Academia Română
Secţia Ştiinţa şi Tehnologia Informaţiei
Institutul de Cercetări pentru Inteligenţa Artificială

Referat II

Arhitectura unei interfeţe avansate pentru un Sistem Suport pentru Decizii

Coordonator ştiinţific: Acad. prof. dr. ing. Florin G. FILIP

Doctorand: Ana-Maria SUDUC

- 2009 -

Cuprins

Introducere

Interacţiunea om-calculator (HCI-Human Computer Interaction) este definită în sens larg ca fiind ştiinţa care studiază oamenii şi tehnologiile computaţionale şi modul în care cele două componente se influenţează reciproc (Dix şi alţii, 2004).

Utilizatorii sistemelor informatice doresc aplicaţii interactive avansate care să fie uşor de utilizat şi uşor de învăţat fără să fie nevoiţi să citească numeroase manuale. Din nefericire astfel de interfeţe sunt dificil de proiectat şi implementat. Myers, în 1992, observa, în urma unor studii, că 48% din codul unei aplicaţii este destinat interfeţei cu utilizatorul şi în jur de 50% din timpul de implementare este folosit pentru implementarea interfeţei. Cu cât interfeţele devin mai uşor de utilizat, cu atât devin mai dificil de creat. Realizatorii de interfeţe au nevoie de unelte care să îi ajute în dezvoltarea interfeţelor avansate cu utilizatorul (Schlungbaum, 1996).

Constantine şi Lockwood (1999) prezintă o colecţie de principii pentru îmbunătăţirea calităţii proiectării unui interfeţe cu utilizatorul:

- *Principiul structurii*. Interfaţa cu utilizatorul trebuie organizată într-un mod util, cu un scop clar şi semnificativ bazată pe modele consistente care sunt cunoscute de utilizatori, grupând componentele care au legătură între ele şi separându-le pe cele diferite, diferenţiind elementele neasemănătoare şi prezentându-le asemănător pe cele similare. Principiul structurii se referă la arhitectura generală a interfeţei cu utilizatorul.
- *Principiul simplicităţii*. Interfaţa cu utilizatorul trebuie să asigure o realizare uşoară a funcţiilor uzuale, prezentând o comunicare clară şi simplă "pe limba" utilizatorului. De asemenea trebuie să furnizeze căi rapide de realizare a unor proceduri lungi.
- *Principiul vizibilităţii*. Interfaţa trebuie să prezinte grupat toate opţiunile şi materialele necesare pentru o anumită sarcină fără a distrage atenţia utilizatorului cu informaţii externe sau redundante. Interfeţele bine proiectate nu încarcă utilizatorul cu prea multe alternative sau prea multe informaţii.
- *Principiul feedbackului*. Interfaţa trebuie să informeze utilizatorul de acţiunile sau interpretările, modificările de stare sau condiţii şi de erorile sau excepţiile

care sunt relevante sau de interes pentru utilizator într-un limbaj clar, concis şi familiar utilizatorului.

- *Principiul toleranţei*. Interfaţa trebuie să fie flexibilă şi tolerantă şi să reducă costul greşelilor şi al utilizărilor necorespunzătoare prin furnizarea posibilităţii de a anula comenzile sau de a le reface şi, în acelaşi timp, să prevină, pe cât posibil, erorile tolerând variate intrări şi secvenţe şi interpretând rezonabil toate acţiunile care sunt rezonabile.
- *Principiul reutilizabilităţii*. Interfaţa trebuie să refolosească componente şi comportamente interne şi externe menţinând consistenţa cu scopul reducerii nevoii utilizatorului de a regândi şi a-şi reaminti.

1. Generalităţi privind interfeţele cu utilizatorul

1.1. Direcţii de proiectare

În proiectarea interfeţelor cu utilizatorul există numeroase abordări, clasificate după diferite criterii. În funcţie de aspectul principal luat în considerare în dezvoltarea interfeţei există două direcţii de proiectare foarte utilizate: proiectarea centrată pe sarcini şi proiectarea centrată pe utilizator.

Proiectarea centrată pe sarcini (task centred interface design) e structurată în jurul unui set de sarcini specifice pe care utilizatorul trebuie să le îndeplinească cu sistemul respectiv. Aceste sarcini sunt selectate încă de la începutul etapei de proiectare şi apoi sunt folosite pentru stabilirea diferitelor detalii legate de proiectare, pentru a ajuta în luarea deciziilor de proiectare precum şi pentru evaluarea proiectării pe măsura dezvoltării ei. Dezavantajele acestei metode faţă de o abordare centrată pe utilizator sunt după cum urmează:

- Software-ul este proiectat conform scopurilor producătorului, nu ale utilizatorului;
- Atenţia este direcţionată spre tehnologie şi uşurinţa implementării, nu spre eficienţa utilizatorului (ergonomia acţiunilor lui);
- Software-ul este focalizat pe caracteristici (feature-centric software), unele niciodată utilizate;
- Software-ul devine mamut, incontrolabil.

Proiectarea centrată pe utilizator (user-centred interface design) se bazează pe înţelegerea domeniului de muncă a utilizatorilor finali vizaţi şi a modului în care aceştia interacţionează cu calculatorul şi urmăreşte să faciliteze acţiunile oamenilor. În cazul unei astfel de abordări:

- Software-ul trebuie să mulţumească, să menajeze, să ajute utilizatorul, proiectarea bazându-se pe abilităţile şi nevoile reale ale utilizatorului, pe context, funcţii, sarcini;
- Proiectarea se realizează având în vedere cerinţele utilizatorului, nu cele ale producătorului;
- Nu trebuie să se utilizeze un limbaj obscur, neînţeles de utilizator;

- Software-ul în prezent abuzează de elementele grafice ale interfeţei (meniuri, ferestre, icon-uri, etc.)

În cazul adoptării acestei direcţii de proiectare, probleme ar putea apărea dacă proiectantul nu reuşeşte să cunoască bine utilizatorul, nevoile acestuia. Astfel de probleme pot fi rezolvate prin implicarea în proiectare a unui grup de utilizatori reprezentativi.

1.2. Stiluri de interacţiune om-calculator

În realizarea modelului conceptual a unei interfeţe cu utilizatorul, este utilă determinarea tipurilor de interacţiuni ce vor fi utilizate şi de ce. Există patru *moduri de interacţiune* (Sharp, 2006):

- *Instructaj* – emiterea de comenzi fie prin tastarea de comenzi, selectarea unor opţiuni din meniurile unei ferestre sau prin intermediul unui touch screen, emiterea de comenzi vocale, prin apăsarea de butoane sau folosirea unor combinaţii de taste. Utilizatorul îi „spune” sistemului ce să facă. Interacţiunea este rapidă şi eficientă.
- *Conversaţie* – interacţiune de tip dialog. Utilizatorul comunică cu interfaţa fie vocal, fie tastând întrebări la care sistemul răspunde textual sau vocal. Exemple: motoare de căutare, servicii telefonice, cumpărături virtuale;
- *Manipulare* – interacţiune directă cu obiectele dintr-un spaţiu virtual/fizic. Exploatează cunoştinţele utilizatorului în ceea ce priveşte mişcarea şi manipularea din lumea reală. Obiectele virtuale pot fi manipulate variat. Obiectele fizice manipulate pot declanşa evenimente concrete sau digitale. Prin manipulare directă, utilizatorii au feedback imediat asupra ceea ce fac, plus câştigă încredere. Novicii pot învăţa rapid funcţionalităţile de bază. Pot apărea probleme de eficienţă – nu toate activităţile pot fi realizate în mod direct.
- *Explorare* – evoluţie în cadrul unui mediu virtual sau spaţiu fizic. Este întâlnită mai ales în cazul mediilor 3D şi sistemele de realitate virtuală. Este folosită de jocurile interactive şi poate implica şi obiecte reale (de exemplu senzori la computerele “la purtător” - wearable computers). Această metodă de interacţiune

este folosită în mediul virtual Second Life[1]. Această metodă are succes deoarece presupune un mediu uşor de personalizat, modificat de utilizator.

Altele: învăţare, rezolvarea de probleme, socializare, căutare, etc.

Cele mai cunoscute şi utilizate *stiluri de interacţiune om-calculator* sunt: în linie de comandă, formulare interactive şi spreadsheet-uri, meniuri, interacţiune grafică, manipularea directă şi hipertextul şi limbajul natural.

În ultimii ani, au apărut noi clase de dispozitive pentru accesarea informaţiei odată cu creşterea conectivităţii. În paralel cu această dezvoltare, au fost explorate noi stiluri de interacţiune: realitate virtuală, realitate mixată (mixed reality), interacţiune 3D, interfeţe utilizator tangibile (tangible user interface), interfeţe conştiente de context (context-aware interfaces) şi interfeţe bazate pe recunoaştere (recognition based interfaces) (Shaer, 2008).

Interacţiunea 3D are loc atunci când utilizatorul poate muta sau realiza interacţiuni într-un spaţiu 3D. Acest tip de interacţiune presupune că utilizatorii îşi îndeplinesc sarcinile şi realizează acţiuni prin schimburi de informaţii cu sistemul într-un spaţiu 3D. Acest tip de interacţiune este unul intuitiv deoarece utilizatorul interacţionează în lumea reală într-un spaţiu tridimensional.

Realitatea virtuală (VR) reprezintă o tehnologie care permite utilizatorului să interacţioneze cu un mediu simulat pe calculator, fie el real sau imaginar. Majoritatea mediilor de realitate virtuală sunt în primul rând experienţe vizuale afişate pe monitorul calculatorului sau prin intermediul unor display-uri stereoscopice. Alte medii includ informaţii adiţionale senzoriale (de exemplu audio). Unele sisteme haptice avansate includ informaţii tactile, cunoscute sub termenul de "force feedback" (folosite în general în aplicaţii de medicină şi jocuri). Utilizatorii pot interacţiona cu mediul virtual fie prin dispozitive standard de intrare precum mouse-ul şi tastatura fie prin dispozitive multimodale (de exemplu mănuşa virtuală – wired glove, braţ de extensie – boom arm, benzi de alergare omnidirecţionale – omnidirectional treadmill). Mediul simulat poate fi similar cu lumea reală (simulări pentru antrenamente de zbor şi de luptă) sau poate fi foarte diferit de lumea reală (ca în jocurile VR).

[1] http://secondlife.com/

Realitatea mixată (mixed reality) reprezintă o îmbinare între lumea reală şi cea virtuală cu scopul de a produce noi medii şi vizualizări în care obiectele fizice şi cele digitale coexistă şi interacţionează în viaţa reală. Un exemplu îl reprezintă platforma creată în cadrul proiectului Binarium care reconstruieşte oraşul Sibiu în spaţiul virtual creând o interfaţă care este populată cu avataruri ale cetăţenilor (www.binarium.ro).

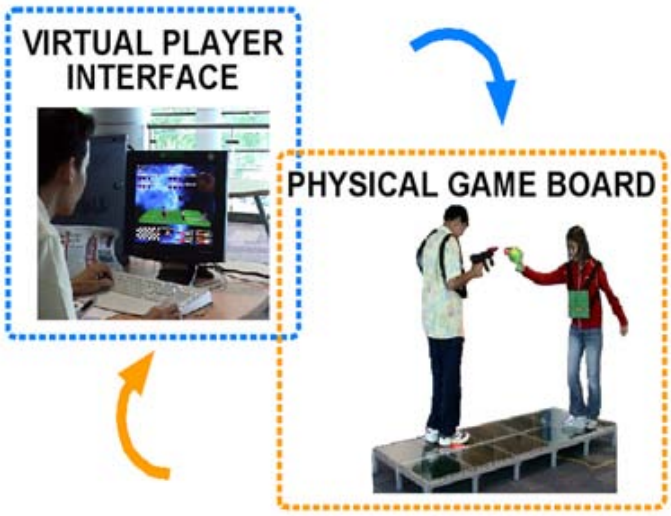

Fig. 1 Realitate mixată (preluare http://www.mixedrealitylab.org)

Interfeţele utilizator tangibile (Tangible User Interface - TUI), numite iniţial *Graspable User Interfaces* de către iniţiatorul acestor tipuri de interfeţe, Hiroshi Ishii, sunt interfeţe în care utilizatorul interacţionează cu informaţiile digitale (de exemplu elemente grafice şi audio) prin intermediul mediului fizic (de exemplu obiecte fizice manipulabile spaţial). Idea lui Ishii a fost de a da formă fizică informaţiei digitale, făcând astfel biţii direct manipulabili şi perceptibili, asociind astfel două lumi foarte diferite: cea a biţilor şi cea a atomilor.

Caracteristicile acestor tipuri de interfeţe sunt: (1) reprezentările fizice sunt asociate computaţional astfel încât să stea la baza informaţiei digitale; (2) reprezentările fizice includ mecanisme pentru controlul interactivităţii; (3) reprezentările fizice sunt asociate perceptual pentru a media activ reprezentări digitale; (4) starea fizică a elementelor fizice încorporează aspecte cheie ale stării digitale a sistemului.

Un exemplu de astfel de interfaţă e sistemul Topobo. Blocurile în Topobo sunt precum piesele lego. Utilizatorul le poate deplasa, roti, amesteca şi aceste blocuri memorează aceste mişcări şi furnizează răspunsuri.

1.3. Tipuri de interfeţe

În momentul de faţă cele mai folosite tipuri de interfeţe sunt cele grafice şi cele bazate pe web.

Interfeţele grafice (graphical user interface - GUI) acceptă intrări de la dispozitive precum tastatura sau mouse-ul şi furnizează ieşiri grafice pe monitorul calculatorului. O interfaţă grafică foloseşte atât imagini grafice cât şi text înlocuind multe funcţii ale tastaturii.

Interfeţele bazate pe web (web user interfaces - WUI) acceptă intrări şi furnizează ieşiri prin generarea de pagini web care sunt transmise via Internet şi vizualizate de utilizator cu ajutorul unui browser web. Implementările mai noi folosesc Java, Ajax, Abobe Flex, Microsoft .NET sau alte tehnologii similare pentru a asigura control în timp real într-un program separat, eliminând nevoia de "refresh" a browsere-lor web bazate pe HTML.

Alte tipuri de interfeţe sunt:

- *interfeţe în linie de comandă* (folosite pentru funcţii de administrare), interfeţe tactile (tactile interfaces) (folosite în simulatoarele computerizate, etc.), interfeţe care folosesc tuchscreen-uri (folosite în muzee, puncte de vânzare, procese industriale, etc.);
- *interfeţe de tip batch* – interfeţe non-interactive în care utilizatorul specifică la început toate detaliile sarcinii care trebuie îndeplinite şi primeşte rezultatul după ce procesarea s-a încheiat. Calculatorul nu solicită alte intrări după ce procesarea a început;
- *agenţi de interfaţă de conversaţie* – încearcă să personifice interfaţa cu calculatorul în forma unei persoane animate, a unui robot sau a altui caracter (exemplu: agrafa din Microsoft Office) şi prezintă interacţiuni într-o formă conversaţională;
- *gesture interface* – interfeţe care acceptă intrări sub forma unei mişcări a mâinii (gest) sau a mouse-ului, schiţat cu un mouse sau un stilou;
- *interfeţe cu utilizatorul inteligente* – sunt interfeţe care vizează să îmbunătăţească eficienţa, eficacitatea şi naturaleţea interacţiunii om-maşină prin reprezentarea,

procesarea şi acţionarea pe bază de modele - a utilizatorului, domeniului, sarcinilor, etc. (prin imagini, limbaj natural, gesturi, etc.).

- *interfeţe multi-ecran* – implică mai multe monitoare pentru a furniza o interacţiune mai flexibilă. Acest tip de interfeţe este cel mai des folosit în jocuri.
- *interfeţe cu utilizatorul non-comandă* – care observă utilizatorul pentru a "deduce" nevoile şi intenţiile acestuia, fără ca el să formuleze explicit comenzi.
- *interfeţe cu utilizatorul reflexive* – utilizatorul controlează şi redefineşte sistemul doar prin intermediul interfeţei. De obicei acest lucru e posibil doar cu o interfaţă grafică foarte bogată;
- *interfeţe cu utilizatorul tangibile* – care pun accentul pe mediul fizic şi pe elementele sale (prezentate anterior);
- *interfeţe text* – sunt interfeţe care furnizează ca ieşire text dar care acceptă şi alte forme de intrări decât cele textuale;
- *interfeţe vocale* – acceptă intrări fie prin apăsarea unor taste sau butoane, fie verbale şi furnizează ieşiri sub formă de mesaje verbale;
- *interfeţe bazate pe limbaj natural* – folosite în motoarele de căutare şi paginile web. Utilizatorul tastează o întrebare / cuvinte / expresii şi aşteaptă un răspuns;
- *interfeţe zero - intrări* (zero-input interfaces) – primesc intrări de la un set de senzori şi nu de la utilizator;
- *interfeţe de mărire* (zooming user interfaces) – obiectele informaţionale sunt reprezentate la diferite nivele de scalare şi detaliere. Utilizatorul poate modifica scara de vizualizate pentru detaliere.

1.4. Moduri de comunicare

Modurile de comunicare pot fi clasice prin intermediul dispozitivelor tradiţionale precum tastatura, monitorul, mouse-ul, boxele audio, joystick-ul, etc.

O altă modalitate este prin *limbaj natural* prin recunoaşterea textului, vocii şi a gesturilor. „Folosirea limbajului natural este menită să servească la emiterea de către utilizator a solicitărilor adresate sistemului şi la furnizarea răspunsurilor" (Filip, 2004) de către sistem într-un mod cât mai natural.

Interfeţele viitoare au potenţialul de a furniza utilizatorului un spaţiu virtual care să-i permită un mod de comunicare mai flexibil, mai natural cu mediul computaţional sau

cu alţi utilizatori, furnizând intrări şi primind răspunsuri, folosind echilibrat toate simţurile disponibile şi canalele de comunicare, în timp ce resursele umane şi cele ale sistemului sunt optimizate. Atingerea unei astfel de performanţe necesită o cunoaştere şi înţelegere profundă a abilităţilor şi necesităţilor utilizatorului, precum şi sarcinile acestuia şi contextul de utilizare. De asemenea, este necesară o experienţă în metodele de descriere şi integrare a acestor cunoştinţe în procesul de proiectare a interfeţei cu utilizatorul (Stephanidis, 1999).

În 1991, Mark Weiser introducea conceptul de "*ubiquitous computing*", "tehnologia calmă". Ubiquitous computing reprezintă o paradigmă în care tehnologia devine practic invizibilă în vieţile noastre. În locul folosirii calculatoarelor de tip desktop sau laptop, tehnologia e încorporată în mediul utilizatorului, în obiectele şi acţiunile zilnice ale acestuia.

1.5. Standarde

Standardele referitoare la interfeţele cu utilizatorul au fost foarte discutate de-a lungul anilor atât în cadrul Organizaţiei Internaţionale de Standardizare cât şi la nivelul Uniunii Europene. În momentul de faţă sunt disponibile numeroase standarde şi ghiduri menite să îndrume dezvoltatorii în crearea sistemelor informatice, în general, şi a interfeţelor, în particular.

Foarte cunoscute în domeniul sistemelor informatice sunt standardele ISO 13407 din 1999 (Human Centered Decision Processes for Interactive Systems), ISO/IEC 14598-1 din 1988 (IT-Evaluation of Software Products – General Guide), ESO/IEC 9126-1 (Software Engineering Product Quality) şi ISO 9241.

Standardul ISO 13407:1999 descrie patru principii ale proiectării centrate pe utilizator:

- Implicarea activă a utilizatorilor finali (sau a reprezentanţilor lor);
- Alocarea corespunzătoare a funcţiilor astfel încât abilităţile umane să fie potrivit folosite;
- Iterarea soluţiilor de proiectare, acordând astfel timp planificării proiectului;
- Proiectare multi – disciplinară;

Standardul cuprinde, de asemenea, patru activităţi centrate pe om:

- Înţelegerea şi specificarea contextului de utilizare (fără presupuneri);

- Specificarea necesităţilor socio-culturale şi cele ale utilizatorului (numeroase puncte de vedere şi individualizări);
- Realizarea de soluţii de proiectare (proiecte multiple încurajează creativitatea);
- Implicarea utilizatorului final în testare.

Standardul, datorită gradului de generalitate, poate fi aplicat oricărui sistem sau produs.

Standardul ISO 9241 (din 1988) cuprinde o serie de îndrumări cu privire la interfeţele cu utilizatorul: principiile dialogului, îndrumări privind utilizabilitatea, prezentarea informaţiilor, îndrumarea utilizatorului, meniuri, comenzi, manipulare directă şi formulare.

Referitor la dialog, standardul furnizează o serie de principii generice care au fost extinse în *ISO 9241-110:2006*.

În partea referitoare la utilizabilitate nu sunt prezentate cerinţe sau recomandări specifice, dar, în schimb, se explică conceptul de utilizabilitate şi cum poate fi identificată informaţia necesară pentru a specifica şi evalua utilizabilitatea sistemelor informatice. Standardul prezintă utilizabilitatea ca dependentă de contextul de utilizare, iar utilizabilitatea unei anumite componente a sistemului este influenţată de sarcini, utilizatori, hardware, mediul fizic şi social, etc.

Printre altele, legat de prezentarea informaţiilor, se recomandă gruparea informaţiei, a listelor şi tabelelor, etichetelor şi câmpurilor, abrevierilor, a codului vizual al informaţiei (inclusiv codificarea pe culori şi grafice) şi necesarului pentru sistemele cu mai multe ferestre.

Îndrumarea utilizatorului se referă la orice informaţie, în afară de dialogul uzual, furnizată utilizatorului la cerere sau furnizată automat de sistem. Îndrumările ajută utilizatorul să obţină rezultatele dorite. Sunt furnizate recomandări legate de diferitele clase de sisteme de ajutor on-line, feedback, gestiunea erorilor (mesaje şi dialoguri asociate cu erorile şi funcţionările necorespunzătoare). În schimb nu se face nici un fel de recomandare legat de tutorialele şi materialele on-line.

Standardul ISO 9241-151:2008 (Guidance on World Wide Web user interfaces) furnizează îndrumări cu privire la proiectarea centrată pe om a interfeţelor cu utilizatorul a aplicaţiilor web. Interfeţele cu utilizatorul web se adresează fie tuturor utilizatorilor de

Internet, fie unui grup anume: de exemplu membrii unei organizaţii, clienţii sau furnizorii unei companii sau alte grupuri specifice de utilizatori. Recomandările standardului se focalizează asupra următoarelor aspecte cu privire la proiectarea interfeţelor web: strategii şi decizii de proiectare de nivel înalt, conţinutul proiectului, navigare şi căutare şi prezentarea conţinutului.

Standardul ISO 9241-110:2006 (Dialogue principles) stabileşte şapte principii de proiectare a sistemelor interactive ergonomice formulate în linii mari (de exemplu fără a face referire la situaţiile de utilizare, aplicaţie, mediu sau tehnologie) şi furnizează un cadru pentru aplicarea acestor principii în analiza, proiectarea şi evaluarea sistemelor interactive. Aceste principii sunt:

- dialogul (interacţiunea dintre utilizator şi sistem) trebuie să fie potrivit sarcinii – dialogul trebuie să fie corespunzător sarcinii utilizatorului şi competenţelor acestuia;
- auto-descriptiv – dialogul trebuie să fie clar astfel încât utilizatorul să ştie ce urmează să facă în continuare;
- controlabil – utilizatorul trebuie să poată controla interacţiunea;
- în conformitate cu aşteptările utilizatorului – trebuie să fie consistent;
- tolerant la erori;
- potrivit pentru învăţare – dialogul trebuie să sprijine învăţarea.

Folosirea standardelor în realizarea interfeţelor cu utilizatorul duce la scăderea timpului de învăţare a componentelor interfeţelor şi a funcţionalităţii lor, îmbunătăţeşte calitatea componentelor produse, promovează principii de proiectare testate şi îmbunătăţeşte colaborarea în cadrul echipei de-a lungul procesului de dezvoltare (Mulawa, 2006).

2. Modelarea interfeţelor cu utilizatorul

2.1. Proiectarea interfeţelor bazată pe modele

Proiectarea interfeţelor poate fi făcută prin mai multe metode: (1) proiectarea informaţională: destktop publishing, multimedia, data-mining; (2) proiectarea interacţiunilor om-maşină: story creating, story telling; (3) proiectarea senzorială: design grafic, ilustraţie şi fotografie, sound design, musical performance, vocal talents, videografie, cinema, tactile, olfactory, kinosthatic design (Buraga, 2008; Tung, 2003; Shedroff, 1994).

Dintr-o altă perspectivă proiectarea interfeţelor poate fi realizată cu unelte bazate pe limbaj (dezvoltatorul va programa interfaţa folosind un limbaj de programare), bazate pe specificaţii de interactivitate grafică (aceste unelte permit o proiectare interactivă a interfeţelor) şi bazate pe modele (folosesc un model sau specificaţie de nivel înalt pentru a genera automat interfaţa).

Dezvoltarea interfeţei cu utilizatorul rămâne dificilă şi consumatoare de timp atunci când se folosesc unelte bazate pe limbaj sau bazate pe specificaţii grafice. Majoritatea uneltelor de dezvoltare a interfeţelor grafice sprijină doar etapa de dezvoltare a ciclului de viaţă a interfeţei şi abstractizările pe care le furnizează au doar o legătură îndepărtată cu rezultatele analizei sarcinilor utilizatorului. Dezvoltarea bazată pe modele a interfeţelor cu utilizatorul şi uneltele sale reprezintă o tehnologie emergentă pentru a remedia aceste probleme a tehnologiilor curente printr-un suport cuprinzător a întregului ciclu de viaţă şi a metodologiei de proiectare centrată pe utilizator (Schlungbaum, 1996).

Paradigma dezvoltării interfeţelor bazată pe modele a atras un interes deosebit încă din anii ’90 datorită potenţialului său de a produce medii de dezvoltare de interfeţe cu utilizatorul integrate care să asiste proiectantul în toate fazele proiectării şi dezvoltării de interfeţe.

Premisa de bază în tehnologia bazată pe modele este că dezvoltarea interfeţei poate fi asistată în totalitate de un model generic, declarativ a tuturor caracteristicilor interfeţei cu utilizatorul, precum componenta de prezentare, de dialog şi toate caracteristicile legate de domeniul asociat, utilizator şi sarcinile utilizatorului. Având un

astfel de model, pachete de unelte care ajută în editarea şi manipularea automată a modelului pot fi create astfel încât este posibil suportul complet a proiectării şi implementării. De obicei, utilizatorii de medii bazate pe model ajustează modelul generic într-un model specific aplicaţiei folosind uneltele furnizate de mediu. Ulterior un sistem de rulare execută modelul modificat (Puerta, 1996).

Uneltele bazate pe modele au fost studiate încă din 1980. Prin aceste unelte, dezvoltatorii pot stabili specificaţiile interfeţelor cu utilizatorul la un nivel de implementare independent. De obicei specificaţiile interfeţelor sunt puse în comun de o serie de componente, numite modele care reprezintă diferite puncte de vedere a caracteristicilor interfeţei. Numărul şi tipul acestor modele este diferit, de la o abordare la alta.

Abordarea bazată pe modele a primit numeroase critici (Myers, 1999; Puerta, 1996; Shneiderman, 2006, citaţi de Stănciulescu, 2008), cele mai importante fiind următoarele:

1. dezvoltatorii trebuie să înveţe un nou limbaj pentru a reprezenta specificaţiile interfeţei cu utilizatorul;

2. fiecare sistem bazat pe modele are limitări stricte din punct de vedere al tipurilor de interfeţe ce pot fi produse iar aceste interfeţe nu sunt la fel de bune ca cele create prin tehnici convenţionale;

3. sistemele bazate pe modele nu suportă o gamă largă de explorări posibile;

4. conexiunile dintre specificaţii este greu de înţeles şi de controlat. De aceea rezultatele sunt imprevizibile;

5. în general modificările de la un anumit nivel nu sunt propagate la celelalte nivele de specificaţii;

6. multe modele sunt strâns legate de sistemul bazat pe modele asociat şi nu poate fi exportat. Mai mult decât atât, unele modele de specificaţii nu sunt nici publice şi nici nu pot fi obţinute prin licenţă.

Unele dintre aceste critici ar putea fi rezolvate după cum urmează:

1. multe modele pot fi create grafic într-un mediu de proiectare, ceea ce uşurează munca proiectantului, nemaifiind nevoit să înveţe limbajul de specificare. Chiar dacă

proiectanţii trebuie să înveţe limbajul de specificare, automatizarea unei porţiuni a procesului de dezvoltare ar trebui să reducă munca acestuia;

2. cea de-a doua critică se aplică anumitor tipuri de generatoare bazate pe modele, care generează interfaţa pornind de la modele de nivel înalt;

3. unele medii permit adăugarea de noi opţiuni de proiectare şi noi valori faţă de cele existente;

4. unele medii conţin seturi de reguli explicite, documentate şi accesibile (Stănciulescu, 2008).

Myers, Hudson, şi Pausch (Myers, 2000) susţineau că uneltele de proiectare bazate pe modele vor avea succes atunci când proiectanţii nu vor avea nevoie de foarte multe cunoştinţe pentru a dezvolta interfeţe iar o interfaţă simplă va putea fi realizată foarte uşor. De asemenea, uneltele de proiectare vor avea suficiente funcţii pentru a produce interfeţe sofisticate cu resurse moderate (din punct de vedere al cunoştinţelor necesare, a timpului, etc.).

Interfeţele bazate pe modele au şi o serie de avantaje:

- *Din punct de vedere al metodologiei:*

a. dezvoltarea unui sistem informatic porneşte de la etapa de specificaţii ale produsului;

b. abordarea bazată pe modele suportă o dezvoltare centrată pe utilizator şi pe interfaţă – proiectanţii lucrează cu elemente precum: sarcini, utilizatori şi domenii, în loc să gândească în termeni inginereşti.

- *Din punct de vedere al reutilizării:*

a. într-un context multiplatformă, uneltele bazate pe modele pot asigura portabilitate automată între diferite dispozitive;

b. existenţa unei descrieri complete a interfeţei într-o formă declarativă permite reutilizarea anumitor componente a interfeţei.

- *Din punct de vedere al consistenţei:*

a. această abordare asigură o formă de consistenţă între fazele iniţiale ale ciclului de dezvoltare (cerinţe, analiza, specificaţii) şi produsul final;

b. într-un context multiplatformă, este garantată, de asemenea, o consistenţă minimală între interfaţa cu utilizatorul generată pentru diferite platforme ţintă. Acest

lucru nu este întotdeauna posibil atunci când se folosește o tehnică tradiţională în care este puţin probabil ca dezvoltarea fiecărei versiuni a interfeţei să se realizeze separat (Stănciulescu, 2008).

2.2. Modelare UML

Modelarea este o parte esenţială a unui proiect software. Folosirea unui model ajută în a stabili dacă funcţionalitatea proiectului va fi corectă şi completă, toate necesităţile utilizatorului final vor fi satisfăcute, designul programului va satisface cerinţele de scalabilitate, robusteţe, extendabilitate etc. Deoarece sistemele ce trebuie dezvoltate sunt complexe, modelarea se face din mai multe perspective: moduri de utilizare, logic, componente, concurenţă şi desfăşurare.

UML (Unified Modeling Language) este un limbaj de modelare standardizat de uz general folosit pentru descrierea, specificarea şi documentarea funcţiilor unei aplicaţii în timpul etapei de proiectare. Limbajul UML constituie un mijloc de comunicare între dezvoltatorii sistemului informaţional şi utilizatori sau reprezentanţii acestora. De asemenea, UML furnizează şi un set de mecanisme de extensie a conceptelor fundamentale, care să asigure personalizările sau adaptările necesare unei situaţii, probleme sau viziuni specifice. Modelarea în UML este independentă de limbajul de programare sau de procesul de realizare, stabilite pentru sistemul care face obiectul studiului.

Pentru a oferi o viziune cât mai completă asupra sistemului de obţinut, UML propune realizarea unui set de diagrame care să faciliteze comunicarea atât în interiorul echipei de dezvoltare a produsului, cât şi cu utilizatori ori alte persoane, neimplicate în mod direct în operaţiile derulate de sistemul informaţional vizat.

Aceste diagrame, 13 la număr, sunt divizate în trei categorii. Şase tipuri de diagrame reprezintă structura aplicaţiei iar şapte reprezintă tipuri generale de comportament, inclusiv patru care sunt grupate într-o categorie aparte, ele reprezentând diferite aspecte ale interacţiunilor (Ambler, 2004).

Diagramele care reprezintă diferite tipuri ale *structurii* aplicaţiei sunt: diagrama de clase, diagrama de componente, diagrama de obiecte, diagrama de structuri compuse, diagrama de desfăşurare, diagrama de pachete. Diagramele de *comportament* reprezintă trăsături comportamentale ale sistemului: diagrame de activităţi, de cazuri de utilizare, de

stare şi cele patru diagrame de *interacţiune*: diagrama de secvenţe, diagrama de interacţiune (interaction overview), de colaborare şi de timp.

Fiecare diagrama oferă o perspectivă distinctă a sistemului, după cum sugerează şi denumirile lor.

O *diagramă a claselor* prezintă structură fizică a claselor identificate în sistem. Clasele reprezintă "elemente" gestionate de sistem; clasele pot fi legate în mai multe moduri: asociate (conectate între ele), dependente (o clasa depinde/foloseşte o altă clasă), specializate (o clasă este specializarea altei clase) sau împachetate (grupate împreună în cadrul unei unităţi). Toate aceste relaţii se materializează în structura internă a claselor în atribute şi operaţii. Diagrama este considerată statică, în sensul că este validă în orice moment din ciclul de viaţă al sistemului.

O *diagramă de componente* prezintă dependenţele existente între diverse componente software (cod sursă, cod binar, executabile, librarii cu legare dinamica etc.) ce compun un sistem informatic. Diagrama de componente este un graf de componente între care există relaţii de dependenţă sau de compunere (componente incluse fizic în alte componente).

Diagramele de obiecte prezintă obiectele şi relaţiile lor, fiind nişte diagrame de colaborare simplificate, fără reprezentarea mesajelor trimise între obiecte.

Diagrama structurilor compuse este o diagramă de obiecte care reprezintă structura de funcţionare şi relaţiile dintre obiecte. Conţinutul aceste diagrame e aproape identic cu cel din diagramele de clase doar că, în anumite situaţii, ar putea mult mai informative decât diagramele de clasă.

Diagrama de desfăşurare este utilizată pentru a modela configuraţia elementelor de procesare la momentul execuţiei şi distribuţia componentelor software, proceselor şi obiectelor pe aceste elemente de procesare.

O *diagrama de pachete* este o diagramă compusă doar din pachete şi relaţiile dintre acestea. Un pachet este o construcţie UML care permite organizarea elementelor modelului (cazuri de utilizare sau clase) în grupuri. Pachetele sunt descrise ca directoare de fişiere şi pot fi folosite în orice diagramă UML.

Diagramele de activităţi reprezintă comportamentul unei operaţii în termeni de acţiuni.

Un *caz de utilizare(use-case)* este o descriere a unei funcţionalităţi (o utilizare specifică a sistemului) pe care o oferă sistemul. Diagrama de cazuri de utilizare prezintă actorii externi şi cazurile de utilizare identificate, numai din punctul de vedere al actorilor (care este comportamentul sistemului, aşa cum este el perceput de utilizatorii lui) nu şi din interior, precum şi conexiunile identificate între actori şi cazurile de utilizare. Diagrama descrie ceea ce face un sistem din punctul de vedere al unui observator extern, această descriere fiind independentă de implementare. Adesea, o astfel de diagramă este folosită în etapele preliminarii a procesului de proiectare pentru a colecta cererile clientului cu privire la proiect. Aşadar, construcţia unui model de cazuri de utilizare, reprezentat printr-o diagrama use-case (sau mai multe) se face de obicei după lungi discuţii între dezvoltatori şi clienţi pe baza specificaţiilor asupra cărora au căzut cu toţii de acord. Un astfel de model va descrie şi va trebui să surprindă felul în care un actor va folosi un anumit sistem.

Pentru a crea o diagramă de cazuri de utilizare trebuie parcurşi mai mulţi paşi: definirea sistemului, identificarea actorilor şi a cazurilor de utilizare, descrierea acestora, definirea relaţiilor dintre cazurile de utilizare iar în final validarea modelului.

Cel mai important scop al unei diagrame de cazuri de utilizare este de a ajuta dezvoltatorii de sisteme software să vizualizeze cerinţele funcţionale ale unui astfel de sistem sau unităţi de sistem.

În ultima parte a acestei lucrări va fi prezentată o astfel de diagrama de cazuri de utilizare pentru o interfaţă cu utilizatorul.

O *stare* este de obicei un complement al descrierii unei clase. O *diagramă de stare* prezintă toate stările prin care trece un obiect al clasei precum şi evenimentele care-i cauzează modificările de stare. Modificarea stării se numeşte tranziţie. Se construiesc diagrame de stare doar pentru acele clase din sistem care au un număr de stări bine definit, iar comportamentul clasei este afectat şi modificat de acestea.

Diagramele de secvenţe prezintă temporal obiectele şi interacţiunile lor. Obiectele sunt văzute ca linii verticale distribuite pe orizontală, iar timpul este reprezentat pe axa verticală de sus în jos. Mesajele sunt reprezentate prin săgeţi între liniile verticale ce corespund obiectelor implicate în mesaj.

Diagramele de interacţiune (interaction overview) sunt variante de diagrame de activităţi.

Diagrama de colaborare surprinde colaborarea dinamică între obiecte, într-o manieră similară cu a diagramei de secvenţă, dar pe lângă schimbul de mesaje (numit şi interacţiune) prezintă obiectele şi relaţiile dintre ele (câteodată referite ca şi *context*).

Diagramele de timp sunt un tip specific de diagrame de interacţiune în care accentul cade pe constrângerile de timp.

2.3. Limbaje de descriere a interfeţei cu utilizatorului

Ca rezultat a creşterii diversităţii dispozitivelor şi a stilurilor de interacţiune, dezvoltatorii de interfeţe întâmpină dificultăţi precum lipsa unui abstractizări corespunzătoare a interacţiunii, nevoia de a crea diferite tipuri de design pentru interfaţa cu un singur utilizator şi integrarea componentelor hardware noi. Ca parte a efortului comunităţii de cercetare în domeniu pentru rezolvarea acestor probleme, conceptul de UIDL (User Interface Description Languages), care îşi are originea în sistemele de management a interfeţelor cu utilizatorul şi dezvoltarea bazată pe modele, a fost introdus ca o abordare promiţătoare. Aceste limbaje permit proiectanţilor să specifice o interfaţă cu utilizatorul folosind construcţii de nivel înalt, care abstractizează detaliile de implementare. Ulterior specificaţiile UIDL pot fi automat sau semiautomat convertite în interfeţe concrete. În ultimii ani au fost dezvoltate câteva astfel de limbaje, majoritatea bazate pe XML (eXtensible Markup Language) cu scopul de a simplifica crearea următoarelor generaţii de interfeţe. În ciuda avansului înregistrat în ultimul timp în acest domeniu, rămân câteva întrebări legate de utilitatea şi eficacitatea UIDL-urilor: Ce modele sunt necesare pentru a specifica comportamentul dinamic al generaţiilor viitoare de interfeţe care sunt caracterizate de interacţiuni multi-utilizator continue şi fizice? Cum pot fi aceste limbaje înţelese şi utile pentru dezvoltatorii de interfeţe cu o pregătire din alt domeniu? Cum vor fi evaluate aceste limbaje? Cum va afecta colaborarea dintre dezvoltatorii de interfeţe şi cercetătorii acestor limbaje cadrul arhitectural al interfeţelor viitoare? (Shaer, 2008)

Limbajul XML (Extensible Markup Language), descrie o clasă de obiecte numite documente XML şi descrie parţial comportamentul unor programe de computer care le procesează. XML este folosit în aplicaţii de structurare a datelor în baze de date,

structurarea documentelor, grafică vectorială (VML, sau Vector Markup Language), prezentări multimedia (SMIL – Synchronized Multimedia Integration Language; HTML + TIME, sau HTML Timed Interactive Multimedia Extensions), comunicarea deschisă între aplicaţii, via Web cu ajutorul mesajelor bazate pe XML (SOAP, sau Simple Object Access Protocol), schimbul de informaţii financiare (OFX, sau Open Financial Exchange), tranzacţii comerciale pe Internet (XFDL, sau eXtensible Forms Description Language), resurse umane (HRMML, sau Human Resource Management Markup Language), formatarea formulelor matematice pe Web (MathML, sau Mathematical Markup Language), descrierea structurilor moleculare (CML, sau Chemical Markup Language), scrierea partiturilor muzicale (MusicML, sau Music Markup Language), buletine meteo (OMF, sau Weather Observation Markup Format), tranzacţii imobiliare (RETS, sau Real Estate Transaction Standard).

Limbajul XISL (eXtensible Interaction Scenario Language) e singurul limbaj bazat pe web care permite dezvoltarea de interfeţe multimodale bazate pe scenarii de interacţiune între utilizator şi sistem. XISL realizează o separare a conţinutului (memorat în fişiere XML/HTML) de interacţiune (descrisă în documente XISL). Acest lucru oferă o serie de avantaje: reutilizabilitatea conţinutului şi/sau a interacţiunii şi îmbunătăţirea specificaţiilor de accesibilitate. De asemenea suportă intrări/ieşiri paralele şi secvenţiale şi intrări alternative.

Limbajul XIML (eXtensible Interface Markup Language) furnizează un cadru pentru definirea şi interrelaţionarea datelor de interacţiune. Datele de interacţiune se referă la date care definesc şi leagă toate elementele relevante ale interfeţei cu utilizatorul. Din punct de vedere structural, XIML include următoarele unităţi de reprezentare:

- Componente - colecţie organizată de elemente de interfaţă grupate în componente de bază din modele de interfaţă:
    - Sarcini – definesc o descompunere ierarhică a sarcinilor în subsarcini şi relaţiile dintre ele;
    - Obiectele domeniului – reprezintă o colecţie organizată de obiecte de date şi clase de obiecte structurată într-o ierarhie;
    - Tipuri de utilizatori – structurate într-o ierarhie de utilizatori;

  - Elemente de prezentare – o ierarhie de elemente de interacţiune compusă din obiecte concrete care comunică cu utilizatorii;
  - Elemente de dialog – o colecţie structurată de elemente care determină acţiunile disponibile utilizatorului;
- Relaţii – definiţie sau declaraţie care leagă două sau mai multe elemente XIML în cadrul aceluiaşi component sau între diferite componente;
- Atribute – caracteristici sau proprietăţi ale elementelor.

XIML permite dezvoltarea interfeţelor cu utilizatorul care trebuie să fie afişate pe diferite dispozitive. XIML poate fi utilizat pentru afişarea efectivă a unui singure definiţii de interfaţă pe orice tip de dispozitiv vizat. Există o serie de aplicaţii care transformă specificaţiile XIML în limbaje binecunoscute (de exemplu HTML, WML). XIML este, de asemenea, suportat de o serie de unelte precum XIML Validator, XIML Editor şi XIML Viewer.

Limbajul UIML (User Interface Markup Language) este un limbaj XML folosit pentru definirea elementelor interfeţei: butoane, meniuri, liste şi alte controale care permit unui program să funcţioneze într-o interfaţă grafică. UIML este folosit pentru definirea locaţiei şi pentru proiectarea controalelor. UIML e un metalimbaj. El defineşte un set redus de tag-uri (folosite pentru descrierea părţilor interfeţei) care e independent de platforma vizată (calculator, telefon, etc.) şi de limbajul vizat (Java, VoiceXML, etc.). UIML separă elementele unui interfeţe şi identifică care părţi compun interfaţa şi stilul de prezentare, conţinutul fiecărei părţi (text, sunet, imagine, etc.) şi legăturile conţinutului cu resursele externe şi comportamentul părţilor exprimat ca un set de reguli.

UIML grupează logic interfaţa într-un arbore de componente ale interfeţei care se modifică pe parcursul vieţii interfeţei. Pe durata vieţii unui interfeţe, arborele iniţial de componente îşi poate modifica dinamic forma prin adăugarea şi ştergerea de elemente.

UIML permite elementelor interfeţei şi a arborelui să fie grupate în şabloane. Aceste şabloane pot fi ulterior refolosite în alte interfeţe.

Limbajul DISL (Dialog and Interface Specification Language) e un subset UIML. Diferenţele faţă de UIML sunt legate de dispozitive (widget) generice şi îmbunătăţiri ale aspectelor comportamentale. Dispozitivele generice sunt introduse pentru a separa modul de prezentare de structură şi comportament. Structura globală a limbajului DISL constă

într-un element head opţional pentru meta informaţii şi o colecţie de şabloane şi interfeţe de unde o interfaţă e considerată activă la un moment dat. Interfeţele sunt utilizate pentru a descrie structura dialogului, stilul şi comportamentul întrucât şabloanele doar descriu structura şi stilul pentru a fi reutilizabile de către alte componente de dialog.

Limbajul VoiceXML e un limbaj pentru crearea interfeţelor cu utilizatorul vocale. Limbajul permite integrarea serviciilor de voce cu serviciile de date folosind paradigma tradiţională client-server. Un serviciu de voce este văzut ca o secvenţă de interacţiuni între utilizator şi o platformă de implementare. VoiceXML furnizează caracteristici pentru a suporta dialoguri complexe: ieşiri de voce sintetizată (text - > voce), ieşiri sub formă de fişiere audio, recunoaşterea intrărilor de voce, recunoaşterea tonurilor DTMF, înregistrarea vocii, funcţii pentru telefonie.

3. Arhitectura unei interfeţe pentru un Sistem Suport pentru Decizii

3.1. Sisteme Suport pentru Decizii şi Sisteme Suport pentru Decizii de Grup

Un *Sistem Suport pentru Decizii* (DSS) reprezintă un sistem informatic *interactiv*, *flexibil* şi *adaptabil* special dezvoltat pentru a oferi suport în găsirea soluţiei unor probleme *nestructurate* (Turban, 1995) şi/sau *semi-structurate*, cu scopul de a îmbunătăţi procesul decizional. El foloseşte date, modele şi/sau baze de cunoştinţe, furnizează utilizatorului o interfaţă intuitivă şi uşor de utilizat şi poate încorpora cunoştinţele utilizatorului. DSS-urile ajută în soluţionarea problemelor complexe prin utilizarea de modele simplificate şi sisteme user-friendly.

Sisteme Suport pentru Decizii de Grup (GDSS) sunt DSS-uri ce asistă luarea deciziilor colective (co-decizii) de către un grup de persoane cu poziţii de autoritate similare. Sistemelor suport pentru decizii de grup trebuie să fie uşor de folosit, flexibile, să permită participări anonime în anumite etape ale şedinţei decizionale asistate de sistem şi comunicarea uşoară între membrii grupului. Printre avantajele folosirii unui astfel de sistem se numără reducerea comportamentului negativ a grupului şi înregistrarea automată a şedinţei.

3.2. Interfeţe pentru GDSS

Sistemele suport pentru decizii de grup pot fi aplicaţii tradiţionale, instalate la o anumită locaţie (de exemplu o cameră decizională) sau pot fi bazate pe web (Lu, 2005). Aplicaţiile bazate pe web au avantajul de a sprijini deciziile de grup în interiorul organizaţiilor distribuite geografic. Astfel sistemul poate fi implementat ca servicii web, la şedinţele decizionale putând participa persoane de oriunde din lume fără să aibă nevoie de nimic altceva decât de un browser web pe un calculator conectat la Internet.

Aplicaţiile bazate pe web au anumite caracteristici comune. Prima caracteristică este legată de faptul că sunt distribuite, adică sarcinile computaţionale au loc în diferite locaţii fizice. De obicei interfaţa cu utilizatorul rulează pe un alt calculator decât cel pe care se află aplicaţia. Arhitectura client-server e folosită ca paradigmă de bază. Interfaţa e

implementată sub forma unui client de mici dimensiuni, universal şi extensibil. Aplicaţia include părţi reutilizabile. Informaţiile specifice aplicaţiei sunt implementate ca date, extensii de server şi scripturi care rulează atât la client cât şi la server. Spre deosebire de aplicaţiile tradiţionale, aplicaţiile web rulează încontinuu. Aplicaţia nu este închisă odată ce utilizatorul şi-a îndeplinit sarcinile (Hejda, 2000). Tehnologia web permite utilizatorilor acces rapid şi ieftin la o cantitate foarte mare de informaţii furnizate pe site-urile web, librăriile digitale sau alte surse de date.

Datorită evoluţiei exponenţiale a web-ului şi a avantajelor sale, au fost create sisteme suport pentru decizii de grup bazate pe web pentru o gamă largă de activităţi decizionale (Group Systems.com[2], Facilitate.com[3], Meetingworks.com[4]).

Filip în 2008 prezenta o listă activităţi decizionale de bază pe care un GDSS „tipic şi complet" ar trebui să le asiste.

*1. Generarea de idei (*plan de acţiune, set de alternative decizionale identificate sau proiectate, mulţimea criteriilor de evaluare etc.), care pot servi la abordarea problemei decizionale. Componentele programelor software ale SSDM (denumite şi instrumente în literatura despre SSDM) care pot fi folosite pentru asistarea generării de idei sunt date sunt: a) *brainstorming electronic*, prin care participanţii pot introduce în sistem, în paralel şi sub protecţia anonimatului, texte care conţin propriile idei privitoare la un subiect dat. La sfârşitul sesiunii, care se recomandă să dureze 30-45 de minute , sistemul produce un raport care conţine ideile propuse, b) *comentarea subiectelor*, cu ajutorul căreia, fiecare participant are acces la o listă de subiecte în vederea introducerii comentariilor proprii la subiectele selectate. Pentru aceasta, el poate să aleagă oricare subiect şi să citească comentariile primite deja de la alţi participanţi, c) *conturarea de grup*, forma cea mai structurată de formulare şi comentare a ideilor, serveşte la prezentarea subiectelor sub forma unui arbore sau a unei liste multinivel, la care participanţii îşi pot asocia, în mod ordonat, comentariile.

2. Organizarea ideilor deja generate prin plasarea acestora sub câteva idei principale. De obicei, această activitate, care se recomandă să dureze 45-90 de minute ,

[2] http://www.groupsystems.com cu GDSS-ul ThinkTank
[3] http://www.facilitate.com cu GDSS-ul FacilitatePro Web Software
[4] http://www.meetingworks.com cu GDSS-ul Meetingworks Connect

reduce mulţimea de idei iniţiale la un număr de cca 20 ori mai mic de idei centrale. Componentele software (instrumentele) care pot fi folosite pentru organizarea ideilor sunt: a) *gruparea ideilor*, cu ajutorul căreia se creează un număr de categorii de idei (uneori pe baza acelor idei care par a fi cele mai importante sau a avea un caracter mai general), în care participanţii pot plasa ideile deja generate, b) *analiza aparitiilor*, care îi asistă pe participanţi să identifice aparitiile cele mai importante din lista de idei deja generate şi să finiseze comentariile anexate acestor elemente.

3. Prioritizarea, prin care se stabileşte importanţa fiecărei idei cheie reţinute. Componentele software (instrumentele) cu ajutorul cărora se obţine, în cca. 10-20 de minute o listă de priorităţi sunt: a) *votarea*, prin care se realizează: asistarea selecţiei unei metode de votare (prin „da" sau „nu", sau prin acordarea de note sau de poziţii într-un clasament), exercitarea votului şi elaborarea raportului privind rezultatele, b) *chestionarul on-line*, care serveşte la crearea de către facilitatorul (sau moderatorul) GDSS-ului a unui set de întrebări şi permite realizarea sintezei răspunsurilor introduse on-line de către participanţi, c) *dicţionarul grupului*, care ajută la crearea interactivă a definiţiilor pentru elementele utilizate în procesul decizional.

4. Elaborarea unor politici, prin care participanţii creează şi adoptă decizii, planuri şi politici decizionale. Componentele software (instrumentele) sunt: a) *formularea politicilor*, care facilitează elaborarea în comun de către participanţi a unor documente referitoare la politici sau misiuni. Aceasta se realizează cu ajutorul unor versiuni succesive ale documentului – din care prima este elaborată de către facilitatorul (sau moderatorul) grupului – până la atingerea consensului de către participanţi. b) *analiza* , prin care se evaluează, în mod sistematic, implicaţiile planurilor şi politicilor.

În cadrul unei sesiuni de lucru asistat de GDSS, activităţile descrise mai sus se desfăşoară iterativ, în cicluri, până când (în cazul sesiunii decizionale) o idee (alternativă decizională) este selecţionată ca soluţie a problemei de decizie, sau atunci când (în sesiunile cu caracter exploratoriu) un set de idei (alternative, criterii de evaluare) sunt reţinute în vederea unor analize ulterioare.

Activităţile de suport care pot fi asistate cu ajutorul unui SSDM privesc:

5. Managementul sesiunilor, care constă în: pregătirea ordinii de zi , controlul desfăşurării sesiunii şi prelucrarea rezultatelor.

6. Gestionarea resurselor comune de grup, pentru care se pot folosi următoarele componente (instrumente): a) *lista participanţilor*, b) *planşeta de desen*, care este un instrument pentru realizarea în comun a unor desene şi a adnotării lor, c) *calibrarea opiniilor*, care este o formă simplă de votare care are menirea de a uniformiza aprecierile, d) *materialele de referinţă* menite să fie văzute de toţi participanţii.

7. Gestionarea resurselor individuale, pentru care se pot folosi următoarele componente (instrumente): a) *monitorul de evenimente*, care are ca menire informarea participanţilor asupra activităţilor, b) *jurnalul individual*, care îi permite fiecărui participant să-şi ia notiţe, c) *servieta* , care conţine o serie de programe de aplicaţie foarte des folosite: poştă electronică, editor de texte, calculator etc.

Aplicaţiile existente pe piaţă la momentul actual asistă aceste activităţi într-o mai mică sau o mai mare măsură.

3.3. Arhitectura unei interfeţe GDSS

Ţinând cont de lista de activităţi decizionale care ar trebui asistate de un GDSS complet prezentată mai sus, în continuare va fi prezentată descrierea şi arhitectura unei interfeţe pentru un astfel de sistem. Prezentarea se va face din perspectiva succesiunilor în timp a activităţilor decizionale asistate de un astfel de sistem.

În cadrul unui astfel de sistem există două tipuri de roluri pe care le poate avea utilizatorul final: *facilitatorul* şedinţei decizionale asistate de sistem şi membrul în echipa decizională, numit, în continuare, *participant*. Nu se va lua în considerare, în această etapă, rolul administratorului sistemului care, uneori se confundă cu facilitatorul.

Ţinând cont de acest lucru, interfaţa trebuie să asigure selectarea unuia dintre cele două roluri.

Odată accesat sistemul pe bază de username şi parolă pentru facilitator şi pe bază de nume si cod de acces sesiune (creată în prealabil de facilitator), sistemul va afişa interfeţe diferite, în funcţie de rol. Facilitatorul va avea acces la toate resursele, iar participantul doar la anumite resurse.

Elementele afişate utilizatorului derivă din acţiunile (conform modelului de GDSS prezentat anterior) pe care acesta trebuie să le desfăşoare pentru o şedinţă decizională de succes.

Facilitatorul trebuie:

1. Să creeze şi să stabilească detaliile sesiunii decizionale (de exemplu: data, ora, numele, obiectivul, comentarii (instrucţiuni) şi cheia de acces pentru participanţi);
2. Să creeze agenda de lucru, momentul începerii, timpul alocat fiecărui eveniment, drepturile participanţilor;
3. Să particularizeze activităţile - setări corespunzătoare fiecărei activităţi în parte;
4. Să încarce materiale de referinţă (fişiere);
5. Să grupeze ideile generate sub câteva idei principale;
6. Să genereze raportul întâlnirii.

Participantul trebuie să poată adăuga idei (în secţiunile dedicate) în conformitate cu obiectivele şedinţei decizionale şi să participe la activităţile de prioritizare a ideilor (vot şi chestionare).

În continuare sunt prezentate o serie de schiţe ale diferitelor pagini ale interfeţei web a sistemului în ordinea în care ele se vor succeda. Spre deosebire de acesta, *participantul*, după logare va fi automat redirecţionat în tab-ul *Întâlnire*.

În Figura 1 este prezentată o schiţă a interfeţei de acces la sistem. Modul de structurate poate fi şi cu tab-uri (tab-ul Facilitator şi tab-ul Participant) după modelul aplicaţiei ThinkTank de la GroupSystems.com.

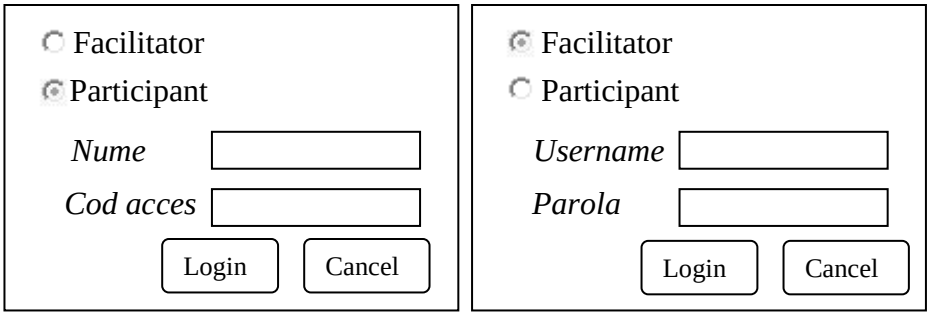

Fig. 2 Exemplu schiţă a unei ferestre de logare

Odată logat, facilitatorului i se va deschide automat fereastra *Detalii Sesiune* iar la apăsarea butonului OK, fereastra *Crează Agenda* se deschide automat. Butonul *Drepturi* îi va deschide facilitatorului fereastra de configurare a drepturilor participanţilor. Implicit, participanţii au drepturi diferite asupra diferitelor tipuri de activităţi.

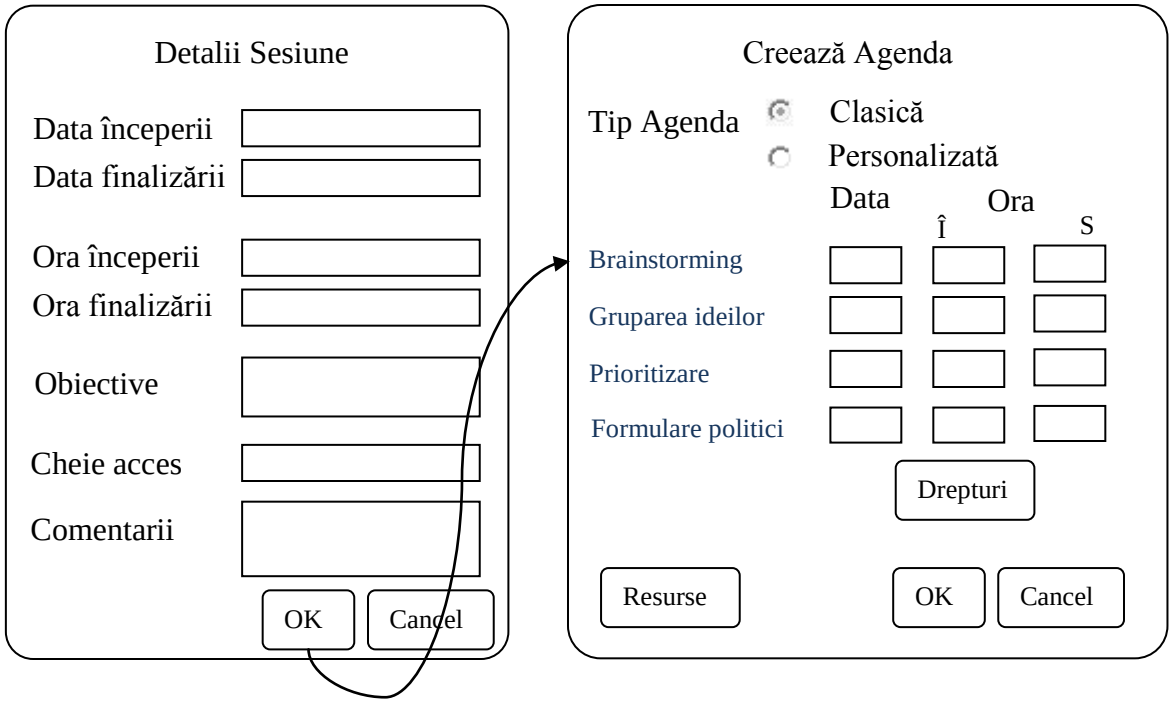

Fig. 3 Schiţe ale ferestrelor de setare a sesiunii şi a agendei

În cazul selectării unei agende particularizate, facilitatorul va avea ocazia să bifeze activităţile dorite dintr-o listă de activităţi. Executarea unui clic pe numele unei activităţi, în această fereastră, permite configurarea activităţii respective.

După setarea detaliilor *Agendei* şi apăsarea butonului OK, facilitatorul va fi redirecţionat în pagina *Întâlnire*.

În tab-ul *Întâlnire* vor exista mai multe secţiuni (sub formă de frame-uri sau sub formă de ferestre). Va exista o secţiune Agenda în care vor fi afişate activităţile prevăzute în agendă cu icoane în dreptul activităţilor încheiate sau în curs de desfăşurare. Icon-ul corespunzător activităţii în derulare va fi animat. În secţiunea activităţii curente (va purta numele activităţii) se vor desfăşura acţiunile corespunzătoare acesteia.

Tab-ul *Întâlnire* va fi avea aproape acelaşi aspect pentru ambele tipuri de utilizatori. Facilitatorul va avea câteva opţiuni (în funcţie de activitate) în plus faţă de participant.

În cazul în care Facilitatorul va opta pentru o fereastra de comunicare instant, aceasta va fi afişată pe tot parcursul întâlnirii. Va exista şi secţiunea *Instrucţiuni, obiective comentarii*.

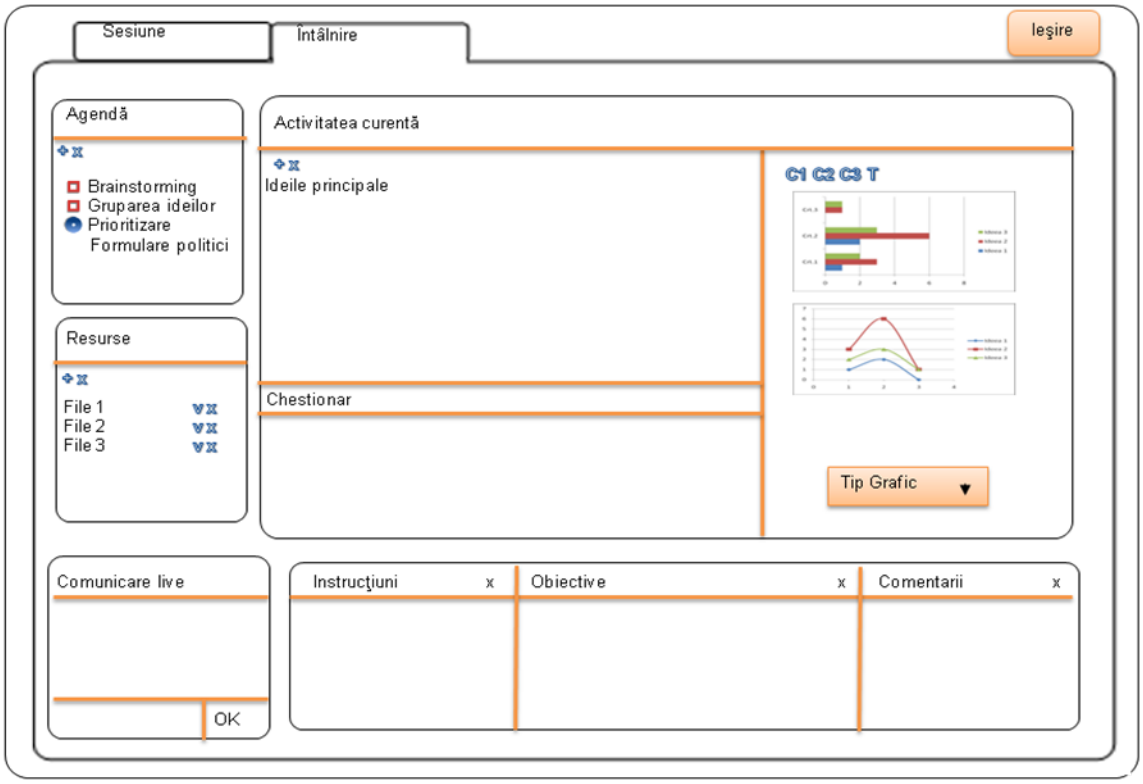

Fig. 4 Schiţa tab-ului *Întâlnire*, unde se va desfăşura întreaga întâlnire

O altă secţiune va fi dedicată resurselor. Participanţii pot adăuga sau şterge din resurse doar dacă facilitatorul le-a permis acest lucru.

Fiecare activitate are o serie de caracteristici particulare care sunt stabilite de facilitator. Interfaţa pentru configurarea acestora şi desfăşurarea lor vor fi detaliate ulterior.

În tab-ul *Sesiune* are acces doar facilitatorul. Vor fi afişate: lista participanţilor (cu posibilitatea de invita şi ulterior alţi participanţi), agenda (care poate fi modificată atât în tab-ul *Întâlnire* cât şi în *Sesiune*), secţiunea instrucţiuni şi secţiunea detalii sesiune (stabilite iniţial şi care pot fi modificate).

După terminarea şedinţei, ieşirea din aplicaţie se face prin apăsarea butonului *Ieşire*.

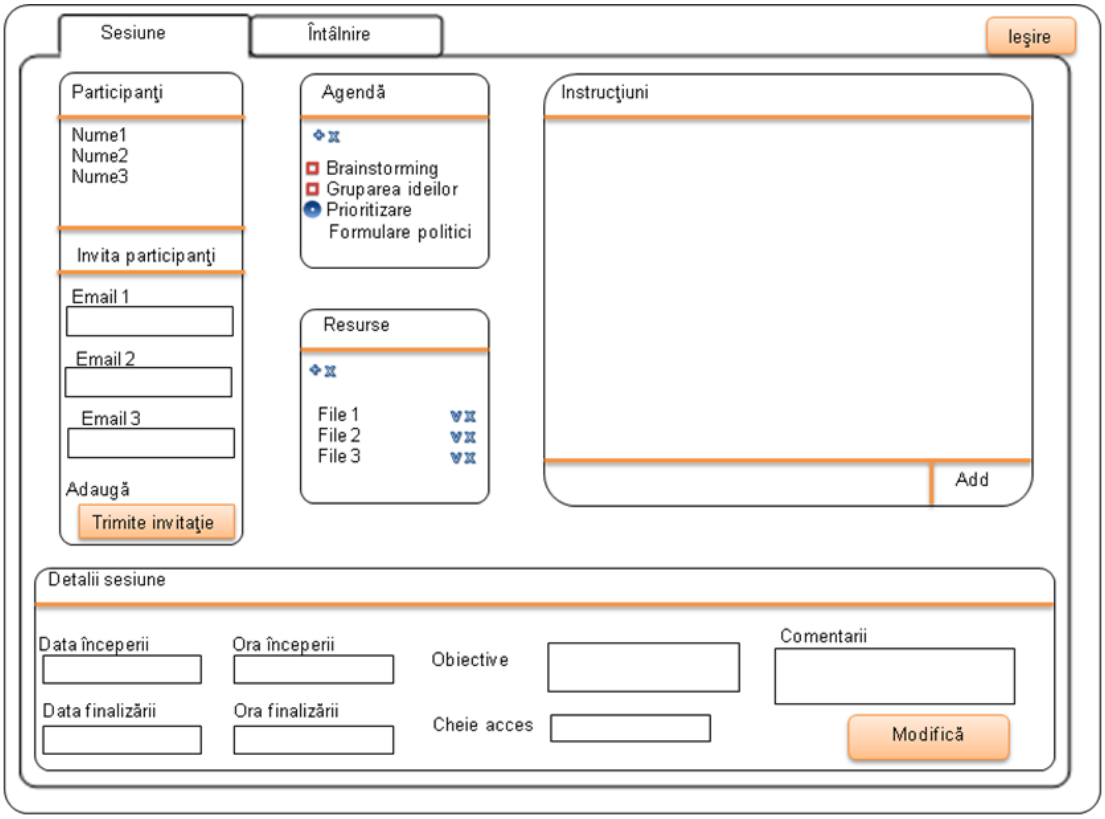

Fig. 5 Schiţa tab-ului *Sesiune*

3.4. Modelul unei interfeţe pentru GDSS

În 2000, Silva afirma că e normal ca interfaţa cu utilizatorul a unui sistem să fie modelată cu UML, chiar dacă nu este tocmai uşor de a identifica cum pot fi reprezentate elementele interfeţei cu ajutorului modelelor UML.

Pentru proiectarea interfeţelor cu utilizatorul este importantă abstractizarea cazurilor de utilizare esenţiale. Acest lucru îi permite proiectantului să modeleze structura esenţială de sarcini fără presupuneri premature legate de detaliile interfeţei (Constantine, 2000).

În figura 6 este prezentat un model realizat în UML folosind diagrame *use case*.

Diagrama scoate în evidenţă tipurile de utilizatori şi cazurile principale de utilizare: crearea sesiunii şi a agendei de către facilitator şi transmiterea ideilor şi participarea la prioritizarea ideilor principale obţinute de către un utilizator (o categorie generică ce include facilitatorul şi participantul). Se poate observa că activităţile pe care

le desfăşoară *Utilizatorul* sunt moştenite de *Facilitator* şi *Participant*. *Participantul* nu e implicat în nici un caz de utilizare special faţă de cele moştenite, mai puţin logarea în calitatea sa de membru a echipei de lucru.

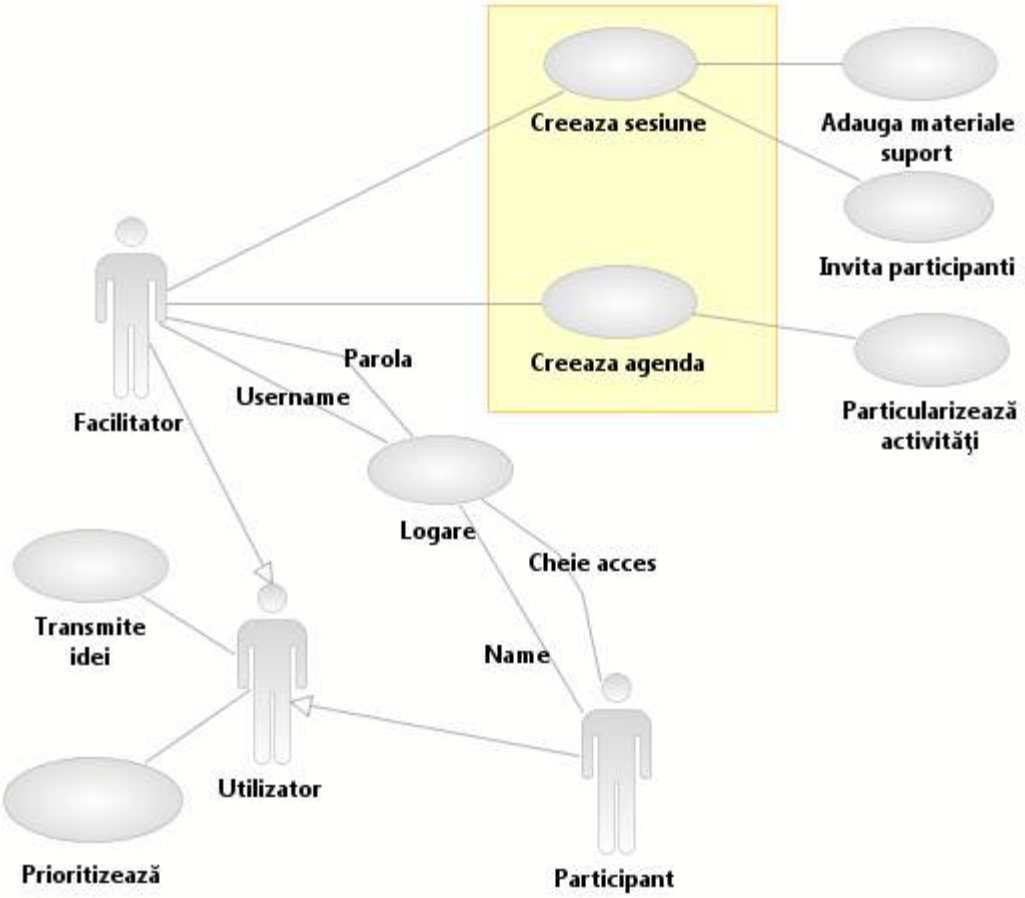

Fig. 6 Modelul UML folosind diagrame de cazuri de utilizare

În diagrama următoare se poate observa cele două tipuri de agende (clasică şi particularizată).

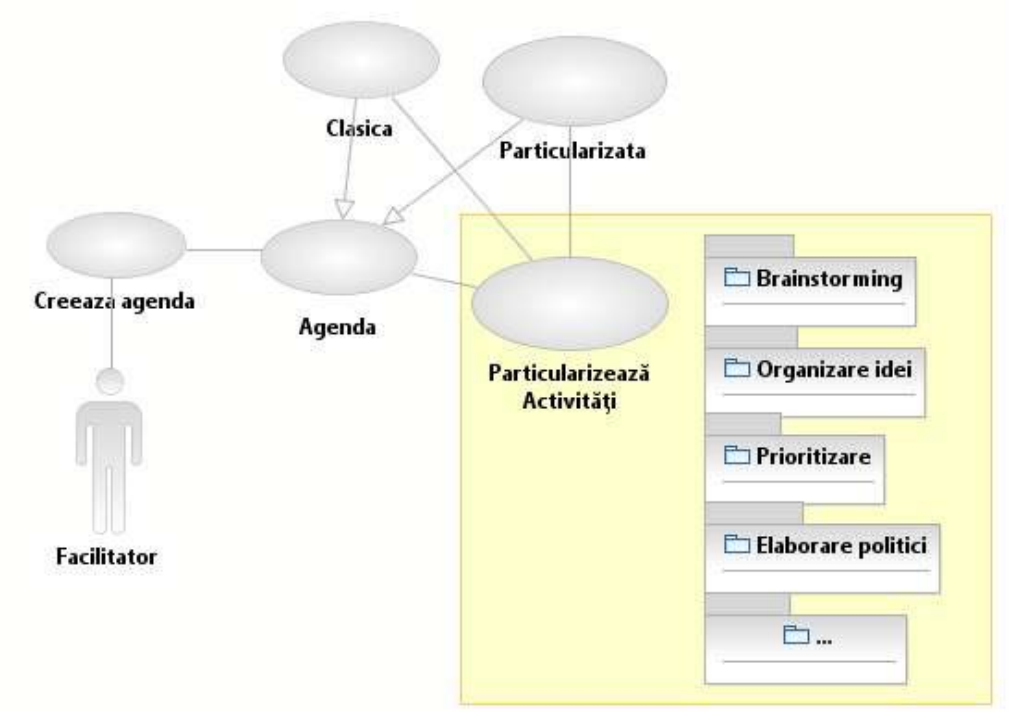

Fig. 7 Crearea agendei

Din figura următoare se poate observa că organizarea ideilor poate avea loc doar dacă se realizează generarea ideilor şi crearea categoriilor.

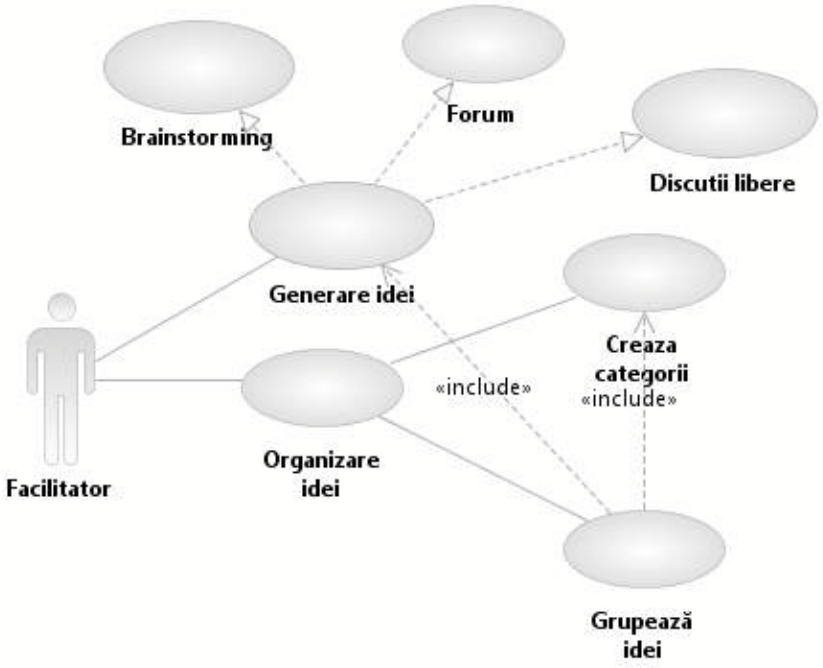

Fig. 8 Organizarea ideilor

Generarea ideilor se poate face prin brainstorming, forum şi discuţii libere.

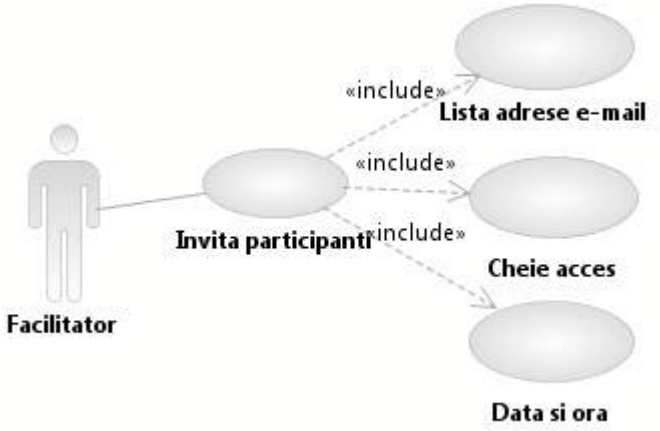

Fig. 9 Invitarea participanţilor

Invitarea participanţilor poate avea loc doar după stabilirea listei de adrese de e-mail a persoanelor ce sunt invitate, stabilirea cheii de acces la sesiune şi a momentului începerii sesiunii.

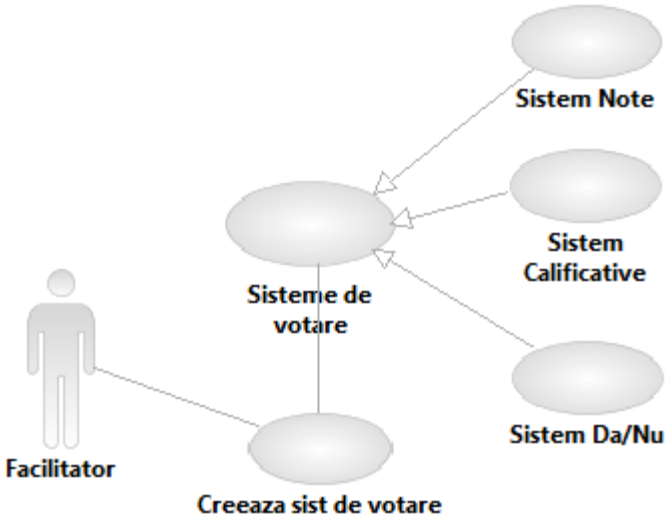

Fig. 10 Sisteme de votare

Sistemele de votare sunt de trei tipuri: sistem bazat pe note (într-un interval stabilit de facilitator), sistem bazat pe calificative (stabilite, de asemenea de facilitator) şi bazat pe răspunsuri de tip DA/NU.

În cazul chestionarelor, trebuie stabilite întrebările, tipul răspunsurilor (casetă text, grilă sau de tip DA/NU).

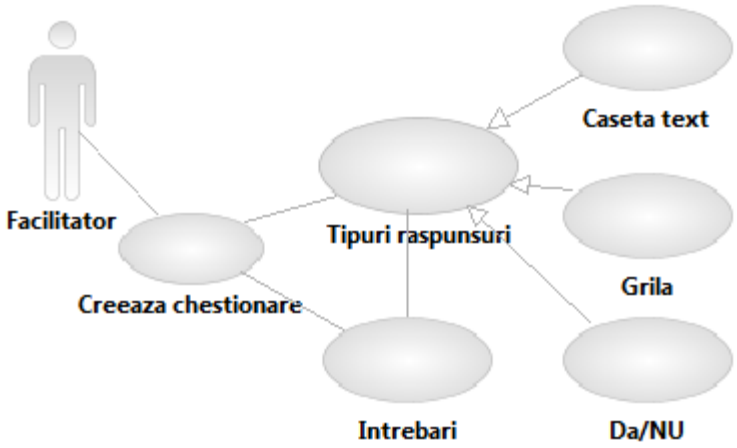

Fig. 11 Crearea de chestionare

Modelul creat are un grad de abstractizare ridicat, specific primei etape de proiectare a unui sistem. Modelul va suferi modificări pe parcursul dezvoltării interfeţei.

Concluzii

În momentul de faţă cele mai folosite tipuri de interfeţe sunt cele grafice şi cele bazate pe web. Pe lângă acestea există şi alte tipuri folosite în funcţie de scopul aplicaţiei şi de tipul utilizatorului: interfeţe în linie de comandă, interfeţe de tip batch, interfeţe cu utilizatorul inteligente, interfeţe cu utilizatorul tangibile, interfeţe bazate pe limbaj natural, Interfeţe vocale, etc.

Proiectarea interfeţelor poate fi făcută prin mai multe metode.

Paradigma dezvoltării interfeţelor bazată pe modele a atras un interes deosebit încă din anii ’90 datorită potenţialului său de a produce medii de dezvoltare de interfeţe cu utilizatorul integrate care să asiste proiectantul în toate fazele proiectării şi dezvoltării de interfeţe.

Premisa de bază în tehnologia bazată pe modele este că dezvoltarea interfeţei poate fi asistată în totalitate de un model generic, declarativ a tuturor caracteristicilor interfeţei cu utilizatorul, precum componenta de prezentare, de dialog şi toate caracteristicile legate de domeniul asociat, utilizator şi sarcinile utilizatorului. Având un astfel de model, pachete de unelte care ajută în editarea şi manipularea automată a modelului pot fi create astfel încât este posibil suportul complet a proiectării şi implementării. De obicei, utilizatorii de medii bazate pe model ajustează modelul generic într-un model specific aplicaţiei folosind uneltele furnizate de mediu. Ulterior un sistem de rulare execută modelul modificat.

Chiar dacă se optează pentru o altă abordare decât cea bazată pe modele în crearea interfeţei, modelarea, în etapa de proiectare, este o activitate foarte importantă deoarece ajută în a stabili dacă funcţionalitatea produsului va fi corectă şi completă, dacă toate necesităţile utilizatorului final vor fi satisfăcute, designul programului va satisface cerinţele de scalabilitate, robusteţe, extendabilitate etc.

Referinţe bibliografice

Ambler, S., (**2007**),*User Interface Flow Diagrams (Storyboards)*, http://www.agilemodeling.com/artifacts/uiFlowDiagram.htm

Ambler, S.W., (**2004**), *The Object Primer 3rd Edition: Agile Modeling Driven Development with UML 2*, Cambridge University Press;

Andersen, P.B. (**2001**). What Semiotics can and cannot do for HCI, *Knowledge-Based Systems*, Volume 14, Number 8, pp. 419-424(6);

Aspinall, D., (**2007**), *Designing Interaction*, University of Edinburgh;

Baxley, B. (**2003**), *Universal Model of a User Interface*, ACM;

Buraga, S., (**2008**), *Interacţiune Om-Calculator*, Universitatea Alexandru Ioan Cuza;

Constantine, L. and Lockwood, L. (1999), *Software for Use: A Practical Guide to the Essential Models and Methods of Usage-Centered Design*. Reading, MA: Addison-Wesley;

Constantine, L. and Lockwood, L. (**2000**), Structure and Style in Use Cases for User Interface Design, http://www.foruse.com/articles/structurestyle2.pdf ;

Filip, F., G., (**1999**), *Informatica industriala: Noi paradigme si aplicatii*, Ed. Tehnica, Bucuresti;

Filip, F., G., (**2002**), *Decizie Asistata de Calculator: Decizii, decidenti - metode si instrumente de baza*, Ed. Tehnica, Bucuresti;

Filip, F., G., (**2004**), *Sisteme suport pentru decizii*, Ed. Tehnica, Bucureşti;

Filip, F.G. (**2008**), *Decision Support and Control for Large-Scale Complex Systems*, Annual Reviews in Control, No. 32, pp. 61-70;

Gadomski, A., M., Bologna, S., DiCostanzo, G., Perini, Anna, Schaerf, M., (**1999**) *An Approach to the Intelligent Decision Advisor (IDA) for Emergency Managers*, TIEMS'99, The Sixth Annual Conference of The International Emergency Management Society, Delft, Netherlands, June 8-11;

Gilliams, S., Van Orshoven, J., Muys, B., Kros, H., Heil, G., Van Deursen, W., (**2005**), *AFFOREST sDSS: a metamodel based spatial decision support system for afforestation of agricultural land*, New Forests, Vol. 30, No. 1. pp. 33-53;

Hennicker, R. Koch, N. (**2001**) Modeling the User Interface of Web Applications with UML, http://www.pst.informatik.uni-muenchen.de/~kochn/pUML2001-Hen-Koch.pdf

Lu, J., Zhang, G., Wu, F., (**2005**), *Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function*, IEEE Intelligent Informatics Bulletin, Vol. 5, Nr. 1, http://www.comp.hkbu.edu.hk/~cib/2005/Jun/iib_vol5no1_article5.pdf;

Mulawa, M., Picking, R., Grout, V., (**2006**), *Defining Development Standards for Reusable User Interface Components*, 6th International Network Conference, University of Plymouth, http://www.newi.ac.uk/groutv/Papers/P4.pdf ;

Myers, B., A., Hudson, S., E., Pausch, R., F., (**2000**), *Past, present and future of user interface software tools*, ACM, Trans. Computer-Human Interaction, Volume 7, No. 1, 2000, pp. 3-28;

Myers, B.A., Rosson, M.B., (**1992**), *Survey on User Interface Programming*. In: P. Bauersfeld, J. Bennett, G. Lynch (eds.): Striking a Balance. Proceedings CHI'92 (Monterey, May 1992), New York: ACM Press, 195-202;

Nichols, J., Faulring, A., (**2005**), *Automatic Interface Generation and Future User Interface Tools*, Proceedings of the Workshop on the Future of User Interface Design Tools at CHI, Portland, http://www.jeffreynichols.com/papers/nichols-faulring-uitools.pdf

Puerta, A. (**1996**), The MECANO project: Comprehensive and integrated support for model-based interface development, https://eprints.kfupm.edu.sa/70502/1/70502.pdf;

Schlungbaum , E. (**1996**), *Model-based User Interface Software Tools*, http://smartech.gatech.edu/bitstream/1853/3516/1/96-30.pdf

Shaer, O., Jacob, R., Green, M., Luyten, K., (**2008**), *User Interface Description Languages for Next Generation User Interfaces*, CHI 2008 Workshop Proceeding, http://www.eecs.tufts.edu/~oshaer/workshop/CHI08_ws_proc.pdf

Sharp, H., Rogers, Y., Preece, J. (**2006**), *Interaction Design: Beyond Human-Computer Interaction. Second Edition*, John Wiley Press;

Shneiderman, B., Fischer, G. (**2006**), *Creativity Support Tools*: Report From a U.S. National Science Foundation Sponsored Workshop, International Journal of Human-Computer Interaction, 20(2), 61–77, http://hcil.cs.umd.edu/trs/2006-17/2006-17.pdf;

Silva, P., Paton, N. (**2000**), *User Interface Modelling with UML*, http://www.ksl.stanford.edu/people/pp/papers/PinheirodaSilva_IMKB_2000.pdf

Stephanidis, C., (**1999**), *User Interfaces for All*, ERCIM News No.39;

Tung, F.W., Deng, Z.S. (**2003**), *A Study on Integrating Interaction Design into Industrial Design Processes*, The 6th Asia Design International Conference, Tsukuba, Japan;