Abordarea semiotică a programării interfețelor bazate pe manipularea directă a obiectelor

Prof.dr. Ion Gh. ROŞCA, conf.dr. Constanța BODEA, conf.dr. Ion SMEUREANU,
Catedra de Informatică Economică, A. S. E., Bucureşti

Lucrarea tratează o serie de aspecte vizând programarea interfețelor bazate pe manipularea directă a obiectelor, activitate interpretată din perspectiva semiotică drept activitate orientată pe semne. În acest context sunt analizate implicațiile teoretice și practice ale abordării semiotice asupra activității de programare a interfețelor bazate pe manipularea directă a obiectelor, cu exemplificare în domeniul instruirii asistate de calculator.

***Cuvinte cheie**: interfață bazată pe manipulare directă, semiotică, metasemiotică, instruire asistată de calculator.*

Introducere

Programarea orientată pe obiecte a promovat un stil propriu de interpretare a procesului de execuție a programelor, bazat pe un model fizic, simulându-se comportamentul unei părți reale sau imaginare din domeniul referențial. Apariția conceptului de interfață bazată pe conceptul de manipulare directă a obiectelor (IMD) poate fi considerată drept o consecință a acestei perspective în înțelegerea programării. În cadrul IMD acțiunile utilizatorilor constau în diferite operații asupra unor obiecte vizibile, operații prin care este posibilă realizarea scopurilor de interacțiune.

Programarea IMD se poate realiza într-un mod tradițional, centrat pe structurile interne ale proceselor de execuție (deci invizibile din perspectiva utilizatorului) și în care se acordă o mică atenție faptului că procesele de execuție trebuie să primească o interpretare din partea utilizatorului. Programarea se bazează în general pe o interpretare matematică a operațiilor de interacțiune om-calculator. Ecranul este văzut ca un sistem de coordonate în care obiectele reprezintă figuri geometrice, fotografii, iconuri sugerând obiecte din lumea reală.

Codul programelor este lung și greu de relaționat cu elementele de manifestare vizuală. Activitatea de programare este complicată, avându-se în vedere faptul că în faza de execuție a programului trebuie să se realizeze următoarele procese:

a) manipularea operațiilor externe (ale utilizatorului) în operații interne:

$$O_{ext} \longrightarrow I_1 \longrightarrow O_{int}$$

unde I_1 reprezintă o funcție de transfer (de interpretare) a operațiilor externe (O_{ext}) în operații interne (O_{int}).

b) executarea operațiilor interne:

$$O_{int} \longrightarrow \text{execuție} \longrightarrow \text{efecte } O_{int}$$

c) manipularea efectelor operațiilor interne în efecte asociate operațiilor externe:

$$\text{efecte } O_{int} \longrightarrow I_2 \longrightarrow \text{efecte } O_{ext}$$

unde I_2 reprezintă o funcție de interpretare a efectelor generate de operațiile interne în efecte ale operațiilor externe.

O altă posibilitate de a realiza programarea IMD o reprezintă programarea orientată pe obiecte, care are avantajul că exprimă operațiile programate în termeni inteligibili pentru utilizator, fiind asociate unor obiecte sau acțiuni reale. Plecând de la programarea orientată pe obiecte, în domeniul IMD se poate introduce conceptul de programare orientată pe semne (POS), prin interpretarea unui

anumit tip de obiecte (respectiv a celor ce prezintă proprietăţi vizuale şi au diferite funcţii relativ la semnificaţia execuţiei unui program) drept semne ([1]). Semnul (în cadrul IMD) reprezintă o redefinire a obiectului în cadrul programării orientate pe obiecte) în sensul că obiectele sunt văzute în planul exprimării, în timp ce conţinutul semnului este generat în cadrul contextului de lucru. Obiectele, ca semne, nu mai sunt entităţi de sine stătătoare (ca în cadrul programării orientate pe obiecte, ci puncte terminale ale funcţiilor dintre sistem şi utilizator. Din această perspectivă, programarea IMD poate fi interpretată drept o activitate de creare a semnelor.

Semne în cadrul IMD - caracteristici, tipologie

IMD poate fi interpretată drept un ansamblu de semne relative la procesele ce sunt percepute, interpretate şi utilizate de o anumită comunitate de utilizatori. Desemnează deci o relaţie între părţile vizibile ale sistemului şi utilizator. Un semn are asociate atribute (caracteristici) şi acţiuni pe care le poate realiza asupra altor semne. Atributele semnelor pot fi de sistem sau de utilizator.

Atributele de sistem sunt cele cu care operează în mod curent sistemul, cum ar fi: numele obiectului, mărimea, culoarea etc. Ele pot fi generale sau specifice unei clase de semne. Spre exemplu, fontul este o caracteristică doar a semnelor de tip text, în timp ce culoarea se menţionează pentru toate semnele ce se desenează pe ecran.

Atributele utilizator sunt proprietăţi asociate de programator unui obiect grafic, în funcţie de conţinutul semantic al acestuia, ca de exemplu: debitul unei conducte într-o schemă tehnologică, punctajul acumulat de un subiect în procesul de verificare-instruire etc.

În raport de persistenţa şi semnificaţia valorilor se pot indentifica trei clase de atribute: persistente,nepersistente (temporare) şi de manipulare.

Atributele persistente sunt cele ale căror valori o dată fixate se salvează şi sunt reţinute între două sesiuni de lucru sau instanţe (de exemplu, culoarea background-ului).

Atributele nepersistente (temporare) sunt cele care au valori specifice doar pe perioada cât este activat obiectul, după care ele revin la valorile standard (de exemplu, volumul la care este redat un clip muzical, poziţia curentă în clip etc.). Semnifică stările în care se poate afla semnul, articulând astfel un sistem de transformări. Această clasă de atribute permite identificarea rapidă a contextului de către utilizator.

Atributele de manipulare sunt generate de utilizator sau în mod automat şi descriu reacţiile obiectelor la evenimentele din sistem. Servesc la programarea acţiunilor.

Combinând caracteristicile semnelor cu acţiunile ce pot fi realizate de aceste semne asupra altor semne se pot defini următoarele clase de semne:

Semnele interactive prezintă toate caracteristicile asociate semnelor, putând, de asemenea, executa acţiuni asupra altor semne. Ca exemplu, se pot menţiona cursorul text şi cursorul mouse din WORD. Astfel, cursorul mouse îşi poate schimba poziţia, afectând şi poziţia cursorului text. Cursorul text îşi poate schimba la rândul lui poziţia, putând fi manipulat prin cheile alfanumerice. Unele semne interactive au trăsături temporare atât de simple incât pot fi considerate ca nereprezentând această clasă de caracteristici. Este cazul, de exemplu, al butoanelor de interfaţă, care reprezintă deci **semne declanşatoare**.

Semnele actori au asociate acţiuni asupra altor semne şi nu prezintă caracteristici de manipulare.

Semnele pasive posedă caracteristici persistente şi temporare, dar nu şi de manipulare. Nu pot influenţa alte semne, dar pot fi influenţate de alte semne, în special de cele interactive. De exemplu, documentul reprezintă un semn pasiv ce poate fi influenţat prin intermediul cursorului text.

		+a		-a
		+m	-m	
+p	+t	semne interactive	semne actor	semne pasive
	-t	semne declanşatoare	semne controler	semne decorative
-p		semne fantomă		

+/- p semnifică prezenţa/absenţa caracteristicilor persistente
+/- t semnifică prezenţa/absenţa caracteristicilor temporare (nepersistente)
+/- m semnifică prezenţa/absenţa caracteristicilor de manipulare
+/- a are semnificaţia asocierii/lipsei acţiunilor asupra altor semne

Semnele controler acţionează asupra altor semne, dar nu îşi pot schimba aspectul vizual. De exemplu, marginile de ferestre schimbă cursorul text dar ele rămân nemodificate.

Semnele decorative reprezintă ansamblul semnelor cu funcţie de îmbunătăţire a formei de prezentare a ecranului, cărora le lipsesc trăsăturile temporare şi cele de manipulare, precum şi acţiunile asupra altor semne.

Semnele fantomă nu prezintă trăsături persistente şi temporare vizibile. Din această cauză nu sunt reprezentate prin iconuri sau alte elemente grafice identificabile pe ecran şi deci nu pot fi manipulate direct. Au însă funcţii în raport cu alte semne, justificându-şi astfel existenţa în cadrul IMD.

Multimedia ToolBook (TBK) reprezintă un mediu de realizare a IMD multimedia ([3]) care articulează operaţiile în cadrul interacţiunii ca procese de prelucrare a semnelor. Tipurile de semne implementate de TBK sunt: butoanele, meniurile câmpurile, ferestrele de dialog, back-ground-ul. Obiectele de interfaţă prezintă propietăţi standard care sunt în mod normal utilizate ca mijloace de expresie, respectiv: localizarea pe ecran, iconul, paleta de culori, clipul (muzical sau video). Varietatea de propietăţi acceptate de TBK oferă utilizatorului mai multe canale de comunicare (text, imagine video, sunet), furnizând deci o interfaţă multimedia. Obiectele pot avea o propietate numită script care descrie acţiunile în termenii programării, servind astfel la implementarea caracteristicilor de manipulare. O parte de scripturi, şi anume handlerele, pot fi generate în mod automat, ca reacţii la evenimentele uzuale din sistem (de exemplu, acţionări de taste) sau pe baza înregistrării unei suite de acţiuni ale utilizatorului asupra obiectelor din sistem (selecţie meniu, schimbarea atributelor de obiecte, animaţie pe o direcţie fixă etc.). Acţiunile descrise de un script sunt declanşate ca reacţie la mesajele primite de obiect. Mesajele pot veni de la utilizator (acţionări de taste, mouse) sau de la sistem. În permanenţă sistemul furnizează un mesaj tact, numit *idle* ce poate fi preluat şi interpretat diferit de diverse obiecte şi poate sta la baza sincroniză-

rii unor acţiuni. Începutul şi sfârşitul acţiunii unui obiect pot genera la rândul lor mesaje pentru alte obiecte ce dispun de proprietăţi de tipul Notify AfterMessages,NotifyBeforeMessages. În cazul sistemelor multimedia, sincronizarea dintre imagine şi sunet joacă un rol deosebit de important. Acest tip special de sincronizare presupune prezenţa în cadrul instrucţiunii *play* a unei clauze Notify.

Programarea orientată pe semne a IMD

Întrucât semnul reprezintă unitatea dintre conţinut şi forma de exprimare, activitatea de programare a semnelor trebuie să se realizeze, pe de o parte, la nivel semiotic, prin exprimarea caracteristicilor relevante în ceea ce priveşte semnificaţia (conţinutul) şi, pe de altă parte, la nivel metasemiotic, prin descrierea aspectelor de implementare (de manifestare într-un anumit mediu) a semnelor.
Descrierea la nivel semiotic a câtorva acţiuni frecvente în modulele de instruire poate apare astfel:

```
to handle buttonDoubleClick
  -stabilire proprietate obiect
    set name of field id20 to ANUNT
  - vizualizare obiect
    show field ANUNT
end
to handle buttonClick
  - stabilire propietate nepersistentă
     prin apelul unei
  - funcţii de tratare
    send marcare to self
end
```

În procesul de instruire, o tehnică frecventă este cea a stabilirii unor conexiuni între cunoştinţe. De exemplu, asocierea unui obiect cu clasa căreia îi aparţine se poate realiza prin mecanismul *drag&drop*. Descrierea acestui mecanism la nivel metasemiotic, prin secvenţe OPENSCRIPT se prezintă astfel:

```
to handle buttonDown
      drag target silently
end
to handle endDrag destObj
  conditions
    when destObj is null or targetType
         of destObj is null
         request "Destinaţie absentă"
         show target
    when targetType of destObj <>
         objectType of target
         request "Asociere incorectă!
                  Mai încearcă!"
          show target
    end
end
```

Utilizarea IMD în sistemele de instruire asistată de calculator

Instruirea asistată de calculator (IAC) necesită facilităţi deosebite în interacţiunea om-calculator. Interfeţele de tip IMD exploatează într-o mare măsură semantica atribuită de utilizator, chiar neinformatician, unor semne ce descriu lumea reală. Spre exemplu, iconul sau imaginea unui obiect real sugerează uşor acţiunile sau atributele ce i se pot asocia: un automobil se poate deplasa printr-o animaţie pe cale fixată, un semn de circulaţie indică o interdicţie sau un drept de acces, un document poate fi deschis pentru editare, un cronometru poate fi pornit şi oprit, un difuzor controlează un mesaj sonor, un TV permite vizualizarea unui clip video etc.
Prin semnele de navigaţie utilizate, o interfaţă cu obiecte manipulabile direct permite o călătorie virtuală într-un model de lume reală, alegerea traseului aparţinând în exclusivitate utilizatorului care va lua decizii specifice fiecărui context. În procesul învăţării, utilizatorul poate fi ajutat în alegerea traseului logic de urmat printr-o înşiruire

articulată a cunoştinţelor. În procesul verificării cunoştinţelor, sistemul poate reacţiona în funcţie de răspunsurile date, trimiţând subiectul pe domeniul insuficient stăpânit.

De cele mai multe ori organizarea cunoştinţelor se face tot pe principiul cărţii, cu capitole, subcapitole, tematicii etc., iar semnele săgeţi de orientare, butoane cu numele capitolelor permit o răsfoire selectivă a tematicii de instruire, cu bifarea subiectelor parcurse şi păstrarea ca atribut persistent a stării la care s-a ajuns pentru o continuare ulterioară. Principiul de navigare hipertext, bazat pe "cuvinte active" ca obiecte sensibile ce te trimit pe nivele de profunzime mai mari este uşor de asimilat de către subiectul care învaţă, fără ca acesta să cunoască script-ul care se ascunde în spatele obiectului. Orientarea script-urilor pe obiecte permite asocierea unor conţinuturi diferite aceluiaşi obiect ("obiect container"), astfel încât o aplicaţie să poată dialoga opţional în limbi diferite, câmpul comportându-se în acest caz ca un container de text.

Metasemiotica structurilor de semne permite construirea unor mecanisme generice, cu principii de funcţionare cunoscute, ale căror script-uri manipulează obiectele prin pointeri (de tip self, target) şi a căror instanţiere se produce la momentul evenimentului utilizator. În acest mod se construiesc mecanisme "matching quiz" pentru evaluarea automată pe bază de chestionar, mecanisme pentru căutarea în index de cunoştinţe pe bază de cuvinte cheie sau de evaluare pe bază de răspunsuri contra cronometru (tratându-se evenimentul Endtime, preluat de la obiectul din sistem care face evaluarea).

Concluzii

Abordarea semiotică a IMD poate fi aplicată nu numai procesului de programare, ci şi celui de proiectare a IMD.

Ceea ce este extrem de important în cadrul programării semnelor este descrierea semnelor compuse. Acestea corespund unor lanţuri de acţiuni ale utilizatorului şi trebuie descrise în program atât în planul conţinutului, cât şi în cel al modului de exprimare, de implementare. Din punct de vedere lingvistic, semnele compuse reprezintă descrieri ale utilizării semnelor. Semnele compuse corespund în acest caz unor concepte lingvistice, ca propoziţiile, frazele etc. În cazul IMD, programarea semnelor poate beneficia de transferul conceptelor şi metodelor din cadrul teoriilor lingvistice. Stările sistemului pot fi descrise prin arbori sin-tactici, care să includă atât părţile vizibile, cât şi cele invizibile ale IMD. Transformările sistemului pot fi interpretate drept mapări ale arborelui într-un alt arbore pe baza unor reguli de transformare. Caracteristicile persistente ale semnelor reprezintă proprietăţi ale IMD, iar acţiunile asociate acestor semne pot fi descrise drept reguli transformaţionale. În promovarea paradigmei programării orientată pe semne este necesară realizarea unor instrumente care să susţină acest mod de programare ce prezintă drept caracteristice esenţiale separarea textului de program la cele două niveluri: al invarianţilor, cu funcţii de comunicare şi al conţinutului.

Bibliografie

1. Andersen P.B. - *A theory of computer semiotics*, C.U. Press, 1990.
2. Bodea C. - *Cerinţe ale dialogului în sistemele de tip expert*, în: Sisteme expert în sprijinul practicii economice, A.S.E., Bucureşti, 1990.
3. *Multimedia ToolBook, OpenScript Reference*, Asymetrix Corporation, Washington USA, 1994.