

Universitatea Babeş-Bolyai, Cluj-Napoca
Facultatea de Matematică și Informatică

Interacțiunea Om-Calculator
- suport de curs -

lect. dr. Adriana-Mihaela Guran

Cuprins

Introducere	7
1 Istoric	15
1.1 Motivație	15
1.2 Scurt istoric al domeniului	15
2 Noțiuni fundamentale	19
2.1 Motive care fac sistemele informaticice greu de folosit	19
2.2 Interacțiunea om-calculator (HCI) și ergonomia cognitivă	22
2.2.1 Model Human Processor	25
2.2.2 Auzul	30
2.2.3 Atingerea (Simțul tactil)	30
2.2.4 Mișcarea	31
2.2.5 Memoria	31
2.2.6 Gândirea	32
2.2.7 Rezolvarea de probleme	34
2.3 Utilizabilitatea sistemelor interactive	36
2.3.1 Definiții ale utilizabilității	36
2.3.2 Un model stratificat al utilizabilității	43
2.3.3 Metode de măsurare a utilizabilității	44
2.4 Interfața utilizator - mediatorul interacțiunii om-calculator	47
3 Metode de descriere a interacțiunii om-calculator	53

3.1	Notării diagramatice	54
3.1.1	Rețele stare-tranzitie	54
3.1.2	Diagrame stratificate de stări (statecharts)	60
3.1.3	Rețele Petri	62
3.2	Notării textuale	64
3.2.1	Gramatici independente de context	64
3.2.2	Formalismul bazat pe evenimente	66
3.2.3	NUAN - New User Action Notation	69
3.3	Modele ale utilizatorului în descrierea interacțiunii om-calculator	72
3.3.1	GOMS - Goals, Operators, Methods and Selection Rules	74
3.3.2	CCT - Cognitive Complexity Theory	76
3.3.3	TAG - Task-Action Grammar	77
3.4	Limbaje de marcare	79
3.4.1	UIML - User Interface Markup Language	80
3.4.2	SUNML - Simple Unified Natural Markup Language	84
4	Analiza sarcinilor în proiectarea sistemelor interactive	87
4.1	Analiza sarcinilor - noțiuni introductive	88
4.1.1	Analiza sarcinilor în proiectarea sistemelor interactive	90
4.1.2	Probleme relevante de analiza sarcinilor	95
4.2	Metode de analiza sarcinilor și notării	96
4.2.1	HTA - Hierarchical Task Analysis	96
4.2.2	GTA - Groupware Task Analysis	98
4.2.3	ConcurTaskTrees	115
4.3	Instrumente pentru analiza sarcinilor	119
4.4	Proiectarea interfeței utilizator pe baza modelelor sarcinilor	121
4.4.1	Generarea modelului dialogului pornind de la modelul sarcinilor	122

4.4.2	Generarea definiției abstracte a structurii interfeței utilizator pornind de la modelul sarcinilor	142
4.5	Metoda DUTCH de proiectare a sistemelor	144
4.6	Studiu de caz	148
5	Utilizabilitatea sistemelor interactive. Inginerie utilizabilității	153
6	Şabloane de proiectare a interfețelor utilizator	155
6.1	Şabloane de proiectare în HCI	156
6.2	O abordare centrată pe utilizator pentru dezvoltarea unui şablon de proiectare a site-urilor web ale unor instituţii de cultură	157
6.2.1	Analiza nevoilor utilizatorilor	157
6.2.2	Evaluarea site-urilor web ale unor teatre bine-cunoscute	160
6.3	Dezvoltarea şablonului de proiectare a site-urilor web ale unor teatre/opere	161
6.3.1	Publicul țintă	162
6.3.2	Structura generală a site-urilor web ale teatrelor .	164
6.4	Rezultatele analizei nevoilor utilizatorilor	167
6.4.1	Fidelitatea chestionarului	167
6.4.2	Concluziile chestionarului și analizei site-urilor web	172
6.5	Şablonul de proiectare a paginilor web ale teatrelor/operelor	177
7	Evaluarea utilizabilității sistemelor soft	189
7.1	Criterii pentru evaluarea interfețelor utilizator	190
7.2	Evaluarea la distanță a utilizabilității	192
7.3	O abordare bazată pe AOP pentru evaluarea automată a utilizabilității	193
7.4	O abordare bazată pe agenți în evaluarea utilizabilității sistemelor informatici	199

8 Asistenți personali ai utilizatorului	205
8.1 Modelul teoretic	206
8.2 Comportamentul agentului LIA	208
8.3 Arhitectura agentului LIA	210
8.4 Evaluare experimentală	211
8.4.1 Măsura de evaluare	212
8.4.2 Studiu de caz	213
Bibliografie	217

Introducere

Într-o lume din ce în ce mai puternic dominată de tehnologie, aproape că nu mai există nici un domeniu în care calculatoarele să nu fi pătruns. Însă nu întotdeauna penetrarea ultimelor rezultate tehnice și tehnologice înseamnă un progres din punctul de vedere al eficienței și satisfacției utilizatorilor finali. Mai mult, succesul sau eșecul unui produs depinde de satisfacția utilizatorilor, ca rezultat al sporirii eficienței acestora în realizarea sarcinilor care le revin. În domeniul dezvoltării de sisteme interactive nu există o rețetă de realizare a unui produs performant și acceptat de utilizatori. Există numeroase exemple de sisteme cu performanțe remarcabile, dar ale căror utilizatori refuză să le folosească. Explicația refuzului de a folosi aceste sisteme constă în comportamentul intimidant și confuz al acestor sisteme. Astfel de probleme se nasc din faptul că în utilizarea unui sistem informatic se întâlnesc două lumi, sau mai științific spus, două modele mentale. Avem pe de o parte modelul proiectantului asupra sistemului, un model extrem de precis, care mai este numit și *modelul conceptual* al sistemului. De cealaltă parte avem modelul mental al utilizatorului asupra sistemului. Acest model se construiește prin interacțiunea utilizatorului cu sistemul și prin preluarea cunoștințelor anterioare de utilizare a altor sisteme interactive. De obicei, modelele mentale sunt neștiințifice și instabile în timp. Problemele de utilizare a sistemelor interactive apar în momentul în care nu există suprapuneri între modelul proiectantului și modelul utilizatorului. Pentru a obține o cât mai bună potrivire între aceste modele, este nevoie

de o abordare centrată pe utilizator în proiectarea sistemelor interactive. Această abordare presupune realizarea procesului de proiectare având în centrul atenției utilizatorii și caracteristicile acestora. Astfel, abordarea procesului de proiectare de sisteme interactive devine una interdisciplinară, incluzând specialiști în ingineria soft, dar și psihologi, experți în factori umani, etnografi, etc. În ultimele decenii s-au dezvoltat discipline preocupate de aspecte ale interacțiunii dintre utilizatori și instrumentele pe care le folosesc în activitatea lor. Este vorba de Ergonomie, disciplină preocupată de adaptarea instrumentelor la om, și în special subramura acesteia numită Ergonomia Cognitivă, preocupată de studiul comportamentului uman când este mediat de instrumente cognitive, cu rol de sporire a capacitatei de procesare a informației. De asemenea, HCI¹ (Human-Computer Interaction) este o disciplină care s-a dezvoltat cu scopul studierii interacțiunii om-calculator, surprinzând aspecte tehnice, cognitive, culturale și sociale ale acesteia.

Prezenta lucrare se situează la intersecția acestor domenii, explorând subiecte aflate la limita ingineriei soft, ergonomiei cognitive, HCI și cu influențe preluate din Inteligența Artificială. Scopul cercetărilor efectuate a fost identificarea și prezentarea unor soluții de proiectare de sisteme interactive care să asigure un sprijin real utilizatorilor în realizarea sarcinilor de muncă și care să se bucure de acceptare din partea utilizatorilor, într-un singur cuvânt, proiectarea de sisteme *utilizabile*. Lucrarea este structurată în şase capitole, al căror conținut este prezentat pe scurt în cele ce urmează.

Capitolul 2 al cărții prezintă conceptele de bază care sunt folosite în cuprinsul lucrării, realizând totodată o încadrare a subiectului cărții în domeniile Human-Computer Interaction și al Ergonomiei Cognitive. Pentru că utilizabilitatea sistemelor interactive este un subiect comun de

¹ traducerea în limba română este *Interacțiunea Om-Calculator*, dar deoarece abrevierea HCI este des utilizată în literatura de specialitate, vom folosi în continuare această abreviere

interes pentru domeniile menționate anterior, sunt prezentate diversele perspective din care a fost abordată utilizabilitatea de-a lungul timpului. Deoarece utilizabilitatea este un obiectiv major în dezvoltarea de soft, în cadrul Secțiunii 2.3, ne-am oprit asupra metodelor de evaluare a utilizabilității, oferind o atenție sporită metricilor utilizabilității.

Modelarea unui sistem informatic implică folosirea de notații. Cu cât aceste notații sunt mai formale, cu atât sporește rigurozitatea modelului. În Capitolul 3 prezentăm o incursiune în domeniul metodelor de descriere a interacțiunii om-calculator. Sunt prezentate atât notații diagramatice (Secțiunea 3.1), precum rețelele stare-tranzitie, rețelele Petri sau diagramele stratificate de stări, cât și notații textuale (Secțiunea 3.2) precum limbajele bazate pe evenimente sau gramaticile independente de context. Atunci când dorim proiectarea de sisteme utilizabile, e nevoie să includem utilizatorii în procesul de proiectare. Cea mai la îndemâna metodă de a realiza acest lucru este crearea de modele ale utilizatorului. De-a lungul timpului s-au dezvoltat numeroase metode de construire a modelului utilizatorului, însă doar câteva dintre ele și-au demonstrat utilitatea. O caracteristică a acestor metode e că sunt orientate pe sarcinile pe care utilizatorii trebuie să le indeplinească. Dintre aceste metode, în cadrul Secțiunii 3.3, sunt prezentate: Hierarchical Task Analysis (HTA), Goals Operators, Methods and Selection Rules (GOMS), Cognitive Complexity Theory (CCT) și Task-Action Grammar (TAG). Progresele recente din domeniul tehnologiei permit ca aplicațiile soft să poată rula nu numai pe calculatoare, ci și pe dispozitive mobile precum PDA-uri sau telefoane mobile inteligente. Problemele care apar atunci când se proiectează sisteme care să poată rula pe diverse dispozitive sunt legate de resursele diferite de care acestea dispun. Dacă ne vom referi la aspectul de prezentare al interfeței utilizator, este suficient să ne raportăm la dimensiunea ecranului sau la modul de interacțiune (taste, touch-screen, etc.). Pornind de la aceste aspecte, s-a dezvoltat conceptul de interfețe plastice (care rulează pe dispozitive multiple). Dezvoltarea unor astfel de interfețe este

sprijinită de dezvoltarea unor limbaje de tip XML care descriu structura interfeței la un nivel abstract. De asemenea, unele limbaje dispun de modalități de specificare a comportamentului interfeței utilizator. Aceste definiții abstracte ale interfeței utilizator sunt convertite în implementări concrete folosind instrumente specifice denumite *rendere*. Din categoria limbajelor de marcăre (de tip XML) folosite pentru crearea de definiții abstracte am prezentat, în cadrul Secțiunii 3.4, limbajele User Interface Markup Language (UIML) și Simple Unified Natural Markup Language (SUNML).

Așa cum am afirmat anterior, o proiectare de succes a unui produs informatic este o proiectare centrată pe utilizator. Pentru a înțelege utilizatorul și sarcinile sale este nevoie de o abordare științifică a acestor aspecte. Astfel, în proiectarea sistemelor informatic, au fost preluate metode din psihologia muncii, precum *analiza sarcinilor*. Capitolul 4 al lucrării este dedicat analizei sarcinilor, cu accente asupra modului în care aceasta poate aduce contribuții în proiectarea sistemelor interactive. Secțiunea 4.1 cuprinde definirea unor noțiuni de bază, precum *sarcina*, *activitatea*, *analiza sarcinilor* și *modelarea sarcinilor*. Sunt prezentate de asemenea modalitățile în care analiza sarcinilor poate fi folosită în HCI: pentru descrierea situației curente a sarcinilor, pentru reproiectarea sarcinilor, respectiv pentru evaluarea interfeței utilizator. Toate aceste trei abordări sunt utilizate pe parcursul cercetărilor descrise în această lucrare. Deoarece anumite aspecte studiate de analiza sarcinilor sunt similare analizei cerințelor din ingineria soft, este prezentată o descriere comparativă a celor două abordări din punctul de vedere al scopului, rezultatelor, subiectelor de interes și principalelor obiecte de analiză. Secțiunea 4.2 este dedicată descrierii metodelor de analiza sarcinilor care au fost utilizate de-a lungul timpului în abordarea centrată pe utilizator a sistemelor informatic, dintre care menționăm: Hierarchical Task Analysis (HTA) - prima metodă de analiza sarcinilor care a fost dezvoltată, Groupware Task Analysis (GTA) - metodă de analiza

sarcinilor care pornește procesul de analiză din perspectiva grupului și ConcurTaskTrees - notație destinată modelării sarcinilor. O atenție sporită este acordată metodei de analiză GTA, care și-a dovedit utilitatea în proiectarea sistemelor interactive de-a lungul timpului, fiind inclusă în dezvoltarea unei metode de proiectare a sistemelor centrată pe utilizator, și anume Designing for Users and Tasks from Concepts to Handles (DUTCH). Această metodă este descrisă în cadrul Secțiunii 4.5. Deoarece aplicațiile soft rulează în prezent nu doar pe computere, ci și pe diverse dispozitive mobile, se dorește evitarea creării unei interfețe utilizator pentru fiecare tip de dispozitiv, disponând de resurse diferite. În acest scop se folosesc abordări orientate pe modele, care sunt transformate în implementări concrete corespunzătoare fiecărui tip de dispozitiv. Secțiunea 4.4 prezintă soluții de construire a modelului dialogului și modelului prezentării pornind de la modelul sarcinilor, asigurând astfel un demers de succes în proiectarea de produse utilizabile.

Atunci când se studiază interacțiunea om-calculator, nu ne limităm doar la aplicațiile desktop. Expansiunea web-ului face ca majoritatea utilizatorilor de calculatoare să intre în contact cu interfețe web. Cu toate că numărul de site-uri existente este impresionant, opiniiile specialiștilor relativ la calitatea interfețelor sunt defavorabile. Dorința de a furniza "rețete de succes" în proiectarea site-urilor web a dus la adoptarea abordării din ingineria soft și anume, formularea de *șabloane de proiectare*. Există dezvoltate deja câteva colecții de șabloane de proiectare destinate interfețelor utilizator, cea mai bogată fiind Amsterdam Pattern Collection. Capitolul 7 prezintă un proces de dezvoltare a unui șablon de proiectare al unui site web dedicat unui domeniu specific pornind de la nevoile utilizatorilor. Abordarea descrisă se desfășoară în trei etape succesive care presupun analiza nevoilor utilizatorilor (în urma acestui proces se va identifica ce își doresc utilizatorii, dar și modul în care își doresc să obțină informațiile). Analiza nevoilor utilizatorilor este urmată de analiza critică (evaluarea) mai multor interfețe din domeniul

vizat pentru a identifica lipsurile sau problemele de interacțiune care pot să apară. Procesul de evaluare se realizează pe un număr de interfețe determinat pe criterii etnografice - *indexul de surpriză* (sunt evaluate atâtea interfețe până când vizitarea unei noi interfețe nu mai aduce nimic nou). În urma culegerii informațiilor din pașii anteriori se trece la formularea şablonului. Secțiunea 6.3 este dedicată prezentării modului de elaborare a unui şablon de proiectare a site-urilor web ale unor instituții de cultură (teatre/opere). În cadrul acestei secțiuni sunt prezentate metodele utilizate pentru culegerea informațiilor legate de nevoile utilizatorilor, rezultatele analizei acestor date cu interpretările aferente și rezultatele evaluării a zece site-uri web ale unor instituții culturale renomate. Secțiunea 6.5 încheie capitolul prin formularea şablonului de proiectare, care utilizează structura şabloanelor din colecția Amsterdam Pattern Collection, menționând faptul că nu există încă un limbaj unanim acceptat de descriere a şabloanelor pentru interfețe utilizator.

Unul din obiectivele cercetărilor în HCI constă în asigurarea unei utilizabilități sporite a sistemelor interactive. În acest sens, este firesc să ne preocupăm de metodele de evaluare a utilizabilității, cu precădere acelea care pot fi aplicate fără o expertiză sporită. Cele mai eficiente metode de evaluare a utilizabilității sunt testelete de utilizabilitate cu utilizatori reali, dar acestea sunt foarte costisitoare. Există și alternativa consultării experților în utilizabilitate, soluție de asemenea costisitoare. Din punctul de vedere al dezvoltatorului, cea mai avantajoasă abordare este evaluarea automată a utilizabilității, bazată pe calculul unor metriki de utilizabilitate (timpul de execuție, frecvența erorilor, frecvența de folosire a unor comenzi, etc.). Evaluarea automată a utilizabilității recurge în mod implicit la jurnalizare (pentru culegerea datelor de interacțiune), dar jurnalizarea impune acces la codul sursă al sistemului, amestec între logica aplicației și codul necesar jurnalizării, iar orice modificare necesară presupune recompilarea codului sursă. Capitolul 7 al cărții este dedicat abordării pe care o propunem în privința eliminării neajunsurilor

menționate anterior. Soluția pe care o furnizăm în Secțiunea 7.3 constă în aplicarea unei paradigmă moderne de programare, și anume Programarea Orientată pe Aspecte, în scopul creării fișierelor de jurnalizare și calculului dinamic al metricilor de utilizabilitate. Astfel, codul destinat preluării informațiilor necesare calculului metricilor de utilizabilitate este grupat într-un singur modul, numit *aspect*, orice modificare se răspândește numai asupra aspectului, iar modulul destinat evaluării utilizabilității poate fi atașat sistemului sau scos din sistem fără necesitatea recompilării sistemului. În cadrul studiilor destinate evaluării automate a utilizabilității ne-am oprit atenția asupra tehnologiilor suport (destinate persoanelor cu dizabilități), deoarece acestea furnizează facilități de captare a evenimentelor dintr-o aplicație fără necesitatea jurnalizării. Un studiu comparativ privitor la aplicabilitatea celor două metode în realizarea unei evaluări automate a utilizabilității este prezentat în cadrul unei secțiuni a Capitolului 8. În încercarea de a depăși limitările impuse de o abordare cantitativă a utilizabilității (dobândită prin calculul unor metri), am propus o soluție care urmărește identificarea problemelor de utilizabilitate, într-o manieră bazată de asemenea pe analiza sarcinilor, deci centrată pe utilizator. Ideea de la care am pornit în demersurile noastre de cercetare, a fost aceea că în interacțiunea dintre utilizator și un sistem informatic se întâlnesc, în esență, două modele ale unui domeniu: modelul proiectantului, numit și model conceptual, și modelul mental al utilizatorului (sau modelul sarcinilor utilizator). Ideal, între cele două modele, ar trebui să existe o suprapunere, iar când această potrivire nu apare, ne confruntăm cu probleme de utilizabilitate. Abordarea descrisă în Secțiunea 7.4, constă în compararea celor două modele pentru a identifica punctele de neconcordanță. Modelul proiectantului este un model al sarcinilor (în contextul unei abordări centrate pe utilizator). Modelul mental al utilizatorului poate fi "reconstruit" din interacțiunea cu sistemul. Sarcina reconstruirii modelului mental al utilizatorului, precum și identificarea problemelor de utilizabilitate, sunt sarcini care pot fi dele-

gate unui agent (intelligent) care are rolul de monitorizare a interacțiunii și de comparare a celor două modele ale domeniului. Arhitectura unui astfel de agent este descrisă în secțiunea 7.4.

Progresele remarcabile realizate în domeniul Inteligenței Artificiale, și mai ales în domeniul învățării automate au creat premisele dezvoltării unui nou domeniu de cercetare, referitor la dezvoltarea de asistenți ai utilizatorului. Aceștia au rolul de a asista utilizatorul în realizarea sarcinilor sale. Cercetările noastre în acest domeniu sunt o continuare firească a abordărilor orientate pe agenți propuse în Capitolul 7. Capitolul 8 prezintă o abordare privind dezvoltarea unui agent intelligent pentru predicția comportamentului utilizatorilor. Agentul învață folosind o tehnică de învățare supervizată, iar după o perioadă de antrenare, emite predicții privind următoarea acțiune utilizator care ar trebui executată astfel încât utilizatorul să îndeplinească sarcina. Predicția se face cu o anumită probabilitate. Pentru evaluarea acurateței predicției sunt folosite două măsuri de evaluare.

Doresc să adresez mulțumiri tuturor colaboratorilor care m-au sprijinit în realizarea acestei cărți: prof. univ. dr. Horia D. Pitariu, prof. univ. dr. Militon Frențiu, prof. univ. dr. Liana Lupșa, conf. dr. Gabriela Czibula, conf. dr. Simona Motogna, lect. dr. Grigoreta Cojocar și asist. drd. Daniela Andrei.

Capitolul 1

Istoric

1.1 Motivație

Lumea în care trăim este dominată de tehnologie, iar viitorul indică o imersiune evidentă în tehnologie. Totuși, măsura în care suntem pregătiți pentru o societate digitalizată este discutabil. Acet aspect depinde modul în care aplicațiile sunt proiectate pentru a fi utilizate de către utilizatorii finali. Pentru a putea oferi experiențe de utilizare potrivite, este nevoie de înțelegerea modului în care oamenii interacționează cu tehnologia. Domeniul Interacțiunii Im-Calculator este preocupat de studiul modului în care oamenii folosesc tehnologia.

1.2 Scurt istoric al domeniului

Conceptul incepe să fie folosit la începutul anilor '80, odată cu primele preocupări legate de bunăstarea oamenilor în mediul în care lucrează. Înaintea acestui moment, în perioada celui de-al Doilea Război Mondial (1949) apare Societatea de Cercetări în Ergonomie.

Ergonomia, ca și știință este preocupată de studiul caracteristicilor fizice ale oamenilor și mașinilor și de modul în care acestea afectează

performanța. Printre fondatorii acestui domeniu se numără și celebrul arhitect Antonio Gaudi (1852-1926), care a proiectat primele modele de scaune ergonomice (vezi Figura 1.1).



Figura 1.1: Scaun ergonomic Antonio Gaudi.

Inițial, ergonomia a fost preocupată de aspectele fizice ale muncii, de adaptarea uneltelor (mașinilor) la caracteristicile fizice ale oamenilor. În timp ce activitățile de muncă încorporează tot mai multe sarcini cognitive, studiile de ergonomie încorporează tot mai multe aspecte cognitive, ducând la apariția unui nou subdomeniu, numit *ergonomie cognitivă*. Odată cu răspândirea mașinilor industriale ca și suport în munca zilnică a oamenilor s-a dezvoltat un nou domeniu de cercetare care include aspecte fizice și psihice ale interacțiunii dintre oameni și mașini, *interacțiunea om-mașină* (Man-Machine Interaction). Cu timpul, locul uneltelor (mașinilor) este să se consideră înțesă că din perspectiva interacțiunii cu calculatoarele domeniul este suficient de amplu, astfel încât să primească denumirea de Computer-Human Interaction (CHI). Ulterior, din perspectiva atenției pe care o acordă domeniul omului, nu tehnologiei, se consideră că locul conceptelor componente trebuie să schimbe, astfel încât să ajungă la denumirea de Human-Computer Interaction, cu accent astfel pe faptul că omul este în centru procesului de proiectare a tehnologiei.

In studiul HCI nu ne limităm la un utilizator care interactionează cu un calculator. În cele ce urmează furnizăm definițiile conceptelor pe care le vom folosi în cadrul acestui material.

Utilizatorul poate fi un individ, un grup de indivizi sau o mulțime de utilizatori care interactionează secvențial pentru una sau mai multe parti din proces. **Computerul** este orice tehnologie care poate însemna un calculator desktop până la un sistem la scară largă, un sistem de control al proceselor sau un sistem integrat. **Sistemul** poate să includă și părți neautomatizate (alți oameni).

Interacțiunea este orice comunicare între utilizatori și computer.

Studiul conceptelor care sunt manipulate în domeniul HCI necesită cunoștințe din domenii multiple, precum psihologia, sociologia, ingineria soft, ergonomia. Astfel, studiul domeniului HCI necesită o abordare interdisciplinară, care să cuprindă cunoștințe din cel puțin următoarele domenii:

- psihologie - pentru a putea înțelege capacitatele cognitive, percepția, rezolvarea de probleme
- ergonomie - pentru a putea analiza potrivirea fizică a sistemului cu utilizatorii
- sociologie - pentru a înțelege contextul interacțiunii, schimbările pe care le poate determina introducerea unui sistem automat;
- inginerie soft - pentru a putea proiecta și implementa efectiv sistemul
- design grafic - pentru a produce o prezentare estetică și minimalistă a sistemului.

În proiectarea unui sistem interactiv se pune întrebarea cât este știință și cât este creativitate? Similar abordărilor din arhitectură, este nevoie de un mix între rezistență și design. Nu ne-am dori să locuim

într-o casă frumoasă dar care nu ar rezista unui vânt puternic, la fel cum nu ne-am dori să locuim într-o casă foarte rezistentă, dar care are un aspect neplăcut. Proiectarea unui sistem potrivit utilizatorilor nu este o sarcină trivială deoarece pentru a proiecta ceva pentru cineva trebuie înțelese capacitatile și limitările sale. De multe ori înțelegerea unui sistem complex e dificilă, de aceea recurgem la *modele*, care sunt simplificări ale realității care păstrează însă caracteristicile esențiale ale sistemului.

HCI este un domeniu interdisciplinar, deoarece necesită înțelegerea fiecărei componente a interacțiunii: omul, calculatorul (sistemul), interacțiunea *în sine*. În plus, oamenii sunt ființe complexe, ale căror caracteristici sunt studiate de diverse științe. Astfel, studiul HCI necesită considerarea unor domenii precum: inginerie soft, psihologie, sociologie, ergonomie.

Capitolul 2

Notiuni fundamentale

2.1 Motive care fac sistemele informatice greu de folosit

1. **De-a lungul dezvoltării produsului se pune accentul pe sistem sau mașină, nu pe utilizatorul acestuia/acesteia.** Există trei componente majore în contextul performanței umane. Acestea sunt omul, contextul și activitatea. Deoarece dezvoltarea unui sistem sau produs este o încercare de îmbunătățire a performanței umane într-un anumit domeniu, proiectanții trebuie să ia în considerare toate aceste aspecte în procesul de proiectare. Toate aceste trei aspecte vor afecta rezultatul asupra performanței umane. Din nefericire, din toate aceste aspecte, proiectanții se concentrează asupra componentei activitate. Relația dintre cele trei componente este de asemenea neglijată. Există câteva explicații pentru aceste abordări neechilibrate. Prima este aceea că oamenii sunt ușor adaptabili și flexibili, astfel încât este mai ușor să se adapteze ei sistemelor, decât să adaptăm sistemele utilizatorilor. În al doilea rând, dezvoltatorii se simt mai confortabil să lucreze dintr-o perspectivă clar delimitată în ceea ce privește dezvoltarea sistemelor, decât

să trateze aspecte ambigue legate de ființele umane. În al treilea rând, dezvoltatorii au fost anagajați de-a lungul timpului pentru potențialul de a rezolva probleme tehnice, nu pentru abilitățile de comunicare interpersonală. Cel mai important aspect care duce la neglijarea factorilor umani în dezvoltarea sistemelor în trecut este acela că se dezvoltau sisteme pentru utilizatori din rândul dezvoltatorilor.

2. **Odată cu penetrarea tehnologiei în masă, audiența țintă s-a schimbat și continuă să se schimbe. Organizațiile dezvoltatoare au reacționat cu încetineală la această evoluție.** Utilizatorii originali ai sistemelor informaticе dețineau cunoștințe de expert relativ la calculatoare și dispozitive mecanice, pasiune pentru tehnologie, mândria de a repara și rezolva orice problemă. Situația s-a schimbat radical în prezent. Dacă în trecut era un lucru neobișnuit pentru o persoană fără pregătire tehnică să folosească calculatoare, astăzi majoritatea persoanelor folosesc astfel de produse fie la locul de muncă, fie în viața privată. Cu toate acestea, utilizatorii de acum au puține cunoștințe tehnice despre calculatoare, puțină răbdare și așteptări complet diferite.
3. **Proiectarea de sisteme utilizabile este dificil de realizat, necesită un efort greu de prevăzut și este tratată în multe organizații ca o chestiune de "bun-simt".** Într-un studiu de cercetare realizat acum câțiva ani, Gould and Lewis au concluzionat că proiectanții au noțiuni eronate despre ceea ce înseamnă proiectarea centrată pe utilizator și, în consecință, asupra utilizabilității. Principiile utilizabilității sunt încă puțin cunoscute și e nevoie de educare, asistență și o abordare sistematică în aplicarea "bunuluisimt" în proiectarea sistemelor interactive.
4. **Organizațiile folosesc echipe foarte specializate în proiectarea și dezvoltarea sistemelor, dar esuează în integrarea**

acestora. În scopul îmbunătățirii eficienței, organizațiile au divizat procesul de dezvoltare a softului în dezvoltarea independentă a mai multor componente sistem (de exemplu interfața, sistemul de ajutor, documentația tehnică). În mod tipic, aceste componente sunt dezvoltate de indivizi/echipe diferite, iar dificultățile apar în momentul în care se încearcă integrarea acestora, datorită lipsei de comunicare.

5. **Proiectarea interfeței utilizator și implementarea interfeței utilizator sunt activități diferite, care necesită aptitudini diferite.** Actual, este nevoie de concentrare asupra părții de proiectare, în timp ce majoritatea dezvoltatorilor dețin rechizitele pentru implementarea tehnică.

Proiectarea în acest caz se referă la modul în care se realizează comunicarea, în timp ce implementarea se referă la modul în care produsul funcționează. Odată cu apariția limbajelor de proiectare orientate pe obiecte și a unor instrumente de generare automată a codului, provocarea implementării tehnice a scăzut. Provocarea lansată însă de proiectarea de sisteme a crescut covârșitor datorită nevoii de abordare a unei populații mai largi de utilizatori și mai puțin ”instruiți”, cu așteptări crescând relativ la ușurința de folosire a sistemelor.

Dezvoltatorii omit uneori faptul că nu dezvoltă produse pentru sine, cine pentru o populație mai largă, și mai ales faptul că proiectează relația dintre om și produsul informatic. În proiectarea acestei relații, proiectanții trebuie să permită utilizatorilor să se concentreze asupra sarcinilor și nu asupra metodelor de realizare a sarcinilor. De aceea este nevoie de metode și tehnici care să-i sprijine pe proiectanți să schimbe modul în care privesc și proiectează produsele, metode care lucrează din-spre exterior spre interior, de la nevoile și abilitățile utilizatorului la o eventuală implementare. Numele dat acestei abordări este **proiectare**

centrată pe utilizator (User Centered Design - UCD) [59]. Proiectarea centrată pe utilizator reprezintă tehnicele, procesele, metodele și procedurile de proiectare a sistemelor și produselor utilizabile, dar și o filosofie care așează utilizatorul în centrul procesului de proiectare [59]. Woodson [82] definește proiectarea centrată pe utilizator ca fiind ”obișnuința de a proiecta produse pe care utilizatorii să le poată folosi pentru a efectua operații, servicii și sarcini de sprijin cu minimul de stres și maximul de eficiență”. Rubin [59] enumerează trei principii ale proiectării centrate pe utilizator:

- Concentrarea atenției din fazele inițiale ale proiectării asupra utilizatorilor și sarcinilor - se cere o abordare sistematică, structurată a colectării datelor de la și despre utilizatori. Proiectanții trebuie să beneficieze de pregătire privind realizarea interviurilor din partea unor experți înainte de a începe o sesiune de culegere de date.
- Măsurători empirice ale folosirii produsului - se pune accentul pe măsurarea ușurinței de învățare și utilizare încă de la începutul procesului de proiectare, prin dezvoltarea și testarea de prototipuri cu utilizatori reali.
- Proiectare iterativă indiferent dacă produsul este proiectat, modificat și testat în mod repetat - proiectarea iterativă permite regândirea și reconstruirea unei proiectări prin testarea *timpurie* a modelelor conceptuale și a ideilor de proiectare.

2.2 Interacțiunea om-calculator (HCI) și ergonomia cognitivă

Interacțiunea om-calculator sau interacțiunea om-mașină face obiectul de studiu al unei discipline care a apărut în jurul anilor '80

și care a fost numită simplu și sugestiv Human-Computer Interaction, sau și mai simplu HCI. Obiectivul cercetărilor HCI este acela de a îmbunătăți interacțiunea dintre utilizatori și sistemele informatiche. HCI își distribuie subiectele de interes în cinci domenii:

- *Tehnicile de interacțiune* - se studiază modul în care tehnologiile de intrare-iesire afectează interacțiunea; scopul e acela de a dezvolta noi tehnici de interacțiune și de a sugera unde și în ce condiții pot fi folosite în modul cel mai potrivit.
- *Modelarea utilizatorului* - furnizarea de teorii și instrumente pentru modelarea cunoștințelor pe care un utilizator le posedă și le folosește pentru a îndeplini o sarcină; scopul e acela de a da proiectanților posibilitatea să construiască sisteme cât mai utilizabile, făcând explicit modelul utilizatorului asupra sarcinilor și sistemului.
- *Potrivirea cu sarcina* - se încearcă găsirea mijloacelor de determinare a nevoilor utilizatorului și potrivirea funcțiilor pe care sistemul le oferă cu nevoie utilizatorului; scopul este de a dezvolta metode prin care să se determine nevoile utilizatorului, asigurând astfel faptul că sistemul oferă funcții de care utilizatorul are nevoie și informații pe care utilizatorul le cere (în formatul în care acesta și le dorește), fără un efort prea mare din partea utilizatorului;
- *Specificațiile și proiectarea sistemelor interactive* - se studiază procesul de proiectare; scopul este acela de a oferi sugestii de îmbunătățire a procesului de proiectare, luând în considerare în mai mare măsură utilizatorul (determinarea unei metode de trecere de la proiectarea orientată pe sistem la proiectarea orientată pe utilizator).
- *Impactul sistemelor informaticice asupra organizațiilor, individelor și grupurilor* - examinarea impactului unui nou sistem

asupra rolurilor utilizatorilor individuali și asupra grupurilor de utilizatori; obiectivul este de a sugera tehnici de proiectare și implementare care ar putea preveni probleme precum decalarea sarcinilor sau conflictele între grupuri.

Ergonomia(psihologia inginerească) este o disciplină preocupată de studiul optimizării și adaptării mașinilor, dispozitivelor tehnice și aparatelor de măsură la om [40]. Există două aspecte distințe ale ergonomiei: *ergonomia fizică*, preocupată de adaptarea instrumentelor la caracteristicile fizice și morfologice ale utilizatorilor și *ergonomia mentală* care se referă la adaptarea instrumentelor la procesele cognitive ale utilizatorului.

Ergonomia cognitivă se ocupă cu studiul comportamentului uman în contextul în care acesta este mediat de aparate și instrumente cognitive (cu rol de optimizare a posibilităților umane de procesare a informației) [53]. Obiectivul urmărit este adaptarea și utilizarea instrumentelor cognitive astfel încât să sporească nivelul prelucrării informațiilor umane în sensul creșterii eficienței și satisfacției și reducerii erorilor și accidentelor [40]. Informațiile furnizate de cercetătorii aparținând celor două discipline prezentate succint sunt de mare folos și cu o reală aplicabilitate în proiectarea sistemelor informatice, deoarece noile tendințe impun proiectarea centrată pe utilizatori, iar utilizatorii pot fi cunoașcuți prin intermediul studiilor efectuate de psihologi, etnografi, specialiști în ergonomie sau factori umani. Preocuparea de a oferi utilizatorului un sistem care să îl ajute în realizarea activităților zilnice, pe care să îl folosească în mod eficient și cu satisfacție a fost înglobată în asigurarea *utilizabilității* sistemelor informatice. Acesta este obiectivul central al tuturor studiilor de HCI și ergonomie cognitivă, iar următoarea secțiune va trata subiectul utilizabilității aşa cum a fost el surprins de-a lungul timpului din diverse perspective.

În cele ce urmează prezentăm abordările esențiale în ceea ce privește modelarea omului în interacțiune.

2.2.1 Model Human Processor

Prima abordare în modelarea omului în contextul interacțiunii este MODEL Human Processor [14], care identifică trei subsisteme esențiale:

- Perceptiv – tratează stimulii din mediul înconjurător
- Motor – controlează acțiunile fizice
- Cognitiv – realizează procesările necesare pentru a le conecta pe cele două anterior menționate.

Fiecare subsistem are propriul procesor și memorie

In acest model, informația este recepționată, apoi memorată și procesată, iar apoi rezultatele sunt transmise spre exterior.

Receptarea informațiilor se realizează prin canalele de intrare, răspunsurile sunt transmise prin canalele de ieșire, iar informația receptată este memorată sau /ci procesată.Trebuie să studiem:

- Canalele de I/O
- Memoria
- Procesarea (rezolvarea de probleme, învățare, erori!!!)

Canalele de I/O

Interacțiunea se realizează prin informație care se primește/transmite. În interacțiunea om-calculator informația primită este output-ul de la calculator și este recepționată prin simțuri. Dintre toate simțurile de care dispunem: văzul, auzul,

atingerea, gustul, miroslul, în interacțiune sunt exploataate primele trei.

Informatia transmisă de către utilizatori este inputul pentru calculator. Toate răspunsurile transmise de către utilizatori sunt răspunsuri motorii realizate prin efectori (degete, ochi, cap, sistem vocal).

În continuare vom trece în revistă caracteristicile simțurilor (capacități și limitări) și modul în care acestea se integrează în interacțiune.

Vederea

Vederea este o cîvitate complexă cu limitări fizice și perceptuale. Este principala sursă de informare pentru majoritatea persoanelor. Se realizează în două faze: percepția fizică a stimulilor și procesarea și interpretarea stimulilor. Există proprietăți fizice ale ochiului care fac ca anumite obiecte să nu poată fi văzute. Capacitățile interpretative ale procesorului vizual permit construcția de imagini din informații incomplete *compensarea*, dar e posibil să apară și *iluzii* (vezi Figura 2.1). Iluziile optice demonstrează faptul că lucrurile nu sunt întocmai cum le vedem.

Modul în care sunt compuse figurile geometrice afectează modul în care le percepem. Figurile geometrice nu sunt percepute aşa cum sunt desenate – tendința este de a percepere să mărită liniile orizontale și de a percepere redusă liniile verticale (un pătrat trebuie desenat ca dreptunghi pentru a-l vedea ca pătrat). Liniile desenate orizontale par mai lungi decât cele verticale. Simetria paginilor este afectată de iluziile optice deoarece percepem centrul paginii puțin deasupra poziției lui reale – ”optical center” (vezi Figura 2.2).

Citirea Percepția și procesarea textului reprezintă un caz special,

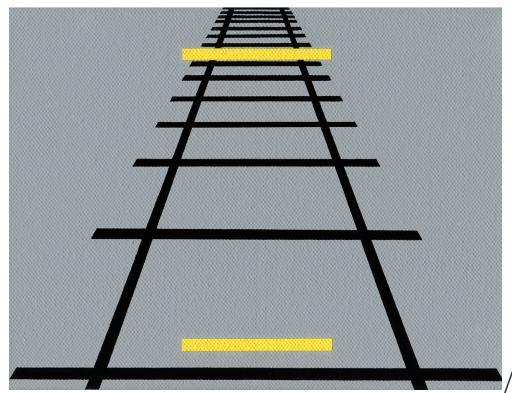


Figura 2.1: Iluzia Ponzo [54]

care se realizează în două etape: percepția şablonului cuvântului și decodificarea şablonului pe baza reprezentării interne a limbajului. Cuvintele nu sunt citite caracter cu caracter, un cuvânt e percepț la fel de repede ca și un caracter, pe baza formei cuvântului. Dacă prima și ultima literă a cuvântului sunt în poziția potrivită, citirea poate fi realizată cu succes.

Exemplu Pe bzaa uonr sudtii a ueni uvetsniariti egzlene, nu are intortmapa in ce odrnie satu liertele itnr-un cvunat. Imatpornt etse ca pmria si umtlia lireta sa fie la lcoul pitovrit. Rtseul leilretor pot sta in ocire odrnie si tustoi ptoi ctii. Atsa e psboiil prntru ca noi ctiim cunelvite irngtei si nu letira cu ltirea.

Culorile Culorile au un impact major în dezvoltarea interfețelor utilizator. Nu întotdeauna impactul este unul pozitiv. Folosirea corectă a culorilor poate îmbunătăți procesul de memorarea și formarea de modele mentale corespunzătoare. Culorile primare sunt: roșu, verde, galben, albastru și este obisnuit ca acestea să aibă atașate semnificații. Proiectarea ținând cont de semnificațiile culturale ale culorilor șăurează utilizarea sistemelor interactive. Numărul magic de culori care ar trebui folosite în implementarea unei interfețe

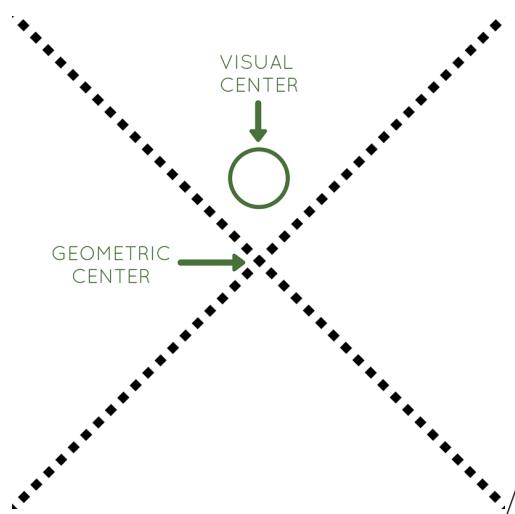


Figura 2.2: Poziția centrului optic

este de $5+/-2$ culori. Ca și recomandare pentru utilizarea eficientă a culorilor se ține seama că pentru concepe diferite, se vor folosi culori diferite, nu nuanțe diferite. În proiectarea interfețelor se va ține cont de diferențele culturale. De asemenea, este important să se țină cont de faptul că există combinații de culori problematice, și acestea au legătură cu utilizarea culorii albastru. Există probleme atunci cand albastrul și galbenul sunt folosite in cadrul graficelor legate de :

- lizibilitate
- viteza de răspuns
- localizare spațială
- percepția formelor geometrice

Această contrângere are legătură cu faptul că există mai puțini fotoreceptori pentru albastru decât pentru verde și roșu și nu există deloc fotoreceptori pentru albastru în zona de acuitate maximă. Recomandarea frecventă este că “Pure blue should not be used for

fine detail or background “ Utilizarea greșită a culorilor poate să facă o interfață mai greu de folosit. De aceea, interfețele ar trebui proiectate astfel încât utilizatorii să poată renunța cu ușurință la ele. E important să proiectăm mai întâi fără culori întrucât dispoziția obiectelor este esențială, culorile sunt folosite doar pentru a îmbunătăți vizualizarea. Schimbarea culorilor se va folosi la schimbarea stării sistemului. Pentru o bună lizibilitatea textul care trebuie citit trebuie să aibă un contrast sporit (recomandabil text negru pe fundal alb sau galben pal), de evitat fundalul gri. Se recomandă atenție sporită nevoilor utilizatorilor mai vîrstnici – după 45 ani majoritatea oamenilor suferă de o reducere a transmisiei luminii în ochi. Se recomandă folosirea fonturilor suficient de mari pentru a putea fi citite pe monitoare standard. Caracterele mai mari vor fi folosite pentru informațiile care se doresc a fi afișate, mai degrabă decât pentru etichete și instrucțiuni (utilizatorii le vor recunoaște).

În ceea ce privește alegerea tipurilor de fonturi (cu serifuri sau fără serifuri), există următoarele remarcări. Fonturile cu serifuri ghidează fluxul orizontal al ochilor în timpul citirii, crește spațiul dintre litere și cuvinte sprijinind lizibilitatea, sporește contrastul și diferențele dintre litere ușurând identificarea, leagă caracterele în cuvinte intregi (sporește coeziunea). În ceea ce privește fonturile fără serif se consideră că sunt mai potrivite pentru paginile web, pentru că în funcție de rezoluție, serifurile sunt greu de reprodus. La dimensiuni mici ale fonturilor sunt mai ușor lizibile și sunt preferabile în cazul copiilor care învață să citească, fiind mai ușor de recunoscut.

2.2.2 Auzul

Auzul este considerat secundar vederii, deși cantitatea de informație pe care o primim prin auz este extrem de mare. Sistemul auditiv realizează o filtrare a sunetelor pe care le primește, permittând ignorarea sunetelor de fundal și concentrarea atenției asupra sunetelor importante – ”cocktail party effect”. Sunetele nu sunt folosite la potențialul lor real în proiectarea interfețelor utilizator, doar pentru avertizări și notificări. Urechea poate diferenția schimbări subtile ale sunetelor și poate recunoaște sunete familiare fără a impune concentrarea atenției asupra sursei sunetului – ar putea fi folosit pentru a furniza informații despre starea sistemului.

2.2.3 Atingerea (Simțul tactil)

Simțul tactil furnizează informații relevante despre mediu și este foarte dezvoltat la persoanele cu dizabilități. Aparatul tactil nu este localizat, receptorul fiind pielea.

Există trei tipuri de receptori senzoriali:

- Termoreceptori (cald/frig)
- Receptori de presiune
- Mecanoreceptori (căldură/presiune/durere) –

Toti acești receptori sunt importanți în studiul HCI pentru că aceștia influențează performanța utilizatorului. Mai mult decât atât, o combinare a acestora poate determina răspunsuri mai rapide din partea utilizatorilor.

2.2.4 Mișcarea

Mișcarea se realizează în etape: stimulul este primit prin intermediul unor receptori – transmis la creier – procesat – mușchii sunt acționați. Fiecare etapă necesită timp. Timpul de mișcare (deinde de vîrstă, starea de sănătate). Timpul de reacție depinde de canalul pe care e primit stimulul (stimul auditiv – 150ms, stimul vizual -200 ms, stimul dureros-700ms). Combinarea lor determină un răspuns mai rapid. Timpul de reacție e foarte important în HCI, atsfel încât s-au încercat variate moduri de estimare a lui.

Legea lui Fitt

Calculează timpul necesar deplasării la o țintă de pe ecran (buton, opțiune de meniu, iconă) în funcție de dimensiunea țintei (S) și distanța față de țintă (D). $T = a + b * \log_2(D/S + 1)$ a, b sunt constante determinate empiric. Astfel, recomandarea este să facem dimensiunea controalelor utilizate mai des mare.

2.2.5 Memoria

Există 3 tipuri de memorie: senzorială, de scurtă durată (de luncru) și de lungă durată. Memoria senzorială este o zonă tampon pentru stimulii primiți de la simțuri. Există câte o zonă de memorie senzorială pentru fiecare canal: memorie iconică, memorie ecoică și memorie tactilă. Memoria senzorială se suprascrie în mod continuu cu informațiile noi primeite. Informațiile din memoria senzorială sunt transferate în memoria de scurtă durată prin atenție (sunt filtrați doar stimulii importanți). Este influențată de modul de prezentare a informației și de scopurile noastre (dacă sunt clare). Memoria de scurtă durată este folosită pentru a reține informații

pentru scurt timp (ex: $35 \times 6 = ?$) sau în citire, pentru a reține începutul unei propoziții. Accesul la informații e foarte rapid – 70 ms, dar informația poate fi reținută maxim 200 ms.

Memoria de lungă durată reține cunoștințe achiziționate din experiență, reguli procedurale, cam tot ce “știm”. Are capacitate mare, dar accesul destul de încet – 1/10 s. Informațiile se uită mai greu, posibilitatea de reamintire este aceeași după 5 minute/1 oră/câteva zile.

Informațiile trec din memoria de scurtă durată în memoria de lungă durată prin repetare. Dacă informațianu are semnificație, e greu de memorat. Acest lucru are legătură cu solicitarea creării de parole cât mai complicate, care să fie formate din litere și cifre aleator – întocmai ceea ce este cel mai dificil pentru oameni.

2.2.6 Gândirea

Gândirea este procesul prin care folosim cunoștințele pe care le avem pentru a trage concluzii noi. Există trei tipuri de gândire: deductivă, inductivă și abductivă.

Gândirea deductivă derivează concluzii logice din premisele existente (de la general la particular). De exemplu:

Dacă este JOI trebuie să mergem la cursul de HCI.

Este joi.

Concluzia: *Mergem la curs.*

Rezultatul deductiei logice poate să intre în conflict cu cunoștințele noastre despre realitate. Apar coliziuni între adevăr și validitate – oamenii aduc în procesul de gândire informații despre lume pentru a crea scurtături în procesul de gândire.

Gândirea inductivă Inducția reprezintă generalizarea unor situații pe care le-am întâlnit pentru a infera informații despre situații pe care nu le-am întâlnit.

De exemplu:

Un elefant pe care l-am văzut avea trompă → toți elefanții au trompă.

Gândirea inductivă surprinde regularitatea, ceea ce este comun, constant, invariant. Ea facilitează extragerea și formularea unei concluzii generale dintr-o multitudine de cazuri particulare. Concluzia va rămâne valabilă până când vom întâlni o excepție-produsele gândirii inductive nu sunt definitive și nici absolut sigure, dimpotrivă, ele pot fi oricând puse în discuție. Procesul nu este riguros, dar e folosit adesea în procesul de învățare despre mediul inconjurător

Gândirea abductivă Abducția funcționează de la un fapt la acțiunea sau starea care a cauzat respectivul fapt. Este folosită pentru a găsi explicații ale evenimentelor pe care le observăm. De exemplu:

Sam conduce cu viteza mare când bea.

Concluzia: *Dacă Sam conduce cu viteză mare → a băut.*

Gândirea nu este riguroasă, dar e folosită de oameni până la identificarea unor probe contrare.

Din perspectiva interacțiunii om-calculator, pot apărea dificultăți dacă un eveniment urmează întotdeauna unei acțiuni, utilizatorul va infera că evenimentul este generat de acțiune, până când contrariul este dovedit. Dacă, de fapt, evenimentul și acțiunea nu sunt relaționate apar erorile și confuzia.

2.2.7 Rezolvarea de probleme

Rezolvarea de probleme este procesul de găsire a unei soluții la o problema nefamiliară, folosind cunoștințele pe care le avem. Există mai multe abordări în explicarea modului în care oamenii rezolvă probleme. **Curentul behaviorist** consideră că oamenii rezolvă probleme prin reproducerea răspunsurilor cunoscute și încercare-eroare.

Curentul gestaltist susține că oamenii rezolvă probleme prin gândire și restructurare a problemei. **Problem space theory** susține că în rezolvarea de probleme oamenii se concentrează asupra spațiului problemei compus din stări ale problemei. Rezolvarea problemei presupune generarea stărilor folosind operatori stare-tranziție permisi. Problema are o stare inițială și un scop – se folosesc operatorii pentru a ajunge de la starea inițială la scop. Spațiul problemei poate fi foarte mare, astfel îcât alegerea operatorilor poate fi dificilă și necesită folosirea unor euristici (starea inițială este comparată cu scopul final și operatorul e astfel ales încât diferența dintre cele două să scadă). Spațiul problemei este limitat de capacitatea de stocare a memoriei de scurtă durată iar viteza de regăsire a informațiilor influențează eficiența metodei.

Analogia se consideră că oamenii folosesc maparea cunoștințelor dintr-un domeniu similar cunoscut problemei noi. Operatorii din domeniul cunoscut sunt transferați în scopul rezolvării problemei noi.

Modelele mentale Un produs de succes este bazat pe un model conceptual care permite utilizatorilor să învețe imediat cum să folosească eficient produsul. În timpul în care utilizatorii folosesc sistemul achiziționează informații despre cum trebuie folosit și despre modul în care acesta funcționează se conturează un model mental. Modelul mental este folosit pentru a face inferențe despre

modul de realizare a sarcinilor cu sistemul și pentru a reacționa când apare ceva neașteptat sau când ne confruntăm cu un sistem cu care nu suntem familiari. Modelele mentale sunt parțiale, instabile, inconsistente, neștiințifice, bazate pe superstiție mai degrabă decât argumente riguroase, bazate pe o interpretare greșită (uneori) a probelor. Un exemplu de model mental este ”more is more” (apăsăm de mai multe ori butonul de la trecerea de pietoni semaforizată pentru a primi liber mai repede, cu toate că sistemul nu funcționează așa).

Existența și caracteristicile modelelor mentale explică apariția erorilor. Există două categorii mari de erori: cele care apar la modificarea unor aspecte în cazul sarcinilor automatizate (din cauza vitezei se omit anumite aspecte din realizarea sarcinilor) sau cele determinate de o înțelegere incorectă a sistemului sau situației (se încearcă realizarea unor acțiuni care nici măcar nu sunt oferite de către sistem). Pentru a obține model mental mai bune există următoarele abordări: educare (manuale de utilizare, help, documentații), feedback util la acțiunile utilizatorului, modalități ușor de înțeles și intuitive de utilizare a sistemului, instrucțiuni clare și ușor de urmat, asistență contextualizată, sistem de help bine construit.

Emoțiile Interacțiunea cu mediul înconjurător este influențată și de emoții. Răspunsul nostru emoțional la anumite situații afectează modul în care reacționăm. Emoțiile pozitive sporesc creativitatea, permit rezolvarea unor probleme complexe, iar emoțiile negative determină o gândire îngustă, concentrată numai pe anumite aspecte. O problemă pe care o rezolvăm cu ușurință fiind relaxați poate deveni dificilă dacă suntem frustrați sau ne temem.

2.3 Utilizabilitatea sistemelor interactive

Atunci când se discută despre calitățile sistemelor informaticе, deseori se menționează conceptul de *utilizabilitate* [57]. Abordările centrate pe utilizator în proiectarea sistemelor interactive, dar nu numai, pun accentul pe această caracteristică a programelor sau a altor dispozitive și echipamente. Este necesară însă clarificarea conceptului, deoarece utilizabilitatea nu este o trăsătură generală a unui produs informatic, ci, aşa cum vom vedea, este dependentă de factori precum: contextul de utilizare, utilizatori și sarcini.

2.3.1 Definiții ale utilizabilității

În această secțiune sunt prezentate și discutate câteva definiții ale utilizabilității. Menționăm că nu există o definiție a utilizabilității unanim acceptată. După cum vom vedea, unele definiții vor fi foarte asemănătoare, în timp ce altele vor prezenta vizuni diferite asupra conceptului.

Shakel [62] definește utilizabilitatea ca fiind capacitatea unui sistem de a fi folosit eficient și ușor de o categorie specifică de utilizatori, care au avut parte de o instruire specifică și asistență pentru utilizarea produsului, pentru a îndeplini un domeniu specific de sarcini într-un cadrul specific. Definiția e operaționalizată prin intermediul a patru criterii: eficiență, ușurință de învățare, flexibilitate și atitudine [62].

Preece [55] privește utilizabilitatea ca măsura ușurinței cu care un sistem poate fi învățat sau utilizat, siguranța, eficiența și eficacitatea sa, și atitudinea utilizatorilor față de sistem.

Standardul ISO 9241-11 definește utilizabilitatea ca fiind: "măsura în care un produs poate fi folosit de utilizatori specifici pentru a atinge scopuri specifice cu eficiență, eficacitate și satisfacție într-un context specific de utilizare" [30].

Eficiența se referă la raportul dintre resursele consumate și

acuratețea și completitudinea cu care utilizatorii îndeplinesc scopuri.

Eficacitatea se referă la acuratețea și completitudinea cu care utilizatorii îndeplinesc sarcini specificate.

Satisfacția este o măsură subiectivă referitoare la confortul și acceptarea produsului de către utilizatorii finali.

Paterno [56] consideră utilizabilitatea a fi un concept multidimensional. Acesta cuprinde mai mult decât "ușurința de învățare" și "ușurința de folosire", deoarece dacă utilizatorii nu pot realiza toate sarcinile pe care și le-ar dori, deoarece o anumită caracteristică îi lipsește sistemului, aceștia nu vor caracteriza sistemul a fi utilizabil. Astfel, definiția utilizabilității ar trebui să includă cel puțin următoarele elemente:

- relevanța sistemului, care se referă la măsura în care acesta servește nevoile utilizatorilor;
- eficiența, care se referă la eficiența cu care utilizatorii pot să își realizeze sarcinile folosind sistemul;
- atitudinea utilizatorilor față de sistem, care se referă la sentimentele subiective față de sistem;
- ușurința de învățare a sistemului, care se referă la cât de ușor este de învățat sistemul pentru utilizarea inițială și cât de ușor își amintesc utilizatorii modul de utilizare a sistemului;
- siguranța sistemului, care se referă la posibilitatea utilizatorilor de a anula acțiuni și de a nu permite sistemului de a acționa într-un mod destructiv.

Dix [22] oferă o definiție operaționalizată a utilizabilității, oferind proiectanților un punct de reper atunci când se proiectează sisteme pentru utilizabilitate. Din perspectiva sa, utilizabilitatea e asigurată de trei factori:

- ușurință de învățare - se referă la ușurința cu care utilizatorii noi pot să înceapă interacțiunea efectivă și să atingă performanță maximă;
- flexibilitate - se referă la multitudinea canalelor prin care utilizatorul și sistemul schimbă informație;
- robustețe - se referă la nivelul sprijinului care îi este acordat utilizatorului în îndeplinierea sarcinilor.

Aceste principii care stau la baza utilizabilității sunt detaliate în cele ce urmează.

§. Ușurința de învățare

Trăsăturile unui sistem interactiv care sporește ușurința de învățare sunt discutate în următoarele paragrafe.

Predictibilitatea

Predictibilitatea se referă la faptul că cunoștințele utilizatorului relative la istoricul interacțiunii sunt suficiente pentru a determina rezultatele interacțiunilor viitoare.

Predictibilitatea este strâns legată de abilitatea utilizatorului de a determina efectele operațiilor asupra sistemului. O formă a predictibilității este abilitatea utilizatorilor de a ști ce operații pot fi efectuate. Vizibilitatea operațiilor se referă tocmai la modul în care disponibilitatea operațiilor îi este sugerată utilizatorului. Dacă o operație poate fi efectuată, atunci trebuie să existe o indicație a acestui fapt pentru utilizator. De asemenea, utilizatorul trebuie să înțeleagă din interfață că o operație pe care dorește să o invoke nu e disponibilă.

Sintetizabilitatea

Predictibilitatea relativă la utilizarea unui sistem informatic presupune existența unui model mental al utilizatorului. Construirea acestui model mental se realizează în fapt prin aceea că utilizatorul evaluează consecințele acțiunilor anterioare asupra stării sistemului. *Sinteza* este abilitatea utilizatorului de a evalua efectele acțiunilor anterioare asupra stării curente.

Când o operație schimbă un aspect al stării interne a programului este necesar ca schimbarea să fie văzută de utilizator. Această notificare ar trebui să apară imediat, fără a necesita alte interacțiuni (de exemplu: crearea unui director folosind un sistem de operare Windows), sau, eventual, după directive explicite ale utilizatorului (crearea unui director folosind un sistem de operare MS-DOS).

Familiaritatea

Utilizatorii noi ai un sistem dețin o bogată experiență provenind din diverse domenii de aplicații, experiență obținută prin interacțiuni din lumea reală și prin interacțiune cu sisteme informatiche. Pentru un utilizator nou al sistemului, *familiaritatea* este o măsură a corelației dintre cunoștințele existente ale utilizatorului și cunoștințele necesare interacțiunii efective. Familiaritatea depinde de prima impresie a utilizatorului asupra sistemului. Este de interes modul în care este percepția sistemul de către utilizator și dacă acesta poate iniția interacțiunea. Modul de prezentare a unui obiect poate sugera modul în care acesta poate fi manipulat (un buton sugerează că trebuie apăsat).

Generalizabilitatea

Utilizatorii încearcă să extindă cunoștințele deținute despre comportamentul specific de interacțiune la situații similare, dar ne-

mai întâlnite. Generalizabilitatea unui sistem interactiv sprijină astfel de inițiative și construirea unui model predictiv al sistemului. Generalizarea se aplică în situații în care utilizatorul vrea să aplique cunoștințe care ajută la atingerea unui scop la situații în care scopul e oarecum similar. Generalizarea poate să apară în cadrul aceleiași aplicații sau între aplicații diverse (de exemplu: funcționalitățile de cut/copy/paste).

Consistență

Consistența se referă la asemănarea în comportament care apare în situații similare sau obiective similare ale sarcinilor. Consistența trebuie aplicată în relație cu ceva (ex: denumirea comenziilor, invocarea comenziilor și argumentelor). Unele din principiile anterior enunțate pot fi "reduse" la consistență: familiaritatea poate fi considerată consistentă în raport cu experiența reală trecută, iar generalizabilitatea poate fi considerată consistentă în raport cu experiențele cu același sistem sau multe de aplicații de pe aceeași platformă.

§. Flexibilitatea

Flexibilitatea se referă la multitudinea canalelor prin care sistemul și utilizatorul schimbă informație. Există două modalități de comunicare: sistemul inițiază dialogul, utilizatorul răspunzând cererilor sistemului - în acest caz vorbim de dialog *preemptiv din partea sistemului*; utilizatorul poate să fie liber să inițieze orice acțiune, caz în care dialogul e *preemptiv utilizator*. Sistemul poate să controleze dialogul până la limita de a împiedica utilizatorul să inițieze orice comunicare privind sarcina curentă sau altă sarcină pe care utilizatorul ar dori să o îndeplinească. O interacțiune condusă de sistem scade flexibilitatea, iar ceea ce se dorește este ca utilizatorul să conducă sistemul, nu viceversa. Există însă situații în care un dialog preemptiv sistem este necesar (dacă

două persoane editează același fișier e necesară împiedicarea unor acțiuni precum ștergerea).

Migrabilitatea sarcinilor

Migrabilitatea sarcinilor privește transferul controlului execuției între sistem și utilizator. Este necesar să se ofere posibilitatea ca o sarcină internă să devină cooperativă sau sistemul și utilizatorul să își transfere controlul asupra execuției. Un exemplu de situație este cea de corectare gramaticală a unui document: această sarcină, lăsată complet în seama sistemului poate duce la tratarea incorectă a numelor proprii.

Substituitivitatea

Substituitivitatea impune ca valori echivalente să se poată substitui (de exemplu: la introducerea dimensiunilor marginilor unui document trebuie să fie posibilă folosirea valorilor atât în inch, cât și în centimetri, la fel ca și situația introducerii unei valori din calculul unei expresii aritmetice).

Customizabilitatea

Customizabilitatea se referă la posibilitatea modificării interfeței utilizator de către sistem (modificări făcute pe baza cunoștințelor despre utilizator) sau de către utilizator. Dacă modificările sunt inițiate de sistem vorbim despre *adaptivitate*, iar dacă modificările sunt inițiate de utilizator vorbim despre *adaptabilitate*. Adaptabilitatea se referă la abilitatea utilizatorului de a ajusta forma intrărilor și a ieșirilor (de obicei se limitează la poziția butoanelor sau numele comenziilor). Acest tip de modificări, restricționate la nivelul de prezentare al interfeței, poartă numele de *customizabilitate lexicală*. Utilizatorii pot fi înzestrăți cu o putere mai mare permitând definirea de macrouri.

Adaptivitatea se referă la customizarea automată a interfeței utilizator de către sistem. Deciziile de adaptare pot fi luate pe baza expertizei utilizatorului sau pe observarea repetării unor secvențe de sarcini. Un sistem poate fi antrenat să recunoască comportamentul unui expert sau al unui novice și să își modifice controlul dialogului sau sistemul de help corespunzător.

§. Robustetea

Robustetea se referă la caracteristicile sistemului care sprijină îndeplinirea cu succes și evaluarea scopurilor. În cele ce urmează vom discuta principiile care sprijină robustetea.

Observabilitatea

Observabilitatea permite utilizatorului să evaluateze starea internă a sistemului pe baza reprezentărilor perceptibile de la nivelul interfeței.

Recuperarea erorilor

Recuperabilitatea este abilitatea de a atinge un scop dorit după recunoașterea unor erori în interacțiunea anterioară. Recuperarea poate avea loc în două direcții: recuperare înainte - când utilizatorul recunoaște starea de eroare și încearcă atingerea stării dorite din starea curentă și recuperare înapoi. Recuperarea înainte este uneori singura posibilitate de recuperare, întâlnită în cazul sistemelor în care efectele interacțiunii nu sunt revocabile. Recuperarea înapoi se referă la încercarea de a anula efectele interacțiunii anterioare cu scopul întoarcerii la o stare anterioară înainte de a continua interacțiunea. Procedura de recuperare trebuie să reflecte munca efectuată, respectându-se principiul efortului comensurat care spune că dacă e dificil să anulezi efectul unei acțiuni într-o anumită

stare, atunci trebuie ca această acțiune să fie greu de efectuat. Reciproc, o acțiune care e ușor de anulat, trebuie să fie ușor de efectuat.

Responsivitatea/Timpul de răspuns

Responsivitatea măsoară rata comunicării dintre sistem și utilizator. Timpul de răspuns e definit a fi durata de timp necesară sistemului pentru a exprima schimbările de stare într-un mod perceptibil utilizatorului. În general e dezirabil ca răspunsul să apară instantaneu, iar dacă acest lucru nu este posibil, atunci utilizatorul trebuie să fie informat asupra faptului că sistemul lucrează pentru a obține răspunsul. Un alt aspect important e *stabilitatea* timpului de răspuns, care se referă la invarianța duratei pentru acțiuni similare sau identice (de exemplu: meniurile pull-down se așteaptă să se deschidă imediat după acționarea butonului mouse-ului).

Conformanța cu sarcina

Sistemele interactive sunt proiectate pentru a sprijini utilizatorii în realizarea de sarcini aparținând unui domeniu de aplicații. Problema care se ridică este dacă sistemul interactiv sprijină toate sarcinile utilizator și dacă sprijinul este oferit într-un mod acceptat de utilizator. Conformanța cu sarcina se referă tocmai la completitudinea sarcinilor și adecvarea cu sarcina din punctul de vedere al utilizatorului.

2.3.2 Un model stratificat al utilizabilității

Martijn van Welie [78] propune un model stratificat al utilizabilității pornind de la definiția dată de ISO, încercând să clarifice care sunt mecanismele care stau la baza utilizabilității, model prezentat în Figura 2.3.

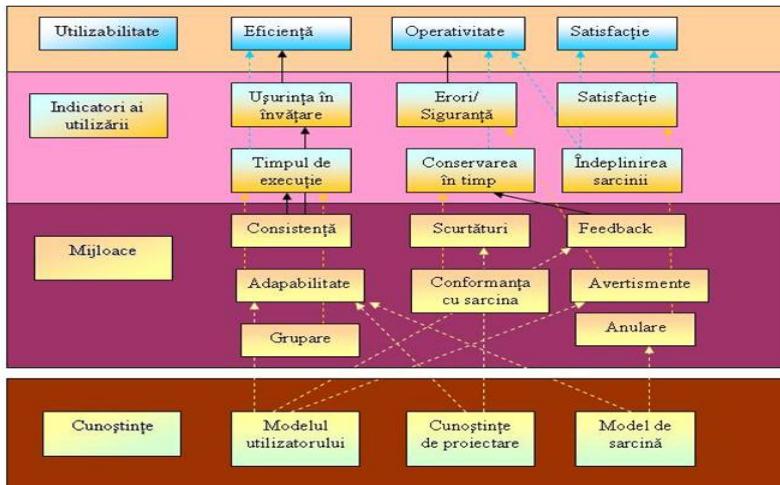


Figura 2.3: Modelul stratificat al utilizabilității [78]

La nivelul cel mai înalt se găsesc conceptele componente ale utilizabilității definite de ISO [30]. La următorul nivel se găsesc indicatorii utilizabilității care pot să fie observați atunci când utilizatorul interacționează cu sistemul. La următorul nivel se găsesc mijloacele, care nu pot fi observate în testările cu utilizatorii și nu sunt scopuri în sine ca și indicatorii. Mijloacele sunt folosite în euristică pentru a îmbunătăți unul sau mai mulți indicatori.

2.3.3 Metode de măsurare a utilizabilității

Evaluarea utilizabilității poate să aibă loc pe parcursul proiectării unui sistem sau după dezvoltarea acestuia. Evaluarea utilizabilității se poate realiza prin testări cu utilizatorii, rezultatele acestor testări fiind cele mai relevante dar în același timp și cele mai costisitoare. Prin folosirea scenariilor se pot obține date despre numărul erorilor sau viteza de execuție, care sunt indicatori ai utilizabilității. Chestionarele de utilizabilitate precum SUMI [65] sau QUIS [16] oferă informații standardizate despre utilizabilitatea sistemelor informaticice.

Asigurarea utilizabilității încă din timpul proiectării este sprijinită de folosirea modelelor formale de proiectare, precum diagramele stratificate de stări, GOMS [13], ConcurTaskTrees [50], TAG [51] sau BNF [10]. Aceste metode sunt foarte puternice în descrierea aspectelor structurale ale proiectării, însă nu surprind aspectele prezentationale. Câteva dintre metodele anterior enumerate vor fi discutate în capitolele 3, respectiv 4 ale prezentei lucrări.

Tabelul 2.3.1 descrie câteva metrii ale utilizabilității și indicatorii utilizabilității care pot fi folosiți, conform modelului prezentat în [78].

Metoda cea mai eficientă de măsurare a utilizabilității rămâne însă testarea cu utilizatorii, aceasta însă este cea mai costisitoare dintre metode și presupune existența unui prototip funcțional al sistemului. Metodele care oferă estimări ale utilizabilității sistemelor încă din faza de proiectare reprezintă alegerea preferată de dezvoltatorii de sisteme. Din această categorie de metode fac parte inspectarea utilizabilității sau evaluarea respectării şabloanelor de proiectare a interfețelor utilizator. Inspectarea utilizabilității este o revizuire a sistemului pe baza unui set de recomandări. Revizuirea este condusă de un grup de experți familiarizați cu conceptele utilizabilității în proiectare. Experții se concentreză asupra unei liste de aspecte din proiectare care s-au dovedit problematice pentru utilizatori. Recomandările de utilizabilitate sunt derivate din studii de HCI, ergonomie, design grafic, proiectarea informației și psihologie cognitivă. Aspectele evaluate pot fi: limbajul folosit în interacțiune, măsura în care se apelează la aducerea aminte a utilizatorului și modul în care sistemul oferă feedback utilizatorilor. Inspectarea utilizabilității poate fi realizată prin diverse metode precum:

- **revizuirile cognitive** - revizuirile cognitive pot fi efectuate în orice etapă a dezvoltării unui produs, folosind fie un document aparținând proiectării conceptuale, un prototip sau produsul final. Pe baza scopurilor utilizatorilor un grup de evaluatori parcurg sarcina pas cu pas, evaluând la fiecare pas cât este de dificil pentru

utilizator să identifice și să opereze asupra elementului din interfață care este cel mai relevant pentru subscopul curent și cât de clar este feedback-ul pe care sistemul îl oferă utilizatorilor. Revizuirile cognitive iau în considerare procesul de gândire care determină luarea deciziilor, precum încărcarea memoriei și abilitatea de a raționa. Această abordare este folosită atunci când se încearcă explorarea utilizabilității pentru utilizatorii care folosesc pentru prima dată sistemul sau utilizatorii ocionali ai sistemului, adică acei utilizatori care învață prin explorare;

- **revizuiri pluralistice** - folosesc întâlniri de grupuri în care utilizatori, dezvoltatori și specialiști în factori umani parcurg un scenariu și discută fiecare element al dialogului;
- **inspectarea consistenței** - un grup de proiectanți care participă la realizarea unor produse similare inspectează interfața pentru a vedea dacă permite realizarea funcționalității în același mod ca și produsele proprii;
- **inspectarea standardelor** - un expert într-un standard verifică respectarea standardului de către o interfață;
- **inspectarea caracteristicilor sistemului** - se listează secvențele folosite în realizarea unei sarcini, se caută secvențele prea lungi, pașii confuzi, acțiunile care nu sunt selectate natural de către utilizator, pașii care necesită cunoștințe extensive sau expertiză pentru a realiza un scop;
- **evaluarea euristică** - este o tehnică de determinare a problemelor de utilizabilitate cu o interfață. Un număr mic de evaluatori experimentați (3-5) inspectează separat o interfață folosind un set de euristici. Apoi, combină rezultatele evaluărilor și fac o clasificare a problemelor de utilizabilitate după gravitatea acestora. Sunt general acceptate cele 10 euristici ale lui Nielsen: vizibilitatea stării

sistemului, potrivirea dintre sistem și lumea reală, controlul utilizatorului și libertatea, consistență și standarde, prevenirea erorilor, recunoaștere (nu reamintire), flexibilitate și eficiență în utilizare, design minimalist și estetic, ajutorul utilizatorilor să recunoască, diagnosticeze și să recupereze erorile, help și documentare [45];

- **focus groups** - focus grupurile reprezintă o metodă foarte eficientă de a culege reacții de la utilizatori și de a aprecia reacțiile inițiale la proiectare, dar și pentru a identifica diferențele dintre așteptările utilizatorilor și implementarea actuală.

2.4 Interfața utilizator - mediatorul interacțiunii om-calculator

Interfața utilizator este locul de întâlnire a utilizatorului cu sistemul informatic fizic, perceptual sau conceptual [43], ea mediind schimbul de informație dintre utilizator și sistemul informatic. Pentru mulți dintre utilizatori, interfața este însuși sistemul, astfel încât o interfață nepotrivită va determina un calificativ slab pentru sistemul informatic. Importanța modului de proiectare a interfeței utilizator a fost subliniată în momentul în care sistemele informatiche nu au mai fost destinate utilizatorilor experți (cei care le concepeau), ci unui segment mai larg al populației care cuprindea utilizatori novici. În acel moment, sisteme informatiche cu performanțe remarcabile și funcționalitate bună au fost clasificate drept nesatisfăcătoare pentru că produceau frustrare și insatisfacție utilizatorilor, care evitau utilizarea lor. A fost momentul în care s-a formulat concluzia conform căreia slaba utilizabilitate a sistemelor interactive se datora proiectării nepotrivite a interfeței utilizator, astfel încât această componentă a sistemelor interactive a primit o mai mare atenție din partea cercetătorilor. De asemenea, utilizatorii sistemelor interactive au devenit subiect de studiu în momentul proiectării siste-

melor interactive, astfel încât în procesul de proiectare au fost incluse și modele ale utilizatorului, alături de modele ale domeniului sau fluxului de date. Prima încercare de abordare a proiectării interfeței utilizator din perspectiva utilizatorului este reprezentată de Command Language Grammar (CLG) și datează din 1981 [43]. CLG încearcă o descrierea a interfeței utilizator pornind de la modelul conceptual al interfeței pe care îl detine utilizatorul. Conform lui Moran, există trei componente majore ale interfeței utilizator: *componenta conceptuală* care cuprinde concepte abstracte în jurul cărora e organizat sistemul, *componenta de comunicare* care conține limbajul de comandă și dialogul, respectiv *componenta fizică* care conține echipamentele fizice pe care utilizatorul le vede și cu care intră în contact. Aceste trei componente sunt rafinate la următorul nivel de descompunere conform Tabelei 2.4.1.

Nivelul sarcinii - descrie domeniul sarcinii pentru care e dezvoltat sistemul. Utilizatorul folosește sistemul pentru a îndeplini un set de sarcini. La acest nivel se analizează nevoile utilizatorului și se încearcă structurarea domeniului sarcinii într-un mod potrivit pentru a fi supus automatizării. Rezultatul analizei de la acest nivel este o structură de sarcini specifice pe care utilizatorul le va îndeplini cu sistemul.

Nivelul semantic descrie conceptele utilizate de sistem. Un sistem este construit în jurul unor obiecte și a operațiilor efectuate asupra obiectelor. Pentru sistem acestea se traduc prin structuri de date și proceduri, pentru utilizator prin entități conceptuale și operații conceptuale asupra acestor entități. Nivelul semantic cuprinde aceste entități și operații care contribuie la îndeplinirea sarcinilor, dar și metode pentru îndeplinirea sarcinilor în termenii acestor entități conceptuale și operații.

Nivelul sintactic - modelul conceptual al sistemului este cuprins în structura limbajului de comandă. Limbajul de comandă cuprinde câteva elemente sintactice: comenzi, argumente, contexte, variabile de stare. Semnificația fiecărei comenzi este definită prin operațiile de la nivelul semantic, iar metodele de la nivelul semantic pot fi descrise

prin intermediul comenziilor de la nivelul sintactic.

Nivelul interacțiunii descrie structura dialogului, care în ultimă instantă se traduce prin acțiuni fizice - apăsări de taste, etc. Nivelul interacțiunii specifică acțiunile fizice asociate elementelor de la nivelul sintactic.

Componența fizică a interacțiunii este considerată de o importanță mai scăzută, astfel încât nu vom insista asupra acestui aspect.

Scopul principal al CLG a fost cel al separării modelului conceptual al sistemului de limbajul de comandă și de evidențiere a relațiilor dintre ele. Astfel, modelul conceptual al utilizatorului e studiat de proiectant și/sau de psiholog, iar apoi se proiectează un limbaj de comandă care implementează modelul conceptual (se poate considera a fi o abordare top-down a proiectării sistemului informatic).

Pentru a evita modul simplist în care este privită interfața utilizator, Tauber [71] a introdus un nou concept - **mașină virtuală a utilizatorului** (UVM sau User's Virtual Machine) - care să surprindă complexitatea interfeței utilizator și să evite reducerea interfeței utilizator la aspectul de prezentare. UVM cuprinde acele aspecte ale sistemului care sunt importante pentru utilizator, și anume:

- funcționalitatea - sarcinile de bază pe care utilizatorul le poate delega sistemului;
- limbajul interfeței - limbajul în care utilizatorul trebuie să se exprime în interacțiunea cu sistemul;
- prezentarea - reprezentarea informațiilor relevante pentru utilizator.

În prezenta lucrare semnificația termenului interfață utilizator va referi aspectele relative comunicării, funcționalității sistemului și prezentării.

Metrică	Indicatorul de utilizabilitate
Timpul de realizare a unei sarcini specifice	Timpul de execuție
Numărul de comenzi folosite	Timpul de execuție
Procentul de sarcină realizat în unitatea de timp	Timpul de execuție
Timpul mediu petrecut pentru acțiuni fizice	Timpul de execuție
Timpul mediu petrecut pentru acțiuni mentale	Timpul de execuție
Timpul petrecut în așteptarea răspunsurilor sistemului	Timpul de execuție
Numărul de sarcini ce pot fi executate într-un interval de timp	Timpul de execuție
Numărul comportamentelor regresive	Memorabilitate
Numărul de caracteristici ale sistemului pe care utilizatorul și le amintește	Memorabilitate
Timpul petrecut în eroare	Erori
Procentul sau numărul erorilor	Erori
Numărul repetițiilor comenziilor eșuate	Erori
Numărul acțiunilor succesive ce conduc la erori	Erori
Timpul folosit pentru help și documentare	Învățare
Frecvența de folosire a helpului și documentației	Învățare

Tabela 2.3.1: Metrici ale utilizabilității

Componenta conceptuală
Nivelul sarcinii
Nivelul semantic
Componenta de comunicare
Nivelul sintactic
Nivelul de interacțiune
Componenta fizică
Nivelul dispunerii spațiale
Nivelul echipamentelor

Tabela 2.4.1: Structura nivelurilor CLG

Capitolul 3

Metode de descriere a interacțiunii om-calculator

Crearea de modele ale interacțiunii om-calculator presupune utilizarea unor notații. Cu cât metodele de modelare sunt mai formale, cu atât rezultatul aplicării modelului este mai ușor de anticipat și corectitudinea sa poate fi demonstrată. În acest capitol sunt descrise câteva din metodele de descriere a interacțiunii om-calculator, dezvoltate din diverse perspective: cea a sistemului, cea a utilizatorului, sau perspective care încearcă tratarea ambilor participanți la interacțiune.

Interacțiunea dintre om și calculator o regăsim uneori și sub denumirea de dialogul om-calculator sau om-mașină. Dialogul este o conversație între doi sau mai mulți participanți și implică un anumit nivel de cooperare. În proiectarea interfețelor calculator, dialogul are o semnificație specială și anume aceea de *structură a conversației dintre om și sistemul informatic*. Limbajul de comunicare cu calculatorul poate fi privit la trei niveluri:

1. lexical - nivelul cel mai de jos referindu-se la forma pictogramelor pe ecran, tastele care trebuie acționate (similar sunetelor și ortografiei cuvintelor);

2. sintactic - se referă la structura intrărilor și a ieșirilor (în limbajul uman e similar gramaticii construcției propozițiilor);
3. semantic - se referă la înțelesul conversației în termeni ai efectelor conversației asupra stării interne a sistemului sau a lumii exterioare (în lumea reală se referă la semnificația asociată de diferenții participanți la conversație).

Pornind de la modelul CLG [43], dialogul reprezintă nivelul sintactic al interacțiunii om-calculator, fiind similar uneori cu scenariul unei piese, doar că uneori utilizatorul sau calculatorul au mai multe opțiuni. Bariera dintre nivelul lexical și sintactic este însă neclară, astfel încât descrierile dialogului includ și caracteristici lexicale. Notațiile pentru descrierea dialogului pot să fie diagramatice - ușor de citit de la o primă privire sau textuale - ușoare pentru analiza formală. Dialogul este strâns legat de semantica sistemului (ce face) și de prezentarea sistemului (cum arată). Avantajul descrierilor formale este că pot fi verificate pentru acțiunile inconsistente, pentru acțiunile dificil reversibile, pentru absența unor elemente, pentru potențiale erori de tastare (miskeying errors).

3.1 Notații diagramatice

3.1.1 Rețele stare-tranziție

Rețelele stare-tranziție au fost utilizate intens în descrierea dialogului. Rețelele de tranziție au la bază *diagramele de tranziție*. O diagramă de tranziție constă dintr-un set de stări (reprazentate prin cercuri) și o mulțime de arce (reprazentate prin săgeți între stări). În cadrul unei rețele există o *stare inițială* unică (marcată printr-un arc neetichetat care are ca destinație starea inițială și nu are o sursă) și o *mulțime de stări finale* (marcate prin două cercuri concentrice). În cea mai simplă formă a rețelelor de tranziție, fiecare arc e etichetat cu o acțiune (simbol de

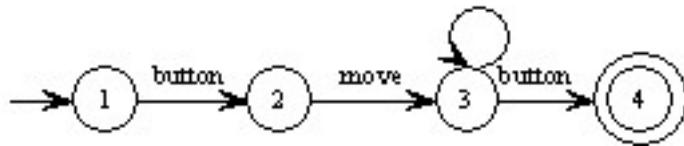


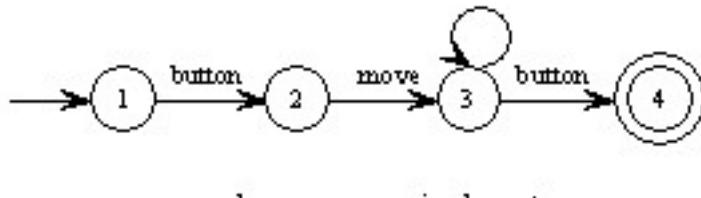
Figura 3.1: Diagrama de tranziție pentru desenarea unei linii curbe

intrare) pe care utilizatorul o poate efectua. Arcele din diagramă descriu modul de evoluție a dialogului de la o stare la alta, dependent de acțiunile utilizatorului. Dialogul poate evoluă de la starea A la starea B dacă între A și B există un arc etichetat. Un *drum* într-o diagramă de tranziție este o secvență de arce care duce de la starea inițială la o stare finală, iar o secvență de acțiuni ale utilizatorului este *acceptată* dacă etichetează arcele unui drum din diagramă.

Un exemplu de diagramă de tranziție pentru desenarea unei linii curbe este prezentat în Figura 3.1. Componenta de prezentare produce simbolul *button* când utilizatorul acționează butonul mouse-ului și simbolul *move* de câte ori utilizatorul mută cursorul mouse-ului.

Această formă simplă de tranziții descrie secvențele de acțiuni utilizator, dar nu surprinde răspunsurile generate de sistem. Pentru înlăturarea acestei deficiențe soluția este atașarea de acțiuni stărilor din diagramă. Interpretarea acestei notații este următoarea: la trecerea sistemului într-o stare se declanșează acțiunea atașată. Astfel, acțiunile utilizatorului sunt atașate arcelor, iar acțiunile sistemului sunt atașate stărilor. În urma atașării de acțiuni stărilor, exemplul anterior devine cel prezentat în Figura 3.2.

Din punct de vedere formal această notație poate fi reprezentată folosind un automat modificat, în care stările automatului corespund stărilor din rețeaua de tranziție, iar arcele sunt convertite în tranziții între stări.



- 1: memorarea primului punct
 2: deseneaza linia pana la pozitia curenta
 3: memorarea urmatorul punct

Figura 3.2: Diagrama de tranziție pentru desenarea unei linii curbe

Astfel, o rețea stare-tranziție se definește prin $\text{STN}^1 = (Q, \Sigma, P, \delta, \gamma, q_0, F)$, unde:

Q este o mulțime finită de stări;

Σ este o mulțime finită de simboluri de intrare (generate de componenta de prezentare);

P este o mulțime finită de acțiuni (cele care etichetează stările din rețeaua de tranziție);

$\delta : Q \times \Sigma \rightarrow Q$ este funcția de tranziție;

$\gamma : Q \rightarrow P$ este funcția de acțiune;

$q_0 \in Q$ este starea inițială;

$F \subset Q$ este mulțimea stărilor finale.

Rețelele stare-tranziție diferă foarte puțin față de un automat: configurația rețelei stare-tranziție este dată de starea curentă a automatului: când STN primește un simbol a , reprezentând o acțiune utilizator, starea sa se schimbă conform funcției de tranziție și în acest moment STN emite numele acțiunii care etichetează noua stare, adică $q' = \delta(q, a)$, iar acțiunea emisă este $p = \gamma(q')$. La începutul dialogului STN se află în starea inițială q_0 . Utilizatorul generează prin acțiunile sale asupra interfeței simboluri de intrare, iar STN trece dintr-o stare în alta, urmând arcele etichetate cu aceste simboluri de intrare. Sfârșitul dialogului are loc când

¹Simple Transition Network

STN ajunge în una dintre stările finale.

Această notație permite descrierea unui număr restrâns de rețele, iar descrierea devine mare. Controlul dimensiunii descrierii se face prin partitioarea rețelei în câteva diagrame și organizarea lor într-o ierarhie în care una din ele este diagrama principală, iar celelalte sunt subdiagrame. Folosirea subdiagramelor face ca dialogul să poată fi partitionat în unități logice (pot exista diagrame pentru fiecare din comenziile interfeței).

Un alt dezavantaj al rețelelor stare-tranzitie este acela că în cazul acțiunilor utilizator neprevăzute, rețelele stare-tranzitie împiedică continuarea dialogului, deoarece diagrama se află într-o anumită stare, iar acțiunea utilizator nu etichetează nici unul din arcele diagramei corespunzătoare respectivei stări. Există două soluții posibile într-o astfel de situație: ignorarea acțiunilor utilizatorului - soluție nerecomandată pentru că acesta nu va mai avea posibilitatea de a ajunge într-o altă stare sau folosirea unei etichete corespunzătoare unui arc ce corespunde oricărei acțiuni utilizator care nu se potrivește celorlalte etichete și acest arc va conduce la o stare unde se încearcă recuperarea erorii.

Dacă permitem ca diagramele să se apeleze ele însese, caz în care formalismul poartă numele de *rețele de tranziție recursive* sau RTN (Recursive Transition Networks), atunci puterea descriptivă a formalismului crește. O altă variantă a acestui formalism o reprezintă *rețelele de tranziție extinse* (sau ATN - Augumented Transition Networks). O ATN constă dintr-o mulțime de diagrame de tranziție, o mulțime de registri și o mulțime de funcții. Regiștrii pot reține o mulțime de valori arbitrară și aceste valori sunt vizibile doar în cadrul componentei de dialog (aplicația nu poate citi sau scrie aceste valori). Funcțiile pot efectua orice calcule pe valorile din regiștri, dar nu pot accesa valorile variabilelor din aplicație. Funcțiile sunt atașate arcelor din diagramă și sunt executate când se încearcă traversarea arcului; dacă valoarea funcției atașate arcului e adevărată, arcul poate fi traversat, altfel nu. O ATN poate descrie

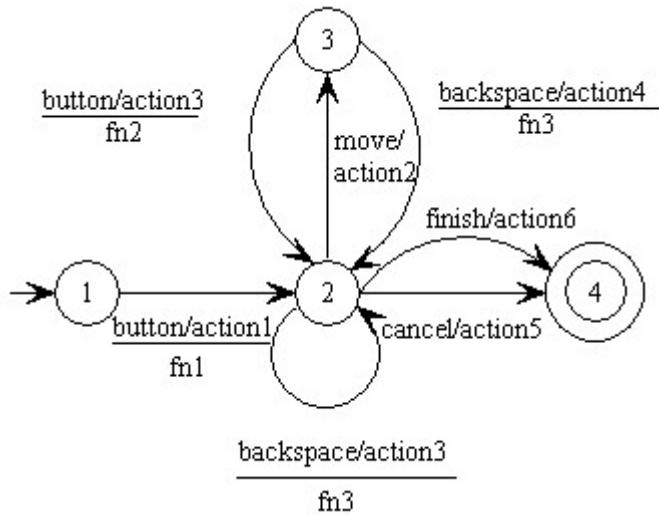


Figura 3.3: Desenarea unei linii poligonale cu opțiunea Cancel

dialoguri senzitive la context, spre deosebire de celelalte tipuri de rețele. În Figura 3.3 este prezentat un exemplu de ATN pentru desenarea unei linii poligonale cu posibilitatea anulării deciziilor anterioare.

În Figura 3.3, avem următoarea listă de acțiuni și funcții:

- action1: înregistrează primul punct;
- action2: desenează linia până la poziția curentă;
- action3: înregistrează următorul punct;
- action4: șterge ultimul punct;
- action5: șterge linia poligonală;
- action6: returnează linia poligonală.

```

fn1: count:=1; return (true);
fn2: count:=count+1; return (true);
fn3: if (count=1) then return (false)
      else count:=count+1; return (true).

```

Din punct de vedere formal, o ATN poate fi reprezentată folosind

un automat push-down, deoarece este nevoie de un mecanism de control al apelului recursiv al subdiagramelor.

Astfel, vom reprezenta o ATN prin $M = (Q, \Sigma, P, \Gamma, \gamma, \delta, q_0, Z_0, F)$

unde:

Q este mulțimea stărilor;

Σ este mulțimea de simboluri de intrare (generate de componenta de prezentare);

P este mulțimea de acțiuni care etichetează stările din diagramă;

Γ este mulțime finită de simboluri, câte un simbol pentru fiecare stare din Q și simbol din Σ ;

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times P \rightarrow Q \times \Gamma^*$ este funcția de tranziție;

$\gamma : Q \rightarrow P$ este funcția de acțiune;

$q_0 \in Q$ este starea inițială;

$Z_0 \in \Gamma$ este vârful memoriei stivă inițial;

$F \subset Q$ este mulțimea stărilor finale.

Configurația rețelei de tranziție este dată de o pereche ordonată (q, z) , unde q este starea automatului, iar z este conținutul memoriei stivă. De câte ori primește o intrare de la componenta de prezentare, rețeaua de tranziție trece la o nouă configurație și emite numele unei acțiuni. Dacă rețeaua de tranziție este în starea $(q, \alpha Z)$ și primește simbolul de intrare a , noua stare și simbolul din vârful stivei sunt date de $\delta(q, a, \alpha)$, iar acțiunea emisă e dată de $\gamma(q')$, unde q' este noua stare. La începutul dialogului configurația rețelei este (q_0, Z_0) . Configurația finală are forma (q, Z_0) , $q \in F$.

Metodele de descriere a dialogului bazate pe diagrame de tranziție au avantajul că dispun de o notație naturală, care poate fi ușor editată și afișată pe terminalele grafice. Primele sisteme de gestiune a interfețelor utilizator (UIMS) s-au bazat pe rețele de tranziție.

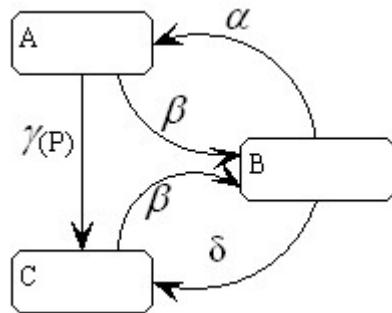


Figura 3.4: Diagramă stratificată de stări încă dinainte de clusterizare

3.1.2 Diagrame stratificate de stări (statecharts)

Diagramele stratificate de stări sunt folosite la specificarea sistemelor reactive² complexe [26]. Folosirea concomitentă a stărilor și a evenimentelor creează un mediu natural pentru descrierea comportamentului dinamic al unui sistem complex. Un aspect important în specificarea unui sistem interactiv este *tranzitia între stări*, căreia îi corespunde în limbaj natural o formulare de tipul: ”când evenimentul α are loc în starea A și condiția C este adevărată, atunci sistemul trece în starea B”. Diagramele de tranzitie exprimă astfel de situații, dar pentru sistemele complexe numărul de stări crește exponential și nu există posibilitatea ierarhizării modelului. Un formalism bun trebuie să permită clusterizarea stărilor, rafinarea, să exprime independența și ortogonalitatea. Diagramele stratificate de stări au posibilitatea de a exprima concurența și sprijină descrierea trecerilor de la un nivel de abstractizare la altul. Nucleul metodei constă în extinderea diagramele de tranzitie prin folosirea descompunerii folosind operatorii AND sau OR și un sistem de comunicare broadcast între componentele concurente.

Pentru descrierea diagramele se folosesc dreptunghiuri cu

²sisteme dirijate de evenimente care reacționaează în mod continuu la stimuli externi și interni

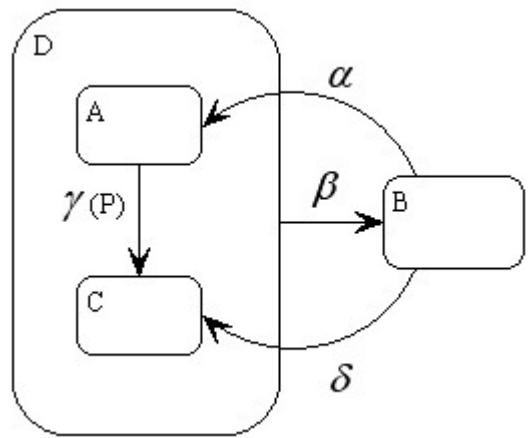


Figura 3.5: Diagramă stratificată de stări după aplicarea operatorului XOR

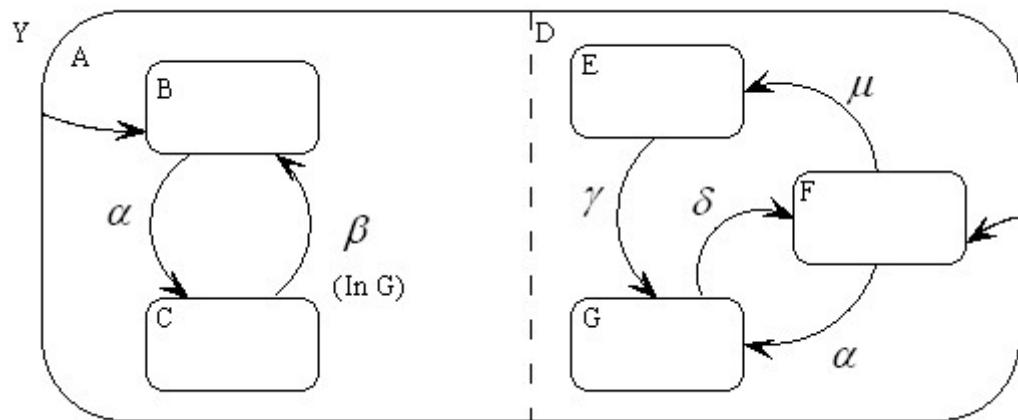


Figura 3.6: Ortogonalitatea în diagrame ierarhizate de stări

colțurile rotunjite care reprezintă stările și arce care reprezintă tranzițiile între stări (respectiv niveluri de abstractizare). Arcele sunt etichetate cu numele unui eveniment și cu o condiție descrisă între paranteze rotunde. Figura 3.5 prezintă un exemplu de diagramă stratificată de stări. Diagrama conține trei stări: A, B și C. Dacă apare evenimentul γ în starea A, sistemul trece în starea C dacă condiția P este îndeplinită. În Figura 3.4, deoarece evenimentul β afectează stările A și C, ele pot fi grupate într-o superstare D, iar cele două arce de la A, respectiv C spre B vor fi înlocuite prin unul singur, conform Figurii 3.5. Din punct de vedere semantic, starea D reprezintă XOR între stările A și C, adică pentru a fi în starea D sistemul trebuie să fie în una din stările A sau C, dar nu în amândouă simultan. Astfel, D este o abstractizare a stărilor A și C, sau A și C sunt o rafinare a stării D.

Ortogonalitatea este legată întrinsec de descompunerea stărilor folosind operatorul AND. Astfel, dacă starea Y este produsul ortogonal al stărilor A și D ($Y=A \text{ AND } D$), trebuie ca sistemul să fie simultan în stările A și D pentru ca sistemul să fie în starea Y. Figura 3.6 prezintă un exemplu de produs ortogonal. Starea inițială a diagramei este perechea (B, F). La apariția evenimentului α , B trece în C și F trece în G simultan. Este un caz de *sincronizare* (un eveniment provoacă două tranziții simultane). Spre deosebire de această situație, dacă în starea (B,F) apare evenimentul μ e afectată doar componenta D, iar sistemul va trece în starea (B,E), deci putem vorbi de *independență*. Ortogonalitatea e necesară pentru descrierea situațiilor în care starea sistemului este dependentă de componentele sale fizice.

3.1.3 Rețele Petri

O rețea Petri modelează un sistem printr-o mulțime de locații reprezentate prin elipse și o mulțime de operații sau tranziții reprezentate prin dreptunghiuri. Locațiile și tranzițiile sunt conectate prin arce

care definesc condițiile în care o tranziție poate avea loc și care sunt efectele sale. Declanșarea unei tranziții scoate cel puțin un simbol din fiecare locație de intrare și depune cel puțin un simbol în fiecare locație de ieșire [36]. Fiecărui arc îi este asociat un întreg n (pondere) care stabilește numărul de simboluri transferate între locațiile pe care le leagă la declanșarea unei tranziții. Starea sistemului este modelată în orice moment de distribuția simbolurilor în locațiile rețelei [47]. O rețea Petri este definită printr-un cvintuplu $(P, T, Pre, Post, M)$, unde:

P este mulțimea de locații;

T este mulțimea tranziților;

Pre este funcția de incidentă reprezentând arcele de intrare ale unei tranziții

$$Pre : P \times T \rightarrow \mathbb{N};$$

$Post$ este funcția de ieșire reprezentând arcele de ieșire ale unei tranziții

$$Post : P \times T \rightarrow \mathbb{N};$$

M este funcția de distribuție a simbolurilor în locații $M: P \rightarrow \mathbb{N}$.

Descrierea paralelismului într-o rețea Petri este surprinsă în Figura 3.7. După declanșarea tranziției T_1 vor exista două fire de control independente, iar după efectuarea tranziției T_4 apare o sincronizare a firelor de control generate de T_1 (cele două fire de control au nevoie să schimbe date sau să coopereze).

Rețelele Petri nu dispun de construcții pentru modelarea entităților, operațiilor și structurii sistemelor informatici. Abordarea orientată obiect dispune de cele mai bune abordări pentru reprezentarea acestor aspecte ale sistemelor informatici, însă există deficiențe în definirea explicită a comportamentului sistemului. În aceste condiții, cele

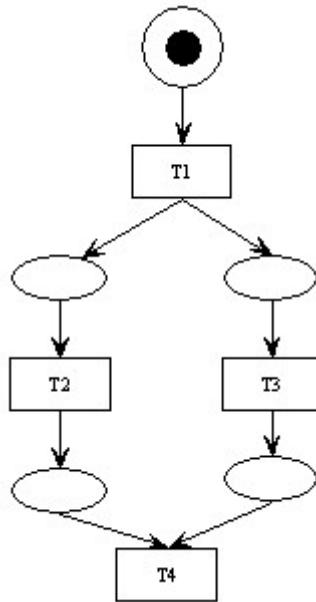


Figura 3.7: Paralelismul în rețele Petri

două abordări fiind complementare, s-au dezvoltat metode care încearcă îmbinarea avantajelor celor două metode, existând în prezent două variante: rețele Petri cu obiecte [63] în care locul simbolurilor este luat de obiecte și rețele Petri în obiecte [48], folosite la modelarea comportamentului obiectelor.

3.2 Notații textuale

3.2.1 Gramatici independente de context

Gramaticile independente de context descriu dialogul utilizator-sistem prin intermediul notației specifice gramaticilor. Motivația acestei abordări se găsește în faptul că dialogul om-calculator este un dialog asemenea celui interuman, iar dialogul interuman folosește un limbaj descris prin intermediul unei gramatici. În HCI, problema care apare este deter-

minată de faptul că utilizatorul folosește un limbaj pentru a da comenzi calculatorului, iar calculatorul folosește un alt limbaj pentru a comunica rezultatele. În practică, gramatica descrie doar limbajul folosit de utilizator pentru a comunica sistemului comenzi, iar răspunsul sistemului este descris prin alte mijloace. Terminalele gramaticii sunt simbolurile de intrare generate de componenta de prezentare și reprezintă acțiunile utilizatorului. Terminalele sunt combinate cu producțiile gramaticii pentru a crea structuri de nivel mai înalt, numite neterminale. Mulțimea de producții definește limbajul folosit în interacțiunea cu calculatorul. Producțiile unei gramatici folosite pentru a descrie desenarea unei linii curbe sunt:

$$\begin{aligned} \text{line} &\rightarrow \text{BUTTON end-point} \\ \text{end-point} &\rightarrow \text{MOVE end-point} \mid \text{BUTTON} \end{aligned}$$

După cum demonstrează exemplul, sunt descrise acțiunile efectuate de utilizator, dar nu sunt surprinse și răspunsurile sistemului. Pentru înlăturarea acestui neajuns s-a decis atașarea de acțiuni fiecărei producții a gramaticii, iar aceste acțiuni sunt efectuate la analiza sintactică a intrărilor furnizate de utilizator. Analiza sintactică poate fi efectuată după modelul top-down (producția e folosită când primul terminal care putea fi generat de partea dreaptă e întâlnit), sau bottom-up (o producție e folosită când toate simbolurile din partea dreaptă au fost regăsite). Folosind o analiză sintactică top-down, exemplul anterior devine:

$$\begin{aligned} \text{line} &\rightarrow \text{BUTTON } d1 \text{ end-point} \\ \text{end-point} &\rightarrow \text{MOVE } d2 \text{ end-point} \mid \text{BUTTON } d3 \\ d1 &\rightarrow \text{memorează primul punct} \\ d2 &\rightarrow \text{desenează linia până la poziția curentă} \\ d3 &\rightarrow \text{memorează următorul punct} \end{aligned}$$

Formal, o astfel de gramatică este reprezentată prin cvintuplul $G = (N, \Sigma, R, P, S)$, unde:

N este o mulțime finită de simboluri neterminale;

Σ este o mulțime finită de terminale (câte unul pentru fiecare

simbol de intrare);

R este o mulțime finită de simboluri ce reprezintă acțiunile atașate producțiilor;

P este mulțimea producțiilor de forma $n \rightarrow \gamma r$, unde $n \in N, \gamma \in (N \cup \Sigma)^*, r \in R$;

S este simbolul de start, $S \in N$.

Un aspect remarcabil în cazul gramaticilor este posibilitatea descrierii limbajului generat de gramatică: $L(G) = \{x | S \xrightarrow{*} x, S - \text{simbolul de start al gramaticii } G\}$. Dacă G descrie componenta de control al dialogului unei interfețe utilizator, $L(G)$ conține doar secvențele legale de acțiuni utilizator.

3.2.2 Formalismul bazat pe evenimente

Noțiunea de bază în acest formalism este aceea de *eveniment de intrare*, specifică pachetelor grafice, în care dispozitivele de intrare sunt privite ca surse de evenimente. Fiecare dispozitiv de intrare generează unul sau mai multe evenimente atunci când utilizatorul interacționează cu el, iar evenimentele au un nume (sau identificator) și o valoare care caracterizează interacțiunea. Evenimentele sunt memorate într-o coadă de evenimente și aplicația șterge aceste evenimente unul câte unul prin apelearea unei rutine din pachetul grafic. În pachetele grafice există un număr fix de evenimente predefinite care puteau fi generate doar de dispozitivele de intrare.

Formalismul bazat pe evenimente este o extensie a acestei idei, permitând existența unui număr arbitrar de tipuri de evenimente, unele generate de dispozitivele de intrare, altele de componenta de control al dialogului. În modelul bazat pe evenimente nu există cozi de evenimente explicate, ci un eveniment o dată generat e trimis unui handler de eveniment care este un proces definit printr-o procedură sau modul capabil să proceseze anumite tipuri de evenimente. Când un handler primește

un eveniment execută o procedură care poate efectua calcule, genera noi evenimente, apela proceduri ale aplicației, crea sau distrugă alți handleri. Comportamentul unui handler este definit de un *șablon* alcătuit din secțiuni care stabilesc parametrii handlerului, variabilele locale, evenimentele pe care le procesează și procedurile folosite pentru tratarea evenimentelor. La creare, unui handler trebuie să îl se specifice un șablon și valorile parametrilor, rezultând un nou identificator prin care ne vom referi la handler. Odată creat, un handler rămâne activ până la distrugere.

În modelul bazat pe evenimente, o interfață utilizator este descrisă prin mulțimea șabloanelor handlerilor pe care îi folosește. La începutul execuției este creată o instanță a uneia dintre șabloane care se va folosi ca handler de evenimente principal pentru interfață. Sursele de evenimente sunt reprezentate de componenta de prezentare care trimite simboluri spre componenta de control al dialogului care sunt convertite în evenimente, respectiv procesul, deoarece un handler poate trimite evenimente către unul sau mai mulți handleri.

Definiția unui handler conține patru secțiuni:

- TOKEN - definește simbolurile pe care handlerul le poate procesa;
- VAR - secțiunea de declarații ale variabilelor locale care sunt folosite de handler;
- secțiunea de definire a procedurilor de tratare a evenimentelor;
- INIT - secțiunea care conține instrucțiunile care trebuie executate la crearea handlerului.

Algoritmul 1 prezintă un exemplu de handler de eveniment pentru desenarea unei linii:

Formal, un handler de eveniment poate fi reprezentat prin $EH=(m, r, Q, P, R)$, unde:

m este numărul de evenimente procesate de handler;

Algorithm 1 EVENT HANDLER line

```
1: TOKEN
2: button Button
3: move Move
4: VAR
5: int state;
6: point first, last;
7: EVENT Button DO{
8: if state == 0 then
9:   first=current position;
10:  state=1;
11: else
12:   last=current position;
13:   deactivate(self)
14: end if}
15: EVENT Move DO{
16: if state == 1 then
17:   draw line for first to current position;
18: end if}
19: INIT
20: i=0;
21: END EVENT HANDLER line;
```

r este numărul de registri din handler ($m \leq r$);
 Q este coada de evenimente , $Q \in E^*$;
 R este mulțimea valorilor din registri;
 P este mulțimea de m proceduri pentru EH, câte o procedură pentru fiecare tip de eveniment procesat de handler.

Configurația unui handler este dată de conținutul Q al cozii de evenimente și de valorile din registri (q, ρ) , unde $q \in Q^*, \rho \in R^*$.

3.2.3 NUAN - New User Action Notation

NUAN este un succesor al notației UAN ³. UAN este o notație orientată pe utilizator și sarcină pentru reprezentarea comportamentului interfețelor concepute pentru manipularea directă a obiectelor și s-a dorit a fi un mecanism de comunicare între proiectanții de interfețe și implementatorii acestora. Specificarea este atât de detaliată încât nici un amănunt nu este lăsat la alegerea sau intuiția implementatorilor [27]. UAN descrie comportamentul utilizatorului și al interfeței în timp ce duc la îndeplinire o sarcină în mod cooperativ. Acțiunile utilizatorului, răspunsul interfeței și starea sistemului sunt descrise simultan, iar relațiile temporale indică ordinea efectuării acțiunilor. Tabelele 3.2.1 și 3.2.2 descriu notațiile UAN pentru acțiunile utilizatorului, respectiv răspunsul interfeței.

Tabela 3.2.3 prezintă un exemplu de descriere UAN pentru mutarea pictogramei unui fișier.

Ceea ce aduce nou NUAN este o descriere mai puțin criptică a operatorilor din UAN, care sunt înlocuiți prin scurte propoziții [78]. O primă extensie adusă la UAN este adăugarea unei coloane optionale pentru a descrie precondițiile care trebuie îndeplinite pentru ca interacțiunea descrisă să poată avea loc (de exemplu, pentru a șterge un fișier, acesta

³User Action Notation

Feedback	Semnificație
!	luminează obiectul (marchează selecția)
-!	deselectează obiectul
!-	clipire (blink)
(!-!) ⁿ	clipire de n ori
@x, y	la punctul (x,y)
@ X	la obiectul X
display (X)	afișează obiectul X
erase (X)	șterge obiectul X
X>~	obiectul X urmărește cursorul
X>>~	obiectul X este redimensionat corespunzător poziției cursorului
outline(X)	evidențiază obiectul X

Tabela 3.2.1: Notațiile UAN pentru răspunsul interfeței

trebuie mai întâi selectat). Coloana precondiției este situată în partea dreaptă a primei linii a tabelului NUAN, partea stângă a primei linii fiind folosită pentru adnotări (ceea ce descrie interacțiunea). O altă modificare adusă UAN este aceea că în NUAN interfața poate să ia singură inițiativa interacțiunii, nu doar să ofere feedback. Situația în care interfața ia inițiativa interacțiunii este descrisă prin folosirea unui semnal exclamării și este o situație întâlnită frecvent în cazul sistemelor de email, securitate sau diagrame de flux. Delimitarea momentului în care o acțiune are loc era confuz definită în UAN, astfel încât NUAN aduce precizări referitoare la acest aspect, astfel încât acțiunile care se găsesc pe linii diferite în tabelul NUAN se desfășoară succesiv, în timp ce acțiunile care se găsesc în aceeași linie a tabelului se desfășoară simultan. Dacă descrierea unor astfel de acțiuni nu începe într-o linie, atunci se folosesc mai multe linii situate în cadrul același celule. În cadrul descrierii NUAN sunt cuprinse și acțiunile mentale, care au fost introduse prima dată de GOMS. Includerea acțiunilor mentale poate să ofere informații despre aspectele

Acțiune	Semnificație
\sim	mută cursorul
$[X]$	“contextul” obiectului X
$\sim[X]$	mută cursorul în contextul obiectului X
$\sim[x,y]$	mută cursorul la punctul de coordonate (x,y) în afara oricărui obiect
$\sim[x,y \text{ in } A]$	mută cursorul la punctul (x,y) în interiorul lui A
$\sim[X \text{ in } Y]$	mută cursorul la obiectul X din cadrul obiectului Y
$[X]\sim$	mută cursorul în afara contextului lui X
v	apasă
\wedge	eliberează
Xv	apasă butonul sau tasta numită X
\bar{X}	eliberează butonul sau tasta numită X
$X\hat{v}$	click pe buton sau tastă numită X
X“abc”	introdu stringul “abc” folosind echipamentul X
X(xyz)	introdu valoare pentru variabila xyz folosind echipamentul X
*	încidere reflexivă - sarcina se execută de zero sau mai multe ori
+	sarcina se execută cel puțin o dată
	sarcină optională
A B	secvențiere - execută sarcina A, apoi sarcina B
OR	disjuncție
&	ordine independentă în efectuarea sarcinilor
\Leftrightarrow	intermitență - efectuarea sarcinilor se poate împieri
	concurrentă
;	întreruperea efectuării sarcinii
\forall	oricare
:	separator între condiție și acțiune sau feedback

Tabela 3.2.2: Notațiile UAN pentru acțiunile utilizatorului

Acțiunea utilizatorului	Răspunsul interfeței	Starea interfeței
[pictograma-fiș.]Mv	pictograma-fiș.-!: pictograma-fiș.!	selected=fiș.
	∀ pictograma-fiș.'!: pictograma-fiș.'-!	
[x,y]* [x',y']	marchează(pictograma-fiș.) > ~	
M^	@[x',y'] display(pictograma-fiș.)	

Tabela 3.2.3: Exemplu UAN pentru mutarea pictogramei unui fișier

cognitive ale interacțiunii, astfel putându-se evalua din descrierea NUAN dacă utilizatorul poate lua o decizie pe baza elementelor vizibile pe ecran sau pe baza conținutului memoriei de lucru.

În Tabela 3.2.4 este descrisă o interacțiune folosind NUAN referitoare la primirea unui email.

3.3 Modele ale utilizatorului în descrierea interacțiunii om-calculator

O categorie aparte de metode de descriere a interacțiunii sau dialogului om-calculator o reprezintă gramaticile care își propun descrierea acestei interacțiuni pornind sau având ca și element central acțiunile utilizatorului. Aceste metode de descriere s-au născut din dorința de a înțelege și anticipa acțiunile utilizatorului astfel încât proiectarea dialogului să fie cât mai aproape de necesitățile și așteptările acestuia.

Email nou			
Despre		Starea interfeței	
<i>Interacțiunea descrie sosirea unui email nou</i>			<i>programul de email=rulează minimizat</i>
Acțiuni utilizator	Acțiuni ale interfeței	Starea interfeței	Conexiunea la logica aplicației
	!MESSAGE('New email has arrived')	mes_wnd=visible	unread_mail=true
	!ASK('Read now?' [Yes, No])		
DECIDE([yes,no])			
POINTERTO (<yesbutton>) CLICK (<yesbutton>) 	MOVEPOINTER (<yesbutton>) SHOW_EMAIL([latest])		unread_mail=false
POINTERTO (<nobutton>)	MOVEPOINTER (<nobutton>)		unread_mail=true
	HIDE_MESSAGE()	mes_wnd=hidden	

Tabela 3.2.4: Exemplu NUAN

3.3.1 GOMS - Goals, Operators, Methods and Selection Rules

Modelul GOMS a fost dezvoltat de Card, Moran și Newell [14] în anul 1983 cu scopul de a furniza predicții cât se poate de precise legate de comportamentul uman în fața interfeței și de timpul necesar îndeplinirii unei sarcini [12], fără a cere un efort prea mare din partea celor implicați. GOMS a fost concepută ca un model cognitiv aproximant pentru procesarea informației de către utilizatori. Un model GOMS este o descriere a cunoștințelor pe care trebuie să le aibă un utilizator pentru a putea efectua sarcini folosind un sistem; este o reprezentare a cunoștințelor "how to do it" solicitată de un sistem pentru a îndeplini o sarcină. O dată ce modelul a fost dezvoltat se poate face predicții asupra performanțelor și timpului de învățare.

Modelul se bazează pe patru componente: *scopuri* (goals), *operatori*, *metode* și *reguli de selecție*.

Scopul este o structură simbolică care definește o stare ce se dorește să fie atinsă și determină o mulțime de metode posibile prin care poate fi atinsă. Descrierea unui scop ia forma unei perechi <acțiune-subiect> (de exemplu: delete-word). Un *operator* este un motor elementar de procesare a informației (act perceptiv, cognitiv sau motoriu), a cărui execuție modifică în mod necesar mediul sarcinii și orice aspect al stării mentale a utilizatorului (de exemplu: apăsarea unei taste, căutarea unei opțiuni dintr-un meniu).

O *metodă* descrie o procedură pentru îndeplinirea unui scop și constă dintr-o serie de pași corespunzători operatorilor pe care utilizatorul îi folosește.

Regulile de selecție sunt folosite deoarece, atunci când se urmărește un scop, pot exista mai multe metode de îndeplinire a sarcinii. Alegerea metodei se face prin selecția de reguli care depind de caracteristicile mediului în care se desfășoară sarcina.

Modelul GOMS funcționează împărțind sarcina într-o stivă de scopuri și specificând operatorii, metodele și regulile de selecție între metodele alternative. Acest model poate fi folosit pentru a prevedea "dru-mul" utilizatorului prin sarcină [11]. Sarcina supusă analizei este descompusă în subsarcini succesive până se ajunge la o "sarcină unitate" (de exemplu: o corectură într-un manuscris poate fi considerată o sarcină unitate).

În cele ce urmează este prezentat un exemplu de analiză GOMS pentru sarcina de selectare a unui text.

```
Selection rule set for goal: select text
If text-is word, then
    Accomplish goal:select word.
If text-is arbitrary, then
    Accomplish goal:select arbitrary text.
Return with goal accomplished.
```

Method for goal: select word

```
Step 1. Locate middle of word. (M)
Step 2. Move cursor to middle of word. (P)
Step 3. Verify that the correct word is located. (M)
Step 4. Double click mouse button. (BB)
Step 5. Verify that the correct text is selected. (M)
Step 6. Return with goal accomplished.
```

Method for goal: select arbitrary text

```
Step 1. Locate beginning of text. (M)
Step 2. Move cursor to beginning of text. (P)
Step 3. Verify that the correct beginning is located. (M)
Step 4. Press mouse button down. (B)
Step 5. Locate end of text. (M)
Step 6. Move cursor to end of text. (P)
Step 7. Verify that correct text is marked. (M)
Step 8. Release mouse button. (B)
Step 9. Verify that the correct text is selected. (M)
Step 10. Return with goal accomplished.
```

Pentru estimarea timpului de realizare a unei sarcini se folosește modelul Keystroke Level Model (KLM) [15] care definește un set de operatori de bază împreună cu timpii necesari execuției acestora. Acești operatori sunt prezenți în cele ce urmează:

K reprezintă apăsarea unei taste;
B reprezintă apăsarea unui buton al mouse-ului;
P reprezintă mutarea cursorului mouse-ului (sau altui dispozitiv de indicare);
H reprezintă mutarea mâinii între tastatură și mouse;
D reprezintă desenarea de linii folosind mouse-ul;
M reprezintă pregătirea mentală pentru o acțiune fizică;
R reprezintă răspunsul sistemului ; poate fi ignorat dacă utilizatorul nu trebuie să aștepte acest răspuns (de exemplu: copierea).

Adâncimea stivei de scopuri poate furniza predicții legate de cerințele relative memoriei de scurtă durată în îndeplinirea sarcinilor, iar 90% din predicțiile legate de comenziile executate de utilizatori în îndeplinirea sarcinii sunt corecte. Timpul de execuție a sarcinii este estimat cu o eroare de 33%.

Dezavantajul GOMS este acela că descrierea necesită comportament de expert (nu se surprinde în nici un fel situația de eroare) și nu se cunoaște cât de adânc trebuie să se meargă cu descompunerea în procesul de analiză.

3.3.2 CCT - Cognitive Complexity Theory

CCT [38] pornește de la descompunerea scopurilor, similar GOMS, dar îmbogățește modelul pentru a-i da mai multă putere predicтивă. CCT folosește două descrieri paralele: una a scopurilor utilizatorilor, iar celalaltă a sistemului. Descrierea scopurilor utilizatorilor este prezentată într-o manieră similară GOMS, dar folosește reguli de producție, iar pentru descrierea stării sistemului se folosesc rețele stare-tranzitie.

Regulile de producție au forma:

if *condition* then *action*

unde *condition* este o condiție relativă la conținutul memoriei de lucru, iar dacă această condiție este adevărată regula este activată. O

acțiune poate să consiste în una sau mai multe acțiuni elementare care pot fi schimbări ale conținutului memoriei de lucru sau acțiuni externe precum acționările de taste. Regulile de producție sunt exprimate într-un limbaj similar LISP. Pentru exemplificare se descrie sarcina ștergerii unui caracter situat la o anumită linie (lin) și coloană (col) într-un text.

```
(PRDELC1
  IF (AND (TEST-GOAL delete character)
            (NOT (TEST-GOAL move cursor to %LINE %COL))
            (NOT (TEST-CURSOR %LINE %COL)))
      )
  THEN
    ((ADD-GOAL move cursor to %LINE %COL))
  )

(PRDELC2
  IF (AND (TEST-GOAL delete character)
            (TEST-CURSOR %LINE %COL)
            )
  THEN ((DO-KEYSTROKE DELETE)
        (WAIT)
        (DELETE-GOAL delete character)
        (UNBIND %LINE %COL)
        )
  )
)
```

CCT poate oferi predicții asupra complexității interacțiunii cu o interfață, iar dubla reprezentare a stării sistemului și a conținutului memoriei de lucru a utilizatorului poate furniza informații despre problemele care apar la nivelul mapării sarcină-acțiune (între ceea ce trebuie făcut și cum se poate face).

3.3.3 TAG - Task-Action Grammar

TAG se bazează pe notația Backus-Naur Form (BNF), dezvoltată în 1986 de Payne și Green [51] în scopul de a descrie modul în care utilizatorul mapează modelul conceptual al sistemului în acțiuni asupra sistemului.

Se consideră că principala cauză a problemelor întâmpinate de utilizator în interacțiunea cu un sistem e lipsa unui model de mapare sarcină-acțiune adecvat care să stabilească relația între ceea ce știm și ceea ce putem face. TAG își propune să prezinte un model despre cum relaționăm ceea ce știm cu ceea ce putem face [11]. Scopul TAG a fost formalizarea modului de mapare a nivelului sarcinii la nivelul interacțiunii într-o manieră în care metrii simple asupra gramaticii să reflecte complexitatea cognitivă a acestei mapări. Analiza unei sarcini folosind TAG presupune descompunerea sa în sarcini simple (cognitiv automate). O sarcină simplă este o sarcină pe care utilizatorul o realizează în mod curent și poate fi la un nivel inferior pentru novici sau mai complex în cazul utilizatorilor experimentați. Spre exemplu sarcina ”mută cursorul cu un caracter mai sus” poate solicita pentru un utilizator novice mai multe acțiuni, însă pentru un utilizator experimentat se va reduce la o acțiune simplă.

TAG oferă două mecanisme de captare a generalităților în cadrul limbajului de comandă. În primul caz, sarcinile simple pot fi definite ca având caracteristicile altor sarcini simple, dar cu una sau mai multe diferențe specifice. Exemplul următor descrie folosind TAG modalitățile de operare cu fișiere sub sistemul de operare Unix:

```
file_op[Op] := command[Op]+ filename+filename|command[Op]+filenames+directory  
command[Op=copy]:=cp  
command[Op=move]:=mv
```

Cel de-al doilea mecanism permite gramaticii TAG să capteze diferitele forme ale cunoștințelor semantice generale în reguli separate. TAG e capabilă să exprime faptul că un simbol al limbajului de comandă e bazat pe cunoștințele limbajului natural ale utilizatorului. Regulile de acest tip îi dau gramaticii TAG un mecanism puternic de captare a informației relevante pentru interacțiune care nu sunt specificate în limbajul în sine. Datorită abilității de a capta influențe din limbajul natural, TAG poate indica potențialele erori datorate alegerii numelor comenzilor. Cunoștințe

despre alte sisteme sau despre versiuni de realizare a unei sarcini pot fi introduse în TAG pentru a demonstra efectele pe care le are interferența acestor surse de cunoștințe. În exemplul următor este descris modul de lansare a comenzi de mutare a unui obiect pe o anumită distanță într-o anumită direcție. Regulile notate cu asterics se consideră a face parte din cunoștințele utilizatorului și nu vor fi luate în calcul la estimarea complexității cognitive a sarcinii.

```
movement[Direction]:=command[Direction] + distance +RETURN
command[Direction]:=known-item[Type=word, Direction]
*command[Direction=forward]:=FORWARD
*command[Direction=backward]:=BACKWARD
*command[Direction=left]:=LEFT
*command[Direction=right]:=RIGHT
```

Consistența modelului e dată de faptul că regulile sunt valabile pentru orice entitate; dacă nu ar fi fost aşa, ar fi fost nevoie de reguli suplimentare pentru a surprinde cazuri speciale, iar numărul de reguli se dorește să reflectă complexitatea cognitivă a unei sarcini. Avantajul TAG asupra BNF este că surprinde inconsistențe pe care BNF nu le sesizează (de exemplu: ordinea acțiune-obiect sau obiect-acțiune în execuția unor comenzi), iar dezavantajul este că inconsistențelor mici le dă aceeași importanță ca și celor majore.

3.4 Limbaje de marcare

Limbajele de marcare au fost folosite în descrierea interfețelor utilizator o dată cu extinderea folosirii aplicațiilor soft pe dispozitive diverse, cu capacitați și resurse variate, care impun restricții asupra interfețelor care pot fi rulate pe aceste dispozitive. Limbajele de marcare permit descrierea abstractă a interfețelor utilizator urmată de generarea implementării concrete pentru dispozitive diverse sau în limbaje de programare diferite folosind instrumente specifice, numite *rendere*. În cadrul aces-

tei secțiuni sunt descrise două limbaje de marcăre cu impact în mediul dezvoltatorilor de soft.

3.4.1 UIML - User Interface Markup Language

Ca și alte limbaje de marcăre, UIML [42] dă o descriere declarativă a unei interfețe. Un limbaj declarativ este mai abstract decât un limbaj imperativ, iar o abstractizare mai mare asigură o portabilitate mai mare.

UIML este un limbaj de marcăre universal proiectat pentru a:

- oferi o separare naturală între codul interfeței utilizator și codul aplicației;
- facilită reutilizabilitatea pentru non-programatori;
- reduce timpul de dezvoltare a interfețelor pentru familii multiple de dispozitive;
- permite prototipizarea rapidă a interfețelor;
- facilită internaționalizarea și localizarea;
- permite descărcarea eficientă a interfețelor prin rețea în browserele Web;
- asigura securitatea;
- asigura extensibilitatea pentru viitoarele tehnologii [6].

Un document UIML poate fi folosit în diverse forme. În prima formă, acesta poate fi stocat pe un server, iar când un utilizator invocă o aplicație, documentul fie este compilat pentru limbajul platformei destinație (de exemplu: WML sau C++), fie este interpretat în timp ce utilizatorul interacționează cu interfața. Compilarea este obligatorie pentru dispozitivele care nu pot descărca aplicații, precum telefoanele

celulare, sau a căror memorie este limitată. UIML poate fi folosit și fără un server, în acest caz UIML sau codul compilat sau limbajul de marcat este instalat o dată cu aplicația pe dispozitivul utilizatorului. La execuție, interfața generată comunică în mod direct cu logica aplicației.

Avantajul rulării unui interpreter la client este evitarea transmiterii de cod executabil spre dispozitive. Spre exemplu, appleturile Java consumă mult timp la încărcare, iar dacă există un interpreter Java este necesară doar transmiterea documentului UIML care este un fișier text mic comparativ cu codul Java. Un alt aspect pozitiv al transmiterii documentelor UIML este securitatea, probabilitatea de a conține viruși sau de a lansa atacuri la client fiind mai mică decât în cazul executabilelor [7].

UIML este un metalimbaj analog lui XML. UIML nu conține taguri specifice unui toolkit particular unei interfețe utilizator (de exemplu: <WINDOW> sau <MENU>), ci folosește în schimb un set de taguri generice (de exemplu: <part>, <property>).

Arhitectura documentelor UIML

Scheletul unui document UIML este:

```
<?xml version='1.0'?>
<!DOCTYPE uiml PUBLIC '-//UIT//DTD UIML 2.0 Draft//EN' ''http://uiml.org/dtds/UIML20.dtd'>
<uiml>
  <head> <meta> ... </meta> </head>
  <peers>

    <presentation>
      <component> ... </component>
    </presentation>

    <logic>
      <component> ... </component>
    </logic>

  </peers>
  <template> ... </template>
```

```
<interface> ... </interface>
</uiml>.
```

Prima linie identifică documentul UIML ca fiind un document XML, iar cea de-a doua linie dă locația documentului DTD care descrie structura sintactică la care se va conforma documentul.

Elementul *<head>* conține metadate despre documentul UIML curent, iar elementele din cadrul tagului *<head>* nu sunt considerate parte a specificării interfeței. Elementul *<peers>* conține mapările între descrierea abstractă a interfeței și implementarea lor actuală (widgeturi din toolkituri, atribute, evenimente specifice platformei și componente ale logicii aplicației). Elementul *<presentation>* este folosit în definirea vocabularelor pentru descrierea documentelor UIML, definind numele legale de clase folosite în elementele *<part>*, valorile legale ale proprietăților fiecărei clase de elemente. Elementul *<logic>* descrie modul în care interacționează interfața cu aplicația de bază care implementează funcționalitatea manifestată prin intermediul interfeței, descriind convențiile de apelare a metodelor din logica aplicației. Elementul *<template>* permite reutilizarea elementelor interfeței. Când un element apare înăuntrul unui template el poate fi inclus altundeva în documentul UIML sau chiar în alt document UIML. Elementul *<interface>* conține descrierea interfeței și este elementul central al documentului UIML. Toate elementele care descriu interfața utilizator sunt cuprinse în acest tag. Un schelet al elementului *interface* este următorul:

```
<interface>
  <structure> <part> ... </part> </structure>
  <style> <property> ... </property> </style>
  <content> <constant> ... </constant> </content>
  <behavior> <rule> ... </rule> </behavior>
</interface>.
```

Elementul *<structure>* se referă la organizarea fizică a interfeței utilizator și include relațiile între elementele care compun interfața. Fiecare tag *<structure>* este compus din mai multe elemente *<part>*,

iar fiecare element *<part>* descrie elementul interfeței utilizator specific platformei și este asociat unei singure categorii de elemente ale interfeței utilizator. Elementul *<style>* conține o listă a proprietăților și valorilor folosite în procesul de transformare a specificării în interfață. Proprietățile UIML specifică atribute ale modului în care interfața va fi transformată pe diferite dispozitive, precum culori, fonturi, layout, etc. Elementul *<content>* asociază cuvinte, sunete, imagini cu părți ale interfeței pentru a facilita internaționalizarea și customizarea pentru diferite gru-puri de utilizatori, iar elementul *<behavior>* enumerează o mulțime de reguli care descriu modul în care interfața ar trebui să reacționeze la diferenți stimuli (din partea utilizatorului, echipamentelor sau logicii aplicației) [7], [6]. Există patru tipuri de acțiuni care pot avea loc: atribuirea unei valori unei proprietăți a unei părți, apelul unei funcții sau metode externe, tratarea unui eveniment sau restructurarea interfeței. Scheletul elementului *<behavior>* este următorul:

```

<behavior>
  <rule>
    <condition>
      <event ...> ... </event>
    </condition>
    <action>
      <property ...> ... </property>
      <method ...> ... </method>
      <event ...> ... </event>
    </action>
  </rule>
  ...
</behavior>.
```

Elementul *<behavior>* conține una sau mai multe reguli. Fiecare regulă constă dintr-o condiție, declanșată fie la apariția unui eveniment, fie când apare un eveniment și un atribut al evenimentului are o anumită valoare. La declanșare, acțiunea asociată condiției este executată și ea poate schimba proprietatea unei părți a interfeței, invoca o acțiune dintr-un script sau apela o metodă din logica aplicației sau genera un nou eveniment.

Maparea documentului UIML la platforma destinație

UIML nu conține nici o informație specifică platformelor, astfel încât nu poate fi folosit direct în forma în care e. Elementul `<peers>` exprimă mapările dintre elementele interfeței și componentele platformei actuale, iar pentru generarea interfețelor specifice platformelor sunt necesare rendere care, pe baza unor vocabulare, translatează specificarea ca document UIML a interfeței la interfața executabilă dorită. Un exemplu de mapare a unei specificări în Java AWT, WML și VoiceXML este prezentată în cele ce urmează:

```
<peers>
  <presentation name="WML">
    <component name="Dialog" maps-to="wml:card"/>
  </presentation>

  <presentation name="VoiceXML">
    <component name="Dialog" maps-to="vxml:form"/>
  </presentation>

  <presentation name="Java-AWT">
    <component name="Dialog" maps-to="java.awt.Dialog"/>
  </presentation>
</peers>.
```

O alternativă de folosire a elementului `<peers>` este următoarea:

```
<peers>
  <presentation base='Java\_1.3\_Harmonia\_1.0' />
</peers>
```

în care se specifică faptul că la generarea interfeței se va folosi vocabularul specificat ca valoare a proprietății *base*.

3.4.2 SUNML - Simple Unified Natural Markup Language

SUNML este un limbaj de specificare a interfeței utilizator bazat pe XML. Diferența dintre SUNML și celelalte limbaje de specificare a

interfeței utilizator (UIML, XIML, XUL) este aceea că SUNML sprijină adaptarea interfeței utilizator și compunerea dinamică [24, 52]. SUNML se bazează pe o mulțime redusă de taguri corespunzătoare principalelor componente ale interfeței utilizator:

- *Element* - un element sprijină intrările și ieșirile de la/spre utilizator. Un element are un tip (string, întreg, boolean, real). Este un tag esențial în descrierea interfețelor utilizator, putând avea reprezentări multiple (etichetă, câmp text, sunet);
- *Link* - este o componentă care declanșează execuția unei acțiuni sau a unui proces. Acest tag corespunde în general butoanelor sau opțiunilor din meniuri sau hiperlegăturilor din paginile web;
- *List* - o listă este o grupare de controale de același tip;
- *Dialog* - un dialog este o listă de controale având tipuri diferite (ar putea fi un panou sau o fereastră). Componentele unui dialog pot fi prezentate secvențial sau simultan;
- *Style* - acest tag se referă la prezentarea controalelor și poate fi specificată folosind fișiere CSS.

Pentru reutilizarea unor părți ale interfeței utilizator, SUNML oferă tagul *<reference>* care are un atribut *file* unde se specifică numele fișierului care conține descrierea interfeței utilizator.

În ultima perioadă au fost dezvoltate numeroase alte limbiage de formatare pentru specificarea interfețelor utilizator precum: XIML (eXtended Interface Markup Language [3], Alternate Abstract Interface Markup Language (AAIML)[2], Abstract User Interface Markup Language (AUIML) [1], Extensible User Interface Language (XUL) [4], Microsoft Extensible Application Markup Language (XAML) [5].

Capitolul 4

Analiza sarcinilor în proiectarea sistemelor interactive

În acest capitol se realizează o incursiune în domeniul proiectării sistemelor interactive bazată pe analiza sarcinilor. Sunt prezentate inițial noțiunile de bază folosite în analiza sarcinilor, urmate de prezentarea cătorva metode de analiza sarcinilor, cu evidențierea acelor metode cu aplicabilitate sporită în proiectarea sistemelor interactive. Câteva din instrumentele destinate analizei sarcinilor și proiectării de sisteme bazate pe analiza sarcinilor sunt prezentate. Deoarece pentru dezvoltatorii de sisteme este esențială transformarea corespunzătoare a modelelor în implementări, capitolul prezintă abordări formalizate pentru generarea modelului dialogului pornind de la modelul sarcinilor, respectiv generarea unei definiții abstracte a interfeței utilizator pornind de la modelul sarcinilor. Aceste două transformări ale modelului sarcinilor asigură dezvoltarea de soft conceput pentru utilizabilitate. Capitolul cuprinde o descriere a metodei de proiectare a sistemelor interactive bazată pe analiza sarcinilor numită DUTCH (Designing for Users and Tasks from Concepts to

Handles).

4.1 Analiza sarcinilor - noțiuni introductive

Analiza muncii este o ramură a psihologiei muncii, iar istoria sa începe din 1967, când Annette și Duncan [8] au fundamentat prima teorie referitoare la analiza muncii, și anume, Hierarchical Task Analysis (HTA). În analiza muncii conceptul fundamental utilizat este acela de *sarcină*. Leplat [40] definește sarcina a fi un obiectiv de atins în condiții (fizice, tehnice, organizaționale, socio-economice) determinate. *Activitatea* este răspunsul individului la aceste condiții - ceea ce face persoana pentru a realiza atât sarcina cât și propriile finalități. *Analiza sarcinilor* este procesul de culegere de date despre acțiunile pe care oamenii le efectuează [78], iar *modelarea sarcinilor* este crearea de reprezentări ale activităților pe care oamenii le efectuează.

Metode și tehnici de culegere a datelor

Analiza sarcinilor poate să utilizeze una sau mai multe din metodele disponibile de colectare a informațiilor despre sarcină, în funcție de tipul activității analizate, scopul analizei, condițiile de realizare a analizei sau experiența analistului. Tehnicile de colectare a datelor se pot grupa în trei categorii:

1. **Tehnici de interogare** - sunt tehnici verbale unde subiectului, care se află în afara situației reale a sarcinii i se cere să-și remintească, să explice, să reflecteze asupra comportamentului său de realizare a sarcinii. Din această categorie fac parte următoarele tehnici: interviul structurat, interviul focalizat, introspectia.
2. **Tehnici observaționale** - subiectul se află într-o situație reală

sau simulată de realizare a sarcinii. Tehnicile pot să presupună sau nu realizarea unui protocol verbal. Din această categorie fac parte tehnici precum: protocolele verbale și tehnica “gândirii cu voce tare”, tehnica ”teach-back”.

3. **Tehnici multidimensionale** - oferă în general date non-verbale; rolul lor este de a forța subiectul să gândească despre domeniul sarcinii într-un mod nou, care uneori poate să pară fără legătură cu sarcinile pe care le realizează. Din această categorie fac parte tehnici precum: grila-inventar, sortarea și clasificarea [29].

O dată culese datele, acestea trebuie structurate și înțelese, acest proces purtând numele de *modelarea sarcinilor*. Modelele sarcinilor sunt rezultatele (produsele) întregii activități de analiză a sarcinilor. Nici analiza sarcinilor și nici modelarea sarcinilor nu sunt subiecte de discuție noi, dar și-au recâștigat atenția o dată cu creșterea interesului pentru proiectarea centrată pe utilizator.

Analiza sarcinilor constituie punctul de pornire în dezvoltarea oricărui sistem/interfețe utilizabile. În proiectarea, validarea și exploatarea sistemelor informatici, analiza sarcinilor poate contribui la realizarea a două obiective principale:

1. ameliorarea caracteristicilor activității umane în sistemele existente;
2. proiectarea și dezvoltarea unor sisteme noi.

Cele două obiective influențează modalitatea de realizare a analizei. Astfel, dacă scopul este proiectarea și dezvoltarea unui nou sistem, este esențial să se analizeze detaliat scopurile și funcțiile sistemului, dar sunt luate în considerare și cerințele formale ale performanței umane, deoarece acestea permit stabilirea unui cadru conceptual adecvat în proiectarea activității și a condițiilor de realizare a sarcinilor. În cazul siste-

melor deja dezvoltate, interesul se îndreaptă spre analiza performanțelor reale și ameliorarea acestora [29].

Scopul analizei sarcinilor este de a culege suficientă informație despre mediul sarcinii pentru a face posibilă proiectarea interfeței utilizator.

În proiectarea interfețelor bazată pe sarcini, modelele sarcinilor sunt sursa primară în determinarea funcționalității aplicației și a modului de structurare a interacțiunii. Modelele sarcinilor sunt folosite pentru restructurarea muncii și pentru a oferi detalii despre utilizatori și despre alte persoane afectate de folosirea sistemului. Aceste informații sunt extrem de importante atunci când se proiectează pentru utilizabilitate.

4.1.1 Analiza sarcinilor în proiectarea sistemelor interactive

Orice metodă de analiză a sarcinilor implică stabilirea domeniului problemei pe care utilizatorul încearcă să o rezolve cu sistemul și formularea sa în termenii scopurilor, intențiilor și operațiilor. În HCI analiza sarcinilor se folosește în următoarele trei moduri:

- descrierea sarcinilor utilizatorului și mediului sarcinilor (modelul *descriptiv* al sarcinilor);
- analiza consecințelor deciziilor de reproiectare a activității (proiectare a sarcinilor - modelul *prescriptiv* al sarcinilor);
- analiza sarcinilor relativ la modul în care este sau ar trebui să fie efectuată cu o interfață utilizator particulară (analiza interfeței utilizator sau evaluarea interfeței utilizator).

Rezultatul analizei sarcinilor este constituit de *modelul sarcinilor* care trebuie să stea la baza procesului de proiectare a sistemelor interactive. Proiectarea bazată pe modele nu este o practică nouă, modelele fiind folosite datorită sprijinului pe care îl oferă în înțelegerea aspectelor esențiale

ale lumii înconjurătoare. Scopul proiectării bazate pe modele este să identifice modele de nivel înalt care le permit proiectanților să specifice și să analizeze aplicațiile soft interactive de la un nivel orientat spre semantică mai degrabă decât să înceapă prin referirea la nivelul de implementare. Acest fapt le permite să se concentreze asupra aspectelor esențiale fără a fi confuzi în legătură cu detaliile de implementare, iar mai apoi să disponă de instrumente care actualizează implementarea în concordanță cu deciziile de nivel înalt. Prin folosirea modelelor care captează aspectele semantice de interes proiectanții pot gestiona mai ușor complexitatea crescândă a sistemelor interactive și să le analizeze atât în faza de dezvoltare a lor, cât și atunci când trebuie modificate sau evaluate.

Există diferite tipuri de modele care sunt folosite în proiectarea sistemelor interactive. Putem menționa tehniciile de modelare orientate pe obiecte, cea mai de succes fiind UML [39], similară cu tehniciile orientate pe sarcini din perspectiva faptului că folosesc descrieri ale obiectelor și sarcinilor, dar care diferă prin notațiile orientate pe obiecte folosite și prin punctele de interes. Abordările orientate pe sarcini identifică mai întâi sarcinile și apoi obiectele care trebuie manipulate. Metodele orientate pe obiecte urmează un drum invers, deoarece urmăresc modelarea obiectelor care alcătuiesc sistemul. În consecință, abordările orientate pe sarcină sunt mai potrivite pentru proiectarea aplicațiilor centrate pe utilizator pentru că atenția este îndreptată spre a sprijini efectiv și eficient sarcinile utilizatorului (să nu uităm că unul din cele mai importante principii de utilizabilitate spune: “concentrează-te asupra utilizatorului și sarcinilor sale”), pe când tehniciile orientate pe obiecte au succes la ingineria nivelului de implementare.

În proiectarea sistemelor interactive trebuie efectuate următoarele activități relative sarcinilor:

1. identificarea domeniului sarcinilor, constrângerilor, competențelor și preferințelor oamenilor și mediului în care se desfășoară activitatea;

2. identificarea acelor sarcini care pot fi efectuate în modul cel mai eficient de către calculator;
3. proiectarea de modele ale domeniului activității (structuri de date) care permit efectuarea computerizată a unui număr maxim de subsarcini;
4. identificarea și dezvoltarea proceselor care permit calculatorului efectuarea acestor sarcini;
5. dezvoltarea interfeței utilizator care acoperă golul dintre nevoile utilizatorului, concepția acestuia asupra activității și modelul sarcinilor. Interfețele bine proiectate realizează o mapare naturală între domeniul conceptual al proiectanților și domeniul utilizatorului prin:
 - crearea unei concepții asupra domeniului activității care să corespundă cât mai bine domeniului conceptual al utilizatorului prin prezentarea obiectelor domeniului într-o formă potrivită, astfel încât utilizatorii să le înțeleagă ușor semnificația și caracteristicile și să stabilească natural legătura între acțiunile calculatorului (funcționalitatea) și acțiunile necesare îndeplinirii sarcinilor;
 - proiectarea de dialoguri care îi oferă utilizatorului libertate în cadrul domeniului activității, oferindu-i toate oportunitățile necesare de a realiza sarcinile fără a pierde controlul asupra sistemului.

Activitățile doi, trei și patru sunt în mod tradițional subiect de studiu pentru ingineria soft. Activitatea cinci este obiect de studiu pentru HCI. Activitatea unu este condusă de analiști, experți în interacțiunea om-calculator, experți din organizație. Atât HCI cât și ingineria soft au

dezvoltat metode proprii de culegere de informații relevante pentru sarcină. Ingineria soft folosește metode de *Analiza sistemului*, iar HCI folosește metode de *Analiza sarcinilor*. Anumite metode din analiza sistemului sunt confundate uneori cu analiza sarcinilor pentru că ambele analizează și modelează componente ale sarcinii. Analiza sarcinilor diferă de analiza sistemului prin scopuri, rezultate și subiecte de interes (vezi Tabela 4.1.1). Distincția dintre analiza sarcinilor și ingineria cerințelor în ingineria softului este aceea că ingineria cerințelor e preocupată de caracteristicile sistemului informatic. Atunci când interfața utilizator e privită ca o parte a sistemului interactiv (modul soft), analiza sarcinilor este o parte a ingineriei cerințelor. Dacă interfața utilizator e privită ca un instrument de efectuare a muncii, iar sistemul informatic este privit ca o implementare funcțională a instrumentului, atunci analiza sarcinilor este o activitate independentă [20].

Aplicarea metodelor de analiză a sarcinilor îi oferă analistului o imagine a implicării umane în sistem, ajutând la dezvoltarea unei imagini detaliate asupra sistemului din perspectiva utilizatorului asupra sarcinilor. Unele părți ale sistemului sunt modelate formal, altele mai puțin formal. Atât modelul sarcinilor cât și procesul de modelare în sine îi oferă analistului o înțelegere profundă a performanței umane. Modelul sarcinilor poate fi dezvoltat înaintea, simultan cu sau după proiectarea sistemului interactiv, în fiecare caz e folosit diferit, după cum urmează:

- analiza sarcinilor poate fi folosită ca intrare pentru toate etapele proiectării, modelul fiind construit pe baza unei activități existente; în acest caz analiza sarcinilor e folosită ca *sursă de informare*;
- modelele sarcinilor pot fi folosite în proiectare, permitând proiectanților să dezvolte scenariile de interacțiune în paralel cu dezvoltarea restului sistemului. Această posibilitate e utilă în dezvoltarea prototipurilor. Modelul trebuie modificat la fiecare versiune nouă a prototipului. În acest caz analiza sarcinilor poate fi

	Analiza de sistem	Analiza sarcinilor
Scop	Intrări pentru proiectarea structurilor de date și proceselor soft	Intrare pentru proiectarea interfețelor utilizator
Rezultat	Specificarea funcțională și specificarea arhitecturii sistemului	Specificări ale interfeței utilizator
Subiecte de interes	Informații tehnice, caracteristici ale datelor, limitări de procesare, considerații referitoare la arhitectura sistemelor	Limitări ale procesării informației ale ființelor umane, caracteristici ale utilizatorilor, considerații asupra sarcinilor
Principalele obiecte de analiză	Structuri de date (domeniul sarcinii computerizate) și funcționalitate	Concepțe despre sarcini și activitate

Tabela 4.1.1: Comparația dintre analiza sistemului și analiza sarcinilor

privită ca *un mijloc de a discuta impactul deciziilor de proiectare asupra activității umane*;

- analiza sarcinilor poate fi folosită pentru a verifica potrivirea dintre sistem și sarcinile utilizatorului. În acest caz analiza este realizată după proiectarea sistemului, constituind un *mijloc de evaluare a eficienței și operativității interfeței utilizator* și ca mijloc de efectuare a analizei erorilor [66].

Din interviurile cu proiectanții de soft în privința utilității analizei sarcinilor în procesul de proiectare a softului, a reieșit că analiza sarcinilor îi ajută să identifice:

- ce așteaptă utilizatorii de la sistem;

- structura și frecvența de folosire a facilităților sistemului;
- numele și forma de reprezentare a obiectelor prezente pe ecran și a evenimentelor care apar;
- informația care ar trebui să fie disponibilă într-un context anumit (écran);
- structura de navigare între contexte (mutarea între ecrane).

Proiectanții și-au exprimat dorința ca analiza sarcinilor să sprijine operațiile de mapare și verificare între sarcinile utilizator și proiectarea propusă. Un alt aspect menționat de proiectanți a fost cel al produselor analizei sarcinilor care ar trebui să poată fi mapate în aspecte ale proiectării precum funcționalitatea și dialogul [34].

4.1.2 Probleme relevante de analiza sarcinilor

Unul din reproșurile care au fost adresate analizei sarcinilor a fost acela că rămâne neclar ce se poate face cu datele culese. Răspunsul este acela că datele trebuie supuse unui proces de analiză care poate evidenția prezența următoarelor tipuri de probleme:

- Probleme în structura sarcinilor - structura sarcinilor nu este optimă pentru că necesită execuția prea multor sarcini sau anumite sarcini sunt mari consumatoare de timp sau au o frecvență de apariție prea mare;
- Diferențe între efectuarea formală și cea curentă a sarcinii - deși există documentații care specifică modul de realizare a sarcinilor, în realitate sarcinile nu se efectuează conform documentelor, ci se efectuează în diverse modalități. Problemele pot să apară atunci când într-un mediu cooperativ persoanele au păreri diferite despre ceea ce trebuie să se facă;

- Interacțiune ineficientă în organizație - sarcinile complexe necesită implicarea mai multor persoane care trebuie să interacționeze și să comunice pentru a împărtăși cunoștințe despre activitate sau datorită responsabilității pentru sarcini. Acestea sunt cauzele pentru care unele activități sunt mari consumatoare de timp sau pot deveni iritante pentru unele persoane implicate;
- Inconsistențe în sarcini - acțiunile sarcinilor sunt definite, dar nu sunt executate de nici un agent sau sunt executate în secvențe contradictorii;
- Oamenii fac lucruri interzise - în mediile complexe adeseori oamenii efectuează activități pentru care nu au primit aprobarea oficială sau folosesc/modifică obiecte pe care nu au voie să le manipuleze.

Unele din aceste probleme pot fi detectate automat sau semi-automat atunci când se folosește un instrument pentru analiza sarcinilor.

4.2 Metode de analiza sarcinilor și notații

4.2.1 HTA - Hierarchical Task Analysis

HTA este o metodă de analiză a sarcinilor care pornește de la descompunerea activității în sarcini. Rezultatele HTA constau în *ierarhii de scopuri, sarcini, acțiuni și planuri* care descriu în ce ordine și în ce condiții se execută sarcinile.

Reprezentarea ierarhiilor de sarcini poate lua formă textuală, caz în care indentarea este folosită pentru a sugera nivelurile în cadrul ierarhiei de sarcini, iar sarcinile sunt numerotate pentru a evidenția această ierarhie. Planurile sunt numerotate corespunzător sarcinilor (de exemplu, planul 0 exprimă modul în care se desfășoară sarcinile 1-5 ale activității 0). În cadrul planurilor observăm că nu toate acțiunile trebuie să se execute sau nu în ordinea prezentată. Găsirea ierarhiei potrivite,

restructurarea acesteia, fac parte din procesul HTA. Procesul de construire a ierarhiei de sarcini și a planurilor este unul iterativ. Se pornește de la o activitate și prima întrebare pe care o adresăm este una referitoare la sarcinile care trebuie îndeplinite pentru a atinge scopul activității, iar răspunsul la această întrebare îl găsim din surse precum: observarea directă, opinia experților, documentație, etc. La următorul pas se analizează fiecare sarcină și se încearcă descompunerea ei în mod similar. Un aspect problematic al acestei abordări este legat de determinarea punctului de oprire în descompunerea ierarhică. Un criteriu de oprire este când sarcina conține răspunsuri motorii complexe (de exemplu, mișcarea mouseului) sau când se ajunge la luarea unor decizii interne (unde activitatea e pur cognitivă) [21]. Pentru descrierea planurilor analistul poate alege folosirea unei metode formale, o diagramă sau o descriere textuală. Un exemplu de descompunere a activității folosind HTA este cel al curățeniei unui apartament, ilustrată diagramatic în Figura 4.1. Descompunerea textuală este prezentată în cele ce urmează:

0. pentru a face curățenie:

1. ia aspiratorul de praf
2. atașează-i componente
3. curăță încăperea
 - (a) curăță holul
 - (b) curăță dormitoarele
 - (c) curăță sufrageria
4. golește sacul de praf
5. așează aspiratorul.

Planul 0: execută 1,2,3,5 în această ordine; când sacul este plin execută 4.

Planul 3: execută oricare din 3.(a), 3.(b), 3.(c) în orice ordine în funcție de încăperea care necesită curățarea.

Deoarece reprezentările grafice sunt mai ușor lizibile, o metodă alternativă de descriere a ierarhiilor rezultate din analiza HTA este cea folosind arbori de sarcini. Descrierea grafică a exemplului prezentat anterior este ilustrată în Figura 4.1:

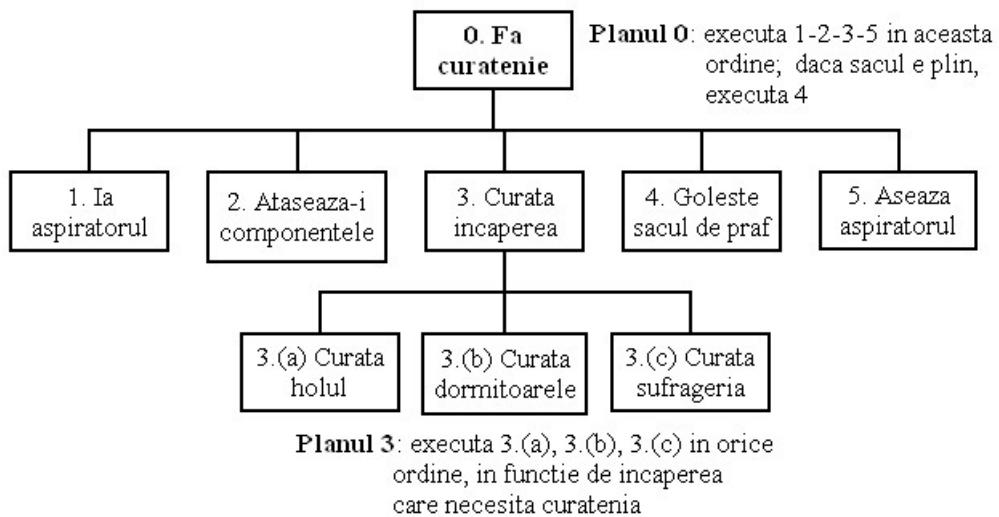


Figura 4.1: Diagrama HTA

4.2.2 GTA - Groupware Task Analysis

GTA este o metodă de analiză a sarcinilor recent dezvoltată la Universitatea Liberă din Amsterdam care combină aspecte preluate și prelucrate de la alte câteva metode de analiză a sarcinilor. GTA se dorește a fi o abordare nouă pentru fazele inițiale din proiectarea sistemelor complexe, în care diferiți utilizatori cu competențe și roluri diferite și persoane a căror activitate este afectată de introducerea tehnologiei folosesc soluții informaticice pentru a efectua sarcini complexe. Termenul “grou-

pware” subliniază faptul că sistemele interactive complexe și procesele de muncă nu pot fi analizate cu folos din punctul de vedere al unui singur post de muncă sau al unei interacțiuni unice utilizator-calculator [77]. GTA oferă un cadru conceptual care specifică aspectele relevante din universul activității care trebuie luate în considerare la proiectarea de sisteme pentru grupuri de utilizatori.

GTA este parte a metodei de proiectare DUTCH [81], în cadrul căreia corespunde primei faze din procesul de proiectare a sistemelor interactive complexe.

Rezultatul analizei GTA este un model formal al sarcinilor care specifică cadrul intenționat pentru sarcini, iar această specificare se dorește a fi intrarea pentru specificarea detaliilor tehnologice (proiectarea propriu-zisă a sistemului) [75].

Analiza sarcinilor GTA urmează trei pași:

- analiza situației curente de muncă și modelarea ei, rezultatul fiind *Modelul 1 al sarcinilor - MS1*;
- imaginarea unei noi situații pentru sarcinile de muncă pentru care se proiectează sistemul, rezultatul fiind *Modelul 2 al sarcinilor - MS2*;
- specificarea semantică tehnologiei informației care este proiectată (modelul 3 al sarcinilor) [77].

Analiza situației curente - Modelul 1 al sarcinilor

În majoritatea cazurilor proiectarea sistemelor este declanșată de o situație existentă a activității. Această situație fie nu este optimă, fie se asteaptă ca introducerea noii tehnologii să îmbunătățească această situație. Analiza stării curente a sarcinilor dă posibilitatea proiectantului să formuleze cerințele de proiectare și permite evaluarea ulterioară a proiectării. Modelarea structurată a sarcinilor ajută proiectantul să rea-

lizeze care sunt lacunele sale în ceea ce privește cunoașterea și înțelegerea acestora. Modelul 1 al sarcinilor servește ca și bază pentru modelul 2 al sarcinilor.

Specificarea situației viitoare a activității - Modelul 2 al sarcinilor

Cel de-al doilea model al sarcinilor este reproiectat pe baza modelului 1 al sarcinilor astfel încât să fie posibilă includerea de soluții tehnice și de răspunsuri tehnice la cerințele utilizatorilor. Deciziile de proiectare care conduc de la modelul 1 al sarcinilor la modelul 2 al sarcinilor sunt bazate pe trei surse:

- Problemele identificate în modelul 1 al sarcinilor - analiza sarcinilor va evidenția părți ale structurii sarcinilor și caracteristici ale obiectelor asociate sarcinilor care trebuie modificate pentru a optimiza îndeplinirea obiectivului.
- Cerințele clientului - clienții sunt persoanele care plătesc pentru îmbunătățirea situației curente de realizarea a sarcinilor, iar cerințele lor includ aspecte economice, constrângeri de timp, norme calitative referitoare la munca efectuată.
- Constrângeri și opțiuni tehnice - pe baza cunoștințelor tehnologice proiectanții identifică posibilele soluții pentru problemă, însă soluțiile tehnologice pot să nu fie fezabile.

Specificarea tehnologiei - Modelul 3 al sarcinilor

Cel de-al treilea model specifică oferta sistemului în privința delegării sarcinilor. Dacă MS 2 prezintă noua structură a sarcinilor global, modelul 3 specifică soluția detaliată în termeni ai tehnologiei. Pentru înțelegerea acestui model se introduce noțiunea de *mașină virtuală a utilizatorului* (User's Virtual Machine - UVM), care cuprinde toate aspectele

sistemului relevant pentru utilizator. Astfel, proiectarea mașinii virtuale a utilizatorului înseamnă, conform modelului lui Seeheim [25], proiectarea funcționalității, dialogului și prezentării. UVM e specifică unui rol și sarcinilor asociate, iar pentru că sistemele sunt proiectate pentru mai multe roluri e necesară proiectarea mai multor UVM-uri care sunt mai apoi integrate, prin urmare orice proiectare este un compromis.

Proiectarea funcționalității este diferită de proiectarea arhitecturii aplicației, referindu-se la funcționalitatea aplicației în măsura în care este relevantă pentru utilizator și care îi va fi prezentată prin intermediul interfeței utilizator. Pentru a realiza activități utilizatorii efectuează acțiuni asupra obiectelor, astfel că la proiectarea funcționalității proiectanții trebuie să aleagă obiectele care vor face parte din sistem, structura și relațiile dintre ele și trebuie să descrie acțiunile care pot fi efectuate asupra lor. Sarcinile sprijinite de sistem vor oferi baza pentru fixarea căilor de navigare la un nivel înalt și care vor fi transpusă în structura interfeței utilizator. Funcționalitatea sistemului trebuie să fie bine aleasă astfel încât să sprijine realizarea sarcinilor din modelul sarcinilor. Funcționalitatea va influența arhitectura softului deoarece funcționalități specifice vor impune constrângerii asupra implementării (de exemplu, nivele multiple de Undo/Redo impun folosirea unei cozi de comenzi).

Proiectarea dialogului - dialogul se referă la structura și comportamentul dinamic al interfeței utilizator, fără a lua în considerare prezentarea. La nivelul dialogului nu este important modul în care arată o componentă a interfeței utilizator, ci dacă e modală sau nu, ce conținut are și cum e structurat. La acest nivel trebuie definite componentele majore din structura interfeței și dinamica interfeței utilizator (de exemplu, deschiderea unei ferestre poate determina închiderea altei ferestre). În dialogul om-calculator un rol important îl are comportamentul utilizatorului care cuprinde acțiuni fizice și acțiuni mentale care ocupă un timp considerabil în execuția sarcinilor. Pentru a îmbunătăți performanțele

utilizatorului trebuie să se ofere o structură navigațională bună însotită de prezentarea elementelor relevante pentru execuția sarcinilor.

Proiectarea prezentării - se referă la aspectele vizibile ale interfeței utilizator, dar și la feedbackul auditiv și cel tactil (în cazul utilizării force-feedback-ului). Folosirea culorilor poate duce la prevenirea erorilor și la memorarea semantică obiectelor interfeței. Gruparea logică a elementelor are un efect pozitiv în găsirea rapidă a informației.

Între MS2 și MS3 trebuie să se păstreze o corespondență. O dată ce UVM este implementată într-un prototip este necesară o revizuire a ambelor modele în scopul evaluării.

Proiectarea prin rafinare a interfeței utilizator aduce avantaje în raport cu proiectarea directă a acesteia prin îmbunătățirea comunicării între echipele de proiectare, deoarece problemele și soluțiile propuse sunt modelate explicit, iar modelarea permite revenirea și evaluarea deciziilor de proiectare în raport cu cerințele.

Nucleul GTA este un cadru conceptual care servește două scopuri: de a oferi recomandări pentru colectarea de informații despre starea curentă a sarcinilor și de a oferi o bază pentru modelarea vechii și noi strucuri a sarcinilor. Analiza GTA are la bază trei aspecte diferite: *agenții*, *munca* și *situația* [77, 75], fiecare din acestea descrie universul sarcinilor dintr-un punct de vedere diferit, dar se află într-o strânsă relație cu celelalte. Această abordare din unghiuri variate permite luarea de decizii de proiectare mai potrivite sarcinilor și permite instrumentelor de proiectare să verifice și să păstreze *coherența* și *completitudinea* modelelor.

§. Agenții

Agenții se referă la oameni (individual sau grupuri), dar și la sisteme. Oamenii sunt descriși prin caracteristicile lor relevante pentru activitate: limba pe care o vorbesc, abilitățile de tastare sau operarea pe diverse sisteme de operare. Agenții sunt grupați în funcție de submulțimile

de sarcini alocate în *roluri*. Un rol poate fi deținut de mai mulți agenți, la fel cum un agent poate să dețină mai multe roluri. *Organizația* este definită de relațiile dintre agenți și roluri în raport cu alocarea sarcinilor. Analiza și reprezentarea unei organizații trebuie să includă informații despre responsabilitatea sarcinilor, delegarea sarcinilor și atribuirea rolurilor, precum și despre competențe și accesul la obiecte.

§. Munca

În studiul muncii conceptul de bază este *sarcina*, care poate fi identificată la diverse niveluri de complexitate. Munca poate fi structurată în una sau mai multe structuri de sarcini, unde structura sarcinilor trebuie să fie descrisă folosind constructori pentru relațiile temporale (secvențierea, declanșarea subsarcinilor, cicluri, alegeri). La nivelul superior al unei structuri a sarcinii se găsește *scopul sarcinii* (business goal) și *sarcinile* legate de acesta. O sarcină nu este delegată unei singure persoane sau unui singur rol, iar pentru scopurile de nivel înalt există câteva sarcini care să ducă la îndeplinirea scopului. La nivelul inferior al structurii sarcinii se găsesc *acțiunile*.

Există două tipuri de sarcini care merită o atenție deosebită: *sarcinile unitate* - adică acele sarcini pe care le descriu oamenii când vorbesc de cel mai jos nivel al muncii lor și *sarcinile de bază* care reprezintă nivelul atomic de delegare a sarcinilor definit de instrumentul folosit în realizarea sarcinii (de exemplu, comanda). Sarcinile complexe pot fi împărțite între actori sau roluri. Sarcinile unitate și sarcinile de bază pot fi descompuse mai departe în *acțiuni sistem* și *acțiuni utilizator*.

Structura sarcinilor

Structura sarcinilor este în cele mai multe cazuri ierarhică, iar pentru indicarea ordinii temporale și a relațiilor de dependență dintre sarcini este nevoie de folosirea unor “constructori”. Structura sarcinii nu

este cunoscută adeseori de actorii individuali, mai ales când diferite roluri sunt implicate în realizarea unor sarcini.

Acțiunile

Acțiunile sunt componente identificabile ale sarcinilor de bază sau ale sarcinilor unitate care au un rol în îndeplinirea unei unități de muncă, dar care își derivă sensul din sarcina a căror parte sunt (de exemplu apăsarea tastei ENTER are o semnificație diferită când urmează unei comenzi sau când confirmă o valoare numerică). În descrierea acțiunilor scopul este de a le identifica semnificația, nu caracteristicile fizice.

Protocolele și strategiile

O mulțime de sarcini care aparțin unui singur rol au un mod de efectuare bine definit. Dacă acest procedeu este considerat a fi o practică comună bună, îl denumim *protocol*. Dacă procedura este specifică experților în domeniu, o numim *strategie*.

§. Situația

Analiza universului sarcinii din punctul de vedere al situației înseamnă găsirea și descrierea mediului înconjurător (fizic, conceptual și social) și a obiectelor din mediu.

Obiectele

Orice lucru care este relevant pentru muncă într-o anumită situație este un *obiect* în cadrul analizei sarcinilor. Obiectele pot fi fizice sau conceptuale (mesaje, gesturi). Referirea la obiecte se face prin reprezentări externe cu caractere diferite: etichete, desene, metafore. Actorii care joacă un anumit rol pot fi obiecte într-o situație a sarcinii diferită

și vor fi etichetați ca “obiecte active”. Obiectele sunt folosite pentru a transfera informații între agenți.

Structura obiectelor

Pentru a descrie semnificația obiectelor trebuie identificate două tipuri de relații între obiecte:

- relații de specializare de tipul subtip/supertip. Subtipurile moștenesc caracteristicile supratipurilor în absența altor specificări. Pe baza acestor relații obiectele vor fi cuprinse într-o ierarhie de obiecte.
- relații de agregare - când un obiect poate fi “în” alt obiect sau “conține” alt obiect sau obiectele pot să se “mute” dintr-o locație în alta.

Pe lângă aceste relații, obiectele sunt legate de sarcini sau agenți.

Obiectele sunt descrise prin structura și atributele lor, dar nu e vorba de sensul oferit termenului în paradigma orientată pe obiecte.

Mediul

Mediul sarcinilor este contextul curent pentru îndeplinirea obiectivului și include actori, roluri, condiții pentru îndeplinirea sarcinilor, pentru strategii și protocoluri, obiecte relevante, artefacturi precum sisteme cărora le sunt delegate sarcini. De asemenea, structura temporală a evenimentelor este parte a mediului. Mediul influențează structura activității.

§. Ontologie pentru universul sarcinilor

Van Welie [78] propune o ontologie a universului sarcinilor care evidențiază aspectele esențiale în analiza sarcinilor și relațiile dintre ele.

Ontologia definește concepțele și relațiile dintre ele care sunt considerate relevante pentru scopul analizei sarcinilor. Ontologia constituie baza conceptuală pentru informațiile care trebuie reținute și modul de structurare și reprezentare a acesteia.

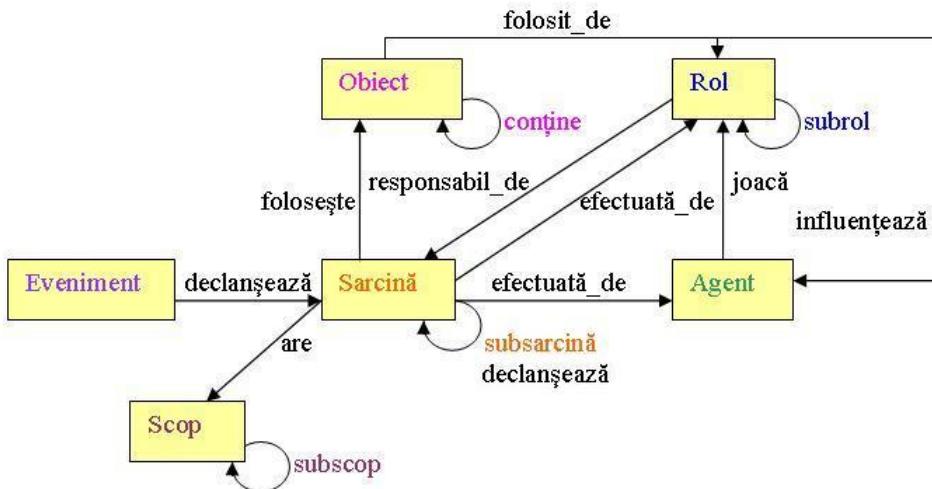


Figura 4.2: Ontologia universului activității [78]

În continuare vom detalia concepțele din ontologie și relațiile care pot să existe între ele:

- **Sarcină** - pași efectuați de agenți pentru a atinge un anumit scop. Sarcinile sunt executate într-o anumită ordine și finalizarea unei sarcini poate declanșa execuția uneia sau mai multor sarcini. O sarcină poate să fie declanșată și de apariția unui eveniment.
- **Scop** - este o stare care se dorește a fi atinsă și care poate fi atinsă prin execuția uneia sau mai multor sarcini;
- **Rol** - colecție de sarcini efectuate de unul sau mai mulți agenți. Rolul are sens atunci când are un scop clar și stabilește o distincție

între grupuri de agenți. Rolul e responsabil de sarcinile pe care le cuprinde. Rolerile pot fi descompuse ierarhic;

- **Obiect** - entitate fizică sau conceptuală, poate fi inclus într-o ierarhie de tipuri și poate fi inclus în alte obiecte;
- **Agent** - entitate activă (oameni, grupuri de oameni sau sisteme). Agenții nu sunt indivizi specifici, ci indică clase de indivizi cu anumite caracteristici;
- **Eveniment** - schimbare în universul activității la un moment dat în timp asupra căreia actorul nu are întotdeauna controlul direct (căderea electricității, îmbolnăvirea unui coleg, etc.). Evenimentele influențează ordinea execuției sarcinilor prin declanșarea de sarcini.

Relațiile care pot apărea între componentele ontologiei sunt:

- **folosește** - specifică obiectele folosite în execuția sarcinii și modul în care sunt folosite;
- **declanșează** - este relația de bază în specificarea fluxului sarcinilor, specificând dacă o sarcină e declanșată de un eveniment sau o altă sarcină;
- **joacă** - fiecare agent trebuie să joace un rol. Această relație indică rolurile asociate agenților;
- **realizată de** - specifică faptul că o sarcină este efectuată de un agent, lucru care nu înseamnă că agentul este responsabil pentru sarcină, responsabilitatea depinzând de rolul din care face parte agentul. Când nu este important care agent efectuează o sarcină se poate menționa rolul care e responsabil pentru sarcină;
- **are** - relație care conectează sarcinile la scopuri; fiecare sarcină are un scop care definește motivul execuției sale;

- **subsarcină/subscop** - descrie descompunerea sarcinilor/scopurilor;
- **subrol** - determină o ierarhie a rolurilor;
- **influențează** - un rol poate influența un alt rol (este o parte a culturii organizaționale);
- **responsabil** - specifică sarcinile pentru care un rol este responsabil;
- **folosit de** - specifică cine poate folosi un obiect și modul în care un rol sau un agent poate folosi obiectul.

Acste relații formează o bază, iar alte relații pot fi derivate pe baza acestor relații (de exemplu, cine este implicat într-o sarcină = responsabil + realizează + joacă + subrol).

§. Verificarea modelelor

Definirea obiectelor și relațiilor dintre ele permite verificarea modelelor construite. Proprietățile care pot fi verificate sunt general valabile și se referă la constrângeri pe care am dori ca modelele să le satisfacă. Aceste constrângeri se pot referi la cardinalitate, tip și atribute ale obiectelor.

Constrângeri de cardinalitate - se referă la cardinalitatea relațiilor dintre concepte. Acestea sunt:

- Fiecare eveniment trebuie să declanșeze cel puțin o sarcină:
 $\forall e \exists t \{e \in Events, t \in Tasks | triggers(e, t)\}$;
- Fiecare agent trebuie să aibă cel puțin un rol:
 $\forall a \exists r \{a \in Agents, r \in Roles | hasrole(a, r)\}$;
- Fiecare rol trebuie să fie responsabil pentru cel puțin o sarcină:
 $\forall r \exists t \{r \in Roles, t \in Tasks | responsible(r, t)\}$;

- Fiecare obiect trebuie să fie folosit în cel puțin o sarcină:

$$\forall o \exists t \{o \in Objects, t \in Tasks | uses(t, o)\};$$

- Fiecare sarcină trebuie să fie îndeplinită de cel puțin un rol:

$$\forall t \exists r \{t \in Tasks, r \in Roles | performs(t, r)\};$$

- Pentru fiecare sarcină trebuie să existe cel puțin un rol responsabil:

$$\forall t \exists r \{t \in Tasks, r \in Roles | responsible(r, t)\};$$

- Fiecare obiect trebuie să aibă un proprietar:

$$\forall o \exists r \{o \in Objects, r \in Roles | uses(o, r, Rights) \vee owner \in Rights\}.$$

Constrângeri de tip - se referă la entități de același tip:

- O instanță a unui obiect nu poate să se conțină pe sine:

$$\neg \exists o \{o \in Objects | contains(o, o)\};$$

- O sarcină nu poate să se aibă pe sine ca subsarcină:

$$\neg \exists t \{t \in Tasks | subtask(t, t)\};$$

- O sarcină nu poate să se declanșeze pe sine:

$$\neg \exists t \{t \in Tasks | triggers(t, t)\};$$

- Un rol nu poate să se aibă pe sine ca subrol:

$$\neg \exists r \{r \in Roles | subrole(r, r)\}.$$

§. Reprezentări pentru conceptele GTA

GTA este un cadru conceptual doar și nu impune folosirea unor reprezentări, dar din experiența aplicării metodei următoarele tipuri de reprezentări corespund cerințelor analizei sarcinilor.

Reprezentarea structurii muncii

Scopul modelării structurii muncii este de a reprezenta modul în care oamenii își împart activitatea în părți mai mici pentru a atinge scopuri. Cunoașterea structurii muncii permite proiectanților să înțeleagă modul în care oamenii gândesc asupra muncii lor și modul în care sarcinile sunt legate de scopuri. Relația dintre sarcini și scopuri permite mai departe alegerea sarcinilor care trebuie să fie sprijinite de sistemul informatic și care sunt scopurile independente de tehnologia folosită. Structura muncii este modelată în mod ușual prin *arbori de sarcini* care descriu o descompunere ierarhică a muncii (vezi Figura 4.3). În construcția arborilor este esențială relația de *subsarcină* dintre sarcini. Alături de sarcini, în ierarhii sunt incluse și scopurile. La cel mai înalt nivel un arbore poate să înceapă cu un scop și subscopurile sale și să continue cu sarcini, subsarcini și acțiuni. În astfel de cazuri sunt folosite relațiile *are* și *subscop*. O descompunere a sarcinii este modelată din punctul de vedere al unui rol sau al unui scop, prin urmare se vor descrie câțiva arbori pentru a surprinde munca din punctul de vedere al tuturor rolurilor. Deseori sunt necesare și informații temporale, în acest caz folosindu-se construcțori precum: SEQ (ordine secvențială), LOOP (cicluri), PAR (paralelism) sau OR (selecție) [76]. Există situații în care anumite sarcini apar *uneori* sau aproape *niciodată*. Dacă se impune precizie în specificarea unor astfel de situații, alegerea cea mai potrivită este de folosire a diagramelor de flux [80]. Detaliile despre sarcină sunt completate în şablonane care includ informații despre schimbarea stării, frecvență și durată. Figura 4.3 prezintă un exemplu de arbore al sarcinilor în cazul multiplicării unor documente.

Structura rolurilor și relația cu sarcinile sunt alte aspecte importante ale structurii muncii. Structura rolurilor poate fi descrisă folosind de asemenea arbori de roluri.

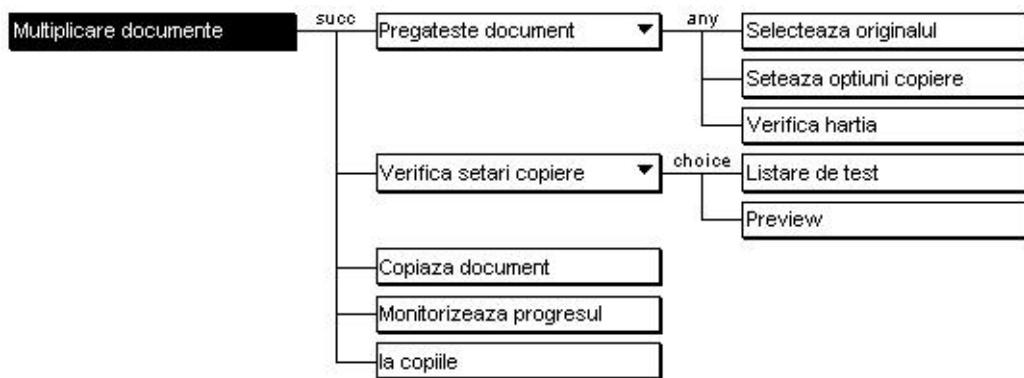


Figura 4.3: Reprezentarea GTA pentru structura muncii

Reprezentarea dinamicii muncii

Dinamica muncii se referă la ordinea în care sunt efectuate sarcinile relativ la rolurile care le efectuează. Pentru descrierea acestor aspecte sunt folosite diagramele de flux a activităților (workflow) și modelele sarcinilor. Un aspect important al dinamicii muncii se referă la colaborare și comunicare, mai ales când sunt implicate mai multe roluri în realizarea unei sarcini. Un model al fluxului muncii poate descrie munca în raport cu rolurile și timpul. Acest model îi oferă proiectantului informații despre ordinea în care sunt efectuate sarcinile și modul de implicare a oamenilor în efectuarea acestora, precum și informații despre modul în care oamenii colaborează și comunică prin schimbul de obiecte și mesaje. Pentru modelarea acestor aspecte se folosește o variație a diagramelor de activitate UML, în care au fost incluse evenimente și scopuri pentru a o face mai potrivită pentru analiza sarcinilor. Fiecare flux descrie un scenariu care este declanșat de un eveniment, reprezentat de un oval conectat la prima sarcină. Sarcinile pot fi grupate în culoare, cîte un culoar pentru fiecare rol. Obiectele transmise între sarcini aparținând diferitelor roluri sunt desenate pe linia despărțitoare a culoarelor (vezi Figura 4.4). Scopurile pot fi reprezentate în prima coloană a fluxului muncii și sunt însotite de

linii verticale care indică durata activării scopului.

Modelul de flux nu dă informații despre ierarhia sarcinilor, ci folosește doar sarcini care se află pe același nivel al ierarhiei. Pentru subsarcini se construiesc noi modele de flux.

În termeni ai ontologiei, modelul de flux e bazat pe conceptele *eveniment*, *sarcină*, *obiect* și *rol*. Relațiile folosite sunt *declanșează*, *responsabil* și *folosește*. Operatorii folosiți sunt **Concurrent**, **Choice**, **Successive**, care sunt parametri ai relației *declanșează*. În cazul obiectelor transmise între roluri, acestea trebuie să fie asociate ambelor sarcini prin relația *folosește*.

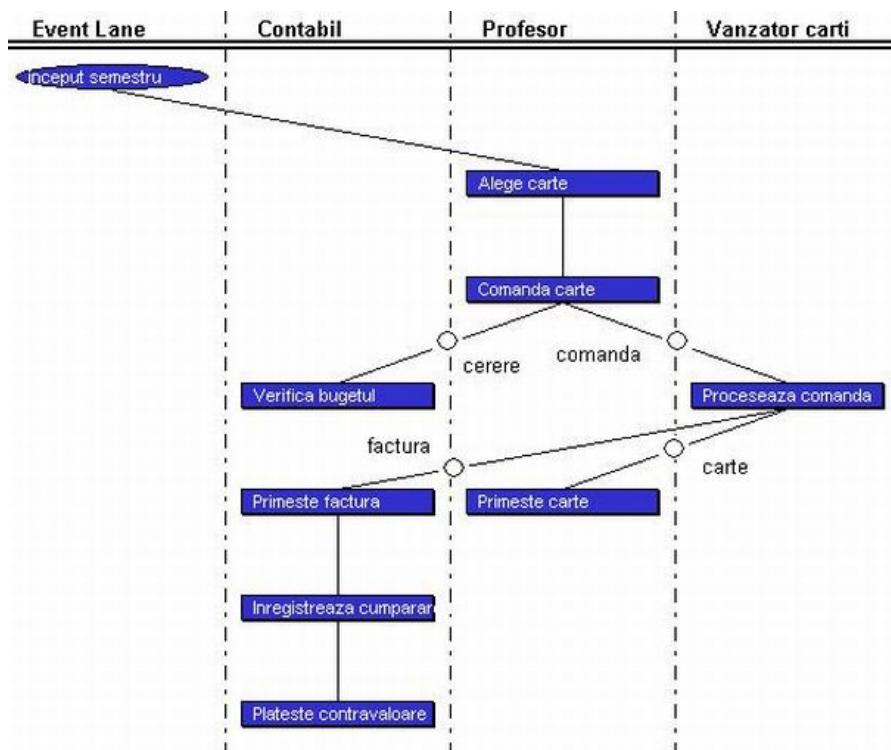


Figura 4.4: Reprezentarea GTA pentru fluxul activității

Reprezentarea artefacturilor

Mediul muncii conține numeroase obiecte, unele din ele fiind folosite direct în realizarea sarcinilor. Modelul artefacturilor surprinde două relații între obiecte: *conține* și *tip*. Pentru acestea e relevant să se descrie structura, tipul, și atributele specifice. Pentru a descrie aceste aspecte se folosesc diagrame de clase, dar fără secțiunea de metode.

Obiectele pot să fie legate de utilizatori (roluri sau agenți), în loc să fie legate de sarcinile în care sunt folosite. Pentru astfel de cazuri se folosesc diagrame în care utilizatorii sunt reprezentați ca și ovaluri, iar obiectele sunt reprezentate prin puncte etichetate poziționate în interiorul ovalurilor. Ovalurile se pot suprapune dacă mai mulți utilizatori folosesc un obiect (vezi Figura 4.5).

Relațiile cu sarcinile și utilizatorii sunt descrise în şabloane specifice.

Reprezentarea mediului muncii

Un aspect al mediului muncii este disponerea fizică a obiectelor și dimensiunile acestora. În descrierea acestor aspecte se folosesc hărți, fotografii, fragmente video la care se adaugă comentarii relative la impactul lor asupra muncii (de exemplu, ușurința de a ajunge la un obiect).

Un alt aspect al mediului muncii este cultura organizațională. În acest model rolurile sunt reprezentate ca și ovaluri, iar acestea sunt conectate prin săgeți dacă există o *influență* între roluri. Intensitatea influenței dintre roluri este ilustrată prin grosimea săgeților. Influențele sunt etichetate cu *comportamentul* determinat de relația de influență (vezi Figura 4.6).

Reprezentări folosite în proiectarea detaliată

În etapa de proiectare detaliată se are în vedere specificarea funcționalității, dialogului și prezentării.

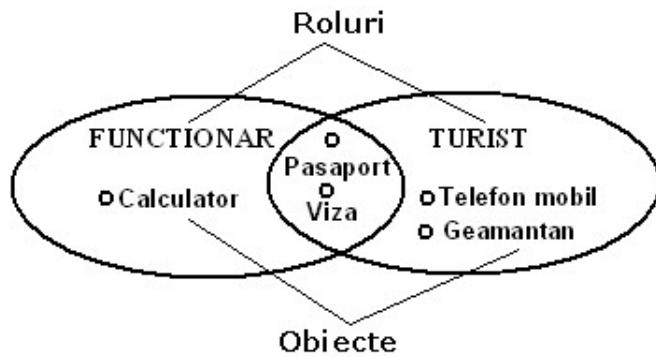


Figura 4.5: Reprezentarea GTA pentru artefacturi și roluri

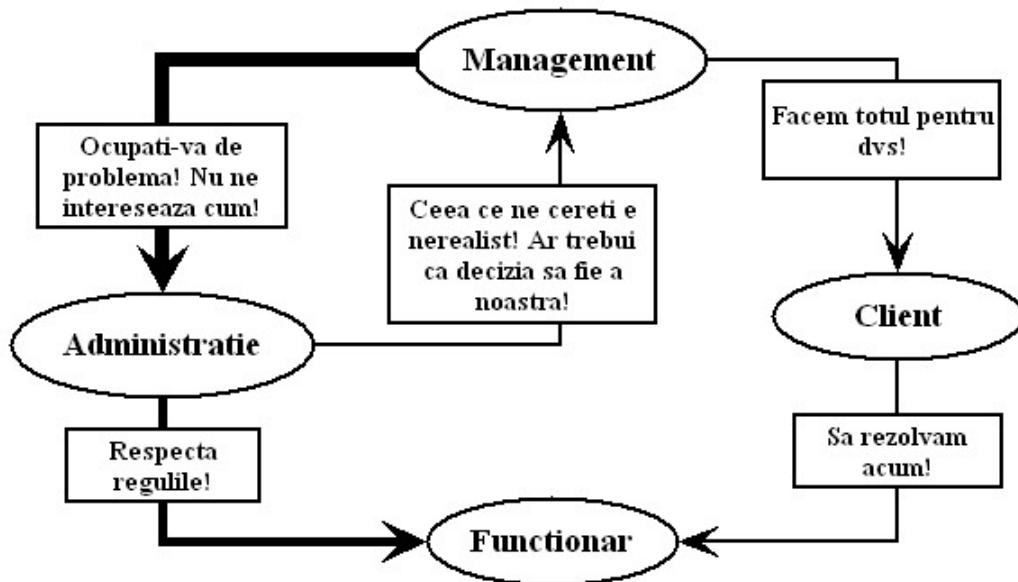


Figura 4.6: Reprezentarea GTA pentru cultura organizațională

Pentru proiectarea dialogului se folosește notația NUAN (New User Action Notation). Diagramele NUAN descriu dialogul dintre utilizator și sistem. Deoarece folosirea singulară a diagramelor UAN nu a dat rezultate încurajatoare, în paralel sunt descrise și schițe ale ecranelor. În descrierea NUAN a interacțiunii sarcinile de bază ale modelului de sarcină devin interacțiunile de nivel înalt din NUAN [76].

4.2.3 ConcurTaskTrees - notație diagramatică pentru specificarea modelelor sarcinilor

ConcurTaskTrees este o notație diagramatică pentru specificarea modelelor sarcinilor dezvoltată din dorința de a oferi o reprezentare grafică a structurii ierarhice rezultate din analiza sarcinilor [50]. Neajunsul major al metodelor de analiză a sarcinilor dezvoltate anterior era acela că nu ofereau posibilitatea exprimării formale a relațiilor temporale dintre sarcini. Mai mult decât atât, metode de analiză a sarcinilor precum GOMS sau TAG nu ofereau posibilitatea descrierii de sarcini care se desfășoară paralel sau concurrent, ci doar descrierea de sarcini care se desfășoară secvențial. ConcurTaskTrees introduce în notația propusă operatorii temporali LOTOS [28], care dispun de o semantică definită formal și care este o notație internațională standardizată.

În ConcurTaskTrees o sarcină este descrisă de următoarele attribute:

- nume - folosit pentru identificarea sarcinii;
- tip - există patru tipuri de sarcini:
 1. utilizator - efectuate în întregime de utilizator care implică activități cognitive sau fizice care nu implică interacțiunea cu sistemul (de exemplu, utilizatorul citește o listă a curselor aeriene și alege una care satisface nevoile sale). Sarcinile utili-

- zator sunt asociate cu o activitate de procesare a informației primite de la mediu;
2. aplicație - sarcini efectuate complet de către sistem, care primesc informație de la sistem și o oferă utilizatorului. Aceste sarcini sunt inițiate de aplicație (de exemplu, compilarea unui program și transmiterea de mesaje de eroare, primirea unor mesaje din rețea și afișarea lor);
 3. interacțiune - executate de utilizator în interacțiune cu sistemul, interacțiune activată de utilizator (de exemplu, editarea unei diagrame, formularea unei interogări pentru o bază de date);
 4. abstrakte - sarcini care presupun acțiuni complexe a căror efectuare completă nu intră doar în una din categoriile de tipuri de activități prezentate anterior.

- subsarcină a - nume al sarcinii părinte;
- obiecte - vector de obiecte, fiecare obiect definit prin: nume, tip, lista acțiunilor asupra obiectului. Tipul obiectelor poate fi:
 - intern - se referă la o entitate care aparține aplicației și care trebuie mapată pentru a fi percepță de utilizator (de exemplu, starea unei interogări a unei baze de date, înregistrările bazei de date);
 - perceptibil - utilizatorul poate interacționa cu obiectul prin intermediul simțurilor (de exemplu: meniuri, ferestre, pictogramme, etc.);
- iterativ - valoare booleană care specifică dacă sarcina e iterativă;
- prima acțiune - mulțimea acțiunilor inițiale posibile;



Figura 4.7: Sarcină abstractă



Figura 4.8: Sarcină cooperativă



Figura 4.9: Sarcină utilizator



Figura 4.10: Sarcină de interacțiune



Figura 4.11: Sarcină aplicație

- ultima acțiune - multimea acțiunilor finale posibile.

Obiectele - sunt entități manipulate pentru îndeplinirea sarcinilor prin acțiunile asociate. Între obiectele care apar pe niveluri diferite ale ierarhiei de sarcini pot apărea două tipuri de relații:

- descompunere - un obiect de la un nivel al sarcinii poate fi descompus în mai multe obiecte la nivelul următor;
- rafinare - numărul acțiunilor asociate unui obiect crește odată cu trecerea la un nou nivel de detaliere a sarcinilor.

Reprezentarea grafică a acestor tipuri de sarcini folosită în instrumentul CTTE (ConcurTaskTreesEnvironment) este prezentată în Figurile 4.7, 4.8, 4.9, 4.10 și 4.11.

Acțiunile - sunt asociate cu obiectele și se clasifică în trei categorii: cognitive, logice și fizice. Pentru sarcinile care nu sunt iterative este importantă specificarea ultimei acțiuni. Este la fel de important să menționăm că sunt acțiunile inițiale deoarece este nevoie de cunoașterea acestora la evaluarea unor expresii temporale (de exemplu, în prezența unui operator de alegere [] sau a operatorului de dezactivare (inhibare) [>].

Operatorii temporali folosiți în notația ConcurTaskTrees sunt:

1. T1||| T2 - întretăierea - acțiunile celor două sarcini se pot desfășura în orice ordine fără constrângeri, spre exemplu monitorizarea unui

écran și vorbitul la un microfon;

2. $T_1 \sqcap T_2$ - alegerea - se poate alege una din cele două sarcini după care începe execuția ei, cealaltă devenind indisponibilă până la terminare sarcinii care a început să se execute, spre exemplu la începutul unei sesiuni de lucru cu un editor de texte este posibil să se aleagă între deschiderea unui document sau crearea unui document nou;
3. $T_1 \parallel T_2$ - concurență cu schimb de informație - sarcinile se execută concurență, dar trebuie să se sincronizeze pentru schimb de informație, spre exemplu într-un editor de texte unde editarea unui fișier și derularea conținutului acestuia se pot realiza în orice ordine, dar trebuie să schimbe informație pentru că editarea se poate realiza doar după ce porțiunea vizată este vizibilă;
4. $T_1 \mid = \mid T_2$ - independentă - ambele sarcini trebuie să se execute, dar odată ce execuția unei sarcini a început, cealaltă se poate executa doar la terminarea primei;
5. $T_1 \gg T_2$ - activarea - la terminarea sarcinii T_1 este activată sarcina T_2 , spre exemplu o aplicație în care utilizatorii trebuie să se autentifice și apoi pot accesa informațiile;
6. $T_1 \sqcap \gg T_2$ - activarea cu schimb de informație - la terminarea sarcinii T_1 este activată T_2 și în plus îi sunt transmise anumite valori lui T_2 , spre exemplu T_1 permite specificarea unor criterii de filtrare, iar T_2 execută interogarea conform criteriilor;
7. $T_1 [> T_2$ - dezactivarea - la efectuarea primei acțiuni din T_2 , T_1 este dezactivată definitiv, spre exemplu aplicațiile în care utilizatorii pot să opteze pentru dezactivarea unui set de sarcini pentru activarea unui alt set de sarcini prin apăsarea unui buton;

8. $T_1 | > T_2$ - suspendare/reluare - T_2 poate să întrerupă execuția lui T_1 , iar la terminarea lui T_2 , T_1 poate să fie reactivată din starea în care se află înainte de întrerupere, spre exemplu editarea unui text poate fi întreruptă de o fereastră modală referitoare la tipărire documentului, iar după finalizarea tipăririi editarea poate fi reluată;
9. T_1^* - iterația - sarcina este iterativă, ceea ce înseamnă că la terminarea sarcinii acțiunile sale se reiau automat, până la execuția unei sarcini de dezactivare;
10. $T_1(n)$ - iterație finită - T_1 se execută de n ori;
11. $[T_1]$ - sarcină optională - execuția lui T_1 nu e obligatorie, spre exemplu completarea unei forme în care unele câmpuri sunt optionale;
12. T - recursivitatea - posibilitatea incluzării în specificarea sarcinii pe ea însăși. Acest lucru este util atunci când la fiecare execuție recursivă se dorește furnizarea posibilității de a adăuga noi sarcini spre a fi executate, până la execuția unei sarcini care să întrerupă recursivitatea [44].

Un exemplu de model al sarcinilor CTT, realizat folosind instrumentul CTTE, pentru operarea cu un bancomat este prezentat în Figura 4.12.

4.3 Instrumente pentru analiza sarcinilor

Deși analiza sarcinilor și-a demonstrat rolul benefic în proiectarea de sisteme utilizabile, numărul instrumentelor de analiza sarcinilor este redus, cu toate că manipularea unor cantități mari de informații necesare efectuării analizei sarcinilor este dificilă. În cele ce urmează vom prezenta două instrumente pentru analiza sarcinilor asociate unor metode de analiza sarcinilor anterior prezentate.

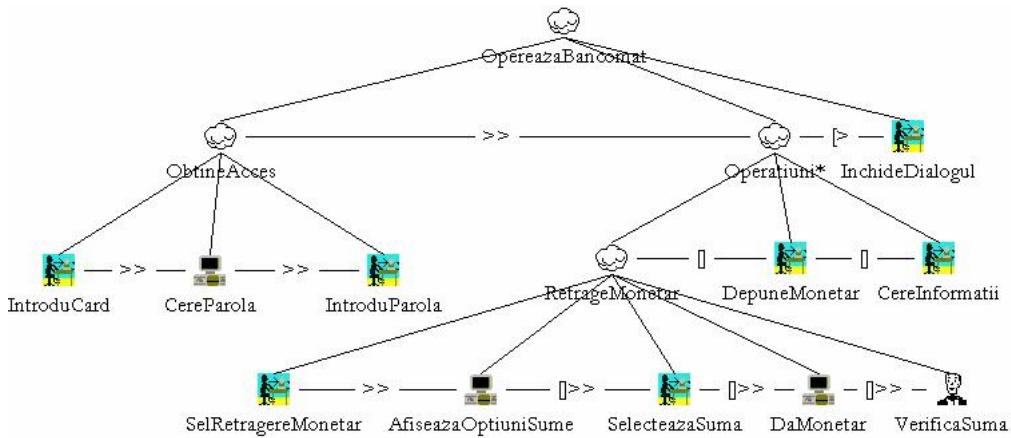


Figura 4.12: Modelul sarcinilor pentru interacțiunea cu un bancomat folosind CTT

Euterpe

Euterpe este instrumentul de analiza sarcinilor asociat metodei GTA. Acesta dispune de funcționalități de editare a modelelor sarcinilor și a tuturor proprietăților sarcinilor. Modelele sarcinilor pot fi convertite în format HTML, iar validitatea acestora poate fi verificată automat. Desi dispune de funcționalitatea completă necesară culegerii datelor necesare analizei, în ceea ce privește modelarea sarcinilor instrumentul prezintă lipsuri. Aceste lipsuri se traduc prin absența folosirii unui formalism necesar exprimării relațiilor temporale dintre sarcini. Specificarea relațiilor temporale dintre sarcini este optională, și chiar și în cazul în care utilizatorul dorește specificarea acestora, o poate face textual folosind cuvintele alese de el, astfel încât este dificilă păstrarea consistenței între modelele dezvoltate de proiectanți diferiți.

CTTE

CTTE (ConcurTaskTrees Environment) este instrumentul asociat notației CTT. Acest instrument este destinat în principal etapei de mo-

delare a sarcinilor. Instrumentul permite editarea modelelor sarcinilor cu stabilirea relațiilor temporale dintre sarcini prin intermediul operatorilor temporali LOTOS. În plus față de Euterpe, CTTE permite editarea modelelor sarcinilor cooperative (acele sarcini la care participă mai mult de un rol). Modelele sarcinilor pot fi salvate în format XML sau sub formă de imagini JPG.

4.4 Proiectarea interfeței utilizator pe baza modelelor sarcinilor

Analiza și modelarea sarcinilor participă la achiziția de informații relevante despre utilizatori și sarcini, informații folosite în proiectarea interfeței utilizator. Informațiile despre utilizatori, sarcini, contextul muncii sunt esențiale în proiectarea de sisteme utile și utilizabile, ajutându-i pe proiectanți să înțeleagă cum pot fi sprijiniți utilizatorii în munca lor.

Proiectarea interfeței utilizator cuprinde proiectarea funcționalității, structurii dialogului și prezentării. Rezultatele analizei sarcinilor se constituie dintr-o descriere detaliată a domeniului problemei și o determinare a aspectelor care pot fi îmbunătățite, cele din urmă constituind scopurile proiectării sistemului (cerințele). Trecerea de la analiză la proiectare este dificilă, deoarece presupune pe lângă o serie de pași bine definiți (inginerie) și creativitate. În această fază se încearcă găsirea răspunsului la întrebări precum:

- care sunt principalele ecrane?
- care sunt datele care trebuie reprezentate și care sunt cele care se constituie mai degrabă în atrbute?
- ce stil de interacțiune este mai potrivit?

- cum ar trebui să navigheze utilizatorul între ecrane?
- cum se va face accesibilă funcționalitatea sistemului?

În practică se pornește cu o proiectare inițială creată pe baza analizei sarcinilor care este supusă unui proces iterativ de evaluare/modificare. Pașii care trebuie urmați în trecerea de la analiză la proiectare sunt următorii:

1. dezvoltarea unui model conceptual al universului activității (fără referiri la instrumente sau sisteme folosite);
2. identificarea sarcinilor majore și obiectelor care trebuie să facă parte din sistem; acestea vor forma structura de nivel înalt a interfeței;
3. structurarea aplicației;
4. crearea căilor de navigare în structura interfeței utilizator în funcție de structura activității;
5. proiectarea prezentării în funcție de stilul platformei [78].

4.4.1 Generarea modelului dialogului pornind de la modelul sarcinilor

Generarea automată a modelului dialogului pe baza modelului sarcinilor presupune determinarea multimilor sarcinilor activate simultan, care vor defini *stările* din modelul dialogului și identificarea *tranzițiilor* dintre stări, care vor defini comportamentul sistemului interactiv. Construirea automată a multimilor de sarcini activate simultan (Enabled Tasks Sets - ETS) [49] necesită definirea unor reguli semantice asociate fiecărui operator temporal utilizat care să surprindă inclusiv modificările asupra familiei de sarcini activate.

În această secțiune descriem o abordare în procesul de determinare a mulțimilor de sarcini activate simultan folosind semantica operatorilor. În acest scop propunem o adaptare a semanticii operaționale la domeniul modelării sarcinilor și descriem un procedeu formal de determinare a mulțimii ETS-urilor folosind regulile de inferență prezentate în [67]. Scopul final al acestui proces deductiv este de a determina într-un mod cât mai corect mulțimea de ETS-uri care va sta la baza construirii *modelului dialogului* interfeței utilizator.

Procesul construirii modelului dialogului se desfășoară în trei pași:

- identificarea stărilor sistemului;
- identificarea stărilor inițiale și finale ale dialogului;
- descoperirea tranzițiilor între stări.

§. Identificarea stărilor din modelul dialogului bazată pe semantica operațională

Semantica operațională a fost folosită cu succes în specificarea diverselor limbaje de programare. Prin efectuarea unor mici modificări, semantica operațională poate fi folosită cu succes pentru determinarea mulțimii stărilor din modelul dialogului.

Notăm mulțimea operatorilor temporali LOTOS prin $\mathcal{O} = \{\[], \gg, [>, \|, ||, | = |, *\}$. Prioritatea operatorilor este descrisă în cele ce urmează: selecție > compunere paralelă (interleaving, sincronizare) > dezactivare > ordine independentă > activare [50].

Scopul propus este ca pornind de la arborele sarcinilor (modelul sarcinilor) să obținem familia mulțimilor sarcinilor active simultan (ETS) care va conduce la determinarea fereștrelor sau prezentărilor unei interfețe utilizator. Pornind de la mulțimea ETS-urilor și folosind anumite caracteristici ale sarcinilor (tipul sarcinilor care poate fi: editare,

monitorizare, selecție simplă, selecție multiplă, control [78]) vom putea determina controalele (widgets) care vor trebui să apară în interfață.

Vom iniția descrierea procesului de construire a familiei de ETS-uri, notată prin \mathcal{E} , pornind de la modelul sarcinilor, oferind o definiție a *arborelui sarcinilor*. Această definiție este o adaptare a definiției conceptului de *arbore* [46].

DEFINIȚIE. Arborele sarcinilor

Un arbore al sarcinilor are un nod rădăcină (reprezentând scopul execuției sarcinii). Fiecare nod poate să fie o frunză (o sarcină unitate) sau un nod interior (reprezentând o subsarcină). Un nod interior are unul sau mai multe noduri fiu și îl vom numi nod părinte. Toți fiile aceluiași nod sunt frați. Oricare doi frați vecini sunt legați printr-un operator temporal aparținând mulțimii \mathcal{O} .

Pentru a defini sintaxa abstractă vom folosi notația introdusă de Haskell:

$$\text{Op} = [](\text{ST}, \text{ST}) \mid ||(\text{ST}, \text{ST}) \mid [> (\text{ST}, \text{ST}) \mid >> (\text{ST}, \text{ST}) \mid \\ \mid \mid (\text{ST}, \text{ST}) \mid (\text{ST})^* \mid |=| (\text{ST}, \text{ST})$$

O secvență de sarcini poate fi definită folosind următoarele reguli de producție:

$$\begin{aligned} ST &\rightarrow T \text{ op } T \mid T \text{ op } ST \mid T \mid T^* \\ T &\rightarrow \text{task} \\ \text{op} &\rightarrow \gg \mid [] \mid \parallel \mid \mid \mid [> \mid |=|. \end{aligned}$$

Formalizat, un arbore al sarcinilor poate fi definit similar unei gramatici, astfel:

DEFINIȚIE. Arborele sarcinilor

Un arbore al sarcinilor este un cvintuplu $T = (N, \Sigma, S, \mathcal{O}, P)$ unde:

- N este mulțimea neterminalelor reprezentate de sarcini care nu sunt frunze în arborele sarcinilor;
- Σ - este mulțimea terminalelor (frunze ale arborelui sarcinilor);

- S - sarcina globală - rădăcina arborelui;
- \mathcal{O} - mulțime de relații temporale, $\mathcal{O} = \{\[], \gg, [>, ||, |||, | = |, *\}$
- P - mulțimea producțiilor de forma $A \rightarrow \alpha_1 r_1 \dots r_i \alpha_{i+1}$ unde $A \in N$, $\alpha_i \in (N \cup \Sigma)$, $r_k \in \mathcal{O}$, $k = 1, \dots, i + 1$ [72].

Pentru a determina stările sistemului, propunem o abordare formală, bazată pe semantica operațională pentru generarea mulțimii stărilor sistemului, notată cu \mathcal{E} . Starea mașinii abstracte pentru modele ale sarcinilor conține o *stivă de sarcini* și *familia de ETS* \mathcal{E} unde sunt salvate mulțimile de sarcini generate de execuția mașinii abstracte.

Semantica operatorilor pentru modelele sarcinilor definește o relație " \rightarrow " care se interprează ca "este transformat printr-un singur pas de execuție în" [64]. Această relație o vom defini prin intermediul unor reguli de inferență. O expresie precum " $\llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket \text{skip} \rrbracket \mathcal{E}$ " se poate citi ca "execuția sarcinii t_1 cu familia de ETS-uri \mathcal{E} înseamnă execuția lui *skip* (nici o acțiune) cu familia de ETS-uri \mathcal{E} ". O expresie precum " $\llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}'$ " se citește ca "execuția sarcinii t_1 cu familia de ETS-uri \mathcal{E} înseamnă execuția subsarcinii t'_1 cu familia de ETS-uri \mathcal{E}' ".

Reguli complementare

Pentru evaluarea expresiilor corespunzătoare modelelor sarcinilor, în afara regulilor asociate operatorilor temporali vom avea nevoie de câteva reguli de inferență suplimentare referitoare la execuția secvențială (notată folosind simbolul ";") a unor sarcini:

$$\boxed{\overline{\llbracket \text{skip}; t \rrbracket \mathcal{E} \rightarrow \llbracket t \rrbracket \mathcal{E}}^{(SR_1)}}$$

Regula SR_1 ne arată că *skip* urmat de execuția unei alte sarcini t se evaluatează la execuția sarcinii t , fără modificări la nivelul lui \mathcal{E} .

$$\frac{[t_1]\mathcal{E} \rightarrow [t'_1]\mathcal{E}'}{[t_1; t_2]\mathcal{E} \rightarrow [t'_1; t_2]\mathcal{E}'} \text{ (SR}_2\text{)}$$

Dacă sarcina t_1 este evaluată la execuția uneia din subsarcinile sale t'_1 , atunci compunerea secvențială a sarcinilor t_1 și t_2 înseamnă execuția subsarcinii t'_1 urmată de execuția sarcinii t_2 . Familia de ETS-uri va fi actualizată cu modificările introduse de execuția lui t'_1 .

$$\frac{t_1 \rightarrow \text{skip}}{[t_1]\mathcal{E} \rightarrow [\text{skip}]\mathcal{E} \cup \{t_1\}} \text{ (ExR}_1\text{)}$$

Regula ExR_1 se aplică atunci când t_1 este o frunză a arborelui a cărei execuție este finalizată. În acestă situație execuția sarcinii înseamnă execuția lui skip și actualizarea lui \mathcal{E} prin adăugarea sarcinii.

Semantica operațională pentru operatorul de activare

Când două sarcini sunt relaționate prin operatorul de activare înseamnă că după efectuarea primei sarcini, cea de-a doua sarcină își va începe execuția. Acest fapt ne duce la concluzia că cele două sarcini vor apartine la două ETS-uri diferite și va fi surprins prin modificările care se vor efectua asupra familiei \mathcal{E} . Regulile de inferență pentru operatorul de activare sunt descrise în cele ce urmează:

$$\frac{[t_1]\mathcal{E} \rightarrow [t'_1]\mathcal{E}}{[t_1 \gg t_2]\mathcal{E} \rightarrow [t'_1; t_2]\mathcal{E}} \text{ (ER}_1\text{)}$$

Regula ER_1 descrie situația în care execuția sarcinii t_1 se realizează prin execuția uneia din subsarcinile ei, notată prin t'_1 . În acest caz, expresia $[t_1 \gg t_2]\mathcal{E}$ se valuează la expresia $[t'_1; t_2]\mathcal{E}$. Remarcăm faptul că \mathcal{E} rămâne neschimbată deoarece e nevoie de procesări ulterioare ale expresiei.

$$\frac{[t_1]\mathcal{E} \rightarrow [skip]\mathcal{E} \cup \{t_1\}}{[t_1 \gg t_2]\mathcal{E} \rightarrow [skip; t_2]\mathcal{E} \cup \{t_1\}} \text{ (ER}_2\text{)}$$

Regula ER_2 tratează situația în care sarcina de activare t_1 este deja încheiată (t_1 se evaluează la *skip*). În această situație expresia $[t_1 \gg t_2]\mathcal{E}$ se evaluează la execuția secvențială a acțiunii *skip* urmată de execuția sarcinii t_2 . Familia \mathcal{E} se îmbogățește prin adăugarea sarcinii efectuate.

$$[skip \gg t_2]\mathcal{E} \rightarrow [t_2]\mathcal{E} \text{ (ER}_3\text{)}$$

Regula ER_3 descrie modul în care se evaluează o expresie în care sarcina de activare este *skip*. În această situație mașina abstractă va executa doar sarcina t_2 , iar \mathcal{E} rămâne neschimbată.

$$[t_1 \gg skip]\mathcal{E} \rightarrow [t_1]\mathcal{E} \text{ (ER}_4\text{)}$$

Regula ER_4 descrie situația în care sarcina de activare este o sarcină finală (este cel mai din dreapta fiu al unui nod), situație în care expresia $[t_1 \gg skip]$ se evaluează la sarcina de activare.

Semantica operațională a operatorului de selecție

Evaluarea unei expresii de forma $t_1[]t_2$ este dependentă de opțiunea utilizatorului. Din acest motiv, s-a recurs la completarea notației operatorului cu simbolul o care reprezintă opțiunea utilizatorului. Facem convenția că dacă o se evaluează la valoarea 1, atunci prima sarcină a fost selectată pentru execuție (vezi ChR_1). Dacă o se evaluează la 2, atunci cea de-a doua sarcină va fi selectată pentru execuție (vezi ChR_3). Din punct de vedere al prezentării interfeței utilizator, pentru ca utilizatorul să poată selecta una din două sarcini trebuie ca ambele sarcini să fie disponibile simultan. În cele ce urmează sunt prezentate regulile de inferență pentru operatorul de selecție.

$$\frac{[\![o]\!] \mathcal{E} \rightarrow 1 \mathcal{E} \quad [\![t_1]\!] \mathcal{E} \rightarrow [\![skip]\!] \mathcal{E}'}{[\![t_1 [o] t_2]\!] \mathcal{E} \rightarrow [\![skip]\!] \mathcal{E}' \cup \{t_1, t_2\}} \text{ (ChR}_1\text{)}$$

$$\frac{[\![o]\!] \mathcal{E} \rightarrow 1 \mathcal{E} \quad [\![t_1]\!] \mathcal{E} \rightarrow [\![t'_1]\!] \mathcal{E}'}{[\![t_1 [o] t_2]\!] \mathcal{E} \rightarrow [\![t'_1]\!] \mathcal{E}' \cup \{t'_1, t_2\}} \text{ (ChR}_2\text{)}$$

$$\frac{[\![o]\!] \mathcal{E} \rightarrow 2 \mathcal{E} \quad [\![t_2]\!] \mathcal{E} \rightarrow [\![skip]\!] \mathcal{E}'}{[\![t_1 [o] t_2]\!] \mathcal{E} \rightarrow [\![skip]\!] \mathcal{E}' \cup \{t_1, t_2\}} \text{ (ChR}_3\text{)}$$

$$\frac{[\![o]\!] \mathcal{E} \rightarrow 2 \mathcal{E} \quad [\![t_2]\!] \mathcal{E} \rightarrow [\![t'_2]\!] \mathcal{E}'}{[\![t_1 [o] t_2]\!] \mathcal{E} \rightarrow [\![t'_2]\!] \mathcal{E}' \cup \{t_1, t'_2\}} \text{ (ChR}_4\text{)}$$

Semantica operatorului de dezactivare

Dacă t_1 și t_2 sunt două sarcini, iar t_2 este o sarcină de dezactivare pentru t_1 , înseamnă că atunci când începe execuția sarcinii t_2 , execuția sarcinii t_1 este întreruptă indiferent de starea în care se află (t_2 poate suspenda execuția oricăreia din subsarcinile lui t_1). Aceasta este un motiv pentru care regula de inferență asociată operatorului de dezactivare nu conține nici o ipoteză. De asemenea, din punctul de vedere al ETS-urilor, t_2 va trebui să aparțină fiecărui ETS care conține subsarcini ale lui t_1 . Acest aspect a fost surprins prin intermediul actualizării care se realizează la nivelul lui \mathcal{E} , adăugând sarcina de dezactivare tuturor ETS-urilor generate anterior:

$$[\![t_1 [> t_2]\!] \mathcal{E} \rightarrow [\![skip; t_2]\!] \mathcal{E}' \text{ (DR)}$$

unde $\mathcal{E}' = \{\{l_1, \dots, l_n, t_2\}, \forall \{l_1, \dots, l_n\} \subseteq \mathcal{E}\}$.

Semantica operațională a operatorului de compunere paralelă - Pure interleaving

Operatorul de interleaving indică faptul că acțiunile sarcinilor aflate în execuție pot să succedă unele altora. Astfel, cele două sarcini vor trebui să aparțină aceluiași ETS.

$$\boxed{\frac{}{\llbracket \text{skip} \parallel \text{skip} \rrbracket \mathcal{E} \rightarrow \llbracket \text{skip} \rrbracket \mathcal{E}}}^{(ConcR_1)}$$

Regula $ConcR_1$ exprimă faptul că prin execuția simultană a acțiunii *skip*, rezultatul este de asemenea *skip*, iar \mathcal{E} rămâne neschimbată.

$$\boxed{\frac{\llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}'}{\llbracket t_1 \parallel \text{skip} \rrbracket M \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}'}}^{(ConcR_2)}$$

$$\boxed{\frac{\llbracket t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_2 \rrbracket \mathcal{E}'}{\llbracket \text{skip} \parallel t_2 \rrbracket M \rightarrow \llbracket t'_2 \rrbracket \mathcal{E}'}}^{(ConcR_3)}$$

Regulile $ConcR_2$ și $ConcR_3$ descriu situațiile în care una din sarcinile paralele este evaluată la execuția uneia din subsarcinile sale, iar cealaltă sarcină este *skip*. În această situație, execuția simultană a celor două sarcini se reduce la execuția subsarcinii cu modificări asupra \mathcal{E} .

$$\boxed{\frac{\llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}'}{\llbracket t_1 \parallel t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \parallel t_2 \rrbracket \mathcal{E}'}}^{(ConcR_4)}$$

$$\boxed{\frac{\llbracket t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_2 \rrbracket \mathcal{E}'}{\llbracket t_1 \parallel t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t_1 \parallel t'_2 \rrbracket \mathcal{E}'}}^{(ConcR_5)}$$

$$\boxed{\frac{\begin{array}{c} \llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}' \\ \llbracket t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_2 \rrbracket \mathcal{E}'' \end{array}}{\llbracket t_1 \parallel t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \parallel t'_2 \rrbracket \mathcal{E} \mathcal{R}}}^{(ConcR_6)}$$

unde \mathcal{ER} este ETS-ul care se construiește astfel: $\mathcal{ER} = \{\{X, Y\} \mid \forall \{X\} \subset \mathcal{E}', \forall \{Y\} \subset \mathcal{E}''\}$. Ultimele trei reguli *ConcR₄*, *ConcR₅* and *ConcR₆* descriu situația în care execuția unuia sau ambelor sarcini se evaluează la execuția unei subsarcini. Rezultatul evaluării execuției simultane a două sarcini constă în execuția simultană a subsarcinilor lor, cu modificări aduse lui \mathcal{E} .

Semantica operațională a compunerii paralele - Sincronizarea

Regulile de inferență pentru operatorul de sincronizare sunt foarte similare celor pentru execuția concurentă, fiind prezentate în continuare.

$$\frac{}{\llbracket \text{sync } t_1 \parallel \text{sync } t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t_1 \parallel t_2 \rrbracket \mathcal{E}} \text{ (SyncR}_1\text{)}$$

$$\frac{\llbracket t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_2 \rrbracket \mathcal{E}'}{\llbracket \text{skip} \parallel \text{sync } t'_2 \rrbracket \mathcal{E} \rightarrow \llbracket t_2 \rrbracket \mathcal{E}'} \text{ (SyncR}_2\text{)}$$

$$\frac{\llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}'}{\llbracket \text{sync } t_1 \parallel \text{skip} \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}'} \text{ (SyncR}_3\text{)}$$

$$\frac{\llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}'}{\llbracket t_1 \parallel \text{sync } t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \parallel \text{sync } t_2 \rrbracket \mathcal{E}'} \text{ (SyncR}_4\text{)}$$

$$\frac{\llbracket t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_2 \rrbracket \mathcal{E}'}{\llbracket \text{sync } t_1 \parallel t_2 \rrbracket \mathcal{E} \rightarrow \llbracket \text{sync } t_1 \parallel t'_2 \rrbracket \mathcal{E}'} \text{ (SyncR}_5\text{)}$$

$$\frac{\llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}' \\ \llbracket t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_2 \rrbracket \mathcal{E}''}{\llbracket \text{sync } t_1 \parallel \text{sync } t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \parallel t'_2 \rrbracket \mathcal{ER}} \text{ (SyncR}_6\text{)}$$

unde \mathcal{ER} este ETS-ul construit astfel: $\mathcal{ER} = \{\{X, Y\} \mid \forall \{X\} \subset \mathcal{E}', \forall \{Y\} \subset \mathcal{E}''\}$

Semantica operațională a operatorului de ordine independentă

Expresia $t_1 \mid = \mid t_2$ înseamnă că t_1 și t_2 pot fi executate în orice ordine, dar ambele sarcini trebuie executate. Aceste sarcini vor trebui să aparțină aceluiași ETS. Regulile de inferență sunt prezentate în cele ce urmează:

$$\frac{\llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}}{\llbracket t_1 \mid = \mid t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1; t_2 \rrbracket \mathcal{E} \cup \{t_1, t_2\}} \text{ (OIR}_1\text{)}$$

$$\frac{\llbracket t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_2 \rrbracket \mathcal{E}}{\llbracket t_1 \mid = \mid t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t_1; t'_2 \rrbracket \mathcal{E} \cup \{t_2, t_1\}} \text{ (OIR}_2\text{)}$$

Primele două reguli descriu situația în care una din cele două sarcini e evaluată la execuția uneia din subsarcinile ei (spre exemplu t_1 se evaluează la execuția lui t'_1). În această situație, expresia $\llbracket t_1 \mid = \mid t_2 \rrbracket \mathcal{E}$ se evaluează la execuția subsarcinii t'_1 urmată de execuția sarcinii t_2 .

Regulile OIR_3 și OIR_4 se referă la situația în care una din sarcinile implicate este *skip*, iar cealaltă se evaluează la execuția uneia subsarcini. Ca rezultat, mașina abstractă va executa subsarcina și va actualiza \mathcal{E} .

$$\frac{\llbracket t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_2 \rrbracket \mathcal{E}}{\llbracket skip \mid = \mid t_2 \rrbracket \mathcal{E} \rightarrow \llbracket t'_2 \rrbracket \mathcal{E} \cup \{t_2\}} \text{ (OIR}_3\text{)}$$

$$\frac{\llbracket t_1 \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E}}{\llbracket t_1 \mid = \mid skip \rrbracket \mathcal{E} \rightarrow \llbracket t'_1 \rrbracket \mathcal{E} \cup \{t_1\}} \text{ (OIR}_4\text{)}$$

Semantica operațională a operatorului de iterare

Operatorul de iterare este un operator unar care aplicat asupra unei sarcini indică faptul că sarcina e iterativă (după finalizarea unei execuții, sarcina poate să își înceapă execuția din nou).

Prima regulă de inferență spune că dacă execuția sarcinii este finalizată atunci evaluarea iterării se face la *skip*, cu adăugarea sarcinii iterative la ETS:

$$\frac{[t_1]\mathcal{E} \rightarrow [skip]\mathcal{E}}{[t_1^*]\mathcal{E} \rightarrow [skip]\mathcal{E} \cup \{t_1\}} (ItR_1)$$

Dacă sarcina iterativă nu este finalizată (una din subsarcinile sale este încă în execuție), iterarea este evaluată la iterația fiului și îmbogățirea ETS-ului cu sarcina părinte.

$$\frac{[t_1]\mathcal{E} \rightarrow [t'_1]\mathcal{E} \cup \{t_1\}}{[t_1^*]\mathcal{E} \rightarrow [t'^*_1]\mathcal{E} \cup \{t_1\}} (ItR_2)$$

§. Identificarea stărilor inițiale și finale ale dialogului

Deoarece este necesar să putem identifica starea/stările inițiale și starea/stările finale ale sistemului, este necesar să putem identifica primele/ultimele sarcini care se execută în utilizarea sistemului modelat de arborele sarcinilor. Determinarea sarcinilor inițiale și finale nu poate fi realizată într-o manieră imediată, de aceea, vom defini inițial două funcții ajutătoare care ne vor furniza cea mai din stânga secvență de sarcini a arborelui, respectiv cea mai din dreapta secvență de sarcini, acestea fiind folosite mai apoi pentru identificarea primelor, respectiv ultimelor sarcini efectuate. În continuare definim mulțimea **LMTS** (Left Most Tasks Sequence) [72] după cum urmează:

DEFINIȚIE. **LMTS**

Dacă X este o sarcină dintr-un arbore de sarcini, atunci $LMTS(X) = \{u | u \in (\Sigma \cup \mathcal{O})^k, X \Rightarrow ur\alpha, r \in \{\gg, \ll\}, u = x_1r_1 \dots x_k, x_i \in \Sigma, r_i \in \mathcal{O}, r_i \notin \{\gg, \ll\} \text{ și } x_1, \dots, x_k \in \Sigma\}$.

Pe baza rezultatului returnat de LMTS este posibilă identificarea sarcinilor care se execută inițial, astfel:

DEFINIȚIE. **FirstTasks**

Dacă X este o sarcină dintr-un arbore de sarcini, atunci $\text{FirstTasks}(X) = \{z | z \in \text{LMTS}(X) \cap \Sigma\}$.

O situație specială apare atunci când părintele unei sarcini din $\text{FirstTasks}(S)$ se află într-o relație de concurență cu o altă sarcină. Fie $t \in \text{FirstTasks}(S)$ și t subsarcină a unei sarcini t_1 , iar $t_1 \parallel t_2$. În acest caz, $\text{FirstTasks}(S)$ va trebui să conțină pe lângă sarcina t și $\text{FirstTasks}(t_2)$.

Pentru a determina starea inițială din modelul dialogului vom calcula $\text{FirstTasks}(S)$, iar starea inițială va fi formată din mulțimea tuturor stărilor care conțin cel puțin una din sarcinile returnate de $\text{FirstTasks}(S)$. Vom nota mulțimea stărilor inițiale prin $\mathcal{IS} = \{E \subseteq \mathcal{E} | x \in E \text{ pentru fiecare } x \in \text{FirstTasks}(S)\}$.

Observație: dacă secvența z din definiția lui FirstTasks conține doar sarcini care nu au predecesori implicați în relații de concurență, starea inițială va fi formată dintr-o singură mulțime de sarcini active. În caz contrar, prima sarcină va apăra mai multor mulțimi active de sarcini care vor participa la formarea stării inițiale.

Determinarea stării finale se realizează prin definirea mulțimii LastTasks [72]. Aceasta se va construi pe baza mulțimii RMTS (**R**ight **M**ost **T**asks **S**equence) care va returna cea mai din dreapta secvență de sarcini care vor fi executate.

DEFINIȚIE. **RMTS**

Dacă X este o sarcină dintr-un arbore de sarcini, atunci $\text{RMTS}(X) = \{u | u \in (\Sigma \cup \mathcal{O})^k, X \Rightarrow \alpha r u, \alpha \in (\Sigma \cup \mathcal{O})^k, r \in \{\gg, \ll\}, u = x_1 r_1 \dots x_k, r_i \in \mathcal{O}, r_i \neq \gg, r_i \neq \ll, x_i \in \Sigma\}$.

DEFINIȚIE. **LastTasks**

Dacă X este o sarcină dintr-un arbore de sarcini, atunci $\text{LastTasks}(X) = \{u | u \in \text{RMTS}(X) \cap \Sigma\}$.

Ne confruntăm și în acest caz cu situația specială în care părintele unei sarcini din $\text{LastTasks}(S)$ se află într-o relație de concurență cu o

altă sarcină. Fie $t \in \text{LastTasks}(S)$ și t subsarcină a unei sarcini t_1 , iar $t_1 \parallel t_2$. În acest caz, $\text{LastTasks}(S)$ va trebui să conțină pe lângă sarcina t și $\text{LastTasks}(t_2)$. Multimea stărilor finale va fi definită de $\mathcal{FS} = \{E \subseteq \mathcal{E} | x \in E, \text{ pentru fiecare } x \in \text{LastTasks}(S)\}$ [72].

§. Identificarea tranzitieiilor

Tranzitieiile între stări sunt determinate de operatorii de *activare*, respectiv *dezactivare*. Pentru descrierea modelului dialogului considerăm potrivită folosirea diagramelor stratificate de stări [26], deoarece acestea permit descrierea concurenței și independenței într-o manieră sugestivă, precum și reprezentarea evenimentelor.

În identificarea tranzitieiilor vom folosi câteva reguli pe care le vom descrie în continuare. Introducem notația $X \mapsto Y$ pentru a sugera faptul că între stările X și Y există o tranzitie. Dacă t_1 și t_2 sunt două sarcini între care există relația de *activare*, ne putem afla în una din următoarele situații:

- între părintii lui t_1 și t_2 nu există o relație de concurență și:
 1. t_1 și t_2 sunt frunze - dacă $t_1 \in X \subseteq \mathcal{E}$ și $t_2 \in Y \subseteq \mathcal{E}$, atunci $X \mapsto Y$ - se creează două stări în diagrama stratificată de stări care vor conține ETS-urile din care fac parte sarcinile t_1 și t_2 și o tranzitie între ele care se declanșează la finalizarea sarcinii t_1 ;
 2. t_1 este o frunză, t_2 nu este frunză - dacă $t_1 \in X \subseteq \mathcal{E}$ și $\text{FirstTasks}(t_2) \in Y \subseteq \mathcal{E}$, atunci $X \mapsto Y$ - se creează o stare pentru ETS-ul care o conține pe t_1 și o stare pentru t_2 în care se vor include stările generate de prelucrarea subsarcinilor lui t_2 și se va adăuga o tranzitie de la t_1 spre $\text{FirstTasks}(t_2)$ care se va declanșa la finalizarea sarcinii t_1 ;

3. t_1 nu e frunză, t_2 e frunză - dacă $\text{LastTasks}(t_1) \in X \subseteq \mathcal{E}$ și $t_2 \in Y \subseteq \mathcal{E}$, atunci $X \mapsto Y$ - se va crea o stare pentru t_1 în care se vor include stările rezultate din prelucrarea subsarcinilor lui t_1 și o stare pentru t_2 cu o tranziție de la $\text{LastTasks}(t_1)$ spre t_2 ;
 4. t_1 și t_2 nu sunt frunze - pentru fiecare $t \in \text{LastTasks}(t_1)$ se aplică regula 2;
- dacă unul din părinții lui t_1 și t_2 e implicat într-o relație de concurență atunci $\forall X \subseteq \mathcal{E}, t_1 \in X$ de forma $(t_1, t_k), \exists Y \subseteq \mathcal{E}, t_2 \in Y$ de forma (t_2, t_k) astfel încât $X \mapsto Y$.

Pentru construirea mulțimii de stări din modelul dialogului se aplică următoarele reguli suplimentare:

- dacă între sarcinile t_1 și t_2 apare operatorul de selecție ($[]$), atunci în modelul dialogului vom avea o stare care va conține două substări etichetate cu numele celor două sarcini (t_1 și t_2);
- dacă între t_1 și t_2 apare relația de ordine independentă ($| = |$), în modelul dialogului vom avea o stare ce va conține două substări, una cuprinzând o tranziție de la t_1 la t_2 , iar cealaltă cuprinzând o tranziție de la t_2 la t_1 (deoarece acest operator semnifică faptul că cele două sarcini pot să se execute în orice ordine, dar trebuie să se execute amândouă).

§. Exemple

În această secțiune vom prezenta câteva exemple de generare a familiei de ETS-uri folosind regulile de inferență propuse. Primul exemplu descrie un model al sarcinilor care folosește operatorul de activare. Arborele sarcinilor este prezentat în figura 4.13.

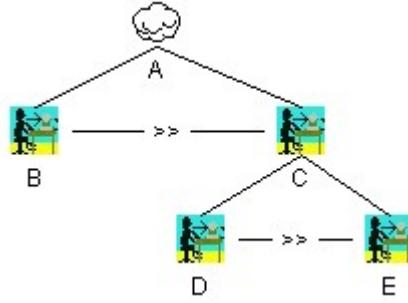


Figura 4.13: Arbore al sarcinilor folosind operatorul de activare

Formal, arborele de sarcini din Figura 4.13 poate fi descris astfel: $T = (N, \Sigma, A, O, P)$, unde $N = \{A, C\}$, $\Sigma = \{B, D, E\}$, $O = \{\gg\}$ și $P: A \rightarrow B \gg C; C \rightarrow D \gg E$. Conform Definiției 4.4.1 avem: $LMTS(A) = \{B\}$, $FirstTasks(A) = \{B\}$, $RMTS(A) = \{E\}$ și $LastTasks(A) = \{E\}$.

Expresia de la care vom porni în construirea familiei \mathcal{E} este frontieră arborelui, adică $\ll B \gg D \gg E \rr \mathcal{E}$. Aplicând regula ER_2 vom efectua primul pas din procesul deductiv astfel:

$$\frac{\ll B \rr \mathcal{E} \rightarrow \ll skip \rr \mathcal{E} \cup \{B\}}{\ll B \gg D \gg E \rr \mathcal{E} \rightarrow \ll skip; D \gg E \rr \mathcal{E} \cup \{B\}}$$

Notăm $\mathcal{E}' = \mathcal{E} \cup \{B\}$. Aplicând regula SR_1 evaluăm expresia $\ll skip; D \gg E \rr \mathcal{E}'$.

$$\overline{\ll skip; D \gg E \rr \mathcal{E}' \rightarrow \ll D \gg E \rr \mathcal{E}'}$$

Acum, aplicând regula ER_2 obținem:

$$\frac{\ll D \rr \mathcal{E}' \rightarrow \ll skip \rr \mathcal{E}' \cup \{D\}}{\ll D \gg E \rr \mathcal{E}' \rightarrow \ll skip; E \rr \mathcal{E}' \cup \{D\}}$$

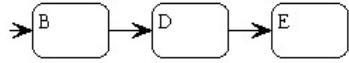


Figura 4.14: Modelul dialogului pentru arborele sarcinilor din Figura 4.13

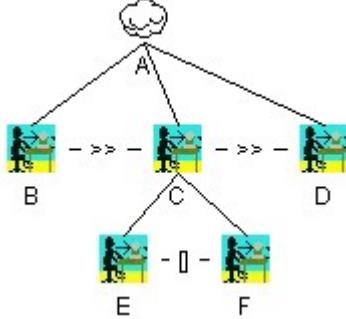


Figura 4.15: Arbore al sarcinilor folosind operatorul de selecție

Notăm $\mathcal{E}'' = \mathcal{E}' \cup \{D\}$. Folosind regula SR_1 , $\llbracket skip; E \rrbracket \mathcal{E}''$ se evaluaază la $\llbracket E \rrbracket \mathcal{E}''$.

$$\overline{\llbracket skip; E \rrbracket \mathcal{E}'' \rightarrow \llbracket E \rrbracket \mathcal{E}''}$$

Aplicăm ExR_1 pentru a evalua $\llbracket E \rrbracket \mathcal{E}''$ și obținem:

$$\frac{\llbracket E \rrbracket \mathcal{E}'' \rightarrow \llbracket skip \rrbracket \mathcal{E}''}{\llbracket E \rrbracket \mathcal{E}'' \rightarrow \llbracket skip \rrbracket \mathcal{E}'' \cup \{E\}}$$

La sfârșitul procesului deductiv conținutul memoriei \mathcal{E} va fi format din trei ETS-uri: $\{B\}$, $\{D\}$ și $\{E\}$. Modelul dialogului determinat conform regulilor enumerate anterior este prezentat în Figura 4.14.

Cel de-al doilea exemplu prezintă un arbore al sarcinilor care folosește operatorul de selecție. Presupunem că utilizatorul alege prima opțiune din cele disponibile (vezi Figura 4.15). Expresia care trebuie evaluată este $\llbracket B \gg E[o]F \gg D \rrbracket \mathcal{E}$.

Aplicăm regula ER_2 pentru sarcina B și obținem:

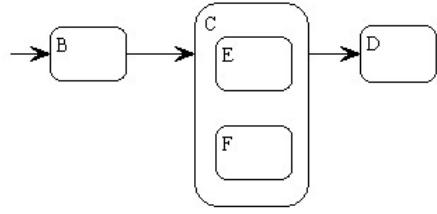


Figura 4.16: Modelul dialogului pentru arborele sarcinilor din Figura 4.15

$$\frac{[\![B]\!] \mathcal{E} \rightarrow [\![\text{skip}]\!] \mathcal{E} \cup \{B\}}{[\![B \gg E [o] F \gg D]\!] \mathcal{E} \rightarrow [\![\text{skip}; E [o] F \gg D]\!] \mathcal{E} \cup \{B\}}$$

Notăm $\mathcal{E}' = \mathcal{E} \cup \{B\}$. Aplicând regulile ChR_1 și ER_2 vom obține:

$$\frac{\frac{[\![o]\!] \mathcal{E}' \rightarrow 1 \mathcal{E}' \quad [\![E]\!] \mathcal{E}' \rightarrow [\![\text{skip}]\!] \mathcal{E}'}{[\![E [o] F]\!] \mathcal{E}' \rightarrow [\![\text{skip}]\!] \mathcal{E}' \cup \{E, F\}}}{[\![E [o] F \gg D]\!] \mathcal{E}' \rightarrow [\![\text{skip}; D]\!] \mathcal{E}' \cup \{E, F\}}$$

Notăm $\mathcal{E}'' = \mathcal{E}' \cup \{E, F\}$. Vom aplica regula SR_1 și următorul pas în procesul deductiv va fi:

$$\overline{[\![\text{skip}; D]\!] \mathcal{E}'' \rightarrow [\![D]\!] \mathcal{E}''}$$

Aplicând regula ExR_1 obținem:

$$\frac{[\![D]\!] \mathcal{E}'' \rightarrow [\![\text{skip}]\!] \mathcal{E}''}{[\![D]\!] \mathcal{E}'' \rightarrow [\![\text{skip}]\!] \mathcal{E}'' \cup \{D\}}$$

La sfârșit, \mathcal{E} ca fi $\mathcal{E}'' \cup \{D\}$, adică $\{E, F\}, \{D\}, \{B\}$. Modelul dialogului este descris în Figura 4.16.

Următorul exemplu ilustrează procesul de determinare a familiei de ETS-uri atunci când modelul conține operatorul de compunere paralelă.

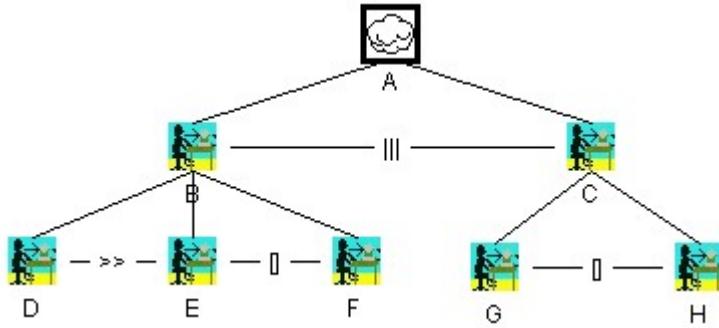


Figura 4.17: Arbore al sarcinilor folosind operatorul de compunere paralela

În acest caz $LMTS(A) = \{D\}$ și inițial $FirstTasks(A) = \{D\}$. Din cauza relației de concurență în care este implicat B, trebuie să calculăm și $FirstTasks(C) = \{G, H\}$. Astfel, $FirstTasks(A) = \{D, G, H\}$. Pentru a determina $LastTasks(A)$, calculăm mai întâi $RMTS(A) = \{G \parallel H\}$. Astfel, $LastTasks(A) = \{G, H\}$. Deoarece, C este implicat în relație de concurență cu B, vom calcula și $LastTasks(B) = \{E, F\}$. Prin urmare, $LastTasks(A) = \{G, H, E, F\}$.

Expresia care trebuie evaluată pentru determinarea stărilor sistemului este $D \gg E \parallel F \parallel G \parallel H$ (vezi Figura 4.17).

$$\frac{[D]\mathcal{E} \rightarrow [skip]\mathcal{E} \cup \{D\}}{[D \gg E [o_1] F \parallel G [o_2] H]\mathcal{E} \rightarrow [skip; E [o_1] F \parallel G [o_2] H]\mathcal{E} \cup \{D\}}$$

Notăm $\mathcal{E}' = \mathcal{E} \cup \{D\}$. Trebuie să evaluăm $[skip; E [o_1] F \parallel G [o_2] H]\mathcal{E}'$ folosind SR_1 .

$$\overline{[skip; E [o_1] F \parallel G [o_2] H]\mathcal{E}' \rightarrow [E [o_1] F \parallel G [o_2] H]\mathcal{E}'}$$

Aplicând SR_1 și ChR_1 obținem:

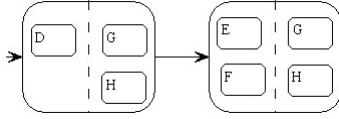


Figura 4.18: Modelul dialogului pentru arborele sarcinilor din Figura 4.17

$$\frac{\llbracket o_1 \rrbracket \rightarrow 1 \mathcal{E}' \quad \llbracket E \rrbracket \mathcal{E}' \rightarrow \llbracket \text{skip} \rrbracket \mathcal{E}'}{\llbracket E [o_1] F \rrbracket \mathcal{E}' \rightarrow \llbracket \text{skip} \rrbracket \mathcal{E}' \cup \{E, F\}}$$

$$\frac{}{\llbracket E [o_1] F \parallel G [o_2] H \rrbracket \mathcal{E}' \rightarrow \llbracket \text{skip} \parallel G [o_2] H \rrbracket \mathcal{E}' \cup \{E, F\}}$$

Notăm $\mathcal{E}'' = \mathcal{E}' \cup \{E, F\}$. Folosind $ConcR_3$ și ChR_2 rezultatul va fi:

$$\frac{\llbracket o_2 \rrbracket \mathcal{E}'' \rightarrow 2 \mathcal{E}'' \cup \{G, H\} \quad \llbracket G \rrbracket \mathcal{E}'' \rightarrow \llbracket \text{skip} \rrbracket \mathcal{E}''}{\llbracket G [o_2] H \rrbracket \mathcal{E}'' \rightarrow \llbracket \text{skip} \rrbracket \mathcal{E}'' \cup \{G, H\}}$$

$$\frac{}{\llbracket \text{skip} \parallel G [o_2] H \rrbracket \mathcal{E}'' \rightarrow \llbracket \text{skip} \parallel \text{skip} \rrbracket \mathcal{ER}}$$

Conținutul final al memoriei va fi $\mathcal{ER} = \{\{D, G, H\}, \{E, F, G, H\}\}$. Modelul dialogului este prezentat în Figura 4.18.

Pentru a ușura procesul de calcul al stărilor sistemului pornind de la modelul (arborele) sarcinilor, vom transforma initial arborele sarcinilor într-un *arbore cu priorități*. Procesul de transformare a arborelui sarcinilor într-un arbore cu priorități s-a realizat prin înlocuirea sarcinilor legate de un operator temporal cu o prioritate mai mare decât cea a operatorilor temporali de la același nivel cu o sarcină de tip abstract care are ca subsarcini tocmai sarcinile înlocuite. Prin această operație s-a păstrat semantica arborelui sarcinilor, însă i s-a extins structura. Pentru o prelucrare corectă a arborelui, va fi nevoie de o traversare în adâncime a acestuia, respectând astfel prioritatea operatorilor. Algoritmul propus pentru construirea arborelui de priorități este prezentat în cele ce urmează:

Algoritmul BuildPriorityTree(t) este:

Require: t sarcină

Ensure: (sub)arborele de rădăcină t va fi un arbore cu priorități creează un vector prior al operatorilor temporali dintre fiii lui t

for fiecare subsarcină t_i a lui t, mai puțin ultima **do**

@adaugă relațiile temporale dintre t_i (fiul curent) și t_{i+1} (fratele său) în vectorul prior

end for

repeat

$ord \leftarrow TRUE$

while $i < dimensiune(prior)$ **do**

$ord \leftarrow TRUE$

if $prior(i) < prior(i + 1)$ **then**

$j = i + 1;$

$ord \leftarrow false$

while $prior(j) \leq prior(j + 1)$ **do**

$j \leftarrow j + 1;$

end while

@creează o nouă sarcină T de tip abstract

@setează relația temporală a lui T la relația temporală dintre t_{j+1} și t_{j+2}

@adaugă toate subsarcinile $t_k, \forall k = i + 1, \dots, j + 1$ ca fii ai sarcinii T

@șterge legăturile subsarcinilor $t_k, \forall k = i + 1, \dots, j + 1$ spre t (șterge fiii)

@setează T ca fiind al $(i+1)$ -lea fiu al sarcinii t

@reconstruiește vectorul de priorități prior

else

if $prior(i) > prior(i + 1)$ **then**

$j \leftarrow i;$

while $prior(j) \geq prior(j + 1)$ **do**

$j \leftarrow j + 1;$

end while

@creează o nouă sarcină abstractă T

@setează relația temporală a lui T la relația temporală dintre t_j și t_{j+1}

@adaugă toate subsarcinile $t_k, \forall k = i, \dots, j$ ca și subsarcini ale lui T

@șterge legăturile subsarcinilor $t_k, \forall k = i, \dots, j$ spre t (șterge fiii)

@setează T ca fiind al i -lea fiu al sarcinii t

@reconstruiește vectorul de priorități prior

end if

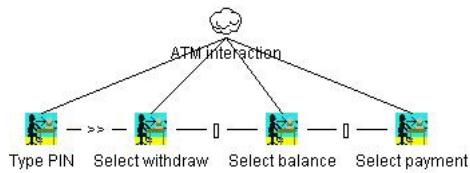


Figura 4.19: Modelul sarcinilor pentru interacțiunea cu un bancomat

```

end if
i  $\leftarrow$  i + 1;
end while
until ord = TRUE
for fiecare subsarcină ti a lui t do
    buildPriorityTree(ti);
end for

```

Pentru o mai bună ilustrare a procesului de transformare a arborelui sarcinilor într-un arbore cu priorități, prezentăm un exemplu referitor la interacțiunea cu un bancomat. Am redus modelul interacțiunii la o situație în care utilizatorul, după introducerea codului PIN poate să efectueze una din operațiile următoare: retragerea de numerar, consultarea soldului disponibil și efectuarea unei plăți (vezi Figura 4.19). Deoarece operatorul de selecție are o prioritate mai mare decât operatorul de activare, instrumentul generează o nouă sarcină abstractă care va avea ca și subarbore sarcinile legate prin operatorul de selecție (vezi Figura 4.20).

4.4.2 Generarea definiției abstracte a structurii interfeței utilizator pornind de la modelul sarcinilor

Următorul pas după construirea modelului dialogului este generația definiției abstracte a interfeței utilizator. Avantajul generării definiției abstracte a structurii interfeței utilizator este acela că ulterior interfața poate fi generată pentru diferite dispozitive fără un efort de reproiectare.

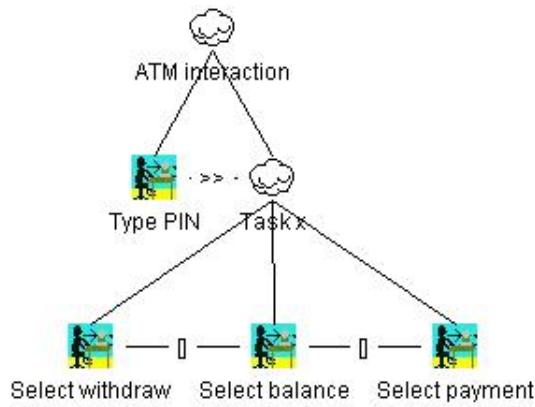


Figura 4.20: Arbore cu priorități pentru interacțiunea cu un bancomat

Pentru exemplificare am ales generarea definiției abstracte a interfeței utilizator în limbajul SUNML, care are o structură simplă, aşa cum s-a arătat în secțiunea 3.4. Generarea definiției abstracte a interfeței utilizator va porni de la familia de ETS-uri, \mathcal{E} . Acest fapt va asigura o grupare mai bună a sarcinilor în unități de dialog. Algoritmul de generare este următorul:

Algoritmul **BuildAbstractInterfaceDefinition(\mathcal{E})**
este:

Require: \mathcal{E} - familia de mulțimi de sarcini activate simultan

Ensure: se generează documentul de descriere abstractă a interfeței utilizator în limbajul SUNML

```

for fiecare  $E \subseteq \mathcal{E}$  do
    @creează un element <dialog> în descrierea SUNML
    for fiecare sarcină  $t \in E$  do
        if categoria sarcinii  $t$  este "Editare" then
            @adaugă un element etichetă cu numele sarcinii în descrierea SUNML
            @adaugă un element câmp de editare (textfield) cu proprietatea read/write
            în documentul SUNML
        else
            if categoria sarcinii  $t$  este "Control" then
                @adaugă un element <link> cu numele sarcinii în descrierea SUNML
            else

```

```

if categoria sarcinii t este "Selection/Single choice" sau "Selection/Multiple Choice" then
    @creează un element <list>
    for fiecare opțiune posibilă din listă do
        @creează un tag <element> în descrierea SUNML
    end for
    @creează un element </list>
end if
end if
end if
end for
    @creează un element </dialog> în descrierea SUNML
end for

```

4.5 Metoda DUTCH de proiectare a sistemelor

DUTCH este o metodă pentru proiectarea sistemelor complexe și necesită participarea unor echipe multidisciplinare (programatori, psihologi, etnografi, proiectanți industriali), fiecare responsabilă pentru un anumit aspect al proiectării. Datorită caracterului multidisciplinar al persoanelor implicate în proiectare apar constrângeri în ceea ce privește folosirea documentației, alegerea reprezentărilor care variază de la metode formale până la schițe informale sau scenarii. Aplicarea metodei presupune alocarea activităților complementare unor grupuri specializate care cuprind 3-5 persoane. Comunicarea între aceste grupuri trebuie gestionată cu atenție, în sensul că e indicată folosirea unei combinații de metode formale și informale.

O metodă practică de proiectare impune respectarea următoarelor cerințe:

1. definirea clară a unui proces;
2. definirea modelelor și reprezentărilor împreună cu semantica lor;

3. dezvoltarea de instrumente care sprijină crearea modelelor [81].

Metoda de proiectare DUTCH este bazată pe sarcina de muncă, ceea ce înseamnă că folosește sarcinile utilizatorilor ca forță de conducere în procesul de proiectare. Se consideră că pentru a dezvolta sisteme utile și utilizabile este nevoie ca proiectarea sistemelor să aibă la bază munca pe care oamenii trebuie să o desfășoare.

Procesul de proiectare constă din patru activități principale: analiza situației curente de muncă, imaginarea unei situații viitoare de muncă pentru care se proiectează soluția informatică, specificarea tehnologiei informaticice care se proiectează și evaluarea activităților anterior specificate, care va face procesul de proiectare ciclic.

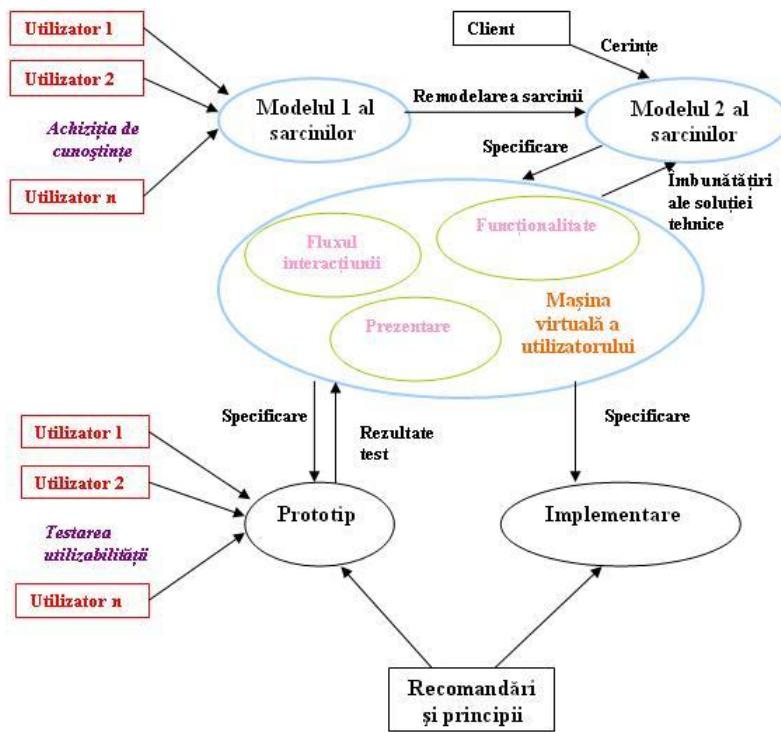


Figura 4.21: Procesul de proiectare DUTCH

Procesul de proiectare începe printr-o analiză extensivă a sarcin-

nilor, folosind metoda GTA. Analiza se va finaliza cu descrierea muncii, situațiilor de muncă, a utilizatorilor și a altor persoane afectate de introducerea unui nou sistem. GTA presupune construirea a două modele ale sarcinilor: modelul 1 al sarcinilor care este folosit în analiza situației curente a activității, este deci un model descriptiv, iar pe baza acestui model se construiește modelul 2 al sarcinilor, care este un model prescriptive. MS1 este construit de către psihologi și etnografi, iar la construirea MS2 participă reprezentanții tuturor disciplinelor anterior menționate.

Trecerea de la modelul 1 al sarcinilor la modelul 2 al sarcinilor este cel mai important pas în procesul de proiectare DUTCH. Pe baza cerințelor clientilor și a problemelor detectate în modelul 1 al sarcinilor și a dorințelor și ideilor utilizatorilor și altor persoane implicate se formulează cerințele de (re)proiectare. Toate disciplinele contribuie la propunerea de îmbunătățiri sau schimbări, iar aceste propuneri sunt evaluate în raport cu resursele disponibile, utilizabilitate, fezabilitate. Modelul 2 al sarcinilor va fi structurat similar modelului 1, dar nu va mai fi un model descriptiv al cunoștințelor utilizatorului, ci un model prescriptive al cunoștințelor pe care un utilizator expert al noii tehnologii trebuie să le posede.

Etapa de proiectare detaliată constă din specificare tehnologiei proiectate. Punctul de pornire pentru proiectarea detaliată este modelul 2 al sarcinilor, la care se adaugă decizii legate de prezentare (look and feel), proiectarea dialogului, proiectarea și ergonomia hardului. Specificările sarcinilor din MS2 sunt folosite ca bază în deciziile legate de funcționalitatea sistemului, în măsura în care aceasta este relevantă pentru utilizator. Pornind de la structura sarcinilor se determină o grupare inițială a funcțiilor și se descrie structura de navigare principală. Pe baza specificațiilor rezultate și a recomandărilor și principiilor de proiectare se construiește un prototip care este supus unor teste de utilizabilitate cu utilizatori direcți. Rezultate evaluării vor determina modificări ale proiectării UVM, până când rezultatele obținute la testele de utilizabilitate

sunt cele așteptate (vezi Figura 4.21). La final, pe baza specificărilor UVM se trece la implementarea sistemului interactiv.

Metoda de reprezentare a rezultatelor analizei sarcinilor a luat forma arborilor de sarcini. Reprezentarea grafică este ușor de urmărit și înțeles de către diverse persoane, mai ales când arborele conține până la 20 sarcini. Sarcinile din arborii de sarcini au atașate imagini, interviuri sau fragmente video, pentru a argumenta structurarea sarcinilor într-o anumită manieră.

Reprezentările folosite în etapa de proiectare detaliată iau forma schițelor, capturilor de ecran (screenshots), prototipuri și descrieri UAN (User Action Notation). Aceste reprezentări sunt atât de detaliate astfel încât pe baza lor se construiește un prototip, iar apoi se implementează întregul sistem. În faza de evaluare se folosesc scenariile, care reprezintă o descriere informală a sarcinilor care e posibil să apară simultan într-o anumită situație și care includ o descriere detaliată a implicării utilizatorului. Scenariile reprezintă o tehnică importantă care se folosește adeseori împreună cu analiza și modelarea sarcinilor. Acestea oferă informații descriptive despre o utilizare specifică a aplicației într-un context specific de folosire. Activitatea de identificare a unui scenariu semnificativ poate să includă majoritatea activităților cuprinse într-un model al sarcinilor. În mod ușual acestea sunt alese astfel încât să evidențieze aspecte specifice. Diferența principală dintre un model al sarcinilor și un scenariu este că modelul sarcinilor descrie toate activitățile principale împreună cu relațiile lor temporale, iar un scenariu descrie doar o secvență specifică de apariție a activităților. Scenariile pot fi folosite pentru scopuri diverse:

- înțelegerea domeniului unei aplicații: în acest caz scenariile sunt folosite pentru a evidenția aspectele esențiale;
- identificarea cerințelor: scenariile pot fi folosite pentru a identifica aspectele ”sensibile” care trebuie evitate;
- suport pentru analiza și modelarea sarcinilor: un scenariu sau un

grup de scenarii poate fi folosit pentru a identifica sarcinile principale și relațiile temporale dintre acestea;

- evaluare: este posibilă crearea de scenarii care să verifice dacă o aplicație satisfac anumite principii de utilizabilitate sau pentru a compara versiuni ale unui produs.

Scenariile își dovedesc utilitatea atunci când sunt puse în practică (jucate). Este recomandat ca “actorii” unor astfel de scenarii să fie utilizatori care sunt sceptici sau se tem de introducerea noii tehnologii, pentru că ei pot releva situații de eșec ale sistemului.

Metoda DUTCH a fost adoptată în patru universități din Olanda, două universități din România și este folosită în practică de o companie din Austria preocupată de proiectare de sisteme critice [81].

4.6 Studiu de caz

Pentru a verifica aplicabilitatea metodei de proiectare DUTCH am realizat un studiu de caz în care ne-am propus să urmăm pașii metodei pentru proiectarea unui sistem informatic pentru evaluarea posturilor de muncă din cadrul unei organizații folosind metoda pe puncte. Proiectarea s-a realizat folosind echipe interdisciplinare incluzând informaticieni, psihologi și experți în evaluarea posturilor de muncă. Am început procesul de proiectare cu analiza sarcinilor aplicând metoda GTA. Modelele sarcinilor rezultate în urma analizei au fost folosite pentru determinarea ecranelor aplicației și a structurii navigării. Folosind aceste informații s-a implementat aplicația care a fost supusă la final unui test de utilizabilitate care urma să confirme aplicabilitatea metodei pentru dezvoltarea de sisteme utilizabile. În cele ce urmează descriem detaliat etapele următoare în dezvoltarea produsului informatic.

Evaluarea posturilor de muncă este o activitate esențială în cadrul organizațiilor, rezultatele evaluării stabilind nivelurile salariale ale pos-

turilor [53]. În esență, la procesul de evaluare participa un psiholog și un număr de experți în domeniu. Așadar, *rolurile* identificate în activitatea de evaluare a posturilor de muncă sunt: *psiholog* și *expert*.

Obiectele manipulate în procesul de analiză sunt: analiza pieții, fișele de post, grila de punctare, ierarhia posturilor, instrucțiuni de evaluare, lista dimensiunilor evaluate, lista factorilor, lista nivelelor pe factori, lista posturilor, organigrame, rezultate brute medie, rezultate individuale brute, rezultate interpretate și dreapta de regresie dintre punctele acordate și salariul aferent fiecărui post de muncă. Pentru fiecare dintre roluri structura sarcinilor este reprezentată printr-un arbore de sarcini. Psihologul trebuie să stabilească lista posturilor, să aleagă experții care vor participa la evaluare, să stabilească dimensiunile, factorii și nivelurile factorilor, iar în final să centralizeze rezultatele evaluărilor, să stabilească ierarhia posturilor și să calculeze linia de regresie. În final el va reconfigura structura salarială corectă a organizației. Expertul care face evaluarea trebuie să parcurgă lista posturilor care i-a fost înmânată de psiholog, și pentru fiecare factor al dimensiunilor stabilite să aleagă un nivel, iar în final să predea evaluarea făcută psihologului. Arborii sarcinilor din MS1 pentru cele două roluri sunt prezentați în Figurile 4.22, 4.23 și 4.24. Arborii sarcinilor au fost construși folosind instrumentul CTTE [49].

În urma studiului posibilității automatizării procesului de evaluare, pe baza modelului 1 al sarcinilor, s-a ajuns la MS2. Rolurile rămân cele din MS1, dar o parte din activitățile realizate de agenții umani pot fi delegate sistemului. Pentru evaluatorul uman consultarea listei posturilor, memorarea evaluărilor făcute și consultarea instrucțiunilor de evaluare sunt realizate automat. Modelul 2 al sarcinilor de evaluare este prezentat în Figura 4.25.

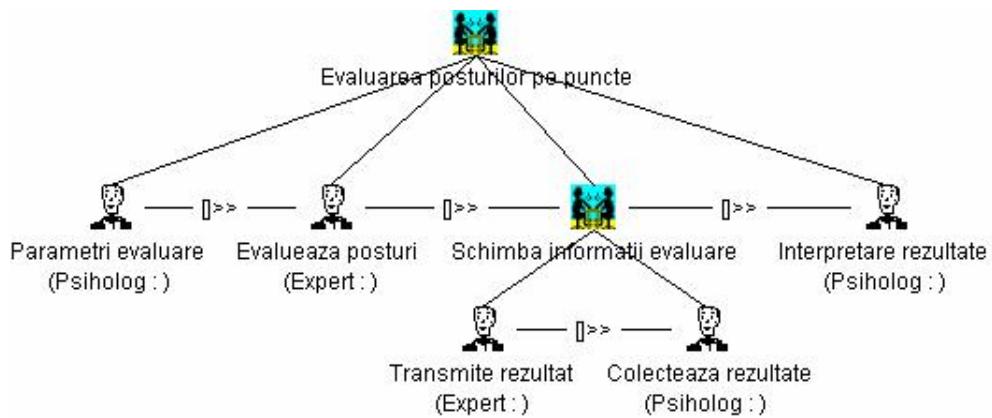


Figura 4.22: Modelul 1 al sarcinilor cooperativ

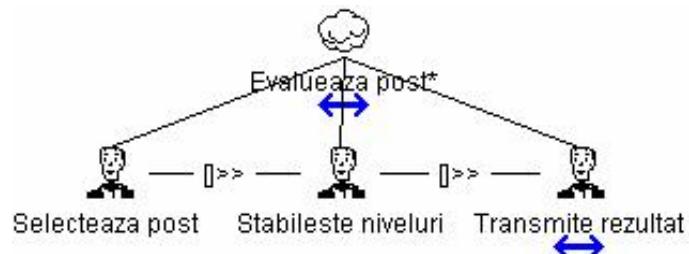


Figura 4.23: Modelul 1 al sarcinilor pentru rolul expert

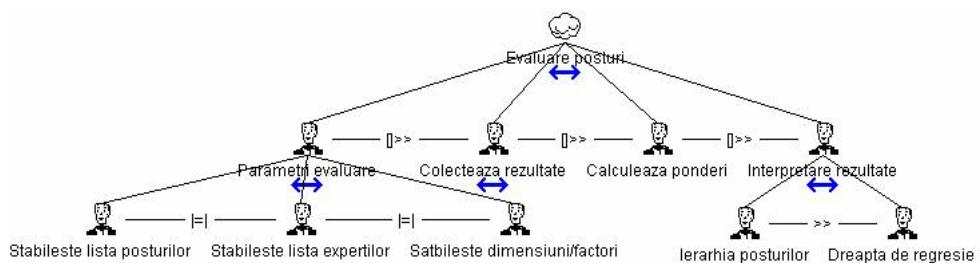


Figura 4.24: Modelul 1 al sarcinilor pentru rolul psiholog

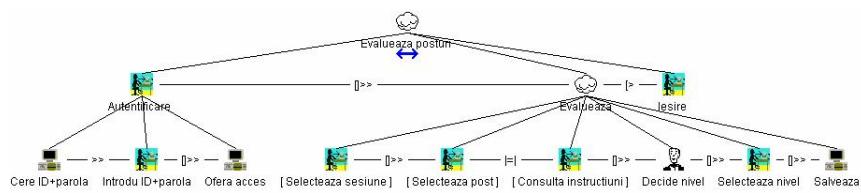


Figura 4.25: Modelul 2 al sarcinilor pentru rolul expert

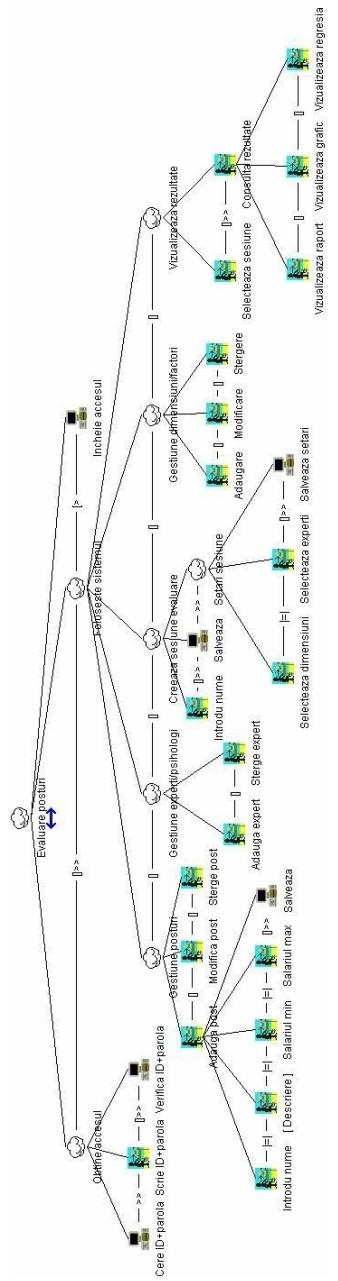


Figura 4.26: Modelul 2 al sarcinilor pentru rolul psiholog

Capitolul 5

Utilizabilitatea sistemelor interactive. Inginerie utilizabilității

Capitolul 6

Şabloane de proiectare a interfeţelor utilizator

Scopul oricărui dezvoltator de aplicaţii este ca aceasta să fie folosită de utilizatori cu uşurinţă. În capitolele anterioare ale prezentei lucrări am descris metode de dezvoltare a unor sisteme utilizabile, care însă pot fi consumatoare de timp. La fel cum în ingineria soft apariţia şablonelor de proiectare a constituit un real succes, în domeniul proiectării interfeţelor utilizator a început dezvoltarea de şabloane de proiectare a interacţiunii sau a prezentării diferitelor tipuri de aplicaţii. Acest capitol face o prezentare a abordărilor existente în formularea de şabloane pentru proiectarea interacţiunii dintre utilizator şi sistemul informatic precum şi şabloane de proiectare a interfeţelor diferitelor aplicaţii web. În cadrul acestui capitol este descrisă o metodă de dezvoltare a şablonelor de proiectare într-o manieră centrată pe utilizator, pornind de la analiza nevoilor utilizatorilor. Rezultatele analizei nevoilor utilizatorilor sunt comparate cu oferta existentă (în urma unui studiu al exemplelor existente) şi în final se trece la formularea şablonului. Abordarea propusă a fost folosită în dezvoltarea unui şablon de proiectare destinat dezvoltării de pagini web ale unor instituţii culturale (teatre/opere) [70], [68], [69].

6.1 Şabloane de proiectare în HCI

Interesul pentru formularea de şabloane pentru proiectarea interfeţelor utilizator s-a manifestat începând cu anul 1994. Deşi de atunci s-au dezvoltat câteva colecţii de şabloane, nu s-a ajuns la adoptarea uniformă a unui limbaj dedicat descrierii acestora. Numărul interfeţelor prost proiectate este mare şi soluţiile bune sunt greu de identificat, de aceea folosirea unor şabloane clar formulate ar putea să ofere soluţii satisfăcătoare. Obiectivul fundamental al şabloanelor este acela de a îmbunătăţi utilizabilitatea, de aceea, în conceperea şabloanelor se acordă o importanţă aparte indicatorilor măsurabili ai utilizabilităţii: uşurinţa de învăţare, uşurinţa de memorare, viteza de execuţie, rata erorilor, satisfacţia şi gradul de îndeplinire a unei sarcini. Dacă un şablon nu dovedeşte îmbunătăţirea a cel puţin unui indicator, atunci acesta nu e un şablon veritabil. Elementele unui şablon, conform formatului propus în [79], sunt:

- **Problema** - problemele sunt legate de utilizabilitatea sistemului şi se referă la utilizatori şi alte persoane interesate de sistem. Spre deosebire de şabloanele din ingineria soft care sunt concentrate asupra problemelor de concepţie cu care se confruntă proiectanţii, descrierile problemelor în acest caz sunt orientate pe sarcină;
- **Principiul de utilizabilitate** - şabloanele de proiectare au la bază un principiu pe care se construieşte soluţia;
- **Contextul** - contextul este focalizat pe utilizator şi descrie caracteristicile cadrului de aplicare în termeni de sarcini, utilizatori, mediu la care se poate aplica şablonul;
- **Soluţia** - soluţia trebuie să fie corectă şi să nu creeze noi probleme. Soluţia va descrie doar nucleul de rezolvare şi probabil va fi nevoie de alte şabloane pentru rezolvarea subproblemelor, iar alte şabloane relevante vor trebui menţionate;

- **Motivația** - această secțiune descrie modul în care funcționează şablonul, de ce funcționează și de ce e bun. Motivația trebuie să ofere argumente viabile pentru impactul asupra utilizabilității, surprinzând atât aspectele utilizabilității care au fost îmbunătățite, cât și cele care au fost negativ afectate, deoarece un şablon optimizează unul sau două aspecte, iar restul au de suferit;
- **Exemple** - această secțiune demonstrează modul în care a fost aplicat cu succes un şablon într-un sistem. Exemplul trebuie însoțit de o captură de ecran din lumea reală.

6.2 O abordare centrată pe utilizator pentru dezvoltarea unui şablon de proiectare a site-urilor web ale unor instituții de cultură

Această secțiune descrie rezultatele unui studiu care își propune determinarea metodei optime de proiectare a site-urilor web ale unor teatre/opere, concretizată în formularea unui şablon de proiectare. Metoda folosită în demersul formulării şablonului de proiectare constă din trei pași:

1. analiza nevoilor utilizatorilor;
2. evaluarea site-urilor web ale unor teatre/opere bine-cunoscute;
3. formularea şablonului [70].

6.2.1 Analiza nevoilor utilizatorilor

Studiul efectuat reprezintă un studiu explorator ce are ca scop identificarea celor mai relevante categorii de conținut în vederea pro-

iectării unui şablon de proiectare util în construirea site-urilor de prezentare a unor organizaţii cu profil artistic (teatru, operă). În vederea atingerii acestui obiectiv s-a recurs într-o primă fază la metodologia interviului pentru a identifica principalele dimensiuni pe care ulterior s-a construit un chestionar [68]. Chestionarul construit este format din 43 itemi grupaţi pe 7 dimensiuni (accesare de informaţii, comunicare, informaţii pentru turişti, colectivul teatrului, proiecte în derulare, managementul teatrului, istoricul teatrului). Aceste dimensiuni au fost identificate prin analiza interviului cu utilizatori precum şi din analiza site-urilor existente. Pentru fiecare dimensiune un grup de 5 experţi au generat itemi care apoi au fost supuşi unei proceduri de redistribuire pe dimensiuni, păstrându-se doar acei itemi care au întrunit un procentaj de acord de peste 70%. Fiecare item al chestionarului este evaluat pe o scală Likert cu 5 trepte, de la informaţie total nefolositoare la informaţie extrem de utilă. Chestionarul, construit în limba engleză, a fost aplicat online cu ajutorul unei aplicaţii Web interactive, pe un eșantion de 112 participanţi cu o medie de vârstă de 27.54 ani cu o distribuţie pe sexe echilibrată (49.1% bărbaţi, 50.9% femei). Participanţii provin din mai multe ţări (67% România, 12% Olanda, 8% Spania, 1.8% Elveţia, 1.8% Polonia, 3.6% SUA, 0.9% Franţa, 0.9% Letonia, 0.9% Indonezia, 0.9% Germania, 0.9% Austria, 0.9% Moldova). S-a căutat, de asemenea, să se includă respondenţi cu o ocupaţie variată pentru a obţine rezultate cât mai puternic generalizabile (0.9% pilot, 0.9% account manager, 2.7% actor, 0.9% agent vânzări, 30.4% studenţi, 4.5% învăţământ mediu, 13.4% învăţământ superior, 31.3% domeniul IT, 0.9% pensionaţi, 10.7% psiholog, 1.8% cadre medicale, 1.8% personal administrativ). Instrucţiunile de completare ale chestionarului au fost următoarele: *This questionnaire is designed to assess the most relevant types of information and actions that a theatre/opera website should provide. Please read carefully the statements listed and choose the answer that fits you best. There is no right or wrong answer and data collected by this survey are confidential and will be used only for academic purposes.*

ses. All the fields are required and for every question you should choose an answer.

Analiza utilizatorilor are obiectivul de a stabili care sunt informațiile pe care vizitatorii unui site web al unui teatru își doresc să le consulte și care sunt acțiunile care să le fie sprijinate de către site-ul web. Acest pas impune colectarea de date care s-a realizat prin conceperea unui chestionar care a fost oferit completării atât în mod obișnuit (pe hârtie), cât și online. Chestionarul a fost conceput cu participarea unor studenți ai Facultății de Psihologie și Științele Educației de la Universitatea Babeș-Bolyai din Cluj-Napoca. Pentru dezvoltarea chestionarului s-au derulat ședințe de brainstorming cu specialiștii din domeniu, precum și vizitarea unor site-uri web ale unor teatre, care au completat imaginea asupra domeniului. În urma acestui proces, s-au identificat următoarele dimensiuni și itemii corespunzători lor (dimensiunile sunt etichetate cu numere, iar itemii cu buline):

1. Accesul și informațiile despre serviciile furnizate de teatru: programul teatrului, prețul biletelor, planul locurilor, agențiiile de bilete, rezervarea/anularea biletelor pentru un spectacol, cumpărarea de bilete online, vizualizarea locurilor libere pentru un spectacol, cumpărarea online de materiale promovaționale sau suveniruri, facilități pentru persoanele cu dizabilități;
2. Comunicare, interacțiune, schimb de opinii: critici, realizarea de comentarii la critici, schimb de opinii cu alte persoane, trimiterea de mesaje actorilor, accesul la alte informații relevante direct din site-ul web al teatrului, crearea unui cont personal, facilități pentru membri;
3. Informații pentru turiști: locația teatrului, detalii tehnice despre teatru, tururi organizate ale teatrului, detalii arhitecturale ale clădirii, tururi virtuale ale clădirii, posibilitatea servirii cinei la teatru, posibilitatea efectuării de căutări în site;

4. Informații despre personalul teatrului: artiștii actuali, posturi vacante, audiuții programate, depunerea CV-ului online pentru un post;
5. Informații despre evoluția istorică a teatrului: evenimente istorice, spectacole și artiști premiați, stele în istoria teatrului, istoric al spectacolelor jucate, vizualizarea de fragmente din spectacolele anterioare, căutarea într-o bază de date cu spectacole, roluri, artiști;
6. Informații referitoare la managementul teatrului: organizarea de evenimente (închiriere de spații, colaborări cu artiști, producători), persoane de contact, înregistrarea de spectacole pentru televiziuni, radiouri, CD, DVD, programarea de întâlniri cu personalul administrativ, informații privind sponsorizările;
7. Informații despre proiectele prezente și viitoare: posibilitatea vizualizării de rezumate ale spectacolelor viitoare, informații despre festivaluri de teatru, proiecte viitoare, distribuția spectacolului, autor, producător, proiecte educaționale.

6.2.2 Evaluarea site-urilor web ale unor teatre bine-cunoscute

Acest pas a fost necesar pentru a obține informații despre starea curentă în domeniu, pentru a compara nevoile utilizatorilor cu ceea ce li se oferă și pentru a face câteva observații legate de proiectarea acestora. Interesul nostru s-a concentrat asupra aspectelor legate de conținut, prezentare și interacțiune. Un număr de zece teatre a fost supus evaluării, acestea fiind:

1. Teatrul The 5th Avenue <http://www.5thavenuetheatre.org/>;
2. Scala din Milano <http://www.teatroallascala.org/public/LaScala/index.html>;

3. Sydney Opera House <http://www.sydneyoperahouse.com/> ;
4. Moulin Rouge <http://www.moulinrouge.fr/home-flash-gb.html>;
5. Royal Opera House Covent Garden <http://www.royalopera.org/>;
6. The Shakespeare's Globe Theatre - <http://www.shakespeares-globe.org/> ;
7. Concertgebouw - <http://www.concertgebouw.nl/> ;
8. Teatrul Odeon - www.teatrul-odeon.ro ;
9. New National Theatre Tokyo - www.nntt.jac.go.jp/cgi-bin/english/index.cgi;
10. Baxter Theatre - www.baxter.co.za .

Criteriul pe care au fost alese site-urile web spre analiză a fost reputația de care se bucură aceste teatre și dispoziția geografică, intenția noastră fiind aceea de a acoperi cât mai multe zone geografice și culturale. Numărul de site-uri evaluate a fost determinat de ”indexul de surpriză” [35], ceea ce înseamnă că ne-am oprit din evaluare în momentul în care nu a mai apărut nimic nou sau surprinzător în paginile vizitate.

6.3 Dezvoltarea şablonului de proiectare a site-urilor web ale unor teatre/opere

Şablonul dezvoltat face parte din categoria şabloanelor de proiectare pentru web și va avea următoarele secțiuni:

- **Problema** - utilizatorii au nevoie de informații despre un teatru sau o operă în scopul de a participa la spectacolele organizate, de a vizita clădirea sau de a obține informații despre alte activități

ale teatrului (închirieri de spații, organizarea de mese, întâlniri cu publicul, spectacole înregistrate, colaborări).

- **Folosit când (contextul de folosire)** - în această secțiune vom folosi informațiile obținute din cestionare despre scopurile principale în care se doresc să fie folosite paginile web ale teatrelor/operelor;
- **Soluția** - această secțiune va sublinia care trebuie să fie principalele secțiuni ale siteului web al unui teatru și acțiunile utilizator care trebuie să fie sprijinite.
- **De ce (Motivația)** - această secțiune va conține argumentele pentru a alege o anumită structură a unui site web și aceste argumente se vor baza de asemenea pe rezultatele studiului cu utilizatorii;
- **Exemple** - secțiunea va conține capturi de ecrane din site-urile web analizate.

6.3.1 Publicul țintă

În această secțiune vom prezenta abordarea inițială pe care am avut-o în ceea ce privește categoriile de oameni interesați de activitatea teatrelor/operelor. Abordarea este intuitivă, iar categoriile prezentate în cele ce urmează au fost determinate în urma explorării siteurilor web ale unor teatre, reflectare asupra posibilelor opțiuni și discuții informale cu persoane implicate în activitatea teatrelor. Categoriile astfel determinate sunt prezentate în cele ce urmează: publicul (vezi Tabela 6.3.1), cercetători/studenți (vezi Tabela 6.3.2), investitori (vezi Tabela 6.3.3), turiști (vezi Tabela 6.3.2), artiști (vezi Tabela 6.3.4) sau alte teatre (vezi Tabela 6.3.5). Fiecare din categoriile menționate are interese diverse. În cele ce urmează vom analiza aceste categorii împreună cu interesele lor folosind o reprezentare tabelară, având două coloane - coloana "CE" - conținând infomația care se dorește să fie găsită și coloana "CUM" -

conținând modul în care informația căutată poate fi găsită fără a avea acces la un site web.

CE	CUM
Programul teatrului	Afișe, agenții teatrale
Cumpărare bilete	Agenții teatrale
Anulare/schimbare rezervări	Agenții teatrale
Consultare critici de teatru	Reviste de critică teatrală
Schimb de informații cu alți iubitori de teatru	Întâlniri, con vorbiri telefonice
Informații despre un artist (favorit)	Ziare, reviste, biografii, arhive
Informații despre turnee	Afișe, mass-media
Proiecte viitoare	Afișe, mass-media
Comunicare cu artiștii	Culise
Vizualizarea de fragmente din spectacole	Casete video, CD, DVD

Tabela 6.3.1: Publicul

Turiști, cercetători/studenți/profesori Această secțiune se referă la două categorii de oameni: turiștii și cercetătorii/studenții și profesorii, deoarece subiectele de interes se suprapun în mare măsură. Vom mai adăuga o coloană formatului de tabel prezentat anterior în care vom specifica cărei categorii de persoane îi corespunde un anumit subiect de interes. În cea de-a treia coloană vom avea fie un ”T” (turist), un ”E” (educație) sau ”TE” care va sugera că ambele categorii sunt interesante de un anumit aspect.

CE	CUM	CINE
Localizarea teatrului	Reviste, hărți, mass-media	T
Vizitarea teatrului	Carduri, reviste	TE
Cumpărarea de bilete	Agenții	T
Programul teatrului	Afișe, agenții	TE
Informații despre turnee în străinătate	Afișe, mass-media	T
Proiecte viitoare	Mass-media	TE
Artiști care au jucat, premii câștigate, date istorice despre teatru	Biblioteci, arhive, ziare	TE
Vizualizare fragmente din spectacole	Casete video, CD, DVD	TE
Genuri dramatice abordate (dramă, comedie)	Arhive, biblioteci	TE
Detalii tehnice (acustică, lumini, etc)	Arhive, biblioteci	E

Tabela 6.3.2: Turiștii

6.3.2 Structura generală a site-urilor web ale teatrelor

În urma analizei site-urilor web ale teatrelor s-a constatat că structura generală unui site web cuprinde următoarele secțiuni:

- Informații generale - aici vizitatorul va găsi informații despre locurile de parcare, ținuta la spectacole, facilități, avantaje pentru grupuri;
- Programul artistic (evenimente și calendar) - secțiune ce cuprinde

CE	CUM
Date de contact	Mass-media, cărți de vizită
Activități: colaborări, proiecte curente și viitoare	Întâlniri cu managerii
Nevoi financiare, avantaje pentru investitori (facilități pentru evenimente cu ușile închise, PR, publicitate, influențe în program, loje	Întâlniri cu managerii
Găzduire de evenimente (închiriere spațiu, colaborări cu artiști, producători	Întâlniri cu managerii
Înregistrări producții pt TV, radio, CD, DVD	Întâlniri cu managerii

Tabela 6.3.3: Investitorii

detalii despre piesele care se joacă și spectacolele viitoare;

- Cumpărare de bilete - în această secțiune sunt furnizate informații despre procesul de rezervare online, avantaje pentru grupuri, reduceri pentru diverse categorii de populație, abonamente.
- Planul sălii - această secțiune prezintă o schiță a sălii și detalii despre numărul de locuri, amplasamentul scenei, etc;
- Planifică-ți vizita - secțiune care informează vizitatorul despre agențiile de bilete, orarul și locația acestora, parcări, acces pentru persoane cu dizabilități; unele teatre oferă chiar și o vizită virtuală.

Site-urile mai complexe oferă un plus de secțiuni:

CE	CUM
Posturi libere	Mass-media
Oportunități de colaborare (auditii)	Mass-media

Tabela 6.3.4: Artiștii

CE	CUM
Repertoriul teatrului	Arhive, reviste
Oportunități de colaborare, schimburi de experiență	Întâlniri cu managerii
Organizare de festivaluri	Întâlniri cu managerii

Tabela 6.3.5: Alte teatre

- Recenzii ale audienței - prezintă feedbackul din partea publicului relativ la reprezentările prezentate;
- Magazin online - secțiune dedicată cumpărării de suveniruri, dar poate să conțină de asemenea cărți, CD-uri, DVD-uri;
- Educație - în această secțiune sunt prezentate activitățile de încurajare a accesului publicului tânăr la reprezentările de teatru și de sporire a gradului de înțelegere a unor genuri precum teatrul, opera, baletul;
- Sprijiniti-ne - este o secțiune care furnizează informații potențialilor sponsori sau altor categorii de persoane interesate în sprijinirea activităților teatrului (sponsori, instituții care doresc să lanseze un produs sau să clădească o marcă).

6.4 Rezultatele analizei nevoilor utilizatorilor

6.4.1 Fidelitatea chestionarului

Analiza coeficientului Alpha Cronbach pentru întreg chestionarul ne relevă o fidelitate bună a acestuia ($\alpha= 0.89$). De asemenea, analiza indicilor alpha pe fiecare scală a chestionarului ne indică o consistență bună a scalelor, cu excepția scalelor de informații turistice (alpha = .50) și proiectele teatrului (alpha= 0.29), scale care se dovedesc a nu avea o reprezentare clară și unitară în ceea ce privește eșantionul nostru. Acest fapt ar putea fi datorat heterogenității eșantionului de respondenți și perceptia diferită a dimensiunilor de proiecte și informații pentru turiști. Cu toate acestea, dat fiind scopul explorator și descriptiv al rezultatelor acestui chestionar, putem considera că nu există probleme de fidelitate, iar în utilizările viitoare putem propune revizuirea distribuției itemelor în cadrul celor două dimensiuni problematice. Un al doilea pas în analiza noastră a fost analiza datelor descriptive pentru cele 7 dimensiuni, pentru a determina care dintre categoriile de informații care ar putea fi incluse într-un site de prezentare a unui teatru sunt percepute ca fiind cele mai utile. Dimensiunile ale căror medii au fost cele mai ridicate sunt cele de informații despre accesul în teatru ($m=3.85$), urmate de informații turistice ($m= 3.65$) și categoria proiectelor prezente și viitoare ($m=3.38$). Informațiile despre colectivul de angajați ai teatrului au întrunit o medie de 3.08, iar cele privind istoricul teatrului o medie de 3.13. Medii scăzute au fost observate pentru categoriile de comunicare ($m=2.97$) și managementul teatrului (2.67) (vezi Figura 6.1).

Din analiza acestor rezultate putem concluziona că, datele acestui studiu pilot indică faptul că în cadrul site-urilor de prezentare a unui teatru ar trebui să predomine informațiile despre accesul în teatru și accesul la spectacole, precum și cele de informații turistice. O posibilă

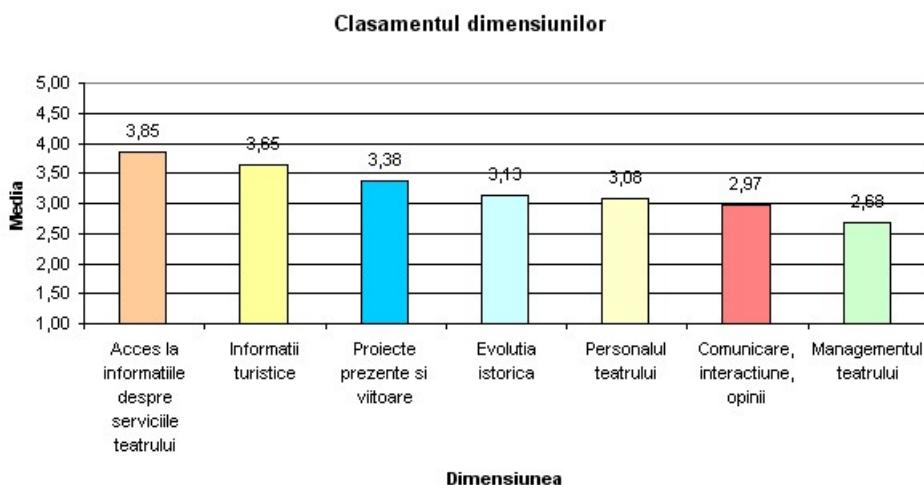


Figura 6.1: Clasificarea dimensiunilor

explicație pentru această ultimă categorie este prin luarea în considerare a caracteristicilor ei funcționale, utile nu doar strict pentru turiști, ci și pentru persoanele care locuiesc în același oraș.

Se observă de asemenea că cele mai puțin relevante categorii de conținuturi sunt considerate a fi cele ce țin de comunicare prin intermediul site-ului și cele legate de managementul teatrului. Acst lucru poate fi explicat făcând apel tot la specificul eșantionului chestionat. Majoritatea respondenților fiind români, și având calitatea de spectatori, nu neapărat implicați foarte puternic în viața artistică, e mai probabil să fie mai puțin interesați de schimbul de opinii avizate despre spectacole sau schimb de opinii cu artiștii implicați, sau de informații despre managementul teatrului. Mergând la o analiză mai aprofundată a mediilor pe fiecare item putem identifica acele categorii de informații sau de acțiuni permise de un site, considerate a fi cele mai relevante în cazul site-urilor de prezentare a unor teatre/opere ($m \geq 4$). Astfel, cele mai relevante categorii de informații sau acțiuni permise sunt cele legate de programul de spectacole al teatrului, prețurile biletelor și cumpărarea biletelor, efec-

tuarea sau anularea rezervărilor online, posibilitatea de a vedea locurile disponibile pe planul sălii. Acestea fac toate parte din dimensiunea informațiilor despre accesul la spectacole (vezi Figura 6.2).

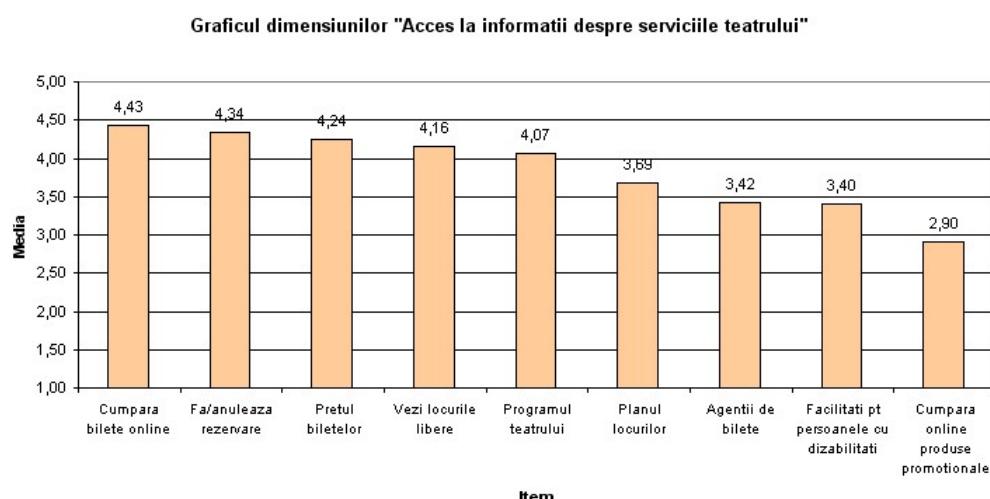


Figura 6.2: Acces la informații despre serviciile teatrului

Din dimensiunea informațiilor turistice o relevanță mare este acordată informațiilor despre locația teatrului precum și posibilitatea acțiunii de căutare a site-ului pentru diverse informații căutate de către posibili spectatori sau vizitatori (vezi Figura 6.3).

De asemenea, analizând celelalte dimensiuni, putem observa că în ceea ce privește comunicarea, cele mai importante sunt considerate a fi posibilitatea de a accesa cronici și informații relevante despre spectacole, și mai puțin posibilitatea de a crea un cont de membru sau a comunica direct cu artiștii (vezi Figura 6.4).

A patra dimensiune, cea de colectiv al teatrului, oferă doar o dimensiune relevantă pentru eșantionul nostru, cea de prezentare a artiștilor. Posibilitatea de a vedea posturile vacante sau de a aplica un CV online nu prezintă interes (vezi Figura 6.5).

În ceea ce privește evoluția istorică a teatrului, cele mai relevante

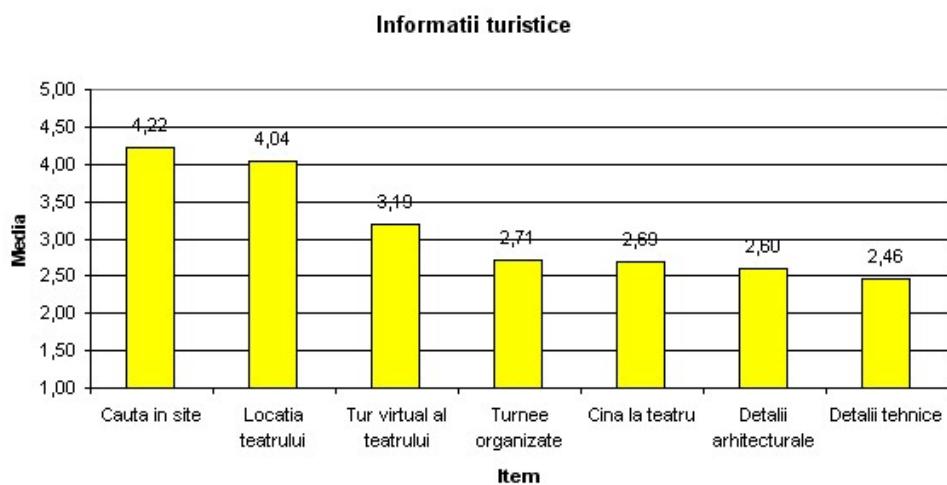


Figura 6.3: Informații turistice

categorii sunt raportate a fi premiile obținute precum și posibilitatea de a revedea fragmente sau de a căuta într-o bază de date cu toate spectacolele (vezi Figura 6.6). Premiile obținute de-a lungul timpului, starurile care au jucat de-a lungul istoriei teatrului sau palmaresul de spectacole puse în scenă în trecut prezintă o relevanță mai scăzută ($m \leq 3,00$).

Segmentul de proiecte în derulare este considerat relevant în totalitate ($m \geq 3,00$) pe toți itemii inclusi în această dimensiune (vezi Figura 6.7), în timp ce în cadrul dimensiunii de administrația teatrului singurul element identificat a fi relevant este constituit de persoanele de contact ($m \geq 3,00$) spre deosebire de aranjarea de evenimente, sponsori principali, posibilități de înregistrare și difuzare a spectacolelor, stabilirea de întâlniri cu personalul de management al teatrului ($m \leq 3,00$) (vezi Figura 6.8).

Superioritatea categoriilor de informații și acțiuni menționate asupra celoralte, din punct de vedere al relevanței lor pentru utilizator poate fi explicată prin prisma categoriei respondenților chestionați. Majoritatea se înscriu în categoria spectator pentru care aceste categorii de informații sunt mai relevante. Lucrurile ar putea fi însă cu totul diferite dacă am

Comunicare, interacțiune, opinii

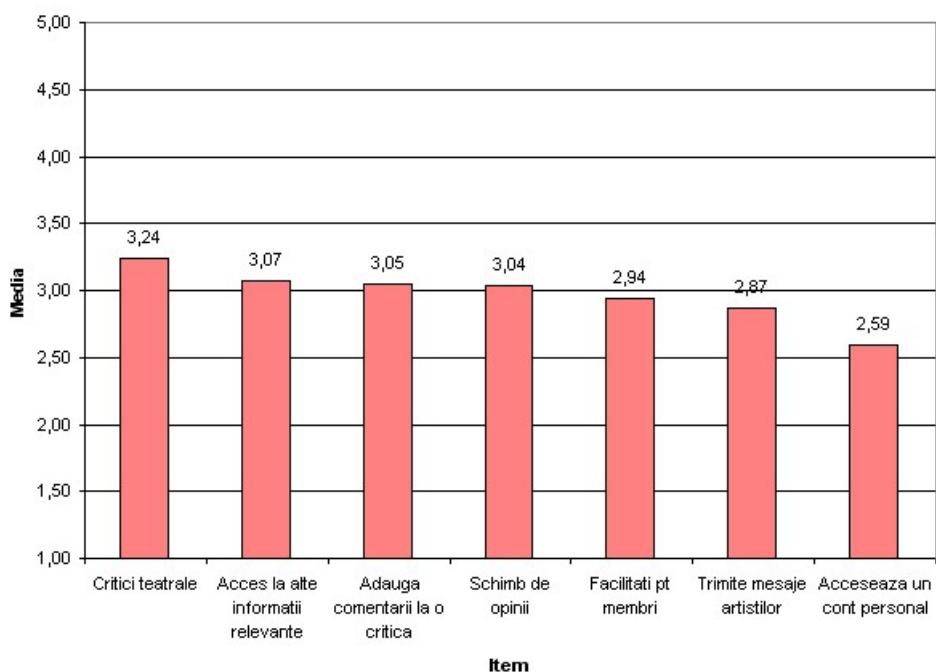


Figura 6.4: Comunicare și interacțiune

putea chestiona categorii diferite de utilizatori, cum ar fi artiști sau critici de artă sau colaboratori ai teatrelor.

O ultimă analiză efectuată a fost cea corelațională. Rezultatele arată că toate dimensiunile chestionarului corelează pozitiv și semnificativ între ele, ceea ce este în concordanță cu consistența internă ridicată a scalei. Din perspectiva eșantionului chestionat, informațiile considerate relevante ar putea să fie înscrise doar într-o singură dimensiune, cea de informații generale despre teatru și spectacole. Aspectul multidimensional al instrumentului trebuie însă investigat suplimentar prin chestionarea unor categorii variate de posibili utilizatori ai acestor tipuri de site-uri. O analiză suplimentară efectuată a fost corelația între vârsta respondenților și scalele chestionarului. Această analiză nu a relevat însă

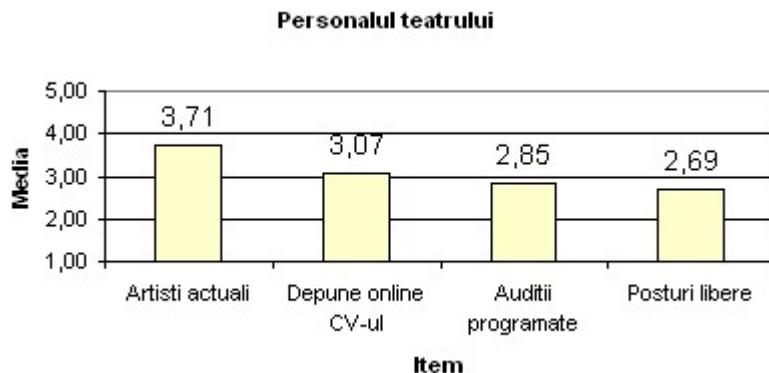


Figura 6.5: Personalul teatrului

nici o corelație semnificativă, situație care era de așteptat dată fiind vârsta destul de redusă a eșantionului, format preponderent din tineri. Posibilitatea existenței unor relații între relevanța anumitor dimensiuni și vârsta participanților trebuie investigată prin includerea unor categorii diverse de vârstă. Această concluzie este susținută și de rezultatele analizei factoriale de tip explorator care ne relevă faptul că datele converg către existența unui singur factor pe care încarcă puternic aproape toți itemii chestionarului. Putem deduce astfel că instrumentul construit are o structură unidimensională centrată preponderent pe informațiile cu referire la aspectul funcțional, de facilitare a accesului la spectacole, rezervări, cumpărări de bilete, informații despre spectacol și actori, și mai puțin pe informațiile de natură diferită care ar putea deveni funcționale pentru alte categorii de utilizatori.

6.4.2 Concluziile chestionarului și analizei site-urilor web

În urma acestei etape de analiză a nevoilor utilizatorilor am identificat categoriile de informații și acțiuni disponibile pe un site de prezen-

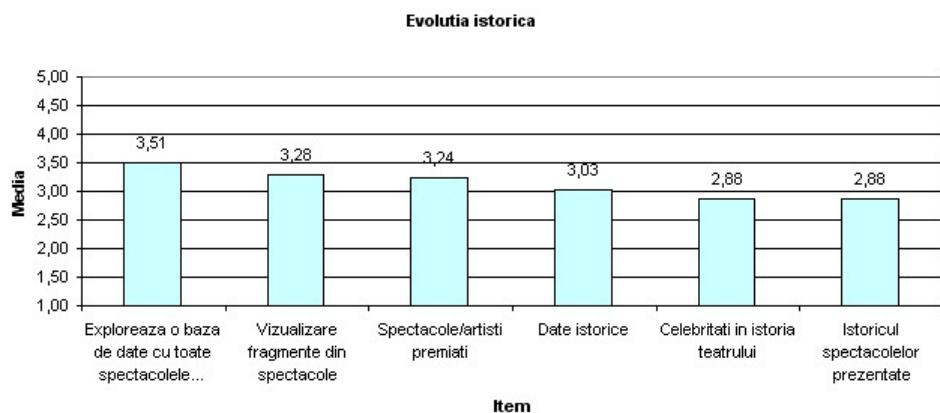


Figura 6.6: Evoluția istorică

tare a unui teatru/opere, considerate a fi cele mai relevante. Acestea se referă în primul rând la dimensiunea funcțională, de acces la spectacole, programul spectacolelor, prețul biletelor și posibilitatea de cumpărare online, vizionarea locurilor disponibile, accesarea informațiilor avizate despre spectacole (vezi Figura 6.9), locația teatrului, vizionare de fragmente, date despre autor (vezi Figura 6.10), distribuție și subiect, proiecte viitoare sau în pregătire, premii (vezi Figura 6.11), posibilitatea de contact cu personalul managerial (vezi Figura 6.12), posibilitatea de căutare în site pentru accesul rapid la informații. Ca urmare, sugerăm ca aceste categorii de informații să devină elemente principale în proiectarea unui astfel de site, prezente în meniul de home-page sau cu posibilitate de accesare rapidă, prin foarte puține operații. Așa cum am subliniat anterior, acest fapt poate fi corelat cu specificul eșantionului utilizat, majoritatea având calitatea de spectator. În condițiile utilizării unor categorii difierite, cum ar fi specialiști în domeniu, actori, critici, sau colaboratori ai teatrelor din lumea audio-vizualului sau a organizării de evenimente, categoriile de informații considerate relevante ar putea fi cu totul altele. Ca urmare este necesară investigarea ulterioară a acestui aspect pentru a identifica diferențele în funcție de calitatea utilizatorilor în interacțiunea

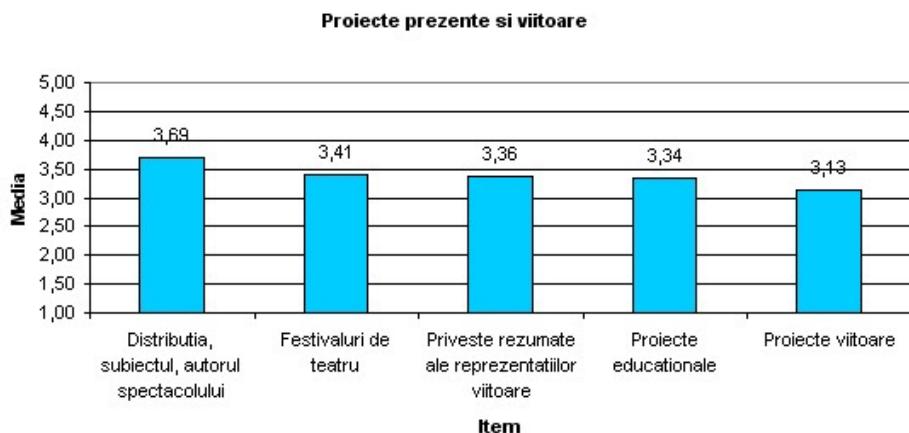


Figura 6.7: Proiecte prezente și viitoare

cu lumea teatrului/operei. Dacă există diferențe semnificative, aşa cum ne aşteptăm, vom putea propune proiectarea unor site-uri adaptate la tipul de utilizatori. Deși am pornit această analiză a nevoilor utilizatorilor de la stabilirea principalelor categorii de persoane interesate să fie la curent cu ceea ce se întâmplă în teatre/opere, definirea intereselor lor specifice și a acțiunilor pe care trebuie să le întreprindă în vederea atingerii acestor interese, eșantionul nostru nu a întrunit persoane din toate aceste categorii. Ca urmare, era de așteptat ca acele informații și acțiuni care corespund intereselor specifice categoriei de spectator să apară și mai relevante în urma analizei noastre. O direcție viitoare de dezvoltare a acestui demers o constituie tocmai investigarea nevoilor utilizatorilor care fac parte din alte categorii și au interese diferite de informare și interacțiune cu teatrul/opera. Ca urmare, soluția de şablon de proiectare propusă pe baza acestor rezultate este destinată preponderent situațiilor în care principalul scop pe care teatrul/opera vrea să îl atingă prin intermediul site-ului este informarea, atragerea și facilitarea accesului publicului la spectacole. Pentru alte categorii de obiective mai extinse, care implică facilitarea interacțiunilor cu furnizori, contractori, lumea audio-vizualului sau organizare de evenimente, nevoile specifice de informare și utilizare

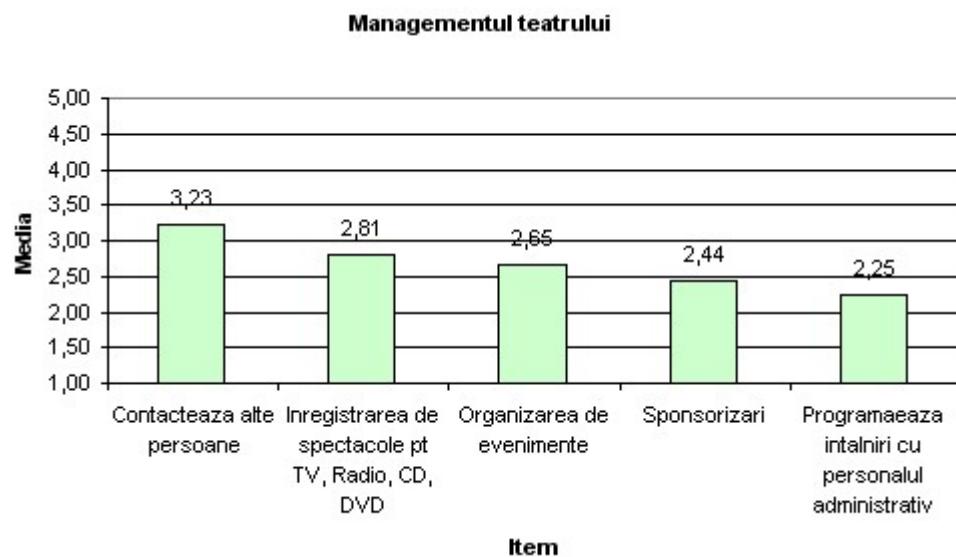


Figura 6.8: Managementul teatrului

a acestor parteneri vor trebui investigate printr-un demers suplimentar. De asemenea, nu au fost identificate relații între vârstă participantilor și preferința pentru informații dintr-o anumită categorie. Acest lucru poate fi explicat însă și datorită mediei reduse de vârstă a eșantionului care nu permite stabilirea unei astfel de relații. Acest aspect necesită la rândul lui investigații ulterioare, pe un eșantion largit și mai puțin omogen din punct de vedere al vârstei pentru a putea oferi sugestii cu privire la organizarea site-urilor. În concluzie, pentru categoria spectator, principalele elemente prezente într-un site oficial de prezentare a unui teatru/opere și posibil de accesat rapid sunt legate de: program, prețul biletelor, rezervări și anulări de rezervări (cu posibilitatea vizualizării locurilor disponibile), cumpărări, acces la cronică, informații despre locație, tur virtual, prezentarea artiștilor, a distribuției, vizualizare de fragmente, informații despre producție, proiecte viitoare, posibilitatea de contactare a persoanelor oficiale din teatru sau cea de căutare a site-ului pentru informații relevante. De asemenea, rezultatele obținute în urma analizei datelor chestionarului

lui ne îndreptățesc să susținem că acesta poate fi un instrument util în vederea explorării nevoilor utilizatorilor acestor categorii de site-uri și în vederea stabilirii principalelor categorii de conținut ale acestora în funcție de statutul și interesele unor grupuri distincte de utilizatori.

Clasificarea caracteristicilor

Conform rezultatelor aplicării chestionarului s-a realizat o împărțire a caracteristicilor în patru categorii: **extrem de utile** - cu valoarea numerică asociată 5, **foarte utile** - cu valoarea numerică asociată 4, **utile, dar neesențiale** - cu valoarea numerică asociată 3, și **mai puțin utile** - cu valoarea numerică asociată 2. A existat și o a cincea categorie, **lipsite de utilitate** - cu valoarea numerică asociată 1, căreia nu i s-a atribuit nici o caracteristică. În continuare vom analiza pe rând fiecare din categoriile de caracteristici menționate anterior.

Item	Procentaj	Număr siteuri
Cumpără bilete online	65%	8
Rezervare/anulare	61%	8
Prețul biletelor	48%	10
Caută în site	47%	4
Vezi locurile libere	46%	2
Programul teatrului	43%	10

Tabela 6.4.1: Caracteristici ”extrem de utile”

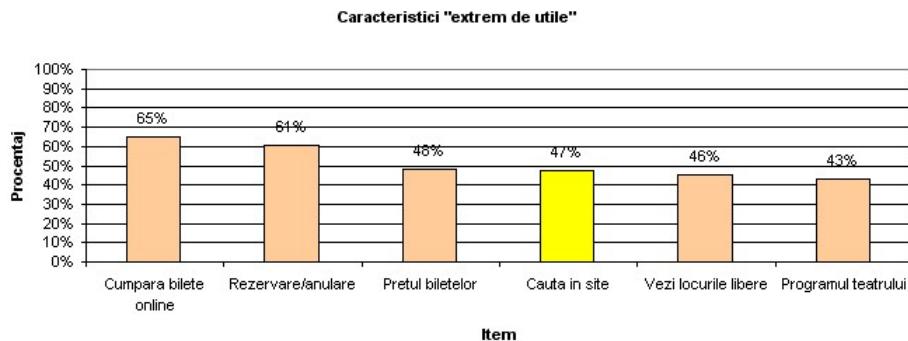


Figura 6.9: Grafic caracteristici "extrem de utile"

6.5 Şablonul de proiectare a paginilor web ale teatrelor/ operelor

Problema

Utilizatorii vor să acceseze informaţii despre spectacolele de teatru/operă, proiecte educaţionale, artişti şi alte activităţi culturale. Scopul principal al utilizatorilor este să acceseze informaţia despre spectacole sau alte activităţi ale teatrului (turnee, proiecte educaţionale, repetiţii, facilităţi pentru membri, sponsorizări).

Contextul de utilizare

Acest şablon se foloseşte la proiectarea unui site web al unui teatru/opere. Site-ul web al unui teatru are scopul principal de a atrage populaţia să participe la spectacolele prezentate de acesta. Spectacolele de teatru/operă (ca şi celealte reprezentaţii "live") au specific interacţiunea dintre artişti şi public. Această relaţie dintre membrii audienţei şi artiştii performer poate fi o bază solidă în procesul de creare a experienţei determinate de formarea unei comunităţi (*Community Building*). Membrii publicului sunt doritori să îşi exprime opinia asupra spectacolelor urmărite, la fel cum artiştii sunt doritori să afle părerea publicului. În funcţie de tipul spectacolului, site-ul web al unui teatru poate să ofere o experienţă

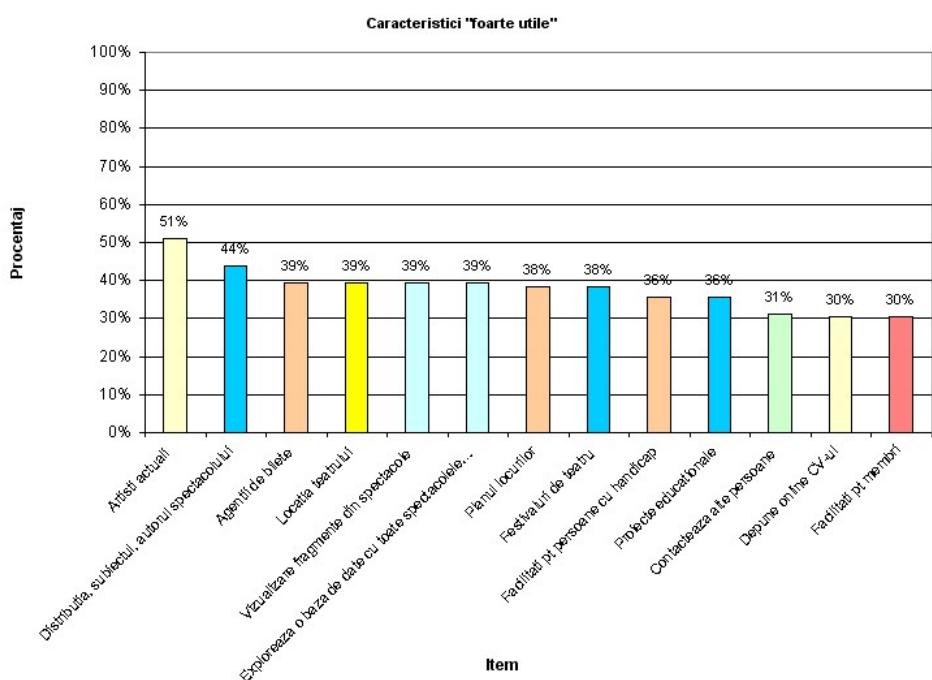


Figura 6.10: Caracteristici ”foarte utile”

Item	Procentaj	Număr siteuri
Artiști actuali	51%	4
Distribuția, subiectul, autorul spectacolului	44%	9
Agenții de bilete	39%	9
Locația teatrului	39%	8
Vizualizare fragmente din spectacole	39%	6
Explorează o bază de date cu toate spectacolele	39%	2
Planul locurilor	38%	8
Festivaluri de teatru	38%	2
Facilități pt persoane cu handicap	36%	7
Proiecte educaționale	36%	4
Contactează alte persoane	31%	8
Depune online CV-ul	30%	0
Facilități pt membri	30%	7

Tabela 6.4.2: Caracteristici "foarte utile"

relaxantă (*Fun Experience*) în prezentarea unui spectacol (prezentarea pe web a turneului național sau internațional al unui teatru este un candidat potrivit pentru o astfel de experiență, impulsionând oamenii să participe la spectacol). Din categoria scapurilor secundare fac parte: strângerea de fonduri prin furnizarea de funcționalități pentru membri și sponsori sau prin prezentarea publică a facilităților de organizare de evenimente precum lansări de produse, cine, petreceri. Teatrele au o misiune culturală în cadrul comunității, de aceea proiectele educaționale trebuie să aibă o secțiune dedicată în cadrul site-ului.

Publicul țintă al site-urilor web ale teatrelor este format în special din public, dar există și alte categorii de persoane interesate în activitatea teatrului, precum: cercetători, studenți, profesori, turiști, cărora trebuie

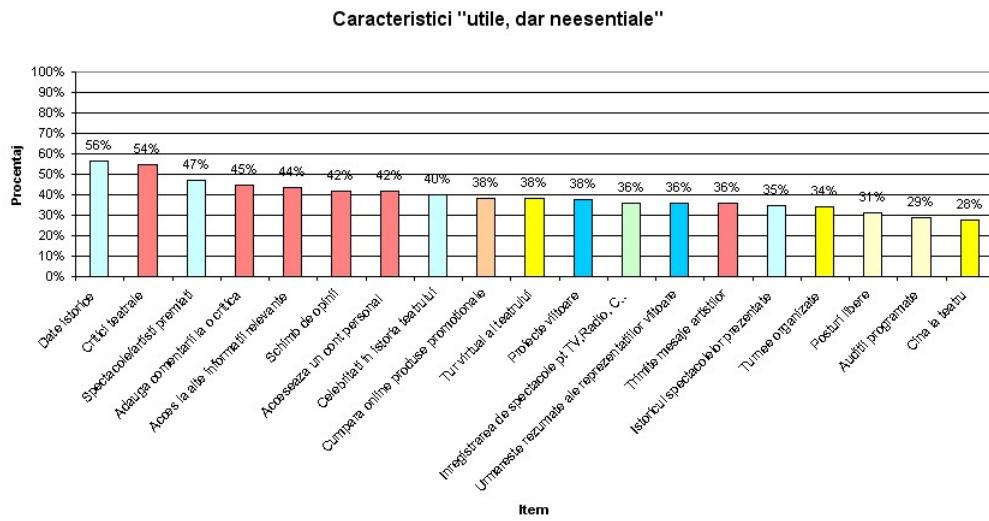


Figura 6.11: Caracteristici ”utile, dar neesentiale”

să li se furnizeze informații de interes.

Soluția

Creați un site web care să se concentreze pe spectacolele teatrului, informații pentru pentru posibilitii spectatori și informații pentru vizitatori. Funcționalitățile esențiale ale site-ului trebuie să includă rezervarea/vânzarea online de bilet, vizualizarea planului sălii și a locurilor disponibile, facilități de căutare și informații pentru membrii și sponsori.

Sarcini utilizator care trebuie sprijinite

Există o varietate de sarcini utilizator pe care site-ul web al unui teatru trebuie să le sprijine, incluzând:

- rezervare online pentru spectacole;
- căutare informații despre spectacole și consultarea de detalii despre acestea;
- vizualizare locuri libere;
- căutare într-o bază de date cu toți artiștii, spectacolele, rolurile

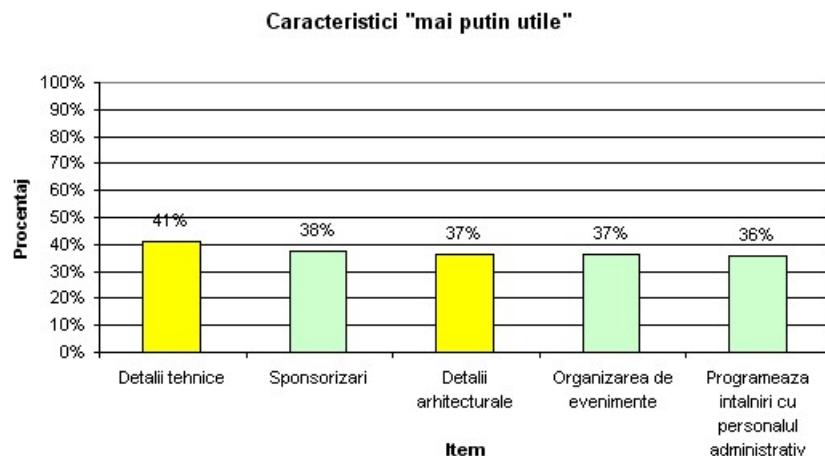


Figura 6.12: Caracteristici ”mai puțin utile”

interpretate;

- consultarea de informații despre evenimentele care vor urma (calendar);
- consultarea de informații necesare unei vizite;
- urmărește fragmente din spectacolele anterioare;
- consultarea de informații despre proiectele educaționale;
- consultarea de informații pentru persoanele cu dizabilități;
- contactarea altor persoane;
- citirea de critici teatrale, adăugarea de comentarii la critici;
- cumpărarea de obiecte de la magazinul online;
- vizualizarea sălilor teatrului, planul locurilor, viziunea asupra scenei de pe un anumit loc;
- consultarea detaliilor legate de organizarea de evenimente;

- obținerea de informații pentru scopuri de cercetare: trecutul istoric, spectacole care s-au jucat, celebrități în istoria teatrului;
- înscrierea pe liste de membri, sponsori.

Site-ul web al unui teatru trebuie să conțină următoarele secțiuni:

- Ce se joacă (What's playing) - această secțiune are scopul de a informa vizitorul despre spectacolele actuale și cele viitoare. Dacă teatru găzduiește diferite genuri artistice (teatru, operă, balet, concerte simfonice), este indicat ca acestea să fie clasificate. În mod uzual, spectacolele sunt prezentate folosind imagini (*Thumbnail*) ce conțin fie afișul spectacolului sau o fotografie din timpul spectacolului însotit de titlul spectacolului care este o hiperlegătură spre o pagină ce conține alte informații precum: distribuția, rezumatul subiectului, durata, link spre critici precum și opțiuni de adăugare a opinilor personale. Tot în această secțiune vor fi prezentate oferte speciale de bilete și un link spre pagina de cumpărare online a biletelor. Este indicat ca pentru fiecare spectacol să se prezinte și facilitățile oferite persoanelor cu dizabilități sau grupuri.
- Calendar - această secțiune are scopul de a informa vizitorii despre evenimentele următoare (fie spectacole, fie expoziții, ateliere de lucru, conferințe sau lansări de produse). Cea mai simplă metodă de implementare a unei astfel de secțiuni este folosirea unui selector de dată (*Date Selector*). Prin selectarea unei date din control, utilizatorul va vizualiza toate evenimentele care vor avea loc la data selectată. Unele site-uri web implementează o funcționalitate mai complexă, care include și criterii precum tipul evenimentului în căutarea evenimentului. O funcționalitate utilă este aceea de a oferi opțiunea "calendarul meu" (My calendar) în care vizitorii pot să adauge evenimente pentru perioada următoare.

- Informații despre bilete (Ticket info) - aceasta este cea mai căutată secțiune a site-ului web al unui teatru, majoritatea vizitorilor fiind interesați asupra prețului biletelor. O opțiune potrivită este aceea de a oferi posibilitatea consultării planului locurilor și, mai mult, consultarea locurilor libere. Puține site-uri ale unor teatre oferă posibilitatea selectării unui loc din sală și a observării perspectivei asupra scenei. Este recomandat ca în această secțiune să existe posibilitatea consultării unui help online sau procesul de cumpărare online să se facă folosind un *Wizard*. Detaliile despre cumpărarea biletelor pot fi reunite într-o subsecțiune numită "Cum/când cumpărați" (how/when to book). Ofertele speciale, informațiile legate de cumpărarea de bilete pentru grupuri sau abonamente trebuie să fie grupate într-o nouă secțiune. Orarul spectacolelor trebuie să fie accesibil din contextul cumpărării de bilete.
- Planifică-ți vizita - această secțiune este dedicată vizitorilor din alte regiuni geografice, dar conține informații utile și pentru alți vizitatori ai teatrului, precum: informații despre parcări, îndrumare, indicații, restaurante. În această secțiune se includ și informații despre agențiile de bilete. Oferta unui tur virtual al teatrului poate fi un factor determinant în decizia de a vizita teatrul. Dacă teatrul are un magazin online, din această secțiune se va furniza un link spre acest magazin virtual.
- Sprijiniți teatrul - secțiune care urmărește strângerea de fonduri prin intermediul membrilor și a sponsorilor. În această secțiune se pot crea subsecțiuni pentru donațiile individuale, sponsorizări din partea corporațiilor sau membri.
- Despre noi - această secțiune prezintă în mod uzual informații despre istoricul teatrului (fondatori, celebrăți din istoria teatrului, lista spectacolelor cu arhive conținând fragmente din spectacole).

Acstea informații trebuie să fie ușor de găsit, astfel încât este indicată prezentarea cronologică (*Timeline*) pentru fiecare categorie de informații. Tot aici se vor prezenta și proiectele educaționale în care este implicat teatrul. Instituțiile sau corporațiile cu care colaborează teatrul pentru diverse servicii (design de costume, lumină, sunet, decoruri) vor fi menționate în această secțiune.

- Colectivul - secțiune în care este prezentat personalul teatrului, cuprinzând manageri, regizori, artiști. Dacă numărul acestora este mare, este indicată folosirea unei indexări alfabetice. Prin selectarea unui nume vizitatorul va vizualiza biografia persoanei respective care va include și repertoriul, premii și cooperări avute de-a lungul timpului.
- Magazin - este posibil ca vizitorii să își dorească să cumpere cărți, CD-uri, DVD-uri legate de activitatea teatrului. Această secțiune este de fapt un site de comerț electronic (*E-commerce site*), oferindu-i utilizatorului experiența *Shopping*.

Site-ul trebuie să sprijine obligatoriu funcționalitatea de căutare (*Simple Search*), iar pentru categorii de vizitatori cu interese bine determinate, e recomandabilă implementarea funcționalității de căutare avansată (*Advanced Search*).

Motivația

Conținutul site-ului web al unui teatru trebuie să ofere informația pe care majoritatea vizitatorilor o caută. Din această cauză toate informațiile despre programul teatrului, prețul biletelor, planul locurilor, rezervare/anulare online, trebuie să fie întotdeauna vizibile și ușor accesibile. Majoritatea vizitatorilor se vor simți în siguranță când vor vedea că există o opțiune de căutare care le oferă oportunitatea de a găsi informația de interes. Posibilitatea căutării într-o bază de date cu informații despre spectacole, artiști, turnee, festivaluri poate fi de interes pentru mulți dintre vizitatorii site-ului. Pentru cei ce își propun să viziteze teatrul, data-

liile referitoare la o viitoare vizită sunt folositoare. Având la dispoziție toate datele necesare privind accesul, parcarea, planul locurilor, oamenii se vor simți mai confortabil înainte de a merge la teatru [69].

Alte exemple

Exemple demne de urmat în privința proiectării site-ului web al unui/unei teatru/opere sunt site-urile de prezentare a Sidney Opera House și Covent Garden Royal Opera House (vezi Figura 6.13).



Figura 6.13: Captură a paginii principale a site-ului web al Covent Garden Royal Opera House

Item	Procentaj	Număr siteuri
Date istorice	56%	9
Critici teatrale	54%	5
Spectacole/artiști premiați	47%	3
Adauga comentarii la o critică	45%	1
Acces la alte informații relevante	44%	7
Schimb de opinii	42%	2
Acceseză un cont personal	42%	2
Celebrități în istoria teatrului	40%	3
Cumpără online produse promotionale	38%	6
Tur virtual al teatrului	38%	3
Proiecte viitoare	38%	1
Înregistrarea de spectacole pt TV,Radio, CD, DVD	36%	1
Urmărește rezumate ale reprezentațiilor viitoare	36%	3
Trimite mesaje artiștilor	36%	1
Istoricul spectacolelor prezentate	35%	6
Turnee organizate	34%	5
Posturi libere	31%	4
Audiții programate	29%	2
Cina la teatru	28%	5

Tabela 6.4.3: Caracteristici ”utile, dar neesențiale”

Item	Procentaj	Număr siteuri
Detalii tehnice	41%	5
Sponsorizări	38%	6
Detalii arhitecturale	37%	4
Organizarea de evenimente	37%	5
Programează întâlniri cu personalul administrativ	36%	0

Tabela 6.4.4: Caracteristici ”mai puțin utile”

Capitolul 7

Evaluarea utilizabilității sistemelor soft

Utilizabilitatea este o calitate determinantă pentru succesul sau eșecul unui produs informatic. Evaluarea utilizabilității se poate realiza prin diverse metode, cea mai eficientă (și costisitoare) fiind testarea sistemului informatic cu utilizatori reali. O abordare promițătoare este automatizarea unor pași din procesul de evaluare a utilizabilității.

Un pas important spre utilizabilitate este evaluarea interfeței utilizator, ale cărei rezultate furnizează informații relative la îmbunătățirea acesteia. Un aspect important în evaluarea interfeței utilizator este identificarea sensului utilizabilității pentru aplicația curentă, fapt pentru care o analiză a utilizatorilor și a nevoilor acestora este esențială. Dacă nu știm ce vreau și ce au nevoie utilizatorii, nu vom ști ce sarcini trebuie să poată efectua folosind aplicația. De exemplu, referindu-ne la trei aplicații precum un joc, o aplicație bancară sau un sistem de control al traficului aerian, e ușor să identificăm cerințe de utilizabilitate complet diferite (pentru jocul pe calculator e important să fie ușor de folosit și să atragă utilizatorul, pentru sistemul bancar eficiența este esențială, iar pentru sistemul de control al traficului aerian o integrare a eficienței și siguranței

este esențială). Înainte de evaluarea interfeței utilizator este important să se stabilească obiectivul evaluării. Evaluarea se poate realiza la diferite momente de timp în cadrul procesului de dezvoltare. În fazele inițiale ale procesului de dezvoltare evaluarea utilizabilității se realizează pentru a alege între diferite posibile proiectări ale interfeței. Mai târziu, evaluarea se face pentru a stabili dacă proiectarea îndeplinește cerințele și îmbunătățește interfața.

7.1 Criterii pentru evaluarea interfețelor utilizator

Motivul principal pentru evaluarea unei interfețe utilizator este unul din următoarele:

- proiectarea având în vedere un obiectiv: este proiectarea suficient de bună?
- compararea unor alternative de proiectare: care este cea mai bună soluție?
- înțelegerea lumii reale: cât de bine funcționează produsul în lumea reală?
- verificarea conformanței la un standard: acest produs se conformată unui standard?

În demersul de proiectare a unui produs utilizabil se încearcă operaționalizarea unei ”definiții” proprii a utilizabilității pentru a putea stabili scopuri măsurabile. Acest lucru înseamnă definirea utilizabilității în termenii unor factori măsurabili, precum:

- performanța utilizatorilor în realizarea unor sarcini, măsurată prin intermediul ratei de efectuare a sarcinilor, timpului de execuție a sarcinilor, numărului erorilor;

- preferințele utilizatorilor sau gradul de satisfacție;
- ușurința de învățare, măsurată prin intermediul ratei de efectuare a sarcinilor, numărului erorilor, frecvenței folosirii helpului;
- flexibilitate care se referă la ușurința cu care sistemul se poate adapta unor schimbări ale cerințelor.

În evaluarea interfeței utilizator este important să se includă aspecte relative sarcinilor. În general, evaluarea interfeței utilizator după realizarea unei testări cu utilizatori reali ar trebui să includă următoarele aspecte:

- Sarcinile pe care utilizatorul a reușit să le realizeze;
- Sarcinile pe care utilizatorul nu a reușit să le realizeze;
- Numărul de execuții ale unei sarcini - dacă sarcina este executată de multe ori, ar putea fi implementată folosind o scurtătură sau un macro;
- Ordinea execuției sarcinilor - dacă utilizatorul încearcă folosirea unei ordini de realizare a sarcinilor diferită de cea stabilită de proiectant, proiectantul trebuie să ofere posibilitatea realizării sarcinilor într-o ordine diferită; în plus, dacă utilizatorul alege întotdeauna să realizeze o sarcină înaintea alteia, atunci cea de-a doua sarcină poate deveni automată la execuția primei sarcini.
- Erorile pe care utilizatorul le efectuează - dacă utilizatorul încearcă realizarea unei acțiuni inutile în realizarea sarcinii, aceasta este o eroare. De asemenea, dacă utilizatorul încearcă să se întoarcă în realizarea sarcinii la un pas anterior, este foarte probabil că acesta are nevoie de o confirmare că este în contextul dorit.
- Opinia subiectivă a utilizatorului asupra interfeței - acest aspect este esențial, pentru că dacă utilizatorului nu îi place o interfață,

este posibil să nu o folosească. Utilizatorii pot furniza informații utile proiectantului relative la părțile din interfață care le-au plăcut și care nu, dar calitatea acestor informații depinde de cunoștințele utilizatorului despre calculatoare și de experiența acestuia relativă la interfețele utilizator.

- Necesitatea utilizării helpului.
- Numărul situațiilor în care utilizatorul reia execuția sarcinii de la început - această măsură indică gradul de dificultate a navigării prin program.
- Timpul necesar utilizatorului pentru a realiza o sarcină - acest aspect e important când timpul este un factor important care apare în specificarea sistemului.

7.2 Evaluarea la distanță a utilizabilității

În ultimii ani s-a remarcat un interes crescut pentru evaluarea la distanță a utilizabilității, adică evaluatorii sunt separați în spațiu și/sau timp de utilizatori. Această abordare a fost inițiată din motive precum:

- disponibilitatea și performanțele sporite ale rețelelor;
- costul mare și rigiditatea metodelor de evaluare a utilizabilității în laboratoare;
- necesitatea minimizării costurilor evaluării utilizabilității pentru a o face mai accesibilă.

Realizarea unei evaluări la distanță a utilizabilității presupune colectarea automată a datelor despre interacțiunea dintre utilizator și sistem, folosind jurnalizarea. În secțiunile următoare popunem două abordări de evaluare la distanță a utilizabilității.

7.3 O abordare bazată pe AOP pentru evaluarea automată a utilizabilității

Pentru evaluarea utilizabilității unui produs informatic sunt efectuați, în mod general, următorii pași [31]:

- colectarea de date - etapă în care sunt adunate informații despre modul în care este folosit sistemul informatic;
- analiza datelor - etapă care cuprinde metode statistice de grupare a datelor și de identificare a problemelor;
- propunerea de sugestii de îmbunătățire a sistemului soft suspus evaluării.

Tipic, la un test de utilizabilitate evaluatorul propune un scenariu de utilizare a sistemului informatic. Sarcinile incluse în scenariu au scopul de a testa anumite aspecte ale interfeței utilizator. În timpul testului de utilizabilitate specialistul în utilizabilitate observă modul în care utilizatorul execută sarcinile, comportamentul său și comentariile pe care le face. La sfârșitul testului are loc o discuție privind modul de realizare a sarcinilor și gradul de satisfacție al utilizatorului în raport cu sistemul folosit.

Primele două activități din cele mai sus amintite pot fi supuse unui proces de automatizare. Avantajele pe care le aduce automatizarea evaluării utilizabilității sunt: reducerea necesarului de resurse umane, reducerea necesarului de expertiză umană, completitudine și posibilitatea comparării rezultatelor testelor de utilizabilitate.

Jurnalizarea și analiza evenimentelor care apar la nivelul interfeței utilizator au fost recunoscute a fi surse valoroase de determinare a problemelor de utilizabilitate. Prin jurnalizarea evenimentelor din interfața utilizator se pot obține informații precise despre ce face utilizatorul, ce date folosește, ce funcționalități folosește, în ce ordine execută acțiunile.

Astfel, metrii ale utilizabilității precum timpul de execuție a unei sarcini, frecvența de folosire a unei funcționalități sau numărul de erori pot fi calculate din fișierele de jurnalizare. Desigur, există și alte modalități de evaluare a acestor metrii (înregistrarea video a sesiunii de testare, notarea manuală), dar folosind jurnalizarea evenimentelor acest lucru se poate realiza automat. Datele înregistrate în fișierele de jurnalizare au un format specific care poate fi transformat în alte formate, operație urmată de supunerea datelor spre analiză. Desigur, jurnalizarea automată a evenimentelor din interfața utilizator are și neajunsuri, unul din acestea fiind imposibilitatea surprinderii reacțiilor utilizatorului.

În mod obișnuit jurnalizarea se realizează prin inserarea unor linii de cod în cadrul codului aplicației pentru a se surprinde elementele de interes din cadrul interacțiunii. Această abordare necesită o foarte bună cunoaștere a structurii aplicației și acces la codul sursă al aplicației. Decizia asupra locațiilor în care să se facă instrumentarea trebuie să aparțină unui expert în utilizabilitate care să colaboreze cu dezvoltatorii ai sistemului.

Propunerea pe care o aducem atenției în acest capitol referitor la jurnalizarea evenimentelor din interfața utilizator este aceea de a folosi o paradigmă de programare recent dezvoltată, și anume programarea orientată pe aspecte (AOP - Aspect Oriented Programming). Această paradigmă este recomandată tocmai în astfel de situații în care trebuie să tratăm situații de cerințe transversale. O cerință transversală este o caracteristică a unui sistem informatic a cărei implementare este împărtășită prin codul sursă al aplicației (exemplu: jurnalizarea, securizarea). Pentru implementarea unor astfel de cerințe AOP introduce patru noțiuni noi [37]:

- *join-point* - punct bine definit în execuția programului;
- *pointcut* - grupează o mulțime de join-pointuri și expune anumite valori din contextul de execuție al join-point-urilor;

- *advice* - secvență de cod care se execută la fiecare join-point dintr-un pointcut;
- *aspect* - este un tip care încapsulează pointcut-uri, advice-uri și proprietăți statice. Un aspect este unitatea de modularizare a AOP.

Aspectele sunt integrate într-un sistem folosind un instrument special care se numește *weaver*. Există extensii AOP pentru limbaje de programare bine-cunoscute precum Java, C++ sau C#.

O abordare în evaluarea automată a utilizabilității folosește paradaigma orientată pe aspecte pentru instrumentarea codului. Acest lucru înseamnă că se vor scrie aspecte pentru diferitele metriki care se doresc a fi evaluate. Avantajul pe care îl aduce abordarea propusă este acela că modulele care realizează procesele de evaluare a utilizabilității sunt separate de codul sursă al aplicației.

Pentru a testa aplicabilitatea metodei propuse s-a realizat un studiu de caz pentru o aplicație de gestiune a bugetului unei familiilor. Funcționalitatea oferită de aplicație include adăugarea unui venit, adăugarea unei cheltuieli, vizualizarea bilanțului pentru o anumită perioadă de timp. Participanții la test au trebuit să realizeze următoarele sarcini:

- adăugarea unui venit la o dată specificată (data1);
- înregistrarea unei cheltuieli la o dată specificată (data2);
- vizualizarea economiilor dintre data1 și data2;
- vizualizarea cheltuielilor dintre data1 și data2;
- vizualizarea tuturor veniturilor.

Pentru evaluarea utilizabilității s-au parcurs următorii pași [73]:

- înregistrarea fiecărei erori împreună cu momentul apariției;

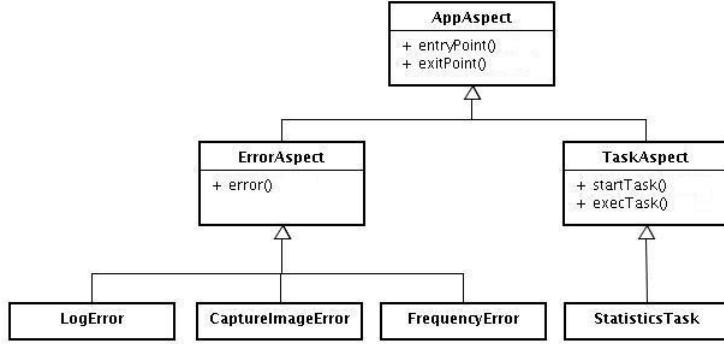


Figura 7.1: Ierarhia de aspecte definită

- preluarea unei capturi de ecran la apariția fiecărei erori pentru a vizualiza datele introduse de utilizator;
- calcularea frecvenței pentru fiecare tip de eroare;
- calcularea timpului de execuție a fiecărei sarcini;
- calcularea numărului de sarcini încheiate cu succes, abandonate (se începe execuția sarcinii) sau eşuate (se începe execuția sarcinii, dar apar erori).

Momentul începerii execuției unei sarcini e considerat acel moment în care utilizatorul selectează butonul corespunzător sarcinii din bara de instrumente sau selectează din meniu opțiunea corespunzătoare.

Aspectele descrise sunt prezentate în Figura 7.1.

Pentru studiul de caz realizat ierarhia de aspecte este prezentată în Figura 7.2.

Aplicația a fost dezvoltată folosind Java [33], iar ca extensie AOP s-a folosit AspectJ [9], [58]. Aplicația folosește metoda statică *show(String msg)* a clasei *ErrorMessage*. Pentru fiecare utilizator s-au creat fișiere jurnal și s-au calculat următoarele metriki de utilizabilitate: timpul de execuție a sarcinilor, numărul de erori, frecvența erorilor de un anumit

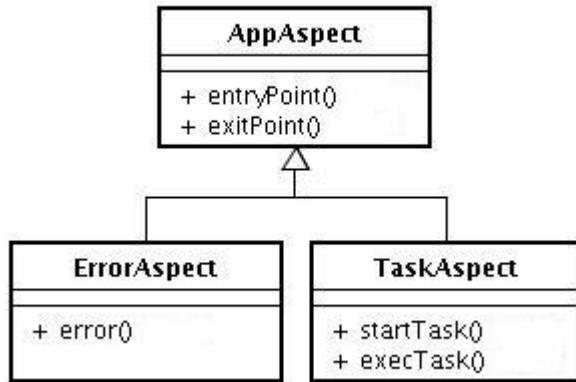


Figura 7.2: Ierarhia de aspecte corespunzătoare studiului de caz

```

abstract aspect ErrorAspect extends AppAspect{
    pointcut error(String msg):
        execution(* MesajEroare.show(String))
            && args(msg);
}

```

Tabela 7.3.1: *ErrorAspect*

tip, numărul sarcinilor efectuate cu succes, numărul sarcinilor abandionate și numărul sarcinilor eşuate. Cu toate că evenimentele de interes au fost scrise în fișierele jurnal pentru o posibilă analiză ulterioară detaliată, calculul metricilor s-a realizat în timpul interacțiunii utilizatorului cu sistemul, astfel încât la sfârșitul sesiunii de testare rezultatele sunt disponibile într-un fișier de tip text. Pentru a completa imaginea problemelor de utilizabilitate, la apariția oricărui eveniment care a fost jurnalizat s-a luat și o captură a interfeței utilizator la momentul respectiv. Codul sursă al aspectului *ErrorAspect* e prezentat în Tabela 7.3.1.

Codul sursă pentru jurnalizarea fiecărui eveniment este prezentat

```

aspect LogError extends ErrorAspect {
    private LogWriter lw;
    before() : entryPoint() {
        //initializes the log writer
        ...
    }
    before(String msg) : error(msg) {
        lw.log(msg,LogWriter.ERROR);
    }
    before() : exitPoint() {
        //closes the log writer
        ...
    }
}

```

Tabela 7.3.2: *.LogError*

în Tabela 7.3.2.

§. Utilizarea bibliotecilor destinate aplicațiilor asistive în jurnalizarea evenimentelor

O metodă alternativă de jurnalizare a informațiilor necesare evaluării utilizabilității aplicațiilor implementate folosind Java este utilizarea Java Accesibility API. Acest API este destinat dezvoltării de aplicații pentru persoane cu dizabilități. Pentru a scrie fișiere de jurnalizare folosind Java Accessibility API e necesară scrierea unei clase

care implementează toate interfețele asociate evenimentelor de care suntem interesați. Clasa *SwingEventMonitor* din Java Accessibility API va fi adăugată ca listener la toate evenimentele de interes. Mașina virtuală Java trebuie configurată înainte de rularea aplicației aflată în evaluare pentru a notifica clasa noastră asupra evenimentelor de interes. Această configurare trebuie făcută o singură dată și este disponibilă pentru toate aplicațiile aflate în evaluare. Aplicând această abordare aplicației prezentate în studiul de caz s-a realizat o comparație a avantajelor/dezavantajelor folosirii celor două abordări. Rezultatele sunt prezentate în Tabelele 7.3.3 și 7.3.4.

7.4 O abordare bazată pe agenți în evaluarea utilizabilității sistemelor informațice

O extindere a abordării evaluării utilizabilității folosind paradaigma de programare orientată pe aspecte apelează la utilizarea agenților (inteligienți) pentru identificarea problemelor de utilizabilitate. Dacă abordarea prezentată în secțiunea anterioară furniza informații canticitative relative la aspecte ale utilizabilității, abordarea prezentată în cele ce urmează încearcă o diagnosticare a problemelor de utilizabilitate prin identificarea acelor puncte din interacțiune în care utilizatorul întâmpină dificultăți. Un *agent* este o entitate care își percepse mediul prin intermediul unor senzori și acționează asupra mediului său prin acțiuni [60]. Un agent intelligent este un agent cu un set de cunoștințe inițiale care are capacitatea de a învăța. Una din sarcinile unui agent este aceea de a asista utilizatorul, de a realiza sarcini în locul utilizatorului sau de a-l învăța pe utilizator ce trebuie să facă.

Un agent este caracterizat prin:

Metrică	AOP	Java Accessibility API
Timpul de execuție	Ușor de calculat în timpul interacțiunii utilizator-sistem; se definește un pointcut pentru începutul execuției și un pointcut pentru sfârșitul execuției.	Necesită căutări în fișierele de jurnalizare (mari); se calculează doar după încheierea sesiunii de testare.
Identificarea erorilor	Fiecare eroare e tratată individual în timpul interacțiunii.	Pot fi identificate numai din context (ordinea evenimentelor).
Starea execuției sarcinilor	Se determină automat în timpul execuției.	Se determină după execuție; se poate automatiza dacă e disponibil un model al sarcinilor.
Numărul erorilor	Se calculează automat prin scrierea de pointcut-uri pentru situațiile tratate dar și pentru excepții neașteptate.	Se pot identifica doar situațiile de excepție tratate (prin căutarea evenimentelor asociate controalelor de afișare a mesajelor de eroare).

Tabela 7.3.3: Comparație între jurnalizarea folosind AOP și Java Accessibility API

- arhitectura (comportament) - acțiunile efectuate după orice secvență;
- program - funcționalitatea internă a agentului.

Evaluarea utilizabilității pornește în demersul propus de la compararea modelului sarcinilor construit de utilizator (**UTT**) cu modelul sarcinilor aparținând proiectantului sistemului (**TT**). În situația ideală, aceste modele sunt identice, caz în care nu vor apărea probleme de utilizabilitate majore. În majoritatea cazurilor însă, există diferențe considerabile între modelul conceptual al proiectantului și modelul conceptual al utilizatorului. Trebuie să remarcăm faptul că modelul conceptual al utilizatorului se construiește pe baza interacțiunii cu sistemul și pe baza experiențelor de operare cu alte sisteme. În abordarea aceasta, modelul conceptual al proiectantului se concretizează într-un arbore al sarcinilor descris cu ajutorul unui instrument de analiza sarcinilor și salvat într-un fișier XML. Rolul agentului este de a monitoriza interacțiunea utilizatorului cu sistemul și de a reconstrui modelul sarcinilor aparținând utilizatorului. Agentul **TAMO** este situat astfel într-un mediu soft. Acesta folosește AOP pentru a culege informații despre mediul său, în modul descris anterior în acest capitol. În Figura 7.3 se prezintă arhitectura generală a unui sistem bazat pe agenți folosit în evaluarea utilizabilității unui sistem informatic **SA**.

Componentele sistemului sunt descrise în cele ce urmează:

- *Aplicația soft (SA)*- aceasta e un sistem interactiv căruia i se evaluatează utilizabilitatea. Aplicația soft are asociat un *arbore al sarcinilor (TT)* dezvoltat de către proiectantul aplicației;
- *Utilizatorul (U)* - o persoană care folosește **SA** pentru a realiza un set stabilit de sarcini (un scenariu);
- *Agentul (TAMO)* - este un agent soft care are rolul de a monitoriza acțiunile utilizatorului și de a compara acțiunile efectuate de

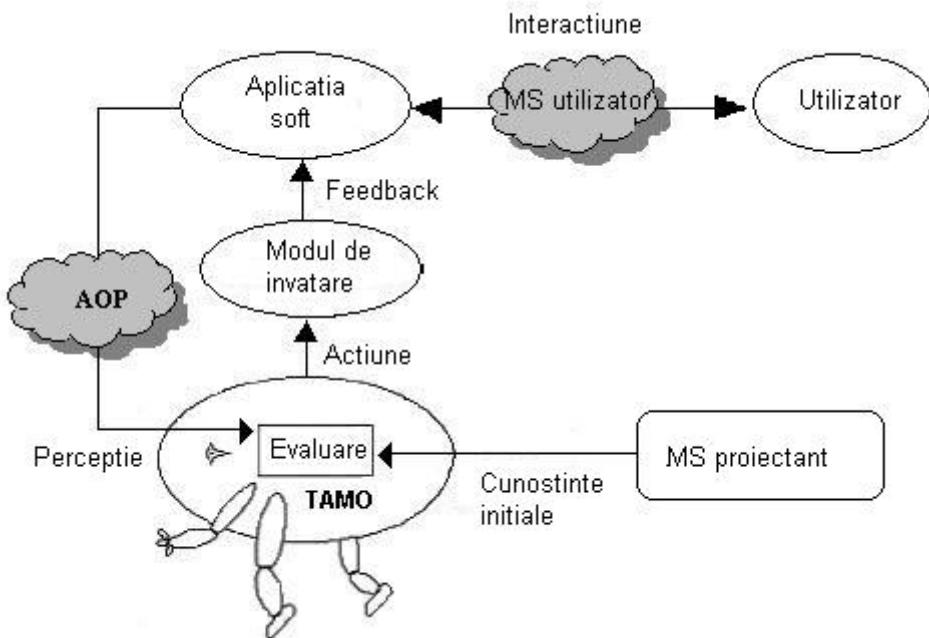


Figura 7.3: Arhitectura sistemului

utilizator cu **TT** pentru a-l asista pe utilizatorul **U**. Cunoștințele inițiale ale agentului constau în arborele sarcinilor **TT** asociat aplicației interactive **SA**. *Percepția* agentului **TAMO** se constituie din acțiunile pe care utilizatorul **U** le efectuează în timp ce folosește **SA**. Partea de program a agentului **TAMO** trebuie să stabilească dacă **UTT** este un subarbore al lui **TT**. În cazul general, *acțiunea* agentului constă în transmiterea rezultatului comparării *modulu-lui de învățare*. **TAMO** poate fi îmbunătățit prin înzestrarea lui cu capacitatea de învățare, situație în care el devine un agent de interfață [41], un fel de asistent personal al utilizatorului;

- Modulul de învățare (**LM**) - este un modul soft care învață comportamentul utilizatorului și transmite feedback evaluatorului interfeței utilizator. Aceasta se poate concretiza într-un agent inte-

ligență.

Modulul AOP este folosit pentru a capta evenimentele utilizator care sunt folosite de **TAMO** pentru a reconstrui modelul sarcinilor utilizator (**UTT**).

Problema de dezvoltare	AOP	Java Accessibility API	
Ușurința integrării	in-	Codul aplicației evaluate rămâne neschimbăt dar e nevoie de cunoștințe profunde legate de codul sursă.	Codul aplicației evaluate rămâne neschimbăt; nu e nevoie de cunoștințe legate de dezvoltarea produsului, dar la analiza fișierului de jurnalizare e necesară cunoașterea profundă a pașilor din execuția sarcinilor.
Captarea evenimentelor		Procesul de dezvoltare a aspectelor pentru captarea evenimentelor este dependent de aplicație. Dezvoltatorul trebuie să cunoască fiecare metodă care tratează evenimente pentru definirea pointcut-urilor.	Captarea evenimentelor se realizează automat de JVM. Dezvoltatorul trebuie să scrie o clasă pentru tratarea evenimentelor și JVM trebuie configurată. Clasa poate fi folosită pentru orice altă aplicație.
Dependența de platformă		Poate fi folosită pentru orice aplicație scrisă într-un limbaj care are asociată o extensie AOP (Java, C++, C#).	Poate fi folosită doar pentru aplicațiile Java dar poate fi adaptată pentru orice limbaj care oferă suport pentru accesibilitate.

Tabela 7.3.4: Comparație privind aspecte de dezvoltare

Capitolul 8

Asistenți personali ai utilizatorului

În trecut, utilizatorii sistemelor informaticice erau cei care le și concepeau, astfel încât nu existau probleme de utilizabilitate. În prezent, în orice domeniu al vieții economice sunt utilizate calculatoarele, iar nivelul de expertiză al utilizatorilor scade, o dată cu creșterea complexității sarcinilor. În acest context, este necesară identificarea de soluții care să ajute utilizatorii novici în îndeplinirea sarcinilor. Soluția descrisă în acest capitol constă în dezvoltarea unui agent intelligent, denumit **LIA** (Learning Interface Agent) pentru predicția comportamentului utilizatorului. Rolul acestui agent intelligent este acela de a învăța, în baza experienței anterioare și a contextului actual al interacțiunii, să prezică următoarea acțiune pe care ar trebui să o efectueze utilizatorul pentru a realiza sarcina. Aceasta este un prim pas spre realizarea unui asistent personal care să ghidizeze utilizatorul în realizarea sarcinilor. Câteva abordări privind predicția următoarei acțiuni utilizator sunt prezentate în cele ce urmează.

În [18, 17, 19] se prezintă o metodă de predicție simplă pentru determinarea următoarei comenzi utilizator dintr-o secvență de comenzi Unix. Abordarea se bazează pe presupunerea că fiecare comandă depinde

doar de comanda anterioară. O abordare similară este prezentată în [32], unde se ia în considerare și timpul dintre două comenzi.

Aplicarea tehniciilor de învățare automată (rețele neuronale și învățare inductivă) a mai fost folosită în analiza urmelor utilizator în [23], dar acestea sunt limitate la şabloane de lungime fixă.

În [74] se prezintă o abordare pentru predicția comportamentului utilizatorilor într-un site Web. Se folosește analiza fișierelor de loguri și pe baza acestora se încearcă determinarea următoarei pagini Web care va fi accesată la pasul următor. Predicția se face folosind o multime de antrenare formată din loguri utilizator, iar evaluarea se face folosind două măsuri. Spre deosebire de această abordare, în [61] se prezintă un model probabilistic de predicție, care presupune că întotdeauna se face o predicție.

8.1 Modelul teoretic

În cele ce urmează, se consideră că agentul **LIA** monitorizează interacțiunile utilizatorilor cu o aplicație interactivă \mathcal{SA} în timp ce realizează o sarcină \mathcal{T} . Vom nota prin \mathcal{A} multimea $\{a_1, a_2, \dots, a_n\}$ tuturor acțiunilor posibile care pot apărea în timpul interacțiunii cu \mathcal{SA} . O acțiune poate fi: apăsarea unui buton, selectarea unei opțiuni dintr-un meniu, completarea unui câmp text.

În timpul interacțiunii cu \mathcal{SA} pentru efectuarea sarcinii \mathcal{T} se generează *urme utilizator* ale interacțiunii. O *urmă utilizator* este o secvență de acțiuni utilizator. Vom spune că o urmă utilizator este *cu succes* dacă sarcina este îndeplinită. În celelalte cazuri avem o urmă utilizator *fără succes*.

În abordarea curentă se vor lua în considerare doar urmele cu succes, pentru care se va da o definiție formală.

DEFINIȚIE. *Urmă utilizator cu succes.*

Se consideră aplicația \mathcal{SA} și o sarcină prestabilită \mathcal{T} care poate fi realizată

folosind \mathcal{SA} . O secvență $t = \langle x_1, x_2, \dots, x_{k_t} \rangle$, unde

- $k_t \in \mathbf{N}$, și
- $x_j \in \mathcal{A}$, $\forall j, 1 \leq j \leq k_t$

care îndeplinește sarcina \mathcal{T} se numește **urmă utilizator cu succes**.

În Definiția 8.1, s-a notat prin k_t numărul de acțiuni utilizator din urma t . În continuare, se va nota prin \mathcal{ST} mulțimea tuturor *urmelor utilizator cu succes*.

Pentru a prezice comportamentul utilizatorului, agentul **LIA** păstrează o colecție de *urme utilizator cu succes* în timpul antrenării. Această colecție va forma *baza de cunoștințe* a agentului.

DEFINIȚIE. Baza de cunoștințe a agentului LIA - \mathcal{KB} .

Se consideră aplicația \mathcal{SA} și o sarcină \mathcal{T} care poate fi realizată folosind \mathcal{SA} . O colecție $\mathcal{KB} = \{t_1, t_2, \dots, t_m\}$ de urme utilizator cu succes, unde

- $t_i \in \mathcal{ST}$, $\forall i, 1 \leq i \leq m$,
- $t_i = \langle x_1^i, x_2^i, \dots, x_{k_i}^i \rangle$, $\forall x_j^i \in \mathcal{A}, 1 \leq j \leq k_i$

reprezintă **baza de cunoștințe** a agentului **LIA**.

Prin m se notează cardinalul bazei de cunoștințe \mathcal{KB} , iar k_i reprezintă numărul de acțiuni din urma t_i ($\forall i, 1 \leq i \leq m$).

DEFINIȚIE. Suburmă a unei urme utilizator Fie $t = \langle s_1, s_2, \dots, s_k \rangle$ o urmă din baza de cunoștințe \mathcal{KB} . Spunem că $sub_t(s_i, s_j) = \langle s_i, s_{i+1}, \dots, s_j \rangle$ ($i \leq j$) este o **suburmă** a urmei t începând cu acțiunea s_i și sfârșind cu acțiunea s_j .

În cele ce urmează se va nota prin $\|t\|$ numărul de acțiuni (sau lungimea) unei (sub)urme t . Menționăm că pentru două acțiuni date s_i și s_j ($i \neq j$) pot să existe mai multe suburme în urma t care încep cu s_i și se termină cu s_j . Vom nota prin $\mathcal{SUB}_t(s_i, s_j)$ mulțimea tuturor acestor suburme.

8.2 Comportamentul agentului LIA

Scopul urmărit este acela de a înzestra agentul **LIA** cu capacitatea de a prezice la un moment dat următoarea acțiune pe care utilizatorul ar trebui să o efectueze pentru a realiza sarcina \mathcal{T} .

Pentru a atinge acest scop, se propune folosirea unei tehnici de învățare supervizată care constă în doi pași:

1. Antrenarea

Pe parcursul acestui pas, agentul **LIA** monitorizează interacțiunea unui set de utilizatori reali în timp ce realizează sarcina \mathcal{T} , folosind aplicația \mathcal{SA} și își construiește baza de cunoștințe \mathcal{KB} (Definiția 8.1). Interacțiunea este monitorizată folosind AOP.

O abordare mai generală poate fi obținută prin construirea a două baze de cunoștințe în timpul antrenării: una cu urme utilizator cu succes, cealaltă cu urme utilizator fără succes.

2. Predicția

În acest pas se urmărește predicția comportamentului pentru un nou utilizator U pe baza datelor culese în timpul antrenării, folosind un model probabilistic. După fiecare acțiune act efectuată de un utilizator U , exceptând prima sa acțiune, agentul **LIA** va prezice următoarea acțiune, a_r ($1 \leq r \leq n$), care ar trebui efectuată cu o anumită probabilitate $P(act, a_r)$, folosindu-se de \mathcal{KB} .

Probabilitatea $P(act, a_r)$ este dată de Egalitatea (8.1).

$$P(act, a_r) = \max\{P(act, a_i), 1 \leq i \leq n\} \quad (8.1)$$

Pentru a calcula aceste probabilități se introduce conceptul de *scor* între două acțiuni. Scorul dintre acțiunile a_i și a_j , notat prin

$score(a_i, a_j)$ indică gradul în care a_j trebuie să-i urmeze lui a_i într-o execuție cu succes a sarcinii \mathcal{T} . Acest fapt sugerează că valoarea $score(a_i, a_j)$ este maximă atunci când a_j trebuie să urmeze imediat după a_i într-o execuție cu succes a sarcinii.

Scorul dintre o acțiune dată act și o acțiune utilizator a_q , $1 \leq q \leq n$, $score(act, a_q)$, se calculează ca în Egalitatea (8.2).

$$score(act, a_q) = \max \left\{ \frac{1}{dist(t_i, act, a_q)}, 1 \leq i \leq m \right\} \quad (8.2)$$

unde $dist(t_i, act, a_q)$ reprezintă distanța dintre două acțiuni act și a_q într-o urmă t_i , calculată folosind \mathcal{KB} .

$$dist(t_i, act, a_q) = \begin{cases} length(t_i, act, a_q) - 1 & \text{dacă } \exists sub_{t_i}(act, a_q) \\ \infty & \text{altfel} \end{cases} \quad (8.3)$$

$length(t_i, act, a_q)$ definește distanța minimă dintre act și a_q în urma t_i .

$$length(t_i, act, a_q) = \min \{ \|s\| \mid s \in \mathcal{SUB}_{t_i}(act, a_q) \} \quad (8.4)$$

$length(t_i, act, a_q)$ reprezintă numărul minim de acțiuni efectuate de utilizatorul U în urma t_i , pentru a ajunge de la acțiunea act la acțiunea a_q , adică, lungimea minimă dintre toate suburmele posibile $sub_{t_i}(act, a_q)$.

Din Egalitatea (8.2), rezultă că $score(act, a_q) \in [0, 1]$ și că valoarea lui $score(act, a_q)$ crește odată cu descreșterea distanței dintre act și a_q în urmele din \mathcal{KB} .

Pe baza scorurilor, $P(act, a_i)$, $\forall 1 \leq i \leq n$, se calculează după cum urmează:

$$P(act, a_i) = \frac{score(act, a_i)}{\max \{ score(a_i, a_j) \mid 1 \leq j \leq n \}} \quad (8.5)$$

Pe baza Ecuației (8.5), probabilități mai mari sunt asociate acțiunilor cele mai potrivite pentru a fi executate.

Rezultatul predicției agentului este acțiunea a_r care satisfacă Egalitatea (8.1). Menționăm că într-un caz nedeterministic (când mai multe acțiuni au aceeași probabilitate maximă P) se poate folosi o metodă de selecție suplimentară.

8.3 Arhitectura agentului LIA

În Figura 8.1 prezentăm arhitectura agentului **LIA**.

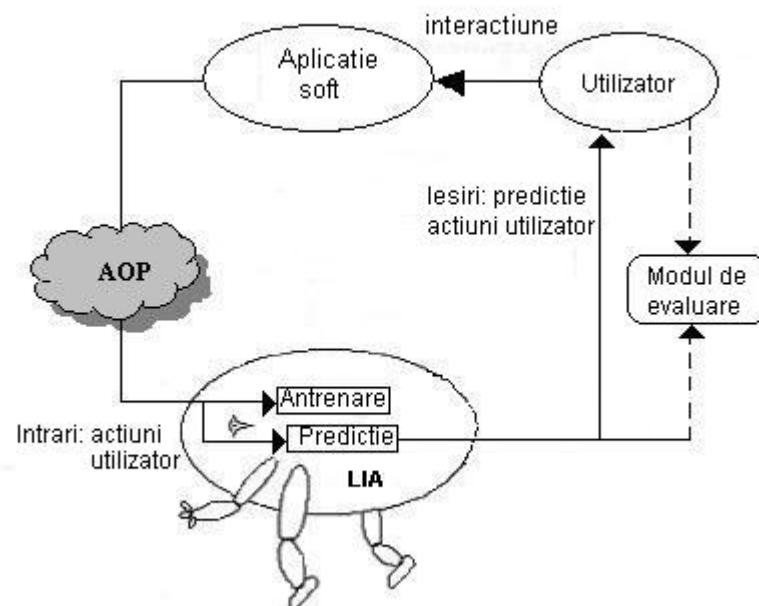


Figura 8.1: Arhitectura agentului **LIA**

Predicțiile agentului **LIA** sunt transmise unui *Modul de Evaluare* care stabilește acuratețea predicțiilor agentului. Prin transmiterea acestor rezultate direct utilizatorului agentul se transformă într-un asistent personal.

Agentul folosește AOP pentru a culege informații despre me-

diul său. Modulul AOP este folosit pentru a capta acțiunile utilizator: acționări ale mouse-ului, introducerea de text, selectarea de acțiuni. Aceste acțiuni sunt receptate de agentul **LIA** și sunt folosite atât în pasul de antrenare (pentru construirea bazei de cunoștințe \mathcal{KB}) cât și în pasul de predicție (pentru a determina cea mai probabilă acțiune utilizator următoare).

Decizia de a folosi AOP în dezvoltarea agentului aduce următoarele avantaje:

- separare clară între aplicație \mathcal{SA} și agent;
- agentul poate fi adaptat ușor și integrat altor sisteme soft;
- aplicația \mathcal{SA} nu trebuie modificată pentru a culege acțiunile utilizatorului;
- codul sursă folosit culegerii acțiunilor utilizator este grupat într-un singur loc - *aspectul*;
- dacă e nevoie de preluarea de noi informații de interacțiune, doar aspectul trebuie modificat.

8.4 Evaluare experimentală

Pentru a evalua acuratețea predicției agentului **LIA**, se compară secvența de acțiuni efectuate de utilizatorul U care duc la îndeplinirea sarcinii cu secvența de acțiuni prezise de agent. Se consideră că predicția unei acțiuni este precisă dacă probabilitatea predicției este mai mare decât un prag fixat.

În acest scop, se definește o măsură de calitate, $ACC(\mathbf{LIA}, U)$, denumită *ACCuracy* (precizie).

8.4.1 Măsura de evaluare

Se consideră că antrenarea agentului **LIA** a fost finalizată. Se dorește evaluarea preciziei predicțiilor agentului în timpul interacțiunii dintre un utilizator U și aplicația \mathcal{SA} . Se consideră că urma utilizator este $t_U = \langle y_1^U, y_2^U, \dots, y_{k_U}^U \rangle$, iar urma generată de predicția agentului este: $t_{LIA}(t_U) = \langle z_2^U, \dots, z_{k_U}^U \rangle$. Pentru fiecare $2 \leq j \leq k_U$, agentul **LIA** prezice cea mai probabilă acțiune utilizator următoare, z_j^U , cu probabilitatea $P(y_{j-1}^U, z_j^U)$.

Următoarea definiție descrie modul de evaluare a preciziei agentului **LIA** relativă la urma utilizator t_U .

DEFINIȚIE. Acuratețea predicției agentului LIA - ACC.

Acuratețea (precizia) predicției în raport cu urma utilizator t_U este dată de Egalitatea (8.6).

$$ACC(t_U) = \frac{\sum_{j=2}^{k_U} acc(z_j^U, y_j^U)}{k_U - 1}, \quad (8.6)$$

unde

$$acc(z_j^U, y_j^U) = \begin{cases} 1 & \text{dacă } z_j^U = y_j^U \text{ și } P(y_{j-1}^U, z_j^U) > \alpha \\ 0 & \text{altfel} \end{cases}. \quad (8.7)$$

$acc(z_j^U, y_j^U)$ arată dacă predicția z_j^U a fost făcută cu o probabilitate mai mare decât un prag α , în raport cu acțiunea utilizator y_j^U . Prin urmare, $ACC(t_U)$ estimează precizia globală a predicțiilor agentului relativ la urma utilizator t_U .

Lema 8.4.1. $ACC(t_U)$ ia valori în intervalul $[0,1]$.

Demonstrație. Pe baza Ecuațiilor 8.7 și 8.6 rezultă că $acc(z_j^U, y_j^U)$ poate lua valorile 0 sau 1. Prin urmare, valoarea minimă a $\sum_{j=2}^{k_U} acc(z_j^U, y_j^U)$ este

0, iar valoarea maximă este $k_U - 1$. Astfel, valorile pentru $ACC(t_U)$ se situează în intervalul $[0,1]$. \square

OBSERVAȚIE. Valori mai mari pentru ACC indică predicții mai bune.

Măsura acurateții poate fi extinsă pentru a ilustra precizia predicției agentului **LIA** pentru mai mulți utilizatori, aşa cum se prezintă în Definiția 8.4.1.

DEFINIȚIE. Acuratețea predicției agentului LIA pentru mai mulți utilizatori - $ACCM$.

Se consideră o mulțime de utilizatori, $\mathcal{U} = \{U_1, \dots, U_l\}$. Se notează prin $\mathcal{UT} = \{t_{U_1}, t_{U_2}, \dots, t_{U_l}\}$ mulțimea urmatorilor utilizatorilor din \mathcal{U} . Acuratețea predicției în raport cu utilizatorul U_i și urma asociată acestuia/acesteia t_{U_i} este dată de Egalitatea (8.8).

$$ACCM(\mathcal{UT}) = \frac{\sum_{i=1}^l ACC(t_{U_i})}{l}. \quad (8.8)$$

unde $ACC(t_{U_i})$ este acuratețea predicției pentru urma utilizator t_{U_i} dată în Egalitatea (8.6).

Lema 8.4.2. $ACCM(\mathcal{UT})$ ia valori în intervalul $[0,1]$.

Demonstrația este similară demonstrației Lemei 8.4.1.

8.4.2 Studiu de caz

Evaluarea predicției agentului **LIA** s-a realizat pe un studiu de caz, folosind măsurile de calitate anterior definite. Pentru studiul de caz efectuat mulțimea acțiunilor utilizator posibile \mathcal{A} conține în jur de **50** elemente ($n \approx 50$). Agentul **LIA** a fost antrenat pe diferite mulțimi de antrenare și s-au evaluat rezultatele pentru diferiți utilizatori.

Rezultate

Pentru evaluare s-a ales valoarea pragului α egală cu **0.75**.

Pentru fiecare pereche (multime de antrenare, multime de test) s-a calculat valoarea măsurii ACC conform Ecuației (8.6). În Tabela 8.4.1 sunt prezentate rezultatele studiului de caz. S-au ales **20** urme utilizator în multimea de antrenare, iar valorile acurateței s-au situat în jurul valorii de **0.96**.

Dimensiunea setului de antrenare	ACCM
67	0.987142
63	0.987142
60	0.987142
50	0.982857
42	0.96

Tabela 8.4.1: Rezultatele studiului de caz

După cum se poate observa în Tabela 8.4.1, acuratețea predicției crește odată cu creșterea dimensiunii setului de antrenare.

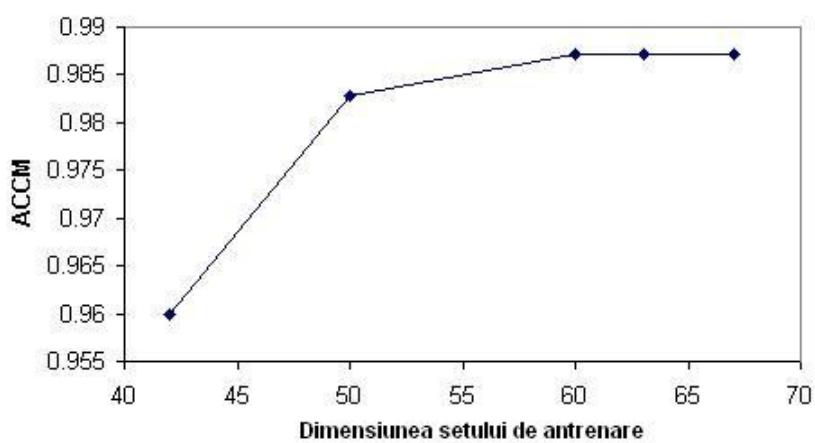


Figura 8.2: Influența dimensiunii setului de antrenare asupra acurateții predicției

Bibliografie

- [1] ***. Abstract User Interface Markup Language.
<http://xml.coverpages.org/userInterfaceXML.html#auiml>.
- [2] ***. Alternate Abstract Interface Markup Language.
<http://xml.coverpages.org/userInterfaceXML.html#aaiml>.
- [3] ***. eXtensible Interface Markup Language. <http://www.ximl.org/>.
- [4] ***. EXtensible User interface Language.
<http://xml.coverpages.org/userInterfaceXML.html#xul>.
- [5] ***. Microsoft eXtensible Application Markup Language.
msdn2.microsoft.com.
- [6] M. Abrams and J. Helm. User Interface Markup Language UIML specification, Language Version 3.0, 2002.
- [7] M. Abrams and C. Phanouriou. UIML: An XML language for building device-independent user interfaces, 1999.
- [8] J. Annett and K. D. Duncan. Task analysis and training design. journal of occupational psychology. *Journal of Occupational Psychology*, 41:211–221, 1967.
- [9] AspectJ Project. <http://eclipse.org/aspectj/>.

- [10] J. W. Backus, J. H. Wegstein, A. van Wijngaarden, M. Woodger, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, and B. Vauquois. Report on the Algorithmic Language ALGOL 60. In *Communications of the ACM, Peter Naur (Ed.)*, pages 299–314, 1960.
- [11] P. Booth. *An Introduction to Human-Computer Interaction*. Lawrence Erlbaum Associates, 1989.
- [12] P.J. Branard, T. R.G. Green, A. MacLean, and M. D. Wilson. *Working with Computers: theory versus outcome.*, chapter Knowledge-Based Task Analysis for Human-Computer Systems, pages 47–87. Academic Press, London, 1988.
- [13] S. Card, T. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Cariere. Lawrence Erlbaum Associates, 1983.
- [14] S. K. Card and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, New Jersey, 1983.
- [15] S.K. Card, T. P. Moran, and A. Newell. The keystroke-level model for user performance time with interactive systems. *Commun. ACM*, 23(7):396–410, 1980.
- [16] J. P. Chin, V. A. Diehl, and K. L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–218, New York, NY, USA, 1988. ACM Press.
- [17] B. D. Davison and H. Hirsh. Experiments in UNIX Command Prediction. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, page 827, Providence, RI, 1997. AAAI Press.

- [18] B. D. Davison and H. Hirsh. Toward an Adaptive Command Line Interface. In *Proceedings of the Seventh International Conference on Human Computer Interaction*, pages 505–508, 1997.
- [19] B.D. Davison and H. Hirsh. Predicting Sequences of User Actions. In *Predicting the Future: AI Approaches to Time-Series Problems*, pages 5–12, Madison, WI, July 1998. AAAI Press. Proceedings of AAAI-98/ICML-98 Workshop, published as Technical Report WS-98-07.
- [20] G. de Haan. *ETAG A Formal Model of Competence Knowledge for User Interface Design*. PhD thesis, Vrije Universiteit Amsterdam, 2000.
- [21] A. Dix and al. *Human-Computer Interaction*. Prentice Hall, 1993.
- [22] A. Dix, J. Finlay, G. Abowd, and B. Russell. *Human-Computer Interaction*. Prentice Hall, 1993.
- [23] A. Dix, J. Finlay, and B. Russell. Analysis of User Behaviour as Time Series. In *Proceedings of HCI'92: People and Computers VII*, pages 429–444. Cambridge University Press., 1992.
- [24] J. Fierstone, A.-M. Pinna-Dery, and M. Riveill. Architecture logicielle pour l'adaptation et la composition d'ihm - mise en oeuvre avec le langage SUNML., 2003.
- [25] M. Green. Report on dialogue specification tools. In *User Interface Management Systems*, pages 9–20. Springer-Verlag, 1985.
- [26] D. Harel. Statecharts: A visual formulation for complex systems. *Sci. Comput. Program.*, 8(3):231–274, 1987.
- [27] H. Hartson, A. Siochi, and D. Hix. The UAN: A user-oriented representation for direct manipulation interface designs. *ACM Transactions on Information Systems*, 8(3):181–203, 1990.

- [28] ISO/IEC. Information Processing Systems Open Systems Interconnection:. *LOTOS, a Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*. IS 8807, 1989.
- [29] G. Iosif and A. Marhan. *Introducere în interacțiunea om-calculator*, chapter Analiza sarcinii, pages 97–116. Matrix Rom, București, 2003.
- [30] ISO. ISO9214-11 Ergonomic Requirements for office Work with VDT's - Guidance on Usability, 1991.
- [31] Melody Y. Ivory and Marti A Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Computings Surveys*, 33(4):470–516, 2001.
- [32] N. Jacobs and H. Blockeel. Sequence Prediction with Mixed Order Markov Chains. In *Proceedings of the Belgian/Dutch Conference on Artificial Intelligence*, 2003.
- [33] Java homepage. <http://java.sun.com/>.
- [34] H. Johnson and P. Johnson. Integrating task analysis into system design. *Ergonomics*, 32(11):1451–1467, 1990.
- [35] B. Jordan. *The Design of Computer-Supported Cooperative Work and Groupware Systems*, chapter Ethnographic Workplace Studies and Computer Supported Cooperative Work, pages 17–42. North Holland/Elsevier Science, Amsterdam, 1996.
- [36] T. Jucan and F. L. Tiplea. *Rețele Petri. Teorie și practică*. Editura Academiei Române, 1999.
- [37] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In *Proceedings European Conference on Object-Oriented Programming*, volume 1241, pages 220–242. Springer-Verlag, 1997.

- [38] D. Kieras and P. G. Polson. An approach to the formal analysis of user complexity. *Int. Journal Human-Computer Study*, 51(2):405–434, 1999.
- [39] C. Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. Prentice Hall, 1997.
- [40] J. Leplat. *Ergonomie cognitivă: teorii, modele, aplicații*, chapter Analiza psihologică a activității în situație de muncă, pages 16–43. Matrix Rom, București, 2004.
- [41] P. Maes. Social Interface Agents: Acquiring Competence by Learning from Users and Other Agents. In O. Etzioni, editor, *Software Agents — Papers from the 1994 Spring Symposium (Technical Report SS-94-03)*, pages 71–78. AAAI Press, 1994.
- [42] F. Mir, M. Pérez-Quiñones, and M. Abrams. A multi-step process for generating multi-platform user interfaces using UIML, 2001.
- [43] T. P. Moran. The command language grammar: a representation for the user interface of interactive computer systems. *Man-Machine Studies*, 15:3–50, 1981.
- [44] G. Mori, F. Paternò, and C. Santoro. CTTE: Support for developing and analyzing task models for interactive system design. *IEEE Transactions on Software Engineering*, 28(9):1–17, 2002.
- [45] J. Nielsen. Ten usability heuristics.
- [46] NIST. National Institute for Standardization and Technology. <http://www.nist.gov/dads/HTML/tree.html>.
- [47] P. Palanque and R. Bastide. Petri net based design of user-driven interfaces using the cooperative object formalism. In F. Paternó, editor, *Design, Specification and Verification of Interactive Systems '94*, pages 383–400, Heidelberg, 1994. Springer-Verlag.

- [48] P. A. Palanque, R. Bastide, L. Dourte, and C. Sibertin-Blanc. Design of user-driven interfaces using Petri nets and objects. *Lecture Notes in Computer Science*, 685:569–585, 1993.
- [49] F. Paternò. Model-based Tools for Pervasive Usability. *Interacting with Computers*, 2004.
- [50] F. Paternò, C. Mancini, and S. Meniconi. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In *INTERACT '97: Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*, pages 362–369. Chapman & Hall, Ltd., 1997.
- [51] S.J. Payne. Task Action Grammar. In Bullinger H. J. and Shackel B., editors, *Proceedings INTERACT'84*, pages 139–144, North-Holland, 1984.
- [52] A. M. Pinna-Dery, J. Fierstone, and E. Picard. *The 5th International Symposium on Human-Computer Interaction with Mobile Devices and Services, number 2795 in Lecture Notes in Computer Science*, chapter Component model and programming: a first step to manage human computer interaction adaptation., pages 456–460. Springer Verlag, Udine, Italy, 2003.
- [53] H. D. Pitariu. *Proiectarea fișelor de post, evaluarea posturilor de muncă și a personalului*. Cariere. Casa de editură IRECSION, 2003.
- [54] Mario Ponzo. Intorno ad alcune illusioni nel campo delle sensazioni tattili, sull'illusione di aristotele e fenomeni analoghi. *Archiv für die gesamte Psychologie*, 16:307–345, 1910.
- [55] J. Preece, D. Benyon, G. Davies, G. Keller, and Y. Rogers. A Guide to Usability, 1990.

- [56] ACM Press, editor. *Proceedings ACM CHI 99*, Pittsburgh, May 1999.
- [57] C. Pribeanu. *Calitatea sistemelor interactive*, chapter Utilizabilitatea sistemelor interactive, pages 69–89. Matrix Rom, Bucureşti, 2004.
- [58] L. Ramnivas. *AspectJ in Action*. Manning Publications Co., 2003.
- [59] Jeffrey Rubin. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. John Wiley & Sons, Inc, 1994.
- [60] S. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice-Hall, Inc., 1995.
- [61] G. Serban, **A. M. Tarță**, and G. S. Moldovan. A Learning Interface Agent for User Behavior Prediction. In *Proceedings of Human Computer Interaction International - HCII 2007*, pages 508–517, Beijing, China, 2007. Springer -Verlag.
- [62] D. Shakes. *Usability - Context, Framework, Definition, Design and Evaluation*. Cambridge University Press, 1991.
- [63] C. Sibertin-Blanc. The object-oriented approach and petri nets for the modelling of information systems' behaviour.
- [64] B. Sufrin and R. Bornat. Animating Operational Semantics with JAPE. citeseer.ist.psu.edu/485702.html.
- [65] SUMI. SUMI Software Usability Measurement Inventory, 1996. <http://www.ucc.ie/hfrg/questionnaires/sumi/>.
- [66] A. M. Tarță. Task modeling in systems design. *Studia Universitatis Babes-Bolyai. Informatica*, 2:37–44, 2004.

- [67] A. M. Tarță and S. C. Motogna. Operational semantics of task models. *Studia Universitatis Babes-Bolyai. Informatica*, 2(LI):81–92, 2006.
- [68] A. M. Tarță, D. M. Onacă, and H. D. Pitariu. The development of a theatre /opera website pattern based on a user need assessment approach. *Psihologia Resurselor Umane - Revista Asociației de Psihologie Industrială și Organizațională*, 1(1):35–48, 2006.
- [69] A. M. Tarță, D. M. Onacă, and H. D. Pitariu. Dezvoltarea unui şablon de proiectare siturilor web ale teatrelor. *Informatica Economică/Economy Informatics*, X(1):121–128, 2006.
- [70] A. M. Tarță and H. D. Pitariu. Proiectarea siturilor web pentru utilizabilitate folosind şabloane de proiectare. In *Conferința Națională de Interacțiune Om-Calculator RoCHI 2005*, pages 173–175, Cluj-Napoca, România, 2005.
- [71] M.J. Tauber. *Working with Computers: theory versus outcome.*, chapter On Mental Models and the User Interface., pages 89–119. Academic Press, London, 1988.
- [72] **A. M. Tarță**. Developing dialog models using GTATool. In *International Conference on Mathematics & Informatics, ICMI45*, pages 589–600, Bacău, România, 2006.
- [73] **A. M. Tarță**, G. S. Moldovan, and G. Șerban. An Agent Based User Interface Evaluation Using Aspect Oriented Programming Techniques. In *ICAM5 - Fifth International Conference on Applied Mathematics*, pages 151–158, Baia-Mare, România, 2006.
- [74] B. Trousse. Evaluation of the Prediction Capability of a User Behaviour Mining Approach for Adaptive Web Sites. In *Proceedings of the 6th RIAO Conference — Content-Based Multimedia Information Access, Paris, France*, 2000.

- [75] G. van der Veer, M. Hoeve, and B. Lenting. Modeling Complex Work Systems - Method meets Reality. In *Cognition and the work system*, 1996.
- [76] G. van der Veer and M. van Welie. Task based groupware design: putting theory into practice. In *DIS '00: Proceedings of the conference on Designing interactive systems*, pages 326–337. ACM Press, 2000.
- [77] R. van Loo, G. van der Veer, and M. van Welie. Groupware Task Analysis in practice: a scientific approach meets security problems. In *7th European Conference on Cognitive Science Approaches to Process Control*, 1999.
- [78] M. van Welie. *Task-based User Interface Design*. PhD thesis, Vrije Universiteit Amsterdam, 2001.
- [79] M. van Welie and H. Troetteberg. Interaction Patterns in User Interfaces. In *7th. Pattern Languages of Programs Conference*, pages 213–218. <http://www.welie.com/about.html>, 2000.
- [80] M. van Welie, G. van der Veer, and A. Koster. Integrated Representations for Task Modeling. In *Tenth European Conference on Cognitive Ergonomics*, pages 129–138, 21-23 August 2000.
- [81] M. van Welie and G.C. van der Veer. Structured methods and creativity: a happy Dutch marriage. In *Co-Designing 2000*, 2000.
- [82] W. E. Woodson. *Human Factors Design Handbook: Information and Guidelines for the Design of Systems, Facilities, Equipment, and Products for Human Use*. McGraw-Hill, 1981.