

LABVirt – basic components

Asist. Codrin NISIOIU, asist. Cătălin SILVESTRU
Catedra de Informatică Economică, A.S.E București

This article presents some aspects about a project integrated in a national research and development project "Information society" (INFOSOC) financed by Romanian Government. A iLab called LABVirt is available for use by Universities, college, high-school.. The goal of this paper is to present the main components of LABVirt, Romanian solution for remote lab access.

Keywords: iLab, remote lab access, client-server architecture, client applet

Introducere

Sistemul educațional prin Internet reprezintă un concept utilizat pe plan național în activitatea instituțiilor de învățământ de toate gradele. Alinierea la ultimele tendințe pe plan mondial în domeniul organizării și funcționării rețelelor de calculatoare după modelele de tip intranet/internet (adică revenirea de la modelul informatic în care datele și prelucrările erau distribuite în mod "egal" în toată rețeaua, la arhitecturi cu servere puternice, servite de soluții software și hardware specializate, care să ofere informații unor "clienți" cât mai diverși, fie aceștia calculatoare fixe sau mobile), creează premisele integrării cât mai apropiate în societatea informatică a viitorului [1]. Articolul va realiza descrierile arhitecturilor aplicației client, respectiv server.

Descrierea aplicației client

În figura 1 este prezentată comportarea dinamică a APPLLET-ului client printr-o succesiune de stări ale acesteia. Fiecare stare corespunde unui nou comportament funcțional al aplicației.

Applet-ul inițializează aplicația client (Starea 1) cu o prezentare generală a laboratorului virtual, prin care i se explică utilizatorului modul de folosire al aplicației, modul în care se navighează în interiorul laboratorului virtual, ceea ce trebuie să facă pentru a duce la bun sfârșit lucrarea de laborator virtual și permite utilizatorului alegerea unei categorii de laboratoare din cele disponibile.

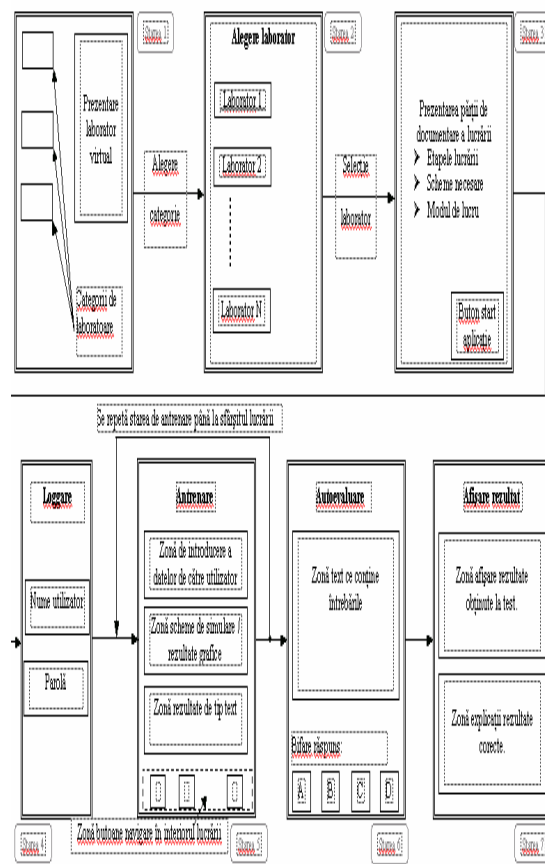


Fig.1. Descriere aplicație client

După ce clientul selectează o categorie, aplicația client trece în Starea 2 în care utilizatorului i se prezintă o listă de lucrări din care trebuie să aleagă laboratorul pe care trebuie să-l efectueze. Utilizatorul poate să aleagă orice lucrare dorește, chiar dacă a mai parcurs-o într-o dată anterioară, sau începe o lucrare nouă. De asemenea, clientul poate efectua lucrările de laborator în ordinea dorită, nu există restricții de acest fel.

În urma acestei selecții, se trece în Starea 3, unde se presupune că utilizatorul este deja

familiarizat din Starea 1 cu modul de efectuare a unei lucrări de laborator virtuale. În acest moment, utilizatorului îi sunt prezentate numai informațiile legate de evoluția lucrării de laborator, cum ar fi etapele lucrării, schemele necesare și modul de lucru. După ce utilizatorul a parcurs toate aceste informații, el poate să acceseze partea de simulare a lucrării, aplicația client trecând în Starea 4.

În această stare utilizatorului i se cere să introducă numele utilizator și parola, necesare pentru a se autentifica în sistem. Dacă clientul nu dispune de un cont și o parolă, el nu poate trece la partea de realizare a lucrării de laborator.

După validarea autentificării utilizatorului, se trece în Starea 5, care reprezintă faza propriu-zisă de antrenare (training), adică efectuează lucrarea de laborator virtuală. Această stare se repetă până ce studentul parcurge toți pașii lucrării. El nu poate parcurge laboratorul decât în ordinea proiectată, anumite butoane fiind inactive pentru a nu permite sărirea peste anumite operații sau navigare de la un modul la altul al lucrării fără realizarea celui precedent.

În cadrul antrenării utilizatorul va interacționa cu simulatorul prin intermediul zonelor definite pentru fiecare modul component al lucrării de laborator în parte prin care studentul introduce date și primește rezultate sau imagini. Conținutul și forma datelor ce sunt trimise sunt verificate înainte de transmiterea acestora de către aplicația client pentru a se evita generarea unor erori care ar îngreuna comunicația client-server.

Când simularea lucrării ia sfârșit, aplicația client trece în Starea 6 în care utilizatorul își verifică cunoștințele acumulate în laboratorul pe care l-a parcurs. Această fază poartă numele de autoevaluare.

Aplicația client se termină cu Starea 7, unde utilizatorului i se afișează rezultatul obținut, întrebările la care a greșit precum și explicațiile rezultatelor corecte. Eventual, el poate primi și unele recomandări asupra căror module să revină pentru o mai bună înțelegere a materialului de studiat.

În varianta pilot a sistemului LABVirt stările 6 și 7 nu au fost implementate, urmand ca

acestea să fie implementate într-o versiune ulterioară, am preferat inversarea stărilor 3 cu 4 pentru o mai bună securitatea a sistemului și am ales ca încărcarea imaginilor să se facă într-o zonă separată de cea de introducere a datelor și de obținere a rezultatelor.

Descrierea aplicației server

Caracteristicile platformei gazda

Aplicația server este proiectată să ruleze pe sisteme de operare compatibile Windows NT. Alegerea platformei Windows are ca argument faptul că există un număr foarte mare de aplicații de simulare cu suport pentru această platformă. Versiunile compatibile NT sunt de preferat datorită stabilității și a suportului îmbunătățit pt. comunicația în rețea.

Separarea sesiunilor de lucru

Aplicația trebuie să suporte un număr de sesiuni de lucru client-simulator ce se vor executa în “paralel”, drept pentru care devine necesară o separare a datelor și a codului pentru fiecare client în parte. Astfel, pentru orice conexiune, serverul va lansa pe un fir de execuție codul ce asigură legătura client-simulator. Datele vor fi alocate pe o stivă proprie, integritatea lor fiind asigurată de mecanismele de protecție ale sistemului de operare.

Pentru cazul în care vor exista spații de lucru comune, (mediul de lucru al simulatorului dacă acesta este unic pentru mai multe sesiuni), fiecare fir de execuție va modifica datele în așa fel încât acestea să nu intre în conflict cu datele celorlalte fire.

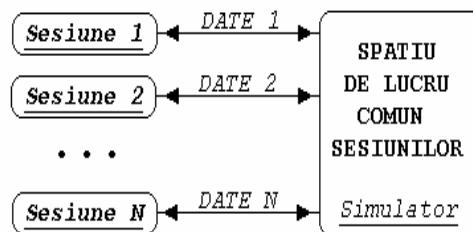


Fig.2. Sesiuni de lucru cu simulatorul

Având la bază ideile susmenționate, se poate proiecta un singur modul, a cărui instanțe vor reprezenta sesiunile și care se vor executa în “paralel”, cu mențiunea că, dacă circumstanțele o cer, datele ce vor fi schimbate cu

simulatorul vor fi independente de la o sesiune la alta.

Arhitectura aplicatiei server. Cooperarea cu modulul simulator

Înca de la început se disting două parti functionale ale aplicatiei:

- un modul de initializare a structurilor aplicatiei si lansarea unei interfete necesara supravegerii si gestionarii activitatii pe server
- un modul ce va realiza conexiunea client – server – simulator. Acest modul se va lansa pe un fir de executie paralel cu cel precedent, la fiecare cerere de conectare venita din parte unei aplicatii client.

Aplicatia debuteaza cu un set de initializari realizate de firul principal. Sunt inițializate:

- un fir de executie pe care va rula socketul de ascultare a rețelei pe portul 1500, port ales ad-hoc de catre echipa executive, din care am facut parte. Structura firului de server o voi prezenta ulterior.

- un alt fir de executie ce va monitoriza firele de executie corespunzatoare sesiunilor de lucru si le va realoca în cazul în care acestea au ajuns la un final.

- interfața cu mediul de simulare

- o interfata cu utilizatorul permitindu-i accesul la vizualizarea si gestiunea starii curente a serverului, urmind ca la comanda de iesire sa opreasca toate firele în executie, sa dezaloc datele alocate dinamic si sa termine aplicatia.

Firul de server

Firul de server are rolul de a asculta rețeaua cu rolul depistării cererilor de conectare venite din partea aplicatiilor client, cit si de a genera un fir de executie tip client-server-simulator pentru fiecare conexiune acceptata. La primirea unei cereri de conectare, socketul firului server creează un nou socket, specific conexiunii în curs de acceptare, care este transmis firului client numai daca el îndeplineste condițiile de securitate prescrise de administratorul sistemului . Altfel, socketul copie este distrus si se reintră în bucla de asteptare a unei noi conexiuni.

Firul de server ruleaza conform schemei din figura 3:

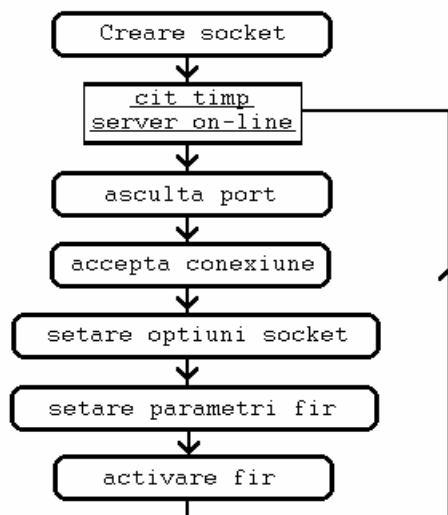


Fig.3. Execuție fir

Modulul alocator de fire

Datorita faptului ca un asemenea sistem nu poate sustine simultan un numar foarte mare de conexiuni client-server-simulator, s-a preferat o alocare statica a datelor, mai eficienta si mai putin pretentioasa decit o alocare dinamica. Pentru a gestiona optim conexiunile ce sunt pornite respectiv oprite s-a recomandat realizarea unui simplu alocator pentru firele de executie, care urmareste conexiunile eliberate si le dezaloca, respectiv aloca noile conexiuni în spatiile precedent eliberate si doar în cazul în care acestea nu exista este alocata o noua pozitie.

Modulul alocator este compus din două parti:

- un fir ce ruleaza în paralel cu sistemul si elibereaza pozitiile firelor de executie terminate;

- o functie ce intoarce prima pozitie de la începutul zonei alocate în care se poate porni un nou fir client-server-simulator. Această funcție este folosită de catre firul de server atunci cind aloca firele de client-server-simulator.

Modulul client-server-simulator

Acest modul va fi activat la fiecare cerere de conexiune acceptată de către server, având rolul de interfață între aplicația client și aplicația de simulare. El are de realizat trei activități:

1. comunicația cu aplicația client, ce se va desfășura pe baza protocolului de comunicație client-server
 2. translatarea datelor între formatele simulator și aplicația client
 3. transmiterea lor către simulator, realizarea simulării și extragerea datelor rezultate.
- Secvența de initializare a datelor firului
Se realizează următoarea secvență de inițializări (figura 4):

- Într-un prim pas este generat un identificator unic pentru firul în execuție, identificator pe baza căruia vor fi modificate și identificate datele ce vor fi trimise spațiului comun al simulatorului, cu scopul precizat anterior, de a nu intra în conflict cu datele scrise de alte fire. Identificatorul este construit pe baza informațiilor legate de conexiunea TCP/IP și este unic.
- Se alocă spații de lucru interne firului
- Se alocă variabile și spații de lucru pentru interfața cu simulatorul
- Se alocă spații de lucru pentru fișierele de configurare.

Secvența de autentificare a utilizatorului

În protocolul de comunicație, după ce serverul trimite sirul de identificare, aplicația client trimite și ea un sir de identificare, unic pentru un utilizator. Acest șir este interpretat și cautat într-un fisier în care vor fi înregistrați utilizatorii autorizați ai aplicației. În cazul în care respectivul utilizator există, serverul raspunde cu "OK" și se trece la secvența următoare. Altfel sesiunea va fi închisă și firul terminat.

Secvența de creare și comunicare a structurilor de date

Odata ce autentificarea s-a încheiat cu succes aplicația client selectează o lucrare de laborator prin comanda din protocol "LAB:xx", aplicația server realizează următoarele activități:

- încarca fișierele de configurare,
- își crează structurile de date funcție de conținutul lor
- le transmite și aplicației client.

De la acest punct, comunicația cu sistemul la distanță va avea loc sincron, avînd la bază aceste structuri. Interpretarea fișierelor de

configurare se va face cu ajutorul unor clase predestinate acestui scop.

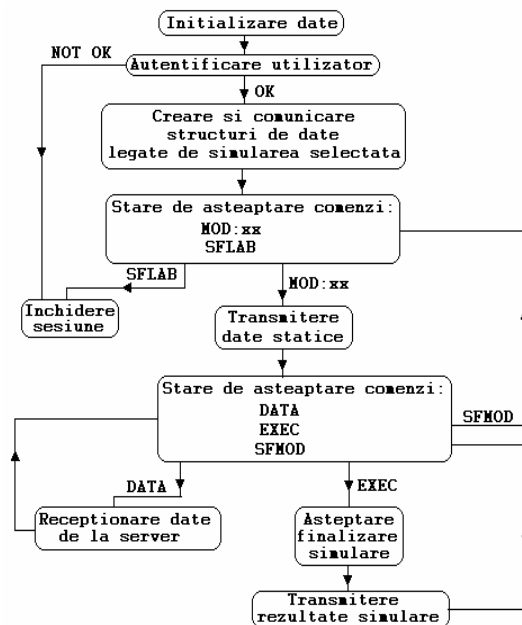


Fig.4. Starile firului client-server-simulator

Comenzile MOD:xx, SFLAB

Începând cu acest pas comunicația se va desfășura sincron între client și server, în sensul în care fiecare va parcurge structura proprie de date creată la pasul anterior și va realiza transmisia de date în concordantă cu aceasta. Comanda "MOD:xx" selectează modulul "xx" din structura lucrării de laborator, modul ce va conține datele unei secvențe complete de simulare. Lucrarea de laborator poate fi alcătuită dintr-un număr suficient de mare de asemenea module, ce vor fi parcurse consecutiv, iesirile simulărilor putînd reprezenta intrari pentru simulări imediat următoare.

O alta comanda care poate fi trimisă de catre client la acest pas este comanda "SFLAB", care permite o încheierea sesiunii, în urma ei ambele sisteme încheind conexiunea.

Secvența încheiere sesiune

La acest pas se ajunge în momentul în care serverul a primit comanda "SFLAB", următoarea acțiune pe care o întreprinde fiind evident închiderea conexiunii, dezalocarea memoriei și iesirea din firul de execuție.

Secvența de transmitere a instanțelor obiectelor

După ce serverul a primit comanda "MOD:xx", el transmite instanțele obiectelor statice descrise în fișierul de configurare .SCF către client, în ordinea în care acestea apar în fișierul de configurare (serverul este sincron cu aplicația client din acest punct de vedere, deoarece ele procesează același fișier de configurare). Datele trimise la acest pas vor fi atât textuale cât și grafice, ele urmând a fi afișate utilizatorului, în cadrul aplicației client.

Comenzile DATA, EXEC, SFMOD

Datele parcurg traseul applet-server-simulator, în ambele sensuri, având aceleași valori dar reprezentări diferite (una la client și alta la simulator). De aceea este nevoie de un "translator" de format (figura 5). Tot aici, rezultă și necesitatea fișierelor de configurare, așa cum sunt descrise în secțiunea următoare. Pentru utilizarea fișierelor de configurare se utilizează un "interpretor" caracteristic fiecărui tip de fișier, care generează în memorie structuri de date. Structurile de date generate fac obiectul comunicății client-server-simulator (în ambele sensuri) ce este gestionată prin comenzile următoare:

Comanda "DATA" precede sirul instanțelor obiectelor de intrare pentru simulare venite de la client. Serverul, după ce recepționează această listă, convertește datele la formatul cunoscut simulatorului, și se întoarce în bucla precedentă, așteptând comanda de începere a simulării "EXEC".

Comanda "EXEC" este executată imediat după ce au fost primite datele de la client. În urma recepționării ei, serverul da comandă de început simulării, și intră într-o buclă de așteptare, ieșirea fiind condiționată de terminarea simulării. Cât timp acesta nu a ajuns la sfârșit, serverul are grija să transmită aplicației client un semnal de prezentă la fiecare 30 de secunde. Când simularea s-a terminat serverul convertește rezultatele în format interpretabil de către aplicația client, și le transmite acesteia tot în ordinea în care apar în fișierul de configurare.

Comanda "SFMOD" marchează sfârșitul unui modul, obligând serverul să se decupleze de la modulul curent, și să revină în prima buclă de așteptare.

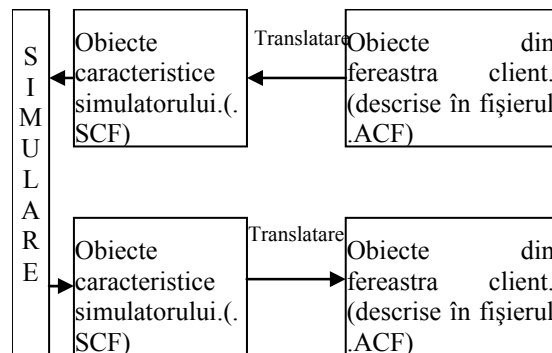


Fig. 5. Traseul client-server-simulator pentru un obiect

Fișiere de configurare: „.SCF”, „.ACF”, „.RES”

Fiecare lucrare de laborator va fi caracterizată prin următoarele trei tipuri de fișiere de configurare:

- „.ACF” (Applet Configuration File) fișierul ce conține configurația aplicației client; Acest fișier este necesar configurării appletului pentru o anumită lucrare. Acest fișier este structurat pe mai multe module corespunzătoare fiecărei etape de simulare în parte. Un asemenea modul conține obiecte prin intermediul cărora se va interacționa cu utilizatorul. În definiția obiectelor se va avea în vedere un nume unic pentru fiecare obiect, coordonatele și dimensiunile în fereastra utilizator, tipul de dată pe care îl va vizualiza (text/imagine), precum și tipul de acces al utilizatorului la ea (citire/scriere). Un exemplu de obiect este descris mai jos:

```
NUME: TIP_DATE: TIP_TR: X0: Y0:
DIM_X: DIM_Y
```

```
tip_date:T-text tip_tr:D-initializat
```

```
B-binari I-intrare
```

```
E-iesire
```

- „.SCF” (Server Configuration File) fișierul ce conține configurația aplicației server. Acest fișier are scopul de a realiza traducerea obiectelor din „.SCF” într-un format accesibil simulatorului și invers. Fiecărui modul în parte îi este atașat un nume de fișier ce va conține codul interpretabil al simulării.

- „.RES” (Resource File) fișierul ce conține resursele statice ale aplicației: texte și imagini. Acestea vor exista pentru fiecare obiect descris în fișierul „.SCF” ca fiind de tipul „D” (inițializat), și vor fi afișate în fereastra

aplicației client, pentru fiecare modul în parte.

Fișierele ce conțin codul interpretabil al simulării vor avea structura unei funcții, în sensul că va avea un număr de date de intrare, un cod interpretabil, și un număr de date de ieșire.

Comunicația aplicație server–simulator utilizat

În general, pentru a fi apt să se integreze aplicației LABVirt, un simulator trebuie să aibă două proprietăți:

- să dispună de posibilitatea de a fi multiuser (mai multe sesiuni pot fi deschise simultan)
- să dispună de un mecanism de comunicare care să permită aplicației server să transfere date în mod eficient și relativ comod și să comande etapele de simulare necesare.

Concluzii

În acest articol noi am încercat să arătăm tehnologia folosită și modalitatea de exploa-

tare pe care noi o considerăm optimă pentru dezvoltarea sistemului de învățământ virtual și mai ales a accesului la distanță a laboratoarelor. Mai sunt numeroase posibilități de dezvoltare a produsului software pe care noi le prezenta în materialele ce vor urma.

Bibliografie

- [1] Nisoiu C.(2003) Realizarea aplicației client a sistemului LABVirt-proiect de diplomă îndrumat de Prof.univ.dr.ing. Minzu V.
- [2] Silvestru C.(2003) Biblioteca digitală pentru informatica economică în cadrul conferinței naționale “ Universitatea virtuală din România “ – Academia de Studii Economice București
- [3] Silvestru C.(2005) E-learning în cadrul conferinței internaționale de informatica economică “ – Academia de Studii Economice București.