Deskriptorji: ELEKTRONSKI IMENIK, KOMUNIKACIJE, IMENIK PORAZDELJENI

Helena Tvrdy
IJS Ljubljana

POVZETEK Opisani so osnovni koncepti elektronskega imenika: funkcionalni, organizacijski in informacijski model. Prikazana so načela, po katerih je fizično in logično porazdeljena informacija v imeniku. Podan je pregled servisov, ki jih nudi imenik. Opisano je delovanje imenika kot porazdeljene aplikacije, specificirano je obnašanje posameznih sistemskih posrednikov imenika, ki pri tem sodelujejo. Našteti so koncepti, ki še niso standardizirani.

ABSTRACT The basic concepts of the Directory are described. The functional, organizational and informational models are introduced. The principles of the logical and physical distribution of the directory information base are given. The services provided by the Directory to its users are presented. The operation of the Directory as a distributed application is described; the behaviour of the directory system agent, that participates in this application is specified. The concepts which are not standardized yet are presented.

1 Uvod

Uporabo komunikacijskih storitev, ki jih omogočajo računalniške mreže (elektronska pošta; prenos in obdelava poslov; prenos, dostop in upravljanje datotek; virtualni terminal; ...) v veliki meri omejujeta čas in napor, potrebna za pridobitev informacij o možnih storitvah, o možnih komunikacijskih partnerjih, o načinu vzpostavitve in uporabi storitev, itd.

Tudi izvajalci storitev se soočajo s številnimi težavami, povezanimi z izvajanjem in upravljanjem le-teh. Pomoč v obliki ustreznih informacij potrebujejo za vključevanje in brisanje naročnikov (uporabnikov); za dogovarjanje z naročniki (na primer za pooblastila); za dogovarjanje z drugimi izvajalci; za merjenje porabe in obračunavanje storitev; ...

Sodobno prepričanje je ([But86], [Ver86], [Ami88]), naj bi se vse informacije, potrebne za različne komunikacijske storitve, *zajele enotno* in *hranile* v nekem univerzalnem *porazdeljenem elektronskem imeniku*. Ta porazdeljeni elektronski imenik (ali na kratko: *imenik*) naj bi združil:

- informacije o osebkih, ki komunicirajo, o izvajalcih storitev in o uporabnikih storitev;
- informacije glede na različna organizacijska področja;
- informacije, ki so potrebne za vse različne komunikacijske storitve.

Ta združitev podatkov naj bi ohranila avtonomnost administrativnih in organizacijskih območij ter sistemov nad vzdrževanjem dela podatkov, za katere so zadolženi.

Imenik bi uporabljali na dva načina:

1. kot informacijsko storitev (neposredni uporabniki so ljudje);
2. v povezavi z ostalimi storitvami – kot pomoč tem storitvam (neposredni uporabniki so aplikacije, na primer elektronska pošta ([Bon88])).

Z obravnavanjem ter izvedbo imenskih strežnikov in elektronskih imenikov se je in se še ukvarja več raziskovalnih projektov. Taka sta bila na primer projekta: Grapevine pri XEROX-PARC in V-System, Spice-project ([Lan86]). DFN (Deutsches Forschungsnetz) načrtuje porazdeljeni imenik ([Ver86], [Ver88]) predvsem kot pomoč pri uporabi njihovega sistema za izmenjavo sporočil (MHS). Projekt THORN (THe Obviously Required Nameserver) ([Kil88], [Sir87]), ki je eden od projektov v ESPRIT (European Strategic Programme for Research into Information Technology), se ukvarja z razvojem in izvedbo storitev imenika, v skladu z OSI.

S standardizacijo na področju imenika se je prva začela ukvarjati ECMA (European Computer Manufacturing Association) ([ECMA85]). Od leta 1986 sta na tem področju intenzivneje začela delovati tudi ISO/TC97/SC21/WG4 in CCITT Study Group VII (Question 35), ki probleme rešujeta skupno in izdajata skupne dokumente ([ISO87]).

V članku so v poglavjih 2–8 opisani koncepti in načela, ki naj bi jih izvajalci upoštevali pri izvedbi porazdeljenega imenika. V poglavju 9 pa je navedeno, katera od obravnavanih področij še niso standardizirana. Podatke sem črpala pretežno iz ([COS88], [ISO87], [VER88]); v poglavju 10 pa so navedeni še ostali viri, na katere sem naletela pri študiju imenikov.

2 Funkcionalni model

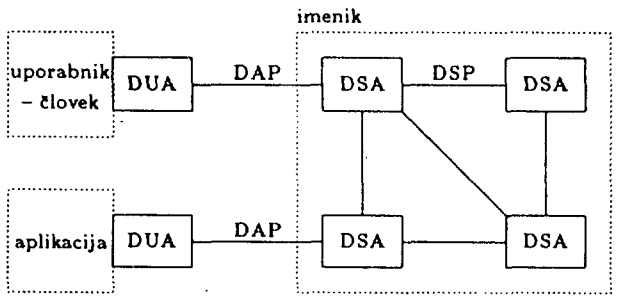
S stališča uporabnika je imenik objekt, ki:

- vsebuje podatkovno bazo, imenovano *informacijska baza za imenik* (DIB – Directory Information Base) in ki
- izvaja dostop do te baze.

DIB vsebuje vhode; vsak vhod predstavlja neki *objekt*. Objekt je karkoli iz realnega sveta (človek, organizacija, aplikativni proces, materialna oprema, ...). Edino, kar zahtevamo od objekta je, da se da identificirati (ima ime).

Imenik tvori množica enega ali več aplikativnih procesov, imenovanih *sistemski posrednik imenika* (DSA – Directory System Agent). Če je imenik sestavljen iz več kot enega DSA, pravimo, da je *porazdeljen*. Uporabniki imenika (ljudje ali aplikacije) dostopajo do imenika preko *uporabniškega posrednika imenika* (DUA – Directory User Agent)(glej sliko 1). DUA je aplikativni proces, ki s stališča imenika predstavlja enega uporabnika.

Imenik nudi svojim uporabnikom množico dostopnih možnosti, imenovano *abstraktni servis imenika*. Oskrbovanje in uporabljanje servisov imenika zahtevata od uporabnika (oziroma dejansko od DUA) in od ostalih funkcionalnih komponent imenika (DSA), da znajo med seboj sodelovati in komunicirati. Če se DUA in DSA nahajata v različnih odprtih sistemih, uporabita *dostopni protokol imenika* (DAP – Directory Access Protocol): DUA pri posredovanju zahteve DSA ter DSA pri vrnitvi odgovora DUA. Vsak DSA ima direktni dostop do informacije, ki jo vsebuje sam;



Slika 1: Funkcionalni model imenika

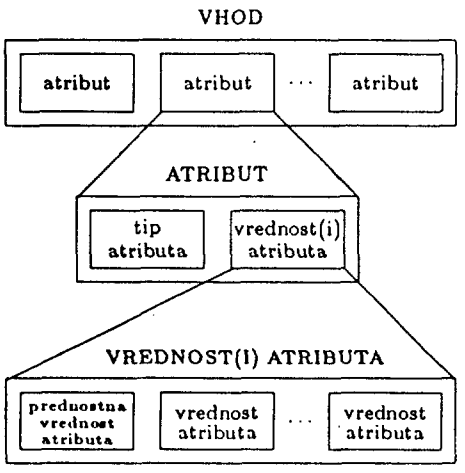
do preostale DIB pa pride tako, da komunicira z drugimi DSA z uporabo *sistemskega protokola imenika* (DSP – Directory System Protocol).

3 Organizacijski model

Množico enega ali več DSA ter nobenega, enega ali več DUA, ki jo upravlja ena sama organizacija, imenujemo *področje upravljanja imenika* (DMD – Directory Management Domain). Področje upravljanja lahko vodi *pošta* (ADDMD – Administration DMD) ali pa kaka druga *organizacija* (PRDMD – Private DMD). Področje upravljanja skrbi za specifikacijo komunikacij med funkcionalnimi komponentami znotraj DMD in za nekatere vidike obnašanja DSA. Vsak uporabniški posrednik mora biti registriran pri enem področju upravljanja. Mednarodne povezave so možne le preko področij upravljanja, ki jih vodi pošta.

4 Informacijski model

Informacija o objektu zanimanja je shranjena v *vhodu*. Vhode, ki jih hrani in vzdržuje imenik, lahko obravnavamo – vsaj konceptualno – kot enotno skupino informacij, imenovano *informacijska baza za imenik* (DIB). Uporabnik vidi imenik kot eno samo podatkovno bazo, čeprav so realni sistemi za imenik fizično porazdeljeni.



Slika 2: Struktura vhoda

Vhod vsebuje enega ali več *atributov*, s katerimi so opisane lastnosti objekta. Atributi, ki opisujejo objekt *oseba*, so na primer: ime in priimek; domači naslov; domača telefonska številka; delovna organizacija, v kateri je oseba zaposlena; službena telefonska številka; funkcija, ki jo ima oseba v delovni organizaciji;...

Vsak atribut sestavljajo *tip atributa*, ki določa vrsto informacije, ki jo opisuje ta atribut in ena ali več *vrednost(i)* tega tipa atributa. Tip atributa je predstavljen z *identifikatorjem objekta*.

Objekti, ki vsebujejo iste tipe atributov, pripadajo istemu *objektnemu razredu*. Vsi objektni razredi so bodisi neposredni, bodisi posredni *podrazredi najvišjega* razreda. Podrazredi podedujejo definicije svojega(-ih) *nad*razreda(-ov). Če na primer v najvišjemu razredu definiramo objekt s tipom atributa "objektni razred", pomeni, da vsebujejo ta tip atributa tudi vsi podrazredi, torej vsak vhod v imeniku.

4.1 Koncept imenovanja

Za uspešno upravljanje vhodov v imeniku je potrebno ustrezno rešiti problem upravljanja imenskega prostora. (Imenski prostor sestavljajo vsa imena, ki jih na osnovi pravil lahko tvorimo nad dano abecedo).

Imenik uporablja hierarhično drevesno strukturo imenskega prostora, ki jo imenujemo *informacijsko drevo za imenik* (DIT – Directory Information Tree). Zaradi hierarhičnega pristopa, se da problem dodeljevanja enoličnih imen vhodom rešiti brez centralne administracije.

V hierarhičnem imenskem prostoru imenika se pojavljata dve konceptualni obliki imen: *del prednostnega imena* – DPI (relative distinguished name) in *prednostno ime* (distinguished name). Vsak vhod – natančneje: upravljalec, odgovoren za ta vhod – je pooblaščen za dodeljevanje enoličnih imen (delov prednostnih imen) naslednikom tega vhoda. Za del prednostnega imena vhoda zadostuje, da je enoličen znotraj omejenega področja – le med vhodi, ki imajo istega predhodnika v DIT. Prednostno ime vhoda je zaporedje, sestavljeno iz delov prednostih imen vseh predhodnih vhodov (začenši pri korenini) in DPI tega vhoda. Ker so vsi DPI enolični znotraj posameznih področij, je prednostno ime enolično znotraj celega imenika. Odnos med deli prednostnih imen in prednostnimi imeni prikazuje slika 3.

DPI so oblikovani iz atributov, zato moramo, če želimo doseči neki vhod v imeniku, le-tega podati v obliki zaporedja, v katerem je vsak člen sestavljen iz tipa atributa in vrednosti atributa.

4.2 Alternativna imena

Prednostno ime mora biti enolično, ni pa nujno, da je to edino ime objekta. Vsak objekt ima lahko poleg prednostnega imena še eno ali več *alternativnih imen*. *Alternativni vhod* ne vsebuje nobene informacije o objektu, ampak kaže na vhod, ki vsebuje prednostno ime danega objekta. Z alternativnimi imeni lahko tvorimo uporabniško prijaznejša imena objektov.

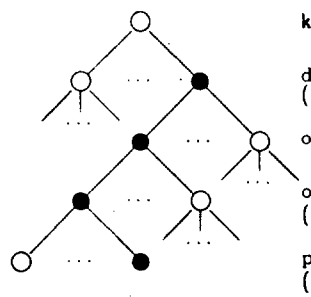
Obratni kazalci – to je kazalci, ki bi kazali iz vhoda (na katerega kaže neki alternativni vhod) na alternativne vhode tega vhoda – ne obstajajo. Zato alternativni vhodi danega vhoda ostanejo v imeniku tudi, če dani vhod odstranimo iz imenika.

4.3 Shema imenika

Shema imenika določa, katera informacija je lahko shranjena v DIB in kakšne so logične relacije med vhodi. Struktura vhodov, tipi informacij, ki jih vhodi vsebujejo ter predstavitev teh informacij so definirani z objektnimi razredi, tipi atributov in sintaksami atributov (slika 4).

Shema obsega množico definicij:

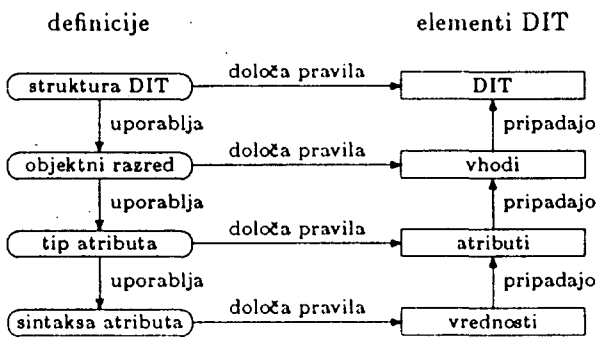
- o *strukturi DIT*,
  ki določajo pravila, po katerih so oblikovana prednostna imena vhodov, ter odnose med posameznimi vhodi v DIT;



	DPI	prednostno ime
korenina	{}	{}
države (C-country)	{C=YU}	{C=YU}
organizacije	{O=IJS}	{C=YU,O=IJS}
organizacijske enote (OU-organisational unit)	{OU=E6}	{C=YU,O=IJS,OU=E6}
posamezni objekti (CN-common name)	{CN=Tvrdy}	{C=YU,O=IJS,OU=E6, CN=Tvrdy}

Slika 3: Hierarhični imenski prostor

- *objektnih razredov*,
  ki določajo obvezne in neobvezne atribute, ki morajo oziroma smejo biti prisotni v nekem vhodu iz danega objektnega razreda;
- *tipov atributov*,
  ki za posamezni atribut določajo njegov objektni identifikator, njegovo sintakso in ali sme imeti več vrednosti;
- *sintaks atributov*,
  ki za vsak atribut določajo ustrezni podatkovni tip in pravila primerjanja.



Slika 4: Shema imenika

Shema imenika je (tako kot DIB) porazdeljena in je lahko v vsakem DSA različna. Upravljalec, ki je pooblaščen za dani DSA, sam določi shemo, ki bo veljavna v delu DIB, za katerega je zadolžen.

5 Servisi imenika

Imenik izvaja servise na zahteve uporabnikov, ki jih zastopajo njihovi DUA. Nekateri servisi omogočajo preiskovanje imenika, drugi pa njegovo spreminjanje. Za vsako zahtevo imenik vedno vrne odgovor, v katerem sporoči rezultat izvedene zahteve, ki pa je odvisen od parametrov podanih v zahtevi. Ti parametri so:

- *servisna krmila*, ki omogočajo uporabnikom, da omejijo uporabo imenika; omejitve se nanašajo na čas dovoljen za izvedbo zahteve, na dimenzijo rezultatov, na način medsebojnega sodelovanja DSA, na prioriteto zahteve,...;
- informacija, potrebna za izvajanje *mehanizmov zaščite*;
- *filtri*, ki se uporabljajo za izbor ustreznih vhodov pri operacijah, ki se nanašajo na več vhodov; filter sestavlja množica pravil za primerjanje, ki so med seboj logično povezana z disjunkcijo, konjunkcijo in negacijo.

Servise (oziroma operacije), ki jih izvaja imenik, razdelimo v tri kategorije:

1. *eno-objektne operacije*:
   so operacije, ki učinkujejo le na en objekt in se izvedejo v DSA, ki vsebuje vhod, ki predstavlja ta objekt:
   - *beri*:
     imenik prebere določeni vhod in vrne nekatere ali vse atribute tega vhoda;
   - *primerjaj*:
     imenik primerja podano vrednost z vrednostjo določenega atributa v vhodu;
   - *dodaj_vhod*:
     imenik doda nov list (končni vhod) v DIT;
     odstrani_vhod:
     imenik odstrani list iz DIT;
   - *spremeni_vhod*:
     imenik izvede na podanem vhodu eno od naslednjih sprememb: doda, odstrani ali zamenja atribut oziroma njegovo vrednost;
   - *spremeni_DPI*:
     imenik spremeni DPI podanega lista;
2. *več-objektni operaciji*:
   sta operaciji, ki delujeta na več vhodih, ki so, ali pa niso vsi v istem DSA:
   - *naštej*:
     imenik vrne seznam neposrednih naslednikov podanega vhoda v DIT;
   - *išči*:
     imenik vrne vse vhode iz določenega dela DIT, ki zadoščajo pogojem navedenim v filtru;
3. *operacija sproščanja*:
   - *prekini*:
     imenik konča z izvajanjem trenutne zahteve, ker povpraševalca rezultati ne zanimajo več.

6 Varnost in zaščita

6.1 Overovljanje

Overovljanje (preverjanje identitete uporabnika) je pomembni razlog za uporabo imenika v aplikacijah. Imenik uporabljamo izvajanje funkcij overovljanja, ker je zanje potrebna globalna podatkovna baza, v kateri so shranjeni javni ključi (public keys) in z njimi povezane informacije.

Za *enostavno overovljanje* s strani imenika zadostuje, da povpraševalec dokaže svojo identiteto (specificirano v imeniku) tako, da navede svoje ime in geslo. Če pa so v imeniku shranjeni zaupni podatki, enostavno overovljanje ne zadošča. Za preprečitev zlorab je potrebno uvesti servis overovljanja, ki nudi večjo varnost (*strogo overovljanje*).

6.2 Nadzor dostopa

Zaščito podatkov pred nedovoljeno uporabo izvedemo z nadzorom dostopa. Za vsak atribut definiramo, kateri uporabnik in pri kateri operaciji lahko uporablja dani atribut.

Nadzor dostopa realiziramo z dostopnimi pravicami za:

- branje zaščitenih atributov;
- zaznavanje zaščite atributov;
- spreminjanje zaščitenih atributov;
- oblikovanje / brisanje atributov oziroma vhodov.

Dostopne pravice se nanašajo na operacije v imeniku. Med izvajanjem servisov mora imenik najprej preveriti dostopne pravice. Če le-te ustrezajo, lahko izvede ustrezno predajo podatkov. Operacije na informaciji, ki zahtevajo posebno overovljanje, lahko izvede le po vnaprejšnji identifikaciji povpraševalca.

7 Porazdeljenost

Porazdelitev DIB (ki navidezno hrani vse vhode imenika na enem mestu) med različne DSA (ki fizično vsebujejo vhode) temelji na principu, da lahko vsak DSA vsebuje enega ali več poddreves v DIT.

Vsakega od vhodov, ki so neposredni nasledniki korenine v DIT, lahko dodelimo nekemu DSA, ki odslej vsebuje popolno informacijo o tem vhodu. Imenujemo ga *izvorni DSA* za ta vhod. Upravljalec tega DSA je zadolžen za vzdrževanje in upravljanje vhoda; pooblastilo za upravljanje vseh ali nekaterih izmed naslednikov tega vhoda, lahko preda drugemu(-im) DSA. Nasledniki tega vhoda postanejo korenine novih poddreves, ki jih na isti način lahko porazdelimo naprej.

Ta postopek omogoča zelo prilagodljivo porazdelitev vhodov: vsak DSA lahko vsebuje poljubno število popolnih ali delnih poddreves. Poddrevesa, ki jih vsebuje posamezni DSA, so si lahko celo disjunktna (korenine med seboj disjunktnih poddreves nimajo istega predhodnika).

7.1 Kopiranje

Kopiranje informacij je v imeniku potrebno za:

1. *zagotovitev visoke uporabnosti imenika*:
   informacija je v imeniku porazdeljena in shranjena na različnih sistemih; če se kateri od teh sistemov pokvari, ali če (iz kakršnihkoli razlogov) ni dosegljiv, potem so objekti imenika, ki se nahajajo v sistemu, nedosegljivi; s stališča uporabnosti imenika je torej nujno, da se nahajajo kopije posameznih informacij na več mestih;
2. *izboljšanje sposobnosti imenika*:
   večina dostopov do imenika se nanaša na iskanje in ne na ažuriranje informacije; več kopij iste informacije omogoča lokalno iskanje – namesto sicer potrebnega vstopanja v oddaljene mreže in s tem povezanih zakasnitev.

Vsak vhod ima natanko eno *izvorno kopijo* (master copy) in morebitne *neizvorne kopije* (non-master copy). Izvorna kopija vhoda leži le v enem DSA – izvornem DSA ostali DSA pa lahko vsebujejo neizvorne kopije tega vhoda. Vzdrževanje in upravljanje nekega vhoda se vedno izvaja v izvornem DSA. Izvorna kopija je vedno popolna kopija vhoda (vsebuje del prednostnega imena in vse atribute).

Neizvorna kopija je bodisi:

- popolna (vsebuje vso informacijo iz vhoda) in se uporablja za optimalnejši (hitrejši in direktni) dostop do informacije; bodisi je
- nepopolna (ne vsebuje vseh atributov; vsebuje na primer le atribute, potrebne za nadaljnje usmerjanje zahtev, ne vsebuje pa atributov, ki opisujejo dani vhod).

V splošnem lahko neizvorno kopijo dobimo na dva načina:

- z dogovorom med dvema DSA, da bo tisti, ki vsebuje izvorno kopijo nekega vhoda pošiljal vse spremembe, ki jih "doživlja" informacija tega vhoda, drugemu DSA. Ta način imenujemo *sledenje* informacije (shadowing);
- če se DSA med izvajanjem operacije dokoplje do informacije, ki je sam ne vsebuje (nima izvorne kopije), se lahko odloči, da to informacijo obdrži. Ta način pridobivanja informacij imenujemo *skladiščenje* informacije (caching).

Uporabnik imenika mora biti obveščen, ali je informacija, ki jo dobi kot odgovor na zahtevo, dobljena iz kopije ali iz originala.

7.2 Znanje

Zaradi porazdeljenosti DIB mora vsak DSA vsebovati neko *znanje*, ki mu omogoča določiti, kateri DSA vsebuje zahtevano informacijo. DSA vsebuje toliko znanja, da je iz vsakega DUA možen dostop do vsakega vhoda. Del znanja uporablja za določitev, ali vsebuje zahtevani vhod v katerem od poddreves; drugi del pa za usmerjanje zahtev na druge DSA.

Poddrevesa, ki jih vsebuje DSA, opišemo s pojmom *konteksta imenovanja*. Kontekst imenovanja je sestavljen iz *kontekstne predpone* (prednostnega imena korenine danega poddrevesa) in iz *referenc*, ki kažejo, kje (v katerem DSA) se nahajajo vhodi danega poddrevesa (*notranje reference* in *reference na naslednike*) in, kje se nahaja predhodnik korenine poddrevesa (*referenca na predhodnika*).

Slika 5 prikazuje DIT, ki je logično razdeljena med pet kontekstov imenovanja (KI-A, KI-B, KI-C, KI-D in KI-E) in ki je fizično porazdeljena med tri sistemske posrednike (DSA-a, DSA-b in DSA-c).

Kontekste imenovanja, ki jih vsebujejo posamezni DSA, v splošnem oblikujemo tako, da lahko zadostijo čimvečjemu številu zahtev (po najrazličnejših operacijah):

- nekatere DSA oblikujemo (na primer) tako, da vsebujejo le vhode, ki predstavljajo področja imenovanja na višjih nivojih (znotraj nekega logičnega dela DIB), pri čemer ni nujno, da vsebujejo tudi vse podrejene vhode;
- druge DSA pa oblikujemo tako, da vsebujejo kontekste imenovanja, ki predstavljajo predvsem končne vhode.

Minimalno znanje

Ker mora imenik zagotoviti dostop do vsakega vhoda, obstaja za vsak DSA tako imenovana *referenčna pot* (reference path), to je zvezno zaporedje referenc, ki vodijo iz danega DSA do vseh kontekstov imenovanja v imeniku. Množico vseh teh referenc in vseh DSA si lahko predstavljamo kot povezan graf; pri čemer minimalno množico referenc, ki še zagotavljajo pravilno obnašanje imenika, obravnavamo kot ogrodje tega grafa. Zmogljivost imenika povečamo, če osnovnemu ogrodju dodamo dodatne prvine (nove reference).

DSA mora vsebovati in vzdrževati:

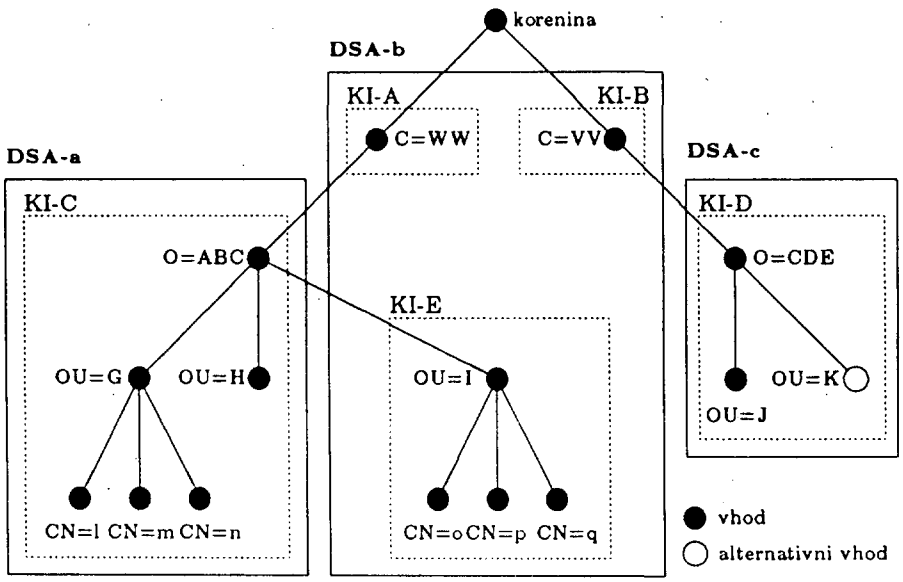
- *znanje o predhodnikih* in
- *znanje o naslednikih* (če seveda obstajajo).

Kontekst korenine

Funkcijo "koreninskega DSA" izvajajo vsi DSA, ki so neposredni nasledniki korenine. Te DSA imenujemo *DSA prvega nivoja*. Vsak DSA prvega nivoja zna simulirati koreninski DSA. To pomeni, da ima popolno znanje o kontekstu korenine. Ker je kontekst korenine kopiran (repliciran) v vsak DSA prvega nivoja, so zanj skupno pristojni vsi (sicer med seboj avtonomno delujoči) upravitelji vseh DSA prvega nivoja, ki postopke za upravljanje konteksta korenine določajo na osnovi medsebojnih sporazumov.

Povzemimo:

- vsak DSA prvega nivoja vsebuje kontekst korenine in zato tudi referenčno pot do vsakega drugega DSA prvega nivoja in



Slika 5: Primer DIT

- vsak DSA, ki ni prvega nivoja, vsebujejo referenco na predhodnika, kar pomeni, da poznamo referenčno pot do poljubnega DSA prvega nivoja.

Dodatno znanje

Optimalnejši dostop do drugih DSA je omogočen z uporabo *navzkrižnih referenc* in *nespecifičnih referenc na naslednike*. Navzkrižne reference vsebujejo "tuje" kontekstne predpone, ki omogočajo direktno interakcijo z DSA, ki vsebuje zahtevani vhod. DSA lahko vsebuje poljubno število navzkrižnih referenc; pridobiva pa jih med običajnim izvajanjem operacij v imeniku. Nespecifične reference na naslednike pa uporabljamo, kadar za neki DSA vemo, da vsebuje naslednike danega vhoda, ne poznamo pa DPI teh vhodov.

Opis posameznih tipov referenc

- *Notranja referenca* vsebuje:
  - del prednostnega imena, ki ustreza vhodu v DIB in
  - kazalec, ki kaže, kje je vhod lokalno shranjen.
- *Referenca na naslednika* vsebuje:
  - del prednostnega imena, ki ustreza neposrednemu nasledniku danega vhoda v DIB,
  - prednostno ime DSA, ki je pristojen za upravljanje naslednika in
  - predstavitveni naslov tega DSA.

  Vse znanje o vhodih, ki jih je dani DSA predal v pristojnost drugim DSA, je v danem DSA predstavljeno v obliki referenc na naslednike.
- *Referenca na predhodnika* vsebuje:
  - prednostno ime DSA, ki vsebuje neposrednega predhodnika korenine danega konteksta imenovanja in
  - predstavitveni naslov tega DSA.

  Vsak DSA, ki ni prvega nivoja, vsebuje natanko eno referenco na predhodnika. Ob oblikovanju novega DSA, ki ni prvega nivoja, moramo le-temu z referenco na predhodnika določiti vsaj minimalno začetno znanje. Dodatno znanje lahko uvedemo kasneje v obliki referenc na naslednike ali/in v obliki navzkrižnih referenc. Ob vključevanju novega DSA prvega nivoja pa moramo le-tega opremiti s kontekstom korenine, ostale (že obstoječe) DSA prvega nivoja pa obvestiti o njegovi vključitvi.
- *Navzkrižna referenca* vsebuje:
  - kontekstno predpono,
  - prednostno ime DSA, ki je pristojen za ta kontekst imenovanja in
  - predstavitveni naslov tega DSA.

  Ta tip reference je neobvezen in služi za optimalnejše razreševanje porazdeljenega imena. DSA lahko vsebuje poljubno število (tudi nič) navzkrižnih referenc.
- *Nespecifična referenca na naslednike* vsebuje:
  - prednostno ime DSA, ki vsebuje enega ali več neposredno sledečih kontekstov imenovanja in
  - predstavitveni naslov tega DSA.

  Tudi ta tip reference je neobvezen. V vsakem kontekstu imenovanja, ki ga vsebuje, ima DSA lahko poljubno število (tudi nič) nespecifičnih referenc na naslednike, ki jih med delovanjem (na primer med postopkom razrešitve imena) upošteva šele, ko neuspešno "obdela" že vse specifične reference (notranje in na naslednike).

7.3 Načini medsebojnega sodelovanja DSA

DSA lahko medseboj sodelujejo na enega od naslednjih treh načinov.

1. *Veriženje* (chaining)
   je način, pri katerem DSA, ki sam ne more razrešiti zahteve, preda zahtevo drugemu (le enemu!) DSA. To pomeni, da DSA preide na izvajanje oddaljene operacije na tistem DSA, za katerega pozna njegove kontekste imenovanja, podane bodisi z navzkrižno referenco, bodisi z referenco na naslednike, bodisi z referenco na predhodnika.
2. *Dodeljevanje večim* (multi-casting)
   je način, pri katerem DSA, ki sam ne more razrešiti zahteve, preda identično zahtevo enemu ali več drugim DSA hkrati ali zaporedno; to je preide na hkratno ali zaporedno izvajanje identične oddaljene operacije na tistih DSA, za katere vsebuje nespecifična referenca na naslednike. Običajno je sposoben nadaljevati izvajanje oddaljene operacije le en izmed sodelujočih DSA, vsi ostali pa vrnejo sporočilo, da izvedba ni možna.
3. *Namigovanje* (hinting)
   je način, pri katerem DSA, ki sam ne more razrešiti zahteve, vrne povprašujočemu (DUA ali DSA) namig, kateri DSA je bližji rešitvi. Namig vsebuje referenco (možna je vsaka referenca) na enega ali več DSA.

7.4 Upravljanje znanja

Če spremembe v enem DSA povzroče, da postanejo reference v drugih DSA napačne, je treba vse reference (na naslednike, na predhodnika in nespecifične na naslednike) ažurirati na osnovi medsebojnih dogovorov upraviteljev ustreznih DSA.

7.5 Protislovnost znanja

Vrsta protislovnosti (znanja) in zaznava te protislovnosti je različna za različne tipe referenc:

- *navzkrižne reference in reference na naslednike*:
  ta vrsta referenc je neveljavna, če referencirani DSA lokalno ne vsebuje konteksta imenovanja s kontekstno predpono, navedeno v referenci; protislovje zazna proces za razrešitev imena med iskanjem konteksta imenovanja (glej razdelek 8.2).
- *reference na predhodnika*:
  napačna referenca na predhodnika kaže na DSA, ki ne vsebuje konteksta imenovanja, ki bi bil nadrejen vsem kontekstom imenovanja, ki jih lokalno vsebuje dani DSA.

Pri veriženju se DSA, ki vsebuje napačno referenco, zave protislovnosti znanja potem, ko kot odgovor na verižen zahtevo dobi sporočilo o napačni referenci. Enak odgovor dobi pri namigovanju povpraševalec, ko hoče uporabiti (od drugega DSA dobljeni) napačni namig. Pri tem pa se DSA, ki je posredoval napačni namig, ne zaveda protislovnosti svojega znanja.

Potem, ko DSA zazna prisotnost napačnih referenc, skuša napake odpraviti in ponovno vzpostaviti neprotislovno stanje. To doseže tako, da (na primer) protislovne navzkrižne reference bodisi:

- pobriše, bodisi
- jih zamenja s pravilnimi, ki jih dobi s (ponovno) uporabo ustreznega mehanizma za pridobivanje navzkrižnih referenc.

7.6 Primer oblikovanja znanja

Slika 6 kaže, kakšno informacijo o znanju vsebuje in vzdržuje DSA-b s slike 5. Uporabljene so naslednje oznake:

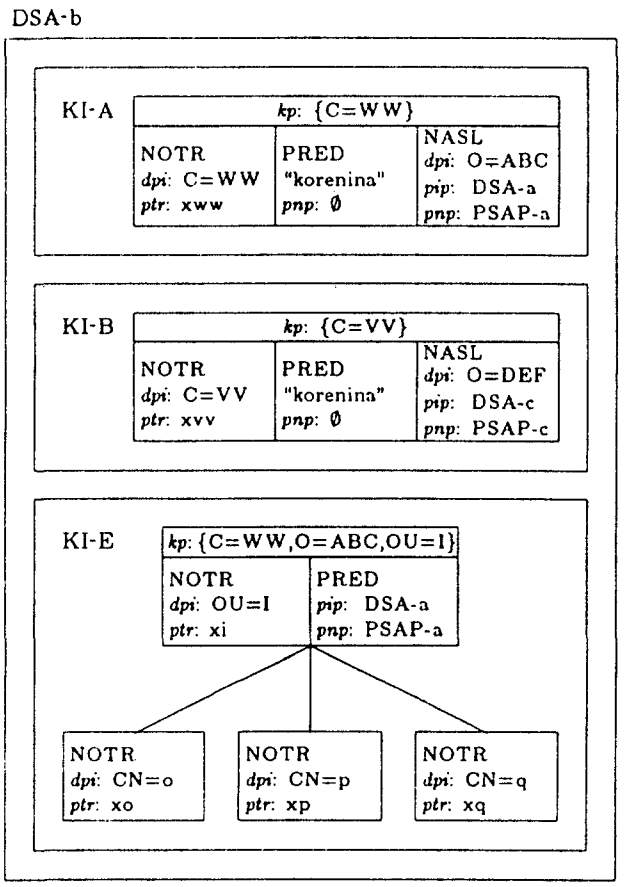
NOTR:	notranja referenca
PRED:	referenca na predhodnika
NASL:	referenca na naslednike
kp:	kontekstna predpona
dpi:	del prednostnega imena
ptr:	kazalec na lokalno pozicijo vhoda
pip:	prednostno ime posrednika
pnp:	predstavitveni naslov posrednika
DSA-n:	prednostno ime posrednika "DSA-n"
PSAP-n:	predstavitveni naslov posrednika "DSA-n"

Znanje, ki ga vsebuje DSA-b, sestavljajo:

kontekstne predpone:	{C=WW} za kontekst KI-A; {C=VV} za kontekst KI-B in {C=WW,O=ABC,OU=I} za kontekst KI-E.
navzkrižne reference:	∅
reference na predhodnika:	
(za kontekst KI-E):	na DSA-a s PSAP-a;
notranje reference:	
(za kontekst KI-A):	na vhod {C=WW}, na poziciji xww;
(za kontekst KI-B):	na vhod {C=VV}, na poziciji xvv;

(za kontekst KI-E):	na vhod {C=WW, O=ABC, OU=I}, na poziciji xi;
	na vhod {CN=o}, na poziciji xo;
	na vhod {CN=p}, na poziciji xp in
	na vhod {CN=q}, na poziciji xq.

reference na naslednike:	
(za kontekst KI-A):	na vhod {C=WW, O=ABC} v DSA-a s PSAP-a in
(za kontekst KI-B):	na vhod {C=VV, O=DEF} v DSA-c s PSAP-c.
nespecifične reference na naslednike:	∅



Slika 6: Znanje v DSA-b

8 Delovanje porazdeljenega imenika

Delovanje imenika lahko opazujemo iz dveh zornih kotov, ki nam združena nudita popolno sliko o delovanju. Prvi zorni kot je osredotočen na:

a) *DSA*,
   ki podaja natančno sliko, kako se izvajajo postopki znotraj vsakega posameznega DSA; drugi pa na
b) *operacijo*,
   ki prikazuje strukturo posameznih operacij, ki se v splošnem izvajajo v več DSA.

Vsak DSA mora izvajati:

1. akcije, ki so potrebne za realizacijo vsake posamezne operacije in
2. dodatne akcije, ki so potrebne za porazdelitev dobljenih rezultatov med druge DSA.

8.1 Obnašanje porazdeljenega imenika

Obnašanje porazdeljenega imenika kot celote je vsota obnašanj posameznih DSA, ki je opisano v razdelku 8.2.

Za porazdeljeno okolje je tipično, da:

- DSA obravnava vsako operacijo kot *prehodni dogodek*: ko dobi (od DUA ali nekega drugega DSA) zahtevo po operaciji, obdela dano zahtevo (kolikor jo pač zmore), nakar jo usmeri k drugemu DSA v nadaljnjo obdelavo in da
- se celotna obdelava neke zahteve izvede v več fazah, ki so skupne vsem operacijam; pri izvajanju vsake od teh faz pa sodeluje več DSA.

Faze izvajanja operacije

Vsaka operacija se v imeniku izvede v treh fazah:

a) *faza razreševanja imena*,
v kateri se na osnovi imena objekta, na katerem naj se izvede operacija, določi, kateri DSA vsebuje vhod, za ta objekt;

b) *faza vrednotenja*,
v kateri se zahtevana operacija (na primer *beri*) tudi dejansko izvede;

c) *faza združitve rezultatov*,
v kateri se povprašujočemu DUA vrnejo rezultati operacije; v primeru veriženja so v to fazo vključeni vsi DSA, ki so posredovali zahtevo drugemu DSA med eno od obeh ali med obema predhodnima fazama.

Vsaka operacija vsebuje argument *napredek operacije*, to je informacijo o fazi, do katere se je operacija, ki jo izvaja več DSA, že izvedla. Vsak DSA potem, ko izvede ustrezni del operacije in preden odda operacijo v nadaljnjo obdelavo drugim DSA, označi, do kje je operacija že izvedena.

Veljavnost zahteve

Po prejemu zahteve za izvedbo neke operacije se DSA najprej prepriča, ali je *zahteva veljavna*, to je, ali se del operacije, ki naj bi jo izvedel, da izvesti. Okoliščine, v katerih se pojavijo zanke v DIT (ob napačni uporabi alternativnih imen, ali ob uporabi napačnih referenc), namreč lahko povzroče, da dobi DSA zahtevo, ki je ne more izvesti.

Stanje operacije

Izvajanje operacije in pogoje, pri katerih pride do zančenja, nadzoruje DSA s *stanjem operacije*, ki je določeno:

- z imenom DSA, ki trenutno izvaja operacijo,
- z imenom objekta, na katerega se nanaša operacija ter
- s parametrom *napredek operacije*.

Poleg trenutnega stanja operacije mora DSA poznati tudi vsa predhodna stanja dane operacije. Predhodna stanja operacije hrani v tako imenovani *beleženi informaciji*, ki jo kot argument operacije preda (drugim DSA) hkrati z operacijo.

Zančenje

Do zančenja (glede na posamezno operacijo) pride vedno, ko je trenutno stanje operacije enako nekemu od predhodnih stanj operacije. Pri obravnavanju zank ločimo dve strategiji:

- *zaznava zanke*:
  DSA preveri dohodno operacijo in javi napako, če najde zanko v njej;
- *izogib zanki*:
  DSA preveri, ali bo operacija, ki naj bi jo poslal naprej (drugemu DSA), povzročila zanko.

Servisna krmila

Omejimo se na obravnavo servisnih krmil, ki so bistvena za pravilno izvajanje operacij v porazdeljenem okolju.

- *prepovedano veriženje*
  DSA upošteva to servisno krmilo pri določanju načina posredovanja operacij drugim DSA. Če je postavljeno, uporabi namigovanje, sicer se odloči med veriženjem in namigovanjem (na osnovi lastne opremljenosti).
- *časovna omejitev*
  DSA skrbi, da ne preseže navedene vrednosti. Ko dobi od DUA zahtevo za operacijo, dobi hkrati tudi časovno omejitev, za izvedbo celotne operacije. V primeru nadaljnjega veriženja operacije preda naslednjemu DSA tudi časovno omejitev, ki je enaka času, ki je še ostal glede na prvotno (v DUA) specificirano omejitev.
- *prostorska omejitev*
  DSA skrbi, da seznam rezultatov ne preseže navedene velikosti. Ob veriženju zahteve drugim DSA, jim posreduje nespremenjeno začetno vrednost omejitve, ki ostane nespremenjena tudi v primeru, ko je potrebno zahtevo razstaviti v več podzahtev: vsaka podzahteva vsebuje celotno začetno vrednost omejitve. Ko DSA dobi rezultate, jih pregleda, upošteva omejitev in vrne skupni odgovor, ki ne presega te omejitve. V odgovoru označi tudi, če je bila v posameznih rezultatih omejitev presežena.
- *prioriteta*
  Pri vseh načinih širjenja operacij DSA skrbi, da se operacije izvajajo v vrstnem redu, ki je naveden v tem servisnem krmilu.
- *lokalna izvedba*
  Operacija je omejena na izvajanje v danem DSA; ni je mogoče širiti na druge DSA.

Dovršitev operacij

Vsak DSA, ki začne ali, ki povzroči širjenje kakšne operacije, mora hraniti podatek o obstoju *nedovršene operacije* vse dokler, od drugih DSA ne dobi rezultatov ali sporočil o napaki; oziroma dokler se ne izteče maksimalni dovoljeni čas za izvedbo operacije. Ta zahteva se nanaša na vse operacije, na vse načine širjenja operacij in na vse faze operacij.

8.2 Obnašanje posameznega DSA

Notranje obnašanje DSA prikazuje slika 7, ki kaže, v kakšnem odnosu je glavni nadzorni postopek (razdelilnik operacij) z ostalimi postopki (razrešitev imena, iskanje konteksta imenovanja, lokalna razrešitev imena, ovrednotenje, ovrednotenje enega objekta, ovrednotenje več objektov, združitev rezultatov).

V nadaljevanju na kratko opišimo vsakega od postopkov, ki skupno tvorijo proces, ki predstavlja obnašanje DSA.

Razdelilnik operacij

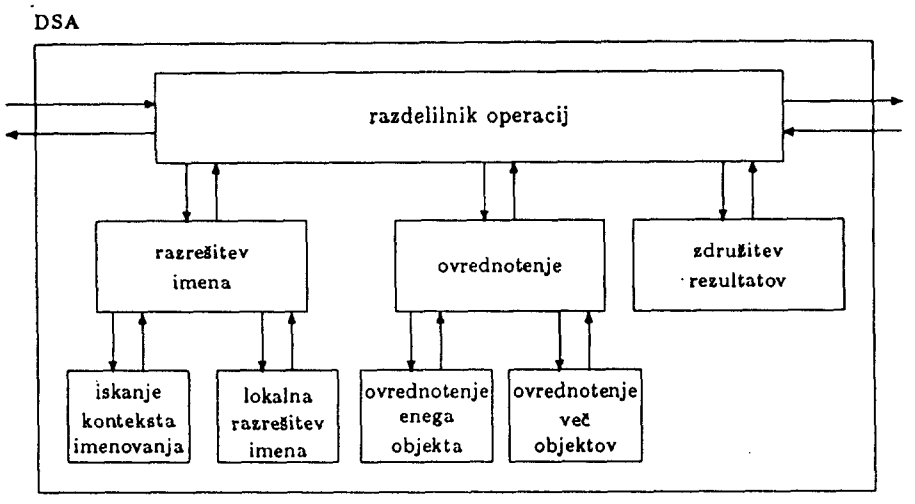
Po prejemu zahteve za izvedbo neke operacije razdelilnik operacij najprej preveri veljavnost zahteve. Če ne najde niti zanke niti overovitvene napake, sproži postopek *razrešitev imena*, ki vrne enega od naslednjih odgovorov:

- sporočilo *ime najdeno*
  v tem primeru razdelilnik operacij pokliče postopek *ovrednotenje*, nakar čaka na rezultate;
- *referenco*
  v tem primeru razdelilnik operacij (odvisno od načina širjenja operacij) bodisi:
  - izvede veriženje ali dodeljevanja večim, s katerim preda operacijo drugemu(-im) DSA, nakar čaka na rezultate; bodisi
  - vrne povprašujočemu (DUA ali DSA) namig, ki je vsebovan v referenci;
- sporočilo *napaka*
  v tem primeru razdelilnik operacij vrne povprašujočemu (DUA ali DSA) enako sporočilo o napaki.

Ko razdelilnik operacij dobi vse rezultate: notranje (od postopka za ovrednotenje) in zunanje (od drugih DSA), jih s postopkom *združitev rezultatov* uredi in vrne povprašujočemu (DUA ali DSA).

Razrešitev imena

Postopek kliče najprej podpostopek *iskanje konteksta imenovanja* in v primeru, da najde ustrezni lokalni kontekst imenovanja, pokliče še podpostopek *lokalna razrešitev imena*. Dobljene rezultate (sporočilo *ime najdeno*, referenco ali sporočilo o napaki) vrne razdelilniku operacij, razen v primeru, ko postopek za lokalno razrešitev imena sporoči, da je izvedel prevedbo alternativnega imena; tedaj postopek za razrešitev imena ponovno izvede celotno analizo (aktivira postopek za iskanje konteksta imenovanja, ...).



Slika 7: Notranje obnašanje DSA

Iskanje konteksta imenovanja

Postopek primerja *predlagano ime* (to je ime objekta, na katerem naj bi se izvedla operacija) s kontekstnimi predponami – išče kontekstno predpono, ki se vsaj delno ujema s predlaganim imenom, če jih je več izbere tisto, ki se najbolj ujema. Če ne najde nobenega ujemanja, primerja predlagano ime s kontekstnimi predponami v navzkrižnih referencah (seveda, če le te obstajajo). Če tudi to ne uspe, poišče referenco na predhodnika. V odgovoru vrne bodisi lokalni kontekst imenovanja, bodisi navzkrižno referenco, bodisi referenco na predhodnika; v vsakem primeru pa nastavi parameter *napredek operacije* tako, da kaže, da se izvaja faza razreševanja imena.

Lokalna razrešitev imena

Postopek primerja dele prednostnega imena (DPI) predlaganega imena z lokalnimi vhodi in vrne sporočilo *ime najdeno*, če se vsi DPI predlaganega imena ujemajo. Če kateri od DPI ne ustreza nobenemu od lokalnih vhodov, potem poskuša najti ustrezno specifično referenco oziroma nespecifično referenco na naslednike, ki jo nato posreduje postopku za razrešitev imena. Če pri primerjanju naleti na alternativno ime, vrne postopku za razrešitev imena sporočilo *prevedba alternativnega imena*; sporočilo *ime najdeno* pa le, če so v trenutku odkritja alternativnega imena, že vsi DPI uspešno obdelani (primerjani).

Ovrednotenje

Postopek izvede zahtevano operacijo na objektu. Odvisno od vrste operacije se deli na dva podpostopka:

- **ovrednotenje enega objekta**:
  deluje nad eno-objektnimi operacijami: *beri, primerjaj, dodaj_vhod, odstrani_vhod, spremeni_vhod* in *spremeni_DPI*; postopek dejansko najde, primerja ali spreminja atribute.
- **ovrednotenje več objektov**:
  deluje nad operacijama *išči* in *naštej*; pri tem uporablja filtre in (če je potrebno) razbije zahtevo na podzahteve, ki jih posreduje razdelilniku operacij, ki jih odda ustreznim DSA.

Združitev rezultatov

Razdelilnik operacij aktivira postopek za združitev rezultatov, kadar obstajajo zunanji rezultati. Seveda lahko hkrati obstaja tudi notranji rezultat. DSA hrani vse rezultate (in vse napake) dokler se postopek za združitev rezultatov v celoti ne zaključi.

Če so zunanji rezultati posledica:

- *veriženja*,
  je postopek za združitev rezultatov trivialen.
- *dodeljevanja večim*,
  se lahko zgodi, da rezultati bodisi ne obstajajo, bodisi obstaja en rezultat, bodisi obstaja več identičnih rezultatov; obstajajo pa lahko tudi napake. Če kakšen od rezultatov obstaja, postopek vrne enega izmed njih, sicer javi napako.
- *razstavitve zahteve*,
  postopek združi rezultate tako, da oblikuje unijo posameznih rezultatov. V primeru, ko so prisotni tako rezultati kot tudi napake, lahko bodisi javi napako, bodisi vrne nepopolni rezultat.

8.3 Izvajanje operacij

Vse *eno-objektne operacije* DSA izvede na enak način z naslednjim zaporedjem dogodkov:

1. aktivira razdelilnik operacij;
2. izvede postopek za razrešitev imena (da najde lokacijo objekta, na katerega se operacija nanaša);
3. izvede postopek za ovrednotenje enega objekta; pri tem
4. upošteva servisna krmila, ki določajo omejitve specificirane s strani uporabnika;
5. vrne rezultate tistemu DUA ali DSA, ki mu je posredoval zahtevo.

Obe *več-objektni operaciji* izvede DSA z naslednjim zaporedjem dogodkov:

1. aktivira razdelilnik operacij;
2. izvede postopek za razrešitev imena;
3. ko najde lokacijo objekta, na katerega se operacija nanaša, izvede postopek za ovrednotenje več objektov;
4. če le-ta izvede razstavitev zahteve na podzahteve, razdelilnik operacij začasno "shrani" lokalne rezultate in počaka, da dobi verižene odgovore, nakar aktivira postopek za združitev rezultatov;
5. med obravnavo operacije preveri in upošteva servisna krmila, da ostane v dovoljenih mejah;
6. vrne rezultate ali sporočila o napakah tistemu DUA ali DSA, ki mu je posredoval zahtevo.

9 Zaključek

V ([COS88]) uvrščajo pri klasifikaciji storitev na *začetne in bodoče* imenik med bodoče storitve; to pa ne zato, ker ni nujno potreben, ampak zato, ker standardi zanj še niso dovolj dodelani. Trenutni standard:

- ne določa, kateri atributi morajo biti uporabljeni pri oblikovanju DPI, predlaga le, kakšna naj bo oblika imen; ne standardizira pa sheme (pravil o strukturi DIT);
- ne omogoča izvedbe operacij *dodaj_vhod* in *odstani_vhod* v porazdeljenem okolju: operaciji sta omejeni na primer, ko sta hkrati vhod, ki naj bi bil dodan oziroma odstranjen in neposredni predhodnik tega vhoda oba v istem DSA;
- ne podpira kopiranja (oziroma natančneje povedano: upravljanja kopij);
- prepušča nadzor dostopa vsakemu DSA kot predmet lokalne izvedbe;
- ne vsebuje globalne zasnove za pridobivanje in upravljanje znanja; operacije, ki vplivajo na konsistentnost znanja izven meja DSA, so dovoljene le na osnovi medsebojnega sporazuma med upravljalci ustreznih DSA.

Kljub vsemu pa so vsaj posamezni deli standarda dovolj stabilni, da je na njihovi osnovi upravičeno začeti z izvedbo pilotskih projektov za imenik (tak je na primer projekt THORN). Pilotski projekti naj bi služili kot demonstacijski projekti za potrditev in izboljšanje starih ter za razvijanje in kasnejšo standardizacijo novih konceptov.

10 Viri

[Ami88] *The Amigo MHS+ Report*, Cost 11ter, Editors: Nottingham University, March 1988

[Ben87] Benford S., *Mapping the DIB to a Relational Schema: Name Representation and Verification*, Internal report, Department of Computer Science, University of Nottingham, 1987

[Ben88] Benford S., *Navigation and Knowledge Management within a Distributed Directory System*, Internal report, Department of Computer Science, University of Nottingham, 1988

[Bon88] Bonač M., *Grupna komunikacija z elektronsko pošto*, Magistrska naloga, Univerza Edvarda Kardelja, Fakulteta za elektrotehniko, Ljubljana 1988

[But86] Butscher B., Tschichholz M., *Directory System, Requirements and the CCITT-Model*, HMI, Berlin, 1986

[Cho86] Chong K. N., Chew E. K., McDonell K. J., *Implementation Considerations of a Name Validation Function for Distributed Directory Services*, Proc. of the IFIP TC 6 International Symposium on Computer Message Systems – 85, North-Holland, 1986

[Com86] Comer D. E., Peterson L. L., *A Model of Name Resolution in Distributed Systems*, IEEE Proc. 6th ICDCS, May 1986

[COS88] COSINE Specification Phase, *Study on Directories*, RARE WG3 Report 7.1, Biber A., Forster J., Softlab, Gmbh, Munich, July 1988

[ECM85] ECMA TC 23, *OSI Directory Access Service and Protocol*, ECMA TR/32

[Hui88] Huitema C., *A Proposal for a naming, data structures, and name date distribution in RARE*, RARE WG3, Internal Report, May 1988

[ISO87] ISO/CCITT, *Directory Convergence Document*, Part 1–8, Gloucester, November 1987

[Kil88] Kille S., *The THORN Large Scale Pilot Exercise*, RARE WG3 Report, May 1988

[Lam86] Lampson B. W., *Designing a Global Name Service*, Proc. 5th ACM Symposium on Principles of Distributed Computing, 1986

[Lan86] Lantz K. A., Eighoffer J. L., Hitson B. L., *Towards a Universal Directory Service*, ACM Operating Systems Review, Vol. 20, No. 2, April 1986

[Lat86] Latella D., *A Model for a Distributed Directory System: Specification of a Distributed Algorithm for Name Resolution*, CNUCE, Internal Report C86-08, July 1986

[Lat88] Latella D., Lenzini L., *The OSIRIDE Directory System: Status and Perspectives*, Computer Standards & Interfaces, Vol.7, No.1/2, 1988

[OSN87] *Directory Standardisation Rethink Results in a more Limited Service*, The Open Systems Newsletter, Vol.1, Issue 2, April 1987

[Par86] Pareta J. S., Huitema C. *THORN – Directory Data Structure and Name Assessment Algorithms*, INRIA, October 1986

[Sir87] Sirovich F., *The ESPRIT Directory System*, ESPRIT 87 – Achievements and Impact, CEC, ed. #2, North Holland, September 1987

[Ter86] Terry D. B., *Stucture-free Name Management for Evolving Distributed Environment*, IEEE Proc. 6th ICDCS, May 1986

[Tvr88] Tvrdy H., *Mehanizmi kopiranja informacij v porazdeljenem elektronskem imeniku*, IJS-DP 5116, Ljubljana 1988

[VER86] *VERDI – A Distributed Directory System for German Research Network (DFN)*, Report, Editors: Santo H., Tschichholz M., July 1986

[VER88] *VERDI II, Konzeption für den Einsatz des Standardisierten Directory Systems im Deutschen Forschungsnetz*, Bonacker K., Schüth H., Thoma G., Tschichholz M., Wenzel O., Meyer A., Tadje R., Zwischenbericht, Juni 1988

[Zat88] Zatti S., Janson P. *Interconnecting OSI and Non-OSI Networks using an Integrated Directory Service*, Computer Networks and ISDN Systems, Vol.15, No.4, 1988