

MDL – Modul de dezvoltare a lecțiilor asistate de calculator

Daniel Safta

Universitatea Tehnică din Cluj-Napoca,
Facultatea de Automatică și Calculatoare
Str. G. Barițiu nr. 26-28
davonkeep@yahoo.com

Dorian Gorgan

Universitatea Tehnică din Cluj-Napoca,
Facultatea de Automatică și Calculatoare
Str. G. Barițiu nr. 26-28
dorian.gorgan@cs.utcluj.ro

REZUMAT

În acest articol vom prezenta o nouă abordare în didactica informaticii - modelarea vizuală a algoritmilor și evaluarea bazată pe forme metaforice - aplicată în nucleul unui sistem de învățământ virtual, un modul de dezvoltare a lecțiilor asistate de calculator (MDL). Am evidențiat structura și caracteristicile procesului didactic implementat în sistemul propus, utilizatori și rolurile lor, metodele didactice aplicate, soluții de evaluare și studii de caz pe un model de lecție. Am expus stadiul actual în domeniul temei alese, evidențind avantajele soluției expuse, dar și extinderi posibile.

Cuvinte cheie

Instruire asistată de calculator, modelare vizuală, proces didactic, gândire algoritmică, limbaj de programare.

Clasificare ACM

K. Computing Milieux, K.3 COMPUTERS AND EDUCATION, K.3.1 Computer Uses in Education, *Computer-assisted instruction (CAI)*.

INTRODUCERE

La ora actuală există o puternică preocupare în domeniul didacticii informaticii spre introducerea utilizatorilor în știința calculatoarelor de la o vârstă cât mai fragedă. În acest sens, sunt binecunoscute eforturile Massachusetts Institute of Technology, care prin Lifelong Kindergarten [1] elaborează și întrețin proiecte educaționale îndreptate spre copii școlari și preșcolari. Motto-ul lor "Sowing the seeds for a more creative society" indică importanța dezvoltării înclinațiilor copiilor spre programare încă de la clasele primare. Trebuie realizată, pe lângă alfabetizarea propriu-zisă, o așa numită alfabetizare în utilizarea calculatorului. În acest scop se utilizează programarea vizuală și programarea prin exemple, implicarea copiilor în proiecte educative, cum sunt Scratch sau Intel Computer Clubhouse Network.

Scratch [2] se adresează copiilor cu vârste cuprinse între 6-16 și încurajează trei direcții: imaginează, programează, împarte cu ceilalți. Iată cum un limbaj de programare este mascat într-un context educativ-distractiv, un mediu în care utilizatorii creează în mod interactiv și împart cu restul comunității online propriile povestiri, jocuri, muzica preferată, artă etc. La ora actuală există mii de proiecte active pe site-ul Scratch iar seriozitatea proiectului este întărită de susținători precum National Science Foundation, Microsoft, Intel Foundation, Nokia, Omega și consorțiul de cercetare de la MIT Media Lab.

Alte medii de promovare a informaticii în rândurile tinerilor sunt așa numitele Computer Clubhouses precum Intel Computer Clubhouse Network [3], Lego

Mindstorms [4], PicoCrickets [5] etc. În aceste comunități online utilizatorii împart propriile idei și experiențe din domeniul programării vizuale elementare aplicate pe domenii practice cum ar fi rețelele de comunicații, roboți lego programabili și chiar artă (muzică, sculptură, dans, ateliere de bijuterie etc).

Pornind de la premisa că sunt deja familiarizați cu utilizarea calculatoarelor și posedă minime aptitudini de programare, tinerii liceeni își pot dezvolta gândirea și stilul de programare cu ajutorul unor sisteme bazate pe programarea vizuală precum Visual Basic Game Programming for Teens [6], Macromedia Authorware [7].

Cu toate acestea, familiarizarea tinerilor cu mediul informatic nu implică faptul că aceștia vor ști să dezvolte un algoritm sau că stăpânesc un limbaj de programare. Între statutul de utilizator și cel de programator există o barieră foarte înaltă [8]. Există numeroase tentative de a ajuta utilizatorul în drumul său însă majoritatea întâlnesc aceeași piedică, stabilirea clară în gândirea umană a conexiunii între modelarea vizuală a soluției și descrierea într-un limbaj de programare [9]. În special utilizatorii foarte tineri, învățați cu acel mod introductiv de "programare distractivă" au deficiențe în adaptarea la un nou mod de abordare a algoritmilor, la abstractizarea noțiunilor modelate vizual și transpunerea în cuvinte. Este nevoie de un anumit nivel de inteligență și maturitate a gândirii individului pentru a putea face acest salt.

Motivație

În didactica informaticii, în interacțiunea dintre utilizator și limbajul de programare, în special în cazul începătorilor, apare o mare deficiență – modul de abordare. Utilizatorul începător nu are gândirea suficient structurată ca să elaboreze un algoritm constrâns de regulile unui pseudocod sau chiar ale unui limbaj de programare, totodată are deficiențe în descrierea traseului logic din rezolvarea unei probleme. Rezultă două mari necesități simultane: nevoia de a-și dezvolta gândirea logică, capabilă de a abstractiza și a transpune ideile într-un algoritm și nevoia de a stăpâni foarte bine rigourile unui limbaj de programare.

Abordarea actuală la nivelul didacticii informaticii, în special la noi în țară, este aceea că utilizatorul trebuie să-și formeze gândirea astfel încât "să gândească precum calculatorul", ceea ce este o mare greșală. În fond, algoritmi sunt realizați după gândirea umană și de aceea vom construi întreg sistemul descris mai departe pe ideea că: utilizatorul scrie codul sursă prin acțiuni și nu prin instrucțiuni!

MDL – MODUL DE DEZVOLTARE A LECȚIILOR ASISTATE DE CALCULATOR

Modulul de dezvoltare a lecțiilor asistate de calculator este nucleul unui întreg sistem de învățământ virtual aflat în curs de dezvoltare, sistem accesibil atât local cât și la distanță, și care oferă o alternativă de abordare a informaticii, în special a studiului algoritmilor și limbajelor de programare. MDL furnizează lecții pe baza unui nou concept în didactica informaticii, axat pe modul de abordare a utilizatorului, de interacțiunea sa cu mediul de programare printr-o interfață orientată spre modelarea vizuală și interactivă a algoritmului, și nu spre simpla editare de cod sursă, cea din urma realizându-se automat.

Noutatea constă în strategia didactică folosită în dezvoltarea pachetului de lecții, în modul de evaluare și de interpretare a rezultatelor: se va urmări forma finală a modelului descris și nu pașii intermediari. Astfel:

- elevul nu este îngrădit în alegerea unei anumite soluții, ci este liber să creeze propria cale de rezolvare a problemei, să modeleze vizual algoritmul după propriile idei;
- nu se pune accentul pe diagrama stărilor parcurse, ci pe rezultatul obținut;
- abia după validarea soluției se va trece la analiza pașilor efectuați;
- dacă răspunsul e corect, se reține traseul acțiunilor descrise ca o nouă metodă în rezolvarea problemei;
- dacă răspunsul nu coincide cu forma metaforică așteptată, sistemul urmărește firul desfășurării evenimentelor și indică prima abatere întâlnită de la traseele cunoscute.

Acest concept stă la baza dezvoltării unui pachet de lecții cu aplicare practică în studiul algoritmilor și a tehnicilor de programare, particularizat pe nivele de vârstă, capacitate de înțelegere și nu în ultimul rând pe baza cunoștințelor anterioare.

Procesul de învățământ

Combinând avantajele programării vizuale cu principiile programării structurate, se evidențiază următoarea alternativă în interacțiunea utilizator – algoritmi: utilizatorul nu este forțat, în prima etapă, spre a folosi un limbaj de programare, ci spre a modela vizual o soluție posibilă, eliminând aparent orice legătură cu editarea de cod sursă. În acest mod cursantul nu se mai împiedică în particularitățile limbajului folosit ci își distribuie întreaga atenție pe componenta logică în determinarea soluției problemei, axându-se doar pe dezvoltarea algoritmului cerut.

Prin urmare, în cadrul lecțiilor din aplicația noastră, utilizatorul folosește - inițial fără a conștientiza - elemente care sunt greu abordabile după modelul de predare clasic, modelează algoritmi după logica proprie, în mod liber, neîngrădit de rigorile unui limbaj de programare, dar în același timp, prin acțiunile sale, “redactează” cod sursă executabil. El poate apoi, prin adnotări grafice, să scrie codul propriu, corespunzător fiecărei acțiuni și să-l supună unei evaluări automate din cadrul MDL.

În selectarea și generarea lecțiilor se ține cont nu doar de nivelul de cunoștințe ci și de nivelul de înțelegere al fiecărui utilizator. Lecțiile pot fi particularizate și în funcție de categoria de vârstă.

Din punctul de vedere al începătorilor, procesul de învățământ are două componente (Figura 1):

- spațiul de modelare, în care va primi cunoștințele teoretice necesare și va modela soluții pentru problemele date spre rezolvare;
- interpretorul de comenzi.

Spațiul de modelare este “sala de clasă” în care elevul își va desfășura activitatea, scena în care el utilizează tehnici vizuale și interactive pentru a construi forma metaforică a algoritmului cerut. După această etapă, în funcție de șabloanele predefinite de către profesor, interpretorul identifică relațiile dintre acțiunile descrise și instrucțiunile corespunzătoare. Interpretorul transformă apoi acțiunile în instrucțiuni pe baza cărora se va scrie codul sursă.

Utilizatorii avansați au posibilitatea să interacționeze direct cu interpretorul, descriind acțiuni prin pseudocod, sărind astfel etapa de modelare vizuală a soluției și apropiindu-se de modul de lucru dintr-un limbaj de programare (Figura 2).

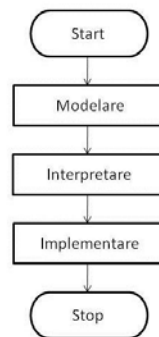


Figura 1. Diagrama procesului de învățământ din punctul de vedere al cursantului începător

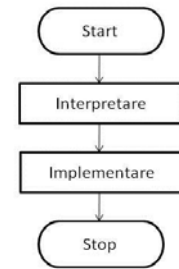


Figura 2. Diagrama procesului de învățământ din punctul de vedere al cursantului avansat

Roluri

În vederea realizării scopului propus, sistemul va interacționa cu trei actori, rolurile în MDL fiind administrator, cadru didactic (profesor, dascăl) și cursant (student, elev) (Figura 3).

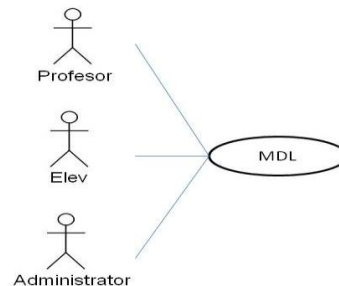


Figura 3. Roluri în MDL

Administratorul asigură integritatea resurselor și supraveghează procesul didactic din punct de vedere tehnic, modul de autentificare și nivelul de acces al

utilizatorilor la nucleul aplicației, întreține bazele de date etc. Administratorul aplicației este cel ce adaugă unelte noi la sugestia cadrelor didactice, creează noi spații de modelare și tehnologii de interacțiune între utilizator și sistem.

Cadrul didactic (profesor sau dascăl) nu dispăre din procesul de învățământ, rolul acestuia fiind cel mai semnificativ. El dispune de documentele școlare pe baza cărora va funcționa întreaga aplicație, introduce lecții și urmărește sincronizarea lor cu programa școlară și planificările calendaristice actuale.

Profesorul utilizează uneltele MDL cu care va gestiona întreg procesul instructiv-educativ: alcătuiește setul de lecții în conformitate cu documentele școlare curente, stabilește noțiunile teoretice și aplicațiile practice care vor fi abordate în cadrul fiecărei lecții în parte.

În generarea problemelor, dascălul creează obiecte, specifică acțiunile posibile asociate, descrie forme metaforice și identifică legătura dintre acțiunile utilizatorilor și instrucțiuni. Astfel, în realizarea contextului aplicativ al lecției, dascălul creează forme metaforice, de la simplu la complex, reprezentate printr-o gamă largă de obiecte predefinite sau create cu ajutorul uneltelor puse la dispoziția sa de către MDL (Figura 4).

Profesorul definește de asemenea datele de test utilizate în validarea soluției și șabloanele privind:

- relaționarea între pseudoinstrucțiuni și evenimentele descrise de elev;
- specificarea structurilor de date corespunzătoare metaforelor utilizate;
- constantele din forma metaforică ce vor fi înlocuite cu variabile.

Pseudocodul rezultat este convertit în cod sursă tot pe baza unor șabloane predefinite de către dascăl.

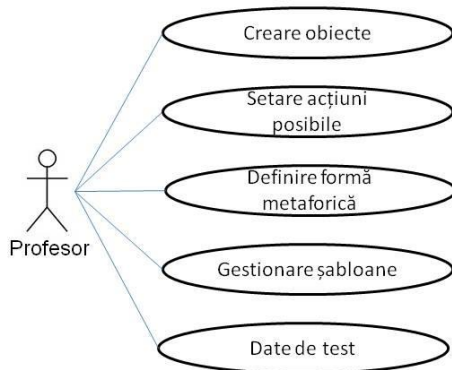


Figura 4. Rolul profesorului

Utilizatorul țintă, cursantul (elev sau student), participă la cursuri corespunzător nivelului de instruire și traiectoriei școlare selectate în prealabil la înregistrarea în sistemul virtual de învățământ.

În cadrul unei lecții, cursantul studiază noțiunile teoretice corespunzătoare și rezolvă probleme, fie interacționând prin modelare vizuală, fie - în cazul celor mai avansați - prin descrierea soluției în pseudocod. În primul caz,

elevul vizualizează o scenă în care are la dispoziție o serie de obiecte predefinite de către dascăl, caracterizate prin atribute specifice și acțiuni permise, cu care interacționează și pe care le utilizează în vederea modelării unei soluții în rezolvarea problemei (Figura 5).

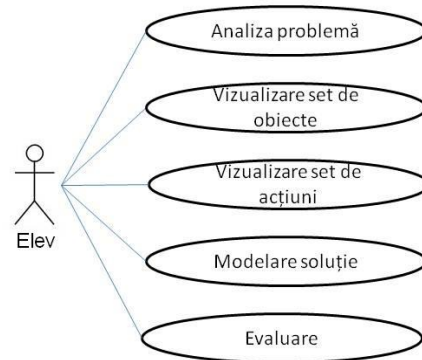


Figura 5. Rolul elevului

Caracteristicile procesului didactic:

- Ușurința în descrierea algoritmului

Prin modelarea vizuală a soluției, gândirea începătorilor nu este îngrădită de lipsa de experiență în utilizarea unui limbaj de programare. Pe lângă această tehnică de interacțiune elev - mediu deja cunoscută și aplicată în familiarizarea începătorilor cu domeniul informaticii, strategia didactică propusă se remarcă prin procesul de evaluare și instruirea bazată pe analiza feedback-ului.

- Libertatea în alegerea soluției

Elevul nu este obligat să utilizeze o anumită metodă de rezolvare predefinită, nu există un șablon care să impună succesiunea stărilor cuplului obiect - acțiune. Dacă în final, forma metaforică descrisă de el corespunde cu cea așteptată, atunci obține nota maximă.

Deși pot exista numeroase trasee alternative în modelarea unei soluții posibile, există o singură formă metaforică finală, astfel încât procesul de evaluare este infailibil. Indiferent de drumul ales, destinația e aceeași. Fie, de exemplu, problema clasică de sortare crescătoare a elementelor unui vector unidimensional. Se pot aplica numeroase metode de sortare, dar datele de intrare și de ieșire rămân aceleași: vectorul dat respectiv vectorul sortat. Indiferent de algoritmul implementat, cât timp nu se urmărește eficiența - ceea ce în cazul începătorilor nu are sens - ci doar corectitudinea soluției, procesul de evaluare nu va eșua.

- Tratarea erorilor de judecată

Deoarece prin modelarea vizuală a algoritmilor am eliminat posibilitatea apariției erorilor de sintaxă, răspunsurile eronate se datorează exclusiv erorilor de judecată. Pentru identificarea lor, se caută în setul de soluții cea mai asemănătoare din punct de vedere al efectului acțiunilor descrise și se face o comparație, pas cu pas, între stările descrise de elev și corespondentul lor în soluție. Se indică astfel căi alternative în traseul logic urmat în modelarea algoritmului cerut.

- Identificarea soluțiilor false

De regulă metaforele utilizate de către dascăl descriu un caz particular al unei probleme generale. Prin interacțiunea sa cu spațiul de modelare a soluțiilor, elevul poate obține un rezultat corect, o formă metaforică identică cu cea predefinită, printr-o serie de acțiuni ce descriu un caz particular și al căror efect nu poate fi generalizat.

De exemplu, sortarea în ordine crescătoare a elementelor unui vector se poate descrie prin următoarele metafore: se dă un număr constant de butoaie, iar fiecare conține o anumită cantitate de pulbere (setul de obiecte). Butoaiile pot fi interschimbate între ele oricare două câte două (setul de acțiuni). Presupunem că în urma modelării realizate de către elev se obține o aliniere a butoaielor astfel încât orice butoi din șir conține cel puțin la fel de multă pulbere ca vecinul din stânga și cel mult la fel de multă pulbere ca vecinul din dreapta. Această soluție poate fi validă, acțiunile elevului descriind o metodă de sortare general valabilă sau una falsă, aranjarea butoaielor realizându-se aleator, iar descrierea procesului de sortare fiind imposibil de generalizat. În acest caz se va proceda la ghidarea elevului spre o soluție corectă utilizând programarea prin exemple.

Identificarea acestor situații are loc în momentul în care se face trecerea la cod executabil când, în urma testării algoritmului pe baza setului de date de intrare, nu se obțin datele de ieșire așteptate în toate cazurile.

- Îmbogățirea bazei de date

Fiecare descriere validată a algoritmului ce nu există în biblioteca sistemului va fi adăugată la setul de soluții corespunzător problemei date. Setul de soluții se folosește în procesul de feedback, proces în care elevul este ghidat, în funcție de caz, spre cel mai apropiat răspuns corect sau spre răspunsul optim.

- Eficiența și optimizarea soluțiilor

În cazul în care elevul obține o soluție validă, sistemul afișează eficiența algoritmului corespunzător raportată la cea maximă și, unde e cazul, realizează o comparație între soluția elevului și soluția optimă. Astfel elevul este îndrumat spre urmărirea performanței în programare.

Lecția

Didactica informaticii se orientează spre rezolvarea de probleme, utilizând prin componenta vizual-interactivă metode activ-participative, insistând pe etapa de analiză a problemei. Lecția are două componente:

- teoretică în care are loc transmiterea de noi cunoștințe;
- practică, bazată pe rezolvarea de probleme.

Lecția este generată în funcție de traseul școlar al cursantului, de cunoștințele și noțiunile dobândite anterior și nu în ultimul rând de programa analitică aflată în vigoare. În desfășurarea lecțiilor se urmăresc o serie de activități de învățare [10] precum:

- utilizarea instrumentelor informatice în modelarea unor activități cotidiene;

- obținerea, în funcție de scopul propus, a unor prelucrări complexe prin combinarea unor operații elementare (pași);
- descrierea unui algoritm prin forme metaforice cât mai naturale;
- descrierea în detaliu a etapelor rezolvării problemelor din punctul de vedere al modelării algoritmilor;
- compararea diversilor algoritmi utilizați în rezolvarea unei probleme pentru a sublinia eficiența, avantajele și dezavantajele fiecărei soluții în parte;
- modelarea unor probleme întâlnite în viața de zi cu zi sau în studiul altor discipline (aplicații interdisciplinare);
- familiarizarea cu un limbaj de programare prin vizualizarea corespondenței acțiunilor întreprinse cu instrucțiuni, pașii unui algoritm și structurile de control specifice;
- proiectarea, modelarea și implementarea unui algoritm;
- utilizarea feedback-ului în depanarea programelor;
- studierea comportamentului programelor în funcție de diferite date de intrare.

În cadrul transmiterii de cunoștințe noi, se folosește îndeosebi programarea prin exemple sau demonstrația. Se expun conceptele fundamentale și se exemplifică pe un număr redus de obiecte. Apoi, prin generalizare, se trasează forma abstractă a algoritmului.

Evaluarea se realizează în etapa a doua a lecției, cea de rezolvare a problemelor, în fiecare din momentele sale:

- descrierea formei metaforice;
- descrierea procesului;
- descrierea pseudocodului;
- codul sursă final;
- rezultatul final, executabil.

În cazul începătorilor ce aplică în rezolvarea problemei exclusiv modelarea vizuală a soluției, evaluarea răspunsului se va axa pe compararea sa cu o formă metaforică predefinită de dascăl ce constituie fie soluția unică, fie o soluție posibilă. Procesul de evaluare include analiza acțiunilor utilizatorului și a rezultatului returnat în comparație cu cel așteptat.

Procesul didactic

Modulul de dezvoltare a lecțiilor asistate de calculator funcționează ca un element de legătură între principalii actori ai sistemului de învățământ, dascălul și cursantul, și furnizează contextul educațional necesar pentru transmiterea respectiv însușirea principiilor programării structurate.

În rezolvarea problemelor, în special în cazul începătorilor, se pune accentul pe modul de gândire al elevului. Se urmărește formarea unei abordări algoritmice, dezvoltarea capacității de analiză a unei probleme date, astfel încât să poată extrage din context datele de intrare și de ieșire, să compună și mai ales să modeleze o soluție după propriile idei, descriind procesul printr-o formă

metaforică, fără a se preocupa inițial cu redactarea de cod într-un limbaj de programare. Acțiunile întreprinse vor descrie instrucțiuni de pseudocod ce vor fi apoi transcrise, pe baza unor șabloane predefinite de către profesor, în cod sursă executabil.

Dupa însușirea noțiunilor teoretice, trecând la partea practică a lecției, elevul studiază problema ce urmează a fi rezolvată, vizualizează setul de obiecte disponibile și construiește prin acțiunile sale o formă metaforică. Sistemul alcătuiește procesul descris de acțiunile utilizatorului și compară răspunsul obținut cu soluțiile existente în baza de date.

În cazul în care răspunsul este corect iar traseul specificat de cursant în modelarea soluției nu există în baza de date, se va reține soluția dată, contribuind la creșterea complexității analizatorului de soluții. Asociind fiecare acțiune cu instrucțiuni corespunzătoare, procesul descris de utilizator este apoi transpus în pseudocod și mai departe, corelat cu șabloanele de cod sursă existente rezultă codul sursă executabil.

Dacă soluția finală a utilizatorului nu este corectă, atunci se indică erorile logice apărute și se afișează indicii pentru obținerea modelului așteptat.

Feedback-ul este un element esențial în sistemul de învățământ. Pentru fiecare cursant se crează și se actualizează după fiecare lecție un set de caracteristici evidențiate în timpul procesului didactic precum capacitatea de asimilare și aplicare în practică a noilor cunoștințe, viteza de lucru în modelarea soluției, tipuri de erori întâlnite și frecvența lor.

STUDIU DE CAZ

Plan de lecție

Unitatea de învățare: Descrierea algoritmilor.

Conținuturi: Structura alternativă (de decizie).

Competențe specifice: Analiza problemei. Identificarea datelor de intrare / ieșire, stabilirea pașilor de rezolvare a

problemei. Modelarea și reprezentarea algoritmilor. Principiile programării structurate în elaborarea algoritmilor.

Activități de învățare: Modul de execuție al instrucțiunii alternative. Exemple de utilizare din viața cotidiană. Urmărirea modului de execuție a instrucțiunii pas cu pas. Aplicații: determinarea minimului și maximului a două numere, verificarea parității unui număr, rezolvarea ecuațiilor de gradul I și II, divizibilitate etc.

Lecția propriu-zisă

Considerații teoretice: definirea și descrierea structurii alternative.

Aplicația practică: selectarea maximului a două numere.

Forma metaforică: compararea a două cantități și selectarea celei mai grele. Umplerea a două găleți cu apă, compararea lor cu ajutorul unui cântar și selectarea celei mai grele.

Setul de obiecte: robinet, apă, găleți, cântar.

Acțiuni posibile: deschidere/închidere robinet, umplere găleată cu apă, așezare găleată pe cântar, selecție găleată.

Desfășurarea lecției

Studentul modelează soluția problemei folosind obiectele disponibile și acțiunile permise în cadrul lecției. El poate să obțină forme metaforice acceptate ca soluție mergând pe trasee alternative, ducând totuși spre rezultatul așteptat. În acest caz se va completa baza de date a sistemului cu un nou algoritm.

În cazul în care utilizatorul nu reușește să ajungă la forma finală, are posibilitatea să vizualizeze forme metaforice intermediare pentru a ieși din impas și de asemenea poate urmări o trasare pas cu pas a modelării soluției corecte.

Bazându-se pe demersul determinist de rezolvare a problemei date, MDL realizează asocierea obiectelor și acțiunilor din procesul de modelare vizuală cu variabilele și instrucțiunile unui limbaj de programare. Lecția prezentată anterior se interpretează conform Tabelului 1.

Tabelul 1. Determinarea maximului a două numere date

Modelarea vizuală a soluției prin acțiuni	Instrucțiunile echivalente în pseudocod	Cod sursa C++
vizualizarea obiectelor disponibile (două găleți și un cântar) și acțiunile permise	două variabile și o structură alternativă	float x,y;
umplerea găleților cu apă	citirea valorilor a două variabile	cin>>x>>y;
cântărirea găleților	test logic if-then-else	if (x<y) cout<<x; else cout<<y;
selectarea găleții mai grele	afișarea soluției	

Din acest exemplu simplu reiese faptul că utilizatorul controlează aparent modelarea soluției într-un mod vizual, interactiv, fără să aibă vreo legătură cu editarea codului sursă. Controlul este aparent deoarece tocmai aici apare marea problemă în programare, în special în cazul utilizatorilor începători – se pierde în soluții. Pentru a evita afundarea în mulțimea de posibilități de rezolvare și

implicit prevenirea pe cât posibil a erorilor logice, utilizatorul este constrâns în a utiliza doar anumite obiecte date.

Am adoptat acest demers determinist pentru a conduce programatorul începător spre găsirea unei soluții corecte prin limitarea posibilităților și implicit a erorilor de

judecată. Rolul profesorului devine cu atât mai important cu cât el este responsabil de a încărca în baza de date - corespunzător fiecărei probleme propuse - obiecte și acțiuni ce permit modelarea cât mai multor soluții, anticipând și totodată ghidând elevul spre modelarea soluției optime.

GENERALIZARE

Analog exemplului descris anterior poate fi modelată orice lecție de programare, principiul urmărit fiind general valabil: utilizatorul scrie codul sursă prin acțiuni și nu prin instrucțiuni. Cunoscând relația acțiune-instrucțiune, MDL generează pseudocod, iar apoi, pe baza unor șabloane existente în baza de date, furnizează cod sursă executabil. La următoarea trasere, codul este vizualizat de utilizator, fiecărei acțiuni fiindu-i adnotată o instrucțiune.

Această abordare se poate extinde la orice lecție de programarea calculatoarelor sau în studiul unor tehnici de programare. Pot fi explicate concepte din programare precum tipuri de date, structuri de control, tipuri structurate, pointeri, arbori, grafuri, liste, subprograme, se pot studia tehnici de programare, proprietăți etc. De exemplu, următoarea formă metaforică: utilizatorul crează cercuri de diverse dimensiuni prin modelare grafică, ilustrând diverse proprietăți precum instanțierea unui obiect dintr-o clasă (clasa cerc), duplicarea, încapsularea, moștenirea, moștenirea multiplă etc.

Pentru avansați, din moment ce utilizatorul s-a familiarizat deja cu manevrarea obiectelor, nivelul de detaliu crește, lecția conține mai multe obiecte văzute acum ca variabilele echivalente într-un mediu de programare și continuă să modeleze, după preferință în mod grafic sau pseudocod, dar pe o cale crescând nedeterministă. MDL va conține în continuare unul sau mai multe soluții și va analiza rezultatul obținut de secvența de acțiuni a utilizatorului. Se vor obține astfel soluții noi, care vor fi memorate și folosite în lecțiile următoare. Deoarece în această abordare soluția e văzută ca o anumită stare a obiectelor date/utilizate, se poate testa orice soluție propusă de utilizator. În cazul în care cursantul nu găsește o soluție, el poate accesa soluții posibile sau îndrumări parțiale prin adnotări grafice pas cu pas.

CONCLUZII

Abordarea prezentată în această lucrare se vrea a fi un pas mai aproape de traversarea barierei dintre utilizator și programator față de metodele existente la ora actuală, în special la noi în țară, în didactica informaticii. Ne-am axat pe aspectul introductiv, pe familiarizarea studenților neinstruiți cu programarea calculatoarelor, pe componentele care realizează interfața dintre acești utilizatorii și mediul de programare. Aplicația prezintă unelte ușor de utilizat de către cadrele didactice în realizarea lecțiilor și un motor de evaluare automată a soluțiilor cursanților. Aceștia beneficiază de un sistem de învățare ușor accesibil, care le stimulează dezvoltarea gândirii algoritmice și îi ferește de constrângerile unui limbaj de programare. Totodată se realizează asocierea între modelarea vizuală și redactarea de cod, obținându-se în final cod sursă executabil.

Extinderi ale modelului propus

Demersul determinist, potrivit începătorilor, trebuie înlocuit cu unul tinzând spre nedeterminism o dată cu evoluția cursantului, deoarece utilizatorul expert nu mai are nevoie să modeleze vizual algoritmul, ci are abilitățile necesare pentru a scrie codul sursă propriu direct într-un mediu de programare.

Automatizarea proceselor. Trecerea de la o problema particulară la un algoritm general, de pildă, pentru studiul de caz prezentat în lucrare, generalizarea la aflarea maximului a n numere date; sortarea a n elemente; parcurgerea unui graf etc

Modul de interacțiune. Extinderea dispozitivelor utilizate în realizarea mediului educațional, de la scene 3D vizualizate pe monitor, interacțiune prin tastatura, mouse și/sau tabletă grafică, la un întreg mediu educațional virtual proiectat 3D și controlat prin mânășă.

Includerea lecțiilor generate de nucleul MDL într-o platformă de eLearning facilitând învățământul la distanță dar menținând, chiar prin construcția lor, controlul asupra feedback-ului și a evoluției liniare, în același timp facilitează, a capacității de dezvoltare a algoritmilor de către utilizatori de pretutindeni.

REFERINȚE

- 1.The Lifelong Kindergarten group, <http://ilk.media.mit.edu/index.php> [2010]
- 2.MIT Media Lab, <http://www.media.mit.edu/> [2010]
- 3.The Intel Computer Clubhouse Network, <http://www.computerclubhouse.org/> [2010]
- 4.Lego Mindstorms NXT. <http://mindstorms.lego.com/en-us/default.aspx> [2010]
- 5.PicoCricket computers, <http://www.picocricket.com> [2010].
- 6.Jonathan S. Harbour, Microsoft Visual Basic Game Programming for Teens, 2nd Ed, ISBN-13: 978-1598633900, Course Technology PTR, 2007.
- 7.Adobe Authorware 7. <http://www.adobe.com/products/authorware/> [2010]
- 8.Andrew J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Chris Scaffidi, Mary Shaw, Susan Wiedenbeck. "The State of the Art in End-User Software Engineering", ACM Computing Surveys. Accepted for publication. <http://faculty.washington.edu/ajko/papers/Ko2010EndUserSoftwareEngineering.pdf> [2010]
- 9.M Grötschel, W R Pulleyblank, Mathematics of Operations Research, Volume 11, Issue 4, ISSN:0364-765X, INFORMS, Linthicum, Maryland, USA, 1986, pp. 537 – 569.
- 10.Programa școlară pentru Informatică intensiv, clasa a IX-a. <http://www.edu.ro/index.php/articles/curriculum/c556+559+580+/> [2010]