UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

DIPLOMSKO DELO

RAZVOJ PROGRAMSKE REŠITVE
(IZDAJA PONUDB ZA PRODUKTE SREDNJE NAPETOSTI)

Ljubljana, december 2004 MARTIN PODGORNIK

IZJAVA

Študent Martin Podgornik izjavljam, da sem avtor tega diplomskega dela, ki sem ga napisal pod mentorstvom dr. Tomaža Turka in dovolim objavo diplomskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne 6.12.2004 Podpis: _____

KAZALO

1 UVOD

Količina podatkov in informacij, ki jih mora določena organizacija predelati in shraniti za svoje uspešno poslovanje, se je v preteklosti povečala in tak trend je še vedno opazen. V spletu obstaja relativno veliko koristnih programskih rešitev, ki omogočajo urejanje na različne načine pridobljenih podatkov in združevanje v koristne informacije in izpise. Vendar pa ima vsaka organizacija svoje posebnosti, ki se bolj ali manj razlikujejo od ostalih organizacij in je težko, če ne že kar nemogoče, dobiti neko splošno programsko rešitev, ki bi ustrezala tudi specifični problematiki. Tema moje diplomske naloge se nanaša na razvoj programske rešitve v obravnavanem podjetju. Glavni izhod programske rešitev je izpis ponudbe, ki se generira na podlagi številnih dreves odločanja in drugih postopkov. Obenem pa omogoča fleksibilno rabo v pisarni kot tudi na terenu.

Podjetje[1], za katerega sem razvil v diplomski nalogi opisano programsko rešitev, je v lasti multinaconalne družbe in je v Sloveniji prisotno od leta 1992, ko je avstrijska podružnica multinacionalke odprla pisarno v Ljubljani. Leta 1994 je bila ustanovljena slovenska podružnica, šest let kasneje (leta 2000) pa je bila ustanovljena družba z omejeno odgovornostjo, kakršno ime in status ima še danes. Na področju Slovenije skrbi za organizacijo prodaje, svojim strankam in partnerjem pa nudi komercialno, tehnično in marketinško podporo. Nastopa na 4 tržnih področjih: energetiki, industriji, na področju inštalacij v zgradbah in v infrastrukturi. Izdelki so namenjeni proizvodnji, prenosu in razdelitvi električne energije od generatorja pa vse do končnih uporabnikov v industriji, infrastrukturnih objektih ter v stanovanjskih zgradbah, opremi za distribucijo električne energije in avtomatizacijo industrijskih procesov. Prisoten je predvsem na severno ameriškem tržišču, kjer ima na nekaterih tržnih področjih največji tržni delež. Proizvaja električno opremo, namenjeno predvsem uporabi v industriji. Kontaktorji in releji, tipkala, končna in tlačna stikala, senzorji in naprave za varno obratovanje, PLC-ji ter predizdelani zbiralnični sistemi so izdelki, s katerimi lahko podjetje ponudi celostne rešitve za večino problemov industrijskega vodenja in avtomatizacije.

Diplomskom delo je razdeljeno na šest poglavji, ki opisujejo posamezne faze razvoja programske rešitve. Drugo poglavje opisuje štiri metode dela, ki jih lahko uporabimo pri razvoju programske rešitve. Na podlagi izbora in preučitve metod sem izbral model prototipa, ki najbolj ustreza okolju v podjetju, velikosti in strukturiranosti problema.

Naslednji dve poglavji opisujeta načrtovanje projekta in analizo, ki sta izjemno pomembna postopka, saj tu določimo odgovornosti, finančni okvir in terminski načrt projekta. V fazi analize se med drugim preučuje obstoječi in željeni sistem v organizaciji. Informacije se

[1] Podjetje se ni strinjalo z objavo njenega imena v diplomski nalogi, saj menijo, da bi razkritje postopkov izdelave ponudbe lahko škodilo njenemu poslovanju.

pridobijo na različne načine. Najpogosteje pa na različnih sestankih, interni dokumentaciji in v mojem primeru, v iteracijah.
Število strani posameznega poglavja in podpoglavja diplomske naloge ne ponazarja količino vloženega dela v posamezen del projekta. Daleč največ truda in energije je bilo vloženo v razvoj programske rešitve same. V iteracijah je prišlo do mnogih idej in dodatnih zahtev, saj uporabnik pogosto šele ob stiku s produktom ugotovi, kaj tehnologija omogoča in kako mu lahko olajša delo.

Izbrana metoda razvoja programkse rešitve omogoča spremembe v zasnovi programske rešitve. Da bi zagotovil celovit pogled nad spremembami, ki so nastale v procesu iteracij, sem v šestem poglavju opisal celotne programsko rešitev.

2 IZBIRA METODE RAZVOJA PROGRAMSKE REŠITVE

Namen tega poglavja je predstaviti različne metode razvoja in utemeljiti izbor tiste metode, ki projektu najbolj ustreza. V preteklosti je bilo razvitih relativno veliko različnih metod razvoja. Ker ni mogoče niti smiselno opisovati vseh, se bom v sledečem poglavju osredotočil na štiri različne metode dela (metoda klasičnega razvojnega cikla, metoda prototipa, pospešen razvoj programske rešitve in agilne metode). Koristi uporabe predpisanih in že definiranih metod razvoja so številne. Večinoma jih delimo na kratkoročne, ki se kažejo neposredno pri izvajanju projektov, in na dolgoročne, ki se zrcalijo predvsem v povečanju organizacije in v izboljšanju stopnje zadovoljstva naročnikov, to pa neposredno vpliva na uspešnost in dolgoročni obstoj organizacije (Hvala, 2004, str. 12).

Kratkoročne koristi so:

- krajši cikli in manjši stroški;
- bolj realni načrti, ki zagotavljajo večjo verjetnost, da bodo predvideni časi tudi doseženi;
- boljša komunikacija o tem "kaj" se pričakuje od projekta in "kdaj".

Dolgoročne koristi so:

- boljši nadzor nad obsegom in s tem hitrejše dokončevanje projektov;
- manjše tveganje;
- izboljšano upravljanje tveganj, ki vodi k boljšim odločitvam;
- povečano zadovoljstvo in zaupanje strank, ki vodita k povečanju obsega poslovanja - stranke začno izvajalce obravnati kot partnerje;
- v organizaciji se poglavitni poudarek prenese z medsebojnega tekmovanja funkcionalnih skupin na povečanje zadovoljstva strank;
- merjenje učinkovitosti (benchmarking) in njeno stalno izboljšavanje postaneta lažja in hitrejša.

2.1 KLASIČNI RAZVOJNI CIKEL

Klasični, linearni ali kaskadni življenjski cikel je najstarejši in še vedno pogost postopek razvoja programske opreme. Zgleduje se po standardnem postopku razvoja tehničnih izdelkov. Vse aktivnosti si sledijo zaporedno. Ker se v klasičnem razvojnem modelu ni možno vračati na predhodne faze, je nujno, da so zahteve že na samem začetku celovito in nedvoumno definirane. To zahteva skrbno analizo in sodelovanje naročnika ali uporabnika. Da ne bi med razvojem prišlo do prevelikega odstopanja med sistemom, ki ga razvijamo in zahtevami uporabnikov, je po vsaki fazi poleg verifikacije rezultatov smiselna tudi njihova validacija. Edino tak način razvoja je bil v preteklosti ekonomičen, saj so bile vse aktivnosti od načrtovanja do kodiranja in testiranja zaradi še neobstoječih orodij za podporo razvoja programske opreme zelo zamudne (Solina, 1997, str. 14).

2.1.1 FAZE KLASIČNEGA RAZVOJNEGA CIKLA

V osnovnem modelu si faze sledijo zaporedno in brez prehajanja na predhodne faze. V prirejenih metodah pa je dovoljeno vračanje na predhodne faze, v kolikor se oceni, da za izpeljavo določeno faze ni dovolj informacij.

2.1.1.1 Opredelitev zahtev

Programska oprema je del poslovnega sistema in zato se delo prične z ugotavljanjem potreb pri izvajanju poslovnih procesov in naknadno olajšanjem izvajanja določenih del z ustrezno programsko opremo. Tak pogled je bistven, ker se mora programska oprema prilagajati procesom ter strojni opremi, uporabnikom in drugim resursom. Poslovni proces je ključen za obstoj programske opreme. Če poslovni procesi niso definirani, jih je potrebno najprej opredeliti. Razvojne ekipe nato analizirajo poslovni proces in ugotovijo potrebe po programski opremi. Razvojna ekipa obišče stranko in analizira njen sistem. Raziščejo se potrebe po avtomatizaciji poslovnih procesov. Po koncu študije ekipa predstavi dokument, ki vsebuje različna priporočila za dotični sistem. V priporočilu so vključeni tudi stroški projekta, terminski načrt in zadolžitve zaposlenih. Zbiranje potreb je še posebej osredotočeno na programsko opremo. Za razumevanje narave programov, ki naj bi se izdelali, mora sistemski analitik razumeti informacijsko domeno programske opreme, kakor tudi potrebne funkcije in njeno delovanje. Namen te faze je ugotoviti potrebe in definirati problem.

2.1.1.2 Analiza

Cilj analize je celostno razumevanje nalog in lastnosti programske opreme. Vse zahteve je potrebno skrbno dokumentirati in se o njih pogovoriti z naročnikom.

Analizo lahko opredelimo na dva sklopa nalog, ki jih moramo opraviti. V prvem sklopu nalog se je potrebno pobliže spoznati s področjem organizacije, ki ga želimo modelirati.

Razumeti moramo cilje organizacije na tem področju in strategije, ki jih organizacija uporablja za dosego teh ciljev. Drug sklop nalog obsega:

- identificiranje uporabnikov in področja uporabe podatkov;
- pregled obstoječe dokumentacije;
- intervjuje z uporabniki.

Najpomembnejša vprašanja, na katera moramo poiskati odgovore v tej fazi, so (Grad, 1996, str. 227):

- Kateri so podatki, ki jih uporabniki potrebujejo?
- Kakšen je pomen teh podatkov?
- Kakšen je tok podatkov?
- Kdo in kako pogosto uporablja posamezne podatke?
- Kaj so vhodi in izhodi transakcije?
- Ali so kakšne dodatne zahteve?

2.1.1.3 Načrtovanje

Načrtovanje programske opreme je večstopenjski proces, ki zahteve s pomočjo uporabe podatkovnih in kontrolnih struktur ter programske arhitekture prevede v načrt programske opreme. Načrt je taka predstavitev programske opreme, da njene lastnosti lahko ocenimo še preden se začne kodiranje. V tej fazi je definirana struktura programske opreme. Pri strežnik/klient tehnologiji je določeno število klientov, pri načrtovanju baz podatkov pa je na primer opredeljen ER-model.

2.1.1.4 Kodiranje

Načrt mora biti preveden v obliko, ki jo strojna oprema razume. Če je bilo načrtovanje podrobno izvedeno, je generacija kode lahko zelo preprosto izvedena. Programerska orodja kot so prevajalniki, intrepreterji in debugerji so uporabljeni za generiranje kode. Različni nivoji programskih jezikov kot C, C++, Pascal, Java so uporabljeni za kodiranje. Z upoštevanjem tipa aplikacije je uporabljen pravi programski jezik.

2.1.1.5 Testiranje

Testiranje ni le izolirana aktivnost, ki preverja rezultate kodiranja, temveč se mora izvajati ves čas razvoja programske opreme. Rezultate vseh aktivnosti je potrebno preveriti. Prej ko odkrijemo neko napako ali pomanjkljivost med razvojem, lažje in ceneje jo lahko odpravimo. Ko testiramo programsko kodo, je vsak element ali modul programske kode potrebno testirati posebej, med integracijo pa postopno, po vsakem dodanem modulu. Po končani integraciji sledi še cela vrsta sistemskih testiranj.

2.1.1.6 Obratovanje in vzdrževanje

Vzdrževanje programske opreme je potrebno zaradi odpravljanja napak po končanem razvoju in zaradi spremenjenih zunanjih okoliščin, v katerih sistem deluje. Razlogov za spremembe je mnogo. Do sprememb lahko pride , ker se nepričakovano spremenijo vhodi

ali izhodi v sistem. Programska oprema mora biti razvita na način, ki omogoča prilagajanje spremembam, ki nastanejo po implementaciji.

2.2 MODEL PROTOTIPA

Beseda prototip pomeni prvi vzorec. Bistvo te metode je, da se že takoj na začetku zgradi prvi vzorec rešitve, nato pa se ta vzorec postopno izpopolnjuje in dograjuje, dokler ne dobimo končne sprejemljive rešitve.

Metoda prototipa poteka v štirih fazah (Gradišar, 2001, str. 430):

- definiranje osnovnih informacijskih potreb uporabnika (analiza in načrtovanje);
- razvoj prototipne rešitve;
- uporaba prototipa za prečiščenje in izpopolnitev uporabnikovih zahtev (iteracije);
- izboljšava prototipa.

V prvi fazi na podlagi sestankov in pridobljene dokumentacije definiramo problem naročnika, določimo cilje projekta, analiziramo obstoječi sistem in pripravimo študijo izvedljivosti projekta. Ko je enkrat izvedena analiza potreb in pripravljen načrt za izvedbo prototipa, se začne faza razvijanja programske rešitve oziroma prototipa. V tej fazi pripravimo konceptualni načrt, izberemo sistem za upravljanje s podatkovno bazo in razvijemo prvo prototipno rešitev. Ko je prototip razvit, je izročen stranki za testiranje. Stranka prototip testira in ugotovitve poda razvijalcu, ki popravi in nadgradi prototip na podlagi strankinih pričakovanj. Faza se ponavlja dokler stranka ni zadovoljna s prototipom. V tej metodi se programska oprema razvije s periodičnimi izmenjavami informacij med stranko in razvijalci.

Metoda prototipa se med drugim navezuje na metodo hevrističnega analiziranja in sicer na področju iskanja rešitev. Uporabnika se spodbuja, da rešitev poišče sam oziroma predlaga ustrezne smernice. Hevristična analiza pri tem uporablja kot izhodišče že znano ugotovitev, da je v vseh fazah razvoja informatike nujno vključevanje uporabnikov.

Prototipni pristop je s stališča postopkov izvajanja podoben simulaciji. Simulacijo v tem primeru opredelimo kot preizkušanje na modelu brez tveganja za izvirnik. Njen namen je spoznavanje raziskovalnega pojava, predvidevanje njegovega obnašanja in iskanje učinkovitih ukrepov. V primeru obdelave dveh sistemov, od katerih se eden preslika v drugega, obravnavamo enostavnejšega (model, prototip) in z njegovo pomočjo raziskujemo kompleksnejšega.

Uporaba prototipa je še posebej uporabna v primeru, ko (Avison, 1995, str. 79):

- niso natančno definirane specifikacije aplikacij;

- organizacija ni seznanjena s tehnologijo (strojna in programska oprema, komunikacije itd.);
- je slaba komunikacija med načrtovalcem in uporabnikom;
- so stroški zavrnitve projekta zelo visoki in je bistveno, da se zagotovi, da končna različica v popolnosti ustreza uporabnikovim zahtevam.

2.3 POSPEŠEN RAZVOJ PROGRAMSKE REŠITVE (METODA RAD)

Izjemno konkurenčno in zato hitro spreminjajoče poslovno okolje je privedlo do razvoja metodologije pospešenega razvoja programske rešitve (RAD - Rapid Application Development). Metodologija je najbolj znana po delu Jamesa Martina, ki je definiral in uveljavil omenjeno metodologijo. Glavne karakteristike metodologije so (Avison, 1995, str. 394):

- metodologija ni bazirana na tradiconalnem življenjskem ciklu, ampak uporabi evolucijski/prototipni pristop;
- osredotoči se na identificiranje najpomembnejših uporabnikov, ki se jih skozi razne delavnice vključi v zgodnje fazo načrtovanja;
- osredotoči se na pridobivanje zaupanja poslovnih uporabnikov v nastajajoči sistem;
- potrebuje CASE orodja z zmogljivim repozitorijem.

Faze RAD metodologije:

- Načrtovanje potreb z uporabo tehnike JRP workshop (Joint requirement planning)
  Tehniko se uporablja za identificiranje potreb in ciljev na najvišjem (strateškem) nivoju organizacije. Udeleženci delavnice so višji management, ki predstavi strateške cilje in vizijo organizacije.
- Uporabniško načrtovanje z uporabo tehnike JAD workshop (Joint Application Design) in CASE orodij in orodij za izdelavo prototipa
  JAD tehnika predpostavlja, da so običajni analitični pristopi kot so intervjuji in vprašalniki na uporabniškem nivoju nezadostni. Dobra analiza je dosežena s pravimi ljudmi, pravim okoljem in dobrimi orodji za izdelavo prototipa.
- Razvoj z uporabo CASE orodij in orodij za razvoj prototipa
  Podlaga za razvoj načrta je pripravljen v prejšni fazi. Generira se prototip, ki se nato z uporabniki podrobno pregleda in naknadno dodela, dokler ne ustreza zahtevam uporabnika.
- Pregled in testiranje
  V tej fazi se sistem podrobno testira z realnimi podatki in v operativni situaciji.

2.4 AGILNE METODE RAZVOJA PRGRAMSKIH REŠITEV

Temelji agilnih metodologij so bili postavljeni decembra 2001, ko se je sestala skupina sedemnajstih gurujev znanih s področja lahkih metodologij (Adaptive Software Development, XP-Extreme Programming, Feature-Driven Development, Crystal, Scrum, Dynamic System Development Method idr.). Sestali so se z namenom, da bi ugotovili, kaj imajo posamezne metodologije skupnega in na osnovi tega postavili skupne metodološke osnove. Preučili so lahke metodologije, katerih število je v zadnjih letih močno naraslo. Njihova temeljna lastnost je učinkovitost in prilagodljivost, s čimer je skupina sedemnajstih strokovnjakov, združena v zvezo Agile Alliance, opisala nov trend agilnih metodologij.

V skupni izjavi (Bajec, 2001, str. 72) so člani zveze kot njihov cilj podali iskanje boljših pristopov razvoja programske opreme, tako da pri tem sami sodelujejo in pomagajo drugim. Iz tega so izpeljali štiri načela:

- posamezniki in njihova komunikacija so pomembnejši kot sam proces in orodja;
- delujoča programska oprema je pomembnejša kot popolna dokumentacija;
- vključevanje (sodelovanje) uporabnika je pomembnejše kot pogajanje na osnovi pogodb;
- upoštevanje sprememb je pomembnejše od sledenja planu.

Proces, orodja, dokumentacija, pogodbe in načrti so pomembni elementi metodologije, vendar pa je v primeru, ko je potrebno postaviti prioritete, pomembneje poskrbeti za posameznike, sodelovanje, delujočo programsko opremo, vključevanje uporabnika in reagiranje na spremembe.

Iz načel, ki veljajo kot osnova agilnih metodologij, so izpeljana priporočila, ki so v pomoč pri gradnji in vrednotenju metodologij (Bajec, 2001, str. 72).

- Delujoče komponente programske opreme je potrebno izdajati pogosto, v ciklu, ki ni daljši od štirih mesecev. Če je naročnik zmožen sprejemati izdelke v krajših razmakih in je tudi razvojna ekipa sposobna upoštevati vse sprotne spremembe, potem so krajši cikli še boljši.
- Spremembe v zahtevah naj bodo dobrodošle tudi v poznih fazah razvoja. Eno izmed načel agilnih pristopov je dober odnos med naročnikom in izvajalcem. Spremembe, ki nastanejo med projektom, in smo jih pripravljeni upoštevati, lahko prinesejo naročniku pomembno konkurenčno prednost, zato se jih ne smemo otepati.
- Projekti naj vključujejo motivirane posameznike. Omogočeno naj jim bo ustrezno delovno okolje ter vse, kar pri svojem delu potrebujejo. Zaupati jim je potrebno, da bodo delo opravili dobro. Nedvomno so na projektu koristnejši motivirani posamezniki, ki med seboj dobro komunicirajo, čeprav ne sledijo predpisanemu procesu, kot pa nemotivirani ljudje. Posamezniki lahko s svojim odnosom do dela v veliki meri vplivajo na uspeh oziroma neuspeh projekta.

- Kakovost programske kode in arhitekture je potrebno stalno preverjati in izboljševati. S tem povečamo prilagodljivost. Pristop, kjer najprej izpeljemo poglobljeno analizo, izdelamo dober načrt in nato razvijemo programsko kodo, je priporočljiv, vendar si ga pogosto ne moremo privoščiti. Zaradi potrebe po hitrih rezultatih je kakovost velikokrat zapostavljena, zato je pomembno, da v vsakem ciklu strmimo k izboljšavam.
- Enostavnost procesa je ključ do uspeha. Kako razviti kakovostno programsko opremo, pri tem pa opraviti čim manj stranskih nalog in izdelkov? Proces mora biti ravno dovolj zahteven, raje malo manj kot pa preveč. To nikakor ne pomeni, da moramo aktivnosti izvajati površno ali pomanjkljivo. Nasprotno, delo mora biti opravljeno kvalitetno, paziti pa je treba, da ni po nepotrebnem zapleteno. Pot do enostavnosti je v resnici težka, saj je včasih preprosteje pustiti stvari kompleksne, kot jih pa poenostaviti.
- Po vsaki iteraciji moramo preveriti proces, ki smo ga uporabljali, ugotavljamo, kaj je bilo pri tem dobrega in kaj odveč. Glede na ugotovitve proces prilagodimo in izboljšamo za naslednje iteracije.

2.5 IZBOR METODE RAZVOJA PROGRAMSKE REŠITVE

Metoda razvoja programske rešitve je zelo odvisna od projekta, ki ga moramo izpeljati. Tem večji je projekt tem bolj je pomembna faza načrtovanja in analiza. Pri manjših projektih si lahko privoščimo večje korake nazaj in več potrpežljivosti z naročniki, saj spremembe ponavadi nimajo prevelikega vpliva na uspešnost projekta. Prav tako je z velikostjo projekta in projektne skupine povezana količina dokumentacije in administracije. Z večanjem skupine nastopi problem komunikacije in pripravljenosti (stopnja obremenjenosti) posameznika na sodelovanje v timu. Zato je pomembno, da se čimveč problemov in rešitev dokumentira. Dokumentacijo se lahko dobro izkoristi za nadaljno uporabo v tekočem in v prihodnjih projektih.

V strateškem načrtu podjetja sta bistveni dve dejstvi. Kot manjše hčerinsko podjetje (20 zaposlenih) multinacionalnega koncerna podružnica nima lastnega IT oddelka niti uslužbenca, ki bi skrbel izključno za področje informacijske tehnologije. Za osnovno informacijsko tehnologijo skrbi lokalno podjetje, ki nudi osnovno podporo (mreža, strojna oprema in manjša popravila). Podporo za informacijski sistem (SAP), elektronsko pošto, povezanost do internih baz podatkov in baz znanj pa nudi IT oddelek na Madžarskem. Drugo dejstvo je, da podjetje na ravni koncerna namerava v prihodnjem letu izvesti outsourcing informacijske tehnologije. Zaradi omejenih možnosti so se na nivoju slovenskega podjetja odločili, da bodo v prihodnjem letu investirali samo v delno obnovo strojne opreme in nakup oziroma razvoj programskih rešitev, ki prispevajo k večji produktivnosti zaposlenih.

Od v prejšnjih poglavjih omenjenih metod dela se je mojemu konkretnemu problemu zaradi uporabnikovih zahtev in načina dela najbolj prilegala metoda prototipa. Pred pričetkom razvoja programske rešitve namreč ni bilo mogoče izdelati natačnih specifikacij in vse potrebe niso bile jasne. Skozi izdelavo delnih rešitev (prototipov) in naknadnih informacij smo prišli do rešitev, ki najbolj ustrezajo končnemu uporabniku.

3 NAČRTOVANJE

Načrtovanje predstavlja proces priprave na delo z namenom doseči točno določen cilj. Z ugotovitvijo, da je projekt koristen organizaciji, je vodstvo dalo projektu zeleno luč. V prvi fazi se je bilo potrebno seznaniti s problemom in na podlagi tega določiti cilje projekta. Iz ciljev je tudi jasno, kaj bodo koristi projekta. Oceniti je potrebno tudi stroške, ki bodo nastali pri izvedbi in urnik projekta, ki je bil v tem primeru dokaj ohlapno narejen. Na podlagi zbranih informacij je bila izvedena prva iteracija (prvi prikaz rešitve), na podlagi katere smo nadaljevali delo.

3.1 DEFINIRANJE PROBLEMA

Ponudbe za produkte srednje napetosti so se do sedaj kreirale ročno. To pomeni, da so se vsi izračuni cen, opisi kompenent, plačilni pogoji itd. vnašali oziroma kopirali v wordov dokument, katerega je zaposleni natisnil in po faksu poslal stranki. Izračun cene za modularni SN stikalni blok tipa SM6 je zaposleni natisnil ali fotokopiral postopke za določanje cen. Nato je določil sestavne dele celice in s kalkulatorjem izračunal skupno nabavno ceno za celico. Nato je postopek ponovil za vsako nadaljnjo celico. Različnih celic je lahko do 20, kar zavisi od zahtev stranke. S pridobljenimi podatki je potrebno izpolniti še obrazec za stikalno omaro, ki vsebuje vse te celice in izračunati dodatno ceno sestavnih delov, ki povezujejo celice med sabo. Po posvetovanju znotraj podjetja je določil popust, ki je pripadal določeni stranki. Na podlagi storjenih operacij je sestavil ponudbo z opisom celic in stikalne omare, s skupno ceno projekta, s plačilnimi pogoji, z roki dobave in ustreznimi prilogami (skice, kreirane s pomočjo Autocada) ter ponudbo poslal ustrezni osebi na vnaprej določeno podjetje. Skratka prvi večji identificirani problem predstavlja čas izdelave ponudbe. Drugi problem se izraža v iskanju podatkov v množici ponudb. Prodajno osebje bi rado imelo pregled nad ponudbami, ki so jih poslali določeni kontaktni osebi znotraj dotičnega podjetja. Zanima jih katere produkte so jim ponudili in kdaj, kakšni so bili popusti itd. Tretji problem je v organiziranosti ponudb. Vsak namreč izdeluje ponudbe sam in vseh ne shrani na predvideno mesto, kar pomeni, da ostali nimajo dostopa do potrebnih informacij.

Produkti srednje napetosti delimo v tri večje sklope. In sicer:

- Suhi energetski transformator tipa TRIHAL;
- Oljni energetski transformator tipa MINERA;

- Modularni SN stikalni blok tipa SM6.

3.2 ZAGON PROJEKTA

Za uspešen zagon projekta je bilo potrebno opredeliti cilje in koristi projekta, ki vodstvu povedo, kakšen rezultat lahko pričakujejo od projekta. Na drugi strani pa je potrebno izvesti študijo izvedljivosti, ki med drugim pove, kaj je potrebno postoriti za uspešno izvedbo projekta in kakšne posledice ima projekt na razpoložljive resurse.

3.2.1 CILJI IN KORISTI

Cilji projekta (razvoj programske rešitve za kreiranje ponudb izdelkov srednje napetosti) so naslednji:

- zmanjšati čas izdelave ponudb za produkte srednje napetosti;
- omogočiti hiter dostop uporabnikov do ponudb (tudi brez povezave na strežnik);
- omogočiti povratne informacije (kdaj, komu, kaj in s kakšnim popustom);
- uporaba obstoječih razvojnih okolij in minimalni dodatni stroški razvoja.

3.2.1.1 Zmanjšanje časa izdelave ponudb za produkte srednje napetosti

Trenutno kreiranje zahtevnejših ponud srednje napetosti traja približno pol delovnega dne, kar zelo obremenjuje zaposlene v prodaji in se zato posledično premalo ukvarjajo s prodajo samo in dodatnim izobraževanjem. Cilj ni bil natančno določen, a se predvideva, da bi lahko delna avtomatizacija čas izdelave ponudbe skrajšala do ene delovne ure pripravljalca ponudbe. Koristi so jasne, manjša obremenjenost, večja produktivnost, več časa za izobraževanje in kar je najpomembneje, več časa za stike s potencialnimi kupci.

3.2.1.2 Hiter dostop uporabnikov do ponudb

Rešitev mora omogočati replikacijo baze na lokalni računalnik, tako da bo uporabnik programske rešitve (vsi imajo prenosne računalnike) lahko poizvedoval po ponudbah tudi doma ali na terenu. Delo prodajnikov je takšno, da malo časa delajo v pisarni, ampak so na terenu, pri strankah, kjer pa ravno tako potrebujejo hitro in zanesljivo informacijo.

3.2.1.3 Povratne informacije

Z uporabo poizvedb je potrebno doseči, da bo imel uporabnik kvalitetne povratne informacije o poslanih ponudbah. V kopici raznoraznih podatkov in informacijah mora prodajnik hitro in učinkovito pridobiti informacijo o že ponujenih izdelkih in popustih, s katerimi je ponudil ta izdelek.

3.2.1.4 Uporaba obstoječih razvojnih okolij

Želja vodstva je, da se projekt izpelje s kar najmanjšimi stroški ter da se po nepotrebnem ne kupuje dodatnih programskih rešitev in licenc.

3.3 IZVEDLJIVOST PROJEKTA

Izvedljivost projekta ali študija izvedljivosti predstavlja pregled stanja z namenom ocene določenega načrta glede njegovih možnosti in racionalnosti. Preden lahko projekt preide v naslednjo fazo, moramo preveriti njegovo izvedljivost, in sicer glede na finance, delovanje ter tehnologijo.

3.3.1 FINANČNA IZVEDLJIVOST

V podjetju za podobne projekte sredstva niso posebej namenjena. Prav tako podjetje nima redno zaposlenega informatika in so se iz tega razloga odločili za najem kadra s potrebnimi znanji preko študentskega servisa. Finančni strošek ne predstavlja bistvene obremenitve.

3.3.2 EKONOMSKA IZVEDLJIVOST

V času načrtovanja projekta ni bila izdelana natančnejša analiza ekonomske izvedljivosti. Vendar pa je eden izmed ciljev projekta zmanjšanje časa izdelave posamezne ponudbe, kar pomeni prihranek na račun programske rešitve vsaj ena človek ura na izdelavo ponudbe in iskanje le teh. V enem koledarskem letu podjetje pošlje do 500 ponudb (produkti srednje napetosti) različnim strankam. Projekt je ekonomsko izvedljiv, saj bodo stroški razvoja manjši od prihrankov, ki se kažejo v večji produktivnosti.

3.3.3 OPERATIVNA IZVEDLJIVOST

Projekt je operativno izvedljiv, saj je kadrovska pozicija izpopolnjena, posebnih tehničnih pripomočkov in licenc pa ni potrebnih. Določeni so bil odgovorni za projekt. Čas porabljen za svetovanje, opis funkcionalnosti in testiranje s strani uporabnikov ne bo bistveno vplival na njihovo vsakdanje delo.

3.3.4 TEHNOLOŠKA IZVEDLJIVOST

Za izvedbo projekta bo uporabljen Microsoft Access, Excel, VBA, API funkcije, tako da naprednejših tehnologij ne bo uporabljenih in s tem ne bodo potrebni dodatni nakupi licenc za razvojne programe.

4 ANALIZA

Analiza je proces razčlembe stvari na njene sestavne dele z namenom spoznati njeno sestavo, posamezne dele ali strukturo kot celoto. Potem, ko sem v fazi načrtovanja preučil zahtevo po prenovi sistema in ugotovil, da je projekt izvedljiv in potrjen s strani vodstva, sem prešel na drugo fazo, to je faza analize.

4.1 OBSTOJEČI SISTEM

V tem poglavju se bom osredotočil na del informacijskega sistema v podjetju, ki vpliva na razvoj moje programske rešitve. Celoten sistem je širše narave in je povezan z materinskim podjetjem v Franciji oziroma z ustreznimi podružnicami podjetja v vzhodni Evropi. Posebnost podjetja je velik delež uporabe prenosnih računalnikov. Kar 15 od 22 zaposlenih uporablja prenosni računalnik za vsakdanje delo. Razlog je v dejavnosti podjetja, ki se ukvarja s prodajo elektronike na slovenskem trgu. Večina zaposlenih je večkrat tedensko izven domače lokacije. Nekateri pa imajo zaradi optimizacije potnih stroškov pisarno kar doma.

V podjetju so izdelali programski rešitvi z imenom Star++ in PonudbeXP. Za obe programski rešitvi razviti v podjetju so bile uporabljene napredne funkcije Microsoft Offica 2000 (VBA, API, replikacije, sinhronizacija itd.).

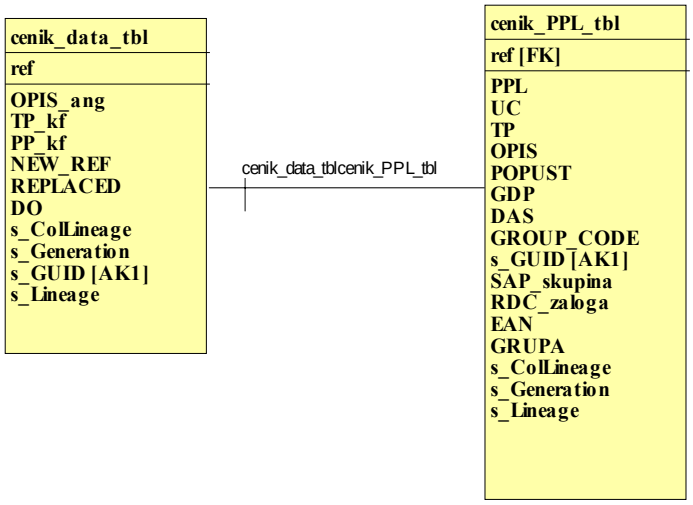
Star++ ima več sklopov z različnimi funkcionalnostmi:

- Cenik
  Cenik je najpomembnejši del celotne programske rešitve, saj se na njegovi podlagi kreira večina ponudb. V ceniku je shranjenih 121 tisoč različnih referenc (produktov oziroma njihovih sestavnih delov) z dobavnimi in prodajnimi cenami, opisom, popustom, skupino itd. Preko vmesnika prodajni referenti, tržniki, logistiki in zaposleni, odgovorni za tehnično pomoč strankam, dostopajo do podatkov posameznih referenc.
- Naslovi podjetij in kontaktnih oseb
  V tem sklopu se nahaja seznam vseh podjetij in kontaktnih oseb, s katerimi sodeluje podjetje. Za vsako stranko je določen komercialist oziroma prodajnik, ki vsak obisk pri stranki zabeleži in na koncu meseca natisne poročilo o preteklem delu. Sistem omogoča nadzor in pregled nad opravljenim delom, ter nudi komercialstu povratno informacijo o številu obiskov določene stranke.
- Preverjanje ustreznosti naročil

Referentka logistike s to funkcijo preveri ustreznost naročil. Najprej generira datoteko iz sistema SAP[2], to datoteko nato program vnese v tabelo in jo primerja z lokalnim cenikom. Morebitna odstopanja se izpišejo na ekran ali na tiskalnik v obliki poročila. Odstopanja se rešujejo v sodelovanju z odgovornim produktnim vodjem.

- Tehnična pomoč
  Stranke pri uporabi produktov pogosto naletijo na težave in nejasnosti. Z uporabo elektronske pošte, faksa ali telefona kontaktirajo zaposlene v tehnološki podpori, ki jim poskušajo v čim krajšem času rešiti nastale probleme. Da bi zagotovili red pri evidentiranju težav strank, je bila razvita dodatna funkcionalnost v sistemu Star++ z imenom Tehnična pomoč. Vsak zaposleni, ki sprejme zahtevo po pojasnitvi problema, vnese v aplikacijo sledeče kategorije: osnovne podatke o stranki, opis problema, datum prejema zahtevka, opis rešitve, datum posredovanja rešitve, produkt, pri katerem se je pojavila težava. Na podlagi vnešenih podatkov je zagotovljena povratna informacija o število še nerešenih zahtev, povprečni čas reševanja enega zahtevka, najbolj problematičnih produktih ipd.

Slika 1: ER-model cenik generiran z uporabo funkcije "Reverse engineer database" (Visible Analyst[3])

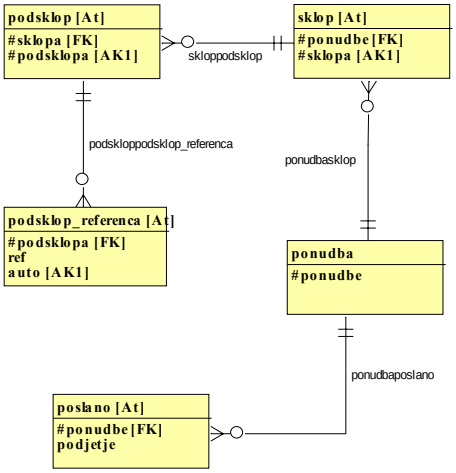


Vir: Lasten vir.

Aplikacija PonudbeXP je prirejena za izdelavo ponudb za projekte nizke napetosti. Sestavljena je iz baze (ponudbe_be.mdb) in uporabniškega vmesnika (ponudbe_fe.mdb). Projekt za nizko napetost je zasnovan iz mnogo sklopov in podsklopov, ki so sestavljeni iz posameznih sestavnih delov. Celotna cena projekta je zasnovana na podlagi seštevka vseh delov in z upoštevanjem popusta za določeno podjetje.

[2] Informacijski sistem SAP razvija in trži nemška korporacija SAP AG.
[3] Program Visible Analyst trži podjetje Visible Systems Corp.

Slika 2: ER-model PonudbXP generiran z uporabo funkcije "Reverse engineer database" (Visible Analyst)



Vir: Lasten vir.

4.2 ŽELJENI SISTEM

Željeni sistem je delno avtomatiziran postopek opisan v poglavju 3.1. Ustrezati mora zapisanim ciljem. To pomeni, da mora pospešiti izdelavo ponudbe in organizirati kreirane ponudbe na način, ki bo omogočil kar najhitrejše iskanje podatkov znotraj ponudbe. Potrebno je vzeti v obzir, da naročnik želi programsko rešitev v prihodnje dograjevati z dodatnimi funkcionalnostmi (npr. naročanje produktov, kalkulacije ipd.).

4.3 IZVORI DEJSTEV

Informacije potrebne za razvoj programske rešitve sem zbral na tri različne načine. In sicer:

- na sestankih,
- v interni dokumentaciji,
- z iteracijami.

4.3.1 SESTANKI

Na temo razvoja aplikacije smo se večkrat sestali. Prvi sestanek je bil z vodjem tehnične pomoči, ki je predstavil idejo razvoja aplikacije in razporedil delo. Pripravljen je bil dokument, kjer so zapisane ideje o prenovi obstoječega sistema, informatizaciji in avtomatizaciji izdelave ponudb izdelkov srednje napetosti. Naslednji sestanek je bil z vodjem prodaje za izdelke srednje napetosti in njegovim pomočnikom, ki sta predstavila sistem dosedanjega dela in željeno stanje. Pripravljen je bil dokument z okvirnimi

zahtevami. Na sledečem sestanku mi je pomočnik prodaje podrobneje razložil specifičnosti dela na ponudbah.

4.3.2 INTERNA DOKUMENTACIJA

Interno dokumentacijo sestavljata v prejšnjem poglavju omenjena dokumenta, ki sta nakazala smer razvoja aplikacije in okvirne cilje. Za samo načrtovanje in razvoj aplikacije so bili najbolj bistveni primerki že narejenih ponudb in predvsem modeli (drevesa odločitve) izračuna cen. Ostalo dokumentacijo sestavljajo raznorazni katalogi in predstavitve izdelkov, njihove slike, kalkulacije ipd.

4.3.3 ITERACIJE

Metodo prototipa sem med drugim izbral tudi zato, da v procesu iteracij dobim povratne informacije s strani uporabnika. V pogovorih in predstavitvah na konkretnem primeru je uporabnik dobil nove ideje in s spoznavanjem možnosti tehnologij podal nove predloge, ki bi najbolj ustrezale njegovim zahtevam. Mnoge podrobnosti, ki v osnovnih specifikacijah niso bile zabeležene ali omenjene na prvih sestankih, so bile kasneje dodane na podlagi informacij pridobljenih v procesu iteracij.

4.4 OPREDELITEV ZAHTEV

Na podlagi zbranih informacij so se jasneje oblikovale zahteve uporabnikov. Zahteve bom razdelil na tri sklope. In sicer na:

- vhode v sistem,
- izhode iz sistema,
- sistem.

Na tem mestu je potrebno poudariti, da so se zahteve skozi iteracije spreminjale in dopolnjevale. To je logična posledica izbire metode, ki uporabnika spodbuja k spoznavanju programske rešitve in podajanju novih predlogov, kakor tudi že sprotnega odpravljanja morebitnih napak, ki imajo izvor v pomanjkljivih začetnih specifikacijah. Spremembe in dopolnitve so posebej omenjene pri opisu iteracij.

4.4.1 VHODI V SISTEM

Vhodi v sistem so v določeni meri podobni izhodom. V sistem je potrebno vnesti, komu se pošilja ponudba (podatki o kontaktni osebi in podjetju), kdaj je bila poslana (podatki o datumu prejema povpraševanja in datumu pošiljanja), kdo jo je pripravil in podpisal (ime in priimek pripravljatelja, overitelja, elektronski naslov, telefon itd.), kaj vsebuje (naziv

izdelkov po sklopih, količina, tip, specifikacije celic itd.), pod kakšnimi pogoji je bila ponujena (cena, popusti, dobavni rok, splošni in ostali pogoji itd.).

4.4.2 IZHODI IZ SISTEMA

Izhodi iz sistema so razdeljeni na dva podsklopa, na ponudbo (in njene različice) in poročila. Izpis ponudbe na zaslon, tiskalnik in v datoteko (format RTF) je pogoj za obstoj sistema. Ponudba mora v glavi vsebovati logo podjetja in v nogi osnovne podatke o podjetju. Pod logom podjetja se zvrstijo podatki o pošiljatelju in prejemniku, zadevi in kratek povzetek vsebine ponudbe (imena izdelkov, ime projekta oziroma objekta). Komercialni pogoji sestojijo iz skupne cene, popusta, cene s popustom, dobavnim rokom itd. Iz vidika prodajnika pa so seveda izjemno pomebni komercialni pogoji poslane ponudbe. Obstajajo pa še splošni in ostali pogoji, ki se od ponudbe do ponudbe razlikujejo. Produkte srednje napetosti delimo v tri večje sklope, ki imajo lahko zelo različne zahteve. Za produkta "Suhi energetski transformator tipa TRIHAL" in "Oljni energetski transformator tipa MINERA" mora ponudba vsebovati informacije o moči in napetosti transformatorja, materialu iz katerega je izdelan, količini ponujenih transformatorjev in dodatnih specifikacijah, ki jih po lastni presoji določi uporabnik.
Najzahtevnejši del izdelave ponudbe produkta "Modularni SN stikalni blok tipa SM6" predstavljata konfiguracija in opis specifikacij celic. Konfiguracija predstavlja stikalno omaro, ki vsebuje posamezne celice. Optično se na ponudbi izraža konfiguracija kot dvodimenzionalna tabela, v kateri so informacije o zaporedni številki celice, tipu celice in nazivnem toku. V tekstu pod tabelo so omenjeni še podatki o vodilu in toku. V nadaljevanju je podan opis vseh vključenih celic v stikalno omaro. Opis celic je sestavljen iz seznama funkcijskih specifikacij posamezne celice.

Poročila oziroma izpisi se večinoma uporabljajo za pridobivanje povratnih informacij o poslanih ponudbah. V začetku so bili predvideni naslednji izpisi:

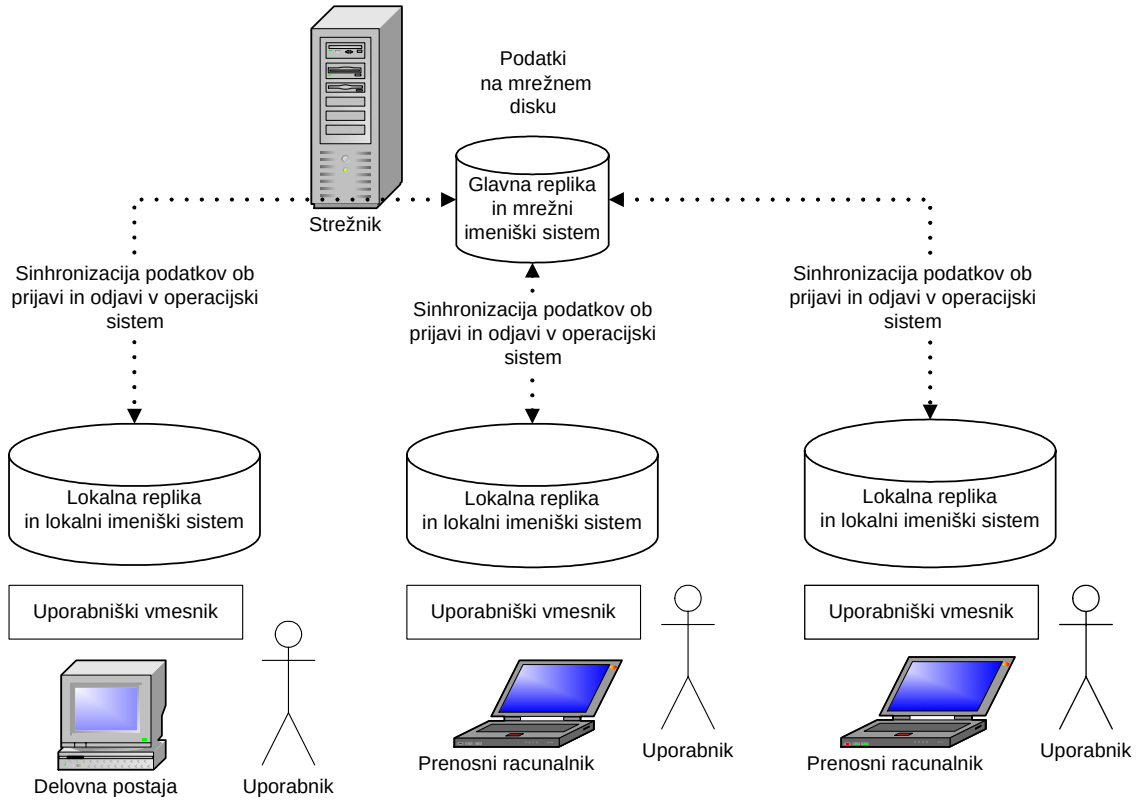
- izpis vseh poslanih ponudb;
- izpis vseh poslanih ponudb tekočega leta;
- izpis vseh poslanih ponudb za določen datum (točno določen datum, vse ponudbe starejše od določenega datuma, vse ponudbe, ki so bile kreirane pred ali po določenem datumu);
- izpis vseh poslanih ponudb določenemu podjetju;
- izpis vseh poslanih ponudb, ki vsebujejo določen izdelek;
- izpis vseh poslanih ponudb, ki jih je pripravila določena oseba.

4.4.3 SISTEM

Osnovno delovanje sistema mora omogočati hiter dostop do pripravljenih ponudb, možnost popravljanja ponudb, predogled in tiskanje ponudb, ter njihovo pretvorbo v tekstovno datoteko.

Naprednejša rešitev mora zagotavljati dostop do informacij tudi, če uporabnik ni priključen na mrežo. Predlagan je bil sistem replikacij, ki je že uspešno uporabljen v podjetju. Sistem replikacij je opisan v poglavju 5.3.2. Uporabljen bo pristop porazdeljene obdelave podatkov. Znotraj tega pristopa obstajajo porazdeljene baze podatkov[4], kar pomeni, da se baza podatkov ne nahaja le na enem, ampak na dveh ali več računalnikih v mreži. Prednost tega pristopa je, da bolj ustreza potrebam (npr. delo izven podjetja) posameznih uporabnikov.

Slika 3: Končna rešitev željenega sistema



Vir: Lasten vir.

5 RAZVOJ PROTOTIPA IN ITERACIJE

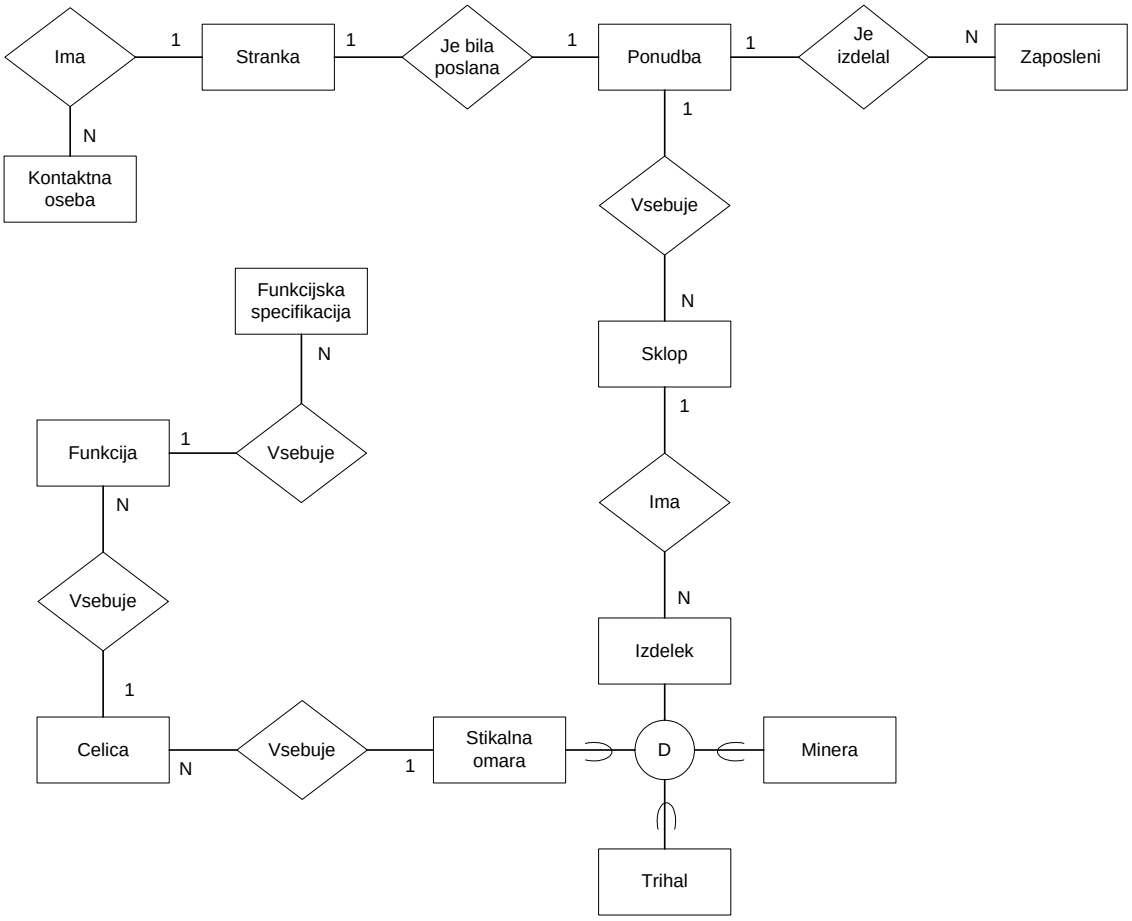
Namen tega poglavja je predstaviti način razvoja programske rešitve. Predstavljen bo konceptualni model baze podatkov, ki na prvi stopnji preslika problematiko podjetja v razvijalcu bolj razumljiv diagram. V drugi fazi je bil izbran sistem za upravljanje baze podatkov, kateremu je sledila faza razvoja prve prototipne rešitve, ki se je prek štirih iteracij zaključila s končno različico programske rešitve.

[4] Baza podatkov je porazdeljena, če se na več mestih nahajajo njene kopije, ali če se na več mestih nahajajo njeni posamezni deli (Gradišar, 2001, str. 296).

5.1 KONCEPTUALNI MODEL BAZE PODATKOV

Konceptualni načrt je predstavitev uporabnikovih podatkovnih zahtev s pomočjo konceptualnega modela. Temelji na percepciji realnega sveta, ki ga modeliramo z bazo podatkov. Svet predstavljajo posamezna dejstva in pravila, ki jih opišemo z uporabo konceptualnega modela. V konceptualni model je potrebno vključiti vsa tista dejstva, ki so za model potrebna in jih ločiti od tistih, ki niso potrebna. Najbolj razširjen konceptualni model je model entiteta - razmerje (model ER), ki je sestavljen iz entitetnih tipov in razmerij med njimi in atributi.

Slika 4 : ER-diagram



Vir: Lasten vir.

Osnovni konceptualni model je preprost. Zaposleni, ki mu je bila zadana naloga, da pripravi ponudbo, jo sestavi na podlagi povpraševanja, ki je prejet v pisni obliki. Mnogo informacij pa se pridobi na podlagi srečanj s potencialnim kupcem. Ponudba je sestavljena iz različnih sklopov, ki vsebujejo ponujeni izdelek. Izdelek se razdeli na tri večje sklope: Minero, Trihal in stikalno omaro. Stikalna omara je sestavljena iz različnih celic. Celico določajo funkcije, ki jim moramo prirediti določeno funkcijsko specifikacijo. Vsaka ponudba je

sestavljena posebej za določeno stranko. Tako se ne more zgoditi, da bi isto ponudbo poslali na dva različna naslova.

5.2 IZBIRA SISTEMA ZA UPRAVLJANJE S PODATKOVNO BAZO

V podjetju imajo že razvite aplikacije, ki uspešno tečejo v Microsoft Access okolju. Zaradi pridobljenega znanja in integracije z že obstoječimi aplikacijami sem se odločil nadaljevati z uporabo Microsoft Accessa 2000. Glavni razlogi so v stroških, saj ima vsak zaposleni v podjetju različico Microsoft Office, ki vsebuje tudi Access. Tako se znižajo stroški, ki bi nastali z nakupom druge rešitve oziroma nakupom Microsoft Office Developer edition, s katerim lahko legalno distributiramo Access aplikacije, četudi uporabnik nima licence Microsoft Office, ki vsebuje Microsoft Access. Drugi razlog je v vzdrževanju, saj ima podjetje zadostno znanje, ki jim bo omogočalo administracijo, popravila morebitnih napak in manjše nadgradnje. Tretji razlog je v večjih nadgradnjah. Podjetje nima redno zaposlenega uslužbenca, ki bi se ukvarjal izključno z informacijsko tehnologijo. Strategija podjetja (multinacionalke) gre v smeri outsourcinga, tako da na tem področju tudi v prihodnje ne načrtujejo zaposlitve. Ker pa jim podobna orodja močno olajšajo in pospešijo delo, nameravajo razvijati programske rešitve tudi vnaprej. Obstoječa programska rešitev pa jim omogoča pridobitev poceni zunanjih kadrov za razvoj dodatnih rešitev.

5.3 UPORABLJENA ORODJA PRI RAZVOJU PROGRAMSKE REŠITVE

V fazi razvoja je moč uporabiti različna orodja, ki omogočajo izdelavo željene programske rešitve. V skladu s ciljem čim nižjih razvojnih stroškov sem uporabil orodja, ki so že bila na razpolago ali pa niso zahtevala nakup dodatnih licenc.

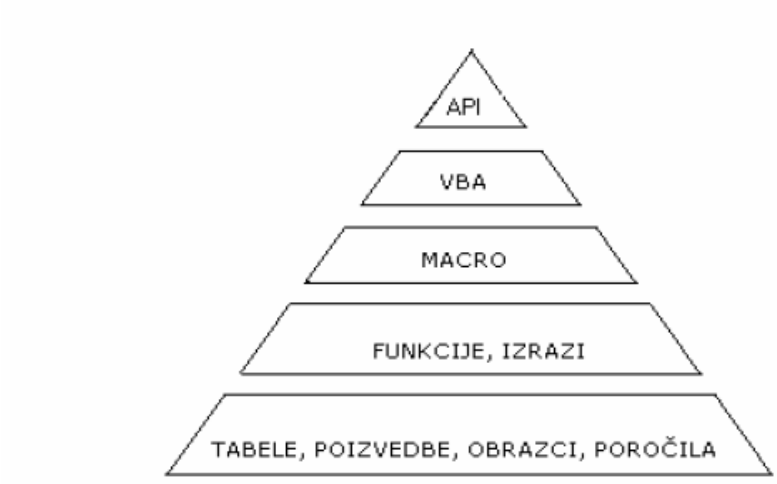
5.3.1 MICROSOFT ACCESS

Microsoftovo orodje Access 2000 je relacijska podatkovna zbirka, ki omogoča skladiščenje, iskanje, urejanje, združevanje in predstavitev podatkov v zaslonskih obrazcih ali natisnjenih poročilih (Prague, 1999, str. 7). Orodje Access je že del izdaje Microsoft Offica 2000. Uporablja se lahko kot namizna podatkovna zbirka za individualno uporabo ali pa na Microsoftovem SQL strežniku, ki omogoča, da podatkovno zbirko lahko uporabljajo tudi drugi uporabniki na mreži. Accessova podatkovna zbirka je sestavljena iz posameznih objektov, ki omogočajo izdelavo aplikacije za obdelavo podatkov. Ti objekti so: tabele, poizvedbe in prikazi, obrazci, poročila, strani za dostop do podatkov, makri in moduli.

Uporaba Accessa je preprosta za posameznega uporabnika, omogoča oblikovanje objektov s pomočjo čarovnikov, poleg tega pa ima zelo razčlenjen in izdelan menu pomoči. Objekte uporabnik lahko po svoji želji obdela, saj ima na razpolago veliko paleto barvnih odtenkov, senc, gumbov itd. Access omogoča tudi uporabo ali izmenjavo podatkov iz drugih baz s funkcijami povezave, izvoza in uvoza (Prague, 1999, str. 9). Povezava se uporablja za podatke v njihovem trenutnem formatu. Omogoča direkten dostop do podatkov v drugih Accessovih tabelah in tabelah v nekaterih drugih bazah podatkov. Uvoz ustvari kopijo zunanjih podatkov in jo prinese v Access, izvoz pa kopijo podatkov prenese v drug program.

Koncept Accessa je oblikovan v hierarhični obliki s petimi nivoji. Najnižji nivo v hierarhiji predstavljajo objekti: tabele, poizvedbe, obrazci in poročila, ki dajejo uporabniku možnost, da jih sam oblikuje po svojih željah in potrebah. Pri tem lahko uporablja različne vrste podatkov: številske, tekstovne, časovne in slikovne. Enostavno procesiranje podatkov je izvedljivo z uporabo funkcij in obrazcev. Makroji omogočajo avtomatsko izvajanje funkcij brez programiranja. Programski jezik VBA (Visual Basic for Application) je za uporabo bolj zahteven, ker so postopki procesiranja bolj kompleksni. Najvišji nivo predstavlja Windows API (Application Programming Interface), ki kliče funkcije ali knjižnice, napisane v drugih programskih jezikih kot so C, Java, Visual Basic. Primer uporabljene funkcije je v prilogi B. To pomeni veliko povezljivost med drugimi programi. Koncept Accessa omogoča, da lahko na vsakem nivoju uporabnik oblikuje ali uporablja vse elemente na nivojih, ki so pod nivojem v hierarhiji, na katerem se nahaja.

Slika 4 : Hierarhični koncept Accessa



Vir: Prague, 1999, str. 7

5.3.2 REPLIKACIJA ZBIRKE PODATKOV

Replikacija zbirke podatkov je sistem, ki kreira kopije zbirke podatkov (replike), ki jih je mogoče sinhronizirati z drugimi replikami v množici. Spremembe podatkov v replicirani

tabeli v eni repliki so prenesene in uveljavljene v drugih replikah. Glede na vidnost replike poznamo globalno, lokalno in anonimno. Vidnost replike med drugim določa, katere vrste replik se lahko iz nje ustvari, ali se lahko replika v množici replik obnaša kot glavna replika in kako obravnava spore med sinhronizacijo.

Globalna replika je tipična replika, iz katere se ustvari vse ostale vrste replik. Spremembam s strani globalne replike je mogoče v celoti slediti in jih lahko izmenjate s katero koli drugo globalno repliko v množici. Globalna replika lahko tudi izmenja spremembe s katero koli lokalno ali anonimno repliko, za katero postane vozlišče. Lokalne replike se sinhronizirajo samo s svojim vozliščem, torej z globalno repliko. Ni jim dovoljeno, da bi se sinhronizirale z ostalimi replikami v množici replik. Ostale lokalne replike ne zaznavajo lokalnih replik. Lokalne replike zaznava samo vozliščna replika, izmenjavo z lokalno repliko pa se lahko razporedi samo iz nje. Tudi vse replike, ki so ustvarjen iz lokalne replike, bodo lokalne in bodo imele isto nadrejeno repliko (Litwin, 2000, str. 560).

Anonimna replika se lahko sinhronizira s svojo nadrejeno, globalno repliko. Anonimne replike se zbirajo preko interneta in nimajo posebne identitete, ampak posredujejo svojo identiteto za posodabljanje z objavljenimi replikami. Sinhronizacija preko interneta ali intraneta deluje dobro, če množica replik ostaja majhna (manj kot 10 posameznih replik) in če je število vstavljanj in posodabljanj podatkov omejeno. Anonimne replike ponujajo način, da se izognemo težavi »omejitve števila replik«. Poleg tega uporaba anonimnih replik pomaga, da se obvarujete nepotrebnih topoloških informacij o replikah, ki sodelujejo samo občasno. Tudi vse replike, ki se jih ustvari iz anonimne replike, bodo anonimne in bodo imele isto nadrejeno repliko. Anonimne replike so priporočljive pri uporabi v internetu za masovno distribucijo, saj se ne vzdržuje sistem sledenja podatkov, velikost replike pa je omejena (Litwin, 2000, str. 560).

Za vzpostavitev dokaj zahtevnega sistema je potrebno upoštevati, da bodo ob uvedbi sistema replikacij na zbirki podatkov vidne določene spremembe, ki bodo vplivale na zahtevnost programiranja, vzdrževanja in uvajanja aplikacij. Spremembe so naslednje:

- Dodana so tri nova polja v vsaki tabeli in sicer: polje s_GUID, ki predstavlja globalno enolični identifikator za vsak zapis; binarno polje s_Lineage, ki vsebuje informacijo o zgodovini sprememb za vsak zapis; polje s_Generation, ki shranjuje informacijo v zvezi s skupinami sprememb.
- Dodanih je 7 novih tabel zbirki podatkov (MSysSidetables, MSysSchemaProb, MSysReplicas, MSysTransAddress, MSysTombstone, MSysRepInfo, MSys-ExchangeLog).
- Dodane so nove lastnosti zbirke podatkov: lastnost Replicable pove, ali je mogoče objekt replicirati; lastnost KeepLocal nam pove, da se objekt ne replicira, ko se replicira zbirka podatkov; lastnost ReplicaID vsakega člana množice replik oskrbi z enolično identifikacijo; lastnost Replication ConflictFunction je uporabljena za

zamenjavo Microsoft Accessovega pregledovalnika sporov s proceduro po meri, ki uporabnikom pomaga pri reševanju sinhronizacijskih sporov.

- Spremembe obnašanja samoštevilčnih polj ob repliciranju zbirke podatkov
  Pri repliciranju zbirke podatkov se vsa samoštevilčna naraščajoča polja v tabelah spremenijo v naključno številčena. Vsa samoštevilčna polja v obstoječih zapisih obdržijo svoje vrednosti, vendar so samoštevilčne vrednosti za vstavljene zapise naključne. Z drugimi besedami, števila zapisa ne odražajo vrstnega reda, v katerem so bili zapisi vstavljeni in zato zapis, vstavljen kot zadnji, nima nujno najvišje vrednosti.

5.3.2.1 Vpliv inštalacije sistema replikacije na velikost podatkovne baze

V prejšnjih poglavjih sem opisal vsa dodatna polja in tabele, ki jih sistem replikacij potrebuje za uspešno delovanje. Ker je teh relativno veliko, sem se odločil, da preverim kako dodana funkcionalnost vpliva na velikost podatkovne baze.

Tabela 2: Primerjava velikost baze z ali brez sistema replikacij

Velikost	Opis
78.188 kB	Produkcijski cenik po kompaktiranju
19.760 kB	Čista baza (brez dodatnih tabel, polj, indeksov in relacij)
25.096 kB	Dodani so primarni ključ, indeks, relacija med tabelami
68.788 kB	Dodana je replikacija

Vir: Lasten vir.

Produkcijski cenik (ER-diagram predstavljen na sliki 1) je bil po kompaktiranju velik 78.188 kB. Z uporabo SQL stavka sem kreiral v popolnoma novi in prazni bazi dve novi tabeli, brez dodatnih polj in tabel, ki jih kreira sistem replikacij. Velikost nove, čiste baze je znašala 19.760 kB. Vendar pa pri prenosu niso bili kreirani primarni ključ, indeks in relacija med tabelama, zato sem jih kreiral ročno. S spremembami je velikost narasla na 25.096 kB. Po tej predpripravi sem kreiral glavno repliko (master design) in hkrati repliko cenika. Velikost glavne replike je znašala 69.628 kB, velikost replike pa nekoliko manj in sicer 68.788 kB. Razlika med bazama (dve tabeli po 121.000 zapisov) je torej kar 43.692 kB.

5.3.3 SINHRONIZACIJA

Eden izmed ciljev projekta je zagotovitev delovanja aplikacije tudi, ko uporabnik nima dostopa do strežnika v podjetju. V ta namen je potrebno sinhronizirati tako bazo podatkov kot tudi točno določeno imeniško strukturo.

5.3.3.1 Sinhronizacija baze podatkov

Sinhronizacija je postopek posodabljanja dveh članic množice replik z izmenjavo vseh posodobljenih zapisov in predmetov v obeh članicah. Članici množice replik sta sinhronizirani, ko so spremembe v prvi repliki uveljavljene v drugi in obratno.

V Microsoft Accessu, Replication Managerju in JRO (Jet and Replication Objects) so na voljo tri metode sinhronizacije podatkov:

- Neposredna sinhronizacija se uporablja za sinhronizacijo podatkov med replikami, ki so neposredno povezane v lokalno omrežje in so na voljo v skupnih omrežnih direktorijih.
- Posredna sinhronizacija je namenjena uporabi v okoljih brez povezave, na primer pri potovanju s prenosnim računalnikom.
- Internetna sinhronizacija se uporablja za sinhroniziranje replik v okolju brez povezave, kjer je na voljo konfiguriran internetni strežnik.

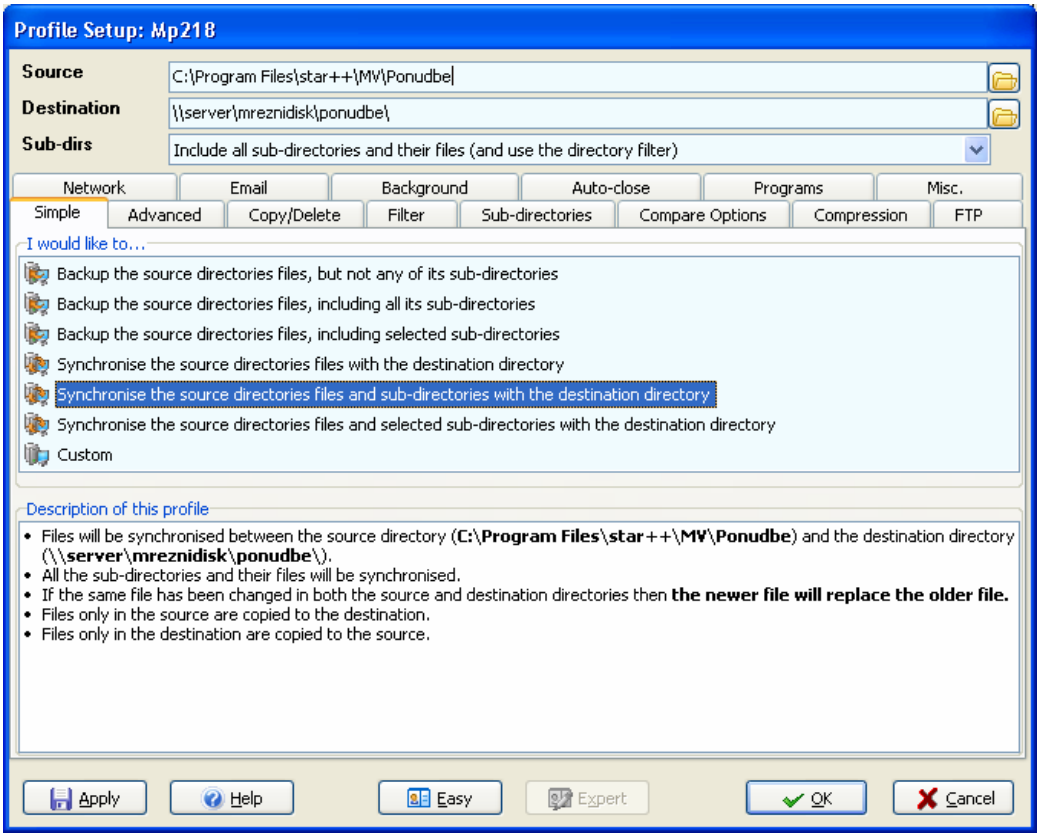
5.3.3.2 Sinhronizacija datotek in map

Program za sinhronizacijo datotek SyncBack[5] nima plačljive licence in kot tak ustreza zadanemu cilju čim manjših dodatnih stroškov za licence. Poleg zanesljive sinhronizacije vsebuje tudi lastnost, ki omogoča klicanje profila iz ukazne vrstice, kar pomeni, da je lahko sinhroniziranje popolnoma avtomatizirano s pomočjo skript in urnika opravil v operacijskem sistemu Windows XP[6]. Program omogoča tudi kreiranje rezervnih kopij, tako da ne bo potrebno izdelati posebne rešitve.

Za sinhronizaranje datotek z mrežnim diskom je bil definiran profil sinhronizacije, ki omogoča sinhronizacijo datotek in celotne imeniške strukture s ciljno imeniško strukturo na omrežnem disku. V primeru sprememb iste datoteke na lokalnem disku in na mreži, bo obveljala najnovejša sprememba. V izogib izgubi podatkov, ko bi na primer dva uporabnika na lokalnem disku spreminjata isto datoteko, smo določili pravilo, da lahko samo pripravljalec ponudbe spreminja ponudbo.

[5] Program Syncback razvija in trži angleško podjetje 2BrightSparks.
[6] Opercijski sistem Windows XP razvija in trži podjetje Microsoft.

Slika 5: Program za sinhroniziranje datotek SyncBack



Vir: Lasten vir.

5.4 ITERACIJE

Namen iteracij je seznanitev uporabnika z delnimi rešitvami in pridobivanje informacij na podlagi uporabnikovega izkustva z uporabo prikazane rešitve. Informacije pridobljene na ta način so bistvenega pomena za nadaljni razvoj aplikacije, ker z njimi zmanjšamo tveganje neustreznega razvoja.

5.4.1 OPIS PRVE ITERACIJE

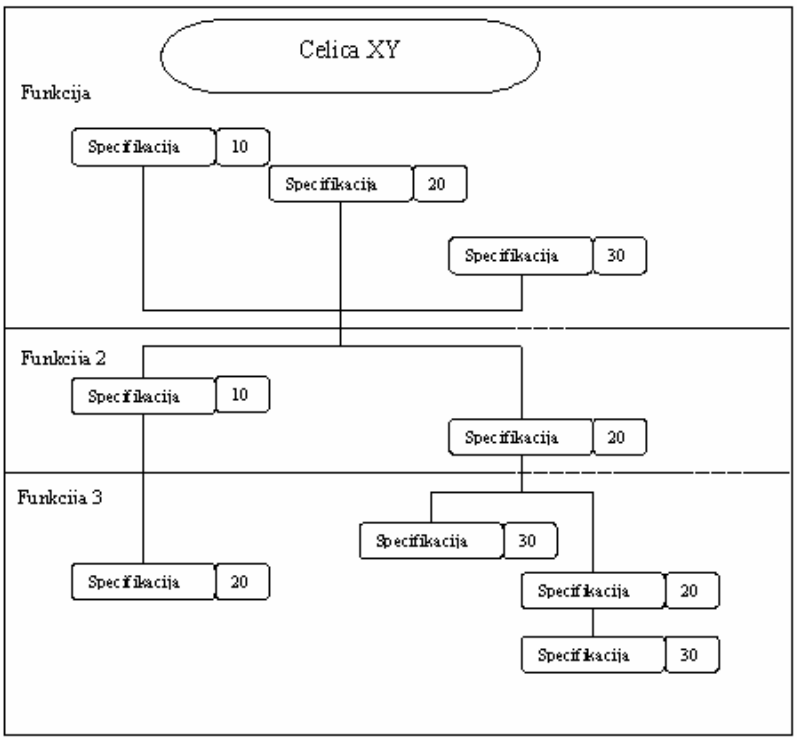
V prvi iteraciji sem se osredotočil na izdelavo baze podatkov in postopka izračuna cene posamezne celice. Zaradi omejenega prostora bom navedel samo bistvene odločitve pri spremembi prototipa in nove predloge

5.4.1.1 Predstavljena rešitev

Pri pripravi ponudbe za izdelek družine "modularni SN stikalni blok tipa SN6" se pripravljalec najprej osredotoči na stikalno omaro, kjer so nameščene celice. Ceno, funkcije

in specifikacijo funkcije celic se določijo na podlagi vnaprej pripravljenih dreves odločitev, ki jih pripravi proizvajalec izdelkov. Preprost primer za izračun celice je prikazan na sliki 6.

Slika 6: Prikaz izračuna cene posamezne celice



Vir: Lasten vir.

Ceno in speficikacije določamo po vrstnem redu funkcij, ki se vrstijo od zgoraj navzdol. Vsaka specifikacija ima svojo točno določeno ceno. Cena je lahko tudi 0. Specifikacije se izberejo na podlagi določenih poti (odločitev), ki so vnaprej definirane. Izhod procedur predstavlja skupna cena vseh izbranih specifikacij in seznam izbranih funkcij s pripadajočimi specifikacijami.

Predlagana rešitev problema je bil čarovnik, ki vodi uporabnika skozi določen postopek. Na podlagi izbora kreira seznam, izračuna ceno in shranjuje izbore, ki jih uporabi za nadaljne operacije.

5.4.1.2 Pripombe

Uporabnik se z rešitvijo ni strinjal, saj z uporabo čarovnika izgubi pregled nad že izbranimi specifikacijami. Ker je vztrajal pri tem, da mora imeti pred sabo sliko odločitvenega drevesa in ker so ta drevesa lahko velika tudi do dve strani (A4 format), smo se odločili, da za naslednjo rešitev uporabimo kar Excel sam.

5.4.1.3 Odprava napak in načrtovanje na podlagi pripomb

Nova rešitev obdrži v popolnosti obstoječa drevesa odločitve. Kreirana sta dva makroja. Prvi označi izbrano specifikacijo. Izbrana specifikacija se obarva v rdečo barvo. Podatki o ceni, pripadajoči funkciji in tipu specifikacije se izpišejo desno od območja tiskanja. Celotna vrstica se obarva rumeno, tako da je jasno, katera funkcija je bila izbrana. Desno zgoraj se avtomatično izračunava skupna cena trenutno izbranih specifikacij. Drugi makro resetira izbor in povrne vrstico v prvotno stanje. Posebna funkcija pred izhodom iz Excela pregleda izbor in kreira zahtevane izhode in jih izvozi v aplikacijo za nadaljno obdelavo. Datoteka se shrani v OLE polje[7] baze podatkov.

5.4.2 OPIS DRUGE ITERACIJE

V drugi iteraciji je bilo predstavljeno jedro programske rešitve. Manjkali so izpisi in iskalnik ponudb. Najbolj smo se posvetili izračunu cen celic, komercialnim pogojem in izgledu ponudbe.

5.4.2.1 Predstavljena rešitev

Uporabnik preko posebne maske določi oznako ponudbe (npr: 0001-MV-MP-04), ki na prvih štirih mestih vsebuje zaporedno številko ponudbe, na naslednjih štirih mestih oznako MV, ki opisuje srednjenapetostni produkt, od devetega do vključno dvanajstega mesta začetnice pripravljalca ponudbe in na zadnjih treh mestih tekoče leto. V primeru ustreznosti določene oznake se odpre vnosna maska za vnos osnovnih podatkov ponudbe (glej sliko 7). Na osnovni maski določimo datum prejema povpraševanja, datum oddaje ponudbe, ime projekta za katerega je ponudba izdelana, komentar objekta, nosilca projekta, pripravljalca ponudbe ter stranko in njeno kontaktno osebo. V okvirju "podrobnosti" določimo komercialne pogoje, ki veljajo za vse sklope v ponudbi ter splošne pogoje, ki prav tako veljajo za vse sklope. Ponudba je lahko sestavljena iz več sklopov, katerim priredimo en izdelek. V primeru izbora izdelka "Modularni stikalni blok tipa SN6" se odprejo podrobnosti tega izdelka. Na tej maski najprej odpremo excelovo datoteko, ki kaže postopek konfiguriranja stikalne omare in na podlagi konfiguracije se izračuna skupna cena ogrodja stikalne omare. Glavni del stikalne omare so celice, ki imajo podoben način izračuna kot stikalna omara. Ko se določi konfiguracija celic in stikalne omare, se uporabnik vrne na osnovno masko, kjer lahko izbere še izdelek Minera in Trihal.

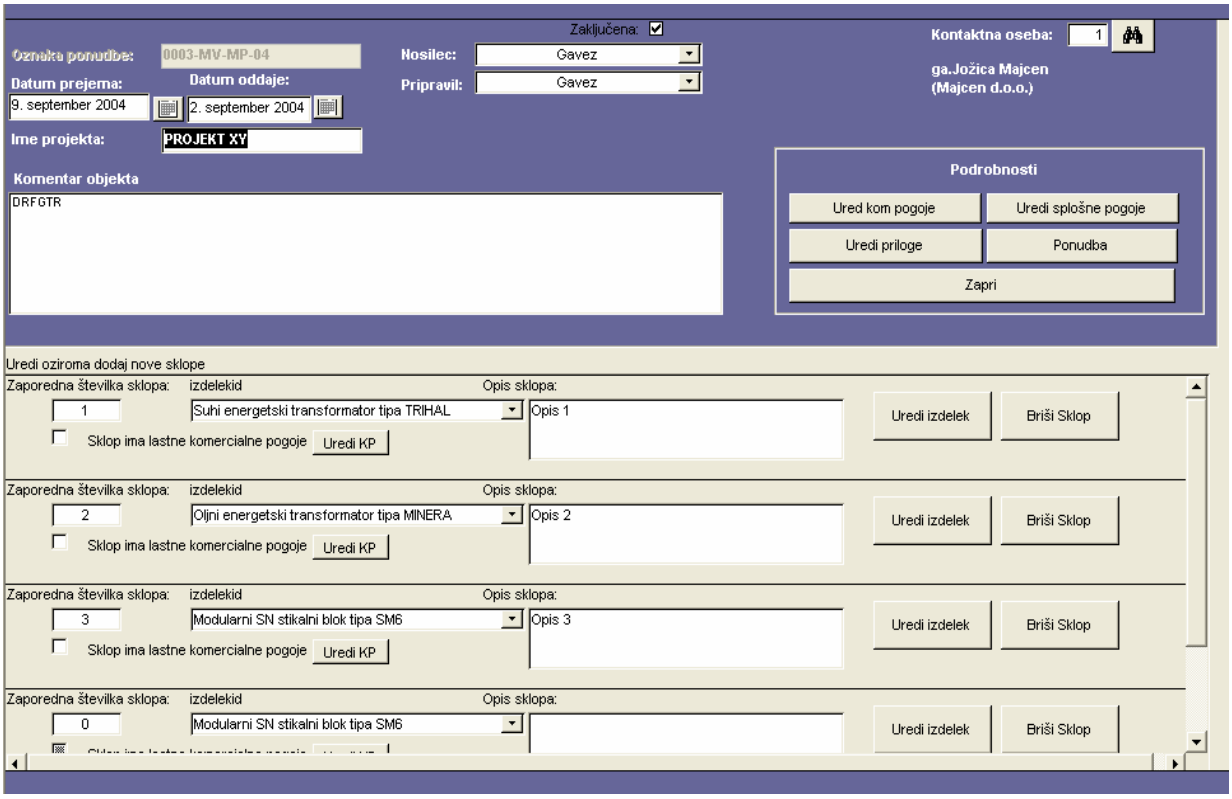
V primeru izbora izdelka Minera se odprejo podrobnosti produkta. Na tem nivoju se določi cena, količina in karakteristike (moč, napetost in ostale specifikacije), ki jih lahko uporabnik pridobi iz posebne maske, ki vsebuje iskalnik, na podlagi katerega lahko uporabnik išče po različnih kriterijih (moč, napetost in material), ki določajo osnovno ceno izdelka.

---

[7] OLE je Microsoftova tehnologija za integracijo programov. Namenjena je prenašanju informacij med programi. V OLE polja lahko shranjujemo različne formate datotek (npr. tekstovne datoteke, slike, preglednice itd.)

Izbor izdelka Trihal ponuja zelo podobne možnosti kot izdelek Minera, le da tukaj med podrobnostmi obstaja več vrst cen (minimalna, zaželjena itd.), ki pripravljalca ponudbe in nosilca projekta usmerjajo pri določitvi cene.

Slika 7: Vnosna maska za kreiranje ponudb



Vir: Lasten vir.

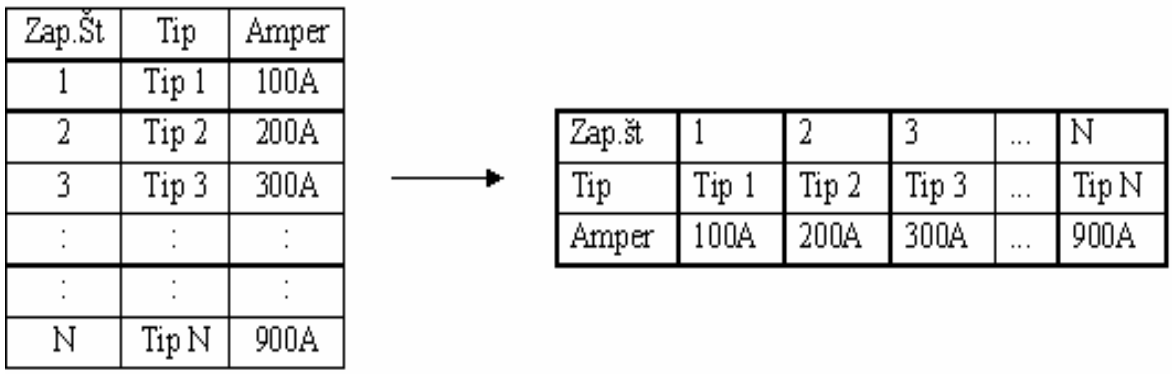
5.4.2.2 Pripombe

Uporabnik je pri testiranju prototipa navedel tri večje pripombe.

- Pri izračunu cene posameznih celic smo pregledali delovanje sistema, barve, izračune in prenos podatkov. Pripombe so bile na izbor specifikacij, kjer je bila predlagana uporaba enega gumba in ne standardne kombinacije za klic makroja (CTRL + *tipka*).
- Pri izpisu ponudb je bi podan predlog, da naj obstaja možnost, ki bo zagotavljala, da se komercialni pogoji lahko določajo tudi za vsak sklop posebej in ne samo za ponudbo kot celoto. Ta posebnost se sicer uporabi v manj kot desetih odstotkih poslanih ponudb.
- V izpisu specifikacij SN celic naročnik zahteva, da se tabela obrne, ker taka postavitev omogoča stranki, da že iz tabele razvidi konfiguracijo stikalne omare.
- Priloge predstavljajo poseben primer. V začetni fazi niso bile omenjene. Uporabnik želi imeti sistem, ki bo shranjeval in uredil priloge. Priloge so v različnih formatih, navadno pa je skica izdelana z uporabo Autocada. V drugi fazi se morajo priloge

kopirati na omrežni disk. Ostalim uporabnikom pa mora biti omogočeno, da avtomatično poberejo priloge na lokalni datotečni sistem. Poleg tega mora imeti uporabnik možnost odpiranja in s tem pregledovanja datotek iz aplikacije same.

Slika 8: Sprememba orientiranosti tabele specifikacij SN celic v izpisu ponudbe



Vir: Lasten vir.

5.4.2.3 Odprava napak in načrtovanje na podlagi pripomb

Pripombe uporabnika so bile smiselne in upoštevane, zato so bile do naslednje iteracije pripravljene spodaj opisane rešitve.

Problem iz naslova prve pripombe smo rešili z uporabo gumba F4 in F5, katerima se ob odpiranju excelove datoteke priredi koda obstoječih makrov. Uporabnik tako s pritiskom na gumb F4 izbere določeno funkcijsko specifikacijo, program na podlagi izbora prišteje ceno specifikacije skupni ceni izdelka. S pritiskom na gumb F5 uporabnik razveljavi spremembe na določeni funkcijski specifikaciji.

Rešitev na pripombo, ki zahteva, da mora programska rešitev omogočiti, da uporabnik vnese komercialne in splošne pogoje za vsak sklop posebej, je bila razvita na slednji način:

- v tabelo, ki obravnava sklop so bila dodana dodatna polja, ki opredeljujejo skupno ceno sklopa, popust in odločitveno polje, kjer uporabnik določi, da komercialni pogoji veljajo samo za določen sklop ali celotno ponudbo;
- kreirana je bila dodatna vnosna maska za vnos komercialnih in splošnih pogojev;
- kreiran je bil dodaten izpis ponudbe, v katerem so po sklopih prikazani komercialni in splošni pogoji;
- spremenjena je bila osnovna maska za vnos nove ponudbe.

Tretja pripomba je zahtevala obrat tabele v izpisu ponudbe, kot jo prikazuje slika 8. Rešitev ne vsebuje samo enostavne poizvedbe, temveč tudi programsko kodo, saj programski jezik SQL ne omogoča preproste transformacije tabele.

Rešitev problema prilog je prinesla korenite spremembe. Dodajanje novih OLE polj v bazo bi prineslo preveliko povečanje baze, ki bi jo Microsoft Access 2000 težko zmogel zanesljivo obvladovati. Odločili smo se, da vse datoteke, ki jih aplikacija uporablja za izdelavo ponudbe, kopira na lokalni disk. Od tam pa se podatki ob vsaki prijavi uporabnika v računalnik, če je le-ta povezan na mrežo, sinhronizirajo na mrežo. OLE polja so bila iz baze odstranjena. Namesto njih so bila uporabljena tekstovna polja, ki shranjujejo informacijo o poti do posamezne datoteke.

Rešitev problema se je razdelila v tri večje naloge:

- preoblikovanje baze podatkov;
- izdelava zaneseljivega sistema kopiranja datotek in kreiranja map na lokalnem disku;
- najti in testirati ustrezen program za sinhronizacijo datotek in map z mrežnim diskom.

5.4.3 OPIS TRETJE ITERACIJE

V tretji iteraciji je bila uporabniku predstavljena v predhodnem poglavju omenjena rešitev. Dodana je vhodna maska, preko katere se uporabnik prijavi v aplikacijo. Glavnemu meniju je bil dodan iskalnik, vnaprej definirani izpisi in nastavitve.

5.4.3.1 Predstavljena rešitev

Predstavljena rešitev je v tej iteraciji obsegala pet večjih tem:

- rešitev prilog in datotek,
- iskalnik,
- vnaprej pripravljeni izpisi,
- varnost,
- distribucija.

5.4.3.1.1 Rešitev prilog in sinhronizacija datotek

Za sistem prilog je bila izdelana posebna forma s pripadajočo tabelo, ki vsebuje informacije o zaporedni številki priloge, opisu priloge in lokacije na lokalnem disku. Priloge se kopirajo na poseben direktorij, ki je poimenovan po oznaki ponudbe, od kjer se s programom Backsync sinhronizirajo datoteke na mrežni disk. Program je opisan v poglavju 5.3.3.2. Ostali uporabniki sinhronizarajo podatke ob prijavi in odjavi v operacijski sistem. Naročniku so bile predstavljene spremembe v delovanju aplikacije tudi v primeru srednjenapetostnih celic, kljub temu da so na prvi pogled spremembe neopazne.

5.4.3.1.2 Iskalnik

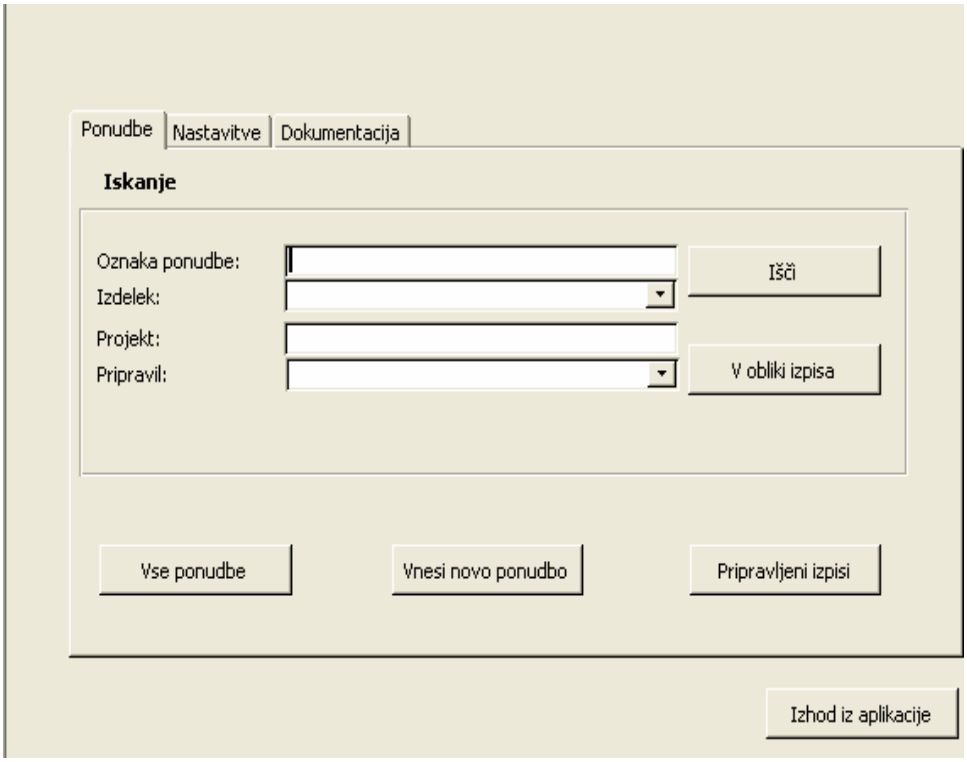
Iskalnik omogoča poizvedbe na podlagi štirih kriterijev. Iskanje po oznaki ponudbe, izdelku, po pripravljalcu in projektu. Izpis rezultatov je mogoč na dva načina: tiskanju

prijaznejši v obliki poročila ali v obliki forme, ki tudi omogoča nadaljno raziskovanje in popravljanje izdelane ponudbe.

5.4.3.1.3 Izpisi

Vnaprej pripravljenih izpisov je sedem. Posamezen izpis poda osnovno informacijo o izdani ponudbi (čas oddaje ponudbe, pripravljalec ponudbe, oznaka itd.), pa tudi podrobnejše informacije o popustih, cenah in posameznih izdelkih. Izpisi so osnovni in so bodo v nadaljevanju odvisno od potreb dopolnjevali. Izpisi so razvrščeni po datumu oddaje ponudbe, kar pomeni, da ima uporabnik možnost pregleda zgodovine izdanih ponudb. Med drugim: kaj je določena stranka naročila pri podjetju; kakšna je bila ponujena cena; kolikšen je bil popust.

Slika 9: Osnovna maska preko katere uporabnik dostopa do različnih funkcij aplikacije



Vir: Lasten vir.

5.4.3.1.4 Varnost

Varnost aplikacije in predvsem podatkov je opredeljena na večih nivojih. Delimo jo na dva večja sklopa.

- Nepooblaščeno dostopanje do podatkov:
  - posamezniki izven podjetja,
  - posamezniki znotraj podjetja.
- Možnost izgube podatkov:

- poškodba diska ali okužba z virusom,
- poškodba mrežnega diska,
- nesreča v stavbi (na primer požar), ki bi uničila vse podatke.

Konkurenčni boj podjetij na trgu je hud in vsakršna informacija o popustih in maržah je za konkurenco ali kupca več kot dobrodošla, zato na nivoju "podjetje-svet" pred vdori v omrežje podjetja varuje zmogljiv požarni zid. Na tem nivoju je bilo podjetje že relativno dobro zavarovano pred nepooblaščenim dostopanjem do internih informacij. Vsi posamezniki v podjetju nimajo pravic za dostop do vseh informacij. V ta namen obstaja zaščita na nivoju operacijskih sistemov in strežnika (mrežni disk), ki ščiti posamezne oddelke in posameznike pred nepooblaščenim dostopom do podatkov. Aplikacija sama ne ponuja popolne zaščite, ampak neveščim oteži dostop do podatkov. Uporabnik mora že na nivoju dostopa v operacijski sistem z ustrezno politko menjave in določevanja gesel poskrbeti za varnost podatkov.

Izgubo podatkov zaradi malomarnosti, nesreče in viruse lahko preprečimo na različne načine. Najpomembnejše pa je shranjevanje podatkov na različne lokacije, od koder jih lahko na relativno lahek način restavriramo nazaj na lokalni disk. V primeru poškodbe lokalnega diska ali okužbe z virusi, ki je pripeljala do popolne izgube podatkov, lahko podatke in aplikacijo obnovimo iz mrežnega diska. V primeru izgube podatkov (brisanje, okvara) na mrežnem disku podatke obnovimo iz kasete. Vse datoteke na mrežnem disku se vsak teden arhivirajo na kaseto, ki jo nato hranimo izven oddelka, tako da se v primeru požara ali druge naravne nesreče podatki lahko restavrirajo.

5.4.3.1.5 Distribucija

Aplikacija se je v tej iteraciji pričela distriburirati s programom InnoSetup[8], za katerega podobno kot za BackSync ne potrebujemo plačljive licence. Program omogoča enostavno kreiranje skripte, ki kreira čarovnika za inštalacijo aplikacije in vseh datotek in direktorijev, ki spadajo zraven. Omogoča tudi enostavno deinštalacijo programa, kar uporabniku omogoča lažjo menjavo verzij. Še posebej je to koristno v času testiranja, ko podatki znotraj baze niso tako pomebni kot so v produkcijskem okolju.

5.4.3.2 Pripombe

Naročnik je predlagal, da bi izdelali sistem, ki bi omogočal izbor predlog za izračun zahtevnejših postopkov, ki se nahajajo v excelovih datotekah. Na podlagi izkušenj je ključni uporabnik ugotovil, da polovica poslanih ponudb pri produktih srednje napetosti vsebujejo standardne konfiguracije celic in stikalne omare.

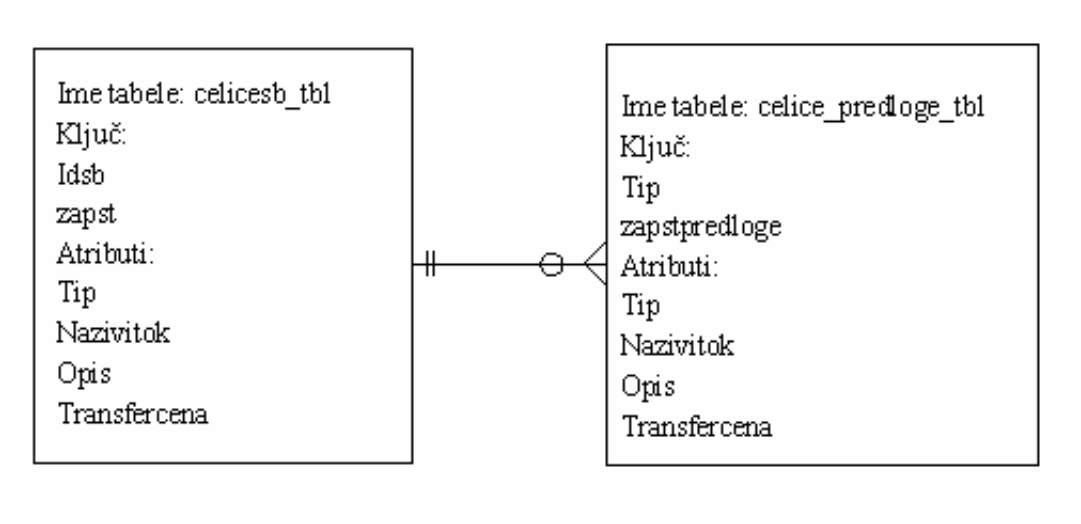
5.4.3.3 Odprava napak in načrtovanje na podlagi pripomb

Excelove datoteke, ki vsebujejo postopke izračuna posameznih cen, delimo na dva sklopa. Prvi sklop na nivoju celotne stikalne omare vsebuje eno datoteko. Za shranjevanje podatkov

[8] Program je izdelal Jordan Russell.

za sistem predlog je bila kreirana nova tabela "sb_predloge_tbl", ki vsebuje informacije o imenu predloge in poti do predloge. Drugi sklop zavzema celice, ki jih vsebuje stikalna omara. Različnih celic in s tem datotek je 20. Obstoječi tabeli, ki vsebuje podatke o posamezni celici, smo priredili novo tabelo ("celice_predloge_tbl") v povezavi 1:N, ki vsebuje podobne informacije kot v primarni tabeli (sestavljeni ključ, opis in pot do datoteke oziroma predloge, nazivni tok, transfer ceno, opis). Direktorijski strukturi na lokalnem disku in mreži smo dodali nov poddirektorij z imenom predloge, kamor se shranjujeo nove predloge, ki jih kreira uporabnik.

Slika 10: Razmerje med tabelo "celicasb_tbl" in "celica_predloge_tbl"



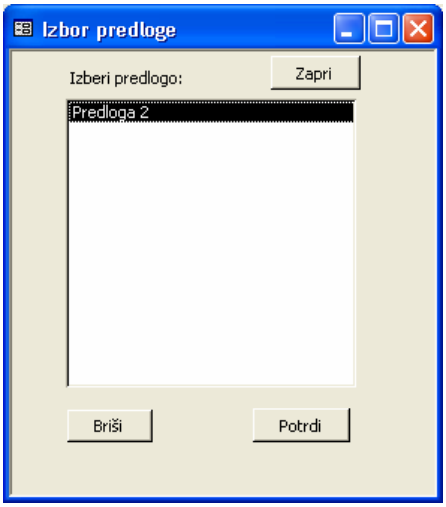
Vir: Lasten vir.

5.4.4 OPIS ČETRTE ITERACIJE

5.4.4.1 Predstavljena rešitev

Uporabniku je poleg standardne (prazne) izbire postopka izračuna cene določene celice ponujena tudi možnost izbire predloge (glej sliko 11). Predlogo uporabnik kreira ob izdelavi ponudbe in če meni, da bo določen postopek koristen v prihodnosti, ga shrani kot prilogo. S tem ne shrani le datoteke ampak tudi informacije, ki jih pridobi v postopku izračuna cene (na primer transfer ceno, seznam specifikacij in nazivni tok).

Slika 11: Maska, ki omogoča izbor vnaprej definirane predloge



Vir: Lasten vir.

Naročniku je bil predstavljen sistem replikacij in način sinhronizacije podatkov in datotek. Ponujene so mu bile tri možnosti:

- Sinhronizacija ob prijavi in odjavi v osebni računalnik (prenosnik)
  V urnik opravil se nastavi program "Preveri povezanost na mrezo.bat". Nastavimo ga tako, da se sproži ob prijavi uporabnika v operacijski sistem. Program najprej preveri povezanost uporabnika na mrežo. V primeru povezanosti pokliče vnaprej pripravljen profil programa Backsync, kjer je definiran način sinhronizacije datotek, izvorni (lokalni) in ponorni (mrežni) direktorij. Po sinhronizaciji datotek se sproži program "synch.mdb", ki sinhronizira bazi podatkov (na mreži in lokalno).
- Sinhronizacija ob prijavi in odjavi v aplikacijo
  Ob sinhronizaciji ob prijavi in odjavi v aplikacijo se ob uspešnem logiranju v sistem preveri, ali je uporabnik povezan v mrežo. V primeru povezanosti se sproži program synch.bat, ki enako kot je opisano v zgornji točki, kliče profil programa Backsync. Po sinhronizaciji datotek se sinhronizira še baza podatkov.
- Ročna sinhronizacija iz aplikacije
  Pri ročni sinhronizaciji iz aplikacije sta v posebni maski z imenom nastavitve pripravljena dva gumba. Prvi kliče program synch.bat in s tem sinhronizira datoteke, drugi pa sproži proceduro z imenom sinhroniziraj, ki sinhronizira repliko na lokalnem disku z repliko na mrežnem disku.

6 KONČNA REŠITEV

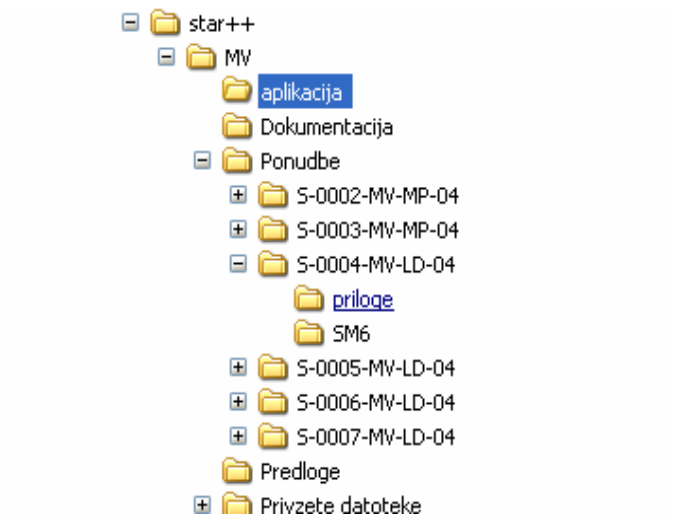
Namen poglavja je opisati končno različico programske rešitve, saj se je aplikacija skozi proces iteracij spreminjala in dopolnjevala. Na podlagi novih spoznanj in proučitve otipljive rešitve je lahko uporabnik podal predloge novih funkcionalnosti, odkrival napake

in predlagal izboljšave obstoječih rešitev. V sledečem poglavju bo predstavljena vpetost aplikacije v okolje podjetja ter funkcionalna dekompozicija aplikacije same.

Programsko rešitev smo poimenovali "Star++ MV". Za to ime smo se odločili, ker se v podjetju uporablja ime "Star++" za programske rešitve razvite znotraj podjetja.

Star++ MV je bil zgrajen z namenom, da lahko uporabnik uporablja podatke tudi kadar ni fizično prisoten v podjetju. Omogočiti mu je bilo potrebno delo doma oziroma na terenu. Za zagotovitev te funkcionalnosti je bila programska rešitev fizično razdeljena na dve različni lokaciji. Prva se nahaja lokalno na uporabnikovem računalniku, druga pa na omrežnem disku. Sinhronizacija med uporabnikom in mrežnim diskom zagotavlja na nivoju podatkovne baze sistem replikacij (opisan v poglavju 5.3.2), na nivoju datotečnega sistema pa program BackSync (opisan v poglavju 5.3.3.2.). Tako lokalno kot tudi na mrežnem disku obstajata enaki imeniški strukturi (glej sliko 12) in podatkovni bazi (repliki), ki jih sinhroniziramo ob vsaki prijavi in odjavi v operacijski sistem. Sistem aplikacije na mrežnem disku je v funkciji agenta, ki sprejema spremembe posameznega uporabnika in spremembe v procesu sinhronizacije izmenja z ostalimi uporabniki. Z običajno prisotnostjo v podjetju ima uporabnik vse potrebne informacije za vsakdanje delo.

Slika 12: Prikaz imeniške strukture



Vir: Lasten vir.

Programska rešitev "Star++ MV" je fizično razdeljena na uporabniški vmesnik, ki se nahaja v datoteki "Medium Voltage.mdb" ter na podatkovno bazo ("Medium Voltage_be.mdb"). Preko uporabniškega vmesnika lahko uporabnik preko vhodne maske (glej sliko 8) dostopa do ponudb, nastavitev aplikacije in dokumentacije.

V okviru ponudb ima uporabnik možnost iskanja in urejanja že izdelanih ponudb, kreiranja novih ponudb ter izpisa vnaprej pripravljenih poročil. Pri kreiranju nove ponudbe se najprej

določi oznaka ponudbe, ki je sestavljena iz zaporedne številke ponudbe, oznake MV, inicialk pripravljalca ponudbe ter tekoče leto. Po uspešni določitvi oznake ponudbe uporabnik vnese osnovne podatke o projektu (npr. nosilec projekta, datum prejema povpraševanja itd.). Trenutno lahko uporabnik izbira med tremi različnimi družinami izdelkov. Vsaka ima svoje posebnosti in ima v aplikaciji drugačen pristop.

Najzapletenejša je obravnava modularnega stikalnega bloka tipa SN6. V datotekah excelovega formata so določene sheme, preko katerih se na podlagi vnaprej določenih pravil sestavi posamezne celice, ki skupaj predstavljajo stikalno omaro. Uporabnik najprej naloži (glej prilogo C) in odpre excelovo privzeto datoteko izbranega tipa celice in s pomočjo definiranih makrojev izbere in označi funkcije in funkcijske specifikacije, ki bodo sestavljale posamezno celico. Aplikacija na podlagi izbora izračuna skupno ceno, shrani izbrane funkcije in funkcijske specifikacije. Excelova datoteka se kopira na natančno določen direktorij, od koder jo lahko uporabnik na preprost način odpre ali izbriše preko uporabniškega vmesnika. Isto datoteko lahko kasneje natisne in jo uporabi za naročilo stikalnega bloka pri matičnem podjetju.

Druga družina izdelkov srednje napetosti se imenuje Trihal. Pri tej družini v ponudbi določimo samo naziv in opis posamezne specifikacije. V pomoč pri določitvi cene je maska (glej prilogo D), v kateri določimo specifikacije, na podlagi katerih aplikacija ponudi različne cene (minimalna, zaželjena itd.).

Tretja družina izdelkov srednje napetosti je Minera. Pri tej družini izdelkov na enak način kot pri družini Trihal določimo specifikacije izdelka (glej prilogo E). V pomoč pri določitvi cene pa je prav tako posebna maska, v kateri se na podlagi izbranih kriterijev izpiše zaželjena cena.

K stroškovni učinkovitosti orientirana podjetja (dobavitelji in kupci) najbolj zanimajo komercialni in splošni pogoji. Komercialne in splošne pogoje lahko opredelimo skupaj za celotno ponudbo ali pa posamezno po različnih družinah izdelkov. V komercialnih pogojih določimo skupno ceno produktov, popust, dobavni rok in ostale pogoje.

Na podlagi vnešenih in izračunanih podatkov lahko ponudbo prikažemo na zaslon, natisnemo ali pa jo izvozimo v tekstovno datoteko.

Posebna funkcionalnost programske rešitve so priloge. Ponavadi jih sestavljajo razne skice in načrti ponujenih izdelkov. Priloge se preko uporabniškega vmesnika kopirajo na lokalno direktorijsko strukturo aplikacije (glej sliko 12), od koder se sinhronizirajo na mrežni disk in so tako enako kot vsi ostali podatki dostopne ostalim uporabnikom.

Pregled nad oddanimi ponudbami dobi uporabnik preko vnaprej pripravljenih izpisov. Izpisi omogočajo ustrezno povratno informacijo o oddanih ponudbah. Med drugim tudi podatke o tem, kakšne popuste je dobivalo v preteklosti točno določeno podjetje.

V razdelku nastavitve ima uporabnik možnost nastavljanja šifrantov, spreminjanja gesla, proženje sinhronizacije in spreminjanja imena strežnika ter omrežnega diska. Šifranti obsegajo seznam nizkonapetostnih celic, strank, kontaktnih oseb in uslužbencev podjetja. Uporabnik ima možnost dodajanja, brisanja in spreminjanja posameznih zapisov v šifrantih.

7 SKLEP

Razvoj programske rešitve zahteva mnogo časa in kreativnega mišljenja, kakor tudi doslednosti in sposobnost pridobivanja novega znanja. Izdelava diplomske naloge in programske rešitve je pokazala, da metoda prototipa da uporabniku oziroma naročniku velik vpogled v rešitev in možnost stalnega dajanja pripomb, novih informacij in možnih rešitev. Prav tako razvojnik hitreje pridobi povratno informacijo o ustreznosti določene rešitve. Z vsem naštetim se mnogo zmanjša tveganje zgrešene investicije.

Programska rešitev je sledila ciljem, ki so bili zadani. Zmanjšala je čas izdelave ponudbe, omogoča delo tudi v času, ko uporabnika ni v pisarni, uporabnik lahko dobi povratno informacijo o preteklih izdanih ponudbah. Ob vseh omenjenih dosežkih pa tudi ni bilo potrebno dokupiti novih licenc.

Programska rešitev ima še kar nekaj možnosti dodelave in dodajanja funkcionalnosti, saj so bile ob razvoju vzete v zakup morebitne spremembe v prihodnosti. Slabost aplikacije je v številu potrebnih operacij, ki jih mora uporabnik izvesti, preden lahko natisne ponudbo, zato bo nadaljna avtomatizacija prednostna naloga nadaljnega razvoja. Ena izmed možnosti vsebinske nadgradnje programske rešitve so naročila sprejetih ponudb, saj je ponudba osnova za pripravo naročila pri matičnem podjetju. Čas pa bo pokazal, koliko bo programska rešitev dejansko pripomogla k večji produktivnosti in s tem večji donosnosti podjetja.

Za konec pa še citat iz odstavka o agilni metodoligiji (Ambler, 2001, str. 5): "Enostavnost procesa je ključ do uspeha. Kako razviti kakovostno programsko opremo, pri tem pa opraviti čim manj stranskih nalog in izdelkov? Proces mora biti ravno dovolj zahteven. Raje malo manj kot pa preveč. To nikakor ne pomeni, da moramo aktivnosti izvajati površno ali pomanjkljivo. Nasprotno, naše delo mora biti opravljeno kvalitetno, paziti pa moramo, da ni po nepotrebnem komplicirano. Pot do enostavnosti je v resnici težka, saj je včasih preprosteje pustiti stvari kompleksne, kot jih pa poenostaviti."

8 LITERATURA

1. Avison David, Fitzgerald Guy: Information systems development (Methodologies, Tehniques and tools). Boston : McGraw-Hill Companies, 1996. 563 str.
2. Bajec Marko, Krisper Marjan: Agilne metodologije razvoja informacijskih sistemov. Uporabna informatika, Ljubljana, 11(2004), 2, str. 68-76.
3. Connoly Thomas, Begg Carolyn: Database systems (A practical approach to design, implementation and management). Harlow, VB : Pearson Education Limited, 1999. 1211 str.
4. Grad Janez, Jaklič Jurij: Baze podatkov. Ljubljana : Ekonomska fakulteta, 1996. 254 str.
5. Gradišar Miro, Resinovič Gortan : Informatika v poslovnem okolju. Ljubljana : Ekonomska fakulteta, 2001. 508 str.
6. Hvala Davor: Obliž za projekt. Monitor, Ljubljana, 14(2004), 9, str. 12-15.
7. Kolar Džangir : Razvoj programske rešitve za zaposlovanje novih kadrov. Diplomsko delo. Ljubljana : Ekonomska fakulteta, 2003. 52 str.
8. Kovačič Andrej: Informatizacija poslovanja. Ljubljana : Ekonomska fakulteta, 1998. 214 str.
9. Litwin Paul, Getz Ken, Gilbert Mike: Access 2000 Developer's Handbook, Volume 2 : Enterprise Edition. Alameda, ZDA : SYBEX Inc., 2000. 808 str.
10. Mason David, Willcocks Lesli: System analysis, system design. Oxford : Alfred Waller, 1994. 337 str.
11. Prague Cary, Irwin Michael: Microsoft Access 2000 Bible. Foster City, ZDA : IDG Books Worldwide Inc., 1999. 1221 str.
12. Pretnar, Barbara: Razvoj informacijskega sistema v oddelku za trženje. Diplomsko delo. Ljubljana : Ekonomska fakulteta, 2003. 43 str.
13. Solina Franc: Projektno vodenje razvoja programske opreme. Ljubljana: Fakulteta za računalništvo in informatiko. 1997. 212 str.

9 VIRI

1. Kevan Lyons: The Agile Approach. URL:http://www.agilemodeling.com/essays/agileDocumentation.htm, 1.10.2004.
2. Scott Ambler: Agile Documentation. [URL:http://www.agilealliance.org/articles/TheAgileApproach.pdf], 1.10.2004.

PRILOGE

PRILOG A: SLOVARČEK TUJIH IZRAZOV IN NJIHOVA RAZLAGA

API (Application Programming Interface) - knjižnica nizkonivojskih funkcij za okolje Windows

SQL (Structured Query Language) - strukturirani povpraševalni jezik, generaliziran jezik do mere, da obsega sredstva za definicijo podatkovnih struktur, specifikacijo integriranih omejitev, podeljevanje prestopnih dovoljenj in zaščito celovitosti baze podatkov

VBA (Visual Basic for Applications) - jezik za programiranje aplikacij v Accessu

OLE objects (Object Linking and Embedding) - tip polja, ki lahko vsebuje slike, zvočne datoteke, grafe in video datoteke

Make table query - poizvedba v MS Accessu, ki kreira tabelo

MV (Medium Voltage) - srednja napetost

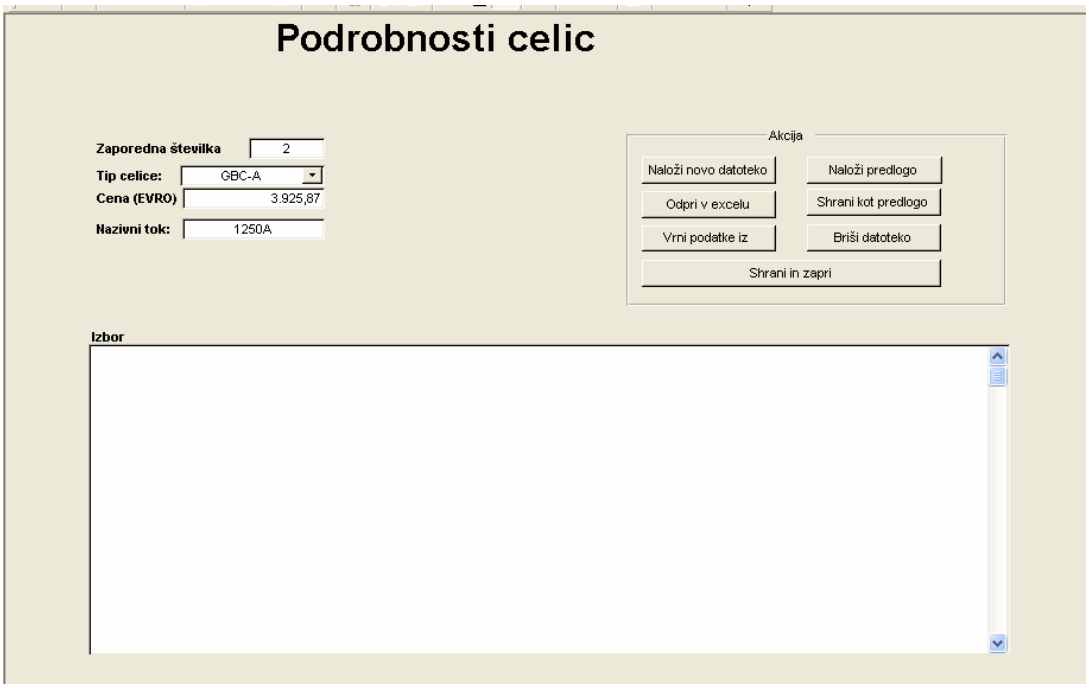
PRILOGA B: PRIMER UPORABLJENE API FUNKCIJE

```
Public Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal
hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters
As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
Const SW_SHOWNORMAL = 1
Const MAX_PATH = 260

Public Sub odpridatoteko(pot As String, hwdnn As Long)
ShellExecute hwdnn, vbNullString, pot, vbNullString, "C:\", SW_SHOWNORMAL
End Sub
```

PRILOGA C: MASKA ZA UREJANJE PODROBNOSTI CELIC

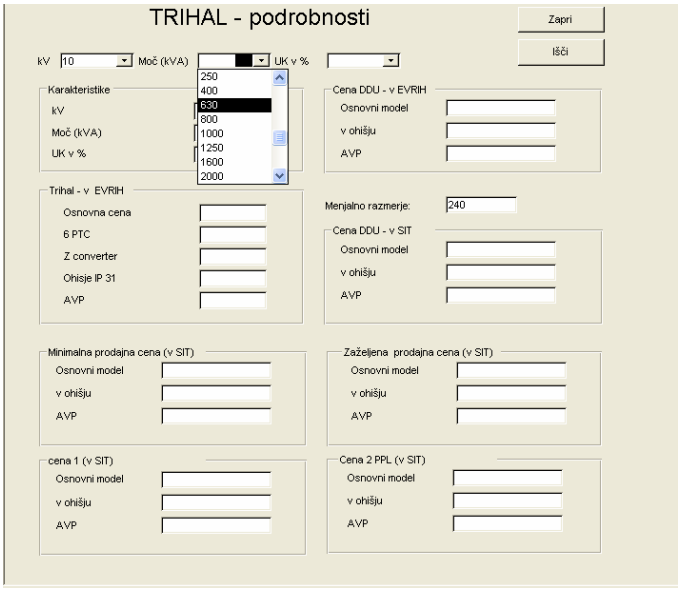
Slika 1: Prikaz uporabnikovega pogleda pri urejanju podrobnosti celic



Vir: Lastna izdelava.

PRILOGA D: MASKA ZA UREJANJE PODROBNOSTI PRODUKTA TRIHAL

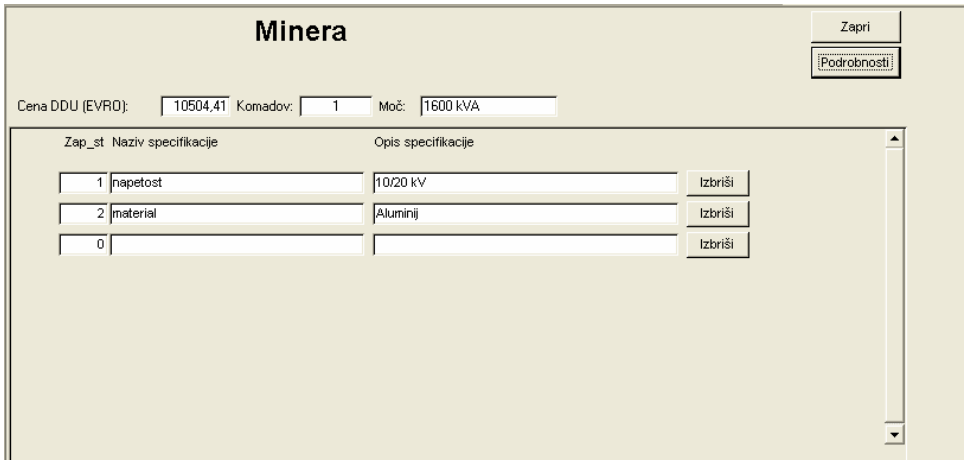
Slika 2: Maska za urejanje podrobnosti produkta TRIHAL.



Vir: Lastna izdelava.

PRILOGA E: MASKA ZA UREJANJE PODROBNOSTI PRODUKTA MINERA

Slika 3: Maska za urejanje specifikacij produkta MINERA.



Vir: Lastna izdelava.