MODEL DE INTERACŢIUNE GRAFICĂ OM-CALCULATOR, ORIENTAT OBIECT

ec. Costin Pribeanu

Institutul de Cercetări în Informatică

Rezumat: În acest articol, se propune un model de interacţiune om-calculator, orientat obiect, care realizează o extensie a conceptului de dispozitiv logic de intrare. Modelul defineşte o ierarhie de instrumente grafice de interacţiune pe mai multe niveluri, care are la bază un manager de evenimente abstracte. Abordarea facilitează descrierea interacţiunii cu ajutorul unei notaţii textuale, orientată pe evenimente.
Cuvinte cheie: interacţiune om-calculator, interfeţe grafice, grafică pe calculator, sisteme grafice orientate obiect.

1. Introducere

În cadrul acestui articol, se prezintă un model de interacţiune grafică om-calculator, orientat obiect, care să realizeze o extensie a conceptului de dispozitiv logic de intrare. Implementarea modelului se bazează, atât pe instrumentele de interacţiune, oferite de sistemul Microsoft Windows, cât şi pe definirea a două clase de intrări grafice, care nu sunt furnizate de sistemul gazdă.

Reorientarea către cerinţele utilizatorului final a grăbit cristalizarea interacţiunii om-calculator (HCI – Human-Computer Interaction) ca domeniu distinct, apărut prin convergenţa unor preocupări din ingineria factorilor umani, ergonomie, psihologie aplicată şi tehnologia informaţiei. Această lărgire a cadrului de lucru teoretic a generat două metamodele de interacţiune om-calculator, prezentate în secţiunea următoare.

În planul realizării componentei de input grafic, există două grupe de abordări. Standardele de grafică pe calculator au definit modelul dispozitivului logic de intrare, ca abstractizare menită să asigure portabilitatea aplicaţiilor faţă de dispozitivele fizice. O abordare alternativă o constituie definirea unor obiecte de interacţiune grafică, caracterizate prin prezentare atrăgătoare şi posibilităţi de manipulare, sugerate de însăşi forma acestora. Cele două abordări sunt descrise în cadrul a două secţiuni şi sunt urmate de o prezentare a unui alt concept – instrumente de interacţiune – prin care se propune o integrare a componentelor de input şi de output grafic.

Definirea modelului de interacţiune grafică om-calculator este făcută în cadrul general de interacţiune definit de ciclul de execuţie-evaluare introdus de Norman şi extins de Abowd şi Beale. La nivelul sistemului, acţiunile, operaţiile şi comenzile, articulate de utilizator într-un limbaj input, sunt transferate de către interfaţa în evenimente fizice, abstracte şi semantice. Avantajul introducerii evenimentelor abstracte îl constituie posibilitatea de a standardiza interfaţa, la nivelul procedurii care tratează evenimentele.

Modelul defineşte o ierarhie de instrumente grafice de interacţiune pe mai multe niveluri, care are la bază un manager de evenimente. Această abordare facilitează descrierea interacţiunii cu ajutorul unei notaţii orientată pe evenimente.

În ultimele secţiuni ale acestui articol se prezintă modelul de interacţiune grafică, un exemplu de descriere a interacţiunii, precum şi unele aspecte privind implementarea modelului de interacţiune.

2. Modelarea interacţiunii om–calculator

Modelele de interacţiune ajută să înţelegem ce se întâmplă în interacţiunea dintre utilizator şi sistem. Întrucât la fundamentarea teoretică a HCI îşi aduc o contribuţie importantă mai multe discipline, este necesar un cadru general al interacţiunii, care să ajute la delimitarea diferitelor subiecte de interes legate de modelarea interacţiunii.

În literatura de specialitate, se întâlnesc două abordări în definirea unui cadru de lucru general pentru HCI: prima, având la bază modelul lui Norman, al ciclului execuţie-evaluare şi a doua bazată pe modelul lui Carroll, al ciclului activitate - artefact.

Cele două abordări evidenţiază aspecte diferite ale aceleiaşi relaţii, între utilizatorul care desfăşoară o anumită activitate şi artefactul computer. Perspectiva este diferită, ciclul execuţie-evaluare fiind mai apropiat de contextul tehnologiei informaţiei în timp ce ciclul activitate-artefact este specific psihologiei.

Cel mai cunoscut este metamodelul definit de Norman [10], al ciclului de execuţie-evaluare, care descrie interacţiunea în termeni de obiective (scopuri) şi acţiuni ale utilizatorului. Ciclul interactiv cuprinde:

- execuţie: stabilirea obiectivului, formarea intenţiei, specificarea secvenţei de acţiuni şi executia acţiunilor;
- evaluare: observarea stării sistemului, interpretarea acesteia, evaluarea în raport cu obiectivele propuse.

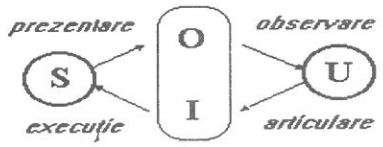

Figura 1. Cadru de interacţiune (după Abowd şi Beale)

O extensie a acestui model, propusă de Abowd şi Beale şi descrisă în [3], defineşte un cadru general de interacţiune. Cele patru componente ale unui sistem interactiv sunt sistemul, utilizatorul, outputul şi inputul. Outputul şi inputul formează interfaţa.

Singura modalitate prin care omul poate interacţiona cu maşina este prin introducerea de comenzi şi date, astfel încât activitatea trebuie articulată într-un limbaj input. Limbajul input este translatat în limbajul nucleului sub formă de operaţii care trebuie efectuate de către sistem.

Sistemul îşi transformă starea ca urmare a operaţiunii input. Starea sistemului este redată în concepte sau trăsături output. Din acest moment începe faza de evaluare: omul observă outputul şi compară rezultatul interacţiunii cu obiectivul iniţial.

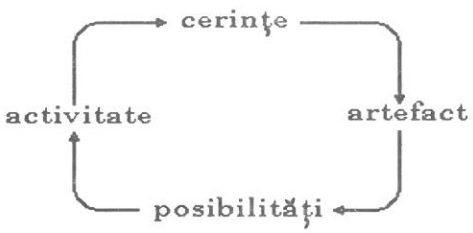

Figura 2. Ciclul activitate – artefact (după Carroll, 1990)

O activitate stabileşte în mod implicit cerinţele pentru proiectarea de artefacte. Utilizarea artefactelor conduce la redefinirea activităţii pentru care acesta a fost iniţial proiectat. Acest fapt conduce la o redefinire a cerinţelor şi necesită studierea artefactelor în contextul utilizării acestora.

În lucrarea menţionată se sugerează că artefactele ar trebui văzute ca teorii HCI încorporate, create de către proiectant. Modelul propus încearcă să integreze concepte din practică care au o existenţă independentă.

În [8] se prezintă o variantă rafinată a acestui model, care pune accentul pe psihologie în conceptualizarea HCI, având în vedere reprezentări bazate pe scenarii, studiul artefactelor în utilizarea, analiza şi evaluarea activităţii în raport cu cerinţele de fundamentare a proiectului.

Fundamentarea proiectului (design rationale) furnizează o descriere detaliată a istoriei şi a înţelesului unui artefact. Constituie un spaţiu de lucru, în care nivelul de detaliere este variabil, de la probleme cheie şi cerinţe de bază la detalii în interiorul proceselor de proiectare. Întrucât oferă o vedere dinamică asupra unui artefact, este adecvată naturii interactive a procesului de proiectare.

Scenariile de interacţiune furnizează descrieri generalizate (decontextualizate) ale acţiunilor pe care utilizatorul doreste să le îndeplinească. Pe parcursul proiectării artefactului, acestea sunt particularizate pentru a corespunde aplicaţiei şi platformei hardware. Proiectarea bazată pe scenarii, constituie o metodă care poate fi aplicată în ilustrarea unei specificaţii, instruirea utilizatorilor, evaluarea aplicabilităţii.

3. Modelul dispozitivului logic de intrare

Standardizarea în grafica pe calculator a adus o ordonare a conceptelor şi o definire mai riguroasă a inputului grafic, structurat pe moduri de operare şi pe dispozitive logice de intrare. Standardele GKS şi PHIGS furnizează un model independent de dispozitiv pentru proiectarea dialogului om-calculator, care are la bază o descriere a inputului grafic în termeni de dispozitive logice de intrare, măsură şi trăgaci (trigger).

Un dispozitiv logic de intrare este o abstractizare a unuia sau a mai multor dispozitive fizice. Furnizează programului de aplicaţie o valoare logică de intrare.

Un set de dispozitive de intrare, echivalente din punctul de vedere al funcţiei input pe care o îndeplinesc, constituie o clasă de intrări. Tipul valorii logice de intrare este determinată de clasa de intrări. Sunt definite şase clase de intrări grafice: *locator, stroke, valuator, choice, pick şi string*, care permit introducerea unei poziţii, unui vector de poziţii, unei valori reale, selectarea unei alternative, intercepţia grafică şi introducerea unui şir de caractere.

Fiecare dispozitiv logic de intrare poate fi utilizat într-un mod de operare, care defineşte modalitatea în care o valoare logică de intrare poate fi introdusă de către operator şi transferată programului de aplicaţie. Modul de operare este selectat de către programul de aplicaţie. Standardele GKS şi PHIGS definesc trei moduri de operare: *request* (la cerere), *sample* (prin eşantionare) şi *event* (prin evenimente).

Un dispozitiv logic de intrare este în interacţiune atunci când este în modurile *event* sau *sample* sau pe parcursul executării unei funcţii în mod *request*.

Un dispozitiv logic de intrare este definit prin:

- o măsură şi un trăgaci: fac parte din implementarea staţiei de lucru;
- o valoare iniţială, un tip de invitaţie (prompter) /ecou şi o înregistrare de date (cu detalii despre tipul de invitaţie / ecou): sunt furnizate de programul de aplicaţie.

Măsura unui dispozitiv logic de intrare este valoarea determinată de unul sau mai multe dispozitive logice de intrare împreună cu o mapare fizic-logic. Mai multe măsuri pot fi determinate de un singur dispozitiv fizic, prin procese de măsurare (mapări) diferite. Un proces de măsurare poate fi văzut ca un proces independent, care există, atâta timp cât dispozitivul logic de intrare este în interacţiune. Starea curentă a procesului este disponibilă ca o valoare logică de intrare.

Ecoul este un output către operator, care arată starea curentă a procesului de măsurare şi furnizează un mecanism de feedback, care permite ajustarea inputului la valoarea dorită. Invitaţia (prompter) este un mesaj către operator de a acţiona, semnalând că dispozitivul este gata pentru utilizare.

Trăgaciul unui dispozitiv logic de intrare este un dispozitiv fizic sau un set de dispozitive fizice, împreună cu o mapare de tragere (trigger mapping). Operatorul acţionează un trăgaci pentru a marca momente semnificative în timp. O singură acţiune a unui operator poate determina mai multe trăgaciuri să fie acţionate. De asemenea, mai multe dispozitive logice de intrare se pot referi la acelaşi trăgaci.

Un proces de tragere poate fi văzut ca un proces independent, care trimite mesaje către unul sau mai mulţi recipienţi atunci când este acţionat. Un dispozitiv logic de intrare este recipient al procesului dacă există o funcţie *request* în derulare sau dacă dispozitivul este în mod *event*.

Efectul mesajului transmis depinde de modul de operare:

- în modul de operare *request* procesul de măsurare returnează valoarea logică de intrare programului de aplicaţie şi încetează;
- în modul de operare *sample*, mesajele sunt ignorate, întrucât programul de aplicaţie poate obţine valoarea logică de intrare în orice moment, fără a aştepta intervenţia operatorului;
- în modul de operare *event*, procesul de măsurare încearcă să adauge o înregistrare (un raport) în coada de evenimente; dacă mai multe dispozitive logice de intrare sunt asociate aceluiaşi trăgaci, este adăugat un grup de evenimente simultane.

Pentru fiecare clasă de intrări există o funcţie de iniţializare, care poate fi apelată numai dacă dispozitivul logic de intrare este în mod *request*. Funcţia permite furnizarea de informaţii cuprinse în lista de stare a staţiei de lucru. Iniţial, un dispozitiv logic de intrare este în mod *request*, ceea ce înseamnă că nu există un proces de măsurare asociat, iar identificatorul dispozitivului nu se află pe lista proceselor de tragere.

Când se invocă o funcţie în mod *request*, se aşteaptă introducerea unei valori de către operator sau anularea comenzii (BREAK, ESC, buton RESET, în funcţie de implementare); în cel din urmă caz, valoarea logică de intrare este invalidă.

Când se invocă o funcţie input în mod *sample*, se returnează valoarea logică de intrare (starea curentă a procesului de măsurare), fără a aştepta acţiunea operatorului. Prin urmare, momentul semnificativ în timp, când informaţia este transferată, este determinat de programul de aplicaţie, şi nu de către operator.

În modul *event*, în coada de evenimente sunt adăugate, la acţionarea unui trăgaci, rapoarte care conţin perechea identificator – valoare logică de intrare. Când este acţionat un trăgaci care aparţine mai multor dispozitive logice de intrare, toate evenimentele rezultate sunt introduse în coadă şi etichetate ca un grup de evenimente simultane.

Evenimentele pot fi extrase cu ajutorul unei funcţii, *await event*, care copiază primul raport din coadă, dacă nu este vidă, într-o înregistrare din lista de stare a nucleului grafic, denumită raport curent. Dacă este vidă, se aşteaptă introducerea unui eveniment sau expirarea timpului de aşteptare specificat. Programul de aplicaţie poate prelua conţinutul raportului curent cu ajutorul unei funcţii *get*.

4. Obiecte de interacţiune

În ultimii ani, se constată o abundenţă pe piaţa software a unor pachete de instrumente (toolkit-uri) care permit tratarea input-ului şi output-ului în mod corelat, la nivelul unor obiecte de interacţiune (widgets, gadgets) care au un comportament predefinit.

Pentru a oferi flexibilitate, acestea pot fi adaptate cerinţelor prin parametrizarea unor caracteristici sau specificarea unui tip de comportament. În acest fel se forţează consistenţa în ceea ce priveşte forma input şi output.

Două trăsături ale acestor obiecte de interacţiune le fac adecvate paradigmei de orientare pe obiecte în programare:

- definirea de clase care pot fi instanţiate;
- construirea de ierarhii, bazate pe moştenire.

Sistemele de vizualizare bazate pe ferestre (sisteme de tip Windows) reprezintă un mediu interactiv, atât pentru programator, cât şi pentru

utilizator, permiţând definirea şi interacţiunea cu o gamă largă de obiecte de interacţiune.

În esenţă, aceste obiecte de interacţiune sunt ferestre de lucru pentru utilizator, definite în cadrul unei ierarhii de generalizare/specializare, capabile de a suporta un anumit tip şi stil de interacţiune grafică.

Din punctul de vedere al programatorului, fereastra este un tip complex de informaţie grafică, definită la nivelul unei clase de bază. Programatorul lucrează cu ferestre definite ca obiecte care recepţionează input-ul utilizator sub formă de mesaje. Comunicarea între ferestre se realizează, de asemenea, prin intermediul mesajelor. Comunicarea prin mesaje între sistem şi fereastră sau între ferestre este o consecinţă a orientării pe obiecte şi conduce la o arhitectură orientată pe mesaje (message-driven).

La producerea unui eveniment semnificativ (de ex. se redimensionează o fereastră), sistemul Windows trimite un mesaj către fereastra respectivă indicând noua dimensiune şi permiţând astfel programului să actualizeze în consecinţă conţinutul ferestrei. Recepţionarea mesajelor transmise ferestrei de către sistem se face în cadrul unei proceduri asociate ferestrei. Procedura procesează mesajul în mod corespunzator, după care returnează controlul sistemului Windows.

Controalele (componentele de dialog) sunt ferestre speciale, dedicate să suporte interacţiunea cu utilizatorul. Există mai multe tipuri de controale, fiecare tip fiind specializat pentru o categorie de operaţii de input şi output grafic. De exemplu, în sistemul Microsoft Windows (marcă inregistrată de Microsoft Corporation) sunt definite următoarele clase de controale: *button, scrollbar, edit, listbox, combobox şi static.*

5. Instrumente de interacţiune grafică

Cele două modele de interacţiune grafică om-calculator, descrise în capitolele precedente, reflectă două orientări diferite, care au la bază cerinţe diferite:

- cerinţe de portabilitate, specifice dezvoltarii software, exprimate de programatorul de aplicatie;
- cerinţe de calitate a interfeţei, specifice interacţiunii om-calculator, exprimate de utilizatorul final.

Dispozitivul logic de intrare este o abstractizare definită pentru programatorul interfeţei grafice, în scopul degrevării acestuia de caracteristici particulare ale diferitelor dispozitive fizice.

Utilizatorul final vede obiectul grafic asupra căruia acţionează. Din punctul său de vedere, dispozitivul logic de intrare este transparent, putând avea diferite ipostaze (implementări). Transparenţa se poate extinde şi asupra unor funcţii din aplicaţie, încorporate în obiectul de interacţiune.

De exemplu, butonul de comandă este o implementare (un stil de prompter/ecou, în termenii dispozitivului logic de intrare) a clasei de intrări *choice* (selecţie). Faptul că imediat după selecţie se invocă o comandă, nu este perceput de către utilizator ca o funcţie în afara modelului input.

Aşa cum s-a arătat în fazele anterioare, fiecare din cele două perspective are importanţa sa. Un model de interacţiune grafică, care să răspundă atât cerinţelor dezvoltării software, cât şi ale utilizării finale, trebuie să integreze concepte din ambele abordări. Un exemplu de model de interacţiune om-calculator stratificat, într-o abordare orientată pe obiecte este prezentat în [9]. Implementarea modelului este realizată deasupra sistemului X Windows.

Modelul propune o integrare, într-o ordine pornind de la hardware la programul de aplicaţie, a modulelor de interacţiune grafică, stratificate pe niveluri de abstractizare. Nivelurile definite sunt evenimente fizice, evenimente abstracte, instrumente de interacţiune şi instrumente compozite (paternuri de interacţiune).

Primul nivel de interacţiune cuprinde evenimente care sunt furnizate de un manager de ferestre : apăsarea unei taste, a unui buton, eliberarea unui buton, mişcarea unui mouse. Constituie nivelul de interfaţă cu un sistem de tip Windows, care preia acţiunile elementare ale operatorului asupra dispozitivului fizic.

Definirea acestor evenimente şi scrierea interfeţei cu un sistem grafic în termeni de evenimente fizice asigură o bază solidă pentru portabilitate.

La al doilea nivel mesajele input sunt definite în termeni de evenimente abstracte. Pentru fiecare eveniment fizic există un corespondent abstract, de ex. down – apăsarea unui buton, key – apăsarea unei taste. Alte evenimente abstracte pot fi definite pentru compunerea unor evenimente fizice (de ex. click pe butonul unui mouse). În acest fel se pot suplini (simula) evenimente fizice care nu au corespondent în sistemul Windows utilizat.

Al treilea nivel de abstractizare conţine un set de instrumente de interacţiune elementare. Exemple de asemenea instrumente de interacţiune sunt butoane, meniuri, valuator, locator. Este de remarcat utilizarea în acest model a unor instrumente definite în termeni de dizpozitive logice de intrare, fapt care conferă o compatibilitate cu alte modele input (GKS, PHIGS).

Modelul de interacţiune stratificat, definit de Laffra şi van den Boss, extinde gama de dispozitive logice de intrare la instrumente de interacţiune cum ar fi triunghi, poligon, polilinie, primitivă generalizată. Această abordare conduce la o legare mai bună a inputului de output prin definirea unor unităţi de interacţiune.

Integrarea dintre input şi output face necesară o distincţie între instrumente interacţiune ca entităţi grafice, care înglobează funcţionalitate input şi output şi obiecte de interacţiune, care sunt create de către sistem (predefinite).

Diferenţa faţă de obiectele de interacţiune definite de sistemele UIMS constă în faptul că unităţile de intrare/ieşire nu sunt numai o proiecţie în interfaţă cu a unor obiecte conceptuale definite de aplicaţie, ci chiar obiectele aplicaţiei.

Această abordare ridică însă unele probleme.

Este necesară o corelare a ecoului, produs de procesul de interacţiune ca feedback pentru operator, cu output-ul care va fi generat; o soluţie ar fi interacţiunea în cadrul unei ferestre de dialog, care să permită şi definirea atributelor.

Este relativ simplă o mapare a primitivelor de ieşire, suportate de nucleu pe instrumente de intrare/ieşire; problema se complică pe măsură ce se avansează pe o ierarhie semantică.

Legatura directă cu informaţia aplicaţiei afectează separaţia dintre informaţia grafică, (în standardul GKS încapsulată în nucleu) şi restul datelor aferente unui obiect; cerinţele de procesare ale celor două categorii de informaţie sunt diferite.

De asemenea, o clasificare utilă pentru interacţiune poate fi diferită de cea utilă pentru generare output şi stocarea informatiei. Astfel de primitive ca triunghi, poligon, pătrat fac parte din categoria unor primitive de suprafaţă, iar polilinia este o primitivă de contur, deşi pentru toate este esenţială introducerea unui vector de poziţii. Faţă de cerinţele input-ului, restul aspectelor sunt atribute care s-ar putea trata la nivelul interacţiunii prin intermediul unui tip de invitaţie/ecou.

Problema realizării de unităţi de intrare/ieşire are sens numai pentru entităţi grafice şi dispozitive logice de intrare care permit interacţiunea cu acestea. În acest sens, primitivele enumerate se pot mapa convenabil pe dispozitive logice de intrare din clasa *stroke*.

Se observă că, în acest caz, clasificarea propusă de standard nu mai este satisfăcătoare. O clasificare adecvată ar trebui să cuprindă două grupe distincte:

- dispozitive logice de intrare destinate introducerii şi editării de entităţi grafice, corespunzător claselor *locator, stroke* şi *pick*;
- dispozitive logice de intrare destinate introducerii şi editării de entităţi alfanumerice, corespunzător claselor *valuator, choice* şi *string*.

Pentru a doua categorie, mediile de dezvoltare pentru interfeţe grafice furnizează un set bogat şi standardizat de obiecte grafice de interacţiune: butoane, meniuri, casete de dialog.

6. Cadrul general de interacţiune

Descrierea modelului de interacţiune om-calculator necesită reluarea discuţiei asupra cadrului general de interacţiune. Aşa cum s-a arătat, în contextul ciclului de execuţie-evaluare, extins de Abowd şi Beale la un cadru general de interacţiune, sfera dialogului cuprinde articularea comenzilor într-un limbaj input şi execuţia acestora de către sistem.

Din punctul de vedere al utilizatorului se disting:

- acţiuni (elementare) ca: apăsare/eliberare buton mouse, apăsare/eliberare tastă, mişcare mouse; corespund nivelului lexical al limbajului de comunicare;
- operaţii, realizate printr-o succesiune (logică) de acţiuni elementare asupra dispozitivului, cum sunt: click, dublu-click pe buton mouse, click pe tastă, apăsare continuă pe buton mouse, mişcare mouse ţinând un buton apăsat; corespund nivelurilor lexical şi sintactic;
- activităţi care realizează parţial sau integral intenţia, reprezentând operaţii şi/sau acţiuni, având o anumită semnificaţie în derularea interacţiunii, cum sunt introducerea unui punct, introducerea unui şir de caractere, anularea efectului unor operaţii, părăsirea interacţiunii, deplasarea unui obiect; corespund nivelurilor sintactic şi semantic.

Distincţia între nivelurile de proiectare (lexical, sintactic şi semantic), definite în [4], depinde de contextul unei interacţiuni. Astfel, introducerea unui punct poate constitui un pas în introducerea unui poligon sau a unei polilinii; în acest caz are o conotaţie sintactică.

Procesul de interacţiune este iniţiat de către utilizator printr-o comandă. Această comandă (de ex. selectarea unui instrument dintr-un meniu sau o paletă) poate fi văzută, atât independent, având rolul de a pune sistemul într-o anumită stare (activare proces de interacţiune), fie (mai degrabă, pentru utilizator) ca pas în interacţiune. Nivelul este sintactic, în primul caz, şi semantic, în al doilea.

La nivelul sistemului acţiunile, operaţiile şi comenzile, articulate de utilizator într-un limbaj input, sunt transformate de către interfaţă în evenimente, astfel:

- evenimente fizice, corespunzător acţiunilor elementare, care surprind schimbarea de stare a unui dispozitiv;
- evenimente abstracte, obţinute prin abstractizarea şi/sau compunerea de evenimente fizice; corespund operaţiilor efectuate de utilizator;
- evenimente semantice, rezultate ca urmare a evenimentelor fizice şi/sau abstracte, evaluate în raport cu starea sistemului.

Compunerea evenimentelor fizice ridică unele probleme. Astfel, este necesară aşteptarea producerii mai multor evenimente, pentru a obţine succesiunea necesară. Acest fapt amână livrarea primului eveniment şi complică interacţiunea cu un control sintactic al succesiunii evenimentelor elementare.

De asemenea, evenimentele abstracte din categoria Click, DubluClick, prezintă riscul unor erori din partea utilizatorului, care poate provoca un DubluClick nedorit sau poate furniza două Click-uri în locul unui DubluClick.

Un alt risc al compunerii evenimentelor elementare îl constituie faptul că, în procesul compunerii, evenimentele elementare sunt consumate, devenind astfel inaccesibile programului de aplicaţie. Din aceste motive este de dorit ca implementarea să asigure flexibilitate la nivelul specificării interacţiunii, lăsând programatorului de aplicaţie libertatea de a alege setul de evenimente abstracte dorit.

Avantajul introducerii evenimentelor abstracte îl constituie posibilitatea de a standardiza interfaţă, la nivelul procedurii care tratează evenimentele trimise de managerul de evenimente, (platformele hard-soft pot furniza game diferite de evenimente abstracte). În [9] se arată că în acest fel, se obţine o portabilitate faţă de un sistem de tip Windows.

Simpla abstractizare a evenimentelor fizice nu surprinde decât acţiunile utilizatorului, fără a ţine seama de schimbările de stare ale sistemului, survenite în urma unor acţiuni precedente. Proiectarea interacţiunii necesită o tratare mai generală, la nivel semantic.

Definirea evenimentelor semantice este necesară şi din alt motiv. Este dificilă şi nerecomandabilă o definire numai pe baza evenimentelor fizice, în cadrul aceluiaşi instrument de interacţiune, întrucât există operaţii comune majorităţii proceselor: anulare/confirmare comandă, undo, care pot fi implementate mai bine cu alte instrumente de interacţiune (meniu de editare, acceleratori).

Ecoul furnizat de sistem în cazul unor comenzi (de ex. conectarea prin segmente a unei succesiuni de puncte) are o conotaţie semantică, fapt care nu permite tratarea acestuia la nivelul managerului de evenimente. Din acest motiv este necesară introducerea de evenimente semantice şi de o stratificare a interacţiunii.

7. Modelul de interacţiune grafică

Modelul propus defineşte o ierarhie de instrumente de interacţiune, pe patru niveluri, având la nivelul superior un manager de evenimente abstracte, la nivelul următor clasele de input grafic LOCATOR şi STROKE, iar la nivelurile inferioare instrumentele de interacţiune. Este o ierarhie de generalizare / specializare, în care instrumentele moştenesc o structură generală de interacţiune şi definesc comportamentul specific.

Interacţiunea om-calculator are loc în cadrul ferestrei aplicaţiei, care constituie un obiect definit de sistemul Windows. În figura 3, este prezentată ierarhia instrumentelor de interacţiune. Interfaţa utilizator cuprinde, atât instrumentele definite de model, cât şi instrumente de interacţiune predefinite de către sistemul gazdă.

Includerea nivelului intermediar, definit de clasele PolylineInput şi FillAreaInput, are rolul de a facilita o legătură mai bună a input-ului cu output-ul grafic, cu efecte pozitive asupra consistenţei interacţiunii om-calculator.

Managerul de evenimente abstracte conţine două componente, care au ca funcţii:

- translatarea/compunerea evenimentelor fizice recepţionate de fereastra aplicaţiei în evenimente abstracte, cu ajutorul cărora se poate modela interacţiunea grafică;
- procesarea evenimentelor abstracte descrise în termeni de ecou, output şi metoda de interacţiune.

Prima componentă, care translatează evenimentele fizice în evenimente abstracte, realizează această funcţie în doi paşi. În primul pas, sunt furnizate evenimente abstracte de uz general, a căror procesare este asigurată de managerul de evenimente. În pasul al doilea, se furnizează evenimentele care pot fi obţinute printr-o translatare specifică, care este implementată la nivelul instrumentelor de interacţiune specializate. Managerul de evenimente furnizează un set predefinit de evenimente abstracte.

Această structură stratificată permite o flexibilitate sporită utilă în extinderea şi adaptarea instrumentelor pentru a răspunde unor cerinţe specifice. Totodată, în acest fel este posibilă şi o particularizare (individualizare) a interfeţei pentru a răspunde abilităţilor şi/sau preferinţelor operatorului.

A doua componentă are rolul de selector al metodei de interacţiune specifice instrumentului grafic. Selecţia se realizează la momentul execuţiei, prin mecanismul de legare ulterioară (late binding) furnizat de limbajul de programare orientat obiect.

Structura generală de interacţiune, furnizată de aceste componente, permite o definire independentă a instrumentelor de interacţiune de pe nivelul următor, asigurând extensibilitatea necesară dezvoltărilor ulterioare.

Totodată, structura modulară a managerului de evenimente abstracte permite o extensie a gamei de evenimente abstracte cu evenimente semantice, necesare definirii unor procese de interacţiune complexe.

Specializarea instrumentelor se face în cadrul a două clase, PolylineInput şi FillAreaInput, care

permit o definire convenabilă a două clase de primitive output, de contur şi de suprafaţă, cu eventuale specializări.

interacţiuni complexe cere un efort mare de reprezentare şi este dificil de urmărit.

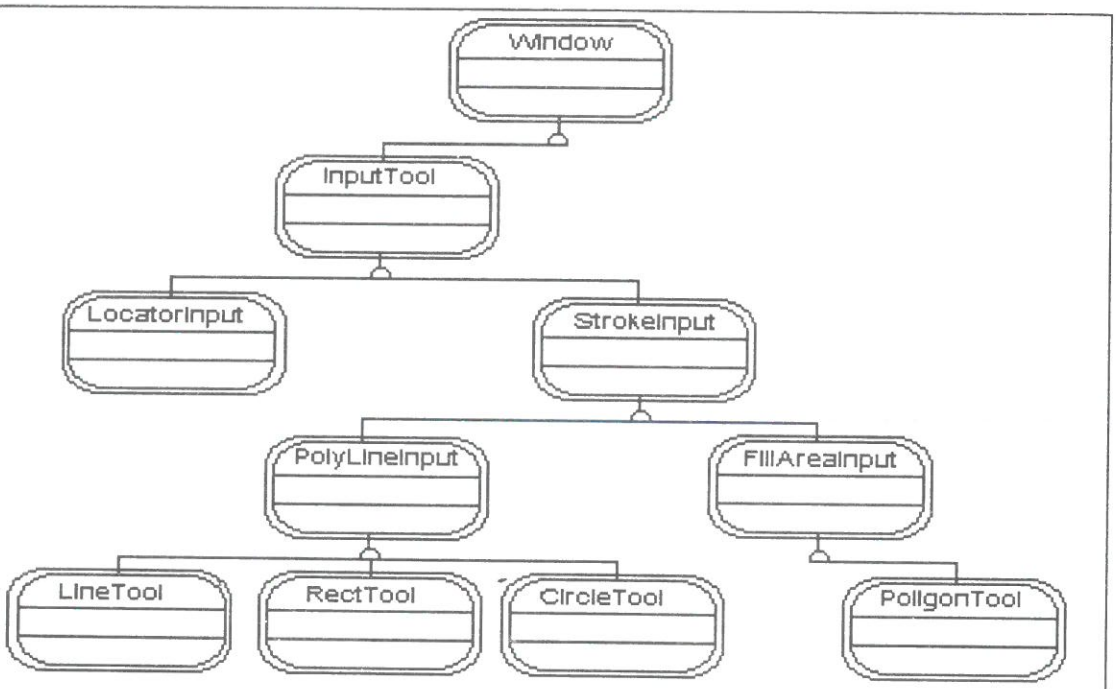

Figura 3. Ierarhia instrumentelor de interacţiune

La nivelul acestor clase se pot specifica atributele primitivelor output. Prin intermediul unor instrumente de interacţiune predefinite, din clasa *choice*, furnizate de sistemul de dezvoltare gazdă, este posibilă implementarea selecţiei atributelor ca proces de interacţiune separat. Aceasta face posibilă selecţia dinamică a atributelor la momentul editării primitivei, asigurând, atât o definire completă, cât şi un feedback rapid pentru operator.

8. Descrierea interacţiunii

Descrierea interacţiunii este necesară pentru proiectarea dialogului dintre utilizator şi sistem. În proiectul unei interfeţe, dialogul determină structura conversaţiei dintre om şi computer.

Pentru descrierea dialogului se utilizează notaţii specifice, care pot fi diagramatice, având avantajul unei înţelegeri rapide şi textuale (lingvistice), oferind facilităţi pentru analiza formală. Descrierile formale pot fi utilizate pentru identificarea acţiunilor inconsistente, dificultăţi în anularea comenzilor (undo), lipsa unor elemente şi identificarea unor posibile erori. O perspectivă asupra notaţiilor pentru dialog este prezentată în [3].

Notaţiile diagramatice au unele dezavantaje, care fac dificilă utilizarea. Astfel, descrierea unor

Între descrierile textuale cel mai frecvent utilizate, se pot menţiona notaţia BNF (Bachus-Naur Form), notaţia bazată pe reguli de producţie, algebre de evenimente. Notaţia BNF are dezavantajul unei reprezentări mai dificile a concurenţei şi a unor comenzi particulare (cum este Escape). Este mai utilă în descrierea unui model al utilizatorului decât în descrierea dialogului.

În cadrul modelului experimental, descrierea interacţiunii se face prin intermediul evenimentelor fizice şi abstracte. O metodă de interacţiune poate fi astfel definită printr-un set de evenimente fizice relevante, un set de relaţii de translatare a acestora în evenimente abstracte şi un set de evenimente abstracte având asociată o secvenţă de procesare.

Managerul de evenimente abstracte are sarcina de a transforma un eveniment fizic în eveniment abstract, funcţie de starea instrumentului de interacţiune. Secvenţa de procesare asociată include actualizarea stării dispozitivului.

În acest scop, poate fi utilizată notaţia pentru definirea regulilor de producţie, orientată pe evenimente. Forma generală pentru această notaţie ar fi:

EVENIMENT : condiţie $\rightarrow$ *Acţiune*

Un exemplu de descriere a setului de relaţii de translatare a evenimentelor fizice în evenimente abstracte, în cazul interacţiunii grafice pentru introducerea unei polilinii ar fi:

CLICK-LB : NoEvent $\rightarrow$ *StartEvent*

CLICK-LB : StartEvent $\rightarrow$ *PointEvent*

CLICK-LB : PointEvent $\rightarrow$ *PointEvent*

CLICK-RB : PointEvent $\rightarrow$ *StopEvent*

În acest caz, CLICK-LB înseamnă click pe butonul stânga mouse, iar CLICK-RB înseamna click pe butonul dreapta mouse. Starea procesului de interacţiune este exprimată în termeni de evenimente abstracte.

În continuare, pentru descrierea acţiunii asociate se poate utiliza o notaţie mai simplă, orientată pe stări, având forma generală:

condiţie $\rightarrow$ *Acţiune*

În acest fel, acţiunile din exemplul considerat, care trebuie executate de către sistem la apariţia unui eveniment, se descriu astfel:

StartEvent $\rightarrow$ *<EchoON> <GetPoint>*

PointEvent $\rightarrow$ *<GetPoint> <DrawEcho>*

StopEvent $\rightarrow$ *<Draw> <EchoOFF>*

În mod asemănător, se pot descrie toate instrumentele de interacţiune. Metoda este adecvată descrierii acţiunilor complexe, întrucât evenimentele abstracte pot avea o conotaţie semantică, reprezentând tranziţia între diferite stări în procesul de interacţiune.

9. Aspecte privind proiectarea aplicaţiilor

Orientarea pe evenimente determinată de mediul de dezvoltare gazdă conduce la o distribuire a componentei input, deci a nucleului grafic, în cadrul aplicaţiei. Acest fapt are efecte pozitive asupra flexibilităţii şi utilizării componentelor de dialog predefinite, dar afectează separaţia dintre aplicaţie şi nucleul grafic.

Aşa cum s-a aratat, aplicaţiile realizate sub sisteme de tip Windows, primesc mesajele de notificare a evenimentelor fizice prin intermediul unei proceduri de tip "callback", asociată ferestrei aplicaţiei.

Structura unui program Windows este astfel determinată de evenimentele procesate de către acesta, fapt care nu creează un cadru favorabil structurării. O critică adresată acestor sisteme, definite generic toolkit-uri, este ca, datorită faptului că nu îşi asumă o formă de organizare, nu furnizează o arhitectură pentru dezvoltarea de aplicaţii [2].

Din acest motiv, este necesară realizarea unui strat intermediar, care să permită o ordonare a inputului şi o agregare a mesajelor în entităţi input de nivel mai înalt. Este şi cerinţa formulată în modelul de referinţă [7] care urmăreşte fluxul informaţiei grafice, pe cele două direcţii - output şi input, în cadrul unor niveluri specifice, denumite medii.

Un alt aspect privind proiectarea aplicaţiilor de grafică pe calculator în context Windows este legat de iniţiativa dialogului. Orientarea pe evenimente a mediului de dezvoltare gazdă oferă maximum de libertate operatorului care deţine controlul dialogului. Acest aspect conduce la necesitatea reconsiderării modurilor de input grafic.

Modelul dispozitivului logic de intrare, prezentat anterior, defineşte trei moduri de intrare: la cerere, prin eşantionare şi prin evenimente. În raport cu modurile de intrare implementate, standardul GKS defineşte trei niveluri de implementare: fără input, input la cerere şi input complet.

Problemele dificile de implementare sunt legate de tratarea evenimentelor. Este de menţionat că, deşi există o separaţie între aplicaţie şi nucleu, programarea în moduri de intrare este caracteristică aplicaţiilor cu buclă internă. Funcţionalitatea interfeţei este definită astfel încât să asigure un control complet al programului de aplicaţie. Evenimentele care apar în cadrul unui input asincron, pentru un dispozitiv logic în modul de intrare EVENT, sunt colectate într-o coadă de evenimente pentru procesare ulterioară.

Deşi definiţia modelului este riguroasă, implementarea are un efect însemnat asupra facilităţilor de dezvoltare. Este important de menţionat că standardul specifică funcţiile de interfaţă cu aplicaţia şi modelul de procesare, lasând libertate implementatorului în realizarea interfeţei cu dispozitivul. Din acest punct de vedere, mediul Windows oferă un set foarte bogat de componente de dialog, standard "de facto", greu de asigurat de către o implementare de standard GKS sau PHIGS.

Efortul de includere în implementare a unor noi facilităţi poate fi însemnat, comparabil cu avantajele obţinute prin independenţa aplicaţiilor faţă de dispozitiv asigurată de către nucleu. Din aceste motive, definirea unor instrumente de interacţiune pe baza componentelor de dialog ale unui mediu de dezvoltare gazdă pare, în prezent o soluţie mai realistă.

Implementarea modurilor de intrare, în cazul unui instrumentar de interacţiune sub Windows, este preferabil să fie lăsată la nivelul aplicaţiei. Un prim motiv ar fi orientarea pe evenimente a unei aplicaţii Windows, deci o transparenţă faţă de mediul gazdă. Al doilea motiv este legat de faptul că, utilizarea unui instrument în mod REQUEST este mai simplă,

revenind la a activa un singur proces de interacţiune, la un moment dat.

Este de menţionat faptul că input-ul în mod REQUEST este caracteristic interacţiunii în care iniţiativa aparţine programului. Funcţiile care actionează în acest mod preiau valoarea introdusă de operator şi reţinută în structura de date a dispozitivului logic şi o furnizează programului de aplicaţie. În cazul utilizării unor instrumente de interacţiune, acestea înglobează funcţionalitate input şi output, degrevând astfel aplicaţia de un transfer de informaţie.

În ceea ce priveşte modul SAMPLE, implementarea este mult mai simplă (se citeşte starea dispozitivului la un moment dat) decât utilizarea de către aplicaţie. În acelaşi timp, o serie de facilităţi care ar necesită acest mod de intrare pot fi asigurate direct de metoda de descriere a interacţiunii, care este implementată la nivelul instrumentului.

10. Implementarea modelului experimental

În cadrul procedurii asociate ferestrei, sunt implementate următoarele funcţii:

- tratarea mesajului Windows de creare a ferestrei, WM_CREATE, care cuprinde iniţializările necesare, şi a mesajului WM_DESTROY, de închidere a ferestrei;
- tratarea mesajelor Windows de modificare a dimensiunii ferestrei, WM_SIZE şi de mutare a ferestrei, WM_MOVE;
- tratarea mesajelor WM_COMMAND de selectare a unei funcţii din meniu;
- tratarea mesajului WM_INITPOPUP, de activare a unui submeniu, în vederea marcării articolului selectat;
- preluarea mesajelor Windows şi expedierea acestora către managerul de evenimente abstracte, pentru transformare şi procesare.

Funcţiile aplicaţiei care implementează modelul experimental sunt descrise în cadrul unui meniu, definit ca resursă. La selectarea unui instrument grafic, se iniţializează şi se activează procesul de interacţiune respectiv.

În afara instrumentelor de interacţiune grafică, din meniu se mai pot apela funcţiile de selecţie a tipului de cursor grafic, precum şi funcţiile de selecţie a atributelor. Ambele categorii de funcţii se pot apela independent de procesul de interacţiune în curs.

Managerul de evenimente abstracte este implementat ca o clasă de bază, din care pot fi derivate clase de obiecte de interacţiune. Cuprinde două componente principale, care asigură funcţiile de transformare a evenimentelor fizice în evenimente abstracte (GetAbsEvent, MapEvent) şi de procesare a evenimentelor abstracte (ProcessEvent). În afara acestora, mai cuprinde o funcţie de testare a poziţiei mouse-ului faţă de aria client (TestInClientArea).

În afara componentelor funcţionale ale managerului de evenimente abstracte, modulul cuprinde şi funcţii care asigură gestiunea cursorului grafic. În afara tipurilor de cursor grafic furnizate de sistemul Windows (săgeată, cruce), în cadrul modelului experimental a fost implementat şi un cursor grafic de tip *cross-hair*, utilizat exclusiv în aria client.

Mesajele transmise ca urmare a acţiunii mouse-ului sunt tratate de managerul de evenimente numai în cadrul ariei client a ferestrei aplicaţiei, destinată interacţiunii grafice. Celelalte mesaje de la mouse sunt returnate procedurii asociate ferestrei.

În cadrul componentei de procesare a evenimentelor abstracte este inclusă structura generală de interacţiune, care apelează procedurile de afişare în ecou, desenare şi actualizare a stării instrumentului. Acestea sunt definite ca funcţii membre virtuale, de-a lungul ierarhiei de clase.

11. Concluzii

Evoluţia cercetărilor în domeniul sistemelor grafice denotă un interes actual în modelarea sistemelor software fără a pierde avantajele obţinute în ceea ce priveşte calitatea dialogului om-calculator.

Orientarea către cerinţele utilizatorului final a complicat structurile de interacţiune, restrânse la un set de dispozitive logice de intrare, din raţiuni de standardizare a formei input. Din această perspectivă, clasificarea propusă de standardele ISO pare rigidă în prezent, îngustând posibilitatile de modelare a interacţiunii.

Obiectivul acestei lucrări l-a constituit elaborarea unui model de interacţiune grafică orientat obiect, care să extindă conceptul de dispozitiv logic de intrare la nivelul cerinţelor actuale. Întrucât mediul de dezvoltare ales este Microsoft Windows, lucrarea a fost axată pe implementarea claselor de intrări *locator* şi *stroke*, pentru care sistemul gazdă nu furnizează suport în programarea aplicaţiilor.

Descrierea interacţiunii se face prin intermediul evenimentelor fizice şi abstracte. O metodă de interacţiune poate fi astfel definită printr-un set de evenimente fizice relevante, un set de relaţii de translatare a acestora în evenimente abstracte şi un set de evenimente abstracte, având asociată o secvenţă de procesare.

Structura generală de interacţiune, furnizată de componentele managerului de evenimente abstracte permite o definire independentă a instrumentelor de interacţiune de pe nivelul următor, asigurând extensibilitatea necesară dezvoltărilor ulterioare.

Bibliografie

1. **CAROLL, J.M., KERLOG, W.A., ROSSON, N.B.:** The Task - Artefact Cycle. In: J.M.Carroll (Ed.), Designing Interaction - Psychology at the Human-Computer Interface, Cambridge University Press, 1991.
2. **COCKTON, G.:** The Architectural Base of Design Reuse. În: D.A.Duce, M.R.Gomes, F.R.A.Hopgood and J.R.Lee (Eds.), User Interface Management and Design, Springer Verlag, 1991.
3. **DIX, A., FINLAY, J., ABOWD, G., BEALE, R.:** Human-Computer Interaction, Prentice Hall, 1993.
4. **FOLEY, J.D., vanDAM, A.:** Fundamentals of Interactive Computer Graphics, Addison-Wesley, Reading, Massachusetts, 1984.
5. **ISO 7942/1985:** Information processing Systems - Computer graphics - Graphical Kernel System (GKS) Functional description.
6. **ISO/IEC 9592-1,2/1989:** Information processing systems - Computer graphics - Programmer's Hierarchical Interactive Graphics System (PHIGS) - Functional description.
7. **ISO/IEC 11072/1992:** Information Technology - Computer graphics - Computer Graphics Reference Model (CGRM).
8. **JOHNSON, P.:** Human-Computer Interaction, McGraw Hill, 1992.
9. **LAFFRA, C., van den BOS, J.:** A Layered Object Oriented Model for Interaction. În: E.H.Blake and P.Wisskirchen (eds.), Advances in Object Oriented Graphics I, Springer Verlag, 1991.
10. **NORMAN, D. A.:** The Psychology of Everyday Things, Basic Books, 1988.