SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1651

SUSTAV UPRAVLJANJA ODREDBAMA NADZORA PRISTUPA U PROGRAMIRLJIVOJ INTERNET OKOLINI

Čedomir Šegulja

Zagreb, lipanj 2007.

Opis zadatka:

Proučiti osnovna načela ostvarenja nadzora pristupa u raspodijeljenim računalnim okolinama. Opisati postojeće sustave nadzore pristupa, te njihove podsustave za upravljanje odredbama nadzora pristupa. Posebice proučiti sustave nadzora pristupa uslugama u raspodijeljenim sustavima zasnovanim na uslugama. Nadalje, proučiti postupke upravljanja uslugama u okolini PIE (Programmable Internet Environment). Ostvariti sustav upravljanje odredbama nadzora pristupa uslugama u okolini PIE. Sustav izgraditi prema načelima računarstva zasnovanog na uslugama te primjenom .NET tehnologija. Opisati arhitekturu i ostvarenje sustava, te navesti korištenu literaturu i primljenu pomoć.

Zahvaljujem se mentoru prof.dr.sc. Siniši Srbljiću na prenesom znanju i omogućavanju daljnjeg školovanja.

Također, zahvaljujem se svim članovima PIE tima na brojnim savjetima i ugodnoj suradnji. Osobito hvala mr. sc. Miroslavu Popoviću i kolegi Marinu Šiliću na pomoći pri pisanju i ostvarivanju diplomskog rada.

Na kraju, najveća hvala obitelji na ljubavi koju mi nesebično pruža.

Sadržaj

1 Uvod

Proteklo desetljeće potvrdilo je globalnu mrežu Internet kao sveobuhvatni računalni sustav bez kojeg je nemoguće zamisliti funkcioniranje suvremenog društva. Sve do nedavno, Internet je prepoznavan kao sustav namijenjen prvenstveno razmijeni informacija. Razvojem računalnih tehnologija proširuje se osnovni skup funkcionalnosti koje Internet pruža. Osim komunikacije i razmjene podataka, danas je moguć razvoj raspodijeljenih primjenskih sustava zasnovanih na Internetu. Korisnici sve više prepoznaju Internet kao platformu za razvoj i povezivanje primjenskih sustava.

Poteškoće pri ostvarenju raspodijeljenih sustava proizlaze iz raznorodnosti programskih i sklopovskih tehnologija mreže Internet. *Računarstvo zasnovano na uslugama* prepoznaje budućnost razvoja programske podrške neovisnu o organizacijskim ili tehnološkim granicama. Usluge su samostalni programski elementi koji prikrivaju tehnološka svojstva svoje radne okoline izlažući funkcionalnosti pomoću općeprihvaćenih sučelja i protokola. Raspodijeljeni primjenski sustavi grade se povezivanjem usluga dostupnih na globalnoj mreži Internet.

Računarstvo zasnovano na uslugama pronašlo je svoju primjenu u poslovnom okruženju, gdje poslovne organizacije ostvaraju dobit objavljivanjem svojih usluga na Internetu i naplatom njihova korištenja. Pri tom, organizacije štite svoje usluge od neovlaštenog korištenja uporabom *sustava nadzora pristupa*. Zadaća sustava nadzora pristupa je tumačiti sigurnosne odredbe organizacije, donositi odluke o dozvoli pristupa, te sukladno odlukama omogućiti ili zabraniti korištenje usluga. Poslovna organizacija definira pravila pristupa svojim uslugama koristeći *sustav upravljanja odredbama nadzora pristupa*.

Programirljiva Internet okolina PIE pruža potporu za razvoj i izvođenje raspodijeljenih primjenskih sustava oslanjajući se na koncepte računarstva zasnovanog na uslugama. Grafičko sučelje okoline PIE prilagođeno je krajnjem korisniku i zasnovano na Internet pregledniku. Okolina PIE razvijena je na Zavodu ze elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu, u suradnji sa tvrtkom Ericsson Nikola Teska d.d iz Zagreba i uz potporu Ministarstva znanosti, obrazovanja i športa Republike Hrvatske.

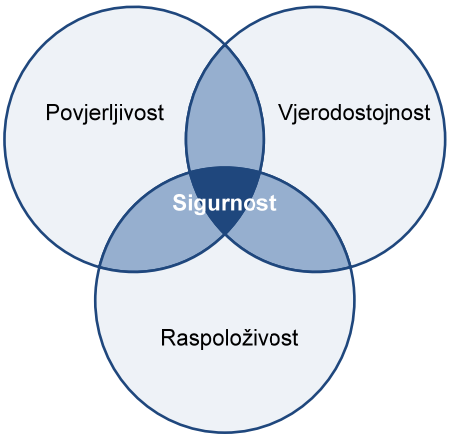
U ovom radu detaljno je opisana problematika nadzora pristupa, s posebnim naglaskom na oblikovanju nadzora pristupa u raspodijeljenim računalnim okolinama. Proučena su načela računarstva zasnovanog na uslugama, kao i dodatni zahtjevi koji se stavljaju pred

sustave nadzora pristupa uslugama. Dodatno, istražene su Web usluge kao najčeće primjenjivana tehnologija za ostvarivanje sustava zasnovanih na uslugama. U radu se razmatraju postojeći sustavi nadzora pristupa u raspodijeljenim okolinama, te se proučavaju postojeća rješenja za upravljanje odredbama nadzora pristupa. Opisana je arhitektura Programirljive Internet Okoline PIE, a poseban osvrt dan je na provođenje nadzora pristupa u okolini PIE. U okviru diplomskog rada osmišljen je i programski ostvaren *Upravitelj odredbama nadzora pristupa*. Korištenjem *Upravitelja odredbama nadzora pristupa* krajnjem korisniku se pruža mogućnost jednostavnog definiranja odredbi prava pristupa sredstvima okoline PIE.

Diplomski rad je organiziran u sedam poglavlja. U drugom poglavlju proučeni su koncepti vezani uz nadzor pristupa. Također, opisan je pristup oblikovanju sustava nadzora pristupa u raspodijeljenim računalnim okolinama. Treće poglavlje opisuje ideju računarstva zasnovanog na uslugama, arhitekturu zasnovanu na uslugama te tehnologiju Web usluga. U četvrtom poglavlju dan je pregled postojećih sustava nadzora pristupa u raspodijeljenim računalnim okolinama. Peto poglavlje sadrži opise glavnih elementa arhitekture okoline PIE i opis odredbi nadzora pristupa u okolini PIE. U šestom poglavlju opisana je arhitektura i programsko ostvarenje sustava *Upravitelj odredbama nadzora pristupa*. U sedmom poglavlju iznesen je zaključak diplomskog rada.

2 Nadzor pristupa

Poslovne organizacije nisu spremne povezivati se i surađivati na tržištu usluga, ako je time ugrožena *sigurnost* njihova poslovanja. *Računalna sigurnosti* (engl. computer security) podrazumijeva ostvarenje tri zahtjeva: povjerljivosti, vjerodostojnosti te raspoloživosti [1] [2].



Povjerljivost (ili tajnost, privatnost, engl. confidentiality, secrecy, privacy) podrazumijeva da pristup podacima imaju samo ovlašteni korisnici. Zahtjevom *vjerodostojnosti* (ili nepovredivost, engl. integrity) postiže se da podatke mogu modificirati samo ovlašteni korisnici, i to na samo na unaprijed određeni način. *Raspoloživost* (engl. availability) podrazumijeva osiguravanje dostupnosti podacima. Drugim riječima, ako određena osoba ima pravo pristupa nekom podatku tada se zahtjeva da joj taj pristup ne bude onemogućen. U literaturi je dosta često korišten suprotan pojam, tj. neraspoloživost (engl. denial of service).

Navedene zahtjeve moguće je ostvariti *ako se svaki zahtjev prema podacima nadzire i ako se samo ovlašteni zahtjevi provedu* [3]. Opisani proces naziva se *nadzor pristupa* (engl. access control). U ovom poglavlju detaljno je opisana problematika nadzora pristupa. Opisane su najvažnije politike, modeli i mehanizmi nadzora pristupa. Poseban osvrt dan je na provođenje nadzora pristupa u raspodijeljenim računalnim okolinama.

2.1 Povijest razvoja nadzora pristupa

Iako su sigurnosna pitanja bila predmet istraživanja već 1960-ih, područje računalne sigurnosti ubrzano se počelo razvijati 1970-ih, uvođenjem višekorisničkih računalnih sustava u državne, vojne i velike industrijske organizacije, te sveučilišta [4]. Stoga su upravo te organizacije značajno doprinijele razvoju računalne sigurnosti. Najraniji rad koji

pokušava definirati formalni, matematički model nadzora pristupa djelo je Lampsona [5] koji uvodi termine *subjekti*, *objekti* i *matrica pristupa* u kojoj su zapisana prava pristupa.

Ministarstvo obrane Sjedinjenih Američkih Država (engl. United States Department of Defense, DoD), uvidjevši rasprostiranje višekorisničkih sustava, odlučilo je ispitati njihovu sigurnost. Tako nastaju istraživanja RAND Corporation-a, i U.S. Air Force-a koji uvode zamisao nadziranja pristupa na osnovi oznaka sigurnosti dodijeljenih subjektima i objektima. Matematički model kojim se formalizira ta zamisao djelo je Bella i LaPadule [6] [7] [8] [9], čiji je rad imao veliki utjecaj na povijesni razvoj nadzora pristupa.

Nastavljajući se na rad Lampsona, Graham i Denninga te preuzimajući njihov način ostvarivanja nadzora pristupa pomoću matrice pristupa, Harrison, Ruzzo i Ullmann 1976. objavljuju značajan rad pod nazivom „*Protection in operating system*“ [10]. Težina rada leži u činjenici da su oni prvi, osim formalizacije nadzora pristupa, pokušali ispitati svojstva sigurnosti modela. Zaključili su da je nemoguće pronaći algoritam koji će za *sve* sustave moći reći jesu li sigurni (s obzirom na njihovu definiciju sigurnosti) ili ne, ali isto tako dokazali da postoje algoritmi koji uspješno ispituju sigurnost *pojedinih* sustava.

1985. godine Ministarstvo obrane Sjedinjenih Američkih Država objavljuje standard *Trusted Computer System Evaluation Criteria* (TCSEC) [11] poznat kao i „Narančasta knjiga” (engl. Orange Book). To je prvi pokušaj standardizacije normi koje sustavi moraju poštivati da bi se smatrali sigurnima. U 1990-ima, tijekom intenzivnog korištenja računalnih sustava u poslovnim okruženjima, javljaju se brojni novi pristupi ostvarivanju nadzora pristupa. Prepoznavanjem koncepta *uloge* u poslovnim i državnim organizacijama javlja se *pravo pristupa zasnovano na ulogama* (engl. Role Based Access Control). Pravo pristupa zasnovano na uslugama koristi mnoge koncepte prethodno uvedene u radovima Baldwina te Brewer i Nasha [12].

Pojavom brojnih komercijalnih rješenja računalnih sustava, javlja se potreba za određivanjem njihove razine sigurnosti. Računalna zajednica poduzima mnoge napore da bi se postigao koncenzus oko vrednovanja sigurnosti pojedinih sustava. 2005. je izdan međunarodni standard pod nazivom *The Common Criteria for Information Technology Security Evaluation*, poznat kraće kao *Common Criteria (CC)* [13]. To je pokušaj uvođenja standarda kojih se proizvođači moraju pridržavati kako bi sigurnost njihovih sustava mogla biti vrednovana.

2.2 Oblikovanje nadzora pristupa

Oblikovanje nadzora pristupa je složen proces koji zahtjeva definiciju pravila po kojim će se provoditi nadzor pristupa, te implementaciju mehanizama koji će osigurati njihovu primjenu.

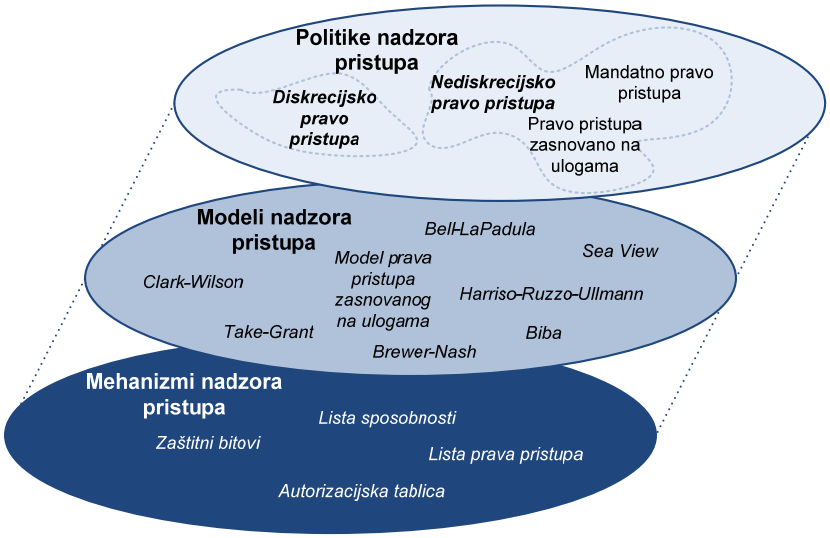
Skup pravila pristupa opisanih na visokom stupnju apstrakcije naziva se *politika nadzora pristupa*. Politika nadzora pristupa definira "tko, kako, i pod kojim okolnostima, može pristupiti određenoj informaciji" [4], ne uzimajući u obzir implementacijske detalje. Politiku nadzora pristupa čini skup *odredbi nadzora pristupa* kojima se definiraju prava pristupa *subjekata* podacima sustava.

Subjekt (engl. subject) je svaki entitet računalnog sustava koji može obavljati operacije nad objektima. Pod pojmom subjekti najčešće podrazumijevamo korisnike računalnog sustava, ali i procese koje ti korisnici pokreću. Zahtjeva se da sve subjekte u sustavu može *jedinstveno identificirati*. Primjerice, procesi najčešće naslijeđuju identifikacijske oznake od korisnika, no ukoliko ih želimo razlikovati definiramo dodatne identifikatore. Dozvoljava se da neki entiteti istovremeno budu i objekti i subjekti u sustavu. Na primjer, kod računalnog programa je objekt, no njegovim pokretanjem izvršava se proces koji se smatra subjektom. Subjekti dolaze do podataka obavljajući *operacije* nad *objektima*.

Objekt (engl. object) je svaki entitet računalnog sustava koji sadrži ili može sadržavati informaciju. Pristup objektu podrazumijeva pristup informaciji koju on sadrži. U različitim okruženjima pod pojmom objekt podrazumijevamo različite stvari. Primjerice, objekti su n-torke i tablice (kada govorimo o bazama podataka); datoteke, direktoriji, programi, memorijske lokacije (kada govorimo o aplikacijama i operacijskim sustavima); ili pak raspodijeljene aplikacije, usluge, metode usluga (kada govorimo u kontekstu arhitekture zasnovane na uslugama).

Iako postoje brojne poznate konkretne politike nadzora pristupa (sigurnosne politike bolnica, banaka ili raznih poslovnih organizacija), njihovo navođenje nije od velike koristi budući da je njihov razvoj vezan za specifične sustave i organizacije. Stoga se politike nadzora pristupa svrstavaju u dvije glavne kategorije [14]: *diskrecijsko pravo pristupa* i *nediskrecijsko pravo pristupa*. Dok diskrecijsko pravo pristupa ostavlja određeni dio nadzora na diskreciji korisnika, nediskrecijsko pravo pristupa podrazumijeva postojanje sigurnosnog administratora sustava koji određuje odredbe nadzora pristupa. Pri tom ta dva skupa nadzornih politika nisu disjunktna (kao što je neprecizno prikazano na slici) već postoje politike koje imaju svojstva po kojima ih možemo svrstati u obje kategorije. Treba

napomenuti da to nije jedina moguća podjela, već je česta i ona na najčešće korištene politike: diskrecionu, mandatnu i pravo pristupa zasnovano na ulogama.



Slika 2-1: Politike, model i mehanizmi nadzora pristupa

Definicija konkretnih akcija nadzora pristupa, na niskoj razini apstrakcije pristupa naziva se *mehanizam nadzora pristupa* (engl. access control mechanism). Mehanizam nadzora pristupa sprovodi u dijelo politiku nadzora pristupa. Postoje brojni mehanizmi nadzora pristupa, svaki sa određenim prednostima i manama pri primjeni raznih politika, te ih je teško grupirati u određene skupine.

Pri razvoju programske podrške, javlja se potreba za dokazivanjem njene ispravnosti. To je posebice istina kada govorimo o sigurnosti. Neučinkovito je za svaku konkretnu realizaciju mehanizma nadzora pristupa testirati njegovu ispravnost. Stoga se uvodi koncept *modela nadzora pristupa* (engl. acccess control model). Model nadzora pristupa pruža formalni model politike nadzora pristupa, čime se omogućava dokazivanje ispravnosti sustava, tj. provjerava se je li sustav siguran s obzirom na dane zahtjeve sigurnosti. Uvođenjem modela nadzora pristupa smanjuje se jaz između politike i mehanizma nadzora pristupa. Modeli i mehanizmi nadzora pristupa razvrstavaju se s obzirom na politike koje podržavaju. Odnos politike, modela i mehanizama nadzora prikazan je na slici 2-1.

Razdvajanje pogleda na nadzora pristupa u tri nezavisne apstraktna nivoa: politika, model i mehanizam nadzora pristupa omogućuje modularno oblikovanje nadzora pristupa. Moguće je definirati samo politiku nadzora pristupa, te zatim izabrati sigurnosni model i mehanizam koji podržavaju danu politiku.

2.3 Registracija, identifikacija i autentikacija

Zadaća nadzora pristupa jest provjeravati zahtjeve subjekata za obavljanjem operacija nad objektima sustava. Da bi se nadziranje pristupa uopće moglo provoditi, nužno je precizno utvrditi identitet subjekta. Tu zadaću provode procesi *registracije*, *identifikacije* i *autentikacije*.

Registracija (engl. registration) je postupak prikupljanja informacija o korisniku. Osnovna zadaća registracije je dodijeliti korisniku jedinstveni identifikator i autentikacijsku značku na osnovi kojih će se korisnik prijavljivat za rad u sustavu. U postupku registracije se mogu sakupljati i dodatne informacije o korisniku, pomoću kojih se sustav može prilagođavati zahtjevima korisnika.

Bez precizne *identifikacije* nije moguće kvalitetno nadzirati pristup objektima sustava. *Identifikacija* (engl. identification) je proces tijekom kojeg se korisnik predstavlja koristeći *jedinstveni identifikator* (engl. unique ID) dodijeljen u postupku registracije. U računalnim sustavima jedinstveni identifikator je najčešće korisničko ime (engl. username). Nakon predstavljanja, od korisnika se obično zahtjeva i potvrda indetiteta, proces poznat kao *autentikacija*.

Autentikacija (engl. authetication) je postupak provjere indentiteta korisnika. Zadaća autentikacije jest utvrditi da li je indetitet korisnika *autentičan*, tj. da li je korisnik uistinu onaj za koga se predstavlja. Postupak autentikacije uobičajen je i u stvarnom životu te služi za uspostavu povjerenja između stranaka u komunikaciji: bankovni čimbenici traže od stranaka osobnu iskaznicu, carinski službenici kao potvrdu indetiteta zahtjevaju putovnicu, itd.

U procesu potvrdu indentiteta korisnika autentikacijski mehanizmi oslanjaju se na ono što korisnik *zna*, *posjeduje* ili ono što korisnik *jest*. Primjer autentikacijskih mehanizama koji se oslanjaju na *znanje* korisnika su sustavi u kojima korisnik prilikom prijave navodi šifru, tajnu frazu ili osobni indetifikacijski broj (PIN, JMBG). Osim onoga što zna, korisnik može priložiti ono što *posjeduje*, poput identifikacijske značke, ključeva ili pametne kartice. Također, razlikujemo i autentikacijske mehanizme koji se oslanjaju se na fizičke karakteristike korisnika, odnosno na ono što korisnik *jest*. Korisnik potvrđuje svoj identitet otiskom prsta, glasom, skeniranjem mrežnice, itd. Iako su ovakve autentifikacijske metode već odavno poznate i u socijalnim okruženjima najrašireniје (poznanike prepoznajemo po izgledu ili po glasu korištenjem telelfona) njihova uporaba u računalnim sustavim primjetna je tek u novije vrijeme.

Procesi identifikacije i autentikacije često su spojeni u jedan korak. Primjerice, prilikom prijave u računalni sustav korisnik navodi svoje korisničko ime (na osnovu kojeg sustav prepoznaje o kome se radi, tj. identificira korisnika) i šifru (na osnovu koje sustav provjerava da li se uistinu radi o tom korisniku, tj. autentificira korisnika).

2.4 Politike nadzora pristupa

Politiku nadzora pristupa (engl. access control policy) čini skup odredbi kojima se definiraju prava pristupa subjekata podacima sustava. Njome se definiraju zahtjevi nadzora pristupa na visokom nivou apstakcije, neovisno o računalnom sklopovlju i programskoj podršci. U nastavku je dan pregled glavnih kategorija politika nadzora pristupa: *diskrecijsko pravo pristupa* i *nediskrecijsko pravo pristupa*.

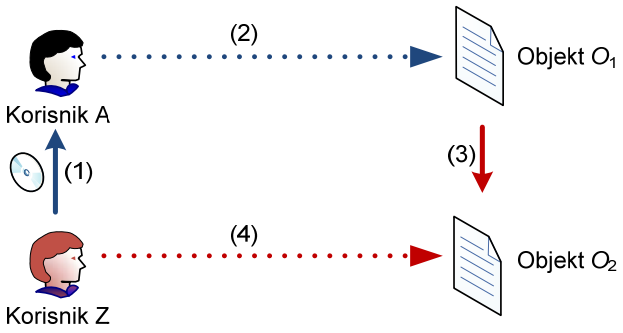
2.4.1 Diskrecijsko pravo pristupa

Diskrecijsko pravo pristupa (engl. Discretionary Access Control, DAC) definirano je u TCSEC-u [11] kao:

> *"Nadziranje prava pristupa objektima na osnovu indentiteta subjekta i/ili grupe kojoj pripadaju. Dozvole su diskrecijske u smislu da subjekt koji ima određenu dozvolu može prenijeti tu dozvolu (možda i indirektno) na drugog subjekta (osim ako nije ograničeno mandatnim pravom pristupa)"*

Diskrecijsko pravo pristupa, kao što ime kaže, ostavlja određeni dio nadzora nad objektima na diskreciji subjekata sustava. Ostvarivanje ove politike obično uključuje uvođenje koncepta *vlasništva* nad objektom, gdje je vlasnik objekta onda taj koji raspolaže pravom da omogući ili zabrani pristup tom objektu. Diskrecijsko pravo pristupa je veoma korišteno u komercijalnom sektoru. U 1980-im i 1990-im mnoge su organizacije smatrale DAC idealnim rješenjem nadzora pristupa. No, određena svojstva diskrecijskog prava pristupa čine ga manjkavom sigurnosnom politikom.

Diskrecijsko pravo pristupa dozvoljava prepisivanje informacija iz jednog objekta u drugi; time ne postoji nadzor nad tokom informacija u sustavu. To uzrokuje ranjivost sustava koji koriste diskrecijsko pravo pristupa na napad *trojanskim konjem. Trojanski konj* (engl. Trojan horse) je računalni program koji je naizgled bezopasan, no njegovom aktivacijom dolazi do ugrožavanja sigurnosti.



Slika 2-2: Primjer napada trojanskim konjem

Primjer napada trojanskim konjem prikazan je na slici 2-2. Korisnik *A* (a time i programi koje korisnik pokreće) ima pristup objektu O_1. Korisnik Z nema pristup objektu O_1, no ima pristup objektu O_2. Korisnik *Z* izrađuje trojanskog konja kojemu je zadatak prepisati sadržaj objekta O_1 u objekt O_2 te ga predaje korisniku *A* (1), koji ga, neznajući njegovu pravu namjeru, pokreće (2). Trojanski konj se izvršava (3) i prepisuje sadržaj objekta O_1 u objekt O_2, čime korisnik *Z* neovlašteno dolazi do pristupa informaciji sadržanoj u objektu O_2 (4).

Diskrecijsko pravo pristupa formalno se opisuje matricama pristupa, kao u modelima Lampsona, Graham-Denninga, te Harrison-Ruzzo-Ullmanna. Najčešće se implementira uporabom lista sposobnosti i mehanizmom zaštitnih bitova.

U velikim organizacijima često se ne dozvolja da odluke o pristupu objektima sustava donose subjekti (najčešće njihovi stvoritelji), već postoje definirana pravila na osnovi kojih se provodi sigurnosna politika organizacije. U tim uvjetima koristi se politike *nediskrecijskog prava pristupa*.

2.4.2 Nediskrecijsko pravo pristupa

Sve politike nadzora pristupa koje nisu diskrecijske svrstavaju se u kategoriju *nedeskrecijskih prava pristupa* (engl. Non-discretionary Access Control, NDAC). Kao što naziv sugerira, kod politika ove kategorije postoje pravila nadzora pristupa o kojima ne mogu odlučivati korisnici. Neki primjeri nediskrecionih politika su *mandatno pravo pristupa* (engl. mandatory access control), *pravo pristupa zasnovano na ulogama* (engl. role-based access control) i *pravo pristupa zasnovano na zabrani* (engl. separation of duty).

Iako nedikrecijske politike pristupa, za razliku od diskrecijskih, sprečavaju neizravni protok informacija (otporne su na navedeni problem *trojanskog konja*), njihovom uporabom ne postiže se nužno potpuna sigurnost računalnih sustava. Naime, politike sigurnosti nadziru

samo *glavne kanale* (engl. overt channels) informacija u sustavu, no ne osiguravaju nedozvoljeni protok informacija putem *skrivenih kanala* (engl. covert channels) [5]. Skriveni kanali su kanali koji nisu namjenjeni normalnoj komunikaciji i nemaju veliki kapacitet, no mogu biti upotrebljeni za otkrivanje povjerljivih informacija. Primjerice, promotrimo slučaj kada korisnik niže sigurnosne razine zatraži pisanje u nepostojeću datoteku više sigurnosne razine (taj zahtjev je legalan u *višerazniskoj sigurnosnoj politici* objašnjenoj kasnije). Ukoliko sustav dojavi nepostojanje datoteke, otkriva se informacija da je neki korisnik više sigurnosne razine obrisao datoteku. Ukoliko se pak pogreška ne dojavi ili sustav automatski stvori datoteku istog imena, korisnik neće biti obavješten o mogućim pogreškama pri legalnim operacijama pisanja u dotičnu datoteku. Slično, ukoliko korisnik više razine intenzivno koristi računalne resurse, korisnik niže razine primjećuje pad performansi sustava i zaključuje da se nešto događa. Time je otkrivena informacija. Za rješavanje problema skrivenih kanala predlagani su *modeli sučelja* (engl. interface models) [15] koji nastoje otkloniti skrivene kanala još tijekom modeliranja sustava.

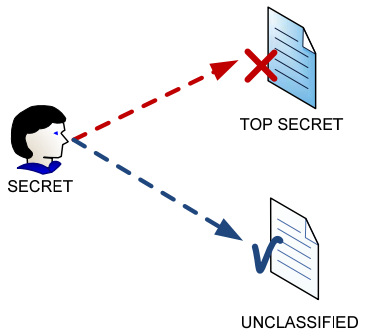
U nastavku su objašnjenje često korištene politike nediskrecijskog prava pristupa: *mandatno pravo pristupa*, te *pravo pristupa zasnovano na ulogama*.

Mandatno pravo pristupa

Mandatno pravo pristupa (engl. Mandatory Access Control) definirano je u TCSEC-u [11] kao:

> *„Nadziranje prava pristupa objektima na osnovu osjetljivosti informacije sadržane u objektu i ovlasti subjekata za pristup informaciji te osjetljivosti“*

Mandatno pravo pristupa podrazumijeva da su objekti klasificirani prema razini *osjetljivosti* (engl. classification level) informacija koju sadrže. Također, svakom subjektu je pridružena odgovarajuća razina *ovlasti* (engl. clearance level). Klasifikaciju subjekata i objekata provodi *centralni autoritet* (engl. central authority), administrator zadužen za uspostavu sigurnosti. Odluke o pristupu donose se uspoređivanjem razina osjetljivosti i ovlasti. Najpoznatiji primjer mandatnog prava pristupa je *višerazinska sigurnosna politika* (engl. multilevel security police).



Slika 2-3: Primjer višerazinske sigurnosne politike

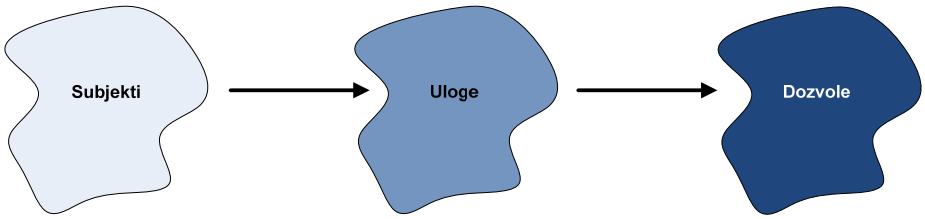
Ideja o razvoju višerazinske sigurnosne politike javlja se tijekom 1960-ih godina kada Američko Ministarstvo obrane uviđa potrebu za razvojem sustava kojim će se zaštiti informacije pohranjene u računalima. Do tada je bilo zakonski zabranjeno obrađivati povjerljive informacije pomoću sustava kojima su pristup imali neovlašteni korisnici, budući da se smatralo da računala ne mogu obavljati zadaću nadzora pristupa nad takvim podacima [2]. Razvijen je sustav u kojem su razine ovlasti subjekata i razine osjetljivosti objekata uređene dvojke oblika $(sigurnosna\ oznaka, kategorija)$. Sigurnosne oznake su mogle poprimiti vrijednosti poput *UNCLASSIFIED, CONFIDENTIAL, SECRET, TOP SECRET*, a kategorije su bile *NATO*, *NUCLEAR, ARMY* itd. i označavale su ogranke vojne orgnizacije. Definirani su odnosi među oznakama i kategorijama, te pravila na osnovu kojih su su donosile odluke o dodjeli ili zabrani prava pristupa. Slika 2-5 prikazuje primjer višrazinske sigurnosne politike. Subjekt, koji ima razinu ovlasti *UNCLASSIFIED* normalno pristupa datotekama označenim kao *UNCLASSIFIED*, no ne može čitati dokumente označene kao *TOP SECRET.*

Prvi matematički model višerazinske sigurnosne politike poznat je pod imenom Bell i LaPadulin model. U svojim radovima, Bell i LaPadula su prvi definirali mnoge termine i koncepte koji su kasnije usvojeni i korišteni u brojnim drugim modelima.

Pravo pristupa zasnovano na ulogama

Pravo pristupa zasnovano na ulogama (engl. Role Based Access Control, RBAC) razvijalo se, za razliku od mandatnog prava pristupa, prvenstveno u poslovnom svijetu. Kod mnogih organizacija korisnici nisu vlasnici informacije, već je sama organizacija *vlasnik* i odlučuje o protoku informacija. Unutar organizacija korisnicima su često dodijeljene *uloge* (engl. roles) na osnovu kojih dobivaju dopuštenja za obavljanje nekih vrsta poslova. Primjerice, unutar bolnice postoje uloge liječnika, medicinskih sestara, pacijenata itd. Liječnik ima dopuštenje postavljati dijagnoze, propisivati lijekove i naručivati

laboratorijske testove. Medicinska sestra nema dopuštenje za obavljanje tih operacija, ali ima za obavljanje nekih drugih, poput unošenja određenih podataka u bolesničke kartone.



Slika 2-4: Idejni prikaz prava pristupa zasnovanog na ulogama. Subjektima se dodjeljuju uloge kojima se dodjeljuju dozvole za obavljanje određenih operacija

Uočavanjem važnosti koncepta uloge u poslovnom svijetu nastalo je pravo pristupa zasnovano na ulogama. Korisnicima se dodjeljuju uloge na osnovi njihovih sposobnosti i odgovornosti unutar sustava. Odluke o operacijama koje korisnik smije obavljati donose se na osnovi uloga, a ne na osnovi identiteta korisnika, što omogućuje laku prilagodljivost na promjene u sustavu. Odnos subjekata, uloga i dozvola prikazan je na slici 2-4. Ukoliko korisnik promjeni ulogu u organizaciji, dovoljno je dodijeliti mu novu ulogu i oduzeti staru. U diskrecijskom pravu pristupa bilo bi potrebno zabraniti prava pristupa svim objektima za koje korisnik više nema dozvolu, i dodati prava pristupa svim objektima za koje je korisnik dobio dozvolu pristupa.

Prednosti pri administriranju koje donosi uvođenje koncepta uloge lako se mogu vidjeti na slijedećem primjeru. Neka je U skup korisnika određene uloge i P skup dozvola potreban za obavljanje te uloge. Ukoliko svakom korisniku po naosob dodijeljujemo dozvolu za obavljanje posla moramo obaviti $|U| \cdot |P|$ administrativnih operacija. S druge strane, korištenjem koncepta uloge, s $|U|$ operacija dodijeljujemo korisnicima ulogu, i s još $|P|$ operacija definiramo dozvole za ulogu, što čini ukupno $|U| + |P|$ administrativnih operacija. Uvjet $|U| + |P| < |U| \cdot |P|$ ispunjen je već za $|U| > 2, |P| > 2$, što je gotovo uvijek ispunjeno u poslovnim organizacijama.

Pravo pristupa zasnovano na ulogama brzo je prošlo put od ideje do komercijalne primjene. Objavljene su brojne publikacije na tu temu, te poduzeti znatni napori da bi se standardizirao model prava pristupa zasnovanog na ulogama. To je konačno učinjeno 2003. od strane ANSI-ja u dokmentu [16]. Model pruža mnoga proširenja osnovnog prava pristupa zasnovanog na ulogama (engl. Core RBAC) poput mogućnosti naslijeđivanja uloga (engl. Hierarchical RBAC), te dinamičkog i statičkog međusobnog isključivanja uloga (engl. Static Separation of Duty, Dynamic Separation of Duty). Također, pokazano je da tako definirani model može podržati i diskrecijske i mandatne politike pristupa [17].

2.5 Modeli nadzora pristupa

Model nadzora pristupa pruža formalni model politike nadzora pristupa, čime se omogućava dokazivanje ispravnosti sustava. Služi kao spona između visoke (politika nadzora pristupa) i niske razinine apstrakcije (mehanizam nadzora pristupa). Korištenjem modela moguće je ispitati politiku nadzora pristupa, te provjeriti ispunjava li ona postavljene zahtjeve.

Prvi modeli nadzora pristupa djelo su Lampsona [18] te Graham i Denninga [19]. Njihovi modeli su konačni automati čije stanje je uređena trojka (S, O, M), gdje je S skup subjekata, O skup objekata (koji sadrži skup S), a M *matrica pristupa* (engl. access matrix) koja ima po jedan red za svakog subjekta i jedan stupac za svakog objekta. Polje $M(s, o)$ sadrži prava pristupa subjekta s prem objektu o.

	subjekt 1	*subjekt 2*	*subjekt 3*	*objekt 1*	*objekt 2*	*objekt 3*
subjekt 1	Ø	{* *vlasnik*}	Ø	Ø	Ø	{*vlasnik*, **čitaj*}
subjekt 2	Ø	Ø	{vlasnik}	{*vlasnik*, *čitaj*, *piši*}	{*čitaj*}	{*piši*}
subjekt 3	Ø	Ø	Ø	{*vlasnik*, *čitaj*, *piši*}	{**vlasnik*}	Ø

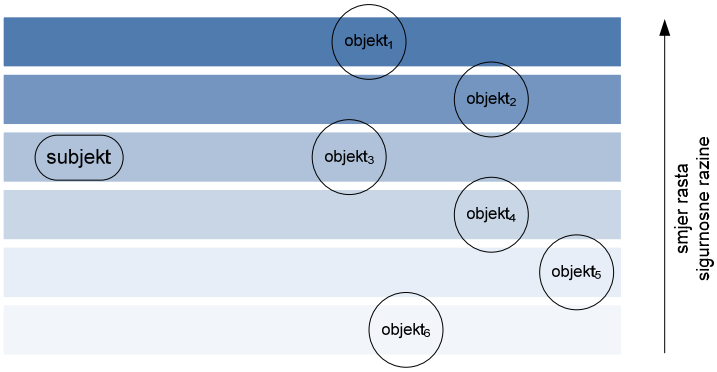
Slika 2-5: Primjer matrice pristupa

Primjer matrice pristupa prikazan je na slici 2-5. Prava označena s $*$ su prijenosna. $Subjekt_1$, na osnovu dozvole $*\,vlasnik$ nad $subjektom_2$ može ukloniti sva prava pristupa koja ima $subjekt_2$, kao mu i dodijeliti ona koja sam posjeduje. Kako je ta dozvola prijenosna, $subjekt_1$ ju može prenijeti primjerice $subjektu_3$ te tada i on upravlja pravima pristupa $subjekta_2$.

Iako konceptualno jednostavna, matrica pristupa mnogo je upotrebljavanja u daljnjim radovima o računalnoj sigurnosti, ponajviše zahvaljujući djelima Harrisona, Ruzza, Ullmana te Bella i LaPadule. Više o modelima nadzora pristupa može se naći u [15].

2.5.1 Bell-LaPadulin model

Bell-LaPadulin model djelo je David Elliott Bella i Lena LaPadule, koji su radeći za organizaciju MITRE formalizirali višerazinsku sigurnosnu politiku definiranu od strane Američkog Ministarstva obrane. Oni formaliziraju pojam sigurnosne razine prema kojima su klasificirani objekti i subjekti sustava. Primjer sigurnosnih razina dan je na slici 2-6.



Slika 2-6: Prikaz sigurnosnih razina

Važnost modela proizlazi iz činjenice da je to prvi uspješan pokušaj modeliranja višerazinsku sigurnosne politike, čime su se mogla detaljno ispitati njena svojstva. Njihov rad je imao velik utjecaj na prosuđivanje sigurnosti sustava u 1970-im i 1980-im godinama (standard TCSEC [11] oslanja se na Bell-LaPadulin model), a samim time i na povijesni razvoj nadzora pristupa.

Model[1] je konačni automat čiji su elementi:

S – skup subjekata

O– skup objekata

L – skup sigurnosnih razina parcijalno uređen relacijom $\leq \subseteq L \times L$ („dominira“)

$A = \{r, w\}$ – skup prava pristup, čitanje i pisanje

$M: S \times O \longrightarrow \mathcal{P}(A)$ - matrica pristupa, gdje je $M(s, o) \subseteq A$ i označava prava pristupa subjekta s nad objektom o

$b \subseteq S \times O \times A$ - skup trenutnih prava pristupa

R - skup zahtjeva (naredbi)

D - skup odluka

Definiraju se funkcije koje subjektima i objektima pridružuju sigurnosne oznake:

$$f = (f_s, f_o), f_s: S \longrightarrow L\ , f_o: O \longrightarrow L$$

[1] Prikazani model sadrži određena pojednostavljenja radi jasnoće, potpuni opis modela moguće je pronaći u [9]

Stanje sustava q je uređena trojka (b, M, f), dok se funkcija prijelaza definira se s $\delta: (Q \times R) \longrightarrow (Q \times D)$, gdje je Q skup stanja . Definiraju se slijedeća svojstva stanja:

Jednostavno svojstvo sigurnosti

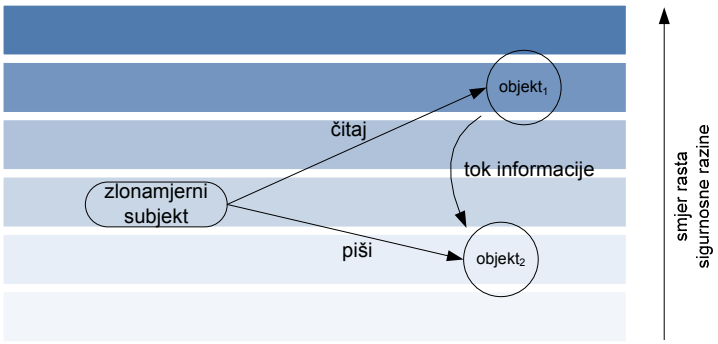
$$\forall s \in S, o \in O: \big((s, o, r) \in b\big) \Rightarrow f_s(s) \geq f_o(o)$$

Jednostavno svojstvo sigurnosti (engl. simple security property) osigurava da subjekti određene sigurnosne razine ne mogu čitati informacije zapisane u objektima više sigurnosne razine.

*** - svojstvo**

$$\forall s \in S, o_1, o_2 \in O: \big((s, o_1, r) \in b \wedge (s, o_2, w) \in b\big) \Rightarrow f_o(o_1) \leq f_o(o_2)$$

Namjera **-svojstva* (engl. star property) jest spriječiti neovlašten tok informacija u sustavu (kao u primjeru napada korištenjem trojanskog konja). Ukoliko subjek čita objekt određene sigurnosne razine, tada mu se zabranjuje istodobno pisanje u sve objekte nižih sigurnosnih razina. U suprotnom, zlonamjerni bi subjekt (kao na slici 2-7) mogao prepisati visokopovjerljivu informaciju u objekt niže sigurnosne razine, čime bi bila narušena njena povjerljivost.



Slika 2-7: Prikaz *-svojstva

Međutim, ponekad je potebno ostvariti komunikaciju među entitetima viših i nižih sigurnosnih razina, te stoga Bell i LaPadula uvode koncept *vjerodostojnog subjekta* (engl. trusted subject). Vjerodostojni subjekt, kao što ime kaže, je subjekt sustava kojemu se vjeruje da neće povrijediti zahtjeve sigurnosti, te se ne mora pridržavati *-svojstva.

Diskrecijsko svojstvo sigurnosti

$$\forall s \in S, o \in O, x \in A: \big((s, o, x) \in b\big) \Rightarrow x \in M(s, o)$$

Uvođenjem diskrecijskog svojstva sigurnosti (engl. dicretionary security property) Bell-LaPadulin model primjenjiv je i na diskrecijske politike nadzora pristupa. Pristup objektima dozvoljava se ukoliko postoji odgovarajući zapis u matrici pristupa (i ukoliko seu zadovoljavena jednostavno i *-svojstvo).

Bell i LaPadula definiraju *sigurno stanje* sustava: stanje q je sigurno ako zadovoljava jednostavno svojstvo sigurnosti, *-svojstvo i razlikovno svojstvo sigurnosti. Također definira se *siguran prijelaz*: prijelaz između stanja q i q' je siguran ako sigurnost stanja q povlači i sigurnost stanja q'. Konačno, *sustav* je *siguran* ako su sva dohvatljiva stanja sigurna. U svojim radovima, autori izvode i dokazuju tzv. *Osnovni teorem sigurnosti* (engl. Basic Security Theorem) koji kaže da je sustav siguran ako i samo ako je početno stanje sigurno i svi prijelazi su sigurni. Iako ovaj teorem nije karakterizacija sigurnosti već posljedica svojstva konačnih automata (kao što je istaknuto u radovima McLeana [20] [21] [22]), on daje potvrdu da se sigurnost može definirati kao induktivno (naslijedno) svojstvo.

Iako mnogo spominjan i primjenjivan, Bell-LaPadulin model ima ograničenja. Jedno od njih je pretpostavljanje konstantnosti sigurnosnih razina (engl. tranquility), tj. nije obuhvaćen problem dinamičke promjene sigurnosih razina[1]. Također, model obraća pozornost samo na povjerljivost informacija, dok ništa ne govori o njihovom integritetu. Bibin model, analogan Bell-LaPadulinom, bavi se integritetom pa se stoga često naziva dualnim modelom.

U svojim kasnijim radovima, Bell i LaPadula rade na konkretizaciji modela i prilagođavaju ga Multics operacijskom sustavu. Definiraju konkretne prijelaze, te pokazujući da oni zadovoljavaju definirana svojstva, dokazuju da je ostvareni sustav siguran.

[1] Autori taj problem pokušavaju riješiti naknadnim preinakama [9], omogućavajući da se korisnik prijavljuje i pod nižim sigurnosnim razinama

2.5.2 Harrison-Ruzzo-Ullmann model

Harrison, Ruzzo i Ullman [10] predložili su varijantu Graham-Denningova modela. Osim same formalizacije politike nadzora pristupa, autori ovog rada ispituju svojstva definiranog modela pokušavajući odgovoriti na pitanje "Ako je sustav u početku *siguran*, hoće li takav i ostati?".

Elementi njihovog modela su:

S – skup subjekata

$O, O \supseteq S$– skup objekata, subjekti su podskup objekata

A – skup prava pristupa

$M: S \times O \longrightarrow \mathcal{P}(A)$ - matrica pristupa, gdje je $M(s, o) \subseteq A$ i označava prava pristupa subjekta s nad objektom o

N – skup prava naredbi oblika

```
naredba α(x_1, x_2, ..., x_k)
  Ako r_1 ⊆ M(s_1, o_1) i
     r_2 ⊆ M(s_2, o_2)
     ...
     r_m ⊆ M(s_m, o_m) i
  tada izvrši
     op_1
     op_2
     ...
     op_n
kraj naredbe
```

, gdje je α ime naredbe, $x_1, x_2, \ldots, x_k$ formalni parametri, a op_i *primitivne operacije* definirane u tablici 2-1.

Tablica 2-1: Opis primitivnih operacija

Primitivna operacija	Uvjet	Akcija
$\boldsymbol{enter\ r\ into\ A[s_k, o_l]}$	$s_k \in S$ $o_l \in O$	$S' = S$ $O' = O$ $A'[s_i, o_j] = \begin{cases} A[s_i, o_j] \cup \{r\}, \text{za } (s_i, o_j) \neq (s_k, o_l) \\ A[s_i, o_j], \quad inače \end{cases}$
$\boldsymbol{delete\ r\ from\ A[s_k, o_l]}$	$s_k \in S$ $o_l \in O$	$S' = S,$ $O' = O,$ $A'[s_i, o_j] = \begin{cases} A[s_i, o_j] - \{r\}, \text{za } (s_i, o_j) \neq (s_k, o_l) \\ A[s_i, o_j], \quad inače \end{cases}$
$\boldsymbol{create\ subject\ s_k}$	$s_k \notin$	$S' = S \cup \{s_k\},$ $O' = O \cup \{s_k\},$ $A'[s_i, o_j] = \begin{cases} A[s_i, o_j], \quad za\ (s_i, o_j) \in S \times O \\ \emptyset, \qquad inače \end{cases}$
$\boldsymbol{create\ object\ o_l}$	$o_l \notin O$	$S' = S,$ $O' = O \cup \{o_l\},$ $A'[s_i, o_j] = \begin{cases} A[s_i, o_j], \quad za\ (s_i, o_j) \in S \times O \\ \emptyset, \qquad inače \end{cases}$
$\boldsymbol{destroy\ subject\ s_k}$	$s_k \in S$	$S' = S - \{s_k\},$ $O' = O - \{s_k\},$ $A'[s_i, o_j] = A[s_i, o_j], \quad za\ (s_i, o_j) \in S' \times O'$
$\boldsymbol{destroy\ object\ o_l}$	$o_l \in O - S$	$O' = O - \{o_l\},$ $A'[s_i, o_j] = A[s_i, o_j], \quad za\ (s_i, o_j) \in S' \times O'$

Za dani sigurnosni sustav kažemo da naredba $\alpha(X_1, X_2, \ldots, X_k)$ *propušta pravo pristupa* r iz stanja $q = (S, O, M)$ ako se izvršenjem naredbe α upisuje pravo r u neki element matrice pristupa koji prethodno nije sadržavao r. Autori definiraju *siguran* sustav kao onaj u kojemu, krenuvši od početnog stanja, nije moguće doći u stanje iz kojeg se propušta pravo pristupa. Time formaliziraju logičan zahtjev; model nadzora pristupa bi morao odgovoriti može li zlonamjerni subjekt dobiti određeno pravo pristupa. No, kao što je dokazano u njihovom radu, nije moguće općenito odgovoriti na ovo pitanje, tj. krenuvši od početne matrice pristupa, nije moguće reći postoji li ili ne slijed naredbi takav da se njihovim izvođenjem u matricu upiše određeno pravo pristupa. Točnije, koristeći načela teorije odlučivosti [23] zaključuju:

> *Problem određivanja da li je određeno stanje sigurno za dano pravo pristupa* r *zadanog sigurnosnog sustava jest neodlučiv.*

Iako je, općenito postavljeni problem neodlučiv, moguće je dobiti odgovore na njegove posebne slučajeve. Autori tako definiraju *mono-operacioni sustav* kao sustav u kojem se sve naredbe sastoje od jedne primitivne operacije. Za takav sustav pronalaze algoritam koji može potvrditi dolazi li do propuštanja određenog prava te zaključuju:

> *Postoji algoritam koji odlučuje da li je početno stanje zadanog mono-operacioniog sigurnosnog sustava sigurno za dano pravo pristupa r.*

Zaključci Harrisona, Ruzza i Ullmana imali su velik utjecaj na područje računalne sigurnosti. Oni su prvi pokazali da je nemoguće pronaći algoritam koji će za *sve* sustave moći reći jesu li sigurni ili ne, ali isto tako dokazali da postoje algoritmi koji uspješno ispituju sigurnost *pojedinih* sustava.

2.6 Mehanizmi nadzora pristupa

Iako matrica pristupa predstavljena u djelu Lampsona [5] predstavlja dobar model za osmišljavanje nadzora pristupa, nije pogodna za implementaciju. Poteškoća proizlazi iz činjenice da je u realnim računalnim sustavima broj subjekata i objekata vrlo velik. Posljedica toga je velika i raspršena matrica pristupa (mnoga polja su nedefinirana). U nastavku se navode neke moguće implementacije matrice pristupa.

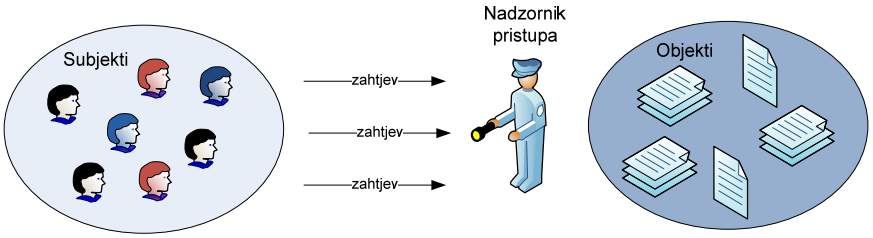
Tablicu autorizacije (engl. authorization table) čine tri stupca: stupac subjekata, stupac operacija i stupac objekata. Popunjava se na osnovi pripadne matrice pristupa, te svaki redak označa jedno pravo pristupa. Ovakav pristup čest je u sustavima za upravljanje bazama podataka, gdje su prava pristupa pohranjena u kataloge (relacije) baze podataka.

Liste sposobnosti (engl. capability lists) stvaraju se pohranjivanjem matrice pristupa po retcima. Svakom subjektu se dodjeljuje lista u kojoj se zapisuju njegove *sposobnosti*, tj. prava pristupa nad svim objektima u sustavu. Omogućuje brzi pregled prava pristupa za pojedinog subjekta, no, s druge strane, nije pogodna za implementaciju politika koje su orijentirane na objekte, što ograničava njenu komercijalnu upotrebu. Primjerice, prilikom definiranja zabrane prava pristupa određenom objektu potrebno je pregledati sve liste.

Liste prava pristupa (engl. access control lists, ACL) zasnivaju se na konceptu pohrane matrice pristupa po stupcima. Svakom objektu sustava dodjeljuje se lista u koju se zapisuju, za svakog subjekta, prava pristupa. Prednost liste prava pristupa je što omogućuje brzu provjeru prava pristupa nad određenim objektom. Naprimjer, zabrana pristupa objektu postiže se jednostavnim brisanjem pridružene liste. Upravo stoga, ovaj mehanizam nadzora pristupa najčešće je korišten.

Mehanizam zaštitnih bitova (engl. protetion bits) čini pojednostavljane liste prava pristupa. Ovaj mehanizam prava pristupa zastupljen je u Unix operacijskim sustavima. Svakom objektu dodjeljuje se lista u koju se zapisuje prava pristupa: *vlasnika*, *grupe*, *ostalih*. *Grupa* označava skup korisnika koji djele prava pristupa objektu. Mehanizam zaštitnih bitova ima manju izražajnost od matrice pristupa, te ne može podržati isti skup politika nadzora pristupa. Uzrokovano time, noviji Unix operacijski sustavi koriste liste prava pristupa.

2.7 Nadzornik pristupa



Slika 2-8: Nadzornika pristupa

Pojam *nadzornik pristupa* (engl. reference monitor) prvi je uveo Anderson u [24]. Nadzornik pristupa ne nameće zahtjev ugradnje specifične sigurnosne politike, već definira apstraktni koncept koji se upotrebljava već više od tri desetljeća u razvoju i implementaciji sigurnih računalnih sustava [4]. Idejno, nadzornik pristupa mora zadovoljavati slijedeće zahtjeve: *nepremostivost*, *neugrozivost* te *dokazivost ispravnosti*.

Nepremostivost (engl. completeness) podrazumijeva da *svaki* pristup subjekta k objektu mora proći kroz nadzornik pristupa. Ovaj zahtjev je teško potpuno ostvariv, zapravo vrlo mali broj računalnih sustava osigurava potpunu nepremostivost [2]. Dva su uzroka teške ostvarivosti potpune nepremostivosti nadzornika pristupa. Pod objektima računalnog sustava obično smatramo datoteke, memoriju, itd. Većina računalnih sustava korektno obavlja zadaću zaštite tih, *očitih izvora informacija*. No, informacija može biti pohranjena u primjerice nazivu datoteka, mapama (koje sadrže informacije o datotekama), itd. Zahtjev nepremostivosti nalaže da pristup svim objektima bude nadziran, a ne samo onim očitim. Drugi izazov u ostvarenju nepremostivosti je kako dopustiti pristup objektu samo putem nadzornika pristupa. Primjerice, sustavi za upravljanje bazom podataka nadziru pristupe tablicama koji dolaze kroz sučelja za rad s bazom podataka. No, kako mogu nadzirati zahtjeve za pristupima koji dolaze preko poziva metoda operacijskog sustava? Slično, zlonamjerni subjekt bi mogao pokušati zaobići datotečni podsustav operacijskog sustava i pokušati čitati memorijske lokacije na kojima su pohranjene informacije.

Zahtjev *neugrozivosti* (engl. isolsation) podrazumijeva da nadzornik pristupa mora biti otporan na moguće prijetnje. Ne smije se dozvoliti da zlonamjerni subjekt utječe ne funkcionalost nadzornika pristupa. Iako očit, poput zahtjeva nepremostivosti, ostvarivanje potpune neugrozivosti zahtjeva ne samo programsku već i sklopovsku podršku. Primjerice, u teoriji operacijskih sustava definira se pojam *jezgra* operacijskog sustava, koja čini osnovu operacijskog sustava i sklopovski je podržana: svaki neovlašteni pristup jezgri operacijskog sustava izazvat će sklopovsku iznimku, te se na taj način onemogućuje upad.

Ostvarenje nadzornika pristupa treba biti jednostavno, kako bi se mogla dokazati *ispravnost* (engl. verifiability). Ispravnost se pokušava pokazati pregledavanjem izvornog koda, rigoroznim testiranjem te formalnim metodama ili matematičkim modeliranjem.

2.8 Nadzor pristupa u raspodijeljenim računalnim okolinama

Objekti su u raspodijeljenim računalnim okolinama smješteni na različitim zemljopisnim lokacijama i u različitim *administrativnim domenama*. *Administrativnu domenu* (engl. administrative domain) čine podaci, računala, mreža računala ili pak skup mreža računala pod nadzorom istog *autoriteta*. *Autoritet* (engl. authority) je zadužen za sigurnost administrativne domene, koju ostvaruje definirajući i provodeći politiku nadzora pristupa. Različite administrativne domene često imaju različite politike nadzora pristupa, različit način opisivanja politika, te različite mehanizme provođenja nadzora pristupa. Kao posljedica raznorodnosti administrativnih domena, ostvarivanje globalnog nadzora pristupa na razini čitave raspodijeljene okoline je vrlo zahtjevan zadatak.

Sustav za nadziranje pristupa u raspodijeljenoj okolini mora moći odgovoriti na brojne zahtjeve [25]. Osnovni zahtjev je ostvarivanje *decentralizacirane administracije*. Zahtjeva se autonomnost admininistrativnih domena u definiranju odredbi nadzora pristupa i provođenju autorizacijskih odluka. Kako različite administrativne domene mogu imati različite politike nadzore pristupa, pa tako i različite načine njihova opisivanja, sustav nadzora pristupa mora podržati *različite jezike opisivanja politike nadzora pristupa*. Također, moraju se ostvariti mehanizmi *kompozicije* različitih politika nadzora pristupa, što je nužan uvjet za izražavanje i provođenje globalne sigurnosne politike nadzora. *Uspostava povjerenja* između administrativnih domena je preduvjet provođenja globalnih operacija (onih u kojima se pristupa objektima iz više administrativnih domena). Stoga, potrebno je ostvariti postupke kojima se uspostavlja povjerenje među entitetima sustava, poput infrastrukture javnog ključa (engl. Pubic Key Infrastructure, PKI). Nadalje, zahtijeva se *prilagodljivost na različite mehanizme autentikacije*. Sustav nadzora pristupa ovisi o

rezultatu provjere autentičnosti korisnikovog identiteta, ali ne bi smio ovisiti o načinu na koji se ta provjera provodi.

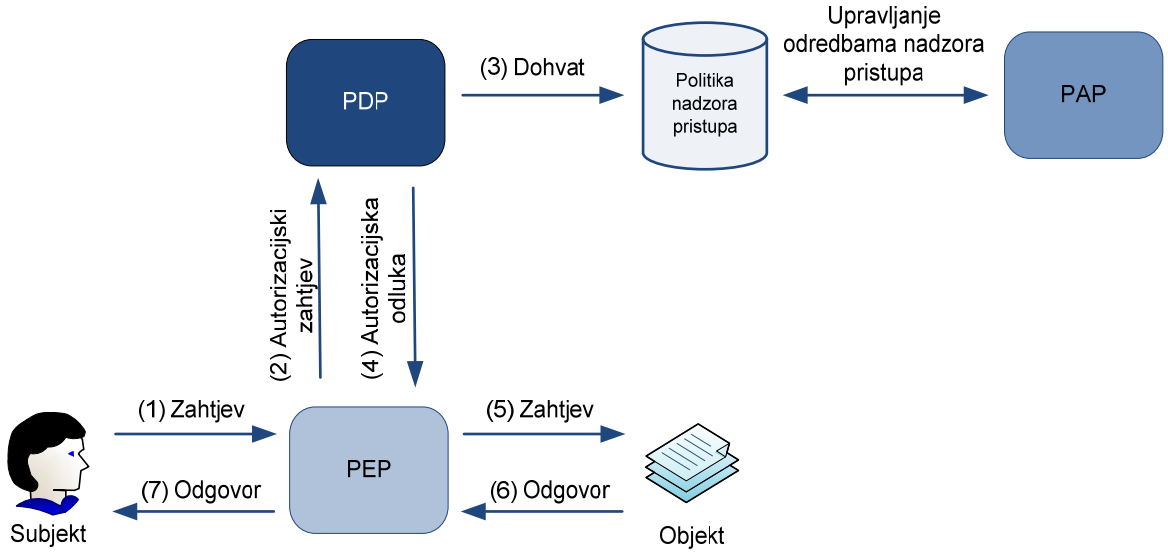
Postojanje više administrativnih domena unutar računalne okoline zahtjeva razlikovanje *globalnog* i *lokalnog identiteta* korisnika. *Globalni identitet* je identitet na razini čitave okoline. *Lokalni identitet* je identitet korisnika u administrativnoj domeni. Nasuprot jedinstvenom globalnom identitetu, korisnik može posjedovati više lokalnih identiteta. Uzrokovano time, sustav za nadzor pristupa u raspodijeljenoj okolini mora ostvariti mehanizme *preslikavanja globalnih i lokalnih identiteta*.

Pri donošenju autorizacijska odluke, sustav nadzora pristupa mora se oslanjati na globalni identitet i *svojstva korisnika*. Nepraktičan i nepregledan način administriranja politike prava pristupa računalne okoline jest postavljati prava pristupa za svaki lokalni identitet korisnika u svakoj administrativnoj domeni. Radije, administrativne domene izriču svoje politike nadzora pristupa objavljujući potrebna svojstva (a ne lokalni identitet) korisnika za ostvarivanje prava izvođenja određenih operacija. Svojstvo ima ulogu ulaznice koju subjekt predočava sustavu nadzora pristupa kada želi pristupiti određenom objektu.

Prvi korak u ostvarivanju sustava nadzora pristupa u raspodijeljenoj računalnoj okolini je razdvajanje funkcionalnosti. *Razdvajanje funkcionalnosti* (engl. separation of concerns) često je korišten koncept pri dizajniranju računarnih sustava. Omogućuje izradu složenih, zahtjevnih sustava dekompozicijom u manje cjeline. Osim što je te, manje, cjeline lakše osmisliti i izraditi, takvim pristupom se pruža mogućnost fizičke raspodijele sustava te mogućnost višestruke iskoristivosti pojedinih cjelina sustava. Primjenom koncepta razdvajanja funkcionalnosti pokušava se ponuditi standardna arhitektura sustava nadzora pristupa, koja treba olakšati njegovu izvedbu, posebice u raspodijeljenim računalnim okolinama. U nastavku je prikazana standardna arhitektura nadzora pristupa, opisana u radu [26].

2.8.1 Radni okvir nadzora pristup

Radni okvir nadzora pristupa (engl. Access Control Framework) [26] predložila je međunarodna organizacija IETF. Arhitektura sustava nadzora pristupa definirana u skladu s IETF radnim okvirom prikazana je na slici 2-9. Glavni elementi arhitekture su *PAP*, *spremnik politike nadzora pristupa*, *PDP* i *PEP*.

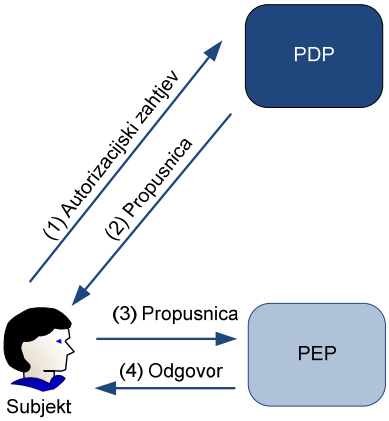


Slika 2-9: IETF arhitektura sustava nadzora pristupa

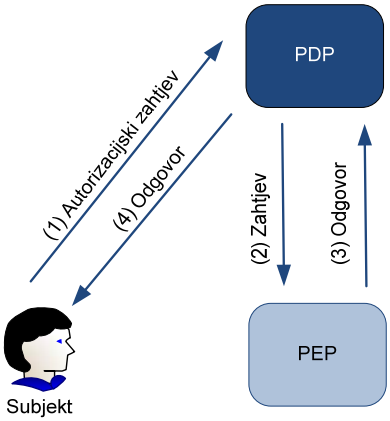
PAP (engl. Policy Administration Point) je entitet sustava zadužen za administriranje politike nadzora politike, odnosno za upravljanje odredbama nadzora pristupa. *PAP* može modificirati postojeće odredbe i spremati nove odredbe nadzora pristupa. Odredbe nadzora pristupa spremaju se u *spremnik politike nadzora pristupa*. *PDP* (engl. Policy Decision Point) je entitet sustava zadužen za donošenje *autorizacijske odluke* (engl. authorization decision). *PDP* prima *autorizacijski zahtjev* (engl. authorization request), dohvaća odredbe nadzora pristupa, tumači ih i donosi odluku. PDP pri donošenju odluke dodatno može komunicirati s računalnom okolinom kako bi pribavio dodatne informacije, poput svojstava korisnika. *PEP* (engl. Policy Enforcement Point) je entitet sustava koji prima korisničke zahtjeve, izdaje autorizacijske zahtjeve i provodi autorizacijske odluke.

Provođenje nadzora pristupa tipično se odvija po scenariji prikazanom na slici 2-9. Subjekt, koji želi izvršiti neku operaciju nad određenim objektom, izdaje korisnički zahtjev (1). *PEP* prima korisnički zahtjev, te oblikuje autorizacijski zahtjev kojeg prosljeđuje *PDP*-u (2). *PDP* prima autorizacijski zahtjev, dohvaća odredbe iz spremnika politike nadzora pristupa te ih tumači (3). Rezultat se zapisuje u autorizacijsku odluku (4), koja može sadržavati dodatne uvjete. Primjerice, dopušta se izvođenje operacije samo u određenom vremenskom intervalu. *PEP* pregledava autorizacijsku odluku te ovisno o njenom sadržaju dopušta ili odbija pristup objektu.

Ovakav protokol suradnje između *PEP*-a i *PDP*-a opisujemo kao *model potraživanja.* U *modelu potraživanja* (engl. pull model) korisnik prvo kontaktira *PEP* koji zatim potražuje autorizacijsku odluku izdanu od strane *PDP*-a. Razlikuju se još dva načina suradnje PEP-a i PDP-a: *model ponude* i *model agenta*.



Slika 2-10: Model ponude



Slika 2-11: Model agenta

U *modelu ponude* (engl. push model) korisnik prvo kontaktira PDP zahtijevajući pristup određenom objektu. *PDP* obrađuje zahtjev i izdaje *propusnicu* u kojoj navodi autorizacijsku odluku. Korisnik nudi propusnicu PEP-u koji mu na osnovi nje dopušta pristup objektu. Model ponude prikazan je na slici 2-10.

Varijanta modela ponude je *model agenta* (engl. agent model). Korisnik, kao u modelu ponude, kontaktira PDP navodeći autorizacijski zahtjev. PDP obrađuje zahtjev i ponaša se kao *agent*, proslijeđuje zahtjev PEP-u koji upravlja izvršavanjem odobrenog zahtjeva i vraća rezultat PDP-u, koji se zatim prenosi korisniku. Model agenta prikazan je na slici 2-11.

2.8.2 Upravljanje odredbama nadzora pristupa

Osnovni zahtjev koji se postavlja pred sustave upravljanja odredbama nadzora u raspodijeljenim računalnim okolinama u prisustu više administrativnih domena jest omogućavanje decentralizirane administracije [27]. Ostvarivanje autonomnosti administrativnih domena pri definiranju i izmjeni odredbi nadzora pristupa zahtijeva razvoj mehanizama *pohranjivanja*, *pronalaženja* i *iskazivanja* odredbi.

Odredbe nadzora pristupa mogu biti pohranjene u jednom spremniku (*centralizirano pohranjivanje*) ili mogu biti raspodijeljene u računalnoj okolini (*decentralizirano pohranjivanje*). Centraliziranim pohranjivanjem politike nadzora pristupa ne posti\e se svojstvo razmjernog rasta (engl. scalability), odnosno jedinstveni spremnik, u raspodijeljenim okolinama širokih razmjera, predstavljao bi usko grlo (engl. bottleneck) sustava. Ukoliko su odredbe nadzora pristupa raspodijeljene najčešće se pohranju u *LDAP* (engl. Lightweight Directory Access Protocol) [28] direktorije. No, decentralizirano pohranjivanje politike nadzora pristupa znatno otežava ostvarivanje mehanizma pronalaženje odredbi.

Prilikom donošenja autorizacijske odluke, zahtijeva se da PDP sakupi *sve* relevantne odredbe nadzora pristupa. To nameće postojanje popisa svih odredbi nadzora pristupa. U slučaju neprolaska nekih odredbi nadzora pristupa, autorizacijska odluka mora biti negativna, budući da nepronađene odredbe mogu zabranjivati izvođenje odabrane opreracije nad objektom.

Sustav upravljanja odredbama nadzora pristupa mora moći *jasno iskazati* definirane odredbe. Kada korisnik želi definirati dodatne odredbe pristupa nad nekim objektom mora raspolagati s razumljivim prikazom trenutnih odredbi. To zahtjeva kreiranje pirodnih jezika za iskazivanje odredbi ili izradu naprednih grafičkih sučelja (engl. graphic user interface, GUI).

3 Računarstvo zasnovano na uslugama

Računarstvo zasnovano na uslugama (engl. Service-Oriented Computing, SOC) je paradigma koja prepoznaje *usluge* kao osnovne elemente pri izgradnji računalnih sustava [29]. *Usluga* (engl. service) je samoopisujući (engl. self-descibing) programski element s dobro definiranim sučeljem koji svoju funkcionalnost izlaže primjenom prihvaćenih, otvorenih standarda.

Usvajanje koncepta usluge (engl. service orientation, SO) je prirodna evolucija dosadašnjih programskih paradigmi, prikazana na slici 3-1.

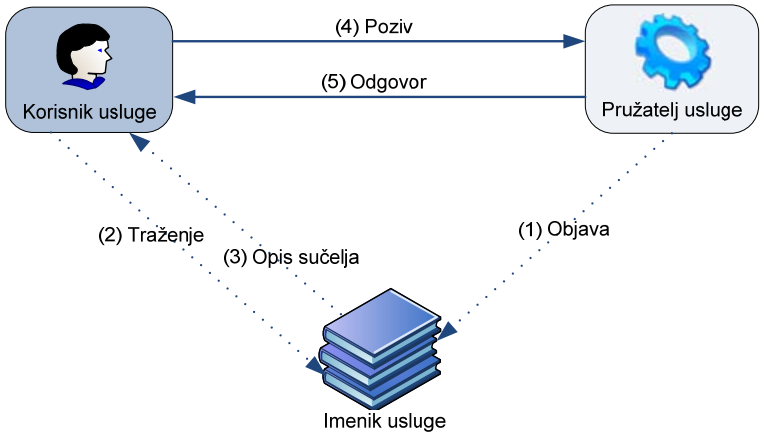
Slika 3-1: Evolucija programskih paradigmi

Razvoj složenih programskih sustava prolazio je tijekom vremena kroz nekoliko koraka. Svaki je korak uvodio određena poboljšanja koja su postupak razvoja programskih sustava činila bržim, lakšim i jednostavnijim. Početno, sustavi su se gradili kao monolitni blokovi. Kako bi se prevladale poteškoće uzrokovane porastom složenosti programske potpore, rodila se zamisao o organizaciji programskog koda u zasebne logičke cjeline (*strukturirano programiranje*). Slijedeći razvojni korak bila je primjena *objektno-orijentirane paradigme* (engl. object-oriented programming). Korištenjem objektno-orijentirane paradigme, programski sustavi se izgrađuju kao skupina *razreda*. *Razredi* (engl. class) predstavljaju modele objekta iz stvarnog života. Mehanizam stvaranja primjeraka i naslijeđivanja razreda znatno su unaprijedili razvoj programske podrške, omogućivši *višestruko korištenje* (engl. reusability) napisanog koda. Također, olakšano je i održavanje izrađenih sustava. Željene ispravke potrebno je provesti samo nad programskim kodom razreda te su one automatski vidljive na svim mjestima gdje se koriste objekti tog razreda. Upotrebom koncepta objekata omogućeno je višestruko korištenje koda unutar jednog sustava. Daljni korak u razvoju jest primjena *paradigme zasnovane na komponentama* (engl. component-based development) koja omogućuje višestruku iskoristivost koda u više nezavisnih sustava. Programska komponenta je dio programskog sustava koja obavlja dobro definiran i nezavisan skup funkcionalnosti. Novi programski sustavi grade se iskorištavanjem gotovih programskih komponenata drugih proizvođača.

Razvojem računalnih mreža javila se potreba za povezivanjem raspodijeljenih računalnih elemenata. Iako oblikovanje oblikovanje zasnovano na komponentama predstavlja zadovoljavajuće rješenje za izgradnju programskih sustava, ne odgovara potrebama razvoja raspodijeljenih sustava. Komponente svoju funkcionalnost izlažu primjenom vlasničkih (engl. propetary), najčešće nestandardiziranih protokola, što otežava njihovo povezivanje. *Računarstvo zasnovno na uslugama* prepoznaje budućnost razvoja programske podrške neovisnu o organizacijskim ili tehnološkim granicama. Usluge, koje prikrivaju tehnološka svojstva radne okoline, objavljuju svoju funkcionalnost na općeprihvaćen, standardiziran način. Prilikom izgradnje računalnih sustava pronalaze se usluge koje ostvaruju dijelove potrebe funkcionalnosti sustava. Cjelokupna funkcionalnost sustava postiže njihovim povezivanjem.

3.1 Arhitektura zasnovana na uslugama

Arhitektura zasnovana na uslugama (engl. Service-Oriented Architecture, SOA) [30] definira model pružanja i korištenja programskih funckionalnosti u obliku usluga. Osnovni elementi arhitekture zasnovane na uslugama prikazani su na slici 3-2.



Slika 3-2: Osnovni model arhitekture zasnovane na uslugama

Arhitektura zasnovanu na uslugama ostvrauje se međudjelovanjem *pružatelja usluge*, *korisnika usluge*, te *imenika usluga*. *Pružatelj usluge* (engl. service provider) izrađuje uslugu, te ju čini javno dostupnom objavljujući njeno sučelje u *imeniku usluga* (engl. services directory) (1). *Korisnik* (engl. service consumer) pretražuje imenik usluga, kako bi pronašao uslugu koja obavlja željenu funkcionalnot (2). Pri pronalasku, korisnik dobavlja opis sučelje usluge (3), koji sadrži sve informacije potrebne za korištenje usluge. Korisnik upućuje zahtjev za izvođenjem operacije iz opisa sučelja (4). Pružatelj usluge prima zahtjev, izvodi odgovarajuću operaciju i rezultat izvođenja šalje kao odgovor korisniku usluge (5).

Izgradnja sustava u skladu sa zahtjevima arhitekture zasnovane na uslugama je *iterativan* proces, prikazan na slici 3-3. Sastoji se od pronalaska već postojećih usluga ili stvaranja novih, njihove kompozicije u složenu uslugu, te objave novoizgrađene usluge radi daljnjeg korištenja.

Slika 3-3: Iterativna izgradnja sustava u skladu s zahtjevima arhitekture zasnovane na uslugama

Prva faza izgradnje uključuje prepoznavanje potrebnih funkcionalnosti sustava. Ukoliko potrebne funkcionalnosti već nisu objavljene kao usluge, tada ih je potrebno implementirati. Razvoju novih usluga može se prići s različitom razinom zrnatosti: svaku pojedinu funkcionalnost može se izložiti kao uslugu ili se skup funkcionalnosti izlaže kao jedinstvena usluga. Kada su sve potrebne funkcionalnosti raspoložive u obliku usluga nastupa njihova kompozicija u novu uslugu. Kako su usluge neovisne o implementaciji, moguće je komponirati usluge iz raznorodnih okolina. Novonastalu uslugu se objavljuje u imeniku uslugu čime ju mogu koristiti krajnji korisnici, ili ponovno ulazi u u proces izgradnje kao dio potrebne funkcionalnosti.

3.1.1 Zahtjevi arhitekture zasnovane na uslugama

Ključni zahtjevi pri izgradnja sustava u skladu s arhitekturom zasnovanom na uslugama su [31]: postizanje *slabe povezanosti* elementa sustava, *platformska neutralnost*, *prilagodljivost*, *logička smislenost* usluga, *postojanost usluga*, te ostvarivanje *ravnopravne suradnje usluga*.

Korisnik usluge treba se oslanjati isključivo na sučelje usluge, a nikako ne ovisiti o implementaciji, internoj strukturi i mehanizmima usluge. Time se ostvaruje *slaba povezanost* (eng. loose coupling), komunikacija u kojoj usluge ne pretpostavljaju više od onog što je potrebno za razmjenu poruka (tj. oslanjaju se samo na opis sučelja), smanjujući rizik da će promjena u implementaciji jedne usluge utjecati na drugu.

Korištenje usluge mora biti *neovisno od njene implementacije* (engl. implementation neutrality). Korisnik usluge ne može ovisiti o implementacijskim detaljima, već način

korištenja usluge mora biti poznat preko dobro opisanog sučelja. Također, zahtijeva se i neutralnost s obzirom na korištene razvojne alate i jezike.

Sustav mora biti lako *prilagodljiv* (engl. flexible), tj. mora biti omogućeno dinamičko konfiguriranje veza među uslugama. To zahtijeva da se usluge razvijaju neovisno o načinu kojim će biti povezane te da se povezivanje obavlja naknadno.

Nadalje, zahtjeva se *postojanost* (engl. long lifetime) usluga. Usluge nisu nužno dugog vijeka, no budući da se međudjelovanje odvija među autonomnim raznorodnim uslugama u dinamičkoj okolini, nužna je mogućnost upravljanja iznimkama. Usluge bi trebale postojati dovoljno dugo da otkriju otkriju važne iznimke, poduzmu odgovarajuće mjere i odgovore na mjere ispravljanja pogrešaka koje su poduzele druge usluge. Usluge moraju postojati dovoljno dugo da ih je moguće pronaći te pouzdati se u njih i njihovo ponašanje.

Pri oblikovanju usluga treba ih oblikovati na prirodnoj razini apstrakcije. To podrazumijeva da ne treba svaku, najmanju funkcionalnost izložiti kao uslugu, već funkcionalnosti treba ujediniti tako da tvorene *smislene logičke cjeline*. Ujedinjavanje na prirodnoj razini apstrakcije smanjuje zavisnost te time omogućuje lakše povezivanje usluga i njihovu uporabu.

Usluge moraju biti *ravnopravne* u obavljanju zadataka. Drugim riječima, umjesto hijerarhijskog modela voditelj/sluge gdje imamo usluge koje su naređene ostalima, zahtjeva se suradnja kao u timu. Oblikovanje zasnovano na timu usluga posljedica je razvoja arhitekture ravnopravnih sudionika (engl. peer to peer architecture).

3.1.2 Nadzora pristupa u arhitekturi zasnovanoj na uslugama

Arhitektura zasnovana na uslugama nameće dodatne zahtjeve [32] pri ostvarivanju nadzora pristupa pored onih razmatranih u 2.8.

Najveća prednost izgradnje sustava u skladu s zahtjevima arhitekture zasnovane na uslugama je slaba povezanost usluga. Usluge su autonomne, te se zahtijeva da sustav za nadzor pristupa ne ugrožava njihovu autonomnost. Prema tome, usluge, odnosno pružatelji usluga, moraju imati *potpunu nezavisnost pri definiranju i provođenju odredbi nadzora pristupa*.

Sustavi ostvareni u skladu s načelima arhitekture zasnovane na uslugama nalaze se u dinamičkom okruženju u kojem se usluge neprestano stvaraju, ukljanjaju, mjenjaju i udružuju. Stoga, primjenjeni sustav nadzora pristupa mora biti *prilagodljiv* na učestale promjene.

Osim samog pristupa usluzi, mora biti moguće i nadzirati pozive metoda usluga, kao i poruke koje usluge izmjenjuju. Zahtjeva se *izražajnost* politike nadzora pristupa, odnosno politikom mora biti moguće specificirati koje metode usluga određeni korisnik smije pozvati, te kakav smije biti sadržaj poruke namjenjene toj metodi.

Dodatno, zahtjeva se ostvarenje *mehanizma praćenja korištenja* (engl. audit trail) *usluga*. Arhitektura zasnovana na uslugama najveću primjenu ima u poslovnim okruženjima u kojima pružatelji usluga naplaćuju njihovo korištenje. Sustav nadzora pristupa, osim što omogućuje definiranje i provođenje odredbi nadzora pristup, mora ostvariti mehanizme vođenja i prikaza evidencije o korištenju usluga na osnovi koje se izvodi naplatu.

Tradicionalne politike nadzora pristupa nisu sasvim prikladne za primjenu u računarstvu zasnovanom na uslugama. Mandatno pravo pristupa, kako je formalizirano Bell-LaPadulinim modelom, uvodi ograničenje konstantnosti sigurnosnih razina čime nije zadovoljen zahtjev prilagodljivosti na učestale promjene. Nadalje, mandatno pravo pristupa podrazumijeva postojanje središnjeg upravitelja (engl. central administration) koji dodjeljuje sigurnosne razine što nije u skladu s zahtjevom autonomnosti usluga. Diskrecijsko pravo pristupa provodi nadzor na osnovi matrice pristupa. Iako postoje razne implementacije matrice pristupa razmatrane u 2.6, zahtjev diskrecijskog prava pristupa da se definiraju prava pristupa za svaku trojku $(subjekt, operacija, objekt)$ ograničava njegovu primjenu u prisustvu velikog broja korisnika i usluga. Pravo pristupa zasnovano na uslugama implicitno podrazumijeva stalnost uloga, odnosno pretpostavlja se da je skup dozvola pridružen ulozi gotovo nepromjenjiv u vremenu. U dinamičkom okruženju arhitekture zasnovane na uslugama, potrebno je osmisliti mehanizme dinamičkog pridruživanja *uloga-dozvole* [33].

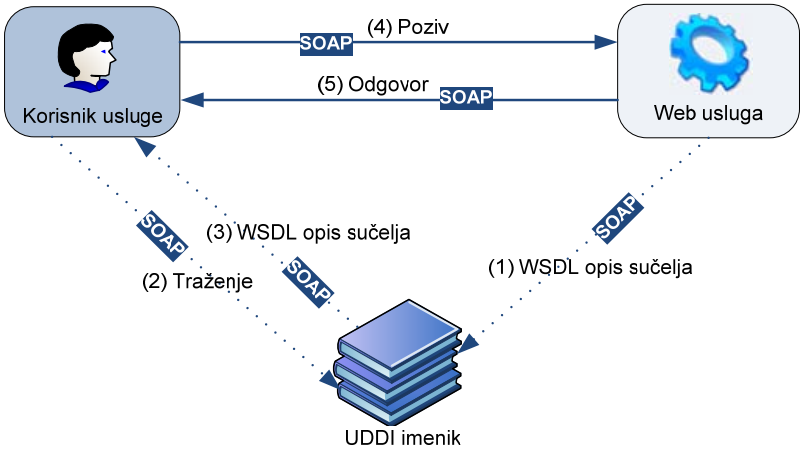
3.2 Web usluge

Arhitektura zasnovana na uslugama je koncept koji je moguće ostvariti korištenjem različitih tehnologija, programskih alata i proizvoda. Prevladavajući skup standardnih tehnologija i protokola koji omogućuju izgradnju sustava u skladu sa zahtjevima arhitekture zasnovane na uslugama poznat je pod nazivom *Web usluge* (engl. Web Services). Standardi, specifikacije i preporuke koje opisuju tehnologiju Web usluga su namjerno modularni i potiču iz raznih organizacija, čime je postignuta njihova široka prihvaćenost. W3C, jedna od organizacija koje istražuju navedenu tehnologiju, definira *Web uslugu* kao [34]:

> *Programski sustav osmišljen tako da podržava interakcije među računalima preko raznih računalnih mreža. Ima sučelje koje je opisano na računalno obradiv način*

(WSDL). Interakcija s ostalim sustavima obavlja se na način opisan sučeljem, putem SOAP poruka koje se presnose korištenjem standardiziranih Internet protokola.

Iako ne postoji jedinstveni dokument koji opisuje Web usluge, već skup specifikacija koje se međusobno nadopunjuju, postoji koncenzus akademske i industrijske zajednice oko osnovnih protokola i tehnologija Web usluga. To su *XML*, *SOAP*, *WSDL* i *UDDI*.



Slika 3-4: Primjena osnovnih standarda Web usluga

Slika 3-4 prikazuje na koji način gradimo sustav zasnovan na uslugama primjenom tehnlogije Web usluga. Sva komunikacija u takvom sustavu zasniva se na SOAP protokolu. Pružatelj usluge razvija uslugu i pruža njeno sučelje putem WSDL opisa koji se dostavlja UDDI imeniku (1). Da bi korisnik mogao koristiti uslugu, informacije o usluzi traži od imenika (2) koji onda korisniku šalje WSDL opis tražene usluge (3). Na osnovi dohvaćenog WSDL opisa korisnik onda može primjenom protokola SOAP pozvati uslugu pružatelja usluge (4) koji mu nakon izvođenja vraća rezultate (5).

U nastavku je dan kratki pregled spomenutih osnovnih standarda: XML, SOAP, WSDL, UDDI.

XML

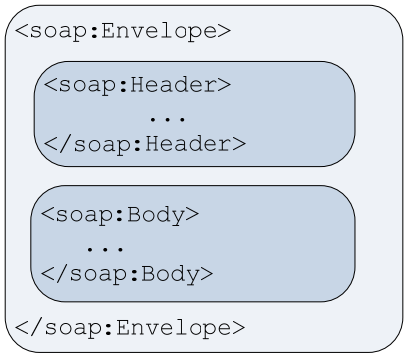
XML (engl. eXtensible Mark-up Language) [35] je proširivi jezik zasnovan na *oznakama* koji pruža tekstualni format zapisivanja podataka neovisan o računalnoj platformi. XML je osnovni jezik na kojeg se oslanjaju drugi standardi Web usluga. Za razliku od HTML-a gdje oznake određuju način prikazivanja podataka, XML oznake nose informaciju o značenju podataka. Također, za razliku od skupa oznaka i svojstava jezika HTML koji je unaprijed definiran, skup oznaka i svojstava jezika XML moguće je definirati i proširiti ovisno o potrebi.

Skup oznaka koji koristi određeni XML dokument naziva se *rječnik* (engl. vocabulary) XML dokumenta. S obzirom da je razvijeno mnoštvo nezavisnih rječnika XML dokumenata, moguće je da oznake i atributi istih imena posjeduju različito značenje u različitim rječnicima. Da bi se izbjeglo miješanje i zabuna uvodi se pojam *prostora imena* (engl. namespace) koji se zadaje pomoću URI-a (engl. uniform resource indentifier). Smještanjem u zadani prostor imena svaki element i svaki atribut jednoznačno su određeni.

SOAP

SOAP je jednostavni (engl. lightweight) komunikacijski protokol namijenjen razmijeni strukturiranih informacija u distribuiranoj okolini [36]. Koristi XML tehnologuju, te omogućuje razmjenu poruka neovisnu o prijenosnom protokolu. SOAP je osmišljen s namjerom da bude nezavisan s obzirom na korišteni programski model.

Osnovni zahtjevi pri izradi protokola bili su jednostavnost i proširivost. To je ostvareno na način da sam standard ne specificira mnoge uobičajene zahtjeve komunikacije u distibuiranim okolinama, poput pouzdanosti, sigurnosti, kvalitete usluga i slično, već su oni (ili će tek biti) definirani dodatnim uredbama. Slično, standard ne specificira prijenosne protokole koji se trebaju koristiti pri prijenosu SOAP poruka. Navode se tek uredbe koje dodatne specifikacije o povezivanju (engl. binding) SOAP poruka s određenim prijenosnim protokolu moraju zadovoljavati. Jedna od takvih dodatnih specifikacija je *SOAP HTTP Binding*, koja opisuje upotrebu HTTP protokola za prijenos SOAP poruka.



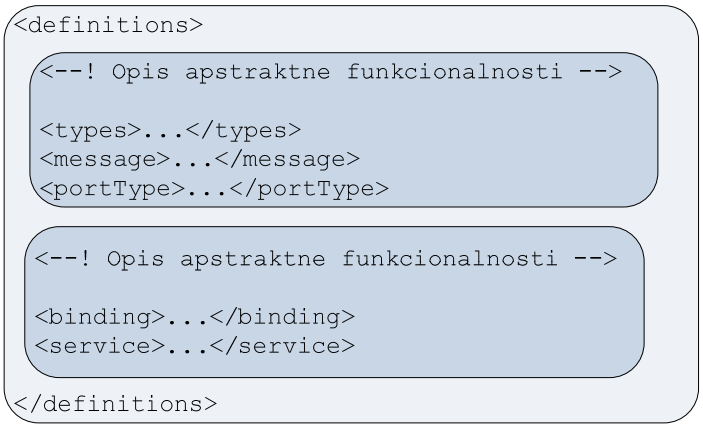
Slika 3-5: Prikaz SOAP poruke

Protokol se zasniva na razmijeni XML poruka, pri čemu se svaka poruka sastoji od tri elementa: omotnice (engl. envelope), zaglavlja (engl. header) i tijela (engl. body) poruke. Prikaz poruke dan je na slici 3-5. Omotnica je korijenski element koji se sastoji od neobaveznog zaglavlja i obaveznog tijela poruke. Zaglavlje je neobavezni element na kojem se zasniva proširivost SOAP protokola, budući da se (za razliku od drugih protokola

namijenjenih prijenosu) aplikacijskoj logici dopušta pristup zaglavlju. Dodvanjem i izuzimanjem elementa zaglavlja omogućuje se evolucija protokola.

WSDL

WSDL (engl. Web Services Description Language Version) pruža model za opisivanje Web usluga zasnovan na XML tehnologiji [37]. WSDL dokument sastoji se od dvije semantičke cijeline: opis apstraktne funkcionalnosti usluge i konkretnih uputa za korištenje usluge (slika 3-6). Pri opisu apstraktne funkcionalnosti usluge definiraju se apstraktni tipovi podataka (element *types*), apstraktne poruke koje se izmjenjuju (element *message*), te skup operacija koje usluga nudi (element *portType*). Konkretne upute za korištenje imenuju prijenosni protokol koji se koristi za pristup usluzi (element *binding*) i opisuje pristupna točka usluge (element *service*). WSDL dokumentom nemoguće je u potpunosti opisati sve aspekte usluga (npr. sigurnosi zahtjevi, dostupnost usluge, itd.). To se rješava objavljivanjem dodatnih specifikacija (*WS-Policy*) koje nadopunjuju osnovne mogućnosti WSDL-a.



Slika 3-6: Prikaz WSDL dokumenta

UDDI

UDDI (engl. Universal Description Discovery & Integration) definira skupa mehanizama koji omogućuju opis i pronalazak poslovnih organizacija, institucija i ostalih ponuđača usluga, usluga koje oni nude te opisa sučelja usluga radi njihova korištenja [38]. UDDI se zasniva na općeprihvaćenim standardima poput XML-a, XML Schema-e i SOAP-a. Standardom se definira način pristupa UDDI katalogu, koji je prema analogiji s telefonskim imenikom podijeljen na bijele, žute i zelene stranice. Bijele stranice (engl. white papers) sadrže podatke o pružateljima usluga. Žute stranice (engl. yellow papers) sadrže podatke o kategoriji usluga. Zelene stranice (engl. green papers) sadrže tehničke podatke o uslugama.

3.2.2 Standardi Web usluga

Web usluge objedinjuju standardne tehnologije i protokole koji omogućuju izgradnju sustava u skladu sa zahtjevima arhitekture zasnovane na uslugama. Ne postoji proces usvajanja standarda koji čine Web usluge, već različite organizacije (akademske i poslovne) objavljuju vlastite specifikacije i preporuke. Iako se takve specifikacije često isprepliću, takav modularan pristup tehnologiji Web usluga omogućava njen dinamičan i ubrzan razvoj. Slika 3-7 prikazuje organizacije koje su objavile najviše standarda Web usluga [39].

Slika 3-7: Organizacije koje su najviše pridonijele razvoju Web usluga

IETF (engl. Internet Engineering Task Force) [40] je velika otvorena međunarodna zajednica arhitekata računalnih mreža, operatera, proizvođača i istraživača koji proučavaju evoluciju Internet arhitekture.

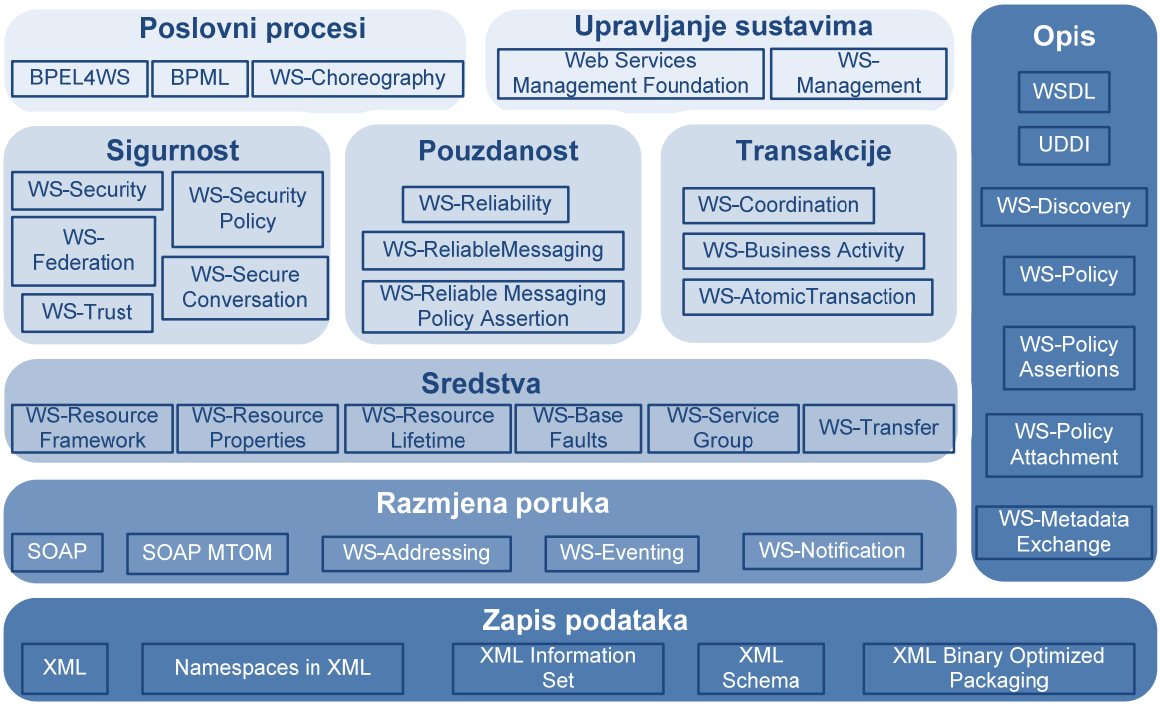
W3C (engl. World Wide Web Consortium) [41] je osnovan u listopadu 1994. kao tijelo koje podupire razvoj World Wide Weba-a objavljivanjem i standardizacijom protokola. W3C ima preko 350 organizacija članica iz čitavog svijeta i međunardno je priznata za doprinost rastu Web tehnologija. U rujnu 2000., W3C je započeo project pod nazivom "XML Protocol Activity" s ciljem razvijanja protokola za razmjenu poruka zasnovanog na XML jeziku. Naslijednik tog projekta jest istraživanje "Web Services Activity".

OASIS (engl. Organization for the Advancement of Structured Information Standards) [42] je neprofitna, međunarodna organizacija koja teži razvoju i primjeni standarda e-poslovanja. OASIS izdaje više standarda Web usluga nego ikoja druga organizacija. Osnovana je 1993., i ima više od 600 članova, što organizacija što individualnih članova, u više od stotinu zemalja.

WS-I (engl. Web Services Interoperability Organization) [43] je otvorena organizacija vodećih proizvođača računalnih sustava koji zajedničkim naporima pokušavaju postići

koncenzus oko primjene postojećih standarda Web usluga i osmišljavanju novih u svrhu postizanja nezavisnosti s obzirom na korištene platforme, operacijske sustave i programske jezike. Gotovo svi glavni proizvođači programske podrške (Microsoft, IBM, BEA, Ericsson, Sun, SAP, HP, Intel i ostali) članovi su ovog udruženja.

Navedene organizacije objavile su brojne specifikacije i preporuke [44] u svrhu nadopunjavanja tehnologije Web usluga. Specifikacijama se definiraju razni aspekti Web usluga, koji se mogu sagledati kao slojevita arhitektura[1] prikazana na slici 3-8.



Slika 3-8: Standardi Web usluga

Osnovni sloj ove arhitekture definira način zapisa (engl. encoding) informacija. U svijetu Web usluga podaci se zapisuju u skladu s XML standardima. Osim specifikacija koje definiraju XML kao zapis informacija u tekstualnom obliku, razvijen je *XML Binary Optimized Packaging (XOP*) standard koji omogućuje efikasan prijenos informacija zapisanih binarnim zapisom u skladu s zahtjevima XML jezika.

Komunikacija među Web uslugama odvija se razmjenom *SOAP* poruka. *WS-Addressing* omogućuje jedinstveno određivanje Web usluga i SOAP poruka neovisno o korištenom prijenosnom protokolu, oslanjajući se na proširivost SOAP-a. *WS-Notification* i *WS-Eventing* podržavaju naprednije oblike komunikacije poput pretplati/objavi (engl.

[1] Iako prikaz na slici ne zadovoljava zahtjeve slojevite arhitekture, jer se definirani slojevi ne oslanjaju samo na one neposredno ispod njih, korištena je ovakva prezentacija da bi se naznačili odnosi između mnoštva specifikacija.

publish/subscribe) mehanizma. *SOAP Message Transmission Optimization (MTOM)* nudi optimirani prijenos netekstualnih podataka korištenjem XOP-a.

Specifikacije kojima je zadatak ponuditi načine opisivanja svojstava Web usluge, poput ponuđenih operacija, korištenih tipova podataka, opisa kvalitete i dostupnosti, način pronalaska i slično, izdvojene su u zasebni vertikalni sloj budući da njegove usluge koriste svi slojevi. Ovdje ubrajamo ranije spomenuti *WSDL* kojim se opisuju osnovna svojstva Web usluge. Uredbama *WS-Policy*, *WS-PolicyAssertions* te *WS-PolicyAttachment* omogućava se opisivanje dodatnih svojstava poput sigurnosnih zahtjeva, raspoloživosti i kvalitete usluge. *UDDI*, *WS-Discovery* i *WS-MetadataExchange* definiraju mehanizme pronalaženja opisa Web Usluga.

Web usluge često se koriste za pristup sredstvima sa stanjem (engl. resource). Neki od primjera sredstava su baza podataka ili pak proces koji „pamti“ koliko je puta pozvan. Specifikacije sloja sredstva opisuju uporabu sredstava putem Web usluga, čim se efektivno omogućuje izrada Web usluga s pohranom stanja, poznatih kao *WS-Resources*.

Mehanizmi razmjene poruka nisu dovoljni da bi se Web usluge široko koristile i u stvarnom, poslovnom sjetu, već treba osigurati određena jamstva pri njihovom izvođenju, poput sigurnosti, pouzdanosti pri komunikaciji, i korektnog izvođenje transakcija. Slojem sigurnosti obuhvaćene su specifikacije koje pružaju povjerljivost i integritet poruka (*WS-Security*, *WS-SecurityPolicy*, *WS-SecureConversation*) te uspostavu povjerenja među Web uslugama (*WS-Trust*). Dodatno, omogućava se i uspostava zaštićene domene između više organizacija (*WS-Federation*). Sloj pouzdanost definira specifikacije kojima je zadaća omogućiti pouzdani prijenos poruka, tj. pružiti mehanizme koji osiguravaju da se svaka poruka isporučuje točno jednom te pravilan poredak među porukama. Transakcijski sloj definira protokole potrebne za primjenu Web usluga u poslovnim okruženjima. Definira se mehanizam koordiniranja Web usluga (*WS-Coordination*), koji podupire specifikacija *WS-AtomicTransaction* omogućavajući upravljanje transakcijama korištenjem poznatih protokola poput protokola dvostrukog potvrđivanja (engl. Two-Phase Commit, 2PC), i *WS-BusinessActivity* definirajući protokole za dugoobavljajuće (engl. long-running) transakcije.

Najviše slojeve čine specifikacije koje definiraju organiziranje poslovnih procesa i upravljanje sustavima. U sloju poslovni procesi (engl. business processes) najznačajnija specifikacija jest *Business Process Execution Languague for Web Services (BPEL4WS)*. BPEL4WS je široko prihvaćeni izvršni jezik za organizaciju poslovnih procesa korištenjem Web usluga. Sloj upravljanja sustavima (engl. management) obuhvaća specifikacije *Web*

Services Management Foundation (WSMF-Foundation) i *WS-Management* kojima se definiraju standardni načini administriranja i upravljanja računalima, uređajima, Web uslugama, aplikacijama, i drugim upravljivim sredstvima (engl. manageable resources), korištenjem Web usluga.

4 Postojeći sustavi nadzora pristupa u raspodijeljenim računalnim okolinama

U ovom poglavlju opisani su postojeći standardi kojima se opisuju protokoli komunikacije između glavnih entiteta arhitekture sustava nadzora pristupa definiranih u poglavlju 2.8.1, te je dan pregled nekoliko sustava nadzora pristupa u raspodijeljenim računalnim okolinama. *PERMIS* se oslanja na infrastrukturu za upravljanje svojstvima kao mehanizam za raspodijeljivanje politike nadzora pristupa i svojstva korisnika. *Akenti* omogućava nadzor pristupa sa svojstvom razmjernog rasta (engl. scalablity) u prisustvu više upravitelja objekta. *Cardea* pokušava ostvariti jedinstveni sustav nadzor pristupa nad više administrativnih domena korištenjem standarda *SAML* i *XACML*.

4.1 Standardi nadzora pristupa

Osim arhitekture sustava nadzora pristupa, potrebno je definirati i podatke te protokole pomoću kojih komuniciraju entiteti sustava. Organizacija OASIS osmislila je dvije usko povezane specifikacije kojima pokušava ponuditi standardne jezike i protokole sustava nadzora pristupa. To su *SAML* i *XACML* koji su detaljnije opisani u nastavku.

4.1.1 SAML

Security Assertion Markup Language (SAML) [45] standard definira okvir razmjene sigurnosnih izjava između entiteta računalnih sustava. Cilj osmišljavanja SAML-a jest *"definirati standardni okvir za stvaranje i izmjenu autentikacijskih i autorizacijskih podataka, korištenjem XML tehnologije"* [46].

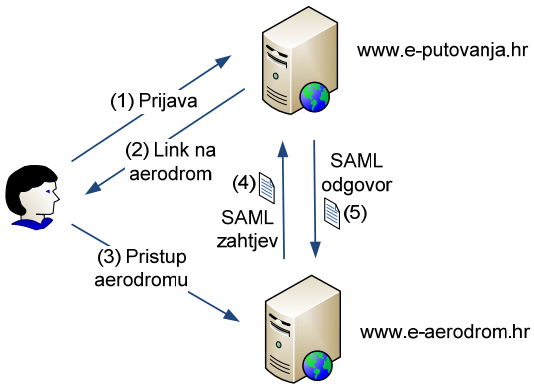
Glavni pojmovi definirani SAML standardom su *izjave*, *protokoli*, *načini povezivanja* te *profili*. *Izjave* (engl. assertions) izdaje *SAML autoritet* (engl. SAML authority) koji uživa povjerenje ostalih entiteta sustava (sam mehanizam uspostave povjerenja nije definiran specifikacijom). Moguće su tri vrste izjava: *autentikacijske izjave*, *izjave svojstva* i *autorizacijske izjave*. *Autentikacijske izjave* (engl. authentication assertions) izdaje entitet sustava koji provodi autentikaciju korisnika, navodeći identitet autenticiranog korisnika te ostale autentikacijske informacije poput vremenskog okvira u kojem je izjava važeća. *Izjavama svojsta* (engl. attribute assertions) vežu se određena svojstva uz korisnika, primjerice njegova uloga. *Autorizacijske izjave* (engl. authorization decision assertions) iskazuju ovlasti koje korisnik ima u sustavu.

SAML standardom je definirano nekoliko *zahtjev/odgovor protokola* (engl. request/response protocols) pomoću kojih entiteti sustava potražuju od SAML autoriteta

jednu ili više sigurnosnih izjava. Sigurnosne izjave moguće je prenositi pomoću različitih komunikacijskih protokola. O tome se brine dio specifikacije u kojem su definirani *načini povezivanja* (engl. binding) SAML poruka sa standardnim protokolima. Primjerice *SAML SOAP Binding* definira način na koji se SAML izjave prenose SOAP-om. Konačno, definirano je nekoliko *profila* (engl. profiles), tj. konkretnih načina uporabe SAML-a. Jedan od njih je *Web SSO Profile* kojim se navodi način korištenja SAML prilikom ostvarivanja mehanizma *jedinstvene uspostave sjednice* (engl . Single Sign On, SSO).

Ostvarivanje programskih sustava u sukladnosti s SAML specifikacijama ima brojne prednosti [47]. SAML pruža standardan način zapisivanja i razmjene podataka o korisniku, čime se postiže *nezavisnost s obzirom na aplikacijsku logiku*. Upotrebom SAML-a moguće je ostvariti zahtjev arhitekture zasnovane na uslugama: *slabu povezanost* entiteta sustava. Prihvaćanjem standardnog načina razmjene korisničkih podataka dijelovi sustava ne ovise o konkretnim autentikacijskim i autorizacijskim tehnikama.

Omogućeno je ostvarenje *jedinstvene uspostave sjednice*. Jedinstvena uspostava sjednice omogućuje korisniku prijelaz između različitih programskih sustava, pri čemu se autentikacijski podaci izmjenjuju u pozadini, bez interakcije s korisnikom (ali s njegovim dopuštenjem).



Slika 4-1: Ostvarivanje mehanizma jedinstvene upostave sjednice pomoću SAML-a

Na slici 4-1 je prikazan jednostavan primjer uporabe SAML autentikacijskih izjava prilikom ostvarivanja mehanizma jedinstvene uspostave sjednice. Korisnik je zainteresiran za putovanje i prijavljuje se na internetsku stranicu www.e-putovanja.hr (1). Prilikom pristupa stranici korisnik se prijavljuje (autenticira) koristeći svoje korisničko ime i šifru. Zahvaljujući prepoznavanju korisnika internetska stranica pronalazi putovanja koja će se vjerojatno svidjeti korisniku na osnovu njegovih navika. Korisnik izabire putovanje koje uključuje let avionom, te mu internetska stranica putovanja dojavljuje stranicu aerodroma

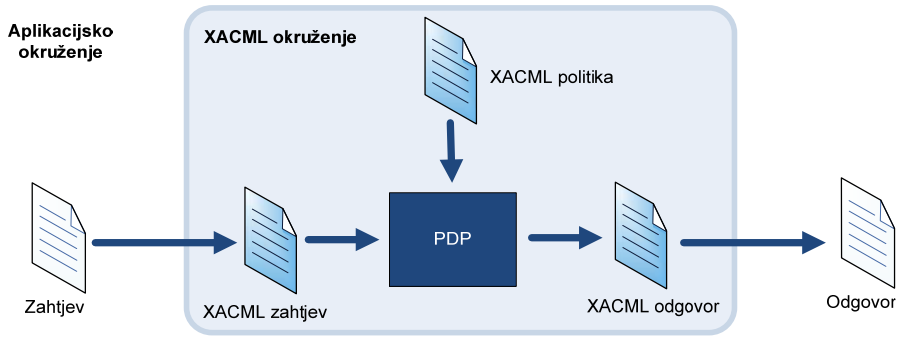
(2). Link na aerodrom uključuje i dodatnu tekstualnu oznaku (engl. query string) pomoću koje internetska stranica www.e-aerodrom.hr prepoznaje korisnika (3). Internetska stranica aerodroma vjeruje stranici putovanja te šalje SAML zahtjev za korisničkim podacima. Nakon što dobiva odgovor (5), pouzdano zna o kojem korisniku se radi te mu omogućuje rezervaciju leta.

Podržavanjem mehanizma jedinstvene uspostave sjednice više nije potrebno da različiti programski sustavi bilježe autentikacijske podatke o korisniku. Time se *smanjuju troškovi administracije*. Nadalje, odgovornost za pouzdanu autentikaciju korisnika sada preuzimaju posebni entiteti. Njihova je zadaća osigurati korektne autentikacijske i autorizacijske tehnike. Dijelovi sustava koji implementiraju aplikacijsku logiku koriste njihove usluge, čime se oslobađaju odgovornosti upravljanja korisničkim indetitetima.

SAML ima mnoge primjene, od ranije spomenutog ostvarivanje jedinstvene uspostave sjednice, korištenja u Web uslugama (WS-Security standard predviđa korištenje SAML izjava pri osiguravanju poruka koje usluge međusobno izmjenju), do mogućnosti suradnje s *XACML*-om prilikom ostvarivanja sustava nadzora pristupa.

4.1.2 XACML

XACML (engl. eXtensible Access Control Markup Language) [48] je jezik za opisivanje politike nadzora pristupa, autorizacijskih zahtjeva i odluka zasnovan na XML-u . U velikim organizacijama čest je slučaj da zasebni dijelovi organizacije opisuju politiku nadzora pristupa na različite načine. Time se otežava administriranje odredbi nadzora pristupa i izricanje cjelokupne sigurnosne politike organizacije. OASIS je osmislio XACML s ciljem standardizacije opisivanja politika nadzora pristupa. Popularizaciji jezika pridonio je Sun Microsystem Inc. slobodnom progamskom (engl. open source) implementacijom XACML standarda.



Slika 4-2: Prikaz XACML okruženja

XACML specifikacija pokušava standardizirati dio sustava nadzora pristupa prikazanog na slici 4-2. Općenitost XACML-a postiže se uvođenjem koncepta *XACML okruženja*. Definiraju se jezici za opisivanje autorizacijskih zahtjeva i odgovora te jezik definiranja politike. Time su na jedinstven način opisani svi ulazi i izlazi PDP-a, što omogućuje njegovu lakšu izradu i povezivanje s ostalim dijelovima sustava nadzora pristupa. Prilikom pregledavanja XACML zahtjeva, PDP konzultira sigurnosnu politiku opisanu u skladu s XACML-om, te na osnovi nje definira odgovor.

Zahtjev korisnika najčešće sadrži opis korisnika, poput imena i pripadne grupe, te zahtjev za pristupom. Sam mehanizam stvaranja korisničkog zahtjeva i odgovora te njihova pretvorba u zahtjeve i odgovore definiran XACML-om, tj. prijelaz iz aplikacijskog okružena u XACML okruženje, nije definiran OASIS-ovim standardom već se koriste komplementarne specifikacije poput SAML-a. Iako postoji mnogo jezika koji pružaju mogućnost opisivanja politika nadzora pristupa, prednosti XACML su njegova *standardiziranosti*, *općenitosti*, *distribuiranosti* i *izražajnost* [49].

XACML je *standardiziran jezik*, što znači da postoji koncenzus široke zajednice računalnih profesionalaca i korisnika oko njegovih svojstava. Njegovom uporabom ubrzava se proces osmišljavanja nadzora pristupa pri izradi novih sustava (nije potrebno osmišljavati vlastiti jezik). Također, sve većom uporabom XACML-a olakšava se međudjelovanje različitih aplikacija koje koriste isti, standardni jezik.

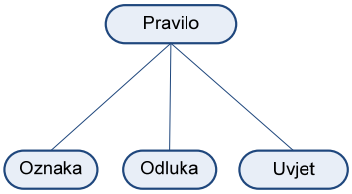
XACML nije fokusiran na nadzor pristupa specifičnim objektima ili u specifičnim okruženjima već teži biti *općenit*, sveobuhvatan. Jednom definirana politika može se koristiti za više aplikacija.

Politika nadzora pristupa može se *raspodijeliti*, tj. biti opisana u više datoteka koje se nalaze na različitim lokacijama. Zahtjev razmjernog rasta sustava nameće da umjesto uređivanja jedne, monolitne sigurnosne politike, korisnici ili grupe korisnika mogu zasebno administrirati i pohranjivati dijelove politike. XACML definira mehanizme koji to omogućuju.

Iako XACML standard dopušta mnoga proširenja, već ovako definiran jezik je vrlo *izražajan*, tj. omogućuje iskazivanje širokog skupa politika nadzora pristupa. Standardom su definirani brojni tipovi podataka, funkcije i pravila o kompoziciji različitih politika.

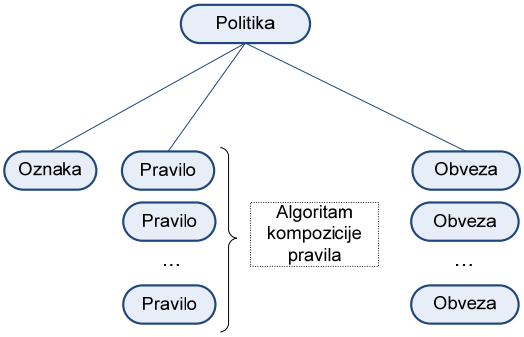
Jezik definiranja politike

Elementi jezika definiranja politike su *pravilo*, *politika* te *skup politika*. Ovdje je naveden njihov kratak opis, bez navođenja preciznih sintaksnih pravila XACML-a.



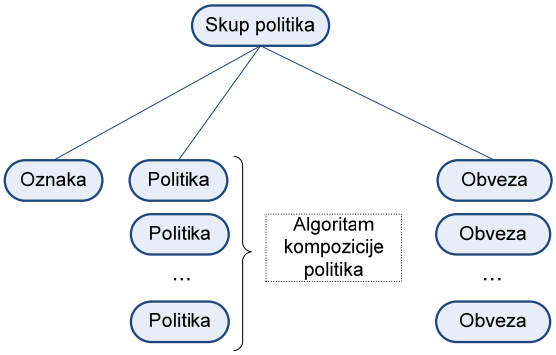
Slika 4-3: Prikaz elementa *pravilo*

Osnovni element politike nadzora pristupa definirane XACML standardom jest *pravilo* (engl. rule). Svako pravilo ima *uvjet* (engl. condition), *odluku* (engl. effect) te *oznaku* (engl. target), kao što je prikazano na slici 4-3. Prilikom nadziranja zahtjeva za pristupom evaluira se *uvjet* i na osnovi njega se donosi *odluka*. Smislena politika nadzora pristupa najčešće zahtjeva definiranje više pravila. Odluka o tome koje pravilo pristupa se uvažava donosi se na osnovi *oznake*. Oznaka se obično određuje navođenjem subjekata, objekata i načina pristupa na koje se pravilo odnosi. Slika



Slika 4-4: Prikaz elementa *politika*

Politika (engl. policy) se definira skupom pravila, *algoritmom kompozicije pravile*, *skupom obveza*, te oznakom (slika 4-4). Kako je moguće navesti više pravila, potrebno je definirati *algoritam kompozicije pravila* (engl. rule combining algorithm) kojim se određuje konačan rezultat na osnovi odluka pojedinačnih pravila. Standardom je definirano nekoliko algoritama poput *algoritma negativne prednosti* (engl. deny-overrides) ili *algoritma pozitivne prednosti* (engl. permit-overrides). Administrator sigurnosne politike može definirati *obveze* (engl. obligations) koje se moraju dodatno ispuniti prije samog pristupa. Nakon što PDP evaluira zahtjev za pristupom, obveze se dojavljuje PEP-u koji ih mora izvršiti prije dopuštanja pristupa. XACML-om je moguće definirati *skup politika*, čime se pruža mogućnost definiranja distribuirane politike nadzora pristupa. Odluka o tome koja politika pristupa se uvažava, slično kao i kod pravila, donosi se na osnovi oznake.



Slika 4-5: Prikaz elementa *skup politika*

Korijenski element sigurnosne politike definirane XACML standardom je *skup politika* (engl. policy set). Skup politika sadrži oznaku, politike, *algoritam kompozicije politika*, te obveze (slika 4-5). Slično kao kod kompozicije pravila, standardom su definirani algoritmi kompozicije politika (engl. policy combining algorithm), kojima se definira konačan rezultat određenog zahtjeva za pristupom.

Precizan opis jezika, kao i točnu definiciju njegovih XML elemenata moguće je pronaći u [48].

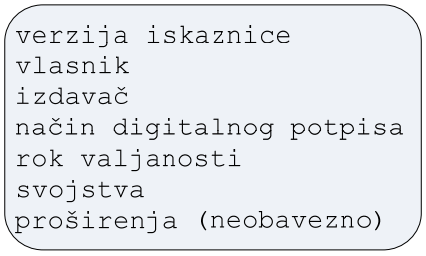
4.2 PERMIS

PERMIS [50] je projekt Europske komisije (engl. European Commission) kojim se želi razviti raspodijeljen sustav nadzora pristupa zasnovan na ulogama koji će biti primjenjiv u različitim programskim okruženjima. U projektu sudjeluju tri europska grada: Barcelona (Španjolska), Bolonja (Italija) i Salford (Velika Britanija) koja razvijaju tri različite aplikacije. Razvijene aplikacije će poslužiti za testiranje i poboljšanje PERMIS-a.

PERMIS se oslanja na *infrastrukturu za upravljanje svojstvima* (engl. Privilege Management Infrasturcture, PMI). Slično kao što *infrastruktura javnog ključa* (engl. Public Key Infrastructure, PKI) omogućuje pouzdanu razmjenu javnih ključeva u raspodijeljenim okolinama, infrastruktura za upravljanje svojstvima treba omogućiti pouzdano upravljanje korisničkim svojstavima.

Glavna zadaća infrastrukture javnog ključa jest omogućiti autentikaciju u raspodijeljenim sustavima, koristeći digitalno potpisivanje. Infrastruktura za upravljanje svojstvima pruža mogućnost ostvarenja raspodijeljenog nadzora pristupa, te ima brojne sličnosti s infrastrukturom javnog ključa. Kod PKI-ja se iskaznicom javnog ključa (engl. Public Key Certificate, PKC) ostvaruje čvrsto povezivanje korisnika i njegova javnog ključa, dok se kod PMI-ja iskaznicom svojstva (engl attribute certificate, AC) povezuje korisnik i pripadajuće mu svojstvo. Entitet sustava koji izdaje i digitalno potpisuje iskaznice u PKI sustavu se

naziva izdavatelj iskaznica (engl. Certificate Authority, CA), dok iskaznice svojstva u PMI-u izdaje izdavatelj svojstava (engl. Attribute Authority, AA).

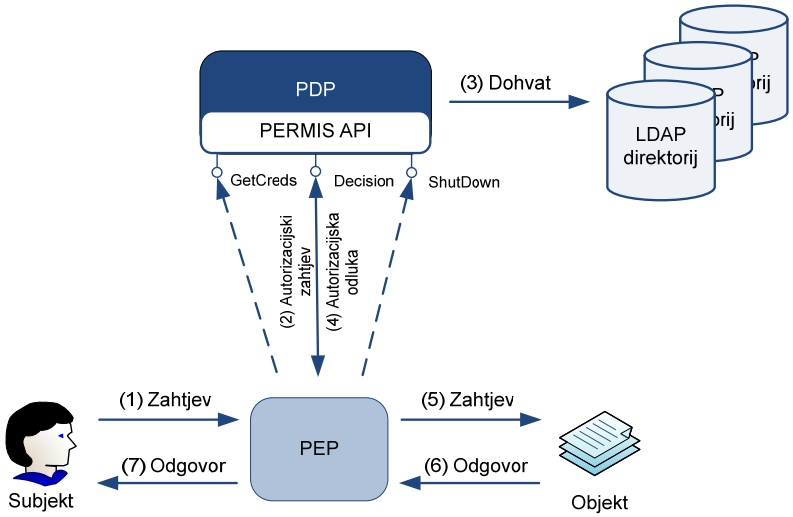


Slika 4-6: Elementi X.509 iskaznice svojstva

Standardni način objave svojstava korisnika su X.509 iskaznice svojstva (engl. X.509 attribute certificate, X.509 AC). One su vrlo slične X.509 iskaznicama javnog ključa, tek je javni ključ zamijenjen svojstvima. Prikaz elemenata iskaznice svojstva dan je na slici 4-6. PERMIS koristi nadzor pristupa zasnovan na ulogama, te se korištenjem X.509 iskaznica svojstava definiraju uloge korisnika. Politika nadzora pristupa također se zapisuje korištenjem X.509 iskaznica svojstva. Za potrebe projekta razvijen je zasebni jezik zasnovan na XML-u koji omogućuje definiranje pravila nadzora pristupa zasnovanog na ulogama. Potpuni opis jezika moguće je pronaći u [51]. Iskaznice su digitalno potpisane od strane ovlaštenih izdavatelja, te se time osigurava njihova autentičnost i integritet.

4.2.1 Arhitektura

PERMIS je razvijan u skladu s IETF radnim okvirom, tj. funkcionalnost je razdvojena u dvije povezane komponente, PEP i PDP (Slika 4-7). Dok je PEP je aplikacijski zavisan i zadužen za provjeru identiteta korisnika (autentikaciju), PDP uvidom u svojstva korisnika i politiku nadzora pristupa odgovara na zahtjev pristupa. PERMIS ne definira način autentikacije korisnika; primjerice može se upotrebljavati infrastruktura javnog ključa ili jednostavan mehanizam korisničkog imena i šifre. PERMIS pokušava standardizirati PDP definiranjem njegovog programskog sučelja. *PERMIS PMI API* predstavlja pojednostavljenje programskog sučelja kojeg je predložila Open Group organizacija, AZN API-ja [52]. Sučelje je specificirano koristeći Java programski jezik.



Slika 4-7: Arhitektura PERMIS sustava

PERMIS PMI API izlaže tri metode: *GetCreds*, *Decision* i *Shutdown* i konstruktor. Konstruktor gradi PERMIS API Java objekt. PEP poziva konstruktor navodeći ovlaštene izdavatelje iskaznica i popis LDAP direktorija gdje su pohranjene iskaznice koje definiraju politiku nadzora pristupa i uloge korisnika. Lokacija iskaznice politike navodi se prva na popisu. Nakon uspješne autentikacije korisnika, PEP poziva metodu *GetCreds*. PDP, na osnovu popisa LDAP direktorija, prikuplja sve iskaznice povezane s korisnikom. Nakon provjere njihove valjanosti, PDP izrađuje skup svih korisnikovih uloga. Metoda *Shutdown* služi za uništavanje trenutne instance PERMIS PMI API-ja.

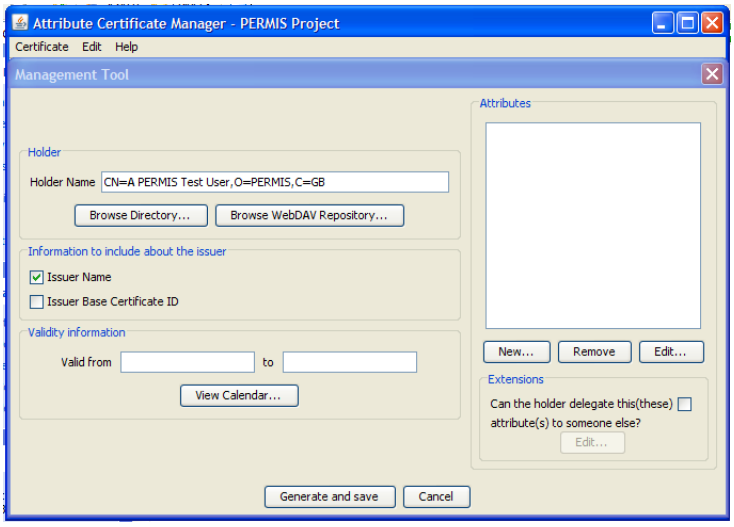
Scenarij obrade korisničkog zahjeva prikazan je na slici 4-7. Kada korisnik zatraži pristup za izvršanjem neke operacija nad određenim objektom (1), PEP poziva metodu *Decision* navodeći autorizacijski zahtjev (2). PDP dohvaća politiku nadzora pristupa (3) te na osnovi korisnikovih uloga donosi autorizacijsku odluku koju vraća kao rezultat poziva metode (4). PERMIS definira dvije autorizacijske odluke: *odobren pristup* (engl. accept) i *zabranjen pristup* (engl. deny). Na osnovu autorizacijske odluke PEP dopušta ili odbija pristup objektu.

4.2.2 Upravljanje odredbama nadzora pristupa

Upravljanje odredbama nadzora pristupa u PERMIS-u odvija se pomoću *Objavljivača svojstva*. Izdavatelj svojstava korištenjem *Objavljivača svojstva* (engl. Privilege Allocation, PA) izdaje X.509 iskaznice svojstava u kojima definira uloge korisnika i politiku nadzora pristupa. Iskaznice se spremaju u LDAP direktorij kojemu kasnije, tijekom obrade zahtjeva, pristupa podsustav provjere svojstava.

Slika 4-8: Objavljivač svojstva

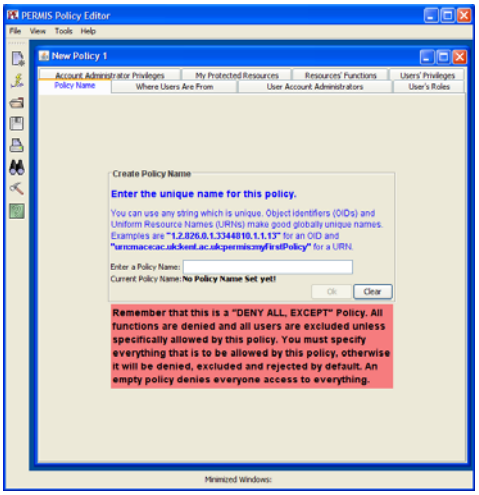
Objavljivač svojstva uključuje tri alata: *Attribute Certificate Manager, Policy Editor* i *Policy Wizard*. *Attribute Certificate Manager* je alat koji omogućuje stvaranje iskaznica svojstava, dok se *Policy Editor* i *Policy Wizard* koriste prilikom stvaranja i izmjeni odredbi nadzora pristupa. Primjerci sučelja alata prkazani su na slikama 4-9, 4-10 i 4-11.



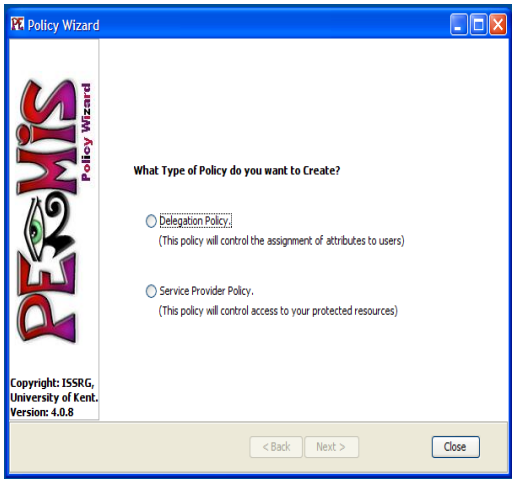
Slika 4-9: Attribute Certificate Manager

Policy Editor je alat koji omogućava stvaranje i modificiranje odredbi nadzora pristupa, te njihovo potpisivanje i spremanje u LDAP direktorij. Odredbe se mogu spremati u XML formatu ili u obliku X.509 iskaznica svojstva. Spremanje u obliku iskaznice zahtijeva posjedovanje tajnog ključa kojim će iskaznice biti potpisane.

Policy Wizard je korisnički-orijentiran (engl. user-friendly) alat kojim se na jednostavan način mogu stvarati odredbe nadzora pristupa. Glavni cilj pri osvarivanju *Policy Wizard*-a bio je ponuditi alat s kojim će se moći definirati odredbe pristupa u najkraćem mogućem vremenu. Zbog toga, *Policy Wizard* pruža samo podskup funkcionalnosti ostvarenih u *Policy Editor*-u. *Policy Wizard* nudi prikaz definirane politike pomoću prirodnog jezika, tako da korisnik može ustanoviti ima li definirana politika željena svojstva.



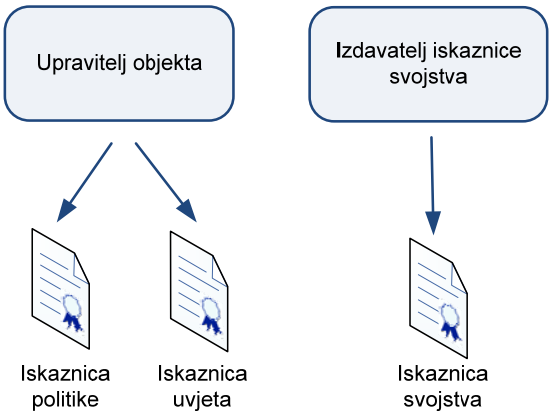
Slika 4-10: Policy Editor



Slika 4-11: Policy Wizard

4.3 Akenti

Akenti [53] je sustav nadzora pristupa razvijan u Berkeley-ovom laboratiju (engl. Ernest Orlando Lawrence Berkeley National Laboratory, LBLB) [54] od 1998. U raspodijeljenim okolinama čest je slučaj da objekt ima više *upravitelja*. *Upravitelj objekta* (engl. stakeholder) je entitet sustava koji ima pravo definirati odredbe pristupa objektu. Primjerice, uređaj za magnetsku rezonanciju u bolnici ima više upravitelja, poput tehničara koji mora nadzirati da ne dođe do oštećenja uređaja prilikom rukovanja, istraživača koji koristi uređaj u znanstveno-istraživačke svrhe, te bolničke administracije koja mora odobriti svaku upotrebu uređaja. Glavni cilj *Akenti*-ja je omogućiti nadzor pristupa sa svojstvom razmjernog rasta u prisustvu više upravitelja. Razmjerni rast sustava postiže se decentrilizacijom politike nadzora pristupa koja se opisuje jezikom zasnovanom na XML-u i pohranjuje u obliku *Akenti iskaznice*. Za opisivanje politike Akenti koristi tri vrste iskaznica: *iskaznice politike*, *iskaznice uvjeta* te *iskaznice svojstava* (slika 4-12).



Slika 4-12: Akenti iskaznice

Iskaznice izdaju i digitalno potpisuju upravitelji objekta i *izdavatelji iskaznice svojstva (eng. attribute authority, AA)*. Za autentikaciju i uspostavu povjerenja između entiteta sustava Akenti se oslanja na infrastrukturu javnih ključeva (PKI). Elementi sustava međusobno se autenticiraju koristeći *X.509 iskaznice,* koje izdaju *ovlašteni izdavatelji iskaznica* (engl. certificate authority, CA).

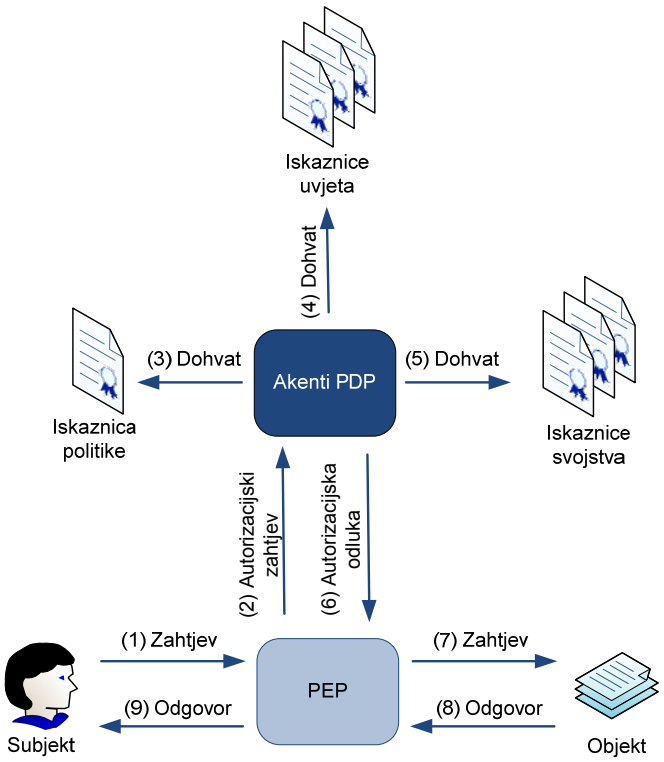
Iskaznica politike (engl. policy certificate) čini osnovni element zaštite određenog objekta i nalazi se na istoj lokaciji kao i objekt kojega štiti. Sadrži naziv objekta kojeg štiti, popis ovlaštenih izdavatelja iskaznica (CA), popis upravitelja objektom, popis lokacija (URL) na kojima se nalaze iskaznice uvjeta, popis lokacija (URL) na kojima se nalaze iskaznice svojstava, te definiraju vremenski okvir unutar kojeg su iskaznice važeće. Iskaznicu politike izdaje i digitalno potpisuje upravitelj objekta.

Iskaznice uvjeta (engl. use-condition certificate) izrađuju upravitelji objekta navodeći ograničenja uporabe objekta. Iskaznica sadrži naziv objekta, popis svojstava (engl. attributes) koja korisnici moraju zadovoljavati za ostvarivanje određenog pristupa objektu, te popis ovlaštenih izdavatelja svojstava. Budući da definiraju prava pristupa objektu, nužno je moći pronaći sve iskaznice uvjeta prilikom donošenja autorizacijske odluke. U suprotnom, pristup objektu se ne dopušta.

Iskaznica svojstava (engl. attribute certificates) veže pojedino svojstvo uz korisnika koje mu može omogućiti pristup određenom objektu. Potpisuju ih ovlašteni izdavatelji svojstava, koji su navedeni u iskaznicama uvjeta. Za razliku od iskaznica svojstva, ne zahtjeva se dostupnost svih iskaznica svojstva prilikom provođenja nadzora pristupa. Nedostupnost iskaznice svojstva može ograničiti pristup objektu, ali nikad neće uzrokovati izvođenje operacije koja inače ne bi trebala biti dozvoljena.

4.3.1 Arhitektura

Primjer obrade korisničkog zahtjeva u Akenti modelu prikazan je na slici 4-13. Korisnik se autenticira koristeći X.509 iskaznicu i SSL/TSL protokol, nakon čega zahtjeva pristup određenom objektu. Akenti ne definira način ostvarenja PEP-a, kao ni protokole komunikacije između korisnika, PEP-a i PDP-a. Definiraju se samo mehanizmi dohvaćanja distribuirane politike nadzora pristupa te način donošenja autorizacijske dozvole.



Slika 4-13: Arhitektura Akenti sustava

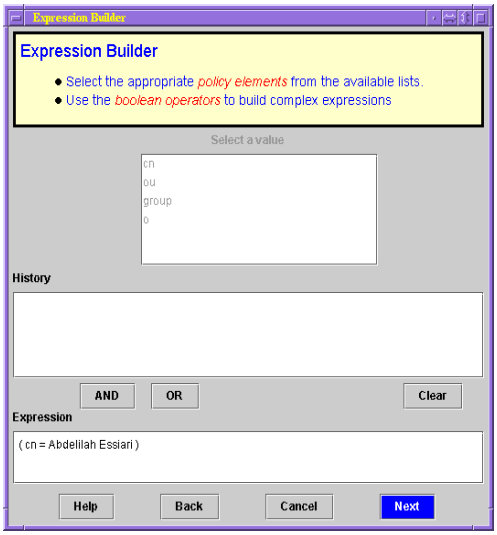
Po primitku zahtjeva autorizacijskog zahtjeva Akenti PDP dohvaća iskaznicu politike u kojoj su zapisane lokacije svih iskaznica uvjeta (3). PDP pronalazi sve iskaznice uvjeta (4) te ih verificira utvrđujući autentičnost upravitelja koji je izdao iskaznicu i provjeravajući digitalni potpis. Zatim se pronalaze iskaznice svojstva (5) na osnovu njihove lokacije definirane u iskaznici politike i/ili iskaznicama uvjeta te se provodi njihove verifikacija. Tada je PDP dohvatio sve informacije potrebne za donošenje autorizacijske odluke koju prosljeđuje PEP-u (6). Akenti PDP interno pohranjuje sve pribavljene iskaznice kao i autorizacijsku odluku, čime se ubrzava obrada slijedećih zahtjeva. Autorizacijska odluka može biti *uvjetovana* (engl. conditional rights), odnosno može navoditi dodatne uvjete koji se moraju ostvariti kako bi subjekt zadobio pravo pristupa, poput uvjetovanja trenutnog opterećenja sustava ili veličine raspoložive memorije.

Akenti, za razliku od većine sustava nadzora pristupa, koristi *čisti model potraživanja* (engl. pure pull model). U čistom modelu potraživanja PEP predočuje PDP-u isključivo identitet korisnika, bez dodatnih svojstva. Motivacija korištenja takvog protokola suradnje proizlazi iz činjenice da njegova primjena ne zahtjeva promjene u tradicionalnim klijent-poslužitelj (engl. client-server) sustavima. No, nemogućnost navođenja dodatnih svojstva korisnika ima ozbiljna ograničenja. Primjerice, u nadzoru pristupa zasnovanom na

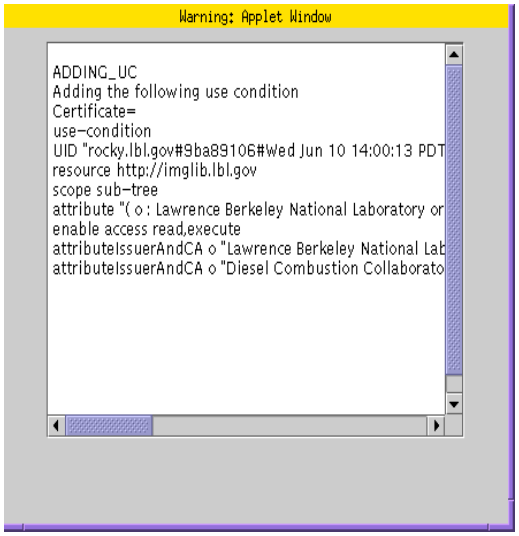
ulogama korisnik može specificirati ulogu (jednu od onih koje obavlja) koju bi želio koristit pri pristupu objektu. Akenti ne dopušta tu mogućnost.

4.3.2 Upravljanje odredbama nadzora pristupa

Akenti iskaznice moguće je izdavati koristeći *alat komandne linije* ili upotrebom *alata s korisničkim sučeljem*. *Alat komandne linije* (engl. command line tool) kao ulaz prima XML dokument (ako se radi o iskaznici politike) ili iskaznicu uvjeta/svojstva te ih potpisuje koristeći privatni ključ i zatim pohranjuje. Alat komandne linije pogodan je za definiranje jednostavnih politika nadzora pristupa, no dočim treba definirati složeniju politiku u prisustvu više upravitelja prikladnije je koristiti *alat s korisničkim sučeljem* (engl. graphic user interface tool). Prilikom stvaranje iskaznice politike alatom s korisničkim sučeljem, korisniku se prikazuje niz obrazac koji omogućuju definiranje svih vrijednosti u iskaznici. Također, koristeći alat korisnik saznaje popis svih upravitelja, mogućih svojstava te operacija nad objektima. U postupku stvaranja iskaznica uvjeta, upravitelju objekta se prikazuje obrazac s podacima o svim upraviteljima tog objekta, mogućim operacijama nad objektom, prethodno definiranim uvjetima i ovlaštenim izdavateljima svojstva. Također, upravitelj može definirati dodatne zahtjeve, poput vremenskog okvira u kojem je iskaznica važeća. Prilikom stvaranja iskaznice svojstva korisniku se prikazuje popis svih svojstava koje je moguće definirati i njihovih mogućih vrijednosti. Prikaz sučelja alata za izdavanje iskaznica dan je na slici 4-14. Osim alata za stvaranje Akenti iskaznica, ostvareno je i sučelje zasnovano na Internet pregledniku koje omogućuje pregledavanje trenutnih odredbi prava pristupa u sustavu, izvedenih na osnovi stvorenih iskaznica politika, uvjeta i svojstva. Primjer sučelja dan je na slici 4-15.



Slika 4-14: Akenti alat za izdavanje iskaznica



Slika 4-15: Akenti web sučelje

4.4 Cardea

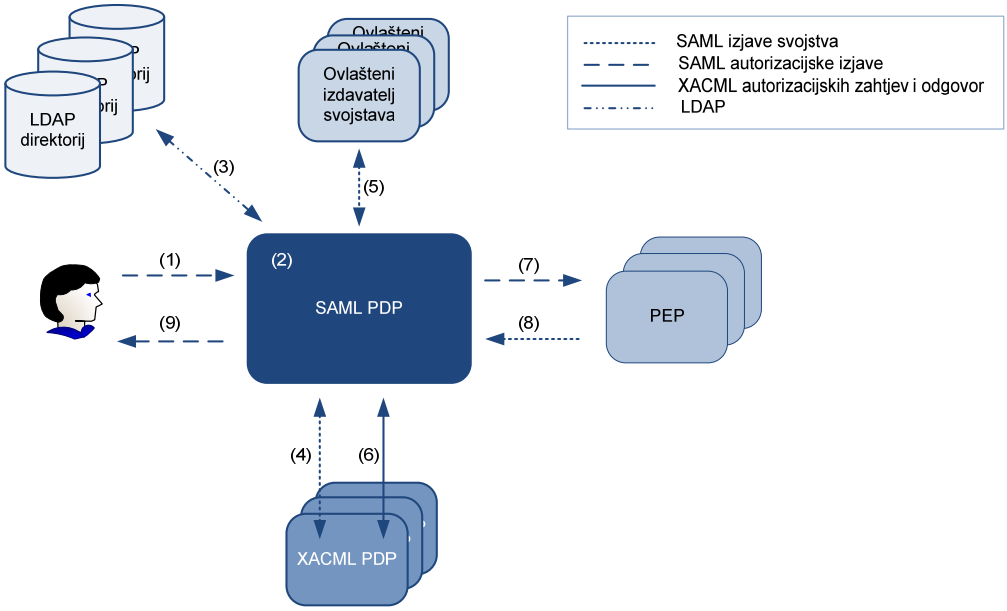
Cardea [55] je raspodijeljeni sustav nadzora pristupa razvijan kao dio NASA-inog računalnog spleta (*NASA Information Power Grid*). Cardea pokušava ostvariti jedinstveni sustav nadzor pristupa nad više administrativnih domena korištenjem standarda SAML i XACML.

Tradicionalni sustavi nadzora pristupa oslanjaju se na postojanje *lokalnog identiteta korisnika*. Pri obradi zahtjeva pristupa nekom objektu, podrazumijeva se da korisnik posjeduje *korisnički račun* (engl. user account) na računalu na kojem se nalazi objekt, te se na osnovu toga donosi odluka o pristupu. To zahtijeva, da prije samo sprovođenja nadzora pristupa, svi administratori računala sustava moraju poznavati sve moguće korisnike te njihova prava pristupa. Očito je da u velikim računalnim spletovima taj pristup ostvarenja sustava nadzora pristupa nije prikladan. Slično kao Akenti i PERMIS, Cardea donosi odluke o pristupu na osnovi skupa svojstava subjekata i objekata, a ne lokalnom indetitetu. Krajnji cilj projekta iz kojeg je proizašla Cardea je *dinamičko upravljanje korisničkim računima.* Sustav za *dinamičko upravljanje korisničkim računima* (engl. dynamic session management) omogućuje automatsko kreiranje i povezivanje lokalnih korisničkih računa potrebnim za provođenje autorizacijskog zahtjeva.

Upotreba postojećih standarda i tehnologija pruža jednostavnije međudjelovanje (engl. interoperating) među dijelovima sustavima. Komunikacija među dijelovima sustava se standardizira definiranjem reprezentacije identiteta korisnika i njegovih svojstava, zahtjeva za pristupom, odgovora, kao i protokola kojim se oni izmjenjuju. Cardea koristi prihvaćene XACML i SAML standarde opisane ranije. Uz njih, integritet poruka štiti se digitalnim potpisom (*XML Digital Signature*).

4.4.1 Arhitektura

Cardea se sastoji od nekoliko međusobno nezavisnih komponenata. Sustav čine SAML PDP, jedan ili više ovlaštenih izdavatelja svojstava, jedan ili više XACML PDP-ova te jedan ili više PEP-ova. Funkcionalnosti komponenata izložene su kao web usluge. Glavna komponenta sustava je SAML PDP koji prihvaća zahtjeve, obrađuje ih, dobavlja potrebne informacije, proslijeđuje ih ostalim komponentama, te daje odgovor.



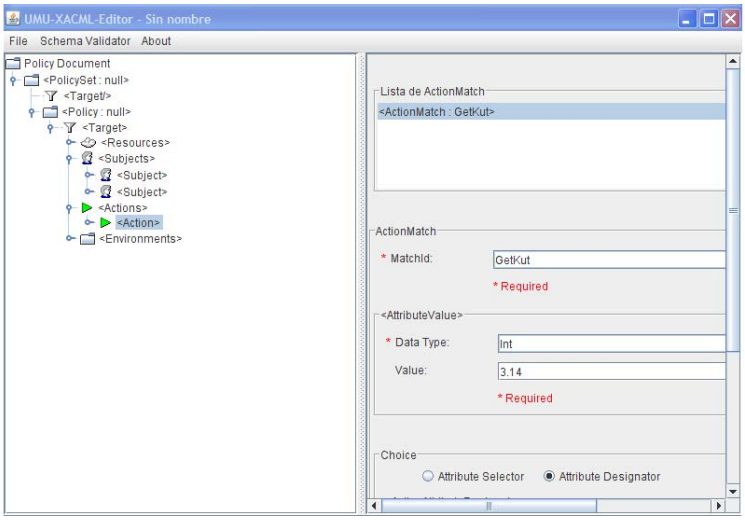
Slika 4-16: Arhitektura Cardea sustava

Obrada korisničkog zahtjeva prikazana je na slici 4-16. Korisnik, koristeći SAML protokol za izmjenu izjava, predaje autorizacijski zahtjev (*SAML AuthorizationDecisionQuery*) SAML PDP-u (1). SAML autorizacijski zahtjev sadrži identitet korisnika te oznaku entiteta koji je autenticirao korisnika. Dodatno, korisnik može specificirati ulogu koju bi želio koristit pri pristupu objektu. Po primitku zahtjeva SAML PDP analizira zahtjev (2). Cilj analize je otkriti kojoj administrativnoj domeni korisnik pripada, odnosno koji XACML PDP je nadležan za donošenje autorizacijske odluke, te tko su ovlašteni izdavatelji svojstva. Na osnovu identiteta korisnika SAML PDP pristupa LDAP direktoriju (3) u kojima su pohranjene lokacije ovlaštenih izdavatelja svojstva i nadležnog XACML PDP-a.

XACML PDP prima autorizacijski zahtjev i daje odgovor. U autorizacijskom zahtjevu potrebno je navesti sve informacije potrebne za donošenje autorizacijske odluke. Zato SAML PDP kontaktira nadležni XACML PDP (4), te saznaje koja korisnikova svojstva su potrebna prilikom odlučivanja o pristupu. Sada SAML PDP stupa u vezu s ovlaštenim izdavateljima svojstva (5) te pomoću SAML upita (*SAMLAttributesQuery*) saznaje relevantna korisnička svojstva. Na osnovu korisničkog zahtjeva i dobavljenih svojstava oblikuje se XACML zahtjev i upućuje XACML PDP-u (6). SAML PDP pretvara XACML autorizacijski odgovor u SAML autorizacijsku izjavu (*SAML AuthorizationDecision*) te ju prosljeđuje odgovarajućem PEP-u (7). PEP izvršava autorizacijski zahtjev te dojavljuje PDP-u dodatne potrebne informacije u pomoću SAML izjave svojstva. Primjerice, dojavljuje se lokalni identitet korisnika koji je korišten prilikom izvršavanja zahtjeva. Konačno, autorizacijska odluka se dojavljuje korisniku.

4.4.2 Upravljanje odredbama nadzora pristupa

Kako Cardea koristi XACML za izražavanje politike nadzora pristupa, nije potrebno razvijati posebne alate za upravljanje odredbama već se mogu koristi postojeći. Jedan od njih je *UMU-XACML-Editor* [56] razvijen na Sveučilištu Murcia (engl. University of Murcia). Korištenjem UMU-XACML-Editora moguće je jednostavno definirati politiku nadzora pristupa izraženu XACML jezikom. Ne zahtjeva se potpuno poznavanje svih konstrukata jezika, već se pomoću sučelja korisniku slikovito prikazuju definirane odredbe i omogućuje njihova izmjena.



Slika 4-17: UMU-XACML-Editor

5 Upravljanje uslugama u okolini PIE

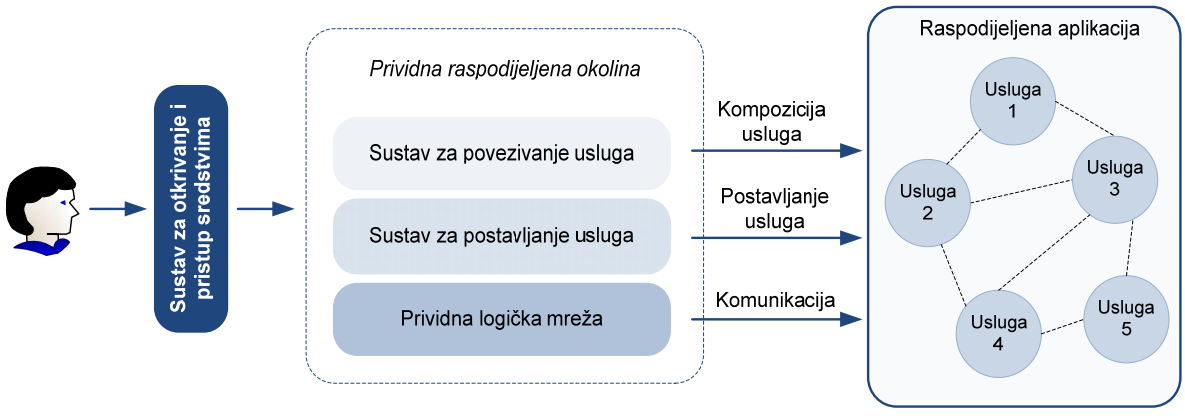
Programirljiva Internet okolina PIE (engl. Programmable Internet Environment) [57] korisnicima omogućuje izgradnju i izvođenje raspodijeljenih primjenskih sustava zasnovanih na programskom modelu zasnovanom na uslugama.

Izgradnja računalnih sustava korištenjem programskog modela zasnovanog na uslugama uslugama (engl. *Service-Oriented Programming Model*) [57] ostvaruje se povezivanjem *primjenskih* usluga (engl. *application services*). Osnovna prednost programskog modela zasnovanog na uslugama nad ostalim metodologijama je raspodjeljivanje *koordinacijske logike povezivanja* (engl. *coordination logic*) na raspodijeljene programe i usluge sinkronizacije i komunikacije [58].

Okolina PIE ostvaruje programski model zasnovan na uslugama. Omogućuje izgradnju raspodijeljenih primjesnkih sustava putem grafičkog sučelja prilagođenog krajnjem korisniku (engl. end-user) zasnovanog na Internet pregledniku. Uporaba Internet preglednika omogućuje nadgledanje i upravljanje sustavom neovisno o sklopovskoj i programskoj platformi računala s kojeg se nadzor ostvaruje.

Okolina PIE razvijena je na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu, u suradnji sa tvrtkom Ericsson Nikola Tesla d.d. iz Zagreb i uz potporu Ministarstva znanosti, obrazovanja i športa Republike Hrvatske.

5.1 Arhitektura okoline PIE



Slika 5-1: Arhitektura okoline PIE

Arhitektura okoline PIE prikazana je na slici 5-1. Osnovni elementi arhitekture su *Sustav za otkrivanje i pristup sredstvima* te *Prividna raspodijeljena okolina*. Svi elementi okoline PIE ostvareni su kao Web usluge.

Sustav za otkrivanje i pristup sredstvima ostvaruje mehanizme nadzora pristupa i izlaže funkcionalnosti *Prividne raspodijeljene okoline*. *Prividna raspodijeljena okolina* je infrastruktura za razvoj, postavljanje i izvođenje raspodijeljenih aplikacija zasnovanih na uslugama. Omogućuje jednostavan razvoj raspodijeljenih aplikacija prilagođen krajnjem korisniku. Prilagodba sposobnostima krajnjih korisnika odgleda se u automatizaciji složenih postupaka, poput postavljanja usluga na čvorove računalne okoline, te u korištenju jednostavnog jezika za izradu raspodijeljenih aplikacija.

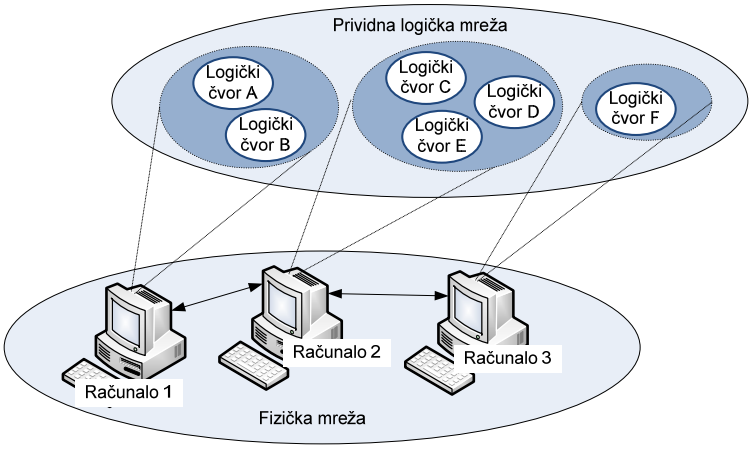
5.2 Prividna raspodijeljena okolina

Prividnu raspodijeljenu okolinu čine tri slojevito postavljena sustava: *Prividna logička mreža*, *Sustav za postavljanje usluga* te *Sustav za povezivanje usluga*. Koncept slojevitosti često se upotrebljava u računalnim arhitekturama. Raspoređivanjem funkcionalnosti po slojevima skrivaju se implementacijski detalji, bez čega je izgradnju velikih računalnih sustava gotovo nezamisliva. Podsustav na višoj razini koristi izložene usluge podsustava na nižim razinama.

Na dnu arhitekture je *Prividna logička mreža* koja ostvaruje nezavisnost raspodijeljenih aplikacija o fizičkoj mrežnoj arhitekturi. Na nju se nadovezuje *Sustav za postavljanje usluga* koji automatizira postupak spremanja, postavljanja i uklanjanja usluga. Potreba za automatiziranim postavljanjem usluga proizlazi iz činjenice da je ručno postavljanje usluga vremenski zahtjevan i greškama podložan proces. *Sustav za povezivanje usluga* omogućuje kompoziciju usluga u raspodijeljene aplikacije, te pruža podršku za njihovo prevođenje i izvođenje.

5.2.1 Prividna logička mreža

Osnovni sloj arhitekture *Prividne raspodijeljene okoline* čini *Prividna logička mreža*. Njen je glavni zadatak ostvariti nezavisnot raspodijeljenih aplikacija s obzirom na fizičku mrežnu infrastrukturu, što je postignuto uvođenjem koncepta *logičkog čvora*. Time se razdvaja logički komunikacijski prostor za izgradnju raspodijeljenih aplikacija od stvarne komunikacijske mreže kojom se ostvaruje fizički prijenos podataka. Također, nesmetanim pristupanje i odjavljivanje logičkih čvorovova iz *Prividne logičke mreže* ostvarena je dinamička priroda mrežne topologije. Radi postizanja svojstva razmjernog rasta informacije o topologiji mreže nisu centralizirane, već je znanje o mrežnoj topologiji distribuirano po logičkim čvorovima. Zbog ograničenog znanja o mreži, logički čvorovi ne mogu uvijek izravno poslati poruke na odredište, već se koriste posebni usmjeravajući čvorovi koji su zaduženi za ispravno usmjeravanje poruka.



Slika 5-2: Struktura *Prividne logičke mreže*

Na slici 5-2 je prikazan primjer konfigracije *Prividne logičke mreže* kao što je prikazan u radu [59]. Šest logičkih čvorova raspoređeno je na tri fizička računala. Budući da je informacija o mrežnoj topologiji distribuirana, čvorovi *Prividne logičke mreže* mogu komunicirati samo s podskupom logičkih čvorova.

Adresa usluge je uređeni par $(ime\ čvora, ime\ usluge)$, gdje $ime\ čvora$ jedinstveno određuje logički čvor mreže, dok $ime\ usluge$ jedinstveno određuje uslugu postavljenu na logičkom čvoru. Prilikom proslijeđivanja poruka, prvi dio adrese se koristi za usmjeravanje zahtjeva do odgovarajućeg logičkog čvora, dok se drugi dio adrese koristi tek na odredišnom čvoru određujući uslugu kojoj se proslijeđuje poruka.

Razdvajanjem logičkog od fizičkog komunikacijskog prostora omogućena je prenosivost raspodijeljenih aplikacija. Moguće je prenositi raspodijeljene aplikacije s jednog skupa računala na drugi, pritom postavljajući odgovarajuću topologogiju *Prividne logičke mreže*. Detalji o *Prividnoj logičkoj mreži* mogu se naći u [60].

5.2.2 Sustav za postavljanje usluga

Sustav za postavljanje usluga nadovezuje se na logičku mrežu. Omogućuje spremanje, postavljanje, uklanjanje te konfiguriranje usluga.

Spremanje usluga je postupak tijekom kojeg administrator računalne okoline sprema izvršni kod usluge unutar *Sustava za postavljanje usluga*. Usluge se spremaju na posebne čvorove računalne okoline koji ostvaruju uslugu skladišta usluga. Usluga se pohranjuje u neaktivnom obliku koji se naziva *instalacijski paket*. Spremljena usluga je spremna za prijenos i postavljanje na čvorove računalne okoline.

Postavljanje usluga je proces prijenosa instalacijskog paketa iz skladišta usluga na čvor računalne okoline te njegove instalacije. Omogućuje automatizirano postavljanje usluga

čime se zamijenjuje ručno postavljanje raspodijeljenje aplikacije koje je vremenski zahtjevno i podložno pogreškama. Postupak postavljanja usluge pokreće administrator računalne okoline ili neka druga usluga. Nakon postavljanja, usluga je spremna za korištenje, tj. obrađivanje poruka.

Konfiguriranje usluga je postupak podešavanja radnih parametara postavljene usluge. Primjer radnog parametra usluge je lokacija *Kataloga usluga*. Lokacija *Kataloga usluga* nužna je informacija za otkrivanje ostalih uluga računalne okoline.

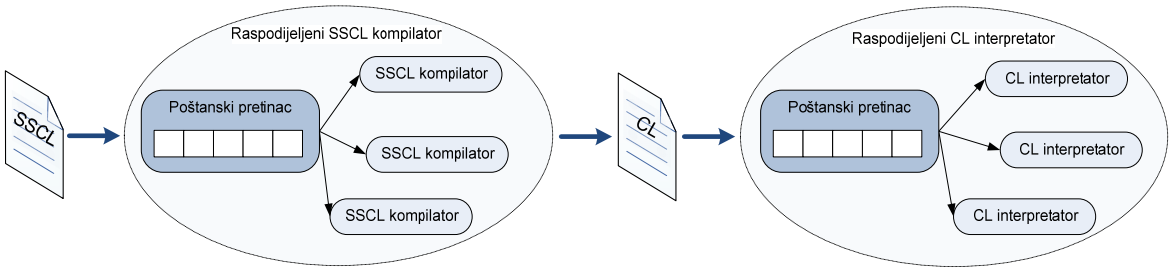
Detalji o *Sustavu za postavljanje usluga* mogu se naći u [59].

5.2.3 Sustav za povezivanje usluga

Sustav za povezivanje usluga ostvaruje proces kompozicije usluga. Usluge postavljene putem *Sustava za otkrivanje i postavljanje usluga* međusobno se povezuju i grade složene raspodijeljene programe. *Sustav za povezivanje usluga* pruža usluge sinkronizacije i komunikacije usluga te prevođenja i izvođenja raspodijeljenih programa.

Prilikom kompozicije usluga, u skladu s zahtjevima arhitekture zasnovane na računalima, potrebno je definirati tok poruka između usluga. *Sustav za povezivanje usluga* omogućuje ostvarivanje sinkronizacije i komunikacije primjenom koncepta *koordinacijskih usluga*. Dakle, korisnik tijekom kompozicije osim *aplikacijskih usluga* (onih koji ustvaruju programsku logiku) raspolaže i *koordinacijskim uslugama*. Razvijene su tri koordinacijske usluge: *semafor*, *poštanski pretinac* i *usmjernik događaja*. *Semafor* ostvaruje mehanizam sinkronizacije i međusobnog isključivanja usluga. *Poštanski pretinac* koristi se za razmjenu poruka između usluga. Omogućavajući asinkronu komunikaciju ostvaruje zahtjev arhitekture zasnovane na računalima za slabim povezivanjem usluga. *Usmjernik događaja* koristi se prilikom komunikacije zasnovane na događajima.

Kompozicija usluga ostvaruje se uporabom programskom jezika *SSCL*-a. *SSCL* (engl. Simple Service Composition Language) je jezik kojim se opisuje međudjelovanje usluga prilagođen krajnjem korisniku. Prevođenje i izvođenje raspodijeljnih programa ostvaruje se uporabom *Raspodijeljenog SSCL kompilatora* i *Raspodijeljenog CL interpretatora*.



Slika 5-3: Proces prevođenja i izvođenja raspodijeljenog programa

Slika 5-3 prikazuje proces prevođenja i izvođenja raspodijeljenog programa. *Raspodijeljeni SSCL kompilator* prima raspodijeljeni program napisam u *SSCL-u* i prevodi ga u *CL* (engl. Coopetition Languague). Sastoji se od *Poštanskog pretinca* i skupa *SSCL kompilatora*. Korisnik šalje raspodijeljeni program napisan u SSCL-u u *Poštanski pretinac*, odakle ga preuzimaju i prevode *SSCL kompilatori*.

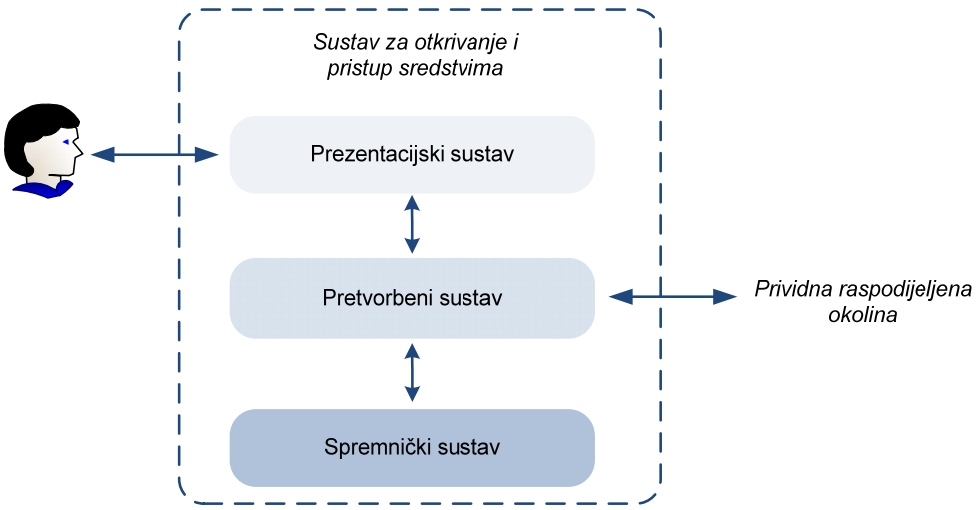
Slijedi izvođenje raspodijeljenog program prevedenog u CL pomoću *Raspodijeljenog CL interpretatora. Raspodijeljeni CL interpretator* sastoji se od *Poštanskog pretinca* i skupa *CL interpretatora. CL interpretatori* uzimaju raspodijeljene programe opisane u CL jeziku iz *Poštanskog pretinca* i izvode ih.

Ovakva arhitektura infrastrukture za izvođenje kompozicije usluga omogućuje decentralizirano prevođenje i izvođenje raspodijeljene aplikacije. Mjesto prevođenja izvođenja raspodijeljenog programa utvrđuje se tek za vrijeme izvođenja. Detalji o *Sustavu za povezivanje usluga* mogu se naći u [61] [62].

5.3 Sustav za otkrivanje i pristup sredstvima

Sustav za otkrivanje i pristup sredstvima djeluje kao posrednik (eng. middleware) između korisnika i *Prividne raspodijeljene okoline*. Svaki korisnički zahtjev prolazi kroz *Sustav za otkrivanje i pristup sredstvima* koji ga prosljeđuje *Prividnoj raspodijeljenoj okolini*. Ostvaruje mehanizme nadzora pristupa i izlaže funkcionalnosti *Prividne raspodijeljene okoline*. Troredna arhitektura *Sustava za otkrivanje i pristup sredstvima* prikazana je na slici 5-4. Slojevitost arhitekture naglašava zavisnost pojedinih sustava.

Prezentacijski sustav izlaže grafičko sučelje prilagođenom krajnjem korisniku (engl. end-user) zasnovano na Internet pregledniku te ostvaruje mehanizme rukovanja koriničkim identitetom: registraciju, identifikaciju i autentikaciju. Sučelje je korisnički svjesno (engl. user-aware), odnosno prilagođeno korisnikovim pravima. Budući da prezentacijski sustav čini radnu okolinu sustava upravljanja odredbama nadzora pristupa koji je tema ovog rada, njegov detaljniji pregled dan je u poglavlju 6 .



Slika 5-4: Arhitektura *Sustava za otkrivanje i pristup sredstvima*

Korisnički zahtjevi definirani pomoću sučelja *Prezentacijskog sustava* predaju se *Pretvorbenom sustavu*. *Pretvorbeni sustav* pregledava korisničke zahtjeve te ih predaje *Prividnoj raspodijeljenoj okolini* koja ih provodi i *Spremničkom sustavu* koji ih bilježi. Dodatno, Pretvorbeni sustav na osnovi podataka *Spremničkog sustava* predaje *Prezentacijskom sustavu* opis podskupa funkcionalnosti *Prividne raspodijeljene okoline* koje korisnik ima pravo koristiti.

Spremnički sustav pohranjuje politiku nadzora pristupa i detaljne podatke o korisnicima i uslugama. Podaci se zapisuju jezikom zasnovanom na XML-u. Detalji o spremničkom sustavu mogu se naći u [63].

5.3.1 Politika nadzora pristupa u okolini PIE

Politika nadzora pristupa okoline PIE ima karakteristike diskrecijskog prava pristupa i prava pristupa zasnovanog na ulogama. Odluke o operacijama koje korisnik smije obavljati donose se na osnovu *identiteta i uloga korisnika*. Razlozi primjene ovakve, hibridine politike nadzora pristupa, navedeni su u poglavlju 3.1.2. Diskrecijsko pravo nije prikladno za primjenu u prisustvu velikog broja korisnika jer zahtjeva definiranje prava pristupa svakog korisnika po naosob. Pravo pristupa zasnovano na ulogama smanjuje broj potrebnih odredbi za opisivanje politike nadzora pristupa, ali donošenje odluke na osnovu uloga umjesto identiteta korisnika smanjuje izražajnost politike nadzora pristupa. Stoga se prilikom kreiranja politike nadzora pristupa okoline PIE kombiniraju oba navedena pristupa.

Politiku nadzora pristupa okoline PIE sukreiraju korisnici i administratori sustava. Administratori sustava stvaraju i uništavaju uloge te pridjeljuje uloge korisnicima. Korisnici prilikom postavljanja *sredstva* definiraju odredbe nadzora pristupa kojima navode identitete i uloge korisnika kojima je dozvoljeno obavljanje *operacija* nad sredstvom. Pregled operacija koje se mogu izvoditi nad sredstvima u okolini PIE dan je u tablici 5-1.

Tablica 5-1: Sredstva i operacije u okolini PIE

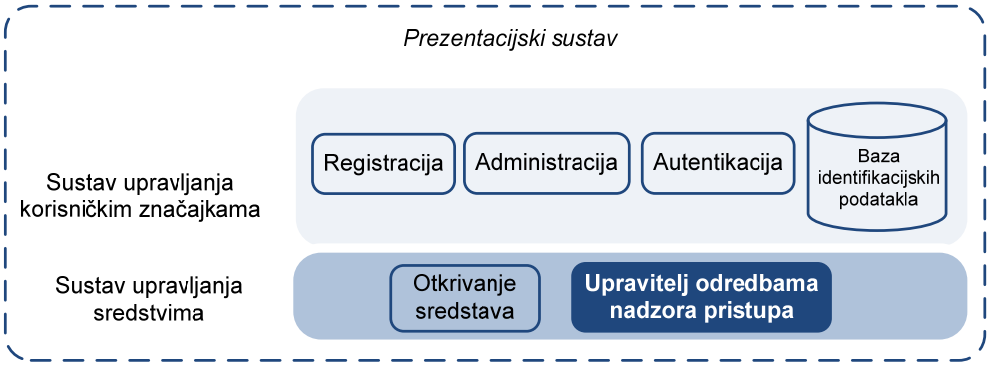
Sredstva	Operacije
Logički čvor	Pregled IP adrese Postavljanje aplikacijske usluge Postavljanje koopeticijske usluge Pozivanje aplikacijske usluge Pozivanje koopeticijske usluge Uklanjanje
Spremljena usluga	Postavljanje Uklanjanje
Postavljena usluga	Pregled Uklanjanje Poziv metode
Primjerak usluge	Pregled Uklanjanje Poziv metode
Udaljena usluga	Pregled Uklanjanje

Sredstva u okolini PIE su *logički čvorovi*, *spremljene usluge*, *postavljene usluge*, *primjerci usluga*, te *udaljene usluge*. Usluge tijekom svog postojanja u okolini PIE poprimaju više oblika. Ako je usluga spremljena u skladištu usluga kao instalacijski paket, onda je usluga u *spremljenom* obliku i nazivamo je *spremljena usluga*. Ako je usluga postavljena na barem jedan čvor računalne okoline, onda je usluga u *postavljenom* obliku te njene postavljene inačice nazivamo *postavljenim uslugama*. Ako je usluga pokrenuta i održava *primjerke* (engl. instances) na barem jednom čvoru, onda govorimo o *primjercima* usluga. Osim usluga koje se pohranjuju, okolina PIE dozvoljava i rad s *udaljenim uslugama*. *Udaljene usluge* nisu spremljene na čvorovima logičke mreže, već je poznat samo njihov WSDL opis. Detalji o životnom ciklusu usluga opisani su u radu [59].

6 Sustav upravljanja odredbama nadzora pristupa u okolini PIE

Sustavi upravljanja odredbama nadzora pristupa moraju omogućiti pregled i izmjenu odredbi nadzora pristupa. Kao što je istaknuto u poglavlju 3.1.2, jedan od osnovnih zahtjeva pri oblikovanju nadzora pristupa u sustavima zasnovanim na uslugama je *izražajnost* politike nadzora pristupa. Okolina PIE namijenja je krajnjem korisniku, stoga sustav upravljanja odredbama nadzora pristupa u okolini PIE mora na razumljiv način prikazati korisniku postojeće odredbe i omogućiti njihovu jednostavnu izmjenu. Izražajnost politike nadzora pristupa i jednostavnost definiranja odredbi dva su suprotna zahtjeva, te je prilikom ostvarivanja sustava upravljanja odredbama nadzora pristupa u okolini PIE potrebno pronaći kompromisno rješenje.

U praktičnom dijelu diplomskog rada oblikovan je i razvijen sustav *Upravitelj odredbama nadzora pristupa*. Korištenjem *Upravitelja odredbama nadzora pristupa* krajnjem korisniku se pruža mogućnost jednostavnog definiranja odredbi prava pristupa sredstvima okoline PIE. *Upravitelj odredbama nadzora pristupa* ostvaren je kao modul *Prezentacijskog sustava*. Arhitektura *Prezentacijskog sustava* s istaknutim *Upraviteljem odredbama nadzora pristupa* prikazana je na slici 6-1.



Slika 6-1: Arhitektura *Prezentacijskog sustava*

Osnovni elementi *Prezentacijskog sustava* su *Sustav upravljanja korisničkim značajkama* i *Sustav upravljajna sredstvima*. *Sustav upravljanje korisničkim značajkama* sadrži module *Registracija*, *Autentikacija*, *Administracija* te *Bazu identifikacijskih podataka*. Prilikom postupka registracije korisnika, modul *Registracija* sakuplja osnovne korisničke podatke, uključujući korisničko ime i šifru, te ih u kriptiranom obliku pohranjuje u *Bazi identifikacijskih podataka*. Modul *Administracija* namijenjen je administratorima okoline PIE. Omogućuje pregled i uređivanje korisničkih računa, stvaranje i uništavanje uloga te dodijelu i opoziv uloge korisniku. Nakon stvaranja korisničkog računa, korisnik pristupa okolini PIE koristeći modul *Autentikacija*. Prilikom procesa autentikacije, korisnik navodi

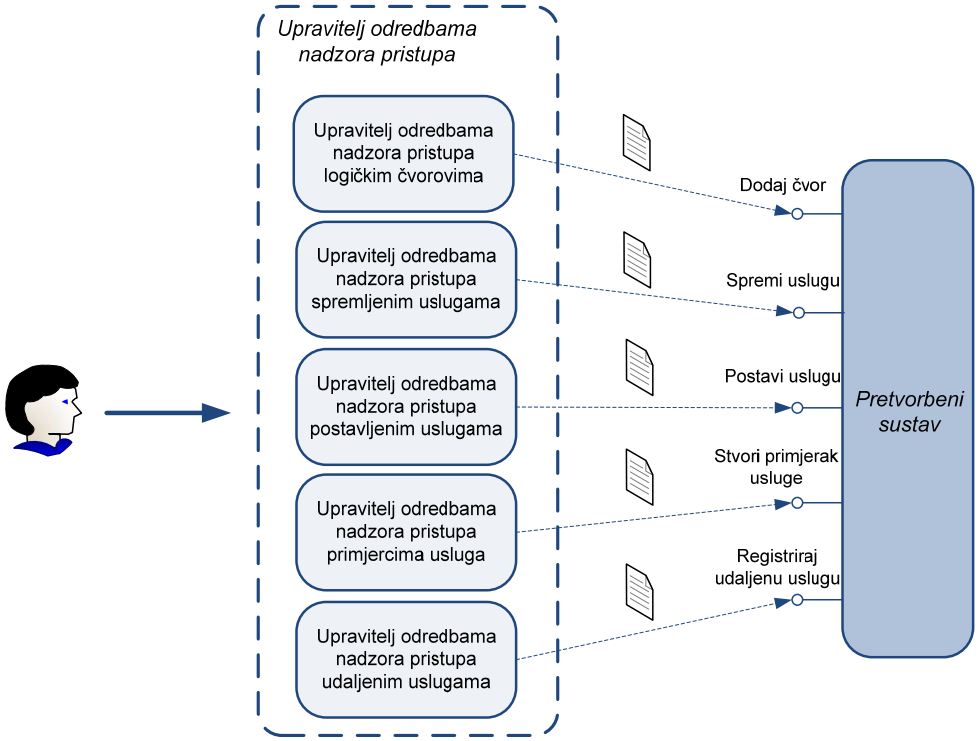
svoje korisničko ime i šifru. Modul autentikacija kontaktira bazu korisničkih podataka te provjerava identitet korisnika. Ukoliko je identitet korisnika autentičan, korisniku se dopušta pristup S*ustavu upravljanja sredstvima*.

Sustav upravljanja sredstvima je *korisnički-svjestan* (engl. user-aware). Kada korisnik pristupa *Sustavu upravljanja sredstvima* njegov identitet je poznat i provjeren što omogućuje izlaganje sučelja prilagođeno zahtjevima i pravima korisnika. Na taj način "korisnik vidi samo ono što smije" (engl. What You See Is What You Get, WYSIWG). *Sustav upravljanja sredstvima* čine moduli *Otkrivanje sredstava* i *Upravitelj odredbama nadzora pristupa.* Modul *Otkrivanje sredstava* kontaktirajući *Pretvorbeni sustav* dobiva informacije o pravima pristupa korisnika te na osnovu njih prikazuje samo ona sredstvima kojim je korišten ovlašten rukovati. *Upravitelj odredbama nadzora pristupa* tema je ovog rada te se detaljnije opisuje u nastavku.

6.1 Upravitelj odredbama nadzora pristupa

Upravitelj odredbama nadzora pristupa je sustav koji krajnjem korisniku (engl. end-user) omogućuje definiranje prava pristupa sredstvima u okolini PIE. Korisnik unosi prava pristupa koristeći *kontekstno-svjesno* (engl. context-aware) sučelje.

Dva su osnovna načina na koja se mogu bilježiti prava pristupa: upotrebom posebno osmišljenih jezika ili korištenjem *izaberi-odaberi* (engl. point-and-click) sučelja. Upotreba posebno osmišljenih jezika zahtjeva da korisnici prije korištenja sustava prođu kroz određenu obuku kako bi savladali osnovne konstrukte jezika. Primjer jezika kojim se mogu izraziti odredbe nadzora pristupa je ranije opisani XACML. *Izaberi-odaberi sučelja* omogućuju definiranje odredbi nadzora pristupa krajnjim korisnicima. Oni korisnika „vode" kroz postupak unosa odredbi nadzora pristupa, ne zahtijevajući veliko predznanje. Primjer takvog načina bilježenja prava pristupa je alat *Policy Wizard* u *PERMIS* sustavu. Kako je okolina PIE prvenstveno namijenjena krajnjem korisniku, osnovni zahtjev pri izgradnji *Upravitelja odredbama nadzora pristupa* jest prilagođenost sposobnostima krajnjeg korisnika. Stoga izrađeni sustav za upravljanje odredbama nadzora pristupa izlaže *izaberi-odaberi* sučelje. Nadalje, sučelje pomoću kojeg korisnik definira prava pristup dinamički se mijenja, ovisno o kontekstu u kojem korisnik radi. Primjerice, tijekom postupka postavljanja usluge, korisniku se prikazuje popis operacija koje se mogu obavljati nad postavljenom uslugom. Nakon odabira operacije, korisniku se nudi popis korisnika i uloga kojima je moguće dozvoliti izvođenje operacije.



Slika 6-2: Arhitektura *Upravitelja odredbam nadzora pristupa*

Arhitektura upravitelja odredbama nadzora pristupa prikazana je na slici 6-2. Osnovni moduli sustava su *Upravitelj odredbama nadzora pristupa logičkim čvorovima*, *Upravitelj odredbama nadzora pristupa spremljenim uslugama*, *Upravitelj odredbama nadzora pristupa postavljenim uslugama*, *Upravitelj odredbama nadzora pristupa primjercima uslugama* te *Upravitelj odredbaman nadzora pristupa udaljenim uslugama*. Moduli upravljaju postupkom definiranja odredbi nadzora pristupa pojedinim sredstvima okoline PIE. Korisnika se tijekom postupka postavljanja sredstva preusmjerava prema modulu zaduženom za definiranje prava pristupa nad tim sredstvom. Modul kontaktira pretvorbeni sustav predajući mu prikupljene podatke.

6.1.1 Arhitektura modula upravitelja nadzora pristupa

Moduli *Upravitelja odredbama nadzora pristupa* imaju zajedničku arhitekturu u kojoj su osnovni elementi *Sučelje*, *Upravitelj sučeljem* i *Upravljačka logika*.

Korištenjem *Sučelja* krajnji korisnik definira odredbe prava pristupa određenom sredstvu. Sučelje sadrži niz *obrazaca* pomoću kojih korisnik definira postavke sredstva. Razlikuju se dvije vrste obrazaca: *obrasci prava pristupa* te *obrasci dodatnih postavki*. Za svaku operaciju koju je moguće izvršiti nad sredstvom definira se po jedan *obrazac prava pristupa* pomoću kojeg korisnik navodi subjekte koji smiju izvršavati operaciju nad sredstvom. *Obrasci dodatnih postavki* omogućuju unos i pregled dodatnih postavki

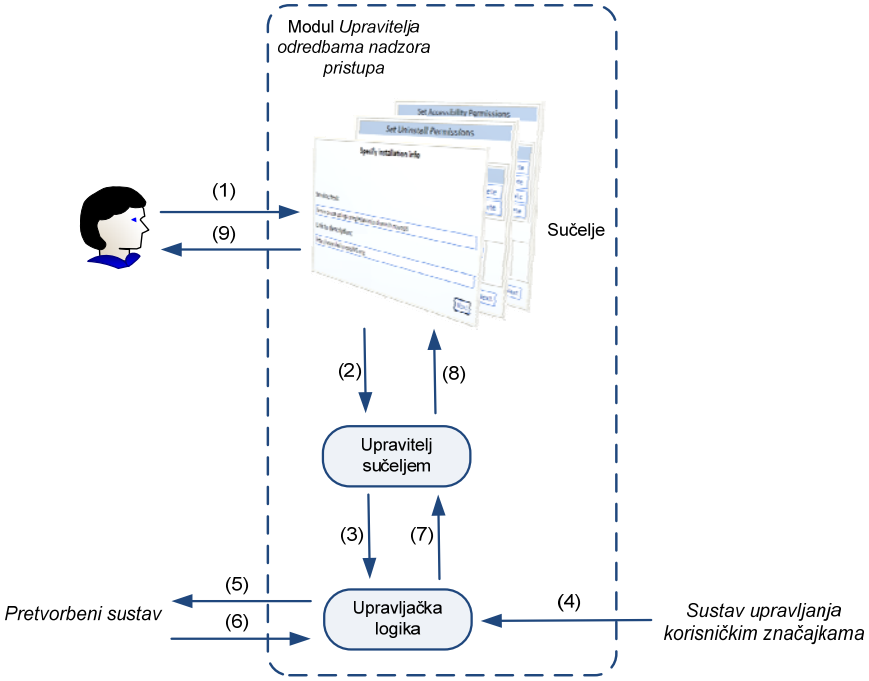
sredstva, poput imena ili opisa sredstva. Prikaz operacija i dodatnih postavki koji se definiraju koristeći pojedine module prikazan je u tablici 6-1.

Tablica 6-1: Operacije i postavke modula *Upravitelja odredbama nadzora pristupa*

	Operacije	Dodatne postavke
Upravitelj odredbama nadzora pristupa logičkim čvorovima	Pregled IP adrese Postavljanje aplikacijske usluge Postavljanje koopeticijske usluge Pozivanje aplikacijske usluge Pozivanje koopeticijske usluge Uklanjanje	Ime logičkog čvora IP adresa
Upravitelj odredbama nadzora pristupa pohranjenim uslugama	Postavljanje Uklanjanje	Ime spremljene usluge Kratak opis Link na opis
Upravitelj odredbama nadzora pristupa postavljenim uslugama	Pregled Uklanjanje Poziv metode	Ime postavljene usluge Ime spremljene usluge Ime logičkog čvora Kratak opis Link na opis
Upravitelj odredbama nadzora pristupa primjercima usluga	Pregled Uklanjanje Poziv metode	Ime primjerka usluge Ime postavljene usluge Ime logičkog čvora Ime spremljene usluge Kratak opis Link na opis
Upravitelj odredbama nadzora pristupa udaljenim uslugama	Dostupnost Uklanjanje	Ime usluge URL Ime logičkog čvora WSDL dokument

Upravitelj sučeljem izgrađuje obrasce i ostvaruje mehanizme povezivanja sučeljem prikazanih podataka s podacima kojima raspolaže *Upravljačka logika*. Programska izgradnja sučelja namijenjenog krajnjem korisniku je vremenski zahtjevan postupak. Stoga se pri izradi modula *Upravitelja odredbama nadzora pristupa* koristi postupak *dinamičke izgradnje sučelja*, koji omogućuje automatiziranje procesa izgradnje korisničkog sučelja. *Upravitelj sučelja* rukovodi navedenim postupkom.

Upravljačka logika dio je modula *Upravitelja odredbama nadzora pristupa* zadužen za obradu prikupljenih podataka i komunikaciju s *Pretvorbenim sustavom*. *Upravljačka logika* poziva *Pretvorbeni sustav* te mu predaje prikupljene podatke i dohvaća rezultat.



Slika 6-3: Postupak definiranja odredbi nadzora pristupa

Postupak definiranja odredbi nadzora pristupa korištenjem modula *Upravitelja odredbama nadzora pristupa* prikazan je na slici 6-3. Korisnik pristupa modulu *Upravitelja odredbama nadzora pristupa* pomoću *Sučelja* (1). Pomoću niza obrazaca korisnik definira odredbe nadzora pristupa određenom sredstvu. Pri tom, *Upravitelj sučelja* pruža podršku za prikupljanje odredbi pristupa (2). Prikupljeni podaci se proslijeđuju *Upravljačkoj logici* koja obrađuje prikupljene podatke (3), te kontaktira *Sustav za upravljanje korisničkim značajkama* te dohvaća identiteta korisnika (4). *Upravljačka logika* poziva *Pretvorbeni sustava* te mu predaje prikupljene informacije (5). *Pretvorbeni sustav* vraća rezultat operacije postavljanja sredstva koji se zatim proslijeđuje korisniku (6 – 9).

6.1.2 Mehanizam dinamičke izgradnje sučelja

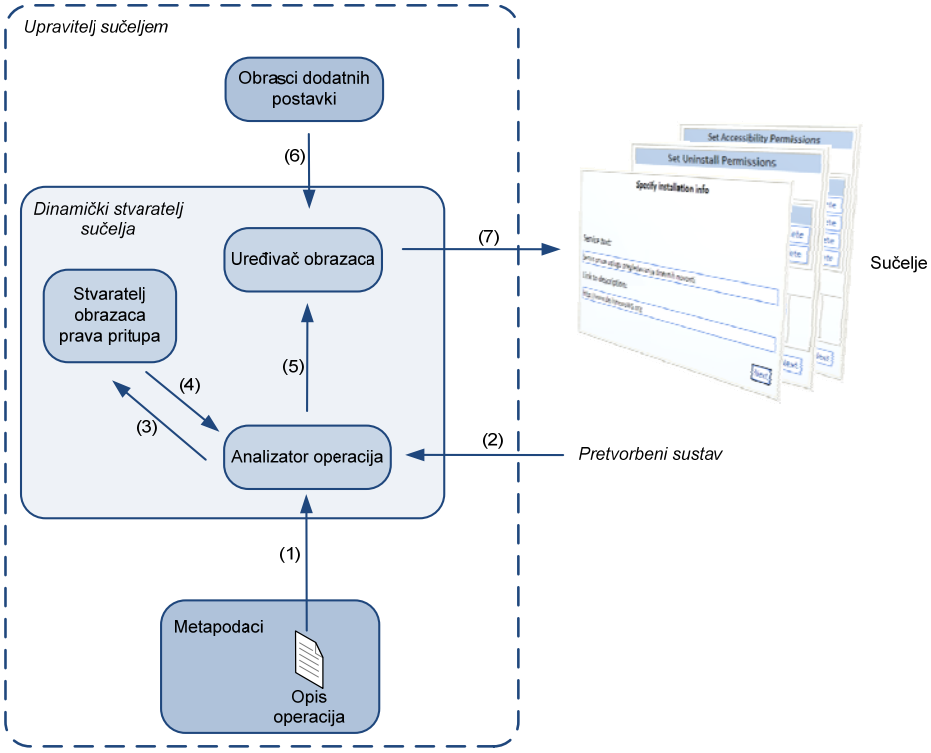
Upravitelj sučeljem oblikuje *Sučelje* modula *Upravitelja odredbama nadzora pristupa i* ostvaruje mehanizme upravljanja *Sučeljem* prikazanih podataka. Obrasci prava pristupa, kao i dio logike kojom se omogućuje njjihovo upravljanje, slični su za sve operacije nad sredstvima. Stoga se za njihovu izgradnju koristi mehanizam *dinamičke izgradnje sučelja*.

Dinamička izgradnja sučelja je automatizirani proces u kojem se oblikuje sučelje na osnovi *metapodataka*. *Metapodaci* sadrže opis podataka koje je potrebno prikazati sučeljem te opis postupaka izmjene podataka. Koristeći opis podataka moguće je na jednoznačan način podatke prezentirati korisniku i omogućiti njihovu izmjenu.

Osnovni dijelovi arhitekture *Upravitelja sučeljem* su *Metapodaci*, *Dinamički stvaratelj sučelja* te *Skup obrazaca dodatnih postavki*. *Metapodaci* stoga sadrže *opis operacija* koje se mogu izvoditi nad sredstvom. U *opisu operacija* za svaku se operaciju navode postupci za pregledavanje, dodavanje i brisanje subjekata koje imaju pravo izvršiti operaciju nad sredstvom. Metapodaci sadrže sve informacije potrebne za dinamičku izgradnju obrazaca prava pristupa.

Za razliku od obrazaca prava pristupa koji se izrađuju dinamičkim postupkom, obrasci dodatnih postavki su specifični za pojedina sredstva te nije moguće automatizirati njihovu izgradnju. Zbog toga ih je potrebno izgraditi prije pokretanja samog postupka dinamičke izgradnje sučelja, te predati *Upravitelju sučelja* kao *Skup obrazaca dodatnih postavki*.

Dinamički stvaratelj sučelja rukovodi postupkom izgradnje sučelja modula *Upravitelja odredbama nadzora pristupa*. Osnovni elementi *Dinamičkog stvaratelja sučelja* su *Analizator operacija*, *Stvaratelj obrazaca prava pristupa* te *Uređivač obrazaca*. *Analizator operacija* dohvaća *opis operacija* te ga analizira. Tijekom postupka analize stvara se popis operacija koje se mogu izvršiti nad sredstvom. Za svaku operaciju poziva se *Stvaratelj obrazaca prava prava* koji koji generira obrasce prava pristupa. Izrađeni obrasci prosljeđuju se *Uređivaču obrazaca*. *Uređivač obrazaca* pribavlja obrasce dodatnih postavki, raspoređuje sve obrasce, te definira konačni izgled korisničkog sučelja.



Slika 6-4: Postupak izgradnje sučelja

Postupak izgradnje sučelja prikazan je na slici 6-4. Dinamički stvaratelj sučelja dohvaća o*pis operacija* (1) koji se analiziraju u *Analizatoru operacija.* Cilj analize je ustvrditi svojstva operacija koje se mogu izvršiti nad sredstvom kako bi se stvorili potrebni obrasci. Također, na osnovi analize *opisa operacija* saznaje se koje je dodatne informacije potrebno prikazati na obrascima, te *Analizator operacija* kontaktira *Pretvorbeni sustav* kako bi ih pribavio (2). Primjerice, na obrascima prava pristupa potrebno je prikazati trenutni popis svih subjekata u okolini PIE. Analizator operacija predaje zahtjev za stvaranjem obrasca za svaku pojedinu operaciju *Stvaratelju obrasca prava pristupa* (3) koji izrađuje obrasce i predaje ih *Analizatoru* (4). Stvoreni obrasci se proslijeđuju *Uređivaču obrazaca* (5). Uređivač obrazaca u primljeni skup obrazaca dodaje o*brasce dodatnih postavki (6).* Nakon obrade, obrasci su putem *Sučelja* dostupni korisniku.

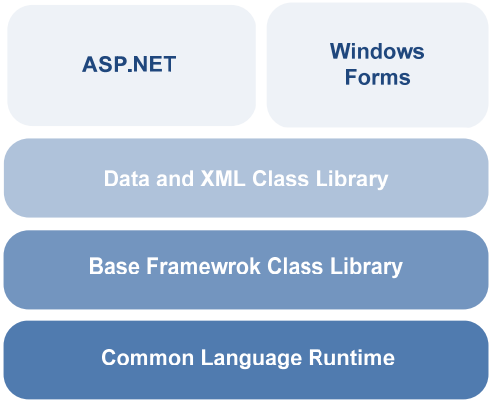
Brojne su prednosti dinamičke izgradnje sučelje. Prvenstveno, olakšana je izrada modula *Upravitelja odredbama nadzora pristupa*. Nadalje, postiže se jednolikost svih sučelja unutar čitavog *Upravitelja odredbama nadzora* pristupa čime se poboljšava iskustvo krajnjeg korisnika. Konačno, ovakav dizajn donosi visok stupanj prilagodljivosti na promjenae u *Pretvorbenom sustavu*. Dodavanje novih sredstava ili promjene postojećih zahtjeva samo izmjenu *Metapodataka*.

6.2 Programsko ostvarenje Upravitelja odredbama nadzora pristupa

Upravitelj odredbama nadzora pristupa programski je ostvaren koristeći *.NET 2.0 radnu okolinu* i *ASP.NET 2.0* tehnologiju. Za razvoj je korišteno radno okruženje *Microsoft Visual Studio 2005* te programski jezik *C#*. Sučelja upravitelja su ostvarena kao *ASP.NET Web stranice* te im se pristupa putem Internet preglednika.

6.2.1 Korištene tehnologije

Microsoft .NET radna okolina (engl. .NET Framework) pruža potporu za razvoj i izvođenje naprednih primjenskih sustava. Sustav sadrži skup programskih jezika, programskih knjižnica i drugih razvojnih alata namijenjenih brzom razvoju raspodijeljenih primjenskih sustava. Nadalje, sustav uključuje i prividni stroj koji omogućuje strojno neovisno izvođenje razvijenih primjenskih sustava. Osnovni dijelovi sustava *Microsoft .NET Framework* prikazani su na slici 6-5.



Slika 6-5: Arhitektura .NET radnog okvira

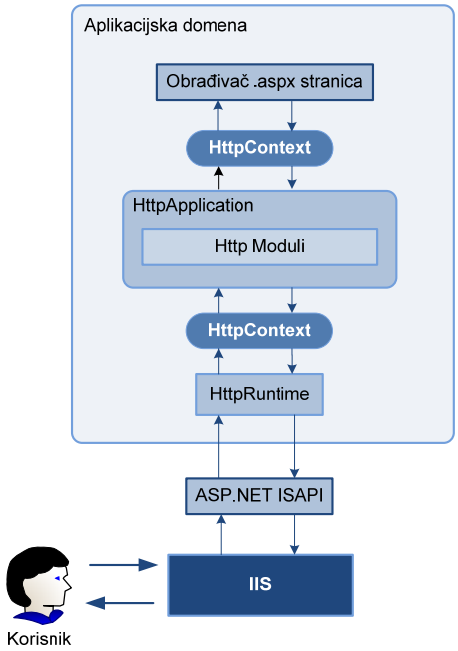
Najniži sloj sustava *.NET Framework* čini podsustav potpore izvođenju *Common Language Runtime (CLR)*. *CLR* podsustav sastoji se od prividnog stroja (engl. *virtual machine*) koji pomoću *JIT* prevoditelja (engl. *just-in-time compiler*) izvodi strojno nezavisni kod *CIL* (*Common Intermediate Language*). Programi prevedeni u *CIL* kod mogu se izvoditi na bilo kojoj sklopovskoj platformi i operacijskom sustavu na kojem je postavljen *CLR* podsustav.

Sljedeći sloj sustava *.NET Framework* čini skup programskih knjižnica *Base Framework Class Library*. Navedene programske knjižnice podupiru rad s podatkovnim tipovima, ulazno-izlazne operacije i ostale osnovne funkcionalnosti potrebne za ostvarenje programske logike. Iznad sloja *Base Framework Class Library* nalazi se skup programskih knjižnica nazvanih *Data and XML Class Library.* Navedene knjižnice pružaju potporu naprednom radu s podacima u bazama podataka te potporu obradi *XML* dokumenata.

Viši slojevi sustava *.NET Framework* omogućuju izgradnju i izvođenje primjenskih sustava. ASP.NET podsustav nudi podršku za izgradnju raspodijeljenih primjenskih sustava povezanih Internetom. Nudi mogućnost izgradnje dinamičkih Web stranica, Web primjenskih sustava i Web usluga. Primjenski sustavi *Windows Forms* zasnovani su na grafičkom sučelju i pristupa im se lokalno.

ASP.NET

ASP.NET je integralni dio .NET radnog okvira te omogućuje razvoj dinamičkih Web stranica, Web primjenskih sustava i Web usluga. *Dinamičke Web stranice* omogućuju razvoj primjenskih sustava s naprednim grafičkim sučeljem dostupnim putem svakog Internet preglednika. Poslužitelj obrađuje korisnikov HTTP zahtjev, proslijeđuje ga ASP.NET-u koji izvršava kod vezan za stranicu i stvara novu stranicu. Poslužitelj korisniku vraća HTTP odgovor koji sadrži novostvorenu stranicu. Detaljan scenarij obrade korisničkog zahtjeva prikazan je na slici 6-6.



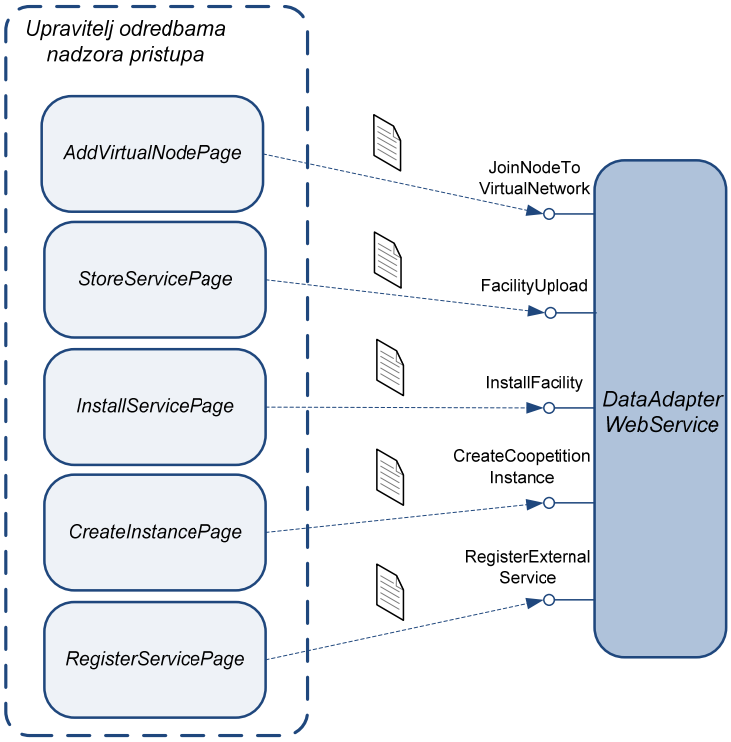
Slika 6-6: Obrada korisničkog zahtjeva u ASP.NET-u

Najčešće korišten poslužitelj za ASP.NET Web stranice je *IIS* (engl. Internet Information Services). ASP.NET se integrira s IIS-om pomoću specijalnog modula, tzv. *ASP.NET ISAPI extension*. IIS proslijeđuje korisnički zahtjev ASP.NET ISAPI-ju koji stvara *aplikacijsku domenu* u kojoj se obavlja proces stvaranja stranice. *Aplikacijska domena* (engl. Application Domain) ostvaruje izolaciju između različitih primjenskih sustava. Aplikacijska domena se stvara samo prilikom prvog zahtjeva te se svi ostali zahtjevi za stranicama istog primjenskog sustava obrađuju se u istoj aplikacijskoj domeni. Zahtjev se predaje *HttpRuntime* objektu koji ga analizira i stvara *HttpContext* objekt. *HttpContext* sadrži sve informacije o koriničkom zahtjevu te ih čini dostupnima za pregled i obradu. Glavne komponente *HttpContext*-a su *HttpRequest*, koji sadrži podatke o trenutnom zahtevu poput korištenog Internet preglednika, i *HttpResponse,* koji će sadržavati odgovor na korisnikov zahtjev. Zatim, *HttpRuntime* stvara *HttpAplication* objekt kojem predaje *HttpContext*. Unutar *HttpAplication* objekta postoje mnogi *Http moduli*, komponente zadužene za obavljanje pripravnih radnji prije samog postupka stvaranja stranice. Primjerice, postoje moduli koji provjeravaju korisnikov identitet i obavljaju autentikaciju (*FormsAuthenticaionModule*) ili uspostavljaju sjednicu s korisnikom (*SessionStateModule*). *HttpAplication* objekt se dodjeljuje svakom korisničkom zahtjevu, no radi poboljšanja performansi primjerci *HttpAplication*-a mogu biti ponovno korišteni za obradu novih zahtjeva. Konačno *HttpContext* objekt se predaje *obrađivaču* (engl. handler) ASP.NET Web stranica koji izvršava kod vezan za stranicu i stvara HTTP odgovor.

HTTP je *bezmemorijski* (engl. stateless) protokol. Poslužitelj svaki zahtjev promatra nezavisno, odnosno sam HTTP protokol ne omogućuje ostvarivanje sjednice (engl. session) s korisnikom. Zbog toga je potrebno osigurati dodatni mehanizam koji će omogućiti povezivanje zahtjeva istog korisnika i očuvanje stanja varijabli korištenih prilikom stvaranja stranice. ASP.NET pruža nekoliko načina za ostvarivanje tog cilja. Prilikom izgradnje upravitelja odredbama nadzora pristupa korišten je mehanizam očuvanja *sjedničkog stanja* (engl. session state). *SessionState* [64] je objekt kojeg ASP.NET pohranjuje unutar aplikacijske domene primjenskog sustava i pomoću njega je moguće sačuvati i dohvaćati varijable između više zahtjeva istog korisnika.

6.2.2 Programska arhitektura

Programska arhitektura *Upravitelja odredbama nadzora pristupa* prikazana je na slici 6-7. Moduli *Upravitelja odredbama nadzora pristupa* ostvareni su razredima *AddVirtualNodePage*, *StoredServicePage*, *InstallServicePage*, *CreateInstancePage* i *RegisterServicePage.* Funkcionalnosti *Pretvorbenog sustava* izložene su kao Web usluga *DataAdapterWebService*. Na slici su prikazane metode sučelja *Pretvorbenog sustava* bitne za rad upravitelja odredbama nadzora pristupa.



Slika 6-7: Programska arhitektura *Upravitelja odredbama nadzora pristupa*

Moduli svoje funkcionalnosti izlažu putem ASP.NET Web stranica. Zajedničke funkcionalnosti potrebne za ispravan rad sučelja, poput naprednih mehanizma upravljanja

sjedničkim stanjem izdvojene su u razred *TemplatePage* kojeg nasljeđuju razredi svih modula. *Upravljačka logika* modula *Upravitelja odredbama nadzora pristupa* definirana je u zasebnim metodama unutar razreda modula. Glavni zadatak upravljačke logike je pretvorba prikupljenih podataka te komunikacija s Web uslugom *DataAdapterWebService* koja se ostvaruje pozivanjem metoda prikazanim u tablici 6-2. Parametri metoda definirani su u Dodatku A.

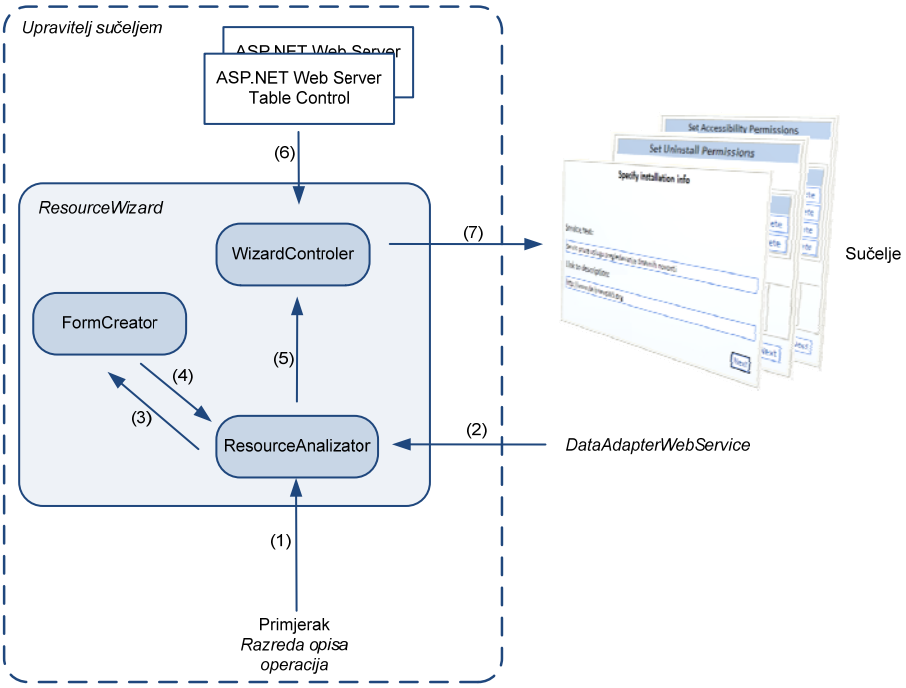
Tablica 6-2: Metode pretvorbenog sustava

Postupci	**Metode Pretvorbenog sustava**		
	Naziv	Ulazni parametri	Povratna vrijednost
Dodavanje logičkog čvora	*JoinNodeToVirtualNode*	VNInfoFull *nodeInfo* string *rootURL*	bool
Spremanje usluge	*FacilityUpload*	SSInfoFull *ssInfo* Package *package*	-
Postavljanje usluge	*InstallFacility*	ISInfoFull *isInfo*	bool
Stvaranje primjerka usluge	*CreateCoopetitionInstance*	SIInfoFull *siInfo*	bool
Registriranje udaljene usluge	*RegisterExternalService*	ESInfoFull *esInfo*	string

6.2.3 Ostvarenje mehanizma dinamičke izgradnje sučelja

Programska arhitektura *Upravitelja sučeljem* prikazana je na slici 6-8. *Dinamički stvaratelj sučelja* ostvaren je razredom *ResourceWizard*. Unutar razreda *ResourceWizard* definirane su metode kojima se ostvaruju *Analizator operacija*, *Stvaratelj obrazaca prava pristupa* te *Uređivač obrazaca*. Metoda *ResourceAnalizator* predstavlja ostvarenje *Analizatora operacija* te kao ulazni parametar prima primjerak *Razreda opisa operacija*. *ResourceAnalizator* provodi analizu primjerka *Razreda opisa operacija* koristeći *refleksiju* (engl. reflection). *Refleksija* [65] je mehanizam programske potpore .NET radnog okvira kojom se, međuostalim, omogućava pribavljanje informacija o tipovima podataka tijekom izvršavanja koda (engl. runtime) te pozivanje metode na osnovi njenog imena. Metoda *ResourceAnalizator* koristi refleksiju za analizu *Razreda opisa operacija*, te tako dobiva popis svih operacija koje se mogu izvršiti nad sredstvom. Metoda *FormCreator* je ostvarenje *Stvaratelja obrazaca prava pristupa* i prima zahtjeve za izgradnjom obrazaca prava pristupa. Izgrađene obrasce, koji su predstavljeni primjercima razreda *ACL*, vraća

metodi *ResourceAnalizator*. Nakon izgradnje svih obrazaca, poziva se metoda *WizardControler* koja predstavlja ostvarenje *Uređivača obrazaca*. *WizardControler* u pribavljeni niz obrazaca uvrštava obrasce dodatnih postavki. Obrasci dodatnih postavki izgrađeni su kao primjerci ASP.NET-ovog razreda *Web Server Table Control*.



Slika 6-8: Programska arhitektura *Upravitelja sučeljem*

Pravila izgradnje razreda opis operacija

Razred opisa operacija sadrži sve informacije potrebne za izgradnju obrazaca prava pristupa. Automatiziranje postupaka oblikovanja i upravljanja obrazaca prava pristupa zahtjeva poštivanje određenih pravila prilikom izgradnje *razreda opisa operacija*. Pravila izgradnje mogu se opisati produkcijama *atributne gramatike* [66] :

(1) $RAZRED \longrightarrow \texttt{public class}\ IDN\ \{OPERACIJE\}$

(2) $OPERACIJE \longrightarrow OPERACIJA_{ime\_operacije,tip\_dozvole}\ OPERACIJE$

(3) $OPERACIJE \longrightarrow \varepsilon$

(4) $OPERACIJA_{ime\_operacije,tip\_dozvole} \longrightarrow DOZVOLA_{ime\_operacije,\ tip\_dozvole}$
$POPIS_{\ ime\_operacije}\ PREGLED_{\ ime\_operacije}$
$UNOS_{\ ime\_operacije,\ tip\_dozvole}$
$BRISANJE_{\ ime\_operacije,\ tip\_dozvole}$

(5) $DOZVOLA_{ime\_operacije,\ tip\_dozvole} \longrightarrow$ `public static readonly Type` $IDN_{ime\_operacije}$ `Object =` `typeof(`$IDN_{tip\_dozvole}$ `);`

(6) $POPIS_{\ ime\_operacije} \longrightarrow$ `public ArrayList` $IDN_{ime\_operacije}$ `{...}`

(7) $PREGLED_{\ ime\_operacije} \longrightarrow$ `public ICollection Select`$IDN_{ime\_operacije}$`() {...}`

(8) $UNOS_{\ ime_{operacije},\ tip_{dozvole}} \longrightarrow$ `public void Add`$IDN_{ime\_operacije}$ `(`$IDN_{tip\_dozvole}$ IDN`) {...}`

(9) $BRISANJE_{\ ime\_operacije,\ tip\_dozvole} \longrightarrow$ `public void Delete`$IDN_{ime\_operacije}$ `(`$IDN_{tip\_dozvole}$ IDN`) {...}`

Nezavršni znakovi gramatike su $RAZRED$, $OPERACIJA$, $DOZVOLA$, $POPIS$, $PREGLED$, $UNOS$, $BRISANJE$ i IDN. $RAZRED$ je početni nezavršni znak gramatike. Nezavršni znak IDN predstavlja proizvoljni identifikator *C#* jezika. Skup završnih znakova čine konstrukti programskog jezika *C#*. Definiraju se dva naslijedna svojstva: $ime\_operacije$ i $tip\_dozvole$, čije se vrijednosti definiraju zasebno za svaku operaciju te prenose prema dnu sintaksnog stabla. Svojstvo $ime\_operacije$ može poprimiti vrijednosti proizvoljnog identifikatora *C#* jezika. Svojstvo $tip\_dozvole$ može poprimiti jednu od vrijednosti definiranih u tablici X.

Tablica 6-3: Raspoloživi tipovi dozvola

Tip dozvole	Opis
Subject	Tip i ime subjekta
SubjectAndAssignedCategory	Tip i ime subjekta te kategorija usluge
SubjectAndWSDLOperation	Tip i ime subjekta te naziv metode usluge

Izgradnja razreda opisa operacije započinje primjenom produkcije (1):

$RAZRED \xrightarrow{(1)}$ `public class` IDN `{`$OPERACIJE$ `}`

IDN u ovoj produkciji predstavlja naziv razreda koji može biti proizvoljan. Nezavršni znak $OPERACIJE$ predstavlja tijelo razreda u kojem se može definirati proizvoljan broj operacija. Primjenom produkcija (2) i (3) započinje se postupak opisivanja jedne operacije:

$\xrightarrow{(2),(3)}$ `public class OpisOperacije {`$OPERACIJA_{op.\ Subject}$`}`

Svojstvu $ime\_operacije$ dodijeljana je vrijednost op, dok je svojstvu $tip\_dozvole$ pridjeljena vrijednost $Subject$. Vrijednosti svojstva prenose se produkcijom (4):

$\xrightarrow{(4)}$ `public class OpisOperacije {`

$DOZVOLA_{op,\ Subject}\ POPIS_{op}\ PREGLED_{op}\ UNOS_{op,\ Subject}\ BRISANJE_{op,\ Subject}\}$

Za svaku operaciju potrebno je definirati tip dozvole, popis svih dozvola kojima se odobrava izvršavanje operacije, te metode koje omogućavaju pregled trenutnih dozvola, unos novih te brisanje postojećih dozvola. Tip dozvole definira se primjenom produkcije (5):

$\xrightarrow{(5)}$ `public class OpisOperacije {`

`public static readonly Type opObject = typeof(Subject);`

$POPIS_{op}\ PREGLED_{op}\ UNOS_{op,\ Subject}\ BRISANJE_{op,\ Subject}\}$

Popis svih dozvola definira se produkcijom (6):

$\xrightarrow{(6)}$ `public class OpisOperacije {`

`public static readonly Type opObject = typeof(Subject);`

`public ArrayList op {...}`

$PREGLED_{op}\ UNOS_{op,\ Subject}\ BRISANJE_{op,\ Subject}\}$

Popis svih dozvola ostvaruje se kao svojstvo tipa *ArrayList*. Produkcijama (7), (8) i (9) definiraju se metode pregleda, unosa i brisanja dozvola.

$\xrightarrow{(7)(8)(9)}$

```
public class OpisOperacije {
    public static readonly Type opObject = typeof(Subject);
    public ArrayList op {...}
    public ICollection Selectop () {...}
    public void Addop (Subject subject) {...}
    public void Deleteop (Subject subject) {...}}
```

U tijelima navedenih metoda potrebno je ostvariti mehanizme kojima se omogućuje modificiranje popisa dozvola.

6.2.4 Ostvareni razredi

Ostvareni mehanizam dinamičke izgradnje sučelja moguće je koristiti za buduća proširivanja funkcionalnosti *Upavitelja odredbama nadzora pristupa*. Ovdje je dan detaljniji prikaz ostvarenih razreda kako bi se olakšalo buduće rukovanje njima.

ResourceWizard

Razred *ResourceWizard* predstavlja ostvarenje *Dinamičkog stvaratelja sučelja.* Izveden je iz razreda *Wizard*. *Wizard* [67] je ASP.NET-ov razred koji omogućuje pregled i prikupljanje podataka kroz više koraka, upotrebom obrazaca (engl. forms). *Wizard* ostvaruje mehanizme koji omogućuju dodavanje koraka, navigaciju među koracima, konzistentnost podataka prilikom navigacije i drugo. *ResourceWizard* koristi funkcionalnosti *Wizard*-a za dinamičku izgradnju sučelja modula *Upravitelja odredbama nadzora pristupa*. Opis sučelja izrađenog razreda dan je u tablici 6-4.

Tablica 6-4: Sučelje razreda *ResourceWizard*

ResourceWizard			
Konstruktor	Ulazni parametri		
	ref object *resource* Type *resourceType* TemplatePage *page*		
Svojstva	-		
Metode	Naziv	Ulazni parametri	Povratna vrijednost
	AddFirstStep	Control *startStepControl*	-
	AddLastStep	Control *lastStepControl*	-
Događaji	*FirstStepDone* *LastStepDone*		

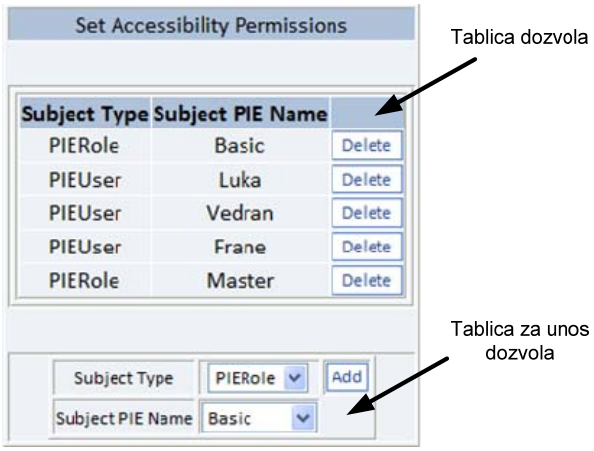
Konstruktor razreda *ResourceWizard* prima tri parametra. *Resource* predstavlja primjerak *Razreda opisa operacija*, koji ne mora biti definiran u trenutku pozivanja konstruktora, te se zato njegov tip navodi pomoću drugog parametra, *ResourceType*. Parametrom *page* definira se stranica tipa *TemplatePage* koja će sadržavati stvorenu instancu *ResourceWizard* razred. Postupci razreda *ResourceWizard* koriste taj parametar kako bi pristupile metodama za upravljanje sjedničkim objektom.

Korištenjem metoda *AddFirstStep* i *AddLastStep* moguće je dodati obrasce dodatnih postavki sredstva. Prilikom završetka prvog i zadnjeg koraka postavljanja sredstva *ResourceWizard* objavljuje događaje (engl. events) *FirstStepDone* i *LastStepDone* na koje

se je moguće pretplatiti te tada obaviti potrebne mehanizme upravljačke logike (primjerice kontaktirati pretvorbeni sustav).

ACL

Razred *ACL* predstavlja programsko ostvarenje obrazaca prava pristupa. Izveden je iz ASP.NET-ovog razreda *Table* [68]. Slika 6-10 prikazuje izgled primjerka razreda *ACL* .



Slika 6-9: Sučelje razreda ACL

Sučelje *ACL* kontrole podijeljeno je u dva dijela: *Tablicu dozvola* ostvarenu razredom *AutoGridView* te *Tablicu za unos dozvola*. Korisnik, koristeći *Tablicu za unos dozvola*, dobiva informaciju o subjektima okoline PIE te im dodjeljuje pravo izvršavanja operacije. Popis subjekata okoline PIE dobiva se kontaktiranjem usluge *DataAdapterWebService*. *Tablica dozvola* omogućuje pregled i brisanje dozvola.

Tablica 6-5: Sučelje razreda *ACL*

	ACL
Konstruktor	Ulazni parametri
	ref object *resource* string *resourceOperationName* TemplatePage *page*
Svojstva	-
Metode	-
Događaji	-

Opis sučelja izrađenog razreda *ACL* dan je u tablici 6-5. Konstruktor razreda *ACL* prima tri parametra. *Resource* predstavlja primjerak *Razreda opisa operacija*. Parametar *resourceOperationName* označava ime operacije za koju se stvara obrazac prava pristupa.

Postupci razreda *ACL* koriste parametar *page* kako bi pristupile metodama za upravljanje sjedničkim objektom.

AutoGridView

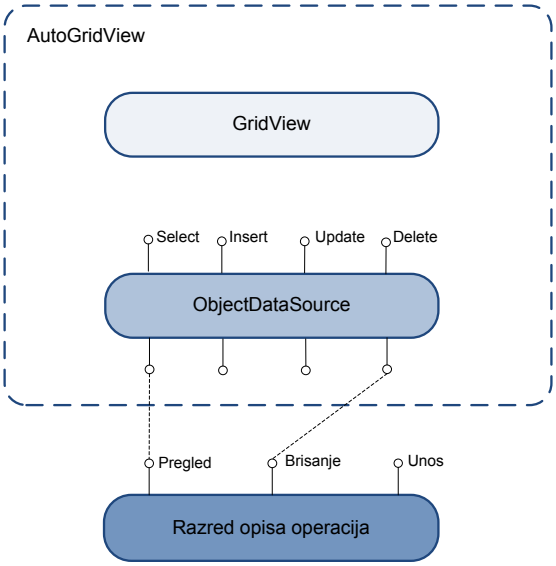
Razred *AutoGridView* prikazuje dvodimenzionalnu tablicu dozvola za izvođenje određene operacije te omogućuje provođenje postupka brisanja dozvola. Primjer sučelja *AutoGridView* kontrole dan je na slici 6-10.

Subject Type	Subject PIE Name	Service Category	
PIERole	Advanced	Coopetition	Delete
PIERole	Basic	Aplication	Delete
PIERole	Master	Aplication	Delete
PIEUser	Luka	Coopetition	Delete
PIEUser	Hrvoje	Aplication	Delete

Slika 6-10: Sučelje razreda *AutoGridView*

AutoGridView omogućuje automatiziranje *postupka povezivanja s podacima*, generirajući sučelje na osnovi metapodataka. U *postupku povezivanja s podacima* (engl. data binding) postiže se konzistentnost između korisničkog sučelja i baze podataka. Svaka promjena podataka u bazi vidljiva je putem sučelja, ali i svaka izmjena izvršena putem sučelja korektno se izvršava u bazi podataka. ASP.NET 2.0 omogućuje provođenje postupka povezivanja s podacima koristeći *kotrole za pristup i prikaz podacima* [69]. *Kontrole za pristup podacima* (engl. data source controls) su ASP.NET-ove kontrole koje omogućuju pristup bazama podataka te čitanje i pisanje podataka. Kontrole za pristup podacima ne izlažu korisničko sučelje, već djeluju kao posrednik (engl. proxy) između baze podataka i korisničkog sučelja. Među kontrolama za pristup podacima razlikujemo kontrole za pristup SQL bazi podataka (*SqlDataSource* kontrola), Access bazi podataka (razred *AccessDataSource* kontrola), XML bazi podataka (*XMLDataSource* kontrola) te kontrole za pristup objektima (*ObjectDataSource* kontrola). *Kontrole za prikaz podataka* (engl. data-bound controls) prezentiraju podatke putem sučelja Internet preglednika. Kontrola za prikaz podataka povezuje se s kontrolom za pristup podacima te prezentira podatke korisniku omogućujući njihov pregled i izmjenu. Neke od kontrola za prikaz podataka su *GridView*, *DetailsView* ili *DropDownList.*

AutoGridView ostvaruje automatiziranje postupka povezivanja s podacima, dinamički povezujući kontrolu za pristup podacima i kontrolu za prikaz podacima. Programska arhitektura *AutoGridView* razreda prikazana je na slici 6-11.



Slika 6-11: Programska arhitektura razreda *AutoGridView*

Za pristup podacima koristi se ASP.NET-ov razred *ObjectDataSource* [70]. Prilikom konfiguracije *ObjectDataSource* kontrole povezuju se njena sučelja s odgovarajućim metodama *Razreda opisa operacija*. Koristeći refleksiju pronalaze se odgovarajuće metode za pregled i brisanje dozvola te se povezuju s odgovarajućim sučeljima *ObjectDataSource* kontrole. Kada je *ObjectDataSource* ispravno konfiguriran izvodi se njegovo povezivanje s kontrolom za prikaz podataka. Za prikaz podataka koristi se ASP.NET-ov razred *GridView* [71].

Tablica 6-6: Sučelje razreda *AutoGridView*

AutoGridView			
Konstruktor	Ulazni parametri		
	ref object *resource* string *resourceOperationName* TemplatePage *page*		
Svojstva		Naziv	Tip
		Modified	ArrayList
Metode	Naziv	Ulazni parametri	Povratna vrijednost
	ShowDeleteButton	-	-
	AddCommandColumn	string *columnName* bool[,] *enabled_marked* string *buttonName*	-
Događaji	*ButtonClick*		

Opis sučelja razreda *AutoGridView* dan je u tablici 6-6. Konstruktor razreda *AutoGridView* prima tri parametra. *Resource* predstavlja primjerak *Razreda opisa operacija*. Parametar *resourceOperationName* označava ime operacije za koju se tablica dozvola. Postupci razreda *AutoGridView* koriste parametar *page* kako bi pristupile metodama za upravljanje sjedničkim objektom. Osim upotrebe pri dinamičkoj izgradnji sučelja modula U*pravitelja odredbama nadzora pristupa*, *AutoGridView* se upotrebljava i u modulu *Otkrivanje sredstava*. Korištenjem svojstva *Modified*, metode *AddCommandColumn* i definiranjem događaja *ButtonClick* moguće je ostvariti napredne mogućnosti *AutoGridView*-a.

TemplatePage

Razred *TemplatePage* sadrži skup funkcionalnosti koji dijele svi moduli *Upravitelja odredbama nadzora pristupa*. Sadrži metode koji osiguravaju ispravan rad sučelja i naslijeđuje ASP-ov razred *Page* [72] . Opis sučelja izrađenog razreda *TemplatePage* dan je u tablici 6-7:

Tablica 6-7: Sučelje razreda *TemplatePage*

TemplatePage			
Konstruktor	Ulazni parametri		
	-		
Svojstva	-		
Metode	Naziv	Ulazni parametri	Povratna vrijednost
	InsertControl	Table *table* Control *control*	-
	RegisterModifiedField	HiddenField *hiddenField*	-
	ReadSelectionFromListBox	ListBox *listBox* string *nameAll*	string
	PopulateList	ListBox *listBox* string *nameAll* string[] *entryList*	-
	SaveToSession	object *resource*	-
	LoadFromSession	Type *resourceType*	object
	SavePropertyToSession	object *resource* string *propertyName*	-
	LoadPropertyFromSession	Type *resourceType* string *propertyName*	string
Događaji	-		

Metoda *InsertControl* postavlja obrasce na Web stranicu i time ih čini dostupnim korisnicima. Metode *RegisterModifiedField*, *ReadSelectionFromListBox* i *PopulateList* ostvaruju dodatne funkcionalnosti koje koristi modul *Otkrivanje sredstava*.

Metode *SaveToSession*, *LoadFromSession*, *SavePropertyToSession* i *LoadPropertyFromSession* ostvaruju napredne mehanizme upravljanja sjedničkim stanjem. Metoda *SaveToSession* kao parametar prima objekt čiju je vrijednost potrebno sačuvati. Za svaki objekt, na osnovi njegovog pripadnog tipa, generira se ključ pod kojim se objekt pohranjuje u *Session* tablici. Metoda *LoadFromSession* dohvaća sačuvani objekt na osnovu ulaznog parametra kojim se označuje pripadni tip objekta. Metode *SavePropertyToSession* i *LoadPropertyFromSession* omogućuju spremanje i dohvaćanje pojedinih svojstva objekata. Oni se koriste za pohranjivanje popisa dozvola operacija iz *Razreda opisa operacija* kojima se ostvaruje pravo izvršavanja pojedinih.

6.3 Primjer modula Upravitelja odredbama nadzora pristupa

Ovdje je dan primjer izvedbe *Upravitelja odredbama nadzora pristupa postavljenim uslugama*. Slika 6-12 prikazuje *Razred opisa operacija* postavljenih usluga izrađen u skladu s definiranom atributnom gramatikom. Nad postavljenom uslugom moguće je izvoditi tri operacije: operaciju pregleda, operaciju uklanjanja te operaciju pozivanja metode usluge

```
public class InstalledServicesOperations
{
    public static readonly Type AccessibilityObject = typeof(Subject);
    public ArrayList Accessibility{...}
    public ICollection SelectAccessibility(){...}
    public void AddAccessibility(Subject accessibilityPermission){...}
    public void DeleteAccessibility(Subject accessibilityPermission){...}

    public static readonly Type UninstallObject = typeof(Subject);
    public ArrayList Uninstall{...}
    public ICollection SelectUninstall(){...}
    public void AddUninstall(Subject uninstallPermission){...}
    public void DeleteUninstantiate(Subject uninstallPermission){...}

    public static readonly Type WsdlObject = typeof(PermissionWSDLoperation);
    public ArrayList Wsdl{...}
    public ICollection SelectWsdl(){...}
    public void AddWsdl(PermissionWSDLoperation WSDLPermission){...}
    public void DeleteWsdl(PermissionWSDLoperation WSDLPermission){...}
}
```

Operacija pregleda

Operacija uklanjanja usluge

Operacija pozivanja metode

Slika 6-12: *Razred opisa operacija* postavljenih usluga

Slika 6-13 prikazuje dio programskog ostvarenja *Upravitelja odredbama nadzora pristupa postavljenim uslugama.* U skladu s prethodno navedenom arhitekturom, istaknuti su dijelovi programa kojima se ostvaruje *Upravitelj sučeljem* i *Upravljačka logika*. *Upravitelj sučeljem* koristi definirani *Razred opisa operacija* postavljenih usluga (*InstalledServicesOperations*) te na osnovi njega automatski stvara potrebne obrasce. Također, koristi se jedan obrazac dodatnih postavki (*FirstStepPanel*). *Upravljačka logika* obrađuje prikupljene podatke, dohvaća korisnikov identitet te poziva odgovarajuću metodu *Pretvorbenog sustava*.

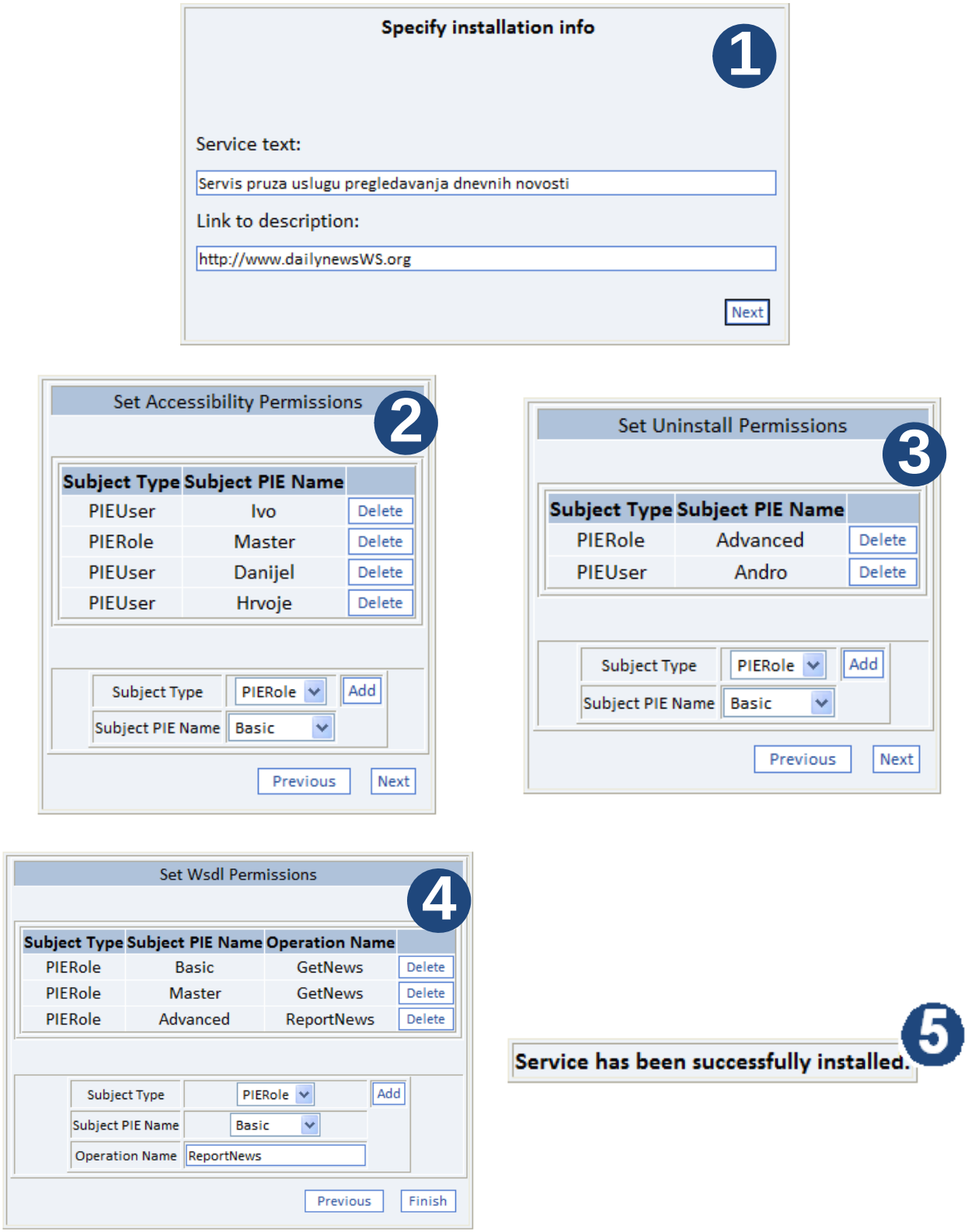
```
public partial class InstalledServices: TemplatePage
{
    object data;
    ResourceWizard resourceWizard;

    protected void Page_Load(object sender, EventArgs e)
    {
        ...
        resourceWizard = new ResourceWizard
                         (ref data, typeof(InstalledServicesPermissions),
                         Page);

        resourceWizard.AddFirstStep(FirstStepPanel);
        InsertControl(table, resourceWizard);
        resourceWizard.FinishButtonClick +=
          new WizardNavigationEventHandler(resourceWizard_FinishButtonClick);
    }

    void resourceWizard_FinishButtonClick(object sender,
                                          WizardNavigationEventArgs e)
    {
        ...
       //Poziv metoda Pretvorbenog sustava
        dataAdapter.InstallFacility(CreateISInfoFull());
        ...

    }

    private DataAdapter.ISInfoFull CreateISInfoFull()
    {
        ISInfoFull isInfoFull = new ISInfoFull();
        ...
        //Dohvat korisnikova identiteta
        isInfoFull.owner = PIE_UserManagement.GetPIEUser(User);
        ...
          return isInfoFull;
    }
}
```

Upravitelj sučeljem

Upravljačka logika

Slika 6-13: Programsko ostvarenje *Upravitelja odredbama nadzora pristupa postavljenim uslugama*

Sučelje *Upravitelja odredbama nadzora pristupa postavljenim uslugama* sastoji se od 4 obrasca i završnog rezultata, prikazanih na slici 6-14.



Slika 6-14: Sučelje *Upravitelja odredbama nadzora pristupa postavljenim uslugama*

7 Zaključak

Računarstvo zasnovano na uslugama slijedeći je korak u razvoju programskih paradigmi. Oslanjajući se na koncept usluge, omogućuje razvoj raspodijeljenih primjenskih sustava velikih razmjera kompozicijom funkcionalnosti koje je moguće izvoditi u različitim sklopovskim i programskim okruženjima. Tehnološke različitosti prevladavaju se primjenom općeprihvaćenih standarda prilikom definiranja sučelja i protokola komunikacije usluga.

Poslovne organizacije mogu ostvarivati dobit objavljivanjem svojih usluga na globalnoj mreži Internet i naplatom njihova korištenja. Preduvjet ostvarivanja tržišta usluga je razvoj sustava nadzora pristupa uslugama. Nadzorom pristupa postiže se da samo ovlašteni korisnici koriste usluge na način opisan odredbama nadzora pristupa. Pružatelj usluga navodi prava pristupa usluzi korištenjem sustava upravljenja odredbama nadzora pristupa.

U diplomskom radu proučeni su postojeći sustavi nadzora pristupa u raspodijeljenim računalnim okolinama te rješenja koja se primjenjuju za upravljanje odredbama nadzora pristupa. Na osnovu proučenih sustava osmišljen je i programski ostvaren *Upravitelj odredbama nadzora pristupa*. *Upravitelj odredbama nadzora pristupa* je sustav koji omogućuje definiranje i izmjenu prava pristupa sredstvima u okolini PIE na način razumljiv krajnjem korisniku.

Izazov pri ostvarenju *Upravitelja odredbama nadzora pristupa* predstavljalo je pronalaženje kompromisa između dvaju suprotnih zahtjeva: izražajnosti politike nadzora pristupa i jednostavnosti definiranja odredbi. Politika nadzora pristupa mora biti dovoljno izražajna da je moguće za svakog korisnika definirati koje metode usluga smije koristiti. Kako je okolina PIE namijenja krajnjem korisniku, sustav upravljanja odredbama nadzora pristupa u okolini PIE mora na razumljiv način prikazati korisniku postojeće odredbe i omogućiti njihovu jednostavnu izmjenu. *Upravitelj odredbama nadzora pristupa* izlaže *izaberi-odaberi* (engl. point-and-click) sučelje koje omogućuje jednostavno definiranje prva pristupa uz dostatnu izražajnost.

Posebna pozornosti pri izradi *Upravitelja odredbama nadzora pristupa* dana je ostvarivanju *prilagodljivosti* na moguće promjene u okolini PIE. Sučelje *Upravitelja odredbama nadzora pristupa* sastoji se od niza obrazaca koji se grade automatiziranim postupkom koji kao ulaz prima opis operacija koje se mogu izvoditi nad sredstvima. Pravila definiranja opisa operacija opisana su atributnom gramatikom. Uz prilagodljivost,

korištenjem postupka dinamičkog generiranja sučelja postignuta je i jednolikost izgleda obrazaca čime se poboljšava iskustvo korisnika.

8 Literatura

1. **Pfleeger, Charles P. i Pfleeger, Shari Lawrence.** *Security in Computing.* Fourth Edition. s.l. : Prentice Hall, 2006. ISBN 0-13-239077-9.

2. **Gasser, Morrie.** *Building a Secure Computer System.* s.l. : Van Nostrand Reinhold, 1988. ISBN 0-442-23022-2.

3. **Pierangela, Samarti and De Capitani, Sabrina.** Access Control: Policies, Models, and Mechanisms. *Foundations of Security Analysis and Design: Tutorial Lectures.* s.l. : Springer Berlin/Heidelberg, 2001, Vol. 2171/2001.

4. **Ferraiolo, David F., Kuhn, D. Richard and Chandramouli, Ramaswamy.** *Role-Based Access Control.* s.l. : Artech House, Inc., 2003. ISBN 1-58053-370-1.

5. **Lampson, Butler W.** *A note on confinement problem.* 1973.

6. **Bell, D.Elliott and La Padula, Leonard J.** Secure Computer Systems: A Mathematical Model. s.l. : The MITRE Corporation, May 1973. Vol. II. MTR-2547.

7. —. Secure Computer Systems: Mathematical Foundations. s.l. : The MITRE Corporation, March 1973. Vol. I. MTR-2547.

8. **Bell, D. Elliott.** Secure Computer System: A Refinement of Mathematical Model. s.l. : The MITRE Corporation, December 1973. Vol. III. MTR-2547.

9. **Bell, D. Elliott i La Padula, Leonard J.** Secure Computer Systems: Unified Exposition and Multics Interpretation. s.l. : The MITRE Corporation, July 1975. MTR-2997.

10. **Harrison, Michael A., Ruzzo, Walter L. and Ullman, Jeffrey D.** Protection in Operating Systems. *Communications of the ACM.* August 1976. Vol. 19.

11. **U.S. Department of Defense.** *Trusted Computer System Evaluation Criteria (TCSEC).* 1985.

12. **Ferraiolo, David D. i Kuhn, D. Richard.** Role-Based Access Control. 1992.

13. *The official website of the Common Criteria Project.* [Mrežno] [Citirano: 18. 4 2007.] http://www.commoncriteriaportal.org/.

14. **Hu, Vincent C., Ferraiolo, David F. and Kuhn, Rick D.** *Assessment of Access Control Systems.* s.l. : NIST National Institute of Standards and Technology, 2006. Interagency Report 7316.

15. **McLean, John.** Security Models. *Encyclopedia of Software Engineering.* s.l. : Wiley Press, 1994.

16. **American National Standard for Information Technology.** *Information Technology - Role Based Acess Control.* 2003. ANSI/INCITS 359-2004.

17. **Osborn, Sylvia, Sandhu, Ravi and Munawer, Qamar.** Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. 2002. Vol. III, 2.

18. **Lampson, Butler W.** Protection. *Proc. 5th Princeton Conf. on Information Sciences and Systems.* 1971.

19. **Graham, G. and Denning, P.** Protection - principles and practice. s.l. : AFIPS Press, 1972.

20. **McLean, John.** A comment on the 'Basic Security Theorem' of Bell and LaPadula.

21. —. Reasoning about Security Models. 1987.

22. —. The Specification and Modeling of Computer Security.

23. **Sipser, Michael.** *Introduction to the Theory of Computation.* 2nd Edition. s.l. : Thomson, 2006.

24. **Anderson, James P.** Computer Security Technology Planning Study. 1972. Vol. II.

25. **Koshutanski, Hristo.** *Distributed Access Control for Web and Business Processes.* 2003. Technical Report # DIT-03-034.

26. **Yavatkar, R., Pendarakis, D. i Guerin, R.** *RFC 2753.* s.l. : Internet Society, 2000.

27. **Lorch, M., et al.** *Conceptual Grid Authorization Framework and Classification.* 2004. Global Grid Forum GFD-1.38.

28. **Wahl, M., Howes, T. i Kille, S.** *Lightweight Directory Access Protocol (v3).* 1997. RFC 2251.

29. **Papazoglou, M. P. and Georgakopoulos, D.** Service-Oriented Computing. *Communications of the ACM.* 2003, Vol. 46, 10.

30. **OASIS.** *Reference Model for Service Oriented Architecture 1.0.* 2006. Committee Specification 1.

31. **Singh, Munindar P. i Huhns, Michael N.** *Service-Oriented Computing: Semantics, Processes, Agents.* s.l. : Wiley, 2005. ISBN-13: 978-0470091487.

32. **Leune, Kees.** *Access Control and Service-Oriented Architectures.* s.l. : CentER Dissetation Series, 2007. Doktorska disertacija. ISBN: 978 90 5668 188 5.

33. **Leune, Keen, Papazoglu, M. i Van Den Heuvel, W. J.** Specification and Querying Security Constraints in the EFSOC Framework. s.l. : ACM Press, 2004. ICSOC: Procceedings of the 2nd international conference on Service oriented computing.

34. **W3C.** *Web Services Architecture.* 2004. W3C Working Group Note.

35. —. *Extensible Markup Language (XML) 1.0 (Fourth Edition).* 2006. W3C Recommendation.

36. —. *SOAP Version 1.2 Part 1: Messaging Framework.* 2003. Recommendation.

37. —. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.* 2007. W3C Working Draft.

38. **OASIS.** *UDDI Version 3.0.2.* 2004. UDDI Spec Technical Committee Draft.

39. **Tilkov, Stefan.** *Overview: Web Services Standards and Specifications.* s.l. : innoQ Deutschland GmbH, 2005.

40. *IETF (The Internet Engineering Task Force).* [Mrežno] [Citirano: 14. 4 2007.] http://www.ietf.org/.

41. *W3C (World Wide Web Consortium).* [Mrežno] [Citirano: 20. 4 2007.] http://www.w3.org/.

42. *OASIS (Organization for the Advancement of Structured Information Standards) .* [Mrežno] [Citirano: 5. 4 2007.] http://www.oasis-open.org/home/index.php.

43. *WS-I (Web Services Interoperaility Organization).* [Mrežno] [Citirano: 15. 4 2007.] http://www.ws-i.org/.

44. **innoQ Deutschland GmbH.** WS-Standards Poster. 2007.

45. **OASIS.** *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0.* 2005. OASIS Standard.

46. —. *Security Assertion Markup Language 2.0 Technical Overview.* 2005. OASIS Working Draft.

47. —. *SAML V2.0 Executive Overview.* 2005. Committee Draft 01.

48. —. *eXtensible Access Control Markup Language (XACML) Version 2.0.* 2005. OASIS Standard.

49. —. A Brief Introduction to XACML. *OASIS.* [Mrežno] 2003. [Citirano: 6. 5 2007.] http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html.

50. **Chadwick, David W. and Otenko, Alexander.** *The PERMIS X.509 Role Based Privilege Management Infastructure.* 2002.

51. —. RBAC Policies in XML for X.509 Based Privilege Management. *Security in the Information Society: Visions and Perspectives: IFIP TC11 17th Int. Conf. On Information Security (SEC2002).* 2002.

52. **The Open Group.** Authorization (AZN) API. 2000. ISBN 1-85912-266-3.

53. **Thompson, Mary R., Essiari, Abdelilah and Mudumbai, Srilekha.** *Cerificate-based Authorization Policy in a PKI Environment.* s.l. : ACM Transaction on Information and Security (TISSEC), 2003.

54. *Lawrence Berkeley National Laboratory.* [Mrežno] [Citirano: 20. 4 2007.] http://www.lbl.gov/.

55. **Lepro, Rebekah.** *Cardea: Dynamic Access Control in Distributed Systems.* 2003. NASA Technical Report NAS-03-020.

56. **Departamento de Ingeniería de la Información y las Comunicaciones.** *UMU-XACML-Editor Home Page.* [Mrežno] [Citirano: 17. 5 2007.] http://xacml.dif.um.es/.

57. **Milanović, Andro.** *Programski model zasnovan na uslugama.* Zagreb : s.n., 2005. Doktorska disertacija.

58. **Žužak, Ivan.** *Sustav praćenja rada posrednika zasnovanog na uslugama.* Zagreb : s.n., 2006. Diplomski rad.

59. **Podravec, Matija.** *Otkrivanje i postavljanje usluga u sustavima zasnovanim na uslugama.* Zagreb : an., 2006. Magistarski rad.

60. **Škvorc, Dean.** *Prividna mreža računalnih sustava zasnovanih na uslugama.* Zagreb : s.n., 2006. Magistarski rad.

61. **Gavran, Ivan.** *Korisnički jezik programskog modela zasnovanog na uslugama.* Zagreb : s.n., 2006. Magistarski rad.

62. **Skrobo, Daniel.** *Raspodijeljeno usporedno interpretiranje programa u arhitekturama zasnovanim na uslugama.* Zagreb : s.n., 2006. Magistarski rad.

63. **Šilić, Marin.** *Spremnički sustav nadzora pristupa u Programirljivoj Internet okolini.* Zagreb : s.n., 2007. Diplomski rad.

64. **MSDN.** SessionState. *MSDN Library.* [Mrežno] 2007. [Citirano: 15. 5 2007.] http://msdn2.microsoft.com/en-us/library/87069683(vs.71).aspx.

65. —. Reflection. *MSDN Library.* [Mrežno] 2007. [Citirano: 10. 5 2007.] http://msdn2.microsoft.com/en-us/library/cxz4wk15.aspx.

66. **Srbljić, Siniša.** *Jezični procesori 1.* Zagreb : Element, 2004. ISBN 953-197-623-6.

67. **MSDN.** Wizard Class. *MSDN Library.* [Mrežno] 2007. [Citirano: 10. 5 2007.] http://msdn2.microsoft.com/en-us/library/system.web.ui.webcontrols.wizard(VS.80).aspx.

68. —. Table Class. *MSDN Library.* [Mrežno] 2007. [Citirano: 10. 5 2007.] http://msdn2.microsoft.com/en-us/library/system.web.ui.webcontrols.table(vs.80).aspx.

69. —. ASP.NET Data Access Overview. *MSDN Library.* [Mrežno] 2007. [Citirano: 10. 5 2007.] http://msdn2.microsoft.com/en-us/library/ms178359(VS.80).aspx.

70. —. ObjectDataSource Class. *MSDN Library.* [Mrežno] 2007. [Citirano: 5. 10 2007.] http://msdn2.microsoft.com/en-us/library/system.web.ui.webcontrols.objectdatasource(vs.80).aspx.

71. —. GridView Class. *MSDN Library.* [Mrežno] 2007. [Citirano: 10. 5 2007.] http://msdn2.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview(vs.80).aspx.

72. —. Page Class. *MSDN Library.* [Mrežno] 2007. [Citirano: 10. 5 2007.] http://msdn2.microsoft.com/en-us/library/system.web.ui.page(vs.80).aspx.

73. **Bell, D. Elliott.** Looking Back at the Bell-La Padula Model. *2005.*

74. **Schackow, Stefan.** *Professional ASP.NET 2.0 Security, Membership, and Role Management.* s.l. : Wiley Publishing, 2006. ISBN-10: 0-7645-9698-5.

75. **Khosravi, Shahram.** *Professional ASP.NET 2.0 Server Control and Component Development.* s.l. : Wiley Publishing, 2006. ISBN-10: 0-471-79350-7.

76. Common Criteria Introduction. [Mrežno] [Citirano: 19. 4 2007.] http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/x595.html.

77. **Rešković, Saša.** *Sinkronizacijski i komunikacijski mehanizmi za sustave zasnovane na uslugama.* Zagreb : an., 2005. Diplomski rad.

78. **Popović, Miroslav.** *Nadziranje pristupa računalnim sustavima zasnovanim na uslugama.* Zagreb : an., 2006. Magistarski rad.

79. Sun's XACML Implementation. [Mrežno] [Citirano: 2. 5 2007.] http://sunxacml.sourceforge.net/.

80. **Čapalija, Davor.** *Objava/pretplata mehanizmi za ostvarivanje mreža zasnovanih na sadržaju.* Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu. Zagreb : s.n., 2005. Diplomski rad.

81. **Westerinen, A. et al.** *RFC 3198.* s.l. : Internet Society, 2001.

9 Dodatak A

U dodatku su definirani tipovi podataka koji se koriste kao parametri prilikom poziva metoda Web usluge *DataAdapterWebService*.

Dodavanje logičkog čvora

```
public class VNInfoFull
{
    public string nodePIEName;
    public string nodeIPAddress;
    public Subject owner;
    public VNResourceACL accessControlList;
 }

public class VNResourceACL
{
    public PermissionInstallOnNode pieInstallation;
    public PermissionInvokeOnNode pieInvocation;
    public PermissionRemoveNode pieRemoval;
    public PermissionViewIP pieViewIP;
}

public class PermissionInstallOnNode
{
    public SubjectAndAssignedCategory[] subjectCategoryList;
}

public class PermissionInvokeOnNode
{
    public SubjectAndAssignedCategory[] subjectCategoryList;
}

public class PermissionRemoveNode
{
    public Subject[] subjectList;
}

public class PermissionViewIP
{
    public Subject[] subjectList;
}
```

Spremanje usluge

```
public class SSInfoFull
{
    public string PIEName;
    public Subject owner;
    public PIEDescription pieDescription;
    public SSResourceACL accessControlList;
}
```

```
public class SSResourceACL
{
    public PermissionInstallation pieInstallation;
    public PermissionUnload pieUnload;
}

public class PermissionInstallation
{
    public Subject[] subjectList;
}

public class PermissionUnload
{
    public Subject[] subjectList;
}

public class Package
{
    public string installScript;
    public System.Byte[] zipPackage;
    public string serviceWsdl;
}
```

Postavljanje usluge

```
public class ISInfoFull
{
    public string installationPIEName;
    public string nodePIEName;
    public Subject owner;
    public StoredServiceParent parentSS;
    public PIEDescription pieDescription;
    public ISResourceACL accessControlList;
}

public class ISResourceACL
{
    public PermissionISAccessible pieAccessibility;
    public PermissionISUninstall pieUninstallation;
    public PermissionWSDLOperation[] wsdlOperationList;
}

public class PermissionISAccessible
{
    public SubjectExtension[] subjectList;
}

public class PermissionISUninstall
{
    public Subject[] subjectList;
}

public class PermissionWSDLOperation
{
    public string operationName;
    public Subject[] subjectList;
}
```

Stvaranje primjerka usluge

```
public class ESInfoFull
{
    public string esAlias;
    public string esURL;
    public Subject owner;
    public PIEDescription esPIEDescription;
    public string serviceWSDL;
    public ESResourceACL accessControlList;
}

public class ESResourceACL
{
    public PermissionESAccessible pieAccessible;
    public PermissionESRemove pieRemovable;
}

public class PermissionESAccessible
{
    public Subject[] subjectList;
}

public class PermissionESRemove
{
    public Subject[] subjectList;
}
```

Registriranje udaljene usluge

```
public class SIInfoFull
{
    public string instanceName;
    public string intallationName;
    public string nodeName;
    public string repositoryName;
    public Subject owner;
    public StoredServiceParent parentSS;
    public PIEDescription pieDescription;
    public SIResourceACL accessControlList;
}

public class SIResourceACL
{
    public PermissionSIAccessible pieAccessibility;
    public PermissionSIUninstantiate pieUninstantiation;
    public PermissionWSDLOperation[] wsdlOperationList;
}

public class PermissionSIAccessible
{
    public Subject[] subjectList;
}

public class PermissionSIUninstantiate
{
    public Subject[] subjectList;
}
```