UNIVERZA V LJUBLJANI

EKONOMSKA FAKULTETA

DIPLOMSKO DELO

PODATKOVNO MODELIRANJE NA PRIMERU PAKETNE PROGRAMSKE REŠITVE

Ljubljana, april 2004 KATJA ŠPEHAR

IZJAVA

Študentka Katja Špehar izjavljam, da sem avtorica tega diplomskega dela, ki sem ga napisala pod mentorstvom Jurija Jakliča in dovolim objavo diplomskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne_____ Podpis: _____

Kazalo

Uvod

Informacijski sistem organizaciji lahko predstavlja ključ do dolgoročno uspešnega poslovanja in konkurenčnih prednosti in vzrok neprimernih odločitev vodstva. Vsaka organizacija je specifična, zato bi morala preučiti, katere informacijske potrebe se pojavljajo, ali bo informacijski sistem razvijala samostojno ali ga bodo razvili zunanji razvijalci.

Vsak model razvoja informacijskega sistema ima prednosti in slabosti, vsekakor pa ni možna opredelitev absolutno najboljšega modela. Informatiki morajo dobro poznati vsakega izmed njih, da v dani situaciji lahko izberejo najprimernejšega. Nekatere modele razvoja informacijskega sistema odlikuje hiter razvoj, druge pa obvladovanje razvoja velikih in kompleksnih informacijskih sistemov. Ne glede na izbrani model so za razvoj informacijskega modela znani postopki, ki jih upoštevamo pri vsakem izmed metod razvoja informacijskega sistema.

Temelj kvalitetnega informacijskega sistema predstavlja dobro zgrajen podatkovni model, ki preprečuje podvajanje podatkov, anomalije, zagotavlja konsistentnost in centralizacijo podatkov. V prvem koraku izdelave podatkovnega modela se potrebno izdelati podatkovni model na logičnem, kasneje pa podatkovni model na fizičnem nivoju. Pri izdelavi podatkovnega modela imamo na voljo različne podatkovne strukture, tako da lahko vsaki organizaciji in metodi razvoja informacijskih sistemov poiščemo najprimernejšo strukturo podatkovnega modela.

Celotni razvoj informacijskega sistema je zahtevna naloga, kar so bistveno olajšala orodja CASE. Zmanjšalo se je število napak, skrajšal čas razvoja in zmanjšali so se celotni stroški, povezani z razvojem informacijskih sistemov. Poleg prednosti imajo orodja CASE tudi svoje slabosti, med katerimi sta najpomembnejši visoka cena in znanje, potrebno za učinkovito delo z orodji CASE. Orodje CASE, ki sem ga uporabljala pri svojem delu, je bilo orodje Oracle Designer, kar je razlog za predstavitev kot konkretni primer tega orodja.

Orodje Oracle Designer sem uporabljala pri projektu izdelave podatkovnega modela poslovnega področja osnovnih sredstev za paketni informacijski sistem Frapis. Finančno-računovodski informacijski sistem Frapis, projekt podjetja Infotrade, je namenjen srednjim in velikim podjetjem. Cilj projekta je bil izdelati prilagodljiv paketni informacijski sistem, ki bi organizacijam pri poslovanju omogočal celovito informacijsko podporo. Izgradnja Frapisa zaradi kompleksnosti projekta poteka po posameznih poslovnih področjih, ki se dodajajo ponudbi podjetja Infotrade. Razvoj informacijskega sistema Frapis je zaznamovalo dejstvo, da je informacijski sistem paketni, zaradi česar so se pojavljale določene posebnosti.

Pri informatizaciji področja osnovnih sredstev sem predstavila začetne faze informatizacije poslovnega področja osnovnih sredstev, in sicer:

- analiziranje poslovnega področja, kamor spada preučevanje poslovnega področja in zbiranje informacij,
- risanje entitetnega modela in
- izdelava podatkovnega modela.

Kasnejše faze izdelave informacijskega sistema:

- programiranja,
- testiranja in
- vzdrževanja

sem le na kratko predstavila, saj se je z izdelanim podatkovnim modelom moje delo na projektu začasno prekinilo, obenem pa se mi zdita izdelavi entitetnega in podatkovnega modela najpomembnejši in najbolj zanimivi fazi razvoja informacijskega sistema.

1. Razvoj informacijskih sistemov

Razvoj informacijskega sistema je kompleksen projekt, povezan z visokimi stroški in dolgoročnimi učinki, zato se ga je potrebno lotiti strokovno in sistematično. Vsaka napaka se odraža s podaljšanjem razvojnega časa in višjimi stroški. Pri razvoju informacijskega sistema sodeluje več skupin ljudi:

- **managerji** organizacije, ki sprejemajo odločitve glede rokov in višine investicije v informacijski sistem,
- **uporabniki**, ki predstavijo vsebinske zahteve glede informacijskega sistema,
- **informatiki** uporabniške zahteve obdelajo in
- **programerji**, ki zgradijo informacijski sistem.

1.1. Zgodovina razvoja informacijskih sistemov

Informacijski sistemi so bili že od samega začetka povezani z visokimi investicijami, tako na področju razvoja kot tudi vzdrževanja informacijskih sistemov. V šestdesetih let je vzdrževanje srednjih in velikih informacijskih sistemov predstavljalo 75% vseh stroškov, povezanih z informacijskim sistemom (Albanna, Osterhaus, 2000, str. 533–538). Posledica tega je bilo preučevanje situacije, ki je privedla do spoznanja, da je informacijske sisteme potrebno graditi na sistematičen način. Razvoj informacijskega sistema se je začel z analizo poslovnega področja in šele nato izdelal informacijski sistem. Kljub večjemu posvečanju pozornosti začetni fazi izdelave informacijskega sistema pa sta višina investicije v informacijski sistem in preseganje rokov ostala problematična.

Desetletje kasneje so se začele uporabljati različne strukturne tehnike in metode razvoja informacijskih sistemov. Uporaba podatkovnega in procesnega modela je nekoliko

zmanjšala investicije v vzdrževanje informacijskih sistemov, saj so v sedemdesetih letih padli na 60% stroškov, povezanih z investicijami v informacijski sistem (Albanna, Osterhaus, 2000, str. 533–538).

V osemdesetih letih se je pojavilo veliko programskih rešitev, saj je bil to čas razcveta majhnih informacijskih sistemov. Razvita so bila orodja CASE ter programska jezika Ada in C, kar dandanes veliko uporabljamo. Razvoj je naprej potekal v smeri objektno orientiranih metodologij (Albanna, Osterhaus, 2000, str. 533–538).

1.2. Modeli razvoja informacijskih sistemov

Vsaka organizacija se ob odločitvi pridobitve informacijskega sistema sreča z nekaterimi dilemami. Prva taka dilema je, ali obstoječ informacijski sistem posodobiti ali se usmeriti v pridobitev novega informacijskega sistema, druga dilema pa je, ali naj se organizacija odloči za lastni razvoj informacijskega sistema, sodelovanje zunanjih sodelavcev ali za nakup izdelanega informacijskega sistema. Specifičnost vsake posamezne organizacije zahteva natančen pregled vsake izmed možnosti, saj ima vsaka svoje prednosti in slabosti.

Predstavila bom najpogostejše načine razvoja informacijskih sistemov (Gradišar, Resinovič, 1996, str. 413–421):
- **življenjski cikel** razvoja informacijskega sistema,
- **tradicionalni** pristop gradnje informacijskega sistema,
- razvoj s **prototipom**,
- razvoj informacijskega sistema **s strani končnih uporabnikov** in
- **nakup** izdelanih informacijskih sistemov.

Metode razvoja informacijskega sistema v večini primerov izhajajo iz metode življenjskega cikla razvoja informacijskega sistema, kar je razlog, da sem se podrobneje posvetila tej metodi. Vsekakor pa je pri razvoju informacijskih sistemov pri vseh metodah potrebno široko vključevanje uporabnikov in gradnja ponovno uporabljivih delov informacijskega sistema, saj ima le na ta način bodoči informacijski sistem dobre možnosti za uspeh (Purba, 2000, str. 481).

1.2.1. Življenjski cikel

Metoda gradnje informacijskega sistema pri pristopu življenjskega cikla je sestavljena iz štirih faz (Jurca, 2000, str. 415–416):

1. začetka,
2. razvoja,
3. uvajanja ter

4. izvajanja in vzdrževanja.

Metodo življenjskega cikla razvoja informacijskega sistema bom predstavila na konkretnem primeru razvoja informacijskega sistema Frapis, čigar razvoj sta zaznamovali dve značilnosti:

- informacijski sistem Frapis je **paketni** informacijski sistem;
- razvoj je potekal po pristopu **življenjskega cikla** razvoja informacijskega sistema.

Prva faza – začetek projekta

Začetna faza razvoja informacijskega sistema je zelo pomembna, saj lahko bistveno zmanjša ali poveča stroške v primeru dobre ali slabe izvedbe prve faze. Večino napak in nesporazumov med uporabniki, informatiki ter managerji, ki bi se sicer pokazali v kasnejših fazah, je mogoče odpraviti že v začetnih fazah. Podfaze, ki se pojavljajo v okviru faze začetka projekta, pa so (Gradišar, Resinovič, 1996, str. 416), (Golob et al., 2003, str. 13):

1. **Študija izvedljivosti** predstavlja preverjanje smotrnosti začetka projekta. Pri tej podfazi sodelujejo uporabniki, managerji in informatiki. Uporabniki podajo zahteve in predloge, ki jih informatiki preverijo, ali so uresničljive, oziroma, v kakšni meri so uresničljive, managerji pa opozarjajo na ekonomske in organizacijske vidike, ki se pri razvoju informacijskega sistema ne smejo zanemariti. Po končani študiji izvedljivosti se začetek izvajanja projekta odobri ali zavrne.
2. Po odobritvi projekta sledi podfaza **načrtovanja projekta**. V tem delu se celotni projekt razdeli na manjše, lažje obvladljive dele. Posamezne dele projekta se dodeli posameznikom ali skupini. Vzporedno s tem je potrebno preučiti soodvisnosti med deli projekta, saj je za začetek enih potreben zaključek drugih. Rezultat podfaze načrtovanja projekta je načrt projekta.

Razvoj paketnega informacijskega sistema Frapis se je v prvi fazi razvoja informacijskega sistema nekoliko razlikoval od razvoja programske rešitve za znanega naročnika, saj pri študiji izvedljivosti bodoči uporabniki rešitve niso sodelovali. Študija izvedljivosti se je izvedla interno. S potrditvijo študije izvedljivosti se je celotni projekt izdelave paketnega informacijskega sistema razdelil na posamezna poslovna področja, ki so dobila različne prioritete. Poslovno področje osnovnih sredstev je imelo v primerjavi s poslovim področjem računovodstva manjšo prioriteto, zato se je razvilo kasneje.

Druga faza – razvoj informacijskega sistema

Po predhodno zaključeni prvi fazi se prične faza razvoja informacijskega sistema. Večina informatikov se strinja, da je najtežji del izgradnje informacijskega sistema pridobitev vseh potrebnih informacij o informatiziranem poslovnem področju. Fazo razvoja

informacijskega sistema sestavljajo podfaze, ki pa niso niti enako dolge niti enako zahtevne (Gradišar, Resinovič, 1996, str. 416–418), (Koletzke, Dorsey, 1999, str. 18–30).

1. Prva podfaza se imenuje **podrobna analiza zahtev** uporabnika, kjer naj bi uporabnik čim natančneje podal zahteve, potrebe ter znanja glede poslovnega področja. Postopek pridobivanja informacij od uporabnika je dolgotrajen postopek in pred dokončnim dogovorom je potrebno večkratno usklajevanje stališč. Načini pridobivanja informacij so (Vintar, 1996, str. 88):
   a) preučevanje razpoložljivega gradiva,
   b) intervjuji,
   c) sestanki,
   d) ankete,
   e) opazovanje,
   f) merjenje in vzorčenje.
2. **Zasnova notranje zgradbe** sistema je narejena na podlagi predhodno zbranih informacij o poslovnem področju. V tej fazi se izdela podatkovni model na logični ravni in kasneje podatkovni model na fizični ravni. Podrobna analiza zahtev ter zasnova notranje zgradbe lahko potekata zaporedno ali izmenično, običajno pa je večkratno popravljanje podatkovnega modela, kar odpravlja nesporazume glede zahtev med informatiki in uporabniki.
3. Neobvezna podfaza razvoja informacijskega sistema je **nabava in namestitev strojne opreme**. Večina podjetij, ki razvijajo informacijske sisteme strojno opremo že imajo. V nasprotnem primeru pa si podjetje priskrbi strojno opremo, ki jo bo pri delu potrebovalo.
4. Podfaza **programiranja** je sestavljena iz treh delov:
   a) prvi del je kodiranje in generiranje, kar predstavlja zapis prej opisanih zahtev in potreb uporabnikov v enem od programskih jezikov. Sem spadajo na primer postopki izračunov, izdelava zaslonskih mask, izpisov, različne kontrole za preprečevanje napak.
   b) Testiranje programskih modulov je sestavljeno iz dveh delov:
      - v prvem delu prevajalnik preveri ali je program napisan v skladu s sintaktičnimi pravili,
      - v drugem delu se preverja logično delovanje programa.

      V fazi testiranja se odkrije veliko napak, vendar se nekatere napake pojavijo le v specifičnih situacijah, zato jih je težko odkriti. Neodkrite napake se odpravijo v fazi uvajanja in vzdrževanja informacijskega sistema.

   c) Sprotno dokumentiranje informacijskega sistema poveča razumljivost programskih modulov, kar skrajša čas pri kasnejšem vzdrževanju in spreminjanju programskih modulov ter odpravljanju napak na informacijskem sistemu.
5. Naslednja podfaza je **testiranje informacijskega sistema v celoti**, saj nam pravilno delovanje posameznih delov informacijskega sistema ne zagotavlja pravilnega delovanja celotnega informacijskega sistema. Test informacijskega sistema je potrebno

načrtovati, kajti v nasprotnem primeru lahko testiranje določenih poslovnih področij spregledamo.

6. Zadnja podfaza v tem sklopu je **dokumentiranje informacijskega sistema**, ki je sestavljeno iz predhodnih faz. Med dokumentacijo spadajo:
   a) navodila o uporabi informacijskega sistema, ki je namenjen uporabniku in
   b) dokumentacija, ki je namenjena programerjem in služi za kasnejše odkrivanje napak ali dograjevanju informacijskega sistema.

Faza razvoja paketne programske rešitve Frapis se je od zgoraj omenjenih podfaz najbolj razlikovala v podfazi podrobne analize zahtev. Bodoči uporabniki namreč niso bili znani, zato je bilo potrebno poiskati druge vire informacij, ki bodo nadomestili intervjuje, ankete in sestanke z bodočimi uporabniki. Glavni viri pri gradnji paketnega informacijskega sistema so:

- literatura konkretnega poslovnega področja,
- intervjuji, sestanki in ankete s potencialnimi bodočimi uporabniki nove programske rešitve in
- sodelovanje sodelavcev, ki imajo izkušnje od preteklih projektov.

Ostale podfaze razvoja paketne informacijske rešitve se niso bistveno razlikovale od informacijskih sistemov, grajenih za znanega naročnika. Poudarila bi le dosledno upoštevanje smernice za čim večjo fleksibilnost bodočega informacijskega sistema, saj v trenutku razvijanja ni bilo znanih dejanskih zahtev in posebnosti naročnika. Zavedali smo se, da je visoka fleksibilnost informacijskega sistema lahko ključnega pomena pri kasnejšem naročnikovem izbiranju med podobnimi paketnimi informacijskimi sistemi, ki se pojavljajo na trgu.

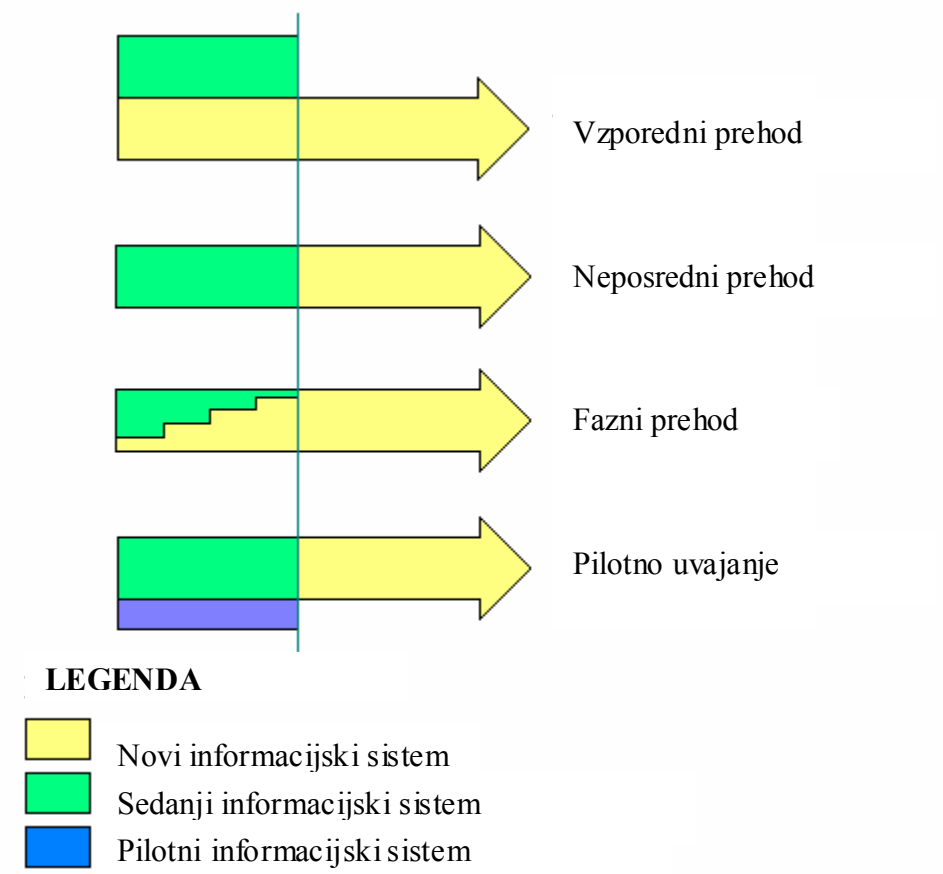
Tretja faza – uvajanje informacijskega sistema

Faza uvajanja informacijskega sistema v organizacijo se lahko prične po predhodno zaključenih fazah, sestavljena pa je iz naslednjih podfaz (Gradišar, Resinovič, 1996, str. 418–420):

- v času prve podfaze **načrtovanja izvajanja** je treba narediti načrt uvajanja informacijskega sistema v organizacijo, kar zajema obdobje in stroške uvajanja informacijskega sistema. Pomemben dejavnik pri načrtu uvajanja predstavlja dejavnost, v kateri posluje organizacija, saj nekatere organizacije brez informacijske podpore ne morejo poslovati in zato v fazi uvajanja novega informacijskega sistema skušajo minimizirati tveganje.
- Po doseženem dogovoru se prične podfaza **urjenja uporabnikov,** pri čemer je potrebno upoštevati predznanje uporabnikov, delo, ki ga opravljajo ter navdušenje nad novim informacijskim sistemom.

- Poznamo več načinov **prehoda organizacije na nov informacijski sistem**, med katerimi ima vsak določene prednosti in slabosti. Razlike med prehodi so predvsem v varnosti ter stroških prehoda iz starega na nov informacijski sistem. Načini prehodov:
  a) vzporedno delovanje je način prehoda, pri katerem določeno časovno obdobje istočasno delujeta novo razviti in obstoječi informacijski sistem. Ključna slabost vzporednega prehoda je povezana z dvojnim vnašanjem podatkov v sistem, kar ob predpostavki morebitnega dvoma uporabnika glede na novi informacijski sistem predstavlja večjo možnost, da si uporabnik ustvari nerealno sliko o novem informacijskem sistemu. Glavna prednost vzporednega prehoda je visoka stopnja varnosti, ker podjetje ne izgubi podatkov in informacijske podpore ob pojavu večjih napak na novem informacijskem sistemu.
  b) Neposredni prehod iz obstoječega na nov informacijski sistem se zgodi v trenutku, kar pomeni prenehanje uporabljati obstoječi informacijski sistem in takojšen prehod na novega. Neposredni in vzporedni prehod predstavljata nasprotujoča si načina uvajanja, kar se odraža pri prednostih in slabostih obeh načinov. Prednost neposrednega prehoda predstavlja enkratno vnašanje podatkov v informacijski sistem, kar razbremenjuje uporabnike sistema. Vsekakor pa se pri neposrednem prehodu pojavlja slabost, ki organizacije odvrača od odločanja za omenjeni način pristopa. Pri neposrednem načinu prehoda se pojavlja izmed vseh načinov prehoda najvišja stopnja tveganja in v primeru večjih napak na novem informacijskem sistemu organizacija lahko izgubi pomembne podatke ali pa začasno ostane brez informacijske podpore.
  c) Postopni ali fazni prehod je način prehoda, ki skuša izkoristiti prednosti in omiliti slabosti zgoraj omenjenih načinov pristopa. Nov informacijski sistem se pri faznem pristopu vključuje postopoma po poslovnih področjih, obstoječi informacijski sistem pa se po poslovnih področjih postopoma umika. Fazni način pristopa je možno uporabljati v primeru velike podobnosti med obema informacijskima sistemoma. Prednost postopnega prehoda je v enkratnem vnašanju podatkov ter majhnemu tveganju ob pojavu napak na novem informacijskem sistemu. Slabost postopnega prehoda pa je v tem, da ga lahko uporabljamo le v primeru, ko sta si informacijska sistema dovolj podobna, da je možno postopno vključevanje poslovnih področij.
  d) Pilotni način uvajanja je kombinacija vseh zgoraj omenjenih pristopov. Pri tem načinu uvajanja se v organizaciji zbere skupina testnih, nad novim informacijskim sistemom navdušenih uporabnikov, ki so novi informacijski sistem pripravljeni nekaj časa vzporedno uporabljati. Prednost pristopa je, da ni siliti vseh uporabnikov v dvojno delo in da se začetne napake in pomanjkljivosti odpravi pri manjši skupini uporabnikov, kar kasneje omogoča lažji in manj tvegan prehod celotne organizacije na novi informacijski sistem. Ključna slabost pilotnega pristopa je v iskanju uporabnikov, ki bi želeli sodelovati in ki so primerni za pilotno skupino uporabnikov, pri katerih bi pilotno uvedli novi informacijski sistem.

Slika 1: Različni načini uvajanja novega informacijskega sistema



Vir: Gradišar, Resinovič , 1996, str. 420

- **Testiranje ustreznosti** izvajajo uporabniki informacijskega sistema, kamor štejemo:
  a) tehnično, pri katerem se preverja, ali informacijski sistem deluje, in
  b) vsebinsko, ki predstavlja preverjanje ali informacijski sistem zadovoljuje vsebinske vidike informatiziranega poslovnega področja.

  Glavni razlog ugotovljene neustreznosti informacijskega sistema v tej fazi je navadno nenatančna izvedba predhodnih faz ali nesporazumi med informatiki in uporabniki.
- **Spremljanje delovanja sistema po uvedbi** je zadnja podfaza uvajanja informacijskega sistema, s katerim primerjamo stroške in koristi novega informacijskega sistema v primerjavi s starim informacijskim sistemom.

Pri paketnih programskih rešitvah se pred fazo uvajanja pojavi nova faza, in sicer iskanje naročnikov za izdelan informacijski sistem. Po pridobitvi naročnika za izdelano programsko rešitev ter namestitvi informacijskega sistema sledi faza uvajanja informacijskega sistema. V fazi uvajanja, razen prilagoditve konkretni organizaciji, ni razlik v primerjavi z razvojem informacijskega sistema za znanega naročnika. Dejansko se proces razvijanja od točke naročnika poenoti z razvojem informacijskega sistema za znanega uporabnika**.**

Četrta faza – izvajanje in vzdrževanje informacijskega sistema

V zadnji fazi je potrebno uvedeni informacijski sistem spremljati in vzdrževati, kar predstavlja odkrivanje in popravljanje skritih napak, ter uporabnikom nuditi podporo ob morebitnih vsebinskih ali tehničnih težavah. Obenem je v tem delu potrebno nuditi možnost nadgradnje informacijskega sistema, saj se poslovno okolje ves čas spreminja, kar posledično pripelje do neizogibnih prilagoditev informacijskega sistema. Faza je sestavljena iz dveh podfaz (Gradišar, Resinovič, 1996, str. 421):

1. **podporo tekočemu delu** nudijo strokovnjaki, ki poznajo področje dela, obenem pa imajo dovolj tehničnega znanja za sporazumevanje s programerji, ki jim sporočajo dodatne potrebe uporabnikov;
2. **vzdrževanje** pa pomeni posodabljanje razvitega informacijskega sistema na način, ki najbolje zadovoljuje potrebe uporabnikov.

Drugi pristopi, ki se pojavljajo pri razvoju informacijskih sistemov, so sicer manj pogosti, kot je pristop razvoja informacijskega sistema z metodo življenjskega cikla, vendar se vseeno pojavljajo.

1.2.2. Tradicionalni pristop

Glavna značilnost tradicionalnega pristopa je, da uporabnik informatiku posreduje svoje potrebe in želje glede programske rešitve, ki na podlagi opisov izdela ustrezno programsko rešitev. Pomembno dejstvo pri tem pristopu je, da informatik ne upošteva formalnih postopkov pri izdelavi programske rešitve.

Slabo lastnost tovrstnega pristopa predstavlja nevarnost, da ne pokrije vseh uporabnikovih potreb, obenem pa je rešitev slabo dokumentirana, kar otežuje morebitno naknadno odpravljanje nepravilnosti ali nadgrajevanje, še posebej v primeru, ko naj bi to delo naredil drug informatik in ne avtor programske rešitve. Kljub naštetim slabostim je pristop pogost, še posebej v Združenih državah Amerike.

1.2.3. Razvoj s prototipom

Pri razvoju informacijskega sistema s prototipom se uporabniku že v zelo zgodnjih fazah razvoja predstavi delujočo rešitev, ki pokriva le osnovne funkcije. Kasneje se dograjujejo in izpopolnjujejo tudi druge funkcije, dokler ne pridemo do končne rešitve (Vintar, 1996, str. 76). Razvoj informacijskih sistemov je praviloma povezan z velikimi stroški, nesporazumi z uporabniki ter dolgotrajnostjo. Vse te slabosti pristop z razvojem prototipa odpravlja, saj je preizkušanje idej povezano z nizkimi stroški, v primerjavi z drugimi pristopi, prav tako

pa si uporabniki lažje predstavljajo, kaj dejansko informatik želi izvedeti od njih. Ključna prednost razvoja informacijskega sistema s prototipom je izboljšanje kvalitete začetnih faz razvoja, kar predstavlja osnovo za kvaliteten informacijski sistem.

Kljub temu da se na prvi pogled zdi, da je razvoj s prototipom rešitev dosedanjih slabosti drugih metod, pa je potrebno upoštevati, da tega pristopa ne moremo uporabljati v vseh primerih. Dobro se obnese v časovnih stiskah, ko drugi pristopi ne bi omogočili pravočasne izdelave informacijskega sistema in kadar je možna delitev celotnega projekta na manjše, obvladljivejše in dokaj samostojne dele oziroma področja. Najbolje je uporabiti ta pristop v primeru, ko uporabnik ne ve točno, kaj dejansko želi imeti, ter kadar za uporabnika prijaznost programske rešitve predstavlja ključen dejavnik.

Primeri, v katerih se pristop z razvojem prototipa ne obnese najbolje, pa so:
- kadar informatik ne pozna orodja, s katerim gradi rešitev,
- kadar uporabnik ni pripravljen sodelovati z informatikom,
- kadar vodstvo ne kaže zanimanja za tak način dela.
- Prav tako pa pristopa s prototipom ne moremo uporabljati, kadar podatki niso enostavno dosegljivi.

1.2.4. Razvoj s strani končnih uporabnikov

Naslednji model, ki ga bom predstavila, je razvoj informacijskega sistema s strani končnih uporabnikov. Kot vsi modeli, ima svoje prednosti in svoje lastnosti. Največja prednost pristopa je vsekakor dejstvo, da si uporabnik natančno predstavlja lastnosti, ki naj bi jih imel informacijski sistem, saj v tem primeru ne pride do nesporazumov med informatiki in uporabniki. Žal se v povezavi s ključno prednostjo pojavi slabost modela, ki jo za razvoj programske rešitve predstavlja pomanjkanje informacijskega znanja uporabnikov.

Model razvoja programskih rešitev s strani končnih uporabnikov je primeren za izdelavo takih programskih rešitev, za katere ima uporabnik dovolj znanja, da lahko več ali manj samostojno razvije rešitev, ki jo potrebuje. Za izdelavo kompleksnih informacijskih sistemov model ni primeren. Omenjene rešitve so s strani podjetja zaželene vsaj do trenutka, ko v organizaciji še ni informacijskega sistema, ki bi zadovoljeval potrebe uporabnika, saj razbremenjujejo oddelek za informatiko.

Vodstvo organizacije mora dobro premisliti, ali bodo v organizaciji dopuščali razvoj informacijskih sistemov s strani končnih uporabnikov, saj se v zvezi s tem pojavljajo naslednji problemi (Hale, 2000, str. 727–747):
- slabo varovanje podatkov,
- neučinkovita raba resursov,
- neprimerno izobraževanje »razvijalcev«,

- neprimerna pomoč uporabnikom,
- nekonsistentni sistemi in
- podvajanje programskih rešitev.

1.2.5. Uporaba programskih paketov

Dandanes se na trgu pojavlja veliko paketnih informacijskih sistemov, med katerimi nekateri bolje, drugi slabše pokrivajo potrebe določene organizacije. Splošna značilnost paketov je, da so večinoma izdelani kvalitetno, saj so rezultat dela strokovnega kadra. Vsak programski paket je namenjen informatizaciji določenih potreb organizacije, zato je pred nakupom potrebno preučiti vsak posamezni paket. Določen paket programske rešitve lahko zadovoljuje večino informacijskih potreb, bolj verjetno pa je, da je bo morala organizacija pridobiti več različnih paketov, da bo pokrila informacijske potrebe organizacije.

Uporaba programskih paketov je smotrna v naslednjih primerih:
1. programske pakete je organizaciji smotrno uporabljati, ko sama nima dovolj resursov, da bi izdelali lastni informacijski sistem.
2. Drugi primer ustreznosti nakupa programskega paketa je, če bi bili stroški izdelave informacijskega sistema višji, kot je cena nakupa, še posebej, če konkretni programski paket pokriva dovolj širok spekter potreb organizacije.
3. Zadnji primer, ki ga bom omenila v prid uporabe programskih paketov, je, ko se organizacija spopade s stisko s časom. Izdelava informacijskega sistema je v večini primerov kompleksna in dolgotrajna naloga, kar je v primeru tesnih rokov razlog, da se organizacije odločajo za nakup paketov in na ta način ne presežejo dogovorjenih rokov.

Vsa dejstva pa ne govorijo v prid uporabe programskih paketov. Ključni problem pri nakupu paketne programske rešitve je, da informacijskih potreb organizacije ne pokriva dovolj dobro, povzroči pa sledeče:
1. nakup večjega števila programskih paketov;
2. lastno delo informacijskega oddelka pri povezovanju paketov;
3. stroški, povezani z nakupom programskega paketa, lastnega razvoja in povezovanja, lahko odtehtajo razvoj informacijskega sistema za konkretno organizacijo.

Predstavljene prednosti in slabosti modelov kažejo na dejstvo, da je pred odločanjem organizacije o modelu razvoja informacijskega sistema potrebno narediti študijo informacijskih potreb in stroškov in se glede na rezultate odločiti za najprimernejši model.

2. Podatkovni model

Konkretne modele razvoja informacijskih sistemov sem predstavila, v nadaljevanju pa sem se osredotočila na predstavitev podatkovnega modela. V začetnih obdobjih pojava

projektiranja informacijskih sistemov je bil največji pomen namenjen modeliranju procesov, modelu podatkov pa je bilo namenjeno manj pozornosti. Vzrok za to je bil predvsem ta, da je bila količina podatkov, ki jo je bilo potrebno obdelati, dokaj majhna in zato ni bila problematična. Programske rešitve so delovale ločeno, po posameznih datotekah, kar je pomenilo večkratno vnašanje podatkov. V primeru spreminjanja teh podatkov je bilo potrebno popraviti zapise v vseh datotekah. Zaradi tega je več programov obdelovalo iste podatke, v ločenih datotekah. Modeliranje podatkov je pomembnejšo vlogo dobilo šele z razvojem podatkovnih baz.

2.1. Splošne značilnosti

Podatkovni model je zbirka konceptov, s katerimi skušamo izraziti statične in dinamične lastnosti podatkov v okviru informacijskega sistema, obenem pa mora realno predstavljati izsek realnosti (Kovačič, Vintar, 1994, str. 79–80). Predstavlja množico pravil, ki določajo, kako smejo biti podatki v bazi podatkov organizirani oziroma strukturirani ter dovoljene operacije nad podatki.

Lastnosti podatkovnega modela (Golob et al., 2003, str. 18), (Svetek, 2003, str. 55–59), (Koletzke, Dorsey, 1999, str. 22):

- struktura podatkov je razčlenjena v enostavne podatkovne modele, saj so stabilnejši od kompleksnih,
- vsi podatki so shranjeni v centralni bazi podatkov, ki je last celotne organizacije, kar pomeni, da vsaka organizacijska enota nima svoje baze podatkov, temveč se podsisteme organizira tako, da lahko pridejo do centralne baze in do podatkov, ki jih vsebuje.
- Dosežena je neodvisnost podatkov, kar pomeni, da vsaka programska rešitev, ki potrebuje podatke, le-te poišče v centralni bazi podatkov, kjer je poskrbljeno za ažurnost podatkov.
- Odpravljena so podvajanja, saj se v bazo podatkov vnaša podatke le enkrat, prav tako se tudi vsako spreminjanje ali brisanje podatkov beleži le enkrat, kar omogoča pravilni prikaz vsem uporabnikom, ki so jim podatki dostopni.
- Omogočen je centraliziran nadzor nad njimi, saj se vsakemu uporabniku, ki ima dostop do programske rešitve in s tem posredno do baze podatkov, določi pravice vnašanja, spreminjanja ter brisanja.
- Avtomatizirana orodja lajšajo delo informatiku že v fazi definiranja podatkov, razvoju novih programskih rešitev, poročil, iskanju napak ter pri delu z bazami podatkov.

Izdelava podatkovnega modela je zahtevna naloga, predvsem zaradi kompleksnosti realnega sveta. Proces izdelave podatkovnega modela ločimo v dve fazi, in sicer:

1. logično in
2. fizično fazo izgradnje podatkovnega modela.

2.2. Logična faza izgradnje podatkovnega modela

Bistvo logične faze izgradnje podatkovnega modela za informacijski sistem predstavlja izdelavo podatkovnega modela, ki bo neodvisen od kasneje uporabljene tehnologije razvoja informacijskega sistema. Pomembno je, da je podatkovni model na logični ravni čim bolj enostaven. Pri izdelavi podatkovnega modela na logični ravni sodelujejo bodoči uporabniki informacijskega sistema, ki so poleg analize zahtev, ki je bila narejena predhodno, vir informacij glede zahtev, ki naj jih ima informacijski sistem.

Izdelava podatkovnega modela na logičnem nivoju se običajno prične graditi na najvišjem nivoju, kasneje se gre bolj v detajle. Na logičnem nivoju izdelave podatkovnega modela se podatkovni model normalizira, vključi se tuje ključe in potencialne indekse. Podatkovni model se na logičnem nivoju v sodelovanju z uporabniki večkrat preverja in ni neobičajno, da se v tej fazi razvoja veliko spreminja (Purba, 2000, str. 485–487).

Kljub temu da je podatkovni model poenostavljena slika realnosti, mora zagotavljati dejansko sliko realnega sveta, kajti v nasprotnem primeru podatkovni model ni uporaben. V logični fazi izdelave podatkovnega modela se realni svet s pomočjo abstrakcije poenostavlja, kar pomeni, da se zavestno odmišlja podrobnosti realnega sveta ter osredotoči na ključne lastnosti pojavov, ki so pomembni za razumevanje ključnega poslovnega področja (Kovačič, Vintar, 1994, str. 81).

Pri logični fazi izdelave podatkovnega modela se v zvezi z informatiziranim poslovnim področjem pojavljajo tri ključna vprašanja (Pressman, 1997, str. 301):
1. kaj je osnovna entiteta,
2. kateri atributi jo opisujejo,
3. kakšne so povezave med posameznimi entitetami.

Odgovore na zgornja vprašanja nam ponuja entitetni diagram, ki je pri logični fazi modeliranja podatkov najpogosteje uporabljen, saj informatiku omogoča nazorno grafično definiranje podatkov (Pressman, 1997, str. 301).
Začetki entitetnega modela segajo v leto 1976, ko ga je prvi predstavil Peter Chen. Njegov model je bil osnova za vse nadaljnje inačice entitetnega modela. Doživel je precej dopolnitev in razširitev, kljub temu pa je ohranil svojo ključno pozitivno lastnost – enostavnost modela. Entitetni model je edini model, ki se je uveljavil pri izdelavi podatkovnega modela na logični ravni (Vintar, Kovačič, 1994, str.100).

Entitetni model je uporaben predvsem v primeru izdelave kompleksnega informacijskega sistema, saj omogoča dober pregled nad podatki. Entitetni model je zgrajen iz treh elementov (Barker, 1995, str.7-1–7-20), (Pressman, 1997, str. 301–309):
1. **entitetni tip** predstavlja realno osebo, dogodek ali koncept. Kvalitetni entitetni tipi imajo pomenska in unikatna imena, ki vsebinsko nakazujejo pomen posameznega

entitetnega tipa, kar olajša razumevanje entitetnega modela in entitetnih tipov samih. Entitetni tipi predstavljajo subjekte v realnem svetu, za katere vodimo zapise.

2. Lastnosti objektov iz realnega sveta so predstavljene z **atributi**, kamor spadajo primarni ključi, obvezni in neobvezni atributi**.**
3. **Povezave med entitetnimi tipi** predstavljajo odnose med posameznimi entitetnimi tipi. Ločimo dve glavni lastnosti povezav:
   - **kardinalnost** nam nudi informacijo o tem, koliko zapisov enega entitetnega tipa je povezano z zapisi drugega entitetnega tipa.
   - Druga lastnost je **eksistenčna** odvisnost, ki nam pove, ali je povezava med entitetnima tipoma obvezna ali ne.

Z izdelanim entitetnim modelom se logična faza izdelave podatkovnega modela zaključi in se prične fizična faza izdelave podatkovnega modela. Prehod iz logične na fizično raven pomeni upoštevanje karakteristik in omejitev strojne in programske opreme. Bistvena razlika med logično in fizično fazo izgradnje informacijskega sistema je v dejstvu, da pri logični fazi zavestno odmislimo tehnologijo, ki jo bomo uporabili pri fizičnem podatkovnem modelu, saj s tem dosežemo večjo neodvisnost bodočega informacijskega sistema od tehnologije (Vintar, 1996, str. 126).

2.3. Fizična faza izdelave podatkovnega modela

Izdelan podatkovni model na logičnem nivoju se pretvori v podatkovni model na fizičnem nivoju, in sicer ob upoštevanju lastnosti baze podatkov, na kateri bo deloval bodoči informacijski sistem, in orodja, s katerim se bo razvil uporabniški vmesnik. V fazi fizične ravni izdelave podatkovnega modela se določijo tuji ključi, indeksi, sprožilci, pogledi in uporabniško določeni tipi podatkov. Podatkovni model se na fizičnem nivoju lahko denormalizira, v primeru, ko to omogoča hitrejše delovanje informacijskega sistema. Podobno kot podatkovni model na logični ravni tudi podatkovni model na fizični ravni ne predstavlja končne verzije podatkovnega modela. Podatkovni model na fizičnem nivoju se navadno dopolnjuje preko celotnega življenjskega cikla razvoja informacijske rešitve, dopolnjevati pa je potrebno tako podatkovni model na fizičnem kot na logičnem nivoju (Purba, 2000, str. 485–487).

Pri fizični fazi podatkovnega modela je ključna organiziranost podatkov na medijih. Obstajata dva načina organiziranosti, in sicer:

- datotečna ter
- baza podatkov.

Datotečna organiziranost je starejša in se dandanes redko uporablja. Zamenjale so jo baze podatkov, kjer so vsi podatki zapisani na enem mestu, po točno določenih pravilih. V nadaljevanju sem se osredotočila na predstavitev baze podatkov, saj so v tem trenutku aktualnejše ter pri informacijskih sistemih pogosteje uporabljene.

Baza podatkov je zbirka, skupina medsebojno povezanih podatkov, ki služijo različnim potrebam neke organizacije in so shranjeni brez nepotrebnega podvajanja (Vintar, 1996, str. 139). Težnja po bazi podatkov se je začela pojavljati pri kompleksnih informacijskih sistemih, ko je velika količina podatkov predstavljala problem iskanja posameznih podatkov, obenem pa gre pri datotečni organizaciji, ob množici podatkov, za večjo porabo prostora na spominskih medijih kot v primeru baze podatkov. Poleg tega so se začeli pojavljati identični podatki z različnimi vrednostmi.

2.4. Tipi struktur podatkovnega modela

Vrste podatkovnih struktur so temelji podatkovnemu modelu. Hierarhična struktura je osnova za hierarhični podatkovni model, mrežna struktura za mrežni podatkovni model, relacijska struktura je temelj relacijskega modela podatkov, pri objektni strukturi pa vpeljuje v bazo podatkov objekte, ki ustrezajo objektom realnega sveta in združujejo podatkovno strukturo, ki se nanaša na neki objekt, s postopki, ki to podatkovno strukturo obdelujejo (Vintar, 1996, str. 147–148).

2.4.1. Hierarhična struktura

Prve hierarhične ali drevesne strukture so se na trgu pojavile že v šestdesetih letih (Kovačič, Vintar, 1994, str. 92). Razvile so se s praktično uporabo, saj je realni svet v večini primerov urejen hierarhično. Iskanje podatka v hierarhični strukturi predstavlja iskanje poti do želenega podatka, zato je pri tem podatkovnem modelu potrebno poznavanje načina logičnega shranjevanja podatkov.
V hierarhičnem modelu poznamo dva koncepta (Barker, 1995, str. 2–10):

- **oče**, ki ima lahko podrejenih več zapisov, nadrejen pa mu je en sam zapis, in
- **sin**, ki predstavlja podrejene zapise. Več zapisov, imenovanih sin, ima lahko istega prednika.

Struktura je v večini primerov uporabe ustrezna, problem se pojavlja pri prikazovanju nehierarhično urejenih povezav (Vintar, 1996, str. 144). V tem primeru je potrebno izbrati primernejšo podatkovno strukturo, saj v primeru nehierarhične urejenosti ni možen dostop do podatkov, ki vnaprej ni bil predviden (Kovačič, 1998, str. 92–93). Kljub omenjeni slabosti se je hierarhična struktura podatkovnega modela veliko uporabljala zaradi njene prednosti, ki je hitra obdelava podatkov, ki imajo v realnem svetu hierarhično strukturo.

2.4.2. Mrežna struktura

Prvi zametki mrežne strukture so nastali že leta 1971, kot rezultat dela CODASYL komiteja (Conferece on Data System Languages), zato se pogosto imenuje CODASYL mrežni model. Kasneje je doživel številne dopolnitve, zadnje leta 1984 (ANSI 1984). Dejansko je

mrežna struktura podobna hierarhični, le da povezave potekajo v vseh smereh, kar tvori mrežo medsebojnih povezav. Podobno kot pri hierarhični, moramo tudi pri mrežni strukturi poznati pot do iskanega podatka.

Ključni prednosti mrežnega modela sta:
- v primerjavi s hierarhično strukturo mrežna struktura omogoča večjo svobodo pri modeliranju podatkov (Vintar, 1996, str. 145).
- Za razliko od relacijskega in hierarhičnega modela, ki omogočata le modeliranje elementarnih atributov, lahko v mrežnem modelu predstavimo tudi sestavljene atribute ter ponavljajoče skupine.

Slabost mrežne strukture podatkovnega modela pa je predvsem v dostopu do podatkov, ki mora biti predviden vnaprej in vgrajen v mrežno strukturo (Kovačič, Vintar, 1994, str. 93–94).

2.4.3. Relacijska struktura

Relacijsko strukturo podatkovnega modela je utemeljil Codd leta 1970. Ima zelo preprosto strukturo in je v primerjavi z drugimi strukturami najbolj formaliziran. Relacijski model je v veliki meri podoben entitetnemu modelu, zato v praksi prihaja do zamenjav konceptov in osnovnih pojmov, vendar pa je med njima pomembna razlika. Relacijski model ima enoznačno matematično podlago, njegovi koncepti so jasno formulirani in omogočajo uporabo relacijske algebre pri izvajanju operacij na modelu. Entitetni model pa je nastal bolj intuitivno, brez enotnega matematičnega formalizma, kar je razlog, da ima v praksi več različnih interpretacij (Kovačič, Vintar, 1994, str. 94–99).

V relacijski strukturi podatkovnega modela so podatki predstavljeni v dvodimenzionalnih tabelah, ki vsebujejo medsebojno povezane podatke. Na podlagi relacijske strukture podatkovnega modela je zaradi »uniformnosti« modela možno zgraditi učinkovite upravljavske informacijske sisteme. Do podatkov pristopamo neposredno z relacijskimi ukazi, in sicer z jezikom SQL, ki je standardiziran jezik za delo z relacijskimi bazami. Iskanje poti v relacijskem podatkovnem modeli je samodejna in jo omogoča sistem relacijskega podatkovnega modela. Do podatkov pristopamo neposredno z ukazi, in sicer z jezikom SQL, namenjenim poizvedovanju. Sicer pa je jezik SQL standardiziran jezik za delo z relacijskimi bazami podatkov.

Relacije si lahko predstavljamo kot dvodimenzionalne tabele, ki vsebujejo medsebojno povezane podatke. Vsaka tabela predstavlja tip entitete, zgrajena je iz:
- **vrstic**, ki predstavljajo posamezen tip entitete ali povezavo,
- **stolpcev**, ki pomenijo posamezne atribute, in
- **vrednosti**, to so elementarni podatki enega atributa določenega entitetnega tipa ali povezave.

- **Glavni, sestavljeni in tuji ključi** v tabelah so potrebni za logične povezave med posameznimi tabelami.

Prednosti relacijske strukture podatkovnega modela so:
- povezave med relacijami niso vnaprej določene in vgrajene v strukturo, kot je pri mrežni in hierarhični strukturi, kar omogoča večjo fleksibilnost pri vzpostavljanju povezav med podatki. Povezave se lahko v vsakem trenutku vzpostavijo v skladu s trenutnimi informacijskimi potrebami (Vintar, 1996, str. 146).
- Relacijska struktura ima matematično strukturo.

Glavna slabost, ki bi jo omenila pri relacijski strukturi podatkovnega modela, bi bila počasno izvajanje poizvedb, še posebej v primeru velike količine podatkov.

2.4.4. Objektno orientiranje strukture

Relacijska, hierarhična in mrežna struktura so zasnovane na skupinah zapisov, ki so med seboj povezani preko kazalcev (hierarhična, mrežna) ali pa logično v relacijah (relacijska). Objekti iz realnega sveta so običajno predstavljeni v več hierarhijah ali relacijah, kar povzroča težave. Čim kompleksnejši so ti objekti, večji je razkorak med objekti realnega sveta in njihovo predstavitvijo znotraj informacijskega sistema. Posledice se kažejo v počasni obdelavi podatkov, ki se nanašajo na posamezni objekt.

Pri vseh doslej omenjenih tehnikah je podatkovna struktura, ki se nanaša na določeni objekt realnega sveta (študent, kupec, vozilo …), povsem ločena od postopkov, ki obdelujejo podatke v strukturi. S tem je možnost napak in nekonsistentnosti pri obdelavi podatkov velika. Objektni pristop odpravlja to slabost tako, da je grajen iz objektov, ki predstavljajo poenostavljeno podobo realnih objektov v realnem okolju s postopkom, ki to podatkovno strukturo obdelujejo. Na ta način dosežemo (Vintar, 1996, str. 147–148):
- objekt je logično zaključena celota podatkovne strukture in postopkov (metod), ki to strukturo obdelujejo;
- objekte je mogoče standardizirati in večkrat uporabiti;
- enostavne objekte je mogoče sestavljati v bolj kompleksne objekte;
- zanesljivost tako izdelanih programskih rešitev je bistveno večja.

Za objektno strukturo podatkovnega modela morajo biti izpolnjeni štirje kriteriji (Borland, 2002, str. 174–176):
1. generalizacija,
2. dedovanje,
3. ograjevanje in
4. mnogoličnost.

2.4.5. Večdimenzionalni model

Večdimenzionalni model v primerjavi z relacijskim modelom omogoča hitrejši dostop do že predhodno agregiranih, večdimenzionalnih podatkov. Izrablja zgodovinske in večdimenzionalne podatke. Zmanjšuje število fizičnih združevanj, potrebnih za poizvedbe (Murtaza, 2000, str. 324).

Bistvo, na katerem sloni večdimenzionalni model, je da se na operativnem nivoju potrebujejo popolnoma drugačne informacije kot na analitičnem. Večdimenzionalni model je enostaven, uporaben je predvsem pri skladiščih podatkov. Podvajanje podatkov se dopušča, še posebej shranjene seštete vrednosti, oblika diagrama pa je običajno zvezdasta.

3. Orodja CASE

Orodja CASE (ang. Computer Aided Software Engineering) so kombinacija grafičnih programskih rešitev, podatkovnega slovarja, generatorja, projektnega managementa in drugih tipov programskih orodij, ki pomagajo razvijalcem razvijati in vzdrževati visoko kvaliteten informacijski sistem (Barker, 1995, str. Gl–2).

Predstavljajo programska orodja, ki avtomatizirajo procesa analize poslovnega področja ter izdelave informacijskega sistema. Informatiku nudijo pomoč pri različnih korakih izdelave informacijskega sistema, in sicer pri (Chou, Chou, 2000, str. 560–561):
- analizi poslovnega področja,
- načrtovanju informacijskega sistema in
- razvoju informacijskega sistema.

Uporaba orodij CASE ključno vpliva na kakovost, produktivnost ter nadzor razvoja informacijskih sistemov, vendar je za kvalitetno delo z orodji CASE potreben usposobljen kader.

3.1. Splošne značilnosti

Orodja CASE so bila razvita iz preprostih, samostojnih risarskih orodij, ki so podpirala risanje in dokumentiranje diagramov, s katerimi so risali strukturne in entitetne diagrame. Korak naprej pri orodjih CASE, v primerjavi s preprostimi risarskimi orodji, predstavlja kombinacija drugih programskih orodij, potrebnih pri razvoju informacijskih sistemov (Barker et al., 1996, str. 364). Komponente orodij CASE so naslednje:
- **orodja za risanje** omogočajo grafični prikaz toka podatkov, procesnih modelov, entitetnih modelov, funkcijskih modelov in struktur podatkovnih modelov.
- **Centralni repozitorij** omogoča integrirano shranjevanje vseh specifikacij, diagramov obrazcev, informacij o vodenju projekta ...

- **Generator poročil** omogoča prikaz informacij, shranjenih v repozitoriju, na različne načine.
- **Orodja za analizo** preverjanja celovitosti, skladnosti, formalne pravilnosti specifikacij, diagramov, obrazcev …
- **Generatorji dokumentacije** izdelajo tehnično in uporabniško dokumentacijo v standardnem formatu.
- **Generatorji kode**: omogočajo avtomatično generiranje programske kode na osnovi podrobnih specifikacij.

Orodja CASE samodejno skrbijo za odpravo podvajanj, omogočajo vzvratno inženirstvo, ponovno uporabljivost programskih modulov in izvorne kode, projektnih planov, prototipnih modelov in načrtovalskih specifikacij.

3.2. Prednosti in slabosti orodij CASE

Orodja CASE so bistveno spremenila način gradnje informacijskih sistemov ter izboljšala njihovo kvaliteto. Ključne prednosti, ki jih prinašajo, pa so (Heričko, 2003, str. 1):

- največja prednost orodij CASE je **večja učinkovitost**, ki je rezultat uporabe že pripravljenih in preizkušenih metod ter programskih modulov, kar vodi v večjo in boljšo usklajenost elementov rešitve, kar posledično privede do
- **večje standardizacije**, kar predstavlja kvalitetnejši razvojni proces.
- **Zmanjša se število napak** ter čas, potreben za odpravo le-teh, kar je posledica večje usklajenosti, napake pa so tudi dovolj zgodaj odkrite.
- Poveča se **kakovost** razvitih informacijskih sistemov.
- Prednosti orodij CASE pred ostalimi načini gradnje informacijskega sistema se odražajo predvsem v **skrajšanju razvojnega časa** informacijskega sistema, saj se programski deli, ki so že bili izdelani, lahko **večkrat uporabijo** (Gradišar, Resinovič, 1996, str. 246).
- Razvijalcem je v veliko pomoč tudi **večja avtomatizacija razvoja**, saj se lahko osredotočijo na kreativnejši del razvoja informacijskega sistema.
- Prednosti orodij CASE pa se ne kažejo le v fazi razvoja, pač pa tudi pri vzdrževanju informacijskega sistema.

Kljub prednostim, ki jih prinašajo orodja CASE, se moramo zavedati, da tovrstna orodja v vsakem primeru niso vedno najboljša rešitev, kajti sama po sebi ne zagotavljajo kvalitetnih informacijskih sistemov ter ne preprečujejo vseh napak (Gradišar, Resinovič, 1996, str. 246). Slabosti orodij CASE:

- **visoki stroški** nabave in uvajanja,
- za delo z orodji CASE je potrebno imeti **usposobljen kader**.

Koristi orodij CASE so navadno dolgoročne, vsekakor pa orodja CASE ne predstavljajo rešitve vseh problemov razvoja informacijskih sistemov. Pri vodstvu, razvijalcih in uporabnikih je potrebno vzbuditi realna pričakovanja.

3.3. Primer orodja CASE: Oracle Designer

Orodje Oracle Designer bom predstavila kot konkreten primer orodja CASE. Razlog za izbor tega orodja je v tem, da sem ga uporabljala pri svojem delu, obenem pa je Oracle Designer v Sloveniji najbolj razširjeno orodje CASE.

Programsko orodje Oracle Designer loči zgornji in spodnji del orodij CASE (Žabkar, 2000, str. 17–38):

1. zgornji del je namenjen strategiji in analizi informatiziranega področja, kamor pa spadajo naslednja orodja:
   a) **procesni modelirnik** je namenjen preučevanju strategije organizacije. Cilj modeliranja procesov je razumevanje sledenja posameznim procesom v organizaciji ter delovanje organizacije kot celote.
   b) **Entitetni modelirnik** je namenjen preučevanju strategije in analizi organizacije.
   c) **Modelirnik dekompozicije poslovnih funkcij v širšem smislu** prikazuje nivoje razčlenitve sistemskih procesov v organizaciji.
   d) **Modelirnik tokov podatkov** prikazuje tok podatkov v organizaciji, kar za razvijalca predstavlja dodaten pogled na celotno organizacijo. Omogoča povezljivost z entitetnim modelirnikom in modelom poslovnih funkcij.
   e) **Matrika kakovosti.**
2. V spodnjem delu imamo povezani razvojni urejevalnik oblikovanja in njegove modele podatkovnih struktur, modulov in aplikacijske logike.

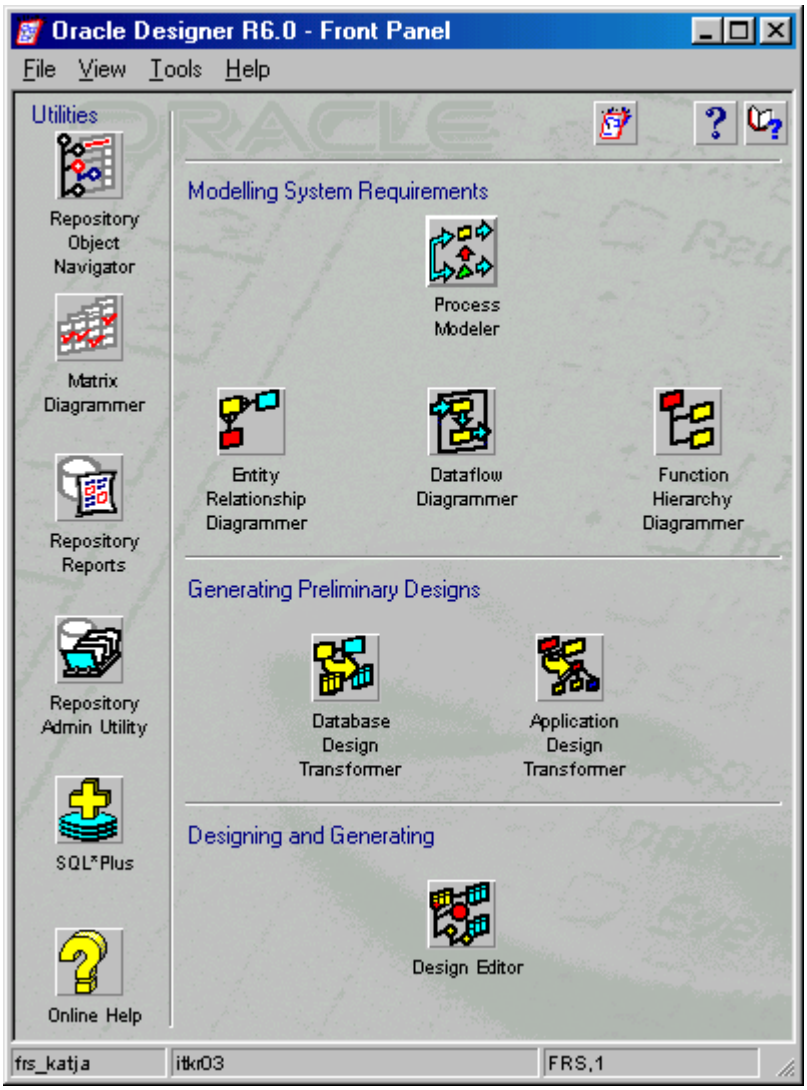
Kot vsako orodje, ima tudi Oracle Designer svoje prednosti in svoje slabosti.

1. Prednosti so:
   a) orodje Oracle Designer velja kot generator najbolj optimalne programske kode, kar je povezano z naslednjimi prednostmi orodja Oracle Designer (Rupnik et al., 1998, str. 28):
      - razvijalcu omogoča **posvečanje ključnim problemom**, s katerimi se pri razvoju programske rešitve srečuje in s tem zmanjšuje količino »ročnega« programiranja,
      - izgub časa s popravljanjem napak v kodi dejansko ni več, saj **se koda generira brez napak**, zato je
      - tretja prednost generiranja **prihranek časa** razvoja programske rešitve, delno zaradi manjšega števila napak, delno zaradi nadomestitve ročnega programiranja z generiranjem kode.

b) Kljub temu da se programski moduli generirajo, je **možno »ročno« poseganje** v generirano kodo, kjer se posamezne SELECT stavke lahko dopolni, spremeni ali doda programske kontrole.
c) **Olajšano je vzdrževanje programskih modulov**, saj se ob vsaki spremembi generira nov programski modul, ki vključi vnešene spremembe.
d) **Standardizacija podatkovnega modela in modulov** se kaže v:
   - poimenovanju objektov,
   - izgledu modulov,
   - uporabniškem vmesniku in
   - v enostavni kontroli upoštevanja sledenju standardom pri razvijalcih.
e) **Močna povezanost** med:
   - podatkovnim modelom in bazo podatkov ter
   - podatkovnim modelom in moduli.
f) Bolj celovita in natančna **sistemska dokumentacija**, saj orodje Oracle Designer poskrbi za shranjevanje le-te med samim razvojem programske opreme.
g) Možna je izdelava **rekonstrukcije** (Rupnik et al., 1998, str. 34–35).
h) Enostavno skrbništvo nad pravicami razvijalcev in uporabnikov po programskih modulih (Vozel, Vožič, 1998, str. 351).

2. Slabosti orodja Oracle Designer pa so:
   a) Oracle Designer je zahtevno orodje, zato je potrebno **veliko časa nameniti šolanju** in usposabljanju razvijalcev programskih rešitev (Žabkar, 2000, str. 27).
   b) Kot drugo slabost bi omenila **visoko ceno** uporabe orodja Oracle Designer, in sicer:
      - visoka cena pridobitve licence za razvijalce informacijskega sistema in
      - visoki stroški izobraževanja razvijalcev programske opreme.
   c) Oracle Designer je **robustno orodje.**
   d) Uporaba orodja Oracle Designer je povezana z visoko stopnjo izrabe razpoložljive strojne opreme.

Slika 2: Čelna maska orodja Oracle Designer



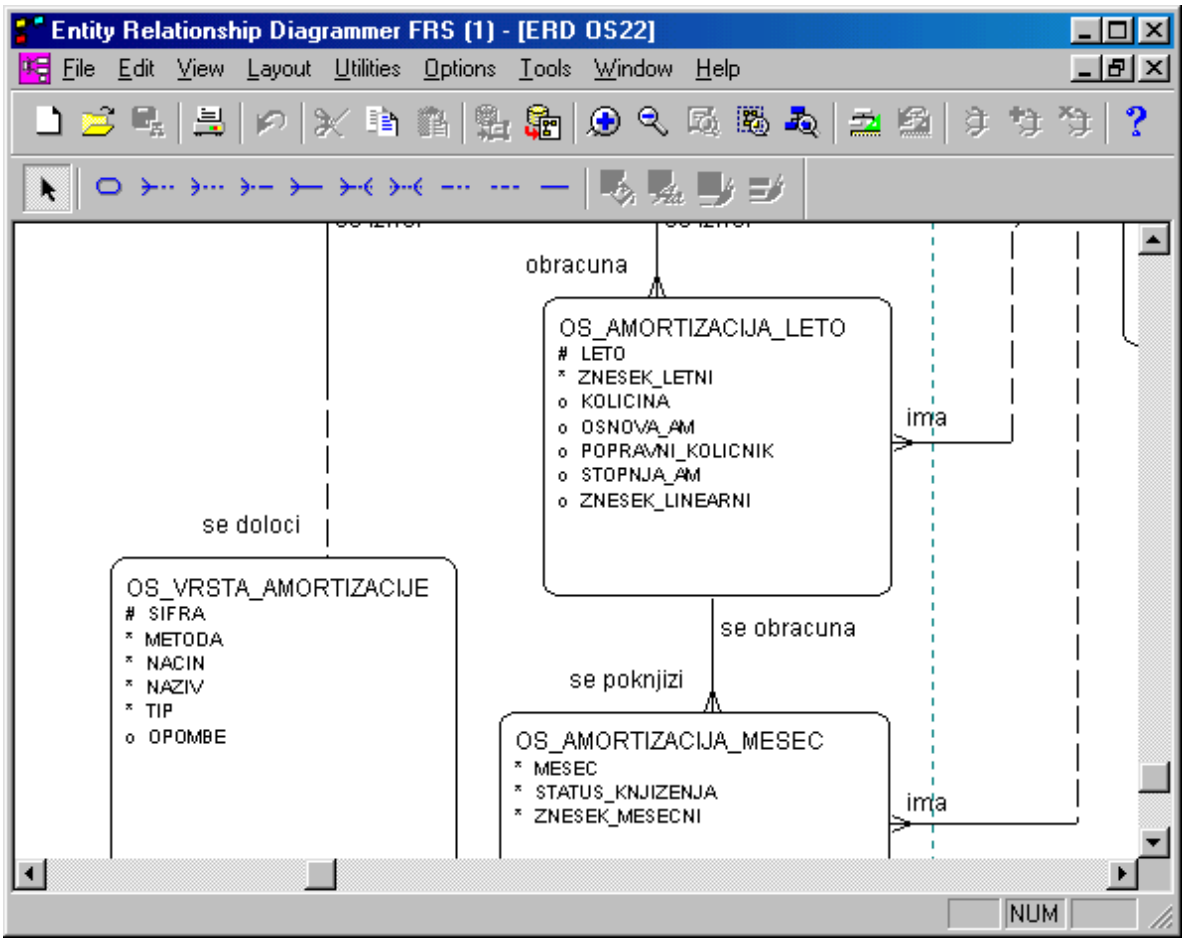
Vir: Interna dokumentacija podjetja Infotrade, 2004

V nadaljevanju poglavja bom predstavila že v začetku poglavja omenjeni zgornji del orodja CASE, ki ga predstavljajo orodja za strategijo in analizo organizacije. Predvsem se bom osredotočila na entitetni modelirnik, saj sem ga v veliki meri uporabljala pri svojem delu.

3.3.1. Entitetni model pri orodju Oracle Designer

Entitetni modelirnik se uporablja za izdelavo entitetnega modela, informatiku omogoča specificirati entitetne tipe in atribute, ki se bodo v bodočem informacijskem sistemu uporabljali, ter obenem določi povezave med posameznimi entitetnimi tipi.

Slika 3: Primer ERM v orodju Oracle Designer

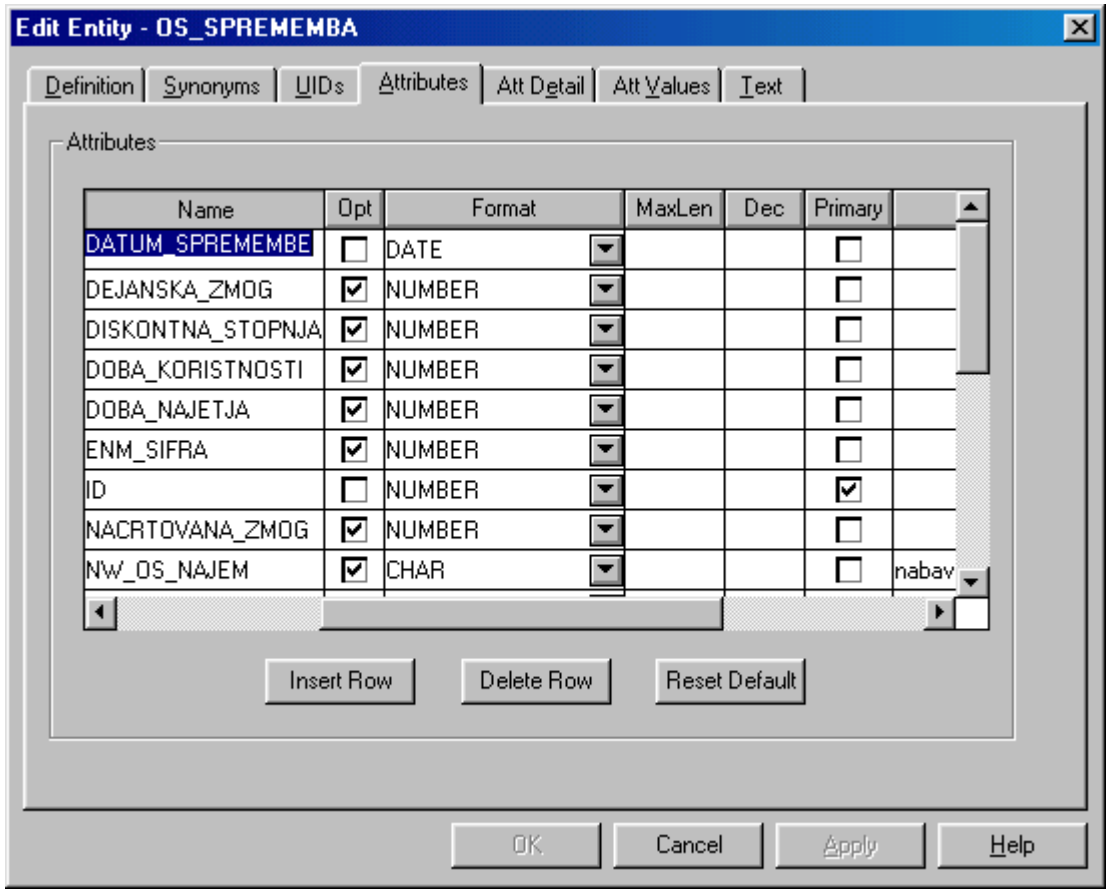


Vir: Interna dokumentacija podjetja infotrade, 2004

Vsi **entitetni tipi** v entitetnem modelu so unikatni, saj nam orodje preprečuje podvajanje imen entitetnih tipov. Entitetni tipi so predstavljene z likom, predstavlja ga na robovih zaobljen kvadrat, ki ima v zgornji vrstici zapisan naziv entitetnega tipa, pod njo pa so navedeni atributi ter simboli atributov. Simboli **atributov** predstavljajo:
unikatno vrednost (primarni ključ),
* obvezno vrednost atributa in
o neobvezno vrednost atributa.

Atributi predstavljajo lastnosti entitetnih tipov, katere določamo preko pogovornega okna. Atributom je potrebno določiti:
- tip podatkov, ki se bodo kasneje vpisovali v polje tabel,
- obveznost vnosa v polje,
- maksimalno dolžino znakov, ki so dovoljeni pri vnosu,
- vnosno masko podatkov (na primer vnosna maska za datum),
- morebitne opombe k posameznim atributom,
- unikatnost atributa entitetnega tipa ter
- imenujemo liste vrednosti, te so v naprej določene vrednosti polja.

Slika 4: Pogovorno okno za določanje atributov entitetnih tipov



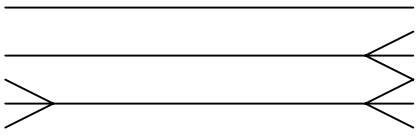
Vir: Interna dokumentacija podjetja Infotrade, 2004

Povezave med entitetnimi tipi v entitetnem modelu ločimo na več vrst, v Oracle Designer entitetnem modelirniku jih označujemo na naslednje načine (Kovačič, Vintar, 1994, str. 111–113):

1. **kardinalnost povezave** označujemo z različnimi spoji povezave in entitetne tipe. Na strani, kjer je možna le ena vrednost, povezavo ponazorimo le z eno črto, kjer pa je možno več vrednosti, pa razvejano.

   Primer:

   a) 1:1,
   b) 1:N,
   c) N:M.



2. **Eksistenčna odvisnost** (obstoj entitetnega tipa je odvisen od obstoja nekega drugega entitetnega tipa):
   a) odvisni tip entitete, ----------
   b) neodvisni tip entitete. _____
3. **Identifikacijska odvisnost** (če je en entitetni tip nemogoče določiti le s pomočjo njenih atributov in potrebuje tudi druge atribute, tujega entitetnega tipa):
   a) samostojna in
   b) nesamostojna.
4. **Obveznost povezave**:

a) obvezna in
b) neobvezna.

V orodju Oracle Designer sem uporabljala predvsem dva tipa orodij, kjer sem:
1. v sklopu orodij za analizo uporabljala entitetni model,
2. v okviru orodja za oblikovanje pa sem uporabila generator za izdelavo podatkovnega modela.

Pri analizi potreb informacijskega sistema se zabeležijo, ugotovijo, določijo in preučijo potrebe bodočega informacijskega sistema. Uporaba orodij za analizo poslovanja organizacije ni možna le za programerje, pač pa tudi za ekonomiste, sistemske analitike, organizatorje dela, kar pomeni, da ga lahko uporabljajo vsi, ki bodo pripomogli k boljšemu razumevanju poslovanje določene organizacije. V orodju Oracle Designer, na področju analize, so funkcijski, entitetni diagram ter diagram toka podatkov (Heap, Speelpenning, 1998, str. 1-5–2-17).

3.3.2. Pretvorba entitetnega modela v podatkovni model

Pri transformaciji ločimo generator baze podatkov ter generator programske rešitve. Generatorji omogočajo izdelavo prototipov.
1. **Generator baze podatkov** avtomatizira izdelavo ter vzdrževanje podatkovnega modela v poslovnem sistemu (Žabkar, 2000, str. 39). Na podlagi vsakega entitetnega tipa se tvori tabela, na podlagi vsakega atributa pa polje v tabeli. Povezave med tabelami v bazi podatkov so zgrajene glede na povezave med entitetnimi tipi v entitetnem modelu.
2. **Generator programske rešitve** pa glede na funkcijski diagram izdela programsko rešitev. Poslovne funkcije predela in nato predlaga kandidate za izpise, procedure, zaslonske maske ter bodoče spletne programske rešitve.

3.3.3. Podatkovni model v orodju Oracle Designer

Vsak entitetni tip se pretvori v tabelo. Za entitetne tipe uporabljamo množinska imena. Unikatni identifikator entitetnega tipa postane primarni ključ tabele. Unikatni identifikator je lahko sestavljen iz kombinacije atributov in/ali povezave med entitetnimi tipi. Opcijske povezave tvorijo kolone, ki imajo lahko prazno vrednost, obvezne povezave pa tvorijo kolone, ki ne smejo imeti prazne vrednosti. Povezave 1:N med entitetnimi tipi naredijo tuje ključe (Barker, 1995, F-1–F-5).

Repozitorij je jedro orodja Oracle Designer, saj so v njem shranjeni vsi objekti in informacije, ki so potrebne za oblikovanje, modeliranje in generiranje baze podatkov ter z njo povezanimi programskimi rešitvami. Za celovito obvladovanje repozitorija so znotraj Oracle Designerja, na voljo orodja, ki nadgrajujejo repozitorij ter omogočajo oblikovanje in

vzdrževanje objektov, shranjenih v njem, in oblikovanje ter pregledovanje povezav med posameznimi objekti.

4. Paketni informacijski sistem Frapis

Začetki Frapisa segajo v leto 1997, ko se je v podjetju Infotrade d.o.o. porodila ideja o izdelavi paketne informacijske rešitve, ki bi bila namenjena srednjim in velikim organizacijam in bi omogočala spremljanje in upravljanje premoženja organizacije. V podjetju je izdelava paketa postala prioritetni projekt in danes se uporablja v nekaterih podjetjih.

Delo na projektu informatizacije celotnega poslovanja organizacije se je razdelilo po poslovnih področij. Najprej so se razvila osnovna poslovna področja, kasneje pa so se razvijala, za poslovanje organizacij, neključna poslovna področja. S tovrstno delitvijo se je izdelava kompleksne rešitve razdelila na več manjših in zato lažje obvladljivih projektov. Pri razvoju vseh poslovnih področij pa so se upoštevali enaki cilji in smernice.

Osnovi cilj pri izdelavi Frapisa je bil izdelati uporabniku prijazen, vendar prek avtorizacije nadzorovan informacijski sistem, ki bi zagotavljal najmanj dnevno ažurne informacije o premoženjskem stanju organizacije, v skladu z zakonodajo ter standardi. Prek uporabniško nastavljivih parametrov bi moral omogočati izdelavo analiz, dokumentov in izpisov. Poleg osnovnega cilja so se pri razvoju upoštevali tudi drugi cilji, ki pa so bili pri razvoju prav tako pomembni in so:

1. dobra povezljivost finančno-računovodskih funkcij z ostalimi poslovnimi funkcijami organizacije.
2. Zmanjšanje obsega ročnega prepisovanja računalniško registriranih podatkov.
3. Enostavno dodajanje in zmanjševanje števila uporabnikov ter dodeljevanje pravic.
4. Avtomatično zmanjševanje obsega operativnih podatkov in
5. dostopnost podatkov za potrebe analiz v daljšem časovnem obdobju ter avtomatizacija postopkov izdelave izhodnih dokumentov in njihovega posredovanja prejemnikom.

4.1. Glavne značilnosti uporabe Frapisa

Vsak uporabnik Frapisa ima svoje uporabniško ime in geslo, s katerim se prijavi v sistem, zato ima vsak uporabnik dostop le do tistega dela programske rešitve, za katerega je pooblaščen. Na enak način se omogoči oziroma onemogoči poizvedovanje, popravljanje, dodajanje ali brisanje zapisov iz baze podatkov. Delovanje vsakega uporabnika se beleži, tako da je možno sledenje, kar olajša ugotavljanje napak.

Ob pregledovanju, vnašanju, spreminjanju in brisanju podatkov se vsaka funkcija odraža na pripadajoči vrstici v tabeli, obenem se izvajajo logične kontrole vpisanih podatkov, ki

skrbijo za konsistenco podatkov v bazi podatkov. Kontrole se praviloma izvajajo ob vnosu podatka v polje, nekatere pa se izvedejo ob potrditvi vnosa. Ob ugotovljeni napaki vnosa ni možno nadaljevati ali potrditi, dokler se napaka ne odpravi ali dokler se zapis, ki vsebuje napako, ne izbriše. Uporabnik ne dela neposredno na podatkovnih tabelah, temveč so vse spremembe shranjene v delovnem pomnilniku računalnika. Spremembe se zapišejo v podatkovno bazo šele takrat, ko jih potrdi uporabnik.

Pri uporabi zaslonskih mask se medsebojno izmenjujeta dva načina:
1. normalni način, kjer se prikazujejo vsi podatki iz baze podatkov, in
2. poizvedovalni način, kjer prikazujemo le podatke, ki ustrezajo določenim pogojem, ki jih narekuje uporabnik.

4.2. Frapisovo pokrivanje poslovnih področij

Proces razvoja paketnega informacijskega sistema Frapis je v tem trenutku na točki, ko podpira osnovna ter nekatera stranska poslovna področja poslovanja organizacije, zato se že uporablja v nekaterih podjetjih v Sloveniji. Področja poslovanja v programski rešitvi Frapis se v grobem ločijo na dva dela:
1. prvi del imenujemo **jedro sistema**, vključuje tiste module, ki omogočajo operativni nadzor nad premoženjskim stanjem organizacije. Osnovni namen je spremljati tekoče poslovanje in pripravljati podatke za računovodsko poročanje.
2. **Nadgradnja sistema** pa je namenjena izvajanju občasnih, pomožnih knjigovodskih funkcij, izmenjavi podatkov z drugimi programskimi rešitvami in pripravi izvlečkov za upravljavske in nadzorne sisteme.

4.3. Osnovna sredstva v programski rešitvi Frapis

Zgoraj omenjena poslovna področja so v paketni programski rešitvi Frapis že informatizirana, veliko poslovnih področij pa v programski rešitvi še ni razvitih. Eno izmed takih poslovnih področij so osnovna sredstva. V nadaljevanju se bom osredotočila na projekt informatizacije poslovnega področja osnovnih sredstev, za katerega sem izdelala podatkovni model.

Projekt informatizacije osnovnih sredstev sem zaradi večje preglednosti in lažje obvladljivosti razdelila na več aktivnosti:
1. prva aktivnost je zajemala zbiranje informacij o področju osnovnih sredstev in ugotavljanje potreb uporabnika.
2. V času druge aktivnosti sta vzporedno potekali dve aktivnosti:
   a) izdelava podatkovnega modela na logični ravni ter
   b) pisanje specifikacije za programske module.
3. Na podlagi dokončno izdelanega podatkovnega modela na logični ravni sem izdelala podatkovni model na fizični ravni.

4. V okviru četrte aktivnosti so se iz podatkovnega modela na fizični ravni, s pomočjo generatorjev, tvorile tabele bodoče programske rešitve. Zgenerirane tabele so v tem delu razvoja programske rešitve prazne in predstavljajo le strukturo baze bodočega informacijskega sistema.
5. Predpogoj za začetek pete aktivnosti, ki predstavlja programiranje programskih modulov, je zgrajena baza podatkov ter napisana specifikacija programskih modulov.
6. Šesta aktivnost je obsegala načrt testiranja in testiranje izdelane programske rešitve, z njo sem odkrivala napake in pomanjkljivosti.
7. V zadnji aktivnosti sem napisala uporabniška navodila, ki predstavljajo eno izmed pomoči uporabnikom ob delu s programsko rešitvijo.

Moje delo na projektu informatizacije področja osnovnih sredstev se je z izdelavo podatkovnega modela prekinilo, zaradi česar se bom v nadaljevanju osredotočila zgolj na predstavitev aktivnosti, povezane z izdelavo podatkovnega modela. Za izvedbo vmesnih aktivnostih so bili zadolženi sodelavci, zato jih bom le na kratko predstavila.

4.3.1. Potek analiziranja poslovnega področja osnovnih sredstev

Zaradi občutljivosti projektov na napake v začetnih fazah izdelave sem analizi poslovnega področja namenila veliko pozornosti. Za osvojitev znanja poslovnega področja osnovnih sredstev sem skušala vključiti čim več virov informacij:

1. glavni vir informacij je bila literatura v zvezi s poslovnim področjem. Oprla sem se predvsem na Slovenske računovodske standarde, Uradni list Republike Slovenije in študijske knjige povezane s področjem osnovnih sredstev.
2. Vir informacij, ki bi si ga želela vključiti v svojo analizo, bi bili intervjuji z bodočimi uporabniki programske rešitve, vendar izdelava paketnega informacijskega sistema za neznanega naročnika bodočih naročnikov informacijskega sistema ne pozna, zato ni možno opiranje na njihove predloge, pojasnila, mnenja in potrebe glede programske rešitve. Odsotnost bodočih uporabnikov informacijskega sistema me je prisilila, da sem poiskala druge skupine ljudi, ki so nadomestili omenjeni vir informacij. Poiskala sem dve skupini, in sicer:
   a) prva skupina so bili potencialni uporabniki programske rešitve, kamor spadajo osebe, ki iz praktičnega vidika poznajo področje osnovnih sredstev a pri svojem delu uporabljajo drug informacijski sistem. Ugotovila sem prednosti in slabosti drugih informacijskih sistemov, na kar sem se oprla pri postavljanju zahtev za lastno programsko rešitev.
   b) Druga skupina so bili sodelavci, ki poznajo poslovno področje osnovnih sredstev in imajo izkušnje iz preteklih projektov.
3. Tretji vir informacij, ki sem ga pri svojem delu uporabila, so bili opisi drugih programskih rešitev za področje osnovnih sredstev. Slednje sem pregledovala in si zapisovala, katere potrebe podpirajo, da bo nova programska rešitev imela vse lastnosti

obstoječih programskih rešitev ter podprla dodatne potrebe, ki bodo morda zanimive za trg.

Največji problem v fazi analiziranja poslovnega področja se mi je zdela odsotnost bodočih uporabnikov, saj so se mi v prvih korakih te faze zdeli ključni vir informacij. Ko sem analizirala področje osnovnih sredstev, sem ugotovila, da se že med strokovnjaki razlikujejo mnenja o poslovnem področju. Zelo so različna in večinoma ozko usmerjena pa so bila pri potencialnih uporabnikih programske rešitve. Na tej točki se je izkazalo, da je literatura predstavljala najbolj objektiven vir informacij. Mnenja in nasvete strokovnjakov ter potencialnih uporabnikov sem upoštevala kot vodilo pri delu, vseeno pa niso bili ključnega pomena.

Ugotovila sem, da je nastala analiza poslovnega področja zajemala širši vidik celotnega področja osnovnih sredstev, kot bi ga v primeru, če bi bili glavni vir informacij bodoči uporabniki informacijskega sistema. V primeru gradnje informacijskega sistema za neznanega naročnika pa je široko zajemanje potreb, velika fleksibilnost in nastavljivost informacijskega sistema bodočemu naročniku ključnega pomena. Odsotnost bodočih uporabnikov se mi zaradi zgoraj omenjenih razlogov ne zdi več pomanjkljivost izdelave paketnega informacijskega sistema, saj je možno poiskati druge, morda objektivnejše vire informacij glede poslovnega področja.

Rezultat analize področja osnovnih sredstev iz zgoraj navedenih virov je bil opis funkcionalnosti bodoče programske rešitve. Opis mi je v naslednjih korakih pomagal pri pisanju navodil za programerje, risanju entitetnega modela, testiranju modulov, pisanju uporabniških navodil ter nenazadnje pri uvajanju uporabnikov v delo z novim delom informacijskega sistema.

4.3.2. Izdelava entitetnega modela za področje osnovnih sredstev

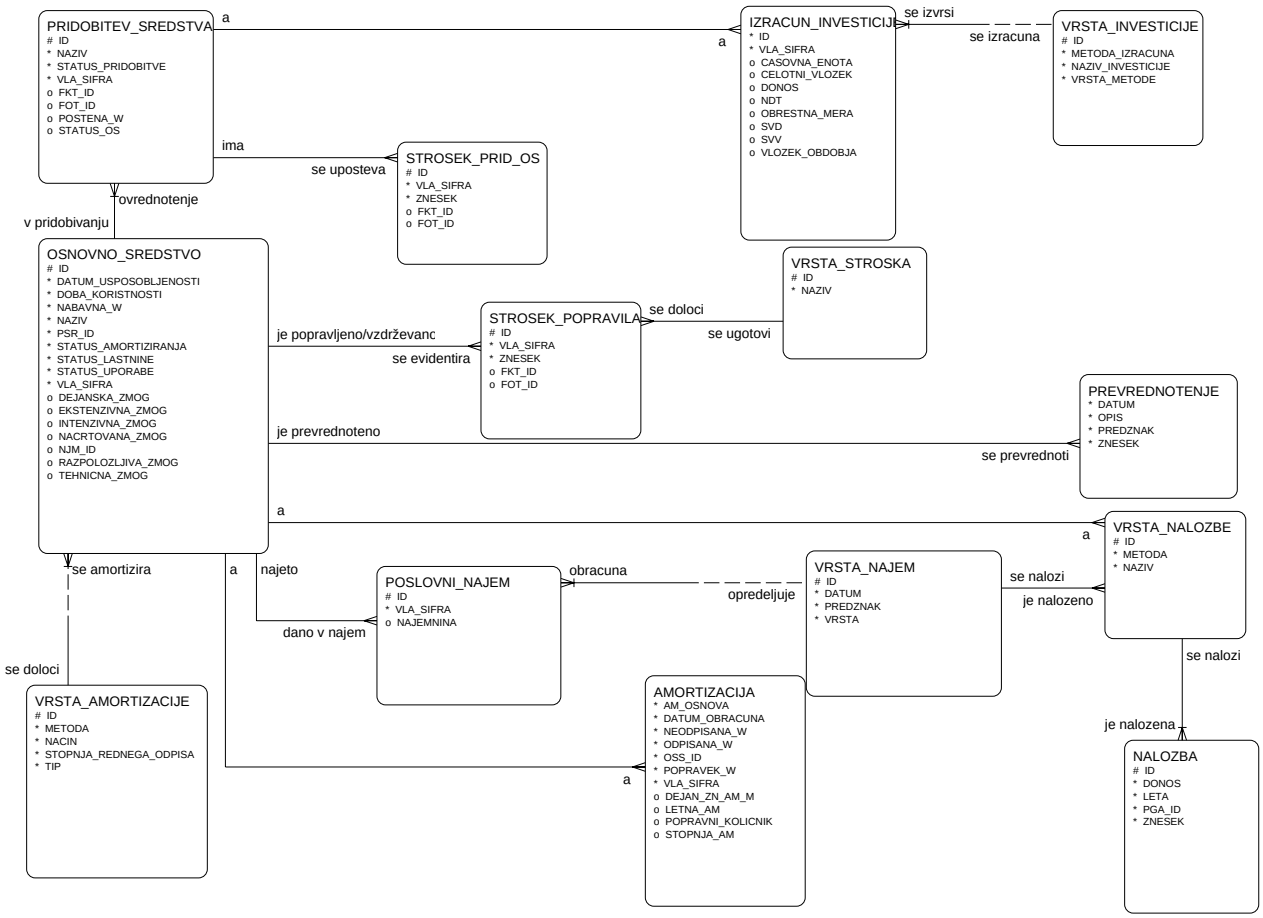
Po končanem preučevanju področja osnovnih sredstev sem se lotila risanja entitetnega modela. Dela sem se lotila po posameznih korakih, delno zaradi obsežnosti dela, delno pa zaradi pričakovanja, da prvi izrisani entitetni model ne bo najboljši in ga bo treba večkrat popraviti ter dodelati. Tako sem izdelavo entitetnega modela razdelila na naslednje faze, ki jih bom v nadaljevanju podrobneje razložila.

1. V prvi fazi sem narisala entitetne tipe, za katere sem ugotovila, da bodo potrebne, jim dodala ustrezne atribute, entitetne tipe pa povezala med seboj. Prvi entitetni model je bil grob osnutek entitetnega modela, ki je nastal v kasnejših fazah.
2. Naslednja faza je predstavljala podrobnejše posvečanje pozornosti posameznim elementom entitetnega modela in s tem popravljanje in spreminjanje osnutka entitetnega modela.
3. V tretji fazi, ko je entitetni model dobil pravo obliko, sem atributom določila lastnosti, ki jih bodo kasneje imeli podatki, shranjeni v posameznih poljih, jim določila tipe

bodočih podatkov (number, varchar2, date ...) ter dolžine polj. Prav tako je bilo potrebno določiti, ali je posamezno polje ključ ali je obvezno oziroma neobvezno in nekatere druge lastnosti.

4. V zadnji fazi se je določenim poljem določila domene, kar pomeni, da so imeli določene vrednosti, tipe in dolžine.

Slika 6: Prva verzija entitetnega modela.



Vir: Informacijski sistem Frapis, 2004

Kot sem že omenila, sem v prvi fazi izdelave entitetnega modela izrisala **osnutek bodočega entitetnega modela** za paketno programsko rešitev področja osnovnih sredstev. Problem, s katerim sem se srečala pri tem, je bil, da nisem vedela, kako bi začela z risanjem entitetnega diagrama. Odločila sem se, da bom najprej poiskala najbolj očitne entitetne tipe, jim dodala atribute, kasneje, ko bi imela več entitetnih tipov, pa jih tudi povezala. Tako sem tudi naredila in najprej poiskala entitetne, kot so:

- OSNOVNO_SREDSTVO,
- NAHAJALISCE,
- VRSTA_AMORTIZACIJE,
- VRSTA_STROSKA ...

kasneje pa sem izrisala tudi manj očitne entitete kot so:

- OBRACUNI_AMORTIZACIJ in
- PRIDOBIVANJA.

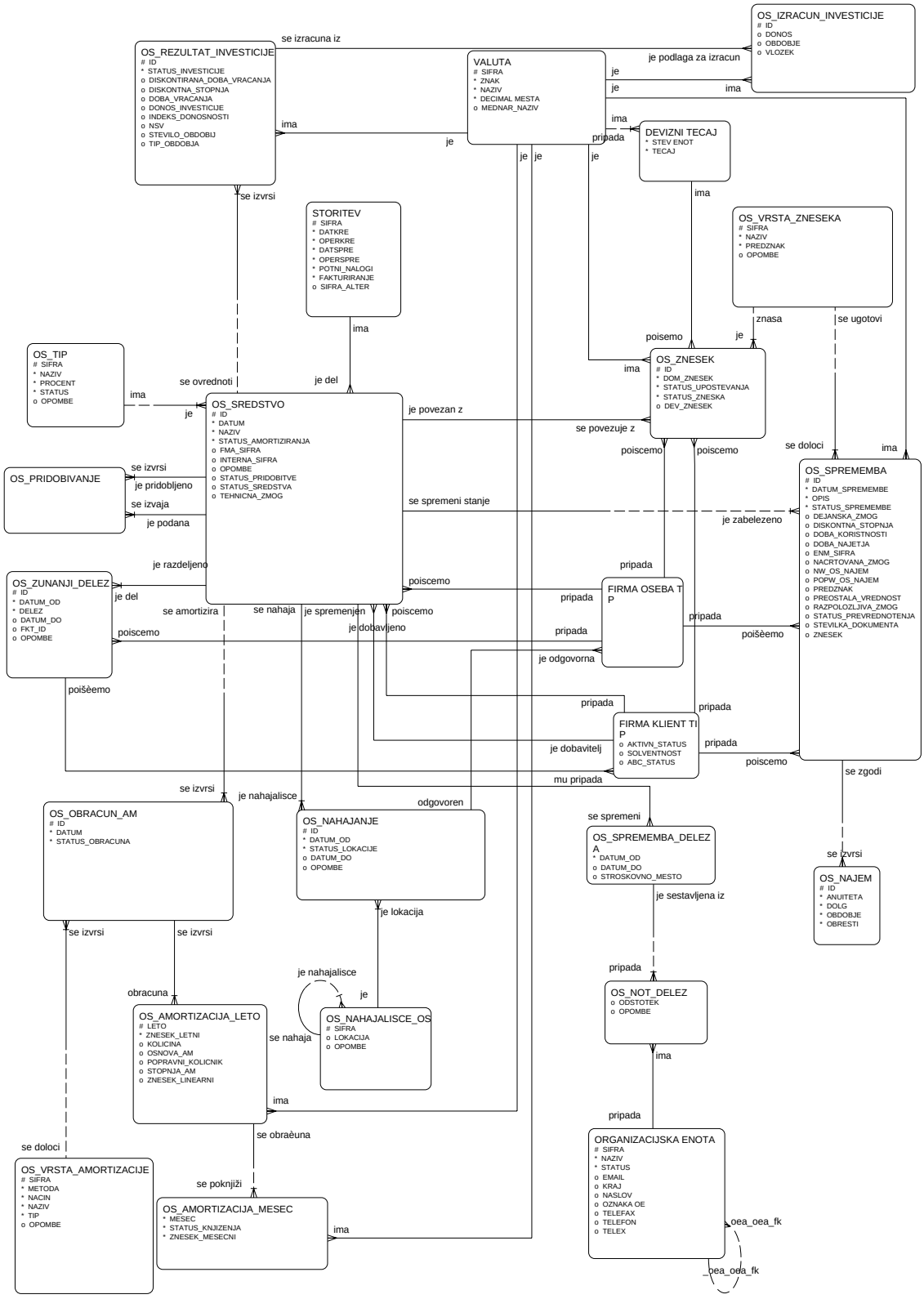
Izrisanim entitetnim tipom sem določila atribute, za katere se je izkazalo, da bodo v programski rešitvi potrebni. Večinoma so bili to enostavni in očitni atributi, kot lahko vidimo na sliki 6**.** V kasnejših fazah izdelave entitetnega modela sem dodajala manj očitne entitetne tipe in atribute. Sočasno z izdelavo entitetnih tipov sem risala tudi povezave med entitetnimi tipi. Pri prvih entitetnih modelih so mi povezave služile zgolj za orientacijo glede povezanosti med posameznimi entitetnimi tipi. Nisem se ukvarjala s tipi in poimenovanji povezav med entitetnimi tipi, kar lahko vidimo na sliki 6, kjer so imena nekaterih povezav označena s črko »a«.

Ko sem dokončala osnutek entitetnega modela, sem pričela z drugo fazo izdelavo le-tega, s katero sem pozornost iz celotnega poslovnega področja preselila na **posamezne dele področja osnovnih sredstev**, ki pa so:

- pregledovanje, pridobivanje in vodenje osnovnih sredstev,
- amortiziranje,
- nahajanje,
- najemanje in oddajanje osnovnih sredstev v najem,
- investicije v osnovna sredstva in nekatere druge.

Poglobila sem se v posamezna poslovna področja in glede na ugotovitve narisala in popravila entitete, atribute in povezave. Slednjim sem določila tip povezave in jih poimenovala. Veliko pozornosti sem namenila določanju obveznosti povezave med entitetnima tipoma ter preverjanju, ali bo tip povezave omogočal pregledovati želene podatke ali ne. V tej fazi sem obenem dodala precej atributov ter nekaj entitetnih tipov, ki jih v prehodni fazi nisem predvidela.

Slika 7: Vmesna verzija entitetnega modela



Vir: Informacijski sistem Frapis, 2004

Tretja faza izdelave entitetnega modela je bila sestavljena iz treh delov:

1. Določanje **lastnosti atributov.** Vsakemu atributu se določijo naslednje lastnosti:

a) ali je atribut primarni ključ ali ne,
b) ali je vnos obvezen ali ne,
c) tip polja vrednosti bodočih podatkov,
d) dolžino polja in
e) vnosne maske nekaterih atributov.

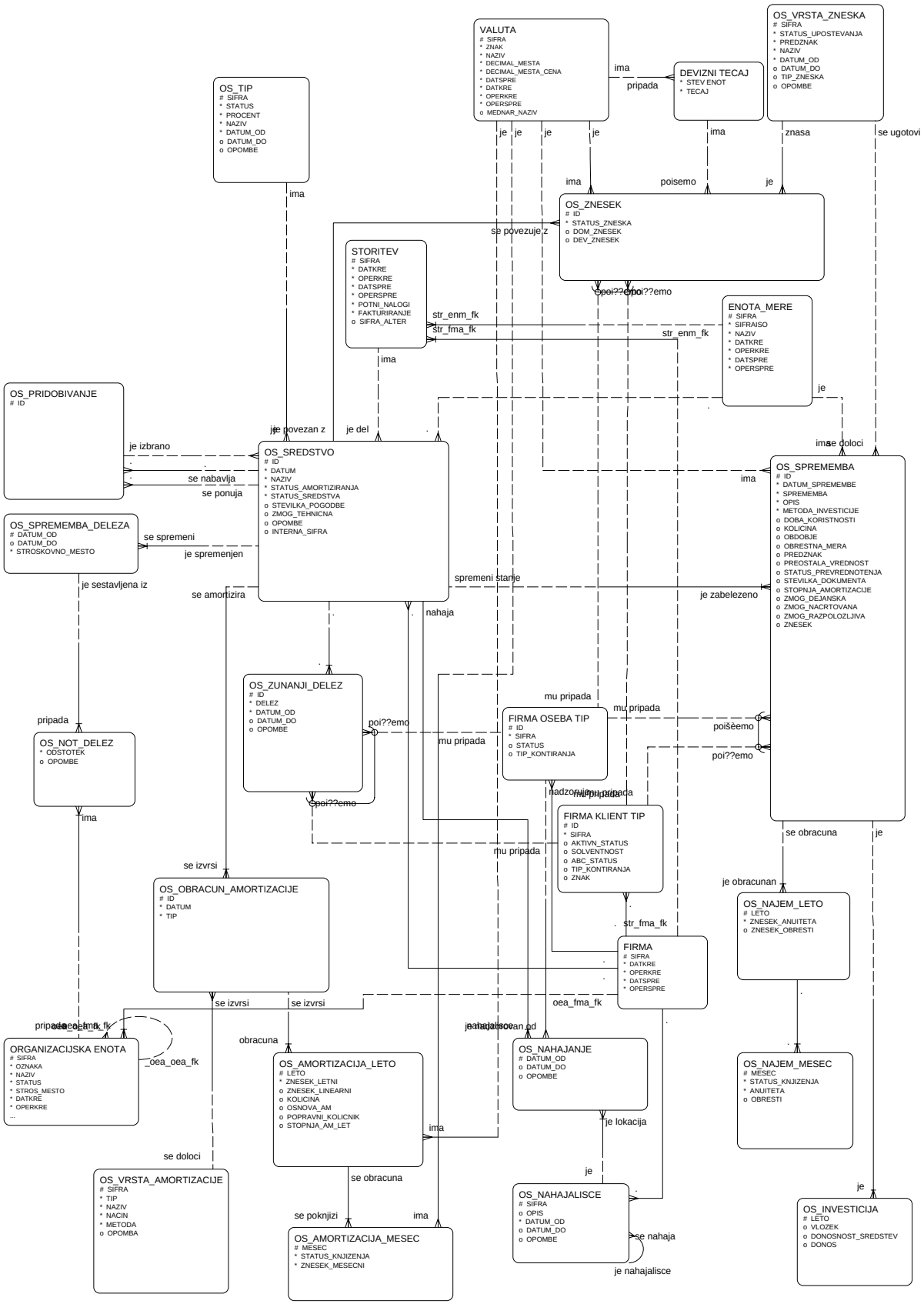
2. Izdelava list vrednosti za posamezne atribute, ki bodo v programski rešitvi namenjeni pomoči uporabniku pri vnosu podatkov. Lista vrednosti vpeljuje niz logičnih vrednosti, rang vrednosti za določeno polje v programski rešitvi. Vrednosti za polja so shranjene v posebni tabeli vrednosti in jo uporabimo vedno, ko je na nekem polju v informacijskem sistemu vnos omejen z listo vrednosti. Liste vrednosti imajo dve bistveni pozitivni lastnosti:
   a) prva je preprečevanje napak ob vnosu podatkov v informacijski sistem,
   b) druga pa je večkratna uporabljivost domen, vnešenih v tabelo vrednosti.
3. Zadnji del pa je bil namenjen povezovanju poslovnega področja osnovnih sredstev s celotnim informacijskim sistemom Frapis, kar sem naredila prek skupnih šifrantov ter vmesne tabele. Končna verzija entitetnega modela je prikazana na sliki 8.

Načeloma atributom ni obvezno določanje zgoraj naštetih lastnosti, še posebno v primeru uporabe entitetnega modelirnika v orodju Oracle Designer le za grafično prikazovanje. V primeru kasnejšega pretvarjanja entitetnega modela v podatkovni model, je potrebno določiti lastnosti atributov.

Sočasno z risanjem entitetnega modela sem pisala specifikacijo programskih modulov. Razlog za vzporedno delo je bil predvsem v tem, da mi je bilo najlažje napisati specifikacijo v trenutku, ko sem se ukvarjala z določenim področjem programske rešitve, vsako ponovno vračanje je terjalo več časa in truda. Predstavljivost bodočega delovanja programske rešitve se mi je v tistem trenutku zdela največja in sem zanjo potrebovala najmanj časa. V specifikacijah sem narisala izgled ekranskih mask in izpisov, določila, kateri entitetni tipi in atributi, kasneje tabele in polja, se bodo kje uporabljali, ter poskrbela za kontrole, ki se morajo ob vnosu ali potrditvi polja ali zapisa izvršiti. Take kontrole so na primer: preverjanje vrednosti atributa, ki ga vnese uporabnik, dolžina vnosa ali samodejno izpolnjevane določenih vrednosti.

Nekatere kontrole so namenjene opozarjanju na neobičajnost vnosa določenih vrednosti, ki bi lahko bil napačen, ni pa nujno, druga pa preprečevanju nadaljevanja vnašanja v primeru, da se želi zaključiti z vnašanjem, ko obvezna polja še niso bila izpolnjena ali ko vrednosti ne ustrezajo predvidenim vrednostim.

Slika 8: Končna verzija entitetnega modela



Vir: Informacijski sistem Frapis, 2004

4.3.3. Podatkovni model

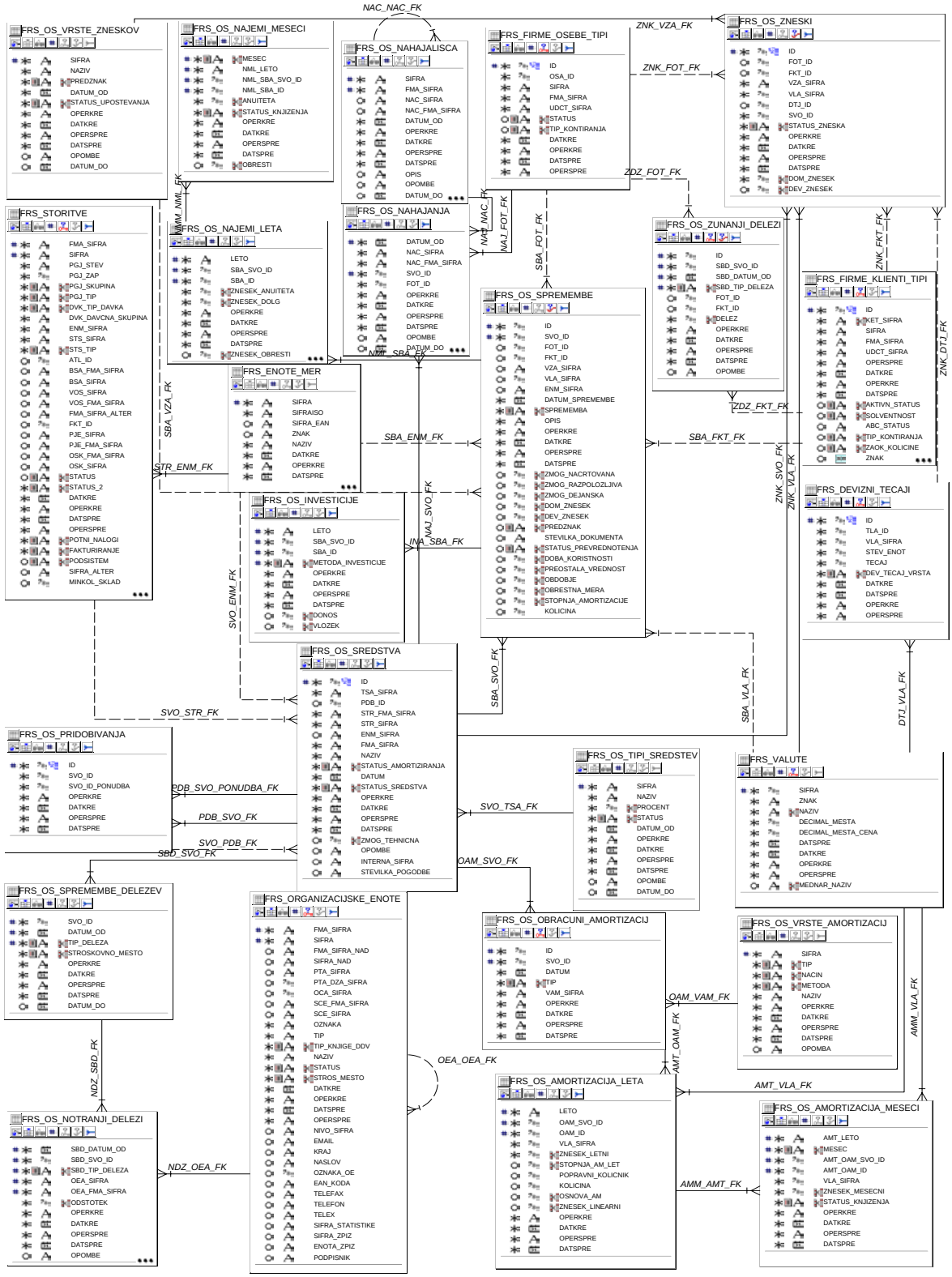
Po končani fazi izdelave entitetnega modela se je prek generatorjev orodja Oracle Designer tvoril podatkovni model, ki je predstavljal strukturo baze podatkov, bodoče programske

rešitve in olajša vzdrževanje baze podatkov (Kovačič, 1998, str. 165). V procesu pretvarjanja entitetnega modela v podatkovni model so se zgodile tri aktivnosti:

1. vsi entitetni tipi entitetnega modela so se pretvorili v tabele.
2. Atributi entitetnih tipov entitetnega modela se med generiranjem pretvorijo v polja tabel. Polja, označena kot unikatni identifikatorji v novih tabelah, predstavljajo primarne ključe.
3. Povezave med entitetnimi tipi pa v podatkovnem modelu predstavljajo unikatne identifikatorje in tuje ključe.

Rezultat pretvarjanja entitetnega modela v podatkovni model je izdelana struktura baze podatkov, kar je osnova za izdelavo programske rešitve, ki predstavlja uporabniški vmesnik za vnos, spreminjanje, brisanje in obdelavo podatkov. Stolpci novonastale tabele predstavljajo vrsto podatkov, vrstice pa posamezen zapis v tabeli.

Slika 9: Podatkovni model paketnega informacijskega sistema Frapis za področje osnovnih sredstev



Vir: Informacijski sistem Frapis, 2004

4.3.4. Kasnejše faze izdelave programske rešitve

Po končani fazi izdelave podatkovnega modela sledi faza izdelave grafičnega vmesnika ter programiranje. Orodje Oracle Designer omogoča prek nastavljanja parametrov generiranje zaslonskih mask ter izpisov, tako da je programiranje programske rešitve skrčeno na izdelavo kontrol, funkcij in procedur. Izdelava programskih modulov se izdela glede na napisana navodila za programerje, ki sem jih začela pisati že v fazi analiziranja poslovnih potreb.

Vsak izdelan programski modul je predstavljal začetek naslednje faze, faze testiranja posameznih modulov. Sintakso modulov je preveril programer, ki je izdelal zaslonsko masko ali izpis, logično preverjanje ustreznosti je bilo potrebno narediti naknadno. Pri načrtu testiranja posameznih modulov kot tudi testiranja celotnega področja osnovnih sredstev sem si pomagala z navodili za programerje, kjer je bilo natančno zapisano, kaj morajo posamezni moduli omogočati. Faza testiranja in popravljanja odkritih napak sta se večkrat ponovili, dokler moduli niso bili ustrezni.

Sledilo je pisanje navodil za pomoč pri uporabi nove programske rešitve, kar sem naredila vzporedno s testiranjem programske rešitve, dokončno pa sem jih napisala po končanem testiranju programske rešitve za področje osnovnih sredstev. Vsekakor pa testiranje poslovnega področja ni dokončen test, potrebno je še testiranje celotnega informacijskega sistema. Podjetje Infotrade d.o.o. testira sistem predvsem z lastno uporabo programske rešitve Frapis, kajti na dejanskih podatkih se najlažje odkrije napake. Po uspešnem testiranju celotne paketne informacijske rešitve Frapis sledi za paketne programske rešitve specifična faza iskanja naročnikov novega informacijskega sistema.

Po vseh uspešno opravljenih predhodnih fazah se prične faza nameščanja in uvajanja informacijskega sistema novemu naročniku. Programska rešitev Frapis se je uvajala glede na dogovor z naročnikom, tako da so zabeležene vse štiri vrste prehodov na novi sistem:

1. vzporedni,
2. neposredni,
3. postopni in
4. pilotni prehod.

Novo poslovno področje osnovnih sredstev bodo stari naročniki lahko dogradili v informacijski sistem, kar pa ni obvezujoče.

Z začetkom uporabe novega informacijskega sistema se začenja podfaza testiranja ustreznosti, ko uporabniki informatike seznanijo z odstopanji novega informacijskega sistema od njihovih pričakovanj ter dodatnimi potrebami, ki se pojavijo kasneje. Sledi faza vzdrževanja programske rešitve, kamor sodijo popravki in dopolnitve informacijskega sistema. Naročniku je potrebno nuditi pomoč pri delu, ki je pri paketnem informacijskem sistemu sestavljena iz treh delov:

1. prvi del je uporabniški priročnik, ki se med delom dopolnjuje;

2. največkrat uporabljeni del pa je telefonska pomoč uporabnikom, ki imajo tehnične ali vsebinske težave, za kompleksnejše probleme je namreč potrebna pomoč strokovnjaka.
3. Zadnja vrsta pomoči pa je javljanje napak prek internetne strani.

5. Sklep

Poslovno okolje podjetja se ves čas spreminja, zato je ključno, da so informacijski sistemi zgrajeni kvalitetno in fleksibilno. Za dosego tega cilja se morajo pri razvoju informacijskih sistemov informatiki držati določenih postopkov, pri čemer ni pomembno, ali se gradi informacijski sistem za znanega ali neznanega naročnika. Pri gradnji paketnega informacijskega sistema je še posebej pomembna zagotovitev visoke fleksibilnosti in nastavljivosti informacijskega sistema, kar pa je posledica dobro izdelanega podatkovnega modela.

Pri razvoju podatkovnega modela mora imeti razvijalec ves čas pred očmi dejstvo, da bodo informacijski sistem uporabljali različni tipi organizacij, zato je nujno, da se v podatkovni model vgrajujejo minimalne predpostavke o bodočem naročniku. Pri izdelavi podatkovnega modela se tak način dela kaže v uporabljanju velikega števila parametrov, ki jih nastavljajo ob namestitvi informacijskega sistema. Poleg tega je potrebno upoštevati, da različni tipi organizacij pri poslovanju uporabljajo različne načine in postopke, ki jih je potrebno vgraditi v informacijski sistem. Vsekakor se razvijalec ne sme odločiti za enega izmed njih, ampak je potrebno upoštevati vsakega izmed njih ali pustiti možnost kasnejšega dograjevanja rešitve.

V splošnem ima izdelava paketnega informacijskega sistema nekatere slabe in nekatere dobre strani. Slaba stran paketnih informacijskih sistemov se kaže v verjetnosti pridobitve naročnika v prvih obdobjih ponudbe paketa trgu, ki je dokaj majhna, saj se podjetja rada odločajo za znane paketne informacijske sisteme, najraje tuje. Razlog je v tem, da so dlje časa na tržišču in zato bolj priznana ter cenjena, obenem pa dosegajo bistveno višjo ceno kot podobne, manj znane paketne programske rešitve. Tako ravnanje podjetij je seveda razumljivo, saj je odločitev za neko programsko rešitev v podjetjih praviloma velika investicija ter obenem ključ za uspešno ali neuspešno poslovanje organizacije v prihodnosti.

Problem, ki sem ga zaznala pri paketnih informacijskih sistemih, je fleksibilnost končnega informacijskega sistema. Vsaki organizaciji se najbrž najbolje prilega informacijski sistem, ki je razvit le zanjo. Z nakupom paketnega informacijskega sistema se lahko potrebe naročnika ter funkcije, ki jih paket omogoča, le bolj ali manj skladajo, saj naročnik pri razvoju paketa ni bil udeležen. Kljub temu pa je naročnik udeležen pri prilagoditvah in nastavitvah parametrov za organizacijo, kar zmanjša odstopanja od izgradnje informacijskega sistema posebej zanjo. V primeru, da bi se organizacija odločila za

izgradnjo informacijskega sistema, pa je smiselno vprašanje, ali bi se to organizaciji časovno in cenovno izplačalo.

Klub temu da ima metoda izdelave paketnega informacijskega sistema dejansko kar nekaj slabih strani, sem mnenja, da prevladajo pozitivne lastnosti. Izgradnja programskih rešitev za neznanega naročnika je zelo primerna in po mojem mnenju najprimernejša, ko gre za srednje in velike informacijske sisteme. V primeru, ko organizacija želi imeti celovit informacijski sistem, zgrajen posebej zanjo, je to v realnem času običajno nemogoče. Poslovno okolje se zelo hitro spreminja, v primeru razvoja bi morda ob dokončno izdelanem informacijskem sistemu morda ugotovili, da je le-ta zastaral ali da so že potrebne prilagoditve in nadgradnje.

Ocenjujem, da je srednje velikim in velikim podjetjem vredno izbrati investicijo v paketni informacijski sistem, in sicer, da realno skušajo ugotoviti, kaj jim posamezni paket nudi, katere informacijske potrebe pokriva, kolikšna je fleksibilnost informacijskega sistema. Pomemben dejavnik pri odločanju za informacijski sistem je prav gotovo cena paketa, vendar cena sama večkrat ni merilo za kvaliteto programske rešitve.

Literatura

1) Albanna J. Sami, Osterhaus Joe: Meeting the Software Challenge. IS Management handbook. Auerbach, Boca Raton: 2000. 788 str.

2) Barker Richard, Barker Barbara, Longman Cliff: CASE*Function and Proces Modelling Workingham. Oracle Corporation UK Limited, 1996.

3) Barker Richard: CASE*Metod Entity Relationship Modelling. Workingham: Oracle Corporation UK Limited, 1995. 128 str., 10 pril.

4) Cerovšek Mitja, Jevšček Matej: Procesni pristop prenove in informatizacije poslovanja v skupini TPV. Uporabna informatika, Ljubljana, X (2002), 4, str. 210–217.

5) Chou C. David, Chou Y. Amy: Analyses of Software Quality and Auditing. IS Management handbook. Auerbach, Boca Raton: 2000. 788 str.

6) Golob Izidor et al.: Analiza in primerjava pristopov pri gradnji podatkovnih skladišč. Uporabna informatika, Ljubljana, XI (2003), 1, str. 12–22.

7) Gradišar Miro, Resinovič Gortan: Informatika v poslovnem okolju. Ljubljana: Ekonomska fakulteta, 1996. 479 str.

8) Hale Ron: End-User Computing Control Guidelines. IS Management handbook. Auerbach, Boca Raton: 2000. 788 str.

9) Jacobson Ivar et al.: Object-Oriented Software Engineeting. Harlow: Addison-Wesley, 1996. 528 str.

10) Jurca Matjaž: Vožnja z vrtiljakom. Zbornik posvetovanja DSI 2000. Ljubljana: Slovensko društvo informatika, 2000, str. 411–419.

11) Kovačič Andrej: Informatizacija poslovanja. Ljubljana: Ekonomska fakulteta, 1998. 214 str.

12) Kovačič Andrej, Vintar Mirko: Načrtovanje in gradnja informacijskih sistemov. Ljubljana: DZS d.d., 1994. 316 str.

13) Murtaza H. Ali: A Framework for Developing Enterprise Data Warehouses. IS Management handbook. Auerbach, Boca Raton: 2000. 788 str.

14) Pressman S. Roger: Software Engineering and Database. New York: The McGraw-Hill Companies, 1997. 852 str.

15) Purba Sanljiv: Database Development Methodology and Organization. IS Management handbook. Auerbach, Boca Raton: 2000. 788 str.

16) Rupnik Rok et al.: Case orodje nove generacije pri prenovitvi in vzdrževanju informacijskih sistemov. Zbornik posvetovanja DSI 1998. Ljubljana: Slovensko društvo informatika, 1998, str. 27–36.

17) Svetek Staš: Razvoj informacijskega sistema za analizo kriminalističnih informacij. Uporabna informatika, Ljubljana, XI (2003), 1, str. 52–61.

18) Vintar Mirko: Informatika. Ljubljana: PACO, 1996. 186 str.

19) Vozel Marko, Vožič Bogdan: Standardizacija uporabniških rešitev. Zbornik posvetovanja DSI 1998. Ljubljana: Slovensko društvo informatika, 1998, str. 350–335.

20) Žabkar Andrej: Popolno generiranje programskih rešitev z orodjem CASE: primer Oracle Designer. Magistrsko delo. Ljubljana: Ekonomska fakulteta, 2000, 89 str., 2 pril.

Viri

1) Delphi 7 Studio Enterprise Essentials, Borland, Training Manual, Copywright: Borland Software Corporation, Version 7.0, release 3.0, 2002.

2) Heap Kate, Speelpenning Jan: Designer/2000 R2: Systems Modeling, Volume 1 Student Guide, Oracle, 1998.

3) Heričko Marjan: Informacijski sistemi, Univerza v Mariboru. [URL: http://lisa.uni-mb.si/student/predmeti/ois/doc/CASE.doc] , 22.10.2003.

4) Informacijski sistem Frapis, 2004.

5) Interna dokumentacija podjetja Infotrade, 2004.

6) Koletzke Peter, Dorsey Paul: Oracle Designer Handbook. Osborne, McGraw-Hill, 1999. 1071 str.

Slovarček

foreign key – tuj ključ
unique key – unikatni identifikator
primary key – primarni ključ
view – pogled
index – indeks
trigger – sprožilec
domain – uporabniško določeni tipi podatkov, lista vrednosti
form – zaslonska maska
main form – čelna maska
report – izpis
encapsulation – ograjevanje
inheritance – dedovanje
abstraction – abstrakcija
polymorphism – mnogoličnost
join - združevanje