Universitatea Tehnică din Cluj-Napoca

Teza de doctorat

Contribuții în controlul accesului bazat pe roluri în aplicații de comerț electronic

Autor: ing. Mihaela Georgetta Ordean

Conducător ştiințific: prof.dr.ing. Dorian Gorgan

- iulie 2008 -

CUPRINS

Lista figurilor

Lista tabelelor

1 Introducere

Controlul accesului are drept obiectiv protejarea resurselor faţă de accesul neautorizat si se realizeaza prin acordarea de permisiuni. Un sistem de control al accesului implică o politică prin care se specifică cine poate accesa o anumită resursă şi în ce manieră o poate face. Politicile sunt în general exprimate prin starea curentă a sistemului şi stările care pot rezulta din aceasta. In momentul în care sistemul de control al accesului este perceput ca şi un sistem de tranziţii între stări, acesta constă din setul de stări, regulile de tranziţii între stări şi un set de proprietăţi sau interogări care sunt de interes într-o anumită stare (vezi si 3.1.).

În Internetul de astăzi există un număr din ce in ce mai mare de aplicaţii care necesită decizii de autorizare. Aceste aplicaţii includ (dar nu sunt limitate la) comerţul electronic, gestionarea şi partajarea resurselor distribuite, execuţia şi descărcarea de cod de la distanţă (exemplu: applet-uri Java şi controale ActiveX). Autorizarea acestor aplicaţii este semnificativ diferită de cea a sistemelor centralizate şi chiar de cea a sistemelor distribuite relativ mici. În sistemele de autorizare pe Internet există mai multe entităţi, multe dintre acestea necunoscându-se una pe cealaltă şi, de multe ori, neexistând o autoritate centrală de încredere pentru toţi actorii. Regulile conform cărora se realizează deciziile de acordare a controlului pot necesita modificări ale acestor aplicaţii, deciziile pentru permiterea accesului la o resursă specifică depinzând de tipul aplicaţiei. Din acest motiv există diferite politici pentru a se realiza controlul accesului.

Odată cu extinderea Internetului, calculatoarele individuale care initial erau protejate prin politici locale de control al accesului, sunt interconectate şi accesează resurse atât local cât şi de la distanţă. Astfel atat numărul resurselor care pot fi accesate a crescut cat şi numărul utilizatorilor care accesează aceste resurse.

Controlul bazat pe roluri este un concept neutru de politici pentru obtinerea controlului accesului. Au fost propuse diferite modele de-a lungul timpului precum modelele de control al accesului discret şi cel obligatoriu (Discretionary Access Control – DAC – şi Mandatory Access Control – MAC), modelul Clark-Wilson, modelul de control al accesului bazat pe roluri (Role Based Access Control – RBAC) şi bazat pe actiuni (Task Based Model - TBAC). Modelul de referinţă în controlul accesului bazat pe roluri este RBAC [FGS01], [SCH+96]. In cadrul RBAC permisiunile de acces nu sunt asignate în mod direct utilizatorilor ci abstractizării cunoscute sub denumirea de “rol”. Au fost propuse diferite extensii dintre care două sunt semnificative: una concentrându-se pe specificarea constrângerilor [SC96], iar cealaltă descriind un cadru pentru delegarea autorităţii prin intermediul rolurilor administrative [SBM98].

RBAC este un model relativ recent pentru controlul accesului, care poate gestiona în mod dinamic seturi de utilizatori şi resurse. Acest model simplifică anumite aspecte ale politicii de administrare şi furnizează un mijloc de luare a deciziilor pentru controlul accesului.

In cadrul modelelor convenţionale existente (amintite mai sus), de control al accesului bazat pe roluri, accesul la resurse se realizează de către utilizatori certificati de catre un inginer de sistem. Aceste modele nu sunt adecvate sistemelor deschise şi

descentralizate în care utilizatorii sunt anonimi şi identitatea acestora nu este cunoscută a priori.

Pentru sistemele deschise s-au propus metode de control al accesului bazate pe credenţiale care implementează noţiunea de access binar bazat pe încredere. Astfel, dacă un utilizator câştigă încrederea pe baza unui credenţial va fi admis în sistem şi va avea acces la resursele acestuia iar, în caz contrar, nu i se va accorda accesul.

Un sistem de control al accesului este totodată şi o instanţă a unei scheme de control al accesului si specifică tipurile de reguli relativ la tranziţiile între stări [TL04]. Un exemplu de model pentru controlul accesului este modelul bazat pe matrici de acces [GD75]. In cadrul acestui model un utilizator iniţiază acţiuni sau operaţii asupra obiectelor, iar acestea (acţiunile sau operaţiile) sunt acceptate sau respinse în funcţie de autorizarea specificată în sistem. Matricea de acces este un model conceptual care specifică drepturile pe care le are fiecare utilizator asupra fiecărui obiect. Acest model se utilizează în special in sistemele de operare [TAP+05].

1.1 Obiectivele tezei

Lucrarea de faţă se referă la un domeniu particular al autorizării şi propune modelul SCAR-ACE pentru controlul accesului bazat pe roluri în aplicaţii de comerţ electronic.

Pe măsură ce creşte complexitatea aplicaţiilor sunt necesare politici noi de administrare şi control al accesului. Aplicaţiile de comerţ electronic devin din ce în ce mai complexe, impunand accesul la resurse heterogene a utilizatorilor cu roluri diferite. Controlul accesului în aplicaţiile de comert electronic este un subiect important al cercetarii stiintifice actuale.

(1) **Direcţia de cercetare**

În [B05] sunt definite direcţiile de cercetare pentru modelele, arhitecturile şi tehnologiile de control al accesului. Teza se subscrie unei direcţii şi anume realizarea unui model care să suporte controlul accesului într-un sistem distribuit. SCAR-ACE extinde abordarea propusă în [B05] prin utilizarea maşinilor de stare.

(2) **Problema de rezolvat**

Unul dintre obstacolele privind accesul pe Web la resurse îl reprezintă inabilitatea de a gestiona autorizarea într-o manieră eficientă si fara costuri relativ mari, in ce priveste timpul si banii. In [PSA01] se prezinta două soluţii de implementare RBAC de control al accesului pe Web. Ambele soluţii sunt modele bazate pe cookies, lasand astfel deschisa o breşă în securitate.

Lucrarea de faţă îşi propune realizarea unui model sigur de control al accesului bazat pe roluri şi care să abordeze dintr-o perspectivă diferită controlul accesului fara a utiliza cookies. Modelul propus permite doar utilizatorilor autorizaţi să acceseze resursele sistem, printr-un control bazat pe ierarhii de roluri si printr-o gestionare a rolurilor mutual exclusive.

Un alt obiectiv al lucrarii este modelarea in RBAC a fluxului de control a accesului în aplicaţiile distribuite.

(3) **Soluția propusă**

Pentru a controla într-o aplicație distribuita fluența şi accesul la resurse se introduce noțiunea de rol [L97] ca un element intermediar între utilizator şi setul de permisiuni ale acestuia (vezi figura 1).

Fiecărui rol i se ataşează un set de permisiuni (regăsite în unele accepțiuni sub denumirea de privilegii) de acces la operații şi resurse.

Figura 1 ilustrează accesul utilizatorilor (vezi şi 2.1, 4.1.) la operații şi resurse prin intermediul rolurilor în SCAR-ACE:

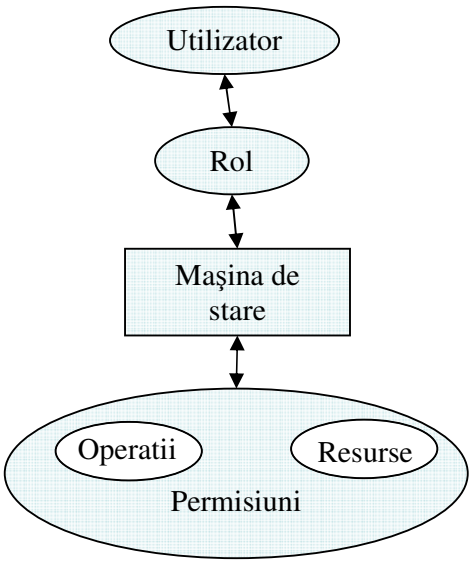

Figura 1. Relația intre utilizator, rol şi permisiuni

Săgeata dublă ilustrează o relație de n-m între utilizator şi rol, rol şi maşina de stare, maşina de stare şi permisiuni.

Scopul oricărui mecanism de control al accesului este de a proteja resursele. În termenii modelului SCAR-ACE resursele reprezinta obiecte: fişiere, date, înregistrări, tabele într-un DBMS sau operații. Atât modelul formal cât şi cel funcțional al SCAR-ACE utilizează maşini de stare pentru realizarea controlului accesului.

Modelul SCAR-ACE are următoarele funcționalități necesare realizării controlului accesului:

- asocierile utilizator/rol: reprezinta utilizatorii autorizați să realizeze un anumit rol;
- constrângeri de tipul separarea îndatoririlor (asocieri de tip utilizator / rol care indică conflict de interese):
    - constrangeri statice (Static Separation of Duties – SSD - vezi şi 2.3.3): un utilizator nu poate fi autorizat pentru două roluri (de exemplu recepționer al cererii şi furnizor de bunuri);
    - constrângeri dinamice (Dynamic Separation of Duties – DSD - vezi şi 2.3.4): un utilizator poate fi autorizat pentru două roluri dar nu poate acționa simultan în ambele (de exemplu vânzător şi cumpărător al aceluiaşi produs).

1.2 Structura tezei

Teza este structurată în sase capitole în care este descris atât modelul cât şi partea de validare a acestuia pentru o aplicaţie de comerţ electronic.

Capitolul întâi contine introducerea în domeniul sistemelor de control al accesului şi obiectivele tezei.

Capitolul al doilea este destinat stadiului actual al cercetării în domeniul tezei şi contine descrierea primelor modele pentru controlul accesului şi implementările curente. Totodată sunt trecute în revistă colectivele care se ocupă de modele de control al accesului.

Capitolul al treilea conţine descrierea modelului SCAR-ACE. Sunt definiţi termenii care sunt utilizaţi în cadrul lucrării după care se detaliază componentele modelului SCAR-ACE în comparaţie cu modelul de referinţă RBAC, respectiv SCAR-ACE$_0$, SCAR-ACE$_1$, SCAR-ACE$_2$ şi SCAR-ACE$_3$. In acest capitol este descris modelul SCAR-ACE, prin analiza constrângerilor, desemnarea privilegiilor şi nivelele de acordare a privilegiilor, nivelele de securitate, politica şi algebra politicii de acordare a privilegiilor. Capitolul detaliază modul în care se realizează controlul accesului în SCAR-ACE. In acest scop sunt descrise componentele pe care se bazează controlul accesului respectiv maşina de stare, rolurile şi constrângerile. Este analizat modelul formal pentru controlul accesului în SCAR-ACE si fazele de realizare a acestuia.

Următorul capitol este destinat validarii modelului SCAR-ACE. Astfel, în capitolul al patrulea sunt analizate componentele şi arhitecturile aplicaţiilor de comerţ electronic, modelele de afaceri în comerţul pe Internet şi asigurarea intimităţii în cadrul aplicaţiilor de comerţ electronic. Se detaliaza implementarea prin care se validează modelul SCAR-ACE. Astfel este descrisă arhitectura aplicaţiei de comerţ electronic implementată si structura aplicaţiei.

Rezultatele experimentale si testele efectuate sunt prezentate in capitolul al cincilea.

In capitolul sase sunt prezentate concluziile şi dezvoltările ulterioare ale modelului.

Anexa întâia conţine bibliografia.

2 Starea actuală a cercetării în controlul accesului

Acest capitol examinează tehnologiile şi cercetările realizate în control accesului. Voi începe cu o trecere în revistă a primelor modele pentru controlul accesului urmată de o introducere în RBAC în care se vor descrie pe scurt conceptele de bază şi extensiile disponibile. Apoi vor fi prezentate cele mai cunoscute implementări ale modelului de baza RBAC şi colectivele implicate în dezvoltarea de sisteme de control al accesului.

2.1 Primele modele în controlul accesului

Incă din 1960 controlul accesului a fost un subiect de interes în special în managementul bazelor de date şi în sistemele de operare. Obiectivul acestuia era, pe de o parte, cel de a proteja resursele sistemului față de accesul neautorizat si, pe de altă parte, de a permite accesul autorizat.

Primele modele în domeniul controlului accesului au fost: Mandatory Access Control (MAC) şi Discretionary Access Control (DAC).

2.1.1 MAC - Mandatory Access Control

MAC a fost introdus in domeniile militar şi guvernamental, realizând controlul accesului prin introducerea de etichete de securitate. Acest model, formalizat inițial de Bell şi LaPadula [BL75], ataşează etichete sau clasificări de securitate fiecărui obiect. Deoarece diferitele forme ale modelului Bell-LaPadula, determinate de multe variațiuni, au dus la confuzii, Sandhu introduce un model minimal [S93] numit BLP care încapsulează părțile esențiale ale modelului Bell-LaPadula.

In MAC accesul este garantat pe baza etichetelor de securitate ataşate subiectelor sau obiectelor. Denotarea etichetelor de securitate era $\lambda(s)$ şi respectiv $\lambda(o)$. Aceste etichete de securitate formează o latice asupra căreia se poate aplica relația de ordine parțială $\leq$.

Deoarece MAC a fost dezvoltat pentru domeniul militar, confidențialitatea este importanta, si se obține prin restricții de fluență între obiecte sau între diferite grupuri de securitate. Aceste restricții se exprimă prin următoarele două proprietăți:

- proprietatea de securitate simplă, cunoscută ca şi „no read up”, impune ca un subiect să poată citi un obiect dacă $\lambda(o) \leq \lambda(s)$;
- proprietatea *, sau „no write down”, permite unui subiect să scrie un obiect numai dacă $\lambda(s) \leq \lambda(o)$. Această proprietate se adresează „scăpărilor” de informații de către programe malițioase. Prin această proprietate nu se permite programelor să scrie informații în obiecte care ar putea fi citite de către subiecți cu privilegii de securitate scăzute. Proprietatea * permite aşadar scrierea obiectelor care au aceeaşi etichetă de securitate ca şi utilizatorii, adică exprimat formal: $\lambda(s) = \lambda(o)$.

Administratorii de rețea sunt responsabili pentru întreținerea etichetelor de securitate; astfel, nici un alt utilizator nu poate schimba aceste etichete. Ori de câte ori un obiect este dublat acestuia i se ataşeaza o etichetă de securitate identica cu cea a obiectului initial.

Un model MAC similar cu BLP a fost publicat de Biba [B75]. Spre deosebire de BLP, modelul lui Biba îşi propune obţinerea integrităţii datelor prin metode opuse confidenţialităţii. Acest model permite fluenţa datelor numai dinspre date cu integritate înaltă către date cu integritate scăzută, exact invers accesului din modelul BLP.

Modelele MAC sunt însă destul de rigide. Acestea suportă un set fix de clase de securitate şi permit doar administratorilor de sistem să modifice politica de securitate.

2.1.2 DAC - Discretionary Access Control

DAC îşi are originile în aria academică. El permite sau restricţionează accesul la un obiect pe baza identităţii utilizatorului care îl accesează. Utilizatorilor le este permis accesul prin permisiuni asupra obiectelor proprii. Totodată utilizatorilor le este permis să delege drepturile proprii altor utilizatori. Un exemplu clasic de DAC este Access Control List (ACL) în care obiectele sunt asociate la o listă de utilizatori (care formeaza grupuri) cărora le este permis accesul.

Intr-o aplicaţie statică, în care accesul nu depinde de acţiuni anterioare, accesul la toate obiectele pentru toţi utilizatorii se poate reprezenta printr-o matrice. Această matrice, denumită matrice de acces [L71], [S92], are câte un rând pentru fiecare utilizator şi o coloană pentru fiecare obiect. Elementele matricii memorează drepturile de acces ale unui utilizator individual pentru un anumit obiect. Această matrice poate avea dimensiuni enorme şi, de obicei, este rar populată. Operaţia de control al accesului include dezvoltarea şi reprezentarea acestei matrici. In sistemele de control al accesului, această matrice a fost implementată în diferite moduri:

- memorată pe coloane astfel încât fiecare coloană este o listă. Acesta este modelul ACL (Access Control List);
- memorată pe linii şi fiecare linie cunoscută sub denumirea de capabilităţi. Astfel de capabilităţi memorează o listă a privilegiilor acordate fiecărui subiect;
- prin memorarea doar a celulelor utile ale matricii împreună cu poziţia acestora în matrice.

Aceste abordări au avut atât avantaje cât şi dezavantaje. Problemele apar de obicei la inserarea de utilizatori, la ştergerea anumitor utilizatori existenţi sau la crearea şi ştergerea obiectelor. De exemplu, în cazul listelor de control al accesului, ştergerea unui utilizator implica verificarea fiecărei liste din sistem. In organizaţiile în care atât utilizatorii cât şi obiectele de protejat se modifică frecvent, DAC s-a dovedit a fi neadecvat.

2.2 Autorizarea în sisteme distribuite

Una dintre primele implementări în domeniul autorizării în sisteme distribuite este Grapevine [BLN82] in care serverele determină dacă un client este membru al unui grup autorizat. O abordare similară se regăseşte în Sun Yellow Pages unde fişierele gestionate centralizat (precum *etc/group*) sunt consultate in prin autorizarea utilizatorilor. În ambele cazuri, cu toate că deciziile de autorizare se realizează local, clientul trebuie să obţină datele de autorizare de la un server aflat la distanţă.

Un alt model de autorizare pentru sistemele distribuite este Kerberos [MN88]. Acest model se bazează pe principiul că fiecare utilizator este autorizat pentru un numar de servicii şi fiecare serviciu îşi gestionează utilizatorii proprii. Scopul sistemului de autentificare Kerberos este de a permite serviciilor să implementeze modele diferite de autorizare prin care se permite accesul utilizatorilor. Determinarea utilizatorilor autorizati se realizeaza pe baza unui tichet specific fiecarui serviciu.

Proiectul SESAME [M95], [PP95] extinde sistemul de autorizare Kerberos şi defineşte o schemă pentru propagarea sigură a privilegiilor, inclusiv roluri şi grupuri, de la clienţi la serverele de aplicaţie.

Alte sisteme propuse oferă soluţii de autorizare off-line, precum delegarea certificatelor în Arhitectura de Securitate a Sistemelor Digitale Distribuite [GGK+89], [GM90], mecanismul de autentificare în cascadă a lui Sollins [S88], autorizarea proxy-based a lui Neuman [N93] şi autorizarea bazată pe certificate a lui Woo and Lam [WL98].

2.3 Utilizarea rolurilor în sisteme distribuite

Conceptul de roluri se utilizează în aplicaţiile software din anii ‘75, dar abia după 1990 autorizarea bazată pe roluri s-a maturizat în modelul RBAC. Acest model vine ca şi o completare a modelelor MAC şi DAC şi este definit prin intermediul entităţilor următoare [S96]: utilizatori, roluri, ierarhii de roluri, permisiuni şi constrângeri.

Rădăcinile RBAC includ utilizarea grupurilor în UNIX şi alte sisteme de operare, si gruparea bazata pe privilegii în sistemele de gestionare a bazelor de date. Modelul RBAC încapsulează toate aceste noţiuni într-un model unic în termeni de roluri şi ierarhii de roluri, activarea rolurilor şi constrângeri asupra rolului pe utilizator.

O constrângere tipică de autorizare, recunoscută şi relevantă, este separarea îndatoririlor (Separation of Duties – SoD) care se aplică asupra rolurilor, asignării rol-utilizator şi asignării rol-permisiuni.

În ultimii ani s-au implementat diverse mecanisme de autorizare bazate pe roluri în: managementul bazelor de date, managementul securităţii şi sisteme de gestionare a reţelelor.

Mecanismele de autorizare determină ca la autentificări diferite ale unui utilizator în cadrul aceleiaşi sesiuni de lucru, acestuia să i se asigneze de fiecare dată un rol nou prin excluderea de la cel vechi. Administrarea securităţii bazate pe roluri este simplificată datorită asignării rol-permisiuni.

Primele eforturi pentru definirea unui standard pentru tehnologiile de control al accesului bazate pe roluri au fost raportate la workshop-ul ACM RBAC în 2000 de catre Shandu si Ferraiolo. Atunci se pun bazele standardului care a fost aprobat cu un an mai târziu, de către NIST (National Institute of Standards and Technology) în autorizarea bazată pe roluri utilizând reţele Petri pentru elaborarea căruia au participat 28 de organizaţii.

Modelele şi implementările existente bazate pe roluri sunt relativ similare în conceptele fundamentale utilizate dar diferă semnificativ în detalii. Similaritatile şi diferenţele nu sunt evidente deoarece modelele folosesc terminologii diferite în descrierea aceloraşi concepte. Deoarece controlul accesului bazat pe roluri este o tehnologie relativ nouă şi deoarece produsele şi modelele vin din medii academice şi comerciale diferite nu prea există un consens asupra denumirii părţilor constituente. Controlul accesului bazat pe roluri este, de asemenea, o tehnologie relativ complexă şi sofisticată şi tratarea acesteia ca un model unic este oarecum nerealistă tocmai datorită inexistenţei unui consens în abordarea acesteia. Un model unic fie ar include fie ar exclude prea mult prezența unui singur punct de vedere asupra unui spectru larg de tehnologii şi variante.

2.4 Implementări RBAC

In [B95] Barkley descrie o implementare RBAC într-un limbaj de programare obiectual. Fiecare rol este reprezentat printr-o clasă diferită. La rulare, obiectele rol servesc ca şi proxy-uri pentru aplicațiile care cer accesul la anumite permisiuni. Abordarea din [B95] nu ia în considerare ierarhiile de roluri şi constrângerile pe care modelul SCAR-ACE le include şi le implementează.

Sistemul hyperDRIVE [B97] utilizează standardul LDAP pentru a implementa servicii de autentificare şi control al accesului. Sistemul face uz de tehnologia Java Applet. Applet-ul Java hyperDRIVE este încărcat într-un navigator şi furnizează o interfață pentru accesarea resurselor web protejate. Un server LDAP serveşte drept depozit centralizat pentru informațiile de autentificare şi control al accesului. Definirea ierarhiilor de roluri şi a rolurilor mutual exclusive este suportată de atribute LDAP cu scop special. Abordarea SCAR-ACE diferă în concept şi implementează rolurile şi privilegiile asociate acestora prin introducerea maşinilor de stare.

Sistemul I-RBAC [TC97] implementeaza controlul accesului în Intranet. O caracteristică a I-RBAC este utilizarea paralelă a ierarhiilor de roluri locale şi globale. Rolurile locale au permisiuni asupra resurselor locale de pe un anumit server. Rolurile globale au permisiuni asupra aşa-numitelor resurse globale care au un identificator unic în cadrul Intranet şi pot fi accesate global. I-RBAC nu permite definirea rolurilor mutual exclusive. Spre deosebire de această implementare, SCAR-ACE este un model pentru aplicații pe Internet care acceptă roluri globale şi permite existența rolurilor mutual exclusive.

In [G99] se descrie un mecanism RBAC bazat pe servleți Java. Se propune o arhitectură denumită JRBAC-WEB care se bazează pe execuția cererilor HTTP (GET, PUT) utilizându-se un servlet Java de securitate. Implementarea suportă definirea ierarhiilor de roluri şi constrângerilor de separare a îndatoririlor. JRBAC-WEB foloseşte autentificarea HTTP. In mod opțional abordarea permite păstrarea sesiunilor de lucru prin utilizarea cookies sau prin tehnici de rescriere a URL-ului pentru servleți Java. SCAR-ACE nu utilizează servleți Java (ci credențiale) şi nici cookies.

RBAC/Web [FBK98] este o implementare a unui serviciu de control al accesului bazat pe roluri pentru servere web. RBAC/Web nu conține un mecanism specific de autentificare şi suportă ierarhii de roluri, constrângeri de cardinalitate şi

separarea dinamică şi statică a îndatoririlor. Un server Web poate utiliza o extensie pentru a accesa direct API-ul RBAC/Web sau să forwardeze apelurile către scripturi CGI. Modelul SCAR-ACE nu este un simplu serviciu ci o aplicaţie care oferă o suită de servicii şi include autentificarea utilizatorilor pe baza de nume utilizator si parola.

2.5 Colective care au implementat modele RBAC

2.5.1 Colectivul de la George Mason University, SUA

Ravi Sandhu este unul dintre pionierii cercetărilor in controlul accesului bazat pe roluri. El, studenţii şi colegii săi, au dezvoltat mai multe modele de control al accesului şi extensii ale acestora. In [M98] Sandhu, Ferraiolo şi Kuhn revizuiesc cele patru modele RBAC (vezi şi 3.4) în încercarea de standardizare, dar descrierea modelelor realizată de acest colectiv rămâne foarte generală. In [FSG01] Ferraiolo et al. descriu o propunere de standard RBAC printr-o specificaţie formală utilizând notaţia Z.

Majoritatea modelelor RBAC în care a fost implicat Sandhu suportă ierarhiile de roluri. In [M98] autorii consideră ierarhiile de roluri ca fiind o prerechizită în realizarea constrângerilor de separare a îndatoririlor.

Lucrarea [S00] este una dintre puţinele care adresează câteva probleme ale RBAC în sisteme distribuite. In acest articol autorii furnizează câteva simulări ale modelelor prin utilizarea de entităţi autonome care sunt administrate separat.

Sandhu şi colectivul său se numără printre puţinii care au acordat atenţie modului de administrare a RBAC şi introduc astfel modelul ARBAC [SBC+97].

Majoritatea modelelor sunt cuprinzătoare şi bine specificate dar, cu toate acestea, nu suportă parametri. Parametrii ajută în abstractizarea rolurilor şi privilegiilor simplificand administrarea politicii de securitate. Necesitatea parametrilor a fost recunoscută în [KS03].

Determinând necesitatea unui mecanism de control al accesului flexibil (care să nu depindă de aplicaţie), în [BJS96] Jajodia si Bertino propun un model de control al accesului care suportă autorizările pozitive şi negative. Acest model suportă autorizarea puternică (cea care este obligatoriu a fi realizată) precum şi autorizarea slabă (pentru permiterea excepţiilor). In cadrul acestui model s-a realizat FAM (Flexible Authorization Manager) [JSS+97], [JSS+01] care este un model formal de forţare de politici de control al accesului multiplu în cadrul unui sistem. Acest model utilizează predicate de tipul *cando*, *do*, predicate derivate de tipul *dercando*, *done*, *error*. Cu acestea se exprimă ceea ce poate sau nu poate face un subiect. Deoarece se consideră ambele politici (atât cea pozitivă cât şi cea negativă), este luată în calcul şi rezolvarea conflictelor. Aceste modele nu sunt bazate pe roluri dar suportă grupurile şi delegarea şi au servit ca un fundament solid pentru RBAC.

In [SRJ01] se defineşte FAF (Flexible Authorization Framework) care introduce rolurile în timp ce este menţinută autorizarea pozitivă şi negativă. In cadrul acestui model este păstrată şi noţiunea de grup.

FAF poate, de asemenea, considera istoricul accesului în luarea deciziilor, iar suportul real pentru constrângeri a fost descris în [SRJ01]. Prin acest articol sunt introduse constrângerile temporale şi se evidentiaza validarea conceptului.

2.5.2 Colectivul de la Imperial College, Londra

Ponder este o implementare RBAC dezvoltată de Imperial College, Londra, de către colectivul condus de Sloman şi Lupu. Inaintea considerării controlului accesului bazat pe roluri colectivual a fost implicat în managementul politicilor de securitate [S94]. Ulterior, s-au luat în considerare politici prin care se exprimau permisiuni şi obligaţii [L98], [LS99], [LS97]. S-au luat în calcul atât politici de securitate negative cât şi pozitive şi s-a acordat o atenţie deosebită rezolvării conflictelor prin reguli bazate pe priorităţi.

In [DDL+01], [D02] se descriu extensiile care includ gruparea rolurilor şi suportul pentru delegarea autoritatii.

S-a ajuns la concluzia că specificarea politicilor de securitate pentru obiecte individuale este ineficientă şi, pentru a rezolva această problemă, s-au introdus domeniile. Domeniile grupează obiecte şi apartenenţa la un domeniu este explicită şi nu definită prin intermediul predicatelor.

Un punct tare al modelului Ponder provine din tipizarea obiectelor determinând o politică extensibilă şi flexibilă. Un exemplu de politică dezvoltat de acest colectiv si descris printr-un limbaj de specificare a politicilor este ilustrat mai jos:

```
inst auth+ fileServerAccess {
subject    /Employees;
target     Servers/PrinterServer;
action     *;                    // permite orice actiune
when       Time.after(''1300''); // constrangere
}
```

Ponder suportă trei tipuri de constrângeri. Constrângerile subiect/ţintă se referă la atributele obiectelor subiect sau ţintă. Constrângerile parametrului acţiune/eveniment sunt expresii care utilizează parametri sau evenimente fundamentale politicii. In final, sunt suportate constrângerile de timp prin utilizarea bibliotecilor temporale.

2.5.3 Colectivul de la Western Ontario University, Canada

Nyanchama şi Osborn au introdus modelul rolurilor sub forma de graf în [NO94], în care se propune o modalitate de administrare a rolurilor utilizându-se teoria grafurilor. Modelul lor defineşte rolurile ca şi un set de privilegii determinate. Din cauză că în cadrul acestui model nici rolurile şi nici privilegiile nu sunt parametrizate, este relativ simplă determinarea relaţiilor dintre roluri. Pentru a realiza conectivitatea rolurilor în graf ei definesc *MaxRole* ca fiind rolul minim senior comun, şi *MinRole* ca fiind cel mai mare rol junior comun. Graful rolurilor se bazează aşadar pe o relaţie între rolurile junior şi senior comune şi este o latice limitată de *MinRole* şi *MaxRole*.

In [NO95] Nyanchama şi Osborn rafinează privilegiile pentru un model orientat obiect. Ei, de asemenea, iau în considerare implicaţiile apelurilor metodelor şi dependenţa între acestea.

In [NO99] Nyanchama şi Osborn îşi continuă munca axându-se pe introducerea managementului ierarhiei de roluri şi investighează constrângerile. Ei clasifică constrângerile în cinci tipuri (utilizator-grup, rol-rol, privilegiu-privilegiu, asignarea utilizator-rol şi asignarea rol-privilegiu). Aceste categorii sunt o expresie a separării statice a îndatoririlor.

In [O97] se tratează proprietățile MAC pentru modelul rolurilor tratate prin teoria grafurilor. Sunt considerate obiecte şi subiecte cu asignări de etichete de securitate şi se tratează modul în care se pot construi ierarhiile de roluri. Acest efort este extins [O02] prin introducerea de grafuri de fluență din grafurile de roluri.

2.5.4 Colectivul de la Universitatea din Roma, Italia

Giuri defineşte rolurile ca un set de domenii protejate [G95], [GI97]. Un domeniu protejat este un set de privilegii, şi identifică setul tuturor operațiilor care pot fi efectuate de către un subiect la un anumit moment de timp. Domeniile protejate au fost introduse pentru a suporta separarea dinamica a îndatoririlor cum ar fi constrângerea la nivelul privilegiilor (de exemplu interzicerea ca un subiect să aibă privilegii conflictuale în acelaşi timp).

Rolurile au fost definite ca şi un set de privilegii şi se suportă ierarhiile de roluri.

Pentru a se implementa separarea îndatoririlor, limbajul propus permite *and-roles* şi *or-roles*. Rolurile *and-roles* specifică un set de privilegii şi roluri care pot fi activate simultan, în timp ce rolurile *or-roles* definesc roluri mutual exclusive.

Modelul lor de bază (Named Set of Protection Domain) nu suportă constrângerile ci doar o extensie [G95] a acestora.

In [G98] Giuri şi Iglio extind modelul propriu pentru a permite controlul accesului bazat pe conținut. Ei au introdus privilegii restrictive care conțin expresii logice care trebuie satisfăcute la momentul în care este utilizat un privilegiu. In acest model ei introduc template-uri de roluri care sunt practic roluri parametrizate.

In [G98] Giuri analizează suportul politicii de securitate propuse în cadrul unei aplicații Java şi cum poate fi extinsă aceasta politica pentru a suporta controlul accesului bazat pe roluri.

2.6 Standardizări în controlul accesului

Modelele RBAC sunt luate ca şi abordare generală pentru controlul accesului.

Standardizarea RBAC (standard dezvoltat de NIST) este organizată în două părți mari: un *Model de referință* şi *Specificația Funcțională* [HTTP4]. Modelul de referință este definit ca şi o colecție de elemente de bază (ex.: utilizatori, roluri, permisiuni, operații şi obiecte) şi relații (tipuri de funcții incluse în cadrul acestui standard). NIST defineşte de asemenea şi scopul caracteristicilor RBAC incluse în standard. Acestea acoperă diferite aspecte: ierarhii de roluri, constrângeri statice SSD şi dinamice DSD. Modelul de referință mai defineşte şi un limbaj în termeni de seturi de elemente şi funcții.

Dar nu toate caracteristicile standard sunt încapsulate în dezvoltările existente ci există diverse variante care utilizează anumite seturi ale acestor caracteristici.

Standardizarea RBAC a atras un interes mărit. Astfel, rolurile au fost luate în considerare începând cu elaborarea SQL3 şi a Oracle versiunea 7 în cadrul sistemelor de gestionare a bazelor de date. Acestea (rolurile) au fost de asemenea încorporate în diverse produse comerciale fie în mod direct fie prin concepte precum "grup de utilizatori". Rolurile mai sunt de asemenea utilizate pentru administrare în sisteme de operare şi în rețele de calculatoare precum Windows NT a Microsoft şi NetWare de la Novell [HTTP5].

3 Modelul SCAR-ACE pentru controlul accesului bazat pe roluri

3.1 Definiții şi termeni utilizați

Definiția 1 (securitate): Securitatea în aplicațiile software se referă în principal la autenticitatea, integritatea şi confidențialitatea datelor, informațiilor şi serviciilor. In mediile actuale aceste caracterisitici sunt oferite din ce în ce mai mult, în mod particular şi în sistemele deschise şi distribuite [TS02]. In mod corespunzător, orice mecanism de securitate trebuie să implementeze proprietățile caracteristice unui asemenea sistem. In principal există două categorii de mecanisme de securitate: (i) protocoale de criptare şi (ii) monitorizarea - ce include controlul accesului.

Definiția 2 (criptografia): Criptografia este tehnologia de asigurare a controlului accesului bazată pe chei asimetrice publice şi private.

Definiția 3 (controlul accesului): Controlul accesului este tehnica inițiată pentru protecția sistemelor centralizate [SC01] care sunt conectate prin canale sigure pentru înregistrarea utilizatorilor şi care sunt gestionate de către un administrator. Recent s-au întreprins cercetări pentru adaptarea controlului accesului şi în sistemele distribuite şi deschise [HTTP06], [LPL+03].

Definiția 4 (sistem de control al accesului): Un sistem de control al accesului este un sistem stare-tranziție $<\Gamma, Q, \vdash, \Psi>$, în care Γ este un set de stări, Q este un set de interogări, $\vdash: \Gamma x Q \rightarrow \{true, false\}$ se numeste relație de necesitate, iar Ψ este setul de reguli de tranziții între stări. O *stare* $\gamma \in \Gamma$ conține toate informațiile necesare pentru luarea unei decizii în controlul accesului la un moment dat. *Relația de necesitate* $\vdash$, determină dacă există o interogare dintr-o anumită stare. Dacă există o interogare, $q \in Q$ pentru o cerere de acces, $\gamma \vdash q$ înseamnă că accesul lui q este permis din starea γ. O *regulă de stare-tranziție* $\psi \in \Psi$ determină modul în care sunt schimbate stările [TL04].

Definiția 5 (utilizator): Un utilizator este o instanță a unui client al sistemului SCAR-ACE.

Definiția 6 (rol): Un rol într-o aplicație de comerț electronic poate reprezenta competențe concrete precum cele ale unui vânzător, cumpărător sau administrator de sistem. Un rol poate încapsula *autoritatea* şi *responsabilitatea* unui actor. Autoritatea şi responsabilitatea sunt diferite de competență: o persoană poate să fie competentă să conducă câteva departamente dar are responsabilitatea de a conduce doar unul. Rolurile pot reflecta asignarea unor îndatoriri. Un *rol administrativ* este un rol care include permisiunea de a modifica setul de utilizatori, rolurile sau permisiunile, sau să modifice asignarea rolurilor la utilizatori.

Definiția 7 (ierarhia de roluri): Ierarhia de roluri este o relație parțial sau total ordonată între roluri. Moştenirea în cadrul unei ierarhii de roluri implică moştenirea permisiunilor rolurilor situate la nivel superior în cadrul ierarhiei.

O ierarhie este o structură de roluri care reflectă autoritatea şi responsabilităţile în cadrul unei organizaţii. Prin convenţie rolurile cu mai multă autoritate (roluri senior) sunt reprezentate în partea de sus a ierahiei iar cele cu mai puţină autoritate (roluri junior) în partea de jos a diagramei.

Definiţia 8 (rol privat): In general, rolurile private sunt roluri maxime într-o ierarhie.

Definiţia 9 (grup): Grupul este un set de utilizatori care fac parte din aceeaşi categorie de roluri.

Definiţia 10 (permisiune): O permisiune este aprobarea de acces la una sau mai multe resurse ale sistemului. Termenii de *autorizare, drepturi de acces şi privilegii* sunt de asemenea utilizaţi în conjunctie cu cel de permisiune.
Permisiunile sunt în general pozitive şi conferă proprietarului posibilitatea de a efectua o acţiune în cadrul sistemului. In literatură se întâlneşte şi termenul de permisiune negativă – cea care împiedică accesul [SCH+96]. In SCAR-ACE se consideră că împiedicarea accesului este mai degrabă o constrângere decât o permisiune negativă.
Modelul conceptual al SCAR-ACE acceptă mai multe interpretări pentru permisiuni, de la faptul că este permis accesul la un set de servicii, la cel în care unitatea de acces este un câmp al unei înregistrări.
Fiecare tip de sistem isi protejează resursele; de exemplu un sistem de operare îşi va proteja fişierele, directoarele, dispozitivele, porturile prin operaţii precum citire, scriere, execuţie iar un sistem DBMS îşi va proteja relaţiile, tuplele, atributele şi vederile prin operaţii precum select, update, delete şi insert.
Permisiunile se pot aplica fie unuia fie mai multor obiecte şi pot fi extrem de specifice precum permisiunea de citire a unui fişier particular, sau permisiunea de citire a tuturor fişierelor aparţinând unui departament. Maniera în care permisiunile individuale sunt reunite într-o permisiune generică care poate fi atribuită unei entităţi depinde de implementare.

Definiţia 11 (privilegiu): Privilegiile sunt permisiunile asignate unui rol.

Definiţia 12 (resurse): Resursele sunt fişiere, conexiuni în reţea, servicii, sau metode ale obiectelor.

Definiţia 13 (sesiunea): Sesiunea este o mapare între un utilizator şi subsetul de roluri care îi sunt asignate, şi este activă un anumit interval de timp după care expiră. Un utilizator stabileşte o sesiune în timpul căreia îşi activează un subset al rolurilor al căror membru este (membru direct sau indirect prin intermediul ierarhiei de roluri). Permisiunile acordate utilizatorului sunt formate din reuniunea tuturor permisiunilor rolurilor activate în cadrul acelei sesiuni. O sesiune este asociată unui singur utilizator iar un utilizator poate avea active mai multe sesiuni concomitent.

Definiţia 14 (constrângere): O constrângere este o limitare şi se poate aplica următoarelor componente: rol, ierarhie de roluri, sesiune, grup. Exemple de constrângeri pot fi considerate rolurile mutual exclusive precum un vânzător şi un cumpărător al aceluiaşi produs.

Definiţia 15 (cerere): O cerere este interogarea de accesare a unei resurse.

Definiţia 16 (autorizator, autentificare): Autorizatorul este entitatea care poate să acceseze o resursă. În mod tradiţional, când un autorizator primeşte o cerere, prima dată identifică emiţătorul acesteia. Autentificarea este acţiunea de a determina identitatea emiţătorului într-o manieră unică. Cu alte cuvinte, autentificarea răspunde întrebării "cine a realizat această cerere?".

Definiţia 17 (autorizare): Autorizare este procesul prin care se realizeaza autentificarea si controlul accesului. In mod tradiţional, autorizarea informaţiei este realizată de către un serviciu. In cadrul aplicatiilor deschise si distribuite creearea şi gestionarea informaţiilor de autorizare se realizeaza în mod dinamic.

Definiţia 18 (Politica de securitate): Politica de securitate a unui sistem desemnează privilegiile ce trebuie acordate utilizatorilor şi condiţiile acordării acestora. In contextul controlului accesului, *politicile de securitate* sunt declaraţii relativ la condiţiile puse pentru accesul la date, informaţii sau servicii, respectiv se specifică pentru fiecare rol tipul de date şi servicii la care are acces.
Pentru politica conceptuală de securitate [BW04] introduce noţiunea de *algebră* a politicii. Aceasta oferă operatori de nivel înalt, orientaţi pe seturi, precum: *enumerare, adunare, scădere*, *selecţie*.

Definiţia 19 (relaţie): O relaţie între un subiect şi un domeniu se exprimă în termenii acţiunilor permise sau interzise, acţiuni care trebuie sau nu efectuate. De exemplu relaţiile între roluri stabilesc modul de propagare a permisiunilor între acestea (între roluri). Noţiunea de relaţie este în conexiune cu cea de constrângere pentru definirea politicii de activare a permisiunilor.

Definiţia 20 (identitate): În sistemele tradiţionale identitatea adesea reprezintă un cont utilizator cunoscut înaintea emiterii oricărei cereri. Pentru aplicaţiile Internet, noţiunea de identitate devine problematică, termenul însemnând "cineva" (un utilizator). Când se întâlneşte un utilizator necunoscut anterior, acesta este asociat în SCAR-ACE cu un vizitator.
Într-un scenariu în care un autorizator şi o cerere nu au relaţii anterioare, cunoaşterea numelui sau identităţii emitentului nu poate ajuta autorizatorul să ia o decizie. Adevărata proprietate necesară cuiva pentru identificare este un credenţial care conţine mai mult decât identificarea utilizatorului.

Definiţia 21 (credenţial): *Credenţialele* sunt părţi digitale similare unei chei publice şi au anumite proprietăţi [BK03] care se referă la aspecte de identitate sau non-identificative ale unui utilizator (precum aptitudini sau profesia). Emiţătorul credenţialului este responsabil de corectitudinea aserţiunilor. Cand se inspectează un credenţial se verifică semnătura şi încrederea în emiţătorul acestuia. Credenţialele se pot utiliza în sistemele distribuite şi deschise.
In SCAR-ACE un credenţial conţine informaţii de determinare a identităţii emiţătorului unei cererii pentru accesarea unei resurse cat si informatii despre tranzitia de efectuat. Credenţialul este compus din identificator utilizator, rol, stare curentă şi eveniment declanşator al cererii. Toate aceste date formează un credenţial necesar acordării permisiunii de a accesa o resursă.

Definiția 22 (RBAC – Role Based Access Control): Conceptul RBAC se referă la o tehnologie care încapsulează termenii de roluri şi ierarhii de roluri, activarea rolurilor, şi constrângeri asupra rolului per utilizator. Este definit prin următoarele componente: nucleul RBAC, ierarhiile RBAC, relații statice şi dinamice şi se detaliază în 4 modele $RBAC_0$, $RBAC_1$, $RBAC_2$ şi $RBAC_3$ (vezi 3.2 şi 3.3).

3.2 Componentele modelului

Componentele de bază ale modelului standard RBAC de control al accesului bazat pe roluri sunt [FSG01]:

1. Nucleul RBAC;
2. Ierarhiile RBAC;
3. Relații statice;
4. Relații dinamice;
5. Extensii.

Modelului SCAR-ACE realizează abordarea componentelor RBAC într-o manieră originală astfel încât să fie soluționate aspectele legate de politica de securitate prin utilizarea unor maşini de stare.

3.2.1 Nucleul

Nucleul modelului SCAR-ACE este similar nucleului RBAC. Nucleul RBAC încapsulează aspectele esențiale ale modelului: unui utilizator i se asignează un rol şi utilizatorul primeşte permisiuni ca urmare a fapului că este membru al acelui rol. Relațiile utilizator-rol şi rol-permisiuni au rangul n:m (many-to-many). Astfel, aceluiaşi utilizator i se pot asigna mai multe roluri şi un rol poate fi asignat mai multor utilizatori. Nucleul mai include de asemenea şi conceptul de sesiune utilizator care permite activarea / dezactivarea rolurilor.

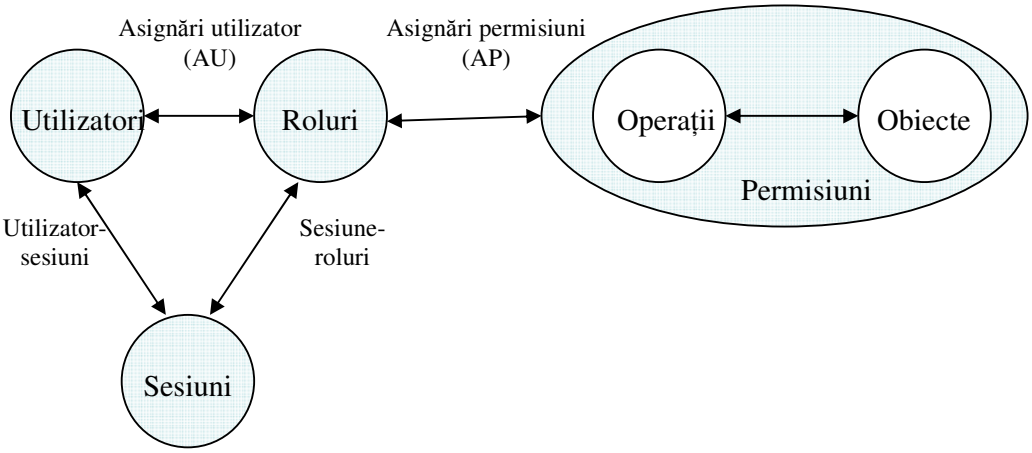

Figura 2. Nucleul RBAC

Una dintre cerințele RBAC este aceea ca utilizatorilor să le fie permisă exercitarea mai multor roluri.

Setul de elemente şi relațiile nucleului RBAC sunt definite în Figura 2. Nucleul RBAC include un set de şase elemente de bază şi anume: utilizatori, roluri, sesiuni, obiecte, operații şi permisiuni. Modelul RBAC este definit prin utilizatori individuali cărora li se asignează roluri şi prin roluri carora li se asignează permisiuni. Astfel, relațiile utilizatori-roluri şi roluri-permisiuni sunt relații many-to-many. Prin asignările AU (utilizator-rol) şi AP (rol-permisiuni) prin tranzitivitate, un utilizator are anumite permisiuni. Figura 2 ilustrează şi relațiile RBAC de asignare a utilizatorilor (AU) şi de asignare a permisiunilor (AP).

Modelul nucleului RBAC include un set de sesiuni unde fiecare sesiune este o mapare între un utilizator şi un subset activat de roluri pentru utilizatorul respectiv.

Nucleul modelului SCAR-ACE este similar nucleului RBAC şi conține urmatoarele elemente: utilizatori, roluri, sesiuni, permisiuni, operații şi obiecte, masina de stare şi relațiile dintre aceste elemente (vezi figura 4).

Un *utilizator în* SCAR-ACE este definit ca şi o persoană cu toate că acest concept poate fi extins pentru a include calculatoare, rețele sau agenți inteligenți autonomi.

Un *rol* este o funcție în contextul sistemului distribuit, cu o semantică asociată, care conferă autoritate şi responsabilitate utilizatorului căruia i s-a atribuit acel rol (de exemplu de cumpărător sau vanzător, recepționer sau distribuitor).

O *permisiune* este aprobarea de a efectua o operație asupra cel puțin unui obiect SCAR-ACE.

O *operație* este forma executabilă a unui modul program sau a unui set de metode care, dupa invocare, efectuează un anumit serviciu pentru utilizator. Tipurile de operații şi obiecte controlate de modelul SCAR-ACE depind de nivelul de detaliu luat in considerare. De exemplu, la nivel de sistem de fişiere, operațiile pot include citirea, scrierea şi execuția; la nivel de sistem de gestionare a bazelor de date, operațiile pot consta din insert, delete, append şi update iar pentru un sistem de comerț electronic pot fi vizualizare catalog de produse, selectare produs, adaugare produs în coş, plata produs, etc.

Un *obiect* este o entitate care conține sau primeşte informații. Pentru modelul SCAR-ACE, obiectele pot fi containere de informații (de exemplu cataloage sau maşini de stare).

Scopul mecanismului de control al accesului este protejarea resurselor sistemului mai precis de protejare a obiectelor şi operațiilor.

O *sesiune* SCAR-ACE este o mapare dintre un utilizator si mai multe roluri, adică un utilizator stabileşte o sesiune în cursul căreia el este asignat cel putin unui rol, rol care activează un anumit subset al operațiilor, operații specifice permisiunilor sale. Fiecare utilizator este asociat unei singure sesiuni deci este o relație de 1:1 între sesiune şi utilizator. Funcția *sesiune-roluri* ilustrează rolurile activate în cadrul unei sesiuni iar relația dintre sesiune şi roluri este una de n:m.

Permisiunile unui utilizator sunt permisiunile asignate rolurilor activate de către utilizator în timpul unei sesiuni.

Fiecare rol are ataşată o *maşină de stare* specifică doar modelului SCAR-ACE. Relația dintre maşina de stare şi roluri este de 1:1 adică fiecărui rol îi corespunde una şi numai una dintre maşinile de stare. Relația dintre maşina de stare şi permisiuni este de 1:n în sensul că unei maşini de stare îi corespund mai multe permisiuni, prin tranzitivitate cele specifice rolului pe care îl descrie.

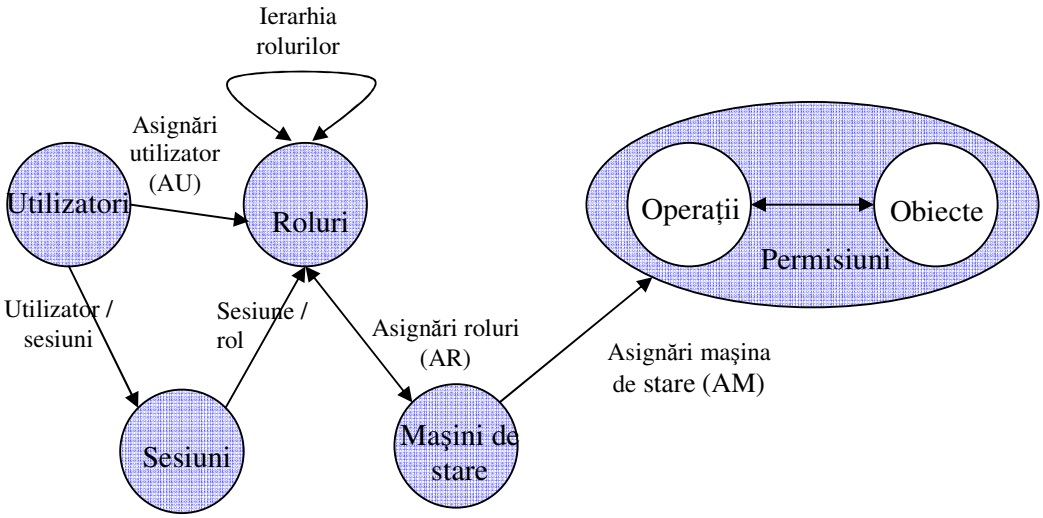

Figura 3. Componentele modelului SCAR-ACE

Accesul unui rol la un set de permisiuni nu poate avea loc decât prin intermediul maşinii de stare aceasta fiind o diferentă majoră față de modelele existente.

Relațiile dintre elemente fac parte din model, unele dintre acestea fiind diferite în SCAR-ACE față de RBAC. Modelul SCAR-ACE impune constrângeri (figura 3) care restricționează următoarele relații:

- relațiile de tip AU (Asignări Utilizator): la un moment dat un utilizator nu poate deține decât un singur rol, relație de 1:1 (similar RBAC);
- relațiile de tip AR (Asignări Roluri) care asignează un rol unei maşini de stare, relație de 1:1 (diferit față de RBAC).
- Relații de tip AM (Asignări Maşini de stare), o maşină de stare are asociate mai multe permisiuni, relație de 1:n (diferit față de RBAC);
- relațiile sesiune – utilizator care sunt relații de tipul 1:1 (similar RBAC);
- relațiile sesiune-rol care sunt relații de tipul 1:n (similar RBAC).

3.2.2 Ierarhiile de roluri

O ierarhie a rolurilor este din punct de vedere matematic o ordine parțială sau totală care defineşte relații între roluri, unde rolurile senior dobândesc permisiunile celor junior [FSG01]. Ierarhiile sunt structuri de roluri care reflectă nivelul autorității şi al responsabilității.

Conceptul de ierarhie a rolurilor este introdus atât în modelul RBAC precum şi în unele dintre implementările acestuia pentru îmbunătățirea eficienței şi descrierea structurilor organizaționale.

Există mai multe tipuri de ierarhii de roluri [SFK00]:

- ierarhii generale. În acest caz există o ordine parțială între roluri care serveşte drept ierarhie şi include moştenirea multiplă a permisiunilor; este definită relația de membru a unui utilizator la un grup de roluri;
- ierarhii limitate. In acest caz se introduc restricții în cadrul ierarhiei rolurilor. Cel mai adesea ierarhiile sunt limitate la structuri simple precum arbori sau arbori inverşi.

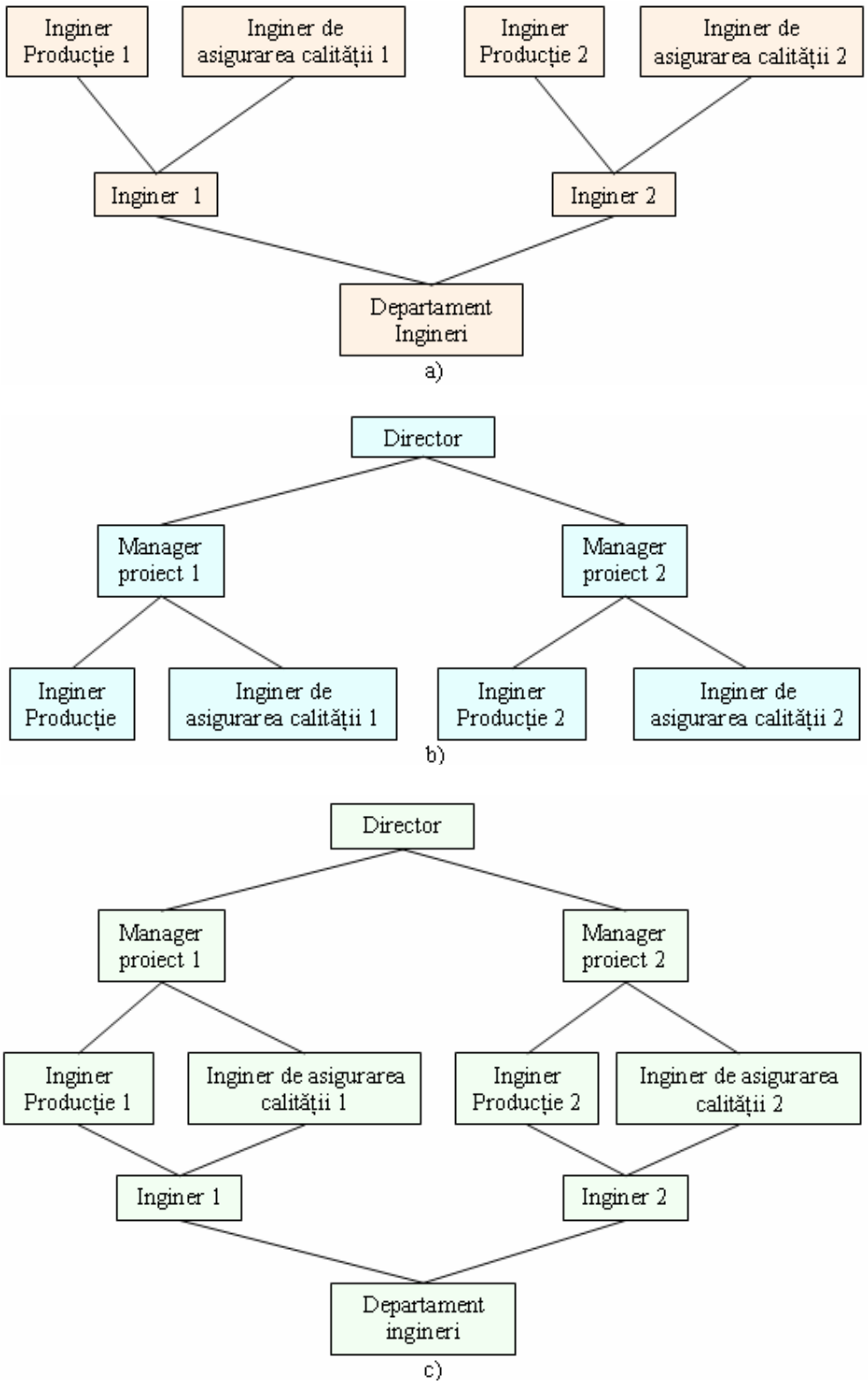

Figura 4 Exemple de ierarhii de roluri: a) arbore; b) arbore invers; c) latice

Justificarea cerințelor de tranzitivitate, reflexivitate şi ale proprietăților de asimetrie pentru ierarhiile limitate este discutata pe larg în literatura de specialitate

[FCK95], [SCH+96], [M98]. Oricum există un număr mic de produse care suportă ierarhiile limitate. In figura 4 sunt exemplificate câteva tipuri de ierarhii.

Ierarhiile de roluri sunt în mod uzual incluse ca şi componente cheie ale modelelor RBAC [FCK95] si reprezinta structuri în care fiecare rol reflectă o poziție organizațională a autorității şi responsabilității proprii.

După cum am mai amintit, ierarhiile de roluri definesc relațiile de moştenire între roluri. Moştenirea se referă la permisiuni astfel: dacă rolul "r_1" moşteneşte drepturile rolului "r_2" toate drepturile lui "r_2" sunt şi ale lui "r_1".

Standardul RBAC NIST [FSG01], [FGL95] recunoaşte atât ierarhiile generale cât şi ierarhiile limitate de roluri. Ierarhiile generale includ conceptul de moştenire multiplă a permisiunilor.

Ierarhiile limitate impun restricții, rezultând o structură mai simplă, de arbore (vezi figura 3) în care un rol poate avea unul sau mai mulți ascendenți imediați dar este restricționat la un singur descendent. De notat că permisiunile sunt transmise de jos în sus în cadrul unei ierarhii şi, prin urmare, ierarhiile generale permit moştenirea multiplă iar ierarhiile limitate permit moştenirea simplă.

In exemplul considerat în figura 3, în ierarhia a), *Inginer 1* moşteneşte permisiunile de la *Departamentul de Ingineri*, la care se adaugă permisiuni proprii specifice. Un *Inginer Producție 1* moşteneste permisiunile de la *Inginer 1*, permisiunilor sale adăugându-li-se şi unele specifice. Aceasta ierarhie este o ierarhie de tip arbore şi este conform definiției o ierarhie limitată.

In figura 3, în ierarhia b), rolul de *Director* moşteneşte permisiuni de la mai mult de un rol junior (de la *Manager 1* şi *Manager 2*), este o ierarhie de tip arbore invers şi conform definiției este o ierarhie generală.

Ierarhia din figura 3 c) este o ierarhie latice şi este tot o ierarhie generală deoarece există moştenire multiplă.

Ierarhia SCAR-ACE este o ierarhie limitată de tip arbore (vezi figura 3 a)). Rolurile în ierarhie sunt restricționate la un singur descendent imediat şi la un singur ascendent imediat (fiecare rol senior are un singur rol junior şi fiecare rol junior are un singur rol senior).

Fiecare rol din cadrul ierarhiei de roluri are ataşată o maşină de stare specifică rolului, accesul unui rol la permisiuni realizându-se doar prin intermediul maşinii de stare corespunzătoare. Dintr-un rol junior (oricare ar fi starea de autorizare a acestuia) se poate trece în oricare rol senior din cadrul ierarhiei de roluri doar prin autentificare. Trecerea de la un rol senior la un rol junior din cadrul ierarhiei se realizează fără autentificare.

3.2.3 Relațiile statice

Relațiile RBAC sunt relații de separare a îndatoririlor şi impun constrângeri asupra modelului. Relațiile de separare a îndatoririlor sunt utilizate pentru a evita conflictul de interese în cadrul unei organizații (pentru a preveni un utilizator de a exceda nivelul de autorizare corespunzător poziției sale).

Ca şi principiu de securitate, separarea îndatoririlor a fost recunoscută pentru larga sa aplicabilitate în business, industrie şi altele [BN89], [CW87]. Pentru a minimiza posibilitatea erorilor intr-o organizatie, indivizilor cu atribute diferite sau interese divergente li se asignează roluri şi permisiuni diferite.

Standardul NIST RBAC permite atât relații statice cât şi dinamice (vezi şi 3.2.4).

Relațiile statice se referă la separarea statică a îndatoririlor şi sunt utilizate pentru evitarea conflictelor în autorizare. Un conflict de interese într-un sistem bazat pe roluri poate apărea ca rezultat al obținerii autorizării de către un utilizator pentru permisiuni asociate cu roluri conflictuale (de exemplu vânzător şi cumpărător al aceluiaşi produs). O posibilitate de evitare a acestor conflicte de interese este prin separarea statică a îndatoririlor (SSD) [K97]. Un exemplu de SSD poate fi o constrângere care impune ca două roluri să fie mutual exclusive; de exemplu dacă un rol necesită cheltuieli şi altul le aprobă, organizația poate să interzică atribuirea ambelor roluri aceluiaşi utilizator.

Relațiile SSD furnizează un mijloc pentru evitarea conflictelor de interese. Constrângerile statice pot lua o varietate de forme. Un exemplu uzual este definirea asignărilor disjuncte relativ la un set de roluri [GI96], [K97]. [GGF98], [SZ83], [AS00] [JT00] au identificat relații SSD pentru includerea constrângerilor asupra utilizatorilor, operațiilor şi obiectelor precum şi a diverselor combinații între acestea.

Modelul SCAR-ACE conține relații SSD şi implementarea acestora are loc prin utilizarea unei maşini de stare pentru fiecare rol. Dacă există stări prin care un utilizator (prin intermediul unui rol) are acces la resurse, pentru realizarea relațiilor SSD maşina de stare va încapsula doar acele stări şi tranziții care sunt permise rolului în cauză.

3.2.4 Relațiile dinamice

Relațiile dinamice DSD, similar celor statice, limitează permisiunile utilizatorilor. Relațiile DSD diferă de cele SSD prin contextul în care aceste limitări sunt impuse. Cerințele DSD limitează drepturile prin plasarea constrângerilor asupra rolurilor care pot fi activate în timpul unei sesiuni utilizator (cu alte cuvinte, in mod dinamic, la rulare).

Relațiile DSD definesc constrângerile ca şi o pereche:

$$R_{DSD} = \{ \bigcup \ a \mid a = (rk, n, ss), rk \in R, n \geq 2, ss \in SS \}$$

unde:

n: un număr natural mai mare sau egal cu 2 cu proprietatea că în nici o sesiune *ss* din *SS* nu pot fi activate simultan *n* sau mai multe roluri *rk* din setul de roluri *R*.

Proprietățile DSD furnizează o extensie pentru principiul privilegiilor minime deoarece un utilizator are nivele diferite de permisiuni la momente diferite, în funcție de sarcina efectuată. Aceasta caracteristica este întâlnita sub denumirea de revocarea temporară a încrederii [K97].

Relatiile SSD se referă la capacitatea de adresare a unui conflict de interese în momentul în care se asignează un rol unui utilizator. DSD permite unui utilizator să fie autorizat pentru un rol care nu cauzează un conflict de interese când acesta se activează independent, dar care produce conflict când este activat simultan de mai mulți utilizatori.

Relaţiile DSD în modelul SCAR-ACE sunt realizate prin restricţia impusă la asignarea rolurilor. Astfel nu pot fi activate simultan de diverşi utilizatori anumite roluri precum cel de administrator, recepţioner al unei cereri sau distribuitor al unui produs. Pentru a realiza relaţiile DSD, modelul SCAR-ACE utilizează tot maşini de stare şi anume: la un moment dat un utilizator face parte din una şi numai una din maşinile de stare şi anume cea care corespunde rolului său. Fiecare maşină de stare este gestionată de un manager care conţine şi numărul de utilizatori care accesează simultan un rol. In cazul în care există restricţii de cardinalitate, acestea sunt rezolvate de către managerul maşinii de stare.

3.2.5 Extensiile

In afara componentelor amintite mai sus, există şi alte caracteristici specifice anumitor modele, caracteristici denumite extensii ale modelului RBAC.

Delegarea

Delegarea permite unui utilizator să imputernicească alţi utilizatori cu o parte a drepturilor sale. Motivarea vine din lumea reală, unde există utilizatori care necesită să acţioneze în favoarea altor utilizatori pentru accesarea resurselor, în cazul în care, de exemplu, se îmbolnăvesc. Există o cercetare intensă în acest domeniu din care amintim [BS[1]00], [BS[2]00], [NC00], [ZAC01]. Anumiţi cercetători consideră delegarea parţială în care doar privilegiile sunt delegate, sau când rolul delegat este cu constrângeri. Diferite tipuri de delegare sunt descrise în [BS[1]00].

Noţiunea de delegare este strâns interconectată cu cea de revocare, care permite utilizatorilor delegaţi să revoce un anumit rol, sau permite rolurilor să fie revocate utilizând constrângeri de timp. Arhitecturile care suportă delegarea şi revocarea diferă prin modul în care acestea au fost implementate.

Parametri

Anumite modele RBAC includ parametri pentru roluri şi/sau privilegii. Rolurile parametrizate pot fi rafinate în timpul activării prin setarea valorii parametrilor. De notat că valoarea acestor parametri este fixă si nu poate fi modificată ulterior. Parametrii privilegiilor, similar parametrilor rolurilor, primesc valori la crearea instanţei; consecvent, parametrii setaţi ai unui privilegiu nu mai pot fi modificaţi. Astfel de privilegii conţin informaţii despre obiectul accesat.

In timp ce parametrii introduc o modalitate de abstractizare a rolurilor şi privilegiilor, aceştia pot crea probleme pentru ierarhiile de roluri. Daca parametrii rolurilor trebuie setaţi intr-o ierarhie, aceasta putand reprezenta o problemă dacă parametrii rolului părinte variază semnificativ faţă de cei ai rolului fiu. In astfel de cazuri relaţia de moştenire trebuie să includă informaţii de setare a parametrilor.

Constrângerile devin, prin suportarea parametrilor, mai complexe [J99]. In mod corespunzător analiza relaţiilor statice devine mai dificilă. De exemplu, parametrii complică verificările de cardinalitate deoarece doar identificatorii rolurilor nu mai sunt suficienti pentru distingerea acestora, necesitându-se memorarea unei cantităţi mai mari de informaţie de stare.

Dependența de context

Cerințele pentru limbaje de specificare a controlului accesului sunt în creştere. Aceasta a determinat sa fie necesară considerarea contextului în care se realizează deciziile de control al accesului [CLS+01]. Suportul pentru dependența de context poate conține de la evaluarea predicatelor temporale [BBF01] până la predicate complexe care furnizează informații despre sistemul de operare [GMS94] sau poate permite evaluarea predicatelor arbitrare [BMY02]. Această extensie a RBAC permite restricții suplimentare adăugate rolurilor, restricții care pot fi temporale sau de acces a bazelor de date.

Scalabilitatea

Scalabilitatea este un factor important în cadrul sistemelor de control al accesului. Un sistem RBAC centralizat s-ar putea să nu respecte cerințele unei organizații mari dispersate geografic; astfel este esențială considerarea sistemelor distribuite.

Un sistem distribuit introduce multe provocări. Pentru a aduce suport utilizatorilor care accesează resurse din diferite locații, majoritatea implementărilor lucrează cu sesiuni. Aceasta complică verificarea constrângerilor: de exemplu este mai dificil de a obține informații relativ la rolurile active ale unui utilizator. Aceste sisteme trebuie, de asemenea, să fie pregatite să controleze eventualele eşecuri de comunicare in rețea. Ca o consecință, politica trebuie extinsă pentru a permite specificații de deteriorare a condițiilor de rulare. Datorită acestor constrângeri există diferite extensii ale modelelor RBAC.

Politici obligatorii

Majoritatea modelelor RBAC sunt permisive, adică specifică ceea ce poate un subiect să facă. In contrast, politicile obligaționale [LS97], care îşi au originea în cerințele de management al aplicatiilor, specifică ceea ce trebuie sau nu trebuie să facă un subiect asupra unor obiecte.

Un exemplu de limbaj RBAC care suportă obligativitatea este Ponder [LS97]. Obligațiile lucrează bine cu alte caracteristici RBAC, de exemplu cu ierarhiile şi contextul. Limbajul propus de Schaad et al. [SM02] propune investigarea integrării obligativității cu extensiile RBAC care permit delegarea.

Politici negative

Permisiunile specificate în majoritatea modelelor RBAC sunt pozitive adică se specifică ce poate un utilizator să facă. In asemenea medii, politica implicită refuză accesul dacă acesta nu este permis explicit.

Pe de alta parte, permisiile negative sunt opusul celor pozitive; acestea specifică ceea ce un utilizator nu poate să facă. Dacă sunt specificate doar politicile negative, controlul accesului va permite tot ceea ce nu este în mod explicit interzis.

Un model poate include atât politici pozitive cât şi negative, un exemplu regăsindu-se în [L98]. Utilizarea ambelor politici poate duce însă la conflicte, anumite acțiuni putând fi atât permise cât şi interzise în acelasi timp.

Roluri relaționate cu poziția versus roluri relaționate cu taskurile

Rolurile grupează atât privilegii cât şi utilizatori. In funcție de abordare, rolurile pot fi grupate în mai multe moduri. In [SBC+97] sunt descrise trei tipuri de entitati: abilități, grupuri, şi roluri UP. Abilitățile sunt roluri care pot avea doar privilegii ca şi membri, şi nu pot grupa utilizatorii. Grupurile, pe de alta parte, colectează utilizatori dar nu consideră în mod implicit privilegiile de acordat. Rolurile UP au atât utilizatori cât şi privilegiile asociate acestora.

O altă clasificare a rolurilor este bazată pe corespondența din lumea reală a acestora [NS02]. Conform acestei clasificări un rol poate fi funcțional sau organizațional. Un rol funcțional grupează privilegii corespunzatoare unei funcții. De exemplu rolul de *database_user* este un rol funcțional. Un avantaj al rolurilor funcționale este că au o durabilitate mai mare în comparație cu rolurile organizaționale. Un rol organizațional reflectă poziția unei persoane în cadrul ierarhiei de roluri din companie. De exemplu, un astfel de rol este cel de *manager*. RBAC suportă rolurile organizaționale prin ierarhii.

Când RBAC nu este suficient

Toate aceste extensii pot crea senzația că RBAC este un model foarte expresiv, dar există cazuri când este destul de dificil de a exprima lumea reală cu ajutorul rolurilor şi a modului de activare a acestora. Un exemplu poate fi o echipă care are un scop şi în care membrii echipei colaborează. In astfel de cazuri se determină alte tipuri de control al accesului precum controlul accesului bazat pe echipe [T97] sau pe taskuri [TS97].

Un alt scenariu în care RBAC nu este suficient este în cazul workflow-urilor. Pentru a lucra cu workflow-uri sistemul RBAC necesită considerarea informațiilor externe de exemplu stadiul actual al proceselor de business, cum ar fi extensiile propuse în [BFA97] şi [HA99].

3.3 Modele conceptuale

In aceasta sectiune analizez modelele conceptuale RBAC si, prin paralelism, modelele conceptuale SCAR-ACE.

3.3.1 Modele RBAC conceptuale

Pentru a analiza diferite versiuni RBAC s-a definit o familie de patru modele conceptuale (figura 5) [SCH+96]. $RBAC_0$ este modelul de bază şi este cerința minimă pentru un sistem de control al accesului bazat pe roluri. Modelele avansate $RBAC_1$ şi $RBAC_2$ includ modelul $RBAC_0$. $RBAC_1$ introduce ierarhiile de roluri (situații în care rolurile moştenesc permisiuni de la alte roluri) în timp ce $RBAC_2$ adaugă constrângeri (care impun restricții între componente). Modelul consolidat $RBAC_3$ include modelele $RBAC_1$ şi $RBAC_2$ şi, prin tranzitivitate, $RBAC_0$.

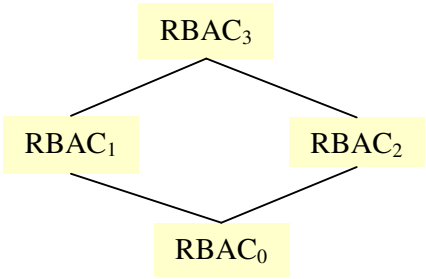

Figura 5. Modele RBAC conceptuale

In abordările RBAC se presupune că, pentru managementul sistemului, există un singur administrator autorizat să configureze diversele setări şi relaţii între componente. Aceasta nu elimină existența unui model de management şi configurare mai sofisticat.

Conceptul RBAC suportă următoarele principii de securitate:

- *Privilegiul minim*: unui rol ii sunt asignate doar acele permisiuni necesare pentru îndeplinirea sarcinilor;
- *Separarea îndatoririlor*: pentru îndeplinirea unei sarcini este necesară, în unele cazuri, invocarea cumulată a rolurilor mutual exclusive (exemplu: necesitatea participării atât a funcționarului cât şi a managerului unui cont în emiterea unui cec);
- *Abstractizarea datelor*: în locul permisiunilor tipice de citire / scriere dintr-o / intr-o bază de date, se pot stabili permisiuni abstracte precum cele de credit şi debit pentru un cont.

Emit două observaţii relativ la aceste principii:

1. RBAC nu impune modul de aplicare a acestor principii. Teoretic, un administrator de sistem poate configura RBAC astfel încât acesta să violeze aceste principii.
2. Nivelul de abstractizare a datelor este determinat de implementare.

3.3.2 Modele SCAR-ACE conceptuale

Modelul SCAR-ACE se referă la un $RBAC_3$ şi, pentru realizarea SSD şi DSD, impune constrângeri. Astfel pentru realizarea SSD se restricționează realizarea statică a maşinii de stare pentru fiecare rol iar pentru DSD se impun constrângeri de simultaneitate în asignarea rolurilor şi restricţii relativ la sesiuni multiple utilizator.

Deoarece $RBAC_3$ include toate modelele RBAC inferioare, prin paralelism, modelul SCAR-ACE este compus din $SCAR\text{-}ACE_0$, $SCAR\text{-}ACE_1$, $SCAR\text{-}ACE_2$ si $SCAR\text{-}ACE_3$. In continuare mă voi referi la toate nivelele conceptuale ale modelului SCAR-ACE şi la specificul acestora [$OG^2$07].

3.3.3 Modelul de bază $SCAR\text{-}ACE_0$

Modelul de bază $SCAR\text{-}ACE_0$ constă din următoarele entităţi: utilizatori (U), roluri (R), masini de stare (M), permisiuni (P) şi sesiuni (S).

In figura 4 sunt ilustrate entitatile mai sus amintite si urmatoarele relatii: *asignările utilizator (AU)*, *asignarile rolurilor (AR)* şi *asignarea maşinilor de stare*

(AM); asignarea utilizatori - roluri este o relaţie de tipul n:m în sensul că un utilizator poate deţine mai multe roluri intr-o sesiune şi un rol poate fi asignat mai multor utilizatori. Relaţia roluri-permisiuni este o relaţie de n:m, un rol având asignate mai multe permisiuni şi o permisiune putând fi asignată mai multor roluri. Această relaţie de roluri – permisiuni este implementată prin intermediul maşinilor de stare. Aşadar, prin tranzitivitate, un utilizator poate exercita permisiuni. Pozitia rolului ca şi intermediar în aceste relaţii permite unui utilizator să exercite permisiuni si furnizează un control mărit pentru controlul accesului.

Utilizatorii stabilesc sesiuni în timpul cărora îşi activează un subset al rolurilor pe care le deţin. Permisiunile disponibile unui utilizator reprezintă reuniunea permisiunilor tuturor rolurilor activate în cadrul acelei sesiuni. In SCAR-ACE_0 fiecare sesiune este asociată unui singur utilizator, această asociere rămânând constantă pe întreaga durată a sesiunii.

Modelul de baza SCAR-ACE_0 suportă principiul minimului privilegiu: un utilizator care deţine câteva roluri pe durata unei sesiuni poate invoca consecutiv orice subset al acestora pentru a permite realizarea unei sarcini. Astfel, un utilizator care poate deţine un rol autorizat poate să păstreze acest rol deactivat şi să îl activeze explicit doar atunci când are nevoie de acesta.

3.3.4 Definirea formală a modelului SCAR-ACE_0

In relaţie cu $RBAC_0$ am introdus definirea formală a modelului SCAR-ACE_0 care are următoarele componente:

- *U, R, P, S şi M* (multimile de utilizatori, roluri, permisiuni, sesiuni şi respectiv maşini de stare);
- *Relaţia utilizator – roluri:* $R_{UR} \subseteq U \times R$, o relaţie de asignare n:m pentru utilizatori-roluri
  $R_{UR} = \{ur \mid ur = (u, r), (\forall u \in U, \exists r \in R) \wedge (\forall r \in R, \exists u \in U)\}$;
- *Relaţia sesiune - utilizator*: $R_{SU} \subseteq S \rightarrow U$, o relaţie 1:1 care mapează fiecare sesiune unui singur utilizator (constant pentru timpul de viaţă al sesiunii)
  $R_{SU} = \{su \mid su = (s, u), (\forall s \in S, \exists ! u \in U)$;
- *Relaţia rol - maşina de stare:* $R_{RM} \subseteq R \rightarrow M$ o relaţie 1:1 care mapează fiecare rol la o singură maşină de stare
  $R_{RM} = \{rm \mid rm = (r, m), (\forall r \in R, \exists ! m \in M)$;
- *Relaţia maşina de stare - permisiuni*: $R_{MP} \subseteq M \rightarrow P$, o relaţie 1:n care mapează fiecare maşina de stare la un set de permisiuni
  $R_{MP} = \{mp \mid mp = (m, p), (\forall m \in M, \exists p \in P)$.

Fiecărui rol este obligatoriu a i se asigna o maşină de stare şi fiecărui utilizator cel puţin un rol. Modelul SCAR-ACE impune aceste condiţii.

Permisiunile de modificare a seturilor R şi M şi a relaţiilor R_{RM} şi R_{MP} sunt denumite permisiuni admistrative şi fac parte din *rolul de administrator*.

Sesiunile sunt sub controlul individual al utilizatorilor. Din punct de vedere al modelului, un utilizator poate crea doar o sesiune. Rolurile active într-o sesiune pot fi modificate la dispoziţia utilizatorului prin autentificare. Sesiunea ia sfârşit fie la iniţiativa utilizatorului fie dacă aceasta activează prea mult. Strict vorbind, aceasta este o constrângere şi aparţine $RBAC_2$.

Anumiți autori [J93] consideră îndatoririle, în plus față de permisiuni, ca fiind un atribut al rolurilor. O îndatorire este obligativitatea unui utilizator de a realiza una sau mai multe sarcini care sunt în general esențiale pentru funcționarea unei implementări. Modelul SCAR-ACE consideră îndatoririle un concept avansat care nu se referă la SCAR-ACE$_0$.

3.3.5 Ierarhiile de roluri şi abordarea acestora în SCAR-ACE$_1$

Modelul SCAR-ACE$_1$ introduce ierarhiile de roluri. Ierarhiile de roluri sunt discutate în literatură în mod invariabil împreună cu rolurile [FK92], [HD95], [NO94], [SM00].

Moştenirea permisiunilor într-o ierarhie este tranzitivă şi, într-o ierarhie, pot exista moşteniri multiple.

Din punct de vedere matematic, aceste ierarhii sunt ordonări parțiale. O ordine parțială este o relație reflexivă, tranzitivă şi asimetrică. Moştenirea este reflexivă deoarece un rol îşi moşteneşte propriile permisiuni; tranzitivitatea este o cerință în acest context; regulile de asimetrie elimină rolurile care moştenesc toate permisiunile unul de la celălalt şi ar fi astfel redundante.

Modelul SCAR-ACE$_1$ este definit pentru ierarhia ilustrată în figura 3.a.

3.3.6 Definirea formală a modelului SCAR-ACE$_1$

In continuare introduc definiția formală a modelului SCAR-ACE$_1$:

- U, R, P, S, M aceleaşi cu cele din SCAR-ACE$_0$;
- $IR \subseteq R \times R$, reprezintă o ordonare parțială pe R denumită ierarhie de roluri sau relație de dominanță a rolului, denotată $\geq$;
- Relațiile sunt aceleaşi cu cele din SCAR-ACE$_0$.

Uneori este utilă limitarea moştenirii. Spre exemplu, în figura 3.c. rolul de manager de proiect este senior rolului de inginer producție şi celui de inginer pentru asigurarea calității. Rolul de inginer de asigurare a calității ar trebui să poată avea anumite permisiuni doar ale sale şi să prevină moştenirea acestora de către rolul managerului de proiect. Aceste permisiuni proprii rolului inginerului pentru asigurarea calitătii pot forma un rol aparte numit *rol privat* [J98]. Rolurile private se obțin în cadrul anumitor sisteme prin blocarea moştenirii unor permisiuni dar această tehnică previne ierarhia de a ilustra cu acuratețe distribuția permisiunilor.

3.3.7 Modelul constrângerilor în SCAR-ACE$_2$

Cel de-al treilea model – SCAR-ACE$_2$ - introduce constrângeri. Cu toate că aceste modele sunt denumite SCAR-ACE$_1$ şi SCAR-ACE$_2$, nu există un progres real implicat. In aceste modele se introduc fie ierarhiile fie constrângerile dar nu ambele.

Constrângerile reprezintă un aspect important al oricarui model de control al accesului. Un exemplu de constrângeri este cel al rolurilor mutual exclusive. In general, unui utilizator nu i se permite apartenența la două roluri mutual exclusive deoarece aceasta ar putea crea posibilitatea comiterii fraudelor. Acest principiu se numeşte *separarea îndatoririlor*.

SCAR-ACE_2 este nemodificat față de SCAR-ACE_0 cu excepția faptului că sunt necesare constrângeri pentru a determina nivelul de acceptare al diferitelor componente SCAR-ACE_0.

Modelul SCAR-ACE_2 realizează două seturi de constrângeri (vezi şi secțiunea 4.4) şi anume constrângeri statice şi constrângeri dinamice.

3.3.8 Definirea formală a modelului SCAR-ACE_2

Componentele modelului formal SCAR-ACE_2 sunt:

- *U, R, P, M* şi *S* (multimile de utilizatori, roluri, permisiuni, masini de stare şi respectiv sesiuni);
- $P \in SP$, permisiunile *P* sunt parte a unui set *SP* (se impun constrângeri asupra permisiunilor şi se specifică permisiunile acceptate şi nu cele care sunt respinse);
- $Ri \neq Rj$, $\forall Ri, Rj \in R$, definesc rolurile mutual exclusive;
- *maşina de stare*: $M \leftrightarrow R$, o funcție biunivocă care mapează fiecare maşină de stare unui rol şi fiecare rol unei maşini de stare. Prin intermediul acesteia se realizează constrângerile relativ la roluri.

Implementările RBAC în general aplică constrângeri simple care pot fi uşor şi eficient verificate sau forțate. In RBAC constrângerile simple pot avea diferite forme. Deoarece majoritatea constrângerilor aplicate relației de asignare a utilizatorilor la roluri au şi partea care se aplică relației de asignare a permisiunilor, se vor discuta în paralel constrângerile pe aceste două componente.

Cea mai uzuală constrângere $RBAC_2$ este cea a rolurilor mutual exclusive. Un utilizator poate fi asignat cel mult unui rol dintr-un set mutual exclusiv (în plus față de alte posibile roluri). Această constrângere realizează separarea îndatoririlor, constrângere care este apoi asigurată prin asignarea permisiunilor.

Constrângerea duală asupra asignării permisiunilor poate furniza asigurări adiționale în separarea îndatoririlor (acesta este însă rareori menționată în literatură). Această constrângere duală impune în SCAR-ACE_2 ca o permisiune să fie asignată cel mult unui rol dintr-un set mutual exclusiv. De exemplu, considerăm două roluri mutual exclusive, cumpărător şi vânzător pentru acelaşi produs. Exclusivitatea mutuală în termenii relației R_{UR} şi R_{SU} specifică faptul că un utilizator nu poate aparține ambelor roluri pentru acelaşi produs.

Mutual exclusivitatea în termenii relațiilor R_{RM} şi R_{MP} mai specifică faptul că aceeaşi permisiune – încasarea cecurilor, de exemplu – nu poate fi asignată ambelor roluri prin intermediul maşinilor de stare specifice. In mod normal această permisiune este asignată vânzătorului. In general ar putea să nu conteze care dintre două roluri *A* sau *B* primeşte autorizarea semnării unui anumit cec, dar contează ca doar unul dintre aceste roluri să primească această permisiune şi nu ambele.

In RBAC este introdus şi conceptul de *roluri prerechizite* care este bazat pe competență şi corespondență (modalitatea prin care unui utilizator îi poate fi asignat rolul *B* doar dacă avea deja asignat rolul *A*).

SCAR-ACE_2 accepta rolurile prerechizite in modul urmator: accesul la rolul de cumpărător (de exemplu) se poate realiza de catre un utilizator dacă inițial a avut un rol junior cu prioritate minimă. Cu alte cuvinte, un utilizator nu poate accede la un rol autorizat decât printr-un proces de autentificare.

O altă constrângere o reprezinta numărul maxim de utilizatori ai unui rol. Numai o persoană poate fi administrator, sau receptioner al unei cereri; de asemenea numărul de roluri căruia îi poate aparține un utilizator poate fi limitat. Acestea reprezintă constrângeri *de cardinalitate* care pot fi aplicate în mod corespunzător şi asignării permisiunilor pentru a se controla distribuirea acestora. Pe de altă parte, constrângerile de cardinalitate minimă pot fi dificil de implementat. De exemplu, dacă un rol necesită un număr minim de membri, este dificil pentru sistem să cunoască modul de răspuns corespunzător în cazul în care unul dintre membri dispare.

Constrângerea dual asupra asignării permisiunilor la roluri se aplică prin intermediul relațiilor R_{RM} şi R_{MP}. Constrângerea dual impune ca permisiunea *p* să poată fi asignată unui rol doar dacă acest rol posedă permisiunea *q*. De exemplu, permisiunea de a citi un fişier necesită permisiunea de a citi directorul în care acesta rezidă. Asignarea doar a primei permisiuni fără cea asupra directorului ar fi incompletă.

O ierarhie de roluri poate fi considerată o constrângere. Intr-o anumită măsură, SCAR-ACE_1 este redundant şi subsumat lui SCAR-ACE_2. Oricum, existența unei ierarhii de roluri trebuie recunoscută ca atare.

In SCAR-ACE_2 constrângerile de asignare a utilizatorilor sunt efective doar dacă este menținută o disciplină în asignarea identificatorilor utilizator (ID). Dacă aceluiaşi individ i se asignează doi sau mai multi identificatori, separarea utilizatorilor şi cardinalitatea acestora determină conflicte. Astfel, este strict necesară o corespondență de 1:1 între identificator şi utilizator pe intreaga durată a unei sesiuni.

In SCAR-ACE_2 un utilizator poate avea mai multe roluri pe durata unei sesiuni dar acestea nu pot fi mutual exclusive şi nu pot fi active în acelaşi timp. Altă constrângere limitează la 1 numărul sesiunilor unui utilizator care ar putea fi active în acelaşi timp.

Relativ la permisiuni, dacă aceeaşi operație aparține la două permisiuni diferite, modelul SCAR-ACE_2 nu poate forța efectiv cardinalitatea şi separarea constrângerilor.

3.3.9 Modelul SCAR-ACE_3

SCAR-ACE_3 este caracterizat de ierarhii şi constrângeri şi combină modelele SCAR-ACE_1 şi SCAR-ACE_2.

Constrângeri asupra ierarhiilor de roluri. Constrângerile se pot aplica asupra ierarhiei de roluri. De exemplu aplicarea unei ordini parțiale în cadrul ierarhiei de roluri este o constrângere. Constrângerile pot limita numărul rolurilor junior sau senior pe care le poate avea un rol dat. Două sau mai multe roluri pot fi, de asemenea, constrânse să nu aibă un rol senior sau junior comun. Asemenea constrângeri sunt utile când autorizarea de a modifica ierarhia de roluri este descentralizată şi

administratorul de sistem doreşte restricţionarea modului în care se realizează aceste modificări ale ierarhiei.

Interacţiuni. Intre ierarhii şi constrângeri intervin interacţiuni. De exemplu cumpărătorul şi vânzătorul au roluri mutual exclusive relativ la acelaşi produs. Supervizorul stocului de produse violează această constrângere mutual exclusivă deoarece are o parte a permisiuniunilor de la cumpărător şi o parte de la vânzător. Modelul trebuie aşadar să poată implementa aceasta violare a constrângerilor.

Constrângeri de cardinalitate. Să presupunem că un utilizator poate fi asignat unui singur rol şi această constrângere s-ar aplica unui rol dintr-o ierarhie de roluri (de exemplu asignarea rolului de inginer de test unui singur utilizator - vezi figura 6). Intrebarea este: se aplică constrângeri de cardinalitate numai unui rol sau şi moştenitorilor acestora? In exemplul din figura 6 aceasta ar determina ca şi cardinalitatea membrilor în proiect să fie egala tot cu 1 ceea ce este eronat. Răspunsul la întrebarea anterioară determină constrângerile de cardinalitate să se aplice punctual asupra fiecărui rol dintr-o ierarhie deci constrângerile de cardinalitate nu reprezintă o caracteristică care se moşteneşte.

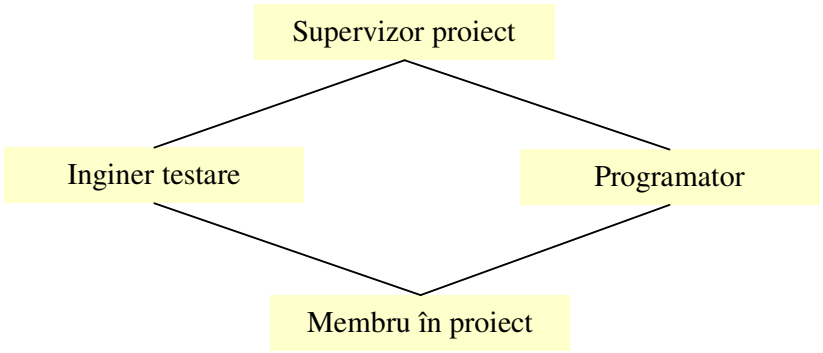

Figura 6. Exemplu de ierarhie de roluri

Modelul SCAR-ACE$_3$ acceptă atât o ierarhie limitată de roluri (vezi figura 7) cât şi constrângeri asupra rolurilor, permisiunilor (prin maşina de stare) şi sesiunilor (constrângeri de timp).

3.4 Analiza constrângerilor

Constrângerile autorizării bazate pe roluri se referă la asignarea rolurilor şi a permisiunilor. Principala caracteristică a modelului este aceea că permite delegarea autorităţii prin acţiuni administrative descentralizate. Un exemplu este cel în care două roluri *r1* şi *r2* sunt declarate ca şi mutual exclusive. O constrângere validă este cea prin care unui utilizator nu îi pot fi asignate cele două roluri exclusive în cadrul aceleiaşi sesiuni în anumite condiţii. Astfel, presupunând că utilizatorul *u1* are rolul *r1*, asignarea rolului *r2* aceluiaşi utilizator *u1* în cadrul aceleiaşi sesiuni ar determina un conflict.

3.4.1 Prerechizite

Primul tip de constrângeri specifică un set de prerechizite pe care un utilizator trebuie să le satisfacă pentru a deține un anumit rol. O astfel de prerechizită poate fi cea prin care unui utilizator i se cere să dețină un anumit rol înainte de a deține un altul (de exemplu: pentru a deține rolul de vânzător un utilizator trebuie să fi deținut anterior rolul de vizitator).

O altă prerechizită poate include evaluări complexe de predicate, precum cele care consideră timpul. Au fost doar câteva încercări de formalizare a limbajelor care să exprime tipuri particulare de constrângeri. O astfel de cercetare a fost realizată de Bertino, Bonatti şi Ferrari care s-au ocupat de constrângeri temporale [BBF00].

3.4.2 Separarea îndatoririlor

Un alt tip de constrângere luat în considerare este cel de separare a îndatoririlor SoD. Este considerată una dintre cele mai importante constrângeri şi este adesea motivarea principală în abordarea controlului accesului bazat pe roluri. SoD este o tehnică prin care se reduc fraudele prin intermediul repartizării responsabilității şi autorității pentru realizarea unei operații, între anumiți utilizatori. Un exemplu uzual este pregătirea şi aprobarea unui ordin de cumpărare. Astfel, ordinul trebuie să fie inițiat şi aprobat de către diferite persoane, frauda în acest caz implicând conspirația a două persoane, puțin probabil a se întâmpla şi uşor de descoperit. In [SZ83] se descriu câteva tipuri de SoD:

- separarea statică a îndatoririlor -SSD-, cunoscută ca şi separarea puternică a îndatoririlor, prin care se obține separarea îndatoririlor prin specificarea de politici într-o manieră care să nu permită nici unui utilizator să dețină roluri care sunt conflictuale;
- separarea dinamică a îndatoririlor -DSD-, cunoscută ca şi excluderea slabă, permite utilizatorilor să acceseze roluri conflictuale dar într-un timp controlat şi limitat. Pe baza controlului efectuat se deosebesc următoarele tipuri de excluziune slabă:
  - separare dinamică simplă a îndatoririlor – împiedică utilizatorii în a deține roluri conflictuale în acelaşi timp;
  - separarea orientată pe obiect a îndatoririlor – permite utilizatorilor să dețină roluri conflictuale, dar privilegiile conflictuale nu pot fi exercitate asupra aceluiaşi obiect;
  - separarea operațională a îndatoririlor – permite activarea rolurilor conflictuale atâta timp cât nu conțin privilegii ale unei anumite operații;
  - separarea bazată pe istoric a îndatoririlor – împiedică utilizatorul de a efectua toate acțiunile ataşate unei operații.

In [GGF98] sunt formalizate tipurile de separări ale îndatoririlor menționate mai sus. In [AS00] acestea sunt extinse prin furnizarea limbajului RSL99 pentru descrierea tipurilor de separări ale îndatoririlor. In [K97] se furnizează un model formal pentru exprimarea separării îndatoririlor dintr-o singură sesiune. Separări adiționale ale îndatoririlor precum Zidul Chinezesc se găsesc în [JSS97].

3.4.3 Cardinalitate

Constrângerile de cardinalitate limitează numărul de sesiuni posibil a fi deținute de către un utilizator sau numărul de utilizatori activi în cadrul unui rol în acelaşi timp. De exemplu, cardinalitatea poate exprima o politică prin care se stabileşte că în sistem există un singur administrator activ la un anumit moment dat. Cardinalitatea este parte a constrângerilor de control a activărilor. Aceste constrângeri trebuie să fie evaluate în momentul rulării.

3.4.4 Constrângerile modelului

Modelul SCAR-ACE impune constrângeri de tipul prerechizite şi separari ale indatoririlor statice şi dinamice.

Ca şi prerechizite sunt acele constrângeri care impun ordinea rolurilor in cadrul ierarhiei de roluri, un utilizator neputând accede la roluri situate mai jos în ierarhie fară a parcurge rolurile anterioare care au responsabilitate şi autoritate mai slabă.

Din punct de vedere al sepaparilor indatoririlor statice este controlată secvența de operații necesare satisfacerii unei cereri prin definirea tranzițiilor posibile dintr-o stare. Separarea dinamica a indatoririlor implica faptul ca un utilizator poate să dețină roluri mutual exclusive în două sesiuni diferite dar nu pentru aceeaşi resursa.

In contextul SCAR-ACE rolurile sunt utilizate pentru a modela diferite poziții şi scopuri în cadrul unei aplicatii de comerț electronic. Un rol poate reprezenta competențe concrete precum cele ale unui vizitator, vânzător sau cumpărător, administrator de sistem, recepționer sau distribuitor de bunuri. Rolurile pot reflecta asignarea unor îndatoriri şi fiecare rol încapsulează autoritatea şi responsabilitatea specifică, adică vizitatorul are autoritate şi responsabilitate slabă şi limitată, vânzătorul poate insera articole în catalogul de produse iar cumpărătorul se poate înregistra pentru a le achiziționa.

Rolurile sunt organizate ierarhic iar în cadrul acestei ierarhii este implementată şi relația de moştenire conform săgeților, respectiv rolurile de cumpărător şi vânzător moştenesc permisiunile acordate rolului de vizitator (vezi figura 7).

Rolul administrativ are permisiunea de a modifica setul de roluri sau permisiuni, sau de a modifica maşinile de stare.

Utilizatorii pot accesa diferitele roluri existente (mai puțin cel de administrator). Acestor roluri le sunt ataşate, prin intermediul maşinilor de stare, permisiunile necesare efectuării anumitor operații. RBAC nu soluționează toate problemele legate de controlul accesului. Sunt necesare metode mai sofisticate care să țină cont de secvențele de control a operațiilor. De exemplu, când o achiziție necesită diverşi paşi înaintea emiterii ordinului de cumpărare, RBAC nu poate controla în mod direct această secvență de evenimente.

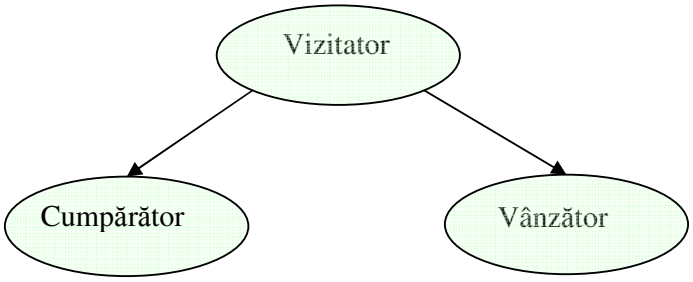

Figura 7. Ierarhia de roluri SCAR-ACE

Modelul SCAR-ACE emite credenţiale alcătuite din mai multe atribute pentru identificarea unei cereri. Secvenţele posibile de operaţii care constituie permisiunile unui rol sunt grupate în maşini de stare a permisiunilor / rol. Introducerea maşinilor de stare in SCAR-ACE reprezintă o extensie adusă constrângerilor care sunt descrise într-un model $RBAC_3$.

Entităţile cu care operează SCAR-ACE sunt cele definite în figura 4. Resursele la care se solicită accesul sunt metode ale obiectelor şi date din baza de date. Atunci când un utilizator doreşte să acceseze o resursă el emite o cerere. Când se primeşte cererea de catre aplicatie, se identifică emiţătorul acesteia şi, dacă acesta are permisiunea de a accesa resursa cerută, primeşte acceptul. Mecanismul însă este complex deoarece ia în considerare maşina de stare şi restricţiile impuse de aceasta. Astfel, pentru a accesa o resursă oferită de sistem, utilizatorul parcurge un set de operaţii permise de stările maşinii de stare. La un moment dat fiecare utilizator este într-o stare bine definită de unde are posibilitatea de a emite cereri pentru anumite resurse. Fiecare cerere poate fi emisă prin lansarea unui eveniment care va interoga sistemul la modul: “Pot rezolva din rolul *r* cererea pentru resursa *x*, din starea *y*, prin lansarea evenimentului *z* în prezenţa parametrilor *v*?” Sistemul verifică combinaţia emisă şi poate răspunde fie prin acordarea permisiunii de acces la resursă în caz de succes, fie printr-un mesaj de eroare în caz de eşec. După ce se realizează accesul la resursă, are loc o tranziţie de la starea din care s-a emis cererea la starea următoare.

Identificarea emiţătorului cererii este realizată prin interpretarea unui credenţial. Modelul crează, memorează şi gestionează informaţiile de autorizare în mod dinamic şi distribuit. Proprietatea necesară unui utilizator pentru autorizare este deci un credenţial care conţine mai mult decât o identificare a utilizatorului. In momentul în care un utilizator emite o cerere pentru o resursă această cerere poartă ştampila tranzitiilor prin care se realizează accesul.

Fiecare utilizator poate accesa resurse prin intermediul maşinii de stare, având anumite permisiuni, şi toate acestea pe durata unei sesiuni. Sesiunile impun constrângeri asupra modelului şi anume: dacă un utilizator nu emite o cerere nouă într-un anumit interval de timp, sesiunea acestuia ia sfârşit automat. Timpul este o variabila parametrizabilă care poate fi modificată dinamic.

Una dintre cerinţele RBAC este aceea ca utilizatorilor să le fie permisă exercitarea rolurilor multiple. Astfel, modelul SCAR-ACE are maşini de stare şi nu reţele Petri tocmai pentru a permite acest mecanism.

Pentru realizarea autorizării într-un sistem distribuit, este necesar un model în care deciziile de control al accesului să se bazeze pe atribute de identificare. Astfel, modelul SCAR-ACE are următoarele caracteristici:

a) Atributele specifice stărilor maşinii de stare sunt distribuite: fiecare stare are atribute specifice care o identifică iar aceste atribute sunt atât pe server cât şi pe client. Trecerea de la o stare la o altă stare se realizează pe baza

unor atribute proprii fiecărei stări. SCAR-ACE asociază unei stări o interfață utilizator grafică sau anumite componente ale acesteia. In acest caz o stare este caracterizată de elemente active şi pasive conținute de către interfața care o defineşte;

b) Caracteristica de delegare a autorității atributelor: o stare îşi deleagă autoritatea unei alte stări în sensul că o stare are încredere în "judecata" altei stări pe baza atributelor acesteia. Controlul accesului implică impunerea unor secvențe logice autorizate pentru fiecare rol în parte (controlate prin maşina de stare) de la care un utilizator nu se poate abate si nu poate forța un alt traseu al starilor în afara celor predefinite. Această secvență are loc pe baza încrederii trecerii de la o stare la alta pe baza unuia sau a mai multor atribute care formează credențiale;
c) Caracteristica de conjuncție a atributelor: o cerere utilizează conjuncția între câteva atribute pentru a trece dintr-o stare în alta. In general o stare este asociată câtorva elemente (active şi / sau pasive) dintr-o pagină a interfeței utilizator. Toate aceste elemente sunt considerate atribute ale stării respective;
d) Atributele au valori: cel puțin unul dintre atributele caracteristice unei stări are ataşată o valoare. De exemplu acesta poate fi un câmp (exemplu produsul de achiziționat) care are o valoare selectabilă dintr-o listă dată. Toate valorile atributelor unei stări formează parametrii de apel în cererea de accesare a unei resurse.

3.5 Privilegiile

Includerea privilegiilor în cadrul designului şi a implementării este adesea, în accepțiunea RBAC, delegată bazei de date [DDT+04].

Includerea privilegiilor în cadrul SCAR-ACE se referă la definirea *nivelelor* şi a *politicilor* de acordare a privilegiilor. Politicile de acordare a privilegiilor se referă la resursele de protejat, utilizatorii şi rolurile sistemului precum şi permisiunile acordate. Nivelele de acordare a privilegiilor se referă la autorizare:

- *autentificarea* – pentru verificarea utilizatorilor şi limitarea acțiunilor lor la privilegiile acordate acestora;
- *controlul accesului* – necesar acordării sau revocării de privilegii utilizatorilor.

Autentificarea si controlul accesului sunt cerințele *minimale* de acordare a privilegiilor şi implică introducerea politicii de securitate în primele etape ale ciclului de viață al aplicației.

Modelul SCAR-ACE se bazeaza pe o politică restrictivă care să nu permită accesul decât într-o anumită secvență impusă cu desemnarea clară a punctului de start (de obicei pagina "Home"). Cu toate că există roluri cu grade diferite de securitate, modelul impune controlul accesului pentru toate grupurile de utilizatori inclusiv pentru cele cu prioritate minimă.

Autentificarea se realizează prin metoda devenită clasică şi anume *identificatorul utilizatorului şi parola* definesc fiecare utilizator autorizat al aplicației de comerț electronic.

Autorizarea impune grupuri de privilegii pentru diverse tipuri de utilizatori.

Asigurarea acordării privilegiilor impune utilizarea credențialelor în controlul secvenței de operații. Fiecare credențial încapsulează mai multe informații care

identifică atât utilizatorul (identificator unic utilizator), rolul utilizatorului în sistem, operația curentă cât şi cererea pentru operația următoare. Toate acestea pot determina fie obținerea accesului la operația şi resursele cerute în caz de succes fie un răspuns de eroare dacă accesul nu este permis.

3.5.1 Specificarea nivelelor de acordare a privilegiilor

Una dintre uneltele importante în proiectarea software este UML (Unified Modeling Language) care este un limbaj de specificare, vizualizare, construire şi documentare de artefacte software. UML unifică mai multe abordări [B+91], [J+92], [R+91] şi altele, într-un standard astfel încât să încapsuleze caracteristicile modelelor constituente.

In UML există diferite tipuri de diagrame (nouă tipuri de diagrame) dintre care amintim: *diagrame use case* pentru interacțiunea utilizatorilor cu componentele software, *diagrama claselor* pentru clasele statice şi relațiile dintre acestea, *diagrame de secvență* pentru comportamentul dinamic a instanțelor din diagramele claselor, *diagrame de stare* pentru ilustrarea tranzițiilor.

Suportul UML pentru definirea cerințelor de securitate şi a privilegiilor prin aceste diagrame şi a elementele lor constituente (exemplu: actori, sisteme, use case-uri, clase, instanțe, relații, metode, date, etc) este precar [DDT+04] dar există eforturi de a le încorpora în UML: [ES99], [SA00] folosesc UML ca pe un limbaj de reprezentare a modelelor şi notațiilor RBAC; [J02], [J05] pentru definiții teoretice ale securității cu UML; [BDL03] introduce SecureUML pentru securitate cu elemente de meta-model; [R+03] a utilizat UML pentru a modela sisteme bazate pe MAC şi RBAC; [AW1 03] şi [AW2 03] prezintă un cadru de încorporare a securității în use case-uri.

Modelul SCAR-ACE încorporează cerințele de securitate în diagrame use case, de clase, de stare, de secvență şi de colaborare pentru definirea elementelor UML relevante. Fezabilitatea acestor diagrame este demonstrată prin implementarea practică.

3.5.2 Nivele de securitate în acordarea privilegiilor

In MAC controlul accesului este bazat pe tupla (*s, op, o*) care indică faptul că subiectul *s* poate executa operația *op* (unul sau mai multe procese) asupra obiectului *o*. In timp ce această abordare este specifică conținutului bazelor de date relaționale, există mai multe motive pentru care acest triplet (*s, op, o*) nu este corespunzător abordării SCAR-ACE orientate obiect. In primul rând în MAC un obiect este o instanță caracterizată de mai multe atribute, care pot fi simple sau referințe la alte obiecte, fiecare având diferite caracteristici ale privilegiilor. In al doilea rând, operația *op* reprezintă în mod tipic un set de operații standard (citire, scriere, actualizare, etc) asupra obiectului *o*, în timp ce metodele (operațiile) unei clase OO conțin seturi complexe de operații şi sarcini.

Pentru abordarea SCAR-ACE propun utilizarea cvadruplului (s_i, s_j, p, o). Această tuplă desemnează trecerea de la starea s_i la starea s_j şi executarea operației o, dacă au fost îndeplinite permisiunile p. O stare încapsulează un obiect complex (specific unui set de atribute ale unei interfețe utilizator – o pagină sau un cadru).

Pentru realizarea tranziției trebuie îndeplinite anumite permisiuni specificate prin *p*. Trecerea la starea următoare se realizează prin executarea unei operații *o* ce reprezintă unul sau mai multe procese executate asupra obiectelor implicate în tranziție. Un proces este mai mult decât o operație de citire sau scriere în baza de date acesta putând conține calcule complexe, verificări de permisiuni, interacțiuni între tabele ale bazei de date, servicii, etc.

In design-ul unui model de acordare a privilegiilor există diferite nivele de abordare, nivele înglobate în proiectare şi care se regăsesc apoi în aplicația propriu-zisă. In cazul concret al sistemului SCAR-ACE există diverse module care poartă amprenta restricțiilor de acordare a privilegiilor. [DD$^+$04] introduce noțiunea de *Nivel de securitate* pentru implementarea regulilor de asigurare a securității. O parte a acestor nivele precum si unele specifice se regăsesc în modelul SCAR-ACE adaptate pentru acordarea privilegiilor. Astfel în [OGI05] am definit *Nivelul 1* care se axează pe diagramele de use case, *Nivelul 2* care subliniază legătura dintre clasele care se utilizează pentru fiecare use case, şi *Nivelul 3* care reprezintă crearea diagramelor de stare, de secvență şi de colaborare cu interacțiunile dintre actori. In fiecare nivel m-am concentrat pe constrângeri de genul (s_i, s_j, *p*, *o*).

Nivelul 1 – diagrame use case

O *diagrama use case* este o colecție de cazuri utilizator. Un caz utilizator (use case) reprezintă o descriere a comportamentului actorilor pentru o anumită parte a aplicației.

Un *actor* este o entitate externă care interacționează cu sistemul software la un anumit nivel pentru a reprezenta simularea evenimentelor posibile (procesele). In cazul concret al modelului SCAR-ACE un actor poate fi asimilat unui utilizator care are asignat un rol.

In diagrama de use case din figura 8 sunt prezentați doi actori: *Vizitator* şi *Cumpărător* şi procesul de înregistrare/login pentru autentificare. Practic în modelul ierarhic RBAC cei doi actori sunt două roluri aflate într-o relație de moştenire. Astfel, permisiunile unui vizitator sunt moştenite de către cumpărător (sunt incluse în cadrul permisiunilor cumpărătorului).

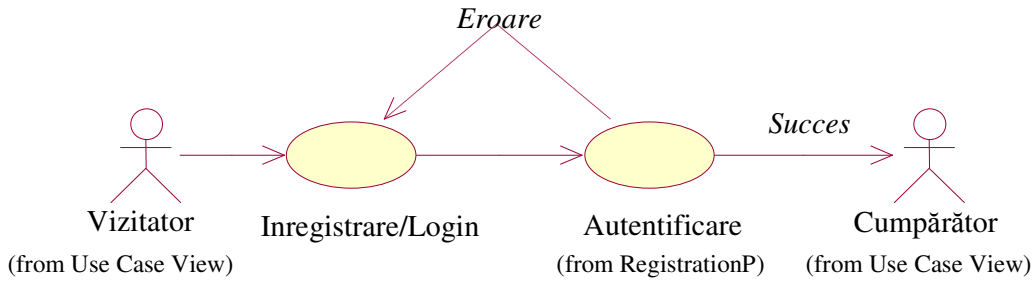

Figura 8. Use case de înregistrare/login a unui vizitator

Se introduce noțiunea de permisiune ataşată unui actor *ac* şi se notează cu *ac.p*. Setul tuturor permisiunilor acordate actorului *ac* se notează cu *ac.PMS*. Deoarece în SCAR-ACE un actor este asociat unui rol, permisiunile *ac.PMS* ale actorului *ac* sunt echivalente cu permisiunile *PMS* asociate rolului r: *r.PMS*. Valoarea lui *r.PMS* este aleasă de către proiectant pe baza politicii de acordare a privilegiilor şi reflectă

permisiunile *PMS* ale unui utilizator la implicarea sa într-un comportament al aplicației în momentul în care deține rolul *r* (specific actorului *ac*).

Se introduce **Regula de acordare a privilegiilor în moştenirea rolurilor:**

> ∀ actori ac_m şi ac_n asociaţi rolurilor r_m şi respectiv r_n, r_n moşteneşte privilegiile lui $r_m \Leftrightarrow r_n.PMS \geq r_m.PMS$ sau $ac_n.PMS \geq ac_m.PMS$.

Cu alte cuvinte permisiunile *PMS* ale rolului senior (actorului părinte) sunt cel puțin egale cu cele ale rolului junior (actorului fiu).

In diagrama use case din figura 8 actorul *Cumpărător* moşteneşte permisiunile actorului *Vizitator* (le încapsulează) după o autentificare cu succes.

In modelul SCAR-ACE un utilizator poate deschide o sesiune de lucru în rolul cu permisiune minimă (*PMS* minime).

Procesul ataşat operației de autentificare are drept rezultat fie permisiunea de log-in sau înregistrare în caz de reuşită, fie semnalarea unei erori în caz de eşec.

Pentru a verifica regula emisă s-a ales ca utilizatorul să îşi deschidă sesiunea de lucru în rolul actorului *vizitator* care are permisiunea cea mai joasă. Această restricție este impusă şi, oricare ar fi starea în care utilizatorul ar dori să acceadă la inițierea unei sesiuni, el este redirecționat către starea *vizitator* cu permisiuni minime. Pentru a intra în orice alt rol, utilizatorul trebuie să treacă printr-un proces de autentificare. Când utilizatorul ia un rol al unui actor părinte ac_m, prin moştenire, sunt necesare şi permisiunile actorului fiu ac_n.

Nivelul 2 – diagrame de clase

O *diagramă de clase* este alcătuită din clase şi reprezintă structura statică a modelului conceptual. O clasă este abstractizarea unui set de obiecte care sunt caracterizate de atribute şi operații. In implementare, o clasă este denumită obiect iar operația unei clase este denumită metodă. Interfețele la alte modele sunt o particularizare în cadrul diagramelor de clase. In figura 9 sunt reprezentate interfețele corespunzătoare rolurilor din figura 8 şi a procesului de selecție a rolurilor înainte şi după autentificare:

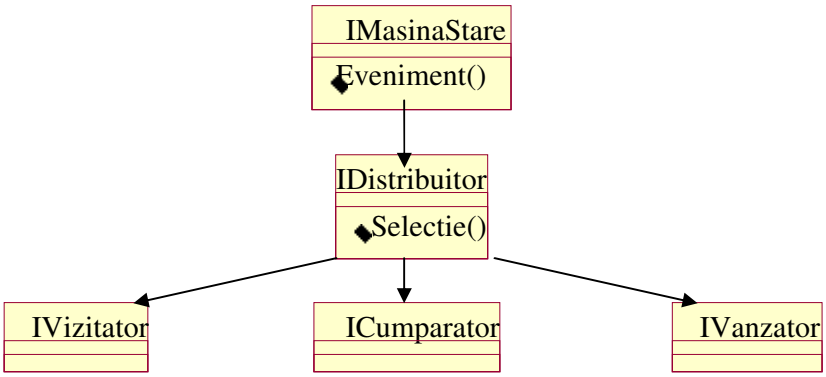

Figura 9. Diagrama de clase: Interfețe pentru selectarea rolurilor

In Nivelul 2 de securitate, pentru acordarea privilegiilor, se pune accentul pe definirea claselor care sunt utilizate într-un use case. Acest pas este anterior celui pentru dependențele bazate pe mesajele care sunt definite în diagrama de secvențe. UML nu suportă un tip explicit de diagramă care să asocieze use case-urile cu clasele

care se utilizează pentru a-l implementa; are însă în mod implicit facilitatea de a alege clasele care sunt implicate în diagrama de secvență pentru a descrie activitățile unui use case.

Se introduce noțiunea de domeniu al permisiunilor *DMP* pentru clasa *c* care are o valoare minimă (*min*) şi o valoare maxima (*max*) pentru metodele pe care le implementează. Cu alte cuvine $c.DMP_{min}$ *şi* $c.DMP_{max}$ specifică domeniul de permisiuni al clasei *c*.

In Nivelul 2, pentru o clasă *c* care se utilizează într-o diagramă de secvență pentru a realiza funcționalitatea unui use case *uc*, *DMP* al *uc* trebuie să fie mai mare decât cel al clasei/claselor pe care le conține. Cu alte cuvinte domeniul permisiunilor al *uc* înglobează toate domeniile de permisiuni ale claselor componente, având nivelul de acordare a privilegiilor cel puțin egal cu nivelul minim al claselor componente.

Se introduce **Regula de acordare a privilegiilor în relația dintre Use Case şi Clasă**.

$\forall$ use case *uc* şi clasă *c*,
domeniul permisiunilor al *uc* relativ la $c \Leftrightarrow uc.DMP \geq c.DMP_{min}$.

De notat este faptul că Nivelul 2 reprezintă stadiul inițial, înaintea creări diagramelor de secvență, în care diagramele de clasă sunt asociate cu use case-urile. In acest moment sunt selectate clasele/obiectele fără includerea dependențelor între metode.

Nivelul 3 – diagrame de stare, de secvență şi de colaborare

Diagrama de stare ilustrează stările, tranzițiile între stări şi evenimentele declanşatoare. O stare încapsulează mai multe proprietăți (are ataşate diferite atribute) şi poate fi modificată în anumite condiții (de obicei la apariția unui eveniment).

Diagrama de stare din figura 10 ilustrează o parte a maşinii de stare şi anume maşina de stare care controlează autentificarea exemplificată în use case-ul din figura 8.

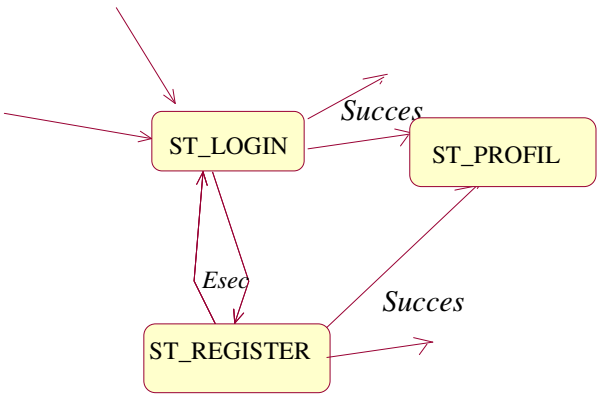

Figura 10. Diagrama de stare ce denotă controlul accesului în autentificare

O *diagramă de secvență* în UML reprezintă comportamentul dinamic al instanțelor (obiectelor) diagramei de clase. Pentru a realiza o anumită operație (adesea un use case), diagrama de secvență indică interacțiunea dintre obiecte. Scopul acestei diagrame este de a modela fluența controlului, de a ilustra un scenariu tipic al procesării.

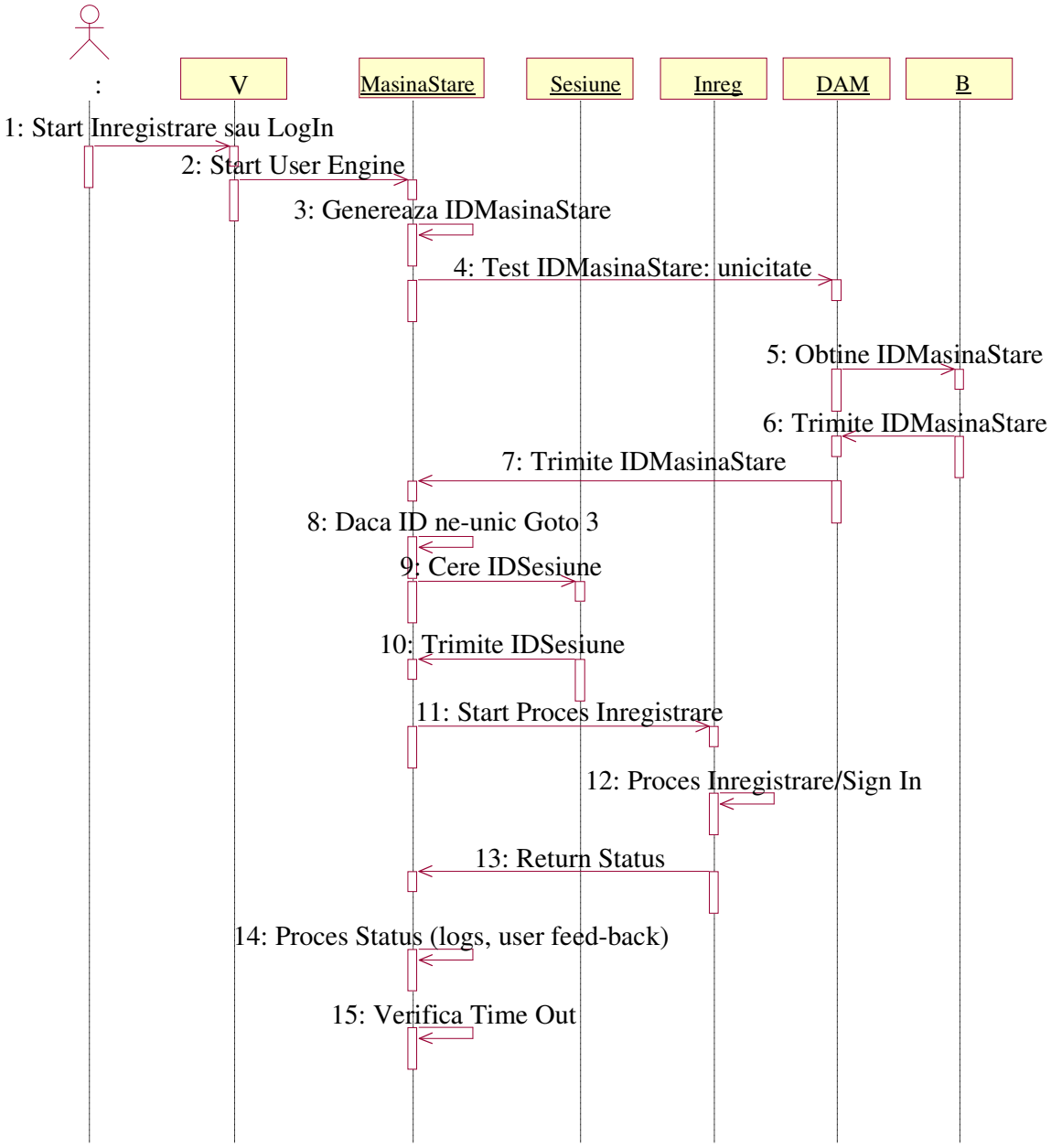

Figura 11. Diagrama de secvență pentru determinarea credențialului

Figura 11 exemplifică diagrama de secvență pentru obținerea credențialului la pentru autentificare.

In strânsă corelație cu această diagramă de secvență este diagrama de colaborare ilustrată în figura 12.

Pentru prezentarea regulilor de acordare a privilegiilor este necesară furnizarea unor notații adiționale. Astfel, o metoda *m* definită în clasa *c* (denotată cu *c.m*) este implementarea unei operații pentru descrierea unui set de activități specifice proceselor clasei *c*. Fie *c.M* setul tuturor metodelor definite în *c*. In proiectarea OO, presupunem că toate atributele clasei sunt implicit private (asigură securitatea), fiind modificate de metode publice sau private (nu pot fi modificate direct).

In continuare adopt notațiile introduse în [ZM90] în care se definesc două tipuri disjuncte de metode în cadrul unei clase: *mutatori* care alterează starea unei instanțe şi *observatori* care conservă starea unei instanțe. Astfel avem:

- $c.M^W = \{c.m \mid c.m$ modifică atributele unei clase$\}$ – mutatori;
- $c.M^{\neg W} = c.M \setminus c.M^W$ – setul metodelor care nu modifică atributele – observatori.

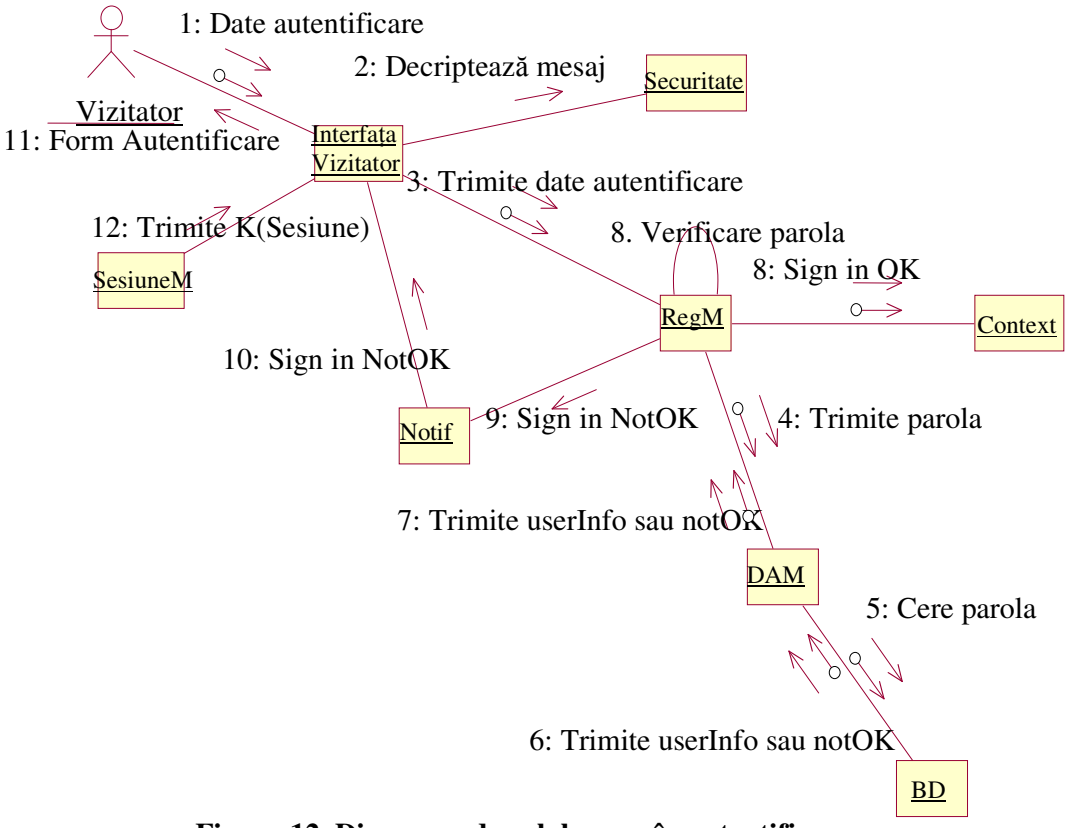

Figura 12. Diagrama de colaborare în autentificare

Un element al $c.M^W$ respectiv $c.M^{\neg W}$ este numit metodă *mutabilă*, respectiv *imutabilă şi* se notează *MuM*, respectiv *IMuM*.

Utilizând aceste metode se pot defini regulile de acordare a privilegiilor într-o diagramă de secvență pentru un actor *ac* respectiv un rol *r*.

Se defineşte **Regula de acordare a privilegiilor pentru metoda unei clase apelate de un actor** (implementarea permisiunilor *PMS* ale unui rol *r* de către clasa *c*).

> ∀ actor *ac* şi metoda *cj.my*, *ac* utilizează în acordarea privilegiilor pe $c_j.m_y \Leftrightarrow$ dacă $c_j.m_y \in c_j.M^{\neg W}$: $ac.PMS \geq c_j.m_y.PMS$.
>
> sau
>
> ∀ rol *r* şi metoda *cj.my*, permisiunile *PMS* ale rolului *r* sunt implementate şi de $c_j.m_y \Leftrightarrow$ dacă $c_j.m_y \in c_j.M^{\neg W}$: $r.PMS \geq c_j.m_y.PMS$.

Astfel, pentru acordarea privilegiilor actorul care apelează o metodă a unei clase imutabile trebuie să aibă permisiunea cel puţin egală cu cea a metodei clasei. Permisiunile unui rol implementat de o metodă a unei clase sunt cel putin egale cu permisiunile implementate de metodă.

Se mai introduce o regulă şi anume **Regula de acordare a privilegiilor pentru metodele dintr-un Use-Case**.

> ∀ use case *uc* şi diagrama de secvenţă D_{sec}, *uc* este descris de metodele din $D_{sec} \Leftrightarrow uc.DMP \leq \min(c_{i1}.m_{x1}.PMS, ..., c_{ik}.m_{xk}.PMS)$ unde $c_{i1}.m_{x1}.PMS, ..., c_{ik}.m_{xk}.PMS \in c.M^W$ şi sunt utilizate în D_{sec} pentru a descrie acţiunile din *uc*.

In această regulă se specifică faptul că proiectantul nu poate utiliza într-un use case *uc* metode mutabile cu permisiunile *PMS* mai mici decât cele ale *uc*.

Se defineşte **Regula de acordare a privilegiilor pentru o clasă relativ la metodele sale**.

Astfel se introduc următoarele noţiuni:

- $c.DMP_{min} \leq \min \{c.m_x.PMS \mid$ metoda m_x este definită în clasa $c\}$;
- $c.DMP_{max} \geq \max \{c.m_x.PMS \mid$ metoda m_x este definită în clasa $c\}$.

Clasa *c* trebuie să aibă cel puţin o metodă mutabilă. Conform principiului celui mai mic privilegiu, valoarea minimă a domeniului privilegiilor clasei *c* are valoarea maximă egală cu cea mai mică valoare a permisiunilor metodelor sale. Valoarea maximă a *DMP* pentru o clasa *c* este cel puţin egală cu maxima permisiunii metodelor componente.

3.5.3 Politica de acordare a privilegiilor

Controlul accesului se referă la alocarea permisiunilor pentru fiecare rol. Permisiunile rolurilor reprezintă tuple *<r, {s}>* în care *r* este un rol iar *{s}* denotă un set de stări.

$PMS_r = \{<r, \{s\}> \mid r \in R, s \in S\}$

adică permisiunile *PMS* ale rolului *r* sunt asocieri între acel rol şi stările *s* corespunzătoare acestuia.

Pentru a exprima relaţia dintre elementele tuplei se defineşte politica conceptuală de acordare a privilegiilor care afirmă: “fiecare rol are permisiunea de a accesa doar un anumit set de stări date, între care sunt stabilite tranziţii bine definite”.

In SCAR-ACE politica conceptuală se referă la introducerea unei maşini de stare pentru acordarea de permisiuni fiecărui rol, maşină de stare care încapsulează setul *{s}* de stări pe care rolul *r* are autorizarea de a le accesa. Astfel, există câte o maşină de stare pentru fiecare rol prin intermediul căreia se acordă permisiunile, maşină de stare care urmăreşte şi fluenţa controlului. O interfaţă utilizator grafică asociată unei stări din setul *{s}* reprezintă o stare în maşina de stare *M* iar fiecare tranziţie reprezintă un arc al acelei maşini de stare.

Politica de securitate mai introduce o constrângere temporală asupra validităţii unui credenţial şi anume: “un credenţial nu poate fi valabil peste un timp *t* de la data ultimei accesări”.

In implementare, aceasta se reflectă prin introducerea unei perioade *t* de validitate a credenţialului după ultimul acces la o resursă sau serviciu. In mod general această perioadă este acceptată a fi de 20 de minute.

3.5.4 Algebra politicii de acordare a privilegiilor

In cadrul politicii de acordare a privilegiilor se introduc diferite operaţii algebrice pe tupla <*r,{s}*> şi asupra credenţialelor, precum cele exemplificate mai jos:

a. *Enumerarea* – este suportat un set enumerativ de interfeţe utilizator grafice *{i}* pentru fiecare rol. Intre aceste interfeţe există o ordine care este prestabilită (deci nu arbitrară) şi nu există posibilitatea accesării aleatoare a acestora. Există o stare iniţială către care un utilizator este redirecţionat oricare alta ar fi adresa pe care acesta o solicită la iniţierea unei sesiuni de lucru. Această stare este prima în cadrul setului enumerativ, se numeşte *Home* şi are gradul de securitate cel mai redus.

b. *Adunarea* (cu sensul de *uniune*) este suportată în două concepte diferite, şi anume:
   - pentru tupla <*r, {s}*> în cadrul relaţiei de moştenire între roluri: un rol senior suportă seturile *{s}* aferente tuturor rolurilor junior pe care le moşteneşte cât şi setul *{s}* specific rolului senior pe care îl deţine;
   - în crearea credenţialelor: acestea sunt o concatenare a mai multor identificatori (vezi şi secţiunea 3.6).

c. *Scăderea* sau mai general toate operaţiile care necesită lucrul cu “informaţii negative” relativ la accesul la o resursă sau la un serviciu nu este suportată. Aceasta nu se regăseşte nici în gestionarea stărilor şi nici în lucrul cu credenţialele:
   - pentru stări: sunt accesate de către un utilizator *doar* acele stări pentru care utilizatorul este autorizat, stări care formează setul *{s}* şi nu se specifică setul pentru care accesul este interzis (nu se specifică “stările interzise”);
   - pentru credenţiale: sunt acceptate credenţialele care întrunesc *toate* condiţiile de acces şi nu se specifică credenţialele care nu sunt acceptate. Mai exact: sunt specificate condiţiile necesare unui

credențial în a fi acceptat şi orice altă variantă este respinsă prin excludere.

d. *Selecția* poate fi aplicată pentru credențial şi interpretată ca şi separarea identificatorilor din cadrul credențialului, identificatori care satisfac anumite constrângeri. Dacă se consideră că un credențial este suma mai multor identificatori aceştia sunt separați prin selecție şi testați individual, în vederea satisfacerii condițiilor de acces la o resursă sau serviciu. Toți termenii componenți ai unui credențial trebuie să fie valizi.

3.6 Credențiale

3.6.1 Componentele unui credențial

Un credențial este emis de fiecare dată când un utilizator cere accesarea unei resurse sau a unui serviciu. Un credențial se setează atât pe server cât şi pe client şi conține următorii identificatori [O00]:

- identificator utilizator;
- identificator rol;
- identificator stare curentă;
- identificator eveniment;

Identificator utilizator este un număr mare generat prin apelul unei funcții de generare identificatori utilizator (GUID) şi este valid pe toată durata unei sesiuni de lucru. Este generat de server la inițierea unei sesiuni de lucru şi este transmis pe client care, la rândul său îl trimite nealterat serverului. Orice modificare asupra acestui identificator determină invaliditatea credențialului şi întreruperea sesiunii.

Identificator rol reprezintă rolul utilizatorului (sau echivalent identificator maşina de stare). Identificatorul este emis de către server şi este transferat clientului care nu il poate modifica. Este în strânsă corelație cu următorii termeni din cadrul credențialului: starea curentă şi evenimentul- şi orice alterare în cadrul acestui tuplu determină invaliditatea credențialului. Aceasta deoarece fiecărui rol i se conferă un set unic de stări şi fiecare stare poate fi modificată doar la apariția anumitor evenimente.

Identificator stare curentă denumeşte în mod unic un set de atribute (o interfață utilizator: o pagină sau un cadru). Acest termen face parte din grupul de permisiuni acordate şi, în funcție de valoarea tuplei (rol, stare curentă, eveniment) se acordă dreptul de acces la anumite resurse şi/sau servicii. Valoarea acestui identificator este inițiată de server care o trimite clientului iar acesta, la rândul său o retrimite serverului. Similar celorlalți identificatori, nici acesta nu poate fi modificat pe traseu fără repercursiuni asupra validității credențialului.

Identificator eveniment de modificare stare, este parte a acordării drepturilor de acces împreună cu identificatorii anteriori. Acest termen este emis pe partea de client la acționarea unui element activ al interfeței - de exemplu un meniu, un buton sau o selecție - şi poate determina:

- modificare identificator rol prin trecerea la un nou rol;
- modificare stare curentă – întotdeauna se trece la o nouă stare (de exemplu o nouă interfață utilizator).

3.6.2 Prelucrarea credențialelor

Pentru a realiza o aplicație sigură aceasta trebuie să asigure printre altele şi următoarele:

- să localizeze toate intrările în sistem care pot fi manipulate de către un atacator;
- să stabilească intrările legale şi să le rejecteze pe toate celelalte;
- să controleze fluența controlului.

Modelul SCAR-ACE realizează aceste cerințe. Mecanismul de control se referă la blocarea tuturor punctelor de intrare posibile prin care un atacator poate avea accces. In cazul unei aplicații distribuite accesul se realizează prin linia de comandă. Prin urmare întregul control trebuie să pornească de la această linie de comandă prin interpretarea acesteia. Linia de comandă a SCAR-ACE este alcătuită din credențial şi parametrii cererii de acces.

In considerarea acestor amenințări se presupune că există puncte de intrare în aplicație prin care un atacator poate introduce valori malițioase.

SCAR-ACE, pentru a reduce posibilitățile de atac, deține credentialul în formă ascunsă (linia de comandă nu este în formă explicită).

Orice cerere emisă de către un utilizator se traduce printr-o linie de comandă compusă din credențial şi parametrii de apel, credențial ale cărui valori sunt în corelație în sensul că:

- unui rol îi corespund anumite stări bine definite;
- dintr-o stare pot fi emise evenimente bine definite;
- pentru fiecare cerere parametrii sunt specifici având numărul şi tipul bine definiti.

Mai mult decât atât, linia de comandă conține informații diverse ilustrate în figura următoare.

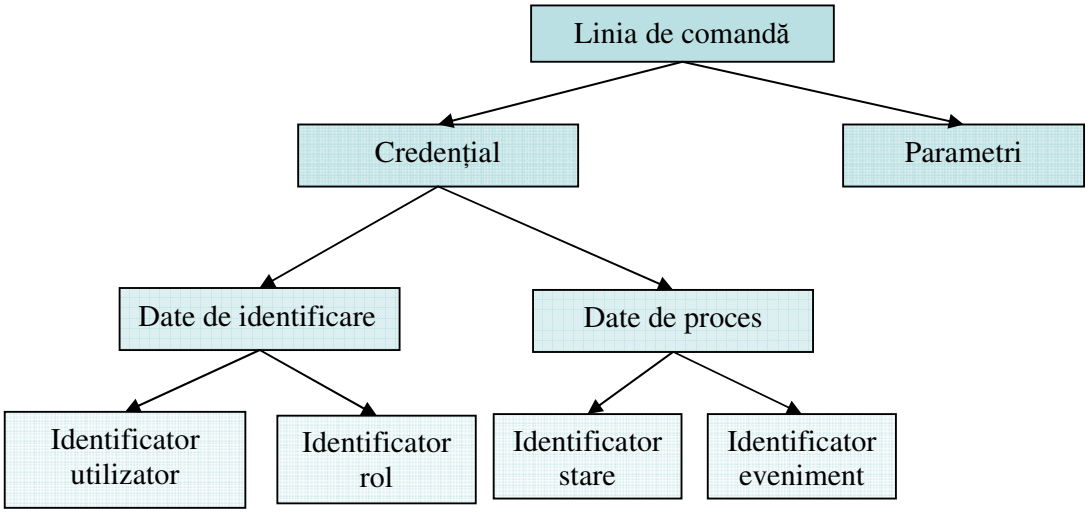

Figura 13. Informațiile liniei de comandă

Serverul, la primirea unui credențial îl interpretează progresiv, identificator cu identificator. Prima validare pe care o face: verifică dacă identificatorul utilizatorului este în memorie. In cazul unei erori (identificator utilizator inexistent) utilizatorul este

notificat şi i se trimite o pagină de eroare cu un mesaj concludent prin care i se specifică faptul că nu i se accordă accesul la nici o resursă sau serviciu.

Dacă primul pas este trecut cu succes se verifică identificatorul rolului şi, în funcţie de rezultatul testului, fie se trece la pasul următor în caz de succes fie se emite o pagină de eroare. Dacă identificatorul rolului este unul valid acesta este trimis unei componente de distribuţie (Dispacher) care dă în prelucrare restul credenţialului corespunzător fiecărui rol (vezi şi figura 14).

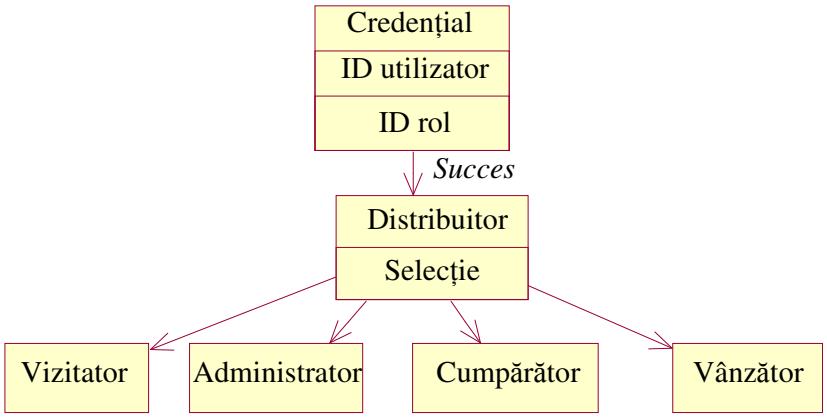

Figura 14. Diagrama de procesare a credenţialului

Starea curentă este specifică fiecărei maşini de stare în parte şi este un element distinct în cadrul grafului de apel. Acest identificator, împreună cu identificatorul rolului şi cu evenimentul, dacă sunt eronate, determină primirea de către utilizator a unui mesaj de acces nepermis. Dacă aceste valori sunt corecte atunci se poate accesa resursa sau serviciile cerute şi se trece la starea următoare. Valoarea acestei stări noi este clar definită şi unică şi serverul o determină din credenţial şi parametrii primiţi şi o trimite clientului. Pentru client această stare va deveni starea curentă.

Dintr-o stare curentă nu se pot emite decât anumite evenimente în funcţie de elementele active ale interfeţei grafice utilizator. In funcţie de valoarea evenimentului se mai trimit serverului parametri suplimentari de apel. Aceşti parametri pot avea valori variabile şi reprezintă atribute obiect sau parametri de metodă în realizarea accesului cerut la resursă sau serviciu. Cu toate că este posibilă transmiterea acestor parametri ei nu fac parte din credenţial ci sunt practic valori ale cererii emise de client în accesarea resurselor sau a serviciilor.

Sursele de cunoştinţe ale modelului sunt divizate în mai multe grupuri şi cuprind diverse entităţi implicate în cadrul procesului de identificare a punctelor vulnerabile.

1. Prima dintre sursele de cunoştinţe o reprezintă *analizorul liniei de comandă* care separă câmpurile constituente ale acesteia (ale liniei de comandă) pentru a le da o semnificaţie validă. In cazul unui eşec se emite mesaj de eroare.
2. Sursa de cunoştinţe următoare este *lista utilizatorilor curenţi* în cadrul căreia se verifică următorul câmp de identificare şi anume identificatorul utilizatorului. In mod uzual acesta este un număr mare generat de către sistem şi memorat în lista utilizatorilor care au o sesiune deschisă pentru rolul determinat. Este un câmp temporar valabil pe durata unei sesiuni

care nu se păstrează în mod persistent. Totodata acest modul verifică atât cardinalitatea cât şi cazul de roluri mutual exclusive. In caz de succes se trece în modulul următor de verificare a datelor de proces.

3. Următorul nivel care reprezintă o sursă de cunoştinţe este un *modul distribuitor* care verifică dacă rolul utilizatorului este unul valid. In caz de succes acesta distribuie câmpurile rămase mai departe spre analiză şi acordarea accesului.
4. Grupul următor de cunoştinţe este alcătuit din *maşina de stare* specifică fiecărui rol. Acest modul verifică informaţiile de proces şi permite accesul la resursele solicitate de către utilizator. Verificarea implică accesarea bazei de date pentru validarea perechii <stare, eveniment>, dacă este o stare permisă, dacă are ataşat un serviciu sau dacă necesită parametri. In cazul în care sunt necesari, parametri verifică dacă aceştia respectă numărul şi tipul necesare. In caz de eşec se emite mesaj de eroare iar în caz contrar, de succes, se trece la accesarea serviciului cerut, rularea acestuia şi trecerea în starea următoare.

3.7 Rolul interfeţelor grafice utilizator

In [P93] este introdus modelul interfeţelor utilizator inteligente care conţine modulele detaliate în figura de mai jos.

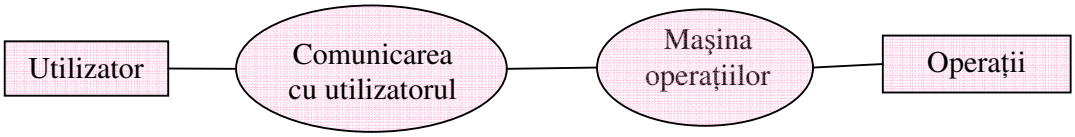

Figura 15. Modelul de comunicare al interfeţelor inteligente

Există cerinţe de cunoaştere a unor funcţiuni suplimentare faţă de cele reprezentate în figura 15 care sunt necesare unei interfeţe utilizator inteligente [C89] precum: cunoaşterea uneltelor software utilizate şi a dispozitivelor hardware pentru furnizarea modelului corespunzător; cunoaşterea mediului organizaţional; interfaţa, pe de altă parte, trebuie să asigure comunicarea cu utilizatorul, să-i realizeze sarcinile în mod intuitiv.

Utilizând modelul abstract din figură se identifică trei funcţionalităţi primare ale interfeţei inteligente: modelarea operaţiilor, modelarea utilizatorului şi translaţia.

Modelarea operaţiilor într-o interfaţă implică un model satisfăcător care să realizeze toate funcţionalităţile importante. *Modelarea utilizatorului* implică realizarea unei reprezentări a cunoştinţelor pentru operaţiile existente. *Translaţiile* convertesc intenţiile utilizatorului în operaţii, şi operaţiile în răspunsuri identificabile.

Aceste funcţiuni implica cunoaşterea cerinţelor: cunoaşterea operaţiilor şi a domeniilor, cunoaşterea tipurilor de utilizatori şi cunoaşterea modalităţilor de interacţiune. Aceste cerinţe adresează interacţiunea dintre un utilizator şi o aplicaţie.

O altă abordare a interfeţelor grafice utilizator este în funcţie de feedback-ul acestora relativ la un domeniu. Figura următoare descrie procesul de feedback a interfeţelor utilizator din punct de vedere al fluenţei controlului şi modul de clasificare al acestora în adaptive şi neadaptive.

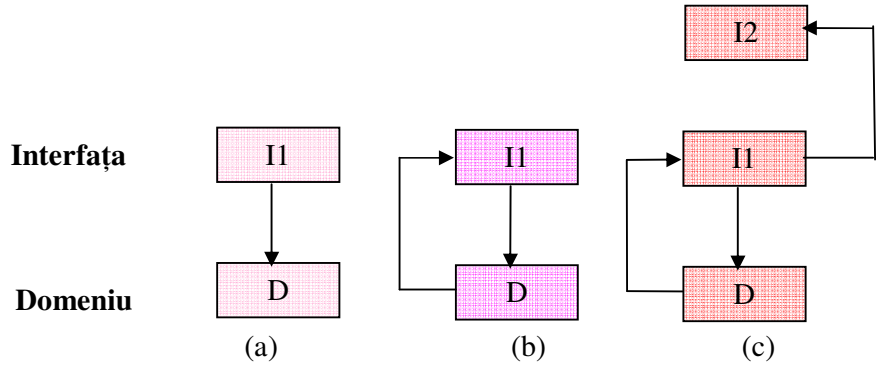

Figura 16. Trei tipuri de sisteme de feedback

In figura 16 se definesc trei tipuri de interacțiuni între un domeniu *D* şi interfețele *I* care îl accesează. In accepțiunea SCAR-ACE, un domeniu *D* reprezintă operațiile posibil a fi realizate pentru un anumit rol iar interfețele *I* reprezintă interfețele grafice prin care utilizatorul comunică cu aplicația.

In figura 16 sistemul (a) este neadaptiv deoarece interfața *I* nu primeşte nici un feedback de la domeniul *D*. Sistemul (b) este adaptiv şi interfața comunică cu domeniul. Sistemul (c) este de asemenea adaptiv dar interfața se modifică în urma comunicării cu domeniul.

In realizarea constrângerilor de autorizare din SCAR-ACE se utilizează interfețe neadaptive (cele de tip (a)) deoarece acestea sunt definite în faza de design şi nu îşi pot modifica atributele constituente ci doar valorile acestora. Aceste interfețe nu pot fi clasificate în mod neapărat ca fiind inteligente. In schimb pot fi calificate ca având cunoştințe relativ la sistemul pe care îl susțin. Aceste cunoştințe sunt încapsulate în stări şi evenimente declanşatoare ale tranzițiilor. Din punct de vedere al adaptabilității, o stare reprezintă instanța unei interfețe sau a unei părți a acesteia şi, împreună cu un eveniment, şi în urma comunicării cu domeniul, determină trecerea la interfața următoare.

Domeniul *D* este o colecție de operații şi o tranziție se execută dacă cererea este autorizată. Mecanismele de navigare între stări sunt implementarea operațiilor prin controlul exercitat de maşina de stare.

Intregul sistem este, în schimb, unul pseudo-inteligent deoarece acțiunea următoare a unui utilizator se realizează prin inferența cu cea anterioară [M+92]. Pentru a fi interfețe complet inteligente ar trebui să participe la un proces de învățare dar aceste interfețe doar recunosc şabloane într-o secvență de interacțiuni şi stabilesc pasul următor.

3.8 Controlul accesului

3.8.1 Maşina de stare

In lucrarea de faţă propun un model pentru controlul accesului în aplicaţiile de comerţ electronic pe baza uneia sau a mai multor maşini de stare. In mod extins orice aplicaţie are o secvenţă de operaţii (procese) şi impunerea unei ordini asupra acestora determină posibitatea tratării prin maşini de stare.

Voi nota cu:

- *U, R* şi *A* setul de utilizatori, setul de roluri şi respectiv setul de acţiuni în cadrul unui sistem distribuit.
- *M, S* şi *T* pentru maşina de stare, stări şi respectiv tranziţii între stări.

Conform [BFA98] o *maşină de stare* este o listă de stări, tranziţii şi operaţii specifice unui rol. Tranziţiile le notez cu $[T_1, T_2, \ldots, T_n]$ unde fiecare T este o 3-tuplă:

$T = <A_i, (RS_i, >_i), act_i>$

unde:

$A_i \in A$,

$RS_i \subseteq R$ este un set de roluri autorizate să execute A_i,

$>_i$ este o ordine locală a rolurilor

act_i reprezintă numărul posibil de activări ale acţiunii A_i.

Definirea noţiunii de *maşină de stare* într-un sistem distribuit conform [GHS95] determină ca în condiţiile în care suprapunem peste un model distribuit o ordine (parţială sau totală) acesta poate fi tratat prin intermediul maşinilor de stare care modelează şi controlează execuţia proceselor într-o aplicaţie distribuită. Fiecare tranziţie este definită ca şi un set de operaţii coordonate care se execută în vederea obţinerii anumitor rezultate. O acţiune poate fi alcătuită din mai multe procese şi realizează deservirea unei cereri prin accesul la o resursă sau un serviciu. Acţiunile într-un sistem distribuit peste care este suprapusă o ordine, sunt interdependente reflectând o activitate coordonată.

In mod tipic o organizaţie stabileşte un set de politici de securitate care impun reguli asupra modului în care se gestionează resursele şi serviciile. In timp ce o politică simplă poate specifica rolul căruia i se poate asigna o acţiune, o politică complexă poate specifica constrângeri de autorizare precum separarea îndatoririlor.

La execuţia unei tranziţii acţiunile sunt asignate rolurilor pe baza unei specificaţii explicite a regulilor de autorizare. Constrângerea de separare a îndatoririlor asigură faptul ca este interzis ca unui singur utilizator să i se dea autoritate suficientă astfel încât să poată să fraudeze sistemul [AW05].

In [BFA98] se examinează diferite tipuri de constrângeri ale autorizării (incluzând cele statice, dinamice şi hibride) şi se propun soluţii de asignare a acţiunilor la utilizatori, într-un sistem bazat pe roluri, astfel încât să nu fie violate constrângerile autorizării. In continuare abordez această tematică în contextul aplicaţiilor de comerţ electronic pe baza maşinilor de stare.

3.8.2 Rolurile

In modelul SCAR-ACE impun o secvenţialitate a operaţiilor în condiţiile în care există constrângeri.

Spre deosebire de aplicaţiile în care un utilizator poate avea mai multe roluri autorizate în cadrul unei sesiuni, în aplicaţiile care necesita controlul accesului fiecare utilizator are un singur rol autorizat, explicit asignat după autentificarea prin nume şi parolă.

Introduc clasificarea următoare asupra rolurilor în funcţie de tipul accesului:

- roluri publice;
- roluri private.

In definirea rolurilor raportarea se face la aplicaţia de comerţ electronic, un rol fiind public respectiv privat în raport cu aceasta. Rolurile publice le definesc ca fiind acele roluri cu acces din afara sistemului (de exemplu vânzător, cumpărător, vizitator).

Rolurile private sunt acele roluri controlate de către sistem, asignate de către inginerul de sistem unor utilizatori cunoscuţi într-un cadru închis. De exemplu recepţionerul cererii sau expeditorul produsului au roluri private.

In contextul lui *R* chiar dacă un utilizator nu poate avea mai multe roluri, un anumit rol poate fi jucat de către mai mulţi utilizatori. Această facilitate, într-o aplicaţie distribuită, este posibilă doar pentru rolurile publice. De exemplu, rolul de cumpărător poate fi avut de către mai mulţi utilizatori. Pentru roluri precum vânzător sau cumpărător utilizatorii sunt mai mulţi, sunt anonimi şi nu pot fi nominalizaţi. Cardinalitatea utilizatorilor asignaţi rolurilor private este, de cele mai multe ori, egală cu 1 sau cu o valoare ştiută şi finită. Utilizatorii rolurilor private sunt de obicei cunoscuţi şi nominalizaţi. De exemplu, pentru rolul de administrator al sistemului este asignat un utilizator, pentru recepţionerul unei cerereri este asignat unul sau mai mulţi utilizatori.

Din punct de vedere al drepturilor detinute există:

- utilizatori *autorizaţi*;
- utilizatori *neautorizaţi*.

Utilizatorii autorizaţi au acces la funcţiile aplicaţiei de comerţ electronic prin perechea *nume_utilizator* şi *parolă* pe când cei neautorizaţi sunt acei utilizatori care folosesc facilităţi pentru care nu este necesară autorizarea.

Intre roluri există o ierarhie, o ordine parţială „>” care încadrează aplicaţia într-un sistem RBAC $_2$. Dacă R_i şi $R_j \in R$ sunt roluri, se numeşte că R_i domină pe R_j dacă $R_i > R_j$. Intre aceste roluri este definită relaţia de moştenire în sensul că rolul dominant moşteneşte funcţionalităţile ale rolului dominat. In cadrul exemplului practic rolurile de vânzător şi cumpărător domină rolul de vizitator şi încapsulează toate facilităţile oferite acestuia.

In afara acestor roluri între care există o relaţie globală există roluri pentru care nu este impusă o relaţie, de exemplu recepţionerul unei cereri nu este subordonat ca şi funcţionalitate unui alt rol ci acesta trebuie doar să respecte ordinea operaţiilor impusă de către aplicaţie şi să se încadreze în restricţiile temporale (de exemplu nu poate

interveni înaintea depunerii unei cereri şi, dacă o cerere a fost emisă, el are un timp rezervat bine stabilit în care să o soluţioneze pozitiv sau negativ).

3.8.3 Constrângeri ale rolurilor şi ale asignărilor utilizator

Constrângerile aplicate asupra utilizatorilor şi asupra asignărilor rolurilor la un set de permisiuni pot fi de câteva tipuri, clasificate în literatura de specialitate [GI96], [AS00], [SC96] în trei categorii în funcţie de momentul evaluării acestora:

(1)Constrângeri statice – acestea pot fi evaluate în afara execuţiei aplicaţiei distribuite. Exemplu de constrângeri statice ar fi: „cel puţin trei roluri pot fi asociate unei aplicaţii".

(2)Constrângeri dinamice – pot fi evaluate doar în timpul execuţiei aplicaţiei. Acestea sunt constrângerile de control al accesului.

(3)Constrângeri hibride – constrângeri care pot fi parţial verificate în afara executării aplicaţiei şi parţial verificate în timpul executării aplicaţiei. O astfel de constrângere ar fi cea conform căreia un utilizator care execută o acţiune A_1 nu poate executa o altă acţiune A_2.

3.9 Definirea formala a modelului

3.9.1 Definiţii şi termeni

In [BFA98] Bertino et all definesc un model formal pentru o aplicaţie de sine stătătoare. In lucrarea de faţă modific şi extind acest formalism în cadrul modelului SCAR-ACE care realizează controlul accesului în aplicaţiile de comerţ electronic.

Pentru definirea formala a modelului SCAR-ACE am realizat un limbaj de exprimare a constrângerilor de autorizare [OG[1]07]. Acest limbaj de specificare a constrângerilor asupra controlului accesului defineşte un set de simboluri constante, variabile şi predicate.

Definiţia 23: O *constantă* poate fi orice membru al uneia dintre următoarele mulţimi: *U* (setul de utilizatori), *R* (setul de roluri), *A* (setul de acţiuni), *S* (setul de stări), *E* (setul de evenimente), *T* (setul de tranziţii) şi *P* (setul de parametri).

Astfel sistemul definit lucrează cu constante din toate aceste tipuri (de exemplu rolurile constante R_VIZITATOR, R_CUMPARATOR, R_RECEPTIONER$\in R$, stările constante ST_HOME, ST_VIEW$\in S$ şi evenimentele constante EV_HOME, EV_VIEW, EV_SELECT $\in E$).

Definiţia 24: Pentru fiecare set de constante se defineşte tipul *variabil.* Astfel există şapte seturi de variabile notate cu V_U, V_R, V_A, V_S, V_E, V_T şi V_P.

Variabilele pot fi cel mai uşor exemplificate pentru cele de tip rol şi anume un utilizator intră în cadrul sistemului în mod obligatoriu pe un rol de securitate redusă (exemplu R_VIZITATOR) şi, la login, el comută pe un rol cu securitate mai mare (exemplu R_CUMPARATOR) variindu-şi astfel starea şi modificându-şi rolul.

In continuare voi nota cu UT termenii utilizator care definesc atât constantele cât şi variabilele utilizator UT = U $\cup V_U$. Similar RT, AT, ST, ET, TT, PT denotă seturile $V_R \cup R$, $V_A \cup A$, $V_S \cup S$, $V_E \cup E$, $V_T \cup T$ şi respectiv $V_P \cup P$.

Un alt set de elemente utilizate în cadrul definirii formale a modelului SCAR-ACE sunt predicatele. Astfel, există mai multe seturi de predicate:

- un set de *predicate de specificație* care definesc modelul formal al aplicației distribuite (tabelul 1);
- un set de *predicate de planificare* care realizează restricțiile impuse de constrângeri asupra seturilor de rol/utilizator care pot executa o operație (tabelul 2);
- un set de *predicate de execuție* pentru execuția aplicației (tabelul 3).

Tabelul 1. Predicate de specificație

Predicat	Definire
`rol`	dacă `rol`(r_i) este adevărat atunci $r_i \in$ RT
`utilizator`	dacă `utilizator` (u_i) este adevărat atunci $u_i \in$ UT
`stare`	dacă `stare` (s_i) este adevărat atunci $s_i \in$ ST
`eveniment`	dacă `eveniment` (e_i) este adevărat atunci $e_i \in$ ET
`tranziţie`	dacă tranziție (s_i, s_j, t_k) este adevărat atunci $\exists$ stările s_i, $s_j \in$ ST pentru care este definită tranziția $t_k \in$ TT
`parametru`	dacă `parametru` (t_i, p_j) este adevărat atunci tranziția $t_i \in$ TT necesită parametrul $p_j \in$ PT
`aparţine`	dacă `aparţine` (u_i, r_j) este adevărat atunci utilizatorul $u_i \in$ UT are rolul $r_j \in$ RT
>	dacă $r_i > r_j$ atunci rolul r_i moşteneşte toate funcționalitățile lui r_j sau, cu alte cuvinte, r_i domină pe r_j în ordinea rolurilor

Tabelul 2. Predicate de planificare

Predicat	Definire
`cannot_do`$_r$	dacă `cannot_do`$_r$ (r_i, a_j) este adevărat, atunci acțiunea a_j nu poate fi asignată rolului r_i
`cannot_do`$_t$	dacă `cannot_do`$_t$ (t_m, s_i, s_j) este adevărat, atunci tranziția t_m din starea s_i în starea s_j nu poate fi realizată
`Must_execute`$_r$	dacă `must_execute`$_r$ (r_i, a_j) este adevărat, atunci acțiunea a_j trebuie să fie executată de către un utilizator aparținând rolului r_i
`Must_execute`$_t$	dacă `must_execute`$_t$ (t_m, s_i, s_j) este adevărat, atunci tranziția t_m trebuie să fie executată din starea s_i în starea s_j
`check`	dacă `check` (t_i, s_j, s_k) este adevărată atunci constrângerile legate de tranziția t_i din starea s_j în starea s_k pot fi verificate în afara executiei aplicației
`panic`	dacă `panic` este adevărată atunci există cel puțin o constrângere care nu poate fi satisfăcută

Tabelul 3. Predicate de execuție

Predicat	Definire
`exec`$_u$	Dacă `exec`$_u$(u_i, a_j, s_k, e_l) este adevărat, atunci acțiunea a_j este executată de utilizatorul u_i prin comutarea de la starea s_k la apariția evenimentului e_l
`exec`$_r$	Dacă `exec`$_r$(r_i, a_j, s_k, e_l) este adevărat, atunci acțiunea a_j este executată de utilizatorul aparținând rolului r_i prin comutarea de la starea s_k la apariția evenimentului e_l
`exec`$_t$	Dacă `exec`$_t$(r_i, t_j, s_k, e_l, p_n) este adevărat, atunci tranziția t_j este executată de utilizatorul aparținând rolului r_i prin comutarea de la starea s_k la apariția evenimentului e_l şi cu parametrii p_n.
`abort`	Dacă `abort` (a_j, s_k, e_l, p_n) este adevărat atunci acțiunea a_i nu a putut fi executată cu succes la comutarea din starea s_k, la apariția evenimentului e_l. Acesta se poate întâmpla fie dacă evenimentul e_l nu este definit pentru starea s_k, fie dacă parametrii p_n nu sunt corespunzători.
`succes`	Dacă `succes` (a_j, s_k, e_l, p_n) este adevărat atunci acțiunea a_i a putut fi executată cu succes la trecerea din starea s_k la apariția evenimentului e_l cu parametrii p_n.

3.9.2 Definirea regulilor

Conform [L89] o regulă este o expresie de forma:

$$H \leftarrow A_1, \ldots, A_n \ \textbf{not}\ B_1, \ldots, \textbf{not}\ B_m, \ n,m \geq 0$$

unde

$A_1, \ldots, A_n$ şi $B_1, \ldots, B_m$ sunt atomi şi **not** denotă negarea prin eşec.

H este *capul* regulii iar $A_1, \ldots, A_n$ **not** $B_1, \ldots,$ **not** B_m sunt corpul acesteia.

In cadrul limbajului utilizat pentru definirea modelului SCAR-ACE există mai multe categorii de reguli în funcție de predicatele accesate.

Definiția 25: O *regulă explicită* este de forma $H \leftarrow$ în care H este fie un predicat de specificație fie unul de execuție.

Regulile explicite reprezintă fapte şi corpul acestora este totdeauna vid.

Regulile explicite al căror cap este un *atom de specificație* determină definirea partii statice a modelului. Sunt definite astfel rolurile, utilizatorii, starile, evenimentele, parametrii si tranzitiile corespunzatoare modelului.

De exemplu:

```
rol (R_VIZITATOR) ←
rol R_CUMPARATOR)←
```

definesc doua roluri valide specifice aplicatiei.

Regulile explicite al căror cap este un *atom de execuție* sunt generate de către sistem în timpul execuției aplicației.

De exemplu:

$\texttt{exec}_t(r_i, t_j, s_k, e_l, p_n)$

ilustrează tranziția t_j pentru rolul r_i.

Definiția 26: O *regulă de atribuire* este de forma:

$$H \leftarrow L_1, \ldots, L_n, \ n \geq 0$$

unde

H este fie $\texttt{must\_execute}_r$, $\texttt{must\_execute}_t$, $\texttt{cannot\_do}_r$, $\texttt{cannot\_do}_t$,

L_i este un *atom de specificație* sau un *atom de execuție*.

O regulă de atribuire exprimă restricții asupra unui set de utilizatori/roluri care pot realiza o anumită tranziție. Regulile $\texttt{must\_execute}_r$ şi $\texttt{must\_execute}_t$ exprimă necesitatea execuției unei acțiuni sau a unei tranziții pentru asigurarea consistenței constrângerilor. In mod similar, dar în sens contrar sunt regulile $\texttt{cannot\_do}_r$ şi $\texttt{cannot\_do}_t$.

Definiția 27: O *regulă de verificare* este de forma:

`check` $(t_i, s_j, s_k) \leftarrow L_1, \ldots L_n$

unde

L_i este un *atom de specificație* corespunzător unei tranziții

Această regulă verifică constrângeri iar în cazul modelului SCAR-ACE acestea sunt reprezentate de către tranziții. In cazul în care constrângerea poate fi verificată în afara aplicației atunci verificarea este statică.

Definiția 28: O *regulă statică* este o regulă care poate fi evaluată fără a se executa aplicația.

Definiția 29: O *regulă dinamică* este o regulă explicită care poate fi evaluată doar în timpul execuției aplicației.

In general există o ordine parțială a rolurilor şi există tranziții specifice fiecărui rol. In anumite cazuri însă (similar celui descris în figura 17) nu se aplică nici o ordine asupra rolurilor care execută anumite tranziții, şi orice rol poate executa acea tranziție. In acest caz tranziția este asociată maşinii de stare vizitator ale cărei tranziții sunt moştenite de către toate rolurile publice autorizate.

In condițiile în care fiecare tranziție are ataşat un rol, se pot defini maşinile de stare specifice rolurilor:

Definiția 30 (maşina de stare): O specificație de *maşină de stare* `M` este o listă de tranziții $[t_1, t_2, ..., t_n]$ în care fiecare tranziție t_i este specificată printr-o tuplă alcătuită din:

- rolul autorizat în efectuarea tranziției;
- starea inițială s_i şi starea finală s_j;
- evenimentul e_k;
- parametrii $p_r, \ldots p_q$.

sau altfel spus

Definiția 31: Fie $t = [t_1, \ldots, t_n]$ secvența dată a tranzițiilor. Maşina de stare a modelului este un graf etichetat G = (N,A) definit după cum urmează:

(1)*N – Noduri*. Fiecare nod în G reprezintă o stare s_i în care `stare(`s_i`)` este adevărat şi $s_i \in$ ST;

(2)*A – Arcuri*. Fiecare arc în G reprezintă o tranziție $t_i \in$ TT care este etichetată cu evenimentul declanşator $e_j \in$ ET, are drept start o stare s_k şi se termină într-o altă stare s_e, $(s_k, s_e) \in$ ST iar pentru execuție are nevoie de parametrii de intrare $p_m \in$ PT.

Definiția 32: Modelul sistemului SCAR-ACE denotat M_M este alcătuit din setul maşinilor de stare şi constrângerile corespunzătoare.

3.9.3 Exemplificare

Modelul formal de control al accesului este ilustrat pe cazul din figura 17. In acest exemplu se selectează două obiecte şi atributele aferente acestora, după care se

realizează o comparare între obiectele selectate. Stările sunt s_0, s_1, s_2, s_3, s_4 şi s_5. Constrângerea constă în impunerea unei secvenţialităţi între stări prin tranziţiile marcate cu săgeţi şi se referă la imposibilitatea comparării acestor obiecte fără o selectare preliminară a acestora. Fiecare obiect are ataşate atribute implicite deci selectarea unor alte atribute nu este obligatorie.

Nu există nici o constrângere relativ la rolul pe care trebuie să-l aibă utilizatorul în momentul efectuării acestei comparaţii, deci oricărui rol i se oferă această facilitate, motiv pentru care aceste tranziţii sunt asignate rolului cu prioritate minimă (R_VIZITATOR). Constrângerile în cadrul exemplului considerat se implementează prin tranziţiile dintre stări.

Astfel, exemplul poate fi descris la modul detaliat astfel:

- Tranziţia t_1: Utilizatorul selectează obiectul 1. Sistemul se găseşte în starea s_1 prin tranziţia din s_0 la apariţia evenimentului EV_START. Este o tranziţie obligatorie;
- Tranziţia t_2: Utilizatorul acţionează un element activ al interfeţei grafice si selecteaza atributele primului obiect. Se emite evenimentul EV_SET_ATRIB şi sistemul realizează tranziţia din starea s_1 în starea s_2. Este o tranziţie opţională;
- Tranziţia t_3: Utilizatorul acţionează un element activ al interfeţei grafice prin care se continuă selectarea altor atribute pentru primul obiect. Se emite evenimentul EV_SET_ATRIB şi sistemul rămâne în starea s_2. Se trimit parametrii care specifică atributul selectat. Este o tranziţie opţională;
- Tranziţia t_4: Utilizatorul acţionează un element activ al interfeţei grafice prin care se trece la selectarea obiectului al doilea. Se emite evenimentul EV_GET_OBJECT şi sistemul realizează tranziţia din starea s_2 în starea s_3. Inaintea realizării tranziţiei este ataşat obiectului 1 atributul/atributele selectate (prin parametrii ataşati tranziţiei). Este o tranziţie obligatorie dacă s-a ajuns în starea s_2 şi s-a terminat selectarea atributelor primului obiect;
- Tranziţia t_5: Utilizatorul acţionează un element activ al interfeţei grafice prin care se trece la selectarea obiectului al doilea fără selectarea atributelor (rămân cele implicite). Se emite evenimentul EV_GET_OBJECT şi sistemul realizează tranziţia din starea s_1 în starea s_3. Este o tranziţie obligatorie dacă nu se doreşte selectarea atributelor pentru primul obiect;
- Tranziţia t_6: Utilizatorul acţionează un element activ al interfeţei grafice prin care se selectează atributele pentru cel de-al doilea obiect. Se emite evenimentul EV_SET_ATRIB şi sistemul realizează tranziţia din starea s_3 în starea s_4. Este o tranziţie opţională;
- Tranziţia t_7: Utilizatorul acţionează un element activ al interfeţei grafice prin care se continuă selectarea altor atribute pentru obiectul al doilea. Se emite evenimentul EV_SET_ATRIB şi sistemul rămâne în starea s_4. Se trimit parametrii care specifică atributele selectate. Este o tranziţie opţională;
- Tranziţia t_8: Utilizatorul acţionează un element activ al interfeţei grafice prin care se trece la compararea între primul şi cel de-al doilea obiect selectat. Se emite evenimentul EV_COMPARE şi sistemul realizează tranziţia din starea s_4 în starea s_5. Inaintea realizării tranziţiei sunt ataşate obiectului 2 atributele selectate (parametrii ataşati tranziţiei). Este o tranziţie obligatorie dacă s-a ajuns în starea s_4 şi s-a finalizat selectarea atributelor celui de-al doilea obiect;

- Tranziția t_9: Utilizatorul acționează un element activ al interfeței grafice prin care se trece la compararea între primul şi cel de-al doilea obiect, fără selectarea atributelor celui de-al doilea obiect. Se emite evenimentul EV_COMPARE şi sistemul realizează tranziția din starea s_3 în starea s_5. Este o tranziție obligatorie dacă nu se doreşte selectarea atributelor celui de-al doilea obiect.

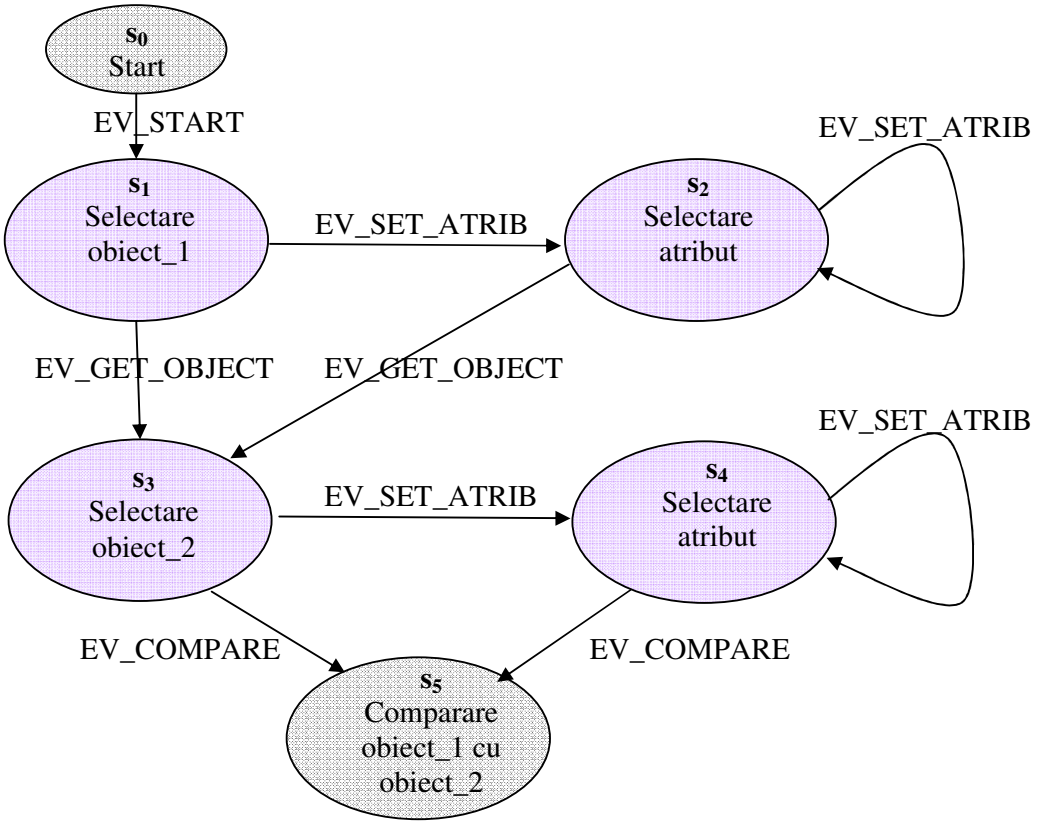

Figura 17. Exemplu de maşină de stare pentru selectarea a două obiecte în vederea comparării lor

In continuare ilustrez specificarea tranzițiilor t_1–t_9 în conformitate cu definiția 30 pentru compararea a două obiecte:

$M = [(t_1, R\_VIZITATOR, (s_0, s_1, EV\_START, \{\})),$
$(t_2, R\_VIZITATOR, (s_1, s_2, EV\_SET\_ATRIB, \{param\_ob_1\})),$
$(t_3, R\_VIZITATOR, (s_2, s_2, EV\_SET\_ATRIB, \{param\_ob_1,$
$param\_atrib_1\})),$
$(t_4, R\_VIZITATOR, (s_2, s_3, EV\_GET\_OBJECT, \{ param\_ob_1,$
$param\_atrib_2\})),$
$(t_5, R\_VIZITATOR, (s_1, s_3, EV\_GET\_OBJECT, \{param\_ob_1,$
$param\_atrib_3\})),$

$(t_6,$ R_VIZITATOR, $(s_3, s_4,$ EV_SET_ATRIB, {param_ob$_1$, param_atrib$_{2|3}$, param_ ob$_2$})),
$(t_7,$ R_VIZITATOR, $(s_4, s_4,$ EV_SET_ATRIB, {param_ob$_1$, param-atrib$_{2|3}$, param_ob$_2$, param-atrib$_4$})),
$(t_8,$ R_VIZITATOR, $(s_4, s_5,$ EV_COMPARE, {param_ob$_1$, param_atrib$_{2|3}$, param_ob$_2$, param_atrib$_5$})),
$(t_9,$ R_VIZITATOR, $(s_3, s_5,$ EV_COMPARE, {param_ob$_1$, param_atrib$_{2|3}$, param_ob$_2$, param_atrib$_6$}))]

Considerând restricțiile impuse de tranzițiile t_1–t_9 se pot ilustra constrângerile CM(M) ale maşinii de stare M construindu-se astfel modelul M_M al SCAR-ACE. Exemplific pe constrângerile tranzițiilor din s_1 în s_4, din s_1 în s_5, din s_2 în s_4 şi din s_2 în s_5.

CM_{1-4}: cannot_do$_t$ (t_{10}, s_1, s_4) ←
(exec$_t$(R_VIZITATOR, t_2, s_1, s_2, EV_SET_ATRIB, {param_ob$_1$}) &
exec$_t$(R_VIZITATOR, t_4, s_2, s_3, EV_GET_OBJECT, {param_ob$_1$, param_atrib$_2$}) &
exec$_t$(R_VIZITATOR, t_6, s_3, s_4, EV_SET_ATRIB, {param_ob$_1$, param_atrib$_{2|3}$, param_ ob$_2$})) |
(exec$_t$(R_VIZITATOR, t_5, s_1, s_3, EV_GET_OBJECT, {param_ob$_1$, param_atrib$_3$}) &
exec$_t$(R_VIZITATOR, t_6, s_3, s_4, EV_SET_ATRIB, {param_ob$_1$, param_atrib$_{2|3}$, param_ ob$_2$}))

CM_{1-5}: cannot_do$_t$ (t_{11}, s_1, s_5) ←
(exec$_t$(R_VIZITATOR, t_2, s_1, s_2, EV_SET_ATRIB, {param_ob$_1$}) &
exec$_t$(R_VIZITATOR, t_4, s_2, s_3, EV_SET_ATRIB, {param_ob$_1$, param_atrib$_2$}) &
exec$_t$(R_VIZITATOR, t_6, s_3, s_4, EV_SET_ATRIB, {param_ob$_1$, param_atrib$_{2|3}$, param_ ob$_2$}) &
exec$_t$(R_VIZITATOR, t_8,(s_4, s_5, EV_COMPARE, {param_ob$_1$, param_atrib$_{2|3}$, param_ob$_2$, param_atrib$_5$})) |
(exec$_t$(R_VIZITATOR, t_5, s_1, s_3, EV_GET_OBJECT, {param_ob$_1$, param_atrib$_3$}) &
exec$_t$(R_VIZITATOR, t_6, s_3, s_4, EV_SET_ATRIB, {param_ob$_1$, param_atrib$_{2|3}$, param_ ob$_2$}) &
exec$_t$(R_VIZITATOR, t_8, s_4, s_5, EV_COMPARE, {param_ob$_1$, param_atrib$_{2|3}$, param_ob$_2$, param_atrib$_5$})) |
(exec$_t$(R_VIZITATOR, t_5, s_1, s_3, EV_GET_OBJECT, param_ob$_1$, param_atrib$_3$}) &

$exec_t$(R_VIZITATOR, t_9,(s_3, s_5, EV_COMPARE, {param_ob$_1$,
param_atrib$_{2|3}$, param_ob$_2$, param_atrib$_6$}))

CM_{2-4}: $cannot\_do_t$ (t_{12}, s_2, s_4) ←
($exec_t$(R_VIZITATOR, t_4, s_2, s_3, EV_SET_ATRIB, {param_ob$_1$,
param_atrib$_2$}) &
$exec_t$(R_VIZITATOR, t_6, s_3, s_4, EV_SET_ATRIB, {param_ob$_1$,
param_atrib$_{2|3}$, param_ ob$_2$}))

CM_{2-5}: $cannot\_do_t$ (t_{13}, s_1, s_5) ←
($exec_t$(R_VIZITATOR, t_4, s_2, s_3, EV_SET_ATRIB, {param_ob$_1$,
param_atrib$_2$}) &
$exec_t$(R_VIZITATOR, t_6, s_3, s_4, EV_SET_ATRIB, {param_ob$_1$,
param_atrib$_{2|3}$, param_ ob$_2$}) &
$exec_t$(R_VIZITATOR, t_8, s_4, s_5, EV_COMPARE, {param_ob$_1$,
param_atrib$_{2|3}$, param_ob$_2$, param_atrib$_5$})) |
($exec_t$(R_VIZITATOR,t_4, s_2, s_3, EV_GET_OBJECT, {param_ob$_1$,
param_atrib$_2$}) &
$exec_t$(R_VIZITATOR, t_6, s_3, s_4, EV_SET_ATRIB, {param_ob$_1$,
param_atrib$_{2|3}$, param_ ob$_2$}) &
$exec_t$(R_VIZITATOR, t_9, s_3, s_5, EV_COMPARE, {param_ob$_1$,
param_atrib$_{2|3}$, param_ob$_2$, param_atrib$_6$}))

3.9.4 Consistența specificării modelului

Un model este specificat într-o manieră consistentă dacă constrângerile încapsulate pot fi satisfăcute. Consistența specificației SCAR-ACE este determinată prin analizarea acestui model (a maşinilor de stare şi a constrângerilor aplicate asupra acestora).

In primul rând dacă predicatul `panic` aparține modelului aceasta implică existența cel puțin a unei condiții care nu poate fi satisfăcută. Din acest motiv, prima condiție de consistență a specificației impune ca predicatul `panic` să nu aparțină modelului.

O altă etapă trebuie să implice analizarea modelului astfel încât să nu conțină informații contradictorii. De exemplu, dacă ambele predicate $must\_execute_t$ (t_m, s_i, s_j) şi $cannot\_do_t$ (t_m, s_i, s_j) aparțin modelului, implică constrângeri în cadrul acestuia care sunt inconsistente deoarece sunt contradictorii.

O constrângere importantă este cea prin care să existe întotdeauna cel puțin un utilizator aparținând unui rol permis care să poată realiza o acțiune dată.

Pentru verificarea inexistenței informațiilor contradictorii trebuie realizată computația tranzițiilor care trebuie să fie permise/nepermise pentru un rol. Această verificare implică baleierea bazei de date care conține maşinile de stare şi verificarea tranzițiilor introduse. Astfel se deosebesc următoarele seturi:

- *Tranziții_Nepermise* (t_m) = $\cup$ { t_m | $\texttt{cannot\_do}_t$ (t_m, s_i, s_j) $\in$ M_M}. *Tranziții_Nepermise* reprezintă setul de tranziții care nu pot fi executate în M_M conform mașinilor de stare;
- *Tranziții_Permise* (t_m) = $\cup$ { t_m | $\texttt{must\_do}_t$ (t_m, s_i, s_j) $\in$ M_M}. *Tranziții_Permise* reprezintă setul de tranziții care pot fi executate în M_M conform mașinilor de stare;
- *Roluri_Nepermise* (r_i) = $\cup$ { r_i | $\texttt{cannot\_do}_r$ (r_i, a_j) $\in$ M_M}. *Roluri_Nepermise* reprezintă setul de roluri care nu pot executa acțiunea a_j în M_M conform mașinilor de stare;
- *Roluri_Permise*(r_i) = $\cup$ { r_i | $\texttt{must\_do}_r$ (r_i, a_j) $\in$ M_M}. *Roluri_Permise* reprezintă setul de roluri care pot executa acțiunea a_j în M_M conform mașinilor de stare
- *Utilizatori_Nepermiși* (u_i) = $\cup$ { u_i | u_i aparține rolului $r_i \wedge \texttt{cannot\_do}_r$ (r_i, a_j) $\in$ M_M}. *Utilizatori_Nepermiși* reprezintă setul de utilizatori aparținând unor roluri care nu pot executa acțiunea a_j în M_M conform mașinilor de stare;
- *Utilizatori_Permiși* (u_i) = $\cup$ { u_i | u_i aparține rolului $r_i \wedge \texttt{must\_do}_r$ (r_i, a_j) $\in$ M_M }. *Utilizatori_Permiși* reprezintă setul de utilizatori aparținând unor roluri care pot executa acțiunea a_j în M_M conform mașinilor de stare.

Definiția 31 (consistența constrângerilor): Fie CM(M) constrângerile unui model bazat pe mașini de stare. CM(M) este consistent dacă și numai dacă sunt îndeplinite următoarele condiții:

- panic $\notin$ CM(M);
- $\forall$ t_m al M_M:
  - Dacă *Tranziții_Permise* (t_m) $\neq \emptyset$, atunci:
    Tranziții_Permise (t_m) $\cap$ *Tranziții_Nepermise* (t_m) = $\emptyset$;
    Tranziții_Permise (t_m) $\subseteq$ { t_m | $t_m \in$ TT};
- $\forall$ r_i al M_M:
  - Dacă *Roluri_Permise* (r_i) $\neq \emptyset$, atunci:
    Roluri_Permise (r_i) $\cap$ *Roluri_Nepermise* (r_i) = $\emptyset$;
    Roluri_Permise (r_i) $\subseteq$ { r_i | $r_i \in$ RT};
- $\forall$ u_i al M_M:
  - Dacă *Utilizatori_Permiși* (u_i) $\neq \emptyset$, atunci:
    Utilizatori_Permiși (u_i) $\cap$ *Utilizatori_Nepermiși* (u_i) = $\emptyset$;
    Utilizatori_Permiși (u_i) $\subseteq$ { u_i | $u_i \in$ UT};
- $\forall$ u_i, r_i ai M_M:
  - Dacă *Utilizatori_Permiși* (u_i) $\neq \emptyset$ și *Roluri_Permise* (r_i) $\neq \emptyset$, atunci:
    $\exists$ $r_j \in$ { $RT_i \setminus$ *Roluri_Nepermise* (r_i)} pentru care
    $\exists$ $u_j \in$ { $RT_i \setminus$ *Utilizatori_Nepermiși* (u_i)}.

3.10 Fazele specificării modelului

In această secțiune prezint paşii în crearea modelului SCAR-ACE. Specificarea schematică a acestor paşi este ilustrată în figura 18, fiecare pas fiind realizat de către o anumită componentă a sistemului de autorizare.

Primul pas, denumit *analiză statică* determină partea statică a constrângerilor modelului şi verifică consistența acestora în confomitate cu definitia 31 (vezi şi 3.9.4). Dacă verificarea se soldează cu eşec pasul este reiterat şi constrângerile sunt modificate până când se elimină inconsistența. Această fază este realizată de către administratorul sistemului de securitate.

Dacă această fază se finalizează cu succes se trece la următorul pas de *verificare/actualizare* prin care se realizează asignarea de acțiuni la roluri, în funcție de faza de analiză statică. In această fază constrângerile se pot modifica astfel încât să fie eliminate regulile redondante.

Faza de *planificare* primeşte la intrare constrângerile şi generează la iesire maşina de stare pentru fiecare rol, prin ataşarea stărilor, tranzițiilor şi a evenimentelor. Dacă acestea nu pot fi realizate se va sesiza administratorul sistemului de securitate pentru reanalizarea modelului.

Dacă această fază se finalizează cu succes se ataşează interfeței grafice partea activă a maşinii de stare respectiv se stabilesc meniurile şi butoanele de comutare de la o stare la alta.

In continuare se vor detaila fazele specificării modelului şi se vor exemplifica pentru cazul descris în figura 17.

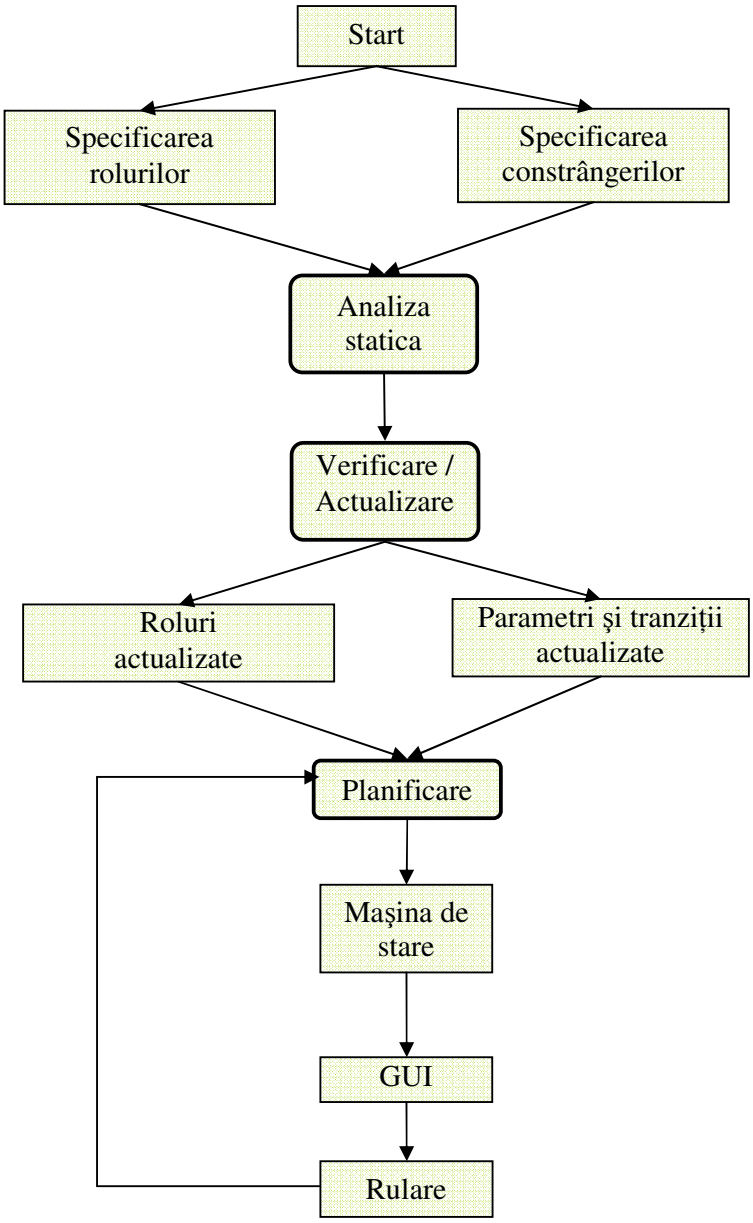

Figura 18. Fazele analizei de specificare a modelului

3.10.1 Faza de analiză statică

In faza de analiză statică se verifică consistența părții statice a modelului. Figura 19 ilustrează intrările şi ieşirile acestei faze.

Algoritmul primeşte la intrare specificarea aplicatiei şi constrângerile acesteia şi verifică consistența părții statice a modelului. In cazul unui eşec se returnează FALSE; în caz contrar se returnează modelul static şi seturile *Roluri_Permise*,

Roluri_Nepermise, Utilizatori_Permişi, Utilizatori_Nepermişi, Tranziţii_Permise şi *Tranziţii_Nepermise*.

Algoritmul 1. *Algoritmul analizei statice*

INPUT: 1) Specificatia aplicatiei;
2) Constrângeri_statice.

OUTPUT: 1) FALSE dacă Constrângeri_statice este inconsistent;
2) Daca Constrângeri_statice este consistent;
a) Seturile de *Roluri_Permise*, *Roluri_Nepermise,*
Utilizatori_Permişi, Utilizatori_Nepermişi
Tranziţii_Permise, Tranziţii_Nepermise;
b) Modelul părţii statice a aplicatiei.

Figura 19. Intrările şi iesirile fazei de analiză statică

Exemplu:

Considerăm exemplul din figura 17. Partea statică a constrângerilor este compusă din tranziţiile t_1–t_9 şi regulile de tipul CM (vezi 5.4.2). Seturile de roluri, utilizatori şi tranziţii pentru figura 17 determinate în faza de analiză statică sunt:

- *Roluri_Permise* = {R_VIZITATOR};
- *Utilizatori_Permişi* = utilizatori în *Roluri_Permise;*
- *Roluri_Nepermise* = *Utilizatori_Nepermişi* = ∅;
- *Tranziţii_Permise* = $\{t_1, t_2, ..., t_9\}$;
- *Tranziţii_Nepermise* = $\{t_{10}, t_{11}, t_{12}, t_{13}\}$.

Consistenţa părţii statice a M_M poate fi verificată imediat prin următoarele:

1) predicatul `panic` ∉ CM (M);
2) *Tranziţii_Permise* $(t_i) \cap$ *Tranziţii_Nepermise* $(t_i) = \varnothing$;
3) *Roluri_Permise* $(r_i) \cap$ *Roluri_Nepermise* $(r_i) = \varnothing$;
4) *Utilizatori_Permişi* $(u_i) \cap$ *Utilizatori_Nepermişi* $(u_i) = \varnothing$;
5) $\exists\ r \in$ RT care să realizeze fiecare tranziţie t_i.

3.10.2 Faza de verificare / actualizare

Scopul fazei de verificare este de a modifica specificaţiile fazei de analiză statică pentru a eficientiza execuţia fazelor următoare.

In cadrul fazei de verificare se determina rolurile permise care aparţin modelului. Dacă setul de *Roluri_Permise* nu este vid, toate rolurile care nu aparţin

acestuia (şi prin urmare aparţin la *Roluri_Nepermise*) pot fi eliminate din specificaţie deoarece asignarea acestora ar viola constrângerile modelului. Dacă setul de *Roluri_Permise* este vid atunci acţiunile din partea statică nu pot fi executate de niciunul dintre roluri deci trebuie revizuite.

In cadrul fazei de analiză statică se examinează seturile de *Tranziţii_Permise* şi *Tranziţii_Nepermise*. Regulile care se aplică în faza de verificare/actualizare asupra rolurilor se aplică şi asupra tranziţiilor. Astfel, dacă setul de *Tranziţii_Permise* nu este nul se elimină din setul de tranziţii toate cele care nu aparţin acestui set. Dacă setul de tranziţii este nul, setul de operaţii asignat acestora îşi pierde consistenţa, deci trebuie revizuit.

Un exemplu în cazul de faţă ar fi extinderea setului de *Tranziţii_Nepermise* cu toate acele tranziţii ce nu pot fi executate pentru exemplul considerat în figura 17.

Intrările/ieşirile fazei de verificare/actualizare sunt ilustrate în figura 20. Astfel se primeşte la intrare rezultatul fazei de analiză statică iar la ieşire, în urma verificării, se actualizează seturile de roluri, utilizatori şi tranziţii. Tot ca şi rezultat, în urma analizării tranziţiilor se determină elementele active şi pasive ale interfeţei grafice caracteristice atât stărilor maşinii de stare precum şi parametrii tranziţiilor.

Algoritmul 2. *Algoritmul fazei de verificare/actualizare*

INPUT: 1) Modelul părţii statice;
2) Seturile de *Roluri_Permise*, *Roluri_Nepermise,*
Utilizatori_Permişi, Utilizatori_Nepermişi
Tranziţii_Permise, Tranziţii_Nepermise.

OUTPUT: 1) Seturile verificate/actualizate de:
Roluri_Permise, *Roluri_Nepermise,*
Utilizatori_Permişi, Utilizatori_Nepermişi
Tranziţii_Permise, Tranziţii_Nepermise;
2) Elementele active şi pasive ale interfeţei grafice corespunzătoare stărilor maşinii de stare.
3) Parametrii tranziţiilor.

Figura 20. Intrările şi iesirile fazei de verificare / actualizare

Pentru determinarea elementelor active şi pasive ale interfeţei grafice corespunzătoare unei stări se iau în considerare tranziţiile posibile din fiecare stare în parte şi, pentru fiecare tranziţie, se generează un element activ. Pentru informaţiile de selectat şi de trimis sistemului ca şi parametri ai tranziţiilor vor fi constituite elementele pasive care sunt atribute ale obiectelor interfeţei grafice. Elementele active şi pasive determinate, specifice interfeţei grafice utilizator, sunt ataşate fiecărei stări în mod consistent şi neredondant.

Pentru determinarea parametrilor fiecărei tranziţii se iau în considerare elementele pasive (sau un subset al acestora).

Exemplu:
Pentru exemplul considerat în figura 17 setul de intrare al fazei de verificare/actualizare este constituit din rezultatele fazei de analiză statică (vezi 5.5.1). Setul de ieşire este alcătuit din:

1) *Roluri_Permise* = {R_VIZITATOR, R_CUMPARATOR, R_VANZATOR};
   Roluri_Nepermise, Utilizatori_Permişi, Utilizatori_Nepermişi, Tranziţii_Permise rămân nemodificate ;
   Tranziţii_Nepermise poate fi extinsă cu $CM_{2,4}$, $CM_{2,5}$ respectiv t_{14} şi t_{15}.
2) Exemplificare elemente active şi pasive pentru starea s_1:
   Elemente active:
   - element activ: un buton ce declanşează tranziţia în s_2 prin emiterea evenimentului EV_SET_ATRIB;
   - element activ : un buton ce declanşează tranziţia în s_3 prin emiterea evenimentului EV_GET_OBJECT;

   Elemente pasive:
   - identificator obiect;
   - atribute obiect;
   - imagine obiect.

3.10.3 Faza de planificare

Această fază generează asignările acţiunilor la roluri şi a utilizatorilor la roluri astfel încât toate constrângerile asociate modelului să fie satisfăcute. Această fază este realizată înaintea execuţiei aplicaţiei.

După ce rolurile private au fost determinate se realizează asignarea utilizatorilor la acestea, pentru cele publice neputându-se face nominalizarea.

Pentru asignarea tranziţiilor la acţiuni se consideră fiecare operaţie şi se determină acţiunile de realizat şi setul de tranziţii corespunzător. De exemplu pentru operaţia de inserare a unui produs în coşul de cumpărături acţiunile de realizat sunt: selectarea produsului şi a atributelor acestuia şi adăugarea acestuia în coş. Există o singură tranziţie necesară realizării operaţiei şi anume cea din starea definită prin pagina în care este setul de produse în starea în care coşul a mai primit un produs nou.

Ultima etapa a planificării o constituie verificarea tranziţiilor constituente ale setului de constrângeri fără executarea aplicaţiei.

Exemplu:

Pentru ca sistemul să poată comuta între stări se execută tranziţii şi fiecărei tranziţii îi corespunde cel putin o acţiune care asigură trecerea în starea următoare.

In faza de planificare se asignează fiecărui rol stările, respective tranziţiile caracteristice în funcţie de acţiunile de efectuat.

Algoritmul fazei de planificare:

INPUT:

1) CM(M);

2) *Roluri_Permise, Utilizatori_Permişi, Tranziţii_Permise.*

OUTPUT:

1) Predicate de planificare;

2) Asignarea actiunilor la roluri;

3) Asignarea utilizatorilor la rolurile private;

4) Secvența de tranziții;

5) Acțiunile specifice fiecărei tranziții.

Figura 21. Intrările şi ieşirile fazei de planificare

Pentru exemplul analizat (figura 17) considerăm t_2 prin care se realizează operația de selectare a obiectului 1. Acțiunea de realizat poate fi identificată (de exemplu) cu a_2 – selectare obiect - şi are predicatul de planificare:

$must\_execute_r$ (R_VIZITATOR, a_2)

Tranziția de executat are predicatul de planificare:

$must\_execute_t(t_2, s_1)$

Verificarea constrângerilor legate de tranziția t_2 se realizează prin:

$check(t_2, s_1, s_2)$

Rolurile care pot executa t_2 sunt următoarele roluri publice (prin moştenirea de la R_VIZITATOR): R_VIZITATOR, R_CUMPARATOR, R_VANZATOR.

Pentru rolul de administrator si receptioner al unei cereri se nominalizeaza utilizatorii.

3.10.4 Definirea maşinii de stare

In această fază se realizează graful de apel, pentru fiecare rol pornindu-se de la predicatele de planificare şi de la secvența de tranziții.

Datele de intrare şi cele de ieşire sunt ilustrate în fig. 22.

Pentru exemplul din figura 17 datele de intrare sunt:

1) rolul R_VIZITATOR
2) tranzițiile t_1– t_9
3) predicatele de planificare rezultate în faza anterioară:

Algoritmul maşinii de stare:

INPUT:

1) setul de roluri;

2) secvența de tranziții;

3) predicate planificare.

OUTPUT:

1) G = (N,A) pentru fiecare rol.

Figura 22. Datele de intrare/ieşire ale maşinii de stare

Datele de ieşire sunt constituite din maşina de stare corespunzătoare ilustrată în figura 23.

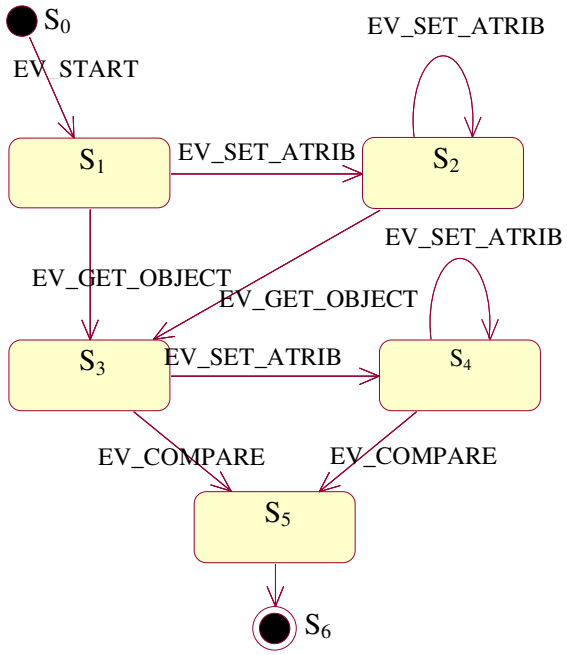

Figura 23. Maşina de stare pentru exemplul din figura 17

3.10.5 Interfața utilizator grafică

După realizarea fazei anterioare componența maşinii de stare este adusă într-o stare funcțională.

Faza curentă are drept scop realizarea interfeței grafice cu utilizatorul pornind de la masina de stare. Astfel, sunt definite paginile utilizator care pot fi alcătuite din unul sau mai multe cadre. Fiecărei scene îi este corespondentă o stare în cadrul maşinii de stare.

Algoritmul fazei de realizare a interfeței grafice cu utilizatorul:

INPUT:
1) G = (N, A) pentru fiecare rol.

OUTPUT:
Interfața grafică cu utilizatorul.

Figura 24. Faza de realizare a interfeței grafice cu utilizatorul

Pentru exemplul din figura 17 voi ilustra interfața grafică rezultată cu pagina reală din implementare:

Figura 25. Interfața grafică pentru selectarea si compararea a două obiecte

3.10.6 Faza de rulare

Faza de rulare este realizată prin executarea acțiunilor specifice tranzițiilor componente ale mașinii de stare. La execuția unei acțiuni utilizatorul este verificat dacă are drepturile de autorizare necesare realizării acesteia.

Acest mecanism se realizează în două etape corespunzătoare autorizării:

- Autentificare;

- controlul accesului.

Autentificarea este realizată de către sistem şi constă în verificarea numelui şi parolei furnizate de către utilizator.

La execuţie utilizatorul urmează trasee în cadrul maşinii de stare specifice rolului pe care acesta îl posedă ceea ce înseamnă că întotdeauna se află în una din stările definite ale lui G.

Dacă utilizatorul doreşte realizarea unei acţiuni, acesta activează un element activ al interfeţei grafice. Această acţiune poate avea două consecinţe:

(1) Dacă elementul activ este valid pentru acel utilizator, selecţia acestuia va determina apariţia unui eveniment, declanşarea tranziţiei şi, drept consecinţă, realizarea acţiunii cerute.
(2) Dacă elementul activ nu este valid nu se va declanşa un eveniment.

Similar, există şi alte surse posibile de eroare precum existenţa stărilor izolate de sistem, sau încercarea realizării unei tranziţii nepermise.

In cazul în care tranziţia a fost executată cu succes se introduc reguli de tipul:

$exec_t\ (r_i, t_j, s_k, e_e, p_n) \leftarrow$

şi

$succes\ (a_j, s_k, e_e, p_n) \leftarrow$

Dacă oricare dintre erori este semnalată în cadrul fazei de execuţie se introduc predicate de tip `abort` şi sistemul este modificat şi trecut din nou în faza de planificare pentru corectarea realizării acestuia. De notat că faza de planificare nu trebuie reluată de la început ci poate fi invocată numai pentru acele tranziţii care au semnalat erori şi doar în cazul cel mai defavorabil se va relua faza de planificare de la început. Acest algoritm este unul incremental până la realizarea completă a aplicaţiei.

4 Analiza aplicațiilor de comerț electronic şi implementarea modelului SCAR-ACE

Pentru validarea modelului SCAR-ACE s-a implementat o aplicație distribuită de comerț electronic. Pentru aceasta s-a realizat preliminar analiza acestui tip de aplicații [O00].

4.1 Introducere

Termenul de *comerț electronic pe Internet* desemnează utilizarea Internetului pentru vânzarea şi cumpărarea de bunuri şi servicii, incluzând suport după vânzare [TS98]. Internetul poate fi un mecanism eficient pentru publicitate şi distribuirea informațiilor despre produse dar scopul lui principal in cazul aplicatiilor comerciale sunt tranzacțiile de afaceri.

Comerțul pe Internet este unul dintre multiplele tipuri de comerț electronic iar implementarea unei aplicații de comerț electronic se supune modelului propus deoarece poate fi asimilată şi tratată ca şi o secventă dată de stări şi tranziții între stări pentru care sunt definite constrângeri.

Comerțul electronic are o istorie mai veche, şi se referă la modul de a pune în legătură furnizorii de bunuri cu cumpărătorii, incluzând utilizarea calculatoarelor şi a tehnologiilor de comunicație în afaceri financiare, sisteme de rezervare a biletelor de avion, procesarea comenzilor, gestionarea inventarelor, etc.

In comerțul pe Internet comunicarea are loc într-o rețea publică partajată. Mai important, Web-ul permite tranzacții spontane între cumpărători şi vânzători fără a exista o relație anterioară între aceştia.

In primul rând, şi cel mai important, comerțul pe Internet se referă la afaceri: utilizarea efectivă a rețelelor pentru a obține scopul propus al afacerii.

Aplicațiile de comerț pe Internet implica mai multe componente şi tehnologii: Web, baze de date, rețele de mare viteză, algoritmi de criptare, multimedia, etc. Punerea acestora impreună pentru a forma un sistem securizat, de mare performanță, poate fi o provocare. In timp, comerțul pe Internet a suferit modificări diverse şi a crescut vertiginos în ultimii ani.

4.2 Lanțul valoric

Comerțul pe Internet se poate privi în termenii lanțului valoric necesar unei afaceri. In parte, lanțul valoric este relativ la afacerea propriu-zisă (de exemplu vânzarea de cărți). Pe de altă parte lanțul valoric se adresează realizării afacerii on line. Ințelegerea acestor două părți (a afacerii propriu-zise si a afacerii on-line) şi a modului de îmbinare a acestora este un pas important în realizarea de aplicații de comerț electronic pe Internet.

Componentele generale ale lanțului valoric sunt [M94]:

- Atragerea clienților;
- Interacțiunea cu clienții;

- Acţionarea la instrucţiunile clientului;
- Reacţia la cererile clienţilor.

Componentele cheie ale lanţului valoric pot fi foarte diferite pentru industrii diferite, şi chiar în cadrul aceleiaşi industrii, pentru afaceri diferite.

4.2.1 Atragerea clienţilor

Prima componentă a lanţului valoric pentru o aplicaţie generică de comerţ electronic este atragerea clienţilor. Prin aceasta se înţeleg mijloacele utilizate de a aduce un client nou, de a-l face interesat într-un site: prin reclama pe Web, mail electronic, televiziune, afişe sau alte forme de marketing [B06]. Scopul acestei faze este trezirea interesului clienţilor în a vedea cataloagele de produse sau a altor informaţii despre produsele şi serviciile propuse vânzării.

4.2.2 Interacţiunea cu clienţii

A doua componentă este interacţiunea cu clienţii si se referă la comandarea produselor. Această fază este în general orientată pe produs şi include cataloage, publicaţii sau alte informaţii disponibile pe Internet. Conţinutul poate fi distribuit în diferite forme precum WWW, poştă electronică sau alte mijloace media precum CD-ROM-uri.

Conţinutul se poate modifica mai rar sau frecvent. Tehnic, conţinutul poate fi static sau dinamic. Conţinutul static constă din pagini pregătite cum ar fi cele dintr-un catalog care sunt trimise unui client la cerere. Aceste pagini se modifică atunci când se schimbă informaţia conţinută pe acestea. Conţinutul dinamic este generat la momentul emiterii unei cereri şi este alcătuit de la una sau mai multe surse pentru a produce o pagină în conformitate cu cerinţa clientului. Sursele de creare a conţinutului dinamic includ bazele de date, iar partea dinamică poate fi de exemplu preţul unui produs.

4.2.3 Acţionarea la instrucţiunile clientului

Componenta următoare este acţionarea la instructiunile clientului. Odată ce un cumpărător a căutat printr-un catalog şi doreşte să cumpere un produs, trebuie să existe o modalitate de a oferi acestuia posibilitatea de a efectua o comandă, să proceseze plata şi să se realizeze livrarea produsului. Componentele acestei etape constau din: procesarea comenzii, plata produselor şi livrarea acestora [HH02].

Procesarea comenzii. Uneori cumpărătorul doreşte să cumpere mai multe produse în acelaşi timp, aşadar procesarea comenzii trebuie să includă posibilitatea ca mai multe entităţi să poată fi grupate pentru cumpărarea ulterioară. Această facilitate este denumită coşul de cumpărături pentru tranzacţiile cu de amănuntul şi, în mod uzual, include posibilitatea de a-şi modifica conţinutul în orice moment. Astfel, cumpărătorul are facilitatea de a descărca entităţi, de a adăuga unele noi sau să modifice cantităţile.

Plata. In funcţie de termenii comenzii, cumpărătorul poate plăti odată ce comanda este finală. Ca şi în viaţa reală există mai multe posibilităţi de a plăti: carţi de credit, ordine de plată, etc.

Livrarea. Pasul următor depinde de tipul item-urilor cumpărate. Dacă sistemul a comandat un bun fizic acesta va fi livrat cumpărătorului. In acest caz sistemul trebuie să aibă un mijoc de plasare a comenzilor care să impacheteze bunul şi să îl trimită.

Un alt tip de comandă este unul pentru un serviciu care să fie executat în lumea reală. De exemplu cineva a comandat o telegramă cântătoare online. Pentru realizarea acestui serviciu se realizează un mecanism similar celui anterior şi anume trebuie să existe un departament către care cererea să fie trimisă spre procesare, departament care va realiza serviciul cerut.

Un al treilea tip de item cerut este mai apropiat sistemului electronic şi poartă denumirea de bun digital. Bunurile digitale includ o paletă largă de servicii cu distribuire online, incluzând software, ziare sau articole de noutăţi, rapoarte, accesul la baza de date pentru o anumită perioadă de timp.

4.2.4 Reacţia la cererile clienţilor

După ce vânzarea a fost completă clientul poate întâmpina dificultăţi sau să aibă întrebări relativ la bunul achiziţionat care necesită service. Cele mai multe întrebări necesită o persoană care să raspundă, la altele poate fi sistemul cel care răspunde.

4.3 Modele de afaceri

In funcţie de cerinţele sistemului şi a opţiunilor de design se pot enumera următoarele tipuri de segmente de afaceri [T05]:

- Retail sau Business-to-Consumer (B2C)

Afacerea prin care se vând bunuri direct unui consumator final individual.

- Business-to-business (B2B)

Sunt afacerile cu cataloage de vânzare online prin care se comercializează bunuri către alte afaceri. In această categorie intră MRO (maintenace, repair and operation) şi COGS (cost of goods and services). COGS implică comenzi mari pentru productie şi tehnologii precum EDI şi XML.

- Comerţul informaţional

Aceasta este o categorie largă si include afaceri care planifică distribuirea bunurilor digitale (produse informaţionale şi servicii). Sunt incluse publicaţiile online şi distribuirea de software online.

In sensul marketingului, un segment de afaceri este o colecţie de companii din aceeaşi arie, cum ar fi producătorii de automobile, redactorii publicaţiilor, care au aceleaşi cerinţe pentru produse şi servicii. Se utilizează denumirea de segment pentru a descrie colecţii de afaceri cu cerinţe similare pentru comerţul pe Internet.

4.3.1 Similarităţi şi diferenţe la nivel de segment

Există mari similarităţi în cerinţele celor trei segmente descrise [LL98]. Toate trebuie să atragă clienţii, să prezinte produse, să asambleze comenzi, să realizeze tranzacţii, şi să distribuie bunurile comandate. Există însă şi diferenţe. Spre exemplu segmentul de retail are mare nevoie de capabilităţi de comercializare, în timp ce aplicaţiile business-to-business au cerinţe majore în sistemele de plată şi pentru aprobarea fluxului producţiei.

Sistemul de comercializare a informaţiilor este cel mai distinctiv dintre cele trei deoarece bunurile digitale nu necesită distribuţie fizică a produselor şi serviciul de vânzare către client (ca şi celelalte), dar are mari necesităţi de îndeplinire online.

4.3.2 Participanţi la sistem

Sistemele de comerţ pe Internet au mai multe tipuri de participanţi în funcţie de rolul şi scopul acestora [PS05], [EE01].

Cumpărătorii cer un set de operaţii; designer-ii de cataloage, reprezentanţii serviciilor client şi operatorii de sistem fiecare are setul lui de cerinţe. Pentru anumite afaceri, în mod particular pentru cele de dimensiune redusă, o persoană poate îndeplini toate aceste sarcini. Pentru afacerile de amploare persoane diferite realizează sarcini diferite. Considerarea separată a rolurilor permite satisfacerea cerinţelor unei aplicaţii de orice mărime, permiţând unei afaceri de mică întindere să crească fără a reconsidera rolurile persoanelor implicate în fiecare etapă.

Discutând în termeni de roluri, acest lucru implică şi eliminarea oricărei confuzii. De exemplu, simpla referire la un client nu distinge cazurile când o persoană selectează un produs pentru a fi cumpărat şi o alta care să realizeze plata. Prin definirea operaţiilor unui rol particular se asigura că toate functionalitatile necesare acelui rol sunt prezente în sistem.

Cumpărători. In context, cumpărător însemnă consumator. Este necesar ca un sistem de comerţ să realizeze necesităţile consumatorilor, dar diferite tipuri de cumpărători necesită lucruri diferite, şi interesele lor sunt uneori antagoniste celor ale vânzătorilor. Consumatorii se departajează în:

- consumatori retail - aceştia sunt consumatorii care utilizează sistemul de comerţ B2C.
- consumatorii de business - sunt acei cumpărători care utilizează sisteme de comerţ B2B.

Vânzători. In context, termenul vânzător include furnizorii angajati în sistemele B2C, B2B sau furnizorii de publicaţii şi conţinut în sistemele de comerţ informaţional.

Procesoare financiare. Acestea operează partea de procesare a cărţilor de credit care acceptă tranzacţii de la furnizori şi trimite datele necesare cumpararii (precum numarul cartii de credit si data la care acesta expira) mai departe băncii furnizorilor. Securitatea este cuvântul cheie pentru un procesor financiar. Securitatea include ca înregistrările să fie private şi clare, şi sa fie asigurate autenticitatea şi integritatea cererilor.

Tehnicieni. Tehnicienii au un rol major în crearea sistemelor de comerţ pe Internet. Pentru a se expanda piaţa pentru comerţul pe Internet este necesară construirea produselor în jurul unui nucleu tehnologic. Ţelul tehnicienilor este acela de a construi un sistem simplu şi general care rezolvă majoritatea problemelor.

In continuare se discută câteva roluri atât pentru cumpărător cât şi pentru vânzător [FL00]:

Roluri client. In orice tranzacţie comercială există un vânzător şi un cumpărător. Se utilizează mai multe cuvinte pentru cumpărător: client, consumator, agent de cumpărări, etc. Pe Internet se utilizează cuvintele client şi browser cu acelaşi înteles, cu referire mai mult la software decât la persoană. Dar există diferenţe subtile între aceste cuvinte şi distincţia reflectă anumite roluri pe partea de cumpărător. In anumite cazuri cum ar fi client de achiziţionări, aceeaşi persoană joacă mai multe roluri fără chiar a ne gândi la aceasta. Există aşadar într-o afacere mai multe roluri:

- Specificator - această persoană selectează bunul de cumpărat;
- Intermediar - această persoană aprobă bunul selectat de specificator;
- Cumpărător - această persoană negociază termenii şi condiţiile unei achiziţii şi specifică plata;
- Recipient - această persoană primeşte produsul cumpărat.

In plus se pot distinge tipuri de cumpărători în funcţie de relaţia acestora cu furnizorul:

- Cumpărător anonim - nu are nici o relaţie cu vânzătorul şi poate să nu creeze nici una la realizarea unei achiziţii;
- Client membru - sunt cei care cumpără în mod repetat de la un vânzător şi au stabilit un soi de relaţie numită relaţie de membru. Membrii se identifică în sistem deoarece acesta le oferă anumite beneficii.

Relaţia de membru dă naştere unui alt rol şi anume *administrator al membrului*: o persoană care acţionează în acest rol şi poate modifica sau actualiza orice informaţie de profil memorată despre membru. Dacă relaţia de membru cuprinde câteva conturi individuale, ca cele pentru o familie de membri sau agenţi multipli, administratorul poate seta limitări în utilizarea conturilor individuale. Aceste limitări pot fi de genul: tipurile de produse de cumpărat, cantitatea acestora, timpul zilei pentru cumpărături, etc.

Aceasta ne spune că un sistem de comerţ pe Internet necesită o modalitate prin care diferiţi oameni pot modela diferite părţi ale unei tranzacţii, şi cum o singură persoană poate mânui toate aceste roluri. Consumatorii nu presupun a-şi schimba rolul foarte des şi în mod explicit la fiecare fază dar se aşteaptă să aibă un proces uşor de realizare a cumpărăturii. Companiile, pe de altă parte, cele care fac distincţie între roluri diferite doresc să gestioneze tranzacţiile dintr-un rol în altul în mod facil şi eficient.

Rolurile furnizorului. De partea cealaltă a unei tranzacţii este vânzătorul. Există diferite roluri într-un sistem de comerţ electronic pentru furnizori. Determinarea tuturor rolurilor la început face posibil ca afacerea pe Internet să crească într-un mod facil pe măsură ce mai multe persoane se raliază echipei şi rolurile devin mult mai distincte. Pentru vânzător există două grupe principale de roluri: rolurile afacerii şi conţinutului şi rolurile echipei operaţionale.

Rolurile următoare sunt cele mai importante roluri ale afacerii:

- Manager afacere: Responsabil cu abordarea afacerii relativ la Internet, crearea şi operarea în prezența Internetului - decide care produse şi servicii urmează a fi comercializate si determină prețurile;
- Arhitectul de comerț pe Internet: Este un analist de sistem care transformă cererile afacerii în design sistem; acesta creaza şi gestioneaza conținutul (cataloagele), proceseaza tranzacțiile, se ocupa de aspectele tehnice ale seviciului clienți;
- Designer de conținut: Este responsabil cu imaginea sistemului de comerț pe Internet incluzând designul grafic;
- Autor de conținut: Crează şi adaptează informațiile despre produse într-o formă care poate fi utilizată în comerțul pe Internet;
- Implementator: Crează programele software necesare funcționării sistemului.
- Administratorul bazei de date: Gestionează crearea şi operarea bazei de date pentru a-i asigura corectitudinea, integritatea şi performanța;
- Serviciul vânzări şi marketing: Responsabil cu promovarea sistemului de comerț bazat pe Internet pentru afacere;
- Serviciul clienți: Răspunde la întrebările despre produse, asistă cumpărătorii în procesul de achiziție, gestionează returnarea produselor.

Rolurile operaționale sunt următoarele:

- Manager operații: Responsabil cu gestionarea activităților pentru sistemul de comerț electronic;
- Supervizor sistem: Gestionează sistemul;
- Administrator sistem: Responsabil cu operațiunile tehnice ale sistemului de calculatoare şi ale rețelei;
- Ofițer de securitate: Asigură că s-au luat măsuri adecvate de securitate în designul şi implementarea sistemului de comerț pe Internet;
- Agent vânzări: Responsabil cu împachetarea şi trimiterea bunurilor sau cu serviciul de distribuire;
- Contabil: Responsabil cu serviciul de contabilitate care este asociat sistemului de comerț pe Internet.

4.4 Asigurarea intimității clientilor

4.4.1 Platforme pentru preferința intimității

Consorțiul WWW a lansat un grup de lucru denumit Platforma pentru preferințele intimității (Platform for Privacy Preferences - http://www.w3.org.P3P/). Pagina Web a acestui grup specifică următoarele [HTTP[1]03]:

> *"Proiectul preferințelelor de intimitate (P3) va rezulta în specificarea şi demonstrarea unui mod interactiv de exprimare a practicilor şi preferințelor de intimitate a site-urilor Web şi respectiv a utilizatorilor acestora."*

Un număr de organizaţii incluzând Netscape, Microsoft, Firefly şi Verisign cooperează în cadrul efortului W3 pentru specificarea intimitatii, utilizând Open Profiling Standard al Firefly ca şi punct de pornire. Asigurarea intimitatii clientilor impune ca oricând un site cere informaţii personale sau de profil ale unui utilizator, acestea pot fi furnizate automat, atâta timp cât intenţiile declarate ale site-ului în utilizarea acestor informaţii sunt conforme cu preferinţele utilizatorului.

4.4.2 Cookies

O tehnologie dezbătută în asigurarea intimităţii utilizatorilor sunt cookies [L05]. Acestea sunt o inovaţie adăugată browserelor de Web în 1995. In 1996 utilizarea cookies-urilor a determinat discuţii controversate relativ la implicaţiile lor în intimitatea utilizatorilor.

Cookie-urile sunt o tehnologie care a transformat hit-urile Web fără stare în sesiuni pentru recunoaşterea automată a unui browser particular când se revine la un site după un anumit interval de timp, şi pentru memorarea informaţiilor de profil ale utilizatorului în cadrul browserului.

Cookies-urile sunt un protocol Web şi un mecanism de browser care permit unui server să spună unui navigator Web să memoreze un bloc de informaţie pe hard-diskul utilizatorului, informaţie pe care să o trimită înapoi serverului la vizitări subsecvente. Cookies-urile furnizează suportul pentru sesiuni, recunoaşterea automată a unui utilizator şi memorarea profilului unui utilizator.

Există anumite avantaje dar şi dezavantaje a cookies-urilor. Avantajele ar fi următoarele [PS00]:

Sesiunile. Fără sesiuni este imposibila furnizarea unui serviciu bazat pe Web care să aibă mai mult de un form. Dacă este necesară o succesiune de formuri, acestea se păstreaza pentru determinarea secventei de acces.

Recunoaşterea automată a unui utilizator. Identificarea automată a unui utilizator particular este importantă pentru un serviciu de securitate care necesită autentificarea şi pentru oferirea unui serviciu care oferă vederi personalizate a utilizatorilor.

Profiluri ale clienţilor. O aplicabilitate a cookies-urilor este că acestea asigură un loc de plasare a anumitor informatii precum profilul utilizatorului sau starea aplicaţiei. Un beneficiu al acestora este că serviciul nu suferă în performanţă în accesul la bazele de date ale serverului pentru fiecare hit.

Cookies-urile însă prezintă şi dezavantaje:
Trasare necunoscută. Cookies-urile pot fi folosite în site-urile Web pentru a memora informaţii personale fără acordul persoanei în cauză. Deoarece un cooky trasează un anumit browser, un site Web poate obţine informaţii cu acurateţe la vizite repetate. Dar informaţiile pot fi şi fără acurateţe, în mod particular dacă un calculator este partajat de mai mult de o persoană. Aceasta permite ca informaţiile despre o persoană să fie arătate alteia.

4.4.3 Tranzacţii electronice securizate

In 1995 Visa şi MasterCard şi-au unit forţele pentru a dezvolta protocolul SET (Secure Electronic Transaction), un standard tehnic pentru plata securizată cu carduri pe reţele deschise.

SET este desemnat să mimeze fluenţa informatiilor unui card. In plus faţă de mesajele operaţionale, SET include utilizarea cheilor publice pentru certificate pentru a autentifica clientul si serverul unul fata de altul [RK00].

4.5 Arhitecturi funcţionale ale aplicaţiilor de comerţ electronic

Definiţia 32: Arhitectura unui sistem defineşte componentele sale de bază şi conceptele importante şi descrie relaţia dintre acestea [HBP+99].

Există mai multe căi de abordare a sistemelor de comerţ pe Internet, de la simplu la complex. In parte, arhitectura depinde de natura afacerii, iar arhitectura sistem dezvoltată pentru un B2C poate fi foarte diferită de cea a unui sistem informaţional. Se crede că mai multe idei de design acoperă un domeniu larg de cereri de comerţ şi că similarităţile în sistemele pentru comerţul pe Internet sunt mai mari decât diferenţele.

Arhitectura poate fi reutilizata ori de câte ori este posibil. Mai important este că, cu cât afacerea ajunge la un rafinament mai mare şi îşi expune ţelurile pentru comerţul pe Internet, cu atât va evolua şi sistemul. Evoluţia poate merge dincolo de cerinţele originale pentru sistem, astfel încât flexibilitatea arhitecturii este importantă în posibilitatea creşterii sistemului.

4.5.1 Idei arhitecturale de bază

Arhitecturile pentru comerţul pe Internet pot arăta foarte diferit, dar ele toate adresează anumite rezultate şi furnizează răspunsuri la un set comun de întrebări. Aceste rezultate trebuie înţelese foarte bine fără a conta abordarea care se ia în considerare. In anumite cazuri aceste întrebări comune sunt considerate explicite pe parcursul fazei de design; în alte cazuri întrebările şi răspunsurile la acestea se reflectă în presupuneri relativ la anumite componente arhitecturale [O00].

Inţelegerea rolurilor. Două dintre întrebarile de bază sunt "Cine utilizează sistemul?" şi "Ce se face în sistem?". Pentru anumite programe există câteva tipuri de utilizatori care partajează ţeluri similare.

Decompoziţia funcţiilor. A doua parte importantă în arhitectură este modul în care sistemul este descompus în unităţi funcţionale. Specificarea acestor unităţi funcţionale şi a interfeţelor dintre ele defineşte arhitectura sistemului. Una din diferenţele dintre arhitecturi este adesea modul în care ele grupează funcţiile în unităţi.

Legarea conţinutului la tranzacţii. Primele două consideraţii pentru arhitectura sistemului, rolurile şi decompoziţia, se aplică design-ului sistemelor. A treia parte a arhitecturii sistem a aplicaţiilor de comerţ electronic pe Internet este modul în care conţinutul, precum cataloagele, este legat de procesarea tranzacţiilor. Într-un sistem

bazat pe hârtie, un cumpărător transcrie pe un ordin de plată numărul entităţilor şi cantităţile. Acestea se fac însă electronic într-un sistem de e-commerce.

Există câteva intrebari care necesita raspuns relativ la aplicatia de comert electronic şi anume:

- Cum realizează utilizatorul tranzacţia?

  In majoritatea cazurilor utilizatorul vede un buton pe care scrie "Apăsaţi aici pentru a cumpăra" sau poate adăuga entităţi într-un coş de cumpărături pentru achiziţia lor ulterioară. Tranzacţia are loc fie pe buton fie la cumpărarea finală pentru coşul de cumpărături.

- Cum este verificată informaţia ?

  In funcţie de tehnologia utilizată, poate fi necesar ca sistemul tranzacţional să verifice că informaţia legată de cumpărături (preţ, identificarea entităţilor cumpărate, etc) nu a fost modificată când a fost trimisă pe reţea. Deoarece Web-ul utilizează un protocol fără menţinerea stării, sistemele pe Internet trebuie să-şi verifice singure starea. Cu alte cuvinte serverul trebuie să realizeze controlul accesului.

- Cum este potrivită informaţia ?

  Anumite sisteme de comerţ pe Internet asigură o verificare în timp real pentru a asigura cumpărătorii că produsele achiziţionate se află pe stoc. Dacă un produs este pe stoc, până când este valabil acesta? Dacă un utilizator pune un produs în coşul de cumpărături pentru achiziţionarea acestuia peste un interval de timp, sistemul promite că acesta va fi disponibil şi în momentul realizării cumpărării?

Răspunsurile la aceste întrebari ajută la decizia pentru design-ul sistemului de achiziţii prin Internet. Deoarece răspunsuri diferite pot determina design-uri diferite, este important ca răspunsurile să fie cunoscute înaintea design-ului.

4.5.2 Modele de încredere

In orice sistem distribuit, diferite componente au încredere una în cealaltă pentru una sau mai multe sarcini. Anumite componente pot avea încredere în altele pentru orice tip de tranzacţie în timp ce alte componente nu acceptă nici un tip de acces la distanţă. Specificarea acestor relaţii între componente este denumită *modelul de încredere (trust model)* pentru sistem [YD05]. Orice model are cel puţin un sistem de încredere, iar specificarea lui explicită ajută la înţelegerea relaţiilor dintre componente atunci când este necesară analiza securităţii sistemului.

4.5.3 Arhitecturi sistem

Se prezintă trei arhitecturi posibile [FSN05] şi arhitectura modelului SCAR-ACE si se discută modul în care au fost construite. Cele patru arhitecturi sunt:

1. un server Web cu formuri de comenzi;
2. o variaţie a unui sistem Web cu formuri de comenzi care utilizează protocolul SET;
3. un sistem de tranzacţii distribuite dezvoltat pentru Open Market;
4. arhitectura SCAR-ACE de B2B.

Există bineînţeles şi alte abordări ale comerţului electronic.

Pentru analizarea arhitecturilor s-au considerat patru componente sistem:

- Cumpărătorul (clientul): este un calculator conectat la Internet prin intermediul unui furnizor de servicii Internet sau prin intermediul unei rețele a corporației. Cumpărătorul utilizează acest calculator client pentru a naviga şi a realiza achiziții.
- Vânzătorul (furnizorul): Sistemul sau sistemele de calculatoare conținând cataloagele furnizorului.
- Sistemul de tranzacție: Sistemul de calculatoare care procesează o comandă particulară şi care sunt responsabile pentru plata, păstrarea înregistrărilor şi alte aspecte ale afacerii relativ la tranzacții.
- Gateway-ul de plată: Sistemul de calculatoare care rutează instrucțiunile de plată către rețele financiare cum ar fi cele de autorizare a cărților de credit.

Arhitecturi diferite utilizează aceste patru componente în moduri diferite. In anumite sisteme câteva dintre aceste componente pot fi combinate pe un singur sistem de calculatoare sau ca şi sisteme separate.

Odată ce designerii sistemului de comerț au selectat o diviziune de funcții există încă o sumedenie de decizii de luat la nivel de funcționalitate. De exemplu funcțiile agregate care permit asamblarea entităților individuale într-o comandă completă.

Arhitectura 1 : Server de Web cu formuri de comenzi

Un server Web cu pagini de catalog şi un form de comandă este construcția cea mai simplă a unui sistem de comerț pe Internet. Acest server este adeseori denumit server furnizor.

In acest exemplu, un singur server Web furnizează atât catalogul cât şi comenzile. Cu alte cuvinte serverul furnizor şi serverul de tranzacții sunt combinate într-un singur sistem şi nu există nici un gateway de plată. Catalogul poate consta dintr-un set de pagini Web care descriu entitățile de vândut, încapsulând imagini, specificații, animații, clipuri video sau audio, şi altele. Paginile Web pot fi create ca şi nişte pagini statice utilizând un editor HTML, sau pot fi create dinamic din baza de date care conține produsele şi informațiile descriptive ale acestora. Pentru fiecare produs există un buton care permite cumpărătorului achizitia acelui produs sau adaugarea sa la coşului de cumpărături. Când cumpărătorul doreşte să achizitioneze produsele, acționează asupra butonului de verificare şi procesul plății demarează ca şi parte a tranzacției.

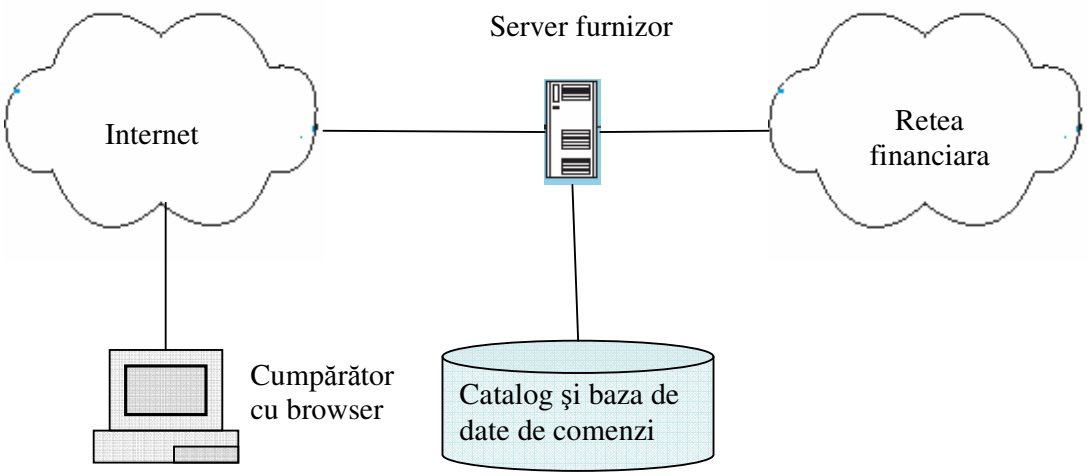

Figura 26. Arhitectura 1: Vedere fizică

Plata prin utilizarea cardurilor este cea mai uzuală în zilele noastre pentru tranzacţii client. Un form simplu de comenzi poate conţine entităţile cumpărate, un set de câmpuri pentru editarea informaţiilor despre card şi adresa de livrat bunurile fizice.

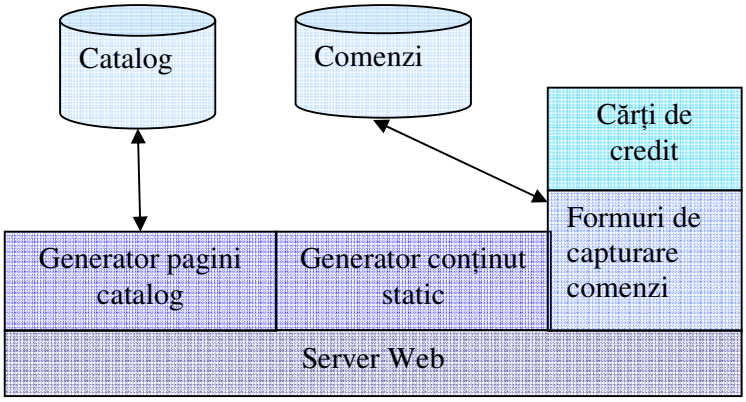

Figura 27. Arhitectura 1: Vedere logică

Este posibil de asemenea ca serviciul Web să includă un mecanism diferit de cumpărare. In versiunea cea mai simplă a acestui model, clientul Web nu are capabilităţi speciale pentru comerţ, aşadar aplicaţia de comerţ nu necesită mecanisme software adiţionale de plată. Cărţile de credit, ordinele de plată sau alte tipuri de plăţi pot fi utilizate în astfel de sisteme ţinându-se cont de capabilităţile de securitate de astăzi ale Web-ului.

Arhitectura aceasta poate fi potrivită şi suficientă pentru anumite tipuri de aplicaţii de comerţ. Caracteristica sa de bază este simplitatea. Pe de altă parte este dificil de expandat pe măsură ce afacerea creşte sau încorporează noi tehnologii şi componente.

Arhitectura 2: Arhitectura SET

SET este un standard pentru tranzacţiile de plată cu cărţi de credit, care se efectuează pe Internet. Într-un sistem cu SET, poarta de acces pentru plăţi este adăugată distinct serverului de tranzacţii.

Diferenţa între un server Web cu formuri de comenzi şi un sistem bazat pe SET constă în modul în care formurile de comenzi sunt gestionate şi în modul în care este realizată comunicaţia pentru efectuarea plăţii [BHM06].

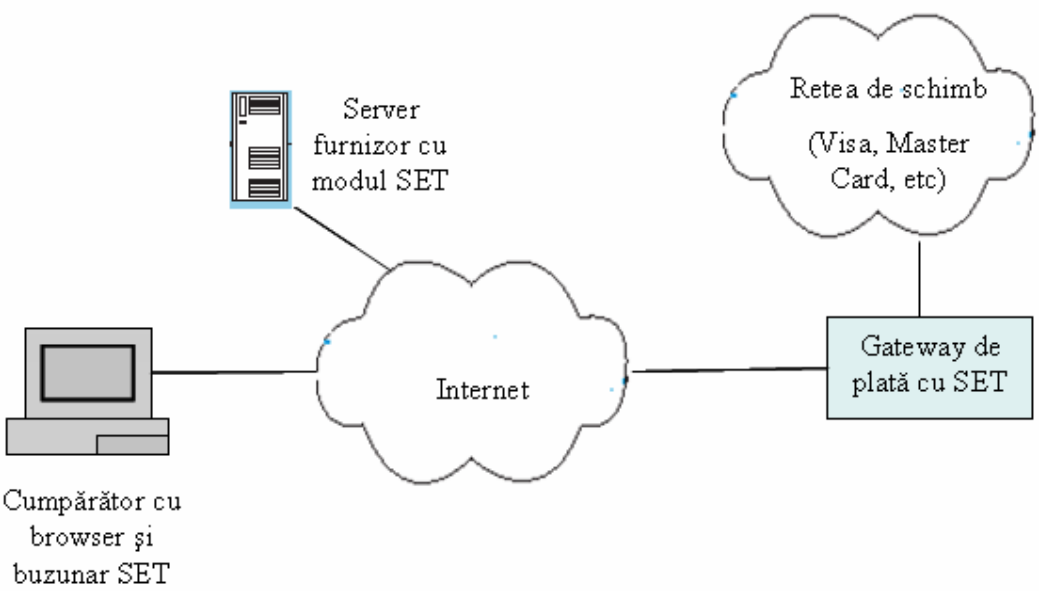

Figura 28. Arhitectura SET

In forma sa cea mai simplă, arhitectura SET comunică de la serverul furnizor până la poarta de acces pentru efectuarea plăţilor. Serverul furnizor, nu se conectează direct la o reţea autorizată în plata cu cărţi de credit ci încorporează un modul SET. Când modulul SET este apelat pentru a se realiza o plată au loc următoarele acţiuni [TT02], [BPM02]:

- Modulul furnizor trimite un mesaj către buzunarul SET care este pe calculatorul cumpărătorului, conţinând o descriere a comenzii şi a preţului total.
- Cumpărătorul utilizează buzunarul SET pentru a selecta un mod de plată şi a aproba plata.
- Buzunarul SET comunică, via calculatorul furnizor, cu poarta de acces SET pentru efectuarea plăţii la banca cerută de furnizor.
- Poarta de acces se conectează la o reţea financiară tradiţională pentru a autoriza tranzacţia.
- Calculatorul furnizor memorează aprobarea şi trimite o confirmare de primire cumpărătorului.

Arhitectura 3: Arhitectura Open Market Commerce

Ideea acestei arhitecturi este separarea gestionării conținutului de gestionarea tranzacțiilor printr-o tehnologie denumită *SecureLink*. Această idee permite serverelor multiple de cataloage să partajeze capacitatea unui singur motor de tranzacții şi permite părților orientate pe conținut ale sistemului, să se scaleze în mod independent fata de părțile orientate pe tranzacții ale sistemului. Figura 29 arată arhitectura fizică [SL98], [DK01], [C03]. In această arhitectură, serverul de tranzacții este separat de serverul furnizor şi poate să existe sau poate să nu existe o poartă separată de plată depinzând de metodele de plată suportate.

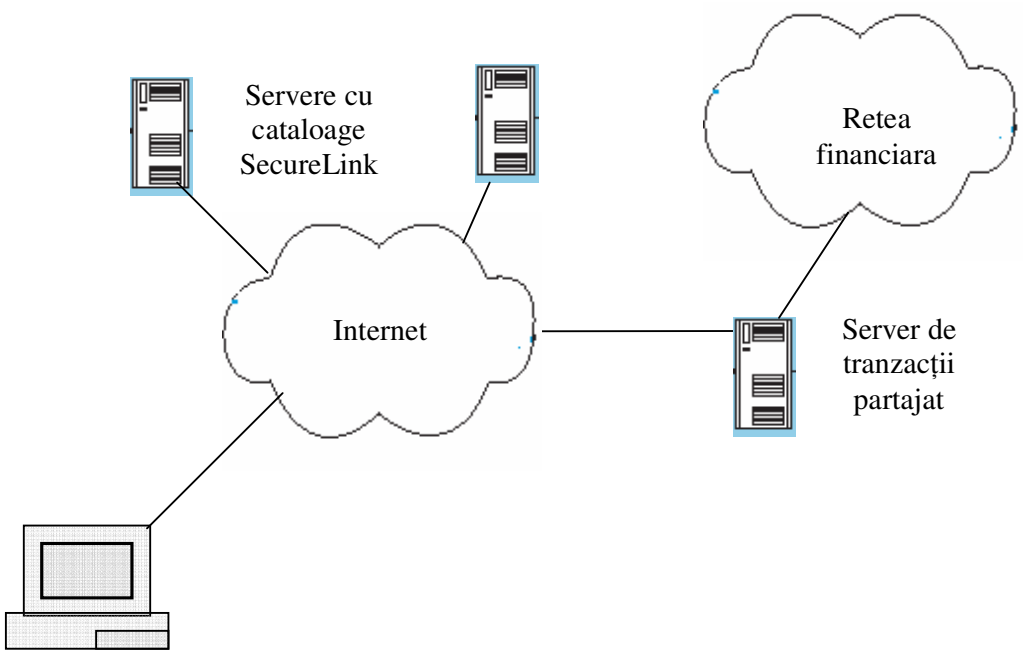

Figura 29. Arhitectura fizică cu SecureLink

4.6 Implementarea modelului SCAR-ACE

4.6.1 Prezentarea arhitecturii

Arhitectura se adresează aplicațiilor de B2B [OG05] în care sunt implicate diverse grupe de actori. Ideea de bază a acestei arhitecturi este separarea funcționalității de comerț între partea orientată pe cumpărare şi cea orientată spre vânzare astfel încât fiecare organizație să isi gestioneze activitățile specifice (figura 31).

Acest design este bazat pe modelul afacerii aratat mai jos:

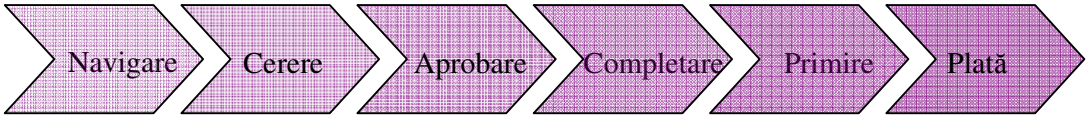

Figura 30. Procesul de achiziție

In acest model logica de separare a activităţilor este astfel: procesele de navigare şi cerere sunt pe partea de cumpărător iar catalogul, gestionarea comenzilor, completarea şi plata sunt pe partea de furnizor. Această arhitectură rezultă în arhitectura logică ilustrată în figura 32. Idea este divizarea serverului de tranzacţii în parte de cumpărător şi parte de vânzător.

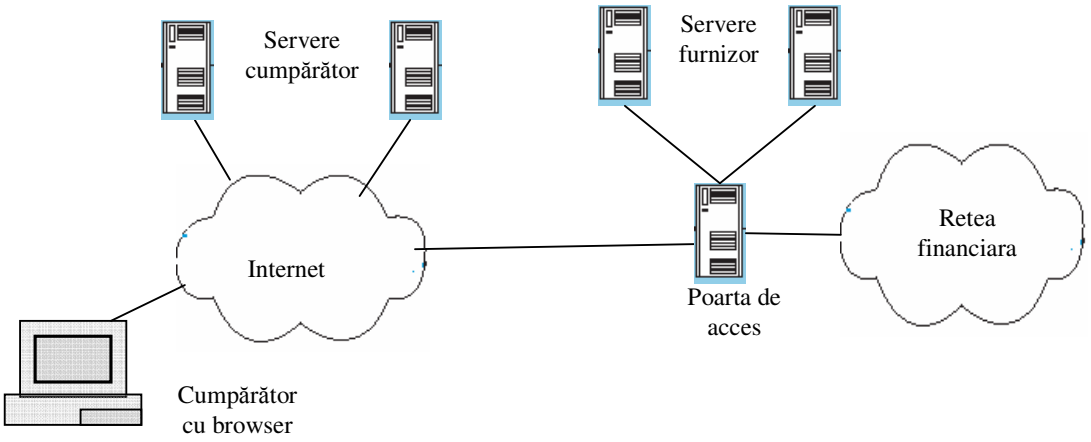

Figura 31. Arhitectura fizică

Pentru ca această arhitectură să fie funcţională sunt necesare două elemente de interoperabilitate între partea cumpărătoare şi partea furnizoare: autentificarea clientului şi gestionarea comenzilor.

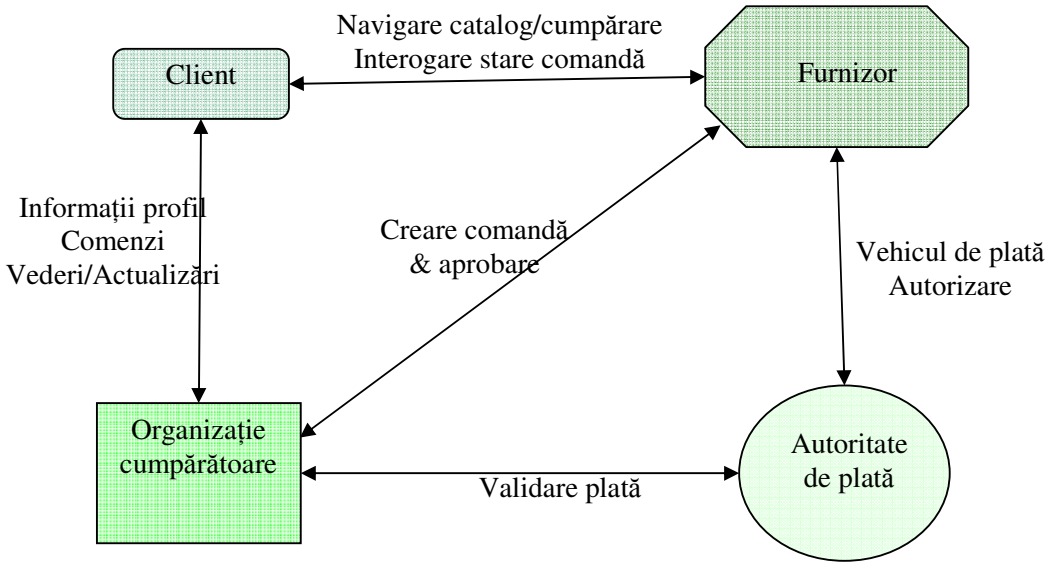

Figura 32. Arhitectura logică

- *Autentificarea clientului* Acest model utilizează autentificarea prin nume utilizator şi parolă.

- *Gestionarea comenzilor* - în cadrul acestui model, clientul crează o comandă prin interacţiunea cu catalogul furnizorului. Această comandă este trimisă unui format standard denumit *Cerere comandă* de pe serverul furnizorului pe serverul cumpărătorului. Odată ajunsă acolo, toate aprobările necesare sunt procesate. După finalizarea comenzii, aceasta este returnată furnizorului ca şi comanda ferma.

4.6.2 Fluenţa tranziţiilor

Pentru a realiza un proces de achiziţie sistemul realizează următorii paşi:

1. Clientul foloseşte un navigator Web şi accesează sistemul. Nivelul de autorizare iniţial este minim şi anume rolul său este cel de *vizitator*;
2. Serverul cu catalogul organizaţiei furnizoare autentifică clientul şi îi permite acestuia să navigheze şi să selecteze articole. După autentificare clientul trece în rolul de *cumpărător;*
3. *Cumpărătorul* mapează comanda într-o cerere, o încapsulează într-un obiect şi transmite cererea de comandă organizaţiei furnizoare utilizând Internetul;
4. Clientul specifică orice adnotări necesare comenzii şi furnizorul realizează aprobarea internă;
5. Organizaţia furnizoare începe distribuirea comenzii.

4.6.3 Funcţionalităţi implementate

Vizitatorul este utilizatorul cu nivelul de securitate cel mai scăzut şi orice încercare de acces a unui utilizator îl direcţionează pe acesta spre pagina de "Home" aceasta fiind prima funcţionalitate implementată. Din această pagină utilizatorul poate vizualiza catalogul de produse, poate intra pe pagina de comparare a produselor sau se poate autentifica în sistem prin login sau înregistrare.

In cadrul catalogului de produse se pot vizualiza produsele în funcţie de anumite criterii: produse de acelaşi tip, produse ale unui furnizor sau cele care au o anumită caracteristică selectată.

Mecanismul implementat în pagina de comparare a produselor este cel descris în figura 17.

Aceste funcţionalităţi sunt moştenite de rolurile de cumpărător şi de vânzător.

Un utilizator are rolul de cumpărătorul după înregistrare, şi îşi poate modifica informaţiile de identificare de pe pagina cu profilul său. Această pagină conţine date de identificare gen nume şi adresa precum şi informaţiir pentru care ar dori notificări.

Totodată cumpărătorul poate selecta produse în coş şi poate emite cereri de livrare. Chiar dacă au fost selectate unul sau mai multe produse pentru achiziţionare, acestea pot rămâne în coş fără a se materializa într-o cerere fermă. Informaţiile acestea precum coşul de cumpărături şi cererile ferme sunt păstrate persistent în baza de date şi pot fi accesate în orice sesiune după autentificare.

Cumpărătorul mai moşteneşte şi toate funcţionalităţile oferite vizitatorului.

4.6.4 Platforma de comerţ electronic

Această secţiune descrie platforma de gestionare a tranziţiilor şi legarea automată a obiectelor în arhitecturile de comerţ electronic. Această platformă a fost programată utilizând tehnologia Microsoft COM/DCOM. Sistem deserveşte mai mulţi

clienți şi mai multe tipuri de actori. Orice tranziție între două stări diferite, bine definite, trebuie gestionată de către aplicația server în momentul rulării ca rezultat a informațiilor parametrizate trimise de către client. In ciuda faptului că selectarea dinamică induce costuri la rulare, flexibilitatea rezultată permite prototipizarea rapidă a aplicației şi reduce timpul de dezvoltare a acesteia. Nucleul arhitecturii este alcătuit din maşini de stare care încapsulează mecanismul de control al accesului bazat pe roluri în care sunt controlate tranzițiile între stări.

Un pas important în designul unui sistem de e-commerce, bazat pe un mecanism de control al accesului este înțelegerea rolurilor client. Aceasta ajută la concentrarea atenției asupra faptului ca fiecare persoană să utilizeze sistemul în îndeplinirea scopurilor propuse, fie că aceasta este realizarea unei cumpărături sau obținerea unor rapoarte.

Pe de altă parte, orice decizie în designul atât a programării bazate pe obiecte cât şi a mediului trebuie să se confrunte cu alegerea între static şi dinamic. Proprietățile statice se referă la alegerile realizate în momentul dezvoltării, înaintea execuției programului în timp ce aspectele dinamice depind de opțiunile şi alegerile care sunt validate doar în momentul rulării aplicației.

Implementarea sistemului SCAR-ACE implică cerințe dinamice legate de controlul accesului, fluența controlului aplicației, astfel încât orice tranziție între două stări este determinată la rulare şi este executată conform parametrilor de transfer între două stări. Dacă parametrii sunt greşiți şi nu potrivesc unei stări consecvente atunci este emisa o excepție.

4.6.5 Structura aplicației. Nivelele prezentare şi de control al fluenței

Această secțiune descrie nivelul de prezentare (Presentation Layer – PL) şi cel de control al fluenței aplicației (Workflow Layer - WFL), tipurile de clienți, descrierea interfeței între aplicație şi navigatoare şi a interfeței dintre WFL şi nivelul de logică a aplicației (Business Logic Layer - BLL). Figura 33 ilustrează nivelele aplicației.

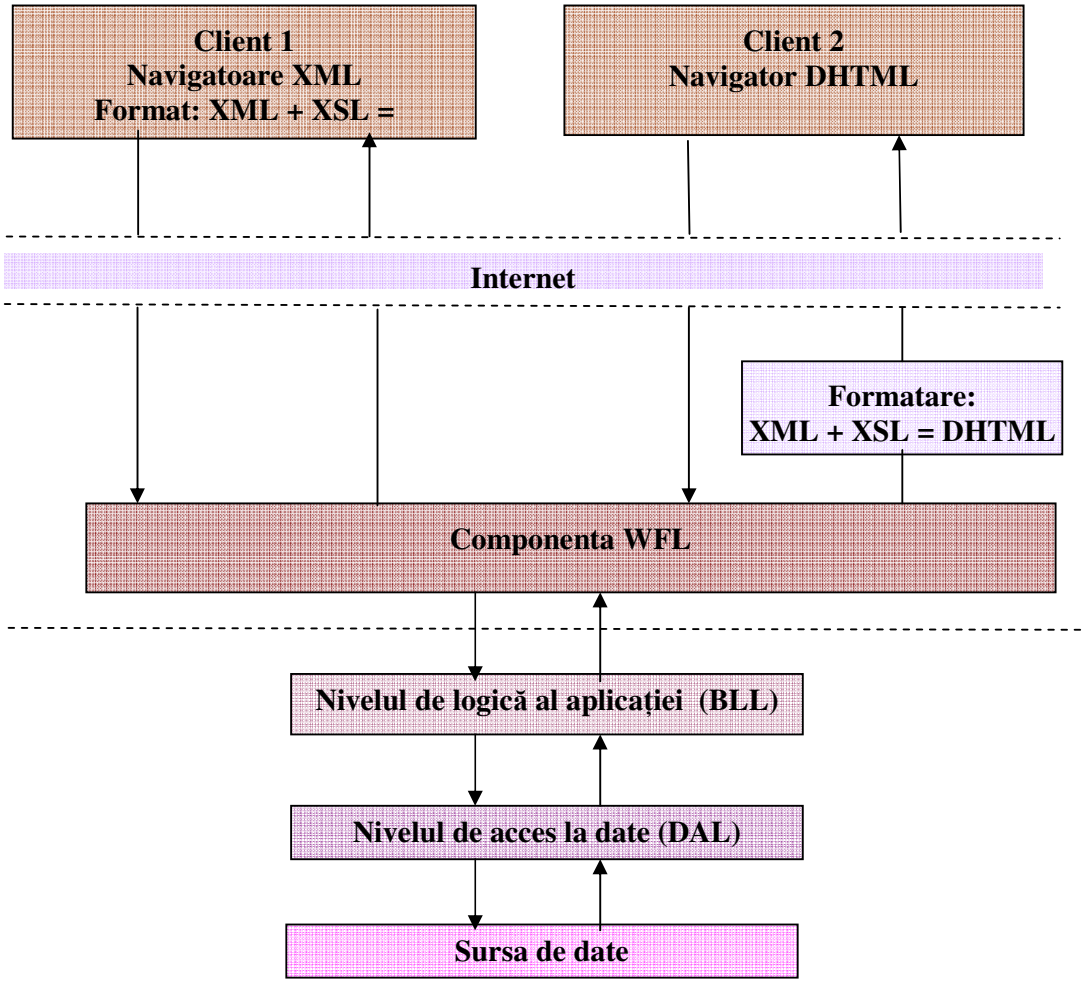

Figura 33. Structura pe nivele a aplicației

Nivelul prezentare - PL

Clienții acceptați de aplicație sunt cei XML şi DHTML. Tehnologiile utilizate pe partea de client sunt DHTML, Java Script şi CSS.

Paginile HTML conțin câmpuri ascunse cu date pentru nivelul de control al fluentei aplicatiei. Câmpurile ascunse sunt declarate în cadrul formurilor, care au aceeaşi structură pe fiecare pagină client:

```
<form name="actionForm">
        <input name="EK_CS" type="hidden">
        <input name="Event" type="hidden" value="" />
        <input name="Param1" type="hidden" value="END" />
                ...
        <input name="Param8" type="hidden" value="END" />
</form>
```

Semnificația variabilelor este după cum urmează:

- `EK_CS` – conține concatenarea următoarelor date: rolul utilizatorului şi identificatorul utilizatorului. Rolul utilizatorului este un caracter ce poate lua diferite valori (în acest exemplu v – rol de vizitator) şi desemnează rolul utilizatorului în cadrul aplicației. Identificatorul utilizatorului este o valoare unică care identifică clientul în cadrul unei sesiuni. Această cheie este generată dinamic de către server;
- `Event` – reprezintă evenimentul emis drept rezultat al acțiunii utilizator la selectarea unui element activ al interfeței grafice. Această valoare este o constantă actualizată pe partea de client;
- `Param1, ..., Param8` **–** reprezintă parametrii dependenți de selecția utilizator în cadrul paginii curente. Valorile acestora sunt actualizate pe partea de client.

Nivelul de control al fluenței aplicației (WFL)

Nivelul de control al fluenței aplicației are rolul de a verifica consistenta si corectitudine tranzitiilor din sistem. El implementeaza componentele TT din M_M respectiv atat partea constanta cat si cea variabila a tranzitiilor modelului.

Nivelul de control al fluentei aplicatiei constă dintr-un fişier script şi o componentă care rulează ambele pe server si furnizează interfața dintre nivelul prezentare (clientul Web) şi nivelul de logică al aplicației.

Interfața cu nivelul prezentare este gestionată de componenta WFL iar secvența evenimentelor este următoarea:

- nivelul prezentare declanşează cererea client;
- se crează o instanță a componentei de control a fluentei careia i se trimite cererea client;
- se trimit parametrii client nivelului de logică al aplicației;
- se obține răspunsul de la nivelul de logică al aplicației şi se trimite înapoi clientului.

Comunicarea între nivelul de control al fluenței aplicației şi nivelul de prezentare este bidirecțională, după cum se arată în figura următoare:

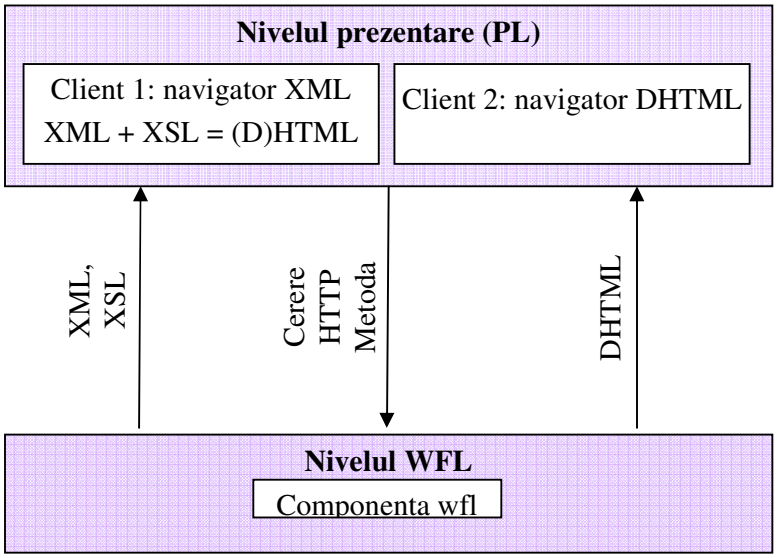

Figura 34. Comunicarea utilizator – componenta de control a fluentei

Comunicarea Client – componenta de control a fluentei

Cererea trimisă de către nivelul prezentare conține parametrii care sunt incluşi în câmpuri ascunse ale paginii HTML. Parametrii sunt trimişi paginii utilizând metoda HTTP POST. Mai apoi aceşti parametri sunt convertiți în structura de mesaj şi sunt trimişi prin intermediul nivelului de control al fluentei catre componenta corespunzătoare din nivelul de logica al aplicatiei.

Exemplificare:

Presupunem că utilizatorul selectează o entitate de pe pagina de cumpărare şi acționează `Print Preview`. Parametrii trimişi către nivelul de control al fluentei aplicatiei sunt:

```
EK_CS = c19efafa8-3634-11d4-a99d-
00d0b7151d330030
Stare = 2 (ST_CUMP_VIEW)
Event = 31 (EV_CUMP_PRINT_PREVIEW)
Param2 = BMS71
Param3 = END
```

unde:

- `c din EK_CS` - rolul utilizatorului care este de rolul de cumparato;r
- `19efafa8-3634-11d4-a99d-00d0b7151d330030` – identificator utilizator;
- `2` - identificatorul stării `ST_CUMP_VIEW` din care se realizează tranziția (unic determinat în baza de date);
- `31` – identificatorul evenimentului `EV_CUMP_PRINT_PREVIEW;` evenimentul semnifică că o entitate a fost aleasă de către utilizator pentru `Print Preview;`
- `BMS71` - codul produsului memorat în baza de date pentru care se doreşte `Print Preview.`

Comunicarea dintre nivelul de control a fluenței aplicației şi nivelul de prezentare

Răspunsul de la nivelul de control al fluenței aplicației la client poate fi trimis în două formate. Dacă clientul suportă XML atunci răspunsul este trimis ca şi un document XML aşa cum este el primit de la nivelul de logica al aplicatiei. Dacă clientul nu suportă XML documentul este convertit în DHTML pe partea de server utilizând documentul XML şi documentul XSL corespunzător.

Comunicarea dintre nivelul de control a fluenței aplicației şi nivelul de logică al aplicației

Comunicația între nivelul de control a fluentei aplicatiei şi nivelul de logică al aplicatiei este de asemenea bidirecțională. Parametrii cererii client sunt convertiți într-o structură de mesaj şi trimişi către nivelul de logică. Se trimit următorii parametri:

- Tipul clientului care a făcut cererea – client XML sau client non-XML;
- rolul utilizatorului;
- identificatorul utilizatorului;
- starea curentă;
- evenimentul;
- un vector cu trei parametri de intrare;
- doi parametri de ieşire, unul pentru datele XML returnate de BLL şi altul pentru calea la fişierul XSL utilizat de XML în conversia către HTML.

Comunicarea dintre nivelul de logică şi nivelul de fluență al aplicației

Răspunsul de la nivelul de logică este primit de către nivelul de fluență într-un document XML, documentul fiind încapsulat într-un şir care conține şi calea către fişierul XSL corespunzător. Fişierul XSL va fi utilizat în convertirea datelor XML în HTML atât pe partea de client cât şi cea de server.

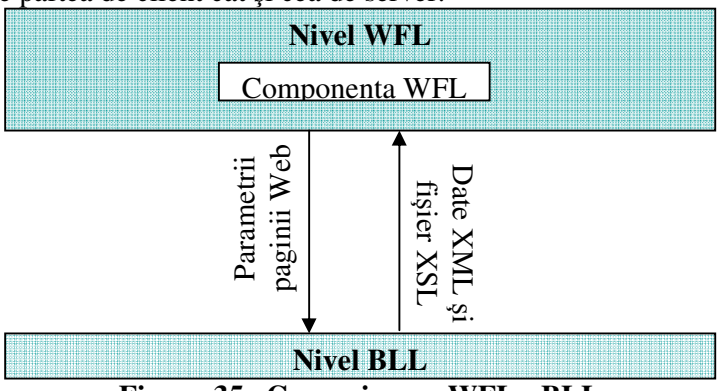

Figura 35. Comunicarea WFL - BLL

4.6.6 Structura datelor

Structura utilizată de către aplicație este divizată în două secțiuni. Prima secțiune este comună fiecărei pagini utilizator iar cea de-a doua secțiune este specifică fiecărei pagini. Structura datelor este următoarea:

```
<data>
  Structura comună...

  Structura specifică...
</data>
```

Structura comună

Structura comuna descrie în format XML mesajul utilizat în comunicarea client-server. Structura este următoarea:

```
<form>
   <EK_CS>va78dc97b-3576-11d4-8f71-00a0d21a4e6e0001</EK_CS>
   <State></State>
   <Event></Event>
   <Parameter>
     <Name>Param1</Name>
     <Value>END</Value>
   </Parameter>
   ...
 </form>
```

Aceste date sunt trimise serverului de catre client. Valoarea campului `EK_CS` rămâne neschimbată la acțiunea utilizatorului pe pagina HTML si este o valoare primita de catre client de la server. Campul `<State>` este tot o valoare care este primita de la server si trimisa inapoi fara a fi modificata reprezentand starea curenta. Serverul va interpreta valorile `<State>` si **<Event>**, si daca combinatia primita este una valida, se va trece la starea urmatoare dupa realizarea actiunii specifice.

Structura specifică

Fiecare pagină conține informație specifică care este reprezentată şi structurată în format XML. De exemplu, informația despre o entitate a catalogului este structurată după cum urmează:

```
<entity>

  <F_P_MAKE>tip1</F_P_MAKE>

  <F_P_SERIE>serie1</F_P_SERIE>
```

```
    <F_P_COD>A41</F_P_COD>

    <F_P_MODEL>A4 1.8 T/2000</F_P_MODEL>

    <F_P_PICTURE>/images/cars/Audi/A4/A41.jpg</F_P_PICTURE>

    <F_P_RP>24760.00</F_P_RP>

    <F_P_LINK>http://www.tip1.com</F_P_LINK>

    <F_P_LOGO>/IMAGES/CARS/Audi/LOGO.JPG</F_P_LOGO>

  </entity>
```

In exemplul prezentat structura de date specifica contine următoarele informatii relativ la modelul unei masini:

- `F_P_MAKE` – constructorul modelului;
- `F_P_SERIE` – seria modelului;
- `F_P_COD` – codul modelului;
- `F_P_MODEL` – tipul modelului;
- `F_P_PICTURE` – calea la imaginea modelului;
- `F_P_RP` – prețul;
- `F_P_LINK` – adresa Web a constructorului.

4.6.7 Nivelul de logică şi nivelul de date

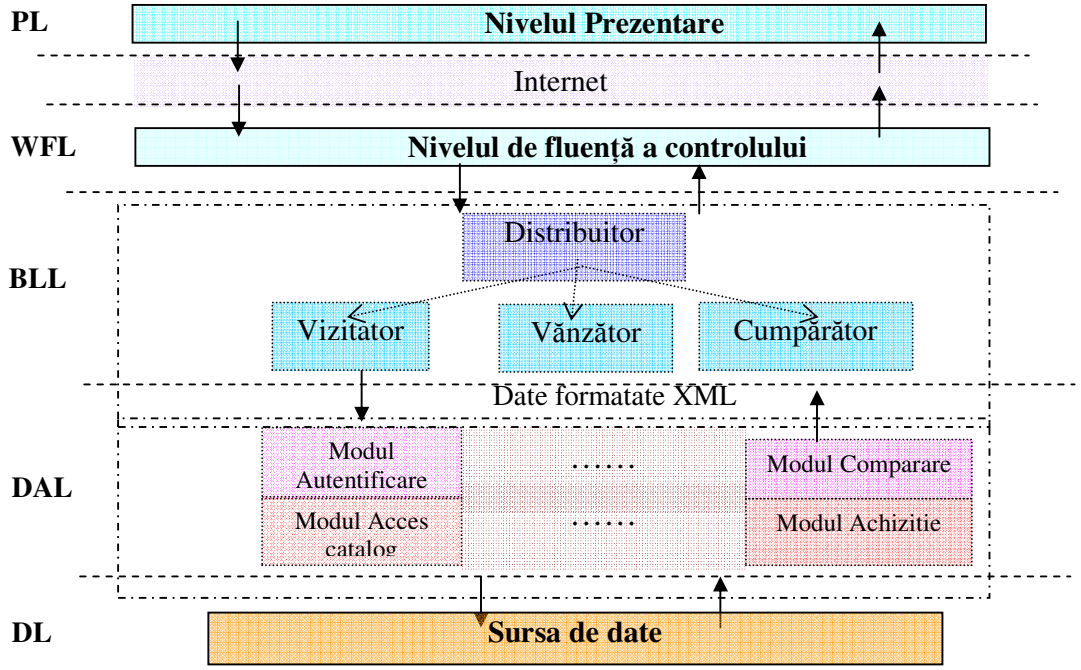

Figura 36. Structura aplicației. Nivelele de logică şi de acces la date

Componentele celor două nivele de prezentat sunt următoarele:

- BLL – nivelul de logică a aplicației;
- DAL – nivelul de acces la date ;
- DL – nivelul de date (server de baze de date).

BLL şi DAL sunt formate din componente dezvoltate ca şi servere in-process care pot rula individual pe diferite maşini.

In cadrul aplicației, un utilizator poate fi în una dintre următoarele trei maşini de stare: vizitator, cumpărător sau vănzător. Componenta *Distribuitor* este cea responsabila cu selectarea masinii de stare corespunzatoare fiecarui utilizator în funcție de rolul acestuia. Tranzițiile în cadrul fiecarei maşini de stare sunt gestionate de una dintre cele trei componente corespondente ale BLL.

DAL conține componente cu facilitate de acces la DL pentru diferite tipuri de acțiuni.

Nivelul de logică a aplicației (BLL)

Memorarea stărilor utilizator şi trasarea contextului

Credențialele specifice fiecărui apel (identificator rol, identificator utilizator, starea, trasarea contextului) sunt păstrate în DL. In anumite cazuri este posibilă accesarea de către BLL a bazei de date evitându-se DAL. Aceasta este necesar la fiecare mesaj primit de la PL pentru identificarea utilizatorului. O soluție alternativă ar fi păstrarea informațiilor utilizator într-o memorie cash şi accesarea acesteia ori de câte ori este necesară identificării unui utilizator sau al contextului acestuia.

Pentru un eveniment de intrare se crează un obiect distribuitor care verifică dacă credențialul primit de la utilizator este unul valid prin căutarea acestuia în DL.

Aplicația are la bază maşini de stare. O stare denotă un context utilizator adică un set de date caracteristice unui anumit moment. Trecerea dintr-o stare în alta sau dintr-o maşină de stare în alta se efectuează pe baza unui eveniment generat în PL. Acest eveniment determină schimbarea contextului de date la trecerea într-o nouă stare. Trecerea de la o maşină de stare la alta se realizează be baza autentificării.

Un utilizator este, la un moment dat, într-o stare clar determinată şi poate efectua un număr determinat de tranziții.

Exemplificare:

Fie două roluri alese precum cel de vizitator şi cel de cumpărător. Rolul de vizitator este rolul inițial a unui utilizator înainte de a se autentifica în cadrul sistemului. Dacă se încearcă forțarea unei stări autorizate se va intra în mod automat în starea de login. Dacă procesul de autentificare se termină cu succes, utilizatorul va trece într-o altă maşină de stare şi îşi va modifica rolul din vizitator în cumpărător (de exemplu). Se disting astfel trei maşini de stare pentru rolurile detaliate mai sus şi anume: maşina de stare vizitator (M_V), maşina de stare login (M_L) şi maşina de stare cumpărător (M_C). In funcție de starea curentă, evenimentul apărut şi alți parametri de intrare, are loc tranziția dintr-o stare în alta, dintr-o maşină de stare în alta.

Maşinile de stare sunt memorate în DL dar, în momentul instalării, acestea sunt încărcate în BLL. Aceasta nu afectează scalabilitatea sistemului: fiecare pachet nou BLL va încărca cele trei maşini de stare în memorie.

Dacă o persoană încearcă să realizeze operații ilegale dintr-un navigator apelând WFL dintr-o pagină, cu parametrii de apel alterați, se realizează verificarea credențialului în BLL:

- rolul trebuie să fie unu valid;
- identificatorul utilizatorului trebuie să fie unul valid;
- starea trebuie să fie validă;
- evenimentul apărut trebuie să declanşeze o acțiune validă şi o tranziție într-o stare validă.

Este implementată şi acționarea butonului **Back** şi modificarea stării curente (trasarea contextului) a unui utilizator în acest caz. Astfel, serverul memorează valoarea actuală a stării clientului (SAS) şi, la sosirea cererii, credențialul conține altă valoare (starea actualizată a clientului - UAS). O altă problemă este cazul în care la solicitarea **Back** utilizatorul va trimite cereri dintr-o pagină aparținând unei alte maşini de stare. In acest moment Dispatcher-ul trebuie să recunoască faptul că s-a acționat un buton **Back** şi nu este un hacker care încearcă să trimită parametri diverşi pentru a intra în sistem. Problema este tratată în funcție de utilizator şi tipul maşinii de stare.

Tabelul de mai jos ilustrează faptul că, în cazul în care SAS are tipul vizitator sau login, acționarea butonului **Back** nu are implicații deoarece datele care se doresc a fi accesate sunt date publice, care nu necesită autentificare.

Când SAS are tipul cumpărător şi UAS are tipul login sau vizitator, utilizatorului i se cere să execute logoff dacă doreşte să acceseze pagini neautorizate. In cazul invers, când UAS are tipul cumpărător şi SAS are tipul login sau vizitator, utilizatorului i se cere să efectueze login dacă doreşte să acceseze pagini (implicit date) autorizate.

Tabelul 4. Gestionarea stărilor la acţionarea butonului Back

UAS \ SAS	VIZITATOR	LOGIN	CUMPĂRĂTOR
VIZITATOR	-	-	Please logoff!
LOGIN	-	-	-
CUMPĂRĂTOR	Please login!	Please login!	Verificări suplimentare

După cum este ilustrat în tabelul 4, în cazul în care UAS este egal cu SAS şi egale cu cumpărător, sunt necesare verificări suplimentare. In baza de date este păstrată o listă conţinând toate stările vizitate de către un cumpărător. Dacă cererea soseşte dintr-o stare memorată înseamnă că utilizatorul a ajuns în acel punct prin acţionarea butonului **Back** şi aceasta este o acţiune autorizată.

Notă: Este important că toate verificările sunt realizate după decriptarea credenţialului care include un UUID (identificator utilizator) care are şanse mari să fie unic mondial. Aşadar, dacă un hacker încearcă să trimită cereri către aplicaţie pentru a accesa stări autorizate, acesta necesită atât o cheie validă criptată cât şi parametri de intrare corecţi.

Componenta de distribuţie

Pentru fiecare intrare utilizator este apelat un modul de distribuţie (Distribuitor). Acesta realizează următoarele acţiuni:

- verificarea credenţialului. Dacă nu este valid se generează mesaj de eroare;
- identificarea rolului utilizator din credenţial;
- crearea unui obiect utilizator corespunzător credenţialului;
- trimiterea evenimentului către obiectul utilizator şi aşteptarea răspunsului (în mod normal un fişier XML);
- trimiterea răspunsului către client;
- gestionarea erorilor.

Componente utilizator

BLL este accesat de către componente WFL la sosirea unei cereri. In funcţie de parametrii de intrare şi de credenţial, Distribuitorul crează o componentă specifică rolului utilizator, şi evenimentele de intrare sunt trimise acestei componente create.

Distribuitorul utilizează polimorfismul la nivelul interfeţelor: apelează metoda *Event* asupra unui obiect utilizator creat fără să ştie tipul obiectului. Obiectul utilizator primeşte starea curentă, evenimentul şi parametrii de intrare, obţine starea următoare şi acţiunea de realizat din maşina de stare. De asemenea, maşina de stare furnizează id-ul interfeţei şi al clasei pentru obiectul DAL de instanţiat (dacă este necesar).

In implementare, se creeaza un obiect DAL care primeste parametrii de intrare curenţi. Asupra acestui obiect se va apela metoda specifica acţiunii de efectuat, obţinându-se în acest mod datele rezultat. Obiectul creat va accesa baza de date, va apela procedura/procedurile stocate şi va transforma rezultatul într-unul în format XML.

In cazul în care se execută logoff aceste acţiuni vor fi rezolvate în cadrul componentelor utilizator fără accesarea bazei de date.

Componente de verificare a constrângerilor

In cazul în care un utilizator are un timp de inactivitate acesta va fi de-logat automat. Pentru aceasta există un utilitar administrativ care are un timer ce va apela automat componenta de verificare. Aceasta componentă verifică expirarea timpului limită de la ultima acţiune a unui utilizator. Dacă rezultatul este **true** (timpul a expirat), informaţia utilizator este menţinută în următoarea manieră: dacă maşina de stare este cea de vizitator, informaţia aferentă acestuia va fi ştearsă din baza de date; dacă utilizatorul este un cumpărător informaţia va fi menţinută în baza de date pentru o perioadă de 24 de ore. Dacă utilizatorul nu revine în acest interval şi nu a emis o comandă informaţia acestuia va fi ştearsă; dacă cumpărătorul e emis o comandă aceasta va fi reţinută în baza de date un timp indefinit.

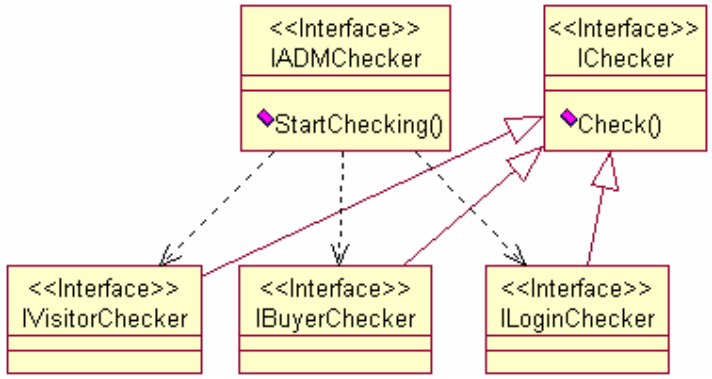

Figura 37. Modelul interfeţelor pentru obiectul de verificare

Există trei tipuri de componente verificatoare pentru fiecare rol. Practic, fiecare tip de obiect verificator şi rol utilizator pot fi instalate pe calculatoare diferite, realizându-se în acest mod scalabilitatea impusă. Modelul interfeţelor pentru obiectele verificatoare este prezentat în figura 37.

Gestionarea erorilor

Pentru gestionarea erorilor fiecare componentă încearcă să-şi rezolve excepţiile şi să-şi forwardeze răspunsul de eroare nivelului superior. In acest mod se regăseşte descrierea exactă a erorii pentru utilizator şi această eroare va fi scrisă şi în fişierul de log.

Dacă intervine o eroare în nivelul DAL se returnează un cod de eroare, iar în funcţie de acest cod se lansează o acţiune de gestionare a erorii, acţiune obţinută în funcţie de maşina de stare şi de componenta DAL. Acest mecanism poate fi reiterat până la producerea unui rezultat valid de gestionare a erorii.

In cadrul BLL a fost introdusă de asemenea o componentă de gestionare a erorilor. Aceasta este apelată de catre Distribuitor în cazul în care se declanşează o eroare. In funcţie de valoarea codului de eroare se va afişa utilizatorului un mesaj de eroare.

4.6.8 Nivelul de acces la date (DAL)

Evenimentele declanşatoare de stări conţinute în cele trei maşini de stare au fost grupate în funcţie de starea din care pot fi declanşate. De exemplu există grupuri de evenimente pentru Actualizare_Ordin, Contul_Meu, Comparare, etc. Gestionarea unui grup de evenimente este realizată de componente separate DAL de exemplu pentru lista de mai sus: DAMUO (Data Access Module Update Order), DAMMA (Data Access Module My Accounts), DAMCF (Data Access Module Comparison Folder).

Aceste componente DAL au ca şi date de intrare codul acţiunii de realizat şi un anumit număr de parametri. In funcţie de codul acţiunii se apelează o procedură iar datele rezultat sunt formatate într-un fişier XML.

In cazul unei excepţii se returnează codul erorii către componenta utilizator.

După cum se poate observa şi în figura 38, interfeţele DAL implementează o interfaţă de bază, pentru crearea unei componente utilizându-se polimorfismul la nivel de interfaţă prin apelarea funcţiei *Action*. Identificatorul clasei şi al interfeţei obiectului DAL se obţin din maşina de stare utilizându-se starea curentă şi evenimentul declanşator.

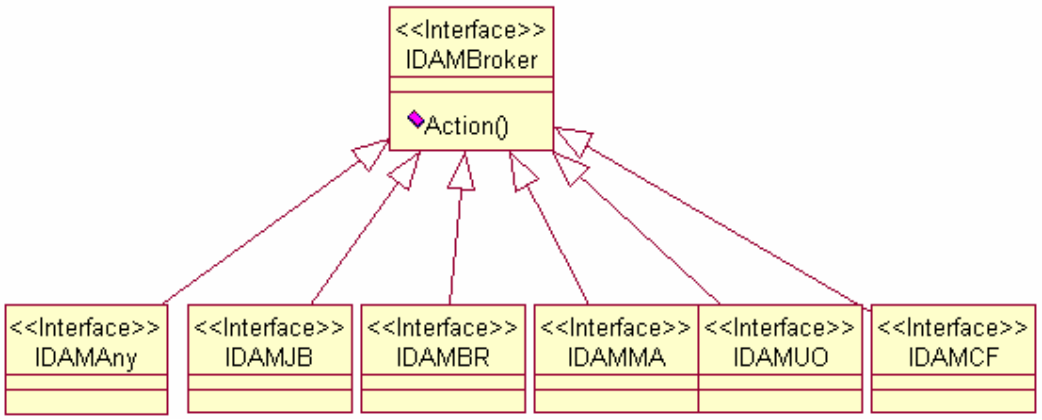

Figura 38. Modelul interfetelor DAL

5 Rezultate experimentale

In cadrul acestui capitol se detaliază testele realizate şi rezultatele experimentale. S-au realizat următoarele tipuri de teste: teste funcţionale şi teste de determinare a nivelului de siguranţă a datelor.

Testele funcţionale realizate sunt:

- teste de performanţă;
- teste de scalabilitate.

Testele funcţionale determină pe de o parte nivelul de performanţă al aplicaţiei distribuite prin evaluarea timpilor de execuţie şi, pe de altă parte, modificarea acestor timpi la scalarea aplicaţiei (teste de scalabilitate). Ceea ce s-a urmărit au fost aspectele legate de performanţa unei aplicaţii de comerţ electronic care implementează modelul SCAR-ACE deoarece, prin introducerea unei maşini de stare, pot aparea eventuale intârzieri. Testele au demonstrat timpi de răspuns mici la o încărcare a aplicaţiei cu 300 de utilizatori simultani şi au validat modelul SCAR-ACE din punct de vedere funcţional.

Testele de determinare a nivelului de siguranţă a datelor sunt:

- teste de răspuns la un atac simulat;
- teste de conformitate cu cerinţele modelului SCAR-ACE.

Testele de răspuns la un atac simulat îşi propun determinarea reacţiei aplicaţiei la încercarea de fraudare a sistemului prin obţinerea şi modificarea credenţialelor. Testele au demonstrat siguranţa aplicaţiei în faţa unui atac.

Testele de conformitate cu cerinţele modelului SCAR-ACE sunt teste care determină modul de realizare a constrângerilor impuse (modul de realizare a separării îndatoririlor statice şi dinamice, modul de realizare a moştenirii rolurilor şi modul de desemnare şi acordare a privilegiilor). Testele au demonstrat conformitatea aplicaţiei cu modelul SCAR-ACE şi au validat modelul.

5.1 Teste de performanţă

Ipoteza de verificat: timpul de răspuns la introducerea maşinilor de stare creşte la încărcarea / descărcarea unei maşini de stare în / din memorie.

Aplicaţia de comerţ electronic a fost testată pentru două tipuri de achiziţii: pentru maşini şi pentru calculatoare. Funcţionalităţile dezvoltate au fost cele pentru rolurile de vizitator, cumpărător, vânzător şi administrator, şi au fost implementate acţiunile necesare realizarii unei comenzi.

Testele efectuate au fost cele de verificare a timpului de răspuns în condiţiile în care au fost lansate 300 de taskuri în paralel.

5.1.1 Descrierea testelor

Sistemul a fost testat în următoarea configurație: serverul - o maşină HP cu procesor Pentium(R) 4 CPU 2.53GHz, 1GB RAM şi 2 clienti cu aceeaşi configurație. Pentru scalabilitate s-au utilizat două servere si doi clienți, toate cu configurația de procesor Pentium(R) 4 CPU 2.53GHz şi 1GB RAM.

Testele au constat din lansarea în execuție a 300 de taskuri care să ruleze în paralel. Astfel:

- primele 100 de taskuri realizează inițierea de sesiuni de către utilizatori (în rolul de vizitator). Pentru fiecare dintre aceste taskuri serverul a executat următoarele acțiuni:
    - crearea unui identificator utilizator şi păstrarea acestuia în memorie;
    - încărcarea maşinii de stare vizitator în memorie din baza de date;
    - memorarea stării curente (ST_HOME) în memorie ca şi stare actuală a utilizatorului nou creat;
    - trimiterea paginii "Home" pe maşina client împreună cu datele de identificare a utilizatorului şi a stării curente.
- următoarele 100 de taskuri sunt pentru trecerea utilizatorilor din rolul de vizitator în rolul de cumpărător prin autentificare. Pentru fiecare task serverul execută următoarele acțiuni:
    - verificarea datelor de autentificare;
    - descarcărea maşinii de stare vizitator din memorie şi încărcarea maşinii de stare cumpărător în cazul în care datele de autentificare au fost corecte (actiunea efectiva de login);
    - emiterea unui mesaj de eroare dacă datele de autentificare au fost greşite;
    - trimiterea unei pagini de răspuns pe maşina client (fie pagina cu informații personale a cumpărătorului fie o pagină de eroare) împreună cu datele de identificare a utilizatorului şi a stării curente.
- ultimele 100 de taskuri reprezintă încheierea sesiunilor de lucru. Serverul execută următoarele acțiuni în acest caz:
    - descărcarea maşinii de stare curente din memorie (acțiunea de logoff);
    - descărcarea datelor de context din memorie pentru fiecare utilizator (starea, informații de acces);
    - scrierea datelor ce necesită a fi păstrate persistent în baza de date (dacă a modificat datele de identificare sau non-identificative, coşul de cumpărături sau cererile ferme).

Testele care s-au făcut au urmărit următorii trei timpi:

- timpul de execuție pentru login (figura 39) si timpul de execuție pentru logoff (figura 40);
- timpul mediu al execuției;
- timpul total al execuției.

A fost considerat timpul de execuție pentru login ca şi test separat deoarece în acest moment este încărcată maşina de stare în memorie (este acțiunea cea mai mare

consumatoare de resurse). Timpul de execuție la logoff reprezintă evoluția sistemului la descărcarea maşinii de stare din memorie si salvarea datelor in baza de date.

Rezultatele obținute sunt cele din figurile de mai jos (figurile 39, 40).

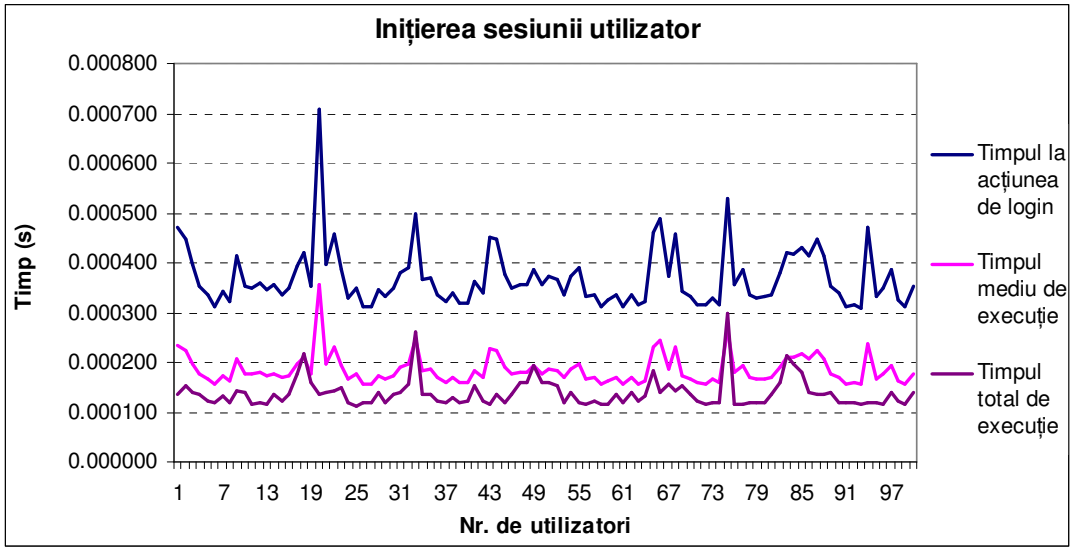

Figura 39. Timpii la autentificarea utilizatorilor

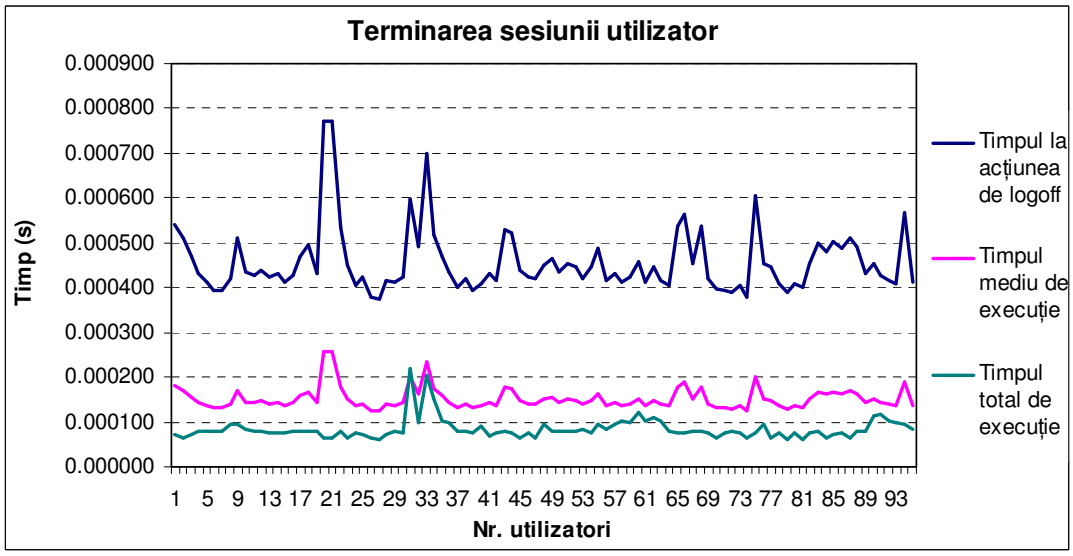

Figura 40. Timpii la terminarea sesiunii utilizator

5.1.2 Interpretarea rezultatelor

Toate cele 300 de taskuri sunt lansate în paralel iar succesiunea acțiunilor executate de server este descrisă în secțiunea anterioară. Trebuie avut în vedere că deşi taskurile au fost lansate în paralel a existat totuşi o secvențialitate a acestora deoarece întreaga aplicație a fost încărcată pe un server cu un singur procesor.

Rezultatele ar fi simțitor mai bune dacă modulele program ar fi încărcate pe mai multe maşini. Aşadar rezultatele obținute sunt cele pentru cel mai defavorabil caz.

Analizând graficele din figurile 39 si 40 se observă că la autentificare si respectiv la terminarea sesiunii utilizator, cei trei timpi studiați nu au aceeaşi valoare.

In graficul din figura 39 timpul mediu de execuție este de circa două ori mai mic față de timpul acțiunii de login, si cu circa 50% mai mare față de timpul total de execuție.

Tot din grafic se observă că la al 20-lea utilizator există un vârf al timpului de execuție. Aceasta deoarece în acel moment server-ul încarcă maşina de stare a cumpărătorului (conform fisierelor de test) pentru primul task care are nevoie de aceasta (în mod firesc, în acest moment primul utilizator care a fost lansat in sistem are nevoie de maşina de stare cumpărător pentru a-şi realiza taskurile impuse).

Incărcarea unei maşini de stare are loc doar pentru primul utilizator dintr-un anumit rol şi descărcarea acesteia are loc la ultimul utilizator care a fost în acel rol.

Aşadar timpul de execuție al celui de-al 20-lea utilizator încapsulează şi încărcarea maşinii de stare cumpărător pentru primul utilizator. Faptul că se realizează încărcarea maşinii de stare (pentru primul utilizator din sistem) simultan cu lansarea celui de-al 20-lea utilizator, este dependentă de configurație şi de puterea de calcul a procesorului (eventual procesoarelor) în lucru. Pentru o configurație diferită aceasta ar putea interveni într-un alt moment şi pentru un alt utilizator.

Analizând în continuare figura 39 (după cel de-al 20-lea utilizator) se observă că timpii devin mai mici şi relativ constanți. Există totuşi şi în continuare anumite vârfuri şi aceasta datorită încărcării serverului, în momentele în care are mai multe acțiuni paralele de executat.

Timpul mediu prezintă vârfuri mai atenuate față de cele ale timpului acțiunii de login si mai accentuate față de timpul total de execuție. Se observă că timpul total de execuție este relativ constant în jurul valorii de 0.0001 s.

Timpii din figura 40 urmăresc evoluția celor din figura 39 şi aceasta deoarece la logoff maşina de stare este descărcată din memorie deci şi in accest caz există timpi de întârziere prin consumarea mai multor resurse.

Concluzie: introducerea maşinilor de stare implică creşterea timpului de execuție la încărcarea /descărcarea acestora în / din memorie. Toți timpii de execuție subsecvenți nu cresc. Timpul total de execuție nu este puternic influențat de lucrul cu maşini de stare.

5.2 Teste de scalabilitate

Ipoteza 1 de verificat: implementarea scalată a modelului SCAR-ACE are timpi de răspuns mai mici decât implementarea nescalată a arhitecturii cu formuri de comenzi.

Sistemul a fost scalat pe două servere şi a fost testat funcțional la use case-ul din figura 41 (test 1) si la use case-ul de autentificare din figura 8 (test 2).

Aplicațiile care au fost selectate pentru testarea timpilor obtinuți prin scalare sunt:

- Prima aplicație implementează modelul SCAR-ACE scalat;
- A doua aplicație implementează arhitectura cu formuri de comenzi nescalata.

Au fost selectate aceste două aplicații pentru a se realiza şi o comparare între cele două implementări de sisteme de comerț electronic.

Primul test (use case-ul din figura 41) implementează următoarea funcționalitate: selectarea celei mai ieftine oferte care respectă anumite criterii funcționale. Oferta este preluată atât din produsele la mâna a doua cât şi din cele noi oferite de sistem, utilizatorului oferindu-i-se mai apoi opțiunea de a o selecta pe cea care îndeplineşte criteriile propri. A fost selectat acest use case deoarece prezintă o acțiune complexă şi se pot testa comparativ timpii de acces pentru scalare.

Al doilea test de scalabilitate verifica timpii de raspuns pentru actiunea de login (use case-ul din figura 8).

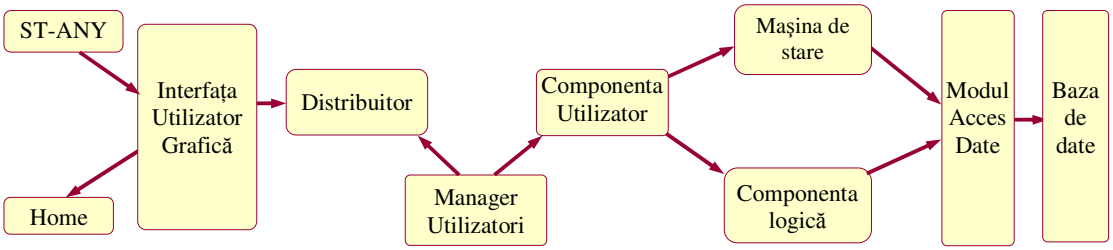

Figura 41. Use case pentru testarea scalabilității

Ipoteza 2 de verificat: timpii de răspuns ai implementarii modelului SCAR-ACE scad prin scalare.

S-a realizat testarea timpilor de răspuns ai implementării modelului SCAR-ACE atât scalat pe două servere cât si nescalat (rulând pe un singur server) pentru acțiunea de login. Au fost comparate cele două variante ale implementării modelului SCAR-ACE (cea scalată si cea nescalată) în funcție de numărul de utilizatori care au participat in sistem.

5.2.1 Interpretarea rezultatelor

Pentru ipoteza 1:

Timpii de răspuns ai primului test de scalabilitate sunt ilustrați in figura 42. S-au luat în considerare 100 de fire de execuție paralele. Se poate observa o diferență între timpii de răspuns pentru cele două situații: prima aplicație (modelul SCAR-ACE scalat) şi cea de a doua aplicație (arhitectura cu formuri de comenzi nescalata), şi anume un timp de execuție mai mic pentru cazul aplicației care implementează modelul SCAR-ACE.

Testul al doilea de scalabilitate a fost selectat pentru o acțiune simplă (înregistrare/login utilizator). Pentru a fi concludent s-a luat in considerare un număr de 170 de utilizatori simultani. Rezultatele testului 2 sunt ilustrate în figura 43. Se observă si in acest caz timpi de răspuns mai mici în cazul aplicației care implementează modelul SCAR-ACE.

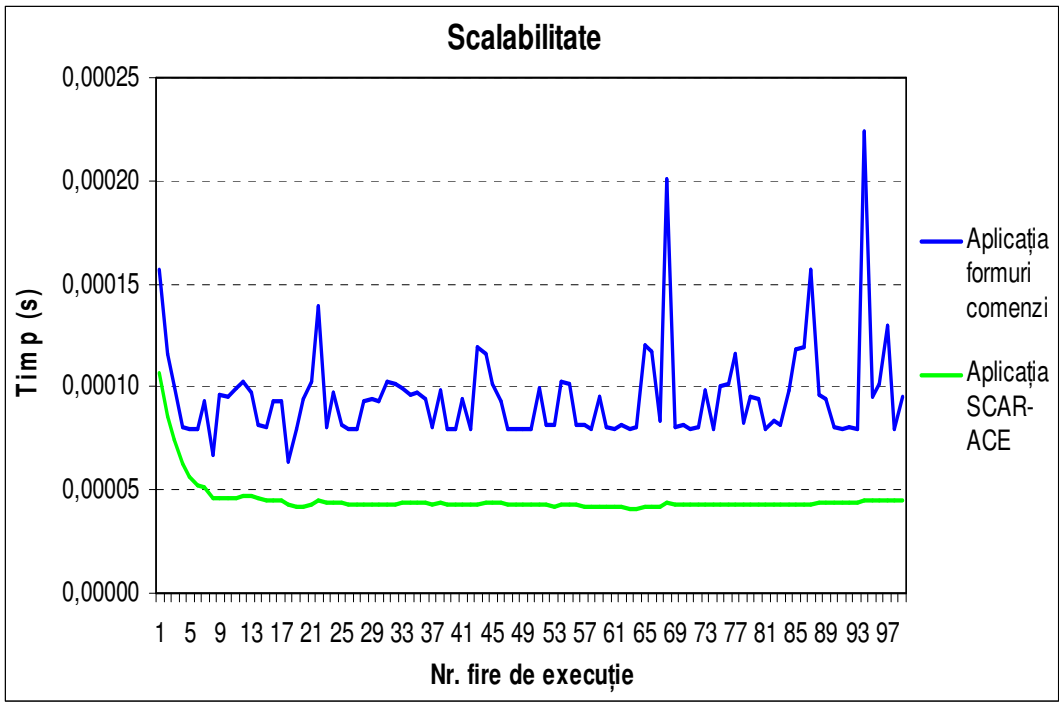

Figura 42. Rezultatele testului 1 de scalabilitate

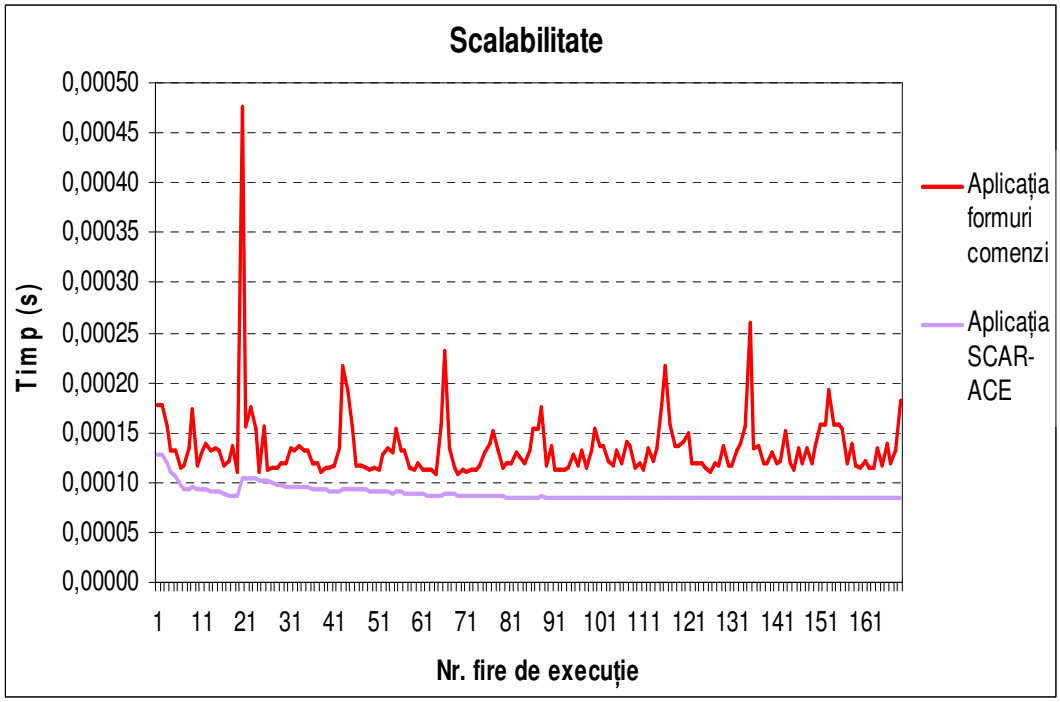

Figura 43. Rezultatele testului 2 de scalabilitate

Concluzia 1: Timpii de răspuns ai aplicației care implementeaza modelul SCAR-ACE scalat sunt mai mici decât cei ai aplicației care implementeaza arhitectura cu formuri de comenzi nescalate.

Pentru ipoteza 2:

Evoluția timpilor de răspuns între cele două variante în funcție de numărul de utilizatori simultani este ilustrată în figura 44. Se observă o diferență din ce în ce mai mare a timpilor de execuție odată cu creşterea numărului de utilizatori.

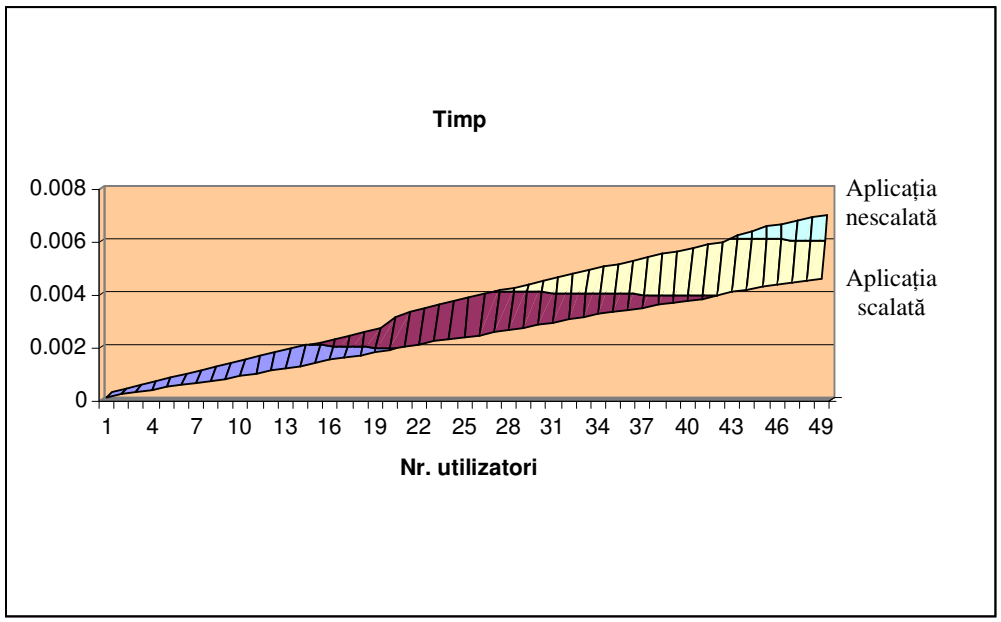

Figura 44. Variatia timpilor totali de executile in aplicatia scalata

Concluzia 2: Timpii de răspuns ai aplicației scalate sunt mai mici decât cei ai aplicației nescalate.

5.3 Teste de răspuns la atac simulat

Ipoteza de verificat: credențialul nu poate fi obținut din nici o stare a aplicației.

Atacul asupra aplicației se referă la încercări de a penetra sistemul în mod fraudulos. In cazul modelului SCAR-ACE acestea ar însemna încercări de acces la informații confidențiale, informații accesibile în mod normal doar prin autentificare. Astfel de informații confidențiale ar putea fi:

- informații de identitate a utilizatorului;
- informații relativ la comenzile realizate;
- informații de plată;
- informații despre starea livrării unui bun.

Pentru a se realiza un atac ar fi posibile următoarele două variante:

a) accesul la credențial;
b) reconstituirea unui credențial.

Accesul la credențial:

Modul în care modelul SCAR-ACE este realizat nu permite vizualizarea credențialului acesta fiind ascuns utilizatorului nefiind prezent în linia de comandă.

Orice încercare de vizualizare este penalizată prin emiterea unei excepţii şi afişarea unei pagini de eroare. Presupunând că în mod fraudulos s-ar fura un credenţial în timpul accesului unui utilizator la informaţiile proprii autorizate, sistemul verifică în mod permanent informaţiile relativ la adresa de IP de la care s-a demarat accesul şi încercarea de acces de la o altă adresă determină încetarea automată a aplicaţiei şi realizarea unui log off forţat.

S-au realizat diverse teste de simulare ale unui atac prin încercarea de vizualizare a unui credenţial în diverse stări ale aplicaţiei şi toate acestea s-au soldat cu închiderea automată a aplicaţiei.

Concluzia 1: Credenţialele nu se pot obţine din aplicaţie.

Reconstituirea unui credenţial:

Dacă un atacator reuşeşte totuşi să obţină în mod fraudulos un credenţial acesta nu ar putea fi utilizat pe durata unei sesiuni utilizator (vezi punctul a). Ar mai avea posibilitatea de a-l utiliza după închiderea sesiunii utilizator. In acest ultim caz există însă următoarea interdicţie: credenţialul nu mai e valabil după închiderea unei sesiuni utilizator şi aceasta deoarece identificatorul unui utilizator expiră după închiderea sesiunii şi la orice acces iniţiat acest identificator utilizator se schimbă. Mai mult decât atât identificatorul utilizatorului este acordat de către server şi nu clientul este cel ce asignează acest număr.

S-a încercat reconstruirea de credenţiale şi s-au simulat atacuri însă nu s-a reuşit penetrarea sistemului nici pentru stări autorizate şi nici pentru cele neautorizate.

Concluzia 2: Credenţialele, chiar dacă ar putea fi obţinute prin diverse mijloace, nu pot fi utilizate în cadrul aplicaţiei deoarece au un timp limitat şi foarte scurt de valabilitate.

5.4 Teste de conformitate cu cerinţele modelului SCAR-ACE

Modelul SCAR-ACE impune anumite cerinţe şi anume:

a) sistemul trebuie să realizeze separarea îndatoririlor;
b) sistemul trebuie să implementeze moştenirea rolurilor.

Separarea îndatoririlor şi moştenirea rolurilor se implementează prin alocarea de maşini de stare diferite pentru fiecare rol. Trecerea de la o maşină de stare la alta se realizează la login. Moştenirea implică posibilitatea de a accesa, dintr-o maşină de stare situată la nivel ierarhic superior, stări dintr-o maşină de stare situată la un nivel ierarhic inferior (de exemplu maşina de stare vizitator poate fi accesată din maşina de stare cumpărător).

Ipoteza de verificat: aplicaţia realizează separarea îndatoririlor.

Testarea s-a realizat cu utilitarul Silktest şi a demonstrat conformitatea funcţionării cu cerinţele modelului SCAR-ACE prin obţinerea de rezultate corecte. S-au efectuat acţiuni în care au fost activi 100 de utilizatori şi au fost lansate simultan 406 fire de execuţie. Pentru verificarea separării îndatoririlor s-a încercat forţarea

atribuţiilor rolului de vizitator prin încercarea de accesare a unor informaţii private specifice rolului de cumpărător:

- informaţii de identitate a unui cumparator – încercare soldată cu eşec;
- informaţii relativ la comenzile realizate de un cumparator - încercare soldată cu eşec;
- informaţii de plată a unei comenzi - încercare soldată cu eşec;
- informaţii despre starea livrării unui bun - încercare soldată cu eşec.

Concluzie: Aplicaţia este conformă cu modelul SCAR-ACE şi implementează separarea îndatoririlor.

Ipoteza de verificat: aplicaţia realizează moştenirea rolurilor.

Pentru verificarea moştenirii rolurilor s-a testat o secvenţă de acţiuni care implică accesul dintr-o maşină de stare în alta. Pentru testarea acestei funcţionalităţi s-au realizat test case-uri care să simuleze secvenţe de acţiuni atât autorizate cât şi neautorizate, teste de la o stare neautorizată la una autorizată şi teste de la o stare autorizată înapoi la una neautorizată.

Exemplu de secvenţă testată:

1. se iniţiază o sesiune;
2. se accesează catalogul de produse;
3. se realizează autentificarea;
4. se selectează un produs în coşul de cumpărături;
5. se efectuează plata;
6. se verifică comanda;
7. se revine la pagina Home.

Această secvenţă a fost realizată pas cu pas şi implică acesarea următoarelor maşini de stare:

1. iniţiere sesiune – maşina de stare vizitator;
2. accesare catalog de produse – maşina de stare vizitator;
3. autentificare – maşina de stare login şi trecere în maşina de stare cumpărător;
4. vizualizare catalog de produse - maşina de stare vizitator (prin moştenire);
5. comparare produse - maşina de stare vizitator (prin moştenire);
6. selectare produs – maşina de stare vizitator (prin moştenire);
7. inserare produs în coşul de cumpărături - maşina de stare cumpărător;
8. efectuare plată – maşina de stare cumpărător;
9. verificare comandă – maşina de stare cumpărător;
10. revenire la pagina Home – maşina de stare vizitator (prin moştenire);
11. logoff.

Concluzie: Aplicaţia este conformă cu modelul SCAR-ACE şi implementează moştenirea rolurilor.

6 Concluzii şi dezvoltări ulterioare

Odată cu creşterea necesităţii de aplicaţii deschise şi distribuite creşte şi cererea pentru politicile de control al accesului. Astfel, din ce în ce mai multe servicii se vor baza pe o infrastructură care să le ofere securitatea accesului. Aceste cerinţe necesita politici adaptive capabile să accepte taskuri colaborative care să îşi îndeplinească sarcinile prin asigurarea autorizării accesului.

6.1 Contribuţia proprie

In această teză s-a descris un model care furnizează o politică de control al accesului bazat pe roluri în sisteme deschise şi distribuite, model care constituie în ansamblul său contribuţia proprie majoră. In detaliu s-a contribuit la următoarele:

- o analiza a modelelor RBAC existente, a cercetărilor actuale şi a colectivelor care au implementat modele RBAC;
- o abordare originală a componentelor RBAC într-o manieră în care să fie soluţionate aspectele legate de controlul accesului bazat pe roluri în sisteme distribuite şi deschise. Astfel, s-a extins modelul standard RBAC cu maşini de stare, prin intermediul cărora se realizează acordarea drepturilor la nivel de acţiune. Aceste maşini de stare au fost introduse în construcţiile modelului astfel încât să poată fi exprimate relaţiile dintre componente. Modelul SCAR-ACE include şi o ierarhie de roluri, ierarhie de tip arbore care permite moştenirea drepturilor;
- un model de realizare a relaţiilor de constrângere şi de separare statică a îndatoririlor tot prin mecanismul bazat pe utilizarea unei maşini de stare pentru fiecare rol. Astfel, un utilizator căruia i-a fost atribuit un rol are ataşată una şi doar una dintre maşinile de stare;
- modelarea sistemului SCAR-ACE prin detalierea pe nivele: SCAR-ACE_0, SCAR-ACE_1, SCAR-ACE_2 şi SCAR-ACE_3;
- definirea modelului formal prin analiza constrângerilor autorizării bazate pe roluri, si a modului de desemnare a privilegiilor în SCAR-ACE. S-au determinat şi s-au descris astfel nivelele de acordare a privilegiilor cu ajutorul UML;
- identificarea modalitatii de acordare a drepturilor în SCAR-ACE prin intermediul credenţialelor şi descrierea componenţei acestora ;
- definirea politicii de acordare a privilegiilor şi a algebrei acestei politici;
- modelarea accesului prin intermediul maşinilor de stare şi descrierea modului în care este realizat controlul accesului în SCAR-ACE prin caracterizarea rolurilor şi identificarea constrângerilor asupra acestora;
- definirea modelului formal al SCAR-ACE şi exemplificarea acestuia pentru un caz concret;
- identificarea fazelor specificării constrângerilor de autorizare:
    - faza de analiză statică;
    - faza de verificare/actualizare;
    - faza de planificare;

    - maşina de stare;
    - interfaţa utilizator grafică;
    - faza de rulare.
- analiza aplicatiilor de comert electronic, a rolurilor si a arhitecturilor existente;
- realizarea implementarii modelului SCAR-ACE si a validarii acestuia prin realizarea de studii de caz si obtinerea de rezultate experimentale.

6.2 Dezvoltări ulterioare

Prezenta teză poate fi considerată un pas iniţial în cercetarea politicilor controlului accesului bazat pe roluri în sistemele deschise şi distribuite. Există mai multe posibile dezvoltări ulterioare detaliate în paragrafele următoare.

Politica abordată extinde modelul de bază RBAC astfel încât sunt suportate componentele standard cărora li se adaugă componente specifice necesare gestionării accesului cu ajutorul maşinilor de stare. Aceste componente proprii pot fi examinate pentru a se determina dacă se pot utiliza în tranzitia de la un sistem RBAC existent la altul.

Modelul SCAR-ACE poate deveni destul de complex dacă se iau în considerare ierarhii diferite de roluri, modelul actual fiind specific ierarhiilor de roluri de tip arbore. O extensie ar putea lua in considerare si alte tipuri de ierarhii.

Abordarea curenta organizează şi structurează politicile de control al accesului în aplicaţiile de comerţ electronic dar acest model poate fi investigat şi pentru alte tipuri de aplicaţii. Maşinile de stare care constituie extensiile sunt, în general vorbind, nişte grafuri şi implementarea acestora necesită cunoştinţe avansate. O direcţie nouă ar putea implementa o interfaţă grafică pentru designul unei aplicaţii bazate pe modelul SCAR-ACE care să aibă drept intrare stările, evenimentele şi tranziţiile dintre stări şi să genereze ca şi ieşire maşinile de stare.

Aplicaţia de comerţ electronic oferă suportul in implementarea modelului SCAR-ACE pentru controlul accesului bazat pe roluri. Această aplicaţie include atât componente de business cât şi componente de administrare, componente care fie formează grupuri fie sunt predefinite în sistem. Unele componente au nevoie de condiţii prerechizite. O direcţie de cercetare constă în extinderea părţii de business astfel încât să poată fi gestionate diverse verticale cu includerea de facilităţi (precum achiziţii de cărţi cu includerea profilului cumpărătorilor pentru notificare şi includerea contextului în care cumpărarea are loc).

Pe de alta parte politica de componente poate fi marcată cu elemente de context, permiţând astfel administratorului să grupeze componente în funcţie de anumite condiţii. Aceasta ar permite vederi asupra politicilor de componente posibil a fi exploatate sub forma unor tools-uri de vizualizare. In funcţie de punctul de vedere considerat s-ar putea determina de exemplu cumpărători care satisfac anumite criterii fie din punct de vedere al securitatii fie din punct de vedere al bunurilor achiziţionate.

O altă extensie posibilă a modelului implică asocierea rolurilor la procese de business. In acest caz s-ar permite utilizarea modelului pentru un workflow în care maşina de stare urmăreşte fluenţa părţii de business a unei aplicaţii. Chiar în contextul actual modelul urmăreşte fluenţa controlului şi, în acest sens, modificările ar fi minore şi ar consta în adăugarea de etichete de proces.

SCAR-ACE ia în considerare restricţii la nivel înalt considerând primitivele modelului ca fiind rolurile, drepturile şi maşina de stare. Pentru a se netezi diferenţele dintre diferite modele RBAC ar fi de interes o cercetare în care primitivele ar fi triplele (obiect, acces, acţiune) şi extinderea prin crearea unor politici echivalente. Această cercetare ar consta, în special, în determinarea constrângerilor.

O posibilitate de extindere ar fi prin încapsularea componentelor modelului SCAR-ACE într-o interfaţă între diferite modele RBAC existente. Fiecare model ar fi o stare în cadrul maşinii de stare şi s-ar putea realiza cooperarea între diferite modele RBAC. In astfel de medii, politicile interfeţei se pot utiliza în medierea diferenţelor dintre diferitele domenii de control al accesului. Primul scop al politicilor de interfaţă ar fi cel de a permite accesul între politicile de domenii. In cazul în care este posibilă o cooperare, aceasta ar putea depinde şi de politicile de administrare. In anumite cazuri tranziţiile între modele s-ar putea realiza pe bază de încredere adică fără necesitatea autentificării. Astfel apare posibilitatea extinderii pentru politici de conformitate pentru deciziile colaborării inter-domenii, decizii bazate pe încredere, după care s-ar putea utiliza politicile de interfaţă pentru a realiza în mod automat colaborarea între modele.

Politica de gestionare a drepturilor furnizează un mijloc de control al accesului la o granularitate foarte fină. Aceste drepturi sunt obtinute în SCAR-ACE prin intermediul maşinii de stare. Aceste drepturi pot fi însă ordonate într-o ierarhie care să permită o politică mai concisă de acordare a acestora. O extensie posibil a fi adusă drepturilor ar fi asocierea acestora cu un factor de risc. Astfel, acţiunile permise nu sunt egale din punct de vedere al răului potenţial pe care îl pot aduce în sistem la o folosire defectuoasă. Prin asocierea acestora cu un factor de risc se pot emite anumite politici care să specifice supervizări suplimentare in acordarea accesului pentru drepturile cu factor ridicat de risc. Astfel, un risc ar fi o valoare care, în funcţie de unul sau mai multe praguri, ar avea sau nu nevoie de supervizări suplimentare.

Printre motivaţiile importante ale cercetării prezente este dezvoltarea unei politici de acces în sistemele distribuite şi deschise care să controleze accesul în funcţie de rolul utilizatorului. O extensie ar fi extinderea în cadrul diverselor aplicaţii precum cele din domeniile: sănătate, telecomunicaţii, gestionarea resurselor umane şi materiale, şi orice alte servicii Web în care se pot evidenţia mai multe roluri cu privilegii necesare diferite.

Anexa 1 Bibliografie

[AB06] M.J. Atallah, M. Blanton, *Key management for non-tree access hierarchies*, Symposium on Access Control Models and Technologies, March 2006, pp. 11-18, ISBN: 1-5959-353-0

[AS00] G. Ahn, R. Sandhu, *Role-based authorization constraints specification,* ACM Transaction Information Systems, Sect. 3, 4, Nov. 2000, pp. 207-226, ISSN: 1094-9224

[AW[1]03] K. Alghathbar, D. Wijesekera, *AuthUML: A Three-phased Framework to model Secure Use Cases,* Proceedings of the Formal Methods în Security Engineering Workshops, (FMSE'03), Oct. 2003, pp. 77-86, ISBN: 4790-7688

[AW[2]03] K. Alghathbar, D. Wijesekera, *Consistent and Complete Access Control Policies in Use Cases,* Proceedings of UML 2003, LNCS, 2003

[AW05] V. Atluri, J. Wagner, *Supporting Conditional Delegation in Secure Workflow Management Systems, SACMAT'05*, Iunie, 2005, ISBN: 1-59593-045-0

[B75] K. J. Biba, Integrity consideration for secure computer systems, *Technical Report MTR-3153*, MITRE Corporation, Bedford, MA, April 1975

[B95] J. Barkley, *Implementing role based access control using object technology,* Proceedings of the ACM Workshop of Role Based Access Control, November 1995, pp. 20-es., ISBN: 0-89791-759-6

[B97] L. Bartz, *hyperDRIVE: Leveraging LDAP to implement RBAC on the web,* Proceedings of the ACM Workshop of Role Based Access Control, 1997, pp.69-74, ISBN: 0-89791-985-8

[B05] K. Beznosov, *Future direction of access control models, architectures, and technologies*, Proceedings of the tenth ACM symposium on Access control model and technologies, SACMAT'05, Iunie, 2005, pp. 48, ISBN: 1-59593-045-0

[B06] C. Bibere, Dezvoltarea comerțului realizat pe Internet, *Teza de doctorat*, 2006, http://www.cnaa.acad.md/thesis/5670/

[B[+]91] G. Booch, et all, *Object-Oriented Design With Applications*, Benjamin/Cummings, 1991

[BBF00] E. Bertino, P. A. Bonatti, E. Ferrari, *TRBAC: a temporal role-based access control model,* Proceedings of the Fifth ACM Workshop on Role-Based Access Control (RBAC'00), pp 21–30, 2000, ISBN: 8970-6549.

[BBF01] E. Bertino, P. A. Bonatti, E. Ferrari, *TRBAC: A temporal rolebased access control model*, ACM Transactions on Information and System Security (TISSEC), pp. 191–233, August 2001, ISSN: 1994-9224.

[BCD[+]05] E. Bertino, B. Catania, M. L. Damiani, P. Perlasca, *GEO-RBAC: a spatially aware RBAC,* Proceedings of the tenth ACM symposium on Access control models and technologies**,** June 2005, pp. 29-37, ISBN: 1-59593-045-0

[BDL03] D. Basin, J. Doser, T. Lodderstedt, *Model driven security for process-oriented systems,* Proceedings of the eight ACM symposium on Access control model and technologies, SACMAT '03, June 2003, pp.100-109, ISBN: 1-58113-681-1

[BFA97] E. Bertino, E. Ferrari, V. Atluri, *A flexible model supporting the specification and enforcement of role-based authorization in workflow management systems*, Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC'97), pp. 1–12, 1997, ISBN: 0-89791-985-8

[BFA99] E. Bertino, E. Ferrari, V. Atluri, *The specification and enforcement of authorisation constraints in workflow management systems*, ACM Transactions on Information Systems Security, November 1999, pp. 65-104, ISBN: 1094-9224

[BHM06] S. Brlek, S. Hamadou, J. Mullins, *Some Remarks on the Certificates Registration on the Electronic Commerce Protocol SET*, Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, February 2006, pp. 119, ISBN: 1094-9225

[BJS96] E. Bertino, S. Jajodia, P. Samarati, *Supporting multiple access control policies in database systems*, IEEE Symposium on Security and Privacy (SSP'96), 1996, pp. 94-107, ISBN: 0-8186-7417-2

[BK03] J. Biskup, Y. Karabulut, *A hybrid pki model – Application to secure mediation,* Research Directions în Data and Application Security, 2003, www.informatik.uni-dortmund.de/uploads/tx_ls6ext/Biskup_ Karabulut_2002a.pdf

[BL76] D. E. Bell, L. J. LaPadula, *Secure computer systems: Unified exposition and Multics interpretation*, Technical Report MTR-2997, The MITRE Corporation, July 1975, www.csrc.nist.gov/publications/history/bell76.pdf

[BLN82] A.D. Birrell, R. Levin, R. M. Needham, M.D. Schroeder, *Grapevine: An exercise în distributed computing,* Communication of the ACM, pp. 260-274, April 1982, ISSN: 0001-0782

[BM+07] G. Booch, R. Mansimchuck and all., *Object Analysis and Design with Applications*, Object Software Engineering, 3rd edition, 2007, ISBN-13: 978-0201895513

[BMY02] J. Bacon, K. Moody, W. Yao, *A model of OASIS role-based access control and its support for active security,* ACM Transactions on Information and System Security (TISSEC), pp. 492–540, November 2002, ISSN: 1094-9224

[BN89] D. Brewer, M. Nash, *The Chinese wall security policy*, Proceedings of th Symposium on Security and Privacy, IEEE Press, pp 215-228, 1989

[BPM02] G. Bella, L. Paulson, F. Massacci, *Cryptographic protocols: The verification of an industrial payment protocol: the SET purchase phase*, Proceedings of the 9th ACM conference on Computer and communications security, November 2002, pp. 12-22, ISBN: 1-58113-612-9

[BS[1]00] E. Barka, R. Sandhu, *Framework for role-based delegation models*, Sixteenth Annual Computer Security Applications Conference, New Orleans, Louisiana, December 2000, Digital ID: 10.1109/ACSAC.2000.898870.

[BS[2]00] E. Barka, R. Sandhu, *A role-based delegation model and some extensions*, Twenty-third National Information Systems Security Conference, Baltimore, MD, October 2000, www.csrc.nist.gov/nissc/2000/proceedings/papers/021slide.pdf

[BW04] J. Biskup, S. Wortmann, *Towards a credential-based implementation of compound access control policies,* Proceedings of the ninth ACM symposium on Access control models and technologies**,** June 2004, pp. 31-40, ISBN: 1-58113-872-5

[C89] S.K. Card, *Human factors and artificial intelligence,* Intelligent Interfaces Theory, Research and Design, pp 27-41, New York, 1989

[C03] D. Clark, *Economics and the Design of Open Systems*, IEEE Internet Computing, March 2003, pp. 94-96

[CK96] A.O. Freier, P.Carlton, P. Kocher, *The SSL Protocol version 3.0*, 1996, http://ieeexplore.ieee.org/iel5/7739/21247/00985877.pdf

[CLS+01] M. J. Covington, W. Long, S. Srinivasan, Anind K. Dev, Mustaque Ahamad, G. D. Abowd, *Securing context-aware applications using environment roles*, Sixth ACM Symposium on Access Control Models and Technologies (SACMAT'01), pp. 10–20, 2001, ISBN: 1-58113-350-2.

[CR06] S. Chakraborty, I. Ray, *TrustBAC: integrating trust relationships into the RBAC model for access control in open systems*, Proceedings of the eleventh ACM symposium on Access control models and technologies SACMAT '06**,** June 2006, pp. 49-58, ISBN:1-59593-353-0

[CW87] D. Clark, D. Wilson, *A comparison of commercial and military computer security policies*, Proceedings of th Symposium on Security and Privacy, IEEE Press, pp. 184-194, 1987, ISBN: 10-1109

[D02] N. C. Damianou, A Policy Framework for Management of Distributed Systems, PhD thesis, Department of Computing, Imperial College of Science, Technology and Medicine, University of London, 2002.

[DD+04] T. Doan, S. Demurjian and all., *MAC and UML for secure software design,* Proceedings of the Formal Methods în Security Engineering Workshops, (FMSE'04), Oct. 2004, pp. 75-85, ISBN:1-58113-971-3

[DDT+04] T. Doan, S. Demurjian, T. C. Ting, A. Ketterl, *Security & analysis II: MAC and UML for secure software design,* Proceedings of the 2004 ACM workshop on Formal methods -în security engineering**,** October 2004, pp. 75-85, ISBN:1-58113-971-3

[DK01] Q. Dai, R. Kauffman, *Business Models for Internet based E-Procurement Systems and B2B Electronic Markets: An Exploratory Assessment*, 34th Annual Hawaii International Conference on System Sciences – Volume 7, January 2001, pp. 704

[DDL+01] N. Damianou, N. Dulay, E. Lupu, M. Sloman, *The Ponder policy specification language*, Policies for Distributed Systems and Networks, International Workshop (POLICY'01), Bristol, UK, pp. 18–38, 2001 [EE01] The electronics industry supply chain: who does what?, *38th Conference of Design Automation*, 2001, www.doc.ic.ac.uk/~mss/Papers/Ponder-Policy01V5.pdf

[EF99] Robert J. Ellison, David A. Fisher, et all, *Survivability: Protecting Your Critical Systems,* IEEE Internet Computing, November1999, pp. 55-63, Digital Object Identifier 10.1109/4236.807008

[ES99] P. Epstein, R. Shandu*, Towards a UML Based Approach to Role Engineering*, Proceedings of the 4th ACM workshop on Role-based Access Control, 1999, pp. 135-143, ISBN:1-58113-180-1

[F00] C. A. Fregly, *Techniques for Optimizing Web Site Development and Runtime Characteristics,* Java Report, November 2000, http://www.adtmag.com/java/articleold.aspx?id=1614

[FBK98] D. Ferraiolo, J. Barkley, D. Kuhn, *A role-based access control model and reference implementation within a corporate intranet*, ACM Transaction on Information and System security, 2, February 1998, pp.34-64, ISSN:1094-9224

[FCK95] D. Ferraiolo, J. Cugini, R. Kuhn, *Role-based access control: Features and motivations,* Proceedings of Annual Computer Security Applications Conference, IEEE Press, 1995, hissa.ncsl.nist.gov/rbac/newpaper/rbac.html

[FGL95] D. Ferraiolo, D. Gilbert, N. Lynch, *An examination of federal and commercial access control policy needs,* Proceedings of the NIST_NSA National (USA) Computer Security Conference, pp. 107-116, 1995, csrc.nist.gov/rbac/sandhu96.pdf

[FK92] D. Ferraiolo, R. Kuhn, *Role-based Access Control,* Proceedings of 15th NIST-NCSC National Computer Security Conference, Gaithersburg, 1992, pp.554-563, csrc.nist.gov/rbac/

[FL00] S. Fong, C. Se-Leng, Modeling personnel and roles for electronic commerce retail, *Proceedings of the 2000 ACM SIGCPR conference on Computer personnel research*, April 2000, pp. 45-53, ISBN:1-58113-212-X

[FSG01] David F. Ferraiolo, Ravi Sandhu , Serban Gavrila , D. Richard Kuhn , Ramaswamy Chandramouli, *Proposed NIST standard for role-based access control*, ACM Transactions on Information and System Security (TISSEC), v.4 n.3, pp.224-274, August 2001 (1), ISSN:1094-9224

[FSN05] J. Froberg, K. Sandstrom, C. Norstrom, *Bussiness situation reflected in automotive electronic architectures: analysis of four commercial cases*, Proceedings of the 2005 workshop on Software engineering for automotive systems, 2005, ISBN:1-59593-128-7

[G96] L. Giuri, *Role-based access control: a natural approach*, Proceedings of the First ACM Workshop on Role-Based Access Control (RBAC'95), pp. II–33–37, 1996, ISBN:0-89791-759-6

[G98] L. Giuri, *Role-based access control in Java*, Proceedings of the Third ACM Workshop on Role-Based Access Control (RBAC'98), pp. 91–100, 1998, ISBN:1-58113-113-5

[G99] L. Giuri, *Role-base access control on the web using java*, Proceedings of the ACM Workshop on Role-Based Access Control, 1999, pp. 11-18, ISBN:1-58113-180-1

[G00] A. Gupta, *Automating Any-to-Any Content Exchange,* IEEE IT Professional, Sept. 2000, pp. 52-54, Digital Object Identifier 10.1109/6294.877499

[GD75] G. S. Graham, P.J. Denning, *Protection – principles and practice*, Procedeengs of the fifth ACM symposium on Operating systems principles, May 1975, pp. 14-24, ISSN:0163-5980

[GGF98] V.D. Gligor, S.I.Gavrila, D.F. Ferraiolo, *On the formal definition of separation-of-duty policies and their composition*, Proceedings of the Symposium on Security and Privacy, IEEE Press, Los Alamitos, California, 1998, www.glue.umd.edu/afs/glue.umd.edu/home/enee/faculty/gligor/pub/oak98.ps

[GGK+89] M. Gasser, A. Goldstein, C. Kaufman, B. Lampson, *The Digital distributed system security architecture*, Proceedings of the 1989 National Computer Security Conference, pp. 305-319, October 1989, research.microsoft.com/Lampson/41-DigitalDSSA/41-DigitalDSSA.htm

[GHS95] D. Georgakopoulos, M. Hornik, A. Sheth, *An overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure,* Distributed and Parallel Databases, pp 119-153, 1995, ISSN:0926-8782

[GI96] L. Giuri, P. Iglio, *A formal model for role based access control with constraints*, Proceedings of the Computer Security Foundations Workshop,

IEEE Press, Los Alamitos, California, pp.136-145, 1996, Digital Object Identifier 10.1109/CSFW.1996.503698

[GI97] L. Giuri, P. Iglio, *Role templates for content-based access control*, Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC'97), pp. 153–159, 1997, ISBN:0-89791-985-8

[GM90] M. Gasser, E. McDermott, *An architecture for practical delegation in a distributed system*, Proceedings of the 1990 IEEE Symposium on Security and Privacy, pp. 20-30, May 1990, ISBN: 63835-X

[GMS94] C. H. Goh, S. E. Madnick, M. D. Siegel, *Context interchange: overcoming the challenges of large-scale interoperable database systems in a dynamic environment*, Proceedings of the third International Conference on Information and Knowledge Management, pp. 337–346, ACM Press, 1994, ISBN:0-89791-674-3

[HA99] Wei-Kuang Huang, V. Atluri, *SecureFlow: a secure Web-enabled workflow management system*, Proceedings of the Fourth ACM Workshop on Role-Based Access Control (RBAC'99), pp. 83–94, 1999, ISBN: 1-58113-180-1

[HBP+99] J. Hands, M. Bessonov, A. Patel, R. Smith, *An Inclusive Architecture for Electronic Brockerage*, 32nd Annual Hawaii International on System Sciences – Volume 8, January 1999, pp. 8025, Digital Object Identifier 10.1109/HICSS.1999.773056

[HD95] M.Y.Hu, S.A. Demurjian, T.C. Ting, *User-Role based Security în the ADAM Object-Oriented Design and Analyses Environment*, Database Security VIII: Status and Prospects, 1995, pp. 333-348, ISBN:0-89791-808-8

[HH02] O. Henfridsson, H. Holmstrom, *Developing e-commerce in internetworked organizations: a case of customer involvment throughout the computer gaming value chain*, ACM SIGMIS, 2002, pp. 38-50, ISSN:0095-0033

[HL04] Horst F. Wedde, Mario Lischka, *Modular authorization and administration*, ACM Transactions on Information and System Security (TISSEC), Volume 7 Issue 3, August 2004, pp. 363-391, ISSN:1094-9224

[HTTP103] *Platform for Privacy Preferences* - http://www.w3.org.P3P/ , 2003

[HTTP203] *Critical Foundations* - http://www.pccip.gov , 2003

[HTTP1] *Petri Networks: Definition and basic ideas*, Examples, Reference, http://staff.um.edu.mt/jskl1/petri.html

[HTTP2] *OASIS. Assertion and Protocol for the OASIS Security Assertion Markup Language (SAML)*, Committee Specification 01, Organization for the Advancement of Structured Information Standard, October 2002, http://www.oasis-open.org/committees/security/

[HTTP3] *Limbajul Alloy*, http://web.mit.edu/~rseater/www/tutorial2/alloy-tutorial.html

[HTTP4] *Standardul RBAC dezvoltat de NIST*: http://csrc.nist.gov/rbac/

[HTTP5] *Componente software RBAC (RBAC for UNIX/POSIX/Linux and RBAC for Windows NT)*, http://csrc.nist.gov/rbac/

[HTTP06] DACS 2006: The Distributed Access Control System, http://dacs.dss.ca/

[J98] W.A. Jansen, *Inheritance Properties of Role Hierarchies*, 21st National Information Systems Security Conference, October, 1998, http://csrc.nist.gov/nissc/1998/papers.html

[J99] T. Jaeger, *On the increasing importance of constraints*, Proceedings of the Fourth ACM Workshop on Role-Based Access Control (RBAC'99), pp. 33–42, 1999, ISBN:1-58113-180-1

[J02] J. Jurgens, *Principles for Secure System Design*, Ph.d. dissertation. Oxford University Computing Laboratory, Oxford University, 2002

[J05] J. Jurgens, *Security: Sound methods and effective tools for model-based security engineering with UML*, Proceedings of the 27th international conference on Software engineering ICSE '05, Mai 2005, pp. 322-331, ISBN:1-59593-963-2

[J+92] I. Jacobson, et all, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992

[J93] D.Jonscher, *Extending Access Control with Duties – Realized by Active Mechanisms*, Database Security VI: Status and Prospects, 1993, pp. 91-111, ISBN:0-444-89889-1

[JSS97] S. Jajodia, P. Samarati, V. S. Subrahmanian, *A logical language for expressing authorizations,* Proceedings of IEEE Symposium on Security and Privacy (SSP'97), pp. 3142, 1997, ISSN:1094-9224

[JSS+97] S. Jajodia, P. Samarati, V. Subrahmanian, E. Bertino, *A unified framework for enforcing multiple access control policies*, SIGMOD'97, pp. 474–485, 1997, ISSN:0163-5808

[JSS+01] S. Jajodia, P. Samarati, M. L. Sapino, V. S. Subrahmanian, *Flexible support for multiple access control policies*, ACM Transactions on Database Systems (TODS'01), pp. 214–260, 2001, ISSN:0362-5915

[JT00] T. Jaeger, J. Tidswell, *Rebuttal to the NIST RBAC model proposal,* Proceedings of the Fifth ACM Workshop on Role-Based Access Control, pp. 65-66, July 2000, ISBN:1-58113-259-X

[K94] M. Kempe, *A framework for the blackboard model*, 1994, http://citeseer.ist.psu.edu/127277.html

[K1 97] R. Kuhn, *Mutual exclusion as a means of implementing separation of duty requirements în role based access control systems*, Proceedings of the Second ACM Workshop on Role Based Access Control, pp. 23-30, 1997, ISBN:0-89791-985-8

[K2 97] D.R. Kuhn, *Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role Based Access Control Systems*, Second ACM Workshop on Role Based Access Control, 1997, pp. 23-30, ISBN:0-89791-985-8

[KA98] S.Kent, R. Atkinson, *Security Architecture for the Internet Protocol*, RFC 2401, Nov. 1998, http://www.ietf.org/rfc/rfc2401.txt

[KH02] C. Joncheng Kuo, Polar Humenn, *Session 4: Web service applications: Dynamically authorized role-based access control for secure distributed computation,* Proceedings of the 2002 ACM workshop on XML security**,** November 2002, pp.97-103, ISBN:1-58113-632-3

[KS03] M. A. Al-Kahtani, R. Sandhu, *Induced role hierarchies with attribute-based RBAC*, Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies (SACMAT'03), pp. 142–148, ACM Press, 2003, ISBN:1-58113-681-1

[L71] B. Lampson, Protection, Proceedings of the Fifth Annual Princeton Conference on Information Sciences and Systems, pp 437–443, Princeton University, 1971

[L89] J.W. Lloyd, *Foundation of Logic Programming*, Springer-Verlag, 1989, http://prism.cs.umd.edu/papers/banquet89/banquet89.html

[L97] P. Lawrence, *Workflow Handbook 1997*, Workflow Management Coalition , Jan. 1997, ISBN-10: 0471969478

[L98] E. Lupu, *A Role-Based Framework for Distributed Systems Management*, PhD thesis, Department of Computing, Imperial College of Science, Technology and Medicine, University of London, 1998.

[L05] J. Linn, *Technology and Web User Data Privacy: A Survey of Riskjs and Countermeasures*, IEEE Security and Privacy, 2005, pp. 52-58

[LL98] L. Lefebvre, E. Lefebvre, *The virtual economy as an emerging paradigm*, 31st Annual Hawaii International Conference on System Sciences, Volume 6, 1998, pp. 33, ISBN: 0-8186-8255-8

[LPL+03] M. Lorch, S. Proctor, R. Lepro, D. Kafura, *First experiences using XACML for access control in distributed systems*, Workshop in XML Security, 2003, pp. 25-37, ISBN:1-58113-777-X

[LS97] E. Lupu, M. Sloman, *Reconciling role based management and role based access control*, Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC'97), pp. 135–141, 1997 ISBN:0-89791-985-8

[LS99] E. Lupu, M. Sloman, *Conflicts in policy-based distributed systems management*, IEEE Transactions on Software Engineering, pp. 852–869, 1999, Digital Object Identifier 10.1109/32.824414

[M+92] D.L. Maulsby and all, *Inferring graphical procedures: The complete metamouse*, Human-Computer Interaction, pp 47-90, 1992, www.srs4702.forprod.vt.edu/pubsubj/pdf/03t3.pdf

[M94] S. Mullender, *Distributed Systems*, ACM Press New York, ISBN 0-201-62427-3, 1994

[M95] P.V. McMahon, *SESAME V2 Public Key and Authorization Extensions to Kerberos,* Proceedings of the 1995 Symposium on Network and Distributed System Security, pp 114-131, February 1995

[M98] J.D. Moffet, *Control principles and role hierarchies*, Proceedings of the Third ACM Workshop on Role-Based Access Control, Oct. 1998, pp. 63-69, ISBN:1-58113-113-5

[M99] C. Metz - *AAA Protocol: Authentication, Authorization and Accounting for the Internet,* IEEE Internet Computing, 1999, pp. 75-79, ISBN: 4236-807015

[MN88] S.P. Miller, B.C. Neuman, J. I. Schiller, J.H. Saltzer, *Section E.2.1: Kerberos Authentication and Authorization System*, Project Athena Technical Plan, MIT Project Athena, Cambridge, Massachusetts, October 1988, http://web.mit.edu/Kerberos/papers.html

[N93] B. C. Neuman, *Proxy-based authorization and accounting for distributed systems,* Proceedings of the 13th International Conference în Distributed Computing Systems, pp 283-291, May 1993, ISBN: 0-8186-3770-6

[NC00] SangYeob Na, SuhHyun Cheon, *Role delegation in role-based access control*, Proceedings of the Fifth ACM Workshop on Role-Based Access Control (RBAC'00), pp. 39–44, 2000, ISBN:1-58113-259-X

[NO94] M. Nyanchama, S. Osborn, *Access Rights Administration în Role-Based Security Systems,* Database Security VIII: Status and Prospects, 1994, pp. 37-56

[NO95] M. Nyanchama, S. Osborn*, The role graph model*, Proceedings of the First ACM Workshop on Role-Based Access Control (RBAC'95), pp. 25–31, 1995, ISSN:1094-9224

[NO99] M. Nyanchama, S. Osborn, *The role graph model and conflict of interest*, ACM Transactions on Information and System Security (TISSEC), pp. 3–33, 1999, ISSN:1094-9224

[NS02] G. Neumann, M. Strembeck, *A scenario-driven role engineering process for functional rbac roles*, Seventh ACM Symposium on Access Control Models and Technologies (SACMAT'02), p. 33–42, ACM Press, 2002, ISBN:1-58113-496-7

[O97] S. Osborn, *Mandatory access control and role-based access control revisited*, Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC'97), pp. 31–40, 1997, ISBN:0-89791-985-8

[O00] M. Ordean, *An application framework for e-commerce platforms*, e-Commerce Simpozion, Bilateral cooperation Romania-Croatia, Cluj-Napoca, oct. 2000

[O02] S. Osborn, *Information flow analysis of an rbac system*, Seventh ACM Symposium on Access Control Models and Technologies (SACMAT'02), pp. 163–168, ISBN:1-58113-496-7

[OG05] M.Ordean**,** D.Gorgan, *Role-based authorization using graphical user interfaces in distributed systems,* ROCHI Cluj-Napoca Romania, September 2005, ISBN 973-7973-24-0

[OG[1]07] M. Ordean, D. Gorgan, *"FORMAL DEFINITION OF SCAR-ACE ROLE-BASED ACCESS CONTROL MODEL"*, Proceedings CSCS-16, 16th International Conference on Control System and Computer Science, 22-25 May 2007, Bucharest, Vol. 2, pp 155-159, ISBN:978-973-718-743-7

[OG[2]07] M. Ordean, D. Gorgan, *RBAC and SCAR-ACE Role Based Access Models*, VIPSI 2007 Venice, International Conferences on Advances in the Internet, Processing, Systems, and Interdisciplinary Research, March 19-22, 2007, Italy, Book of Abstracts, ISBN 86-7466-117-3, pp. 12

[OGI05] M. Ordean, D. Gorgan, I. Ignat, *Security Tiers Specification by RBAC and UML,* Scientific Journal of Automation Computers Applied Mathematics (ACAM), vol. 14, 2005, no 1, ISSN 1221-437X, pp. 43-50

[OMG1] OMG. ptc/01-06-17: *CSIv2 proposed finalized specification, Non-change barred proposed finalized specification of CSIv2*, http://www.omg.org/cgi-bin/apps/doclist.pl

[OMG2] OMG. formal/02-06-01 *(CORBA 3.0 version)* http://www.omg.org/cgi-bin/doc?formal/02-06-01

[OMG3] OMG. *Authorization Token Layer Acquisition Service Specification*, Object management Group, October 2001, http://cgi.omg.org/cgi-bin/doc?ptc/2001-10-22

[P93] R. Puerta, The *study of models of intelligent interfaces,* International Conference on Intelligent User Interfaces, pp: 71 - 78, 1993, ISBN: 0-89791-556-9

[PP95] T. Parker, D. Pinkas, *"SESAME V4 – Overview"*, December 1995, http://www.esat.kuleuven.ac.be/cosic/sesame/doc-txt/overview.txt

[PS00] J. Park, R. Sandhu, *Secure Cookies on the Web*, IEEE Internet Computing, July 2000, pp. 36-44, Digital Object Identifier 10.1109/4236.865084

[PS05] S. Park, N. Suresh, *An investigation of roles of electronic marketplace in the supply chain*, Procedeengs of the 38th Annual Hawaii International Conference on System Sciences, 2005, pp. 161b, Digital Object Identifier 10.1109/HICSS.2005.93

[PSA01] Joon S. Park, Ravi Sandhu, Gail-Joon Ahn, *Role-based access control on the web*, ACM Transactions on Information and System Security (TISSEC), February 2001, pp. 37-71, ISSN:1094-9224

[R[+]91] J. Rumbaugh, et all, *Object-Oriented Modeling and Design*, Prentice-Hall, 1991

[R+03] I. Ray, et all, *Using Parametrized UML to Specify and Compose Access Control Models,* Proceedings of the 6th IFIP Working Conference on Integrity & Internal Control în Info. Systems, 2003, ISBN:1-58113-872-5

[RK00] P. Ratnasingham, K. Kumar, *Trading partner trust in electronic commerce participation*, Proceedings of the 21st international concerence on Information systems, December 2000, pp 544-552, ISBN:ICIS2000-X

[S88] K. R. Sollins, *Cascaded Authentication*, Proceedings of the 1988 IEEE Symposium on Security and Privacy, IEEE Computer Society, 1988, pp. 156, ISBN: 0-8186-0850-1

[S92] R. Sandhu, *The typed access matrix model*, Proceedings of the Eleventh IEEE Symposium on Security and Privacy (SSP'92), pp. 122–136, 1992, Digital Object Identifier 10.1109/RISP.1992.213266

[S93] R. Sandhu, *Lattice-based access control models*, IEEE Computer, 26(11), pp. 9–19, 1993, ISSN: 0018-9162

[S94] M. Sloman, *Policy driven management for distributed systems*, Network and Systems Management, pp. 333–360, 1994, ISBN:1-59593-116-3

[S96] R. Sandhu, *Rationale for the RBAC96 family of access control models*, Symposium on Access Control Models and Technologies, 1996, Article no. 9, ISBN:0-89791-759-6

[S00] R. Sandhu, *Engineering authority and trust in cyberspace: the OM-AM and RBAC way*, Proceedings of the Fifth ACM Workshop on Role-Based Access Control (RBAC'00), pp. 111–119, 2000, ISBN:1-58113-259-X

[SA00] M. Shin, G. Ahn, *UML-Based Representation on Role-based Access Control*, Proceedings of the 9th IEEE workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000, Digital Object Identifier 10.1109/ENABL.2000.883728

[SBC+97] R. Sandhu, V. Bhamidipati, E. Coyne, S. Ganta, C. Youman, *The ARBAC97 model for role-based administration of roles: preliminary description and outline*, Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC'97), pp. 41–50, 1997, ISBN:0-89791-985-8

[SBM98] R. Sandhu, V. Bhamidipati, Q. Munawer, *The ARBAC97 model for role-based administration of roles*, ACM Transactions. Inf. Syst. Security, 1998, pp. 105-135, ISSN:1094-9224

[SC96] R. Sandhu, F. Chen, *Constraints for role-based access control*, 1st ACM Workshop on Role-based access control, 1996, ISBN:1-58113-681-1

[SC01] P. Samarati, S. Capitani, *Access control – policies, models and mechanisms*, Foundations of Security Analysis and Design, October 2001, pp 137-196, www.springerlink.com/index/80wrewj7j1a716wb.pdf

[SCH+96] R. Sandhu, E.Coyne, H. Feinstein, C. Youman, *Role-based access control models,* IEEE Computer, pp 38-47, 1996, csrc.nist.gov/rbac/sandhu96.pdf

[SH97] Hans Schuster, Petra Heinl, *A workflow data distribution strategy for scalable workflow management systems*, Proceedings of the 1997 ACM symposium on applied computing, April 1997, pp. 174-176, ISBN:0-89791-850-9

[SFK00] R. Sandhu, D. Ferraiaolo, R. Kuhn, *The NIST model for role-based access control: Towards a unified standard*, Proceedings of the Fifth ACM Workshop on Role-Based Access Control, July 2000, pp. 47-63, ISSN:1094-9224

[SM00] S.H. von Solms, I. van der Merwe, *The Mangement of Computer Security Profile,* 2000

[SL98] B. Schmid, M. Lindemann, *Elements of a Reference Model for Electronic Market*, 31st Annual Hawaii International Conference on System Sciences – Volume 4, January 1998, pp. 0193, Digital Object Identifier 10.1109/HICSS.1998.655275

[SM[1]02] A. Schaad, J. Moffett, *Delegation of obligations*, Third IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'02), pp. 25–35, 2002, Digital Object Identifier 10.1109/POLICY.2002.1011290

[SM[2]02] A. Schaad, Jonathan D. Moffett, *A lightweight approach to specification and analysis of role-based access control extensions,* Symposium on Access Control Models and Technologies, pp. 13-22, 2002, ISBN:1-58113-496-7

[SRJ01] P. Samarati, M. K. Reiter, S. Jajodia, *An authorization model for a public key management service*, ACM Transactions on Information and System Security (TISSEC), pp. 453–482, 2001, ISSN:1094-9224

[SZ83] R. Simon, R. Zurko, *Separation of duty în role based access control environments,* Proceedings of New Security, Paradigms Workshop, Sept. 1997, pp. 183, ISSN:0018-8670

[T05] B. Travica, *Virtual organizations and electronic commerce*, ACM SIGMIS, 2005, pp. 45-68, ISSN:0095-0033

[T97] R. K. Thomas, *Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments*, Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC'97), pp. 13–19, 1997, ISBN:0-89791-985-8

[TAP+05] W. Tolone, G. Ahn, T. Pai, S. Hong, *Access control in collaborative systems*, ACM Computing Surveys (CSUR), March 2005, pp. 91-100, ISSN:0360-0300

[TC97] Z. Tari, S. Chan, *A role based access control for intranet security,* IEEE Internet Computing, September/October 1997, pp. 24-34, Digital Object Identifier 10.1109/4236.623965

[TL04] M. V. Tripunitara, N. Li, *Comparing the expressive power of access control models,* Proceedings of the 11th ACM conference on Computer and communications security, October 2004, pp. 62-71, ISBN:1-58113-961-6

[TS97] R. K. Thomas, R. Sandhu, *Task-based Authorization Controls (TBAC): A family of models for active and enterprise-oriented authorization management*, Proceedings of the IFIP WG 11.3 Workshop on Database Security, pp. 166–181, August 1997, ISBN:0-412-82090-0

[TS98] G. Winfield Treese, L. C. Stewart, *Designing Systems for Internet Commerce*, Addison-Wesley Longman Publishing Co., 1998, ISBN: 0-202-57167-6

[TS02] A.S. Tanenbaum, M. Steen, *Distributed Systems – Principles and Paradigms*, Prentice Hall, NJ, September 2002

[TT02] Y. Tan, W. Thoen, *Formal Aspects of a Generic Model of Trust for Electronic Commerce*, Decision Support System, Volume 33, July 2002, pp. 233-246

[W96] L. Wang, *A framework for map recognition based on blackboard model*, 1996, http://citeseer.ist.psu.edu/11673.html

[WK05] Jacques Wainer, Akhil Kumar, *A fine-grained, controllable, user-to-user delegation method in RBAC,* Proceedings of the tenth ACM symposium on Access control models and technologies, June 2005, pp. 59-66, ISBN:1-59593-045-0

[WL98] T.Y.C. Woo, S.S. Lam, *Designing a Distributed Authorization Service*, Proceedings of IEEE INFOCOM'98, March 1998, ISBN: 0-7803-4383-2

[YD05] F. Yi, L. Diao, *E-commerce models, structure, mechanisms, globalization and strategy: Electronic market and operating intermediary*, Proceedings of the 7th international conference on Electronic commerce, ICEC, 2005, pp. 1-5, ISBN:1-59593-112-0

[YLS96] N. Yialelis, E. Lupu, M. Sloman, *Role based security for distributed object systems*, Proceedings of the Fifth Workshops on Enabling Technologies: Infrastructure for Collaborating Enterprises (WET-ICE'96), pp. 80–85, 1996, ISBN:0-8186-7445-8

[ZAC01] L. Zhang, Gail-Joon Ahn, Bei-Tseng Chu, *A rule-based framework for role based delegation*, Sixth ACM Symposium on Access Control Models and Technologies (SACMAT'01), pp. 153–162, 2001, ISSN:1094-9224

[ZK98] J. Leon Zhao, Akhil Kumar, *Data management issues for large scale, distributed workflow systems on the Internet,* ACM SIGMIS Database, Volume 29 Issue 4, 1998, pp. 22-32, ISSN:0095-0033

[ZM90] S. Zdonik, D. Maier, *Fundamentals of Object-Oriented Database*, Readings in Object-Oriented Database Systems, 1990, www.toa.com/pub/reading_list_article.txt