LABORATOR VIRTUAL MULTIDISCIPLINAR DESTINAT ÎNVĂȚĂMÂNTULUI TEHNIC UNIVERSITAR

stud. Mircea Nedelcu
conf. dr. ing. Daniela Saru

Universitatea Politehnica Bucureşti
mirceaned@k.ro, saru@aii.pub.ro

Rezumat: Lucrarea prezintă rezultatele obținute în activitatea de proiectare şi de implementare a unui laborator virtual, cu caracter multidisciplinar ,în cadrul Facultății de Automatică şi Calculatoare a Universității "Politehnica" Bucureşti. Sunt discutate conceptul de laborator virtual şi avantajele implementării acestuia, folosind o infrastructură informatică de tip client-server, orientată obiect. Este prezentată, ca studiu de caz, crearea unui laborator virtual multidisciplinar, format din echipamente cu particularități hardware/software eterogene, situate în zone geografice distincte. Sunt analizate perspectivele proiectului "Laborator Virtual" în contextul învățământului tehnic românesc.

Cuvinte cheie: laborator virtual, comandă la distanță, învățământ la distanță., arhitectură client-server, middleware, CORBA, programare distribuită, obiecte distribuite.

1. Introducere

Dezvoltarea din ultimii ani a tehnologiei informației nu putea să nu influențeze radical felul în care trăim şi gândim. Comunicațiile au devenit globale, iar informația a devenit un bun accesibil tuturor. Educația a fost unul dintre primele sisteme care au beneficiat de posibilitățile oferite de rețeaua Internet şi de progresele din domeniul tehnicii de calcul. Şi în învățământul românesc au început să fie folosite materiale didactice *on-line* şi teste grilă pentru evaluarea cunoştințelor, puse la dispoziția cursanților prin intermediul tehnologiilor World Wide Web. Această tendință are, fără îndoială, efecte pozitive asupra procesului instructiv-educativ, care trebuie, totuşi, nuanțate.

În contextul educațional modern, şcoala devine o instituție care trebuie să se ocupe de o comunitate numeroasă. Necesitățile învățământului actual țin de formarea continuă a indivizilor şi de dezvoltarea capacității acestora de a se adapta la sarcini multiple şi diverse. De aceea, procesul de învățare nu trebuie să se limiteze doar la transmiterea de cunoştințe, ci să formeze, în primul rând, aptitudini şi deprinderi practice. Produsele de învățare trebuie atent proiectate, pentru a nu îl transforma pe cel care învață într-un "consumator" de cunoştințe, ci într-o persoană practică şi creativă, ce are capacitatea de a formula întrebări şi de a găsi soluții.

O componentă esențială în cadrul unui *curriculum* din domeniul tehnic sau ştiințific trebuie să fie activitatea practică, în cadrul căreia se sistematizează cunoştințele teoretice dobândite şi se valorifică aceste cunoştințe sub formă de soluții pentru probleme reale. În învățământul universitar românesc, de cele mai multe ori, activitatea practică se realizează sub formă de ore de laborator în cadrul cărora se folosesc pachete software de simulare (de tipul instrumentație virtuală). Această variantă necesită costuri reduse, dar nu oferă un mediu de lucru "adevărat", cu probleme reale. Una dintre alternativele viabile este folosirea unui "Laborator Virtual", concept care va fi prezentat, pe scurt, în cele ce urmează.

2. Conceptul de "Laborator Virtual"

Termenul de "Laborator Virtual" este folosit aici cu sensul de laborator ale cărui componente sunt distribuite în cadrul unei rețele de comunicație şi permite interacțiunea dintre utilizatori şi instrumentația reală disponibilă. Principalele avantaje ale unei abordări de acest tip sunt [6]:

- suplinirea lipsei personalului şi a resurselor (acces 24 de ore din 24);
- oferirea de servicii tip învățământ la distanță pentru studenții care nu pot participă la activitățile de laborator;
- folosirea în comun de către cei implicați în procesul educativ (studenți, cercetători etc.) a unor instalații foarte scumpe, care în mod normal nu pot fi dedicate procesului educativ;
- posibilitatea schimburilor de laboratoare între universități;
- colaborarea cu întreprinderile ce activează în industria locală.

Este important de menționat faptul că utilizarea laboratorului virtual nu trebuie să substituie în totalitate structura clasică, ci să suplinească anumite limitări ale acesteia. De aceea, chiar universități prestigioase, cum ar fi Carnegie Mellon sau Massachusets Institute of Technology au adoptat, deja, în structura *curriculum*-ului lor metode specifice laboratoarelor la distanță.

Scenariul de desfăşurare a unei şedinţe de laborator poate fi următorul [9]:

- studentul face o cerere pentru un anumit experiment;
- se eliberează un nume de utilizator şi o parolă;
- cu numele de utilizator şi parola, solicitantul se conectează la server şi îşi face o programare, care va fi memorată într-o bază de date;
- la data stabilită, cursantul se conectează la maşina server folosind un *browser* Web sau un client executabil de sine stătător şi îşi realizează experimentul.

În proiectarea unei structuri pentru învăţământ la distanţă trebuie avute în vedere următoarele aspecte:

- portabilitatea arhitecturii pe diferite platforme hardware şi software (sisteme de operare etc.);
- considerente de modularitate a soluţiei (să poată fi extinsă uşor prin adăugare de noi componente).

Prin prisma cerinţelor enumerate anterior se observă că arhitectura client/server şi utilizarea reţelei Internet reprezintă infrastructura ideală a unui laborator virtual bine proiectat. Principalele componente ale laboratorului ar putea fi, de exemplu, (figura 1):

- o aplicaţie Server care controlează direct procesul tehnologic;
- una sau mai multe aplicaţii Client, care folosesc simultan echipamentele laboratorului virtual;
- o conexiune video între instalaţia popriu-zisă şi calculatorul care o accesează (asigurarea unui *feedback* corespunzător).

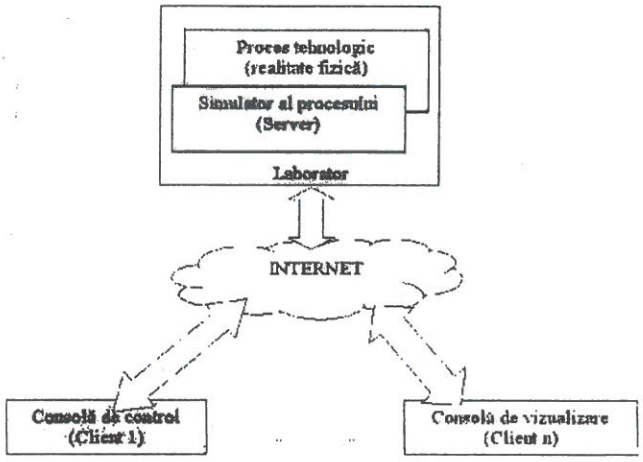

Figura 1. Infrastructura de comunicaţie a unui laborator virtual

Figura 1. Infrastructura de comunicaţie a unui laborator virtual

Modelul client-server avut în vedere oferă o arhitectură stratificată pe două niveluri: nivelul utilizator, care conţine interfaţa grafică, şi nivelul server, care conţine logica aplicaţiei. Între cele două niveluri se află un set de servicii transparent utilizatorului, numit *middleware*, care are rol de intermediar între sistemele software eterogene.

Pentru a ilustra rolul stratului *middleware*, se poate folosi următorul scenariu: un intermediar în domeniul vânzărilor *(broker)* comercializează obiecte. Dacă un comerciant trimite o cerere pentru un anumit obiect, intermediarul poate localiza rapid un producător care să îndeplinească cererea. Intermediarul poate comunica producătorului cerinţele exacte ale comerciantului. În plus, intermediarul poate face reclamă obiectelor în numele producătorului şi poate să transporte obiectele de la producător la comerciant.

Prin analogie cu scenariul anterior, se poate analiza următoarea problemă: există obiecte software localizate pe server, ce trebuie accesate de la nivelul clientului. Sarcina clientului este de a formula cereri, el nu este implicat în mecanismul de comunicare/activare a server-ului şi nici în procedura de stocare a datelor. Aceste sarcini sunt îndeplinite de către *middleware*.

Având în vedere multitudinea de sisteme de calcul şi sisteme de operare prezente în Internet, proiectarea unor produse *middleware* performante a impus o amplă activitate de standardizare la nivelul marilor consorţii software şi organizaţii nonprofit. Soluţiile cu cea mai mare pondere pe piaţă sunt oferite de Object Management Group (metodologia CORBA), Sun Microsystems (Enterprise Java Beans) şi Microsoft (tehnologia DCOM). În alegerea făcută pentru implementarea laboratorului la distanţă, s-a optat pentru standardul CORBA, deoarece oferă avantaje importante, care vor fi detaliate în cele ce urmează.

3. Prezentare succintă a tehnologiei CORBA

CORBA *(Common Object Request Broker Arhitecture)* poate fi descrisă pe scurt ca fiind o arhitectură client-server distribuită şi orientată obiect care oferă un mecanism standard de specificare a interfeţelor dintre componente (interfaţa este văzută ca un contract prin care componenta ce oferă interfaţa se angajează să satisfacă cererile venite din partea altor componente) [2].

Orientarea obiect a acestei tehnologii asigură avantaje importante, care ajută la atingerea scopului pe care ni l-am propus în cadrul proiectului de cercetare:

- oferă programării distribuite paradigmele programării orientate obiect: încapsulare, moştenire, polimorfism;
- separă interfaţa unui obiect de implementarea sa;
- înlesneşte construirea sistemelor complexe.

Posibilitatea utilizarii conceptelor orientate-obiect în mediu distribuit înseamnă, mai ales, că vor putea fi accesate obiecte aflate la distanţă în acelaşi fel în care sunt accesate obiectele locale.

CORBA separă interfaţa unui obiect de implementarea sa, deci, permite utilizarea unor aplicaţii mai vechi *(legacy applications)* în cadrul unui sistem nou, modern, prin "împachetarea" *(wrapping)* acestora într-un "înveliş" corespunzător. De exemplu, prin scrierea unei interfeţe se poate transforma codul existent al unei aplicaţii mai vechi, de comandă locală a instalaţiei de laborator, într-un obiect CORBA.

Avantajele programării orientate obiect în implementarea sistemelor complexe sunt evidente, fiind date chiar de conceptul de refolosire a codului şi de posibilitatea de modificare a implementării fără a modifica interfaţa.

CORBA oferă avantaje importante şi prin comparaţie cu alte tehnologii *middleware* [2]:

- permite independenţa faţă de limbajul folosit (de exemplu, un client implementat într-un limbaj de programare poate să comunice cu un server scris în alt limbaj de programare);
- clientul nu este obligat să cunoască detaliile de implementare de la nivelul server-ului;
- oferă independenţă faţă de sistemul de operare folosit: clientul nu este nevoit să cunoască sistemul de operare folosit de server, acesta putând fi implementat chiar fără ajutorul unui sistem de operare, cum este cazul sistemelor încapsulate (*embedded systems*);
- clientul ignoră straturile de transport şi de legătură de date (se alege în mod transparent tehnologia de reţea disponibilă - Ethernet, ATM, Token Ring).

Unul dintre elementele cheie ale arhitecturii CORBA este ORB *(Object Request Broker)*. Acesta are rolul unei magistrale de obiecte (figura 2)[10].

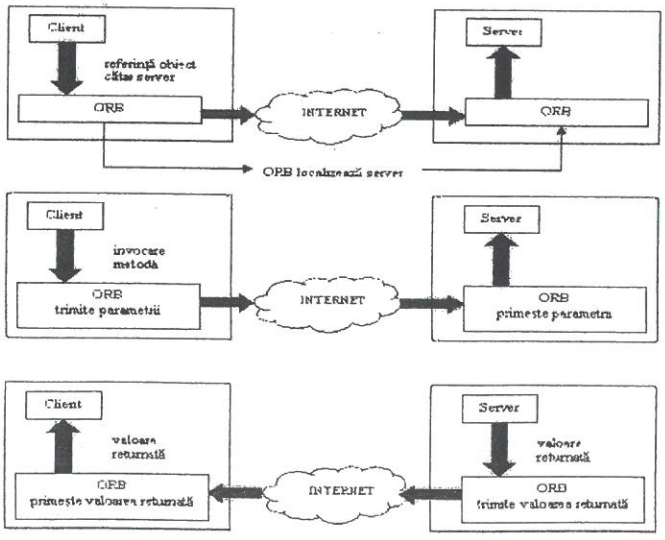

Figura 2. Rolul ORB în localizarea serverului şi transmiterea datelor

Figura 2. Rolul ORB în localizarea serverului şi transmiterea datelor

În esenţă, funcţionalitatea ORB poate fi descrisă astfel: furnizează aplicaţiei client o referinţă către obiectul server, transmite apelul de funcţie, convertind parametrii într-un format adecvat circulaţiei în reţeaua de comunicaţie, aşteaptă ca execuţia la nivelul server-ului să ia sfârşit şi apoi trimite către client valoarea returnată de apel. Întregul proces are loc

fără intervenția programatorului care nu face decât să apeleze o metodă ca și cum ea ar fi în spațiul de adresă local. Modul de transmitere în rețea a valorilor parametrilor metodei este standardizat prin IIOP *(Internet Inter-Orb Protocol)*, ceea ce permite interoperabilitatea între implementări CORBA provenite de la producători software diferiți și independența față de platforma hardware/software folosită.

Independența față de limbaj este asigurată de un alt element important al arhitecturii CORBA: IDL *(Interface Definition Language)*. După cum arată și numele, IDL este un limbaj pentru definirea interfețelor ce descriu funcționalitatea componentelor unei aplicații. Două aplicații (un client și un server) pot comunica doar dacă folosesc o interfață comună. Independența față de limbaj este realizată prin mecanismul de translatare a elementelor IDL în limbaj de programare (*language mapping*), ceea ce permite creatorilor de software să aleagă limbajele dorite pentru implementare: de exemplu, clientul poate fi dezvoltat în Java (pentru a permite funcționarea lui pe orice platformă care acceptă o mașină virtuală Java), iar server-ul poate fi scris în C++ (din considerente de viteză).

Pentru înțelegerea modului de lucru, se poate folosi un exemplu de generare a interfețelor pentru crearea unui sistem în care conlucrează (sunt integrate) componente software implementate în limbajul Java și limbajul C++ (figura 3) [2]. Creatorul aplicației va scrie o interfață în IDL, apoi va prelucra această interfață cu ajutorul unui compilator IDL. În urma compilării rezultă două tipuri de fișiere: *stub* și *skeleton*. Aceste fișiere se comportă ca un liant între interfețele IDL (independente de limbaj) și detaliile de implementare (specifice limbajului ales). Fișierele *stub* sunt folosite de către aplicația client, ele oferind o pseudoimplementare pentru metodele asociate obiectului server. Această pseudoimplementare apelează, de fapt, funcționalități ORB, într-un mod transparent pentru utilizator. Pe de altă parte, fișierele de tip *skeleton* oferă infrastructura pe baza căreia cel ce implementează aplicația va scrie ce anume se dorește a fi executat atunci când se face apelul de la distanță.

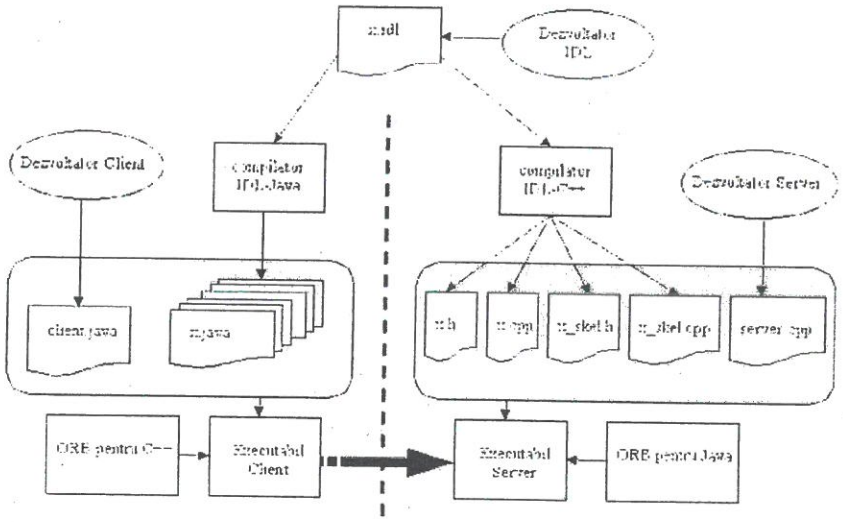

Figura 3. Compilarea IDL în cazul folosirii unor limbaje de programare diferite

Figura 3. Compilarea IDL în cazul folosirii unor limbaje de programare diferite

Câteva detalii legate de utilizarea acestor tipuri speciale de fișiere vor fi prezentate în secțiunea următoare, care ilustrează aspecte practice de implemetare a unui laborator virtual folosind tehnologia CORBA.

4. Studiu de caz: crearea unui laborator virtual multidisciplinar

Principalul obiectiv al proiectului nostru de cercetare a fost crearea unui laborator virtual multidisciplinar, destinat, în primul rând, instruirii practice a studenților facultății de Automatică și Calculatoare a Universității "Politehnica" București și a cursanților Centrului de Pregătire a Resurselor Umane (CPRU) din cadrul aceleiași universități. Printre disciplinele de studiu avute în vedere se numară: "Sisteme cu evenimente discrete" (anul III de studiu), "Sisteme de operare în timp real" (anul IV de studiu), "Sisteme CIM" (anul IV de studiu), "Standardizarea proiectării aplicațiilor informatice" (anul VI - studii aprofundate), "Sisteme cu arhitectură deschisă" (anul VI - studii aprofundate - CPRU). În acest stadiu de dezvoltare a proiectului, echipamentele folosite în cadrul laboratorului virtual creat aparțin celor două principale entități universitare beneficiare: facultatea de Automatică și Calculatoare și Centrul de Pregătire a Resurselor Umane (CPRU). Laboratorul virtual creat permite studenților să comande și să vizualizeze comportarea unei celule flexibile de fabricație tip DEGEM

2000, situate în sediul CPRU, să observe particularităţi specifice sistemelor cu evenimente discrete, să experimenteze diferite strategii şi mecanisme de conducere a sistemelor în timp real.

Celula de fabricaţie DEGEM 2000 (figura 4) este un produs al firmei DEGEM, cu rol educaţional. Ea este alcătuită din patru posturi de lucru, deservite de roboţi manipulatori:

- alimentare,
- magazie,
- conveior (bandă rulantă),
- prelucrare şi asamblare.

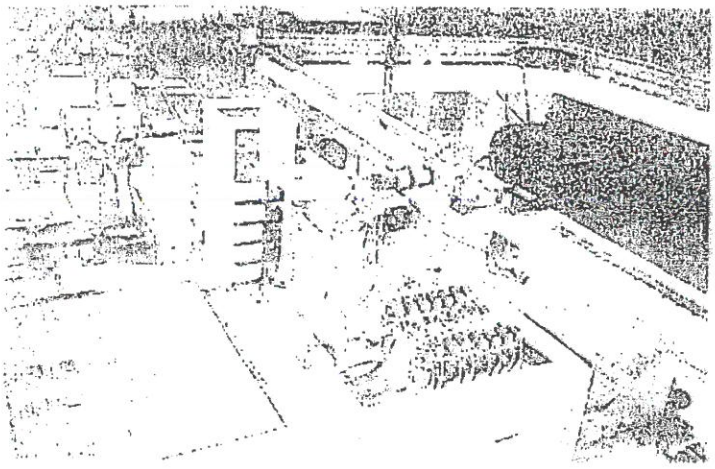

Figura 4. Celula de fabricaţie DEGEM 2000

Figura 4. Celula de fabricaţie DEGEM 2000

Conducerea celulei flexibile se realizează prin intermediul automatelor programabile, care au şi rolul de a asigura comunicaţia între diferitele module ale instalaţiei.

Ca exemplu, un scenariu de funcţionare a celulei poate fi descris astfel:

- unul dintre roboţii manipulatori ia o piesă de la staţia de alimentare şi o aşează pe un cărucior aflat pe banda rulantă (conveior);
- începe deplasarea benzii, dar ea se va încheia în dreptul staţiei de prelucrare, când apariţia căruciorului este detectată de către un senzor;
- piesa este luată de manipulator, este aşezată pe bancul de lucru, prelucrată şi pusă din nou pe banda rulantă, de unde este transportată spre magazie.

Iniţial, monitorizarea întregii celule flexibile de fabricaţie se realiza local, cu ajutorul unui sistem software, implementat în limbaj C, ce se executa sub sistem de operare QNX (sistem de timp real). Noul sistem de monitorizare, gândit astfel încât să ofere posibilitatea folosirii celulei în cadrul laboratorului virtual, permite coexistenţa unor module QNX, Windows, UNIX (diverse versiuni) şi LINUX, integrate prin tehnologia CORBA [11].

Fiecare post de lucru al celulei se caracterizează prin funcţii şi stări specifice, monitorizate în cadrul sistemului software complex. Pentru postul de lucru "Conveior" ales ca exemplu în acest studiu de caz, automatul programabil acceptă informaţii (date de intrare) de la senzori (sesizarea apăsării butonului de Eroare, detectarea prezenţei paleţilor la celelalte trei staţii etc.), comenzi date de utilizator (citirea seriei paletului aflat la o anumită staţie, eliberarea paletului, blocarea acestuia, activarea/dezactivarea conveiorului, activarea/dezactivarea alarmei, activarea/dezactivarea lămpilor de semnalizare optică ale celor trei staţii) şi poate memora stări de funcţionare (starea lămpilor celor trei staţii, starea conveiorului etc.).

Toate aceste particularităţi au fost luate în considerare în cadrul etapei de analiză [5] a sistemului informatic ce urma a fi proiectat, pe baza lor fiind formulate următoarele cerinţe:

- monitorizarea senzorilor şi a stărilor celulei;
- emiterea de comenzi către celulă;
- asigurarea persistenţei informaţiei primite de la senzori, pentru analize ulterioare şi tipărire de rapoarte;
- posibilitatea comutarii comandă automată - comandă manuală pentru un anumit post de lucru.

Etapa de proiectare a aplicației "Client" a avut în vedere asigurarea accesului la stările și intrările în automatul programabil prin definirea unei interfețe IDL corespunzătoare.

Etapa de implementare a laboratorului virtual a presupus alegerea uneia dintre următoarele două variante:

- realizarea aplicației client sub formă de *applet* Java, ce se execută în cadrul unei pagini Web, soluție ce ar fi asigurat o independență sporită față de platformă;
- crearea modulului client ca program executabil de sine stătător, folosind mediul Microsoft Visual Studio.

Deoarece într-o etapă anterioară a proiectului fuseseră deja create module de conducere pentru o parte a posturilor de lucru aferente celulei [11], module ce utilizau a doua variantă de implementare, am considerat necesar ca, în cadrul prezentei etape, să folosim aceeași metodologie. Folosirea tehnologiei CORBA asigură prin ea însăși o mai mare libertate de alegere a limbajelor de programare pentru eventuale module software, care să îmbogățească în viitor funcționalitatea laboratorului virtual proiectat. În plus, deoarece urmează ca în cadrul CPRU să fie realizată o modernizare a celulei DEGEM 2000 prin dotare cu echipamente care să ofere o alternativă la conducerea cu automate programabile AEG, am decis ca, pentru moment, să nu creăm componenta server destinată „Conveior"-ului, ci să proiectăm și să folosim un modul prototip, cu rol de simulare. Această soluție permite ca, în viitor, să se realizeze particularizări rapide pentru orice tip de echipament folosit, modificările fiind făcute numai la nivelul componentei server ce va apela funcții specifice *driver*-ului asociat echipamentului ales.

Pașii parcurși în cadrul etapei de implementare pot fi sistematizați astfel:

PAS 1: Definirea interfeței pentru obiectul condus (în cazul nostru, automatul programabil care controlează banda rulantă)

```
interface Automat{
//scrierea unei comenzi
  void ScrieComanda(în unsigned short usNrComanda);
//accesarea stărilor automatului programabil
  unsigned short CitesteStare(în unsigned short usNrStare);
  void ScrieStare(în unsigned short usNrStare,în unsigned short usValStare);
//accesarea intrărilor automatului programabil
  void ScrieIntrare(în unsigned short usNrIntrare,
                    în unsigned short usValIntrare);
  unsigned short CitesteIntrare(în unsigned short usNrIntrare);
//s-a modificat starea automatului față de citirea anterioară
  unsigned short CitesteModificare();
//citirea timpului local (al mașinii server) pentru păstrarea unui istoric al funcționării
  void CitesteDataTimp(out unsigned short usAn, out unsigned short usLuna,
                       out unsigned short usZi, out unsigned short usOra,
                       out unsigned short usMin, out unsigned short usSec);
//metoda pentru obținerea dreptului de accesare a instalației - verific o variabilă de stare și returnează "adevărat"
//dacă accesul este permis; este apelată doar de clientul "Consola Conveior", care are drept de scriere
  int InregistreazaAcces()
//metoda pentru eliberarea accesului la instalație; este apelată la terminarea execuției clientului "Consola
Conveior"
  void ElibereazaInstalatia()
};
```

PAS 2. Generarea fișierelor *stub* (Automat.h, Automat.cpp) și *skeleton* (Automat_skel.h, Automat_skel.cpp) prin compilarea interfeței IDL cu ajutorul compilatorului *eidl* (inclus în ORBacus/E, implementare CORBA a firmei Iona Technologies).

PAS 3: Scrierea aplicației client.

Obiectivul principal al acestei etape a fost crearea unei funcționalități specifice lucrului în timp real pentru aplicațiile de tip client, ce se conectează simultan la celulă. Acestea fac o citire periodică a stărilor automatului și împrospătează informația existentă pe ecranul utilizatorului. Varianta de implementare aleasă este folosirea firelor de execuție multiple *(multi-thread)*: fiecare client conține un *thread* care se ocupă de manipularea interfeței grafice și un *thread* care se ocupă de citirea periodică a modificărilor aparute în server. Datele ce caracterizează funcționarea celulei sunt păstrate într-o bază de date tip Microsoft Access, folosind tehnologia ODBC. Dintre acestea pot fi menționate informații despre: starea butoanelor consolei operator (buton de eroare, buton de start), stările senzorilor (disponibilitatea paleților la cele 3 stații), momentul de timp, data curentă,

intrarea modificată la momentul curent. Pentru ca baza de date să poată fi extinsă prin adăugarea unor tabele suplimentare, s-a prevăzut un câmp cu rol de cheie primară, care să permită realizarea de relaționări în cazul analizelor statistice complexe.

Metodologia de autentificare a potențialilor utilizatori se concretizează în verificarea unor informații introduse prin intermediul unei ferestre de dialog (figura 5). Orice aplicație ce dorește să inițieze conlucrarea cu aplicația server trebuie să cunoască adresa de Internet (IP) a sistemului de calcul care conduce celula, port-ul pe care așteaptă server-ul cererile clienților și o parolă de acces. Verificarea acestor informații are drept scop prevenirea accesului neautorizat la dispozitivele aflate la distanță.

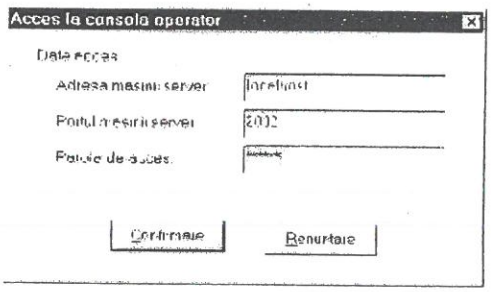

Figura 5. Fereastră de acces la consola operator

Figura 5. Fereastră de acces la consola operator

În cadrul actualei etape de realizare a proiectului de cercetare, au fost implementate două tipuri de aplicații client: o consolă operator pentru acces la distanță și un modul de evaluare a activității de laborator.

Funcționalitatea aplicației client cu rol de consolă operator a postului de lucru "Conveior" (figura 6) oferă posibilitatea emiterii de comenzi și a monitorizării stărilor de interes de către cel care accesează Laboratorul Virtual. Operarea se realizează în mod transparent, din punctul de vedere al clientului neexistând deosebiri între accesul local și accesul la distanță. Suplimentar, se oferă și funcționalitate de tip "Comandă automată".

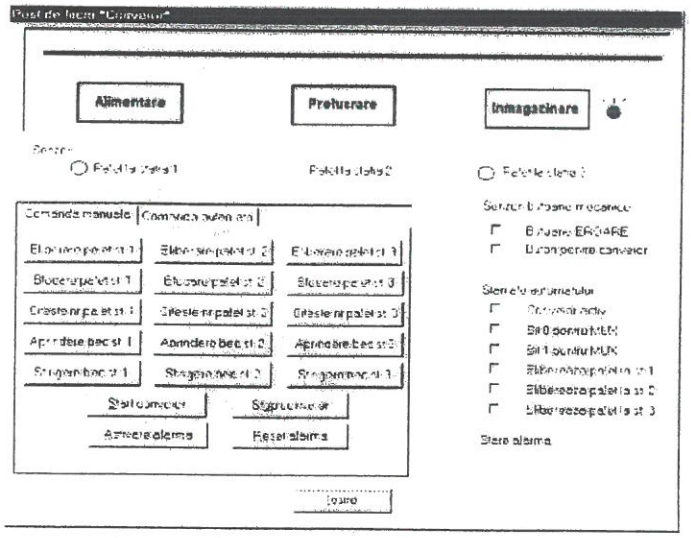

Figura 6. Consola operator a postului de lucru "Conveior"

Figura 6. Consola operator a postului de lucru "Conveior"

Celălalt tip de aplicație client (figura 7), destinat preponderent activității de evaluare a activității de laborator, poate fi folosit, de exemplu, de către cadrul didactic coordonator și oferă următoarele facilități:

- tratarea elegantă a excepțiilor generate de oprirea aplicației server și/sau configurarea greșită a aplicațiilor client;
- bară de unelte pentru acces rapid la informații;
- sistem de asistență utilizator cu opțiuni de căutare și index;
- "Print/Print Preview" pentru înregistrările din tabel, cu numerotarea paginilor și afișarea unui antet;
- analiza înregistrărilor corespunzătoare unei anumite date calendaristice, prin selectarea zilei dorite;

- poziționare rapidă pe anumite înregistrări din baza de date, folosind meniul contextual al *mouse*-ului.

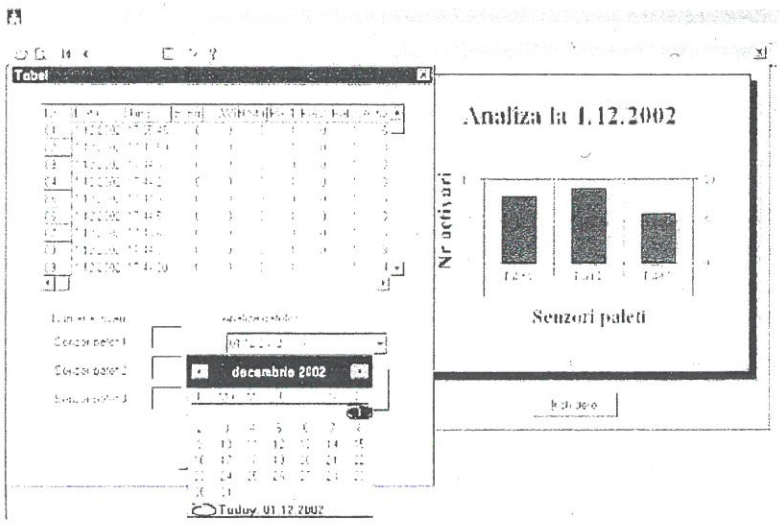

Figura 7. Consola de evaluare a activității de laborator

Figura 7. Consola de evaluare a activității de laborator

PAS 4: Implementarea funcționalității obiectului server.

Modulul server este cel care emite comenzile propriu-zise către automatul programabil. De aceea, este necesar ca el să apeleze funcțiile oferite de către producători în cadrul *driver*-elor, care să activeze/dezactiveze anumite intrări ale automatului. În modulul software de simulare, creat în cadrul activității noastre de cercetare, obiectul server a fost implementat în limbajul C++ ca aplicație de consolă (figura 8) care afișează pe ecran modificările survenite în starea automatului programabil, împreună cu "ştampila" de timp a acestora.

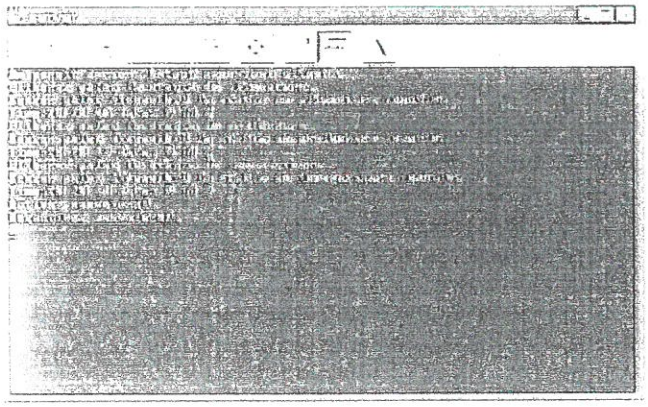

Figura 8. Modulul de simulare a obiectului Server

Figura 8. Modulul de simulare a obiectului Server

Modelul interacțiunilor dintre componentele sistemului de monitorizare este sintetizat în figura 9. Funcția de comunicație între componentele aplicației este asigurată prin folosirea tehnologiei CORBA. Comanda efectivă a instalației este realizată de către server, prin apelarea unor rutine specializate *(drivere)*, ce modifică stările echipamentelor dedicate (automate programabile sau alt tip de echipament). Serverul poate deservi simultan mai mulți clienți, dintre care unul singur poate iniția comenzi asupra instalației în timp ce restul pot urmări efectul obținut. Excluderea mutuală la nivelul resursei critice (instalația) se realizează prin apelul obligatoriu al metodei *ÎnregistreazăAcces()* de către aplicațiile client cu drept de scriere asupra stărilor automatului programabil şi prin eliberarea resursei la terminarea execuției acestora (metoda *Eliberează Instalatia()*). Clienții de tipul "Consolă evaluare" nu prezintă probleme de sincronizare a accesului la resurse datorită faptului că se limitează la citirea stărilor server-ului, fără a încerca să le modifice. Comunicația dintre server şi aplicatiile client se conformează modelului de "concurență reactivă": server-ul nu comunică simultan cu mai mulți clienți ci deservește cererile depuse de aceştia într-un şir ordonat.

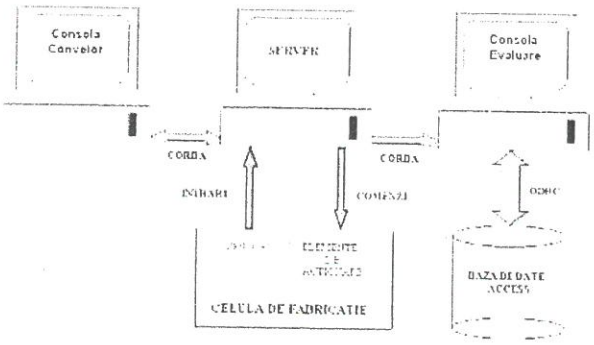

Figura 9. Modelul interacţiunilor din sistem

Figura 9. Modelul interacţiunilor din sistem

5. Concluzii

Construirea sistemului informatic complex cu rol de laborator virtual ne-a oferit posibilitatea utilizării unor metode şi tehnologii moderne de proiectare şi implementare a unei aplicaţii software distribuite eterogene cu o valoare practică deosebită. Conectând săli de laborator şi echipamente situate în zone geografice distincte, laboratorul virtual creat oferă procesului de instruire avantaje multiple, expuse (în esenţă) în prima parte a acestei lucrări. Funcţionalitatea laboratorului cuprinde, între altele:

- monitorizarea locală şi de la distanţă a unei celule flexibile de fabricaţie;
- urmărirea activităţii studenţilor, realizarea de statistici, evaluări şi rapoarte;
- vizualizarea codului sursă, aferent modulelor software ce compun sistemul de conducere;
- utilizarea unor pachete software de simulare a comportării proceselor;
- utilizarea unor pachete software de modelare.

Toate acestea conferă laboratorului virtual un caracter multidisciplinar, flexibil (amănunte, în secţiunea a 4a). Datorită utilizării tehnologiei CORBA ca infrastructură informatică, funcţionarea laboratorului poate fi uşor extinsă prin adăugarea unor noi echipamente şi module software.

Ca obiectiv de cercetare viitoare, ne-am propus realizarea unor componente software cu rol de client sub formă de aplicaţii Java, care să permită accesarea laboratorului de pe orice sistem de calcul, capabil să găzduiască o maşină virtuală Java. Aceasta va permite cursanţilor atât folosirea sistemelor de calcul "clasice", cât şi a dispozitivelor moderne, din domeniul comunicaţiilor mobile. Experinţa dobândită va putea fi utilizată şi pentru crearea unor aplicaţii informatice complexe, eterogene, destinate industriei sau serviciilor.

Bibliografie

1. **BONIVENTO, C., et al:** A Web-Based Laboratory for Control Engineering Education, documentaţie Internet, http://www-lar.deis.unibo.it/woda/spider/b60c.htm
2. ***: CORBA/ C++ Programming with ORBacus, Iona Tehnologies Inc., 2001.
3. ***: CORBA documentation, http://www.corba.org , documentaţie Internet.
4. ***: Introduction to CORBA, Magelang Institute, documentaţie Internet, http://developer.java.sun.com/developer/onlineTraining/corba/corba.html
5. **IONIŢĂ, A., D. SARU:** Dezvoltarea orientată pe obiecte a programelor medii şi mari în limbajul C++, MatrixRom, Bucureşti, România, 1998.
6. **LICKS, V., et al:** Building Virtual Laboratories - A Web Based Experience, documentaţie Internet, http://www.eece.unm.edu/~vlicks/papers/icece2000final.pdf
7. ***: Object Management Group, http://www.omg.org , documentiaţie Internet.
8. **PATIL, A., et. al:** Comparison of Middleware Tehnologies - CORBA, RMI & COM/DCOM, documentaţie Internet, http://www.egr.msu.edu/~patilabh/Final_Report.pdf

9. **RŐHRIG, C., A. JOCHHEIM:** The Virtual Lab for Controlling Real Experiments via Internet, documentație Internet, http://prt.fernuni-hagen.de/virtlab/cacsd99/mp3_4/mp3_4.html

10. **ROSENBERG, J.:** Teach Yourself CORBA în 14 Days, SAMS Publishing, New York, USA, 1998.

11. **SARU, D., A. PETCU, D. MEREZEANU**: Server QNX/CORBA de supervizare a unui modul de intreprindere virtuală. În: Revista Român de Informatică și Automatică, vol. 12, nr.3, 2002, pp. 52-65.