FORMAREA VIITORULUI PROFESOR DE INFORMATICĂ DIN PERSPECTIVA TEHNOLOGIEI ORIENTATĂ OBIECT

***Silviu Gîncu**, doctorand*
Universitatea de Stat Tiraspol

Abstract*: *Information technologies are increasingly being used in various fields. A pillar that underlies their development is technology object oriented programming . In this context we consider necessary for future teacher of informatics to possess this technology
Key-words: *technology, object oriented programming, competency, situations, actions, resources, programming language, visual programming language.*

Preliminarii

Actualmente tehnologiile informaționale se află într-o expansiune continuă. Popularitatea lor în majoritatea domeniilor sociale impune pregătirea de specialişti, care să posede competențe în domeniul tehnologiilor informaționale. Un rol important în formarea competențelor specifice informaticii îl are profesorul de informatică. Anume în timpul orelor de informatică, sunt formate deprinderi de a utiliza calculatorul şi tehnologiile informaționale.

Începând 2005, pregătirea profesorilor de informatică este realizată în două cicluri: licență şi masterat. Pentru primul ciclu, disciplinele ce formează viitorul profesor de informatică, sunt divizate în patru componente:

a) componenta fundamentală,
b) componenta de formare a abilităților şi competențelor generale,
c) componenta de orientare socio-umanistă,
d) componenta de **orientare spre specializare**.

Tehnologiile informaționale sunt caracteristice ultimului compartiment. Una dintre tehnologiile care țin nemijlocit de programare este cea orientă pe obiect. Conform lui Grady Booch, „programarea orientată pe obiecte (POO) este o metodă de implementare în care programele sunt organizate ca colecții de obiecte ce cooperează între ele, fiecare obiect reprezentând instanța unei clase; fiecare clasă aparține unei ierarhii în cadrul căreia clasele sunt legate prin relații de moştenire”. [2, p.35]

Evoluția tehnologiei orientată pe obiect

Evoluția limbajelor de programare a trecut prin mai multe generații. S-a observat însă două tendințe:

- trecerea de la unități de programare mai mici la unele mai mari, care să ofere mai multe posibilități;
- dezvoltarea limbajelor de programare de nivel înalt.

Prima generație a limbajelor de programare o constituie perioada 1954-1958. Limbajele din această generație au fost concepute pentru a simplifica complexitatea formulelor matematice. Sunt caracteristice limbajele FORTRAN I, ALGOL-58, Flowmatic şi IPL V.

A doua generație datează din anii 1959-1961 şi se caracterizează prin concentrarea atenției asupra abstracțiunilor algoritmice. În acea perioadă calculatoarele au devenit mai

puternice, ele fiind utilizate în soluționarea problemelor de ordin economic, în evidența personalului etc. Sunt aplicate limbajele FORTRAN II, ALGOL-60, COBOL, Lisp.
Odată cu apariția tranzistoarelor, costul unui calculator (PC) a scăzut considerabil, astfel producerea PC a crescut exponențial. PC a dus la apariția unei noi generații de limbaje de programare în perioada 1962-1970: PL/I, ALGOL-68, Pascal, Simula. Aceste limbaje se caracterizează prin faptul că oferă un alt nivel de abstractizare a datelor, oferind programatorului posibilitatea de a crea noi tipuri de date.
Perioada 1970-1980 se caracterizează prin apariția mai multor limbaje de programare, care nu au făcut față cerințelor pieții. Problema abstractizării obiectelor a fost caracteristică pentru decada 1970-1980, astfel a apărut limbajul Simula, iar în baza acestuia şi altele, precum Smalltalk, Object Pascal, C++, CLOS, Ada şi Eiffel. Aceste limbaje au fost numite orientate pe obiecte.
Crearea sistemului de operare Windows a determinat elaborarea de programe, „capabile” să utilizeze resursele sistemului. Astfel au apărut şi limbaje de programare vizuală. În cadrul limbajelor de programare vizuală (VPL - Visual Programming Language), obiectele nu au o reprezentare vizuală, însă pentru realizarea unei interfețe prietenoase, se utilizează o reprezentare vizuală a obiectelor. De asemenea, construcțiile program şi regulile de combinare a acestor construcții sunt prezentate vizual. Domeniul aplicațiilor limbajelor de programare vizuală include grafica pe calculator, proiectarea interfețelor utilizator, interfețe cu bazele de date, gestiunea formelor (ferestre) şi proiectarea asistată de calculator.
În prezent cele mai populare limbaje de programare sunt limbajele orientate pe obiecte, impuse datorită celor patru elemente esențiale:
1. *Abstractizare* - procesul care captează caracteristicile esențiale ale unui obiect, caracteristici ce diferențiază acest obiect de toate celelalte tipuri de obiecte. Prin abstractizare se rezolvă problemele cu un grad sporit de complexitate. Hoare susține: „Abstractizarea se manifestă prin determinarea similarității dintre anumite obiecte, situații sau procese din lumea reală şi în luarea deciziilor ignorând pentru momente deosebirile dintre obiecte.” [1, p.83].
2. *Încapsulare* - proces mental executat pe o abstractizare în care elementele structurale ale abstractizării sunt izolate de cele ce constituie comportamentul său. Serveşte la separarea interfeței vizibile a unei abstractizări față de implementarea sa.
3. *Modularitate* - proces de divizare a unui sistem complex în părți (module) manevrabile. Conform lui Liskov, „modularitatea nu este altceva decât divizarea programului în mai multe unități, care deşi compilate separat, comunică între ele” [3, p.66.].
4. *Ierarhizare* - proces de clasificare pe nivele de abstractizare.

Modelul orientat pe obiecte

Din perspectiva programării, rezolvarea unei probleme se poate face pe 3 direcții:
a) rezolvarea orientată pe algoritm,
b) rezolvarea orientată pe date,
c) rezolvarea orientată obiect, combină tendințele primelor două abordări.

Acest mod de rezolvare se bazează pe crearea de obiecte şi pe interacțiunea dintre ele. Cele mai importante caracteristici ale tehnologiei orientate pe obiecte sunt:

A. *Obiectul* – asociază datele împreună cu operațiile necesare prelucrării. Datele sunt informații de structură descrise de o mulțime de atribute ale obiectului, iar operațiile (metode) acționează asupra atributelor obiectului şi eventual, asupra altor obiecte;
B. *Clasa* – reuneşte o colecție de obiecte care partajează aceeaşi listă de atribute informaționale şi comportamentale;
C. *Încapsularea* – principiul care se bazează pe combinarea datelor cu operațiile asupra unui obiect şi proprietatea obiectelor de ascundere date şi operațiile proprii față de alte obiecte;
D. *Instanțierea* –operația prin care se creează un obiect şi apoi se inițializează cu date specifice;
E. Crearea *ierarhiilor de clase*. Clasele pot fi ierarhizate prin:
1. *Moştenire* – principiul prin care putem reutiliza şi extinde clasele existente. Acest mecanism permite unei noi clase să beneficieze de atributele şi metodele definite într-o clasă deja existentă;
2. *Agregare* – operațiile în care clasele sunt descompuse în unități mai mici, care la fel sunt clase. O clasă agregat este o clasă ale cărei obiecte conțin alte obiecte.
F. *Polimorfismul* – posibilitatea de a putea aplica în moduri diferite aceeaşi operație mai multor clase. Prin operație înțelegem orice metodă sau operator. Polimorfismul reprezintă abilitatea unor obiecte similare de a răspunde la acelaşi mesaj în moduri diferite.

Avantaje oferite de tehnologia orientă obiect

Edificarea unei societăți sănătoase solicită formarea unei societăți bine educate. Una dintre componentele care participă la educarea societății o reprezintă cadrele didactice. Formarea profesională a acestora trebuie să răspundă în mare măsură la două întrebări: Ce va preda? Şi cum va preda?
Cercetătorii ruşi V. Matrosov, S. Jdanov, S. Karakazov şi N. Râjova în [4] evidențiază patru componente ale formării profesionale a profesorului de informatică:
A. Formarea teoretico-metodologică;
B. Formarea specială în domeniul informaticii;
C. Formarea metodică;
D. Formarea psihopedagogică.
Prin utilizarea tehnologiei orientă obiect se va ameliora formarea profesională a profesorului de informatică. Să analizăm unele dintre aceste avantaje.
Formarea unui *stil de gândire orientat pe obiecte*. Tehnologia orientată obiect impune un nou stil de gândire. Dacă în sistemul de învățământ preuniversitar accentul este pus pe formarea şi dezvoltarea gândirii algoritmice, atunci în cazul nostru accentul este pus pe dezvoltarea unei gândiri obiectuale. Acest model reprezintă modul uman de gândire. Datorită mecanismelor POO este mai uşor de a rezolva o problemă care necesită a prelucra diverse tipuri de date, predefinite de către utilizator.
Sistemul de învățământ din Republica Moldova este organizat pe niveluri şi trepte. Una dintre ele este şi învățământul mediu de specialitate (colegiu). Într-o bună parte din aceste instituții există specialitatea Informatica. Conform planurilor de învățământ la această specialitate figurează disciplina „*Programarea orientă pe obiecte*”.

Instituția	Discipline conform planului de învățământ	Numărul de ore
Colegiul politehnic	Programarea orientată pe obiecte	128 ore
Colegiul de Informatică	Programarea orientată pe obiecte (Delphi)	128 ore
	Programare Java	128 ore
	Bazele de date (SQL, Delphi)	128 ore
Colegiul Financiar-Bancar	Programarea orientată pe obiecte I	128 ore
	Programarea orientată pe obiecte II	256 ore

În acest sens este necesar de a pregăti cadre didactice competente pentru a profesa în toate instituțiile de învățământ existente în R.M.

Aşa cum mediile de programare vizuale orientate pe obiecte (în acest caz Borland Delphi) sunt foarte populare, elevii claselor liceale ar putea studia modul de utilizare acestora în cadrul orelor opționale.

Odată cu apariția limbajelor de programare orientate pe obiecte, evoluția produselor software a cunoscut un salt important. Anume în baza acestei concepții sunt elaborate o bună parte din produsele software. Aceasta va permite adaptarea mai rapidă a cadrului didactic la noi platforme de dezvoltare.

O importanță majoră în procesul didactic îl are resursele utilizate. Din acest punct de vedere, profesorul de informatică are posibilitatea de a selecta diverse platforme educaționale. Un dezavantaj al acestora îl constituie costul platformelor educaționale, care sunt foarte mari în condițiile R.M.. Acesta este unul dintre motivele care impune elaborarea de materiale didactice necesare pentru lecții. Prezentările Power Point dețin întâietate în elaborarea de materiale didactice. Deşi utilizarea acestora aduce un plus de calitate, sunt cazuri când este necesar de a prezenta elevului exemple concrete. În acest sens, prin intermediul mediilor de programare vizuală orientate pe obiecte, pot fi elaborate diverse programe educaționale pentru diverse compartimente.

Din propria experiență am observat că elevii întâmpină dificultăți la compartimentul „*Structuri dinamice de date*”. Pentru acest compartiment pot fi create programe care să vină în întâmpinarea elevului. Un astfel de program poate fi utilizat pentru temele: „*Arbori binari. Arbori binari de căutare*”.

Prin intermediul unui astfel de program elevul va avea posibilitatea de a urmări modalitatea de creare, inserare, parcurgere şi excludere a datelor dintr-un arbore binar de căutare.

Pentru realizarea unui astfel de produs va fi necesar de a crea un meniu cu opțiunile

a) crearea arborelui,
b) inserarea unui nod,
c) excluderea unui nod,
d) căutarea unui nod,
e) parcurgeri (vom adăuga alte trei opțiuni: inordine, preordine şi postordine, câte o opțiune pentru fiecare tip de parcurgere).

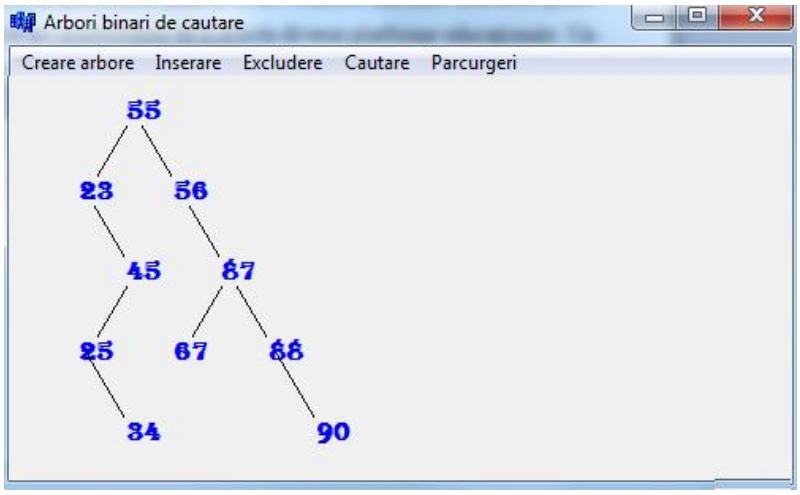


Fig.1 Forma aplicației *arbori binari de căutare*

Pentru realizarea unui astfel de produs va fi necesar de a crea un meniu cu opțiunile

f) crearea arborelui,

g) inserarea unui nod,

h) excluderea unui nod,

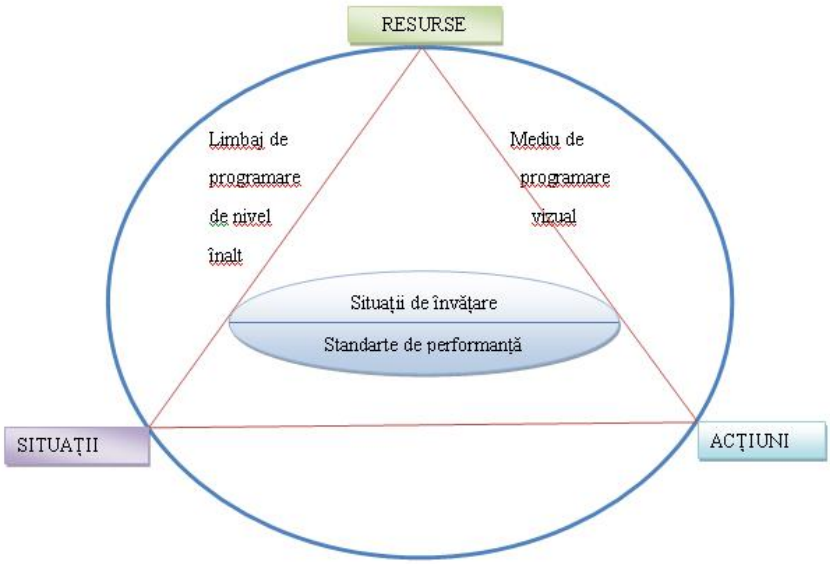
i) căutarea unui nod,

j) parcurgeri (vom adăuga alte trei opțiuni: inordine, preordine şi postordine, câte o opțiune pentru fiecare tip de parcurgere).

Acest program va servi şi drept model de evaluare pentru elev, deoarece, prin intermediul acestuia, el va avea posibilitatea de a se autoevalua, urmărind datele din program.

Astfel de programe pot fi elaborate şi pentru alte structuri dinamice de date, cum ar fi liste, cozi, stive. Pot fi elaborate aplicații de modelare pentru rezolvarea anumitor probleme de la compartimentul „*Tehnici de programare*”, cum ar fi turnurile din Hanoi sau tabla de şah, etc.

Tehnologia orientată obiect include în sine mai multe elemente. În mare măsură aceste elemente țin de conceptele tehnologiei, de modul de implementate şi capacitatea de utilizare a lor. În mod schematic aceasta poate fi reprezentată în felul următor:



Pentru a fi competent în utilizarea tehnologiei orientată obiect este necesar de a forma anumite competențe specifice:

CS 1. Cunoaşterea noțiunilor, principiilor şi mecanismelor caracteristice tehnologiei orientate pe obiecte.

CS 2. Identificarea domeniului de valori şi a setului de operații admisibile ale unui obiect.

CS 3. Implementarea conceptelor POO în baza unui limbaj de programare.

CS 4. Modificarea stării şi comportamentul obiectului în dependență de specificul problemei.

CS 5. Elaborarea aplicațiilor orientate pe obiecte.

CS 6. Capacitatea de proiectare a unei interfețe grafice.

CS 7. Capacitatea de a utiliza componente pentru gestiunea unei baze de date.

Formarea competențelor specifice poate fi efectuată prin prisma limbajelor de programare orientate pe obiecte. În aceste sens, formarea de competențe se va desfăşura în două etape:

Studierea conceptelor POO. Această etapă presupune a studia modul de definire a unei clase, de creare a unui obiect, precum şi modul de stabilire a relațiilor dintre acestea. Se recomandă a utiliza limbajele Object Pascal şi C++.

Aplicarea în practică a conceptelor POO. Această etapă presupune a utiliza un mediu de programare vizual orientat pe obiecte, prin intermediul căruia va fi studiată modalitatea de utilizare a componentelor. Aşa cum componente sunt foarte multe, studiul nu va fi concentrat pe o componentă sau alta, ci pe situații, adică vor fi prezentate situații (spre exemplu, aplicații gen calculator, editor de texte, etc.) şi propuse metode de soluționare a acestora. Se recomandă a utiliza platformele Borland Delphi sau Borland C++ Builder.

Programele prezentate nu sunt unice, pot fi utilizate şi alte limbaje precum Java sau mediul Visual Studio, însă datorită faptului că aceste limbaje nu sunt cunoscute de către studenți, ar fi necesar ca un anumit număr de ore să fie destinat studiului asupra sintaxei limbajelor menționate.

BIBLIOGRAFIE

[1] Dahl, O., Dijkstra, E. and Hoare, C.A.R. 1972. Structured Programming. London, England: Academic Press;

[2] Grady Booch. Object-Oriented Design with Applications. Benjamin/Cummings, Redwood City, California, 2nd edition, 1994 534 p.

[3] Liskov, B. 1980. A Design Methodology for Reliable Software Systems, in Tutorial on Software Design Techniques. Third Edition. New York, NY: IEEE Computer Society;

[4] Матросов В. Л.; Жданов С. А.; Караказов С. Д.; Рыжова Н. И. Перспективы развития предметной подготовки учителей информатики. http://bjalony.ucoz.ru/publ/8-1-0-103 vizitat 18.02.2012