UNIVERZA V LJUBLJANI

EKONOMSKA FAKULTETA

DIPLOMSKO DELO

RAZVOJ INFORMACIJSKEGA SISTEMA NA PODLAGI PROSTO DOSTOPNE PROGRAMSKE OPREME NA PRIMERU PODJETJA ACK, D.D.

Ljubljana, marec 2008 MARJAN FLIS

IZJAVA

Študent MARJAN FLIS izjavljam, da sem avtor tega diplomskega dela, ki sem ga napisal pod mentorstvom mag. ALEŠA POPOVIČA, in dovoljujem objavo diplomskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne 12.03.2008 Podpis:_____

KAZALO

1 UVOD 1
2 ACK, D.D. IN TRG POŠTNIH STORITEV 2
2.1 ZGODOVINA DRUŽBE 2
2.2 OSNOVNI PODATKI IN DEJAVNOST DRUŽBE 2
2.3 TRŽNI DELEŽ 3
3 OPREDELITEV OSNOVNIH POJMOV 3
3.1 INFORMACIJSKI SISTEM 3
3.1.1 Življenjski cikli razvoja in razvojni modeli 5
3.1.2 Celotni stroški lastništva (TCO) 7
3.2 BAZE PODATKOV 8
3.2.1 Sistemi za upravljanje z bazami podatkov (SUBP) 9
3.2.2 Podatkovno modeliranje 10
3.3 ODPRTA KODA 11
3.3.1 Odprtokodne licence in njihove omejitve 13
4 RAZVOJ PROGRAMSKE REŠITVE 15
4.1 VZROKI ZA PRENOVO 15
4.1.1 Pomanjkljivosti pri naročanju pošiljk 15
4.1.2 Pomanjkljivosti pri dostavi pošiljk 16
4.1.3 Pomanjkljivosti pri izdaji računov 16
4.1.4 Pomanjkljivosti, povezane z računovodstvom 17
4.2 ANALIZA POTREB 17
4.2.1 Izhodišča 17
4.2.2 Cilji razvoja novega informacijskega sistema 18
4.2.3 Analiza posameznih poslovnih procesov 19
4.3 PODATKOVNI MODEL 24
4.4 ARHITEKTURA SISTEMA 26
4.4.1 Prednosti in slabosti dodatnega nivoja 28
4.4.2 Izbrana arhitektura 29
4.5 IZBIRA RAZVOJNE PLATFORME 30
4.6 IZBIRA SISTEMA ZA UPRAVLJANJE Z BAZAMI PODATKOV 31
4.6.1 Stroški licenc in podpore 31
4.6.2 Zmogljivost 33
4.6.3 Nabor funkcij 34
4.6.4 Izbrani SUBP 36
5 UVEDBA NOVEGA INFORMACIJSKEGA SISTEMA 37
5.1 TESTIRANJE PROTOTIPA 37
5.2 SPREMEMBE IN DOPOLNITVE ZAHTEV 38
5.3 TEŽAVE PRI UVAJANJU 38
6 ANALIZA USPEŠNOSTI IN NADALJNJI RAZVOJ 39

6.1 ANALIZA USPEŠNOSTI, STROŠKOV IN KORISTI 40
6.2 NADALJNJI RAZVOJ IN POMANJKLJIVOSTI 41
7 SKLEP 43
LITERATURA 44
VIRI 45
PRILOGE 1

1 UVOD

Informacijski sistem se danes v marsikaterem večjem podjetju jemlje kot nekaj samoumevnega. Zakonitosti trga v konkurenčnem okolju narekujejo podjetjem, da neprestano sledijo dogajanju in da se na spremembe odzivajo kakor hitro je le mogoče. Kakšen in kako hiter bo ta odziv, je v veliki meri odvisno prav od lastnosti informacij, ki jih informacijski sistem zagotavlja. Dober sistem zagotavlja informacije, ki so pravočasne, točne in primerne. Učinkovitost oziroma kakovost sistema lahko pri tem od podjetja do podjetja precej variira.

Kljub ključni vlogi, ki jo informacijski sistem zavzema v podjetju, se lahko zgodi, da podjetje sistema nima izdelanega v želenem obsegu, oziroma le-ta potreb podjetja ne pokriva dovolj dobro. Stroški, ki so povezani z nepokritim delom, pri povečanju obsega poslovanja lahko skokovito narastejo in takrat je običajno čas za večje spremembe oziroma prilagoditve obstoječega sistema.

Namen diplomskega dela je predstaviti razvoj dela informacijskega sistema v podjetju, katerega glavna dejavnost je opravljanje poštnih storitev, in pri razvoju upoštevati prosto dostopno programsko opremo kot potencialno cenejšo alternativo plačljivi.

Cilj diplomskega dela je obravnavati implementirano programsko rešitev in vrednotiti ustreznosti posameznih izbir na poti do nje, poleg tega pa na podlagi pridobljenih izkušenj tudi predlagati smer nadaljnjega razvoja informacijskega sistema v obravnavanem podjetju.

Uporabljena metoda temelji na preučevanju strokovne literature in virov s področij informacijskih sistemov, baz podatkov ter prosto dostopne programske opreme. V delu sem razčlenil posamezne korake v razvoju uvedenega informacijskega sistema in na podlagi razpoložljivih informacij tudi analiziral alternativne poti do implementirane rešitve.

Diplomsko delo sem razdelil na sedem večjih sklopov. Uvodu v drugem poglavju sledi kratka predstavitev podjetja, v katerem je bil vpeljan nov informacijski sistem, in dejavnosti, s katero se podjetje ukvarja. V tretjem delu bom s pomočjo domače in tuje literature ter različnih virov predstavil nekatere metodologije ter osnovne pojme, povezane z razvojem informacijskih sistemov. V omenjenem poglavju se posvečam tudi z njimi nerazdružljivo povezanim bazam podatkov ter na kratko predstavim področje prosto dostopne programske opreme. V četrtem delu se bom podrobneje posvetil samemu postopku, katerega posledica je bil uveden informacijski sistem, predstavil vzroke za prenovo in cilje, ki jih podjetje z novim sistemom želi doseči. Vsebina petega poglavja bo uvedba informacijskega sistema in obravnava težav, povezanih z njo. V šestem poglavju sledi analiza uspešnosti izvedenega projekta, v katerem bom predstavil pomanjkljivosti implementirane rešitve in nakazal smeri nadaljnjega razvoja. V sedmem poglavju povzemam še nekaj sklepnih misli.

2 ACK, D.D. IN TRG POŠTNIH STORITEV

V naslednjih dveh poglavjih bom na kratko predstavil družbo ACK, d.d. in njeno zgodovino, v tretjem delu pa bom opisal tudi vlogo, ki jo podjetje zavzema na trgu poštnih storitev oziroma tržni delež podjetja glede na ostale udeležence.

2.1 ZGODOVINA DRUŽBE

ACK, d.d. je pravni naslednik Splošne trgovske delniške družbe, ki je bila vpisana v register družb v Kopru daljnega leta 1949. Po reorganizacijah v šestdesetih letih je leta 1964 začela poslovati kot trgovsko podjetje pod firmo Adriacommerce, d.d., in s to dejavnostjo, kateri se je vmes pridružilo tudi nekaj manjših industrijskih obratov, nadaljevala vse do leta 1991. V tem letu je bila družba zaradi velike odvisnosti od trgov bivše Jugoslavije prisiljena opustiti trgovsko dejavnost, več kot tristo zaposlenih pa je dobilo možnost v lastnih podjetjih nadaljevati nekatere dejavnosti družbe. Po 1991. letu je bila dejavnost družbe usmerjena predvsem v upravljanje s premoženjem. V letu 1996 je družba začela razvijati poslovanje na področju logistike in z logistiko povezanimi storitvami ter se v nadaljnjih letih tako preko organske rasti kot kapitalskih povezav na področju logistike uvrstila med najpomembnejše ponudnike logističnih storitev v Sloveniji (Spletna stran družbe ACK, d.d., 2007).

2.2 OSNOVNI PODATKI IN DEJAVNOST DRUŽBE

Družba je registrirana ter po standardni klasifikaciji razvrščena pod šifro 63.400 (dejavnost drugih prometnih agencij) in organizirana v devet organizacijskih enot. Njena glavna dejavnost je opravljanje logističnih storitev. Deluje torej na področju organizacije letalskega, cestnega in železniškega transporta, pretovora in skladiščenja blaga, carinskega posredovanja ter na področju malih pošiljk in paketne distribucije (Grošić, 2006, str. 25). Organizacijske enote, ki jih obravnavam v diplomskem delu, so vezane predvsem na glavno dejavnost, ki po oceni predsednika uprave predstavlja nekaj več kot polovico celotnega obsega poslovanja.

Družba formalno sodi med srednje velika podjetja, njen osnovni kapital je v letu 2006 znašal 426.961.000,00 SIT in je bil razdeljen na 426.961 delnic. Leto 2006 je družba zaključila s 311.945.000 SIT čistega dobička. ACK, d.d. je od leta 2002 dalje tudi imetnik certifikata kakovosti ISO 9001:2000 (Letno poročilo družbe ACK, d.d. za leto 2006).

2.3 TRŽNI DELEŽ

V letu 2006 je na trgu poštnih storitev v Sloveniji poleg Pošte Slovenije nastopalo še 11 drugih izvajalcev poštnih storitev s pridobljenim ugotovitvenim sklepom, med njimi tudi podjetje ACK, d.d. Obseg poštnih storitev Pošte Slovenije je bistveno drugačen od storitev ostalih izvajalcev poštnih storitev. Edina primerjava je mogoča pri storitvah »poslovni paket« in »hitra pošta« Pošte Slovenije in KEP storitvah prenosa paketov in dokumentov izvajalcev poštnih storitev z ugotovitvenim sklepom. KEP storitve vključujejo kurirske, ekspresne in paketne storitve. Poštna pošiljka mora biti pri tem zavarovana za primer izgube, izropanja, kraje ali poškodbe. Uporabnik poštnih storitev pa prejme dokazilo o poslani oziroma vročeni poštni pošiljki (Analiza trga poštnih storitev v Republiki Sloveniji v l. 2006).

Pri prenosu paketov z dodano vrednostjo v notranjem poštnem prometu ima Pošta Slovenije 78 % delež. Slednji se je v letu 2006 v primerjavi z letom 2005 zmanjšal za 2,5 %. Sledi podjetje TNT Express Worldwide z 12,3 % ter GLS s 6,7 %, delež podjetja ACK, d.d. pa znaša 0,84 %. Vsi udeleženci na trgu med drugim delujejo tudi pri prenosu paketov v mednarodnem poštnem prometu (v prispetju). Tu znaša delež Pošte Slovenije le 0,3 % in se je v letu 2006 glede na leto 2005 zmanjšal za več kot 1,5 %, najmočnejši položaj pa ima podjetje Intereuropa (Analiza trga poštnih storitev v Republiki Sloveniji v l. 2006).

Storitev Pošte Slovenije »hitra pošta« je mogoče primerjati s storitvijo KEP prenosa dokumentov v notranjem poštnem prometu. Na tem segmentu trga poštnih storitev ima Pošta Slovenije 55,8 % delež, dokaj visok delež imata tudi podjetji City Express (25,6 %) in Business Express (15,8 %). Precej podobna sta deleža podjetij OD VRAT DO VRAT (1,5 %) in ACK, d.d. (1,3 %) (Analiza trga poštnih storitev v Republiki Sloveniji v l. 2006).

3 OPREDELITEV OSNOVNIH POJMOV

V tej točki bom na kratko opisal nekatere ključne pojme, ki so bistveni za obravnavo v tem diplomskem delu. Najprej je to informacijski sistem, za njim pa še baze podatkov, ki so z informacijskim sistemom neločljivo povezane. Na koncu bom izpostavil pojem odprte kode in posledice, povezane z njeno uporabo.

3.1 INFORMACIJSKI SISTEM

Informacijski sistem lahko opredelimo kot množico medsebojno odvisnih komponent (strojna oprema, komunikacijska oprema, programska oprema, ljudje), ki zbirajo, procesirajo, hranijo in porazdeljujejo podatke (informacije) in s tem podpirajo tako temeljne kot tudi odločitvene procese v organizaciji (Bajec, 2004, str. 44). Informacijske sisteme je možno razdeliti na formalne in računalniško podprte ter na neformalne informacijske sisteme. Medtem ko ima

formalni informacijski sistem jasno določene podatke, s katerimi operira, določene postopke za njihovo obdelavo ter jasno definirana pravila, je neformalni informacijski sistem odvisen od implicitnih dogovorov in nedefiniranih pravil. V neformalnih informacijskih sistemih ni predpisov o tem, kaj so podatki in kaj mehanizmi za njihovo obdelavo. Pomembna lastnost informacijskega sistema je računalniška podpora, s čimer imamo v mislih sisteme, ki delujejo na osnovi računalniških/informacijskih tehnologij. Kljub temu, da lahko obstajajo informacijski sistemi, ki delujejo brez uporabe omenjenih tehnologij, je danes velika večina informacijskih sistemov vsaj delno računalniško podprta (Bajec, 2004, str. 44).

Pri zgornji definiciji izstopata dva pojma, ki sta nezdružljivo povezana z informacijskim sistemom:

- **Informacija**: je tako zaporedje znakov v danem znakovnem sistemu, ki je (Gradišar, Resinovič, 2001, str. 54):
    - sintaktično pravilno,
    - ima nedvoumno semantično vsebino, ki je zadostna slika pojava, na katerega se nanaša,
    - ima za upravljavca pragmatično vrednost.

  Informacija je znanje, ki se nanaša na objekte kot so dejstva, dogodki, stvari, procesi ali ideje, vključno s koncepti, ki imajo v okviru nekega konteksta določen pomen (Bajec, 2004, str. 17).
- **Podatek**: je predstavitev informacije na formaliziran način, ki je primeren za komunikacijo, interpretacijo ali obdelavo (s strani človeka ali stroja). Predstavimo ga lahko s pomočjo simbolov ali analognih veličin, ki jim je pripisan nek pomen oziroma jim ga je mogoče pripisati. Je zapis, opis ali predstavitev nekega dogodka, pojava ali dejstva iz realnega sveta v numerični, besedni ali grafični obliki (Bajec, 2004, str. 17).

Ker je informacija pojem, odvisen od podatka, lahko njun odnos zapišemo tudi z Borje Langefors-ovo informacijsko enačbo: **I = i(D, S, t)**. Pri tem je I informacija, ki jo posredujejo podatki, i informacijska funkcija, D podatki, S prejemnikovo znanje in t čas, ki je na voljo prejemniku za interpretacijo podatkov.

Iz te enačbe sledi (Bajec, 2004, str. 17):

- podatki niso informacija;
- podatki ne vsebujejo informacije;
- podatki posredujejo informacijo prejemniku, katerega znanje je konsistentno z izbrano predstavitvijo podatkov in modelom sveta, na katerega se nanašajo;
- če je količina podatkov tako velika, da se jih v času, ki je na voljo za ukrepanje na njihovi osnovi, ne da interpretirati, se lahko zgodi, da s podatki ni posredovana nobena informacija.

Lastnosti dobrega informacijskega sistema so zagotavljanje podatkov, iz katerih lahko zaposleni na različnih ravneh v organizaciji pridobivajo informacije, ki jih potrebujejo pri

svojem delu. Informacijski sistem naj bi tudi dajal podlago tako za reševanje vsakodnevnih vprašanj kot tudi za izvajanje upravljavskih ukrepov ter sprejemanje strateških odločitev. Predpogoj za vse našteto pa je seveda usklajenost s poslovnim sistemom, katerega del je (Bajec, 2004, str. 45).

3.1.1 Življenjski cikli razvoja in razvojni modeli

Kot večina razvojnih procesov tudi razvoj informacijskega sistema sledi določenemu življenjskemu ciklu oziroma razvojnemu modelu, ki določa zaporedje faz razvoja. Razvojni modeli informacijskega sistema zajemajo analizo, načrtovanje, izvedbo ter vpeljavo. Med seboj se razlikujejo predvsem po podrobnejši delitvi faz na aktivnosti ter v zaporedju in načinu njihovega izvajanja.

3.1.1.1 Zaporedni ali slapovni model

Zaporedni ali slapovni model (ang. "waterfall model") je najstarejši razvojni model, značilen za prve oblike strukturnega pristopa. Značilnost tega modela je, da si faze sledijo zaporedno. Ko smo z neko fazo zaključili, vračanje vanjo ni več mogoče. Slabosti zaporednega modela so (Bajec, 2004, str. 45):

- zahteve nikoli niso statične in se spreminjajo;
- posamezne faze ne moremo preprosto zaključiti, potrebno je vračanje nazaj;
- brez vračanja nazaj razvit sistem lahko ob koncu ne ustreza dejanskim zahtevam;
- tveganje, da sistem ne ustreza zahtevam, je visoko vse do zadnje faze razvoja.

3.1.1.2 Iterativni model

Iterativni model je bil razvit kot odziv na pomanjkljivosti slapovnega pristopa. Pri iterativnem pristopu izvajamo korake slapovnega pristopa v več iteracijah. V vsaki iteraciji razvijemo določen del funkcionalnosti celotnega sistema. V začetnih iteracijah razvijemo najbolj tvegane dele sistema. Bistvena značilnost je, da se sistem razvija inkrementalno. Najbolj tvegane so začetne iteracije, zato najprej razvijemo najbolj tvegan del sistema. Rezultat vsake iteracije je izvršljiv dodaten del celotnega sistema. Vsaka iteracija vključuje povezovanje v celoten sistem in preizkušanje. Prednosti iterativnega razvoja pred zaporednim so (Yeates, Wakefield, 2004, str. 134-142):

- najbolj tvegani deli so razrešeni še preden postane investicija velika;
- začetne iteracije omogočijo zgodnje povratne informacije s strani uporabnikov;
- preizkušanje in povezovanje v sistem sta nepretrgana;
- ciljni mejniki omogočajo kratkoročno osredotočenje;
- napredek merimo z ocenjevanjem izvedenega dela;

- možna je predaja izvedenega dela projekta še preden je dokončan celoten projekt.

3.1.1.3 Prototipni razvoj

Prototipni razvoj se je pojavil skupaj z iterativnim modelom. Danes se uporablja pri večini razvojnih modelov, obstaja tudi poseben prototipni model. Prototipni model temelji na izdelavi prototipov in se uporablja v različnih fazah razvoja. Prototip označuje predhodno izdelane in navadno nepopolne verzije sistema. Za izdelavo prototipov so bila razvita posebna razvojna okolja, ki omogočajo vizualno sestavljanje zaslonskih mask, izpisov in poizvedb ter vsebujejo mehanizme za avtomatsko generiranje kode. Izdelava prototipov kot tehnika razvoja aplikacij temelji na tesnem sodelovanju med uporabnikom in analitikom. Ko se uskladita glede zahtev sistema, analitik razvije prototip in ga da uporabniku v testiranje oziroma v uporabo. Ta analitiku sporoči, kaj mu je pri prototipu všeč in kaj ne. Povratna informacija služi analitiku za pripravo nove verzije prototipa. Proces traja, dokler uporabnik s prototipom ni zadovoljen. Prototipi se lahko uporabljajo kot del specifikacije sistema, za pridobitev jasnejše podobe bodočega sistema in se v nadaljevanju zavržejo, ali pa kot osnova za izdelavo produkcijskega sistema (Kovačič, 1998, str. 160).

3.1.1.4 Inkrementalni model

Inkrementalni ali postopni model je zelo popularen model razvoja informacijskega sistema, pri katerem celoten problem razdelimo na podprobleme ali module, ki jih rešujemo posebej in neodvisno od ostalih. Aplikacijo tako razvijamo po delih, ki zajemajo le določen del funkcionalnosti želenega sistema, vendar so obenem dovolj samostojni, da jih lahko vključimo v produkcijo. Z moduli sčasoma pokrijemo celotno funkcionalnost in zaključimo razvoj. Razvoj posameznih modulov jemljemo kot samostojne podprojekte. Delitev projekta na podprojekte je velikokrat izvedena na osnovi določitve pomembnosti posameznih funkcionalnih zahtev, njihove uvrstitve v module ter določitve vrstnega reda izvedbe posameznih modulov. Prav tako popularen pristop je delitev, kjer en modul ustreza enemu funkcionalnemu področju (začnemo s strateškim planiranjem, nadaljujemo pa z naslednjimi fazami za vsak modul posebej). Posamezne module na koncu integriramo v celotno rešitev. Inkrementalni model v sebi skriva kar nekaj dobrih lastnosti. Ena pomembnejših je ta, da dele končnega produkta že relativno zgodaj predamo v uporabo, kar poveča možnost za odkrivanje in odpravljanje morebitnih napak in pomanjkljivosti. Velja tudi, da je razvoj informacijskega sistema po inkrementalnem modelu v splošnem cenejši in manj tvegan kot razvoj aplikacije v enem kosu. Kljub mnogim prednostim pa inkrementalni model ni primeren za vse vrste projektov. Ne moremo na primer pričakovati, da bi sistem za kontrolo zračnega prometa lahko v praksi deloval, dokler ne bi bil v celoti dokončan (Yeates, Wakefield, 2004, str. 134-142).

3.1.1.5 Kombinirani model

Kombinirani model je zasnovan na osnovi zaporednega modela, pri čemer omogoča vračanje v predhodne faze ter vključuje dosledno rabo prototipov. S tem ohranja dobre lastnosti zaporednega modela (razvijalcem nudi hrbtenico, ki je pri večjih projektih neobhodna) ter odpravlja njegove bistvene pomanjkljivosti, upoštevaje glavne prednosti pristopa razvoja po iterativnem modelu. Kombinirani model je mogoče uporabljati tudi v kombinaciji z inkrementalnim razvojem, kjer posamezne module razvijamo po kombiniranem modelu. V praksi se zelo pogosto uporablja, njegove glavne odlike izhajajo iz dejstva, da je zelo blizu resničnemu procesu razvoja, saj dovoljuje poljubna prehajanja med fazami, hkrati pa definira osnovno zaporedje, v katerem naj se aktivnosti izvajajo, in s tem preprečuje popolnoma nedisciplinirano delo. Dodatna dobra lastnost kombiniranega modela je uporaba prototipov v vseh fazah razvoja, kar izboljšuje komunikacijo med razvijalci in uporabniki, nosilci poslovnega znanja (Bajec, 2004, str. 90-104).

3.1.2 Celotni stroški lastništva (TCO)

Celotni stroški lastništva informacijskega sistema v organizaciji so vsi stroški, ki se nanašajo na nakup in vzdrževanje sistema. So pomemben pokazatelj, ki služi zagotavljanju usklajenosti strateškega načrta informatike s finančnimi odločitvami, in široko uporabljen parameter pri merjenju stroškovne učinkovitosti programske in strojne opreme oziroma poljubnih kombinacij teh dveh znotraj organizacije (Coopworks - Total cost of ownership, 2008).

V obdobju e-poslovanja, osebnih računalnikov in sistemov tipa odjemalec/strežnik je količina dela, ki je potrebno za vzdrževanje informacijskega sistema, stroškovno gledano postala prevladujoča komponenta celote. Pred tem so v času velikih strežnikov (»mainframe«) stroški programske in strojne opreme predstavljali največji del stroškov. Pri izračunu celotnih stroškov lastništva se poskuša zajeti vse realne stroške vpeljave informacijske rešitve v vseh fazah. Pomembno pri tem pa je, da celotni stroški lastništva ne poskušajo izmeriti učinka izboljšav, ki jih prinaša dana informacijska tehnologija. Za merjenje učinka so običajno v rabi drugi kazalniki, med drugim tudi donos na investicije - ROI (Coopworks - Total cost of ownership, 2008).

V izračun celotnih stroškov lastništva zajeti stroški so (Coopworks - Total cost of ownership, 2008):

- Stroški pridobitve oziroma nakupa sistema, ki vključujejo naslednje stroške:
    - analize za nakup primernih rešitev;
    - zasnove sistema in potrebnih pripadajočih komponent, z namenom zagotavljanja kompatibilnosti;
    - zbiranja ponudb in iskanja najboljšega možnega dobavitelja za izbrano rešitev;
    - nakup izbrane rešitve – z dobaviteljem dogovorjena cena;

- namestitve sistema;
- stroški prilagoditve aplikacije za samo uporabo - stroški prilagoditev lahko povzročijo, da celotni stroški lastništva precej narastejo, a so obenem prilagoditve običajno podcenjene;
- izobraževanja uporabnikov;
- vpeljave sistema v uporabo, vključno s prehodom starega sistema in poslovnih procesov na nov sistem.

- Stroški vzdrževanja razpoložljivosti za končne uporabnike, ki pokrivajo naslednja področja:
  - upravljanje sistema, vključno z vsemi vidiki zagotavljanja normalnega delovanja, kot na primer aktivacija in izklop sistema, kontrola nalog, upravljanje z izložki, varnostne kopije, obnovitve;
  - vzdrževanje strojne opreme in programskih komponent, vključno s preventivo in odpravljanjem napak;
  - podpora uporabnikom, vključno s kontinuiranim izobraževanjem podpornega osebja, in reševanje problemov;
  - faktorji okolja, vključno s klimatizacijo, napajanjem, hrambo;
  - ostali dejavniki, ki ne spadajo v nobeno od zgornjih kategorij, so pa odvisni od specifik sistema in prevladujočih okoliščin.

Čeprav so zgoraj naštete kategorije precej lahko razumljive, je izmera oziroma ocena vsake izmed njih danes težavna ali celo nepraktična, saj imajo le redke organizacije dovolj razvito računovodstvo, ki bi omogočalo drobitev stroškov do zadostnih podrobnosti. Dodatni problem predstavlja netočno vodenje inventarja oziroma pomanjkljive informacije, povezane z računalniškim sistemom. To je še posebej izrazito v velikih organizacijah, kjer odločitve, povezane z nakupom opreme, niso sprejete le na enem mestu. Točen znesek celotnih stroškov lastništva je sicer le redko pomemben, potrebno pa se je zavedati, kako visoki bodo ti stroški, in jih v procesu planiranja upoštevati, četudi jih je možno le grobo oceniti (Coopworks - Total cost of ownership, 2008).

Celotni stroški lastništva nudijo tudi dobro osnovo za primerjavo različnih načinov vpeljave sistema med platformami in med konkurenčnimi proizvodi. Kljub temu pa ti ne bi smeli biti edini faktor pri sprejemanju odločitev, saj je pri tem vedno zelo pomembno ravnotežje med stroški in koristmi, ki jih dani sistem ali tehnologija prinese v podjetje (Coopworks - Total cost of ownership, 2008).

3.2 BAZE PODATKOV

Baze podatkov so z informacijskim sistemom neločljivo povezane. Običajno se vsak proces pod okriljem nekega informacijskega sistema začne in konča pri bazi podatkov. Od kakovosti

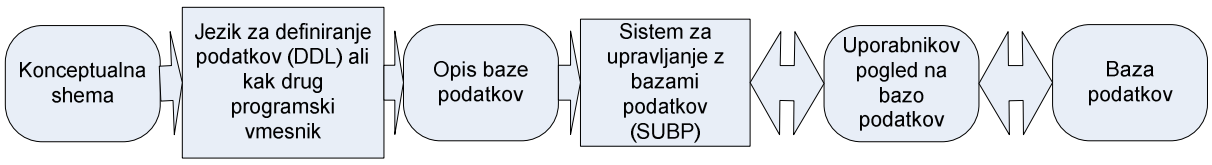
podatkov, shranjenih v bazi podatkov, je neposredno odvisna tudi kvaliteta izhodnih podatkov, ki se na te nanašajo, posledično pa tudi kakovost informacijskega sistema kot celote.

Baza podatkov je urejena zbirka medsebojno povezanih podatkov, ki je shranjena na računalniškem nosilcu. Baza podatkov je abstrakcija nekega dela realnega sveta, kar pomeni, da se morajo v njej odražati tudi spremembe v realnem svetu. Baza naj bi bila integrirana in naj bi vsebovala podatke za mnoge uporabnike, pri čemer vsakega od njih zanima le manjši del celote. Prekrivajoče potrebe uporabnikov, ki uporabljajo iste podatke, obenem pomenijo zmanjšanje podvajanja podatkov na najmanjšo možno mero. Centralizirana hramba omogoča sistematičen pregled nad podatki, kar eliminira podvajanje dela pri skrbi za njihovo ažurnost. Značilnost baze podatkov je tudi, da je neodvisna od programov, ki jo bodo uporabljali. Ker pri razvoju informacijskega sistema vseh načinov uporabe baze podatkov običajno ne moremo predvideti, ta lastnost omogoča večjo fleksibilnost. Pri tem lahko del baze, ki strukturno ni vezan na aplikacijo, spreminjamo brez da bi vplivali na njeno funkcionalnost. To nam omogoča lažje prilagajanje spremembam, hkrati pa tudi izvajanje »ad hoc« opravil oziroma poizvedb nad obstoječimi podatki, ki jih razvoj aplikacije, ki se povezuje z bazo podatkov, ni predvidel (Jaklič, 2002, str. 70-72).

3.2.1 Sistemi za upravljanje z bazami podatkov (SUBP)

Sistem za upravljanje z bazami podatkov je zbirka programov, ki omogočajo tvorbo, uporabo in vzdrževanje podatkovnih baz. Z njimi izvajamo vse aktivnosti, povezane s podatkovno bazo, razen načrtovanja, čemur so namenjena druga programska orodja. SUBP so namenjeni raznolikim uporabnikom, od načrtovalca podatkovne baze, programerja, skrbnika do uporabnika (Jaklič, 2002, str. 75). SUBP je sistem, ki omogoča prevedbo podatkov iz njihove logične predstavitve v fizično in nazaj. Omenjeni proces prikazuje spodnja slika.

Slika 1: Struktura DBMS



Vir: Sumathi, Esakkirajan, 2007, str. 4.

Glavni cilji SUBP so (Sumathi, Esakkirajan, 2007, str. 3-4):

- **Razpoložljivost podatkov:** nanaša se na dejstvo, da so podatki na voljo širokemu krogu uporabnikov v smiselnem, uporabljivem formatu za sprejemljivo ceno (z vidika potrebnega truda, časa...). Uporabnik naj bi do podatkov torej enostavno dostopal.
- **Integriteta podatkov:** nanaša se na pravilnost zapisa podatkov v bazi podatkov. Z drugimi besedami, podatki, ki so na voljo v bazi podatkov, so zanesljivi.

- **Varnost podatkov:** nanaša se na dejstvo, da do podatkov smejo dostopati le pooblaščeni uporabniki. Dostop do podatkov je tako običajno varovan z geslom. Varnost podatkov pa zadeva tudi konfliktne spremembe v bazi podatkov. Slednje mora namreč sistem preprečevati.
- **Podatkovna neodvisnost:** SUBP dovoljuje uporabniku shranjevanje, spreminjanje in priklic podatkov na učinkovit način. Pri tem SUBP zagotavlja abstrakten pogled na to, kako so v bazi podatkov le-ti dejansko shranjeni. Z namenom omogočanja učinkovitega priklica podatkov so za njihovo predstavitev uporabljene kompleksne strukture. Sistem pri tem skrije določene podrobnosti v zvezi s tem, kako se podatki shranjujejo in vzdržujejo.

3.2.2 Podatkovno modeliranje

Podatkovno modeliranje je tehnika, ki beleži zalogo, obliko, velikost, vsebino in pravila uporabljenih podatkovnih elementov v okviru poslovnega procesa. Rezultat podatkovnega modeliranja pa je nekakšen zemljevid (model), ki opisuje v procesu uporabljene podatke (Siau, 2007, str. 64). Podatkovno modeliranje je tesno povezano z modeliranjem poslovnih procesov. Slednje omogoča enotno razumevanje in analizo poslovnih procesov, ki je osnova za temeljito razumevanje posameznega procesa. Skozi poslovne procese je možno analizirati in povezati organizacijo (Popovič et al., 2004, str. 80).

Model entitet-povezav (ER model) je bil sprva predstavljen kot sredstvo, s katerim je mogoče z malo napora pridobiti dober vpogled v strukturo baze podatkov. Omogoča nam, da na bolj formalen način predstavimo ali opišemo podatke, s katerimi podjetje operira (Davis, 1999, str. 199). Opis vključuje splet objektov oziroma logičnih enot v podjetju in pripadajočih povezav. Običajno se ta model uporablja v začetnih fazah snovanja baze podatkov (Ramakrishnan, Gehrke, 1998, str. 24-35).

Pri modelu ER se srečujemo z velikim številom pojmov, med njimi pa bi izpostavil dva:

- **Entiteta**: je logična enota oziroma objekt v realnosti, ki jo je moč ločiti od ostalih objektov (Celko, 1998, str. 14). V podjetju ACK, d.d. so ključne enitete pošiljka, stranka, paket ipd.
- **Atribut**: je pojem, s katerim opisujemo entiteto. Vsi objekti, ki ustrezajo eni entiteti, imajo enake atribute. Tako imata na primer Janko Študent in Micka Novak, ki sta pojava entitete stranka, v realnosti oba atribute kot so ime, priimek, naslov ipd. Izbor naših atributov je odsev stopnje podrobnosti, s katero želimo predstaviti podatke o entitetah. V danem primeru nas na primer zanima strankin spol, v drugem pa nam povsem zadostuje le naslov (Hočevar, 2005, str. 30).

3.3 ODPRTA KODA

Odprta koda je pojem, ki nas v zadnjih letih spremlja vse pogosteje. Podjetja so nanjo postala pozorna predvsem zaradi prihrankov pri stroških programske opreme, ki jih omogoča. Odprta koda je bila na začetku zgolj miselnost ali neke vrste gibanje, katerega pionir je bil Richard Matthew Stallman, danes pa se vse pogosteje kaže kot močna sila na trgu, ki je ne more spregledati niti veliko podjetje kot je Microsoft. Pojem odprte kode se marsikdaj povezuje z novo obliko komunizma in posledično z vsemi problemi, s katerimi je bil slednji obremenjen. Tovrstna povezovanja lahko zasledimo predvsem s strani podjetij, ki se na velike spremembe na trgu programske opreme še niso prilagodila. Nasprotno pa se prilagojena podjetja zavedajo omejitev večine zaprtokodnih licenc in nevarnosti, ki jih s seboj prinaša zaprtokodna programska oprema, kot so na primer (Walnes et al., 2003, str. 3):

- programski patenti;
- nezdružljivost (datotek, omrežnih protokolov ipd.);
- netransparentno delovanje programov (koncept črne škatle);
- majhna možnost vplivanja na delovanje naše programske opreme izven vnaprej predvidenih okvirjev (z izjemo povratnega inženiringa, ki je v večini zaprtokodnih licenc eksplicitno prepovedan);
- vprašanje zaupanja avtorju programske opreme;
- tveganje premočne navezanosti na enega ponudnika in posledično slab pogajalski položaj kupca programske opreme;
- vprašanje podpore v primeru težav ponudnika (npr. če to podjetje propade).

Četudi je odprtokodna alternativa za dan problem na voljo, se pogosto zgodi, da moramo kljub temu poseči po zaportokodni rešitvi, saj je velikokrat preprosto boljša, ali pa na primer edina pokriva naše potrebe. To še posebej velja za nove odprtokodne projekte, ki še niso na dovolj visoki stopnji razvoja, da bi lahko konkurirali že več let prisotnim komercialnim rešitvam. Je pa ne glede na dostopnost alternativ pomembno, da se zgoraj naštetih nevarnosti zavedamo. Odprtokodna programska oprema v naše roke polaga precejšnje število pravic, ki jih pri zaprtokodnih rešitvah nismo vajeni, to pa lahko za nas pomeni tudi nove, prej lahko celo nepredstavljive poslovne priložnosti.

Obstaja več definicij odprte kode, najbolj široko sprejeta je tista, ki jo podaja Iniciativa za odprto kodo - OSI (Open Source Initiative, 2007). Gre za organizacijo, ustanovljeno februarja 1998, po objavi izvorne kode spletnega brskalnika podjetja Netscape Communications. Spletni brskalnik je bil najpomembnejši med njihovimi proizvodi, razlog za to odločitev pa je bilo padanje profitnih stopenj in močna konkurenca s strani Microsoftovega Internet Explorer-ja.

Pogoji, ki jim mora programska oprema zadostiti, da se lahko uvršča pod odprtokodno, so po OSI naslednji:

- **Prosta redistribucija:** Licenca ne sme nikomur prepovedovati prodaje in brezplačne distribucije programske opreme, ki med drugim vključuje s to licenco krito programsko opremo kot enega izmed delov. Licenca za to ne sme zahtevati nikakršnih plačil.
- **Izvorna koda:** Program mora vključevati izvorno kodo in dovoljevati distribucijo tako v obliki izvorne kode kot v prevedeni obliki. Kjer neka oblika proizvoda ni distribuirana skupaj z izvorno kodo, mora biti priloženo izpostavljeno obvestilo, ki omogoča dostop do izvorne kode za razumno ceno, ki ni višja od stroškov reprodukcije. Preferiran način je možnost prenosa preko interneta brez dodatnih stroškov. Sprememba izvorne kode mora biti preferenčna oblika spremembe programa. Nedovoljeno je zakrivanje ali drugačno oteževanje njene uporabe. Vmesne oblike, kot je izhod prevajalnika ali predprocesoja, niso dovoljene.
- **Izvedena dela:** Licenca mora dovoljevati spreminjanje in izdelavo izvedenih programskih rešitev, prav tako mora dovoljevati distribucijo pod enakimi licenčnimi pogoji kot to velja za izvorno programsko opremo.
- **Integriteta avtorja izvorne kode:** Prepoved distribucije izvorne kode v spremenjeni obliki je sprejemljiva le v primeru, ko licenca dovoljuje distribucijo paketov, ki omogočajo spremembe v času izgradnje programske rešitve. Eksplicitno mora biti dovoljena distribucija programske opreme, zgrajene na temelju modificirane izvorne kode. Licenca lahko zahteva, da morajo biti izpeljani izdelki distribuirani pod spremenjenim nazivom, ali pa spremenjeno oznako verzije.
- **Prepoved diskriminacije posameznikov in skupin:** Licenca ne sme diskriminirati nobenega posameznika ali skupine posameznikov.
- **Prepoved diskriminacije posameznih področij dejavnosti:** Licenca ne sme nikogar omejevati pri uporabi programa na določenem področju dejavnosti. Ne sme na primer omejevati uporabe programa v komercialne namene ali za namene genetskih raziskav.
- **Distribucija licence:** Licenčne pravice se morajo prenesti na vsakogar, ki prejme programsko rešitev, brez dodatnih postopkov licenciranja, izvedenih s strani prejemnika.
- **Licenca ne sme biti specifična za proizvod:** Pravice, vezane na program, ne smejo biti odvisne od tega, ali je program del določene programske distribucije. Če je program odstranjen iz distribucije in uporabljen ali distribuiran naprej pod pogoji licence programa, morajo vse stranke, katerim je bil program distribuiran, imeti enake pravice, kot tiste, ki so jim bile dodeljene pravice za originalno programsko distribucijo.
- **Licenca ne sme omejevati druge programske opreme:** Licenca ne sme postavljati omejitev nad programsko opremo, ki je distribuirana skupaj z licencirano programsko opremo. Licenca na primer ne sme zahtevati, da mora biti vsa programska oprema, ki je distribuirana na istem mediju, prav tako odprtokodna programska oprema.

- **Licenca mora biti nevtralna do tehnologije:** Prevzemanje licenc ne sme biti omejeno na posamezne tehnološke rešitve ali vmesnike.

Programska oprema je lahko prosto dostopna, a ne zadosti vsem zgoraj naštetim kriterij. V nekaterih primerih so lahko omejitve pri tem precej podobne običajni, zaprtokodni programski opremi. Razlika je lahko v ceni licence, ki nam jo pri tem ni potrebno plačati. Omeniti velja, da v tem primeru programska oprema nosi celotno tveganje zaprtokodne programske opreme.

3.3.1 Odprtokodne licence in njihove omejitve

Celoten spisek licenc, ki zadoščajo zahtevam »odprte kode«, OSI vodi tudi na svoji spletni strani. Licence, ki so v svetu odprtokodne programske opreme pogosto v uporabi, pa so naslednje:

- GNU General Public Licence (GPL),
- BSD Licence,
- MIT Licence,
- GNU Library ali »Lesser« Public Licence (LGPL),
- Mozilla Public Licence (MPL).

Naštete licence je možno glede na raven pravic, ki jih puščajo uporabniku, kljub ustrezanju precejšnjemu številu pogojev še vedno razdeliti v več kategorij. Omejitve se nanašajo predvsem na to, ali mora nekdo pri distribuciji izvedenega programa dati na razpolago tudi izvorno kodo za avtorju izvedenega dela lasten del celote (restriktivna licenca). Tako je za program, ki temelji na osnovi nekega drugega, ki je izdan pod izbrano restriktivno licenco, relativno težko iztržiti visoko ceno. Ob pogoju enostavne dostopnosti izvorne kode se namreč večina tovrstnega poslovanja spremeni v vzdrževalne pogodbe, saj ob predpostavki dovolj visoke informiranosti ni veliko razlogov za nakup nečesa, kar je možno dobiti zastonj. V povezavi s tem se pogosto uporablja pojem »virusni učinek licence«, saj na primer še tako majhen del uporabljene kode pod restriktivno licenco pomeni veliko obvezo za celoto. Poznavanje omejitev, ki jih posamezna licenca nalaga uporabniku odprtokodne programske rešitve, je torej pri razvoju lastne programske opreme zelo pomembno ("We Don't Need the GPL Anymore", 2007).

Najbolj restriktivna med naštetimi licencami je tako GPL, sledita ji LGPL in MPL, najmanj restriktivni pa sta MIT in BSD licenci, ki uporabniku kode puščata skoraj povsem proste roke. Pri GPL licenci in podobnih alternativah velja omeniti, da posledice spoštovanja licence povprečni uporabnik čuti le v primeru, ko izdelek, temelječ na programu pod GPL licenco, poskuša distribuirati naprej. Razvite programske rešitve, ki temelji na GPL licencirani programski opremi, tako ni potrebno dati na vpogled javnosti, dokler je ne prodamo naši

stranki ali kakemu drugemu partnerju. Skrb, da bi pri notranjem razvoju tvegali izgubo poslovnih skrivnosti iz tega naslova, je tako odveč.

Restriktivna licenca kljub skladnosti z vsemi OSI pogoji ponuja tudi poslovno priložnost za podjetje, ki je na začetku razvojne verige programske opreme. Podjetje namreč lahko izda program pod neomejenim številom licenc. Običajno je kombinacija restriktivne licence in običajne komercialne licence tisto, kar omogoča podjetju, da dobi »najboljše iz obeh svetov«. Na eni strani je podjetje deležno koristi s strani skupnosti razvijalcev programske opreme, ki so tudi sami uporabniki programske rešitve. Izdana programska rešitev namreč lahko pritegne zanimanje razvijalcev in ti preko vložka lastnega prostega časa v razvoj proizvoda težijo k izboljšanju njegove kvalitete. Na drugi strani pa lahko podjetje še vedno beleži prihodke iz naslova prodaje te programske rešitve tistim podjetjem, pri katerih bo slednja le delček v mozaiku večjega projekta. Tega načina licenciranja se med drugim poslužuje tudi MySQL AB, katerega sistem za uporabljanje z bazami podatkov bom v nadaljevanju obravnaval kot poglavitni del pri razvoju informacijskega sistema ACK, d.d. (Spletna stran MySQL AB, 2007).

V odkprtokodni skupnosti se pogosto pojavlja vprašanje, katero licenco pri odprtokodnem programu uporabiti. Argumentov na strani bolj restriktivnih licenc je precej, enako pa je možno trditi tudi za nasprotno stran. Ravnanje velikih korporacij, kot je Microsoft, so dober zgled slabe prakse oziroma zlorabe dobronamerne poteze skupine razvijalcev. Protokol Kerberos, ki je danes eden od najbolj pogosto uporabljenih pri preverjanju verodostojnosti uporabnikov v računalniških omrežjih znotraj podjetij, je na primer Microsoft nekoliko modificiral, implementacijo sprememb v konkurenčnih proizvodih pa je na začetku življenjske poti njihovega novega proizvoda prepovedal. Ker mu licenca ni narekovala, da spremembe osnove razkrije pod enakimi pogoji, kot so bili tisti, pod katerimi je osnovo dobil, tega ni storil. Namesto tega je dokumentacijo ob izidu označil za poslovno skrivnost (Linux Gazette, 2002).

Po drugi plati pa restriktivno licenco lahko razumemo tudi kot neke vrste ukrep pri zaščiti programske opreme na začetku svoje poti. Kot je na primer za industrijo v povojih včasih opravičljiva carinska zaščita pred tujo konkurenco, je tudi pri programski opremi veliko tveganje, da bo razvijajoči projekt pristal v vodah zaprtokodne programske opreme. Ker je na začetku svoje življenjske poti programska oprema običajno še precej nedodelana, zahteva precejšen vložek ur dela programerjev, ti pa v primeru, da je njihov proizvod v malo drugačni obliki nekje drugje komercialno dostopen, lahko izgubijo začetno motiviranost in zavezanost projektu. To pa seveda ni v interesu podjetja, ki odprtokodno programsko opremo uporablja, saj to pomeni konec nadaljnjega razvoja in podpore, lahko pa pripelje tudi do precejšnjih stroškov iskanja alternativ v primeru močne integracije v našem podjetju ("We Don't Need the GPL Anymore", 2007).

4 RAZVOJ PROGRAMSKE REŠITVE

V naslednjih poglavjih bom predstavil proces razvoja programske rešitve v obravnavanem podjetju. V prvih treh podpoglavjih se bom posvetil obravnavi vzrokov, ki so pripeljali do razvoja nove programske rešitve, potrebam, ki jih bo slednja morala pokrivati, in podatkovnemu modelu, ki je logična posledica prvih dveh korakov. V zadnjih treh podpoglavjih bom obravnaval nekatere odločitve, povezane s tehničnim delom implementacije.

4.1 VZROKI ZA PRENOVO

Obstoječe poslovanje obravnavanega podjetja pred uvedbo novega sistema pokriva večje število bolj ali manj neodvisnih aplikacij in orodja, ki spadajo v domeno pisarniških programskih paketov. Uporaba rezultatov enega dela v tem mozaiku načeloma ni neposredno uporabljiva v drugem delu sistema. Tako na primer s podatki o pošiljkah, vnesenimi v sistem poslovnega partnerja, ni možno enostavno in neposredno operirati, ko potrebe narekujejo izdelavo kakega poročila ali analize, sploh če slednje služi kot podlaga za obračun plačil. Nič neobičajnega ni niti pojav napak, katerih teža se zaradi soodvisnosti enot, v primeru, da nastanejo na začetku, po verigi lahko le še stopnjuje. Zaznava napak je zaradi omejene možnosti kontrole težka, šibkost varoval pa je relativno lahko izkoristiti tudi za pridobitev osebne koristi. Omenjene pomanjkljivosti vodijo v podvajanje dela in ustvarjajo precejšnja ozka grla, ki otežujejo širitev obsega poslovanja podjetja in neposredno vplivajo tudi na zadovoljstvo strank.

V nadaljevanju bom poskusil splet različnih aplikacij in postopkov, potrebnih dodelave oziroma prenove, razdeliti v večje kategorije. Področji, ki ga te pokrivajo, sta predvsem logistika in izdaja računov, kategorije iz naslednjih 4 podpoglavij pa približno ustrezajo tudi pripadajočim poslovnim procesom. Okvirno bom razložil možnosti, ki jih ponuja vsaka izmed njih, izpostavil pa bom tudi njihovo povezljivost z ostalimi deli celote. Pri tem bom upošteval le dele, ki jih bo pokrivala nova programska rešitev.

4.1.1 Pomanjkljivosti pri naročanju pošiljk

Naročanje pošiljk s strani strank je v celoti neinformatizirano (pri procesu je potrebno precej ročnega dela s strani referenta). Posledice tega se najbolj občutijo pri majhnih pošiljkah, saj so pri teh zato relativni manipulativni stroški, povezani z naročanjem, največji. Manj problematičen je periodičen prevzem večjih količin po predhodnem dogovoru pa tudi enkratno naročilo pošiljke z večjo težo ali prostornino.

V glavnini primerov naročilo prevzema pošiljke poteka preko telefona. Stranka pri tem referentu zaupa osnovne podatke, ki se nanašajo na pošiljko, od te točke dalje pa je osnovna dokumentacija, povezana s pošiljko, špeditersko potrdilo. Problem tega potrdila je v težavnem sledenju pošiljki in veliki količini ročnega dela ter s tem povezanem številu mest, kjer zaradi človeškega faktorja lahko pride do napake.

Očitno ozko grlo pri tem delu je posledica izpolnjevanja špediterskih potrdil in pretipkavanja le-teh v program za izdajo računov. Ta korak je potreben zaradi podatkov o opravljenih storitvah na računu, ki med drugim služijo za lastno evidenco in so v kombinaciji s še nekaterimi drugimi faktorji (teža, prostornina…) tudi osnova za izračun cene. Odsotnost teh podatkov bi bila tudi v nasprotju z željami večine strank.

4.1.2 Pomanjkljivosti pri dostavi pošiljk

Po prevzemu pošiljke na sedežu stranke ali v poslovalnici ACK, d.d. se večina podatkov, povezanih s pošiljko, nahaja na špediterskih potrdilih ali njim pripadajočih listih. V danem trenutku je tako relativno težko določiti, koliko pošiljk se nahaja na določenem delu med prevzemom in dostavo. Sledenje pošiljki je namreč mogoče le v primeru, ko je pošiljka pri dostavi v neko drugo državo zaupana posredniku, ki to storitev podpira, in še to le od mesta predaje naprej.

Težavno je tudi razporejanje pošiljk na dostavne voznike in kontrola dejansko dostavljenega dela pošiljk. Pri dostavi voznik prevzame pošiljke glede na dnevno predvideno pot. Poti so povezane z dostavnim območjem in poštno številko. Če označba na pošiljki ni ustrezna – če je oseba, ki je dano pošiljko uvrstila na določeno pot, pri tem storila napako, ali če je označba neberljiva, to pomeni dodatne stroške zaradi izgubljenega časa in nepotrebnih prevozov pošiljke.

Za analize in obračune plačil je potrebno pozorno spremljanje špediterskih potrdil. Posledica nepozornosti v tem postopku je običajno neustrezen obračun, ki je lahko izveden v škodo podjetja, stranke, ki je pošiljko naročila, ali voznikov transportnih vozil, preko katerih so pošiljke potovale do cilja. Vključeni v tem procesu si pri delu pomagajo z orodji za delo s preglednicami. Končna pot potrdila pa je račun, ki storitev s tega potrdila vključuje.

4.1.3 Pomanjkljivosti pri izdaji računov

Izdajo računov v precejšnji meri pokriva za okolje DOS pisan program, katerega pomanjkljivosti se tekom časa kažejo kot vedno bolj omejujoče. Aplikacija je precej neprilagodljiva in spremembe izven vnaprej določenih okvirov so bolj ali manj neizvedljive. Zaradi zastarelega uporabniškega vmesnika in relativno neintuitivne postavitve preko

tipkovnice dostopnih ukazov je delo s programom uporabniku tudi manj prijazno. Po drugi strani pa je uporabnik, ko enkrat osvoji programu lastne bližnjice preko kombinacij tipk, v okviru razpoložljive funkcionalnosti lahko precej učinkovit. Glavna hiba tega sistema dela je predvsem necentralizirana hramba podatkov in nezmožnost enostavne integracije z ostalimi deli sistema.

Delno razvit, trenutno niti še v procesu testiranja, je tudi nov program za izdajo računov, katerega osnova je Microsoft Access. Bistveni cilji slednjega so poenostavitev izdaje računov na način, ki bi omogočal avtomatično grupiranje posameznih postavk in periodično izdajo glede na nastavljene parametre. Daljnosežni cilj je tudi povezljivost z računovodstvom preko ustreznih izvoznih datotek. Program je v zgodnji fazi razvoja in trenutno vključuje le majhno podmnožico želenih funkcij.

4.1.4 Pomanjkljivosti, povezane z računovodstvom

Računovodstvo je v družbi ACK, d.d. pokrito z aplikacijo podjetja SAOP. Aplikacija kljub nekaterim pomanjkljivostim zadovoljivo opravlja svojo nalogo. Se pa na tem mestu pojavi precejšnje ozko grlo. Knjiženje računov je dolgotrajen, zamuden proces, število le-teh pa se je, še posebej po vstopu Slovenije v Evropsko unijo, precej povečalo. Večino računov je na podlagi ustrezne listine potrebno ročno (še enkrat) vnesti v sistem. Zaradi povečane količine potrebnih vnosov je čas za vnos podrobnejših podatkov še toliko bolj omejen, zato se slednji večinoma izpuščajo. Posledica tega je težaven nadzor, saj se v ozadju posamezne storitve skrivajo pomembne informacije, ki omogočajo podrobnejše razporejanje stroškov na stroškovne nosilce. Aplikacija sicer omogoča avtomatično knjiženje, žal pa vir, ki bi podatke lahko zagotovil v ustreznem formatu in obsegu, ni na voljo.

4.2 ANALIZA POTREB

V naslednjih podpoglavjih bom obravnaval informacijske potrebe, ki jih bo nova programska rešitev morala pokrivati. Na začetku bom predstavil izhodišča in cilje razvoja novega informacijskega sistema, v nadaljevanju pa se bom posvetil podrobnejši obravnavi posameznih izbranih poslovnih procesov.

4.2.1 Izhodišča

Osnova v fazi načrtovanja razvoja novega informacijskega sistema je bila delno razvita aplikacija na platformi Microsoft Access (programski jezik Visual Basic for Applications). Aplikacija je logično in fizično ločena na podatkovni del, ki je trajna hramba za vse pri uporabi programa relevantne podatke, in del, ki vključuje poslovna pravila oziroma poslovno logiko. Slednja je v neki meri že odraz predhodnih zahtev oziroma posledica predhodne

analize potreb. Ključno področje, ki naj bi ga aplikacija pokrivala, pa je le izdaja računov. Razlog za tako odločitev vodstva podjetja in za relativno ozko namembnost aplikacije so bile predvsem že opisane pomanjkljivosti. Aplikacija bi pri tem imela poudarek na avtomatičnem združevanju vnesenih postavk in izračunu cen na podlagi vnesene lestvice. Omenjeno je možno razbrati tudi iz Slike 1 v Prilogi 1, ki prikazuje diagram modela podatkovne baze, ki je bil rezultat začetnih dogovorov v zvezi s podatkovnimi zahtevami predhodnika nove programske rešitve. Eden od sklepov na osnovi tega je bil, da se nadaljuje z razvojem v smeri, ki je že nakazana v razvijajoči se programski rešitvi. Sklep je izhajal iz ocene ustreznosti celotnega projekta in platforme, na kateri je temeljil.

Po prvih testiranjih prototipa, ki so sledili nekaj tednom začetnega razvoja, so se pojavile prve večje pomanjkljivosti zastavljenega projekta. Poleg manjše spremembe zahtev, ki so bile posledica prvih vtisov testiranja prototipa aplikacije, se je pokazalo, da bo izbrana platforma za nadaljevanje razvoja že v obstoječi fazi razvoja neustrezna. Kompromisi, ki bi jih bilo potrebno sprejemati ob vztrajanju na začrtani poti, bi bili dolgoročno preveliki. To je po ponovni oceni ustreznosti vodilo do nekaterih sprememb podatkovnega modela in menjave uporabljene platforme.

V poglavju, ki se nanaša na uvajanje nove programske rešitve, bom izpostavil tudi nekatere težave, na katere bi neizogibno naleteli pri uvedbi aplikacije, bazirane na omenjeni platformi, v redno uporabo.

4.2.2 Cilji razvoja novega informacijskega sistema

Veliko povečanje obsega poslovanja kmalu pokaže, kako pomembna je informacijska podpora poslovnim procesom. Upoštevaje obstoječe stanje, si s strani vodstva podjetja podani cilji razvoja novega sistema sledijo po naslednjem vrstnem redu:

- Doseči čim višjo raven informacijske podprtosti pri ključnih delovnih nalogah, ki so trenutno v precejšnji meri ročno izvajane. To bi namreč pomenilo stroškovno in časovno optimizacijo poslovanja podjetja, saj se tako zmanjšuje število delovnih nalog, ki so potrebne pri vnosu podatkov v sistem, in nad njimi izvedenih osnovnih operacij. Glavnino teh nalog bi potem prevzela programska oprema. Razsežnost zastavljenega cilja in precejšnja časovna občutljivost do neke mere že definirata metodo razvoja in vpeljave novega sistema - inkrementalni razvoj s kolikor je le mogoče visoko ravnijo uporabljivosti vmesnih stopenj. Za dosego slednjih bo potrebna prioritetna obravnava tistih procesov, ki so vzrok za največja ozka grla in pri katerih bo tudi učinek prenove največji.
- Kompatibilnost z ostalimi deli sistema, predvsem izmenljivost podatkov z že vpeljanim računovodskim sistemom. To se navezuje tudi na zgornjo točko, saj omenjeni razkorak trenutno premošča ročno delo.

- Možnost nadzora nad ključnimi delovnimi nalogami in nadaljnja analitična obravnava rezultatov. Dodaten motivator pri tem je uvajanje kontrolinga, ki je že tako tesno povezan z računovodskim sistemom, bolj podrobne informacije pa bodo lahko osnova za bolj kakovostne odločitve. Za slednje so bili navadno na voljo le agregirani podatki.
- Povečanje varnosti in zanesljivosti poslovanja preko možnosti spremljanja poslovnih dogodkov v realnem času. Najpomembnejši del se nanaša na možnost spremljanja poti posameznih enot v procesu in trajno hrambo pridobljenih podatkov. Vključena je tudi možnost omejevanja dostopa. Če bi vsakemu uporabniku sistema lahko skladno s pooblastili omogočili obseg razpoložljivih funkcij, bi se zmanjšala tudi možnost zlorab.
- Dvig kakovosti storitev, ki so jih deležne stranke, ki bo med drugim tudi posledica krajših odzivnih časov.

4.2.3 Analiza posameznih poslovnih procesov

V poglavju, ki sledi, bom opisal pomembnejše procese, ki jih bo pokrivala nova programska rešitev. Pri tem bom upošteval tudi spremembe zahtev, do katerih je prihajalo v postopku razvoja nove programske rešitve. Kompleksnost nekaterih delov opisanih procesov je lahko tudi precej večja, a je celota zaradi nazornejšega prikaza ključnih delov poenostavljeno predstavljena.

4.2.3.1 Proces naročanja in prevzemanja pošiljk

Pri procesu naročanja pošiljk običajno sodelujejo 4 ključne osebe:

- **stranka**, ki naroči storitev, povezano s pošiljko;
- **referent oziroma dispečer**, ki sprejme strankino naročilo;
- **voznik**, ki opravi prevzem pošiljke pri stranki;
- **skladiščnik**, ki izbrano pošiljko sprejme v skladišče.

Običajen potek procesa naročanja pošiljk je naslednji:

Proces se začne pri stranki, ki naroči storitev, povezano s pošiljko. Pošiljka je v tem in v vseh nadaljnjih procesih vedno širši pojem kot paket. Paket je sestavni del pošiljke in vsi paketi, ki neki pošiljki pripadajo, so namenjeni na isto lokacijo. Paketi, ki temu kriteriju ne zadostijo, morajo biti obvezno priključeni k drugi pošiljki. Paketi iste pošiljke na dveh različnih lokacijah (npr. v dveh ločenih skladiščih) namreč pomenijo napako v postopku. Najbolj pogosta pot za posredovanje naročila je telefonski klic, možna načina pa sta tudi fax, elektronska pošta ali osebna oddaja naročila skupaj z oddajo pošiljke v enem od skladišč. Pri zadnjem načinu se proces zaključi, ko skladiščnik sprejme pošiljko v skladišču, pri ostalih pa sta v potek vključena še referent oziroma dispečer in voznik. Referent pri sprejemu klica

naročilo posreduje ustreznemu vozniku, ki ga izbere na podlagi naslova stranke oziroma želene lokacije prevzema pošiljke, ki jo stranka ob naročilu sporoči. Pogoj za dodelitev voznika je, da se le-ta nahaja na ustrezni lokaciji in da ni že preobremenjen z ostalimi dostavami oziroma prevzemi. Stranka pošiljki ob oddaji priloži še izpolnjeno špeditersko potrdilo oziroma ga izpolni ob predaji in nato oboje preda vozniku. Prevzeto pošiljko voznik na koncu dostavi v enega izmed skladišč, kjer jo prevzame skladiščnik. V nekaterih primerih lahko isti voznik pošiljko prevzame pri stranki in jo dostavi še preden jo odda v skladišču. Razlogi za to so približno sovpadanje dostavnega naslova in lokacije nekega drugega prevzema ali neke druge dostave pošiljke.

Najbolj pereči problemi pri tem poslovnem procesu so:

- neobstoj podatkov, povezanih s tem procesom, v elektronski obliki;
- podvajanje dela in veliko prostora za napake – podatki, povezani s posamezno pošiljko, se v procesu posredujejo od ene do druge, v proces vključene osebe vsaj trikrat;
- relativno visoka obremenitev dispečerja;
- visoki manipulativni stroški posamezne pošiljke;
- neobveščenost o zaključku posamezne faze – informacijo o tem, da se je določena faza zaključila, ima v trenutku zaključka le oseba, ki je vanjo neposredno vpletena.

Rešitev prvega problema je čim bolj celovit in natančen zajem podatkov. Glede na potrebe tega procesa in od njega odvisnih procesov, bodo v zajem podatkov vključene naslednje večje skupine le-teh:

- podatki o pošiljatelju (naslov, naziv, …);
- podatki o prejemniku (naslov, naziv, …);
- podatki o storitvi (vrsta storitve, dodatne storitve, cena, …);
- podatki o pošiljki in pripadajočih paketih (teža, prostornina, vsebina, …);
- podatki o vozniku (dodeljen voznik, datum in ura oddaje, …);
- podatki o skladišču (lokacija skladišča, vrsta sprejema, datum in ura sprejema, …).

Namen sprememb v poslovnem procesu pa je rešitev preostalih štirih problemov. Proces se še vedno začne pri stranki, njegov potek pa je od tega mesta naprej nekoliko spremenjen. Pot naročila od stranke do referenta oziroma dispečerja preko zgoraj omenjenih kanalov v nekoliko spremenjeni obliki še vedno obstaja, je pa dopolnjena z novimi možnostmi. Stranka ima za posredovanje naročila sedaj še tri različne možnosti, cilj vseh pa je elektronski zajem tem večjega števila podatkov in odprava podvojenosti podatkov ter zmanjšanje možnosti napak na komunikacijskih kanalih.

Prvi od novih načinov je spletni vmesnik, ki je prikazan na Sliki 2 v Prilogi 2. Gre za spletno aplikacijo, ki stranki omogoča upravljanje s pošiljkami in je obenem najbolj zaželen način oddaje naročila. Primarna funkcija te aplikacije je zajem vseh podatkov o pošiljki, pošiljatelju, prejemniku, storitvi in ostalem na enem mestu. Omogoča tudi združevanje pošiljk v skupine in predaje celotne skupine naenkrat. Stranka preko uporabniškega vmesnika lahko

administrira prejemnike, ki jim pošiljke pošilja večkrat, slednji pa so potem na voljo kot hitra izbira pri vnosu nove pošiljke. Vse oddane pošiljke so v nadaljevanju vidne v arhivu oddanih pošiljk. Pošiljke, ki so oddane preko spletnega vmesnika so takoj dostopne dispečerju, stranka pa si po zaključku vnosa natisne nalepko, s katero nato opremi pripadajoče pakete. Na nalepki se nahajajo vsi potrebni podatki o pošiljki, obenem pa tudi enodimenzionalna črtna koda, ki služi razpoznavi pošiljke v skladišču. Kot potrdilo o prevzemu s strani voznika služi dnevni manifest, ki zagotavlja kvantitativno skladnost prevzetih pošiljk in pripadajočih paketov.

Drugi način oddaje naročila je posebna aplikacija, nameščena na računalniku stranke, ki je neodvisna od prej omenjenega spletnega vmesnika. Hramba vseh za to aplikacijo relevantnih podatkov je računalnik stranke. Aplikacija je po funkcionalnosti sorodna prej opisani spletni aplikaciji, pomembna razlika pa je v možnosti s spletom nepovezanega delovanja. Pošiljanje podatkov aplikacijskemu strežniku poteka le v primeru, ko je na voljo povezava z internetom. Ko povezava ni na voljo, so vsi relevantni podatki na voljo le na nalepki, s katero je pošiljka opremljena. Ta nalepka se od nalepke, s katero operira spletni vmesnik, razlikuje po tem, da je opremljena še z dvodimenzionalno črtno kodo. Vanjo je možno zapisati veliko več podatkov, kot jih dopušča enodimenzionalna črtna koda. Podatki, ki ob prihodu pošiljke v skladišče še niso v podatkovni bazi, se preberejo iz črtne kode in ustrezno shranijo. V načinu nepovezanega delovanja je pri aplikaciji še vedno potrebna komunikacija z enim od poslovalnic ACK, d.d. preko telefona.

Tretji nov način oddaje naročila predstavljajo predtiskane nalepke. Gre za način, ki je komplementaren običajnemu telefonskemu naročilu oziroma predhodno dogovorjenemu intervalnemu naročanju. Prednost tega načina je, da so na predtiskani nalepki že podatki o pošiljatelju. Slednji se nahajajo tudi v črtni kodi na nalepki, ki je enodimenzionalnega formata. Pri opremljanju pošiljke z nalepko je potrebno dodati tudi podatke o pošiljatelju. Ta vrsta naročanja je predvidena za stranke, ki v skladišču oziroma dislocirani poslovalnici podjetja nimajo računalnika, obenem pa storitve, povezane s pošiljkami, naročajo v relativno velikem obsegu.

Pri vseh obstoječih načinih naročanja s pošiljkami povezanih storitev je predvideno dopolnjevanje oziroma elektronski zajem podatkov ob prihodu pošiljke v skladišče. Referent oziroma dispečer ima večino pošiljk običajno v sistemu že ob samem naročilu pošiljke in tako precej olajšano delo. Sistem mu na podlagi vnesenih podatkov pove, preko katerega območja in po kateri poti voznika bo pošiljka prispela do skladišča, njegova naloga pa je, da na ustrezen naslov napoti prostega voznika in to označi v sistemu. Voznik pakete pošiljke ob prevzemu odčita z ročnim skenerjem in označi tudi osebo, od katere je pošiljko prevzel.

Ročni skenerji so novost pri tem procesu in vseh, ki mu sledijo. Njihova glavna naloga je beleženje kod posameznih paketov skupaj z njihovimi statusi, datumom in uro ter voznikom, ki z njim operira. Prenosni (brezžični) ročni skener predstavlja vmesni pomnilnik za omenjene podatke med mestom zajema podatkov na terenu in skladiščem. Opremljen je z optično enoto

za prebiranje črtne kode in šibkejšim računalnikom, ki poleg tega da shranjuje prebrane podatke, le-te v ustreznem formatu tudi posreduje terminalu ob priklopu na bazno postajo.

Ko voznik pošiljko dostavi do skladišča, njene pakete položi na tekoči trak, ki jih odpelje do tehtnice. Na tej jih skladiščnik odčita s skenerjem, obenem pa se v sistem shrani tudi teža posameznih paketov. Elektronska, z računalnikom povezana tehtnica nadomešča, pri velikih težah in velikostih paketov pa dopolnjuje prejšnje načine tehtanja. Bistvena prednost je občutno višja hitrost tehtanja in od osebe, ki tehtanje opravlja, neodvisno posredovanje podatkov sistemu. Na tem mestu se v sistem shranijo tudi podatki o skladišču, ki je pošiljko prevzelo, in o vrsti prevzema pošiljke (npr. navadni sprejem, tranzitni sprejem, napake povezane s prevzemom…). Voznik na terminal v skladišču priklopi tudi ročni skener, ki vsebuje podatke o osebi, od katere je pošiljko prevzel, uro in datum prevzema ter nekatere druge parametre. S tem dopolni že vnesene podatke in zaključi proces.

4.2.3.2 Proces dostave pošiljk

Pri procesu dostave pošiljk običajno sodelujejo vsaj tri od naslednjih oseb oziroma poslovnih subjektov:

- **voznik**, ki pošiljko dostavi končnemu prejemniku oziroma jo preda skladišču;
- **skladiščnik**, ki pošiljko sprejme v (tranzitno) skladišče;
- **prejemnik**, ki pošiljko na koncu postopka prejme;
- **posrednik (zunanji partner)** v primeru, ko dostave končnemu prejemniku ne opravi podjetje.

Proces dostave pošiljk običajno poteka na naslednji način:

Proces dostave pošiljk sledi procesu naročanja in prevzemanja pošiljk. Začne se z voznikovim prevzemom pošiljk, ki so razporejene po lokacijah dostave. V primeru, ko je lokacija končnega prejemnika pošiljke znotraj območja skladišča, v katerega je bila pošiljka dostavljena ob prevzemu, in je voznik za to na voljo, pošiljko neposredno dostavi. V nasprotnem primeru je na poti do cilja vključeno vsaj eno skladišče. Pri sprejemu v skladišče se pošiljka glede na predvideno pot dostave oziroma območje dostave ustrezno razporedi. V primeru zapletov pri dostavi ima lahko pošiljka še več vmesnih postaj, možno je tudi njihovo ponavljanje – če prejemnik ni dosegljiv, lahko pošiljka pot od skladišča do njegovega naslova prepotuje tudi več kot enkrat. Če pošiljke končnemu prejemniku ne bo dostavilo podjetje samo, jo preda posredniku. Storitev odkupne pošiljke in pošiljke, pri kateri je prejemnik plačnik prevoza, prejemnik poravna ob prevzemu, v nasprotnem primeru pa pošiljatelj poravna opravljene dostave pošiljk za preteklo obdobje na podlagi naknadno izdane fakture ali manj pogosto ob predaji pošiljke.

Najbolj pereči problemi pri tem poslovnem procesu so:

- težavno sledenje pošiljki na njeni poti do prejemnika;
- oteženo reševanje zapletov pri dostavah, saj je podpora pripisa odgovornosti z dokazi za nastalo težavo lahko sporna;
- v primeru, da pošiljka krene iz predvidene poti, je njeno lociranje lahko dolgotrajno;
- velik potencial za pridobitev osebne koristi iz naslova odtujitve lastnine in prikrivanja oziroma navajanja neresničnih podatkov;
- delitev pošiljk po dostavnih poteh na posamezne voznike terja nesorazmerno veliko časa.

Začetni korak za rešitev problemov iz prvih štirih točk je elektronski zajem podatkov na vmesnih postajah pošiljke tekom poti do prejemnika. Ti podatki vključujejo:

- status pošiljke;
- osebo (voznik, skladiščnik, prejemnik), ki s pošiljko manipulira;
- datum in uro posamezne operacije nad pošiljko;
- lokacijo dogodka, povezanega s pošiljko.

Proces razporejanja pošiljk je z zajemom večjega števila podatkov in poslovne logike, ki le-te ustrezno transformira, precej olajšan. Na podlagi zgornjih podatkov in podatkov, zajetih v predhodnem procesu, sistem pošiljko že ob prihodu v skladišče na podlagi nastavljivih pravil avtomatično usmeri k naslednjemu skladišču oziroma jo napoti neposredno k prejemniku. Proces razporejanja pošiljk po posameznih dostavnih voznikih je ob prenovljenem procesu mogoče opraviti na dva načina, posledica obeh pa je evidenca spremljajočih podatkov. Prvi način je razporejanje s strani skladiščnika preko programskega vmesnika po različnih parametrih pošiljke. Drugi način pa je potrjevanje pošiljk z ročnim skenerjem. Pri slednjem voznik razporejene pošiljke sprejme in posamezne pakete znotraj vsake ob nalaganju odčita z ročnim skenerjem. Prenos podatkov potem pošiljke v sistemu zaznamuje kot prevzete s strani voznika. Po zaključku tega postopka voznik s sprejemom poročila o prevzetih pošiljkah potrdi, da se strinja z navedeno količino pošiljk, s prostornino in težo. Na poročilu se nahaja tudi podatek o zneskih odkupa, voznin in ostalih storitev, za katere mora voznik plačilo prevzeti že ob trenutku dostave. Ob vsaki predaji pošiljk se le-te vse do končnega prejemnika ponovno evidentirajo v sistemu, skupaj z vsemi prej omenjenimi podatki.

4.2.3.3 Proces izdaje računov in njihovega knjiženja

V procesu izdaje računov običajno sodelujeta dve osebi. Ena od njiju račun izda, druga pa je zadolžena za njegovo kontrolo. Izdaja računa se začne z vnosom vseh potrebnih podatkov v aplikacijo, namenjeno izdaji računov. Pri tem je potreben vnos vsake postavke računa in ročni obračun popustov, ki strankam pripadajo na podlagi dogovorov. Ob zaključku vnosa se posamezen račun natisne, pregleda in odpošlje na naslov stranke. Izdane račune računovodja ročno vnese še v računovodsko aplikacijo, s čimer se proces zaključi.

Pri tem procesu so najbolj problematične naslednje pomanjkljivosti:

- vnos postavk na posamezen račun terja veliko časa;
- ročni vnosi in izračuni so dovzetni za napake;
- evidentiranje posameznih računov je potrebno izvesti dvakrat, prvič pri izstavi, drugič pa pri njegovem knjiženju.

Izdaja računov je v večini primerov povsem deterministične proces. Možno je postaviti pravila oziroma vzpostaviti mehanizem, ki na podlagi predhodno vnesenih podatkov in parametrov oblikuje račune. Mehanizem pri tem združi postavke v skupine in opravi ustrezne izračune. V novem sistemu je posledično za vsako stranko nastavljiva dolžina obdobja izdaje računa, plačilni rok in načini obračunavanja popustov. Posamezen račun pri tem predstavlja skupino oziroma krovno enoto, v katero se na podlagi nastavljivih parametrov razvrstijo ustrezne postavke (npr. pošiljke, ki so v sistemu od naročila naprej). Kriteriji pri tem so plačnik storitve, dolžina obdobja, tipi postavk (dodatne storitve, neobdavčene storitve…) in popusti. Račun gre z vidika aplikacije skozi tri življenjska obdobja. Prvo predstavlja stanje, ko je račun v pripravi (še nepotrjen), drugo, ko je račun izdan, in tretje oziroma končno, ko je račun izvožen. Nepotrjene račune se po potrditvi natisne, in sicer v večjih skupinah. Izdane račune se preko izvoznih datotek prenese v računovodstvo. Te datoteke tudi odpravijo potrebo po ponovnem vnosu računa s postavkami v računovodskem programu.

4.3 PODATKOVNI MODEL

V bazi podatkov ACK, d.d. je trenutno v uporabi nekaj več kot 50 tabel. Pomembnejše med njimi bom v nadaljevanju glede na vlogo v procesu razdelil v 6 logičnih skupin in vsako skupino oziroma splet tabel, ki v skupini zavzamejo pomembnejšo vlogo, okvirno opisal. Na koncu vsake skupine so naštete tudi tabele, na katere se opis nanaša. Pomembnejše tabele z večino atributov so sicer prikazane na Sliki 4 v Prilogi 4.

- Vsaki stranki pripada poljubno število kontaktov, en kontakt le eni stranki. Vsaka stranka ima poleg osnovnih lastnosti tudi določene storitve, za katere se pri naročilu pošiljke lahko odloči, obenem pa še popuste zanje. Vsaka stranka ima enega komercialista in poljubno število predhodno vnesenih prejemnikov (se najpogosteje pojavljajo kot prejemniki). Vsaki stranki je poleg splošne opombe oziroma zaznamka možno opombo beležiti tudi glede na posamezen dan (kot nekakšen dnevnik) (relevantne tabele: »stranke«, »kontakti«, »strankestoritve«, »popustistoritve«, »strankeporocila«, »komercialisti«, »webprejemniki«).
- Pošiljke imajo glede na vir lahko najprej pojavno obliko web-pošiljke oziroma so pošiljke vnesene s spletnega vmesnika, lahko pa so samostojne enote. Vsaka pošiljka ima poljubno število paketov, se pa zaradi potencialnih razlik med vnesenimi in dejanskimi lastnostmi paketa, le-ta hrani dvakrat. Vsaka pošiljka je povezana le z eno storitvijo. Pošiljke, pri katerih nalog za izbris pride iz zunanjega vira, so posebej

označene (relevantne tabele: »webposiljke«, »webpaketi«, »posiljke«, »storitve«, »paketi«, »paketinarocila«, »posiljkeizbris«).

- Vsak paket ima v procesu dostave večje število z dostavo povezanih dogodkov. Vsak dogodek ima začetni in končni status. Če dogodek nima končnega statusa, je ali še v postopku obdelave ali pa iz tega s posebnim parametrom eksplicitno izvzet. Dogodku lahko pripada določen voznik oziroma druga oseba na poti do cilja. Za nekatere kombinacije dogodkov so v sistemu nastavljena pravila, ki pa so odvisna tudi od nekaterih parametrov pošiljke. Dva od teh sta depo in routa. Enemu depoju lahko pripada poljubno število rout, ena routa pa lahko vsebuje poljubno število poštnih številk. Poštna številka prevzema in dostave pa sta parametra pošiljke. Pošiljki lahko pripada le en posrednik, kateremu je bila pošiljka predana. Za namene agregiranega poročanja o dogodkih, povezanih s pošiljko, obstajajo v bazi še sheme za pretvorbo leteh na raven pošiljk. Glede na namen lahko obstaja poljubno mnogo shem, vsaka shema lahko pokriva poljubno število statusov, vsak od teh pa ima določeno mejo prehoda med nivoji (relevantne tabele: »paketidogodki«, »statusikoncni«, »statusizacetni«, »statusiavtkoncni«, »statusiavtdepo«, »statusisheme«, »statusimeje«, »statusiposiljke«, »posredniki«, »soferji«, »route«, »depoji«).
- Vsaki storitvi glede na datum in cono pripada lestvica tež s cenami za oblikovane razrede. Cena nad zgornjim razredom je za vsako dodatno enoto enaka. Vsaka država pripada vsaj eni coni, ena cona pa ima vsaj eno državo. Poleg lestvice obstaja še procentualni in pavšalni cenik za pošiljkam pripadajoče storitve (relevantne tabele: »cene«, »cenedodatni«, »cenestoritve«, »storitvecenika«, »cone«, »drzavecone«).
- Vsak račun ima vsaj eno postavko. Ena postavka lahko pripada le enemu računu. Račun ima v danem trenutku le en status, obenem pa še določen kraj izdaje ter osebo, ki je račun izdala in kontrolirala. Na račun je možno dodati poljubno število opomb. Lahko je izstavljen v določeni valuti, v primeru različnih tečajev in valut postavk pa bodo te glede na izbrano valuto ustrezno pretvorjene (relevantne tabele: »racuni«, »racunistatusi«, »racunikrajiizdaja«, »racuniosebeizdaja«, »racuniosebekontrola«, »racuniopombe«, »tecaji«, »valute«).
- »Nepošiljke« so posebna entiteta, ki se od pošiljk razlikuje v količini zajetih podatkov in namembnosti. Nepošiljka je lahko le enega tipa (relevantni tabeli: »storitvenep«, »neposiljke«).
- Poleg standardnih šifrantov je posebna entiteta tudi del nastavitev, ki so skupne vsem na bazo priklopljenim klientom (relevantne tabele: »poste«, »drzave«, »jeziki«, »obcine«, »skd«, »nastavitvenum«, »nastavitvetext«).

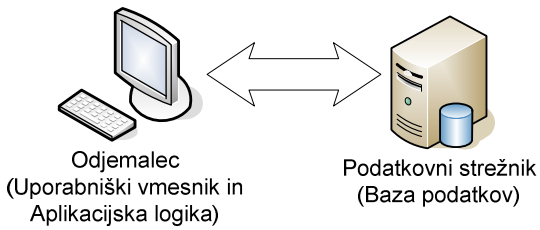
4.4 ARHITEKTURA SISTEMA

Pri načrtovanju informacijskega sistema je že v relativno zgodnji fazi potrebno sprejeti odločitev o tem, kakšna bo arhitektura naše programske rešitve oziroma koliko fizičnih nivojev bo sistem dopuščal. Nivoji se nanašajo na število enot strojne opreme (strežnikov), na katerih je implementirano rešitev možno poganjati. Število nivojev v informacijskem sistemu določajo predvsem njegove potrebe, kljub temu pa sta danes najbolj običajni dvonivojska in tronivojska arhitektura. Skoraj vse poslovne aplikacije so danes vsaj dvonivojske.

DVONIVOJSKA ARHITEKTURA

Predstavlja prvo generacijo arhitekture odjemalec/strežnik. Nekateri to generacijo imenujejo model debeli-odjemalec (ang. »Fat Client«) zaradi hkratne implementacije uporabniškega vmesnika in aplikacijske logike na odjemalcu. Slednja vsebuje tudi vse potrebno za dostop do baze podatkov. Dvonivojsko arhitekturo, kot že samo ime pove, sestavljata dva nivoja - uporabniški nivo (nivo odjemalca) in nivo baze podatkov oziroma SUBP (nivo strežnika). Aplikacija, ki teče na odjemalcu, tekom delovanja izdaja zahtevke podatkovnemu strežniku, ta pa ji na podlagi teh posreduje podatke, ki ustrezajo zahtevkom. Prednost takšnega dostopa je velik izkoristek odjemalca in možnost izgradnje zelo odzivnega uporabniškega vmesnika (Sajko, 1997, str. 33).

Slika 2: Dvonivojska arhitektura



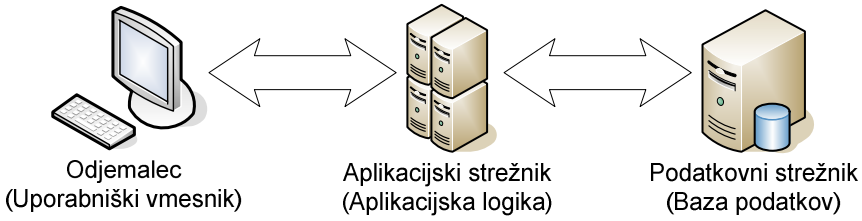
Vir: Lasten prikaz.

TRONIVOJSKA ARHITEKTURA

Pomeni premik dela aplikacijske logike in logike dostopa do baze v tretji nivo - aplikacijski strežnik. Koliko aplikacijske logike je distribuirane na odjemalcu in koliko na strežniku, je odvisno od faktorjev, ki upoštevajo zahteve po zmogljivosti, stroške vzdrževanja in nadgrajevanja, dostopnosti do strežnika in razpoložljivosti razvojnega orodja, ki zadovolji potrebo po različnih modelih odjemalec/strežnik.

Aplikacijski strežnik je popolnoma samostojen programski del aplikacije, ki se lahko odvija na posebni platformi, kjer izvaja celotno logiko aplikacije in je izključno orientiran na problematiko, ki jo aplikacija pokriva. Z odjemalčevega stališča je aplikacijski nivo strežnik, medtem ko je za nivo podatkovne baze aplikacijski strežnik odjemalec.

Slika 3: Tronivojska arhitektura



Vir: Lasten prikaz.

V povezavi s tronivojsko arhitekturo bi na tem mestu omenil še implementacijo poslovnih pravil. Poslovno pravilo je izjava, ki definira ali omeji nek vidik poslovanja. Njen namen je postavitev poslovne strukture, nadzor ali vpliv na potek poslovanja. Poslovno pravilo je elementarno oziroma nerazdružljivo v smislu, da ga ni moč razbiti na manjše enote, ne da bi pri tem tvegali izgubo pomembnih informacij. Primer poslovnega pravila bi bil recimo: »če stranka pri naročilu preseže vrednost 10.000,00 EUR, se ji prizna 5 % popust«.
Možnih je več načinov implementacije poslovnih pravil, vsak pa ima svoje prednosti in pomanjkljivosti. Danes najbolj pogosto uporabljeni načini so (Bajec, Krisper, Rupnik, 2000, str. 77-84):

- **Poslovna pravila so del programske kode:** Prednost tega načina implementacije je hitrost izvedbe. Slabost tega načina je, da vsaka sprememba zahteva programiranje in vpletenost informatika, poslovna pravila so raztresena po aplikacijski logiki in jih je zato težje locirati. Prepletenost pravil v programski kodi zahteva tudi precejšnjo pazljivost pri spremembah.
- **Parametrizirana aplikacija:** Pri tej implementaciji poslovnih pravil obstaja možnost spremembe pravila preko uporabniku prijaznih nastavitev parametrov. Četudi so poslovna pravila še vedno del aplikacijske kode, parametri omogočajo njihovo usmerjanje. Slabost tega načina je, da veliko breme leži na razvijalcu aplikacije, saj mora vnaprej predvideti karseda veliko število scenarijev, ki se lahko odvijejo tekom življenjske dobe aplikacije.
- **Uporaba strežniških procedur SUBP:** Prednost tega načina je, da so poslovna pravila spremenljiva neodvisno od aplikacijske logike. Formalna predstavitev je izvedena v jeziku SQL, ki je precej podoben naravnemu jeziku. Poslovna pravila je možno spreminjati tudi na daljavo, ta pa so obenem locirana na enem mestu. Dodajanje novega poslovnega pravila prav tako ne zahteva nujno spremembe programske kode. Slabost te metode je odvisnost od ponudnika SUBP zaradi razlik med verzijami jezika SQL. Implementacija zelo kompleksnih pravil je vprašljiva, obenem pa je problematično tudi dejstvo, da je SUBP v tem primeru v funkciji, za katero je slabše optimiziran kot običajni programski jeziki.
- **Uporaba strežniških procedur, neodvisnih od ponudnika SUBP:** Lastnosti so enake tistim v predhodni točki, razlika je v tem, da gre tu za dodatni vmesni nivo med

SUBP in aplikacijskim strežnikom. Izhod so še vedno strežniške procedure, ki pa so zaradi abstrakcije neodvisne od SUBP.

- **Procesor poslovnih pravil:** Poslovna pravila so zajeta v repozitoriju poslovnih pravil na aplikacijskem strežniku. Slednji predstavlja nivo med bazo podatkov in namizno aplikacijo. Prednost tega načina je tehnološka neodvisnost, saj je ta nivo preko vmesnikov povezljiv z ostalimi deli aplikacije in je neodvisen od ponudnika SUBP.
- **Motor za poslovna pravila (ang. »Rules Engine«):** Gre za poseben, od aplikacijskega strežnika ločen nivo, ki je zadolžen izključno za procesiranje poslovnih pravil. Prednosti tega načina so predvsem končnemu uporabniku lažje razumljiv programski jezik, ki se za razliko od klasičnih ne izvaja vedno od zgoraj navzdol. Pravila se lahko začnejo izvajati s katerekoli točke v skupini poslovnih pravil. Zaradi te in še nekaterih drugih lastnosti so nepopolna ali podvojena pravila hitro opazna, verjetnost usklajenega delovanja pa je tako večja. Prednost tega načina je tudi enostavnejše delo z nepopolnimi ali manjkajočimi informacijami, saj lahko operira tudi z vrednostmi, kot je na primer »neznano«. Neodvisnost od ostale programske logike prinese hitrejše prototipiranje, saj spremembe ne zahtevajo ponovne izgradnje programa (ang. »Recompilation«).

4.4.1 Prednosti in slabosti dodatnega nivoja

Kompleksnost načrta programske rešitve raste sorazmerno s številom nivojev. Večja kompleksnost pomeni več potrebnega časa za načrtovanje in implementacijo. To se odrazi v zvišanju stroškov razvoja, sčasoma pa tudi stroškov vzdrževanja. Ključne lastnosti, ki nam jih pravilna izbira števila nivojev lahko nudi, so (Lhotka Rockford – Should all apps be n-tier?, 2005):

- zmogljivost;
- razširljivost (ang. »scalability«);
- odpornost na napake;
- varnost.

Ker je praktično nemogoče doseči maksimalen rezultat na vseh štirih oseh, je potrebno sprejeti kompromis na podlagi naših prioritet.

ZMOGLJIVOST

Pri izbiri se je potrebno zavedati, da je prečkanje meje med posameznimi nivoji drago. Z vidika programske rešitve je klic oziroma povezava iz enega na drug nivo lahko tudi tisočkrat počasnejša kot operacija na istem nivoju. To je očitna posledica ovir, ki jih mora izdan zahtevek premostiti – od programskih mej znotraj nivoja do omrežja izven fizičnega okvira izbranega nivoja (Lhotka, 2006, str. 1-33).

RAZŠIRLJIVOST

Prednost tronivojske arhitekture je predvsem razširljivost. Povečanje števila uporabnikov dvonivojske aplikacije istočasno povečuje tudi količino podatkov in število operacij, ki se izvajajo na strežniku. Edina možnost, da se ohranijo zmogljivosti celotnega sistema, je povečanje kapacitete strežnika, kar pa ima seveda svoje meje. Ta problem se lahko pri tronivojski aplikaciji relativno enostavno reši z dodajanjem novega strežnika za aplikacijski nivo. Na vsakem od strežnikov se lahko odvija isti proces, uporabnik pa dostopa do tistega strežnika, ki je trenutno najmanj obremenjen. Vsi strežniki poslovnega nivoja komunicirajo s strežnikom, na katerem se nahaja podatkovna baza. Strežnik je v tem primeru manj obremenjen, saj se na njem ne odvijajo več procesi poslovnega dela aplikacije (Lhotka, 2006, str. 1-33) .

ODPORNOST NA NAPAKE

Gre za bolj kompleksen problem od razširljivosti. Doseganje odpornosti na napake zahteva pregled vseh točk potencialne odpovedi, ki obstajajo med uporabnikom in bazo podatkov in nenazadnje tudi pregled SUBP. Pri natančni analizi bi bilo kot možno mesto odpovedi potrebno obravnavati tudi uporabnika, še posebno pri procesno orientiranih ali storitveno orientiranih sistemih. V večini primerov dopolnitev sistema z aplikacijskim strežnikom ne poveča odpornosti na napake. Precej bolj verjeten scenarij je, da se celoten postopek zagotavljanja občutljivosti na napake zaradi dodatnega nivoja le še podraži, saj nam nov element v sistemu naloži še dodatno skrb (Lhotka, 2006, str. 1-33).

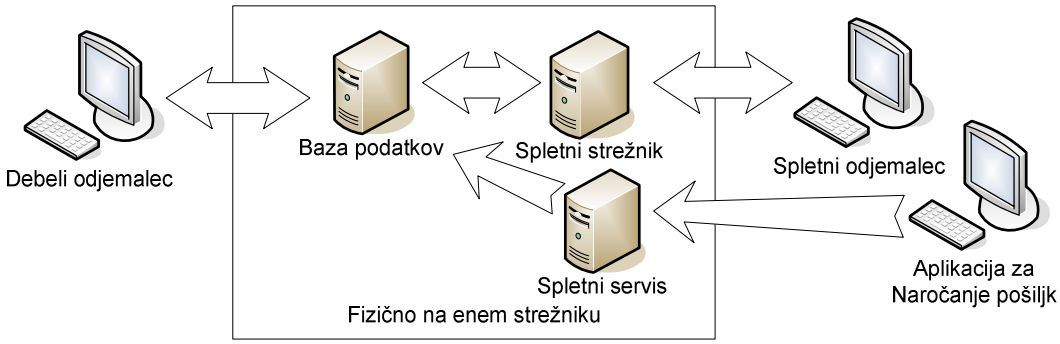
VARNOST

Varnost je zelo širok pojem, vseeno pa v večini organizacij pomeni predvsem varovanje baze podatkov oziroma omejevanje dostopa do nekaterih njenih delov. Dodatni nivo lahko pri tem koristi, saj centralizira vso aplikacijsko logiko oziroma del celote, ki komunicira z bazo. Tako ima le ta nivo ključ za dostop do baze podatkov, kar lahko potencialno dvigne nivo varnosti celotnega sistema (Lhotka, 2006, str. 1-33).

4.4.2 Izbrana arhitektura

Na podlagi prvotnih zahtev in analize potreb je bila pri informacijskem sistemu ACK, d.d. izbrana dvonivojska arhitektura, bi pa bilo mogoče gledati na del sistema, ki teče na podatkovnem strežniku skupaj s SUBP, kot na tretji nivo. Vsebuje namreč precejšen del poslovne logike, ki je v uporabi preko dveh tipov odjemalcev – preko spletne aplikacije in preko namizne aplikacije. Preostali del poslovne logike, ki ne teče na podatkovnem strežniku, je implementiran preko parametriziranega dela namizne aplikacije in v manjši meri tudi kot del spletne aplikacije. Razlogi za izbiro tega tipa arhitekture so predvsem želja po čim višji odzivnosti sistema in v fazi razvoja izražena želja po čim hitreje dostopni rešitvi. Ustrezen nivo pooblastil bodo tako določale pravice uporabniških računov na SUBP.

Slika 4: Arhitektura informacijskega sistema ACK, d.d.



Vir: Lasten prikaz.

4.5 IZBIRA RAZVOJNE PLATFORME

Na razvojno platformo se pogosto gleda kot na manjšo tehnično podrobnost v razvoju oziroma na nekaj, kar je izključno v domeni programerja. Se pa hitro lahko zgodi, da nepazljiva izbira s seboj prinese precej dodatnih stroškov. Ti so običajno posledica vzdrževanja informacijskega sistema na daljši rok. Manjša zastopanost na trgu pomeni manjšo ponudbo razvijalcev programskih rešitev, običajno slabše vzdrževanje razpoložljivih orodij (skrb za njihovo kvaliteto), slabšo dokumentacijo in nenazadnje višjo ceno. Vse to seveda velja ob predpostavki, da izvorna koda za vse programske komponente informacijskega sistema ostane v rokah podjetja, v nasprotnem primeru pa je podjetje vezano na dobavitelja za celotno obdobje uporabe sistema in je razprava o tem brezpredmetna.

Izbira platforme za informacijski sistem ACK, d.d. je potekala med platformama .NET in JAVA. Prva sodi pod okrilje Microsofta, druga pa pod okrilje SUN-a. Razlogi za omejitev na ti dve platformi so bili velika razširjenost, relativno velika prenosljivost, dobra hitrost razvoja in relativna enostavnost. JAVA je odprtokodna in ima precej dobro podporo tudi v zelo velikem številu ne-Microsoftovih programskih okolij (Linux, FreeBSD, Solaris, OSX…). .NET je zastopan v nekoliko manjšem številu okolij, podpora platformi pa je najboljša v okolju Windows (Microsoft). V Linux-u se platforma razvija pod okriljem posebnega odprtokodnega projekta, imenovanega Mono, in ima prav tako relativno dobro podporo, nekatera druga programska okolja (npr. FreeBSD) pa so slabše podprta.

Od zgoraj omenjenih platform je bila izbrana Microsoftova. Na končno odločitev je v precejšnji meri vplivalo dejstvo, da bodo vse razvite aplikacije tako ali tako tekle le v okolju Windows. Če pa bi se kdaj pokazala večja potreba po prenosljivosti na kako drugo okolje, bi bilo, kot sem že omenil, tudi to možno izvesti. Drug pomemben faktor pri odločitvi je dejstvo, da je bil prototip razvit v Accessu, zaradi česar je bil možen prenos precejšnjega dela programske kode na novo platformo brez večjih sprememb.

4.6 IZBIRA SISTEMA ZA UPRAVLJANJE Z BAZAMI PODATKOV

Izjemno pomemben del vsakega poslovnega informacijskega sistema je baza podatkov. Poslovne aplikacije namreč praviloma spadajo v skupino podatkovno orientiranih aplikacij (ang. »Data-centric Applications«). Ker se skoraj vsak proces v takem informacijskem sistemu začne in konča pri bazi podatkov, zmogljivost sistema za upravljanje z njo (SUBP) pogojuje tudi zmogljivost celotnega sistema. Ponudba na trgu SUBP je velika in zajema vse od manjših sistemov, namenjenih predvsem integraciji v manjše programske rešitve (npr. SQLite, VistaDB…), pa do tistih, ki lahko pokrivajo tudi potrebe kompleksnejših sistemov in velikih podatkovnih skladišč, katerih količina podatkov se meri v terabajtih. Kljub temu pa omenjeni trg lahko razdelimo na dva, po značilnostih precej različna segmenta. Tako na enem kot na drugem obstaja le nekaj velikih ponudnikov. Glede na potrebe ter predvidene obremenitve informacijskega sistema ACK, d.d. bom v nadaljevanju večjo pozornost posvetil naslednjim:

- Odprtokodni SUBP:
  - MySQL,
  - PostgreSQL.

- Lastniški (ang. »Proprietary«), komercialni in zaprtokodni SUBP:
  - Oracle,
  - DB2,
  - SQL Server.

Najbolj očitna ločnica med prvo in drugo skupino z vidika podjetja je prodajna cena posameznega SUBP. Zelo omejeni viri pri razvoju informacijskega sistema ACK, d.d. so izbiro implicitno omejili le na prvo skupino, končna odločitev je tako bila sprejeta po presoji prednosti in slabosti MySQL oziroma PostgreSQL.

Skupek faktorjev, ki vplivajo na oceno primernosti posameznega SUBP, bom zajel v nadaljevanju. Obenem bom poskušal pokazati, da se odprtokodna rešitev pri vprašanju SUBP dolgoročno lahko izkaže celo kot manj tvegana, vsekakor pa cenejša izbira.

4.6.1 Stroški licenc in podpore

LASTNIŠKI SUBP
Stroški licenc lastniških SUBP lahko pri manjšem informacijskem sistemu predstavljajo kar zajeten del celotnih stroškov. Njihove prodajne cene so ob prihodu večjedrnih procesorjev še dodatno narasle. Razlog za to je v cenovni politiki vezave licence na jedro procesorja. Prav to je za razliko od predhodnih let, ko se je moč posameznega procesorja večala predvsem na račun njegove frekvence, sedaj botrovalo nesorazmernemu povečanju stroškov. Oracle je na

primer, da bi to nekoliko ublažil, v začetku leta 2006 nekoliko prilagodil cenovno politiko, v februarju 2007 pa je pri svoji standardni različici SUBP sledil Microsoft-u in licenco vezal na posamezen čip (Industry Pushes Back On Oracle's Multicore Licensing Policies, 2007).

Cene licenc lastniških SUBP so poleg že zgoraj omenjenega odvisne od:

- različice – vsak ponudnik jih ima na trgu več, od okrnjene verzije (pogosto je v njihovem imenu beseda »Standard«), namenjene manjšim podjetjem, do svojega glavnega proizvoda (»Enterprise«);
- števila hkratnih uporabnikov;
- bilateralnih dogovorov oziroma predhodnih pogodb ter ponudbe, ki spada v skupino vezane prodaje.

Zaradi vseh zgoraj omenjenih faktorjev ima pri lastniških SUBP nabavna cena, če upoštevamo celotni asortiment, relativno velik razpon. Osnovne različice, namenjene manjšim podjetjem (tako recimo pri Microsoft-u in IBM-u), trenutno nosijo ceno okoli 8.000,00 EUR, pri Oracle-u pa nekaj več kot 10.000,00 EUR. Cene za najmočnejše različice se pri vseh začnejo okoli 30.000,00 EUR, navzgor pa so načeloma omejene le s potrebami stranke in razpoložljivimi sredstvi.

Vsak od obravnavanih ponudnikov lastniških SUBP ima v svoji ponudbi poleg osnovne verzije še dodatno okleščeno različico. Pri IBM-u se imenuje DB2 Express-C, pri Oracle-u ji pravijo Oracle Database Lite, pri Microsoft-u pa SQL Server 2005 Express Edition. Vsaka od omenjenih verzij ima svoje omejitve, ki so pri dveh od njiju kar precej hude. Pri Microsft-u je velikost celotne baze podatkov omejena na 4 gigabajte (GB), prosto dostopna različica ima le en procesor in je zmožna uporabiti le 1 GB pomnilnika, nima pa možnosti indeksiranega iskanja po polnem tekstu. Omejitev velikosti celotne baze je enaka tudi pri Oracle-u, ki temu dodaja še omejitev na največ 64 hkratnih povezav. IBM-ova ponudba je najbolj velikodušna, saj sta omejitvi 2 jedri procesorja in 2 GB pomnilnika, prostorsko pa velikost baze podatkov ni omejena. Microsoft ima v svoji ponudbi SUBP poleg omenjenega Server 2005 Express Edition tudi Access, ki je del pisarniške zbirke Microsoft Office. Cena te pisarniške zbirke z Access-om se giblje okrog 300,00 EUR, ima pa še nekoliko ostrejše omejitve kot njegov večji brat – najbolj očitna je omejitev velikosti baze na 2 GB (DB2 Express-C, 2008; Oracle Database Lite 10g, 2008; SQL Server 2005 Express Edition, 2008).

Omenjene precej okrnjene SUBP ponudniki le-teh predstavljajo na različne načine – kot pripomoček pri razvoju programskih rešitev pred redno uporabo, testiranje obremenitev pred uvedbo, kot hrambo za dokumente in potrebe manjših pisarn… Želja ponudnikov pa je seveda, da s časom, ko se potrebe podjetja povečajo, to preide na plačljivo različico SUBP. Eden pogostejših razlogov, zakaj bi si tak prehod želeli, je tako imenovani »Vendor Lock« oziroma odvisnost od ponudnika. Zaradi manjših razlik med prej omenjenimi ponudniki in tudi med odprtokodnimi alternativami prenos razvitega informacijskega sistema iz ene platforme na drugo običajno ni trivialen. Četudi naše ocenjene potrebe v času razvoja ne presegajo

postavljenih omejitev okrnjenih verzij, bi bilo dolgoročno priporočljivo planirati tudi nakup licenc.

Eden izmed zadržkov pri odločanju za enega od omenjenih SUBP je, da uporabnik slednjega nima nobene garancije, da bo izbrana prosto dostopna različica tudi na dolgi rok na voljo pod enakimi pogoji. Vprašljivi sta tudi podpora ponudnika za prosto dostopno različico na dolgi rok in odprava napak, ki so pri vsakem večjem skupku programske opreme običajno neizogibni. Pri vseh plačljivih različicah je za to sicer poskrbljeno, je pa podpora v okviru licenčne pogodbe časovno omejena. Do novih različic SUBP pri nobenem od ponudnikov praviloma ne bomo upravičeni, so pa v sklopu pogodbe krite manjše izboljšave in odprava programskih napak.

ODPRTOKODNI SUBP
MySQL in PostgreSQL sta prosto dostopna SUBP, tako da o stroških za nakup licenc ne moremo govoriti. Tako pri izbiri nismo omejeni ne pri številu, ne pri velikosti naše podatkovne baze, cena pa je neodvisna tudi od strojne opreme. PostgreSQL je na voljo pod BSD licenco, ki nas pri uporabi praktično ne omejuje, MySQL pa je na voljo pod GPL licenco, pri kateri v primeru notranjega razvoja tudi ni ovir pri uporabi. V primeru nadaljnje prodaje programske rešitve je na voljo tudi komercialna licenca, ki to omogoča. Oba sistema imata zelo dobro urejeno dokumentacijo in navodila za uporabo, ki so dostopna preko spletnih strani. Pomoč je MySQL uporabnikom na voljo tudi preko uporabniških skupin na forumu domače strani, pri PostrgreSQL pa preko seznamov za elektronsko pošto (ang. »Mailing Lists«). Če bi se izkazalo, da to ne pokriva naših potreb, pa je tako pri enem kot pri drugem na voljo tudi plačljiva pomoč preko raznih neodvisnih ponudnikov.

Zaradi relativno velike in dejavne skupnosti uporabnikov in razvijalcev je pri PostgreSQL-u tveganje, da bi nadaljnji razvoj obstal ali da bi prešel v območje lastniške programske opreme, relativno majhno. Tveganje pri MySQL-u pa je zaradi povezav z Oraclom nekoliko večje. Oracle je namreč lastnik enega od pomembnih sestavnih delov njihovega glavnega proizvoda. Pritiski s strani siceršnjega konkurenta pa se že manifestirajo kot težnja po vedno večjem uvajanju plačljivih storitev (Oracle Announces the Acquisition of Open Source Software Company, Innobase, 2008).

4.6.2 Zmogljivost

Zmogljivosti posameznega SUBP je relativno širok pojem in v veliki meri odvisen od okoliščin oziroma področja uporabe. Težko bi za katerega od sistemov trdili, da je na vseh področjih boljši od drugega. Najboljšo možno odločitev bi lahko sprejeli le, če bi vsak SUBP testirali v povezavi z našim informacijskim sistemom pod različnimi obremenitvami in z različno količino podatkov. Pri tem bi vsako za nas pomembno lastnost oziroma rezultat specifičnega dela testa posebej ponderirali in na podlagi končne ocene izbrali zmagovalca.

Ker pa je to v praksi težko izvedljivo, se lahko zanesemo le na rezultate, ki jih objavljajo neodvisne institucije, kot je na primer »The Standard Performance Evaluation Corporation« (SPEC). Problem nekaterih plačljivih SUBP je v tem, da njihova licenčna pogodba prepoveduje objavo neodvisnih primerjav brez privolitve ponudnika (Oraclov sporazum o licencah in storitvah, 2008). Kompleksnost administracije SUBP in zgoraj omenjena odvisnost od okoliščin to omejevanje po eni strani opravičujeta, po drugi strani pa porajata dvome o objektivnosti primerjav s potrjenimi testi.

Vsi obravnavani lastniški SUBP sodijo med visoko zmogljive. Iz rezultatov v Tabeli 1 iz Priloge 5 pa je mogoče razbrati, da tudi odprtokodni ne zaostajajo prav veliko.

Iz omenjene tabele je razvidno, da je razlika pri danem testu med Oraclom in PostgreSQL-om 12 % oziroma 42 % (pri močnejši strojni opremi) v prid Oracla. Rezultati seveda zaradi razlike v strojni konfiguraciji niso neposredno primerljivi, test pa tudi ni odvisen izključno od SUBP. Testi so zanimivi, če se pri njih upošteva še cenovna komponenta – v celoti »odprtokodna« platforma sicer nekoliko zaostaja, bi pa za relativno majhno razliko v zmogljivosti morali plačati nesorazmerno več. Test tudi pokaže, da je PostgreSQL dosegel nekoliko boljši rezultat kot MySQL pri zelo podobni strojni konfiguraciji.

4.6.3 Nabor funkcij

Obravnavani SUBP se pri naboru osnovnih funkcij bistveno ne razlikujejo in bolj ali manj sledijo standardu (Comparison of different SQL implementations, 2008).

Glede na planirane potrebe informacijskega sistema ACK, d.d. naj bi izbran SUBP vseboval tudi nabor naslednjih funkcij (Spletna stran PostgreSQL, 2008; Oracle, MySQL and PostgreSQL feature comparison, 2008).

- **Strežniške procedure:** Vsi obravnavani SUBP imajo integrirano možnost poganjanja strežniških procedur. Gre za možnost izvedbe programa, pisanega v programskem jeziku, specifičnem za SUBP, in tesno povezanim s SQL-om ter bazo podatkov. MySQL je slednje sicer dobil šele v različici 5.0, a je bila ta opcija na voljo že na začetku razvoja informacijskega sistema. Strežniške procedure bodo služile za od odjemalca neodvisno implementacijo dela poslovne logike – kot vmesni nivo med bazo podatkov in odjemalcem. Obenem bo njihova posledica tudi povečana odzivnost nekaterih kompleksnejših delov sistema, saj bo obdelava podatkov pri teh potekala na samem strežniku. Prenos podatkov bo posledično po omrežju do klienta potekal tudi v zmanjšanem obsegu, hkrati pa bo potrebno med njima manjše število poti.
- **Sistem pogledov:** Pogledi pomenijo nivo abstrakcije med podatki, shranjenimi v eni ali več tabelah, in končnim uporabnikom. Pomembna pri tem je predvsem možnost omejevanja količine podatkov, ki so vidni trenutnemu uporabniku. Njegov pogled na podatke se lahko razlikuje od pogleda drugega uporabnika zaradi razlik v dodeljenih

pooblastilih. Poglede podpirajo vsi obravnavani SUBP, se pa razlikujejo pri možnosti spreminjanja podatkov v pogledu. PostgreSQL podpira spreminjanje podatkov, vključenih v izbran pogled preko ročno nastavljivih pravil, po katerih se spremembe teh podatkov odrazijo na izvornih tabelah. Pri vseh ostalih SUBP je ta proces v precejšnji meri avtomatiziran, ponuja pa PostgreSQL-ov način precej več fleksibilnosti, saj s pravili lahko določimo tudi, kako se bodo na izvoru odrazile spremembe agregiranih podatkov.

- **Posnetke ali matearilizirane poglede:** Pri teh gre za podobno funkcionalnost kot pri sistemu pogledov, s to razliko, da je pogled na dane podatke ustvarjen na določen časovni interval in shranjen za uporabo do naslednje osvežitve – pri navadnem pogledu se dinamično ustvari ob vsakem dostopu do podatkov, ki jih vsebuje. Prednost posnetkov pred navadnimi pogledi je predvsem v hitrosti izvedbe poizvedb. Ker so podatki v posnetku materializirani (shranjeni do naslednje osvežitve), so precej hitreje dostopni. Vsi SUBP to funkcionalnost do neke mere podpirajo, vendar je le pri lastniških SUBP ta funkcionalnost privzeta in v celoti na voljo. Pri obeh odprtokodnih SUBP je delovanje posnetkov mogoče posnemati s strežniškimi procedurami.
- **Povezljivost z izbrano platformo:** Povezljivost pri različnih SUBP zagotavlja veliko število temu namenjenih vmesnikov. Na začetku razvoja informacijskega sistema ACK, d.d. pri izbrani platformi za PostgreSQL še ni na voljo vmesnika, ki bi bil s kvaliteto na zadovoljivi ravni, ostali SUBP pa so ga imeli.
- **Sprožilce:** Pomenijo možnost vezave ukaza ali strežniške procedure na nek določen dogodek, ki je lahko sprememba, vnos ali izbris zapisa oziroma kakšna druga, za SUBP specifična sprememba. Vsi obravnavani SUBP to funkcionalnost podpirajo.
- **Učinkovito izvršitev kompleksnih SQL stavkov:** Z izjemo MySQL-a so obravnavani SUBP zmožni uspešno izvršiti tudi zelo kompleksne SQL stavke. Pri tem uporabljajo podrobno nastavljiv optimizator poizvedb, ki temelji na stroških posameznih elementov. Stroški nekaterih izmed njih so implicitno določeni s statistično analizo podatkov v tabelah. MySQL ima pri kompleksnejših poizvedbah precejšnje težave, še posebno, če poleg velikega števila povezav vsebujejo še veliko število podpoizvedb. Precejšnje število tovrstnih težav pa je mogoče premagati z nekoliko bolj pazljivim oblikovanjem poizvedb oziroma »ročnim« usmerjanjem njihovega izvajanja. PostgreSQL si po drugi strani pri zelo zahtevnih problemih pomaga z naprednimi genetskimi algoritmi, ki izdelavo načrta za izvršitev take izvedbe znotraj SUBP časovno precej skrajša.
- **Podpora transakcijam:** Transakcije zagotavljajo, da se zaporedje ukazov, ki jih naložimo, SUBP izvede v celoti, ali pa sploh ne. Obenem zagotavljajo tudi nivo izolacije uporabnikov, kar ob ustrezni izbiri nivoja izolacije preprečuje, da bi bile spremembe podatkov, ki jih določen uporabnik še ni potrdil, vidne ostalim uporabnikom SUBP. Vsi obravnavani SUBP zagotavljajo to funkcionalnost, ki je pri PostgreSQL-u še nekoliko naprednejša. Podpora s strani transakcij je namreč zagotovljena tudi za ukaze, ki spreminjajo metapodatke. Pri Oraclu na primer kreiranje nove tabele podatkov pomeni implicitno potrditev transakcije, ki je v teku (četudi je to

v nasprotju z našimi željami) in z vidika razveljavitve transakcije nepreklicno potezo. Pri PostgreSQL-u pa lahko v primeru napake, ki se pojavi v enem od ukazov po dopolnitvi baze z novo tabelo, celotno transakcijo še vedno razveljavimo.

- **Varnost – omejevanje dostopa:** Vsi SUBP z izjemo MySQL-a poleg uporabniških računov podpirajo tudi vloge oziroma uporabniške skupine. To precej olajša administracijo uporabniških računov, saj lahko uporabnike glede na nivo pooblastil združimo v skupine oziroma jim dodelimo ustrezno vlogo.

4.6.4 Izbrani SUBP

SUBP, ki je bil izbran pri načrtovanju informacijskega sistema ACK, d.d., je MySQL. Razlog za to je bila na prvem mestu dobra predhodna seznanjenost z MySQLom in dobre izkušnje pri uporabi tega SUBP. Gre predvsem za hitrost MySQL-a pri izvajanju enostavnih poizvedb. Pomemben argument v prid izbiri je bila tudi dostopnost programskega vmesnika za delo na izbrani platformi.

Izkušnje po dveh letih uporabe pa niso bile predvsem pozitivne. Razlogi za to so v težavah, ki so se pojavile tekom razvoja in uporabe tega SUBP. Mogoče bi bilo trditi, da je za manj zahtevne aplikacije, kot so recimo enostavnejši spletni portali in spletni forumi zmogljivost SUBP v smislu števila enostavnih transakcij na sekundo, ki jih ta zmore najpomembnejša komponenta. Trditev je mogoče utemeljiti z dejstvom, da relativna enostavnost podatkovnega modela in poizvedb, povezana z željo stranke po čim večjem odzivu, lahko odtehta občasne nepravilnosti. Malenkostne neskladnosti, povezane s statistiko obiska, so zlahka opravičljive, če ob tem zaradi odzivnosti stranko sploh pridobimo. Povsem druga kategorija pa so finančni podatki in kompleksne poizvedbe, ki se nanašajo na stranke in njihove storitve. MySQL je tako že v procesu razvoja terjal več pazljivosti pri preverjanju formata v bazo vnesenih podatkov in s tem povezano dodatno testiranje. Hibe, ki so se pri tem pokazale, pa so naslednje:

- ob preverjanju tujih ključev zapisa v primeru napake sistem ni vrnil informacije o tem, na katerem mestu oziroma pri katerem polju je do napake dejansko prišlo;
- neupoštevanje smeri razvrščanja seznama zapisov pri poizvedbi;
- nepravilno izvajanje podpoizvedb, pri katerih je iskan parameter podpoizvedbe »null«;
- nepravilno vrnjena vrednost skalarne podpoizvedbe;
- nekajkratno zmrzovanje SUBP.

Poleg omenjenega se je pokazalo tudi, da je večja kompleksnost nekaterih poizvedb že iz domene tistega, kar MySQL opravlja dobro. Tako bi bil PostgreSQL ob trenutnih razmerah na trgu odprtokodnih SUBP veliko boljša izbira. Zanimiva bi bila tudi primerjava stroškov zgoraj omenjenih težav z nabavno ceno komercialnega SUBP, ki pa je zaradi relativno težavnega vrednotenja posameznih kategorij in nepopolnih podatkov o frekvenci in posledicah pripetljajev ne morem z opraviti s sprejemljivo zanesljivostjo.

5 UVEDBA NOVEGA INFORMACIJSKEGA SISTEMA

V naslednjih treh poglavjih bom opisal sklepno fazo razvoja programske rešitve - uvedbo. V prvi točki bom predstavil uvedbo prototipa in jo preko dopolnitve zahtev v drugi točki povezal z uvedbo končne različice.

5.1 TESTIRANJE PROTOTIPA

Prototip programske rešitve na platformi Microsoft Access je predstavljal prvi poizkus pokritja informacijskih potreb poslovanja. Razlogi za izbiro Accessa na samem začetku so bili:

- poznavanje okolja s strani uporabnikov;
- zaradi njegovih lastnosti ga je možno uvrstiti v skupino RAD (ang. »Rapid Application Development«) orodij – aplikacije, razvite v Accessu, pogosto za enak (ali vsaj podoben) učinek zahtevajo precej manj programske kode in zato omogočajo relativno hitro dostopne rešitve, ki posledično pomenijo manj porabljenih potrebnih sredstev v fazi razvoja (Microsoft Access within an Organization's Database Strategy, 2008);
- integracija z ostalimi orodji iz skupine Microsoft Office;
- enostavno upravljanje z bazo podatkov – preprosto kopiranje datoteke pri arhiviranju, trivialna instalacija na drugem računalniku;
- ocenjene potrebe so bile v okviru omejitev, ki jih predstavlja okolje.

Kljub temu, da je prototip pokrival le majhen del funkcij, ki jih ponuja trenutno uporabljana različica programske rešitve, se je izkazal kot arhitekturno neustrezen. Razlogi so bili večplastni – delno vsebinski oziroma posledica spremenjenih zahtev (ki jih bom predstavil v naslednji točki – v okviru sprememb zahtev), delno pa so bile to omejitve platforme, vidne že v času testiranja.

Testiranje prototipa je bilo delno izvedeno z realnimi, delno pa s fiktivnimi podatki, ki so bili pred tem preneseni v bazo podatkov. Četudi je bila velikost baze podatkov še krepko znotraj omejitev platforme, je bil prvi vtis počasno delovanje aplikacije, in sicer že pod majhnimi obremenitvami. Odzivni časi izvajanja bolj kompleksnih poizvedb, ki so bile v ozadju nekaterih obrazcev uporabniškega vmesnika in ki bi se izvajale zelo pogosto, so bili nekajkrat nad mejo sprejemljivega – tudi do več sekund. Še posebno dolgotrajno je bilo izvajanje poizvedb, ko se je baza nahajala na drugem računalniku in je šlo za dostop do baze podatkov preko računalniškega omrežja. Testna različica baze podatkov je imela do 300.000 zapisov v posamezni tabeli in skupaj 29 tabel. Velikost celotne baze podatkov je bila 150 megabajtov. Za primerjavo naj omenim, da ima trenutno uporabljana različica aplikacije v bazi podatkov znotraj posameznih tabel do nekaj manj kot milijon zapisov, velikostno pa dosega nekaj manj kot 500 megabajtov. Do baze podatkov se skoraj konstantno dostopa z več lokacij hkrati, v konicah ima lahko precej več kot deset hkratnih uporabnikov (kot je približno maksimum pri

Accessu). Če bi nespremenjen prototip pripeljali do končne različice, bi bil pri teh obremenitvah in naboru funkcij verjetno povsem neuporaben (What are the limitations of MS Acces?, 2008).

5.2 SPREMEMBE IN DOPOLNITVE ZAHTEV

Tekom razvoja programske rešitve je pogosto prihajalo do sprememb in dopolnitev v zahtevah, glavni razlogi za to pa so bili:

- večje spremembe poslovanja, predvsem pri povezavah s poslovnimi partnerji;
- slabše poznavanje omejitev, ki so posledica zahtevanega načina delovanja programske rešitve;
- neusklajenost zaposlenih glede prioritet pri razvoju programske rešitve;
- konflikt interesov pri zaposlenih, povezan z želeno funkcionalnostjo.

Prvi paket sprememb v zahtevah je sledil testiranju prototipa. Delno je bil povezan s spremembami v poslovanju, delno posledica rezultatov prvih testiranj in delno tudi novih želja glede končne funkcionalnosti. Na prvem mestu sta bila hitrost in odzivnost aplikacije, ki naj bi bila precej višja od tiste, ki jo je ponujal prototip. Sledile so želje po omejevanju dostopa oziroma možnost različnih nivojev pooblastil uporabnika sistema ter precejšnja širitev podatkovnega modela pri entitetah stranka in pošiljka (v številu atributov in povezav). Ena od zahtev, ki je sledila, je bila možnost uporabe programske rešitve v nepovezanem delovanju in sinhronizacija z bazo podatkov na glavnem strežniku.

Preostale zahteve, ki so bile podane pred radikalnimi spremembami v delovanju programske rešitve, so na primer vključevale možnost podrobnejšega obračuna popustov in različne načine združevanja postavk, vključenih na račun. Naslednji paket zahtev je vključeval spremembe v izmenjavi podatkov, izključno nepovezano delovanje, možnost naročanja pošiljk preko spletnega vmesnika, dopolnitve pregledov, večjo raven nastavljivosti, različne vrste izpisov in poročil, fakturno knjigo ipd.

Posledice sprememb so bile precejšen odmik začetka uporabe novega informacijskega sistema in relativno veliki stroški sprememb.

5.3 TEŽAVE PRI UVAJANJU

Uvajanje nove programske rešitve ni potekalo brez težav, je pa morda zaradi izmenjujočih si faz razvoja in uvajanja nekoliko težje potegniti ločnico med koncem enega in začetkom drugega. Odpora na strani uporabnikov na srečo ni bilo, zaradi spreminjajočih zahtev pa je bilo delo s programom zanje mestoma precej stresno. Programska rešitev je bila po prvem

stiku z uporabnikom še večkrat spremenjena, tako v izgledu uporabniškega vmesnika, kot programski logiki oziroma v odzivih sistema, ki jih je bil uporabnik deležen ob delu z njim.

Poleg nekaterih hroščev, ki so se pokazali šele ob uporabi, so manjšo težavo predstavljala tudi stičišča uvedene rešitve z drugimi sistemi in posameznimi, v sistem vključenimi orodji. Eno od teh je uporaba mobilnega ročnega skenerja, ki je zahtevala posebne prilagoditve. Poleg fizičnih ovir je bilo v obtoku večje število različnih formatov zapisa v črtnih kodah in modul za prepoznavo le-teh je moral pokrivati vse. Manjše popravke je zahtevala tudi metoda branja podatkov preko na serijski vhod računalnika priklopljene tehtnice.

6 ANALIZA USPEŠNOSTI IN NADALJNJI RAZVOJ

Razvoj programske opreme vedno poteka v nekih vnaprej določenih okvirih. Od njihovih meja je v največji meri odvisno, kako dober izdelek bo na koncu pristal v rokah naročnika. Meje, ki so pri tem običajno navedene kot najbolj pomembne, so čas, stroški in kakovost – splet teh treh dejavnikov je največkrat omenjen kot projektni trikotnik (ang. »Project triangle«). Premik ene od mej pri tem ni mogoč brez premika neke druge meje. Dvig kvalitete obravnavane rešitve ob dani časovni omejitvi neizogibno pomeni dvig stroškov (npr. uporabo dražjih orodij, ki omogočajo večjo produktivnost na enoto vloženega človeškega dela). Lewis omenjenim trem komponentam dodaja še obseg oziroma velikost projekta, ki ga obenem označuje kot glavnega krivca za neuspešnost projektov (Lewis, 2000, str. 12).

Po podatkih, ki jih povzema Stepanek (2005, str. 4), je bila v analizi uspešnosti projektov, povezanih z izdelavo programske opreme, med vsemi zajetimi projekti le 28 % tistih, ki bi jih bilo mogoče označiti za uspešne. Ocena, ki pove, ali je nek projekt s tega področja uspešen ali ne, je lahko zelo subjektivna. Morda je še najbolj ustrezna kategorizacija, pri kateri upoštevamo le vidik naročnika programske opreme. Pri tem bi upoštevali naslednje kriterije (Why Software Projects Tend to Fail, 2008):

- izdelana programska oprema ne ustreza potrebam naročnika;
- zamujen je bil zadnji rok za predajo funkcionalne različice programske opreme;
- število hroščev v izdelani rešitvi je preveliko.

V naslednjih dveh poglavjih bom opisal, v kolikšni meri implementirana programska rešitev pokriva potrebe ACK, d.d. in ocenil, ali je dosežen rezultat možno vrednotiti kot uspeh. Izpostavil bom tudi pomanjkljivosti, ki so se pojavile v procesu načrtovanja ali izdelave. V drugi točki bom skupaj z omenjenim nakazal še smernice nadaljnjega razvoja informacijskega sistema ACK, d.d.

6.1 ANALIZA USPEŠNOSTI, STROŠKOV IN KORISTI

Največje odstopanje od načrta do uvedbe nove programske rešitve je bilo v časovni komponenti. Če bi le-to upošteval kot ključni kriterij pri presoji uspešnosti, bi projekt težko označil za uspešnega. Če izvzamem testiranje prototipa, je od začetka projekta do začetka dela uporabnikov z novim informacijskim sistemom preteklo slabo leto. Če pri tem upoštevam še nadaljnje spremembe oziroma dopolnitve v funkcionalnosti, se meja premakne še za nekaj mesecev višje. Po prvotnih načrtih naj bi bili za celoten proces potrebni približno štirje meseci, razlogov za precejšen odmik od tega pa je več. Poglavitni so naslednji:

- Zahteve, izražene ob načrtovanju informacijskega sistema, niso bile skladne s pričakovanji glede končne različice oziroma so bile pomanjkljivo predstavljene. Večkrat se je zgodilo, da je razlika, ki je zahtevo ločila od njene popolne oblike (s celotnimi specifikacijami), precej vplivala na poslovna pravila, zajeta v programu. Takšen primer je način izračuna cene pošiljke s sistemom cenikov in dejavnikov, ki jih je poleg tega še potrebno upoštevati. Ob nepopolnem opisu zahteve je prostora za interpretacijo veliko, zato je bil ta postopek večkrat spremenjen.
- Podane zahteve so se večkrat, tudi precej pozno v razvoju, drastično spremenile. Tudi število zahtev se je precej povečalo. Tako je bila ena od dodatnih zahtev že prej omenjen nepovezan način delovanja, ki je bil v veliki meri preslikava nekaterih lastnosti prototipa na novo platformo.
- Posledica zgornjih dveh točk je bilo precejšnje povečanje obsega projekta. Skladno s tem je implementacija terjala tudi precej večje število ur potrebnega dela, kar pa je bilo ob zelo omejenih virih težko dosegljivo.

Če pri oceni uspešnosti upoštevam zgoraj omenjeno in časovni zamik kot neizogibno posledico, bi projekt razvoja in vpeljave novega informacijskega sistema v ACK, d.d. označil za uspešnega. Najmočnejši argument za to so doseženi zastavljeni cilji, predvsem precejšen dvig v informacijski pokritosti obravnavanega dela poslovanja podjetja.

Bistveno vprašanje pri vsaki investiciji, vključno z investicijami v informatiko, je: Ali je naložba ekonomsko upravičena? Problema, ki se v povezavi z informatiko pojavita pri odgovoru na to vprašanje, sta dva. Prvi je v tem, da se v večini primerov učinki naložbe v informatiko ne odrazijo neposredno v višjem dobičku, ki ga ustvari podjetje s svojim delovanjem na trgu. Drug problem pa je povezan z negotovostjo oziroma z dejstvom, da mora odgovor na omenjeno vprašanje vsebovati znatno mero napovedovanja prihodnosti (Turk, 2005, str. 156).

Ker z informacijami o ozadju izbire informacijske rešitve pred začetkom razvoja vpeljanega informacijskega sistema ne razpolagam, lahko z razpoložljivimi obravnavam stroške in koristi programske rešitve le glede na nespremenjeno stanje. Večino koristi uvedenega sistema sem omenil že pri analizi posameznih procesov, stroški pa so zaradi precejšnje uporabe notranjih virov zelo težko ocenljivi. Strošek, ki je najmanj sporen, saj ni pod vplivom ocen, je cena

programske rešitve. Končna cena, ki vključuje proces od začetka razvoja do namestitve na strežnik in delovne postaje, pripadajoče svetovanje in odpravo napak, je znašala 3,3 mio SIT. Stroškov spremljajoče programske opreme (orodja za delo z bazo ipd.) zaradi baziranja na odprtokodnih rešitvah ni bilo. Vse prej kot zanemarljiv je bil tudi strošek zakasnitve pri uporabi rešitve in strošek, nastal zaradi nepopolnega pokrivanja dela kakega procesa. Žal gre pri tem za področje, ki je povsem odvisno od ocen, pa tudi precej v domeni notranjih informacij. Težko bi bilo tudi izločiti elemente procesov pred prenovo, saj se je marsikateri pri tem pojavil v spremenjeni obliki (problem relativnosti).

6.2 NADALJNJI RAZVOJ IN POMANJKLJIVOSTI

Uveden informacijski sistem običajno pomeni tudi zavezo stalnemu usklajevanju s spremembami v poslovnem procesu. Pomanjkljivosti programske rešitve, vpeljane v ACK, d.d., tako že implicitno narekujejo nadaljnji razvoj.

Pomanjkljivosti v uvedenem sistemu bi glede na razloge zanje lahko razdelil v dve skupini. V prvo bi uvrstil pomanjkljivosti, za katere je bilo že na začetku znano, da jih ob danih omejitvah glede na uveljavljeno prakso oziroma trenutno stanje ne bo možno odpraviti oziroma da se bodo zaradi določenih predpostavk pojavile. Drugi skupini pa bi pripisal tiste, ki so nastale kot posledica sprememb v procesu razvoja oziroma kompromisov, ki so bili pri tem sprejeti zaradi doseganja nekega drugega, prav tako pomembnega cilja. Ne glede na razlog v ozadju pa je pomanjkljivosti glede na resnost oziroma prioriteto pri njihovi sanaciji možno enotno razvrstiti. V nadaljevanju jih bom razdelil na štiri segmente in jih podrobneje opisal. Ti segmenti obenem pomenijo tudi oris nadaljnjega razvoja.

V prvi, najpomembnejši sklop pomanjkljivosti sodijo različni povezovalni moduli. Najpomembnejši del teh so moduli za povezavo s poslovnimi partnerji. Zaradi precejšnjih razlik med informacijskimi sistemi poslovnih partnerjev in različnimi formati, ki jih slednji uporabljajo pri izmenjavi podatkov, neposredna povezljivost brez večjih prilagoditev ni mogoča. Težavi pri tem sta dve. Prva je vsebinske narave in se nanaša na količino in strukturo podatkov, povezanih s posamezno enoto. V primeru ACK, d.d. gre za entitete stranka, pošiljka, paket, dogodek s pripadajočimi atributi, na drugi strani poslovnega partnerja pa je ta hierarhija ali količina pripetih podatkov lahko drugačna. Zato je za povezljivost potrebna vmesna faza, ki eno stran ob določenih pravilih pretvori v drugo. Drug problem naj bi načeloma v precejšnji meri reševali standardi, a so ti v praksi, sploh pri starejših implementacijah, redkeje upoštevani. Tako je rokovanje z datotekami s polji fiksne dolžine ali variabilne dolžine z ločili še vedno povsem običajno. Informacijski sistem ACK, d.d. ima predvsem zaradi časovnih omejitev trenutno le delno dokončan modul za povezljivost.

V drugi sklop spadajo nekateri pomembni, a informacijsko slabše pokriti deli poslovanja. Pri tem je na prvem mestu členjeno obračunavanje storitev. Bistvo le-tega je možnost spremljanja

postavke na računu v načinu »od zgoraj navzdol«. Na primeru pošiljke kot postavke, zaračunane naročniku, to pomeni, da so s slednjo povezane tudi pripadajoče storitve, kot so na primer zavarovanje, podatek o vrnjeni dobavnici in drugo. Omenjeno pa ni vodeno kot posamezna neodvisna storitev na računu, ampak logično povezano s pošiljko, saj nudi potrebne informacije pri kalkulaciji stroškov. Naslednji del tega sklopa je povezan s prejšnjim, a se v nasprotju z njim (vezan na naročnika) nanaša na podizvajalce oziroma partnerje na poti pošiljke do cilja, vloga tega dela pa je enaka prejšnji. V ta sklop spadajo še nekateri obračuni, ki sedaj trpijo na račun ročnega dela oziroma slabše transparentnosti, ter poročila in analize.

Tretji sklop pripada varnosti in personalizaciji uporabniškega vmesnika na spletu. Čeprav trenutna programska rešitev omogoča nastavljanje uporabniških pravic oziroma pooblastil, je to relativno težavno ali vsaj nepregledno, obenem pa tudi ni vezano na posamezne aktivnosti, ampak na celotne skupine podatkov. Naslednja pomanjkljivost, povezana s tem, je nezmožnost sledenja aktivnostim uporabnika. Trenutni obseg teh informacij namreč obsega zgolj časovne žige spremembe in vnosa posameznega zapisa. Če izvzamemo strežniške dnevnike SUBP, pri katerih so stroški razveljavitve sprememb relativno visoki, ob napaki uporabnika povratek v predhodno stanje ni mogoč. Del informacijskega sistema, ki prav tako spada v ta sklop, je spletni uporabniški vmesnik, ki je zaenkrat voden na ravni posameznega podjetja. To je zaradi večjega števila enot podjetja pod istim okriljem problematično iz dveh razlogov – prvi so naročila pošiljk, drugi pa izdaja računov.

V zadnji sklop spadajo vse ostale pomanjkljivosti, ki bi v razumnem času morale biti odpravljene, a imajo nekoliko nižjo prioriteto. Mednje spada optimizacija nekaterih delov sistema, ki so nekoliko slabše odzivni. Precejšen del teh je neposredno povezan s slabšo izvedbo kompleksnih poizvedb s strani MySQL-a. Rešitev za to bi precej verjetno moral biti drug SUBP. Kot sledi iz poglavja o izbiri SUBP, bi bil najboljši prosto dostopni kandidat PostgreSQL. Zaradi dveh migracij (iz Access-a in starejše verzije programske platforme) slabša kvaliteta programske kode je naslednja pomanjkljivost, ki ima za posledico višje stroške vzdrževanja in bi morala biti odpravljena. Nenazadnje pa bo nekaj izboljšav potrebnih tudi na uporabniškem vmesniku, ki je sicer relativno enostaven za uporabo, a so nekatere operacije v njem manj priročno dostopne. K temu spadajo na primer obrazci za hitrejši ročni vnos večjih količin pošiljk v tabelaričnem formatu (podobno programom za delo s preglednicami).

7 SKLEP

Konkurenčni pritiski na podjetje so v informacijski dobi veliko večji kot so bili včasih, kar se kaže tudi na področju logistike in poštnih storitev. Odzivni časi na zahteve strank in na spremembe na trgu, ki se danes od podjetja pričakujejo, so precej krajši, slabo razvit informacijski sistem pa temu večkrat ni kos. Ker je razvoj informacijskega sistema pogosto zelo zahtevna naloga, je pomembno, da ji v podjetju posvečamo dovolj pozornosti. Napake v zgodnjih fazah razvoja bomo drago plačevali proti koncu le-tega. Investicija v informatiko mora biti tako kot vsaka druga investicija ekonomsko upravičena. Če razvit informacijski sistem ne dosega zastavljenih ciljev, lahko tudi koristi, ki jih bomo od njega deležni, ne bodo zagotavljale ustreznega donosa.

Trg programske opreme je zadnja leta priča precejšnjim spremembam. Relativno nova sila na njem, odprtokodna programska oprema, se je sprva začela kot ideologija in do danes že izoblikovala v velik, pogosto povsem neizkoriščen potencial. Dobro informirano podjetje ga lahko izkoristi in z njim doseže precejšnje prihranke pri investicijah v informatiko. To še posebej velja za velike organizacije, pri katerih lahko stroški licenc zaradi velikega števila namestitev predstavljajo velik del celotnih investicij v informatiko. Poleg cenovne komponente, ki ima na podjetja neposreden učinek, je morda še pomembnejši del sporočila ideologije tisti, ki se nanaša na število pravic, ki se jim moramo pri običajni zaprtokodni licenci odreči. Kot posledico tega bi izpostavil predvsem nevarnost nezdružljivosti kupljene nove programske opreme z ostalo, ki jo uporabljamo v podjetju, in s tisto, ki jo uporabljajo naši partnerji in stranke. Pri tem mislim predvsem na težave zaradi nedostopnosti specifikacij za datoteke, protokole in ostale načine izmenjave podatkov. Čeprav se pri slednjem stanje zadnja leta nekoliko izboljšuje, nezdružljivost za podjetje lahko še vedno pomeni precej dodatnih stroškov.

V diplomskem delu sem predstavil, kako je potekal razvoj in zatem tudi vpeljava novega informacijskega sistema v podjetju ACK, d.d.. Čeprav se projekt ni odvijal brez zapletov, bi ga označil za uspešnega. Največje odstopanje od zastavljenih ciljev je bil precejšen odmik dejanskega začetka uporabe novega informacijskega sistema od prvotno načrtovanega. Razlogov, ki so pripeljali do tega, je bilo več. Na prvo mesto bi postavil slabo definirane zahteve in njihovo pogosto spreminjanje. Slednje je bilo v precejšnji meri krivda neusklajenih zahtev in prioritet znotraj podjetja, prišlo pa je tudi do konflikta interesov v zvezi z želeno funkcionalnostjo. Drugi razlog je precejšnje povečanje obsega projekta. Ta se je povečal tako v smislu področij, ki naj bi jih programska rešitev pokrivala, kot v smislu nabora funkcij znotraj predhodno definiranih. Tretji razlog je posledica prvih dveh in omejitev na strani virov. Količina vloženih ur v razvoj programske rešitve je zaradi omenjenega precej presegla prvotno zastavljene okvire in odmik dejanskega začetka uporabe je bil ob danih virih neizogiben. Kljub omenjenemu pa vpeljana programska rešitev dobro pokriva v razvoju upoštevane potrebe in nudi dobro podlago za nadaljnji razvoj.

LITERATURA

1. Bajec Marko: Osnove informacijskih sistemov. Ljubljana : Fakulteta za računalništvo in informatiko, 2004. 545 str.
2. Bajec Marko, Krisper Marjan, Rupnik Rok: Using business rules technologies to bridge the gap between business and business applications. Peking : Publishing House of Electronics Industry, 2000, str. 77-84.
3. Celko Joe: Joe Celko's Data And Databases - Concepts In Practice. San Francisco : Morgan Kaufmann, 1998. 400 str.
4. Davis S. William: Information System Consultants Handbook Systems Analysis And Design. Florida : Crc Press, 1999. 765 str.
5. Gradišar Miro, Resinovič Gortan: Informatika v poslovnem okolju. Ljubljana : Ekonomska fakulteta, 2001. 508 str.
6. Grošić Sanela: Uvajanje standarda kakovosti v špedicijsko dejavnost. Diplomsko delo. Koper : Fakulteta za management Koper, 2006. 51 str.
7. Hočevar Jožko: Podatkovni model proizvodnega podjetja. Diplomsko delo. Ljubljana : Ekonomska fakulteta, 2005. 46 str.
8. Jaklič Jurij: Upravljanje in uporaba podatkov. Ljubljana : Ekonomska fakulteta, 2002. 213 str.
9. Kovačič Andrej: Informatizacija poslovanja. Ljubljana : Ekonomska fakulteta, 1998. 214 str.
10. Lewis P. James: Project Planning, Scheduling & Control. 3rd Edition. New York : McGraw Hill, 2000. 550 str.
11. Lhotka Rockford: Expert C# Business Objects. 2nd Edition. Berkeley : Apress, 2006. 696 str.
12. Popovič Aleš et al.: Poslovno modeliranje v teoriji in praksi: izkušnje in napotki. Uporabna informatika, Ljubljana, 12(2004), 2, str. 80-89.
13. Ramakrishnan Raghu, Gehrke Johannes: Database Management Systems. 2nd Edition. New York : McGraw Hill, 1998. 906 str.
14. Sajko Uroš: Razvoj sistemov tipa odjemalec/strežnik z orodjem powerbuilder. Diplomsko delo. Maribor : Fakulteta za elektrotehniko, računalništvo in informatiko, 1997. 80 str.
15. Siau Keng: Contemporary Issues in Database Design and Information Systems Development. London : IGI Publishing, 2007. 300 str.
16. Stepanek George: Project Secrets: Why Software Projects Fail. Berkeley : Apress, 2005. 192 str.
17. Sumathi Sai, Esakkirajan S.: Fundamentals of Relational Database Management Systems. New York : Springer Verlag, 2007. 776 str.
18. Turk Tomaž: Analiza stroškov in koristi naložb v informatiko. Uporabna informatika, Ljubljana, 13(2005), 3, str. 153-169.

19. Walnes Joseph et al.: Java Open Source Programming: with XDoclet, JUnit, WebWork, Hibernate. Indianapolis : Wiley, 2003. 480 str.
20. Yeates Donald, Wakefield Tony: Systems Analysis and Design. 2nd Edition. New York : Prentice Hall, 2004. 520 str.

VIRI

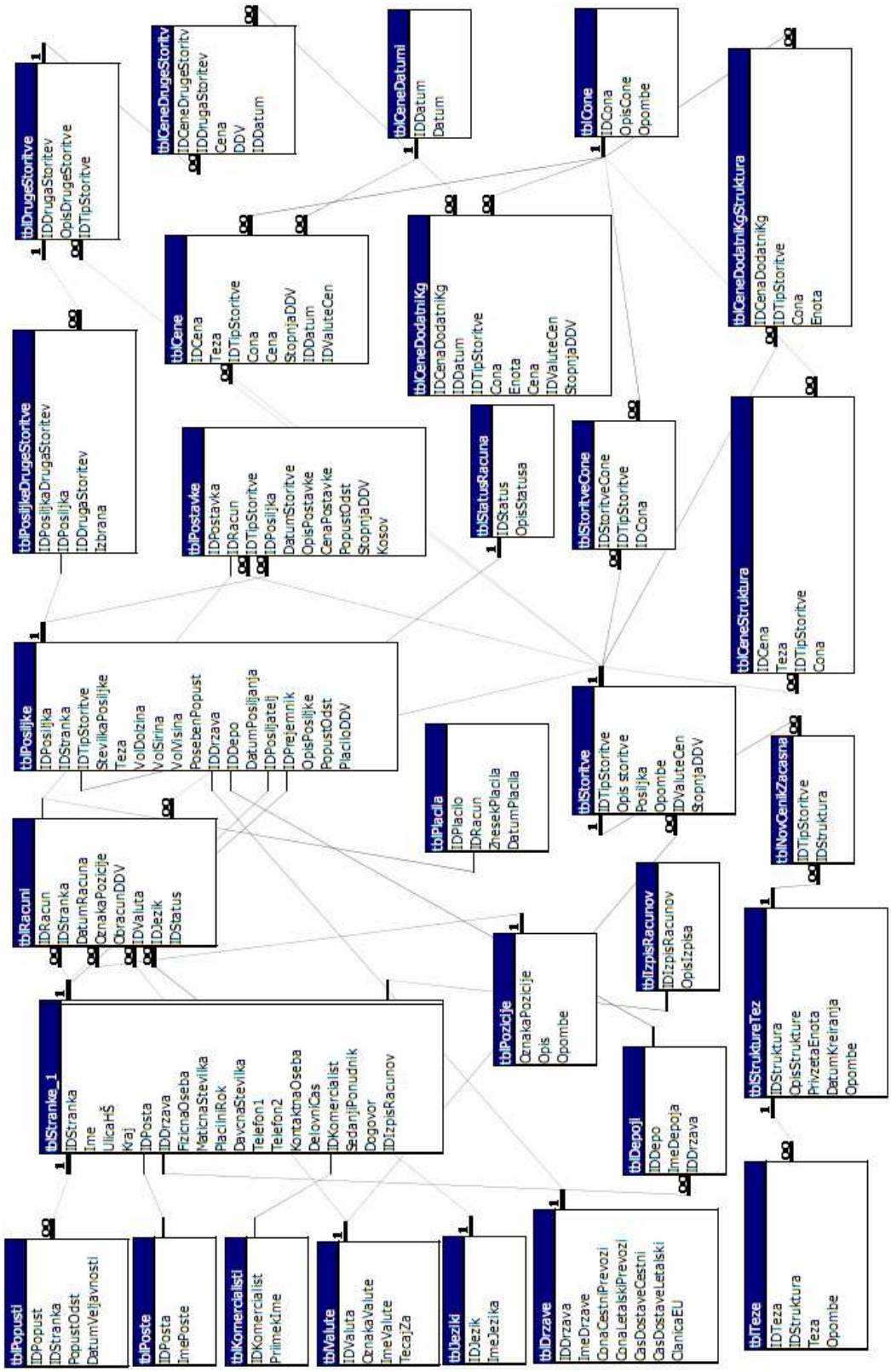
1. All SPEC jAppServer2004 Results: SPEC. [URL: http://www.spec.org/jAppServer2004/results/jAppServer2004.html/], 14.02.2008.
2. Analiza trga poštnih storitev v Republiki Sloveniji v l. 2006. Agencija za pošto in elektronske komunikacije Republike Slovenije. [URL: http://www.apek.si/sl/analiza_trga_postnih_storitev_v_rs_v_letu_2006/], junij 2007.
3. Comparison of different SQL implementations. [URL: http://troels.arvin.dk/db/rdbms/], 14.02.2008.
4. Coopworks - Total cost of ownership (TCO). [URL: http://www.coopworks.org/index2.php?option=com_content&task=view&id=45&pop=1&page=0&Itemid=39/], 14.02.2008.
5. DB2 Express-C: IBM. [URL: http://www-306.ibm.com/software/data/db2/express/], 14.02.2008.
6. Industry Pushes Back On Oracle's Multicore Licensing Policies: CMP Channel. [URL: http://www.crn.com/hardware/175003555/], 12.12.2007.
7. Letno poročilo družbe ACK d.d. za leto 2006.
8. Lhotka Rockford - Should all apps be n-tier?. [URL: http://www.lhotka.net/weblog/ShouldAllAppsBeNtier.aspx/], 14.02.2008.
9. Linux Gazette. [URL: http://linuxgazette.net/issue75/nielsen.html/], februar 2002.
10. Microsoft Access within an Organization's Database Strategy. [URL: http://www.fmsinc.com/tpapers/genaccess/DBOD.asp/], 14.02.2008.
11. Open Source Initiative. [URL: http://www.opensource.org/docs/osd/], 12.12.2007.
12. Oracle Announces the Acquisition of Open Source Software Company, Innobase: Oracle Press Release. [URL: http://www.oracle.com/corporate/press/2005_oct/inno.html/], 14.02.2008.
13. Oracle Database Lite 10g: Oracle. [URL: http://www.oracle.com/technology/products/lite/index.html/], 14.02.2008.
14. Oracle, MySQL and PostgreSQL feature comparison. [URL: http://www.databasejournal.com/features/oracle/article.php/3692566/], 14.02.2008.
15. Oraclov sporazum o licencah in storitvah. [URL: http://www.oracle.com/corporate/license/OLSA-sl-v111003.pdf/], 14.02.2008.
16. Spletna stran družbe ACK, d.d. [URL: http://www.ack.si/zgodovina.html], 22.11.2007.
17. Spletna stran MySQL AB. [URL: http://www.mysql.com/], 12.12.2007.
18. Spletna stran PostgreSQL. [URL: http://www.postgresql.org/], 14.02.2008.

19. SQL Server 2005 Express Edition: Microsoft. [URL: http://www.microsoft.com/sql/editions/express/default.mspx/], 14.02.2008.
20. "We Don't Need the GPL Anymore": ESR. [URL: http://www.onlamp.com/lpt/a/5998/], 22.11.2007.
21. What are the limitations of MS Access?. [URL: http://databases.aspfaq.com/database/what-are-the-limitations-of-ms-access.html/], 14.02.2008.
22. Why Software Projects Tend to Fail. [URL: http://www.codeproject.com/KB/work/WhySoftwareProjectsFail.aspx/], 28.02.2008.

PRILOGE

PRILOGA 1: Diagram modela podatkovne baze prototipa

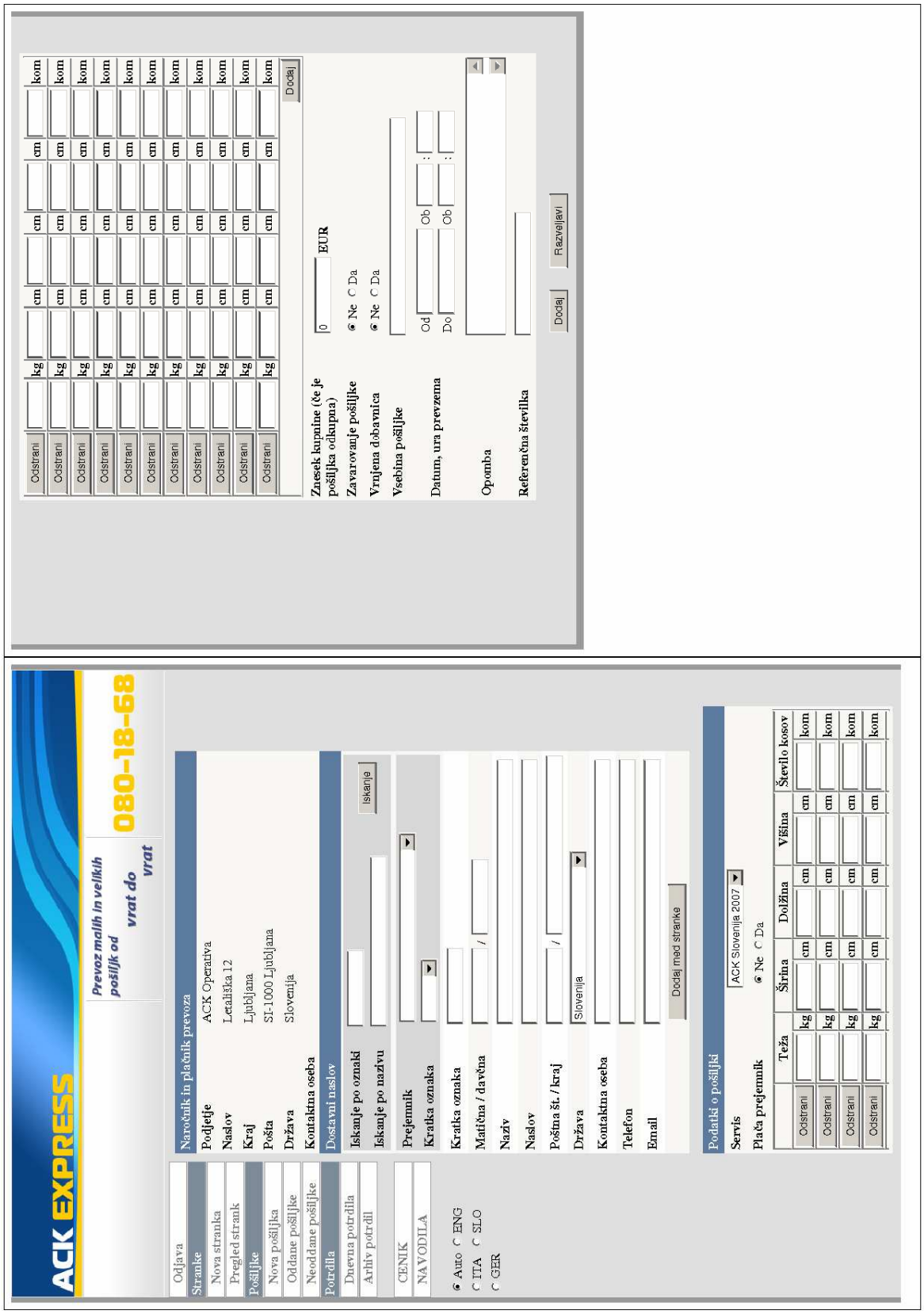
Slika 1: Diagram modela podatkovne baze prototipa



Vir: Lasten prikaz.

PRILOGA 2: Spletni vmesnik za naročanje pošiljk

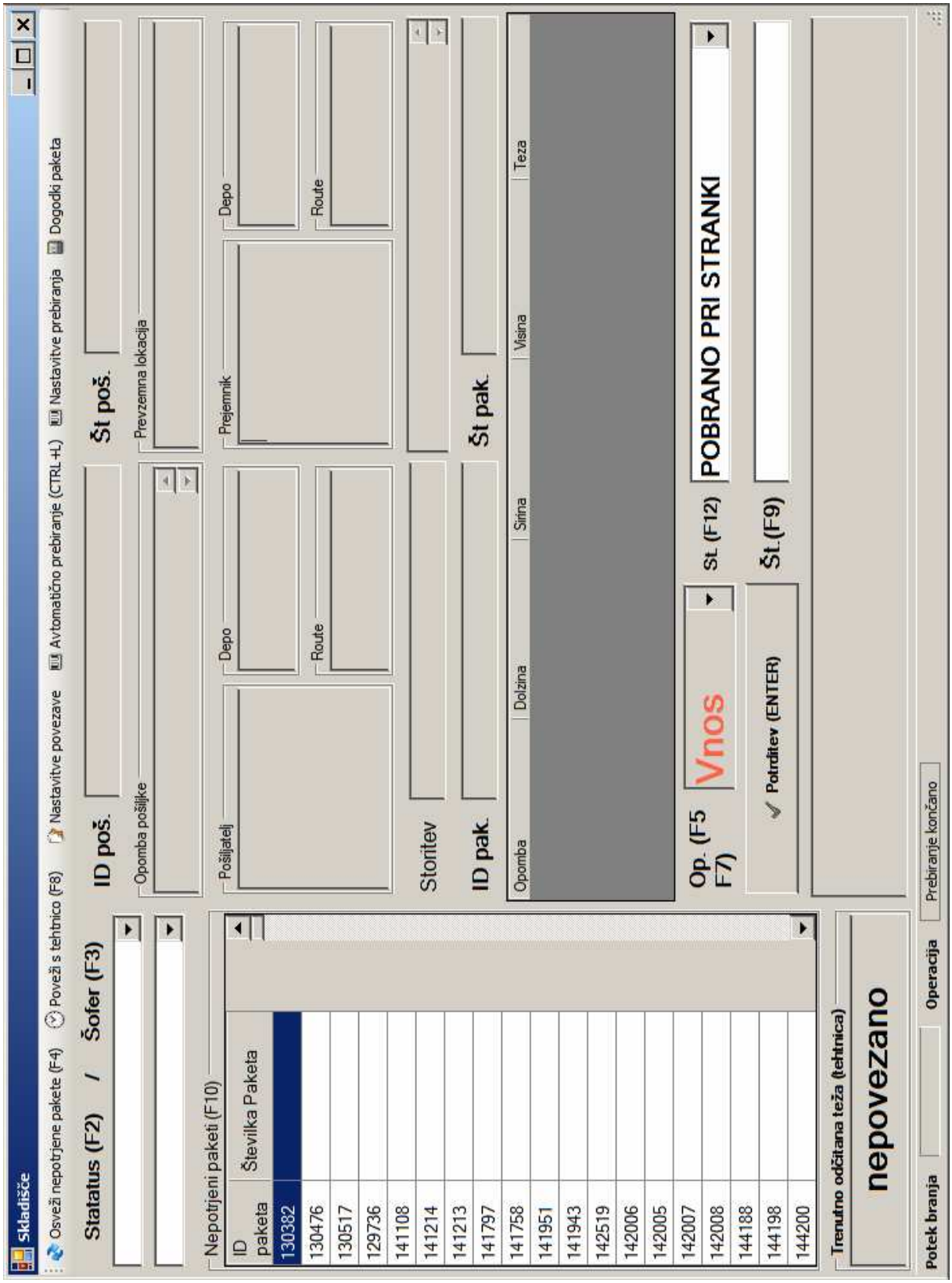
Slika 2: Spletni vmesnik za naročanje pošiljk



Vir: Lasten prikaz.

PRILOGA 3: Uporabniški vmesnik modula skladišče

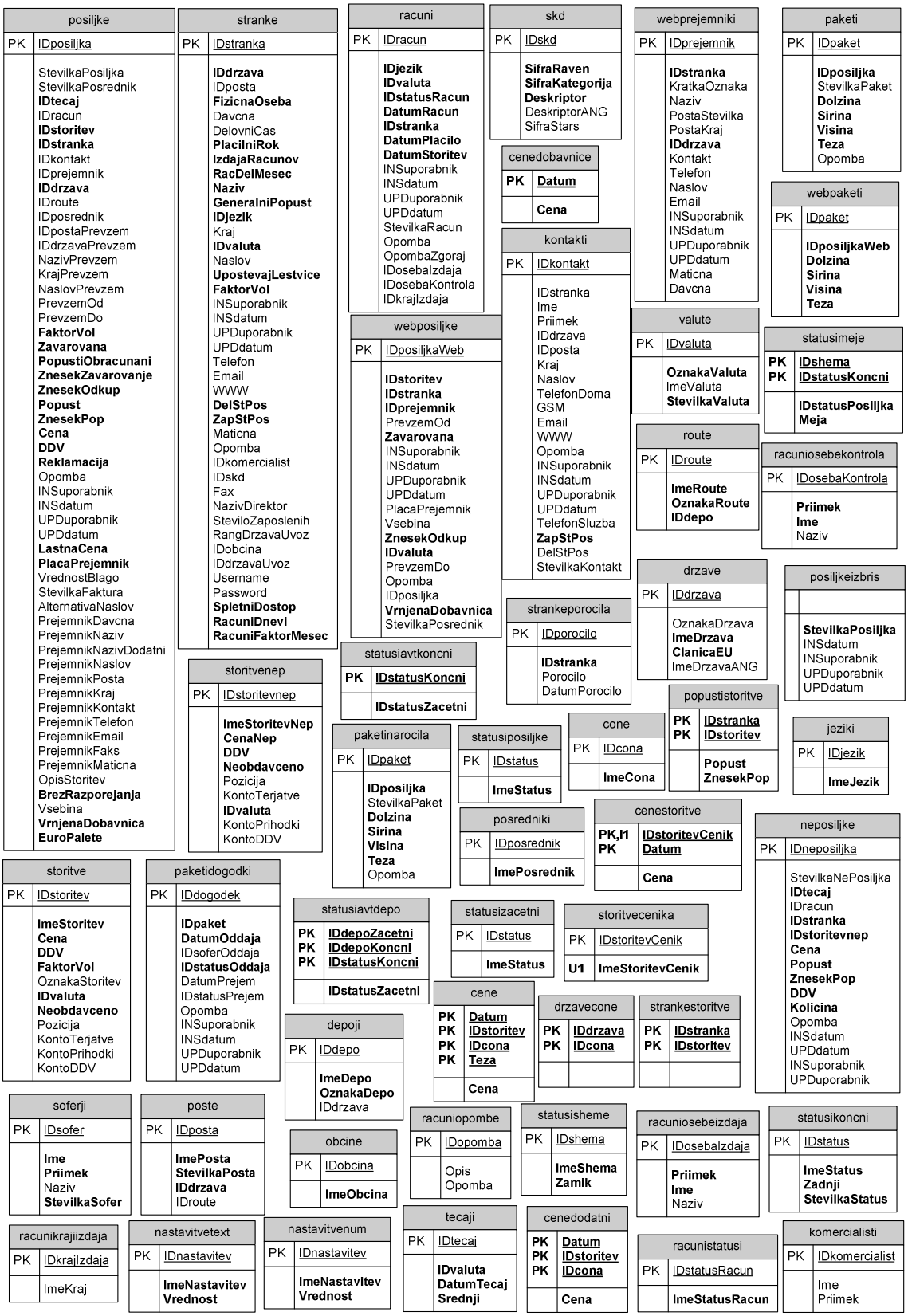
Slika 3: Uporabniški vmesnik modula skladišče



Vir: Lasten prikaz.

PRILOGA 4: Diagram modela podatkovne baze novega sistema

Slika 4: Diagram modela podatkovne baze novega sistema



Vir: Lasten prikaz.

PRILOGA 5: Rezultati testa SPEC jAppServer2004

Tabela 1: Rezultati testa SPEC jAppServer2004

Izvajalec testa	Naziv sistema	Točke JOPS	J2EE strežnik št. vozlišč	Procesorji	J2EE strežnik št. instanc	SUBP strežnik št. instanc
HP	Oracle Application Server 10g Release 10.1.3.2 - Java Edition on HP-UX rx2660 cluster	1578.16	2	8 jeder / 4 čipi / 2 jedri na čip	4	1
HP	Oracle Application Server 10g Release 10.1.3.2 - Java Edition on HP-UX rx2660	874.17	1	4 jedra / 2 čipa / 2 jedri na čip (z HT)	2	1
Sun Microsystems Inc.	Sun Java (TM) Systems Application Server 9.0 Platform Edition with Postgres 8.2	778.14	3	12 jeder / 6 čipov / 2 jedri na čip	3	1
Sun Microsystems Inc.	Sun Java (TM) Systems Application Server 9.0 Platform Edition on SunFire X4100 Cluster with MySQL 5	720.56	3	12 jeder / 6 čipov / 2 jedri na čip	3	1

Vir: All SPEC jAppServer2004 Results, 2008.