UNIVERZITET U BEOGRADU

FAKULTET ORGANIZACIONIH NAUKA

Ivan M. Bojičić

MODELOM VOĐEN RAZVOJ SKLADIŠTA PODATAKA ZASNOVANOG NA DATA VAULT PRISTUPU

doktorska disertacija

Beograd, 2017.

UNIVERSITY OF BELGRADE

FACULTY OF ORGANIZATIONAL SCIENCES

Ivan M. Bojičić

MODEL-DRIVEN DEVELOPMENT OF DATA VAULT BASED DATA WAREHOUSES

Doctoral Dissertation

Belgrade, 2017.

Mentor:

prof. dr Zoran Marjanović, redovni profesor,

Fakultet organizacionih nauka, Univerzitet u Beogradu

Članovi komisije:

prof. dr Milija Suknović, redovni profesor,

Fakultet organizacionih nauka, Univerzitet u Beogradu

prof. dr Milica Vučković, vanredni profesor,

Fakultet organizacionih nauka, Univerzitet u Beogradu

prof. dr Vladan Jovanović, redovni profesor,

Allen E. Paulson College of Engineering and Information Technology, Georgia Southern University

prof. dr Ivan Luković, redovni profesor,

Fakultet tehničkih nauka, Univerzitet u Novom Sadu

Datum odbrane:

_____

Datum promocije:

_____

Zahvalnica

Veliku zahvalnost želim da izrazim mentoru ***Prof. dr Zoranu Marjanoviću*** *na poverenju i idejama koje mi je pružio i što je svojim velikim iskustvom, smernicama i savetima vodio celokupan proces izrade rada.*

Izuzetnu i neizmernu zahvalnost želim da izrazim ***Prof. dr Vladanu Jovanoviću*** *na inicijalnoj ideji i kasnijim razmatranjima i konstruktivnim razgovorima koji su u velikoj meri uticali na konačan sadržaj i kvalitet rada.*

Zahvaljujem se ***Prof. dr Miliji Suknoviću***, ***Prof. dr Milici Vučković*** *i* ***Prof. dr Ivanu Lukoviću*** *na njihovoj podršci i sveukupnoj saradnji koju smo ostvarili tokom izrade ovog rada.*

Zahvaljujem se mojim prijateljima i kolegama ***dr Marku Petroviću***, ***Ognjenu Turkoviću*** *i* ***dr Nini Turajlić*** *na pomoći koju su mi nesebično pružali tokom izrade disertacije.*

Posebnu zahvalnost želim da izrazim supruzi ***Sanji*** *na neiscrpnom razumevanju i strpljenju svih ovih godina, kao i deci* ***Matiji, Aleksi*** *i* ***Mili*** *uz izvinjenje za sve petice, reči i prve korake koje sam propustio.*

Ivan Bojičić

Beograd, februar 2017. godine

MODELOM VOĐEN RAZVOJ SKLADIŠTA PODATAKA

ZASNOVANOG NA DATA VAULT PRISTUPU

Rezime

U tezi je razmatrano više problema vezanih za projektovanje i razvoj skladišta podataka, kao što su:

- neusaglašenost skladišta podataka sa izvorima podataka, nastala kao rezultat permanentnih promena strukture izvora,
- nekompletnost podataka u skladištu podataka,
- heterogenost modela izvora i njihova semantička neusaglašenost
- nepostojanje standardnog konceptualnog modela i modela strukture skladišta podataka

Usled potrebe za prevladavanjem datih problema, prvenstveni cilj teze je da se formalizuje empirijski Data Vault model i da se definiše odgovarajuće okruženje za primenu modelom vođenog razvoja skladišta podataka.

Definisan je C-DV metamodel sa dvojakom namenom. Kao okvir za konceptualno modelovanje skladišta podataka i kao jezik koji može da obuhvati semantiku različitih konceptualnih modela. Pored toga, C-DV metamodel se koristi kao međumodel automatskih transformacija iz izvornih u ciljni model skladišta podataka.

Definisan je L-DV metamodel koji opisuje integrisano skladište podataka i pohranjuje platformski nezavisne verzije konkretnih modela. Takođe, data su pravila transformacije iz C-DV modela u L-DV model. Konkretan L-DV model predstavlja integrisan model korporativnog skladišta podataka, koje sadrži strukturu svih modela izvora sa svim promenama tih modela.

Prilagođavanje L-DV modela konkretnoj tehnološkoj platformi se vrši transformacijom u odgovarajući P-DV model. P-DV metamodel je definisan na taj

način da obuhvata dobre osobine postojećih modela podataka skladišta podataka. U tezi su data i pravila transformacije iz L-DV modela u ciljni P-DV model.

Dat je predlog metodološkog okvira baziranog na modelom vođenom razvoju. Definisan je skup postupaka koji upravljaju razvojem skladišta podataka korišćenjem definisanih metamodela koji rezultuju ciljnom arhitekturom skladišta podataka.

Ključne reči:
Skladište podataka, Data Vault, Modelom vođen razvoj, Model Domen/Preslikavanje

Naučna oblast:
Tehničke nauke

Uža naučna oblast:
Informacioni sistemi

UDK broj:
004.6

MODEL-DRIVEN DEVELOPMENT OF
DATA VAULT BASED DATA WAREHOUSES

Abstract

Several issues, related to the design and development of data warehouses, are analyzed in this thesis:

- inconsistency between the data warehouse and data sources due to the permanent changes in the structure of the data sources,
- incompleteness of the stored data,
- heterogeneity of data source models and their semantic inconsistency,
- absence of standardized conceptual or structural data warehouse models.

In view of the need for overcoming these issues the primary objective of this thesis is to formalize an empirical Data Vault model, and define the appropriate environment for applying model-driven data warehouse development.

A C-DV metamodel is defined and its purpose is twofold: as a framework for the conceptual modeling of data warehouses, and as a language which can encompass the semantic heterogeneity of various conceptual models. Furthermore, the C-DV metamodel can be used as an intermediate model in the automated transformation of source models into target data warehouse models.

An L-DV metamodel is defined for describing the integrated data warehouse and storing platform-independent versions of concrete models. The rules for transforming C-DV models into L-DV models are also given. A concrete L-DV model represents an integrated enterprise data warehouse model, incorporating the structure of all source models, along with the changes in these models.

The mapping of a concrete L-DV model to a particular technological platform is accomplished through its transformation into a corresponding P-DV model. The P-DV metamodel is defined to encompass the advantages of existing data warehouse

data models. The rules for transforming an L-DV into a target P-DV model are also given.

A methodological framework based on model-driven development is proposed. A concrete approach, governing the development of a data warehouse using the introduced metamodels, which results in a target data warehouse architecture, is defined.

Keywords:
Data Warehouse, Data Vault, Model Driven Development, Domain/Mapping Model

Scientific Area:
Technical Sciences

Specific Scientific Area:
Information Systems

UDK Number:
004.6

SADRŽAJ

1 UVOD

Skladište podataka (eng. *Data Warehouse*, u daljem tekstu *DW*) održava skup istorijskih podataka koji predstavljaju niz stanja sistema prouzrokovanih dogadjajima ili aktivnostima koje se obavljaju u organizaciji. Ovi podaci se periodično prikupljaju, najčešće iz više različitih izvora, kako bi se izvršila analiza poslovanja, pratili trendovi i definisala strateška opredeljenja u organizaciji. Takođe, pomoću analize podataka u DW, moguće je praćenje ispunjenosti postavljenih operativnih i strateških ciljeva u organizaciji ili njenim delovima.

Usled raznovrsnosti i promenljivosti poslovnih funkcija, njihovih međusobnih veza i neminovne neusklađenosti ciljeva pojedinih delova organizacije, skladište podataka treba da zadovolji određene kriterijume kako bi se na adekvatan način podržao proces odlučivanja, planiranja i usklađivanja u datom poslovnom sistemu. Stoga je neophodno da skladište podataka bude jasno orijentisano ka pojedinačnim poslovnim funkcijama i organizaciji u celini na taj način da nedvosmisleno može da iznađe stanje sistema u datom trenutku vremena. Takođe, skladište podataka treba da obuhvati i niz stanja sistema u vremenskom periodu kako bi se omogućila analiza i upoređivanje istorijskih stanja, a sa druge strane i predviđanje i planiranje budućih stanja sistema. Pored toga, neophodno je da skladište podataka ima mogućnost integracije podataka usled heterogenosti izvora, pri čemu se integracija vrši na taj način da se podaci prikupljaju bez suštinskih izmena, odnosno bez promene semantike uvezenih podataka.

Zbog svega navedenog, skladište podataka se može definisati i kao model realnog sistema koji predstavlja skup stanja tog sistema u određenom vremenskom periodu. Stalne promene (organizacione, legislativne, funkcionalne i dr.) kojima je poslovni sistem izložen, odražavaju se i na odgovarajuće skladište podataka. Stoga je jedan od osnovnih problema razvoja i održavanja DW nekonzistentnost realnog sistema i skladišta podataka koja se vremenom povećava. Problem se dodatno usložnjava jer

ne postoji jasno definisan metod koji bi promene u realnom sistemu (izvorima) na formalan način transformisao u promene na strani skladišta podataka. U ovom radu će biti dat metod i odgovarajući modeli pomoću kojih će se sve promene na modelima izvora adekvatno transformisati u ciljni model skladišta podataka. Sve promene u strukturi skladišta podataka će se svoditi na nadogradnju postojećeg modela, a ne i na izmene postojećih koncepata.

Sledeći problem sa kojim se realizacija skladišta podataka suočava je kompletnost podataka. Način na koji se podaci najčešće skladište narušava integrativnu funkciju skladišta podataka. Naime, uobičajeni postupak je da se unapred vrši priprema i izbor podataka koji će biti skladišteni u DW. Postupak rezultuje time da se između podataka iz više različitih izvora koji se odnose na isti koncept realnog sistema, na osnovu različitih kriterijuma, najčešće pri ETL procesu, bira jedan koji će da bude istinit, tj. zabeležen u skladištu podataka. Pri tome se svi podaci koji ne odgovaraju 'istini' odbacuju, tj. ne postanu deo skladišta podataka. Na taj način skladište podataka ima samo deo podataka izvora, pri čemu je kasnije nemoguće analizirati sve vrednosti koje postoje u izvorima. U radu će biti definisan odgovarajući model koji omogućava čuvanje svih vrednosti svih modela izvora (i njihovih verzija) bez gubitka informacija korišćenjem ELT pristupa. To praktično znači da će u skladištu podataka biti uvezeni svi podaci izvora na odgovarajući način, a da će se sve transformacije podataka vršiti nakon te operacije.

Naredni problem koji će u radu biti razmatran je i heterogenost modela izvora i njihova semantička neusaglašenost. Transakcioni sistemi se realizuju korišćenjem različitih metamodela za opis i implementaciju sistema, i nad raznovrsnim tehnološkim platformama. Stoga je jedan od osnovnih zahteva koji se postavlja pred skladište podataka omogućavanje integracije modela izvora. U ovom radu će biti definisan model zasnovan na empirijskom Data Vault modelu koji može da obuhvati specifikaciju raznorodnih izvora podataka jedinstvenim jezikom čime se omogućava integracija izvornih modela i automatska transformacija izvora podataka do nivoa izvršivog objedinjenog skladišta podataka.

Pored navedenih, još jedan od osnovnih zahteva je da skladište podataka treba da omogući praćenje prošlih, trenutnih i budućih stanja objekata i događaja, kao i trenutaka kada su se te promene dešavale. Neophodno je da skladište podatak prati stanje objekata po bar dve vremenske dimenzije: vremenu važenja i vremenu ažuriranja. Vreme važenja predstavlja trenutak ili period u kojem je neki koncept realnog sistema važeći tj. validan. Vreme ažuriranja predstavlja trenutak ili period u kojem je koncept realnog sistema uveden u skladište podataka ili je došlo do njegove izmene. U radu se prikazuju postojeći modeli podataka sa ovog stanovišta i razmatraju njihovi nedostaci, i daje se predlog modela koji može da ispuni zahteve vezane za temporalni aspekt skladišta podataka.

Na osnovu navedenog, prvenstveni cilj ovog rada je formalizacija empirijskog Data Vault modela i definisanje odgovarajućeg okruženja za primenu modelom vođenog pristupa razvoju skladišta podataka kako bi se savladale prikazane neusaglašenosti i problemi razvoja skladišta podataka. Rad je organizovan na način kako je prikazano u daljem tekstu.

Nakon uvodnih napomena, u drugom poglavlju su predstavljene vrste arhitektura koje se koriste u razvoju skladišta podataka. U prvom odeljku su predstavljene osnovne komponente logičke arhitekture od kojih skladište podataka može, ali ne mora da se sastoji. Drugi odeljak razmatra „modelom vođenu arhitekturu“ koja predstavlja pogodan okvir za transformacioni razvoj i interoperabilnost različitih jezika za opis sistema. Ova arhitektura predstavlja i „meta“ opseg u okviru kojeg će biti predstavljeni svi modeli u kasnijim poglavljima. Treći odeljak drugog poglavlja je posvećen fizičkoj arhitekturi skladišta podataka, odnosno načinu i mogućnostima struktuiranja i organizovanja logičkih komponenti skladišta podataka.

U narednom, trećem poglavlju, razmatraju se postojeći konceptualni modeli koji se koriste za realizaciju skladišta podataka. U prvom delu poglavlja je dat kratak prikaz OMG standarda – CWM metamodela i njegovih komponenti. U nastavku poglavlja je dat opis postojećih konceptualnih modela koji se zasnivaju na MOV, UML ili semantičkim modelima. Svrha poglavlja je, osim upoznavanja sa trenutnim stanjem

konceptualnih modela iz oblasti projektovanja skladišta podataka, i ukazivanje na njihove nedostatke u opisu i izgradnji svih slojeva DW.

Kako je već navedeno, neophodno je da modeli podataka koji se koriste za razvoj skladišta podataka mogu da podrže praćenje sistema kroz različita stanja i da na adekvatan način omoguće održavanje promena nastalih u realnom sistemu, odnosno izvorima podataka. Trenutno ne postoji standardni model koji je predviđen za definisanje strukture skladišta podataka. Stoga je u četvrtom poglavlju izvršen pregled postojećih pristupa. Dva najstarija predlažu organizaciju podataka u 3NF ili u višedimenzionom modelu. Oba pristupa imaju ograničenja koja se ogledaju u otežanom održavanju promena strukture izvora. Sa druge strane, oba pristupa su standardizovani kroz odgovarajuće metamodele definisane u CWM metamodelu. Pored toga, prikazana su još dva pristupa koji pokušavaju da nadomeste ove nedostatke. To su 'Anchor modeling' baziran na podacima normalizovanim u 6NF i 'Data Vault' pristup koji takođe može (ali ne mora) da skladišti podatke normalizovane u 6NF. Ni Anchor Modeling ni Data Vault nisu standardna ekstenzija CWM metamodela.

Analiza ovih modela je izvršena sa nekoliko aspekata koji određuju upotrebljivost modela podataka za razvoj skladišta podataka. Pored kratkog opisa modela podataka i njihovih osnovnih koncepata, svaki pojedinačni model je detaljno razmatran uz sagledavanje njihovih dobrih osobina i postojećih nedostataka.

Na osnovu prikazanog u prethodnim poglavljima, u petom poglavlju se definiše C-DV metamodel sa dvojakom namenom. Prvo, C-DV predstavlja okvir za konceptualno modelovanje skladišta podataka proširenjem empirijskog Data Vault modela tako što ga formalizuje i proširuje konceptima koji omogućavaju praćenje strukturnih promena na modelima izvora podataka. Drugo, C-DV se definiše na taj način da može da izrazi i obuhvati semantiku složenih modela izvora podataka i da bude međumodel automatskih transformacija iz izvornih u ciljne modele skladišta podataka.

Pored navedenog, u petom poglavlju je izvršeno pozicioniranje C-DV metamodela u okviru modelom vođene arhitekture i definisani su C-DV koncepti kao ekstenzija

CWM metamodela. Takođe, data su pravila transformacije izvornih modela u ciljni C-DV.

Povezivanje C-DV modela (koji predstavljaju pojedinačne verzije modela izvora), sa integrisanim skladištem podataka, vrši se preko L-DV modela koji je prikazan u šestom poglavlju. L-DV metamodel je definisan kako bi se opisali platformski nezavisni modeli skladišta podataka. Konkretan L-DV model predstavlja integrisan model korporativnog skladišta podataka, koje sadrži strukturu svih modela izvora sa svim promenama tih modela.

U drugom delu šestog poglavlja je izvršeno pozicioniranje L-DV metamodela u okviru modelom vođene arhitekture i prikazani su L-DV koncepti kao ekstenzija CWM metamodela. Takođe, data su pravila transformacije izvornih C-DV modela u ciljni L-DV model.

Da bi se integrisani L-DV model prilagodio konkretnom okruženju, vrši se njegova transformacija u odgovarajući P-DV model. U sedmom poglavlju je definisan P-DV metamodel čija je osnovna namena da opiše detalje ciljne tehnološke platforme i da omogući automatsku transformaciju iz integrisanog L-DV modela. P-DV metamodel je definisan na taj način da obuhvata dobre osobine prikazanih modela podataka i da ispunjava sve postavljene zahteve kako je dato u četvrtom poglavlju. Slično kao i L-DV model, i P-DV se izgrađuje isključivo nadogradnjom, a ne brisanjem ili izmenom postojećih koncepata neke konkretne tehnološke platforme.

U poglavlju su detaljno razmatrani problemi sa kojima se suočavaju modeli podataka pri definisanju strukture skladišta podataka, kao i odgovarajući zaključci proizašli iz tih primera. Nakon toga je predstavljen postupak definisanja P-DV metamodela koji uzima u obzir sve postavljene zaključke.

U drugom delu sedmog poglavlja je P-DV pozicioniran u okviru modelom vođene arhitekture i data su pravila preslikavanja iz izvornog L-DV modela u ciljni P-DV model.

Poslednje, osmo poglavlje razmatra metodološki postupak razvoja skladišta podataka. U okviru poglavlja se daje predlog skupa postupaka koji upravljaju

razvojem skladišta podataka korišćenjem prikazanih DV metamodela u okvirima *modelom vođene arhitekture*. Takođe, definisan je predlog ciljne arhitekture skladišta podataka.

U zaključku se analiziraju rezultati rada u odnosu na postavljene ciljeve. Posebno se diskutuju mogući pravci daljeg istraživanja na ovakvoj primeni novih metoda i modela u jednom aktivnom okruženju kakvo danas predstavljaju sistemi za razvoj skladišta podataka.

Nakon zaključka, dat je spisak korišćene literature i odgovarajući prateći prilozi.

2 SAVREMENI PRISTUPI RAZVOJU SKLADIŠTA PODATAKA

U ovom poglavlju će se izvršiti analiza skladišta podataka sa stanovišta logičke i fizičke arhitekture, sa ciljem korišćenja ovih zapažanja kao svojevrsnog ustanovljenog „rečnika“ koji će biti korišćen za dalja razmatranja u okviru rada.

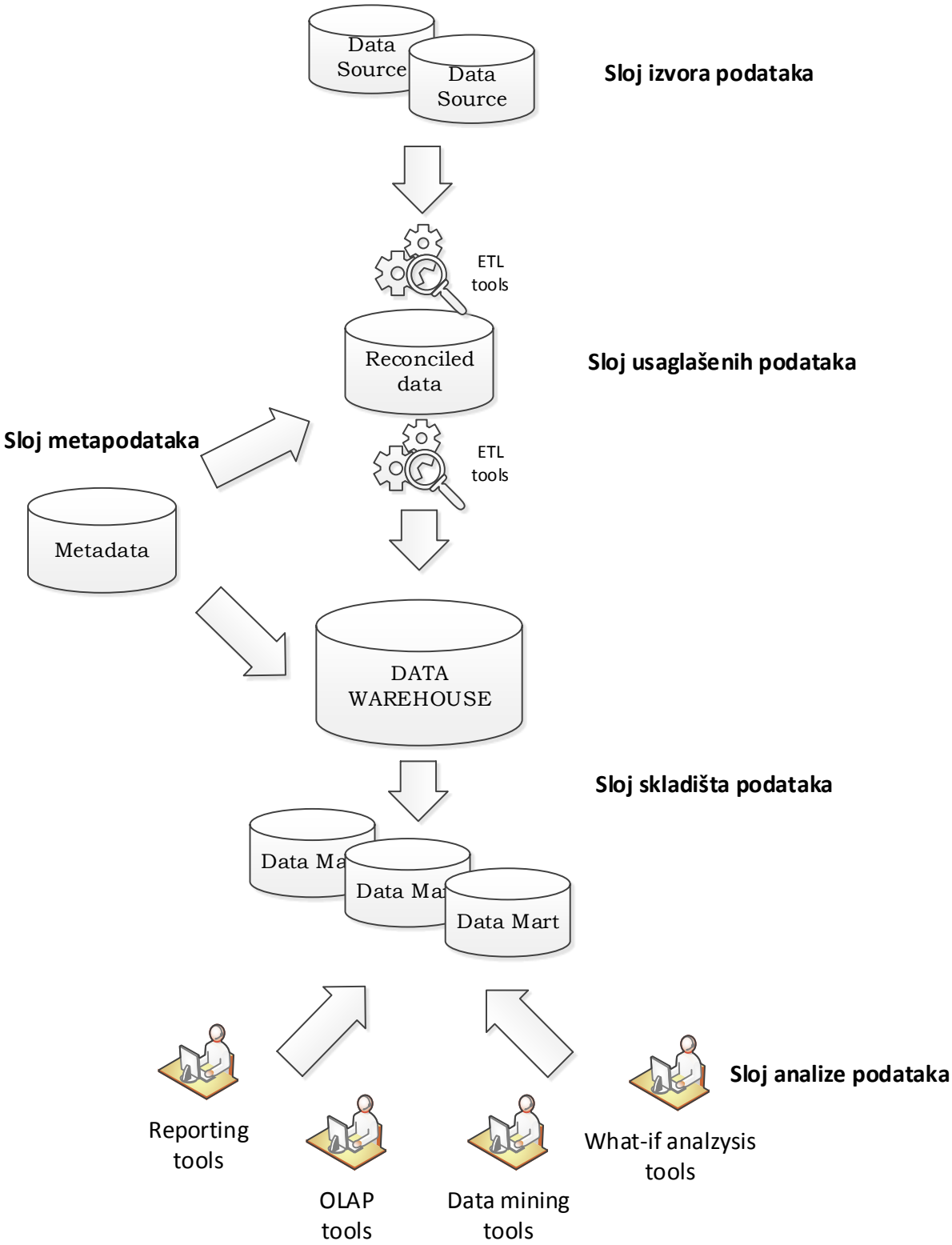
2.1 Komponente logičke arhitekture skladišta podataka

U kojoj arhitekturi će DW sistem biti realizovan, uslovljeno je sa više ulaznih kriterijuma kao što su: nefunkcionalni zahtevi, odabrani proces razvoja skladišta podataka, raspoloživi resursi, dostupno vreme realizacije i slično. Osnovne (najčešće korišćene) logičke komponente DW arhitekture su:

- Sloj izvora podataka
- Sloj pripreme podataka
- Sloj usaglašenih podataka
- Sloj skladišta podataka
- Sloj analize podataka
- Sloj metapodataka

uz ogradu da nije neophodno korišćenje svih komponenti (što potvrđuju primeri realizovanih skladišta podataka) u razvoju DW sistema, što će biti predstavljeno u odeljku *Fizička arhitektura skladišta podataka* ovog poglavlja.

Navedene logičke komponente, su prikazane na Slici 1, koja je u celini preuzeta iz [9].



Slika 1. Komponente logičke arhitekture DW

2.1.1 Sloj izvora podataka

Skladište podataka je zasnovano na podacima koje preuzima iz različitih sistema (eng. *Data Source Layer*) koji proizvode te podatke pri obavljanju poslovnih funkcija organizacije. Izvorni sistemi mogu biti heterogeni, tj. realizovani na različitim tehnološkim platformama. Takođe, heterogenost izvornih sistema se ogleda i u različitim modelima i formatima u kojima skladište podatke, stepenu njihove struktuiranosti, semantici i slično.

Postoji nekoliko kategorija podataka koji se prikupljaju iz izvornih sistema:

- *produkcioni podaci* su podaci nastali pri svakodnevnom obavljanju poslovnih funkcija u organizaciji na unapred definisan i usaglašen način.
- *interni podaci* su svi podaci koje pojedinci u organizaciji kreiraju kao ulazne ili pomoćne podatke u obavljanju poslovnih funkcija.
- *arhivski podaci* su podaci koji su već skladišteni na različitim medijumima u okviru izvornih sistema i posebno su značajni pri uspostavljanju skladišta podataka, tj. inicijalnom importu.
- *eksterni podaci* su podaci koji nastaju izvan granica organizacije, ali koji će se zbog svog značaja pohranjivati u skladištu podataka.

2.1.2 Sloj pripreme podataka

Pošto se podaci preuzimaju iz više različitih izvora, moguće je izvršiti niz akcija nad podacima na sloju pripreme podataka (eng. *Data Staging Area*) čime se vrši objedinjavanje i usklađivanje podataka pred njihovo učitavanje u skladište podataka. Akcije koje se preduzimaju su poznate kao proces preuzimanja, transformacije i učitavanja podataka. (eng. *Extract, Transform & Load - ETL*). Ovaj proces može da obuhvati preuzimanje, prečišćavanje, integraciju heterogenih modela, određene konverzije podataka, kao i transformaciju podataka u format koji je pogodan za kasnije punjenje skladišta podataka. ETL proces se izvršava na dva načina: kao inicijalni import pri početnom punjenju skladišta podataka i kasnije u predefinisanim intervalima kada se skladište podataka dopunjuje razlikom u podacima.

Preuzimanje podataka. Deo procesa koji treba da se prilagodi podacima koji se nalaze u struktuiranom, polustruktuiranom ili nestruktuiranom formatu i koji nastaju u izvorima podataka realizovanim na raznorodnim tehnološkim platformama. Podaci pri preuzimanju mogu biti čuvani na izvorima podataka ili češće, na fizički odvojenim mestima predviđenim za tu namenu kako bi se izvori podataka u najkraćem roku rasteretili za obavljanje uobičajenih aktivnosti.

Transformacija podataka. Izuzetno zahtevna operacija koja se sastoji iz niza nezavisnih koraka u cilju pripreme podataka, tj. konvertovanja podataka iz formata u kojem se čuvaju u izvorima podataka u format pogodan za učitavanje u skladište podataka. Pored filtriranja, korekcije, šifriranja, sortiranja i sličnih operacija, transformacija uključuje i standardizaciju podataka, tj. usaglašavanje podataka iz više različitih izvora, kao i semantičku standardizaciju, kao što je razrešavanje sinonima i homonima u podacima. Složenost ovog dela ETL procesa se ogleda i u tome što ova operacija treba da podrži ne samo inicijalni import podataka, već i sve strukturne i druge promene na izvorima podataka, prilikom kasnijih iterativnih učitavanja.

Učitavanje podataka. Učitavanje je poslednja faza ETL procesa, kada se pripremljeni podaci prenose u skladište podataka. Zavisno od zahteva skladišta podataka, moguće je vršiti učitavanje celokupnog DW, uz prethodno brisanje postojećih podataka, ili, što je češće, vrši se učitavanje razlike podataka od prethodnog učitavanja. U ovoj fazi se vrši konačna provera strukture i ispravnosti podataka kao što su referencijalni integritet i druga ograničenja ciljnog skladišta podataka.

Pored uobičajenog ETL pristupa prikupljanju podataka, postoji i tzv. ELT [3] pristup (eng. Extract, Load & Transform) koji podrazumeva da se transformacije nad podacima vrše tek nakon preuzimanja i učitavanja. Na ovaj način se kontrola promene i prilagođavanja podataka prenosi na kasnije faze u izgradnji skladišta podataka, tj. neposredno pre definisanja Sloja analize podataka. Ovaj pristup takođe podrazumeva da skladište podataka sadrži sve podatke izvora, a da se shodno potrebi krajnjih korisnika u analitičkom izveštavanju vrši odgovarajuće filtriranje i

transformacija podataka. U ovom radu će se zastupati ELT pristup preuzimanja podataka iz izvora.

2.1.3 Sloj usaglašenih podataka

Usaglašeni podaci predstavljaju objedinjene podatke iz transakcionih sistema (izvora podataka) koji se konsoliduju nakon integracije i čišćenja izvornih podataka. Sloj usaglašenih podataka (eng. *reconciled data layer* ili *operational data store*) donekle menja strukturu izvornog modela, jer ima za cilj definisanje jedinstvenog modela za sve izvorne modele. Na taj način *Sloj usaglašenih podataka* obezbeđuje zajedničke referentne podatke za skladište podataka. Pored toga, vrši razdvajanje operacija preuzimanja i učitavanja podataka.

2.1.4 Sloj skladišta podataka

Podaci se pohranjuju na Sloju skladišta podataka koje predstavlja logički centralizovani i jedinstveni repozitorijum podataka [9]. Na ovom sloju egzistiraju svi podaci koji postoje u skladištu podataka.

Shodno ciljnoj arhitekturi skladišta podataka i odabranom pristupu razvoja, ovaj sloj može biti organizovan na nekoliko načina:

- kao integrisan model svih izvora sa minimumom redundanse podataka sa najvećim nedostatkom da za ovakav pristup treba najviše vremena i resursa
- kao skup centara podataka koji predstavljaju najčešće skup podataka o pojedinačnim poslovnim funkcijama realnog sistema pri čemu se tada ovaj sloj može razvijati iterativno
- kao kombinacija prethodna dva pristupa, kada je sloj podeljen na dva dela, gde integrisani model predstavlja izvor za izgradnju centara podataka. U ovom slučaju, integrisani model se tada uobičajeno naziva *primarno* ili *korporativno* skladište podataka, dok centri podataka predstavljaju *zavisno* skladište podataka [9].

U sva tri slučaja, moguće je direktno pristupati podacima sa ovog sloja, a takođe je moguće da ovaj sloj služi kao osnova za definisanje *Sloja analize podataka*.

2.1.5 Sloj analize podataka

Na Sloju analize podataka egzistiraju alati koji olakšavaju pristup i omogućavaju analizu skladištenih podataka krajnjim korisnicima skladišta podataka. Alati za analizu podataka mogu biti jednostavni, kao što su alati za pristup i pregled već pripremljenih podataka ili postavljanje ad hoc upita, ali i vrlo složeni kao što su alati za analizu i pronalaženje međuzavisnosti nad podacima (eng. *data mining*) ili alati za simulacije (eng. *what-if analysis*) [12].

2.1.6 Sloj metapodataka

Metapodaci (eng. *meta data*) su podaci koji se koriste da opišu druge podatke. Metapodaci se održavaju u Rečniku skladišta podataka (eng. *Meta Data Repository*) koji je informacioni sistem koji specifikuje celokupno skladište podataka, i ujedno je osnova za razvoj ostalih slojeva skladišta podataka. Prema [9], pored toga što opisuje sve slojeve skladišta podataka, Rečnik skladišta podataka je dostupan ostalim slojevima pružajući im informacije neophodne za njihov operativni rad. U nekim radovima [82], je posebno naglašena uloga Rečnika skladišta podataka pri učitavanju podataka iz izvora podataka i u vođenju ETL procesa.

Rečnik skladišta podataka sadrži [59]:

- informacije o lokaciji, strukturi i sadržaju skladišta podataka
- informacije o procesima i transformacijama koje se izvršavaju prilikom preuzimanja ili prosleđivanja podataka između različitih slojeva skladišta podataka
- informacije o semantici podataka
- informacije o infrastrukturi i fizičkim karakteristikama izvora i komponenti skladišta podataka
- informacije o bezbednosti, pravima pristupa i korišćenju skladišta podataka

Kompleksnost skladišta podataka i tehnološka različitost proizvođača komponenti skladišta podataka su uslovile potrebu za standardizacijom sloja metapodataka. Ova inicijativa je imala za cilj da omogući platformski nezavisni mehanizam razmene

metapodataka u standardnom formatu između različitih slojeva skladišta podataka bez obzira na tehnološku platformu na kojoj su komponente realizovane.

Prvi pokušaj standardizacije je specifikacija *MetaData Interchange Specification (MDIS)* [81] koja je predložena od strane industrijskog, neprofitnog *MetaData Coalition (MDC)* konzorcijuma. Ovaj platformski nezavisan standard je omogućavao razmenu metapodataka koji su se odnosili pre svega na strukturu izvora, centara podataka i na veze između osnovnih koncepata skladišta podataka [44]. Zbog svoje ograničene funkcionalne primene, i zatvorenosti za neke već postojeće standarde za razmenu poruka, ova specifikacija svojevremeno nije postala *de facto* standard.

Istovremeno, od strane *Microsoft* korporacije, razvijan je i *Open Information Model (OIM)* koji je bio zasnovan na UML i XML jezicima kao standardnim formatima za razmenu metapodataka. U narednom periodu, OIM inicijativa je priključena MDC koaliciji kako bi se obezbedila otvorenost ovog standarda. OIM je prevazilazio okvire skladišta podataka, jer je bilo predviđeno da obuhvati i analizu, projektovanje i ostale faze procesa razvoja informacionih sistema [44].

Nakon ova dva pokušaja usledila je OMG inicijativa objavljivanjem *Common Warehouse Metamodel* (CWM) [23] [46] standarda zasnovanog na već postojećim OMG standardima: UML, MOF i XMI. Osnovna namena CWM je da omogući razmenu metapodataka, u domenu skladišta podataka i poslovne inteligencije, između različitih alata, platformi i repozitorijuma u distribuiranim heterogenim okruženjima [23]. Nakon jednog perioda egzistiranja dva otvorena standarda, CWM i OIM, MDC koalicija je odlučila da napusti dalji rad na OIM standardu i da zajedno sa OMG grupacijom učestvuje u daljem razvoju CWM standarda čime je CWM postao jedinstveni standard za razvoj i razmenu metapodataka u domenu skladišta podataka. Ovaj standard će detaljnije biti opisan u odeljku *Opšti metamodel skladišta podataka – CWM* poglavlja *Konceptualni modeli skladišta podataka*.

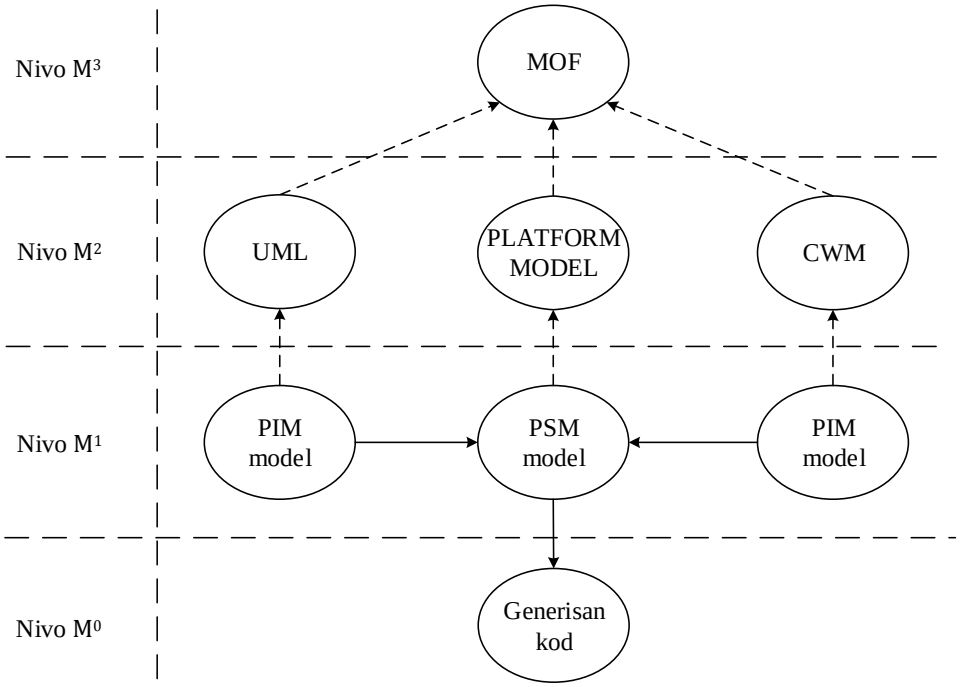
2.2 Modelom vođena arhitektura – MDA

Modelom vođena arhitektura [21] (eng. *Model Driven Architecture – MDA*) je OMG standard ustanovljen 2001. godine. OMG (eng. *Object Management Group*) je predstavila MDA kao arhitekturu u kojoj su modeli osnovni gradivni blokovi i kao okvir koji omogućava interoperabilnost različitih jezika za opis sistema.

2.2.1 Osnovni koncepti MDA arhitekture

MDA arhitektura se sastoji iz četiri hijerarhijska nivoa, pri čemu model na višem nivou predstavlja metamodel koji opisuje modele na nižem nivou.

Na najvišem nivou hijerarhije – M^3 nivou, kao što je prikazano na Slici 2, nalazi se MOF [24] (eng. *Meta Object Facility*) kao apstraktni jezik za specifikaciju jezika za opis, odnosno meta modela. Pošto je MOF meta meta model, i definiše kako će se neki meta model opisivati, često se naziva ontologijom.



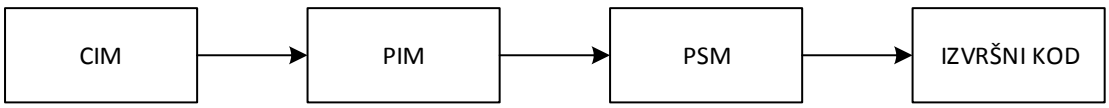
Slika 2. Modelom vođena arhitektura - MDA

Na tom nivou su definisani koncepti, sintaksa i opis strukture opštih metamodela (npr. UML) ili specifičnih (npr. CWM). Tako definisani metamodeli su pozicionirani na M^2 nivou. Ovi jezici se koriste za formalnu specifikaciju sistema. Rezultat specifikacije tj. modeli koji su njima opisani su tehnološki nezavisni i oni se nalaze na M^1 nivou. Takvi modeli se kasnije prevode u tehnološki zavisne modele generisanjem u konkretno implementaciono okruženje [99].

2.2.2 Vrste modela MDA arhitekture

Modeli koji su osnovni elementi MDA arhitekture se mogu razvrstati na osnovu aspekta koji je korišćen za njihovo kreiranje, odnosno da li su korišćene apstrakcije bliže poslovnom sistemu ili tehnološkoj platformi. Na osnovu ovog kriterijuma razlikuje se tri vrste modela, kao što je prikazano na Slici 3:

- Domenski modeli (eng. *Computation Independent Models – CIM*) su modeli koji prvenstveno služe kao tačka sporazumevanja između eksperata poslovnog sistema i analitičara. CIM modeli opisuju poslovni sistem bez bilo kakvih detalja vezanih za informacione sisteme, odnosno ne prejudiciraju način realizacije u nekom informacionom okruženju.
- Platformski nezavisni modeli (eng. *Platform Indipendent Models – PIM*) su modeli koje uobičajeno definišu analitičari sistema, i oni predstavljaju specifikaciju sistema bez detalja o konkretnoj tehnološkoj platformi.
- Platformski zavisni modeli (eng. *Platform Specific Models – PSM*) predstavljaju specifikaciju sistema na ciljnoj tehnološkoj platformi. PSM modeli su prilagođeni za automatsku transformaciju u izvršni kod.



Slika 3. MDA modeli

Sve prikazane vrste modela pripadaju M^1 nivou MDA arhitekture.

Korišćenjem CIM, PIM i PSM modela u specifikaciji sistema, postiže se odvajanje specifikacije funkcionalnosti sistema od specifikacije implementacije te funkcionalnosti na konkretnoj tehnološkoj platformi [21].

Pored toga, podržana je interoperabilnost jer je moguće da se na osnovu jedne specifikacije, definisane PIM modelom izvrši transformacija u više PSM modela koji odgovaraju različitim tehnološkim platformama.

Pristup zasnovan na MDA okviru je zbog svojih dobrih karakteristika pogodan i za razvoj skladišta podataka, jer se za specifikaciju komponenti logičke arhitekture, kao što je opisano u prethodnom odeljku, koristi više različitih jezika za opis.

2.3 Fizička arhitektura skladišta podataka

Projektovanje i realizacija arhitekture skladišta podataka, od prethodno definisanih logičkih komponenti, treba da bude zasnovano i vođeno određenim nefunkcionalnim zahtevima kako bi ciljno skladište podataka odgovorilo na brojne zahteve pri korišćenju i održavanju takvog sistema. Nefunkcionalni zahtevi koji definišu arhitekturu skladišta podataka su [9][2][11]:

- Dostupnost informacijama (eng. *Accessibility*)
- Konzistentnost podataka (eng. *Data Consistency*)
- Prilagodljivost (eng. *Adaptivity*)
- Proširivost (eng. *Extensibility*)
- Modularnost (eng. *Separation*)
- Skalabilnost (eng. *Scalability*)
- Sigurnost podataka (eng. *Security*)
- Potpunost podataka (eng. *Single Version of Fact*)
- Pratljivost (eng. Traceability)

U odnosu na ispunjenost nefunkcionalnih zahteva i način mapiranja komponenti logičke arhitekture skladišta podataka, fizička realizacija DW se može razmatrati sa dva stanovišta: strukturnog i organizacionog. Strukturno orijentisana realizacija se odnosi na broj slojeva arhitekture na koji se logičke komponente DW preslikavaju, dok organizacioni aspekt sagledava topologiju skladišta podataka sa stanovišta primene i načina korišćenja DW u okviru organizacije ili njenih delova.

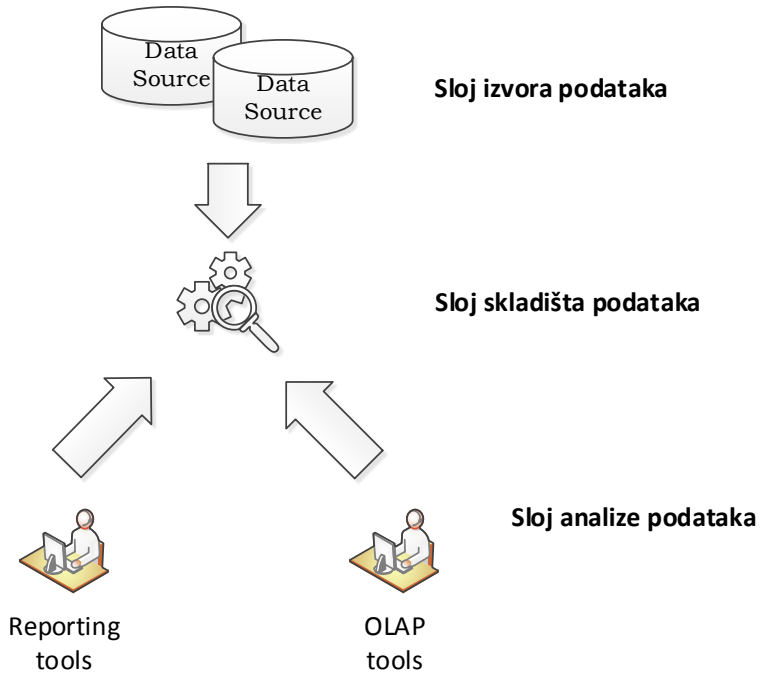
2.3.1 Strukturni aspekt arhitekture skladišta podataka

Strukturni aspekt arhitekture skladišta podataka razmatra broj slojeva sa kojim je realizovano skladište podataka, kao i preslikavanje logičkih komponenti na pojedine slojeve arhitekture.

2.3.1.1 Jednoslojna arhitektura

Kao što je prikazano na Slici 4, jedini sloj ove arhitekture je sloj izvora podataka. Ova arhitektura u kojoj je skladište podataka virtuelno, nije često korišćena u praksi [9],

a nastaje prvenstveno u organizacijama koje uoče potrebu za analitičkim izveštavanjem, a pre ulaganja u skladište podataka. Takođe, koristi se ukoliko ne postoje uslovi da se podaci redundantno čuvaju, pri čemu je skladište podataka realizovano kao multidimenzioni pogled nad transakcionim podacima [77].



Slika 4. Jednoslojna arhitektura

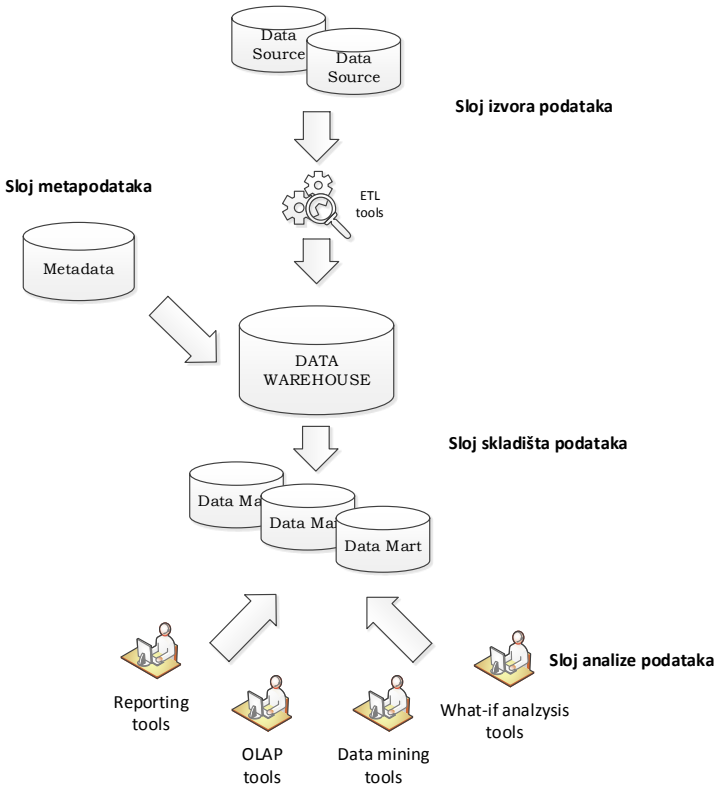
Slabost ovakve arhitekture je što nije ispunjen jedan od osnovnih nefunkcionalnih zahteva, razdvojenost transakcionog i analitičkog dela skladišta podataka, što dovodi do značajnog opterećenja transakcionog sistema prilikom analitičkih upita. Takođe, virtuelno skladište podataka nije sposobno da čuva više podataka nego što ih ima u samim transakcionim sistemima. Stoga je ovakva arhitektura pogodna jedino u slučajevima kada su za analizu podataka dovoljne ograničene informacije [9].

2.3.1.2 Dvoslojna arhitektura

U dvoslojnoj arhitekturi se pored izvora podataka uvodi i sloj skladišta podataka, čime se postiže nezavisnost transakcionog sistema od analitičkog dela. Podaci su smešteni u jedinstvenom centralizovanom repozitorijumu koje predstavlja izvor informacija analitičkom sistemu. Sa druge strane, centralizovano skladište podataka

može da bude izvor za logički organizovane centre podataka (eng. *Data Mart*) čiji je prvenstveni cilj da pružaju informacije prema poslovnim domenima, organizacionim jedinicama, kategoriji korisnika ili sličnim kriterijumima. Ukoliko je to slučaj, centralizovano skladište podataka se naziva *primarno* skladište podataka [9], dok skup kreiranih centara podataka predstavlja *zavisno* skladište podataka.

Centri podataka se mogu posmatrati kao lokalizovana skladišta podataka koja skladište deo dostupnih podataka. Jedna od osnovnih karakteristika centara podataka su dobre performanse, tj. analitički upiti ne traju dugo kao pri dobijanju informacija iz primarnog skladišta podataka, jer su podaci već pripremljeni (npr. agregirani) i prilagođeni specifičnim potrebama krajnjih korisnika.



Slika 5. Dvoslojna arhitektura

U praksi su poznati slučajevi da skladište podataka bude organizovano isključivo od centara podataka [2], što može voditi do neispunjavanja nefunkcionalnog zahteva o konzistentnosti podataka, usled nepostojanja objedinjenog modela iz kojih se podaci učitavaju. Dobra strana ovakvog pristupa je što se razvoj skladišta podataka može

vršiti postupno pri čemu se određeni rezultat javlja ranije nego što je to slučaj pri realizaciji objedinjenog tj. centralizovanog skladišta podataka.

Pored ova dva sloja, kao što je prikazanao na Slici 5, dvoslojna arhitektura podrazumeva i sloj pripreme podataka i sloj analize podataka.

2.3.1.3 Troslojna arhitektura

Troslojna arhitektura uvodi kao dodatni sloj u odnosu na dvoslojnu arhitekturu, sloj usaglašenih podataka. Kao što je prikazano na Slici 1, sloj usaglašenih podataka se nalazi između sloja izvora podataka i sloja skladišta podataka. Sloj usaglašenih podataka predstavlja *referentni* model podataka sveukupnih stanja celokupne organizacije. Pored toga, značaj ovog sloja se ogleda u tome što umanjuje probleme koji se javljaju pri promeni strukture izvora ili preuzimanju i integraciji tih podataka. Na taj način je zadovoljen nefunkcionalni zahtev dostupnosti podacima jer promene na izvorima podataka ne utiču na dostupnost sloja skladišta podataka.

Dobre osobine troslojne arhitekture su:

- Postoji stalna dostupnost informacijama, jer nemogućnost pristupa nekom izvoru podataka ne utiče na pristup centralizovanom skladištu podataka
- Analitički upiti ne utiču negativno na transakcione izvore podataka zbog nezavisnosti pojedinih slojeva arhitekture
- Sloj usaglašenih podataka i sloj skladišta podataka mogu biti realizovani preko različitih modela podataka, kao što su multidimenzioni model, model objekti veze i sl.

2.3.2 Organizacioni aspekt arhitekture skladišta podataka

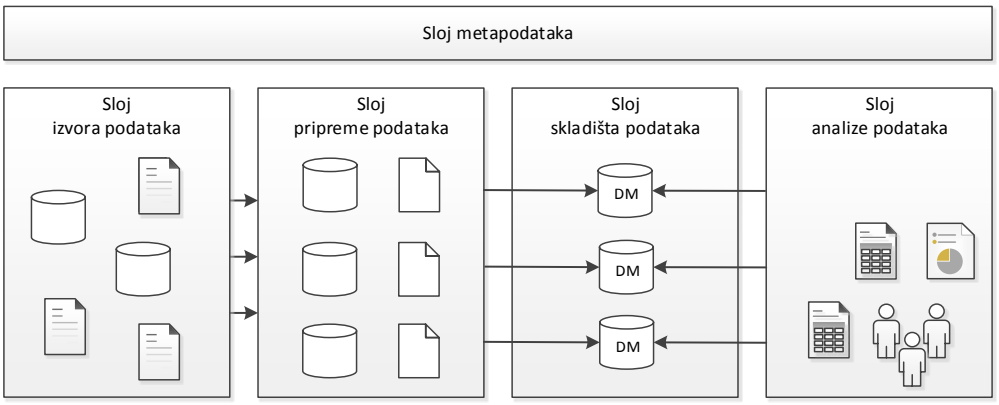
Prema [78], u naučnoj literaturi definisano je nekoliko različitih topologija skladišta podataka, koje se razlikuju prema načinu organizacije komponenti zasnovanih na već prikazanim osnovnim slojevima skladišta podataka:

- arhitektura nezavisnih centara podataka,
- arhitektura usklađenih centara podataka,
- arhitektura zavisnih centara podataka,
- centralizovana arhitektura i
- federativna arhitektura.

Navedene arhitekture se nazivaju i referentne, jer predstavljaju polaznu osnovu za razvoj skladišta podataka [79].

2.3.2.1 Arhitektura nezavisnih centara podataka

Arhitektura nezavisnih centara podataka (eng. *independent data mart architecture*) podrazumeva da su centri podataka u sloju skladišta podataka realizovani nezavisno, pri čemu je ovakvo skladište podataka neintegrisano i može dovesti do nekonzistentnih podataka unutar skladišta. Problemi koji mogu nastati korišćenjem ovakve arhitekture utiču da se arhitektura nezavisnih centara podataka zamenjuje drugim arhitekturama kako bi se ispunio nefunkcionalni zahtev za konzistentnošću podataka.

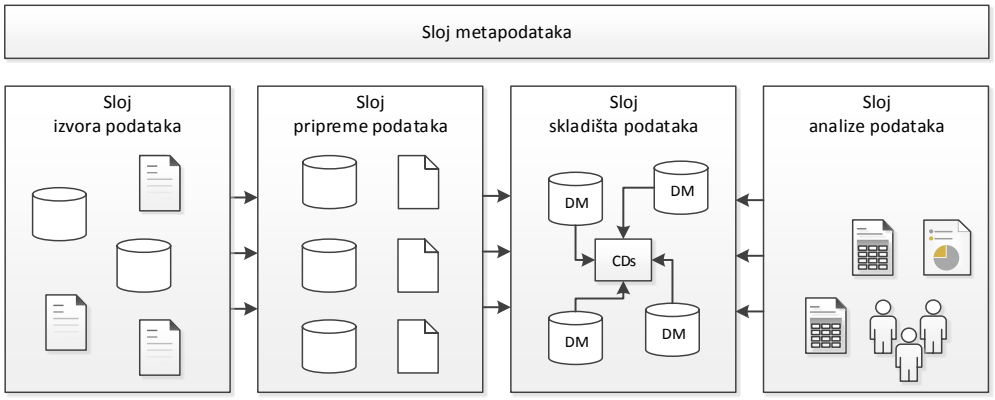


Slika 7. Arhitektura nezavisnih centara podataka

2.3.2.2 Arhitektura usklađenih centara podataka

Arhitektura usklađenih centara podataka (eng. *bus architecture*) je vrlo slična prethodnoj arhitekturi, ali sa jednom bitnom razlikom: dimenzije centara podataka su u arhitekturi usklađenih centara podataka usaglašene (eng. *conformed dimensions*). To praktično znači da svi centri podataka koriste iste dimenzije za

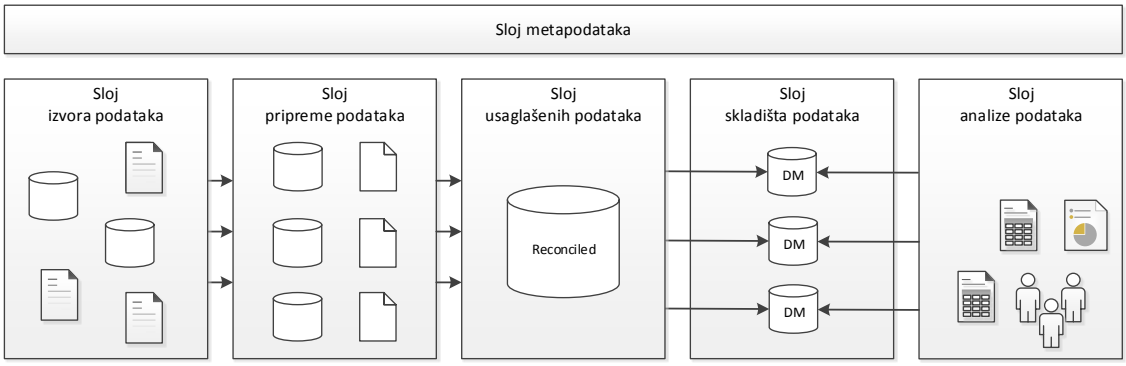
celokupnu organizaciju [2]. Na taj način se korišćenjem ove arhitekture izbegava nekonzinstentnost podataka u okviru skladišta podataka.



Slika 8. Arhitektura usklađenih centara podataka

2.3.2.3 Arhitektura zavisnih centara podataka

Arhitektura zavisnih centara podataka (eng. *hub and spoke architecture*) omogućava skalabilnost i proširivost skladišta podataka, kao i centralizovani pogled na informacije organizacije [9]. U ovoj arhitekturi centri podataka, uobičajeno u multidimenzionom modelu, preuzimaju i agregiraju podatke od sloja usaglašenih podataka (eng. *reconciled layer*) koji skladišti normalizovane podatke.

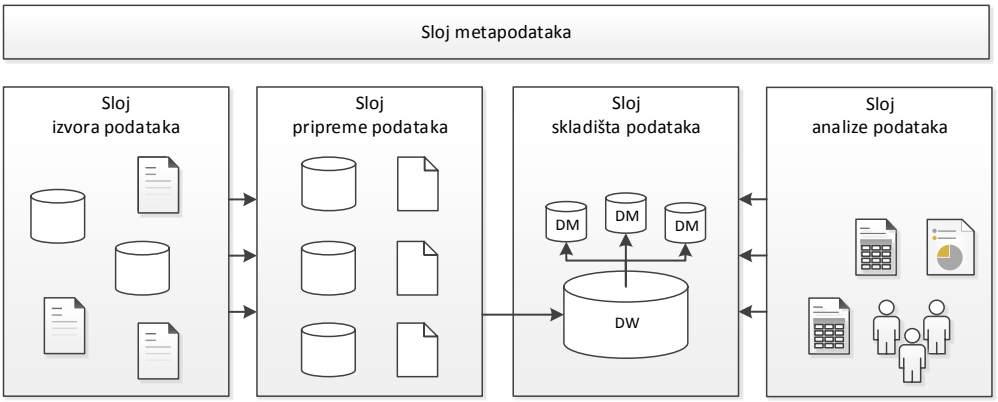


Slika 9. Arhitektura zavisnih centara podataka

2.3.2.4 Centralizovana arhitektura

Centralizovana arhitektura (eng. *centralized architecture*) za razliku od prethodno opisane arhitekture zavisnih centara podataka podrazumeva sloj skladišta podataka

kao centralizovanog modela koji suštinski obuhvata sloj usaglašenih podataka i centre podataka [11] [78]. Ova arhitektura ne isključuje realizaciju centara podataka na osnovu primarnog skladišta podataka. Razvoj skladišta podataka korišćenjem ove arhitekture je sporije u odnosu na prethodno opisane arhitekture, jer pre bilo kakvog punjenja sloja analize podataka, neophodno je projektovati sloj skladišta podataka za celokupnu organizaciju.



Slika 10. Centralizovana arhitektura

2.3.2.5 Federativna arhitektura

Federativna arhitektura (eng. *federated architecture*) se koristi u slučajevima potrebe za integracijom više različitih (u tehnološkom, organizacionom, funkcionalnom smislu i sl.), ali već postojećih, skladišta podataka. Integracija može biti na logičkom ili fizičkom nivou, a postiže se korišćenjem sloja metapodataka, distribuiranim upitima i sličnim metodama [78].

Sve opisane arhitekture su potvrđene u praksi, a njihov izbor prilikom projektovanja konkretnog skladišta podataka zavisi od sledećih faktora [9]:

- Količina međuzavisnih podataka koji se razmenjuju između organizacionih jedinica neke organizacije utiče na izbor arhitekture. Ukoliko postoji potreba snažnije povezanosti organizacionih delova na nivou cele organizacije, izabrana arhitektura može biti centralizovana arhitektura ili arhitektura usklađenih i zavisnih centara podataka, dok, ukoliko je slučaj da su

organizacioni delovi u značajnoj meri nezavisni, izabrana arhitektura može biti arhitektura nezavisnih centara podataka.

- Finansijska i resursna ograničenja, kao i kratko vreme za realizaciju skladišta podataka, utiču na izbor arhitektura koje se iterativno i brže razvijaju, kao što je arhitektura nezavisnih centara podataka.
- Potreba da se integrišu već postojeća skladišta podataka, posebno ukoliko su razvijena na različitim tehnološkim platformama, mogu rezultovati u izboru federativne arhitekture.

3 KONCEPTUALNI MODELI SKLADIŠTA PODATAKA

3.1 Opšti metamodel skladišta podataka – CWM

CWM standard je proširenje MDA arhitekture u domenu skladišta podataka sa osnovnom namenom da omogući opis i razmenu metapodataka skladišta podataka nezavisno od tehnološke platforme, alata i repozitorijuma u distribuiranim heterogenim okruženjima [23]. Korišćenjem ostalih OMG standarda, kao što su UML i XMI omogućena je interoperabilnost platformski nezavisnih okruženja razmenom metapodataka preko odgovarajućih XML dokumenata [45].

Opšti metamodel skladišta podataka je skup metamodela iz domena skladišta podataka organizovanih u više logičkih celina – paketa. Ovakva modularnost CWM omogućava korišćenje samo pojedinih, ne obavezno svih, metamodela koji su neophodni za projektovanje skladišta podataka shodno korisničkim zahtevima i ciljnoj arhitekturi.

Kao što je prikazano na Slici 11, metamodeli su razmešteni u više međusobno zavisnih slojeva.

3.1.1 Bazni sloj

Bazni sloj (eng. *Object Layer*) je podskup UML koji preuzima samo one koncepte koji su neophodni za opis CWM [23]. Sastoji se iz *Core*, *Behavioral*, *Relationship* i *Instance* metamodela. *Core* metamodel opisuje osnovne koncepte koji egzistiraju u UML jeziku. *Behavioral* metamodel proširuje osnovne koncepte sa konceptima koji služe za opisivanje ponašanja sistema, kao što su operacije i procedure. *Relationship* metamodel obuhvata moguće veze između koncepata nekog sistema, dok *Instance* metamodel definiše koncepte koji predstavljaju pojavljivanja datih modela.

3.1.2 Opšti sloj

Opšti sloj (eng. *Foundation Layer*) predstavlja skup metamodela koji obezbeđuju zajedničke koncepte na koje se oslanjaju metamodeli u višim slojevima CWM arhitekture. *Data Types* metamodel definiše osnovne skupove - tipove podataka iz kojih metapodaci uzimaju vrednosti. *Type Mapping* opisuje koncepte koji predstavljaju preslikavanje između raznorodnih sistema tipova podataka čime se omogućava interoperabilnost različitih tehnoloških platformi i alata. *Keys Indexes* metamodel definiše identifikatore različitih modela podataka, kao što su relacioni ili dimenzioni model. *Business Information* metamodel sadrži koncepte koji opisuju dodatne karakteristike okruženja u kojem ostali elementi obitavaju [44].

3.1.3 Resursni sloj

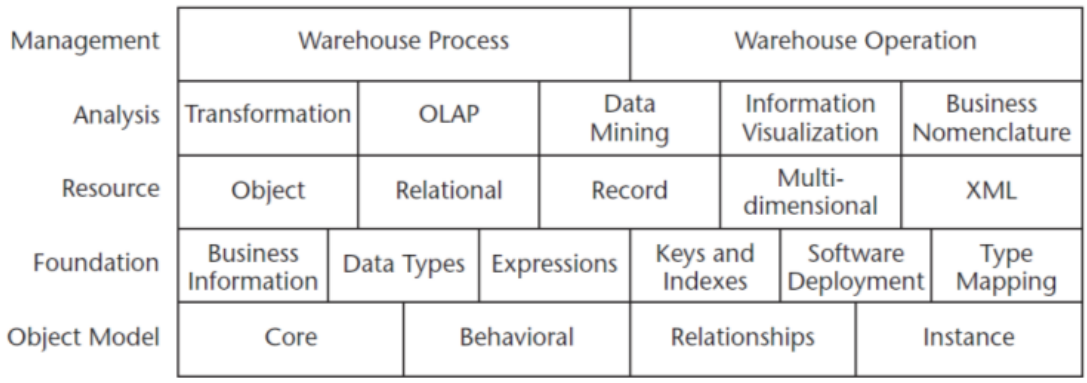
Resursni sloj (eng. *Resource Layer*) obuhvata metamodele koji definišu različite jezike za opis strukture i/ili skladištenje podataka. U okviru ovog sloja nalaze se *Relacioni*, *Rekord*, *Multidimenzioni* i *XML* metamodel. Metamodel *Objekti-Veze* definisan je u proširenju CWM specifikacije (eng. *Extended CWM – CWMX*) [46].

3.1.4 Analitički sloj

Analitički sloj (eng. *Analysis Layer*) predstavlja skup metamodela koji se odnose na analizu podataka skladišta podataka. Osnovni metamodel ovog sloja je *Transformation* metamodel koji, korišćenjem koncepata metamodela sa Resursnog sloja, može da opiše mapiranja i transformacije iz datog izvornog u ciljni model. Takođe, transformacije mogu da se odnose i na ostale modele Analitičkog sloja, kao izvornih ili ciljnih modela. *OLAP* metamodel sadrži koncepte koji izlažu konsolidovane poslovne podatke u višedimenzionom formatu. *Data Mining* je metamodel koji omogućava specifikaciju metapodataka pridruženih različitim izvorima podataka kako bi se izvukli željeni paterni i trendovi iz poslovnih podataka. *Business nomenclature* metamodel omogućava definisanje taksonomija i nomenklatura poslovnih termina i koncepata. *Visualisation* metamodel sadrži koncepte koji mogu da opišu metapodatke koji se odnose na napredno izveštavanje i analitičke alate [44].

3.1.5 Upravljački sloj

Upravljački sloj (eng. *Management Layer*) se sastoji iz dva metamodela: *Warehouse Process* koji omogućava formalni opis poslovnih procesa, kao što je definisanje ETL procesa i *Warehouse Operation* koji može da opiše različite specifične ili periodične aktivnosti nad objektima skladišta podataka.



Slika 11. CWM metamodel

3.2 Model objekti / veze

Model objekti veze – MOV (eng. *Entity Relationship Model - ERM*) [40] je najpopularniji i u praksi najkorišćeniji model podataka koji se koristi za projektovanje transakcionih sistema [83]. Takođe, MOV model je vrlo zastupljen i pri projektovanju skladišta podataka (bilo da se koristi u originalnoj Čenovoj verziji ili kroz jedno od svojih mnogobrojnih proširenja [17] [18] [19]), iako jedan od pionira u ovoj oblasti, Ralf Kimbal, smatra da MOV model ne može da se prilagodi i bude osnovni model strukture za EDW [12].

Model objekti veze je osmislio Piter Čen (eng. *Peter Chen*) 1976. godine i zasnovan je na teoriji skupova i relacionom modelu. Kao što sam autor navodi [40], MOV je nastao sa prvenstvenom namenom da apstrahuje razlike i objedini pogled na tadašnja tri aktuelna modela podataka: mrežni, relacioni i hijerarhijski. MOV je prepoznat kao semantički model [84] koji je prilagođen da prirodnije inkorporira znanje o nekom sistemu, jer se sam sistem sastoji iz skupa objekata i njihovih međusobnih veza. Čen smatra da je MOV okvir iz kojeg mrežni, relacioni i hijerarhijski model mogu da budu izvedeni i stoga zaključuje da je MOV na određen način generalizacija, odnosno ekstenzija pomenutih modela [40].

Originalni Čenov model se sastoji od tri esencijalna koncepta: tipa objekta, tipa veze i atributa. Objekat predstavlja bilo koji realni ili apstraktni koncept koji se modeluje. Tip objekta nastaje korišćenjem apstrakcije *klasifikacije* gde se objekti istih osobina predstavljaju zajedničkim tipom. Tip objekta se prema notaciji predstavlja pravougaonikom. U [40] Čen ne koristi i ne razrađuje apstrakcije generalizacije i agregacije, što je urađeno u kasnijim ekstenzijama osnovnog modela.

Zbog objekata u realnom sistemu koji ne mogu da postoje bez veze sa njemu nadređenim objektom (eng. *Existence dependency*) definisan je *slab* objekat (eng. *Weak entity*). Slab objekat je predstavljen sa dva pravougaonika, jednim unutar drugog.

Veza u MOV predstavlja način povezivanja (uzajamna dejstva) objekata [83]. Svaka veza je određena sa dve role koje predstavljaju funkcije preslikavanja između

skupova kojima entiteti (koji su u vezi) pripadaju. Uvodi se *kardinalnost* preslikavanja koja definiše donju i gornju granicu broja pojavljivanja objekata iz skupa u koji se posmatrani objekat preslikava. Grafički, koncept veze se predstavlja rombom. Takođe, eksplicitno se definiše *slaba* veza, koja povezuje slab i njemu nadređen objekat.

Atributi predstavljaju zajedničke karakteristike objekata istog skupa i dele se u dve klase: *identifikatore* koji jedinstveno određuju pojavljivanje objekta i *deskriptore* koji bliže određuju osobine objekta. Preuzete iz relacionog modela, funkcionalne zavisnosti su podeljene u dve grupe, odnosno funkcionalne zavisnosti koje se odnose na atribute objekta i koje se odnose na objekte u vezi. U Čenovoj originalnoj notaciji ne postoji grafički element za atribut, već atribute prikazuje kao imenovana preslikavanja između tipa objekta i domena, tj. skupa iz kojih atribut uzima vrednost.

MOV je u kasnijem periodu pretrpeo značajna proširenja, razmatranjem n-arnih veza i strukturnih dinamičkih pravila integriteta [86], ali posebno u smislu obogaćivanja semantike modela uvođenjem apstarakcija generalizacije i agregacije [85].

U sledećim odeljcima će biti prikazani najzastupljeniji konceptualni modeli skladišta podataka zasnovani na MOV modelu.

3.2.1 starER

Konceptualni model starER je u svojoj osnovi MOV model koji je proširen sa konceptima dimenzionog modela. Autor (Nectaria Tryfona) smatra da je MOV model potvrdio svoj kvalitete dugogodišnjom praktičnom upotrebom i da je semantički dovoljno bogat da bi se koristio kao konceptualni model u projektovanju skladišta podataka, uz napomenu da je neophodno da bude obogaćen konceptima star šeme kako bi se na najpogodniji način opisalo skladište podataka [19]. Naime, skladišta podataka zbog svoje prirode, pretpostavljaju drugačiju strukturu podataka u odnosu na transakcione sisteme. Središnji koncept u skladištu podataka je fekt - skup događaja koji su rezultat obavljanja poslovnih procesa u realnom sistemu. Sa

skupovima događaja su povezane dimenzije - objekti koji definišu kontekst za svaki pojedinačan događaj ili grupu događaja.

Fekt može dodatno biti opisan sa odgovarajućim karakteristikama – agregatnim atributima. Vrednosti ovih atributa su uglavnom numeričke i stoga se nad njima mogu definisati agregacioni izrazi kako bi se dobile željene informacije. Tipovi agregatnih atributa su:

- Atributi koji opisuju stanje ili vrednost u nekom trenutku vremena (eng. *stock*)
- Atributi koji opisuju kumulativne vrednosti u vremenskom periodu (eng. *flow*)
- Atributi koji opisuju vrednosti u odnosu na neku jedinicu mere (eng. *value-per-unit*)

Na osnovu ovih razmatranja, predviđeno je da starER model bude proširen konceptom događaja. Korišćenjem teorije skupa, definiše se skup dogadjaja (eng. *Fact Set*) koji predstavlja događaje sa istim karakteristikama koji su se izvršili u realnom sistemu. Ovi događaji su uvek povezani sa vremenskom osom i u ovom modelu su događaji predstavljeni kružnicom. Ovaj koncept može da bude proširen i dodatnim karakteristikama – agregatnim atributima koji, za razliku od atributa na tipu objekta u MOV, moraju biti definisani tipom agregatnog atributa. Takođe, za razliku od drugih dimenzionih modela, starER omogućava M:N vezu između fekta i dimenzije.

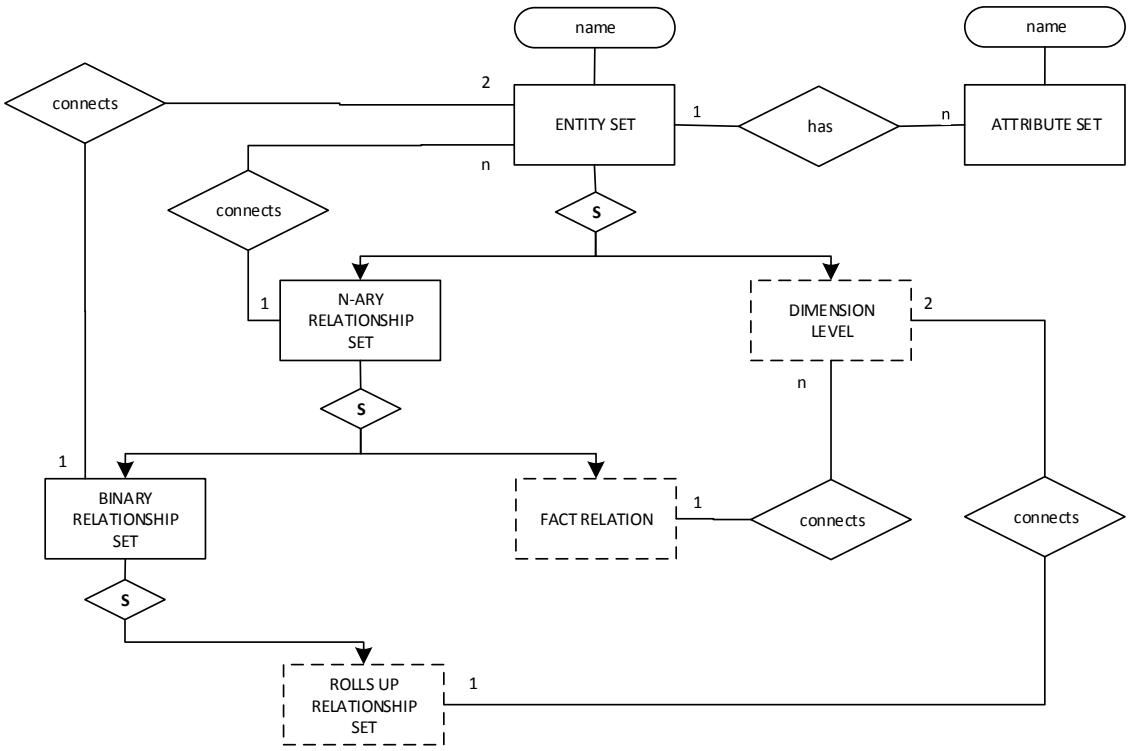
Kao još jedna razlika sa Čenovim MOV modelom, proširena je semantika tipa objekta koji takođe može biti obogaćen agregatnim atributima. U starER modelu, tip objekta ima ulogu dimenzije koji definišu kontekst fekt koncepta.

Potrebno je napomenuti još dve karakteristike starER modela. Prva, da je omogućeno postojanje objekata koji nisu dimenzije i koji nisu direktno vezani za fekt. Drugo, dozvoljene su specifične relacije nad dimenzijama, kao što su generalizacija i agregacija [19].

3.2.2 Multidimenzioni model objekti veze (ME/R)

Autori pri definisanju Multidimenzionog modela objekti veze (eng. *Multidimensional Entity Relationship Model – ME/R*) polaze od nekoliko osnovnih tvrdnji: da multidimenziona paradigma najbliže oslikava potrebe sistema za podršku odlučivanju i da konceptualni modeli zasnovani na relacionim ili objektno orijentisanim postavkama ne daju odgovarajuću podršku definisanju multidimenzione strukture. Stoga je neophodno definisati konceptualni multidimenzioni model koji je sposoban da izrazi semantiku multidimenzione strukture na odgovarajući način.

ME/R model proširuje semantiku inicijalnog MOV modela korišćenjem apstrakcije specijalizacije. Kao što je prikazano na Slici 12, gde je dat metamodel ME/R, svi novouvedeni koncepti (iscrtani isprekidanom linijom) su izvedeni iz već postojećih koncepata MOV.



Slika 12. Metamodel ME/R [18]

Novouvedeni koncepti su:

- Specifičan tip objekta 'Dimension level' koji ima sve osobine tipa objekta MOV sa dodatnom semantikom da predstavlja jedan hijerarhijski nivo dimenzije čime se obezbeđuje granularnost dimenzije i mogućnost obavljanja specijalnih operacija nad dimenzionim modelom (npr. *roll-up* i *drill-down*).
- Specifičnu n-arnu 'Fact' vezu koja predstavlja skup događaja u sistemu i koja može da poseduje dodatne karakteristike (eng. *quantifying data*).
- Specifičnu binarnu vezu 'Rolls-up' koja predstavlja vezu između dva *Dimension level* objekta čime se gradi hijerarhija dimenzije.

Primetno je da ME/R model ne poseduje jedan od središnjih koncepata multidimenzione paradigme – dimenziju, već je ovaj koncept izveden iz 'Dimension level' objekata koji su povezani 'Rolls-up' vezom.

3.2.3 Konceptualni GMD Model

CGMD je konceptualni model zasnovan na MOV modelu koji je proširen agregiranim multidimenzionim entitetima. Novouvedeni tipovi objekata su podeljeni u dve vrste:

- Prosti agregirani objekti – tipovi koji nastaju agregiranjem više tipova objekata koji pripadaju istom domenu, čime je moguće izgraditi dimenzije sa više hijerarhijskih nivoa (npr. izgradnja vremenske dimenzije koja je sastavljena od dana, meseca, kvartala, godine i sl.).
- Višedimenzioni agregirani tipovi – tipovi objekata koji predstavljaju hijerarhijske nivoe čime je moguće definisati dimenzije koje agregiraju više različitih domena.

Obe vrste agregiranih objekata su strukturni objekti i mogu imati dodatne atribute. Iako se u MOV uvodi dodatna semantika za nove tipove objekata (prvenstveno za Višedimenzioni agregirani tip), dodati elementi se ne razlikuju mnogo u odnosu na već postojeća proširenja MOV kao što je pokazano u [85] i [83].

3.3 Jedinstveni jezik za modelovanje - UML

Nakon nastanka i sve učestalije primene objektno orijentisanih jezika u implementaciji, usledio je razvoj jezika i alata za objektno orijentisanu analizu i projektovanje. Pristup koji se izdvojio kao de facto standard je Jedinstveni jezik za modelovanje (eng. *Unified Modeling Language* – *UML*). UML [41] je sublimacija tri, u to vreme postojećih objektnih pristupa: Object Modeling Technique – OMT, Booch Method i Object-Oriented Software Engineering – OOSE metode. Tri autora (Grady Booch, Jim Rumbaugh i Ivar Jacobson) su na osnovu postojećih pristupa definisali Jedinstveni metod (eng. *Unified Method*) koji je pre podnošenja OMG - nezavisnom telu za standardizaciju, preimenovan u UML. Na taj način nastao je jezik koji je trenutno (verzija UML 2.0) skup 14 metamodela za specifikaciju strukture i ponašanja sistema.

Posebna snaga UML jezika je u tome što omogućava definisanje UML profila čime se postiže proširenje osnovnih UML koncepata, odnosno pridaje se dodatna semantika ograničavanjem osnovnih koncepata korišćenjem apstrakcije generalizacije.

Svi UML metamodeli su opisani kroz MOF [24][99] meta metamodel (eng. *Meta Object Facility*) koji predstavlja M^3 model MDA arhitekture. Na taj način MOF i sama organizacija metamodela u okviru MDA arhitekture predstavljaju stabilnu osnovu koja se može koristiti za definisanje metamodela koji opisuju skladište podataka jer se obezbeđuje interoperabilnost metamodela i preuzimanje semantike osnovnih koncepata.

U daljem tekstu će biti prikazani pristupi zasnovani na UML jeziku. Jedan od realizovanih primera je i ranije opisani CWM metamodel.

3.3.1 OOMD

OOMD (eng. *Object Oriented Multidimensional Model*) [100] je verovatno prvi multidimenzioni model zasnovan na objektno orijentisanim konceptima. Ovaj model je u sledećem radu preimenovan u GOLD model [101]. Kao osnovni razlog zašto je model zasnovan na objektno orijentisanim konceptima, autori navode

korišćenje apstrakcija koje omogućavaju potpunu nezavisnost od načina fizičke implementacije modela.

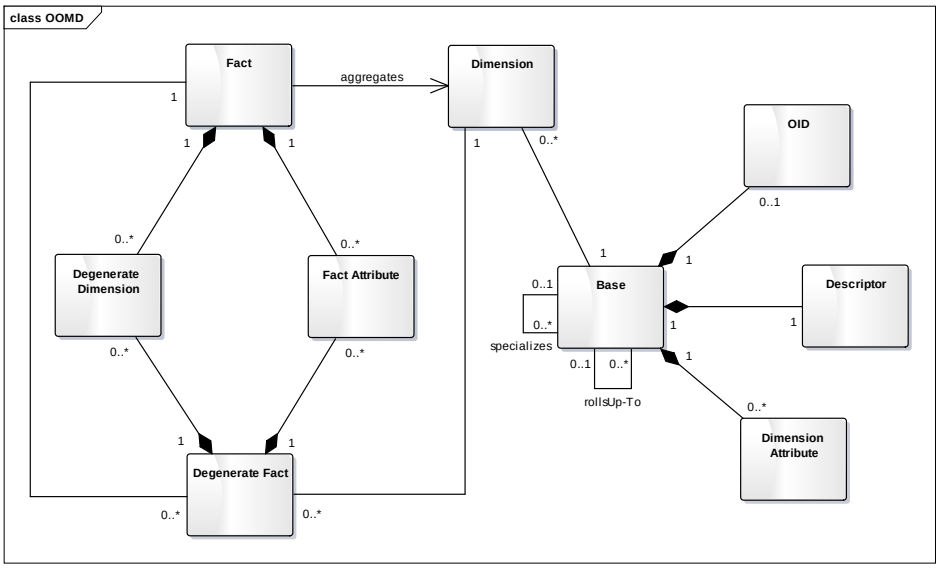
Definisana su tri osnovna koncepta [100] koji opisuju multidimenzioni model:

- Dimenzija (eng. *Dimension Class - DC*) predstavlja klasu čija se struktura sastoji od ključnih atributa (KA) koji jedinstveno identifikuju svaki dimenzioni objekat (DO), neključnih atributa (DA) koji opisuju posmatranu dimenziju, roll-up preslikavanja (ARR) između različitih nivoa hijerarhije dimenzije i operacija nad dimenzionim objektima - ‘new' i 'delete'.
- Fekt (eng. *Fact Class - FC*) je klasa sa sličnom strukturom, sa razlikama da KA skup atributa predstavlja agregaciju svih KA atributa dimenzionih objekata sa kojima je fekt objekat (FO) u vezi, i što umesto DA atributa, Fekt klasa poseduje agregatne (FA) atribute (eng. *Measure*).
- Kocka (eng. *Cube Class - CC*) je agregacija DC i FC klasa, kompletnih KA i podskupa pripadajućih DA i FA atributa. Pored mogućih operacija (‘new' i 'delete', u strukturu CC klase su uključeni i uslovi koji definišu vrednosti koje atributi objekata treba da imaju kako bi pripadali ekstenziji skupa CC.

Prateći osobine objektno orijentisane paradigme, sve klase enkapsuliraju statičke i dinamičke karakteristike dimenzionog modela. Dinamičke karakteristike su praktično sve moguće operacije nad konceptima strukture u koje spadaju: drill-down, roll-up, slice/dice, pivoting i slično.

U kasnijim radovima [15], OOMD model je definisan kao UML profil, kao što je prikazano na Slici 13, sa određenim unapređenjima u odnosu na dotadašnji rad. Naime, određeni koncepti su dodati ili su postojeći revidirani, i to:

- Rola. Eksplicitno se uvodi postojanje preslikavanja na asocijacijama kako bi se izbegle ciklične hijerarhije na dimenzijama.
- Dimenzija. Za razliku od prvobitne zamisli, gde su hijerarhijski nivoi dimenzija bili definisani kao atributi unutar dimenzije, u novom pristupu se dimenzija i hijerarhijski nivoi definišu nezavisno.

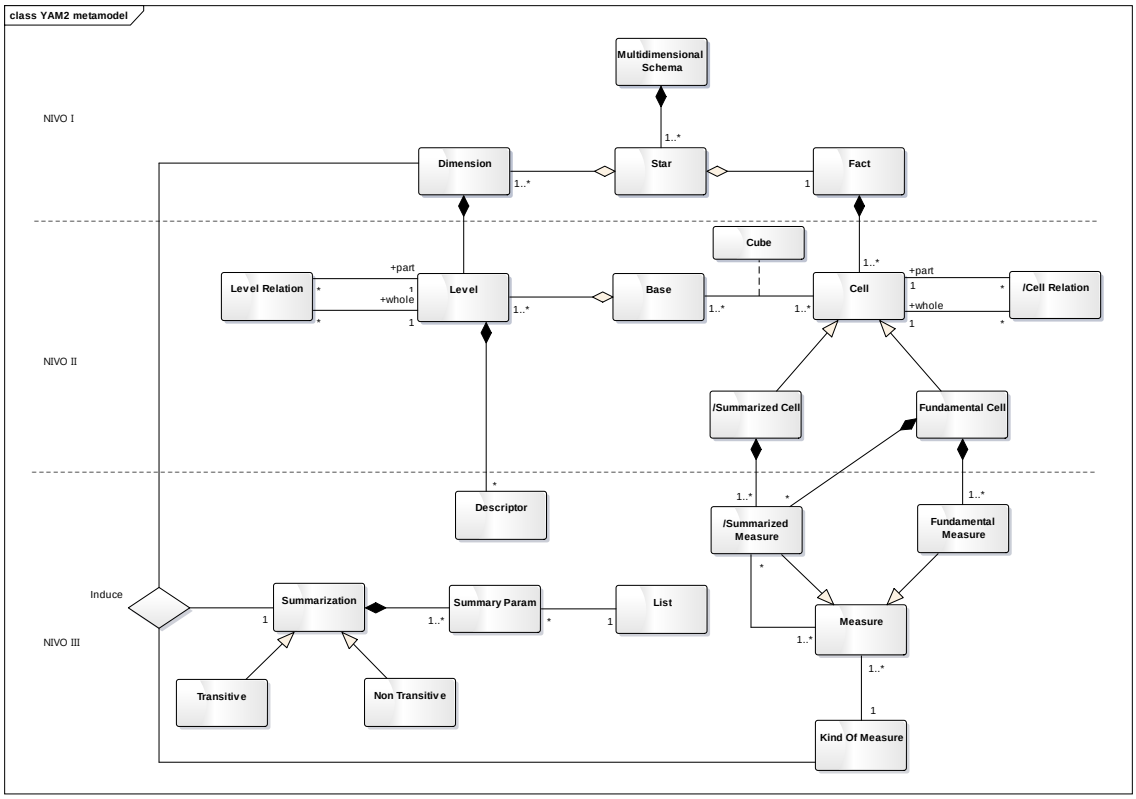


Slika 13. Metamodel OOMD [15]

- Asocijacija kao klasa se koristi za opisivanje M:M veza između Fekta i Dimenzije čime se omogućava definisanje dodatnih atributa na ovim vezama, što u prethodnom radu nije bilo izvodljivo.

3.3.2 YAM^2

Yet Another Multidimensional Model - YAM^2 [14] je konceptualni model za specifikaciju dimenzionog modela definisan kao UML profil. Njegovi koncepti su izvedeni iz MOF modela čime YAM^2 praktično pretenduje da postane sastavni deo predložene OMG MDA arhitekture. Kao što je prikazano na Slici 14, u definisan model su ugrađeni koncepti dimenzionog modela se posebnim akcentom na rešavanju problema povezivanja nezavisnih Centara podataka. Stoga, autori preuzimaju apstrakcije objektno orijentisane paradigme kao što su generalizacija, agregacija, tok, asocijacija i sl. i definišu koje apstrakcije je moguće koristiti u različitim slučajevima povezivanja Fektova i Dimenzija.



Slika 14. Metamodel YAM^2 [106]

Autori ističu da su osnovne prednosti YAM^2 modela:

- Semantička izražajnost (eng. *Semantic Power*) koja predstavlja visok stepen semantike u kojem model može da opiše koncepte realnog sistema.
- Semantička prilagodljivost (eng. *Semantic Relativism*) predstavlja mogućnost modela da istovremeno obuhvati više različitih subjektivnih predstava realnog sistema.
- Omogućavanje grupisanja i klasifikacije podataka koje olakšava kasniju primenu agregatnih funkcija nad podacima.

Postupak projektovanja je zasnovan na postepenom uvođenju detalja u model i to:

- Nivo I u kojem se identifikuju osnovni koncepti multidimenzionog modela (*Fact* i *Dimension*)
- Nivo II povezuje identifikovane koncepte odgovarajućom vrstom dozvoljene veze (*Cell Relation* i *Level Relation*).

- Nivo III predstavlja postupak dodavanja pripadajućih atributa na definisane koncepte (*Measure*, *Descriptor*...)

Nivoi koji omogućavaju postepenu izgradnju modela su prikazani na Slici 14, zajedno sa pripadajućim konceptima.

3.4 Semantički modeli

Semantički modeli nisu nastali iz nijednog opštepoznatog jezika za modelovanje, već su definisani konceptima koji ne pretpostavljaju prethodno poznavanje ili korišćenje jezika kao što su CWM, UML ili MOV. Pored toga, semantika svakog koncepta je jasna pri čemu se multidimenzionom modelu pridaje adekvatan značaj čime se projektantu omogućava intuitivan rad [97].

U proteklom periodu je definisano više semantičkih modela [103] [104] [105], međutim, najpotpuniji i najpoznatiji je Dimenzioni Fekt Model, opisan u sledećoj sekciji.

3.4.1 Dimenzioni Fekt Model

Dimenzioni Fekt Model – DFM (eng. *Dimensional Fact Model*) je model koji je, uz prateću metodologiju razvoja skladišta podataka, nastao 1998. godine i koji je u proteklom periodu doživeo više unapređenja.

Realni sistem modelovan DFM modelom se naziva Dimenziona šema. Dimenziona šema se sastoji iz više Fekt šema koje su formalno opisane kao usmereni, aciklični grafovi [13].

Osnovni elementi Fekt šeme su fekti, agregatni atributi, dimenzije i hijerarhije:

- Fekt je koncept koji je relevantan u procesu odlučivanja i predstavlja skup događaja koji se odvijaju u nekom sistemu.
- Agregatni atribut je numerička vrednost fekta koja opisuje njegove kvantitativne aspekte.
- Dimenzija je konačan skup vrednosti koji predstavlja jednu osobinu fekta.

- Hijerarhija je usmereno stablo čiji čvorovi su dimenzioni atributi i čije grane definišu 1:M vezu između dimenzionih atributa.

DFM je konceptualni model prvenstveno nastao da podrži fizički Dimenzioni model. Stoga se u značajnoj meri bavi situacijama koje su specifične za dimenziono modelovanje, pri čemu predviđa koncepte kao što su [9]:

- Deskriptivni atribut (eng. *Descritpive Attributes*) je osobina koja bliže opisuje atribute dimenzije. Osnovna karakteristika deskriptivnih atributa je da pri tom ne učestvuju u agregatnim operacijama koje se obavljaju nad hijerarhijom dimenzije. Pored toga, deskriptivni atribut može biti osobina Fekta, u smislu bližeg opisa događaja, ali ne i igranja uloge agregatnog atributa.
- Međudimenzioni atributi (eng. *Cross-Dimensional Attributes*) je dimenzioni ili deskriptivni atribut čije vrednosti su definisane sa dva ili više dimenziona atributa koji mogu pripadati različitim hijerarhijama.
- Konvergencija (eng. *Convergence*) je kada se kroz hijerarhiju sa bar dve različite putanje može doći do istog čvora. U tom slučaju struktura hijerarhije nema jedinstvenu putanju u stablu.
- Deljene hijerarhije (eng. Shared Hierarchies). Usled postojanja potrebe da pojedini koncept učestvuje u više hijerarhija, kada mu se pridaje različita semantika, uvode se Deljene hijerarhije. One predstavljaju delove hijerarhija koji su zajednički za više hijerarhija.
- Višestruke grane (eng. *Multiple Arcs*) predstavljaju koncept kada kardinalnost između nivoa hijerarhije nije 1:M već postoji potreba da atribut dimenzije uzme više vrednosti iz nadređenog nivoa čime kardinalnost veze između nivoa hijerarhije prerasta u M:M preslikavanje.
- Opciona grana (eng. *Optional Arcs*) je grana koje definiše vezu koja nije primenljiva za ceo skup događaja nekog Fekta. Ovako definisana grana praktično ima donju granicu kardinalnosti 0. Takođe, u slučajevima kada dimenzija generalizuje karakteristike različitih objekata, opciona grana se može koristiti i za definisanje klasifikacije (preklapajuća, disjunktna,

potpuna, nepotpuna) date generalizacije (eng. *Overlapping*, *Disjoint*, *Complete*, *Incomplete*).

- Nepotpuna hijerarhija (eng. *Incomplete Hierarchies*) predstavlja hijerarhiju kod koje pojedini atributi dimenzije (tj. nivoi hijerarhije) za određena pojavljivanja dimenzije nisu primenljivi, odnosno, ne postoje.
- Rekurzivna hijerarhija (eng. *Recursive Hierarchies*) predstavlja hijerarhiju koja može biti povezana sama sa sobom proizvoljan broj puta, pri čemu broj nivoa u rekurziji ne mora biti isti za sve instance u dimenziji.

Potrebno je napomenuti da DFM, pošto definiše hijerarhiju kao stablo, ne podržava M:M vezu između Fekta i Dimenzije [102].

4 MODELI STRUKTURE SKLADIŠTA PODATAKA

Model podataka je intelektualno sredstvo za opis statičkih karakteristika sistema, tj. opis objekata, njihovih atributa i međusobnih veza u nekom stacionarnom stanju [83]. Kako se skladište podataka definiše kao model realnog sistema koji predstavlja skup stanja tog sistema u određenom vremenskom periodu, neophodno je da modeli podataka koji se koriste za razvoj skladišta podataka mogu da podrže opis sistema kroz različita stanja i da na adekvatan način omoguće održavanje promena nastalih u realnom sistemu, odnosno izvorima podataka.

Jedna od neusaglašenosti na polju razvoja DW je nepostojanje standardnog modela strukture skladišta podataka. Postojeći pristupi predlažu organizaciju podataka u 3NF [1] ili u višedimenzionom modelu [2]. Oba pristupa imaju ograničenja koja se ogledaju u otežanom održavanju promena strukture izvora. Sa druge strane, oba pristupa su standardizovani kroz odgovarajuće metamodele definisane u CWM metamodelu. Poslednjih godina su se izdvojila još dva pristupa koji pokušavaju da nadomeste ove nedostatke. To su Anchor modeling [5] baziran na podacima normalizovanim u 6NF i Data Vault [6] pristup koji takođe može (ali ne mora) da skladišti podatke normalizovane u 6NF. Ni Anchor Modeling ni Data Vault nisu standardna ekstenzija CWM metamodela.

U ovom poglavlju su prikazani najzastupljeniji modeli podataka koji učestvuju u izgradnji strukture skladišta podataka. Pored kratkog opisa modela podataka i njihovih osnovnih koncepata, u drugom delu poglavlja je data analiza prikazanih modela sa stanovišta više kriterijuma važnih za razvoj skladišta podataka.

4.1 Relacioni model normalizovan u 3NF

Relacioni model (eng. *Relational Model*) je formalni matematički model zasnovan na teoriji skupova. Ubrzo nakon nastajanja, relacioni model je bio ispraćen razvojem odgovarajućih sistema za upravljanje bazama podataka koje su ga podržavale, tako da je danas najzastupljeniji model za realizaciju transakcionih sistema i skladišta podataka.

Relacioni model je osmislio Edgar Kod (eng. *Edgar Frank Codd*) sa ciljem da ukloni probleme redundanse i nekonzistentnosti podataka koje su posedovali mrežni i hijerarhijski model, kao i savladavanja kompleksnosti tih modela. Osnovne koncepte je postavio 1969. godine u delu [89], da bi sledeće godine proširio taj rad sa normalizacijom relacija čime je definisao postupak da vrednosti atributa relacije budu atomske [90]. Naime, u prvom radu, relacioni model je dozvoljavao koncept „relacije u relaciji“ (kada su elementi Domena relacije), koji je na neki način bio preteča objektnog i XML modela.

Središnji koncept modela je **Relacija** koja predstavlja podskup Dekartovog proizvoda kolekcije skupova S1, S2, ... , Sn. Svaki od skupova koji učestvuje u Dekartovom proizvodu predstavlja **Domen** relacije. Domeni mogu da budu *predefinisani* – skupovi podataka koji pripadaju ugrađenim tipovima podataka podržanim u SUBP i *semantički*, kada su ti skupovi korisnički definisani nad nekim predefinisanim ili semantičkim domenom, pri čemu im se dodeljuje odgovarajuće značenje [83]. Broj skupova tj. domena nad kojima je definisana relacija predstavlja **Stepen** relacije. **Kardinalnost** relacije predstavlja broj n-torki jedne relacije. Kako je definisano da su sve n-torke jedne relacije različite, postoji atribut (ili kolekcija atributa) koji jedinstveno identifikuje n-torku u tabeli i koji se naziva **Ključ** relacije [89]. U sledećem radu Kod ključ relacije (odabrani identifikator, jer relacija može da ima više ključeva koji jedinstveno identifikuju n-torke) naziva **Primarni ključ** i uvodi koncept **Spoljnog ključa** koji predstavlja atribut (moguće složen) relacije R koji je primarni ključ relacije S, pri čemu nije isključeno da su relacije R i S identične [89].

Kao što je već navedeno, Kod u [90] uvodi koncept normalizacije, čime definiše još jedno pravilo koje skup treba da zadovolji da bi bio relacija – da svi atributi relacije uzimaju vrednosti iz prostih domena, tj. da sve vrednosti relacije budu atomske, čime se relacija nalazi u Prvoj normalnoj formi. Normalizacija je postupak dovođenja relacija u odgovarajuću normalnu formu čime se otklanjaju anomalije u ažuriranju (dodavanju, brisanju ili promeni sadržaja relacije) [83]. U narednom periodu Kod je uveo Drugu, Treću i Bojs-Kodovu normalnu formu kao pokušaj da u potpunosti ukloni redundansu relacija. Dakle, normalizacija se može posmatrati kao opšti postupak dekompozicije relacija na relacije koje predstavljaju fundamentalne objekte sistema i njihove veze, čime se garantuje minimum redundanse podataka i održavanje relacija bez anomalija [83].

U domenu skladišta podataka, neformalno, se za model koji se nalazi bar u 3NF ustalio naziv "normalizovan model". Na polju razvoja skladišta podataka dugi niz godina traje rasprava da li je pogodnije za model strukture koristiti tzv. model u 3NF (čiji je zagovornik Inmon), koji ukida redundansu podataka, ili dimenzioni model (potenciran od strane Kimbala), koji se nalazi u 1NF i koji poboljšava performanse i razumljiviji je krajnjim korisnicima u fazi analize podataka.

Obzirom da postoje jasna pravila transformacije između MOV i relacionog modela [83], i da pravilno projektovan MOV model može biti transformisan u normalizovan relacioni model koji zadovoljava bar 3NF (a najčešće i 5NF) u ovom radu će se za prikaz tzv. „skladišta podataka zasnovanih na 3NF" biti korišćen FMOV (Fizički model objekti veze) [83].

4.2 Data Vault

Data Vault je empirijski model koji se koristi u razvoju skladišta podataka više od dve decenije. Osmislio ga je Den Linsted (eng. *Dan Linstedt*) sa ciljem da omogući potpunu pratljivost podataka, kao i veću skalabilnost i adaptivnost nego što su podržavale aktuelne arhitekture skladišta podataka [29]. Bil Inmon, koji je predložio sledeću generaciju arhitektura skladišta podataka – DW 2.0 [11], je prepoznao Data Vault kao optimalan izbor te arhitekture.

Prilikom definisanja Data Vault modela, Linsted je pošao od pretpostavke da objekti realnog sistema imaju jedinstven identifikator koji ih jednoznačno određuje, a da su sve ostale karakteristike objekta promenljive u vremenu. Stoga je jedan tip objekta definisao kao dva koncepta, prvi - entitet sa identifikatorom i drugi – koji sadrži ostale pripadajuće, tzv. opisne atribute koji mogu da budu organizovani u više različitih skupova – domena.

Entitet sa pripadajućim identifikatorom je ujedno i osnovni koncept DV modela - Hab (eng. *Hub*) [30] koji je predstava realnih ili apstraktnih objekata realnog sistema koji mogu biti jedinstveno identifikovani, pri čemu je identifikator svojstvo koje je nepromenljivo ili je promenljivo u izuzetnim slučajevima. Prema Linstedovoj notaciji, Hab se predstavlja pravougaonikom koji je imenovan sa prefiksom „H_“ i nazivom Haba.

Struktura Haba je podložna promenama i ona se definiše preko jednog ili više Satelita. Satelit [30] je domen, moguće složen i predstavlja jednu ili više karakteristika Haba. Karakteristike Haba od kojih se Satelit sastoji se mogu grupisati prema više različitih kriterijuma: izvoru podataka iz kojeg dolaze, frekventnosti promene vrednosti karakteristika, organizacionoj jedinici i slično. Sateliti se na DV modelu predstavljaju pravougaonikom koji je imenovan sa prefiksom „S_“ i nazivom Satelita.

Habovi mogu biti povezani preko događaja u sistemu, strukturnih i drugih veza što se ostvaruje korišćenjem Link koncepta [30]. Link može imati svoje dodatne karakteristike – Satelite. Empirijski DV model dozvoljava Link – Link vezu. Pored

toga Link je koncept koji omogućava n-arnu vezu Habova. Link se na modelu predstavlja pravougaonikom koji se imenuje prefiksom „L_“ i nazivom Linka.

Pri realizaciji DV modela, sve instance osnovnih koncepata implicitno poseduju metapodatke koji ih dodatno opisuju, i to:

- *Surogat ključ* koji jedinstveno identifikuje objekte Haba, Linka ili Satelita
- *Izvor podatka* je atribut u kojem se čuva izvor iz kojeg je podatak prvi put učitan
- *Identifikator žurnala* je veza ka log tabeli koja skladišti vreme učitavanja podatka, trajanje učitavanja i slične informacije o ETL procesu

Pored nabrojanih, Satelit poseduje još najmanje dva metapodatka:

- *Početak važenja*, koji predstavlja datum i vreme od kada vrednosti atributa egzistiraju u sistemu (ili su učitane u Satelit)
- *Prestanak važenja*, koji predstavlja datum i vreme kada su vrednosti atributa zamenjene drugim vrednostima.

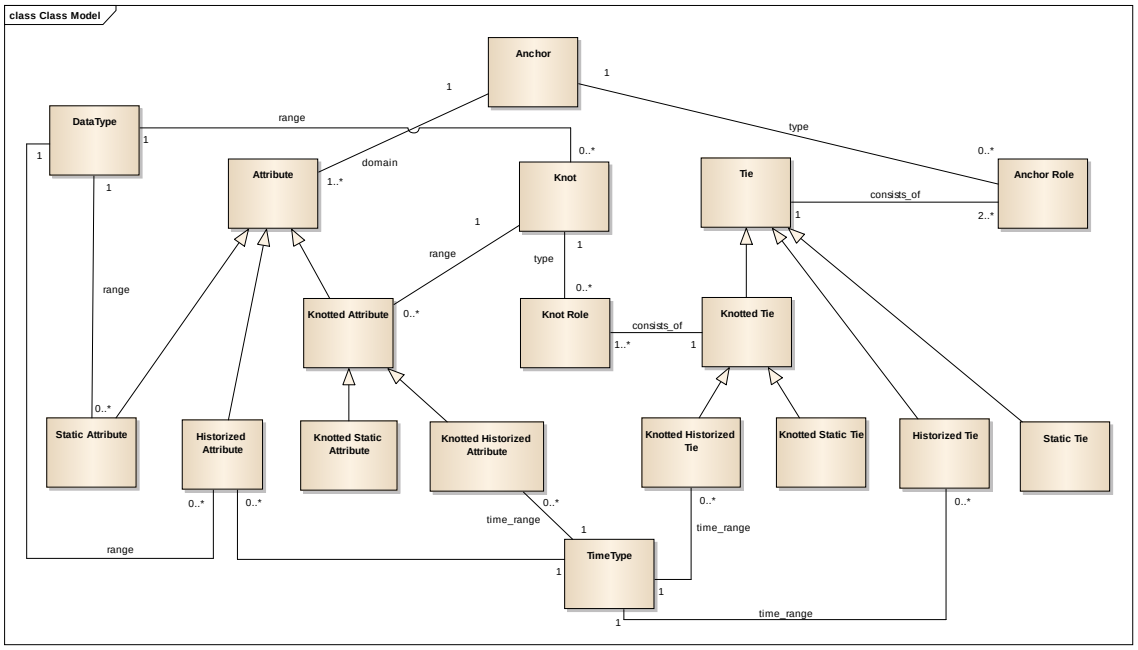
čime se definiše period važenja atributa koji su upisani u Satelit.

Kao što je već naglašeno, konkretan Data Vault model je isključivo fizički model koji nastaje primenom jednostavnih pravila transformacije nekog izvornog modela (npr. relacionog) u koncepte (ciljnog relacionog modela) koji se semantički razlikuju prema korišćenim prefiksima. Na primer, koncept spoljnog ključa RM se uvek realizuje kao Link koncept, što omogućava otpornost DV modela na kasnije strukturne promene ukoliko se, radi demonstracije, kardinalnost između dva izvorna koncepta promeni, te od početne veze nastane agregacija, odnosno veza koja ima dodatne karakteristike.

4.3 Anchor Modeling

Iako je prvobitno realizovan kao empirijski model (i metod) za razvoj skladišta podataka 2004. godine [5], Anchor modeling je formalizovan 2010. godine [36]. Nastao je prvenstveno da bi obezbedio proširivost modela skladišta podataka i da bi podržao agilne metodologije razvoja skladišta podataka. Na fizičkom nivou, Anchor model je „visoko normalizovan" tj. normalizovan je do nivoa 6NF, tako što su svi atributi koncepata realizovani kao tabele koje su preko koncepta spoljnjeg ključa povezane sa tabelom koja predstavlja tip objekta. Primetno je i da je Anchor model isti na logičkom i fizičkom nivou, tj. transformacija 1:1 preslikavanje između ova dva modela.

Anchor model se sastoji od četiri koncepta koji se semantički mogu proširivati korišćenjem predefinisanih podtipova čime se detaljnije opisuje njihova uloga u sistemu (npr. da li je koncept nepromenljiv ili se njegovo stanje prati istorijski). Metamodel Anchor modela [36] je prikazan na Slici 15.



Slika 15. Metamodel Anchor modela u UML notaciji

Osnovni koncept je Sidro (eng. *Anchor*) koji predstavlja skup objekata realnog sistema. Koncept Čvor (eng. *Knot*) je vrlo sličan Sidru, ali sa jednom bitnom razlikom – predstavlja skup objekata koji nisu promenljivi u vremenu. Takođe, objekti Čvora

mogu biti deljeni između više instanci nekog Sidra. Atribut (eng. *Attribute*) je koncept koji opisuje strukturu Sidra. Postoji više podtipova Atributa i to:

- Nepromenljivi atribut (eng. *Static Attribute*) koji opisuje karakteristiku Sidra za koju nije bitno voditi istoriju promena
- Promenljivi atribut (eng. *Historized Attribute*) je atribut kod kojeg se pamti svaka promena vrednosti posmatranog atributa.
- Nepromenljivi šifarski atribut (eng. *Knotted Static Attribute*) predstavlja preslikavanje između Sidra i Čvora pri čemu ovakvo preslikavanje nije promenljivo sa vremenom
- Promenljivi šifarski atribut (eng. *Knotted Historized Attribute*) je atribut koji vrednost uzima iz skupa vrednosti nekog Čvora pri čemu se ova vrednost može menjati sa vremenom.

Da bi se definisao poslednji koncept Veza, uvodi se koncept Rola (eng. *Role*) koji predstavlja preslikavanje dva skupa objekata. Kako su i Sidro i Čvor skupovi objekata, definišu se dve vrste rola: Rola Sidra i Rola Čvora. Veza (eng. *Tie*) je koncept koji predstavlja asocijaciju između dva ili više Sidra (ili Čvora) i sastoji se od bar dve role. Analogno atributima, postoji četiri podtipa koji preciznije definišu Vezu:

- Nepromenljiva veza (eng. *Static Tie*) je skup najmanje dve Role Sidra.
- Promenljiva veza (eng. *Historized Tie*) predstavlja skup najmanje dve Role Sidra i vremenske dimenzije.
- Nepromenljiva šifarska veza (eng. *Knotted Static Tie*) je skup bar dve Role Sidra i jedne ili više Rola Čvora.
- Promenljiva šifarska veza (eng. *Knotted Historized Tie*) je skup bar dve Role Sidra, jedne ili više Rola Čvora i vremenske dimenzije.

4.4 Dimenzioni model

Dimenzioni model je u praksi verovatno najzastupljeniji model strukture koji se koristi u realizaciji skladišta podataka. Najveća prednost dimenzionog modela je sa stanovišta korišćenja i analize podataka, jer je ovaj model dovoljno intuitivan i razumljiv krajnjim korisnicima koji olakšano mogu sami da kreiraju upite nad skladištem podataka. Pored toga, dimenzioni model zbog svoje strukture i redundanse koju podrazumeva, pruža dosta dobre performanse pri vraćanju rezultata postavljenih upita.

Sam dimenzioni model je rezultat istraživačkih univerzitetskih projekata šezdesetih godina prošlog veka koji su imali za cilj pojednostavljenje prezentovanja analitičkih podataka [12]. Najveći zagovornik korišćanja ovog modela je Ralf Kimbal (eng. *Ralph Kimball*) koji je nad dimenzionim modelom definisao celokupnu metodologiju razvoja skladišta podataka [2].

Osnovni koncepti dimenzionog modela su Dimenzija i Fekt. Fekt (eng. *Fact*) predstavlja skup događaja koji se dešavaju u poslovnom sistemu. Dimenzija (eng. *Dimension*) je koncept predstavlja informacije koje opisuju fekte [9]. Dimenzija može biti denormalizovana (eng. *Star Schema*) kada se celokupna hijerarhija dimenzije redundantno čuva u jednoj tabeli. Drugi pristup je da se dimenzija organizuje kao normalizovan podmodel (eng. *Snowflake Schema*) kada se redundansa dimenzije svodi na minimum, pri čemu je moguće tabele koje učestvuju u hijerarhiji dimenzije koristiti i u drugim dimenzijama. Uobičajeno je da se pri realizaciji skladišta podataka koristi isključivo jedan od ova dva pristupa.

Kimbal predlaže denormalizovane dimenzije [12] zbog performansi, pri čemu uvodi koncept usaglašene dimenzije (eng. *Conformed Dimension*) čija je odlika da se koristi jedinstveno u celokupnom skladištu podataka, deljena između više različitih centara podataka (eng. *Data Mart*).

Pored toga, prema Kimbalu, centri podataka (u čijoj osnovi je Fekt sa pripadajućim Dimenzijama) se definišu prema osnovnim jedinicama posla organizacije, tj. za svaku poslovnu funkciju se realizuje odgovarajući centar podataka. Na ovaj način se

gradi matrica poslovnih procesa i dimenzija nad kojim se poslovni procesi izgrađuju, čime se izbegava redundansa na nivou celokupne dimenzije u okviru jednog skladišta podataka.

Kako se promene u Dimenzijama ređe dešavaju nego u Fektima, Kimbal je u [12] definisao tri tipa promene dimenzije (eng. *Slowly Changing Dimensions* - *SCD*), dok je u [39] dodao još četiri tipa promene dimenzija (pre svega zbog slučajeva učestale promene vrednosti Dimenzija), ukoliko se ne računa tip 0 koja predstavlja specijalan slučaj SCD u kojoj se promene nikad ne beleže:

- SCD tip 1 postojeću vrednost zamenjuje novom što znači da se istorija vrednosti nekog atributa ne čuva
- SCD tip 2 dodaje novu vrednost u dimenziju što uslovljava korišćenje surogat ključa, jer identifikator objekta, zbog praćenja istorije, više nije jedinstven na nivou dimenzije. Svaka dimenzija koja se održava na ovaj način ima tri dodatna atributa: *Početak važenja, Kraj važenja i Indikator trenutno važeće vrednosti*.
- SCD tip 3 u dimenziju dodaje novi atribut koji skladišti staru tj. alternativnu vrednost (eng. *alternate reality*) kako bi se ona sačuvala. Korisnici sistema mogu da grupišu podatke ili po osnovu važeće ili po osnovu alternativne vrednosti.
- SCD tip 4 promene dimenzije se koristi kada se jedan deo atributa učestalo menja (ili koristi u analitici) kada se taj podskup atributa izmesti u tzv. *mini dimenziju*. Novonastala dimenzija ima sopstveni surogat ključa, pri čemu se primarni ključevi obe dimenzije spuštaju u pridruženi Fekt.
- SCD tip 5 predstavlja kombinaciju tipa 4 i tipa 1, jer su mini dimenzija i osnovna dimenzija povezane referencijalnim integritetom preko primarnog ključa mini dimenzije. Na taj način je sa Fektom povezana samo osnovna dimenzija. Dimenzija i pripadajuća mini dimenzija se na prezentacionom sloju prikazuju kao jedna tabela.
- SCD tip 6 je nastao kombinacijom korišćenja SCD tipova 1, 2 i 3. U ovom tipu se za svaku promenu atributa dodaje nova n-torka sa periodima važenja (tip 2), pri čemu se uvodi dodatna kolona za trenutno važeću vrednost

posmatranog atributa (tip 3). Dalje, svaka promena uslovljava ažuriranje sa trenutno važećom vrednošću (tip 1).

- SCD tip 7 je prilagođen za Dimenzije koje imaju veliki broj atributa. U ovom tipu Dimenzija pored surogat ključa, u Fekt upisuje i poslovni ključ, tj. istorijski nepromenljivi identifikator objekta. Na ovaj način se Fekt može analizirati i kao tip 1 i kao tip 2.

4.5 Analiza modela strukture skladišta podataka

U ovom odeljku će prikazani modeli strukture biti razmatrani sa nekoliko stanovišta, i to:

- analiza ugrađene semantike za skladišta podataka,
- analiza vremenskog aspekta modela strukture,
- analiza otpornosti modela na promene strukture modela izvora podataka, i
- analiza kompletnosti i pratljivosti podataka

kako bi se uočile prednosti i nedostaci postojećih modela sa ciljem definisanja odgovarajućeg modela koji bi adekvatno mogao da odgovori na postavljene zahteve projektovanja i realizacije skladišta podataka.

4.5.1 Analiza ugrađene semantike za skladišta podataka

Na najvišem nivou apstrakcije, svi prikazani modeli su zasnovani na nekoliko fundamentalnih koncepata, kao što je prikazano (Tabela 1).

Tabela 1. Osnovni koncepti modela strukture

	Objekat	**Veza**	**Atribut**	**Identifikator**
Normalizovan model	Relacija	Spoljni ključ	Domen	Primarni ključ
Data Vault	Hab	Link	Satelit	Poslovni / primarni ključ
Anchor Model	Sidro / Čvor	Tie	Atribut	Primarni ključ
Dimenzioni model	Dimenzija	Fekt	Atribut	Poslovni / primarni ključ

Osnovna razlika prikazanih modela je u nivou semantike koji su u njih ugrađeni. Normalizovan model nema nikakvih semantičkih ograničenja i kao takav je vrlo opšt, jer se izgradnja bilo kog modela zasniva na funkciji preslikavanja skupova. Ovaj model nema implicitnu predstavu praćenja istorije promene strukture ili objekata.

Data Vault pretpostavlja da objekti realnog sistema imaju nepromenljivi identifikator i donekle menja strukturu izvora na taj način što dozvoljava proizvoljnu organizaciju strukture objekata podelom te strukture na više satelita. Sateliti su prilagođeni praćenju promene vrednosti atributa, što nije slučaj za identifikator Haba.

Anchor Model je visoko normalizovan, ali za koncepte realnog sistema predviđa da mogu da se realizuju kroz dva koncepta: Sidro i Čvor. Takođe, Anchor Model ima mogućnost praćenja istorije svih objekata osim u slučaju Čvora.

Dimenzioni model je zbog performansi i razumljivosti prilagođen potrebama krajnjih korisnika i stoga je zasnovan na događajima koji povezuju objekte u sistemu. Zbog takvih zahteva je denormalizovan, pri čemu se praćenje promena vrednosti u sistemu zasnivaju na složenim pravilima promene dimenzija kao što je prikazano ranije.

Svi modeli imaju po jednu predstavu objekta (ili entiteta) osim Anchor modela koji koncepte realnog sistema može definisati preko Sidra ili Čvora. Osnovna razlika je da su Normalizovan model, Data Vault i Anchor Model normalizovani, dok je Dimenzioni model denormalizovan (više objekata u jednoj tabeli sa redundantnim podacima).

U Normalizovanom modelu, veze između objekata su predstavljene preko *spoljnog ključa*, koji ostvaruje „čvrstu vezu“ jer se promenom strukture relacije referencira primarni ključ neke druge relacije. Data Vault, Anchor Model i Dimenzioni model veze između objekata definišu preko koncepata *link*, *veza* i *fekt*, respektivno, pri čemu se veza realizuje kao tabela koja skladišti primarne ključeve objekata koji su u vezi (ne mora da bude binarna). Pored toga, uobičajeno je da Dimenzioni model u strukturi fekta ima i dodatne izvedene atribute ili atribute agregiranih vrednosti.

Kada se razmatraju atributi, Data Vault i Anchor Model strukturu objekta čuvaju nezavisno od objekta, korišćenjem *satelita* i *atributa* koji se referenciraju na objekat preko spoljnog ključa. Razlika ova dva modela je što se u Anchor Modelu svaki pojedinačni atribut realizuje kao nezavisna tabela, dok se u Data Vault modelu

atributi mogu grupisati po različitim kriterijumima u više satelita za jedan objekat, kao što je ranije navedeno. Normalizovan model i Dimenzioni model održavaju atribute u okviru strukture objekta, tj. relacije i dimenzije.

Koncepti svih modela koji predstavljaju tipove objekata imaju identifikator, osim što Data Vault model pretpostavlja postojanje poslovnog ključa, tj. identifikatora koji jedinstveno određuje objekat u realnom sistemu. Donekle slično, Dimenzioni model, pri korišćenju SCD tipa 2 očekuje postojanje poslovnog ključa, preko kojeg se vrednosti dimenzije grupišu.

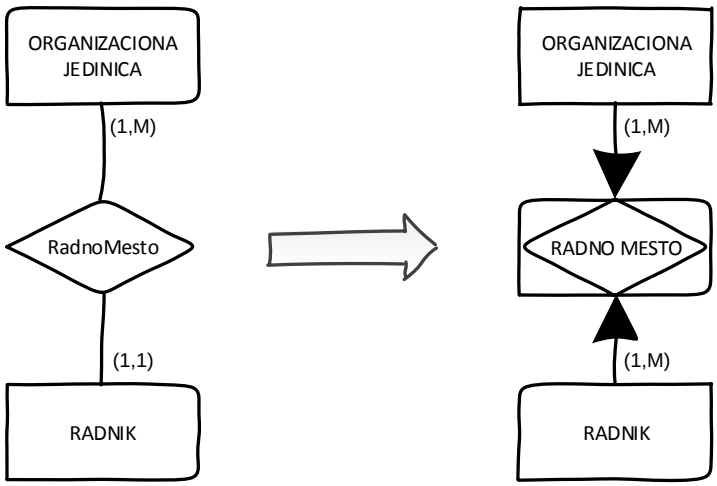
4.5.2 Analiza otpornosti modela na promene strukture izvora podataka

Stalne promene (organizacione, zakonske, funkcionalne i dr.) kojima je izložen realni sistem, odražavaju se i na odgovarajuće skladište podataka. Iz navedenog proizilazi jedan od osnovnih problema razvoja i održavanja DW: nekonzistentnost realnog sistema i skladišta podataka koja se vremenom povećava. Stoga je jedan od osnovnih zahteva pri projektovanju skladišta podataka sposobnost skladišta podataka da apsorbuje promene strukture izvora bez menjanja sopstvene strukture, sa ciljem da se olakša održavanje i buduća proširenja.

Ovaj odeljak razmatra kako prikazani modeli strukture ispunjavaju nefunkcionalne zahteve kao što su prilagodljivost i proširivost. Analiza je pre svega izvedena tako što će se sagledati da li analizirani modeli imaju sposobnost da u slučaju promene strukture izvora, isključivo dodaju, a ne i menjaju već postojeće koncepte skladišta podataka na fizičkom nivou.

Normalizovan model se pokazao kao optimalan model podataka za transakcione sisteme, jer garantuje minimum redundanse podataka i jer dekomponuje strukturu sistema do nivoa fundamentalnih objekata. Međutim, u domenu skladišta podataka iskazuje određene slabosti, jer zahtevi koji se postavljaju pred skladištem podataka nisu isti kao za transakcione sisteme. Osnovni nedostatak Normalizovanog modela je što su i atributi i veze deo strukture objekata sistema, što vodi ka otežanoj prilagodljivosti i proširivosti. Naime, bilo koja promena strukture izvora (atributa ili veza) će kao rezultat imati promenu strukture u Normalizovanom modelu, kao što

je prikazano na Slici 16, gde je pokazana promena kardinalnosti preslikavanja između koncepata Radnik i Organizaciona jedinica.

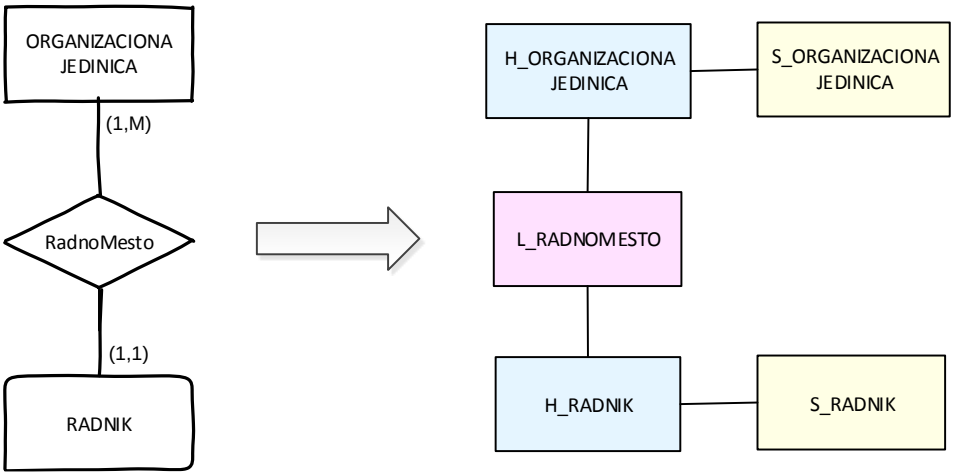


Slika 16. Promena kardinalnosti na Normalizovanom modelu

U prikazanom slučaju je neophodno kreirati tabelu Radno mesto, prebaciti već postojeće podatke koji su odražavali vezu dva koncepta i izbrisati spoljni ključ sa tabele Radnik koji je predstavljao vezu ka tabeli Organizaciona jedinica.

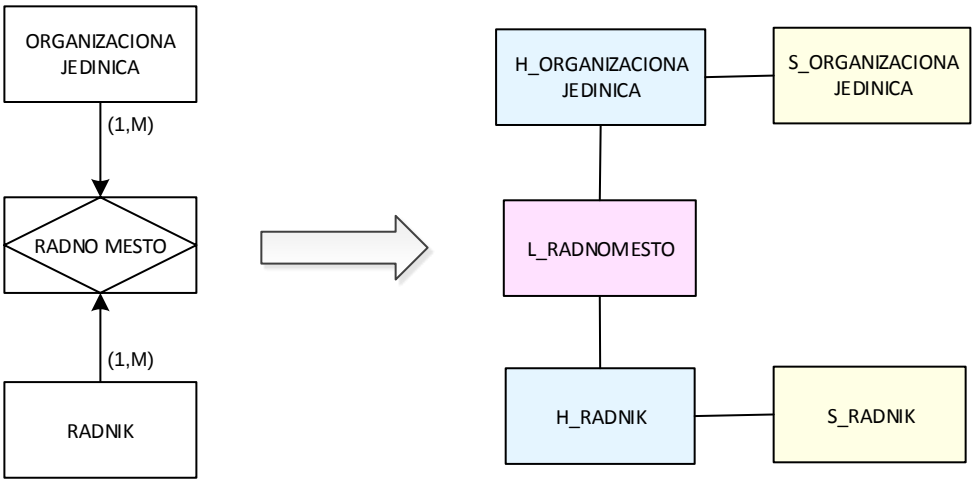
Pored toga, Normalizovan model ima poteškoće pri konsolidaciji modela izvora (eng. *Reconciliation*) jer usled semantičke različitosti koje mogu postojati u modelima izvora, nije moguće projektovati jedinstven model, kao što je Golfareli pokazao u [9].

Data Vault model pokazuje mnogo veću fleksibilnost sa stanovišta promene strukture izvora. Ova fleksibilnost proističe iz samog jezika, tako što je struktura objekta „izmeštena“ u drugi, fizički odvojen koncept – Satelit i što se svaka veza između koncepata tj. Link uvek realizuje kao tabela agregacije objekata koji su u vezi, bez obzira na kardinanost preslikavanja.



Slika 17. Transformacija izvora (1,M veze) u Data Vault model

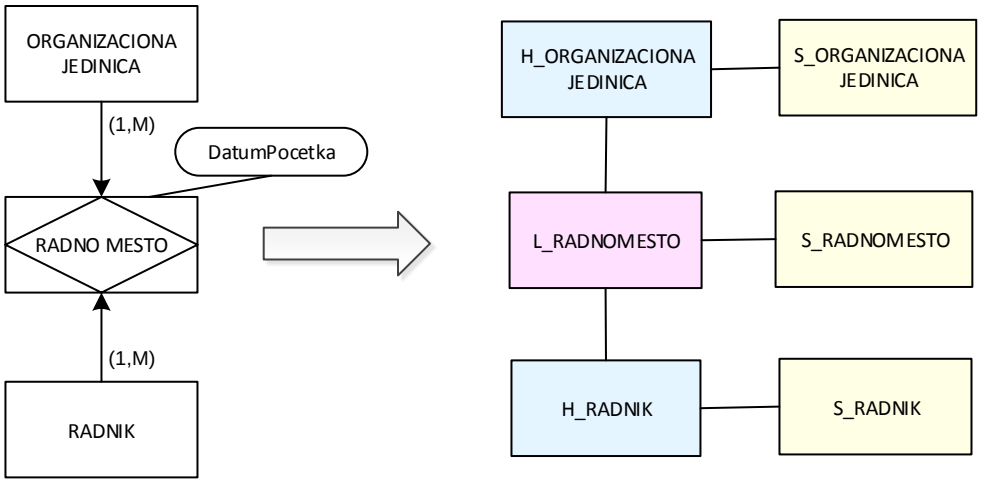
Kao što je prikazano (Slika 17), u početnom strukturnom stanju izvornog sistema, **Radnik** je mogao da bude u radnom odnosu u jednoj i samo jednoj **Organizacionoj jedinici**. Rezultujući DV model ima Habove *H_Radnik* i *H_OrganizacionaJedinica* sa pripadajućim Satelitima. Habovi su povezani odgovarajućim linkom. Ukoliko se struktura modela promeni (npr. zbog uvođenja radnika koji su zaposleni na osnovu ugovora), uslovljena promenom kardinalnosti preslikavanja, i izvorni model dozvoli da jedan Radnik istovremeno radi u više Organizacionih jedinica, ciljni DV model neće pretrpeti nikakve promene, kao što je prikazano na Slici 18.



Slika 18. Transformacija izvora (agregacije) u Data Vault model

Pored toga, ukoliko koncept izvora Radno mesto, dobije dodatnu karakteristiku, npr. *Datum početka* (kao što je prikazano na Slici 19), DV model dokazuje svoju

proširivost na taj način što se na postojeći Link L_RadnoMesto jednostavno doda Satelit *S_RadnoMesto* sa pripadajućim atributom.



Slika 19. Transformacija izvora (dodavanje atributa) u Data Vault model

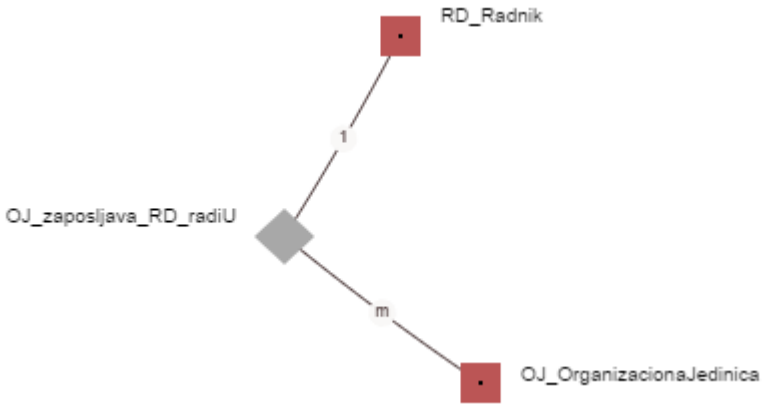
Čak i kasnija proširenja (pored već opisanog dodavanja atributa), kao što je vezivanje izvornog koncepta Radno mesto za druge koncepte izvornog modela neće usloviti promenu postojeće strukture u DV modelu, jer je dozvoljena veza dva Linka.

Međutim, iako na fizičkom nivou u prikazanim slučajevima DV model ne trpi nikakve promene, implicitno se menja semantika izvornog sistema tako što se jak objekat (u ovom slučaju agregacija Radno mesto) u DV modelu predstavlja kao Link, odnosno događaj ili veza.

Uz prethodni, DV model iskazuje i sledeći nedostatak, da reverzibilan postupak, tj. dobijanje strukture ciljnog modela iz postojećeg DV modela nije moguće, jer je DV model semantički siromašniji od izvornih modela, i prilikom transformacije dolazi do gubitka informacija o izvornom modelu. Korišćenjem prethodnih primera, nije moguće utvrditi u koji od prva dva izvorna modela bi se DV model reverzibilno preslikao.

Anchor Model. Korišćenjem istog primera promene strukture za Data Vault model, razmotriće se prilagodljivost Anchor modela na prikazane promene izvora podataka. Svi modeli su realizovani korišćenjem originalnog alata za izradu Anchor modela [88].

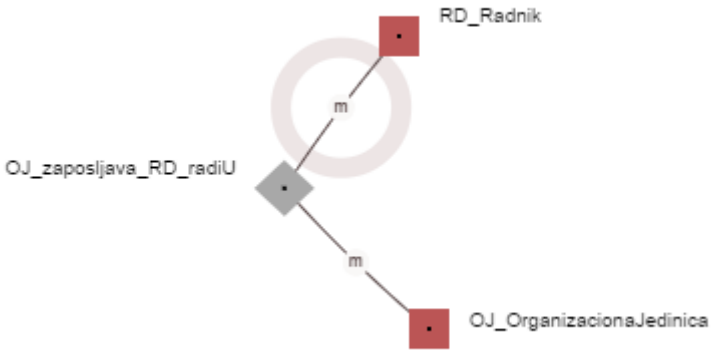
Kao što je prikazano na Slici 20, inicijalni model se sastoji od Sidra radnik (**RD_Radnik**) koji može da bude zaposlen u jednoj i samo jednoj organizacionoj jedinici (**OJ_OrganizacionaJedinica**).



Slika 20. Realizacija izvora (1,M veze) u Anchor modelu

Veza između dva Sidra (*OJ_zapošljava_RD_radiU*) se imenuje implicitno na osnovu naziva pripadajućih rola. Nakon transformacije, Veza prerasta u tabelu čiji je primarni ključ, ključ koncepta čija je gornja granica kardinalnosti preslikavanja 1.

Ukoliko se posmatrana kardinalnost promeni, na osnovu toga da radnik može da radi na više radnih mesta (kao što je prikazano na Slici 21), jedina promena na rezultujućem modelu će biti što će i primarni ključ organizacione jedinice postati deo primarnog ključa na tabeli koja predstavlja Vezu.



Slika 21. Realizacija izvora (M,M veze) u Anchor modelu

Ukoliko u izvornom sistemu dođe do strukturne promene da radno mesto (*OJ_zapošljava_RD_radiU*), dobije dodatnu karakteristiku - *DatumPočetka*, Anchor model ne dozvoljava vezivanje atributa, jer Veza (iako realizovana kao tabela nakon transformacije) ne prerasta u jak (ili agregirani) objekat. Umesto toga, dozvoljeno je Vezu referencirati na Čvor koji je skup svih datuma početka rada na nekom radnom mestu. Opisani slučaj je prikazan na Slici 22.

Slika 22. Dodavanje atributa na vezu u Anchor modelu

Promena koja nastaje na fizičkom nivou, je da se tabela koja predstavlja Vezu proširuje za još jedan atribut (primarni ključ koncepta Čvor) koji ujedno postaje i deo složenog ključa posmatrane Veze.

Dimenzioni model iskazuje slične nedostatke kao i Normalizovan model, jer svako dodavanje atributa u izvornom modelu rezultuje u promeni strukture odgovarajuće Dimenzije u skladištu podataka. Veze u Dimenzionom modelu su stabilnije jer je ovaj model procesno orijentisan, tj. projektuje se na osnovu fundamentalnih poslovnih procesa u organizaciji. Ipak, svaki dodatni koncept u izvornom modelu koji prerasta u dimenziju rezultuje promenom Fekta na taj način što se vrši dodavanje spoljnog ključa na Fekt koji ga povezuje sa dodatom Dimenzijom.

4.5.3 Analiza vremenskog aspekta modela strukture

Definišući skladište podataka [1] kao skup podataka promenljivih u vremenu (eng. *Time-Variant*), Inmon je postavio jedan od osnovnih zahteva da skladište podataka treba da omogući praćenje prošlih, trenutnih i budućih stanja objekata i događaja, kao i trenutaka kada su se te promene dešavale [10]. Da transakcioni sistem treba da podrži više vremenskih dimenzija prvi put je obrazlagano u [92], kada je Snodgrass vremenski aspekt podelio u dve dimenzije: vreme važenja i vreme ažuriranja (eng. *Bitemporal*).

Danas je uobičajeno da se vremenski aspekt (kako je dato u [91]) tretira kroz tri dimenzije:

- Vreme važenja (eng. *Valid Time*) koje označava trenutak ili period u kojem je koncept realnog sistema (npr. objekat ili događaj) važeći tj. validan.
- Vreme ažuriranja (eng. *Transaction Time*) predstavlja period u kojem je koncept realnog sistema kreiran, menjan ili izbrisan iz skladišta, odnosno baze podataka.
- Korisničko vreme (eng. *User Defined Time*) se koristi za proizvoljne strukturne vrednosti objekata i nije od interesa za dalje razmatranje vremenskog aspekta u skladištu podataka.

Smatra se da model podataka podržava vremenski aspekt ukoliko ima ugrađene mehanizme koji omogućavaju implicitno korišćenje vremena važenja i vremena ažuriranja [87].

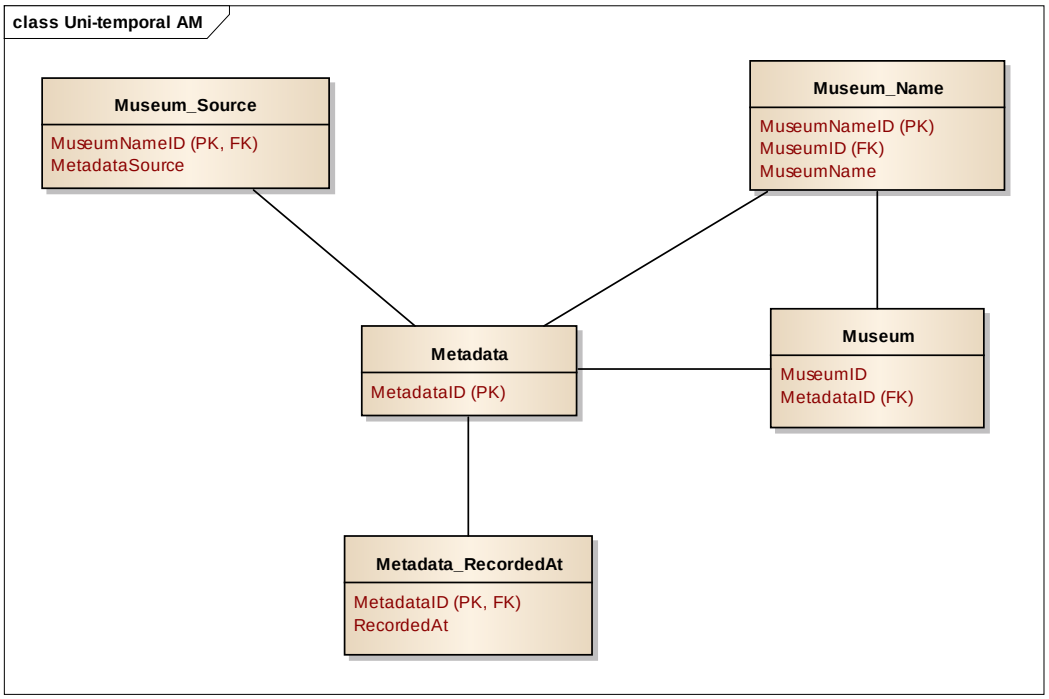
U ovom odeljku se razmatra da li postoji mogućnost i na koji način postojeći modeli strukture skladišta podataka prate istoriju životnog ciklusa objekata realnog sistema.

Normalizovan model ne poseduje nikakvu ugrađenu podršku vremenskom aspektu, već se primena vremenskih koncepata koristi isključivo na osnovu izbora projektanta skladišta podataka [87], dakle na fizičkom nivou.

Data Vault model poseduje implicitno *vreme ažuriranja* na svim konceptima, Habu, Linku i Satelitu (eng. *Load Start & End Date*). Kako ovaj model pretpostavlja da se struktura objekata menja vremenom na Satelitu se mogu voditi promene vrednosti koje nastaju tokom životnog ciklusa objekta. Na ovaj način je *vreme važenja* primenljivo za strukturu objekta, ali ne i za sam objekat ili vezu koju on ostvaruje. Usled toga, a zbog pretpostavke nepromenljivosti poslovnog identifikatora objekta, Data Vault model ne poseduje predefinisan način održavanja vremena ažuriranja za poslovni identifikator. Jedan od načina da se ovakva situacija prevaziđe je korišćenje Linka ekvivalentnosti (eng. *Same-As Link*) kojim se dva Haba sa različitim poslovnim identifikatorima proglašavaju identičnim [6].

Anchor Model podržava da se u fazi projektovanja određeni atributi i veze predstave promenljivim kategorijama (eng. *Historized*) i to samo za *vreme važenja*. Kada se koristi ova opcija, svakom odabranom konceptu se u strukturi dodaje vreme promene (eng. *ValidFrom*) [94] kao atribut u koji se upisuje vreme početka važenja vrednosti (npr. zvanični alat [87] za vreme važenja generiše *ChangedAt* atribut). Implicitno se za kraj vremena važenja vrednosti atributa prethodne vrednosti vodi vrednost – početka važenja nove vrednosti. Sidro i čvor nisu predviđeni da mogu da budu promenljivi koncepti u vremenu.

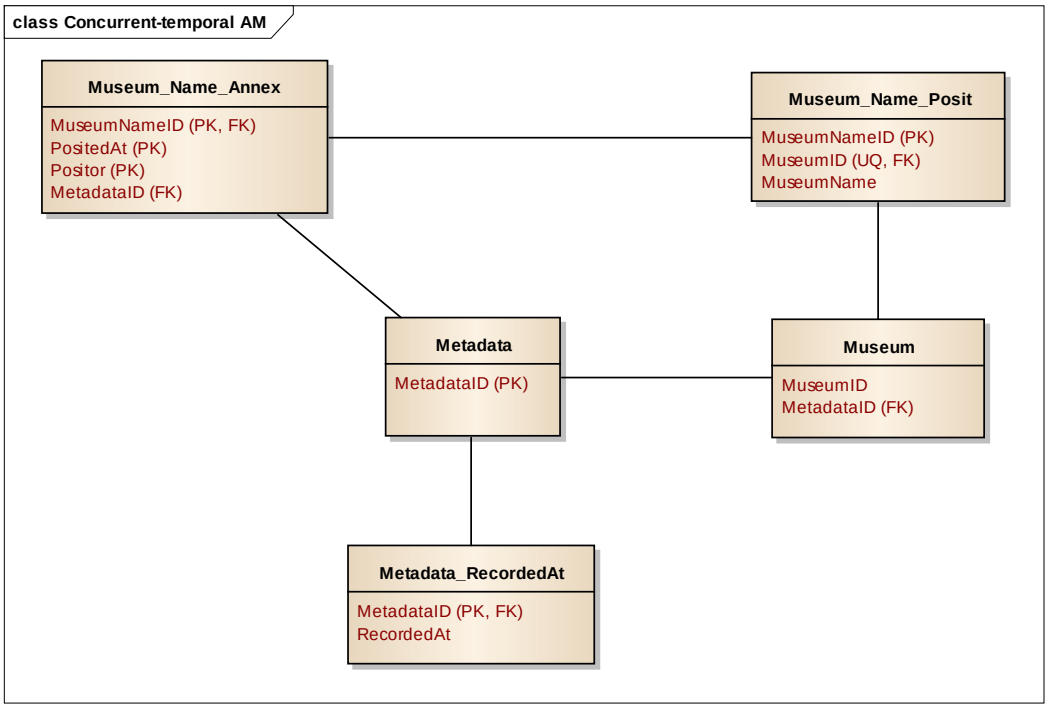
Takođe, ukoliko se pri modelovanju koristi opcija nepromenljivosti (eng. *Static*), gubi se mogućnost praćenja promene vrednosti za ove atribute ili veze. Navedeni razlog za to je da se ovakva opcija koristi za vrednosti koje su konstantne, kao što je Datum rođenja. Međutim, ne uzima se u obzir da je moguće da je vrednost ovog atributa u izvoru pogrešno upisana, ili da postoji više izvora podataka u kojima se za ovakav atribut čuvaju različite vrednosti.



Slika 23. Metapodaci - jedna vremenska dimenzija

Anchor model *vreme ažuriranja* (eng. *RecordedAt*) ne vodi kroz strukturu koncepata koji predstavljaju objekte realnog sistema već kroz odgovarajuće metapodatke u posebnim tabelama [36], kao što je prikazano na Slici 23. I u slučaju organizacije metapodataka Anchor Model se pridržava principa koji vladaju u modelovanju osnovnih koncepata, odnosno za svaki atribut metapodatka se kreira odgovarajuća tabela.

Nakon formalizacije u [36], zaključeno je da vremenske dimenzije – vreme važenja i vreme ažuriranja nisu dovoljne za razvoj i održavanje skladišta podataka, te je ovaj model 2011. godine [95] proširen i za *vreme ispravke* (eng. *Positing Time*) koje se beleži preko tzv. aneks (eng. *Annex*) tabele koja poseduje odgovarajući atribut (eng. *PositedAt*) u koji se ovo vreme upisuje pri svakoj izmeni vrednosti atributa, kao što je prikazano na Slici 24. Model prikazan na slici još uvek nije postao sastavni deo zvaničnog alata za Anchor Model [88], niti je formalno deo Anchor modela, već još uvek egzistira kao predloga autora ovog modela.



Slika 24. Metapodaci - više vremenskih dimenzija

Kao što je već pokazano, Dimenzioni model realizuje praćenje *vremena važenja* koncepata realnog sistema korišćenjem SCD tipova, pri čemu SCD tip 1 ili SCD tip 3 ne omogućavaju praćenje vremena važenja objekata. Jedan od problema koji se odnosi na praćenje promena na dimenzijama je sam izbor SCD tipa promene, jer kasniji prelaz sa jednog na drugi SCD tip neizbežno vodi promeni strukture dimenzije. Pored toga, ovaj model, ukoliko se koristi SCD tip 2 iskazuje nemogućnost praćenja promene poslovnog identifikatora objekta, jer je on osnov za praćenje promena ostalih atributa. Jedan od načina za prevazilaženje ovog problema je premošćavanje (eng. *Bridge Table*) koji se po svojoj prirodi ne uklapa u standardne koncepte dimenzionog modela.

Što se tiče druge vremenske dimenzije, *vreme ažuriranja* se u Dimenzionom modelu ne vodi na nivou koncepata skladišta podataka koji predstavljaju poslovne objekte, već u žurnal (log) tabelama.

4.5.4 Analiza kompletnosti i pratljivosti podataka

Prema već pominjanoj definiciji skladišta podataka koju je dao Inmon [1], jedna od osnovnih karakteristika DW je nepromenljivost (eng. *Non-Volatility*) podataka koja podrazumeva da svi podaci koji se integrišu u skladištu podataka u njemu i ostanu – nepromenjeni, što praktično znači da upit postavljen u bilo kom trenutku daje isti, konzistentan rezultat [56].

Zahtevi koji se odnose na kompletnost i pratljivost podataka su u direktnoj koliziji sa konceptom *jedinstvene istine* (eng. *Single Version of Truth*) koji predstavlja postupak gde se unapred pripremaju i filtriraju podaci koji će biti skladišteni u DW. Jedna verzija istine je dakle rezultat integracije podataka u skladištu podataka iz više različitih izvora radi stvaranja osnove za analizu podataka i kako bi se izbegla redundansa [93]. Ipak, da bi se dostigao cilj jedinstvene istine pri projektovanju skladišta podataka, između podataka iz više različitih izvora koji se odnose na isti koncept realnog sistema, na osnovu različitih kriterijuma, najčešće pri ETL procesu, bira se jedan koji će da bude istinit, tj. zabeležen u skladištu podataka. Pri tome se svi podaci koji ne odgovaraju 'istini' odbacuju, tj. ne postanu deo skladišta podataka. Na taj način skladište podataka ima samo deo podataka izvora, pri čemu je nemoguće analizirati sve vrednosti koje postoje u izvorima.

Stoga je uveden koncept *više verzija istine* (eng. *Single Version of Facts*) [3] koji pretpostavlja da skladište podataka sadrži sveukupne podatke izvora i da na taj način omogućava drugačije poglede različitim korisnicima shodno njihovim potrebama. Praktično, zagovara se ELT proces, kada se transformacija podataka vrši posle punjenja *Sloja skladišta podataka*, a ne između *Sloja pripreme podataka* i *Sloja skladišta podataka*, kao što je to uobičajeno. Na ovaj način skladište podataka sadrži sve podatke svih izvora u svim periodima. U ovom slučaju se podaci transformišu, odnosno prilagođavaju tek na zahtev krajnjeg korisnika [3].

U ovom odeljku će se razmatrati prikazani modeli strukture skladišta podataka sa stanovišta kompletnosti prenešenih podataka iz izvora, kao i sa stanovišta mogućnosti praćenja unešenih podataka do izvora u kojima su nastali.

Slično kao u slučaju vremenskog apekta, Normalizovan model ne poseduje implicitno koncepte koji omogućavaju pratljivost podataka ili istovremeno čuvanje podataka iz više različitih izvora. Pratljivost podataka, na primer, može da se obezbedi dodatnim metapodacima nad svakom n-torkom u tabelama u slučaju primene 'jedinstvene istine'. Međutim, ukoliko se ispunjava zahtev više verzija istine, Normalizovan model iskazuje određene nedostatke. Jedini način da se više izvora skladišti u istom modelu bi značio po jednu n-torku za isto toliko izvora u kojima se isti koncept realnog sistema javlja. Ovo stvara problem povezivanja n-torki, odnosno korišćenja dodatnih koncepata koji bi povezali više n-torki u skladištu podataka koji predstavljaju jedan (isti) objekat realnog sistema. Takođe, na ovaj način se uvodi redundansa (u n-torkama koje predstavljaju isti objekat realnog sistema postoje atributi koji imaju iste vrednosti), čime se ukida jedna od dobrih osobina Normalizovanog modela.

Data Vault model podržava koncepte koji omogućavaju pratljivost podataka, tako što obavezno beleži iz kog izvora je Hab prvi put upisan. Takođe, Satelit i Link poseduju ugrađene atribute koji beleže, pored vrednosti, i iz kojeg izvora je vrednost upisana (atribut *RecordSource*) [27]. Osnovni nedostatak ovog modela je što se u strukturi Haba nalazi poslovni ključ, koji je nepromenljive (ili retko promenljive) vrednosti. Pored toga, kao što je ranije pokazano, pored vrednosti, može doći do promene i strukture poslovnog ključa objekta realnog sistema. Data Vault model ove slučajeve razrešava uvođenjem novog Haba koji se 'Same As' Linkom povezuje sa prvobitnim Habom. Na ovaj način egzistira najmanje dva Haba koji imaju istovetne atribute i iste veze sa ostatkom sistema.

Pored toga, Data Vault omogućava praćenje više izvora tako što se u Satelitu beleži vrednost atributa zajedno sa izvorom podataka.

Anchor Model obezbeđuje pratljivost preko koncepta *metapodatka* kao što je prikazano na Slikama 23 i 24. Naime, svaki promenljivi koncept modela (ranije je napomenuto da su nepromenljivi Sidro i Čvor) može imati vezu ka metapodacima koji beleže *izvor podataka.* Iako je druga verzija nastala kao pokušaj da se omogući preciznije beleženje metapodataka, tj. izvora i dimenzije vremena, u ovoj verziji nije

moguće zabeležiti da u istom trenutku važi ista vrednost za isti objekat u dva izvora. Razlog je što na implementacionom nivou postoji ograničenje (eng. *Unique Constraint*) nad grupom atributa: identifikator atributa, vreme važenja i vrednost atributa. Na taj način će se u skladištu podataka uvesti samo vrednost iz prvog izvora iz kojeg bude upisana. Sa druge strane, novija verzija AM omogućava preciznije beleženje izvora podataka, tako što je moguće voditi sve promene atributa i veza (kada je nastala, ko je promenu uneo, stepen pouzdanosti unosa i slično).

Dimenzioni model ne poseduje ugrađeni mehanizam kojim bi se obezbedila pratljivost podataka do samog izvora. Takođe, u dimenzionom modelu nije moguće voditi više vrednosti iz više različitih izvora koje su važeće u istom trenutku.

4.5.5 Presek modela strukture skladišta podataka na osnovu analize

Kao što je prikazano u Tabeli 2, analizirani modeli ne ispunjavaju u potpunosti zahteve koje DW modeli podataka treba da zadovolje.

Tabela 2. Poređenje DW modela podataka

	Normalizovan model	**Data Vault**	**Anchor Model**	**Dimenzioni model**
Ugrađena semantika	Ne	Da	Da	Da
Otpornost na promene	Ne	Delimično	Delimično	Ne
Vremenski aspekt	Ne	Delimično	Delimično	Da
Kompletnost	Ne	Da	Delimično	Ne
Pratljivost	Ne	Da	Da	Ne

Normalizovan model iskazuje slabosti za sve prikazane kriterijume DW modela podataka.

Data Vault pokazuje nedostatke pri složenijim izmenama strukture izvora i zbog „čvrste“ veze između poslovnog identifikatora objekta i samog objekta. Pored toga, Data Vault ne poseduje eksplicitno koncept *vreme važenja*.

Anchor Model omogućava neistorijsko praćenje objekata i pokazuje nedostatke u ispunjavanju kriterijuma kompletnosti zbog podrške konceptu *'jedinstvene istine'*.

Iako je Dimenzioni Model verovatno najpogodniji za pripremu i analizu podataka, otežana otpornost ovog modela na promene i nemogućnost praćenja više vrednosti različitih izvora podataka ovaj model udaljava od kandidata za EDW model.

U poglavlju *Fizički Data Vault (Model Domen / Preslikavanje)* je data specifikacija projektovanog fizičkog modela DW koji u celini ispunjava postavljene zahteve i koji objedinjuje dobre osobine prikazanih modela.

5 KONCEPTUALNI DATA VAULT

Konceptualni model predstavlja specifikaciju realnog sistema na takvom nivou apstrakcije da ne uključuje sve detalje realnog sistema, niti poseduje detalje vezane za tehnološku platformu. Takođe, konceptualni modeli su bliži korisnikovom poimanju realnog sistema u odnosu na sve druge modele koji se koriste u projektovanju i implementaciji informacionih sistema.

Jezik kojim se definiše konceptualni model treba da bude dovoljno opšt da može da opiše različite klase realnih sistema, ali i u odgovarajućoj meri specifičan, jer koncepti i semantika koju jezik poseduje treba da usmeravaju i olakšavaju definisanje modela.

Često se ukazuje na razlike koje postoje u konceptualnim modelima za transakcione sisteme i za skladišta podataka, iako postoji niz sličnosti u obe vrste konceptualnih modela koje su prvenstveno vezane za opis objekata, njihovu strukturu i međusobne veze.

Razlike koje postoje nastaju iz same definicije i prirode skladišta podataka. Naime, kao što je ranije primećeno, skladište podataka predstavlja skup stanja nekog sistema u određenom vremenskom periodu. Stoga, jezik koji definiše DW konceptualni model implicitno treba da podrži vremenski aspekt životnog ciklusa objekata i događaja realnog sistema.

Pored toga, da bi skladište podataka ispunilo svoju integrativnu funkciju, jezik kojim se opisuje treba da ima koncepte dovoljno opšte kako bi bilo moguće obuhvatanje različitih jezika za opis i realizaciju koji se koriste u izvorima podataka.

Kao što je pokazano u poglavlju *Konceptualni modeli skladišta podataka*, svi predstavljeni jezici za konceptualno modelovanje su orijentisani isključivo na *Sloj analize podataka* (tj. odnose se na Dimenzioni model), iako bi konceptualni model skladišta podataka trebalo ili da opiše ili da omogući odgovarajuće transformacije u

sve slojeve skladišta podataka koji su predstavljeni u odeljku *Komponente logičke arhitekture*.

Takođe, modeli prikazani u poglavlju *Modeli strukture skladišta podataka* nisu odgovarajući kandidati za DW konceptualni model zbog nedostataka predstavljenih u odeljku *Analiza modela strukture skladišta podataka* i što, dodatno, iskazuju značajnu povezanost sa ciljnim fizičkim modelom. Praktično, modeli definisani ovim jezicima (tačnije je zvati ih logičkim modelima pre nego konceptualnim) imaju 1:1 preslikavanje sa krajnjim fizičkim modelom skladišta podataka.

5.1 Konceptualni Data Vault Model – C-DV

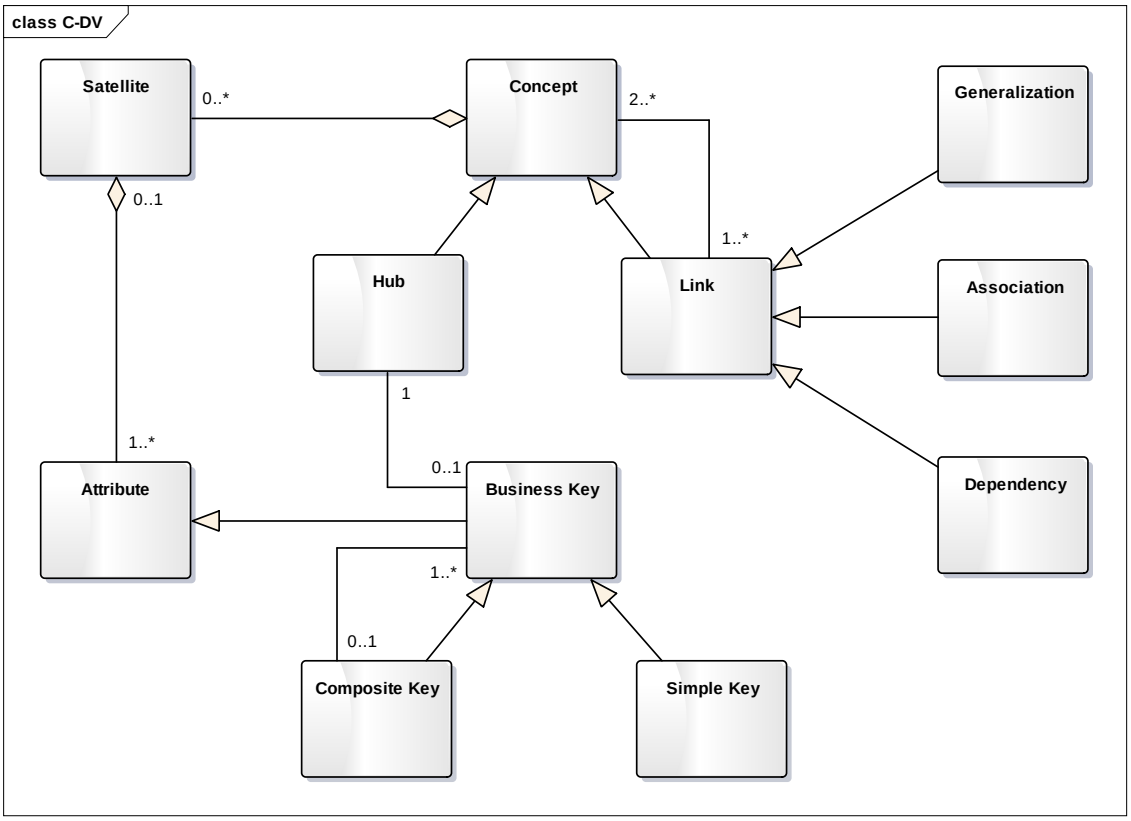
Konceptualni Data Vault (eng. *Conceptual Data Vault – C-DV*) [7] je metamodel za opis DW modela čija je namena dvojaka. Sa jedne strane, C-DV predstavlja okvir za konceptualno modelovanje skladišta podataka proširenjem osnovnog Data Vault [6] modela tako što ga formalizuje i ekstenduje konceptima koji omogućavaju praćenje strukturnih i promena na meta nivou na modelima izvora podataka.

Sa druge strane, C-DV je definisan na taj način da može da izrazi i obuhvati semantiku složenih modela izvora podataka i da bude međumodel automatskih transformacija iz izvornih u ciljne modele skladišta podataka. Takođe, moguće je vršiti i inverznu transformaciju iz izvora podataka opisanih u relacionom modelu. Tako definisan C-DV model predstavlja specifikaciju raznorodnih izvora podataka jedinstvenim jezikom čime se omogućava automatska transformacija izvora podataka do nivoa izvršivog objedinjenog skladišta podataka.

U ovom poglavlju će se definisati odgovarajući metamodel (C-DV) koji može da opiše semantiku raznorodnih modela izvora podataka iskazanu kroz različite metamodele (UML Dijagram klasa, Model objekti-veze, Relacioni model...) koji je zasnovan na postojećim i proširenim konceptima empirijskog Data Vault modela.

Kao što je prikazano na Slici 25, osnovni koncept C-DV modela je **Hub** koji je predstava realnih ili apstraktnih objekata realnog sistema koji mogu biti jedinstveno identifikovani. Svaki Hab je identifikovan jedinstvenim **Business Key** identifikatorom, koji može biti prost ili složen (**Simple Key** ili **Composite Key**) koji

su na modelu predstavljeni kao specijalizacija poslovnog ključa. Za razliku od empirijskog Data Vault modela, poslovni ključ je izdvojen iz Hab koncepta zbog postojanja jezika za opis koji ne ograničavaju model postojanjem identifikatora objekta.



Slika 25. C-DV metamodel u UML notaciji

Habovi mogu biti povezani preko događaja u sistemu, strukturnih i drugih veza što se ostvaruje korišćenjem **Link** koncepta. Link se specijalizuje u više vrsta veza koje mogu da izraze moguće veze između objekata realnog sistema, i to:

- **Dependancy** je veza koja predstavlja vezu posedovanja, najpribližnije slabom objektu iz MOV ili kompozicije iz UML jezika (eng. *„has a“ relationship*).
- **Generalization** predstavlja apstrakciju generalizacije/specijalizacije (eng. *“is a” relationship*)

- **Association** obuhvata sve ostale binarne ili n-arne veze. Postojanje n-arnih veza je definisano kako bi se podržali svi jezici za opis sistema koji ne predviđaju isključivo binarne veze.

Struktura Haba se definiše preko jednog ili više **Satellite** koncepta. Satelit je domen, moguće složen i predstavlja jednu ili više karakteristika Haba. Svaka karakteristika je opisana konceptom **Attribute**. Atribut je, pored toga što je domen koji definiše skup vrednosti, ujedno i nadtip poslovnog ključa, koji je i sam atribut, ali sa dodatnom semantikom identifikatora nekog objekta.

Iako većina jezika za opis definiše strukturu objekta isključivo preko atributa, u C-DV modelu je zadržan i koncept Satelita i Atributa, kako bi model mogao da podrži izvore ili modele koji su definisani u empirijskom Data Vault modelu.

Takođe, i Link koncept može imati strukturne karakteristike (slično konceptima 'asocijacija kao klasa' iz UML, odnosno 'agregacija' iz MOV modela) i stoga je uveden koncept **Concept** kojem se mogu pridružiti Sateliti i iz kojeg su izvedeni Hab i Link. Na taj način se definiše struktura za oba koncepta.

Celokupno skladište podataka treba da bude opisano u Rečniku skladišta podataka, kao što je opisano u odeljku *Komponente logičke arhitekture skladišta podataka*. Skup svih C-DV modela predstavlja osnovnu komponentu Rečnika skladišta podataka. Ovakav repozitorijum modela praktično opisuje celokupno skladište podataka sa istorijatom promene strukture izvornih i DW modela. Na taj način je moguće pratiti sve strukturne promene koje su se desile u skladištu podataka.

Pored repozitorijuma modela, Rečnik skladišta podataka treba da poseduje još najmanje dva modela koji su neophodni za ispravno funkcionisanje repozitorijuma modela:

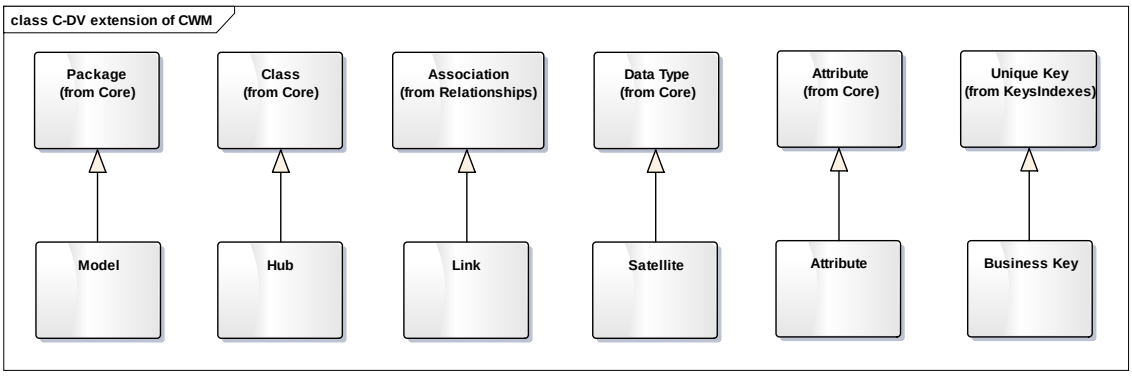
- Model verzionisanja (eng. *Versioning Model*) koji prati verzije modela i razlike koje u njima postoje i koji podržava istorijsku funkciju skladišta podataka, i

- Model preslikavanja (eng. *Weaving Model*) je model koji održava vezu između koncepata koji opisuju isti objekat realnog sistema, ali u dva različita izvora. Ovaj model podržava integrativnu funkciju skladišta podataka. Kako je ovim modelom mapiranje definisano kao preslikavanje opštih koncepata, može služiti za definisanje preslikavanja i nad M^2 i nad M^1 modelima.

Kako postoji više standardnih rešenja za modele verzionisanja i preslikavanja, definisanje ovih modela neće biti predmet istraživanja ovog rada.

5.2 C-DV u okviru MDA arhitekture

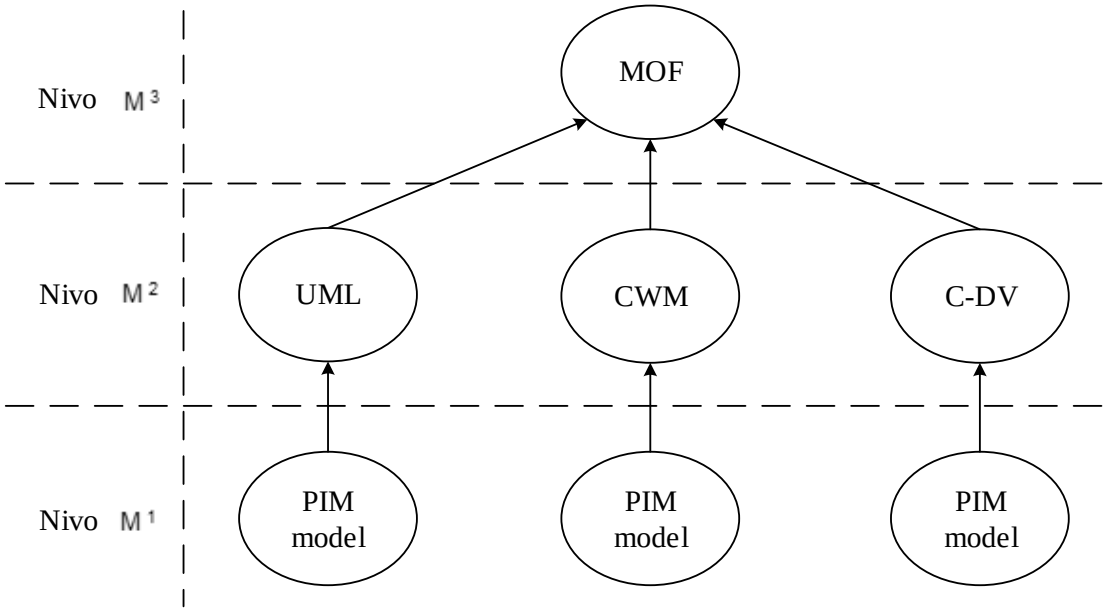
Da bi se omogućila interoperabilnost C-DV metamodela u okviru platformski nezavisnih okruženja, C-DV koncepti mogu da budu definisani i kao ekstenzija CWM metamodela, kao što je prikazano na Slici 26.



Slika 26. C-DV koncepti kao ekstenzija CWM

Svaki pojedinačni koncept se izvodi iz odgovarajućeg CWM koncepta. Kao što je prikazano, C-DV Model je *Package* koji predstavlja domensko znanje o nekom sistemu ili njegovom delu. Hab je skup objekata specijalizovan iz *Class* koncepta. Link je *Association*, dok su pripadajući atributi Haba i Linka definisani kao *Attribute.* Satelit je opisan kao kolekcija atributa i stoga je specijalizovan iz *Data Types* koncepta. Na kraju, identifikator haba je izveden iz koncepta *Unique Key*.

Na taj način C-DV metamodel je pozicioniran kao M^2 model u okviru MDA arhitekture kao što je prikazano na Slici 27.



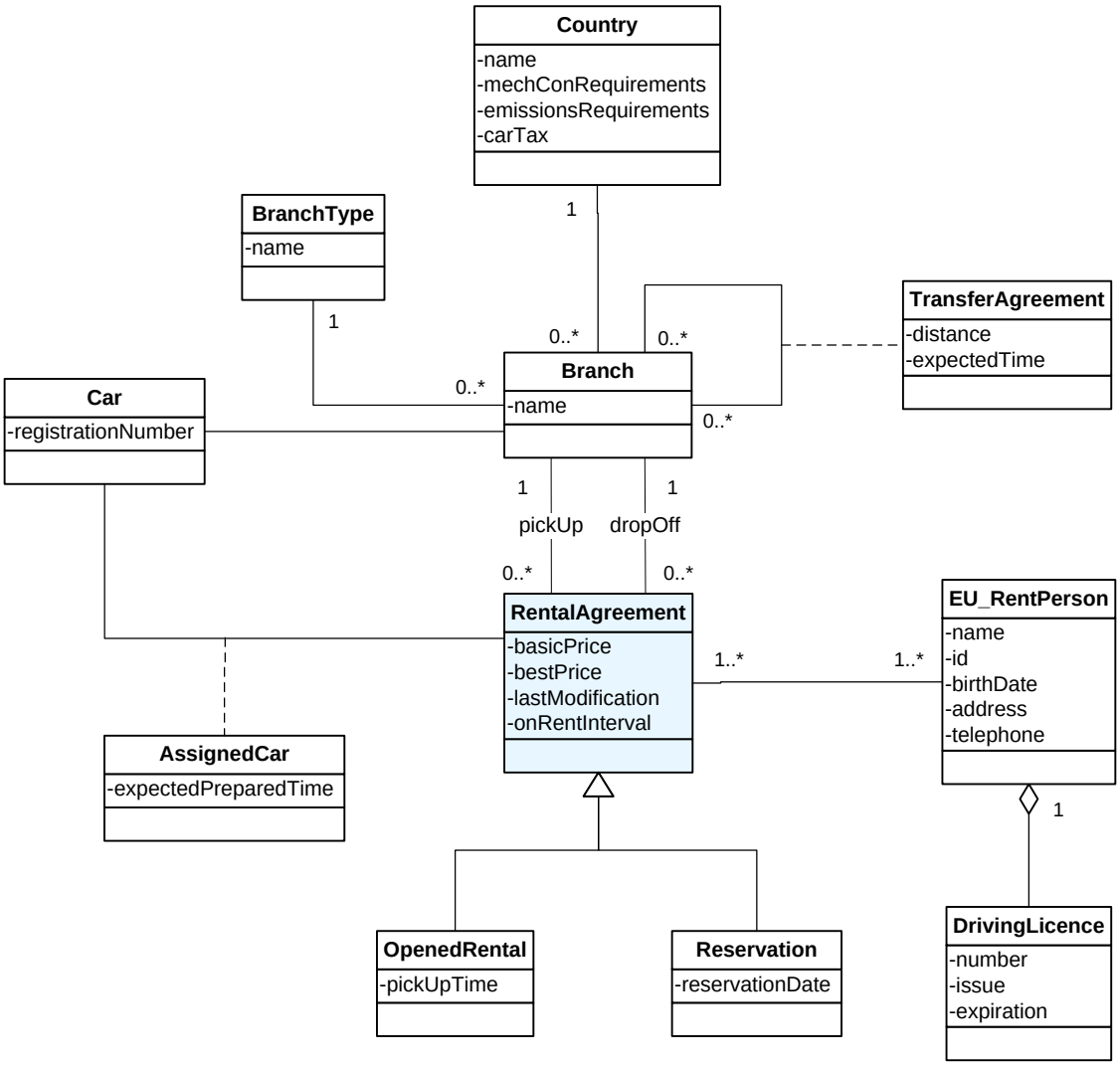
Slika 27. C-DV u okviru MDA arhitekture

Kao što je prikazano, C-DV model se nalazi na istom nivou MDA arhitekture kao i CWM i modeli koji ga čine (Objektni, Relacioni, Multidimenzioni) ili su pak njegove ekstenzije (model objekti veze). Takvim pozicioniranjem C-DV modela omogućeno je da se na standardan način izvrši primena odgovarajućih transformacija, korišćenjem XMI jezika definisanog za razmenu metapodataka.

5.3 Pravila transformacije UML u C-DV

U ovom odeljku se definiše skup C-DV pravila transformacije iz izvora podataka u ciljni C-DV model. Prikazani skup transformacionih paterna obrađuje elementarne strukture i konstrukcije koje se javljaju u modelovanju i definišu rezultujući ciljni podmodel opisan C-DV modelom.

Ilustracija transformacionih paterna će se obaviti upotrebom '*EU Rent Car*' modela koji je opšte poznat i često korišćen za različite studije slučajeva pretežno u akademskoj zajednici. Kao što je prikazano na Slici 28, model predstavlja strukturu

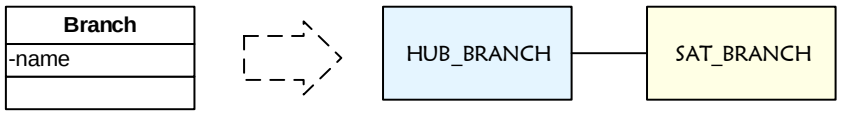


Slika 28. 'EU Rent Car' podmodel

organizacije za iznajmljivanje automobila sa filijalama u više zemalja i koja pruža uobičajene usluge rentiranja. Originalna i potpuna specifikacija se može naći na [107]. Model je neznatno izmenjen u odnosu na originalni, kako bi zadovoljio neophodne zahteve za prikaz svih transformacionih paterna. Izvorni model je definisan u UML Dijagram klasa modelu, i stoga bi prikazani transformacioni paterni trebali biti korišćeni za transformacije UML → C-DV.

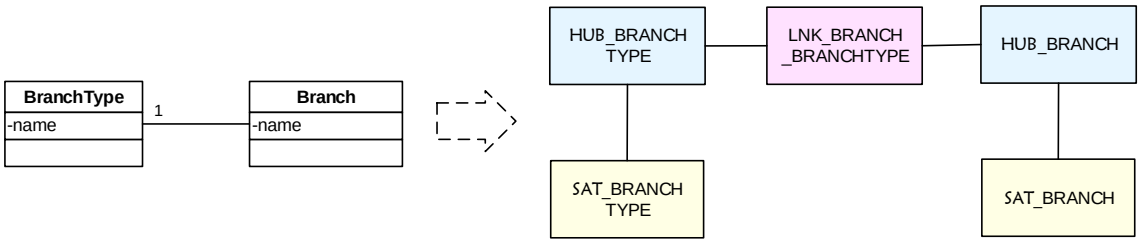
Pored toga, u sledećem odeljku su dati transformacioni paterni za MOV → C-DV preslikavanje, čime će biti pokazano da nezavisno od semantike modela kojim su opisani izvorni, rezultujući model može da bude uniformno izgrađen.

Objekat – Atribut patern: Svaki objekat se transformiše u Hab. Identifikator objekta prerasta u Poslovni ključ haba. Atributi objekta se izdvajaju u Satelit koji je povezan sa habom, kao što je prikazano na Slici 29.



Slika 29. Objekat – Atribut patern UML

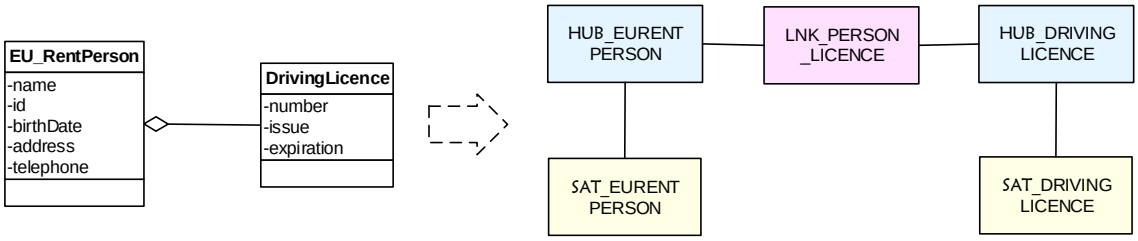
Asocijacija patern: Objekti koji su u vezi se transformišu u habove i pripadajuće satelite, kao u prethodnom primeru. Asocijacije se transformišu u koncept koji predstavlja vezu, odnosno Link bez obzira na kardinalnosti u izvornom modelu. Praktično, asocijacija se uvek realizuje kao Link koji predstavlja realizaciju M:M veze, čak iako je u izvornom modelu gornja granica kardinalnosti 1. Jedna od varijacija Veza paterna je prikazana na Slici 30.



Slika 30. Asocijacija patern UML

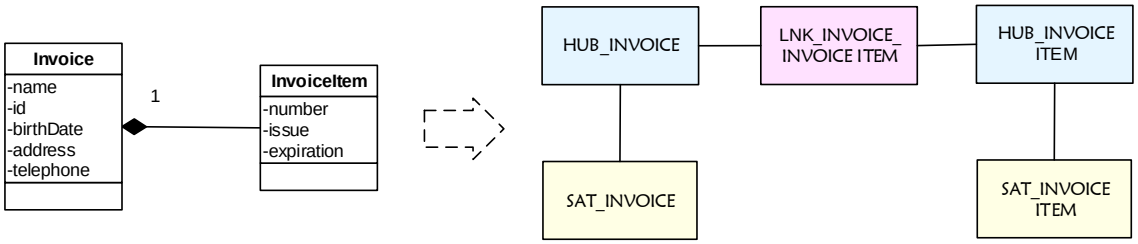
Postavljanje Link koncepta između dva povezana entiteta omogućava fleksibilnost modela na taj način što ukida strukturnu vezu između entiteta koja postoji na izvornom modelu. Ovaj princip će biti zadržan i na ostalim paternima, kao što će biti ilustrovano u sledećim primerima.

Agregacija patern: Kao i u prethodnim primerima, objekti se transformišu korišćenjem *Objekat – Atribut paterna*, dok se veza agregacija transformiše u odgovarajući Link, kao što je prikazano na Slici 31.



Slika 31. Agregacija patern UML

Kompozicija patern: U slučaju kompozitne veze, objekti se transformišu korišćenjem *Objekat – Atribut* paterna, dok se sama kompozicija transformiše u odgovarajući Link, kao što je prikazano na Slici 32.

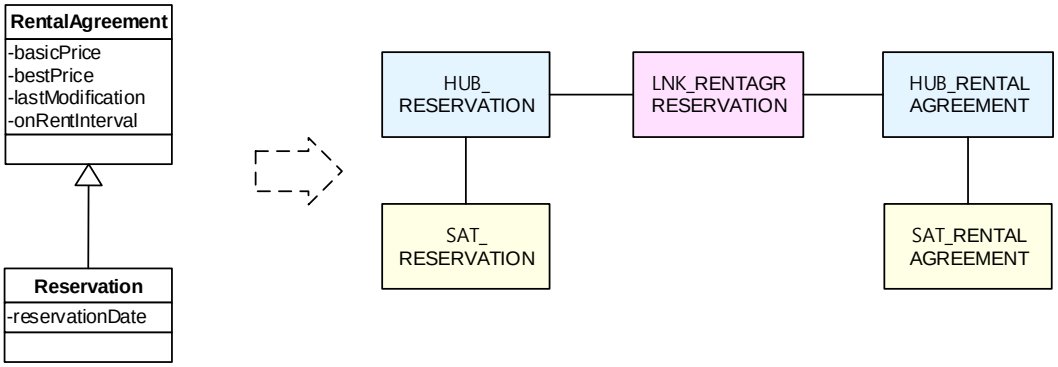


Slika 32. Kompozicija patern UML

Kao što je primetno, sva tri prethodna primera imaju istovetan rezultujući model, bez obzira koja veza između objekata je uspostavljena na izvornom modelu. Za svaki od ovih primera se koristi odgovarajuća vrsta *Dependancy* linka, kako bi se sačuvala semantika izvornog modela.

Osnovna namera ovako definisanih paterna je da se izvrši strukturno razdvajanje objekata koji su u vezi, tj. da se postigne da nijedan objekat na rezultujućem modelu ne sadrži delove ili celinu drugog objekta.

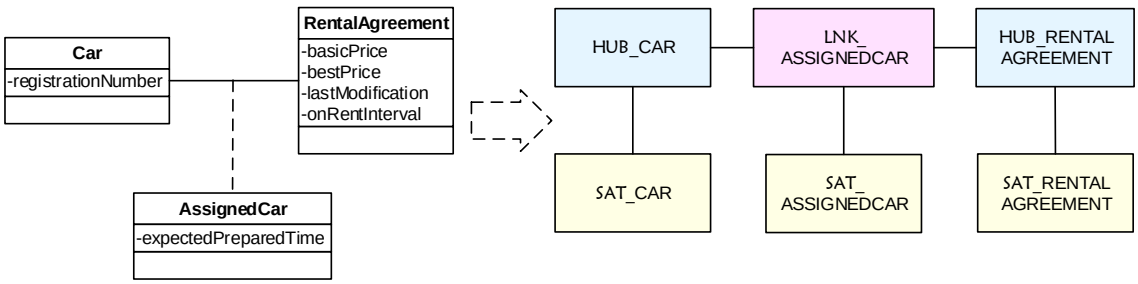
Generalizacija patern: Apstrakcija generalizacije/specijalizacije na izvornom modelu se transformiše u odgovarajući Link koncept, dok se objekti transformišu korišćenjem paterna *Objekat – Atribut*.



Slika 33. Generalizacija patern UML

Kao što je prikazano na Slici 33, i ova (tzv. "is a") veza rezultira istim C-DV modelom kao i prethodne veze, sa tom razlikom što se u ovom slučaju koristi *Generalization* vrsta linka.

Asocijacija kao klasa patern: U slučaju pojavljivanja koncepta 'asocijacija kao klasa' u izvornom modelu, transformacija se vrši na sledeći način: za objekte koji učestvuju u asocijaciji, koristi se patern *Objekat – Atribut*, dok se sama asocijacija transformiše u odgovarajući Link sa pripadajućim satelitima, kao što je prikazano na Slici 34.

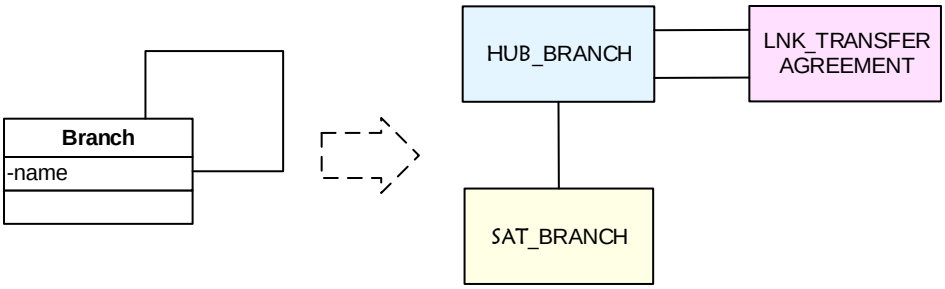


Slika 34. Asocijacija kao klasa patern UML

Kao što je pri definisanju C-DV metamodela napomenuto, dozvoljeno je da veza ima pripadajuće atribute, kako bi se omogućila transformacija različitih izvora podataka u ciljni C-DV.

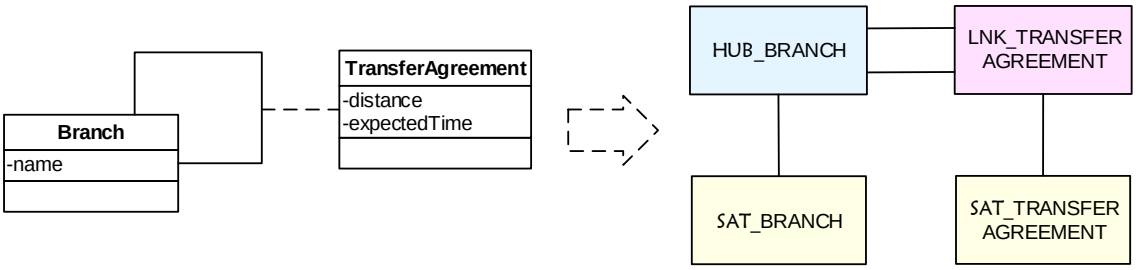
Rekurzivna asocijacija patern: Pri postojanju rekurzivne veze u izvoru podataka, transformacija se vrši na taj način što se za objekat primenjuje *Objekat – Atribut*

patern, dok se veza transformiše u Link koji je dva puta povezan sa habom, po jednom za svaku rolu veze, kao što je prikazano na Slici 35.



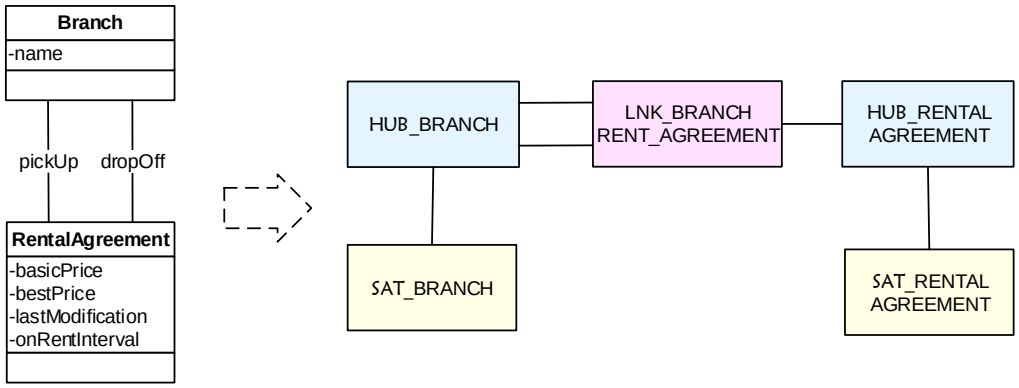
Slika 35. Rekurzivna asocijacija patern UML

Rekurzivna asocijacija kao klasa patern: Ukoliko se rekurzivna asocijacija na izvoru predstavi kao asocijacija kao klasa, tj. ukoliko asocijacija poseduje neke strukturne karakteristike, transformacija se vrši na taj način što Link iz prethodnog primera dobije pridružen satelit, kao što je prikazano na Slici 36.



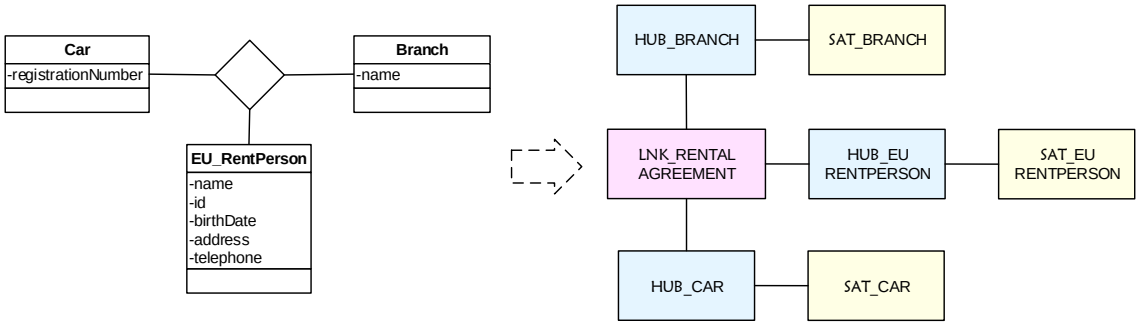
Slika 36. Rekurzivna asocijacija kao klasa patern UML

Hijerarhija patern: Iako je postavka Hijerarhija paterna semantički bliska prethodnom primeru, rezultujući C-DV model se značajno razlikuje. Naime, ukoliko se na izvornom modelu definiše tzv. sastavnica, ciljni model se realizuje na taj način što se uvodi dodatni hab (u odnosu na prethodni primer) za koji se vezuje pripadajući satelit, kao što je prikazano na Slici 37.



Slika 37. Rekurzivna asocijacija kao klasa patern UML

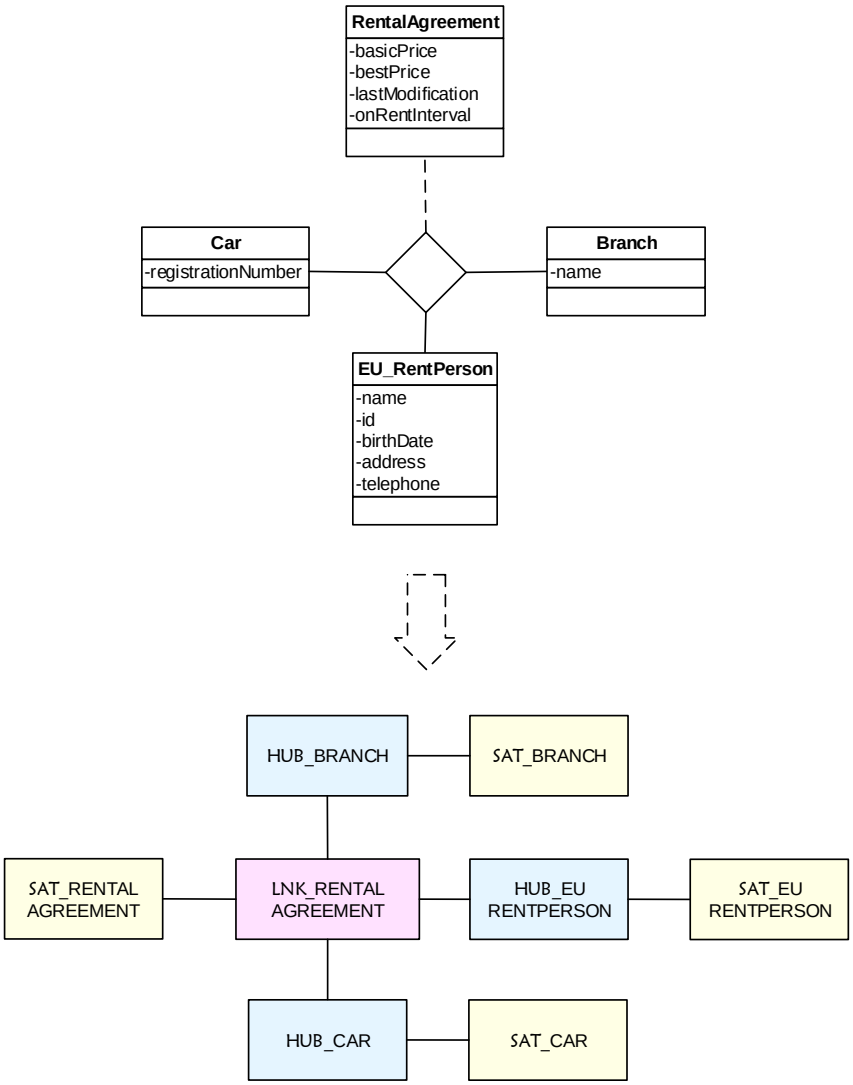
N-arna asocijacija patern: N-arna asocijacija se transformiše tako što se definiše Link koji povezuje odgovarajuće habove sa pripadajućim atributima, kao što je dato na Slici 38.



Slika 38. N-arna asocijacija patern UML

Primetno je da *N-arna asocijacija* patern i *Asocijacija* patern imaju isti rezultujući C-DV model.

N-arna asocijacija kao klasa patern: Ukoliko se izvorni UML podmodel definiše kao N-arna asocijacija kao klasa, rezultujući C-DV podmodel predstavlja kombinaciju dva paterna: *Objekat – Atribut* i *Asocijacija kao klasa*. Svaki objekat se transformiše u odgovarajući hab sa pripadajućim satelitima, dok se rezultujućem link konceptu pridružuje satelit, kao što je prikazano na Slici 39.



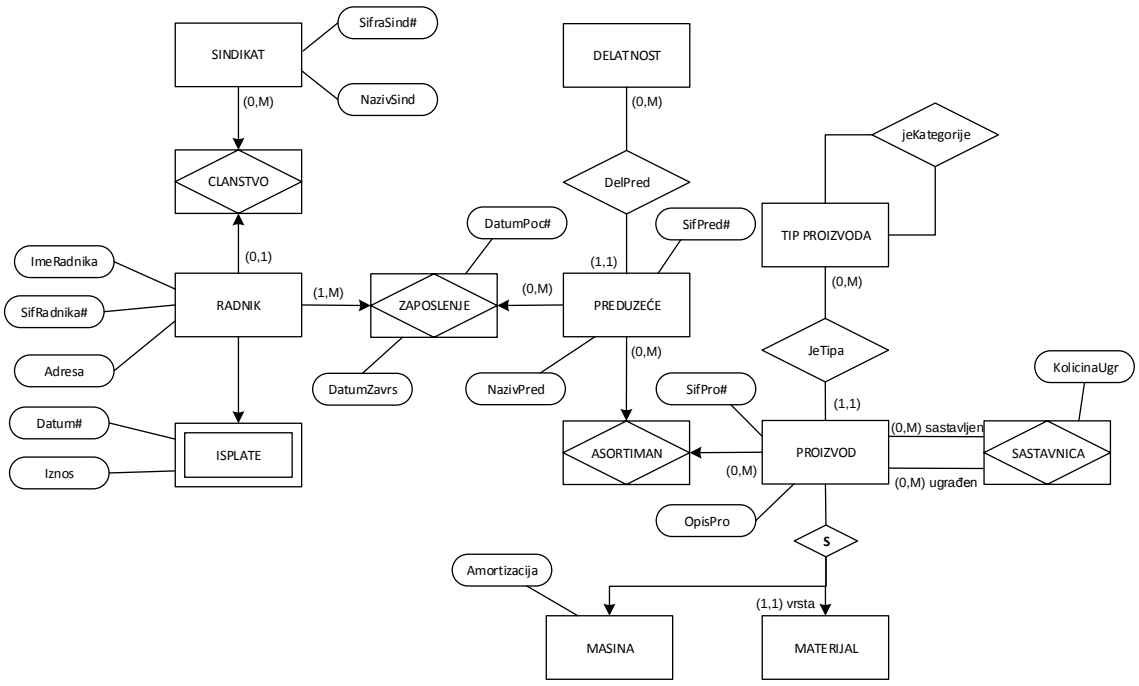
Slika 39. N-arna asocijacija kao klasa patern UML

Ilustracijom u prethodnim primerima data su pravila preslikavanja izvornog modela (opisanog UML jezikom) u ciljni C-DV. Prikazana pravila obuhvataju i obrađuju sve apstrakcije jezika izvornog modela. Na taj način (uz slična pravila transformacije za ostale jezike, npr. za MOV, kao što je dato u sledećem odeljku), omogućeno je opisivanje modelom C-DV celokupnog modela EDW na konceptualnom nivou. Da bi se to pokazalo, za opisivanje transformacionih paterna su uzeta dva semantički najbogatija jezika za opis strukture sistema.

5.4 Pravila transformacije MOV u C-DV

U ovom odeljku se definiše skup C-DV pravila transformacije iz modela koji su opisani MOV jezikom. Prikazani skup transformacionih paterna obrađuje elementarne strukture i konstrukcije koje se javljaju u modelovanju i definišu rezultujući ciljni podmodel u C-DV modelu.

Ilustracija transformacionih paterna će se obaviti upotrebom MOV modela koji je preuzet iz [83] gde se može naći originalna i potpuna specifikacija, jer je model neznatno izmenjen kako bi ispunio neophodne zahteve za ilustrovanje svih transformacionih paterna. Kao što je prikazano na Slici 40, model predstavlja strukturu neke proizvodne organizacije.

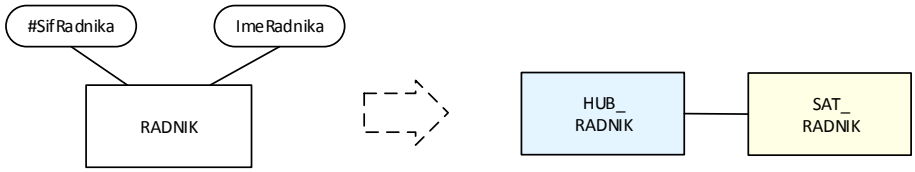


Slika 40. MOV podmodel

Izvorni model je definisan u MOV modelu, i stoga bi svi prikazani transformacioni paterni trebali biti korišćeni za transformacije MOV → C-DV.

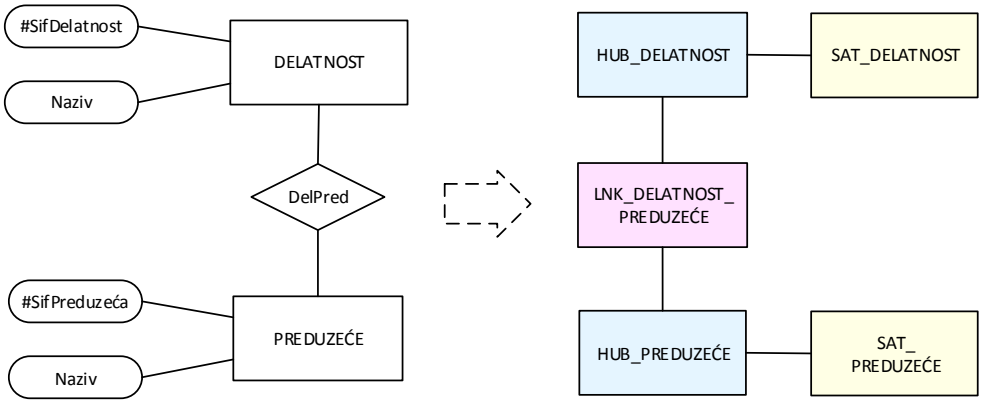
Kako bi se pokazalo da rezultujući model može da bude uniformno izgrađen, pored transformacija iz UML jezika, date su i transformacije iz MOV, kako bi se obuhvatila dva semantički najbogatija jezika za opis strukture sistema.

Objekat – Atribut patern: Svaki objekat se transformiše u Hab. Identifikator objekta prerasta u Poslovni ključ haba. Atributi objekta se izdvajaju u Satelit koji je povezan sa habom, kao što je prikazano na Slici 41.



Slika 41. Patern 'Objekat – Atribut MOV'

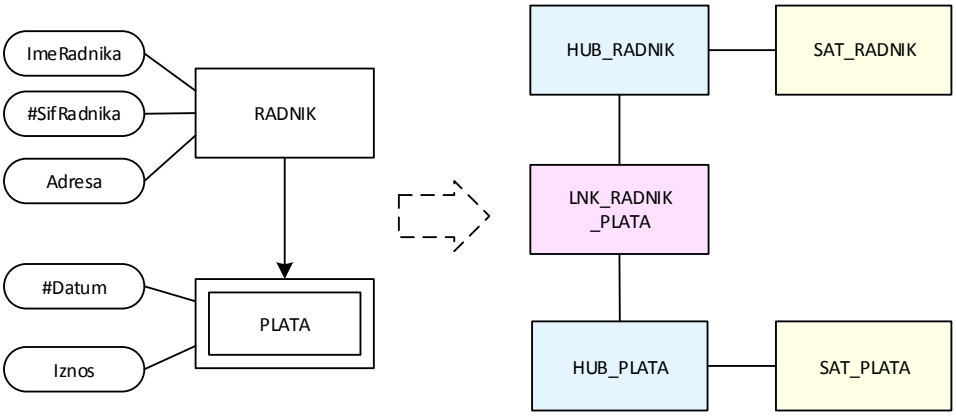
Veza patern: Objekti koji su u vezi se transformišu u habove i pripadajuće satelite, kao u prethodnom primeru. Veze se transformišu u koncept koji predstavlja vezu, odnosno Link bez obzira na kardinalnosti u izvornom modelu. Praktično, asocijacija se uvek realizuje kao Link koji predstavlja realizaciju M:M veze, čak iako je u izvornom modelu gornja granica kardinalnosti 1. Jedna od varijacija Veza paterna je prikazana na Slici 42.



Slika 42. Patern 'Veza MOV'

Postavljanje Link koncepta između dva povezana entiteta omogućava fleksibilnost modela na taj način što ukida strukturnu vezu između entiteta koja postoji na izvornom modelu. Ovaj princip će biti zadržan i na ostalim paternima, kao što će biti ilustrovano u sledećim primerima.

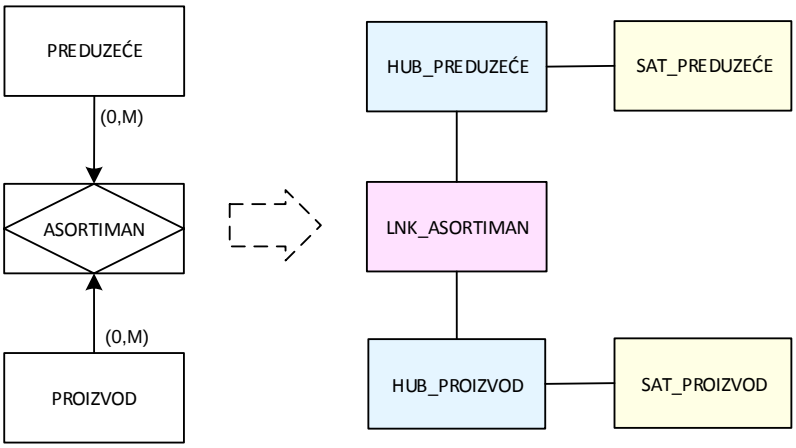
Slab objekat patern: Kao i u prethodnim primerima, objekti se transformišu korišćenjem *Objekat – Atribut paterna*, dok se veza ka slabom objektu transformiše u *Dependancy* link, kao što je prikazano na Slici 43.



Slika 43. Patern 'Slab objekat MOV'

Poslovni ključ slabog objekta postaje složen ključ formiran od identifikatora slabog i objekta sa kojim je u posmatranoj vezi.

Agregacija patern: Kao i u prethodnim primerima, objekti se transformišu korišćenjem *Objekat – Atribut paterna*, dok se agregacija transformiše u odgovarajući Link, kao što je prikazano na Slici 44.

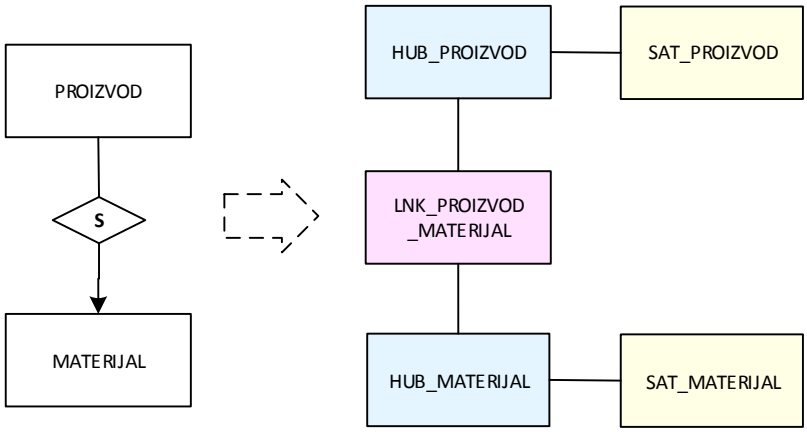


Slika 44. Patern 'Agregacija MOV'

Kao što je primetno, sva tri prethodna primera imaju istovetan rezultujući model, bez obzira koja veza između objekata je uspostavljena na izvornom modelu. Za svaki od ovih primera se koristi odgovarajuća vrsta *Dependancy* linka, kako bi se sačuvala semantika izvornog modela.

Osnovna namera ovako definisanih paterna je da se izvrši strukturno razdvajanje objekata koji su u vezi, tj. da se postigne da nijedan objekat na rezultujućem modelu ne sadrži delove ili celinu drugog objekta.

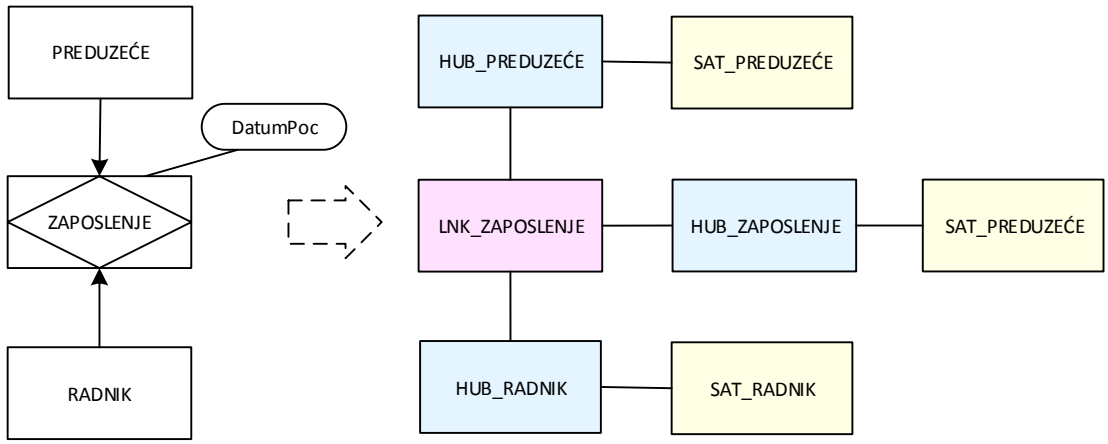
Generalizacija patern: Apstrakcija generalizacije/specijalizacije na izvornom modelu se transformiše u odgovarajući Link koncept, dok se objekti transformišu korišćenjem paterna *Objekat – Atribut*.



Slika 45. Patern 'Generalizacija MOV'

Kao što je prikazano na Slici 45, i ova (tzv. "is a") veza rezultira istim C-DV modelom kao i prethodne veze, sa tom razlikom što se u ovom slučaju koristi *Generalization* vrsta linka.

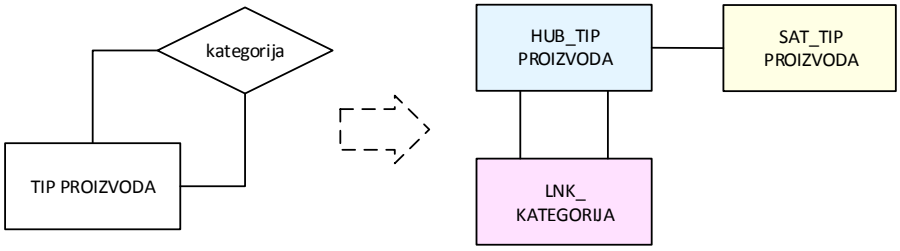
Agregacija sa atributima patern: U slučaju agregacije koja ima pripadajuće atribute, transformacija se vrši na sledeći način: za objekte koji učestvuju u asocijaciji, koristi



Slika 46. Patern 'Agregacija sa atributima MOV'

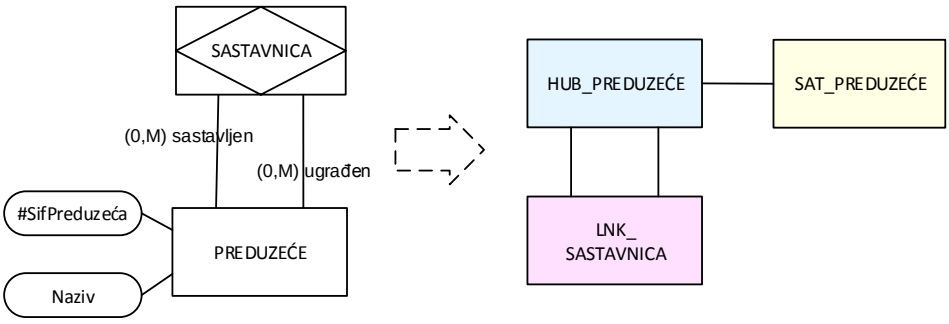
se patern *Objekat – Atribut*, dok se sama agregacija transformiše u odgovarajući hab sa pripadajućim satelitima, kao što je prikazano na Slici 46.

Rekurzivna veza patern: Pri postojanju rekurzivne veze u izvoru podataka, transformacija se vrši na taj način što se za objekat primenjuje *Objekat – Atribut* patern, dok se veza transformiše u Link koji je dva puta povezan sa habom, po jednom za svaku rolu veze, kao što je prikazano na Slici 47.



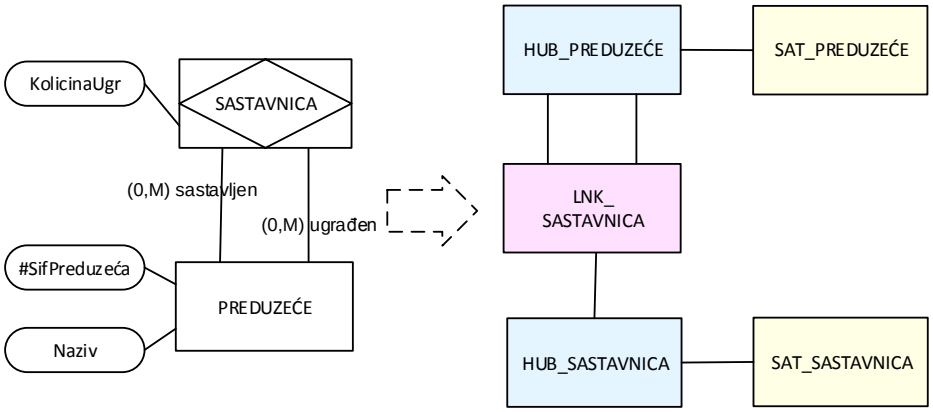
Slika 47. Patern 'Rekurzivna veza MOV'

Rekurzivna agregacija patern: Rezultujući C-DV model je istovetan kao u prethodnom primeru. Naime, ukoliko se na izvornom modelu definiše agregacija bez dodatnih atributa, takva agregacija se posmatra kao veza koja je prerasla u agregaciju zbog pravila preslikavanja MOV → FMOV. Ciljni model se realizuje na taj način što se za objekat primenjuje *Objekat–Atribut* patern, dok se veza transformiše u link koji je dva puta povezan sa habom, kao što je prikazano na Slici 48.



Slika 48. Patern 'Rekurzivna agregacija MOV'

Rekurzivna agregacija sa atributima patern: Ukoliko je na izvornom modelu definisana agregacija sa atributima, agregacija se smatra jakim objektom, tako da i ona prerasta u hab sa pripadajućim satelitom, kao što je prikazano na Slici 49. Link povezuje dva kreirana haba.



Slika 49. Patern 'Rekurzivna agregacija sa atributima MOV'

Prethodnim primerima data su pravila preslikavanja izvornog modela (opisanog MOV jezikom) u ciljni C-DV. Prikazana pravila obuhvataju i obrađuju sve apstrakcije jezika izvornog modela. Na taj način, omogućeno je opisivanje modelom C-DV celokupnog modela EDW na konceptualnom nivou, od modela koji dolaze iz više izvora koji su opisani različitim jezicima za opis. Da bi se to pokazalo, za opisivanje transformacionih paterna su uzeta dva semantički najbogatija jezika za opis strukture sistema, kao što je prikazano u ovom i prethodnom odeljku.

Iz primera je uočljivo da date transformacije mogu da budu automatizovane. Pored toga, opisivanje skladišta podataka na jedinstven i uniforman način, daje stabilnu osnovu za dalje projektovanje i realizaciju skladišta podataka.

Iz datih primera se takođe može izvesti zaključak da C-DV modeli nisu bliski korisniku kao standardni jezici za opis. Stoga bi alat koji bi se koristio u izgradnji EDW, usled mogućnosti automatskih transformacija i afiniteta korisnika, mogao inverznom transformacijom da prikazuje sve skladištene modele u notaciji koja korisniku najviše odgovara, kao što su Dijagram klasa ili MOV. To praktično znači da bi korisnik sve modele obrađivao u željenoj, jedinstvenoj notaciji bez obzira u kojem i na koliko različitih jezika su definisani na strani izvora podataka.

6 LOGIČKI DATA VAULT

Za razliku od konceptualnih modela koji treba prvenstveno da omoguće opisivanje semantike realnog sistema iz ugla korisnika, logički modeli predstavljaju pogled projektanta na dati sistem. Iako ne sadrži detalje o konkretnoj tehnološkoj platformi, logički model se definiše za neku postojeću paradigmu, kao što su relacioni ili dimenzioni model. Logički model, pored toga, sadrži sve atribute i veze između objekata.

6.1 Logički Data Vault Model – L-DV

Logički Data Vault (eng. *Logical Data Vault – L-DV*) je metamodel čija je namena opis platformski nezavisnih DW modela, kao i njihova integracija. Pored toga, namena L-DV metamodela je omogućavanje automatske transformacije C-DV → L-DV čime se postiže prilagođavanje izvornih modela ciljnoj tehnološkoj paradigmi (najčešće se relacioni model podrazumeva kao implementaciona platforma).

Kao što je opisano u poglavlju *Konceptualni Data Vault*, Rečnik skladišta podataka čuva sve modele izvora i njihove verzije. Kako ovaj pristup ima za cilj čuvanje svih podataka svih (integrisanih) izvora u svim stanjima, što će detaljnije biti opisano u sledećem poglavlju, moguće je dobiti istorijsko stanje nekog sistema sa strukturom i podacima koji su važili u bilo kom trenutku vremena.

Model opisan L-DV modelom predstavlja integrisani pogled na celokupno skladište podataka, odnosno konkretni L-DV model obuhvata sve transformisane C-DV modele i njihove verzije. L-DV model se dopunjuje vremenom na taj način što se svaka promena izvora, tj. definisanje nove verzije odgovarajućeg C-DV modela ugradi u postojeći integrisani L-DV model.

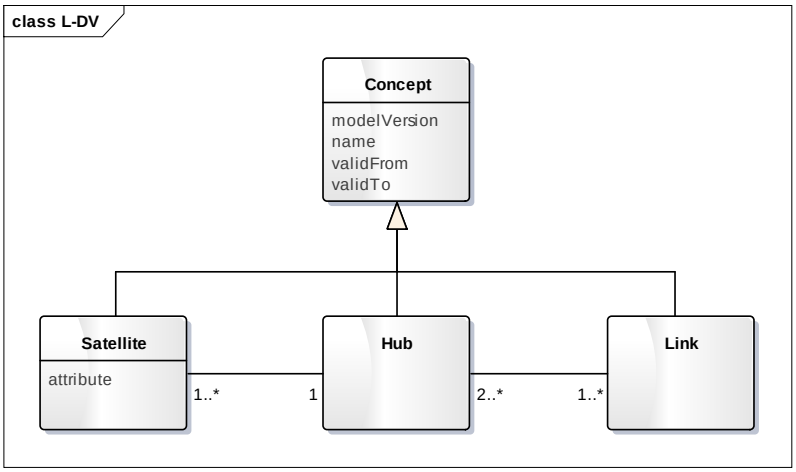
Kao što je prikazano na Slici 50, L-DV model je sveden na osnovne koncepte empirijskog Data Vault modela sa sledećim ograničenjima:

- Link ne može da ima pridružene atribute
- Veza između dva Linka nije dozvoljena
- Veza između Habova koja je n-arna nije dozvoljena
- Sateliti sadrže isključivo jedan atribut
- Poslovni ključ se izdvaja iz objekta i postaje Satelit

Sva predstavljena ograničenja su ugrađena u C-DV → L-DV transformacione paterne, kao što će biti prikazano u odeljku *Pravila transformacije C-DV u L-DV*.

Kao što je prikazano, osnovni koncept modela je Hub koji predstavlja objekat realnog sistema bez strukture. Struktura Haba se definiše Satelitima, pri čemu se svaki Satelit sastoji iz isključivo jednog atributa. Na taj način se model normalizuje do nivoa 6NF čime se postiže da se promena u vremenu strukture objekta može održavati jednostavnom aktivacijom ili deaktivacijom atributa, odnosno bez strukturne izmene samog modela.

Poslovni ključevi se definišu na isti način kao Sateliti. Informacija koji atribut ima ulogu poslovnog ključa se preuzima sa C-DV modela. Ova informacija se čuva na „meta“ nivou zbog toga što u istoriji objekta može doći, kako do promene strukture, tako i do promene vrednosti poslovnog ključa, kao što je opisano u primerima u odeljku *Postupak definisanja Modela Domen / Preslikavanje* (Problem br. 3) u sledećem poglavlju.



Slika 50. L-DV metamodel u UML notaciji

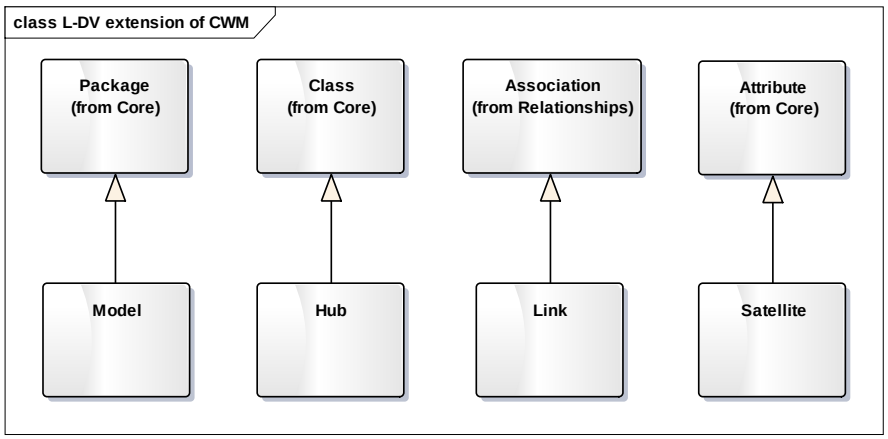
Veze između Habova se održavaju preko Link koncepta. Link povezuje najmanje dva Haba. U L-DV modelu se sve vrste veza, prikazane pri opisu C-DV metamodela, tretiraju podjednako. Semantika veze se, kao i za poslovne ključeve, održava preko verzije C-DV modela.

Svi prikazani koncepti na sebi nose vezu ka verziji C-DV modela sa kojim su prvi put uvedeni u sistem. Takođe, poseduju i dva datuma (*validFrom, validTo*) koji im određuju period validnosti, odnosno vreme nastanka i vreme gašenja objekta, veze ili atributa u C-DV modelu koji predstavlja model izvora.

Za ispravno funkcionisanje L-DV modela i odgovarajućih transformacija, neophodno je korišćenje već pomenutog *Modela preslikavanja* koji će omogućiti preslikavanja koncepata C-DV modela u koncepte L-DV modela.

6.2 L-DV u okviru MDA arhitekture

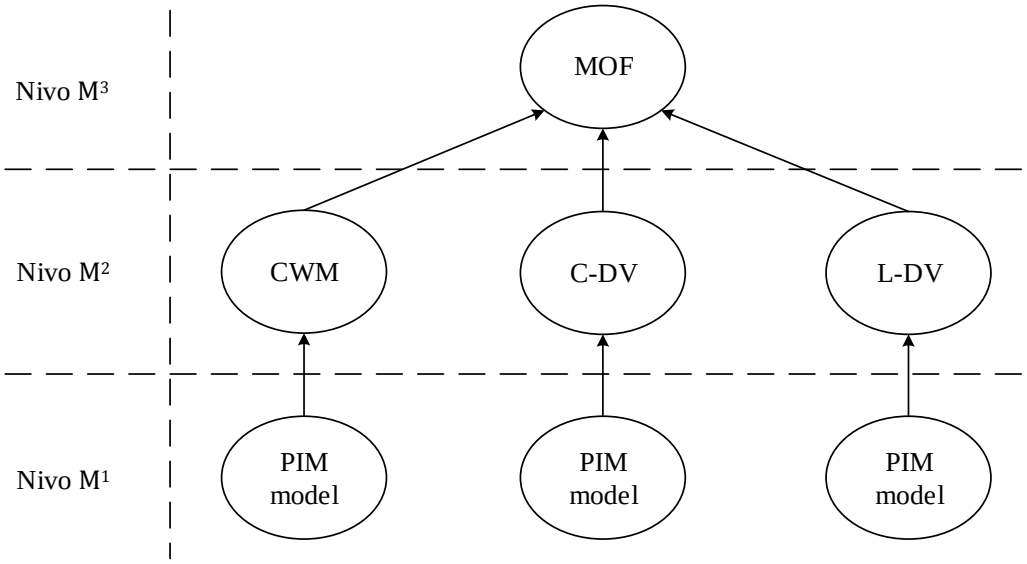
Slično kao C-DV model, i L-DV model se definiše kao ekstenzija CWM metamodela, kao što je prikazano na Slici 51, čime se omogućavaju odgovarajuće transformacije između ova dva modela.



Slika 51. L-DV koncepti kao ekstenzija CWM

Svaki pojedinačni koncept se izvodi iz odgovarajućeg MOF koncepta. Kao što je prikazano, L-DV Model je izveden iz *Package* koncepta i predstavlja strukturu nekog sistema ili njegovog dela. Hab je skup objekata specijalizovan iz *Class* koncepta. Link je *Association*, dok su pripadajući Sateliti haba i linka definisani kao *Attribute.*

Na taj način L-DV metamodel je pozicioniran kao M² model u okviru MDA arhitekture kao što je prikazano na Slici 52.



Slika 52. L-DV u okviru MDA arhitekture

Kao što je prikazano, L-DV model se nalazi na istom nivou MDA arhitekture kao CWM i C-DV. Postavljanjem L-DV modela na M² nivou, omogućeno je da se na standardan način izvrši primena odgovarajućih transformacija, korišćenjem XMI jezika definisanog za razmenu metapodataka.

6.3 Pravila transformacije C-DV u L-DV

Jedan od postupaka izgradnje skladišta uključuje i transformacije iz polaznog C-DV modela u ciljni L-DV model. Ove transformacije predstavljaju prilagođavanje izvornih modela ciljnoj tehnološkoj platformi. Za razliku od modela opisanih C-DV metamodelom, koji predstavljaju domensko znanje o realnom sistemu, ciljni L-DV model predstavlja platformski nezavisan model posmatranog sistema.

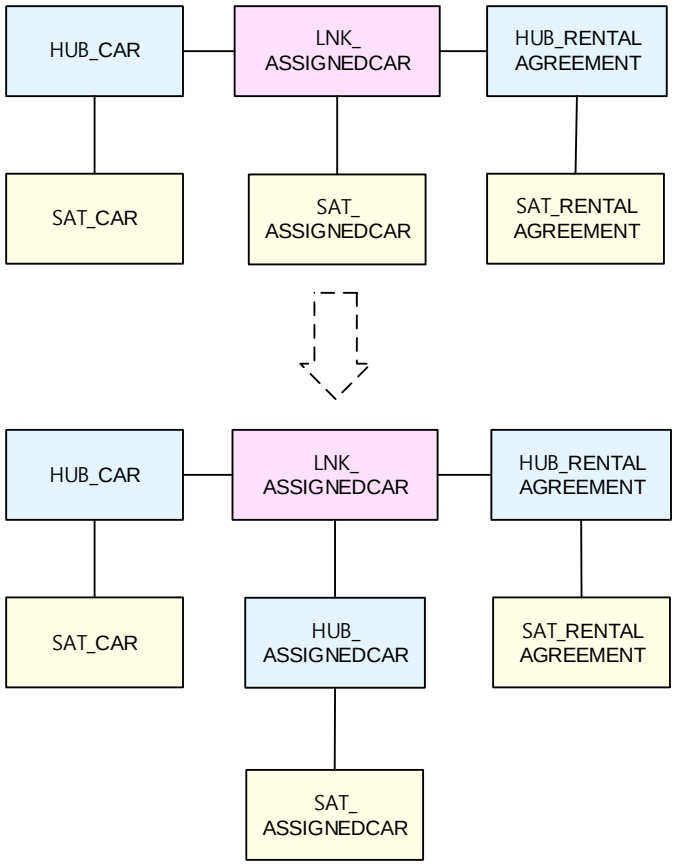
Transformacija se vrši svaki put kada se pojavi potpuno novi C-DV model ili se definiše nova verzija nekog od postojećih C-DV modela. Rezultat transformacije u oba slučaja je isključivo proširenje integrisanog L-DV modela (koncepti koji prestaju da postoje novom verzijom modela, ostaju deo integrisanog L-DV modela zbog

istorijskih razloga). Transformacije se vrše uz konsultovanje modela verzionisanja i modela preslikavanja. Na primer, isti Hab koji postoji u dva C-DV modela, kreiraće se u L-DV modelu samo jednom.

C-DV → L-DV transformacije su trivijalne (svaki pojedinačni C-DV koncept se transformiše u L-DV koncept) osim za ograničenja definisana u odeljku *Logički Data Vault Model – L-DV.*

U ovom odeljku se definiše skup pravila transformacija iz izvornog C-DV modela u ciljni L-DV model. Prikazani skup transformacionih paterna obrađuje sve specifične slučajeve transformacija, odnosno sve slučajeve gde transformacija nije trivijalna. Ilustracija transformacionih paterna će biti izvršena preko već prikazanih rezultujućih modela opisanih u odeljku *Pravila transformacije UML u C-DV.*

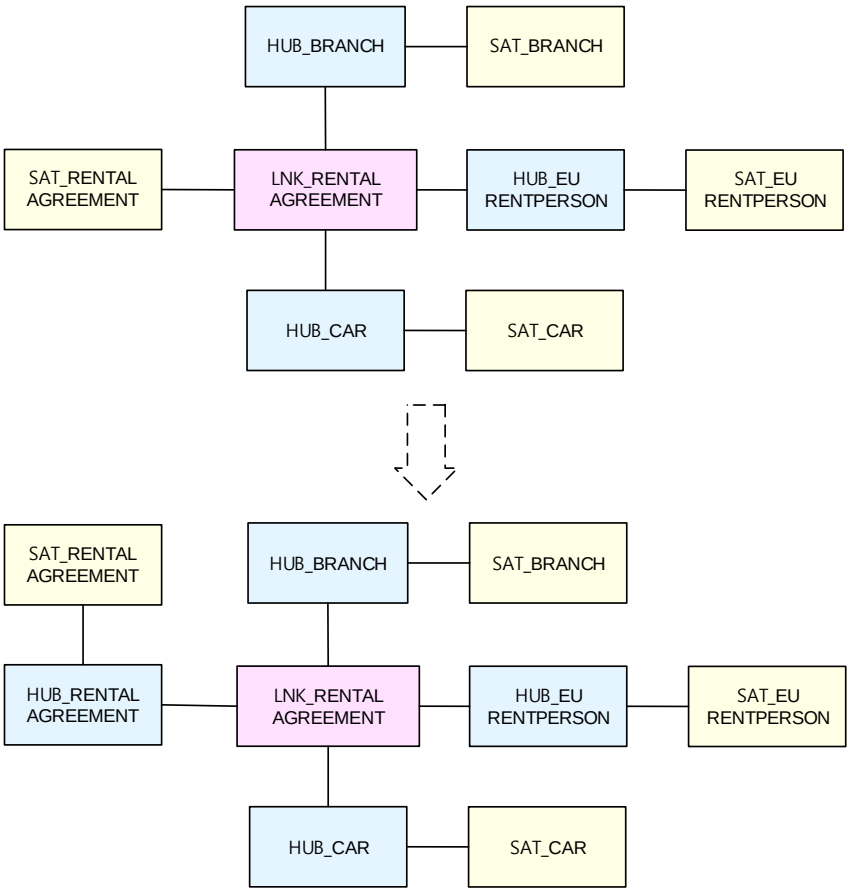
Asocijacija kao klasa patern: Kao što je prikazano na Slici 53, za ovaj primer je korišćen rezultujući model paterna sa istim nazivom iz skupa UML → C-DV transformacija. U ovom primeru se taj model koristi kao ulazni model.



Slika 53. Asocijacija kao klasa patern C-DV → L-DV

Kao što je ilustrovano, ulazni model sadrži link *LNK_ASSIGNEDCAR* sa dodatnim atributima, što nije dozvoljeno prema opisanim ograničenjima. Stoga se kreira dodatni hab *HUB_ASSIGNEDCAR* koji se vezuje za posmatrani link i kojem se pridružuju odgovarajući sateliti. Pored toga, svi sateliti se dekomponuju na onoliko satelita koliko je svaki imao atributa u izvornom C-DV modelu (tj. prikazani sateliti na rezultujućem modelu *SAT_CAR, SAT_ASSIGNED_CAR i SAT_RENTALAGRIMENT* predstavljaju kolekcije satelita).

N-arna asocijacija kao klasa patern: U sledećem primeru je kao ulazni model dat rezultujući model u slučaju n-arne asocijacije kao klase. Prikazani model ima link *LNK_RENTALAGRIMENT* sa pridruženim atributima, kao u prethodnom primeru.

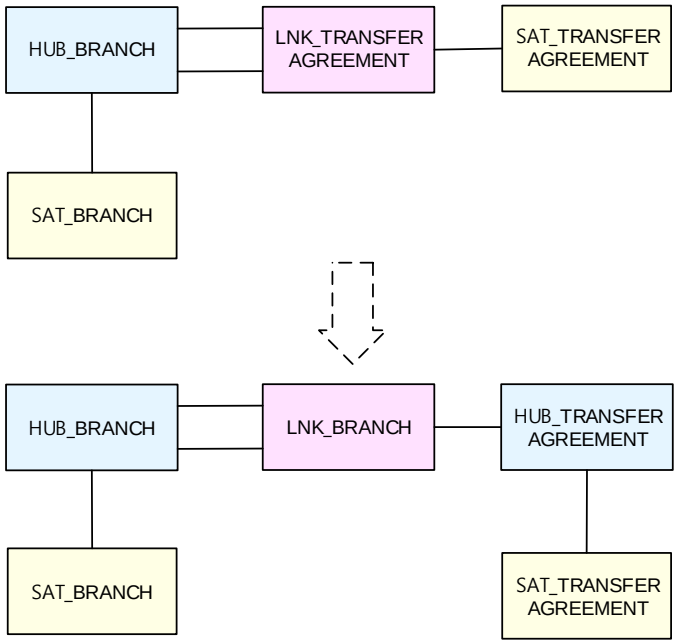


Slika 54. N-arna asocijacija kao klasa patern C-DV → L-DV

Na isti način uvodi se dodatni hab *HUB_ RENTALAGRIMENT* koji se vezuje za posmatrani link i kojem se pridružuju odgovarajući sateliti. Kao i u prethodnom

slučaju, svi sateliti se dekomponuju na onoliko satelita koliko je svaki imao atributa u izvornom C-DV modelu (tj. prikazani sateliti na rezultujućem modelu *SAT_CAR, SAT_BRANCH, SAT_EURENTALPERSON i SAT_RENTALAGRIMENT* predstavljaju kolekcije satelita).

Rekurzivna asocijacija patern: Kao ulazni model dat je rezultujući model paterna *Rekurzivna asocijacija* iz UML → C-DV preslikavanja. Ovaj model je kao rezultat sadržao link *LNK_TRANSFERAGREEMENT* koji ima pridružene satelite.



Slika 55. Rekurzivna asocijacija patern C-DV → L-DV

Transformacija se vrši na taj način što se u ciljni model uvodi odgovarajući hab *HUB_TRANSFERAGREEMENT* kojem se pridružuju odgovarajući sateliti. Dodatno, rezultujući sateliti *SAT_BRANCH* i *SAT_TRANSFERAGREEMENT* predstavljaju kolekcije satelita, tj. za svaki atribut iz satelita na ulaznom modelu, definiše se tačno jedan satelit na ciljnom L-DV modelu.

Primenom prikazanih paterna transformacije, trivijalnih transformacija i dekomponovanjem ulaznih satelita u više pojedinačnih satelita, izgrađuje se jedinstveni L-DV model koji sa svega tri koncepta opisuje celokupnu strukturu izvornih modela na integrisan i uniforman način.

L-DV model je takođe zamišljen na taj način da se u njemu održavaju sve verzije izvornih modela. To praktično znači da se jedinstveni L-DV model gradi isključivo nadograđivanjem bez izmene ili brisanja postojećih koncepata.

Na taj način, L-DV model ispunjava dve najvažnije funkcije skladišta podataka, integrativnu i istorijsku.

Ovako definisan model može da se transformiše u različite fizičke PIM modele skladišta podataka. Sa druge strane, moguće je za ovako definisan model direktno napraviti PSM i implementirati korporativno skladište podataka direktnim korišćenjem L-DV koncepata. Takav pristup bi praktično bio vrlo sličan onom koji koristi empirijski Data Vault. Međutim, nedostaci koji su opisani u odeljku *Analiza modela strukture skladišta podataka* i koji proizilaze iz takvog pristupa ostaju nerešeni.

U sledećem poglavlju je dat predlog jednog mogućeg modela u koji L-DV može da se transformiše i koji anulira sve prikazane nedostatke postojećih modela podataka.

7 FIZIČKI DATA VAULT (MODEL DOMEN / PRESLIKAVANJE)

Model podataka koji opisuje fizičku strukturu skladišta podataka treba da bude dovoljno opšt da može da pomiri semantičke različitosti konceptualnih ili logičkih modela. Takođe, fizički model treba da opiše sve neophodne aspekte tehnološke platforme na kojoj će se skladište podataka realizovati. Pored toga, treba da ispuni odgovarajuće zahteve, kao što su:

(1) Prihvatanje različitih izvora podataka

(2) Prihvatanje svih podataka izvora i pratljivost

(3) Otpornost na promene izvora

(4) Integrisani, ali ne i konsolidovani podaci

U ovom poglavlju će biti predstavljen predlog modela koji opisuje detalje ciljne tehnološke platforme, kao i način transformacije iz polaznog L-DV modela u ciljni fizički model.

7.1 Fizički Data Vault Model – P-DV

Fizički Data Vault (eng. *Physical Data Vault – P-DV*) je metamodel čija je osnovna namena opis platformski nezavisnih (PIM) i platformski zavisnih (PSM) modela koji opisuju detalje ciljne tehnološke platforme.

Pored navedenog, namena P-DV modela je omogućavanje automatske transformacije L-DV → P-DV čime se postiže prilagođavanje izvornih modela ciljnoj tehnološkoj platformi na kojoj će se realizovati skladište podataka. Transformacije uključuju i P-DV(PIM) → P-DV(PSM) preslikavanja. Na taj način dobijeni P-DV(PSM) modeli praktično predstavljaju DDL strukturu koja može da se izvrši nad konkretnim sistemom za upravljanje bazom podataka.

Predloženi P-DV model nije zavisan u odnosu na prikazane C-DV i L-DV modele, već se može koristiti nezavisno kao model implementacije za većinu razmatranih konceptualnih modela u prethodnim poglavljima.

Modeli (PIM i PSM) opisani P-DV metamodelom su deo repozitorijuma modela Rečnika skladišta podataka. Pored do sada definisanih modela Rečnika skladišta podataka, za ispravno funkcionisanje PIM → PSM transformacija, neophodno je postojanje najmanje sledećih komponenti:

- Model preslikavanja (eng. *Weaving Model*) koji se koristi za mapiranje L-DV modela u koncepte P-DV modela
- Model za opis tehnološke platforme (eng. *Platform Description Model*) je model koji opisuje detalje ciljne tehnološke platforme i njene specifičnosti, i
- Generator - koji na osnovu PIM modela i definicije tehnološke platforme generiše odgovarajući PSM model koji praktično predstavlja trivijalno preslikavanje u DDL naredbe za konkretnu tehnološku platformu.

U sledećim odeljcima će biti prikazan Model Domen / Preslikavanje – MDP (eng. *Domain/Mapping Model – DMM*) koji je dovoljno opšt da pomiri semantičke različitosti konceptualnih i logičkih modela skladišta podataka i da ispuni specifične zahteve date na početku poglavlja. MDP je jezgro, tj. osnova P-DV modela.

7.2 Postupak definisanja Modela Domen / Preslikavanje

U ovom odeljku će biti objašnjen postupak projektovanja MDP modela kroz definisanje i razmatranje problema uočenih u odeljku *Analiza modela strukture skladišta podataka*, kao i na osnovu odgovarajućih zaključaka koje predloženi model treba da zadovolji.

Problem br. 1: Proširivost i prilagodljivost modela strukture

Kao što je pokazano, modeli strukture kod kojih je veći stepen integrisanosti atributa i veza sa objektom (Normalizovan model i Dimenzioni model) pokazuju manji stepen fleksibilnosti i prilagodljivosti promenama koje se dešavaju u strukturi izvora, u odnosu na modele koji takvu integraciju nemaju ili je imaju u manjoj meri (Data Vault i Anchor Model).

Primer br. 1: Prikazani primeri u odeljku *Analiza otpornosti modela na promenu strukture izvora*

Zaključak br. 1: Model strukture treba da omogući da atributi i veze nemaju „čvrstu“ vezu ka objektima sistema, kao i automatsku transformaciju koncepata modela izvora u 6NF.

Problem br. 2: Otpornost modela na promenu vrednosti identifikatora objekta

Iako ovaj problem nije eksplicitno prikazan u analizi modela strukture (jer identifikator jeste deo strukture objekta), značajan je stoga što prikazani modeli strukture identifikator objekta različito tretiraju. Identifikator objekta je *jednoznačni atribut*, kod kojeg i inverzno preslikavanje Domen → Objekat jeste (0 ili 1, 1). Sa stanovišta transakcionih sistema je ovaj stav ispravan, međutim, u domenu skladišta podataka koje imaju vremensku dimenziju i potencijalno više izvora podataka, ova definicija se ne može primeniti na *poslovni identifikator* objekta.

Primer br. 2.1: PIO fond Republike Srbije se sastoji iz dva skoro nezavisna informaciona sistema koji vode filijale Beograd i Novi Sad. Postoji određen broj

osiguranika koji, kao poslovni identifikator, imaju dva različita važeća Lična broja – LB, koji su dodeljeni istoj osobi u ove dve filijale. Prilikom integracije dva sistema oba identifikatora moraju da budu sačuvana. Na taj način, isti objekat ima dve različite vrednosti identifikatora istovremeno. Takođe, postoje slučajevi kada je ista vrednost LB dodeljena različitim osiguranicima.

Primer br. 2.2: MUP Republike Srbije kontroliše izdavanje JMBG brojeva, jedinstvenih identifikatora građana. Postoji određen broj građana koji imaju dva različita JMBG broja istovremeno važeća. Takođe, postoji određen broj slučajeva kada je isti JMBG broj dodeljen različitima licima.

Zaključak br. 2: Model strukture treba da ima definisan identifikator objekta, ali i da omogući da preslikavanje objekta i identifikatora bude M:M, odnosno da identifikator objekta bude *višeznačni atribut*. Isto važi i za ostale atribute objekta.

Problem br. 3: Otpornost modela na promenu strukture identifikatora objekta

Tip objekta može, istorijski gledano, da iskusi promenu (grupe) atributa koji predstavljaju poslovni identifikator. Problem se posebno odnosi na modele koji implicitno podrazumevaju postojanje poslovnog identifikatora (Data Vault i Dimenzioni model).

Primer br. 3.1: Poslovni identifikator objekta Osiguranik u PIO fondu do 1982. godine je bio LB. Promenom pravne regulative, od tog perioda je uveden JMBG kao identifikator objekta.

Primer br. 3.2: Poslovni identifikator objekta Obveznik u PIO fondu do 2003. godine je bio Registarski broj. Promenom pravne regulative, od tog perioda uvedena je grupa atributa: Poreski identifikacioni broj – PIB, Opština i Ulica kao složeni identifikator objekta Obveznik.

Zaključak br. 3: Model strukture skladišta podataka treba da omogući skladištenje semantički različitih poslovnih identifikatora istog objekta u različitim periodima vremena.

Problem br. 4: Redundansa podataka

Iako većina prikazanih modela strukture (osim Dimenzionog modela) posebnu pažnju pridaje ovom problemu, redundansa podataka se neizostavno javlja ukoliko se više izvora podataka uključuje u skladište podataka u različitim vremenskim periodima. Izvor ovog problema je što svi modeli strukturno vezuju atribute za pripadajuće objekte.

Primer br. 4: MUP Republike Srbije u svom sistemu, pored ostalih, ima dve funkcije: Matičnu evidenciju (obrada i dodeljivanje JMBG brojeva) i Kadrovsku evidenciju. Bez obzira koji od ovih podsistema se prvi unese u skladište podataka, sam JMBG broj postaje, kao atribut, sastavni deo objekta kojem pripada, npr. Fizičkom licu. Uvođenjem drugog sistema, Kadrovske evidencije, i objekta npr. Zaposleni koji kao atribut takođe ima JMBG. Na ovaj način se skup vrednosti JMBG na dva mesta vodi u istom skladištu podataka, nezavisno jedan od drugog.

Zaključak br. 4: Domeni, tj. skupovi iz kojih atributi objekata uzimaju vrednosti treba da budu strukturno nezavisni, kako bi se omogućilo da te vrednosti koriste atributi različitih tipova objekata istovremeno, kako bi se redundansa podataka svela na najmanju moguću meru.

Problem br. 5: Praćenje vremenskog aspekta

Kao što je pokazano, neki modeli poseduju odgovarajuće mehanizme za praćenje vremenskog aspekta samo delimično (Data Vault, Anchor Model i Dimenzioni model).

Primer br. 5: Uprava Carina Republike Srbije u svom informacionom sistemu, pored ostalih, ima dva podsistema: Tranzitni sistem – NCTS i Referentne podatke – RD. NCTS se u svakodnevnom radu oslanja na RD šifarnike. RD šifarnici imaju period važenja i promene u šifarnicima se uvode u sistem bar mesec dana pre početka korišćenja. Očigledno je da svi objekti sistema podležu vremenskom aspektu.

Zaključak br. 5: Svi objekti, atributi i veze sistema treba da imaju mogućnost praćenja po vremenskoj dimenziji, vremenu važenja i vremenu ažuriranja. Ova karakteristika modela treba da bude implicitna, tj. da ne zavisi od stepena poznavanja realnog sistema ili ekspertize projektanta.

Problem br. 6: Održavanje više verzija istine

Ovaj problem je opisan u odeljku *Analiza kompletnosti i pratljivosti podataka*. Prikazani modeli strukture uglavnom nisu inicijalno definisani da ispune ovaj zahtev. Data Vault i Anchor Model iskazuju određenu fleksibilnost iz razloga koji je već pomenut – atributi i veze nisu strukturno integrisani sa objektom. Nedostaci se i u ovom slučaju ispoljavaju kada se uvede vremenska dimenzija. Zamenom jednog izvora drugim, iz istog poslovnog domena, mehanizmima koji ova dva modela poseduju nemoguće je ispratiti kontinuitet dva izvora, tj. ne mogu se posmatrati kao dve verzije istog modela.

Primer br. 6: U više primera je pokazano da skladište podataka može da očekuje skladištenje više različitih, ali istovremeno validnih podataka.

Zaključak br. 6: Model strukture treba implicitno da omogući skladištenje više različitih vrednosti istovremeno za jednu karakteristiku objekta (vezu ili atribut) i da pri tom ima referencu ka *verziji* izvora (modela) iz kojeg je ta vrednost preuzeta.

Problem br. 7: Svi podaci su jednaki

Primer br. 7: Dva prikazana modela, Data Vault i Anchor model, predviđaju koncept semantički drugačiji od koncepata koji su predviđeni da opišu osnovne tipove objekata realnog sistema. Data Vault definiše Referentne podatke, dok Anchor model podrazumeva korišćenje Čvor koncepta. Oba koncepta igraju ulogu predstavljanja statičkih, nepromenljivih podataka koji klasifikuju poslovne podatke od interesa.

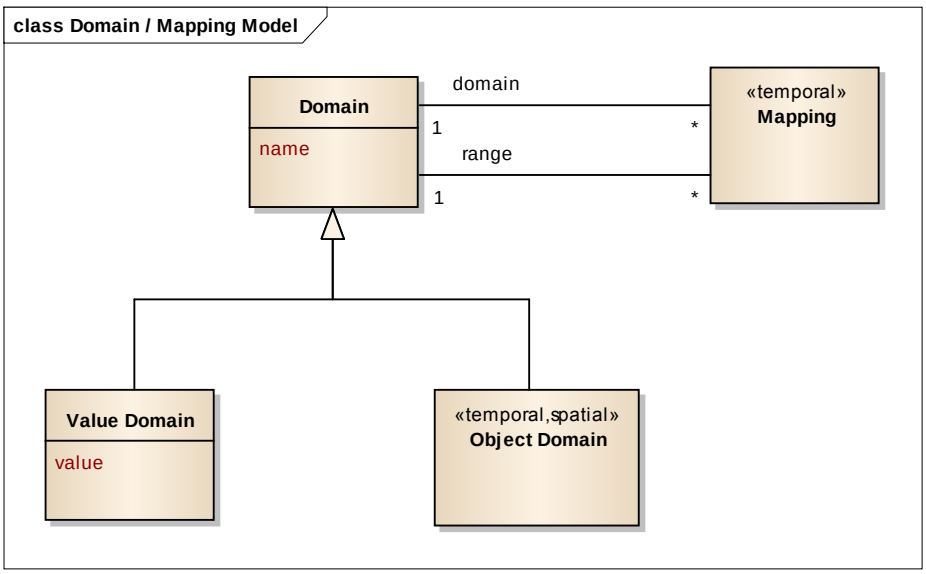
Ministarstvo zdravlja održava više grupa podataka kao što su podaci vezani za organizacionu strukturu, kadrove ili medicinsku opremu zdravstenih ustanova u njenoj nadležnosti. Prilikom obrade, ovi podaci se oslanjaju na odgovarajuće referentne podatke koji pripadaju zasebnom segmentu Nomenklatura i klasifikacija, koji čine jedinstveni šifarski sistem dostupan transakcionim podacima.

Ovako definisani šifarnici bi se u Data Vault i Anchor modelu prikazali kao Referentni podaci i Čvorovi, respektivno. Osnovno ograničenje ovakvog pristupa je što se šifarnici tretiraju kao nepromenljivi, iako su i sami predmet obrade neke poslovne funkcije, kada postaju objekti obrade kao i bilo koji drugi unutar sistema.

Zaključak br. 7: Model strukture treba da tretira podatke na taj način da im pridaje isti značaj bez obzira da li su metapodaci, da li nastaju unutar sistema ili kolika im je frekvencija promene.

7.3 Model Domen / Preslikavanje

Uzimajući u obzir probleme i zaključke opisane u prethodnom odeljku, definisan je Model Domen / Preslikavanje (u daljem tekstu MDP), prikazan na Slici 56. MDP je vrlo opšt model prilagođen pomirenju semantičke različitosti postojećih konceptualnih modela skladišta podataka, ukidanju redundanse, vođenju vremenskog aspekta, omogućavanju pratljivosti, proširivosti i prilagodljivosti, kao i održavanju više verzija istine.

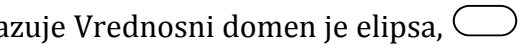


Slika 56. Model Domen / Preslikavanje – Konceptualni model

Osnovni koncept modela je **Domen** (eng. *Domain*) koji predstavlja skup – vrednosti ili objekata. Skup vrednosti je **Vrednosni Domen** (eng. *Value Domain*) koji je univerzalan i nezavisan u odnosu na vremenski i prostorni aspekt. Sastoji se od konačnog broja atomskih celina iz kojih atributi objekta uzimaju vrednosti. Svaki takav skup predefinisano poseduje i dodatni element - nepoznatu (eng. *null*) vrednost.

Definicija 1: Vrednosni domen $D_v = (I, V)$ je uređeni par, gde je:

- I = konačan skup identifikatora
- V = konačan skup vrednosti

Simbol kojim se prikazuje Vrednosni domen je elipsa,  sa nazivom vrednosnog domena unutar nje.

Objektni domen (eng. *Object Domain*) predstavlja skup objekata realnog sistema koji je zavisan u odnosu na vremenski i prostorni aspekt, odnosno objekti su promenljivi u vremenu i prostoru.

Definicija 2: Objektni domen $D_o = (I, T, S)$ je uređena n-torka, gde je:

- I = konačan skup identifikatora
- T = odrednica vremenskog aspekta
- S = odrednica prostornog aspekta

Simbol kojim se prikazuje Objektni domen je dve elipse, jedna unutar druge,  sa nazivom Objektnog domena unutar simbola.

Mapiranje (eng. *Mapping*) je koncept koji omogućava formiranje strukture objekata realnog sistema i njihovih međusobnih veza. Ukoliko se definiše atribut objekta, preslikavanje je između Objektnog i Vrednosnog domena. Kada je u pitanju definisanje veze između objekata, preslikavanje se vrši između Objektnih domena. Izgradnja strukture objekata i njihovih veza je zavisno od vremenskog aspekta. Takođe, definisanje strukture skladišta podataka se vrši na osnovu odgovarajuće *verzije modela* (različitih izvora ili verzija istog modela).

Definicija 3: Mapiranje $P = (F, T, M)$ je uređena n-torka, gde je:

- F = preslikavanje dva domena
- T = odrednica vremenskog aspekta
- M = verzija modela koja definiše preslikavanje

Simbol kojim se prikazuje Mapiranje je neusmerena linija sa nazivom mapiranja na njoj.

Definicija 4: Preslikavanje F je par funkcije preslikavanja, odnosno preslikavanja $f(x) = D_d \rightarrow D_r$ i inverznog preslikavanja $f'(x) = D_r \rightarrow D_d$ gde je:

- D_d = domen preslikavanja
- D_r = kodomen (eng. *Range*) preslikavanja

Definicija 5: Preslikavanje dva vrednosna domena nije dozvoljeno.

Definicija 6: Vremenski aspekt $T = (T_{tt}, T_{vf}, T_{vt})$ je skup vremenskih dimenzija gde je:

- T_{tt} = vreme ažuriranja
- T_{vf} = početak vremena važenja
- T_{vt} = kraj vremena važenja

Definicija 7: Prostorni aspekt $S = (L_t, L_g)$ je uređena dvojka prostornih dimenzija gde je:

- L_t = latituda
- L_g = longituda

Definicija 8: Latituda $L_t = (G_d, G_m, G_s, G_{dw})$ je uređena četvorka gde je:

- G_d = stepeni u rasponu 0 - 90
- G_m = minuti
- G_s = sekunde
- G_{dw} = pravac geografske dužine

Definicija 9: Longituda $L_g = (G_d, G_m, G_s, G_{dh})$ je uređena četvorka gde je:

- G_d = stepeni u rasponu 0 - 180
- G_m = minuti
- G_s = sekunde
- G_{dh} = pravac geografske širine

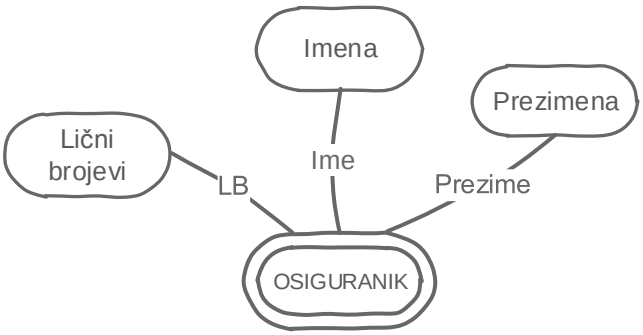
7.4 Način korišćenja Modela Domen / Preslikavanje

U ovom odeljku će kroz nekoliko primera biti prikazan način korišćenja MDP modela. Ovi primeri ujedno pokazuju njegovu primenljivost na rešavanje problema i ispunjavanje zahteva opisanih u odeljku *Postupak definisanja Modela Domen / Preslikavanje*.

Na Slici 57, su data dva pojednostavljena podmodela koji prikazuju promenu strukture koncepta Osiguranik u nekom vremenskom periodu. Kao što je prikazano, početni koncept Osiguranika je promenjen dodavanjem atributa JMBG (Jedinstveni matični broj građana), koji ujedno i postaje poslovni identifikator tog objekta, promenom zakonske regulative. Poslovni identifikator objekta je do te promene bio LB (Lični broj). Kroz dati primer će biti opisano ispunjavanje *Zaključaka 1 do 3*.

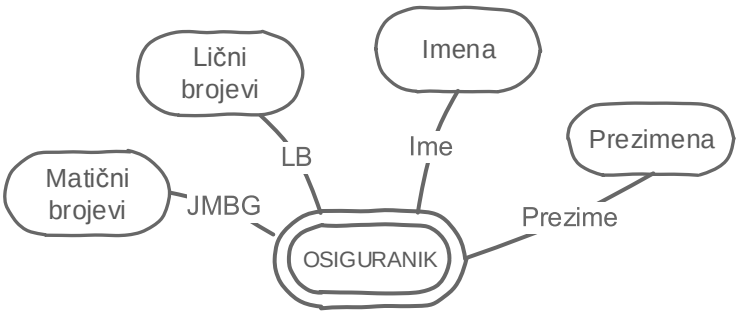
Slika 57. Dodavanje atributa i promena poslovnog identifikatora

Na sledećoj slici je dat model predstavljen u MDP koji prikazuje tri definisana domena – Ličnih brojeva, Imena i Prezimena. Takođe, prikazan je i koncept Osiguranika, koji se preko koncepta Mapiranja, za svoja konkretna pojavljivanja povezuje sa prikazanim domenima.



Slika 58. Početni MDP model – Osiguranik

Kardinalnosti svih preslikavanja su M:M (odnosno svi atributi su višeznačni) čime se ispunjava zahtev da se u jednom trenutku može voditi više vrednosti istog atributa, i to po dve dimenzije: vremenskoj i izvoru podataka. Naime, ukoliko više različitih sistema čuvaju različite vrednosti atributa i ukoliko vrednost atributa bude promenjena u vremenu ovakav model je sposoban da te promene podrži. To je omogućeno time što je preslikavanje domena realizovano preko koncepta Mapiranje koji implicitno održava vreme važenja i metamodel na osnovu koje je preuzeta vrednost važeća. Na ovaj način se u skladištu podataka čuvaju sve *vrednosti* svih *verzija* modela izvora podataka bez gubitka informacija.

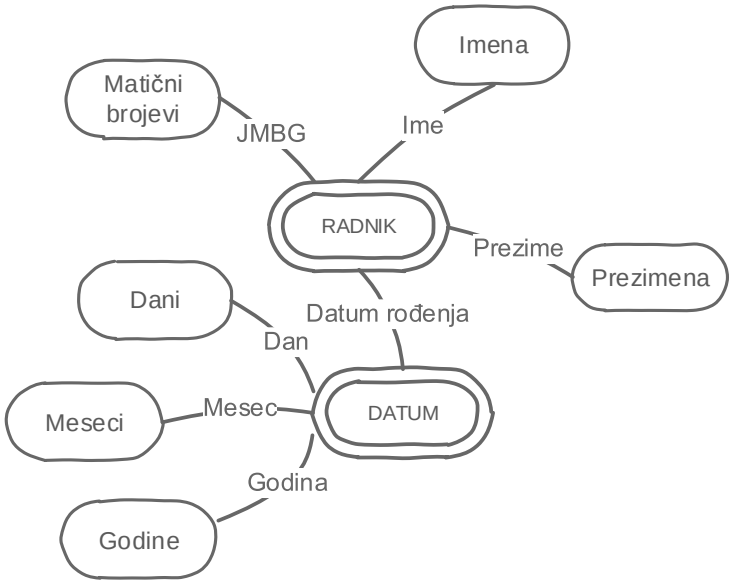


Slika 59. Promenjeni MDP model – Osiguranik

Na Slici 59, dat je primer dodavanja atributa koji ujedno postaje i novi identifikator objekta. Na fizičkom nivou svi pomenuti koncepti se realizuju kao tabele. Kao što je prikazano, izmenjen model je nastao samo dodavanjem novih struktura (ili povezivanjem sa već postojećim strukturama), bez izmena ili brisanja postojećih, odnosno mapiranjem odgovarajućih domena.

U sledećem primeru je prikazano uklanjanje redundanse podataka na taj način što su domeni iz kojih atributi uzimaju vrednosti strukturno nezavisni što omogućava da više različitih atributa istovremeno koristi definisane domene. U primeru je pokazano kako različite poslovne funkcije nastale iz različitih izvora podataka koriste iste vrednosne domene.

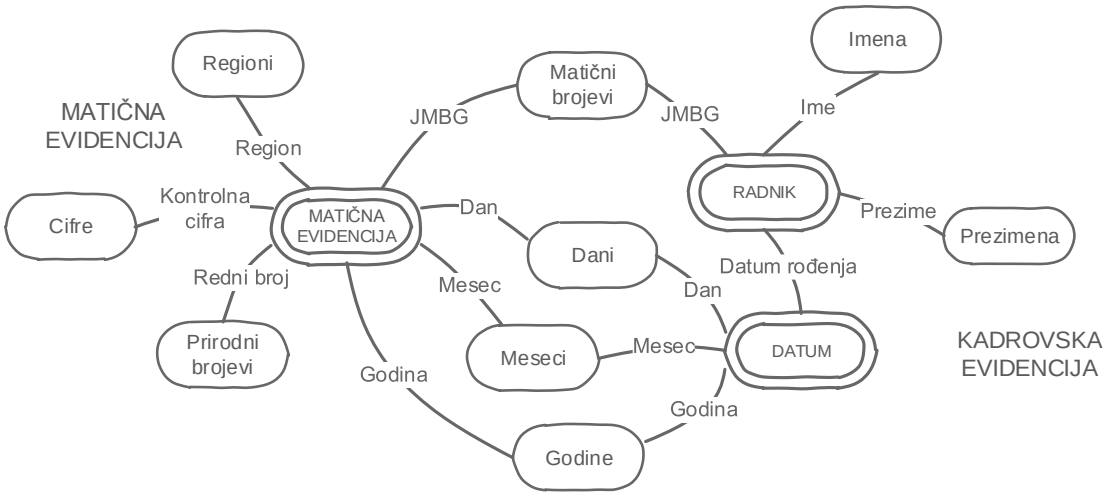
Na Slici 60, prikazan je pojednostavljen podmodel Kadrovske evidencije koji prikazuje koncept Radnik koji se sastoji od atributa JMBG, Ime, Prezime i Datum. Prva tri atributa su preslikavanja sa vrednosnim domenima, dok je Datum mapiranje sa objektnim domenom Datum, koji se sastoji od preslikavanja sa tri vrednosna domena, Dani, Meseci i Godine.



Slika 60. Kadrovska evidencija - podmodel

Ukoliko se u skladište podataka uvede novi izvor podataka, moguće je koristiti postojeće domene sa ciljem ukidanja redundanse, bez izmene već postojeće strukture skladišta podataka. Na sledećoj slici (Slika 61) je prikazan podmodel matične evidencije koji opisuje deo vezan za Jedinstvene matične brojeve građana. U ovom podmodelu se definišu JMBG brojevi svih stanovnika i on sadrži i JMBG brojeve koji se već nalaze u kadrovskoj evidenciji, tj. koji su vezani za zaposlene u organizaciji. Uobičajeno bi se ovi JMBG brojevi redundantno čuvali, pri čemu postoji anomalija u ažuriranju. Na predstavljen način, korišćenjem MDP, ova redundansa se

ukida preslikavanjem oba koncepta, Radnik i MatičnaEvidencija, sa vrednosnim domenom MatičniBrojevi.



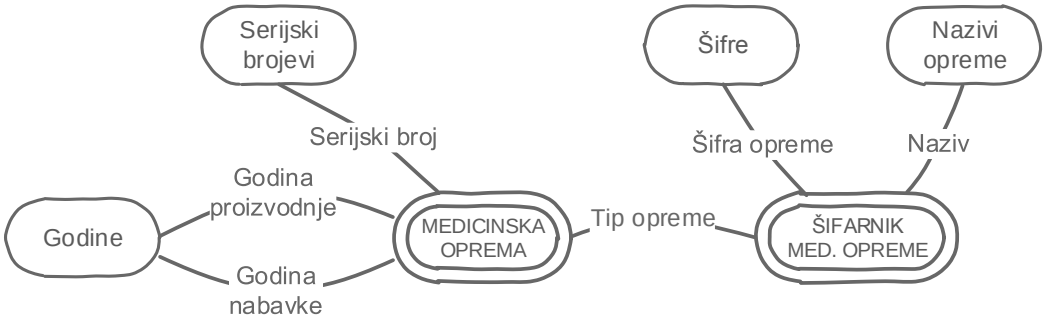
Slika 61. MDP model i redundansa podataka

Takođe, prikazano je kako koncept MatičnaEvidencija koristi vrednosne domene Dani, Meseci i Godine direktno, jer se konkretne vrednosti ovih domena koriste u izgradnji JMBG broja. Na prikazani način se, izgradnjom mreže domena koji se koriste za različite koncepte, redundansa podataka svodi na najmanju moguću meru, čime je premošćen *Problem br. 4* opisan u odeljku *Postupak definisanja Modela Domen / Preslikavanje*.

Pored toga, model MDP odgovara na zahtev postavljen u *Zaključku br. 5* na taj način što svi objekti sistema u skladištu podataka, kao i preslikavanja koja oni ostvaruju sa atributima ili drugim objektima implicitno poseduju vreme važenja i vreme ažuriranja.

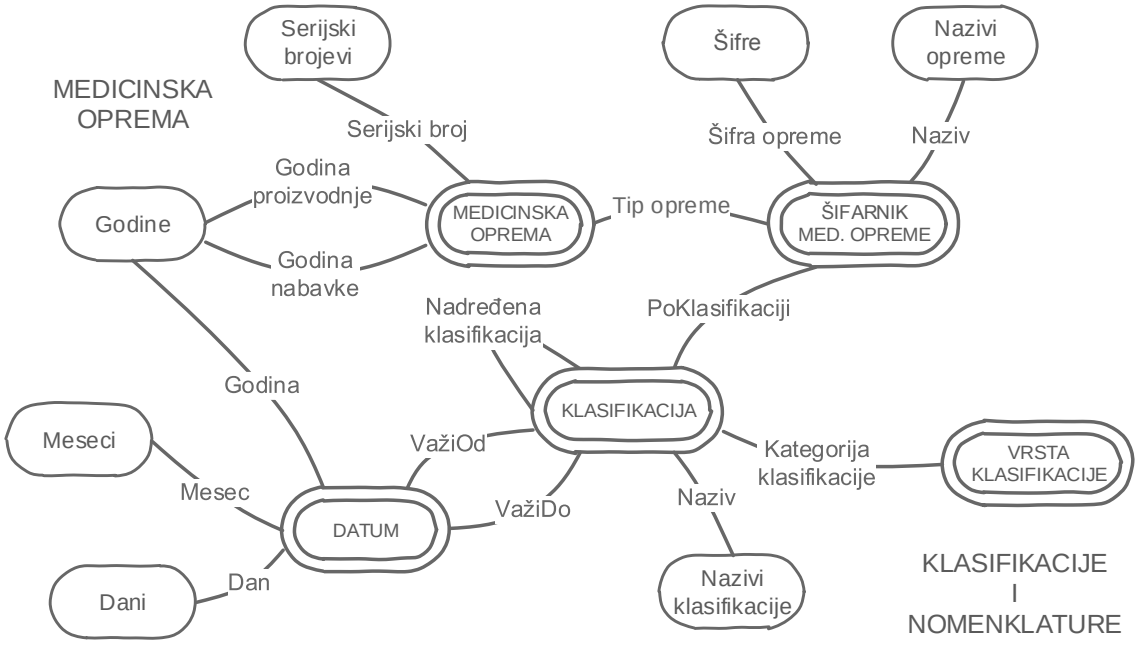
Sledeće, realizacija zahteva iz *Zaključka br. 6* se ostvaruje preko predefinisane M:M kardinalnosti preslikavanja između objekata i atributa ili objekta i drugih objekata. Na taj način je omogućeno da isti objekat istovremeno ima više različitih vrednosti istog atributa ili da istovremeno bude povezan sa različitim objektima koji su istog tipa.

Konačno, ispunjavanje zahteva na osnovu Zaključka br. 7 je prikazano na sledeće dve slike. Na početnoj (Slika 62), prikazan je pojednostavljen podmodel Medicinska oprema.



Slika 62. **Medicinska oprema**

Ukoliko bi se u sistemu skladištio i model koji opisuje klasifikacije, to ne bi bilo moguće bez promene strukture modela, ako je koncept Šifarnik medicinske opreme realizovan tako da ne može da istorijski prati promene svojih vrednosti. Na sledećoj (Slika 63), je prikazano kako se na jednostavan način uvodi potpuno nova funkcija u skladište podataka bez promene postojećeg modela, iako nije na istom nivou apstrakcije kao i podmodel Medicinska oprema.



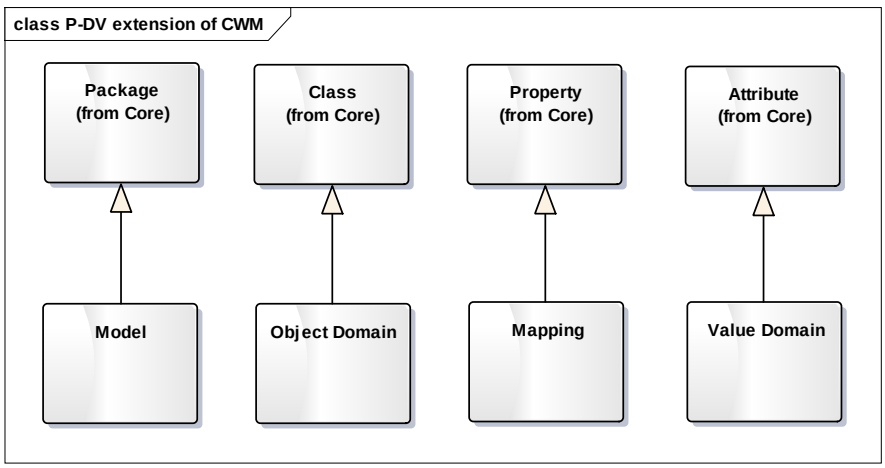
Slika 63. Referentni podaci u MDP

Kao što je u prethodnim primerima pokazano, MDP se sastoji iz dva fundamentalna koncepta, Domena i Preslikavanja, čime je omogućena potpuna fleksibilnost skladišta podataka na fizičkom nivou i otklanjanje nedostataka savremenih modela strukture skladišta podataka. MDP je sposoban da apsorbuje promene strukture koje nastaju u izvorima podataka, da prati podatke do njihovog izvora, da implicitno prati vremensku i prostornu dimenziju podataka i da ujedno pruži integrisani pogled na podatke u skladištu podataka.

Takođe, zbog svoje opštosti, MDP može da pomiri semantičke razlike konceptualnih modela za opis skladišta podataka i stoga je potpuno nezavisan od odabranog jezika kojim je skladište podataka opisano. To stvara mogućnost da se odgovarajućom transformacijom MDP primenjuje u definisanju strukture DW sa većinom postojećih konceptualnih modela.

7.5 P-DV u okviru MDA arhitekture

P-DV model (tj. njegov podmodel MDP), se definiše kao ekstenzija CWM modela, slično kao i C-DV i L-DV modeli, kao što je prikazano na Slici 64.

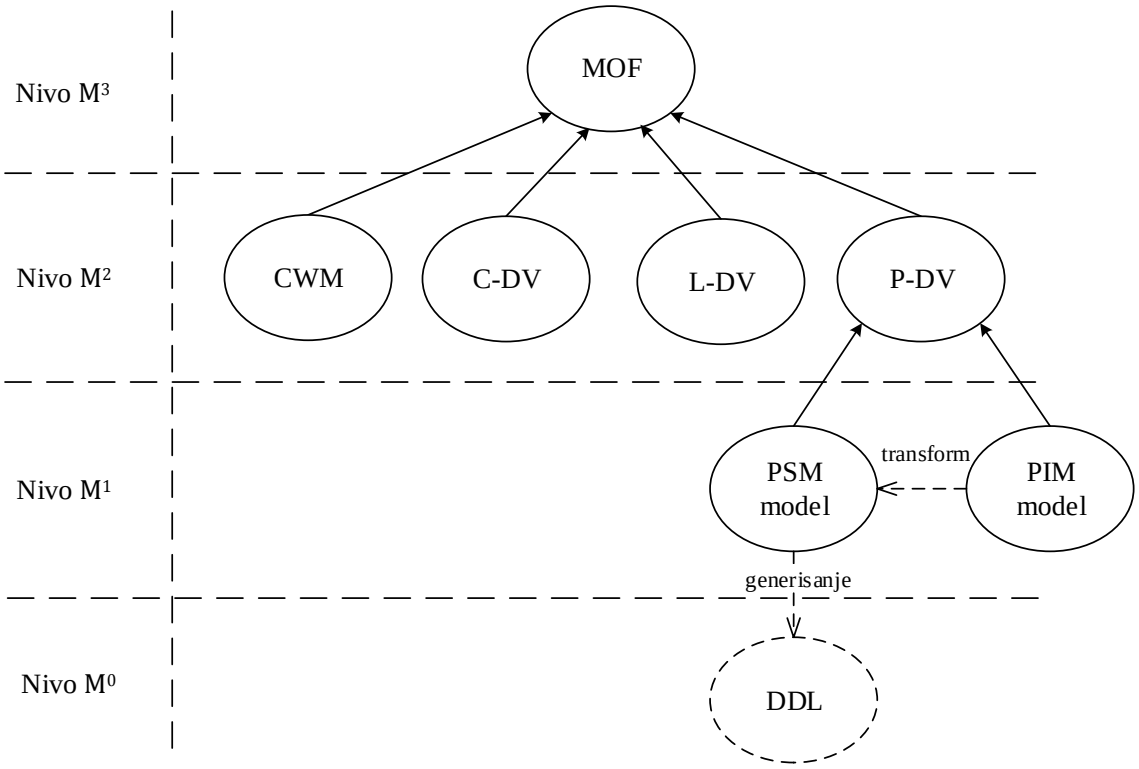


Slika 64. P-DV koncepti kao ekstenzija CWM

Svaki pojedinačni koncept se izvodi iz odgovarajućeg MOF koncepta. Kao što je prikazano, *Model* je izveden iz *Package* koncepta i predstavlja strukturu nekog sistema ili njegovog dela. *Object Domain* je skup objekata specijalizovan iz *Class*

koncepta. Koncept *Mapping* je izveden iz koncepta *Property*, dok je *Value Domain* definisan kao *Attribute.*

Na taj način P-DV metamodel je pozicioniran kao M^2 model u okviru MDA arhitekture kao što je prikazano na Slici 65.



Slika 65. P-DV u okviru MDA arhitekture

P-DV se nalazi na istom nivou MDA arhitekture kao C-DV i L-DV. Postavljanjem P-DV modela na M^2 nivou, omogućeno je da se na standardan način izvrši primena odgovarajućih transformacija, korišćenjem XMI jezika definisanog za razmenu metapodataka.

Kao što je prikazano, definisani PIM model može da se transformiše u odgovarajući PSM model, uz korišćenje Modela za opis tehnološke platforme ili standardnog XMI skupa interfejsa. Nakon toga transformisani PSM model trivijalnom transformacijom generiše skup DDL naredbi za odabranu tehnološku platformu.

7.6 Pravila transformacije L-DV u P-DV

Sve transformacije L-DV → P-DV su trivijalne. Svaki hab i link koji povezuje više od dva haba postaje *Object Domain*, svaki satelit (inicijalno atribut na C-DV modelu) prerasta u *Value Domain* (ukoliko već ne postoji) i prateći *Mapping* objekat koji predstavlja vezu objekta i atributa, pri čemu preuzima naziv satelita. Za svaki link koji vezuje tačno dva haba, formira se *Mapping* objekat koji predstavlja uređeni par - preslikavanje i njegovo inverzno (eng. *opposite*) preslikavanje između dva domena koji predstavljaju objekte.

Važan, ali ne i nužan korak, koji prethodi transformacijama, je *konsolidacija vrednosnih domena*, odnosno projektantska odluka koji vrednosni domeni će biti kreirani, a koji ponovo korišćeni, kako bi se redundansa podataka svela na najmanju moguću meru. Naime, pre transformacije, u administrativnom delu Rečnika skladišta podataka, korišćenjem *Modela preslikavanja*, odgovarajući sateliti se mogu usmeriti na već postojeće, definisane vrednosne domene. Ukoliko se koristi, ovaj korak ne može biti automatizovan.

8 MODELOM VOĐEN RAZVOJ SKLADIŠTA PODATAKA

U prethodnim poglavljima su definisani C-DV, L-DV i P-DV metamodeli i pravila transformacije između koncepata koji pripadaju tim modelima. Takođe, svi metamodeli su pozicionirani kao M^2 modeli u okviru MDA arhitekture čime se stiče osnova za modelom vođen razvoj skladišta podataka. U ovom poglavlju će biti dat predlog metodološkog postupka razvoja skladišta podataka i ciljne arhitekture kao rezultata tako definisanog razvoja.

8.1 Pristup razvoju skladišta podataka

Postupak izgradnje skladišta podataka se izvodi kroz nekoliko koraka kao što je prikazano na Slici 66.

Početni korak podrazumeva prilagođavanje modela izvora za automatsku transformaciju u ciljni C-DV model. Prilagođavanje modela podrazumeva da se konkretan model (M^1) definiše konceptima odgovarajućeg jezika za opis (M^2), ukoliko takva specifikacija već ne postoji. Na taj način se vrši priprema za automatsku transformaciju Izvorni model → C-DV model. Za svaki novouvezeni model izvora se kreira odgovarajući konkretni C-DV model.

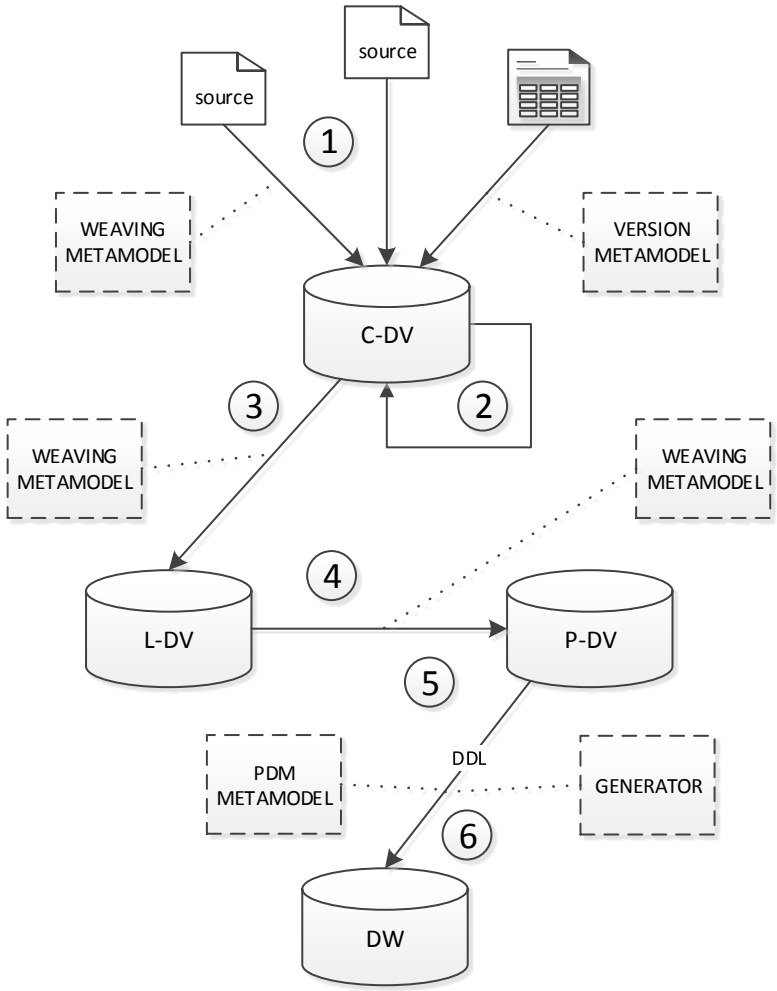
Ovaj korak se obavlja za svaku promenu strukture modela izvora i zavodi se kao nova verzija posmatranog C-DV modela.

Za ovaj korak se koriste, pored C-DV i izvornih metamodela, i sledeći modeli:

- Model preslikavanja koji opisuje pravila transformacije između koncepata kojima je izvorni model opisan i C-DV koncepata.
- Model verzionisanja koji skladišti različite verzije istih C-DV modela, uključivši i prvu, ukoliko je jedina.

Drugi korak je korisnički orijentisan kada projektant vrši tzv. konsolidaciju C-DV modela u različitim slučajevima, a pre svega:

- Identifikacija i označavanje onih koncepata realnog sistema koji su u više modela izvora opisani na različite načine (naziv, struktura itd.). Ovaj postupak se sprovodi kako bi u kasnijim preslikavanjima (u integrisanom L-DV modelu) predstavljali jedinstveni koncept.
- Identifikacija i promena podrazumevanih identifikatora (nastalih u procesu transformacije) u odgovarajuće poslovne ključeve na C-DV modelu, gde je to izvodljivo. U ostalim situacijama poslovni ključ ostaje kakav je nastao u procesu automatske transformacije.



Slika 66. Postupak izgradnje skladišta podataka

Treći korak predstavlja automatske PIM → PIM transformacije iz C-DV modela u integrisani L-DV. Za izvršavanje ovog koraka se, pored opisa u C-DV i L-DV metamodelima, koristi i Model preslikavanja u kojem su opisane transformacije između koncepata dva metamodela. Transformacije uključuju konsolidovane podatke iz prethodnog koraka. Na primer, hab koji je identifikovan kao već postojeći u nekom drugom modelu, neće se ponovo kreirati u rezultujućem integrisanom L-DV modelu.

Četvrti korak je korisnički orijentisan jer se u ovom koraku vrši konsolidacija vrednosnih domena iz integrisanog L-DV u odgovarajući P-DV model. Pojedine satelite je moguće preslikati u već postojeće vrednosne domene kako bi se izbegla redundansa podataka kao što je to opisano u odeljku *Postupak definisanja MDP* u poglavlju *Fizički Data Vault*. U okviru ovog koraka se koristi, pored metamodela L-DV i P-DV, i *Model preslikavanja* koji opisuje pravila transformacije ova dva modela.

Peti korak predstavlja automatske PIM → PIM transformacije iz L-DV modela u ciljni P-DV. Za izvršavanje ovog koraka se, pored opisa u L-DV i P-DV metamodelima, koristi i *Model preslikavanja* u kojem su opisane transformacije između koncepata dva metamodela. Transformacije konsultuju konsolidovane podatke iz prethodnog koraka. Na primer, satelit koji je predviđen da se preslika u već postojeći vrednosni domen, neće rezultovati kreiranjem novog vrednosnog domena, već će biti povezan sa već postojećim. Za sve preostale satelite se kreiraju odgovarajući vrednosni domeni.

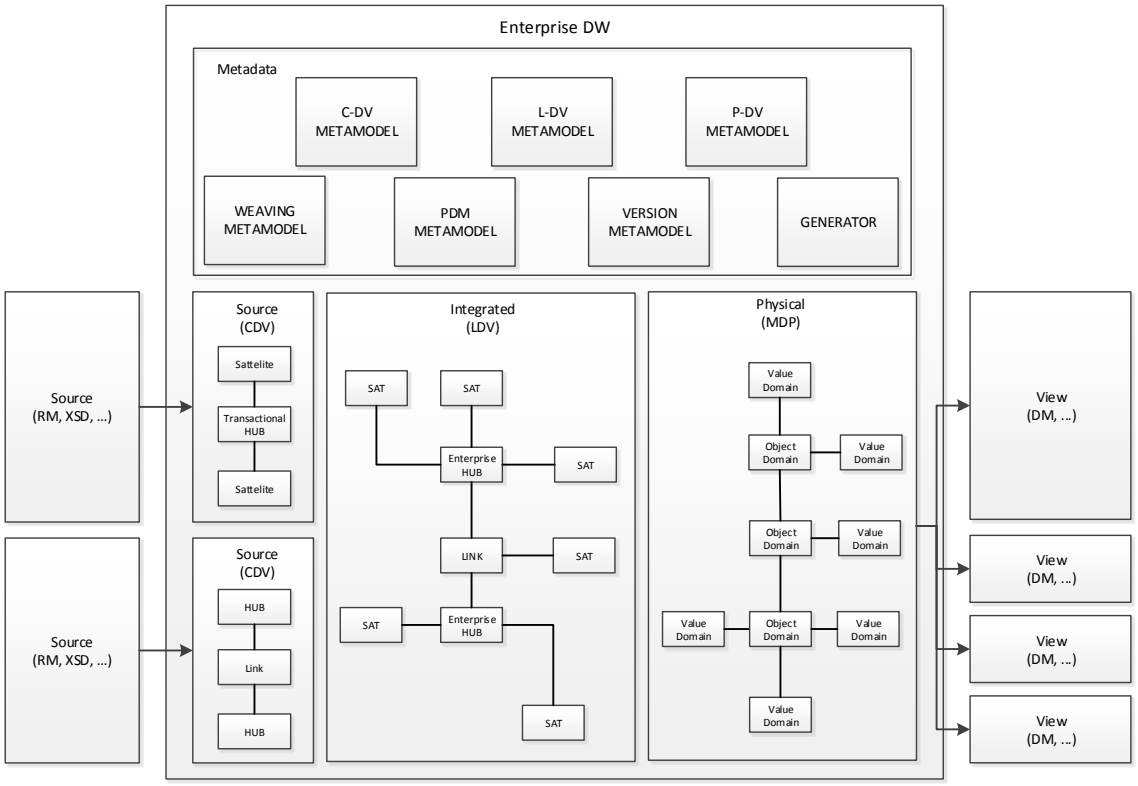
Šesti, i završni, korak predstavlja automatsku transformaciju iz P-DV modela u skup odgovarajućih DDL naredbi na ciljnoj tehnološkoj platformi. Za izvršenje ovog koraka, pored specifikacije konkretnog P-DV modela, koristi se i Model za opis tehnološke platforme i Generator koda.

Nakon ovih koraka korporativno skladište podataka u jedinstvenom, integrisanom modelu poseduje sve verzije izvora sa svim originalnim informacijama. U suštinski ELT pristupu, *Sloj skladišta podataka* u ovom trenutku pruža sve neophodne preduslove za definisanje *Sloja analize podataka* primenom nekog od standardnih i dobro ustanovljenih načina izgradnje *Centara podataka (eng. Data Marts).*

Postupak izgradnje skladišta podataka je vođeno, ali i rezultuje ciljnom arhitekturom koja je prikazana u sledećem poglavlju.

8.2 Arhitektura skladišta podataka

Kao što je prikazano na Slici 67, ciljna arhitektura je organizovana korišćenjem svih komponenti koje su prikazane u odeljku *Komponente logičke arhitekture skladišta podataka* iz poglavlja *Savremeni pristupi razvoju skladišta podataka.*



Slika 67. Arhitektura skladišta podataka

Sloj metapodataka se sastoji iz odgovarajućih metamodela koji opisuju modele izvora i modele konkretnog skladišta podataka. Konkretni modeli se definišu u prvom koraku kako je to opisano u prethodnom odeljku. Takođe je opisano i da se ovaj sloj dodatno definiše konsolidacijom M^1 modela, i to u koracima dva i četiri, kada se vrši usaglašavanje realnih objekata i konsolidacija preslikavanja, respektivno.

Sloj metapodataka takođe sadrži i opise predefinisanih preslikavanja, kako na M^2 nivou, tako i na nivou M^1, odnosno konkretnih modela poslovnog sistema.

Sloj pripreme podataka je odgovoran za održavanje modela izvora i njihovih verzija kako bi se izvršilo praćenje promene strukture izvora. Na taj način se priprema podataka vrši isključivo na „meta" nivou, tj. održavanjem C-DV modela. Kao opcija, postoji mogućnost da se zbog performanse i operativnosti izvora podaci prvo učitaju u originalnoj strukturi i formatu. U oba slučaja se učitavanje podataka vrši tek nakon izvršavanja svih šest koraka iz prethodnog odeljka.

Sloj skladišta podataka se izgrađuje nad odgovarajućim integrisanim pogledom nad izvorima podataka, tj. L-DV modelom i pripadajućim P-DV modelom. U njemu je sadržano celokupno korporativno skladište podataka i sve verzije struktura modela izvora i pripadajućih podataka. Ovaj sloj se izgrađuje izvršavanjem trećeg, četvrtog, petog i šestog koraka, kako je to opisano u prethodnom odeljku.

Sloj analize podataka obuhvata sve centre podataka koji su izgrađeni nad podacima koji se nalaze na Sloju skladišta podataka.

U narednom odeljku će kroz odgovarajući primer biti detaljnije prikazan postupak izgradnje različitih slojeva ciljne EDW arhitekture.

8.3 Implementacija

Modelom vođen razvoj skladišta podataka predstavljen u prethodnom poglavlju zahteva odgovarajuće konceptualno i tehnološko okruženje bazirano na MDA arhitekturi. U ovom poglavlju će biti predstavljeno tehnološko okruženje za kreiranje, skladištenje i održavanje C-DV, L-DV i P-DV metamodela i transformaciju PIM modela koji su njima specifikovani do nivoa izvršivog skladišta podataka.

U drugom delu poglavlja će biti predstavljen jednostavan primer realizacije transformacija izvornog modela do nivoa DDL naredbi sa ciljem verifikacije i validacije predloženog pristupa razvoju skladišta podataka.

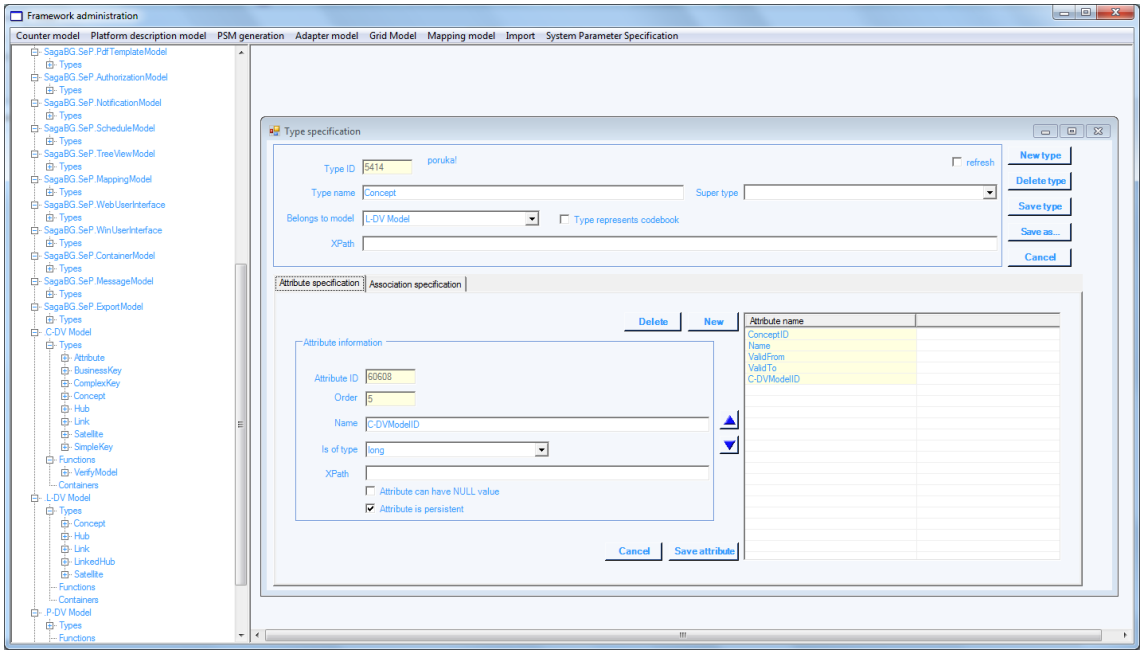
8.4 Implementaciono okruženje – DDF

Pristup razvoju skladišta podataka predstavljen u prethodnim poglavljima za svoju realizaciju zahteva implementaciono okruženje zasnovano na MDA arhitekturi. Za demonstraciju modelom vođenog razvoja skladišta podataka zasnovanog na Data Vault pristupu je korišćen DDF (eng. *Domain Description Framework*) alat, razvijen u kompaniji Saga d.o.o Beograd, koji je primenjen nad više desetina softverskih rešenja realizovanih MDD pristupom.

DDF [108] [109] omogućava formalnu specifikaciju poslovnih i tehničkih modela i ciljne tehnološke platforme. Na osnovu te specifikacije omogućena je automatska transformacija i dobijanje izvršivih delova novog sistema.

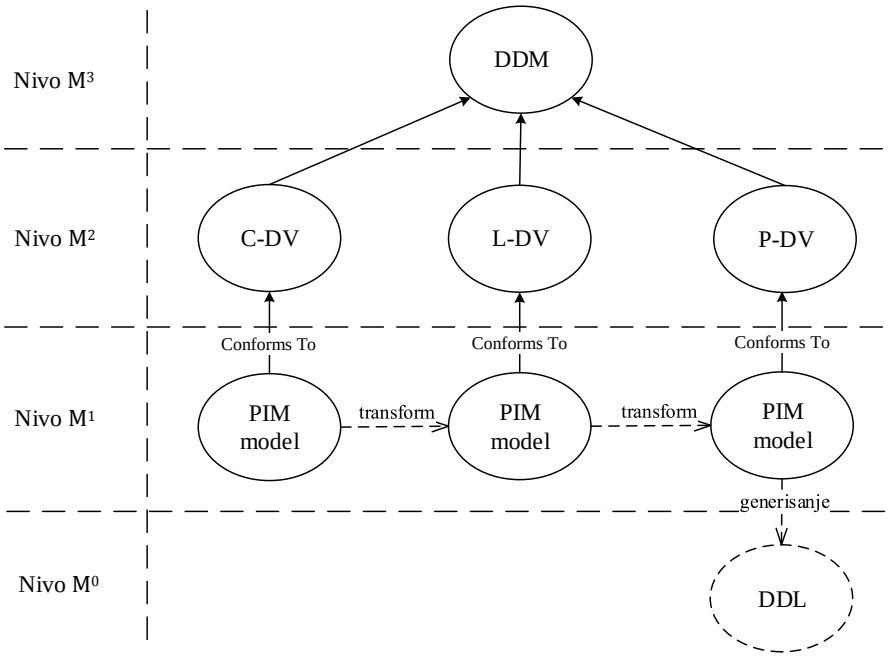
DDF okruženje se sastoji iz četiri hijerarhijska nivoa modela, pri čemu model na višem nivou hijerarhije predstavlja metamodel koji opisuje modele na nižem nivou. Prva tri nivoa arhitekture se opisuju i održavaju kroz Rečnik IS, dok nivo konkretnih instanci predstavlja izvršivi sistem.

Kao priprema za demonstraciju, u DDF repozitorijumu modela je izvršeno opisivanje C-DV, L-DV i P-DV metamodela, prema specifikacijama na slikama 25, 50 i 56, respektivno. Za uvođenje ovih metamodela je korišćena pripadajuća aplikacija za administraciju i održavanje DDF repozitorijuma modela kao što je prikazano na Slici 68.



Slika 68. Administracija DDF

Na ovaj način je pripremljeno odgovarajuće okruženje za demonstraciju modelom vođenog pristupa razvoju skladišta podataka, kao što je prikazano na Slici 69.

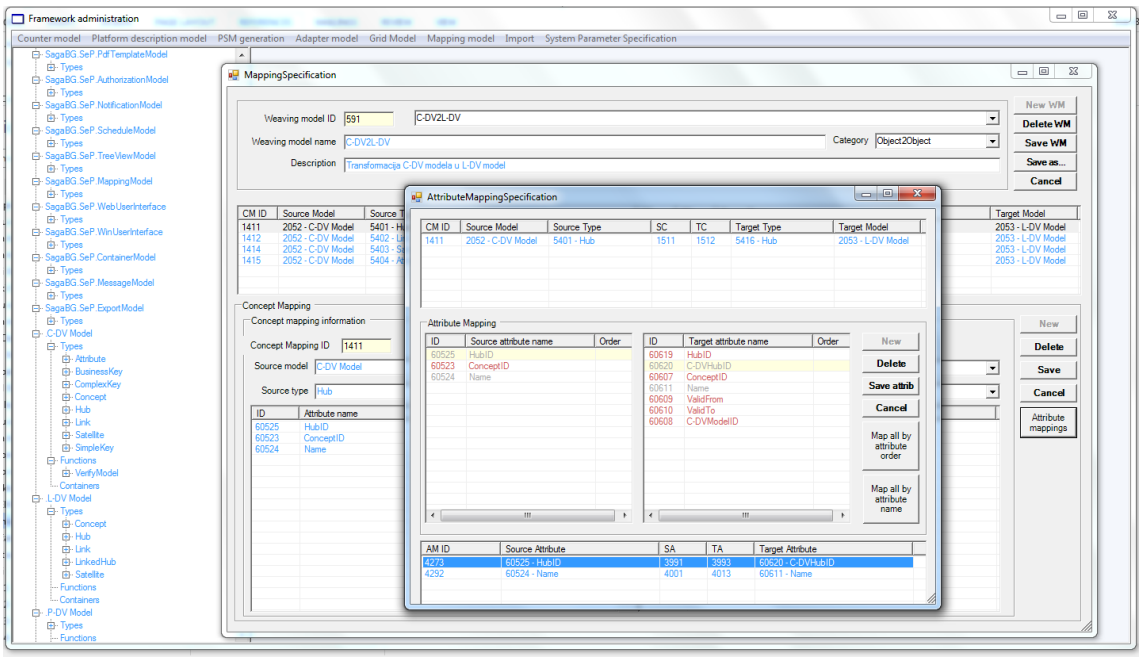


Slika 69. DDF sa definisanim C-DV, L-DV i P-DV metamodelima

To praktično znači da definisani modeli postaju standardni M2 modeli pogodni da se pomoću njih opišu odgovarajući poslovni (PIM) modeli koji predstavljaju modele izvora podataka.

Usvojeni meta metamodel za opis ovih metamodela je DDM (eng. *Domain Description Model*). DDM [108] omogućava opis poslovnih i tehničkih M^2 modela nekog sistema i njihovih međusobnih veza, kao i tehnički domen u kojem će sistem biti realizovan. Posmatrano iz ugla MDA, omogućene su PIM → PIM transformacije između korišćenih modela, kao i PIM → PSM i PSM → kod, pri čemu se može iz jednog PIM modela dobiti više različitih PSM modela, odnosno, jedan PSM model moguće je transformisati u više različitih tehnoloških okruženja. Ove transformacije su osnova generisanja izvršivog sistema iz date specifikacije.

Pored toga, u DDF okruženju su definisana pravila transformacije kroz pripadajući *Model preslikavanja* u administraciji DDF, kao što je prikazano na Slici 70.



Slika 70. Definisanje pravila transformacije

Takođe, u demonstraciji su korišćeni već razvijeni modeli i pripadajući infrastrukturni elementi, kao što su softverske komponente, aplikativni okviri i generator. Korišćene komponente i modeli DDF alata su detaljnije predstavljeni u pratećim prilozima.

8.5 Realizacija skladišta podataka

Celokupna realizacija skladišta podataka se izvodi prema opisanoj proceduri u odeljku *Pristup razvoju skladišta podataka* poglavlja *Modelom vođen razvoj skladišta podataka.*

U ovom odeljku će se izvršiti prikaz pojednostavljenog postupka radi demonstracije i potvrde ispravnosti definisanog pristupa i procedure razvoja.

Primer koji će biti prikazan je podmodel preuzet sa modela prikazanog na Slici 40, koji se sastoji iz koncepata: Tip proizvoda, Proizvod, Asortiman i Preduzeće, i on je prikazan u XML formatu na Slici 71.

```
<CDVMODEL>
  <HUB id="101" name="PROIZVOD">
    <BUSINESSKEY type="long" name="SIFPRO"/>
    <SATELLITE name="PROIZVOD">
      <ATTRIBUTE type="string" name="OPISPRO"/>
    </SATELLITE>
  </HUB>
  <HUB id="102" name="TIPPROIZVODA">
    <BUSINESSKEY type="string" name="SIFTIPPRO"/>
    <SATELLITE name="TIPPROIZVODA">
      <ATTRIBUTE type="string" name="NAZIVTIPPRO"/>
    </SATELLITE>
  </HUB>
  <HUB id="103" name="PREDUZECE">
    <BUSINESSKEY type="long" name="SIFPRED"/>
    <SATELLITE name="PREDUZECE">
      <ATTRIBUTE type="string" name="NAZIVPRED"/>
    </SATELLITE>
  </HUB>
  <LINK id="104" name="JEKATEGORIJE" type="association">
    <HUBID>102</HUBID>
    <HUBID>102</HUBID>
  </LINK>
  <LINK id="105" name="ASORTIMAN" type="association">
    <HUBID>101</HUBID>
    <HUBID>103</HUBID>
  </LINK>
  <LINK id="106" name="JETIPA" type="association">
    <HUBID>101</HUBID>
    <HUBID>102</HUBID>
  </LINK>
</CDVMODEL>
```

Slika 71. Izvorni podmodel u XML formatu

Nakon importa ulaznog podmodela, rezultujući model se skladišti u odgovarajućem C-DV repozitorijumu, kao što je prikazano na Slici 72.

Model

ModelID	Name
1001	Demo

Concept

ConceptID	Name
101	PROIZVOD
102	TIPPROIZVODA
103	PREDUZECE
104	JEKATEGORIJE
105	ASORTIMAN
106	JETIPA

Hub

HubID
101
102
103

Link

LinkID	Type
104	association
105	association
106	association

Satellite

SatelliteID	Name	ConceptID
101	PROIZVOD	101
102	TIPPROIZVODA	102
103	PREDUZECE	103

Attribute

AttributeID	Name	SatelliteID
101	SIFPRO	
102	OPISPRO	101
103	SIFTIPPRO	
104	NAZIVTIPPRO	102
105	SIFPRED	
106	NAZIVPRED	103

BusinessKey

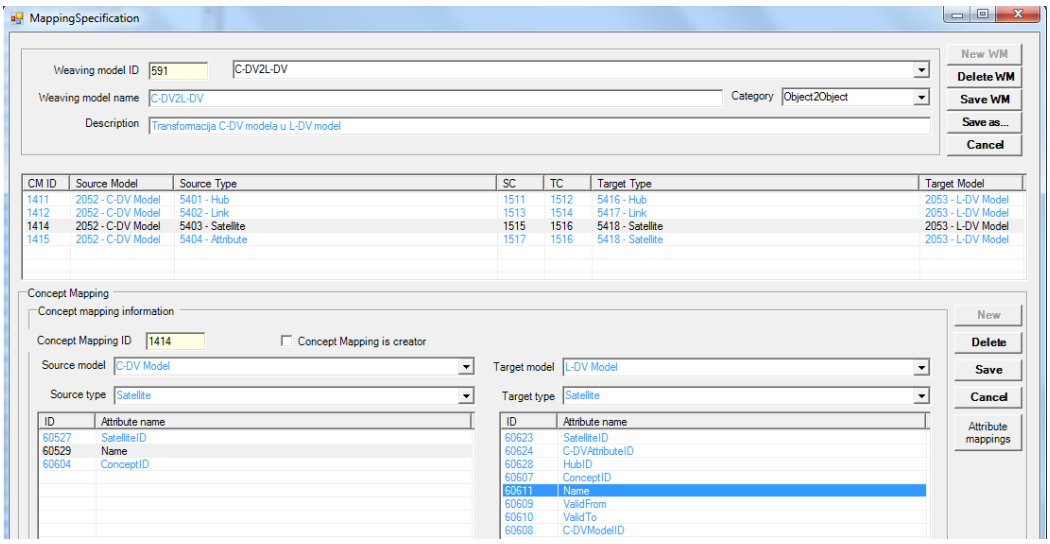
BusinessKeyID	HubID
101	101
103	102
105	103

SimpleKey

SimpleKeyID
101
103
105

Slika 72. Ciljni C-DV model

Sledeća transformacija je C-DV → L-DV zasnovana na skupu pravila kao što je prikazano na Slici 73.



Slika 73. Transformacije C-DV → L-DV

Ovako definisana transformacija rezultuje u zasnivanju integrisanog modela EDW u notaciji L-DV metamodela, koji je prikazan na Slici 74.

Model

ModelID	Name
1001	Demo

Concept

ConceptID	Name	validFrom	validTo	CDVModelID
101	PROIZVOD	5.1.2016.	null	1001
102	TIPPROIZVODA	5.1.2016.	null	1001
103	PREDUZECE	5.1.2016.	null	1001
104	JEKATEGORIJE	5.1.2016.	null	1001
105	ASORTIMAN	5.1.2016.	null	1001
106	JETIPA	5.1.2016.	null	1001
107	SIFPRO	5.1.2016.	null	1001
108	OPISPRO	5.1.2016.	null	1001
109	SIFTIPPRO	5.1.2016.	null	1001
110	NAZIVTIPPRO	5.1.2016.	null	1001
111	SIFPRED	5.1.2016.	null	1001
112	NAZIVPRED	5.1.2016.	null	1001

Hub

HubID	CDVHubID
101	101
102	102
103	103

Link

LinkID	CDVLinkID
104	104
105	105
106	106

Satellite

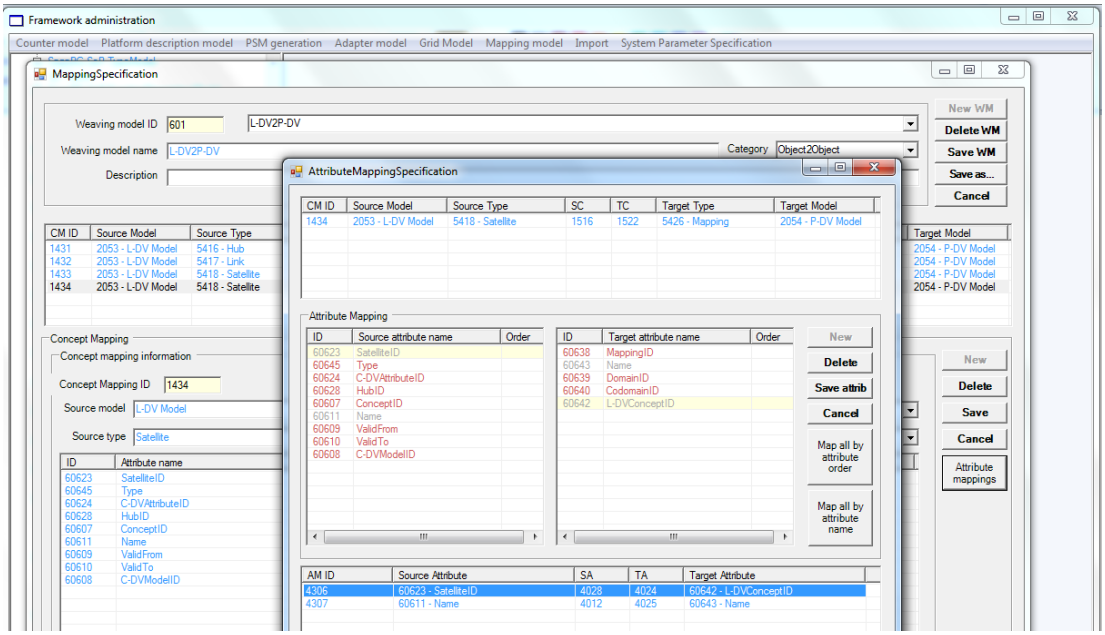
SatelliteID	CDVAttributeID	HubID
107	101	101
108	102	101
109	103	102
110	104	102
111	105	103
112	106	103

LinkedHub

LinkedHubID	LinkID	HubID
101	104	102
102	104	102
103	105	103
104	105	101
105	106	101
106	106	102

Slika 74. Ciljni L-DV model

U sledećem koraku, vrši se transformacija iz skladištenog L-DV modela u ciljni P-DV model prema definisanim pravilima transformacije koja su prikazana na Slici 75.



Slika 75. Transformacije L-DV → P-DV

Rezultujući P-DV model je prikazan na Slici 76, gde su prikazani svi oformljeni objektni i vrednosni domeni, kao i koncepti preslikavanja koji ih povezuju.

Domain

DomainID	Name
101	PROIZVOD
102	TIPPROIZVODA
103	PREDUZECE
104	SIFRE
105	NAZIVI
106	OPISIPRO

ObjectDomain

Object DomainID	LDVConceptID
101	101
102	102
103	103

ValueDomain

Value DomainID	Type
104	string
105	string
106	string

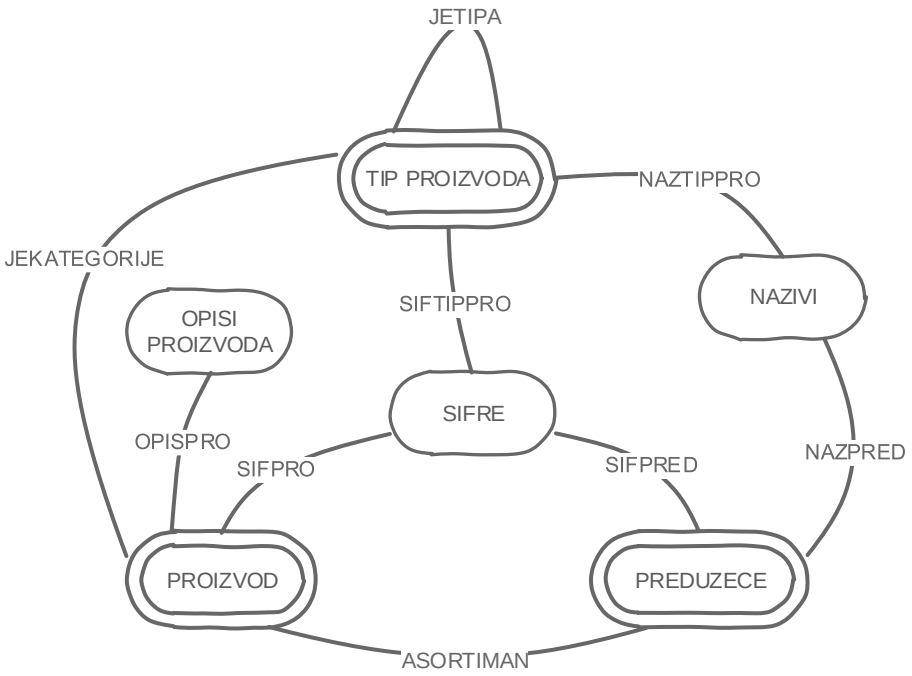
Mapping

MappingID	Name	DomainID	CodomainID	LDVConceptID
101	JEKATEGORIJE	101	102	104
102	ASORTIMAN	101	103	105
103	JETIPA	102	102	106
104	SIFPRO	101	104	107
105	OPISPRO	101	106	108
106	SIFTIPPRO	102	104	109
107	NAZIVTIPPRO	102	105	110
108	SIFPRED	103	104	111
109	NAZIVPRED	103	105	112

Slika 76. Ciljni P-DV model

Domeni su prikazani ne samo kao rezultat automatske transformacije, već nakon konsolidacije vrednosnih domena. Radi demonstracije, sve šifre su stavljene u isti vrednosni domen (*SIFRE*), slično kao i nazivi proizvoda, tipova proizvoda i preduzeća (*NAZIVI*), što ne mora biti slučaj u realnoj situaciji.

Na sledećoj slici (Slika 77) je dat grafički prikaz rezultujućeg modela u MDP notaciji.

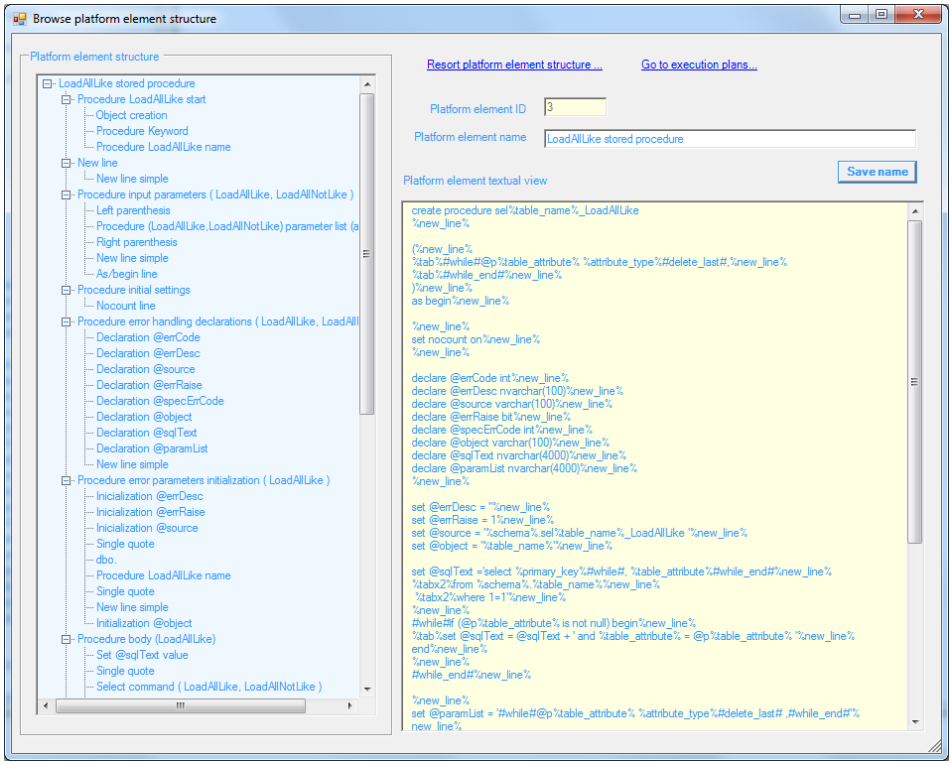


Slika 77. MDP rezultujući model

Kao što je prikazano, Objektni domeni predstavljaju objekte realnog sistema, Preslikavanja igraju ulogu atributa ili veza između objekata, dok su Vrednosni domeni skupovi vrednosti iz kojih atributi uzimaju odgovarajuću vrednost.

Ovako pripremljen model je spreman za finalnu PIM → PSM transformaciju u ciljno tehnološko okruženje. Za ovaj korak demonstracije je korišćen SQL Server SUBP koji je opisan u *Modelu za opis tehnološke platforme*. Rezultujuće DDL naredbe su proizvod generisanja odgovarajućeg PSM modela na osnovu ulaznog PIM modela i odabranog tehnološkog okruženja, kao što je opisano u Prilogu 4.

Jedna od definicija za SQL Server platformu je prikazana na Slici 78.



Slika 78. Primer opisa tehnološke platforme

Rezultat finalnog koraka, odnosno generisanja DDL naredbi, je prikazan na Slici 79, gde je dat po jedan primer naredbe za svaki koncept MDP modela (Objektni domen, Preslikavanje, Vrednosni domen).

```
CREATE TABLE dbo.Proizvod -- OBJECT DOMAIN
(
	ProizvodID bigint NOT NULL PRIMARY KEY,
	ValidFrom datetime NOT NULL,
	ValidTo datetime NULL,
	TransactionTime datetime NOT NULL
);
CREATE TABLE dbo.Sifre -- VALUE DOMAIN
(
	SifreID bigint NOT NULL PRIMARY KEY,
	Value nvarchar(max) NOT NULL
);
CREATE TABLE dbo.SifPro -- MAPPING
(
	SifProID bigint NOT NULL,
	ProizvodID bigint NOT NULL,
	SifreID bigint NOT NULL,
	ValidFrom datetime NOT NULL,
	ValidTo datetime NULL,
	ModelVersionID bigint NOT NULL,
	TransactionTime datetime NOT NULL
);
```

Slika 79. Rezultujuće SQL Server DDL naredbe

Kao što je prikazano, implicitno se Objektnom domenu i Preslikavanju dodaje vremenska dimenzija kroz kolone koje predstavljaju vreme važenja i vreme ažuriranja.

Preslikavanje se takođe proširuje sa dodatnom kolonom *ModelVersion* kako bi se moglo odrediti po kojem modelu izvora potiče unešena vrednost veze ili atributa.

Vrednosni domen je uvek tipa string kako bi se mogle upisati i nevalidne vrednosti koje egzistiraju u izvorima podataka, a pravi tip kojima ove vrednosti pripadaju se održavaju na metamodelu.

9 ZAKLJUČAK

Osnovni cilj ovog rada je formalizacija empirijskog Data Vault modela i definisanje odgovarajućeg okruženja za primenu modelom vođenog pristupa razvoju skladišta podataka kako bi se savladale neusaglašenosti razvoja skladišta podataka opisane u uvodnom poglavlju ovog rada.

U ovom radu su prikazani postojeći savremeni konceptualni modeli i modeli podataka koji se koriste u razvoju skladišta podataka. Pored toga, izvršena je detaljna analiza modela strukture i razmatrane su njihove prednosti i nedostaci.

Na osnovu tih zapažanja i definisanja okruženja za razvoj skladišta podataka, u radu je pokazano da je:

- Moguće definisati odgovarajući metamodel (C-DV) koji može da opiše semantiku raznorodnih modela izvora podataka iskazanu kroz različite metamodele (UML Dijagram klasa, Model objekti-veze, Relacioni model...) koji je zasnovan na postojećim konceptima empirijskog Data Vault modela.
- Moguće definisati automatske transformacije iz modela izvora podataka u definisani C-DV model bez gubitka informacija.
- Moguće definisati metamodel (L-DV) koji će moći da podrži strukturne i promene na meta nivou koje se vremenom dešavaju na modelima izvora podataka, kao i da podrži praćenje podataka do njihovog izvora, u vremenskoj i prostornoj dimenziji.
- Moguće da L-DV bude definisan kao platformski nezavisan i kao formalno proširenje standardnog metamodela za opis skladišta podataka CWM.
- Moguće definisati automatske transformacije iz izvornog C-DV metamodela u ciljni L-DV metamodel, kao neophodan korak usklađivanja sa konceptima na izvršnoj tehnološkoj platformi, prvenstveno sistemima za upravljanje relacionim bazama podataka.

- Moguće definisati metamodel (P-DV) koji će opisati fizičku arhitekturu skladišta podataka i detalje ciljne tehnološke platforme.

Prikazani postupak realizacije skladišta podataka predstavlja u mnogim segmentima potpuno inovativan metod i način razvoja skladišta podataka. Ovakav metod se može direktno primenjivati u procesu razvoja skladišta podataka čime uklanja nedostatke postojećih pristupa i olakšava projektovanje i razvoj skladišta podataka.

Definisani modeli i pravila transformacije pružaju snažnu podršku prevazilaženju osnovnih problema razvoja skladišta podataka koji su predstavljeni u ovom radu, a pre svega je dato: ujednačavanje semantičkih različitosti konceptualnih modela za opis, jedinstveni jezik za opis skladišta podataka, pružanje integrisanog pogleda na celokupno skladište podataka, apsorbovanje promena strukture koje nastaju u izvorima podataka, praćenje kompletnih podataka za sve izvore u svim periodima na jedinstven način, zatim praćenje podataka do njihovog izvora, kao i praćenje vremenske i prostorne dimenzije podataka.

Dalji pravci istraživanja koji proizilaze iz ovog rada se mogu podeliti na dve grupe: istraživanje vezano za osnovnu tezu rada (poboljšanja postojećeg rešenja i proširenja grupe ciljeva), i istraživanje koje se odnosi na uočene probleme u primenjenoj metodologiji i arhitekturi.

U okviru prve grupe izdvajaju se:

(i) definisanje pravila preslikavanja L-DV modela u odgovarajući DFM model,
(ii) proširenje definisanog P-DV modela i
(iii) potpuna integracija definisanih metamodela (C-DV, L-DV i P-DV) u CWM standard.

Definisanje L-DV → DFM pravila preslikavanja bi omogućilo zaokruživanje procesa definisanja skladišta podataka zaključno sa *Slojem analize podataka*, čime bi se primena ovog pristupa sprovela na sve slojeve skladišta podataka.

Proširenje definisanog P-DV modela bi omogućilo specifikaciju i održavanje svih aspekata fizičkog nivoa skladišta podataka.

Potpuna integracija definisanih metamodela (C-DV, L-DV i P-DV) u opšteprihvaćeni CWM standard zahteva definisanje XMI interfejsa čime bi se omogućila šira praktična primena ovog pristupa.

Drugu grupu pravaca istraživanja predstavljaju izmene i prilagođavanja metodološkog pristupa i predložene arhitekture, kao i svih prikazanih metamodela i pravila transformacije, nakon primene ovog pristupa u realnim uslovima. Cilj ove grupe pravaca istraživanja je definisanje jedne potpune metodologije razvoja primenljive u svim fazama životnog ciklusa razvoja skladišta podataka.

LITERATURA

[1] W. Inmon. *Building the Data Warehouse*. Wiley, 2002.

[2] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*. Wiley, 1998.

[3] R. Damhof, *The next generation EDW*, available at: prudenza.typepad.com/files/damhof_dbm0508_eng-1.pdf, avgust 2008.

[4] Object Management Group. *Common Warehouse Metamodel (CWM)*, available at: http://www.omg.org/cgi-bin/doc?formal/03-03-02.pdf, 2002.

[5] O. Regardt et al. *Anchor Modeling An Agile Modeling Technique using the Sixth Normal Form for Structurally and Temporally Evolving Data,* ER 2009 [Lecture Notes in Computer Science, vol. 5829, no.1, pp. 234-250], Gramado, Brazil, 2009.

[6] D. Linstedt. *Data Vault Modeling Specification v1.0.9.* available at: http://danlinstedt.com/datavaultcat/standards/dv-modeling-specification-v1-0-8/, 2010.

[7] V. Jovanović, I. Bojičić. *Conceptual Data Vault Model*. SAIS Conference 2012 Atlanta, USA, pp. 131-136

[8] V. Jovanović et al. *Persistent Staging Area Models for Data Warehouses*. IACIS, Issues in Information Systems, South Carolina, 2012.

[9] M. Golfarelli, S. Rizzi. *Data Warehouse Design – Modern Principles and Methodologies*. McGraw-Hill, 2009.

[10] E. Malinowski, E. Zimanyi. *Advanced Data Warehouse Design – From Conventional to Spatial and Temporal Applications*. Springer-Verlag Berlin Heidelberg, 2008.

[11] W. Inmon, D. Strauss, G. Neushloss. *DW 2.0 - The Architecture for the next generation of data warehousing*, Morgan Kaufman, 2008.

[12] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley, second edition, 2002.

[13] M. Golfarelli, D. Maio, S. Rizzi. *The dimensional fact model: a conceptual model for data warehouses,* International Journal of Cooperative Information Systems, Vol.7, Nos. 2 and 3 pp. 215-247., 1998.

[14] A. Abello, J. Samos, F. Saltor. *YAM²: A multidimensional conceptual model extending UML*. Information Systems, 32(6) : 541-567, 2006.

[15] S. Lujan-Mora, J. Trujillo, I. Song. *A UML profile for multidimensional modeling in data warehouses.* Data & Knowledge Engineering, 59(3):725-769, 2006.

[16] T.B. Nguyen, A.M. Tjoa, R. Wagner. *An Object Oriented Multidimensional Data Model for OLAP.* In Proceedings 1st International Conference on Web-Age Information Management, Shanghai, pp. 59-82, 2000.

[17] E. Franconi, U. Sattler. *A data warehouse conceptual model for multidimensional aggregation*. In Proceedings 1st International Workshop on Design and Management of Data Warehouses, Heidelberg, Germany, 1999.

[18] C. Sapia, M. Blaschka, G. Hofling, B Dinter. *Extending the E/R model for the multidimensional paradigm* In Proceedings International Conference on Conceptual Modeling, Singapore, 1998.

[19] N. Tryfona, F. Busborg, J.G. Borch Christiansen. *starER: A Conceptual Model for Data Warehouse Design* In Proceedings ACM International Workshop on Data Warehousing and OLAP, pp. 3-8, 1999.

[20] F. Jouault, J. Bezivin. *KM3 - a DSL for Metamodel Specification*. In proceedings of 8th FMOODS, pp. 171-185, Springer, 2006.

[21] Object Management Group. *Model Driven Architecture.*

available at: http://www.omg.org/cgi-bin/doc?omg/00-11-05, 2000.

[22] Object Management Group. *MDA Guide v1.0.1.*

available at: http://www.omg.org/cgi-bin/doc?omg/03-06-01, 2003.

[23] Object Management Group. *Common Warehouse Metamodel.*

available at: http://www.omg.org/spec/CWM/1.1/, 2003.

[24] Object Management Group. *Meta Object Facility (MOF) v1.4.*

available at: http://www.omg.org/spec/MOF/2.4.1/, 2011.

[25] D. Mihajlović. *Metodologija naučnih istraživanja*. Fakultet organizacionih nauka, Beograd, 1999.

[26] D. Cvetković, S. Simić. *Diskretna matematika - matematika za kompjuterske nauke.* Prosveta, Niš, 1996.

[27] D. Linstedt. *Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault*, CreateSpace Independent Publishing Platform, 2011.

[28] D. Linstedt, K. Graziano, H. Hulgeren. *The New Business Supermodel*. Lulu.com, 2008.

[29] D. Linstedt. *Data Vault Series 1 - Data Vault Overview.*

available at: http://www.tdan.com/view-articles/5054/, 2002.

[30] D. Linstedt. *Data Vault Series 2 - Data Vault Components.*

available at: http://www.tdan.com/view-articles/5155/, 2003.

[31] D. Linstedt. *Data Vault Series 3 - End Dates and Basic Joins.*

available at: http://www.tdan.com/view-articles/5067/, 2003.

[32] D. Linstedt. *Data Vault Series 4 – Link Tables.*

available at: http://www.tdan.com/view-articles/5172/, 2004.

[33] D. Linstedt. *Data Vault Series 5 – Loading Practices.*

available at: http://www.tdan.com/view-articles/5285/, 2005.

[34] M. Casters, R. Bouman, J. Dongen. *Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration*. Wiley, 2010.

[35] C. Knowles, *6NF Conceptual Models and Data Warehousing 2.0*, SAIS Conference 2012, Atlanta, USA, pp160-165, 2012.

[36] L. Ronnback et al. *Anchor modeling –Agile information modeling in evolving data environments*. Data & Knowledge Engineering, Volume 69, Issue 12, pp. 1229-1253, 2010.

[37] B. Inmon. *Data Warehousing 2.0 - Modeling and Metadata Strategies for Next Generation Architectures.* White Paper, available at: http://www.devgear.co.kr/pdf/bill-inmon-data-warehousing-2-0-whitepaper.pdf, 2010.

[38] R. Kimball, L. Caserta. *The Data Warehouse ETL Toolkit – Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data.* Wiley, 2004.

[39] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley, Third edition, 2013.

[40] P.S. Chen. *The Entity-Relationship Model: Toward a unified view of Data*. ACM TODS, 1(1):9-36, 1976.

[41] Object Management Group. *Unified Modeling Language (UML), V2.4.1*

available at: http://www.omg.org/spec/UML/2.4.1/, 2011.

[42] P. Janičić. *Matematička logika u računarstvu*. Matematički fakultet, Univerzitet u Beogradu, četvrto elektronsko izdanje, Beograd, 2008.

[43] P. Janičić, M. Nikolić. *Veštačka Inteligencija*. Matematički fakultet, Univerzitet u Beogradu, elektronsko izdanje, Beograd, 2010.

[44] J. Poole et al. *Common Warehouse Metamodel*. Wiley, 2002.

[45] J. Poole et al. *Common Warehouse Metamodel Developer's Guide*. Wiley, 2003.

[46] Object Management Group. *Common Warehouse Metamodel (CWM) Volume 2. Extensions*, 2001.

[47] J. Cahoon. *Fast Development of a Data Warehouse using MOF, CWM, and Code Generation*. available at: http://www.cubemodel.com, 2006.

[48] V. Stefanov, B. List. *A UML Profile for Modeling Data Warehouse Usage*. ER'07 Proceedings of the 2007 conference on Advances in conceptual modeling: foundations and applications, 2007.

[49] J.N. Mazon, J. Trujillo. *An MDA approach for the development of data warehouses*. Decision Support Systems, Volume 45, Issue 1, pp. 41-58, 2008.

[50] S. Rizzi. *Conceptual modeling solutions for the data warehouse.* in: R.Wrembel, C. Koncilia (Eds.), DataWarehouses and OLAP: Concepts, Architectures and Solutions, Idea Group, 2006.

[51] B. List, R. Bruckner, K. Machaczek, J. Schiefer. *A comparison of data warehouse development methodologies: case study of the process warehouse.* Proceedings DEXA, 2002, pp. 203–215, 2002.

[52] A. Sen, A. Sinha. *A Comparison of Data Warehousing Methodologies*. Communications of the ACM, Volume 48 Issue 3, pp. 79-84, March 2005.

[53] D. Pan. *Metadata Version Management for DW 2.0 Environment*. In Proceedings of JCIT, 54-60, 2010.

[54] P. Vassiliadis. *Data Warehouse Modeling and Quality Issues*. PhD Thesis. Knowledge and DataBase Systems Laboratory, Dept. of Electrical and Computer Engineering, National Technical University of Athens, June 2000.

[55] B. Bebel et al. *Creation and Management of Versions in Multiversion Data Warehouse*. Proceedings of the 2004 ACM symposium on Applied computing, pp. 717-723, 2004.

[56] M. Golfarelli, J. Lechtenbörger, S. Rizzi, G. Vossen. *Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation*. Data Knowl. Eng. 59(2): 435-459, 2006.

[57] M. Blaschka. FIESTA: *A Framework for Schema Evolution in Multidimensional Databases*. PhD Thesis, Technischen Universität München, 2000.

[58] R. France, B. Rumpe. Model-driven Development of Complex Software: A Research Roadmap. In Proceedings FOSE '07 Future of Software Engineering, pp. 37-54, 2007.

[59] P. Vassiliadis, *Data Warehouse Metadata*. In Encyclopedia of Database Systems, Springer, 2009.

[60] W.H. Inmon. *Metadata in the Data Warehouse.*

http://www.billinmon.com//library/whiteprs/earlywp/ttmeta.pdf

[61] W.H. Inmon. *Metadata in the Data Warehouse: A Statement of Vision.*

http://www.billinmon.com/library/whiteprs/techtopic/tt10.pdf

[62] L. Carneiro, A. Brayner. *X-META: A Methodology for Data Warehouse Design with Metadata Management*. In Proceedings of the 4th Intl. Workshop on Design and Management of Data Warehouses, DMDW'02, 2002.

[63] T. Vetterli, A. Vaduva, M. Staudt. *Metadata standards for data warehousing: open information model vs. common warehouse metadata*. ACM SIGMOD, Volume 29 Issue 3, pp 68-75, 2000.

[64] V. Stefanov, B. List. *Business Metadata for the Data Warehouse: Weaving Enterprise Goals and Multidimensional Models*. EDOCW'06, 10th IEEE International Enterprise Distributed Object Computing Conference Workshops, pp.20, 2006.

[65] E. Malinowski, E. Zimanyi. *A conceptual model for temporal data warehouses and its transformation to the ER and the object-relational models*. Data & Knowledge Engineering, Volume 64 Issue 1, pp. 101-133, January, 2008.

[66] H.-C. Wei, R. Elmasri. *Schema versioning and database conversion techniques*. Annals of Mathematics and Artificial Intelligence 30: 23–52, Netherland, 2000.

[67] J. Eder, C. Koncilia & T. Morzy. *The COMET Metamodel for Temporal Data Warehouses*. Proceedings of the 14th International Conference on Advanced Information Systems Engineering (Caise 2002), Toronto, Canada, LNCS 2348, pp 83-99, 2002.

[68] R. Wrembel, B. Bebel. *Metadata Management in a Multiversion Data Warehouse*. Journal on data semantics VIII, pp. 118-157., Springer, 2007.

[69] X. Zhao, Z. Huang. *A Formal Framework for Reasoning on Metadata Based on CWM*. In: ER, Vol. 4215 Springer, p. 371-384, 2006.

[70] R. Wall. *ScaDaVer – A Framework for Schema and Data Versioning in OLTP Databases.* Research Prospectus, Department of Computer Science, Montana State University, 2011.

[71] S. Lujan-Mora, J. Trujillo. *Physical Modeling of Data Warehouses using UML*. DOLAP'04, Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, pp. 48-54, 2004.

[72] Z. Akkaoui, E. Zimanyi, J.N. Mazon, J. Trujillo. *A model-driven framework for ETL process development*. DOLAP'11, Proceedings of the 14th ACM international workshop on Data warehousing and OLAP, pp. 45-524, 2011.

[73] M. Golfarelli, S. Rizzi. *Temporal Data Warehousing: Approaches and Techniques*. In Integrations of Data Warehousing, Data Mining and Database - Innovative Approaches, Information Science Reference, 2011.

[74] T. Martyn. *Reconsidering Multi-Dimensional Schemas*. SIGMOD Record, Volume 33 Issue 1, pp. 83-88, March 2004.

[75] P. Vassilliadis, T. Sellis. *A survey of logical models for OLAP databases*. SIGMOD Record, Volume 28, Issue 4, pp. 64-69, 1999.

[76] G. Viswanathan, M. Schneider. *On the requirements for user-centric spatial data warehousing and SOLAP*. DASFAA'11 Proceedings of the 16th international conference on Database systems for advanced applications, pp. 144-155, 2011.

[77] B. Devlin. *Data Warehouse: From Architecture to Implementation*. Addison Wesley, 1997.

[78] S. Rizzi. *Data Warehouse. Wiley Encyclopedia of Computer Science and Engineering*, B. Wah (Ed.), John Wiley and Sons, 2008.

[79] T. Ariyachandra, H Watson. *Key organizational factors in data warehouse architecture selection*. Decision Support Systems, 49, 200-212, 2010.

[80] R. Jindal, A. Acharya, *Federated data warehouse architecture*, White Paper for ITtoolbox Data Warehouse, 2008. Available at: http://hosteddocs.ittoolbox.com/Federated%20data%20Warehouse%20Architecture.pdf

[81] Meta Data Coalition, *Meta Data Interchange Specification (MDIS)*, Version 1.1, 1997.

[82] M. Staudt, A. Vaduva, T. Vetterli. *Metadata management and data warehousing*. Technical Report 21, Swiss Life, Information Systems Research, July 1999.

[83] B. Lazarević, Z. Marjanović, N. Aničić, S. Babarogić. *Baze podataka.* Fakultet organizacionih nauka, Beograd, 2010.

[84] C. J. Date, *An Introduction to Database Systems*, 6th Edition, Addison Wesley, 1995.

[85] J. M. Smith, *Database abstractions: aggregation and generalization*, ACM Transactions on Database Systems (TODS), Volume 2 Issue 2, June 1977.

[86] J. Teorey, *A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model*, ACM Computing Surveys, Volume 18, pp. 197-222, 1986.

[87] H. Gregersen and J.S. Jensen, *Temporal Entity-Relationship models a survey*, IEEE Transactions on Knowledge and Data Engineering, vol. 11, pp. 464-497, 1999.

[88] *Anchor Modeling Tool*, Available at: http://www.anchormodeling.com/modeler

[89] E. F. Codd, *Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks*, IBM Research Report, San Jose, California, 1969.

[90] E. F. Codd, *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, Volume 13 Issue 6, pp. 377-387, June 1970.

[91] C. Jensen, J. Clifford, R. Elmasri, S. K. Gadia, P. J. Hayes & S. Jajodia, *A Consensus Glossary of Temporal Database Concepts*. SIGMOD Record, 23(1), pp. 52–64, 1994.

[92] R. T. Snodgrass, I. Ahn, *Temporal Databases*, IEEE Computer 19(9), pp. 35–42, September 1986.

[93] B. Inmon, *The Single Version of The Truth*, Business Intelligence Network (Powell Media LLC), 2004. Available at: http://www.b-eye-network.com/view/282

[94] L. Ronnback, O. Regardt, M. Bergholtz, P. Johannesson, and P. Wohed, *From Anchor Model to Relational Database*, valid at September 16th 2010. Available at: http://www.anchormodeling.com/wp-content/uploads/2010/09/AM-RDB.pdf

[95] L. Ronnback, *Back to The Moment*, Anchor Modeling, 2011. Available at: http://www.anchormodeling.com/wp-content/uploads/2011/05/Back-to-the-Moment.pdf

[96] H.-J. Lenz and A. Shoshani. *Summarizability in olap and statistical data bases*. Proceedings of Ninth International Conference on Scientific and Statistical Database Management-SSDBM'97, pages 132–143, 1997.

[96] A. S. Kamble, *A conceptual model for multidimensional data*, in *APCCM'08: Proc.* of the 15th on Asia-Pacific Conf. on Conceptual Modelling. Australia: Australian Computer Society, Inc., pp. 29–38, 2008.

[97] Nouha Arfaoui, Jalel Akaichi, *Data Warehouse: Conceptual And Logical Schema-Survey*, International Journal of Enterprise Computing and Business Systems, Vol. 2 Issue 1 January 2012.

[98] E. Franconi, A. S. Kamble, *The GMD data model and algebra for multidimensional information*, in 'Proc. of 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004), Riga, Latvia, June 7-11', pp. 446–462.

[99] I. Bojičić, *Rečnik podataka kao alat za opis i razvoj informacionog sistema*, Magistarski rad, Fakultet organizacionih nauka, Beograd, 2004.

[100] J.Trujillo, M. Palomar, *An Object Oriented Approach to Multidimensional Database Conceptual Modeling (OOMD)*, Conference on Information and Knowledge Management, ACM Press, New York, pp.16-21, 1998.

[101] J. Trujillo, *The GOLD Model: An Object Oriented Multidimensional Data Model For Multidimensional Databases*, In Proceedings of the 9th ECOOP International Workshop For PhD Students In Objects Oriented Systems, Lisboa, Portugal, 1999.

[102] J. Trujillo, M. Palomar, J. Gomez, I.Y. Song, *Designing data warehouses with OO conceptual models*, IEEE Computer 34 (12) 66–75 (Special issue on Data Warehouses), 2001.

[103] B. Husemann, J. Lechtenborger, G. Vossen, *Conceptual data warehouse design*, In Proceedings 2nd International Workshop on Design and Management of Fata Warehouses, Stockholm, 2000.

[104] A. Tsois, N. Karayannidis, T. Sellis, *MAC: Conceptual data modelling for OLAP*, In Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW-2001)', pp. 5–1, 5–13, 2001.

[105] P. Vassiliadis, *Modeling multidimensional databases, cubes and cube operations*, In Proceedings 10th International Conference on Statistical and Scientific Database Management, Capri, Italy, 1998.

[106] A. Abello, J. Samos, F. Saltor, *YAM2 (Yet Another Multidimensional Model): an extension of UML*, International Database Engineering & Applications Symposium (IDEAS 2002), pp. 172–181, Edmonton, Canada, 2002.

[107] L. Frias, A. Queralt, A. Olive, *EU-Rent Car Rentals Specification*, available at: http://www.cs.upc.edu/dept/techreps/llistat_detallat.php?id=690, 2003.

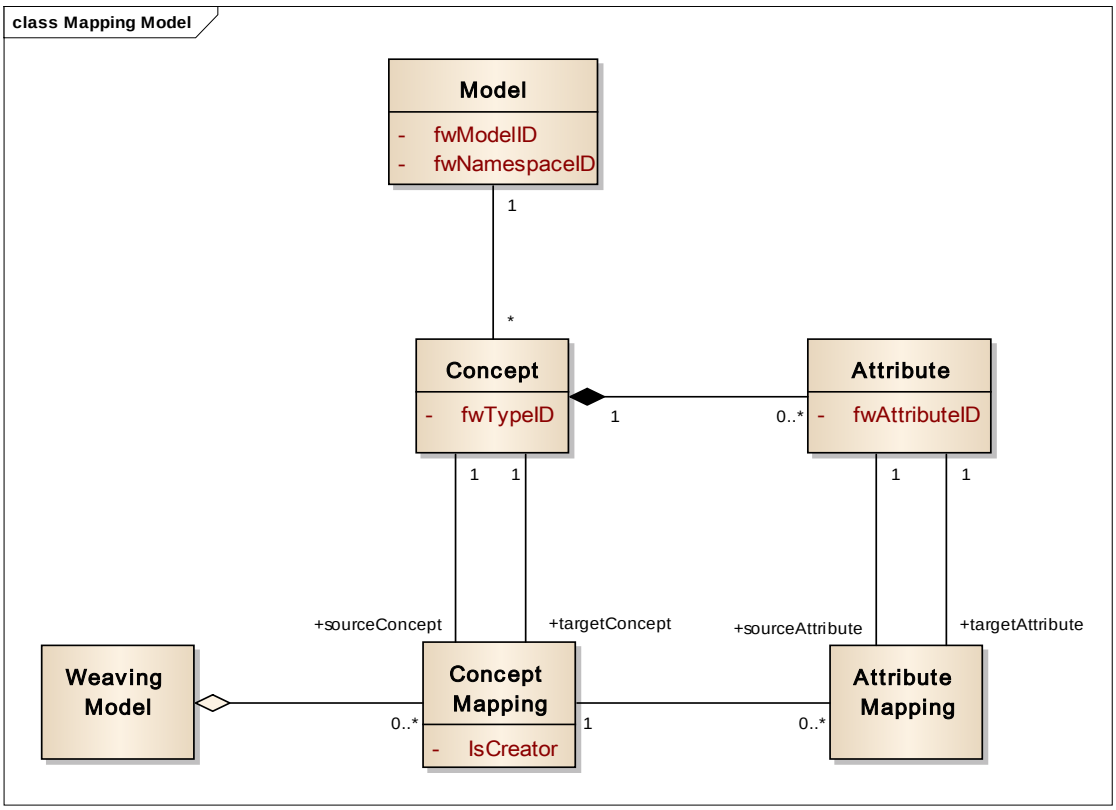
[108] I. Bojičić: *Domain Description Framework – alat za podršku modelom vođenom razvoju softvera*, InfoM, 2005.

[109] I. Bojičić, M. Bjeković: „*Pristup razvoju informacionog sistema automatskom transformacijom modela*“, InfoFest, Budva, 2006.

PRILOZI

Prilog 1 – Model preslikavanja

Model preslikavanja (eng. *Weaving Model*) omogućava transformacije koncepata različitih modela, poslovnih ili tehnoloških. Ovaj model je osnova opšte softverske komponente za preslikavanje koja je poslužila za definisanje preslikavanja između koncepata C-DV, L-DV i P-DV metamodela kako bi se transformacije između njih automatizovale, kao što je opisano u prethodnim poglavljima.



Slika 80. Model preslikavanja

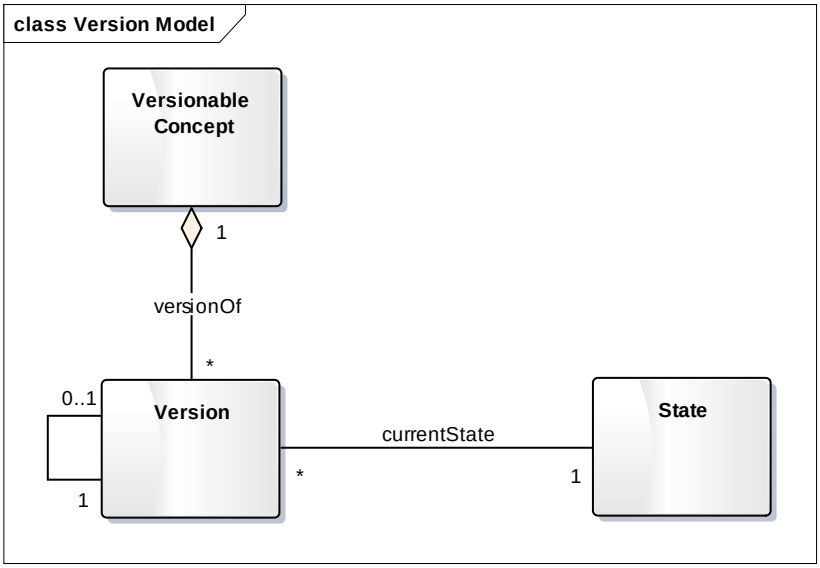
Kao što je prikazano na slici, koncept **Model** jedinstveno određuje poslovni ili tehnološki domen kojem pripadaju koncepti koji se mapiraju. Osnovni koncept modela je **Concept**, koji može predstavljati bilo koji konkretan ili apstraktan tip opisan u metamodelu (npr. Hab, Klasa, Link, Atribut i sl.). Definicija preslikavanja

koncepata se održava kroz **ConceptMapping**, rekurzivnu vezu (predstavljenu kao osnovna klasa). Ovaj koncept preko atributa *IsCreator* definiše da li se ciljni koncept ujedno i kreira preko tog preslikavanja.

Atributi koji se mapiraju sa izvornog na ciljni koncept se definišu preko preslikavanja na **AttributeMapping** objektu. Željene vrednosti preslikavanja se prenose na osnovu vrednosti pripadajućih **Attribute** struktura sa kojim je Concept povezan. Skup koncepata koji se preslikavaju mogu logički da budu grupisani preko koncepta **WeavingModel**.

Prilog 2 – Model verzionisanja

Komponenta za održavanje verzija modela zasnovana je na modelu koji je prikazan na Slici 81. Ova komponenta je predviđena za održavanje verzija bilo kog koncepta realnog sistema jer određeni objekti u sistemu zahtevaju da se vodi trag njihovih verzija, odnosno istorijat samog objekta. Ovako definisana opšta komponenta u ovom slučaju je prvenstveno predviđena za održavanje verzija C-DV modela koji predstavljaju transformiasane modele izvora.



Slika 81. Model verzionisanja

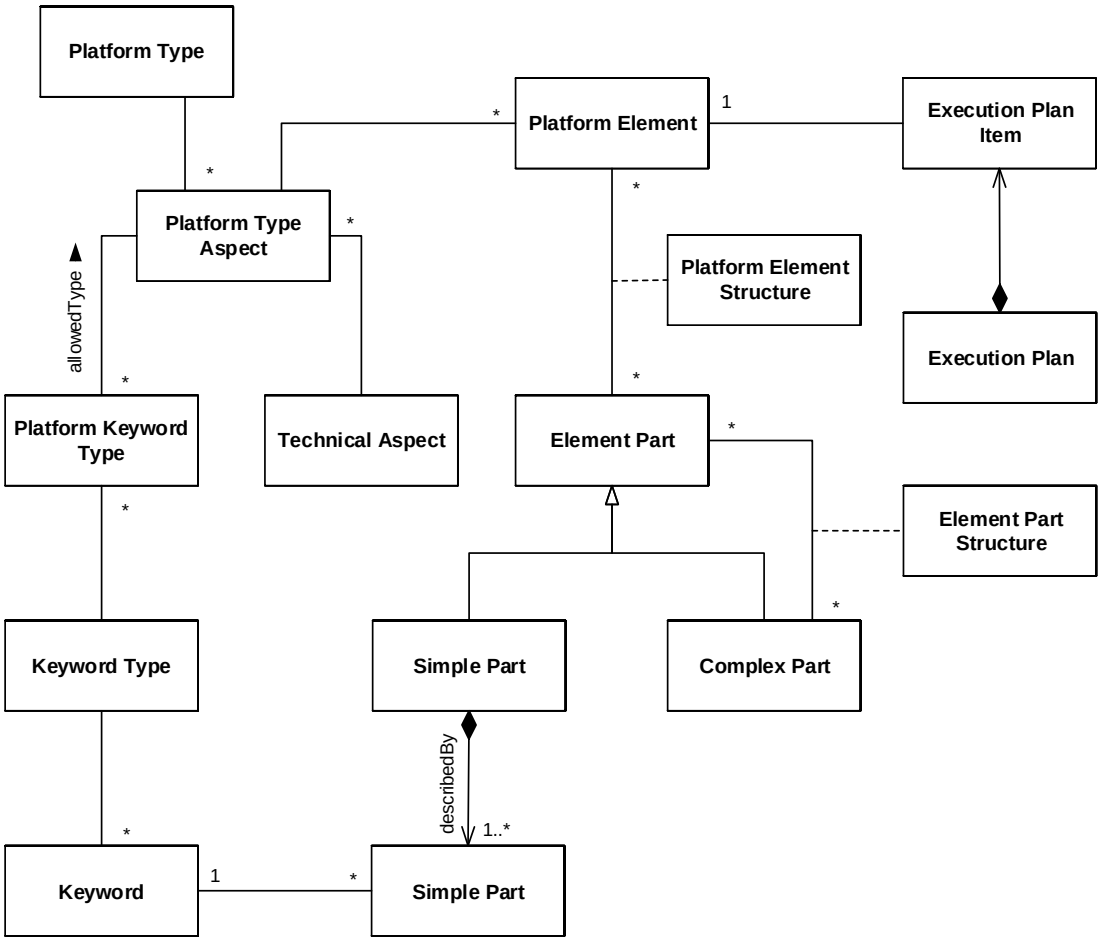
Kao što je prikazano na slici, koncept **VersionableConcept** je registrovani objekat, potencijalno iz bilo kog modela ili sam model, koji podleže verzionisanju. Različite verzije objekata se čuvaju preko koncepta **Version**, pri čemu se rekurzivnom vezom održava iz koje prethodne verzije je posmatrana verzija nastala. Svaka verzija ima odgovarajući *currentState,* koji je definisan konceptom **State**.

Održavanjem svih verzija konkretnog izvornog modela koji se transformiše u C-DV model čuvaju se sve verzije i promene koje nastaju na modelima izvora.

Prilog 3 – Model za opis tehnološke platforme

Model za opis tehnološke platforme (eng. *Platform Description Model - PDM*) omogućava prenosivost i implementaciju specifikacije sistema na više različitih tehnoloških okruženja.

Da bi se omogućila transformacija PIM modela na osnovu formalnih definicija tehnoloških platformi sa istim razvojnim alatom, neophodno je kreirati PDM model koji će skladištiti definicije. Kao što je prikazano na Slici 82, za svaku tehnološku platformu u koju se može vršiti transformacija, definiše se **PlatformType** (SQLServer, Oracle, DB2…) kao predstava te platforme.



Slika 82. Model za opis tehnološke platforme

Sa druge strane, **TechnicalAspect** predstavlja skup koncepata, tj. "reči" kojima raspolažu odgovarajući jezici za opis, odnosno metamodeli sa M^2 nivoa MDA arhitekture (Table, Primary Key, Foreign Key, Column...). Ograničenje koje definiše

pripadnost koncepata pojedinog metamodela tehnološkoj platformi, prikazuje se kroz entitet **PlatformTypeAspect**. Središnji enitet modela je **PlatformElement** i on je konkretizovani Platform Type Aspect za datu tehnološku platformu. Takođe, definisani platform elementi predstavljaju osnovnu logičku jedinicu koja može biti generisana. Ovako fleksibilno rešenje omogućava laku promenu definicije Platform Elementa usled promene tehnološkog okruženja.

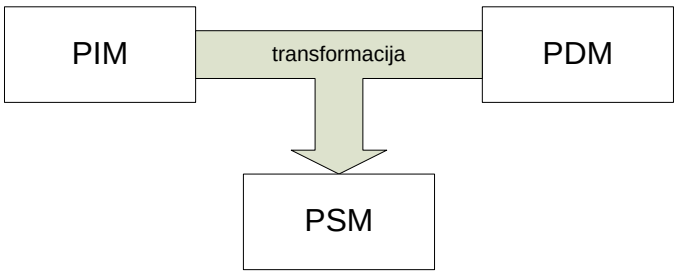
Svaki Platform Element se sastoji iz skupa **ElementPart**-ova, kao što je opisano u **PlatformElementStructure** entitetu. Zbog složenosti strukture pojedinih Platform Elemenata, **ComplexPart** može biti struktuiran od više prostih (**SimplePart**-ova) ili drugih složenih elemenata. Simple Part je opisan skupom **SimplePartDescription** elemenata koji su nosioci informacije o elementarnim ključnim rečima (**Keyword**). Keyword je koncept koji predstavlja osnovnu gradivnu jedinicu svakog Platform Elementa, pri čemu se njihova pripadnost određenoj grupi reči (**KeywordType**) predefiniše. Pripadnost tipa ključnih reči nekoj tehnološkoj platformi i konceptima metamodela je ograničena preko koncepta **PlatformKeywordType**. Zbog potrebe logičkog grupisanja određenih elemenata pri generisanju, **ExecutionPlan** predstavlja redosled izvršavanja koje se definiše skupom **ExecutionPlanItem** entiteta.

Prilog 4 – Generator

Sa stanovišta vezanog za pristup transformacije modela izdvajaju se dva problema: definisanje procesa (načina) transformacije i potencijalno veliki broj pravila transformacije koja se definišu.

Automatizacija transformacije se ističe kao najprihvatljiviji način jer se mogućnost greške svodi na najmanju moguću meru. Automatizacija se realizuje pomoću generatora – skupa softverskih komponenti koje interpretiraju poslovni model, opis tehnološke platforme i pravila transformacije.

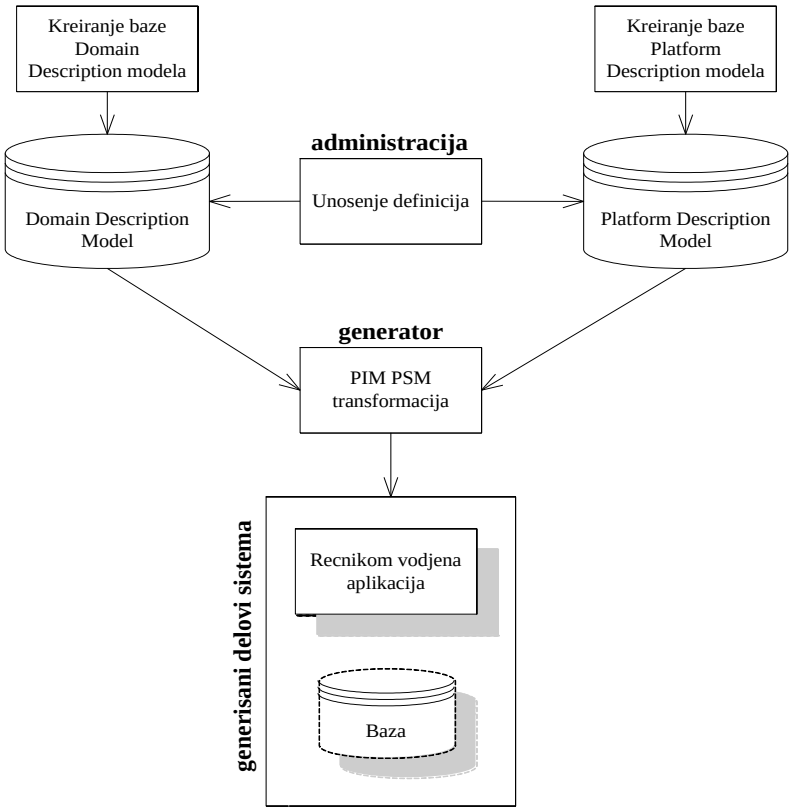
Sa stanovišta drugog problema, da bi se transformacija učinila prenosivom na više različitih okruženja, pogodno je definisati jezik za opis tehnoloških platformi (npr. PDM model). Ovakav pristup, prikazan na Slici 83, omogućava korišćenje jednog generatora za sve tehnološke platforme koje su opisane PDM modelom.



Slika 83. PIM → PSM transformacije na osnovu PDM definicija

Da bi se ostvario pristup zasnovan na MDA arhitekturi i automatizovanoj transformaciji modela, neophodno je, pre svega, realizovati odgovarajuće okruženje. Okruženje podrazumeva implementaciju ciljne MDA arhitekture i razvoj Rečnika IS.

Nakon toga se pristupa realizaciji alata (generatora) koji će da podrži proces transformacije. Kao što je prikazano na Slici 84, nakon unošenja definicija tehnoloških platformi i domenskih modela, vrši se pokretanje generatora koji interpretira uskladištene informacije dva prikazana metamodela, i kao izlaz transformacije generiše se kod, DDL naredbe ili XML dokumenti, shodno vrsti ciljne tehnološke platforme.



Slika 84. Koraci korišćenja generatora

Na prikazani način dobijaju se kompletni delovi izvršivog sistema, pri čemu je na minimum sveden procenat greške u komunikaciji između delova sistema koji su realizovani različitim jezicima za opis (relacioni model, objektni model, XML Schema...).

BIOGRAFIJA

Ivan Bojičić je rodjen 13.07.1974. godine u Nišu. Završio je Vojnu gimnaziju i Vojnu akademiju u Beogradu.

U oktobru 2000. godine upisuje se na poslediplomske studije Fakulteta organizacionih nauka.

Marta 2001. godine postaje član Laboratorije za informacione sisteme Fakulteta organizacionih nauka.

Marta 2004. godine brani magistarsku tezu *Rečnik podataka kao alat za opis i razvoj informacionog sistema* kod mentora Prof. dr Branislava Lazarevića i stiče zvanje magistar tehničkih nauka za informacione sisteme.

Od septembra 2004. godine radi u preduzeću Saga d.o.o. Beograd, trenutno na poziciji Direktor sektora za posebne softverske projekte.

Od 2009. do 2015. godine je angažovan kao asistent na Visokoj školi elektrotehnike i računarstva u Beogradu, na predmetu „Projektovanje informacionih sistema“.

Od 2014. do 2016. godine je angažovan kao asistent na Visokoj tehnološkoj školi u Aranđelovcu, na predmetu „Baze podataka“.

Učestvovao je na više od 30 istraživačkih, razvojnih i komercijalnih projekata u ulogama rukovodioca, tehničkog rukovodioca, analitičara, arhitekte rešenja i programera.

Autor je više naučnih radova u oblasti razvoja informacionih sistema i skladišta podataka.

Njegova naučno-istraživačka interesovanja su Razvoj informacionih sistema i skladišta podataka, Modelom vođen razvoj i Savremene softverske arhitekture.

Pregled publikovanih i objavljenih radova

Радови објављени у зборницима са скупова националног значаја (М60)

Саопштења са скупова националног значаја штампана у целини (М63)

- **Bojičić I.**, Turković O.: „*Metodološki pristup razvoju arhitekture informacionog sistema*“, InfoFest, Budva, 2007.
- **Bojičić I.**, Bjeković M.: „*Pristup razvoju informacionog sistema automatskom transformacijom modela*“, InfoFest, Budva, 2006.
- Babić S., **Bojičić I.**, Tamburić I: „*Realizacija platnih sistema korišćenjem standardnih softverskih komponenti*“, InfoFest, Budva, 2006.
- **Bojičić Ivan**: „*Aplikativni okvir za podršku modelom vođenom razvoju softvera*“, InfoFest, Budva, 2005.
- **Bojičić I.**, Lazarević B.: "*Aspektno orijentisan pristup razvoju rečnika informacionog sistema*", YU INFO, Kopaonik, 2004.
- Nešković S., **Bojičić I.**, Matić R.; “*Strategija razvoja informacionog sistema opštine*”, VIIth conference ISDOS, Zlatibor, 2003.
- Nešković S., Babarogić S., **Bojičić I.**, Matić R. “*Automatizacija integrisanog sistema menadžmenta kvalitetom i zaštitom životne sredine*”, DQM-2004.
- Nešković S., Vučković M., Babarogić S., **Bojičić I.**, Matić R.: “*Modelovanje i automatizacija menadžmenta kvalitetom i zaštitom životne sredine* ”, Symorg, Zlatibor, 2004.

Радови објављени у часописима националног значаја (М50)

Радови објављени у часопису националног значаја (М52)

- **Bojičić Ivan**: „*Domain Description Framework – alat za podršku modelom vođenom razvoju softvera*“, InfoM, 2005.
- **Bojičić Ivan**: “*Modelom vođen razvoj rečnika informacionog sistema*”, InfoM, 2004.

Радови објављени у зборницима међународних научних скупова (М30)

Саопштења са међународних скупова штампана у целини (М33)

- **Bojičić I.**, Marjanović Z., Jovanović V., Turajlić N., Petrović M., Vučković M.: „*A Comparative Analysis of Data Warehouse Data Models"*, ICCCC2016- IEEE proceedings (ISBN 978-1-5090-1735-5)
- Jovanović V., **Bojičić I.**: „*Conceptual Data Vault Model*", SAIS, Proceedings of the Southern Association for Information Systems Conference, Atlanta, 2012. pp. 131-136
- Jovanović V., **Bojičić I.**, Knowles C., Pavlic M.: „*Persistent Staging Area Models for Data Warehouses*", IACIS, Issues in Information Systems, South Carolina, 2012.

Радови објављени у научним часописима међународног значаја (М23)

- **Bojičić I**., Marjanović Z., Jovanović V., Turajlić N., Petrović M., Vučković M.: "*Domain/Mapping Model: a Novel Data Warehouse Data Model*", IJCCC 2017, Volume 12, Issue 2, pp. 157-173, (ISSN: 1841-9836)

Pregled projekata

Pregled izabranih projekata i softverskih rešenja u čijoj je izradi učestvovao kao član ili vođa tima:

- „*IIS NSZ – Integrisani informacioni sistem NSZ*", Ministarstvo za rad, zapošljavanje, boračka i socijalna pitanja, NSZ, 2016 - 2017.
- „*ISIB - Informacioni sistem za izvršenje budžeta Republike Srbije*", Ministarstvo finansija, Trezor, 2015 – 2016.
- „*Softver za akviziciju podataka od interesa za održavanje elektroenergetske opreme"*, Elektrovojvodina, Novi Sad, 2015.
- "*NCTS*" (New Computerized Transit System), Ministarstvo finansija, Uprava carina Republike Srbije, 2012 – 2014.

- "*CRIS - Informacioni sistem Centralnog registra obaveznog socijalnog osiguranja*", Ministarstvo rada i socijalne politike, Centralni registar, 2011 – 2013.
- "*Popis poljoprivrednih gazdinstava i domaćinstava*", Republički zavod za statistiku, Srbija, 2012 – 2013.
- „*Resursni portal*", Ministarstvo zdravlja, Crna Gora, Podgorica, 2012.
- „*Popis stanovništva*", Republički zavod za statistiku, Srbija, 2011-2012.
- "*Pravno informacioni sistem*", Ministarstvo pravde, Crna Gora, 2011 – 2012.
- *"Idejni projekat informacionog sistema JKP Vodovod – Valjevo",* Vodovod – Valjevo, Srbija, 2011.
- "*Elektronsko bankarstvo*", Raiffaisen banka, 2008 – 2010.
- „*Tehnis*", Ministarstvo ekonomije i regionalnog razvoja, Srbija, 2010.
- "*A3C2 Security Server*", Raiffaisen banka, 2009.
- "*Sistem direktnog zaduženja*", Udruženje banaka Srbije, 2008 – 2009.
- *„Rečnik resursa informacionog sistema državnih organa Republike Srbije",* Telekom, Srbija, 2007.
- **"***Vocalia***",** Societte General Bank, Srbija, 2007.
- **"***Web Credit***"**, Piraeus banka, 2006.
- **"***Klirinig čekova***"**, Udruženje banaka Srbije, 2004 – 2005.
- **"***ADMIS"*, Ministarstvo nauke i tehnologije, Srbija, 2002 – 2004.
- *"Informacioni sistem upravljanja menadžmenta kvalitetom i zaštite životne sredine"*, Luka Bar, Crna Gora, 2002 – 2003.
- "*Strategija razvoja informacionog sistema opštine Požarevac"*, Opština Požarevac, Srbija, 2002 – 2003.
- *"Informacioni sistem za administrativne procedure"*, Ministarstvo poljoprivrede Crne Gore, 2001.

Izjava o autorstvu

Potpisani: Ivan Bojičić

Izjavljujem

da je doktorska disertacija pod naslovom

MODELOM VOĐEN RAZVOJ SKLADIŠTA PODATAKA ZASNOVANOG NA DATA VAULT PRISTUPU

- rezultat sopstvenog istraživačkog rada,
- da predložena disertacija u celini ni u delovima nije bila predložena za dobijanje bilo koje diplome prema studijskim programima drugih visokoškolskih ustanova,
- da su rezultati korektno navedeni i
- da nisam kršio autorska prava i koristio intelektualnu svojinu drugih lica.

U Beogradu,
22.02.2017. godine

Potpis doktoranda

_____

Izjava o istovetnosti štampane i elektronske verzije doktorskog rada

Ime i prezime autora:	Ivan Bojičić
Studijski program:	Informacioni sistemi
Naslov rada:	Modelom vođen razvoj skladišta podataka zasnovanog na Data Vault pristupu
Mentor:	prof. dr Zoran Marjanović

Potpisani Ivan Bojičić

Izjavljujem da je štampana verzija mog doktorskog rada istovetna elektronskoj verziji koju sam predao za objavljivanje na portalu **Digitalnog repozitorijuma Univerziteta u Beogradu.**

Dozvoljavam da se objave moji lični podaci vezani za dobijanje akademskog zvanja doktora nauka, kao što su ime i prezime, godina i mesto rođenja i datum odbrane rada.

Ovi lični podaci mogu se objaviti na mrežnim stranicama digitalne biblioteke, u elektronskom katalogu i u publikacijama Univerziteta u Beogradu.

U Beogradu,
22.02.2017. godine

Potpis doktoranda

_____

Izjava o korišćenju

Ovlašćujem Univerzitetsku biblioteku „Svetozar Marković“ da u Digitalni repozitorijum Univerziteta u Beogradu unese moju doktorsku disertaciju pod naslovom:

MODELOM VOĐEN RAZVOJ SKLADIŠTA PODATAKA ZASNOVANOG NA DATA VAULT PRISTUPU

koja je moje autorsko delo.

Disertaciju sa svim prilozima predao sam u elektronskom formatu pogodnom za trajno arhiviranje.

Moju doktorsku disertaciju pohranjenu u Digitalni repozitorijum Univerziteta u Beogradu mogu da koriste svi koji poštuju odredbe sadržane u odabranom tipu licence Kreativne zajednice (Creative Commons) za koju sam se odlučio.

1. Autorstvo
2. Autorstvo – nekomercijalno
3. **Autorstvo – nekomercijalno – bez prerade**
4. Autorstvo – nekomercijalno – deliti pod istim uslovima
5. Autorstvo – bez prerade
6. Autorstvo – deliti pod istim uslovima

U Beogradu,
22.02.2017. godine

Potpis doktoranda

_____