UDC: 007:004.056.52 INFO M: str. 34-41

MODELOVANJE KONTROLE PRISTUPA
ACCESS CONTROL MODELING

Branislav Trninić - FTN Novi Sad, Goran Sladić - FTN Novi Sad, Gordana Milosavljević - FTN Novi Sad

REZIME: Sa sve većom primenom informacionih tehnologija u različitim sferama, oblast kontrole pristupa postaje jedan od jako važnih faktora u razvoju softvera. Rastuću kompleksnost nije moguće rješavati ad-hoc pristupima već je potrebno uvesti metodologije za modelovanje sistema kontrole pristupa i modelovanje zadatih prava. U okviru rada prikazani su različiti pristupi problemu modelovanja kontrole prustupa. Predstavljeni načini modelovanja svrstani su u 4 grupe: formalno-jezički bazirani, XML bazirani, UML bazirani i DSL bazirani.

KLJUČNE REČI: kontrola pristupa, MDS, XML, UML, DSL

ABSTRACT: Access control is growing concern in software industry. With expansion of computer based bussines systems and widespread communication a software engineer is facing tremendous security requrements. Ever increasing complexity cannot be addressed using ad-hoc approaches. It is necessary to develop methodologies for modeling access control systems and defining access control policies. This paper presents different methodologies for access control modeling, grouped into four approaches: formal language based, XML based, UML based and DSL based.

KEY WORDS: access control, MDS, XML, UML, DSL

1. UVOD

Kontrola pristupa (*Access Control,* AC), odnosno autorizacija, se odnosi na proces kontrolisanja ko može da pristupi kojem resursu i na koji način može dati resurs da koristi. Kontrola pristupa zahtjeva definisanje prava pristupa datih u vidu autorizacijskih pravila i ograničenja. Ona spada u preventivne mjere zaštite računarskih sistema i predstavlja samo jedan aspekt složene i obimne oblasti bezbjednosti računarskih sistema, ali je i jedna od njenih najvažnijih djelova.

Danas, izazovi u oblasti sigurnosti kompleksnih i distribuiranih sistema ne leže samo u kontroli pristupa jednom sloju, već u cjelokupnoj zaštiti sistema. To podrazumjeva jasno definisanje prava pristupa na različitim nivoima apstrakcije, kao i objedinjen i ispravan menadžment i sprovođenje definisanih prava pristupa na svim relevantnim tačkama unutar sistema. Za implemenaciju sistema kontole pristupa potrebno je definisati sledeća tri koncepta [1]:

- **Prava pristupa** (*Security policy*): definišu pravila visokog nivoa u skladu sa potrebama kontrole pristupa
- **Model** predstavlja *formalnu* reprezentaciju pravila u kontroli pristupa i njihovo značenje. Formalizacija dozvoljava analizu sigurnosnih karakteristika sistema kontrole pristupa koji se dizajnira.
- **Mehanizam** definiše softverske i hardverske funkcije niskog nivoa potrebne za sprovođenje kontrole pristupa u skladu sa pravima pristupa koja su definisana modelom.

Mehanizam kontrole pristupa sačinjavaju [2]:

Autentifikacija (i identifikacija) podrazumjeva proces utvrđivanja identiteta subjekta koji želi pristup. Korisnik se identifikuje na osnovu nečega sto zna (*password*), što ima (*smartcard*) ili što sam korisnik jeste (otisak prista). Uglavnom se implementira kao jednokratan postupak u okviru jedne sesije (instance interakcije). Autentifikacija, sama po sebi, ne definiše koje operacije korisnik može da radi ili ne radi u sistemu.

Autorizacija je proces odlučivanja da li upućeni zahtjev subjekta za pristup resursu može biti odobren. Svaki novi zahtjev u toku korisničke sesije treba biti autorizovan, tako da se autorizacija stalno ponavlja i može biti zahtjevna sa stanovišta performansi. Naravno, preduslov za autorizaciju je prethodna identifikacija i autentifikacija korisnika.

Audit podrazumjeva praćenje aktivnosti subjekta na sistemu i drugih vezanih događaja radi kasnijeg korišćenja za pronalaženje griješaka ili istraživanje bezbjednosnih incidenata, redosljeda izvršenja kao i za identifikaciju zagušenja. Tip događaja može biti početak i kraj sesije, uspješna ili neuspješna autentifikacija, uspješna ili neuspješna autorizacija i sl. Osim ispravne implementacije i sprovođenja kontrole pristupa, važna osobina nekog sistema za kontrolu pristupa je način zadavanja prava pristupa. Sistem treba da administratoru obezbjedi efikasan način za inicijalno zadavanje prava pristupa, kasnije ažuriranje i validaciju.

U svim modelima kontrole pristupa postoje entiteti *subjekta* i *objekta*. Subjekat predstavljaju entiteti računarskog sistema koji obavljaju operacije nad objektima. *Objekti* su resursi računarskog sistema prema kojima treba ograničiti pristup. Subjekat može biti čovjek, ali i autonomni računarski agent. Čovjek - korisnik može uticati na sistem samo kroz softverske entitete nad kojima ima kontrolu. Česta praksa je da jedan subjekat odgovara jednom korisniku, mada jedan isti korisnik može da djeluje u sistemu kroz više različitih subjekata. U distribuiranim sistemima, subjekat je bilo koji agent, (klijent, aplikacija) koji zahtjeva pristup resursu putem servisa.

U kontroli pristupa, izazov modelovanja leži u jasnoj i korektnoj specifikaciji prava pristupa i mehanizma za njihovo sprovođenje. Način administracije korisnika, njihovog organizovanja u grupe ili uloge, privilegija, dozvola i relacija

između njih predstavlja značajan problem zbog potencijalno ogromnog broja entiteta. Na primjeru Dresden Banke, studija [3] je identifikovala preko 1300 korisničkih uloga i 40 000 korisnika. Kao moguća solucija za rješenje problema ovakvog tipa predložena je *Model Driven Security* (MDS) metodologija [4]. MDS predstavlja specijalizaciju *Model Driven Engineering* (MDE) metodologije [5] za oblast sigurnosti sistema i kontrole pristupa.

Ostatak rada je organizovan na sledeći način: u narednom poglavlju su date osnovne apstrakcije i modeli kontrole pristupa. U poglavlju 3. dat je pregled tehnika modelom-vođenog inženjeringa (MDE). Poglavlje 4. uvodi koncepte modelom-vođene inženjeringa bezbjednosti (MDS). U poglavlju 5. su predstavljene i analizirane različite metode za modelovanje prava pristupa, i to: formalno-jezički, XML bazirani, UML bazirani i DSL bazirani.

2. MODELI KONTROLE PRISTUPA

Odnose između subjekta i objekata možemo prikazati kroz matricu kontrole pristupa.

Matrica kontrole pristupa (*Access Control Matrix*, ACM) je fundamentalna apstrakcija i model kontrole pristupa u računarskim sistemima, i služi za prikazivanje odnosa između subjekata i objekata [6]. ACM opisuje koja prava ima svaki subjekat nad svakim objektom. Pošto nije definisana granularnost, matrica za kontrolu pristupa može biti korišćena kao statički model privilegija u bilo kojem sistemu kontrole pristupa, ili kao projekcija trenutnog stanja dinamičkog sistema. S obzirom da ne modeluje pravila i načine kako se neka dozvola može mijenjati, predstavlja djelimičan opis prava pristupa i u praksi se rijetko primjenjuje u doslovnom obliku. Bukvalna realizacija matrice kao dvodimenzionalog niza, u kojem svaki element ima listu dozvola, takođe je nepraktična zbog potencijalno ogromnih memorijskih zahtjeva.

Discretionary access control - DAC

U DAC-u [7] korisnik ima potpunu kontrolu nad dokumentima koje posjeduje, i u mogućnosti je da postavlja dozvole nad njima i za druge korisnike. Najbolji primjer DAC-a je kontrola pristupa fajlovima u Unix fajl sistemu. Danas se koristi u većini operativnih sistema za kontrolu pristupa fajlovima. U praksi se DAC implementira korišćenjem lista kontrole pristupa (*Access Control Lists*, ACL), gdje je svakom objektu pridružena lista korisnika i korisničkih grupa sa dozvolama za *read*, *write* i *execute*.

Mandatory Access Control – MAC

MAC [7] kontrola pristupa je pravljena za potrebe Američke vlade i najstriktnija je od svih prethodno navdenih. Samo administrator može da postavlja dozvole. MAC koristi sigurnosne oznake svim resursima prisutnim na sistemu. Oznaka sadrži dva podatka: oznaku klasifikacije (*secret, top secret, confidental*) i oznaku kategorije na koju se klasifikacija odnosi. Analogno, svaki korisnički nalog ima podatke o klasifikaciji i kategoriji. Za uspjesan pristup korisnik mora imati istu ili veću klasifikaciju i pripadati kategoriji kojoj pripada i resurs. Iako jedan od najsigurnijih, MAC je nepraktičan i tezak za odrzavanje.

DAC i MAC modeli kontrole pristupa su prvi implementirani u praksi. Vremenom su se pokazali kao nepraktični za primjenu u većim i dinamičnim poslovnim okruženjima.

Role-Based Access Control - RBAC

RBAC [8] je danas jedan od najzastupljenijih modela kontrole pristupa, naročito u poslovnim sistemima. Glavna prednost ovog modela je uvođenje pojma *korisničke uloge*. *Korisnička uloga* je dodatni sloj indirekcije između korisnika i privilegija. Na taj način je moguće grupisati korisnike i privilegije u logičke cjeline na većem nivou apstrakcije. Time je definisanje prava pristuipa višestruko olakšano. Osnovni entiteti definisani RBAC modelom su *skup korisnika, korisnička sesija, skup uloga i skup privilegija*. U okviru RBAC-a, privilegija je definisana apstrakntno. U praksi se često se implementira kao uređen par (*objekat, operacija*).

Najveća prednost RBAC-a je u tome što je uvođenjem pojama uloge približio koncepte iz kontrole pristupa ka realnosti. Korisnicima se dodjeljuju uloge zavisno od njihovih obaveza, kompetentnosti za obavljanje posla, odgovornosti i kvalifikacije.

Osim osnovnog RBAC modela, u praksi se upotrebljavaju i proširenja RBAC-a. Standardna proširenja, data u okviru RBAC96 [8] standarda su: hijerarhija uloga, statičko i dinamičko razdvajanje obaveza (*Separation of Suties, SoD*) i kombinacija hijerarhije uloga i razdvajanja obaveza.

3. MODEL-DRIVEN ENGINEERING

U zadnjih 20-tak godina, napredak programskih jezika opšte namjene nije dao očekivane rezultate po pitanju produktivnosti programera [9]. Veći efekat na produktivnost je imao napredak ravojnih okruzenja (*Integrated Develepment Enviroment*, IDE) koji omogućava lakše editovanje i rukovanje s programskim kodom, vizuelizaciju i statičku analizu. Značajan pomak u produktivnosti je moguće postići podizanjem nivoa apstrakcije na kojem se programski kod pise ili crta – inženjering zasnovan na modelima [10].

Model-Driven Engineering (MDE) je metodologija razvoja softvera, zasnovana na modelima i njihovom intenzivnom korišćenju u svim aspektima procesa razvoja. Modeli se dalje progresivno dorađuju i popunjavaju potrebnim detaljima dok se iz njih automaski ne generiše programski kod konačnog softverskog proizvoda. Često citirana definicija modela data je u [9]:

> *Model je koherentan skup formalnih elementa koji opisuju nešto, a napravljen da bi poboljšao analizu i omogućio razmjenu ideja između covjeka i mašine, provjeru kompletnosti, konkurentnosti, generisanje test slučajeva, ocjenu indikatora isplativosti, standardizovanje i tranformisanje u implementaciju.*

Model može biti predstavljen grafičkom i tekstualnom notacijom, ili kombinacijom obe. Tekst može biti napisan u ma-

šinski čitljivom jeziku za modelovanje ili u slobodnoj formi. Modeli se prave sa ciljem da daju formalne opise sistema na višem nivou apstrakcije i olakšaju specifikaciju, razumjevanje i automatizaciju procesa razvoja softvera.

Prije formulacije MDE metodologije, modeli su se takođe koristili tokom razvoja softvera, za bolje razumjevanje i kao način komunikacije između ljudi. Model je paralelno ažuriran sa programskim kodom, a u kasnijim fazama bi obično postajao zapušten. Model nije bio aktivni dio razvoja softvera; nije bio interpretiran, kompajliran, transformisan niti izvršavan od strane računara, a često ni formalno verifikovan.

Svaki model je definisan u skladu sa svojim metamodelom. Metamodel definiše jezik i način zadavanja modela, odnosno njegovu apstraktnu sintaksu [11]. Model predstavlja apstrakciju realnog sistema, a metamodel je eksplicitna specifikacija apstrakcije [12]. Đurić i dr. u [13] definišu metamodel:

Metamodel je model koji definiše strukturu, semantiku i ograničenja neke familije modela.

Metamodel ne označava nikakvu posebnu vrstu modela. Prefiks *meta* označava relativnu poziciju na skali apstrakcije između dva modela. Metamodel je "meta" samo za model čije koncepte opisuje.

Najčešće primjenjivane tehnike u okviru MDE-a su:

- Domain-specific modeling languages (DSMLs)
- Mehanizmi transformacije i generatori koda

Transformacija je jedan od ključnih metoda MDE pristupa. Njome je omogućeno sukcesivno prevođenje i rafiniranje modela, datih za različite apsekte i nivoe apstrakcije, sve do izrade konačne izvšne reprezentacije softvera. Proces transformacije je automatizovan i rezultuje nizom konzistentnih modela. Zbog jasno definisanih načina transfor–macije i s obzirom da svaki od modela u nizu prolazi validaciju, vjerovatnoća greške je svedena na minimum [14].

4. MODEL-DRIVEN SECURITY

U praksi se pokazalo da se u dosta slučajeva sigurnosni podsistem implementira u *ad-hoc* maniru, čak i kad je ostatak poslovno orjentisanog djela aplikacije urađen naprednim metodologijama. Razvoj aplikacije obično kreće sa specifikacijom i razvojem poslovno orjentisanog djela i kad on uđe u fazu finalizacije i testiranja, prelazi se na realizciju "trivijalnih" stvari. Takvim pristupom nije moguće ostvariti zadovoljavajuću integraciju poslovnog i sigurnosnog dijela. Rezultat je hardkodirana sigurnosna logika prepletena sa poslovnom logikom, sa otežanim mogućnostima za održavanjem i ažuriranjem politike sigurnosti.

Kao rješenje, Basin i dr. u [4] predlažu korišćenje *Model Driven Security* (MDS) metodologije kao specijalizaciju MDE-a. U njemu dizajner pravi model sistema uključujuci bezbjednosne zahtjeve od samog početka i automatski generiše kompletnu i konfigurisanu infrastrukturu. MDS daje načine i alate za efikasno uključivanje sigurnosnih aspekata u razvoj aplikacije. Metodologija koristi vizuelne modele visokog nivoa apstrakcije za opisivanje sigurnosnog podsistema kao i načina integracije sa ostatkom aplikacije. Iz modela se dalje, putem automatizovanih mehanizama transformacije i generisanja dobija funkcionalna infrastruktura za sprovođenje kontrole pristupa. Kontrola pristupa je problem koji se iznova javlja u svakom narednom softverskom projektu. Korišćenje MDE pristupa omogućava da se jednom dizajniran, verifikovan i tesiran model kontrole pristupa na lak način integriše u novi proizvod, uz minimalan manuelni rad i mogućnost greške [15].

Prednosti MDS-a [16]:

- efikasna integracija sa poslovnom logikom,
- omogućiti potpuni SoC (*Separation of Concerns*) aplikacije i sigurnosnog podsistema
- uključenje sigurnosnih aspekata od početka razvoja,
- fleksibilan način zadavanja sigurnosnog modela i prava pristupa, i
- formalizacija modela sigurnosti i prava pristupa da bi omogućili standardizaciju i sertifikaciju sistema.

4. MODELOVANJE KONTROLE PRISTUPA

U ovom poglavlju predstavljene su najčešće korišćene metode za modelovanje kontrole pristupa (prava pristupa). Opisana su reprezentativna rješenja za svaku od metoda.

4.1 *Formalno - jezičko modelovanje prava pristupa*

Jedan od prvih radova u pravcu deklerativnih specifikacija sigurnosnih apsekata su radovi [17] i [18] gdje su dati formalni jezici za specifikaciju SoD tipova ograničenja u RBAC sistemima, RSL99 i RCL2000, respektivno. Jezici su formulisani kao logičko-predikatski iskazi. RSL99 (*Role-Based Separation of Duty Language*) nema mogućnost modelovanja durgih aspekata prava pristupa, osim SoD ograničenja.

U radu [18] Ahn i Sandhu definišu formalni jezik za definisanje SoD ograničenja – RCL2000 (*Role Constraint Language*). RCL2000 je unapređena verzija RSL99. RCL2000 je zasnovan na standardnom RBAC96 modelu i koristi 6 skupova: korisnici U, uloge R, objekti OBJ, operacije OP, dozvole $P=OP\times OBJ$ i sesije S. Funkcije preslikavanja su: *user:* $S\rightarrow U$ i $R\rightarrow 2^U$, *roles*: $U\cup P\cup S\rightarrow 2^R$, *sessions*: $U\rightarrow 2^S$, *permissions*: $R\rightarrow 2^P$, *operations*: $R\times OBJ\rightarrow 2^{OP}$, *object*: $P\rightarrow 2^{OBJ}$. Funkcije *roles** i *permissions** su proširenja funkcija *roles* i *permissions*, respektivno, u prisustvu hijerarhije uloga.

RCL2000 definiše i dvije nedeterminističke funkcije, *oneelement(X)* ili *OE(X)* i *allother(X)* ili *AO(X)*. OE(X) vraća jedan elemenat x_i iz skupa X. Više pojava funkije OE(X) u okviru jednog izraza vraća isti elemenat x_i. Putem funkcije AO(X) možemo da dobijemo skup iz koga je izdvojen jedan element. ove dvije nedeterminističke funkcije su povezane kontekstom u smislu da za bilo koji skup S vijedi:

$$\{\mathrm{OE}(S)\} \cup \mathrm{AO}(S) = S.$$

Korišćenje funkcija je dato na primjeru „Ni jedan korisnik ne smije imati dodjeljenje dvije konfliktne uloge“:

$$|roles(\mathrm{OE}(U)) \cap \mathrm{OE}(CR)| \leq 1$$

gdje je CR skup konfliktnih uloga.

Izraz za dinamičku verziju istog ograničenja glasi:

$$|roles(\mathrm{OE}(sessions(\mathrm{OE}(U)))) \cap \mathrm{OE}(CR)| \leq 1$$

Funkcije definisane u RCL2000 ne uključuju vrijeme ili stanje sistema, te nije u mogućnosti da iskaže vremenska ograničenja ili ograničenja zasnovana na istoriji. U okviru rada nije dat radni okvir koji bi izvršavao zadate iskaze.

Ahn i Shin u [19] koriste OCL (*Object Constraint Language*) kao osnovu za definisanje ograničenja baziranih na ulogama: ograničenja preduslovom, ograničenja kardinalnosti i SoD ograničenja. Upotrebu OCL-a možemo ilustrovati sledećim primjerima:

Ograničiti dodjelu 2 konfliktne uloge jednom korisniku:

```
context User inv:
let M : Set = {{clerk, manager}, ….} in
M->select(m | self.role -> intersection(m)->size > 1) -> isEmpty
```

Ograničiti dodjelu dvije konfliktne dozvole istoj ulozi:

```
let M : Set = {{prepare check, issue check}, ….} in
M->select(m | self.permission -> intersection(m)->size > 1) -> isEmpty
```

Korisnik može imati ulogu r1 samo ako prethodno ima ulog r2:

```
context User inv:
self.role ->includes(`tester`) implies self.role-> includes(`project_team`)
```

4.2. *XML bazirano modelovanje kontrole pristupa*

OASIS-ova XACML [20] specifikacija definiše jezik za definisanje prava pristupa kao i arhitekturu za sprovođenje kontrole pristupa. XACML jezik je XML baziran i u njemu je moguće izraziti različite autorizacione politike, uzimajući u obzir osobine subjekata i objekata, kao i kontekstne informacije. OASIS je definisao XACML RBAC profil, koji obuhvata osnovni RBAC, hijerarhiju uloga, statički i dinamički SoD. Standard obezbjeđuje: univerzalno razmjenjiv format definicije prava pristupa, podršku za grube i fine detalje u pravima pristupa, kondicionalnu autorizaciju, kombinovanje prava pristupa i rješavanje konfliktnih pravila, nezavisnost od načina implementacije.

XACML autorizaciona pravila se sastoje iz tri dijela: ciljni entitet na koji se pravila odnose, skup pravila i algoritam za kombinovanje pravila. Ciljni entiteti su subjekti, resursi, akcije i okruzenje na koje se prava pristupa mogu odnositi. Svako pravilo se dodatno može sastojati od još jednog opcionog ciljnog entiteta, uslova i efekta. Uslov definiše koje vrijednsti atributa moraju biti u zahtjevu da bi za zahtjev bio primjenjen efekat. Efekat može biti “odbijeno” ili “dozvoljeno”. Algoritam za kombinovanje pravila se koristi za rješavanje konflikata između pravila koja su u datom kontekstu primjenjiva, a imaju različite efekte [20].

XACML zahtjev se sastoji od liste atributa koji karakterišu subjekt, njegovo okruženje zajedno sa atributima akcije i atributima resursa. Osim standardne specifikacije, mogu biti dati dodatni profili napravljeni u cilju podrške za određene vrste modela bazirane na XACML-u. Npr. osnovni XACML RBAC profil se sastoji iz tri skupa autorizacionih politika: uloge, dozvole i SoD. Uloge su izražene preko XACML atributa subjekta. Zavisno od datih atributa, korisnik je pridružen određenom skupu uloga. Skup pravila pridruženih ulogama je dalje povezan sa dozvolama. Hijerarhija uloga je predstavljena povezivanjem sa dozvolama putem mehanizma referenci [21].

Arhitektura XACML-a se satoji od 4 komponente:

- Policy Administration Point (PAP)
- Policy Decision Point (PDP)
- Policy Enforcement Point (PEP)
- Policy Information Point (PIP)

PAP kreira prava pristupa, koje PDP koristi u procesu odlučivanja. PDP evaluira zadata prava i relevantne atribute zajedno sa parametrima iz zahtjeva za pristupom, te šalje rezultate PEP-u. PEP sprovodi kontrolu pristupa na osnovu podataka dobijenih od PDP. PIP pribavlja vrijednosti atributa potrebnih za proces evaluacije.

U radovima [22] i [23] Sladić i dr. daju je pregled kontekstno zavisne kontrole pristupa XML dokumentima. Za modelovanje prava pristupa autori uvode XXACF (*eXtensible Role-Based XML Access Control Framework*) model prava pristupa zasnovan na RBAC modelu. Glavne karakteristike XXACF-a su:

- baziran je na RBAC modelu sa hijerarhijom korisničkih uloga,
- podržano je kontekstno zavisno sprovođenje prava pristupa, i
- mogućnost definisanja dozvole i zabrane sa različitim prioritetima i granularnostima.

Definisanje prava pristupa počinje sa <*policy*> elementom sa dodjeljenim jedinistvenim id atributom. <*permission*> elemenat dozvoljava (*grant*) ili zabranjuje (*deny*) operaciju <*operation*> nad objektom <*object*>. Objekat može biti XML šema dokumenta ili instanca dokumenta označena jedinstvenim ID atributom. Prava pristupa definisana za šemu odnose se na sve instance XML dokumenta ustrojenih prema toj šemi. Ako je pravo pristupa definisano za određeni fragment šeme ili dokumenta, onda je potrebno definisati XPath izraz koji selektuje fragment. Korištćnje XPath izraza sa uslovima omogućuje dodjelu prava zavisnu od sadržaja.

Kontekstno zavisna dodjela prava pristupa se zadaje elementom uslova <*condition*>, koji može da sadrži druge poduslove i predikatske iskaze <*predicate*>. Predikatski iskaz je funkcija koja vraća vrijednost *true* ili *false*. Korišćenjem predikata čija vrijednost zavisi od trenutnog okruženja realizuje se kontekstno zavisna kontrola pristupa. Atributom *propagation_direction* u okviru elementa <*object*> kontroliše se smjer propagacije prava na dole (*down*) ili na gore (*up*), a artibutom

propagation_level se definiše maksimalan broj nivoa propagacije. Pravila definisana za čvorove potomke imaju prioritet nad pravilima za čvorove roditelje.

Sprovođenje kontrole pristupa odvija se u 4 koraka:

- vrši se selekcija prava koja su primjenjiva u datom slučaju,
- označavaju se čvorovi XML dokumenta,
- razrešavaju se eventualni konflikti pravila, i
- izvršava se tražena operacija.

4.3. *UML bazirano modelovanje kontrole pristupa*

Unified Modeling Language (UML) [24] je *defacto* industrijski standard za modelovanje, kako softverski baziranih, tako i drugih sistema. Konkretna sintaksa UML-a je vizuelna, a apstraktna sintaksa je definisana MOF metamodelom. Osnovni UML je moguće proširivati i prilagođavati određenom domenu putem *profila*. UML-om je moguće modelovati statičke i dinamičke karakteristike. S obzirom na vizuelnu konkretnu sintaksu, za efektivan rad potrebno je koristiti generički ili posebni UML softverski alat.

U radu [25] Hafner i dr. definišu SECTET radni okvir za MDS koji podržava dizajn, implementaciju i menadžement sigurnih *workflow*-a namjenjenih saradnji među različitim organizacijama bez kontrole od strane centralnog autoriteta. Radni okvir je namjenjen distribuiranim i servisno orjentisanim sistemima kao što su e-vlada, e-zdravstvo i sl. Sigurnosni zahtjevi se integrišu na nivou UML-a. UML model se dalje transformiše alatima. Sectet omogućuje saradnju organizacija na bezbjedan način. Pristup podrazumjeva korišćenje *Global Workflow Model*-a (GWfM), Local Workflow Model-a, (LWfM) za specifikaciju inter-organizacionih i intra-organizacionih workflow-a, respektivno.

U radu [26] Lodderstedt, Basin i Doser definišu SecureUML, jezik za modelovanje sistema računarske bezbjednosti. SecureUML definiše vokabular za dopunjavanje UML baziranih modela sa informacijama o kontroli pristupa. Dodate informacije se kombinuju sa UML modelom poslovnog procesa i generiše se infrastruktura za sprovođenje kontrole pristupa. Autori koriste MOF da definišu novi UML bazirani jezik za modelovanje RBAC prava pristupa i ograničenja. SecureUML ne definiše koji resursi su predmet kontrole pristupa, pa isti autori u radu [27] dodaju ComponentUML, jezik za modelovanje komponentno-baziranih sistema. ComponentUML sadrži entitete koji su specijalizacija *Action* i *Resource* entiteta iz SecureUML metamodela. Njihovim instanciranjem se modeluju aplikativni resursi koji su predmet kontrole pristupa.

UMLSec [28] je jedan od prvih radova na temu modelovanja bezbjednosnih aspekata distribuiranih poslovnih sistema. U radu autor prezentuje UMLSec proširenje UMLa koje omogućuje da se informacije bitne za kontrolu pristupa mogu iskazati u okviru dijagrama specifikacije sistema. UMLSec je dat u vidu UML profila koristeći standardne mehanizme za proširenje UML-a. Pristup obuhvata proširivanje dijagrama klasa, stanja i interakcije. Dijagram klasa predstavja strukturu sistema i služi za obezbjeđenje razmjene podataka. Dijagram stanja opisuje ponašanje određenog objekta i koristi se za sprečavanje indirektnog toka podataka sa viših na niže nivoe bezbjednosti. Dijagram interakcije opisuje razmjenu poruka među objektima i služi za verifikaciju korektnosti distribuiranih aspekata.

Dae-Kyoo i dr. u [29] opisuju metodologiju za uključivanje RBAC prava pristupa u UML modele sistema. Pristup podrazumjeva:

- tehniku za specifikaciju generičkih RBAC pravila pristupa u vidu šablona, gde se šabloni mogu instancirati u cilju definisanja namjenskih struktura specifičnih za datu oblast,
- tehniku za sistematsko uključivanje definisanih struktura u postojeće UML modele sistema, i
- tehniku za definisanje šablonskih struktura koje krše ograničenja data RBAC-om.

RBAC šabloni se po potrebi proširuju sa aplikativno-zavisnim konceptima i integrišu sa glavnim modelom poslovne aplikacije. Specifikacija šablona kršenja prava pristupa se zadaje u vidu OCL izraza. Na taj način se u toku razvoja identifikuju greške u statičkom dizajnu prava pristupa.

Sanchez i dr. u radu [30] predlažu ModelSec pristup. Za razliku od ostalih MDS pristupa ModelSec definiše generativnu arhitekturu zasnovanu na ulančanim transformacijama *security* modela datih na različitim nivoima apstrakcije. U radu je takođe dat i SecML, DSL jezik za specifikaciju sigurnosnih zahtjeva. SecML nije baziran na UML-u, već na osnovu *security requirements* metamodela autora, za SecML je definisana samo grafička konkretna sintaksa. ModelSec je uopšteni MDS pristup i nije fokusiran na neku posebnu oblast (npr. kontrola pristupa). Pristup je generativan i usko povezan sa procesom razvoja aplikacije. Pristup ne omogućuje potpuno sprovođenje SoC principa, niti izmjene prava pristupa u *runtime*-u.

U radu [31] Ray i dr. koriste parametrizovane UML dijagrame za modelovanje RBAC i MAC radnih okvira. Radni okviri se dalje ručno kombinuju i stvaraju *Hybrid Access Control* model za definisanje prava pristupa. Iako univerzalniji od MAC-a ili RBAC-a, model nije dalje proširiv. Autori ne definišu metamodel prava pristupa, niti DSL za unos prava, osim UMLa.

Slimani i dr. u [32] ide korak dalje u smislu da definišu metamodel prava pristupa kao osnovu za razvoj UML-baziranog jezika za modelovanje. Metamodel se sastoji od 4 klase: *Subject, Category, Resourc i Action*. Da bi modelovali MAC ili RBAC, potrebno je specijalozovati metamodel uvođenjem novih klasa koje nasleđuju 4 početne i time specijalizuju semantiku. Od svih analiziranih, ovaj pristup je najsličniji našem. Međutim, izloženi metamodel je više hibridni model kontrole pristupa nego metamodel. Metamodel nije meta u smislu da služi za modelovanje modela kontrole pristupa, već je generički šablon od kojeg daljom specijalizacijom dobijamo RBAC, MAC ili neki drugi model kontrole pristupa.

U radu [33] prezentovan je UML profil RBAC-a, putem kojeg specifikacija prava pristupa može biti modelovana grafički, zajedno sa specifikacijama domenske problematike, od samog

početka procesa razvoja. U radu se obrađuje način realizacija SoD-a, kardinalnosti i drugih ograničenja, kao i korišćenje OCL-a za validaciju modela u skladu sa RBAC pravilima. Rad se ne upušta u implementaciju radnog okvira za sprovođenje polike pristupa, kao ni razvoja druge konkretne sintakse pored UML grafičke. Dati UML profil obuhvata osnovne četiri komponente: osnovni RBAC, hijerarhijski RBAC, statički i dinamički SoD, i dodatne elemente: preduslovne uloge, vremenski zavisna ograničenja i kritične privilegije. Preduslovne uloge (*prerequisite roles*) za neku ulogu *R* su uloge koje korisnik mora prethodno da ima da bi mu bila moguće dodjeliti ulogu *R*. Kritične privilegije su privilegije potrebne za obavljanje zadatka koji podliježe SoD ograničenju. Kritična privilegija može biti dodjeljena samo jednoj korisničkoj ulozi.

U radu [34] prikazan je komplet alata (*toolchain*) namjenjen web-orjentisanim okruženjima, koji omogućava UML notaciju prava pristupa i njegovu konverziju u izvršnu formu. Prava pristupa se zadaju grafički korišćenjem UWE (*UML-based Web Engineering*) radnog okruženja. UWE alatom se zadaje:

- model sadržaja - opisuje domenske koncepte web aplikacije i odnose među njima,
- model uloga - definiše uloge i hijerarhiju korisni-čkih grupa,
- model osnovnih prava - izražava prava pristupa definisanih uloga nad domenskim konceptima koristeći RBAC bazirani model,
- model navigacije - grafička reprezentacija putanja kojima se korisnik kreće kroz web okruženje.

Korišćenjem UWE2XACML i XACML2FACPL alata, definisana prava pristupa se transformišu u XACML, i dalje u FACPL iskaze. *Formal Access Control Policy Language* (FACPL) je jezik formalne semantike namjenjen za definisanje prava pristupa. FACPL je čitljivija i upravljivija alternativa XML baziranom XACML jeziku. Korišćenje XACML-a u međukoraku omogućava integrisanje drugih standardnih alata za specifikaciju i verifikaciju prava pristupa. Korišćenje FACPL jezika u drugom koraku omogućava formalno-matematičku analizu prava pristupa. Izvršavanje FACPL iskaza je realizovano u Java programskom jeziku. Na osnovu zadath iskaza generiše se izvorni kôd i kompajlira u java klase i JAR arhive, koje postaju sastavni dio web aplikacije.

4.3. *DSL bazirano modelovanje*

Korišćenje standardnih jezika tekstualne i grafičke sintakse za modelovanje kontrole pristupa ne daje uvijek zadovoljavajuće rezultate. Npr. UML standard je obiman, kompleksan i nepraktičan za korišćenje od strane domenskog eksperta za sigurnost. Da bi se proces modelovanja učinio lakšim i efikasnijim, u literaturi je predloženo definisanje i korišćenje namjenskih programskih jezika (*Domain Specific Language,* DSL), prilagođenih oblasti kontrole pristupa.

U [15] Mouelhi i dr. predlažu korišćenje metamodela kontrole pristupa za generisanje test-slučajeva. U skladu sa metamodelom se definišu prava pristupa koja se u sledećem koraku transformišu u XACML. Integracija aplikacije i XACML platforme se vrši putem AOP-a. U radu se metamodel koristi za verfikaciju unesenih prava i za generisnje *fault injection* test-slučajeva. Metamodel podržava koncepte AC sistema baziranih na pravilima, bez direktne podrške RBAC familije i DSL-a za njeno modelovanje. Verifikacija modela se odvija u dvije faze: *a priori* -model se verifikuje prije generisanja PEP i PDP komponenti, i *a posteriori* – generisane komponente se verifikuju test-slučajevima.

Predloženi MDE proces je baziran na DSL-u i modeluje bezbjednosne formalizme/jezike kao i prava pristupa zadata u skladu sa ovim formalizmima. DSL je baziran na generičkom metamodelu koji obuhvata sve relevantne koncepte kontrole prisupa zasnovane na pravilima. Proces se oslanja na nekoliko automatizovanih koraka: generisanje posebnog radnog okvira, generisanje izvršnoe verzije PDP-a na osnovu prava pristupa, i automatsko ubacivanje PEP tačaka u poslovnu logiku putem aspekt-orjentisanog programiranja. Sintaksa DSL-a je ilustrovana na primjeru bibliotečkog sistema:

```
POLICY LibraryOrBAC (OrBAC):
R1 -> Permission(Library Student Borrow Book
WorkingDays)
R2 -> Prohibition( Library Student Borrow Book Holidays )
R3 -> Prohibition( Library Secretary Borrow Book Default )
R4 -> Permission( Library Personnel ModifyAccount
UserAccount WorkingDays )
R5 -> Permission( Library Director CreateAccount
UserAccount WorkingDays )
```

Iskaz *Rnnn->Permission(p q r s)* definiše pravilo *Rnnn* kojim se korisničkoj ulozi *p* omogućuje operacija *q* nad resursom *r* u datom kontekstu *s*. Analogno, iskazom *Rnnn->Prohibition(p q r s)* se definiše pravilo kojim se zabranjuje ulozi *p* da izvrši *q* nad *r* pri kontekstu *s*.

Hummer i dr. u [35] definišu DSL za RBAC koji omogućava definisanje IAM (*Identity and Access Management*) prava pristupa u servisno-orjentisanim okruženjima. DSL okruženje automatski generiše WS-BPEL [36] specifikacije na osnovu RBAC modela zadatog putem DSL-a. Autori koriste WS-BPEL mehanizam proširenja za označavanje i dopunjavanje definicija procesa sa direktivama vezanim za IAM. Dati pristup zahtjeva *redeployment* aplikacije nakon svake izmjene DSL definicije RBAC modela. Primjer sintakse DSL-a je dat u nastavku:

```
SUBJECT "jane"
SUBJECT "bob"
SUBJECT "135-222-001"
ROLE "staff"
ROLE "physician"
ROLE "patient"
ASSIGN "jane" "staff"
ASSIGN "bob" "physician"
ASSIGN"135-222-001" "patient"
INHERIT "staff" "physician"
MUTEX "patient" "physician"
CONTEXT "default"
CONTEXT "emergency"
RESOURCE "h1.com/patients"
RESOURCE "h1.com/emergency"
OPERATION "getPersonalData"
OPERATION "getCriticalHistory"
PERM "staff" "getPersonalData" "h1.
com/patients"
PERM "physician"
"getCriticalHistory" "h1.com/
emergency"
 WHEN "emergency"
```

Iskazi SUBJECT, ROLE, CONTEXT, RESOURCE, OPERATION definišu korisnika, korisničku ulogu, sigurnosni kontekst, resurs i operaciju, respektivno. Iskaz ASSIGN *x y* pri-

družuje korisnika *x* ulozi *y*. INHERIT *x* i označava da uloga *y* nasleđuje ulogu *x*. MUTEX *x y* označava da su uloge *x* i *y* međusobno isključive. Iskaz PERM i, *y*, *z* dodjeljuje ulozi *x* dozvolu za operaciju *y* nad resursom *z*. Ključna riječ WHEN povezuje dodjelu dozvole sa kontekstom u kojem je validna.

Zarnett i dr. u [37] predlaže implementaciju RBAC modela putem uvođenja *proxy* objekata i anotacija u Java programskom jeziku, tako što programer koji definiše prava pristupa dodaje anotacije na metode, interfejse i klase. Anotacije opisuju kojim korisničkim ulogama je dozvoljeno da pristupe nekoj klasi ili interfejsu ili da izvrše određenu metodu, tj. operaciju. Sistem automatski generiše posredničke (*proxy*) objekte sa samo onim metodama za koje je klijent autorizovan, kako je definisano pravima pristupa. Klijenti koji koriste RMI (*Remote Method Invocation*) interfejs dobijaju *proxy* objekte umjesto originalnih. U radu se razmatra proces anotacije, semantika anotacija, kako se izvode *proxy* objekti na osnovu datih anotacija i kako da RMI klijenti pozivaju metode putem *proxy* objekata. Prednosti datog pristupa je da je granularnost definisanja prava pristupa data na nivou metode u Java programskom jeziku.

5. ZAKLJUČAK

U prvim fazama razvoja računarskih sistema, aspekte kontrole pristupa je bilo moguće iskazati jednostavnim apstrakcijama u vidu matrice prstupa. U današnje vrijeme sigurnosne zahtjeve i prava pristupa gotovo da nije moguće modelovati bez primjene *Model Driven Engineering* tehnologija, odnosno, *Model Driven Security* metodologije. U radu je dat pregled različitih MDS pristupa za modelovanju kontrole pristupa.

Prvi radovi na datu temu su definisali modele implicitno, u vidu formalno – matematičkih izraza, dok su kasniji radovi bazirani na eksplicitno definisanim modelima. Eksplicitni definisanje modela je dalje gradirano prema rastućem nivou apstrakcije: XML, UML i DSL. Modelovanje kontrole pristupa bazirano na UML-u je najčešće korišćeno, zbog rasprostranjenosti UML standarda i dostupnosti UML alata.

Korišćenje DSL u svrhu modelovanja kontrole predstavlja sintezu tehnologija MDE, MDS i DSM, i dobija sve veću pozornost. Značajniji problem u ovom pristupu predstavlja nedostatak odgovarajućih alata za rad sa DSL-ovima. Prednosti ove metodologije su velike mogućnosti specijalizacije načina zadavanja prava pristupa.

6. LITERATURA

[1] Pierangela Samarati and Sabrina De Capitani di Vimercati, "Access Control: Policies, Models, and Mechanisms," 2001.

[2] R. Sandhu and P. Samarati, "Authentication, Access Control, and Audit," *ACM Computing Surveys*, vol. 28, no. 1, pp. 241-243, 1996.

[3] A. Schaad, J. Moffett, and Jacob J., "The Role-Based Access Control System of a European Bank: A Case Study and Discussion," *SACMAT '01*, 2001.

[4] D Basin and J Doser, *Model Driven Security: from UML Models to Access Control Infrastructures*.: Information Security Group, ETH Zurich, 2005.

[5] D. Schmidt, *Model-Driven Engineering*.: IEEE Computer Society, 2006.

[6] Butler W. Lampson, "Protection," in *Proceedings of the 5th Princeton Conference on Information Sciences and Systems*, 1971, p. 437.

[7] Department of Defense National Computer Security Center,.: Trusted computer system evaluation criteria (Orange book), 1985.

[8] D., Kuhn, R., Chandramouli, R. Ferraiolo, *Role-based access control*.: Artech House, 2003.

[9] S.J. Mellor, A.N. Clark, and T. Futagami, "Model-driven development - Guest editor's introduction," *IEEE Software*, vol. 20, no. 5, pp. 14-18, 2003.

[10] R. France and B. Rumpe, "Model-driven Development of Complex Software: A Research Roadmap," in *Future of Software Engineering*, 2007, pp. 37-54.

[11] J. Bézivin, "Model Driven Engineering: An Emerging Technical Space," in *Proceedings of the 2005 international conference on Generative and Transformational Techniques in Software Engineering*, 2005, pp. 36-64.

[12] I. Dejanović, *Metamodel, editor modela i generator poslovnih aplikacija*, Magistarska teza ed. Novi Sad: FTN, 2008.

[13] D. Djuric, D. Gasevic, and V. Devedzic, "The Tao of Modeling Spaces," *Journal of Object Technology*, vol. 5, no. 8, pp. 125-147, 2006.

[14] S. Kent, "Model Driven Engineering," in *Proceedings of the Third International Conference on Integrated Formal Methods*, 2002, pp. 286-298.

[15] T. Mouelhi, F Fleurey, B. Baudry, and Y.L. Traon, "A Model-Based Framework for Security Policy Specification, Deployment and Testing," in *MoDELS '08 Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems*, 2008, pp. 537-552.

[16] U. Lang and R. Schreiner, "Model driven security management: Making security management manageable in complex distributed systems," in *MODELS ć08*, 2008.

[17] G.J. Ahn and R. Sandhu, "The RSL 99 Language for Role-Based Separation of Duty Constraints," *ACM RBAC'99*, 1999.

[18] G.J. Ahn and R. Sandhu, "Role-Based Authorization Constraints Specification," *ACM Transactions on Information and System Security,* vol. 4, no. 3, pp. 207-226, November 2000.

[19] G.J. Ahn and M.E. Shin, "Role-based Authorization Constraints Specification Using Object Constraint Language," in *WETICE '01 Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2001, pp. 157-162.

[20] OASIS, *eXtensible Access Control Markup Language (XACML) Version 3.0*.: http://www.oasis-open.org/, 2010.

[21] J. Crampton, "XACML and Role-Based Access Control," in *DIMACS Workshop on Secure Web Services and e-Commerce*.

[22] G. Sladić, B. Milosavljević, Z. Konjović, and M. and Vidaković, "Access Control Framework for XML Document Collections," *Computer Science and Information Systems (ComSIS)*, vol. 8, no. 3, pp. 591-609, 2011.

[23] G. Sladić, B. Milosavljević, D. Surla, and Z. and Konjović, "Flexible Access Control Framework for MARC Records," *The Electronic Library*, vol. 30, no. 5, pp. 623-652, 2012.

[24] OMG, *Unified Modeling Language: Infrastructure*., version 2.0, formal/05-07-05.

[25] M. Hafner, R. Breu, B. Agreiter, and A. Nowak, "Sectet: an extensible framework for secure inter-organizational workflows," *Internet Research*, vol. 16, no. 5, pp. 491-506, 2006.

[26] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," in *Proceedings of the 5th International Conference on The Unified Modeling Language*, 2002.

[27] D. Basin, M. Clavel, J. Doser, and M. Egea, "A Metamodel-Based Approach for Analyzing Security-Design Models," in *MoDELS 2007*, 2007, pp. 420–435.

[28] Jan Jürjens, "UMLsec: Extending UML for Secure Systems Development," in *UML'02 Proceedings of the 5th International Conference on The Unified Modeling Language*, 2002, pp. 412-425.

[29] K. Dae-Kyoo, R. Indrakshi, R. France, and Na Li, "Modeling Role-Based Access Control Using Parameterized UML Models," in *FASE*, 2004, pp. 80–193.

[30] O. Sanchez, F. Molina, J. Garcıa-Molina, and A. Toval, "ModelSec: A Generative Architecture for Model-Driven Security," *Journal of Universal Computer Science*, vol. 15, no. 15, pp. 2957-2980, 2009.

[31] I. Ray, N. Li, D.K. Kim, and R. France, "Using parameterized UML to specify and compose access control models," in *Proceedings of the 6th IFIP TC-11 WG 11.5 Working Confrence on Integrity and Internal Control in Information Systems*, Lausanne, Switzerland, 2003, pp. 49-65.

[32] N. Slimani, H. Khambhammettu, K. Adi, and L. Logrippo, "UACML: Unified Access Control Modeling Language," in *4th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2011, pp. 1-8.

[33] Ç. Cirit and F. Buzluca, "A UML Profile for Role-Based Access Control," in *SIN '09 Proceedings of the 2nd international conference on Security of information and networks*, pp. 83-92.

[34] M. Busch, N. Koch, M. Masi, R. Pugliese, and F Tiezzi, "Towards Model-Driven Development of Access Control Policies for Web Applications," in *MDsec 2012*, Innsbruck, Austria, 2012.

[35] W. Hummer, P. Gaubatz, M. Strembeck, U. Zdun, and S. Dustdar, "An Integrated Approach for Identity and Access Management in a SOA Context," in *SACMAT'11*, Innsbruck, Austria, Austria, 2011, pp. 21-30.

[36] OASIS, *Web Services Business Process Execution Language Version 2.0*, OASIS Standard ed.: http://www.oasis-open.org, April 2007.

[37] J Zarnett, M. Tripunitara, and P. Lam, "Role-Based Access Control (RBAC) in Java via Proxy Objects using Annotations," in *Proceedings of the 15th ACM symposium on Access control models and technologies*, 2010, pp. 79-88.

[38] M. Strembeck and Mendling J., "Modeling process-related RBAC models with extended UML activity models," *Information and Software Technology*, vol. 53, pp. 456–483, 2011.

[39] J. Jürjens, *Secure Systems Development with UML*.: Springer-Verlag Berlin Heidelberg, 2005.

[40] S. Barker, "The Next 700 Access Control Models or a Unifying Meta-Model ?," *SACMAT'09*, pp. 187-196, 2009.

[41] W Sun, R. France, and I. Ray, "Rigorous Analysis of UML Access Control Policy Models," in *POLICY '11 Proceedings of the 2011 IEEE International Symposium on Policies for Distributed Systems and Networks*, 2011, pp. 9-16.

[42] F. Massacci and N. Zannone, "A Model-Driven Approach for the Specification and Analysis of Access Control Policies," in *OTM '08 Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008*, 2008, pp. 1087-1103.

[43] M. Alam, M. Hafner, M. Memon, and P. Hung, "Modeling and Enforcing Advanced Access Control Policies in Healthcare Systems with SECTET".

[44] M Mankai and L. Logrippo, "Access Control Policies: Modeling and Validation," in *Logrippo-Proceedings of the 5th NOTERE Conference*, Gatineau, Canada, 2005, pp. 85-91.

[45] W ESSMAYR, S. PROBST, and E. WEIPPL, "Role-Based Access Controls: Status, Dissemination and Prospects for Generic Security Mechanisms," *Electronic Commerce Research*, vol. 4, pp. 127–156, 2004.

[46] R Baird and R. Gamble, "Developing a Security Meta-Language Framework," in *HICSS '11 Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, 2011, pp. 1-10.

[47] J. Reznik and T. Ritter, "Model Driven Development of Security Aspects," *Electronic Notes in Theoretical Computer Science*, vol. 163, pp. 65-79, 2007.

[48] B. D. Olivier-Nathanaël and B. Benoit, "Toward a Model-driven Access-control Enforcement Mechanism for Pervasive Systems," in *MDsec 2012*, Innsbruck, Austria, 2012.

[49] S. Martínez, J. Cabot, J. Garcia-Alfaro, F. Cuppens, and N. Cuppens-Boulahia, "A model-driven approach for the extraction of network access-control policies," in *MDsec 2012*, Innsbruck, Austria, 2012.

[50] G. Sladić, *Proširivi sistem za kontrolu pristupa xml dokumentima zasnovanu na korisničkim ulogama*, magistarska teza ed. Novi Sad: FTN, 2006.

Branislav Trninić, Fakultet tehničkih nauka, Univerzitet u Novom Sadu
Kontakt: trninic@gmail.com
Oblasti interesovanja: razvoj softvera upravljan modelima, domen-specifični jezici, kompajleri i interpreteri

doc. dr Goran Sladić, Fakultet tehničkih nauka, Univerzitet u Novom Sadu.
Kontakt: sladicg@uns.ac.rs
Oblasti interesovanja: bezbednost informacija, upravljanje dokumentima, workflow sistemi, XML tehnologije

doc. dr Gordana Milosavljević, Fakultet tehničkih nauka, Univerzitet u Novom Sadu.
Kontakt: grist@uns.ac.rs
Oblasti interesovanja: poslovni sistemi, razvoj softvera vođen modelima, agilne metodologije