server* trebuie să le ia în considerare. De asemenea, trebuie considerat că aceleaşi staţii puternice de lucru vor necesita un suport tehnic susţinut. Cu toate că este, în general, acceptat faptul că echipamentele utilizate în sistemele *client/server* sunt în mod semnificativ mai ieftine dacât *mainframe-urile* pe care le înlocuiesc, cercetări efectuate indică faptul că, atunci când costurile de suport, pregătire, conducere şi întreţinere sunt luate în calcul, nu se obţine nici o reducere semnificativă a costurilor prin tranziţia la sistemele *client/server*.

În tabelul de mai jos, sunt prezentate câteva dintre dezavantajele sistemelor client/server:

Dezavantaj	Exemplu/Explicație
Costuri ale tranziției	Adeseori pierdute din vedere, acestea sunt costuri „ascunse", apărute la trecerea către arhitecturile *client/server*
Cost ridicat pentru pregătire, suport, întreținere și conducere	Datorită în, primul rând, arhitecturii deschise, multivendor pregătirea, suportul, întreținerea și conducerea sunt mult mai complexe și, în consecință, mai scumpe.
Arhitectura multivendor	Prezentată și la avantaje, o arhitectură deschisă. susținută de tehnologii ce provin de la diverși furnizori. poate conduce la probleme de incompatibilitate.
Lipsa unor instrumente de conducere pentru mediul distribuit	Deși sunt făcute eforturi pentru a satisface această nevoie, instrumente sofisticate de conducere, capabile să conducă medii distribuite multivendor, sunt doar incipiente.
Lipsa standardelor	Această problemă apare, în special, atunci când două standarde în competiție sunt susținute de furnizori sau de consorții rivale.
Tehnologii nepregătite pentru aplicații critice	Cu toate că devin tot timpul mai sigure, server-ele și sistemele de operare ale server-lor încă nu garantează siguranța și puterea *mainframe-urilor*.
Lipsa instrumentelor software de conversie	Instrumente sofisticate care pot evita rescrierea în totalitate a aplicațiilor mainframe pot reduce enorm costurile de tranziție.

2. Trecerea de la mainframe la sisteme client/server

În momentul în care se preia o aplicație bazată pe mainframe și se dezvoltă din nou pentru o platformă distribuită client/server, trebuie avută în vedere natura proceselor care realizează această preluare. Printre cele mai utilizate terminologii folosite pentru descrierea acestor procese sunt:

- **Downsizing**

Implică faptul că aplicația bazată pe *mainframe* a fost dezvoltată din nou, pentru a rula pe o platformă mai mică. Această platformă mai mică poate sau nu să fie un sistem client/server. De asemenea, termenul de *downsizing* nu implică neapărat reproiectarea sau reingineria.

Exemplu: Codul sursă inițial poate fi doar recompilat și va rula nealterat pe noua platformă mai mică. Ca alternativă, aplicația poate fi rescrisă într-un nou limbaj și dezvoltată din nou. Altă posibilitate ar fi ca logica programului să fie reexaminată, reproiectată și reprogramată.

Downsizing-ul urmărește, în primul rând, să se folosească de avantajele computerelor din ce în ce mai puternice, disponibile la prețuri mai atrăgătoare. Aceasta nu implică neapărat ca procesarea să fie mai eficientă.

- **Rightsizing**

Implică faptul că aplicațiile trebuie să fie proiectate și dezvoltate pe o platformă sau un sistem de calcul, de un anumit tip și mărime, care să se apropie cel mai mult de mărimea optimă. Calculatorul sau platforma optimă presupune ca alegerea să se facă pe baza maximizării eficienței cu care rulează aplicațiile. Potrivirea unei aplicații particulare optime, cu o platformă de calcul particulară, urmărește eficiența nu numai din punct de vedere al performanței, dar și din perspectiva costului.

Datorită faptului că execuția eficientă a aplicației stă la baza optimizării, acest proces implică, de obicei, dezvoltarea nouă sau reproiectarea aplicațiilor, în contrast cu eforturile pentru o simplă reducere a dimensiunii.

- **Upsizing**

Poate fi considerat un caz particular de *rightsizing*. Când aplicațiilor le lipsește puterea de calcul pe platformele de calcul existente, ele pot fi reproiectate și dezvoltate din nou, pe platforme mai mari, mai puternice. De la caz la caz, se apreciază dacă aplicațiile au sau nu nevoie să fie regândite înainte de a fi „upsize"- ate.

- **Smartsizing**

Merge dincolo *de rightsizing*, implicând un alt nivel al studiului și al reingineriei. Prin *smartsizing* se înțelege mai mult decât o simplă reevaluare a aplicației, mergându-se un pas mai departe, prin reevaluare și studiul procesului economic care stă la baza aplicației. O dată ce procesul economic a fost în totalitate și obiectiv reevaluat, programele aplicației sunt reproiectate și rescrise. În final, noua aplicație este dezvoltată pe platforma optimă, care maximizează raportul performanță / cost.

3. Arhitecturi logice

Sistemele *client/server* folosesc avantajele puterii de calcul, disponibile acum pe PC, prin împărțirea sarcinii de livrare a informației către utilizatorii finali, între mai multe computere.

Arhitectura P-A-D (Presentation-Application-Data)

Cel mai uzual model logic al sistemelor clint/server analizează atent ceea ce implică livrarea informației de calitate către utilizatorii finali.

Această perspectivă funcțională este ilustrată în figura de mai jos :

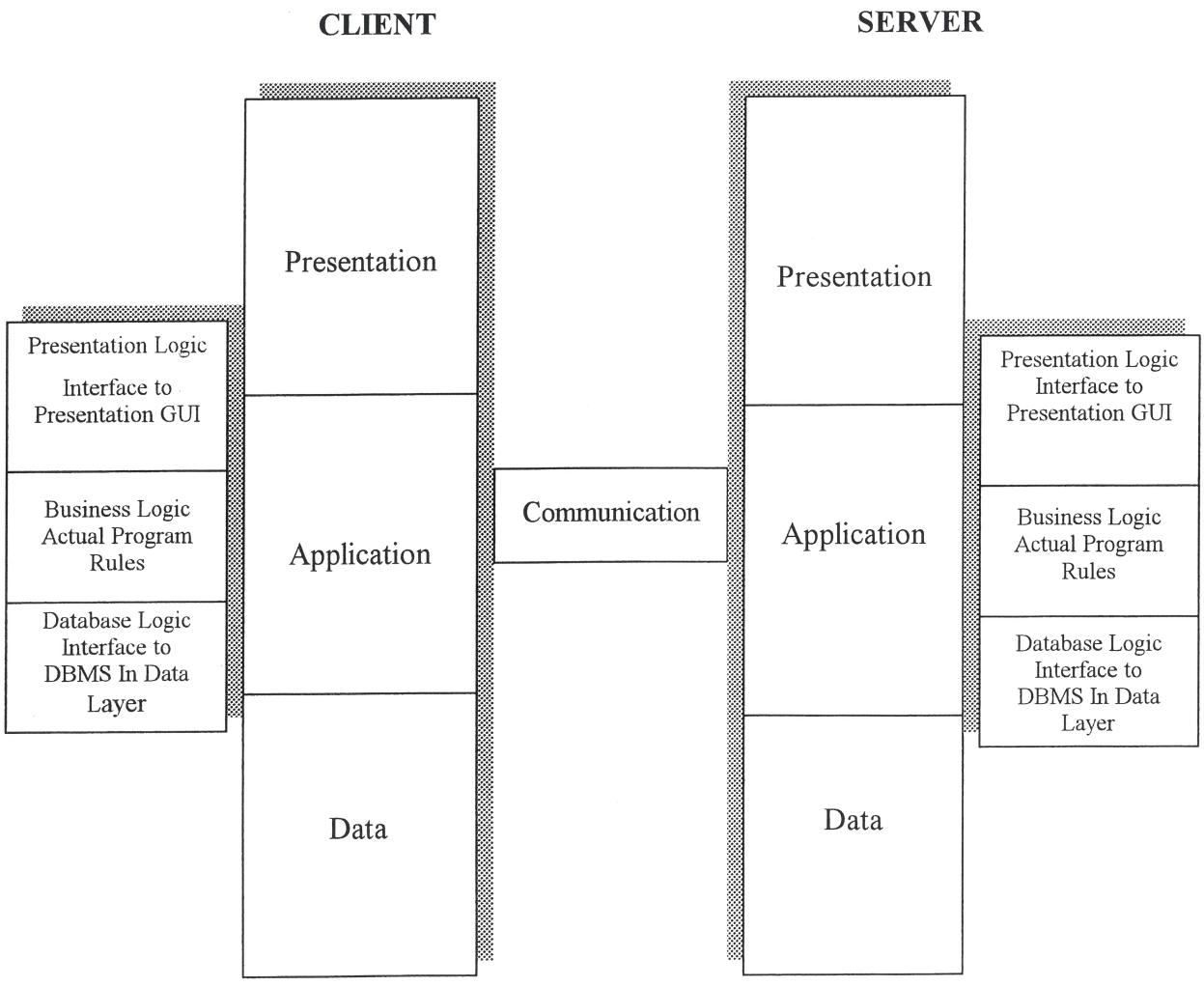

Figura 1. Modelul logic P-A-D (Prezentare-Aplicație-Date)

După cum se observă în figura 1, livrarea informației de calitate la utilizatorii finali depinde de interacțiunea a trei procese fundamentale:

1. prezentare (sau interfața utilizator);
2. aplicație (sau logica aplicației ori procesarea);
3. date (manipularea datelor).

Există modele logice ale sistemelor client/server care împart nivelul aplicație în următoarele trei subniveluri, evidențiind astfel interfețele acestui nivel cu nivelurile Prezentare şi Date:

1. Prezentare – partea aplicației responsabilă de interfața cu utilizatorul;
2. Tranzacție – regulile programului responsabil de controlul execuției programului şi întărirea regulilor de tranzacție;
3. Baza de date –partea aplicației responsabilă de interfața cu DBMS-ul.

Aceste trei funcțiuni importante (*Prezentare, Aplicație, Date*) se află şi la baza celor mai vechi sisteme informatice, bazate pe *mainframe-uri*. Noutatea adusă de sistemele *client/server* constă în posibilitatea scindării unora sau a tuturor acestor funcții pe mai multe platforme de calculatoare. Prin scindarea acestor funcții apare necesitatea comunicațiilor sofisticate între calculatoarele cooperante. Această comunicare este asigurată prin intermediul rețelei, de multe ori rețea locală (LAN). Importanța rețelei pentru calculul *client /server* este, deci, evident. Fără rețea, nu există calcul *client/server*.

4. Categorii de sisteme client/server

Prin combinarea elementelor Prezentare, Aplicație, Date ale arhitecturii P-A-D se obține o matrice a categoriilor posibile de sisteme client/server. În figura 2 de mai jos este prezentată această matrice, adăugându-se o nouă variabilă: platforma de execuție pentru aceste elemente.

Executing Platforms

	Client – Only	**Cooperative Client and Server**	**Server - Only**
Presentation	Client – Based Presentation	Distributed Presentation	Host – Based Presentation (Remote Presentation, Dumb Terminals)
Application	Client – Based Processing	Distributed Computing (Cooperative Computing Cooperative Processing)	Host - Based Processing
Data	Client –Based Management	Database Distribution (Distributed Data Management, Distributed Database)	Host – Based Data Management (Remote Data Management)

Figura 2. Categorii posibile de sisteme client/server

După cum se observă , fiecare funcţiune sau element poate fi executat în trei moduri:

1. funcţia este executată în totalitate da către client;
2. funcţia este asigurată prin cooperarea dintre client şi server;
3. funcţia este executată în totalitate de către server

Pentru ca un sistem să fie consideart *client/server*, clientul trebuie să manipuleze cel puţin primul element al arhitecturii logice P-A-D, numit ***Prezentare.*** Prezentarea realizată de client este, adesea, realizată prin intermediul interfeţelor grafice (GUI) ca Windows, OS/2 Presentation Manager, sisteme bazate pe X Windows precum Motif sau Open Look, sau Macintosh Desktop.

Calculul distribuit se referă la împărţirea unei aplicaţii în două sau mai multe părţi şi distribuirea şi procesarea acestor aplicaţii distribuite pe două sau mai multe calculatoare fie clienţi, fie server. Însă, divizarea optimă a aplicaţiei pentru mediul client/server nu se realizează uşor.

Principalele sarcini ale părţii client a programului (front-end) sunt:

- oferă interfaţa cu utilizatorul;
- formatează cereri pentru date sau pentru calcule pe care le transmite server-ului;
- formatează datele primite de la server pentru ieşirea către utilizator.

Principalele sarcini ale părţii server a programului (back-end) sunt:

- regăseşte şi stochează date la cerere;
- realizează calcule şi procesări ale aplicaţiilor;
- oferă securitatea necesară precum şi funcţii de management.

Două dintre cele mai importante atribute ale procesării distribuite sunt:

- *transparenţa* care reprezintă abilitatea sistemelor de procesare distribuită de a combina clienţi şi servere cu diferite sisteme de operare, sisteme de operare pentru reţea şi protocoale într-un sistem coerent fără a ţine cont de diferenţele dintre ele;
- *scalabilitatea* care reprezintă abilitatea sistemelor de procesare distribuită de a adăuga noi clienţi fără să reducă performanţa globală a sistemului.

4.1. Comunicarea între procese (IPC)

După împărţirea logică, cu succes, a aplicaţiei între client (front-end) şi unul sau mai multe server (back-end), trebuie ca aceste părţi separate ale aplicaţiei să se comporte ca şi cum ar fi un singur proces ce relurează pe un singur calculator. Este necesar un mecanism ce va permite părţilor distribuite ale aplicaţiei să comunice unele cu altele.

Protocoalele de comunicare interprocese sunt responsabile pentru trimiterea de mesaje şi comenzi între procese şi sincronizarea tranzacţiilor complicate, care necesită executarea a multiple procese pe calculatoare diverse.

Două dintre cele mai populare categorii de protocoale sunt:

- Remote Procedure Call (RPC);
- Massage passing.

4.2. Distribuirea bazei de date

Într-un mediu distribuit, datele trebuie să poată fi stocate în multiple locaţii fizice, pe diferite tipuri de servere, într-o manieră transparentă, utilizatorului final. Aceste servere multiple de baze de date trebuie să poată comunica unele cu altele chiar dacă ele nu rulează sub acelaşi DBMS. Faţă de DBMS-urile clasice, acestea trebuie într-un fel adaptate şi pentru medii distribuite.

DBMS-urile distribuite şi uneltele sale front-end asociate servesc ca o variabilă adiţională, împreună cu programele de aplicaţie, sistemele de operare şi sistemele de operare pentru reţele care trebuie să interopereze transparent pentru reţeaua de firmă.

4.3. Prelucrări de tranzacţii distribuite

Pentru sistemele informatice, tranzacţia este o secvenţă sau serii de paşi sau acţiuni predefinite, luate în cadrul unui program, pentru a înregistra corect tranzacţia economică. Înregistrarea adecvată a unei tranzacţii economice implică, de obicei, înregistrarea schimbărilor în mai multe fişiere sau baze de date. Procesarea tranzacţiilor necesită o monitorizare atentă pentru a se asigura că toate, şi nu numai unele dintre schimbările aferente respectivei tranzacţii economice, au fost terminate cu succes. Această monitorizare este făcută de un tip specializat de software cunoscut sub denumirea de *monitor TP* (*Transaction Posting*).

Caracteristicile acestuia sunt:

- atomicitatea (totul sau nimic);
- blocarea sincronizată, în timpul înregistrării tranzacţiei;
- toleranţa la erori.

Informaţiile economice trebuie să fie cât mai actuale. De multe ori, nu este suficient să se cunoască doar situaţia de la începutul zilei. Sistemele informatice care dau informaţii actualizate aproape instantaneu trebuie înregistrate imediat, şi nu în loturi nocturne, folosind sisteme *OLTP* (*on–line transaction processing*).

Între subnivelurile logice ale procesului *Aplicaţie* din cadrul arhitecturii P-A-D şi arhitectura fizică în care aceste procese subnivel sunt executate, se stabileşte o relaţie care este referită prin termenul *entitate (tier)*. Cele mai populare tipuri de arhitecturi sunt cele cu *două entităţi (two-tiered)* şi cu *trei entităţi (three-tiered)*.

În implementarea two-tiered, cele trei componente ale unei aplicaţii (Prezentare, Procesare şi Date) sunt divizate între două entităţi: codul aplicaţiei client şi baza de date server.

Arhitectura three-tiered aduce ca noutate separarea prezentării, procesării şi datelor în entităţi software distincte. Aceleaşi tipuri de unelte care în arhitectura precedentă erau utilizate pentru prezentare, acum sunt dedicate doar pentru prezentare. Când clientul solicită o cerere pentru acces la date sau o procesare a datelor, cererea se face la nivelul de mijloc, care este un server. Acest nivel poate efectua procesări de date sau cereri asemănătoare unui client la alte servere. Server-ele din nivelul mijlociu pot fi multi-treaded şi pot fi accesate de clienţi multiplii, asemeni unei aplicaţii separate.

În prezent, arhitectura two-tiered este mult mai uzuală şi lucrează bine pentru sistemele *client/server* departamentale. Instrumentele de dezvoltare şi back-end pentru bazele de date sunt disponibile pe scară largă pentru arhitectura two-tiered. La nivel de firmă, corporaţie, sunt preferate arhitecturile three-tired. Arhitectură mai nouă, apărută datorită limitărilor celei precedente, arhitectura three-tiered are dezavantajul unor relativ puţine instrumente de dezvoltare şi a unor puţine instalări cu succes, care să poată fi folosite ca modele.

5. Concluzii

Sistemele client/server necesită ca o gamă largă de tehnologii hard şi soft, provenite de la o multitudine de furnizori, să fie îmbinate armonios, prin intermediul reţelei de firmă, pentru a asigura utilizatorului, în final, în mod transparent, interoperabilitate. Această armonie implică legarea cu succes, prin protocoale mutual compatibile, a tuturor interfeţelor soft/soft ,soft/hard şi hard/hard.

Bibliografie

1. **NĂSTASE, F.**: Arhitectura Reţelelor de Calculatoare, Editura Economică, Bucureşti, 1999.
2. **DODESCU, G., A. VASILESCU, B. OANCEA**: Sisteme de Operare Unix şi Windows, Editura Economică, Bucureşti, 2003.
3. **HOARE, C.**: Communicating Sequential Processes, Englewood Cliffs NJ, Prentice Hall, 1985.

4. **HENNESY, J., D. PATTERSON**: Computer Architecture: A Quantitative Approach, San Mateo, CA, Morgan Kaufamnn, 1996.

5. **GRAG, V.**: Principles of Distributed Systems, Kluwer Academic Publishers, Boston, 1996.

6. **BEN-ARI, M.**: Principles of Concurrent and Distributed Programming, Englewood Cliffs, NJ, Prentice Hall, 1990.

7. **BRESON, A.** A Client/Server Architecture, New York, McGraw Hill, 1996.

8. **BACON, J.**: Concurrent Systems, Reading, MA, Addison-Wesley, 1998.

9. **AXFORD, T.**: Concurrent Programming: Fundamental Techniques for Real-Time and Parallel Software Design, New York, Wiley, 1988.