Universitatea Tehnica din Cluj-Napoca

Rezumat al tezei de doctorat

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

AUTOR

Ing. Mihaela Georgetta Ordean

CONDUCATOR STIINTIFIC

Prof.Dr.Ing. Dorian Gorgan

- iulie 2008 -

CUPRINS

1 Introducere 9
1.1 Obiectivele tezei 10
1.2 Structura tezei 12
2 Starea actuală a cercetării în controlul accesului 13
2.1 Primele modele în controlul accesului 13
2.1.1 MAC - Mandatory Access Control 13
2.1.2 DAC - Discretionary Access Control 14
2.2 Autorizarea în sisteme distribuite 14
2.3 Utilizarea rolurilor în sisteme distribuite 15
2.4 Implementări RBAC 16
2.5 Colective care au implementat modele RBAC 17
2.5.1 Colectivul de la George Mason University, SUA 17
2.5.2 Colectivul de la Imperial College, Londra 18
2.5.3 Colectivul de la Western Ontario University, Canada 18
2.5.4 Colectivul de la Universitatea din Roma, Italia 19
2.6 Standardizări în controlul accesului 19
3 Modelul SCAR-ACE pentru controlul accesului bazat pe roluri 21
3.1 Definiţii şi termeni utilizaţi 21
3.2 Componentele modelului 24
3.2.1 Nucleul 24
3.2.2 Ierarhiile de roluri 26
3.2.3 Relaţiile statice 28
3.2.4 Relaţiile dinamice 29
3.2.5 Extensiile 30
3.3 Modele conceptuale 32
3.3.1 Modele RBAC conceptuale 32
3.3.2 Modele SCAR-ACE conceptuale 33
3.3.3 Modelul de bază SCAR-ACE0 33
3.3.4 Definirea formală a modelului SCAR-ACE0 34
3.3.5 Ierarhiile de roluri şi abordarea acestora în SCAR-ACE1 35
3.3.6 Definirea formală a modelului SCAR-ACE1 35
3.3.7 Modelul constrângerilor în SCAR-ACE2 35
3.3.8 Definirea formală a modelului SCAR-ACE2 36
3.3.9 Modelul SCAR-ACE3 37
3.4 Analiza constrângerilor 38
3.4.1 Prerechizite 39
3.4.2 Separarea îndatoririlor 39
3.4.3 Cardinalitate 40
3.4.4 Constrângerile modelului 40
3.5 Privilegiile 42
3.5.1 Specificarea nivelelor de acordare a privilegiilor 43
3.5.2 Nivele de securitate în acordarea privilegiilor 43
3.5.3 Politica de acordare a privilegiilor 49
3.5.4 Algebra politicii de acordare a privilegiilor 50
3.6 Credenţiale 51
3.6.1 Componentele unui credenţial 51
3.6.2 Prelucrarea credenţialelor 52
3.7 Rolul interfeţelor grafice utilizator 54
3.8 Controlul accesului 56

3.8.1 Maşina de stare 56
3.8.2 Rolurile 57
3.8.3 Constrângeri ale rolurilor şi ale asignărilor utilizator 58
3.9 Definirea formala a modelului 58
3.9.1 Definiţii şi termeni 58
3.9.2 Definirea regulilor 61
3.9.3 Exemplificare 62
3.9.4 Consistenţa specificării modelului 66
3.10 Fazele specificării modelului 68
3.10.1 Faza de analiză statică 69
3.10.2 Faza de verificare / actualizare 70
3.10.3 Faza de planificare 72
3.10.4 Definirea maşinii de stare 73
3.10.5 Interfaţa utilizator grafică 74
3.10.6 Faza de rulare 75
4 Analiza aplicaţiilor de comerţ electronic şi implementarea modelului SCAR-ACE 77
4.1 Introducere 77
4.2 Lanţul valoric 77
4.2.1 Atragerea clienţilor 78
4.2.2 Interacţiunea cu clienţii 78
4.2.3 Acţionarea la instrucţiunile clientului 78
4.2.4 Reacţia la cererile clienţilor 79
4.3 Modele de afaceri 79
4.3.1 Similarităţi şi diferenţe la nivel de segment 80
4.3.2 Participanţi la sistem 80
4.4 Asigurarea intimităţii clientilor 82
4.4.1 Platforme pentru preferinţa intimităţii 82
4.4.2 Cookies 83
4.4.3 Tranzacţii electronice securizate 84
4.5 Arhitecturi funcţionale ale aplicaţiilor de comerţ electronic 84
4.5.1 Idei arhitecturale de bază 84
4.5.2 Modele de încredere 85
4.5.3 Arhitecturi sistem 85
4.6 Implementarea modelului SCAR-ACE 89
4.6.1 Prezentarea arhitecturii 89
4.6.2 Fluenţa tranziţiilor 91
4.6.3 Funcţionalităţi implementate 91
4.6.4 Platforma de comerţ electronic 91
4.6.5 Structura aplicaţiei. Nivelele prezentare şi de control al fluenţei 92
4.6.6 Structura datelor 97
4.6.7 Nivelul de logică şi nivelul de date 99
4.6.8 Nivelul de acces la date (DAL) 103
5 Rezultate experimentale 105
5.1 Teste de performanţă 105
5.1.1 Descrierea testelor 106
5.1.2 Interpretarea rezultatelor 107
5.2 Teste de scalabilitate 108
5.2.1 Interpretarea rezultatelor 109
5.3 Teste de răspuns la atac simulat 111

5.4 Teste de conformitate cu cerințele modelului SCAR-ACE 112
6 Concluzii şi dezvoltări ulterioare 115
6.1 Contribuția proprie..... 115
6.2 Dezvoltari ulterioare 116
Anexa 1 Bibliografie 119

1 Introducere

Controlul accesului are drept obiectiv protejarea resurselor față 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 tranziții între stări, acesta constă din setul de stări, regulile de tranziții între stări şi un set de proprietăți sau interogări.

În Internetul de astăzi există un număr din ce in ce mai mare de aplicații care necesită decizii de autorizare. Aceste aplicații includ (dar nu sunt limitate la) comerțul electronic, gestionarea şi partajarea resurselor distribuite, execuția şi descărcarea de cod de la distanță. Autorizarea acestor aplicații 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 entități, multe dintre acestea necunoscându-se una pe cealaltă şi, de multe ori, neexistând o autoritate centrală de încredere pentru toți actorii. Regulile conform cărora se realizează deciziile de acordare a controlului pot necesita modificări ale acestor aplicații, deciziile pentru permiterea accesului la o resursă specifică depinzând de tipul aplicației.

Un sistem de control al accesului este totodată şi o instanță a unei scheme de control al accesului si specifică tipurile de reguli relativ la tranzițiile între stări [TL04]. Un exemplu de model pentru controlul accesului este modelul bazat pe matrici de acces [GD75]. 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].

Obiectivele tezei

Teza se referă la un domeniu particular al autorizării şi propune modelul SCAR-ACE pentru controlul accesului bazat pe roluri în aplicații de comerț electronic.

Aplicațiile de comerț electronic devin din ce în ce mai complexe, impunand accesul la resurse heterogene a utilizatorilor cu roluri diferite. Controlul accesului în aplicațiile de comert electronic este un subiect important al cercetarii stiintifice actuale.

(1) **Direcția de cercetare**

În [B05] sunt definite direcțiile de cercetare pentru modelele, arhitecturile şi tehnologiile de control al accesului. Teza se subscrie unei direcții 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 soluții de implementare RBAC de control al accesului pe Web. Ambele soluții sunt modele bazate pe cookies, lasand astfel deschisa o breşă în securitate.

Lucrarea de față își 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 autorizați 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 fluxului de control a accesului în aplicațiile distribuite.

(3) **Soluția propusă**

Pentru a controla într-o aplicație distribuită fluența şi accesul la resurse se introduce noțiunea de rol ca un element intermediar între utilizator şi setul de permisiuni ale acestuia.

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 la operații şi resurse prin intermediul rolurilor în SCAR-ACE:

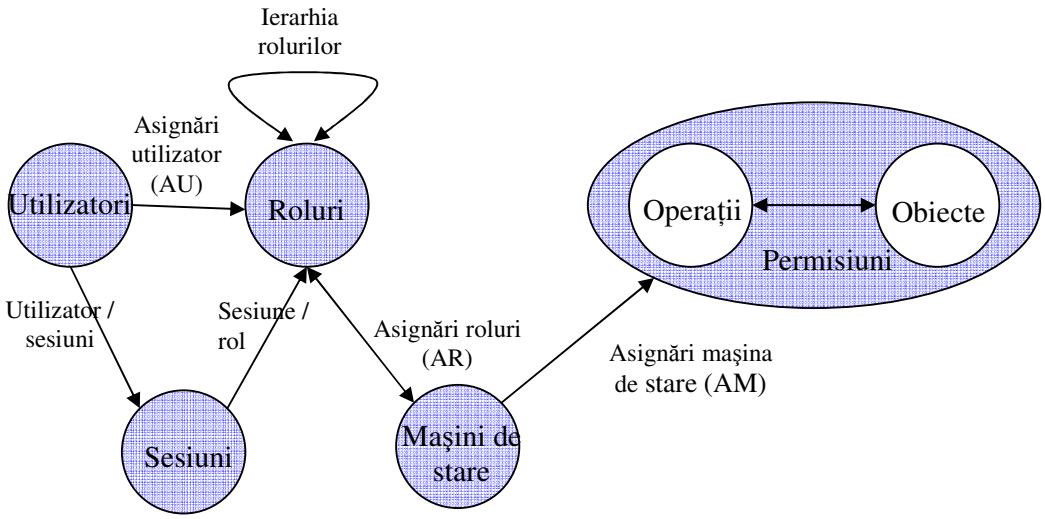

Figura 1. Componentele modelului SCAR-ACE

Structura tezei

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 conține descrierea modelului SCAR-ACE. Sunt definiți termenii care sunt utilizați în cadrul lucrării după care se detaliază componentele modelului SCAR-ACE în comparație cu modelul de referință RBAC, respectiv $SCAR\text{-}ACE_0$, $SCAR\text{-}ACE_1$, $SCAR\text{-}ACE_2$ şi $SCAR\text{-}ACE_3$. 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 aplicațiilor de comerț electronic, modelele de afaceri în comerțul pe Internet şi asigurarea intimității în cadrul aplicațiilor de comerț electronic. Se detaliaza implementarea prin care se validează modelul SCAR-ACE. Astfel este descrisă arhitectura aplicației de comerț electronic implementată si structura aplicației.

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

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

Anexa întâia conține bibliografia.

2 Starea actuală a cercetării 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).

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.

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

DAC îşi are originile în aria academică. El permite sau restricționează accesul la un obiect pe baza identității 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.

Aceste abordări au avut atât avantaje cât şi dezavantaje. Problemele apar de obicei la inserarea de utilizatori, la ştergerea anumitor utilizatori existenți sau la crearea şi ştergerea obiectelor.

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 propri.

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].

Controlul accesului bazat pe roluri se regaseste in literatura de specialitate sub denumirea de RBAC (Role Based Access Control).

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 aplicaţiile 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. 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. I-RBAC nu permite definirea rolurilor mutual exclusive. Spre deosebire de această implementare, SCAR-ACE este un model pentru aplicaţii pe Internet care acceptă roluri globale şi permite existenţa rolurilor mutual exclusive.

In [G99] se descrie un mecanism RBAC bazat pe servleţi Java. Se propune o arhitectură denumită JRBAC-WEB care se bazează pe execuţia cererilor HTTP 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 opţional abordarea permite păstrarea sesiunilor de lucru prin utilizarea cookies sau prin tehnici de rescriere a URL-ului pentru servleţi Java. SCAR-ACE nu utilizează servleţi Java (ci credenţiale) 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 conţine un mecanism specific de autentificare şi suportă ierarhii de roluri, constrângeri de cardinalitate şi

separarea dinamică şi statică a îndatoririlor. Implementarea modelului 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.

3 Modelul SCAR-ACE pentru controlul accesului bazat pe roluri

Componentele modelului

Componentele de bază ale modelului standard RBAC de control al accesului bazat pe roluri sunt [FSG01]: Nucleul RBAC, Ierarhiile RBAC, Relaţii statice, Relaţii dinamice si Extensii.

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

Nucleul modelului SCAR-ACE este similar nucleului RBAC. Nucleul RBAC include un set de şase elemente de bază şi anume: utilizatori, roluri, sesiuni, obiecte, operaţii şi permisiuni.

Nucleul modelului SCAR-ACE conţine urmatoarele elemente: utilizatori, roluri, sesiuni, permisiuni, operaţii şi obiecte, masina de stare şi relaţiile dintre aceste elemente.

Fiecare rol are ataşată o *maşină de stare* specifică doar 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 faţă de modelele existente.

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

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

- ierarhii generale. În acest caz există o ordine parţială între roluri şi include moştenirea multiplă a permisiunilor;
- ierarhii limitate. In acest caz se introduc restricţii în cadrul ierarhiei rolurilor.

Ierarhia SCAR-ACE este o ierarhie limitată de tip arbore. Rolurile în ierarhie sunt restricţionate la un singur descendent imediat şi la un singur ascendent imediat.

Relaţiile RBAC sunt relaţii de separare a îndatoririlor şi impun constrângeri asupra modelului. Relaţiile statice (SSD) sunt relatii de separare a îndatoririlor si sunt utilizate pentru a evita conflictul de interese în cadrul unei organizaţii (pentru a preveni un utilizator de a exceda nivelul de autorizare corespunzător poziţiei sale).

Modelul SCAR-ACE conţine relaţii statice şi implementarea acestora are loc prin utilizarea unei maşini de stare pentru fiecare rol.

Relaţiile dinamice (DSD), similar celor statice, limitează permisiunile utilizatorilor. Relaţiile DSD diferă de cele SSD prin contextul în care aceste limitări sunt impuse. Cerinţele 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).

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, recepționer al unei cereri sau distribuitor al unui produs. Pentru a realiza relațiile DSD, modelul SCAR-ACE utilizează tot mașini de stare.

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

Delegarea permite unui utilizator să imputernicească alți utilizatori cu o parte a drepturilor sale.

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

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 si a scalabilitatii acestora.

Modele conceptuale

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

Pentru a analiza diferite versiuni RBAC s-a definit o familie de patru modele conceptuale [SCH+96]. $RBAC_0$ este modelul de bază și este cerința minimă pentru un sistem de control al accesului bazat pe roluri. Modelele avansate $RBAC_1$ și $RBAC_2$ includ modelul $RBAC_0$. $RBAC_1$ introduce ierarhiile de roluri în timp ce $RBAC_2$ adaugă constrângeri. Modelul consolidat $RBAC_3$ include modelele $RBAC_1$ și $RBAC_2$ și, prin tranzitivitate, $RBAC_0$.

Modelul SCAR-ACE se referă la un $RBAC_3$ și, pentru realizarea SSD (Static Separation of Duties) și DSD (Dynamic Separation of Duties), impune constrângeri. Prin paralelism cu modelul RBAC, modelul SCAR-ACE este compus din SCAR-ACE_0, SCAR-ACE_1, SCAR-ACE_2 si SCAR-ACE_3. In continuare mă voi referi la toate nivelele conceptuale ale modelului SCAR-ACE și la specificul acestora [OG^{2}07].

Modelul de bază SCAR-ACE_0

Modelul de bază SCAR-ACE_0 constă din următoarele entități: utilizatori (U), roluri (R), masini de stare (M), permisiuni (P) și sesiuni (S).

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

- *U, R, P, S și M* (multimile de utilizatori, roluri, permisiuni, sesiuni și respectiv mașini de stare);
- *Relația utilizator – roluri:* $R_{UR} \subseteq U \times R$, o relație de asignare n:m pentru utilizatori-roluri
  $R_{UR} = \{ur \mid ur = (u, r), (\forall u \in U, \exists r \in R) \wedge (\forall r \in R, \exists u \in U)\}$;
- *Relația sesiune - utilizator*: $R_{SU} \subseteq S \rightarrow U$, o relație 1:1 care mapează fiecare sesiune unui singur utilizator (constant pentru timpul de viață 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
  $R_{MP} = \{mp \mid mp = (m, p), (\forall m \in M, \exists p \in P)$.

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]. Modelul SCAR-ACE$_1$ este definit pentru ierarhia de tip arbore.

In continuare introduc definiţia 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 parţială pe *R* denumită ierarhie de roluri sau relaţie de dominanţă a rolului, denotată $\geq$;
- Relaţiile sunt aceleaşi cu cele din SCAR-ACE$_0$.

Modelul constrângerilor în SCAR-ACE$_2$

SCAR-ACE$_2$ este nemodificat faţă de SCAR-ACE$_0$ cu excepţia 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 şi anume constrângeri statice şi constrângeri dinamice.

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 funcţie 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.

Modelul SCAR-ACE$_3$

SCAR-ACE$_3$ este caracterizat de ierarhii şi constrângeri şi combină modelele SCAR-ACE$_1$ şi SCAR-ACE$_2$. Sunt definite constrângeri asupra ierarhiilor de roluri, interacţiuni si constrângeri de cardinalitate.

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

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 si sunt tratate prin utilizarea limbajului UML.

Nivele de securitate în acordarea privilegiilor

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).

Nivelul 1 – diagrame use case

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*.

Introduc *Regula de acordare a privilegiilor în moştenirea rolurilor***:**

$\forall$ actori ac_m şi ac_n asociați 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).

Nivelul 2 – diagrame de clase

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.

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*.

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.

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

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.

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

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*).

> $\forall$ 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
>
> $\forall$ 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 puțin 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.*

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

- $c.DMP_{min} \leq \min \{c.m_x.PMS \mid$ 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 puțin 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 puțin egală cu maxima permisiunii metodelor componente.

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.

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".

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};*
b. *Adunarea* (cu sensul de *uniune*) este suportată în două concepte diferite:
   - pentru tupla *<r, {s}>* în cadrul relaţiei de moştenire între roluri;
   - în crearea credenţialelor.
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;
d. *Selecţia* poate fi aplicată pentru credenţial.

Credentiale

Un credenţial este emis de fiecare dată când un utilizator cere accesarea unei resurse sau a unui serviciu şi conţine următorii identificatori [O00]:

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

Controlul accesului

Maşina 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.

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.

Rolurile

Introduc clasificarea următoare asupra rolurilor în funcţie de tipul accesului:

- roluri publice;
- roluri private.

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. Cardinalitatea utilizatorilor asignaţi rolurilor private este, de cele mai multe ori, egală cu 1 sau cu o valoare ştiută şi finită.

Din punct de vedere al drepturilor detinute există:

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

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;

(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.

Definirea formală a modelului

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.

Definiţie: 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).

Definiţie: 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.

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;
- un set de *predicate de planificare* care realizează restricțiile;
- un set de *predicate de execuție* pentru execuția aplicației.

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
`Tranziție`	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
`Aparține`	dacă `aparține` (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 execuţie

Predicat	Definire
$\texttt{exec}_u$	Dacă $\texttt{exec}_u(u_i, a_j, s_k, e_l)$ este adevărat, atunci acţiunea a_j este executată de utilizatorul u_i prin comutarea de la starea s_k la apariţia evenimentului e_l
$\texttt{exec}_r$	Dacă $\texttt{exec}_r(r_i, a_j, s_k, e_l)$ este adevărat, atunci acţiunea a_j este executată de utilizatorul aparţinând rolului r_i prin comutarea de la starea s_k la apariţia evenimentului e_l
$\texttt{exec}_t$	Dacă $\texttt{exec}_t(r_i, t_j, s_k, e_l, p_n)$ este adevărat, atunci tranziţia t_j este executată de utilizatorul aparţinând rolului r_i prin comutarea de la starea s_k la apariţia evenimentului e_l şi cu parametrii p_n.
abort	Dacă abort (a_j, s_k, e_l, p_n) este adevărat atunci acţiunea a_i nu a putut fi executată cu succes la comutarea din starea s_k, la apariţia 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 acţiunea a_i a putut fi executată cu succes la trecerea din starea s_k la apariţia evenimentului e_l cu parametrii p_n.

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

Consistenţa specificării modelului

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

In primul rând dacă predicatul panic aparţine modelului aceasta implică existenţa cel puţin a unei condiţii care nu poate fi satisfăcută. Din acest motiv, prima condiţie de consistenţă a specificaţiei impune ca predicatul panic să nu aparţină modelului.

O altă etapă trebuie să implice analizarea modelului astfel încât să nu conţină informaţii contradictorii. De exemplu, dacă ambele predicate $\texttt{must\_execute}_t$ (t_m, s_i, s_j) şi $\texttt{cannot\_do}_t$ (t_m, s_i, s_j) aparţin modelului, implică constrângeri în cadrul acestuia care sunt inconsistente deoarece sunt contradictorii.

O constrângere importantă este cea prin care să existe întotdeauna cel puţin un utilizator aparţinând unui rol permis care să poată realiza o acţiune dată.

Pentru verificarea inexistenţei informaţiilor contradictorii trebuie realizată computaţia tranziţiilor care trebuie să fie permise/nepermise pentru un rol. Această verificare implică baleierea bazei de date care conţine maşinile de stare şi verificarea tranziţiilor introduse. Astfel se deosebesc următoarele seturi:

- *Tranziții_Nepermise (t_m)* = ∪ { t_m | cannot_do$_t$ (t_m, s_i, s_j) ∈ M_M}. *Tranziții_Nepermise* reprezintă setul de tranziții care nu pot fi executate în M_M conform mașinilor de stare;
- *Tranziții_Permise (t_m)* = ∪ { t_m | must_do$_t$ (t_m, s_i, s_j) ∈ M_M}. *Tranziții_Permise* reprezintă setul de tranziții care pot fi executate în M_M conform mașinilor de stare;
- *Roluri_Nepermise (r_i)* = ∪ { r_i | cannot_do$_r$ (r_i, a_j) ∈ 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)* = ∪ { r_i | must_do$_r$ (r_i, a_j) ∈ 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)* = ∪ { u_i | u_i aparține rolului r_i ∧ cannot_do$_r$ (r_i, a_j) ∈ 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)* = ∪ { u_i | u_i aparține rolului r_i ∧ must_do$_r$ (r_i, a_j) ∈ M_M }. *Utilizatori_Permiși* reprezintă setul de utilizatori aparținând unor roluri care pot executa acțiunea a_j în M_M conform mașinilor de stare.

Definiție (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 ∉ CM(M);
- ∀ t_m al M_M:
  - Dacă *Tranziții_Permise* (t_m) ≠ ∅, atunci:
    Tranziții_Permise (t_m) ∩ *Tranziții_Nepermise* (t_m) = ∅;
    Tranziții_Permise (t_m) ⊆ { t_m | t_m ∈ TT};
- ∀ r_i al M_M:
  - Dacă *Roluri_Permise* (r_i) ≠ ∅, atunci:
    Roluri_Permise (r_i) ∩ *Roluri_Nepermise* (r_i) = ∅;
    Roluri_Permise (r_i) ⊆ { r_i | r_i ∈ RT};
- ∀ u_i al M_M:
  - Dacă *Utilizatori_Permiși* (u_i) ≠ ∅, atunci:
    Utilizatori_Permiși (u_i) ∩ *Utilizatori_Nepermiși* (u_i) = ∅;
    Utilizatori_Permiși (u_i) ⊆ { u_i | u_i ∈ UT};
- ∀ u_i, r_i ai M_M:
  - Dacă *Utilizatori_Permiși* (u_i) ≠ ∅ și *Roluri_Permise* (r_i) ≠ ∅, atunci:
    ∃ r_j ∈ { RT_i \ *Roluri_Nepermise (r_i)*} pentru care
    ∃ u_j ∈ { RT_i \ *Utilizatori_Nepermiși (u_i)*}.

Fazele specificării modelului

In această secțiune prezint pașii în crearea modelului SCAR-ACE, fiecare pas fiind realizat de către o anumită componentă a sistemului de autorizare.

Primul pas, denumit *analiză statică* determină partea statică a constrângerilor modelului și verifică consistența acestora. Dacă această fază se finalizează cu succes se trece la următorul pas de *verificare/actualizare* prin care se realizează asignarea de

acţiuni la roluri, în funcţie de faza de analiză statică. Faza de *planificare* primeşte la intrare constrângerile şi generează la iesire maşina de stare pentru fiecare rol, prin ataşarea stărilor, tranziţiilor şi a evenimentelor. Dacă această fază se finalizează cu succes se ataşează interfeţei 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.

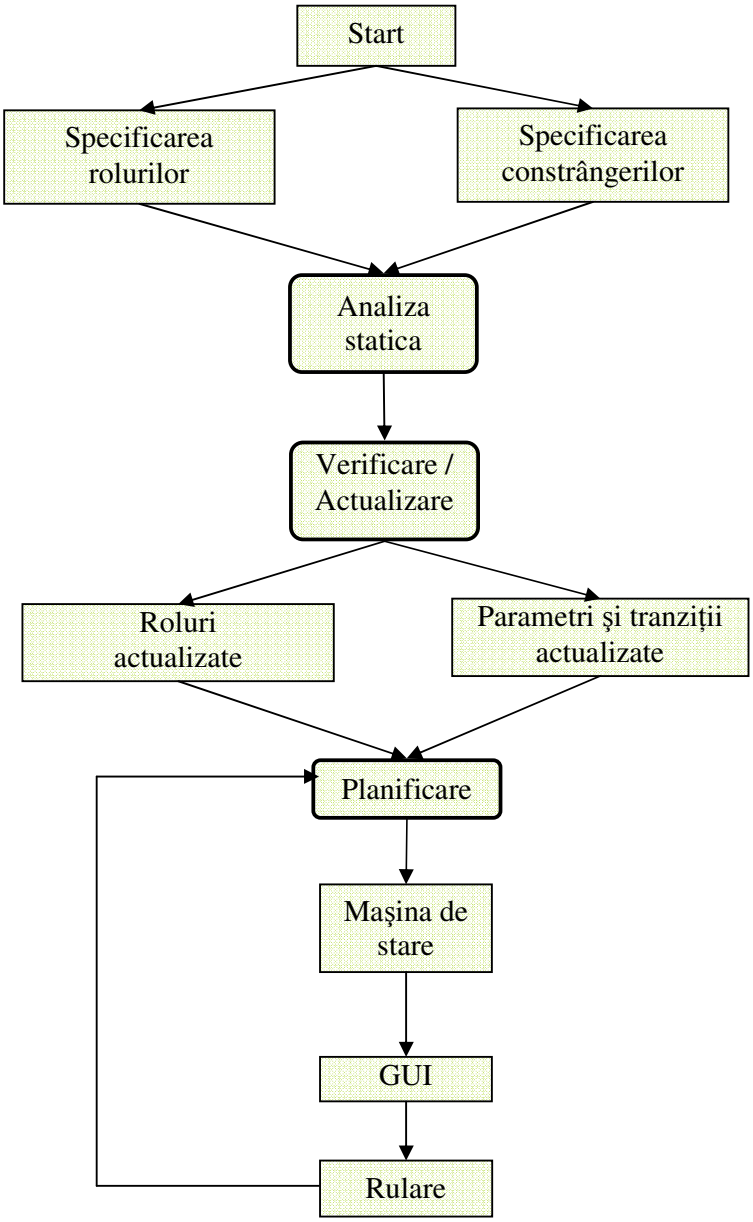

Figura 2. Fazele analizei de specificare a modelului

Faza de analiză statică

In faza de analiză statică se verifică consistenţa părţii statice a modelului.

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.

Faza de verificare / actualizare

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

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.

Faza de planificare

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

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;

3) Secvenţa de tranziţii;

4) Acţiunile specifice fiecărei tranziţii.

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 secvenţa de tranziţii.

Algoritmul maşinii de stare:

INPUT:

1) setul de roluri;

2) secvenţa de tranziţii;

3) predicate planificare.

OUTPUT:

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

Interfaţa utilizator grafică

După realizarea fazei anterioare componenţa maşinii de stare este adusă într-o stare funcţională.

Algoritmul fazei de realizare a interfeţei grafice:
INPUT:
1) G = (N, A) pentru fiecare rol.
OUTPUT:
Interfaţa grafică cu utilizatorul.

Faza de rulare

Faza de rulare este realizată prin executarea acţiunilor specifice tranziţiilor componente ale maşinii de stare.

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].

Introducere

Termenul de *comerț electronic pe Internet* desemnează utilizarea Internetului pentru vânzarea şi cumpărarea de bunuri şi servicii, incluzând suport după vânzare [TS98]. 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.

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. 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.

Modele de afaceri

In funcție de cerințele sistemului şi a opțiunilor 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;
- Comerțul informațional

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

Sistemele de comerț pe Internet au mai multe tipuri de participanți în funcție de rolul şi scopul acestora [PS05], [EE01].

Cumpărătorii cer un set de operații; designer-ii de cataloage, reprezentanții serviciilor client şi operatorii de sistem fiecare are setul lui de cerințe. 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 cerințelor unei aplicații de orice mărime, permițând unei afaceri de mică întindere să crească fără a reconsidera rolurile persoanelor implicate în fiecare etapă.

Asigurarea intimității clientilor

O tehnologie dezbătută în asigurarea intimității utilizatorilor sunt cookies [L05]. Acestea sunt o inovație adăugată browserelor de Web în 1995.

Există anumite avantaje dar şi dezavantaje a cookies-urilor. Avantajele ar fi următoarele [PS00]: determinarea sesiunilor, recunoaşterea automată a unui utilizator, profiluri ale clienților.

Cookies-urile însă prezintă şi dezavantaje: 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.

Arhitecturi funcţionale ale aplicaţiilor de comerţ electronic

Definiţie: 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.

Se prezintă trei arhitecturi posibile şi arhitectura modelului SCAR-ACE. 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.

Există şi alte abordări ale comerţului electronic.

Implementarea modelului SCAR-ACE

Arhitectura se adresează aplicaţiilor de B2B [OG05] în care sunt implicate diverse grupe de actori. Acest design este bazat pe modelul afacerii aratat mai jos:

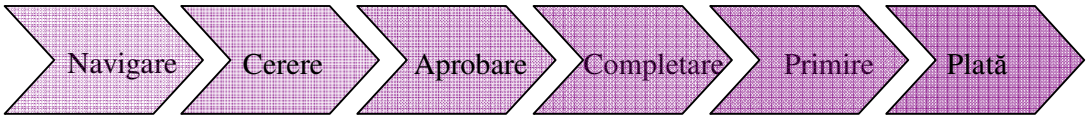

Figura 3. Procesul de achiziţie

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.

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.

Platforma a fost programată utilizând tehnologia Microsoft COM/DCOM. 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.

Orice decizie în designul atât a programării bazate pe obiecte pentru implementarea modelului SCAR-ACE 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.

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).

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.

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.

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.

Nivelul de logică şi nivelul de 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.

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.

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 şi, la sosirea cererii, credențialul conține altă valoare (starea actualizată a clientului).

Componenta de distribuție

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

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

5 Rezultate experimentale

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

Testele funcționale realizate sunt:

- teste de performanță;

- teste de scalabilitate.

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.

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.

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

Testele au constat din lansarea în execuţie a 300 de taskuri care să ruleze în paralel. Testele care s-au făcut au urmărit următorii trei timpi:

- timpul de execuţie ;
- timpul mediu al execuţiei;
- timpul total al execuţiei.

Rezultatele obţinute sunt cele din figurile de mai jos (figura 4 si figura 5).

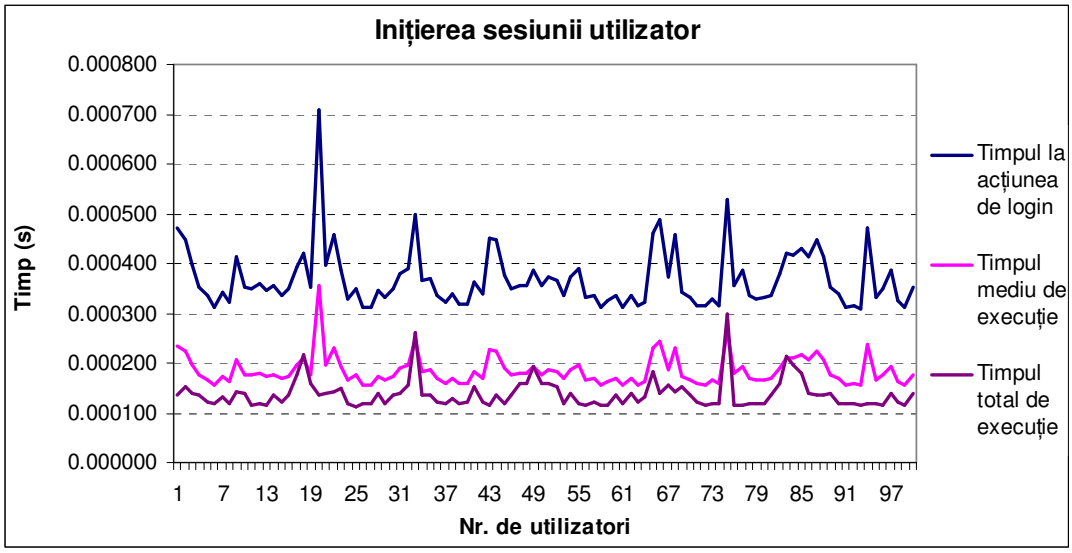

Figura 4. Timpii la autentificarea utilizatorilor

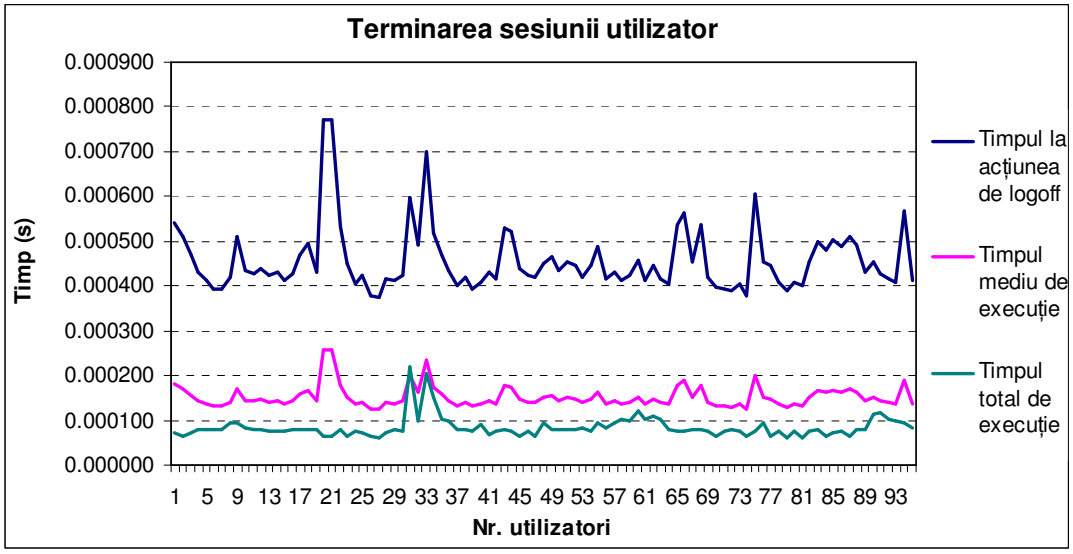

Figura 5. Timpii la terminarea sesiunilor utilizator

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 4 si 5 se observă că la autentificare si respectiv la terminarea sesiunii utilizator, cei trei timpi studiați nu au aceeaşi valoare.

In graficul din figura 4 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 4 (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 acţiuni paralele de executat.

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

Timpii din figura 5 urmăresc evoluţia celor din figura 4 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 execuţie la încărcarea /descărcarea acestora în / din memorie. Toţi timpii de execuţie subsecvenţi nu cresc. Timpul total de execuţie nu este puternic influenţat de lucrul cu maşini de stare.

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.

Aplicatia SCAR-ACE a fost scalata pe două servere şi a fost testata funcţional pentru 2 cazuri ilustrate in teza (figura 6).

Aplicaţiile care au fost selectate pentru testarea timpilor obtinuţi prin scalare sunt:

- Prima aplicaţie implementează modelul SCAR-ACE si a fost scalata;
- A doua aplicaţie implementează arhitectura cu formuri de comenzi si nu a fost scalata.

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

Primul test implementează următoarea funcţionalitate: selectarea celei mai ieftine oferte care respectă anumite criterii funcţionale.

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 acţiunea de login. Au fost comparate cele două variante ale implementării modelului SCAR-ACE (cea scalată si cea nescalată) în funcţie de numărul de utilizatori care au participat in sistem.

Interpretarea rezultatelor

Pentru ipoteza 1: Timpii de răspuns ai primului test de scalabilitate sunt ilustrati in figura 6. S-au luat in considerare 100 de fire de executie paralele. Se poate observa o diferenta intre timpii de raspuns pentru cele doua situatii: prima aplicaţie (modelul SCAR-ACE scalat) şi cea de a doua aplicaţie (arhitectura cu formuri de comenzi

nescalata), şi anume un timp de execuţie mai mic pentru cazul aplicaţiei care implementează modelul SCAR-ACE.

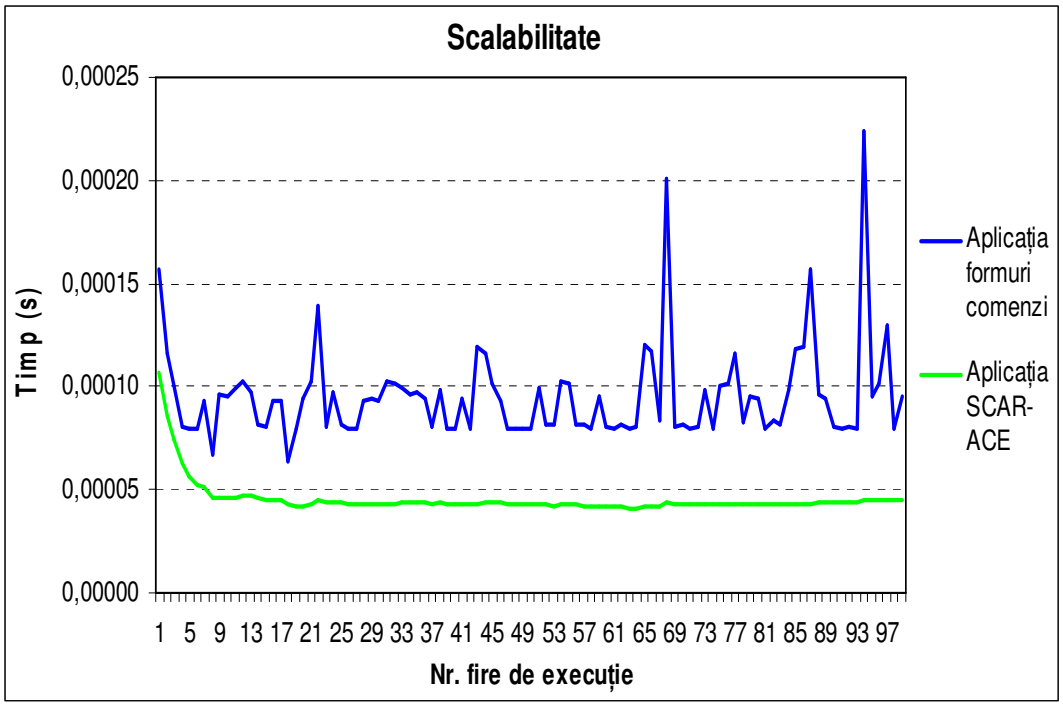

Figura 6. Rezultatele testului 1 de scalabilitate

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

Pentru ipoteza 2:

Evoluţia timpilor de răspuns între cele două variante în funcţie de numărul de utilizatori simultani este ilustrată în figura 7. Se observă o diferenţă din ce în ce mai mare a timpilor de execuţie odată cu creşterea numărului de utilizatori.

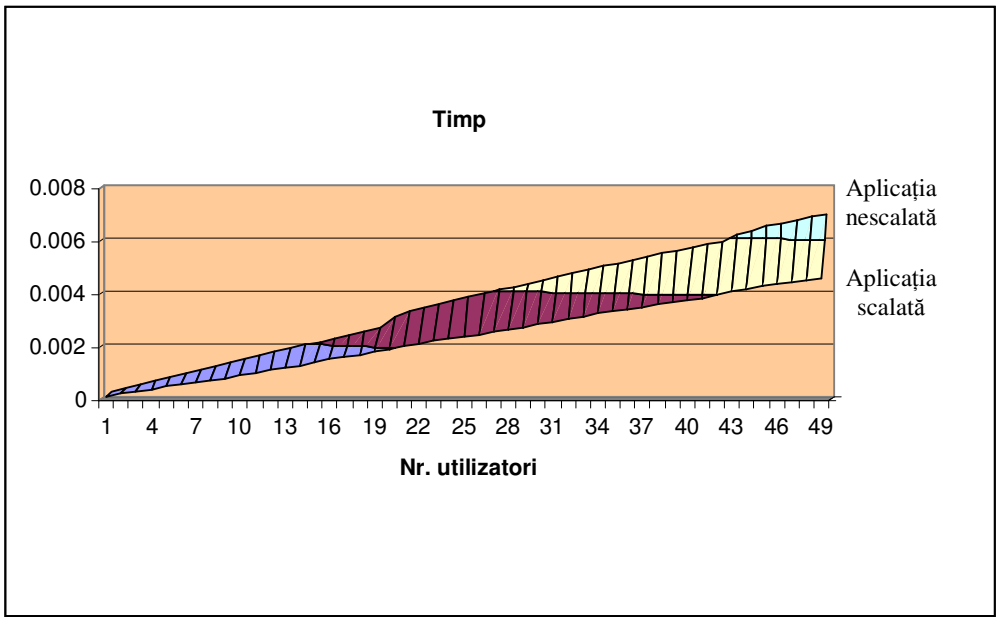

Figura 7. Variatia timpilor totali de executile in aplicatia scalata

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

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 excepții şi afişarea unei pagini de eroare. Presupunând că în mod fraudulos s-ar fura un credențial în timpul accesului unui utilizator la informațiile proprii autorizate, sistemul verifică în mod permanent informațiile 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 aplicației şi realizarea unui log off forțat.

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

Concluzia 1: Credențialele nu se pot obține din aplicație.

Reconstituirea unui credențial:

Dacă un atacator reuşeşte totuşi să obțină în mod fraudulos un credențial 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 interdicție: credențialul nu mai e valabil după închiderea unei sesiuni utilizator şi aceasta deoarece identificatorul unui utilizator expiră după închiderea sesiunii şi la orice acces inițiat 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 credențiale şi s-au simulat atacuri însă nu s-a reuşit penetrarea sistemului nici pentru stări autorizate şi nici pentru cele neautorizate.

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

Teste de conformitate cu cerințele modelului SCAR-ACE

Modelul SCAR-ACE impune anumite cerințe 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: aplicația realizează separarea îndatoririlor.

Testarea s-a realizat cu utilitarul Silktest şi a demonstrat conformitatea funcționării cu cerințele modelului SCAR-ACE prin obținerea de rezultate corecte. S-au efectuat acțiuni în care au fost activi 100 de utilizatori şi au fost lansate simultam 406 fire de execuție. Pentru verificarea separării îndatoririlor s-a încercat forțarea atribuțiilor rolului de vizitator prin încercarea de accesare a unor informații private specifice rolului de cumpărător:

- informații de identitate a unui cumparator – încercare soldată cu eşec;
- informații relativ la comenzile realizate de un cumparator - încercare soldată cu eşec;
- informații de plată a unei comenzi - încercare soldată cu eşec;
- informații 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

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.

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

7 Bibliografie selectivă

[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

[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

[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

[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

[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

[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

[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/

[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

[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

[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

[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

[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

[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

[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

[L05] J. Linn, *Technology and Web User Data Privacy: A Survey of Riskjs and Countermeasures*, IEEE Security and Privacy, 2005, pp. 52-58

[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

[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

[NO94] M. Nyanchama, S. Osborn, *Access Rights Administration în Role-Based Security Systems,* Database Security VIII: Status and Prospects, 1994, pp. 37-56

[O00] M. Ordean, *An application framework for e-commerce platforms*, e-Commerce Simpozion, Bilateral cooperation Romania-Croatia, Cluj-Napoca, oct. 2000

[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

[OG107] 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

[OG207] 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

[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

[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

[S93] R. Sandhu, *Lattice-based access control models*, IEEE Computer, 26(11), pp. 9–19, 1993, ISSN: 0018-9162

[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

[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

[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

[T05] B. Travica, *Virtual organizations and electronic commerce*, ACM SIGMIS, 2005, pp. 45-68, ISSN:0095-0033

[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

[TS98] G. Winfield Treese, L. C. Stewart, *Designing Systems for Internet Commerce*, Addison-Wesley Longman Publishing Co., 1998, ISBN: 0-202-57167-6

[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

[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

Articol indexat ISI

1. Floricica Parauan, Mihaela Ordean, Andrei Diamandi, Cloud Mask Algorithm, SYNASC 2006, IEEE Computer Society, 26-29 September 2006, Timisoara, Romania, ISBN 0-7695-2740-X, ISBN 978-0-7695-2740-6, pp. 259-266

Articole

2. Mihaela Ordean, Dorian 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

3. Mihaela Ordean, Dorian 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

4. Mihaela Ordean, Floricica Parauan, Clouds Mask Algorithm Architecture, 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

5. Floricica Parauan, Mihaela Ordean, Andrei Diamandi, Cloud Mask Algorithm, SYNASC 2006, IEEE Computer Society, 26-29 September 2006, Timisoara, Romania, ISBN 0-7695-2740-X, ISBN 978-0-7695-2740-6, pp. 259-266

6. Ioan Alexandru Dragan, Mihaela Ordean, Tibor Gal, Premise de securitate in MEDIOGRID, ISBN 973-713-092-8, pag. 126-125

7. 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

8. M. Ordean, C. Melenti, D. Gorgan: MEDIOGRID System in Meteorological and Environment Applications, IPSI – 2005 Amalfi, Italy, February 17 – 20, 2005, pp: 203-207, ISBN: 86-7466-117-3

9. Ordean M., Gorgan D., Ignat I.: 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

10. C. Melenti, M. Ordean, D.Gorgan, S. Oancea: Grid computing-based Satellite
    Image Processing for Fire Detection, International Conference on Advances in
    the Internet, Processing, Systems and Interdisciplinary Research, IPSI 2004, 11-14 Dec. 2004, Prague, Cehia, pp.101-107, ISBN: 86-7466-117-3

11. M. Ordean, D. Gorgan: Distributed Active Object Model, Proceedings of the IEEE - INES 2004 Conference, 19-21 Sept. 2004, Cluj-Napoca, Romania, pp. 557-560, ISBN 973-662-120-9

12. C. Melenti, M. Ordean, Electronic book, electronic library – as the image destiny, "Interferente" Art and Design University, april 2002

13. T.Le, M.Stankovics, E. Schwartzkopf, M. Ordean, Report on Observations and Preliminary Analysis in the Domain of Museums and Art Galleries, ETH Zurich, September, 2002

14. Mihaela Ordean, An application framework for e-commerce platforms, e-Commerce Simpozion, Cluj-Napoca, Colaborare bilaterala Romania-Serbia, oct. 2000

15. M.Ordean, G. Karetsos – Advanced Brokerage System, Architecture guideline for the implementation of Broker V.1, National Technical University of Athens, 1997.

16. G. Cimoca, M. Ordean, O. Pop, T. Trif - Applying new concepts of chaos theory in a new statistical-geometrical technology elaboration and the localization of gas resources undiscovered potential in a geological-tectonical area. Scientific colocvium C.C.I.T. Gaz Metan Medias, 3-4 Dec. 1996 (printed volume).

Carti

Mihaela Ordean – Programarea Aplicatiilor C sub Windows, Editura MicroInformatica, 1996, ISBN 973-9215-19-X

Proiecte

1. **Proiect INOVARE:** Sinteza text-vorbire, 2008, coordonator

2. **Proiect IMPACT:** Dotarea unui laborator la standarde europene, 2007, coordonator

3. **Proiect Leonardo Mobilitati**: Training of project managers in order to attain the quality of work according to European standards – P-LEAD, 2006, coordonator – premiu pentru buna practica acordat de Agentia Nationala Leonardo Da Vinci

4. **Proiect CEEX:** MEDIOGRID - Prelucrarea grafica paralela si distribuita pe structura grid a datelor geografice si de mediu, Universitatea Tehnica Cluj-Napoca, Catedra de Calculatoare, 2005 -2008, Coordonator partener iQuest Technologies SRL

5. **Proiect CNCSIS:** Procesarea locala si distribuita a imaginilor si aplicarea multimedia in medicina, Universitatea Tehnica Cluj-Napoca, Departamentul de Electronica Aplicata, 2004, colaborator

6. **Proiect PHARE:** Di@log – Introducerea tehnicilor performante in socio–dinamica – audit social si chestionare de feed-back ale angajatilor, regiunea Muntenia-Sud, iQuest Technologies, 2002-2003, Administrator