UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

ALEŠ PREGEL

# RAVNANJE S KADRI IN VODENJE PROJEKTOV OBJEKTNEGA RAZVOJA INFORMACIJKIH SISTEMOV

## MAGISTRSKO DELO

Mentor: prof. dr. Franc Solina

Ljubljana, december 2006

# ZAHVALA

Zahvaljujem se svojemu mentorju prof. dr. Francu Solini, za njegovo pomoč in nasvete pri izdelavi naloge.

Zahvaljujem se prof. dr. Miranu Mihelčiču za pregled naloge in dane pripombe.

Zahvaljujem se podjetju RRC d.d., ki mi je finančno omogočilo študij ter mi v času študija nudilo vso podporo.

Na koncu pa se zahvaljujem še mojim najdražjim Petri, Gali in Pii, ki so mi bile velika vzpodbuda pri študiju in so z razumevanjem sprejemale moje odsotnosti.

# POVZETEK

Moja glavna zadolžitev v podjetju RRC Računalniške storitve d.d. je razvoj informacijskih sistemov za naše naročnike. Kot vodja enote in vodja projekta pa sem tesno povezan tudi s politiko zaposlovanja novih kadrov.

V začetku naloge opisujem metodologijo objektnega razvoja informacijskih sistemov, ki smo ga uporabili tudi na našem projektu KPIS. Teoretični del je osnovan na Enotni metodologiji razvoja informacijskih sistemov v Sloveniji - EMRIS.

Večni problem v podjetjih za razvoj informacijskih sistemov je, da primanjkuje delavec (zaposlencev) z ustreznimi znanji, prav tako pa je potrebno skrbeti, da so zaposleni delavci zadovoljni s svojim statusom v podjetju, kar je potrebno uskladiti z oblikovanjem delovnih skupin. Poleg tega, da je težko dobiti kadre, pa se poraja še vprašanje povezano s stroški ali zaposliti izkušenega delavca (višja plača) ali novinca (manjša plača).

V zadnjih poglavjih magistrske naloge opisujem postopek objektnega razvoja informacijskega sistema za našega naročnika. Za vodenje projekta smo uporabili naše preverjene postopke, ki so opisani v Poslovniku kakovosti podjetja RRC Računalniške storitve d.d.



# ABSTRACT

My major occupation in the firm RRC Računalniške storitve d.d. is the development of information systems for our customers. As a head of a work unit and a project manager I am also strongly involved in company employment policy.

The first goal of my thesis is to describe the methodology of object oriented development in information systems, which we used on our project KPIS. The theory part is based on EMRIS (unified methodology of information systems development in Slovenia).

The constant problem in our business is the insufficiency of employees with proper knowledge for the development of information systems. It is also necessary to take care of employees in the firm. They should be pleased with their position in the firm, so we must be careful how to establish project groups. How to employ a right person is also a hard work, but it's very important. I also research if it is better to employ an expert (higher salary) or a beginner (lower salary).

In the last chapters of my thesis I describe the process of object oriented development in information systems. For the project management we used our firm tested procedure written in our quality policy.

# KAZALO VSEBINE

# KAZALO SLIK



# KAZALO PREGLEDNIC

# 1 UVOD

Podjetja, ki se ukvarjajo z razvojem informacijskih sistemov, se soočajo z različnimi problemi, vsekakor pa je eden od glavnih, kako pridobiti ustrezne delavce za razvoj informacijskih sistemov. Če uporabljamo objektni razvoj informacijskih sistemov, pa je to še težje, saj je takšen pristop razmeroma nov. Zaposlen sem v podjetju RRC računalniške storitve d.d., kjer sem vodja enote za razvoj informacijskih sistemov za javno upravo, glavna naloga pa je vodenje projekta za Ministrstvo za šolstvo in šport. Pri svojem delu se poleg razvoja informacijskega sistema stalno soočam s problemom pridobivanja delavcev za potrebe enote in projekta, ki ga vodim. Oblikovanje skupine za razvoj je vse prej kot lahko.

Objektna tehnologije se tudi v praksi vse bolj uveljavlja kot vodila paradigma razvoja informacijskih in programskih sistemov. Objektni proces razvoja predvideva štiri faze (stopnje) in sedem aktivnosti (dejavnosti), ki se izvajajo znotraj vsake faze. Objektni razvoj temelji na jeziku za objektno modeliranje UML (*Unified Modeling Language*). Poleg glavnih aktivnostih razvoja smo se morali na projektu ukvarjati tudi s podpornimi aktivnostmi, kot so upravljanje (ravnanje) konfiguracij, upravljanje (ravnanje) sprememb in spremljanja verzij.

Kot vodja enote se soočam tudi s kadrovanjem. Izbor delavcev je ena od osnovnih nalog kadrovske funkcije. V podjetjih, v katerih se ukvarjamo z razvojem informacijskih sistemov, je ta naloga toliko težja, saj ljudi na trgu delavcev s primernim znanjem in izkušnjami primanjkuje, če pa le ti so, so za podjetja zelo dragi, saj pričakujejo za svoje znanje primerno nagrado in ugodnosti. Druga možnost, ki ostane podjetjem, kot je RRC d.d., pa je zaposliti ljudi brez izkušenj, s primernimi sposobnostmi in jih vključiti v izobraževanje na področju, kjer bodo delovali. Seveda tudi ta druga možnost za podjetja ni zastonj, saj je treba delavca poslati na draga izobraževanja, poleg tega pa si mora z delom, ki ga ponavadi ne moremo tržiti in katerega delo predstavlja v začetku le strošek, še pridobiti ustrezna znanja in izkušnje. Kje je meja? Ali se da oceniti znanje, ki ga nek kandidat ima? Koliko bi podjetje stalo, da vzgoji delavca s podobno ravnjo znanja?

Pri svojem delu sem se srečal z vodenjem projektov, vodenjem skupine in mrežno načrtovanje. Podjetje RRC ima razvito svojo podporo za vodenje projekta, ki se še razvija. Kot vodja projekta imam kar nekaj predlogov za dopolnitev sistema spremljanja projektov.

Vse tri vsebine (objektni razvoj IS, vodenje projekta in pridobivanje delavcev) je bilo potrebno povezati pri projektu prenove kadrovsko plačnega sistema za osnovnošolske in srednješolske zavode za naročnika MŠŠ (Ministrstvo za šolstvo in šport). Projekt se je začel v letu 2004 za obdobje treh let (v letu 2006 se zaključuje). Glavni cilj projekta je poleg sprotne podpore aplikaciji ŠKIS, prehod iz arhitekture odjemalec strežnik na večnivojsko arhitekturo. Izdelava spletne aplikacije KPIS se v decembru 2006 zaključuje. Pri vzpostavitvi projekta smo izbrali objektni razvoj za aplikacijo KPIS.

V magistrski nalogi bom raziskal osnove Objektnega razvoja IS in projektnega vodenja, prikazani bodo osnovni koncepti objektnega razvoja. Na primeru bom prikazal, kaj smo delali na projektu in kakšno znanje zaposlencev, ki so delali na projektu, je bilo potrebno,

kako smo se lotili pridobivanj novih delavcev za projekt in za enoto, katere vodja sem. Del magistrske naloge bo tudi razvoj odločitvenega modela za izbiro delavca z izkušnjami ali delavca brez izkušenj.

## 1.1 Namen in cilj magistrskega dela

Glavni cilj magistrske naloge je najti rešitve pri zaposlovanju delavcev, ki sodelujejo na projektih objektnega razvoja IS. Rezultat magistrske naloge je tudi natančneje opisati, kakšna znanja in v kakšnem obsegu rabimo pri objektnem razvoju IS, ter podati smernice pri izboru delavcev. Ugotovitve bom primerjal s podatki, ki sem jih pridobil kot vodja projekta ŠKIS/KPIS za Ministrstvo za šolstvo in šport.

Glavni prispevek magistrske naloge je pomagati podjetjem kot je RRC d.d., da se na podlagi izsledkov naloge lažje odločajo pri izbiri delavcev za sodelovanje na projektih objektnega razvoja IS.

## 1.2 Struktura poglavij

V prvem poglavju bom najprej opisal koncept (zasnutek) in pravila za objektni razvoj informacijskih sistemov ter nekaj splošni teoretičnih pravil ravnanja z delavci.

Svoja videnja in opažanja pri izbiri delavcev bom podkrepil z razvojem odločitvenega modela izbire kandidatov. V odločitvenemu modelu postavljam v ospredje primerjavo med zaposlitvijo delavca z izkušnjami in takšnega brez izkušenj.

Vodenje projekta KPIS bom obdelal v zadnjih dveh poglavjih magistrskega dela. Peto poglavje opisuje, kako smo se lotili razvoja po objektni metodologiji, šesto poglavje pa opisuje, kako opredeljuje vodenje projekta poslovnik kakovosti podjetja RRC in kako smo to uresničili v praksi.

## 1.3 Predstavitev podjetja RRC Računalniške storitve d.d.

Podjetje je bilo ustanovljeno leta 1968, kot Elektronski računski center in je kot RRC - Republiški računski center, do leta 1979 delovalo kot samostojna enota v okviru Inštituta Jožef Stefan. 1. 1. 1980 je postalo samostojno podjetje, po končanem procesu lastninjenja, pa od leta 1995 posluje kot delniška družba [28].

Osnovni podatki podjetja so:
**Ime podjetja:** RRC Računalniške storitve, d.d.
**Naslov:** Jadranska 21, SI-1000 Ljubljana

Podjetje RRC, d.d. izvaja projekte za velike naročnike, kot so Ministrstvo za finance RS, Gospodarska zbornica Slovenije, Ministrstvo za obrambo RS in Ministrstvo za šolstvo in

šport RS, pri katerih smo izvedli že vrsto rešitev, ki so neposredno primerljive z rešitvami za velike in srednje poslovne sisteme.

Kje in kako lahko RRC pomaga svojim naročnikom [28]:

- **Izkušnje z distribuiranimi bazami in velikimi količinami podatkov ter uporabnikov**

Večina naših dosedanjih naročnikov je organiziranih na območju celotne Slovenije, kjer avtonomno zbirajo in obdelujejo podatke. Ti podatki se periodično stekajo na centralne lokacije združb, kjer jih dalje obdelujejo in analizirajo. Število uporabnikov pri našem najvišjem naročniku presega 2.500. Podatki pa se dodatno izmenjujejo tudi s sorodnimi organizacijami, ki so prav tako večinoma organizirane dislocirano. Tako pripravljene podatke uporabljajo tudi pri odločanju vlade RS.

- **Izkušnje z visoko zahtevnimi sistemi, v smislu varnosti delovanja in sposobnost hitrega odzivanja na napake**

Naši naročniki so večinoma večje združbe, ki zaradi narave svojega dela zahtevajo zanesljivost in varnost pri delovanju svojih informacijskih rešitev. To podjetje RRC tudi uspešno zagotavlja. Delo se ureja po vnaprej določenih načrtih, z vmesnimi koordinacijami aktivnosti. V primeru napak zagotavljamo hiter odziv in rešitev problema. Daljši zastoji niso sprejemljivi, ker bi lahko začasno ohromili delovanje kritično pomembnih dejavnosti.

- **Visoka raven usposobljenosti in učinkovitosti zaradi dolgoletnega delovanja na podobnih projektih**

RRC d.d. se je v okviru projektov, ki jih izvaja, že večkrat srečal s podobno vsebinsko, strukturno in tehnološko problematiko, kar nam je omogočilo razviti učinkovite inštrumente za odpravo napak, spremljanje in diagnostiko delovanja sistemov. To usposobljenost nam priznava tudi Center vlade za informatiko.

- **Izkušnje pri zagotavljanju podpore uporabnikom (telefonska pomoč, HelpDesk, baza znanja)**

Svojim naročnikom nudimo za svoje rešitve jamstva, dodatno pa lahko nudimo tudi telefonsko oz. tehnično pomoč na lokaciji naročnika. Kot rezultat dolgoletnih posredovanj pri tehničnih problemih naših uporabnikov, smo razvili obsežno notranjo zbirko znanja.

- **Izkušnje z raznovrstnimi tehnologijami in arhitekturami**

RRC d.d. razpolaga z obilico strokovnjakov s področja velikih podatkovnih baz tipa ISAM-Cobol, MS SQL ter Oracle RDBMS in raznovrstnih razvojnih orodij od Cobola in Delphija do najnovejših Microsoftovih .NET (jezik C#) razvojnih orodij ter SAS metodologije. Arhitekturo in tehnološko rešitev informacijskih sistemov vedno prilagodimo potrebam in željam svojih naročnikov.

Podjetje RRC, d.d. je na razpisu Centra vlade za informatiko (OMIS.IT-4/2001), na katerega se je prijavilo za priznanje sposobnosti opravljanja storitev za državno upravo, dobilo potrjeno sposobnost za naslednja področja [28]:

1. Analiziranje, načrtovanje, izvedba, vpeljava oz. implementacija ter vzdrževanje in dopolnjevanje IS

2. Usposobljenost za aktivnosti iz prve točke za sledeča področja IS:

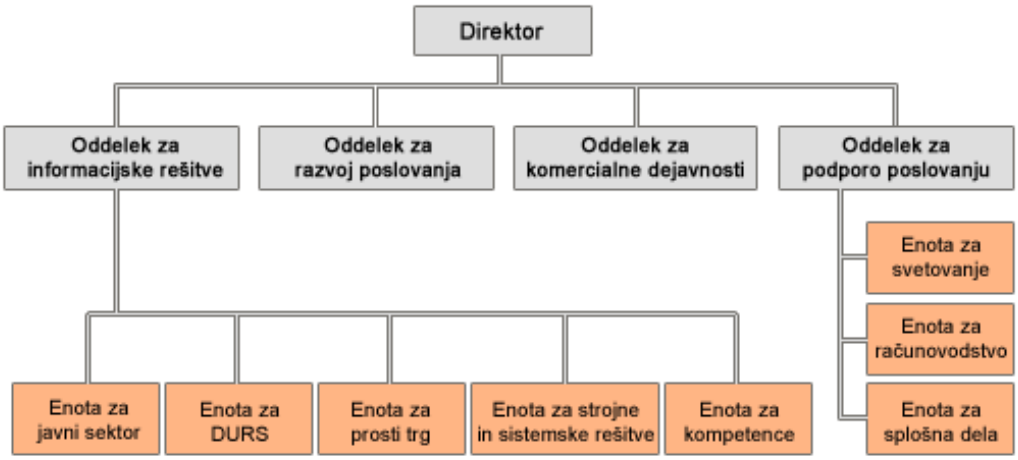
   a. finance
   b. pravosodje

c. obramba
d. šolstvo
e. e-poslovanje
f. javna naročila
g. podpora odločanju

3. Usposobljenost velja za naslednje kriterije IS:

a. enostavni informacijski sistemi
b. zahtevnejši informacijski sistemi
c. spletni informacijski sistemi
d. poslovno kritični informacijski sistemi

Organizacijsko strukturo podjetja predstavlja slika 1.1. Vseh zaposlenih v podjetju je trenutno 46. Sem vodja Enote za javni sektor, v katero je zaenkrat vključena še Enota za prosti trg. V enoti je trenutno zaposlenih 12 delavcev, se pa ta številka spreminja, saj je ravno ta enota najbolj podvržena širjenju.




**Slika 1.1: Organizacijska struktura podjetja RRC d.d.**

V okviru Enote za javni sektor je trenutno vzpostavljeno 9 projektov, ki so podrobneje opisani v preglednici 1.1.

| **NAROČNIK** | **OPIS** |
|---|---|
| BANKA SLOVENIJE | Storitve programiranja aplikativnih sistemov v Oracle okolju - POOL Banke Slovenije in Podpora upravljanja z vrednostnimi papirji. |
| CASINO LEV (Projekt BINGO) | Aplikacija za izdelavo listkov z unikatnimi kombinacijami, preverjanjem zadetkov in shranjevanje podatkov preteklih kol za igro Bingo. TEHNOLOGIJA: Microsoft .NET |
| Ministrstvo za kmetijstvo, gozdarstvo | Aplikacija Obračun NRN 2004 za obračun zahtevkov za povrnitev nastale škode kmetom ob naravnih nesrečah. |

| | |
|---|---|
| in prehrano, Agencija RS za kmetijske trge in razvoj podeželja | TEHNOLOGIJA: Oracle (RDBMS Oracle 8.1.7, Oracle Forms, Oracle reports 6i. |
| Ministrstvo za obrambo, Uprava za zaščito in reševanje (Projekt AJDA) | Spletna aplikacija za ocenjevanje škode na kmetijskih pridelkih.<br><br>TEHNOLOGIJA: Microsoft .NET (.NET 2.0, SQL Server 2005) |
| Ministrstvo za šolstvo in šport RS (Projekt KPIS) | Kadrovsko plačni informacijski sistem. Sistem KPIS je spletna aplikacija za vodenje matičnih podatkov zaposlenih na osnovnih in srednjih šolah, njihovih delovnih mestih in delovnih nalogah. V sistemu KPIS je možen vpogled na centralni lokaciji naročnika izračunane količnike za plače delavcev.<br>**TEHNOLOGIJA:** Microsoft .NET, RDMBS Oracle 10g |
| Ministrstvo za šolstvo, znanost in šport RS (Projekt ŠKIS) | Šolski kadrovski informacijski sistem, izdelava računalniško podprtega šolskega kadrovskega informacijskega sistema: **TEHNOLOGIJA:** Delphi, RDMBS Oracle 8i |
| Ministrstvo za finance RS, DURS (WDohod, WObr, WDobiček) | Programi, ki jih DURS nudi davčnim zavezancem, da v elektronski obliki oddajo kontrolne podatke za dohodnino, podatke za obračun davka od dobička pravnih oseb in podatke za obračun davek iz dejavnosti.<br>**TEHNOLOGIJA:** Delphi |
| Ministrstvo za pravosodje RS (Projekt EZO) | Informacijski sistem s centralno bazo podatkov za spremljanje evidence zaprtih oseb v RS:<br>**TEHNOLOGIJA:** Oracle (RDBMS Oracle 8.1.7, Oracle Forms, Oracle reports 6i. |
| Ministrstvo za obrambo RS (Projekt ISOJAN) | Informacijski sistem za spremljanje javnih naročil pri MORS: **TEHNOLOGIJA:** Oracle (RDBMS Oracle 8.1.7, Oracle Forms, Oracle reports 6i. |

**Preglednica 1.1: Pregled projektov podjetja RRC, ki so vzpostavljeni v okviru enote za javni trg.**

Poslanstvo podjetja RRC, d.d. je zagotavljanje kakovostnih in celovitih rešitev na področju informacijske tehnologije. Pri uresničevanju tega poslanstva se oziramo na trg in potrebe naročnika, ki mu nudimo strokovnost, zanesljivost, dolgoročno partnersko sodelovanje, z izkazano dodano vrednostjo in ustvarjalnost pri iskanju rešitev. V prihodnosti želimo na svojem področju ostati v koraku z novostmi ter razširiti in obogatiti svojo ponudbo v sodelovanju z drugimi podjetji doma in v tujini.

Podjetje RRC je leta 1999 prejelo certifikat ISO 9001. Certifikat smo prejeli za dejavnost razvoja in vzdrževanja informacijskih sistemov, vzdrževanja strojne računalniške opreme in nudenja sistemske podpore. Našim naročnikom certifikat pomeni zagotovilo, da na tem področju nudimo standardizirane delovne postopke, ki zagotavljajo kakovost izdelkov in s tem uspeh in zadovoljstvo naših poslovnih partnerjev.

# 2 OBJEKTNI RAZVOJ INFORMACIJSKIH SISTEMOV

## 2.1 Objektna tehnologija in osnovni koncepti (zasnutki) objektne tehnologije

Uveljavljanje objektnega načina v prakso in vse večje tekmovalne sposobnosti podjetij, ki ta koncept sprejmejo in ga razvijajo naprej, je pripeljalo podjetja do nedvoumne odločitve, da je potrebno čim prej koncepte razvoja IS prilagoditi objektni tehnologiji.

Največja sprememba, ki jo prinaša objektni proces, je obravnavanje podatkov in procesov kot eno. Klasične metode dosledno ločujejo podatkovno in procesno stran IS. Najpomembnejši cilj razvoja je zmanjšati čas razvoja IS, in posledično tudi stroške razvoja IS. Največji del skupnih stroškov razvoja IS predstavljajo prav stroški dela.

Objektni pristop poizkuša objekte realnega sveta preslikati v objekte objektnega modela. Model tako postane abstrahirana in poenostavljena stvarnost, ki je narejena za boljše razumevanje sistema ali procesa, ki ga modeliramo [15]. Prav zato je ena od osnovnih prednosti objektnega razvoja in konceptov večje razumevanje med uporabniki in razvijalci, saj je stopnja abstrakcije realnega sistema (stopnja konkretizirane implementacije modela) odvisna od namena neke faze modeliranja sistema. Tako se npr. izvaja modeliranje na višjem nivoju (ravni) (*High-level*), v katerega so vključeni praviloma vsi udeleženci pri izgradnji informacijskega sistema in obsega zgodnje osredotočenje na sam modelirani proces, na zbiranje zahtev in predvidevanje opcij. V kasnejši fazi izdelave se izvaja modeliranje na nižjem nivoju (*Low-level*), v katerega so vključeni neposredni razvijalci programske opreme in obsega podrobno kodiranje funkcionalnosti in aplikacij [14].

## 2.2 Osnovni koncepti objektne tehnologije

Osnovni koncepti objektne tehnologije so:

- objekti,
- ograjevanje,
- razredi,
- dedovanje in delegiranje,
- prenos (pošiljanje) sporočil in
- polimorfizem (mnogoličnost).

### 2.2.1 Objekti

Objekt (*object*) je skupek podatkov in logike (operacij). Podatki določijo stanje objekta, operacije pa predstavljajo funkcije in transformacije, ki jih uporabimo nad objektom ali pa jih le ta zagotavlja. Metoda je implementacija neke operacije. Tisti, ki objekt uporabljajo, morajo poznati le operacije, saj le te zagotavljajo komuniciranje z objektom. Metode pa mora poznati le načrtovalec objekta.

### 2.2.2 Ograjevanje

Tehniko, ki omogoča ločevanje definicij od podrobnosti implementacije, imenujemo ograjevanje (*encapsulation*). Stanje objekta ne more biti neposredno spremenjeno, pač pa je to možno le preko operacij. Možno je, da objekt neposreden dostop dovoljuje.

Ograjevanje namreč omeji število domen, o katerih je uporabniku objekta potrebno razmišljati. Uporabnik objekta mora poznati le operacije, ki so na voljo, ter vedeti kako operacije uporabljati (katere podatke obravnava operacija in katere vrne).

Vse to pomeni, da lahko objekt pred uporabnikom skrije svoje metode in podatke. To je pomembno v primeru sprememb. Verižna reakcija ob spremembi objekta je le v primeru, da je spremenjen vmesnik objekta, ne pa spremembe znotraj objekta.

### 2.2.3 Razredi

Razred (*class*) je opis podobnih objektov. Vsak objekt je primerek ali predstavnik (*instance*) nekega razred. Razred definirajo operacije, ki jih zagotavljajo objekti tega razred. Njihovo implementacijo (metode) ter podatkovne tipe, ki definirajo dovoljeno stanje, v katerem se objekt nahaja. Za podani razred lahko v času izvajanja obstaja več primerkov - objektov.

Kadar si objekti istega razreda delijo neke podatke oz. atribut po imenu in po vrednosti, govorimo o razrednih spremenljivkah. (*class variables*). Metode, ki dostopajo in uporabljajo le razredne spremenljivke in atribute, imenujemo razredne metode (*class metods*). Če ena ali več metod razreda nima določene implementacije, govorimo o abstraktnih metodah (*abstract metods*). Takšen razred ne more imeti primerkov in ga imenujemo abstraktni razred (*abstract class*). Razred, katerega primerki so razredi, pa imenujemo metarazred (*metaclass*).

### 2.2.4 Dedovanje in delegiranje

Kadar imamo primer, da imajo razredi objektov podobne, vendar ne identične atribute in operacije, uporabimo možnost dedovanja (*inheritance*). Podrazredi (*subclasses*) podedujejo operacije in atribute nadrazreda (*superclass*). Podrazred ima poleg podedovanih atributov in metod tudi lastne. Pri organiziranju razredov v razredne hierarhije srečujemo enkratno oziroma večkratno dedovanje. Enkratno dedovanje je takrat, ko nek podrazred podeduje značilnosti enega razreda. Če razred podeduje značilnosti od več neposrednih razredov, govorimo o večkratnim dedovanju. Poznamo še bolj zapletene načine dedovanja in sicer ponavljajoče dedovanje ter selektivno dedovanje.

Koncept delegiranja je podoben dedovanju. Osnovna razlika je, da delegiranje vzpostavlja povezave med konkretnimi objekti, dedovanje pa povezuje razrede.

### 2.2.5 Prenos (pošiljanje) sporočil

Osnovni namen prenosa sporočil je proženje operacij objekta. Sporočilo vsebuje ime, ki pove, kateremu objektu je sporočilo namenjeno in morebitne parametre. Običajno je ime sporočila enako imenu operacije, ki jo je potrebno izvesti. Rezultat (izid) operacije se vrne do izvora preko povratnega sporočila.

### 2.2.6 Polimorfizem

Polimorfizem pomeni, da je ista operacija implementirana na več različnih načinov, oziroma zanjo obstaja več metod. Poznamo več osnovnih tipov polimorfizma [8]: vključeni polimorfizem, operacijski polimorfizem in parametrski polimorfizem.

## 2.3 Objektni proces razvoja informacijskih sistemov

### 2.3.1 Faze (stopnje) objektnega razvoja informacijskih sistemov

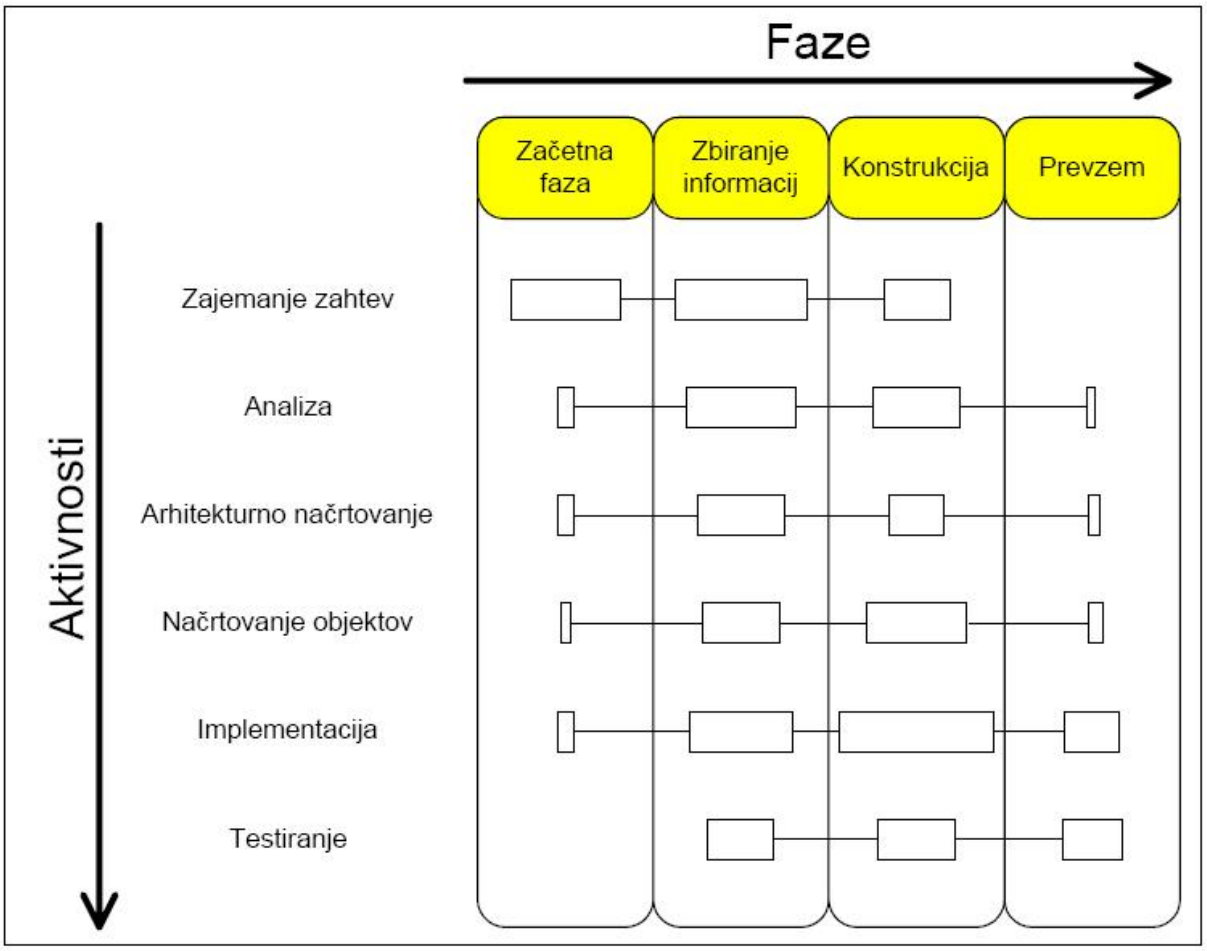
Proces razvoja programske opreme predstavlja opredelitev aktivnosti, ki so potrebne za implementiranje in spreminjanje potreb uporabnikov v ustrezne proizvode, ki predstavljajo programsko rešitev, ter aktivnosti za spreminjanje dodatnih sprememb teh zahtev v nov, ponovno skladno ustrezen niz produktov [14].

Postopno naraščajoča narava objektnega pristopa ima za posledico prekrivanje med posameznimi fazami razvoja, kar narekuje predstavitev procesnega modela v dveh dimenzijah: časovno po fazah razvoja in organizacijsko po aktivnostih, ki jih opravljamo znotraj vsake ponovitve. Po časovni komponenti so opredeljene faze razvoja, ki zajemajo celoten življenjski cikel razvoja informacijskega sistema [1], in sicer:
- začetna faza,
- zbiranje informacij,
- konstrukcija,
- prevzem.

Znotraj posameznih faz (stopenj) izvajamo aktivnosti (dejavnosti). Aktivnosti se delijo še na opravila in korake. Zelo pomembna mehanizma objektnega razvoja sta iterativni in inkrementalni (naraščajoči) pristop k razvoju. Končnemu cilju se približujemo v iteracijah, rezultati predhodnih iteracij pa služijo za nadgradnjo v tekoči iteraciji.

Slika 2.1. prikazuje dvodimenzionalni pogled na razvojni proces. Horizontalna smer predstavlja časovno os, kjer si sledijo posamezne faze (stopnje) procesa, na vertikalni osi pa so prikazane aktivnosti (dejavnosti). Na sliki lahko vidimo, kako so posamezne faze organizirane po času in si sledijo zaporedno, hkrati pa je potrebno gledati tudi organizacijo po aktivnostih in njihov delež, ki ga uporabljamo v posamezni fazi. Vidimo, da se posamezne aktivnosti lahko uporabljajo v vseh fazah oziroma vsaj v večih fazah procesa razvoja.

Življenjski cikel vsake generacije programske opreme oziroma produktov posamezne faze razvoja razdelimo na navedene štiri faze. Vsaka faza se zaključi z definiranim časovnim mejnikom in zahteva sprejetje odločitev, ki lahko v veliki meri vplivajo na uspešnost projekta.




**Slika 2.1: Shema objektnega procesa razvoja IS [8]**

Bistveni cilji posamezne faze so sledeči [1]:

- **Začetna faza** definira namen in cilj projekta, izvedljivost projekta glede na poslovne zahteve, definira projekt s poslovnega stališča, določi njegov obseg in dobrobiti izvedbe. Pod pojmom projekt je tu mišljen proces razvoja informacijskega sistema. Poleg tega ocenimo, kako obsežen bo projekt, kakšna so potrebna vlaganja, kakšni so potrebni viri in katera so kritična tveganja pri izvajanju projekta. Že v tej fazi zajamemo bistvene primere uporabe in se predvsem osredotočimo na izvajanje aktivnosti definiranja zahtev in analize. Če v to fazo vključimo tudi izdelavo prototipa za potrditev osnovnega koncepta, potem se v določeni meri izvajata tudi aktivnosti načrtovanja in implementacije prototipa. Prototip nima vpliva na končni produkt, zato je kot tak kasneje zavržen in zato tudi ni podvržen aktivnosti testiranja.

- **Faza zbiranja informacij** v grobem obsega analizo problemskega področja, postavitev ogrodja arhitekture, izdelavo projektnega načrta za fazo konstrukcije, zbiranje začetnih zahtev za funkcionalnost sistema in izpopolnitev opredeljenih tveganj. V prvi fazi razpolagamo z okvirnimi zamislimi o zahtevah glede informacijskega sistema. Nato zberemo podrobne podatke o zahtevah, izvedemo analizo in načrt ter določimo osnovno arhitekturo novega sistema. Na koncu določimo načrt konstrukcije. Na osnovi vseh teh podatkov opredelimo, kaj in kako bomo delali, ter definiramo tveganja:
  - tveganje potreb, ki vključuje nevarnost, da zgradimo informacijski sistem, ki ne ustreza postavljenim zahtevam in je lahko posledica nesporazuma med razvijalcem in uporabnikom;
  - tehnološko tveganje, kjer se poraja vprašanje ali imamo na razpolago ustrezno tehnologijo za rešitev problema, saj se namreč lahko dogodi, da uporabimo več različnih tehnoloških rešitev, ki so dobre, povezava med njimi pa je težavna ali celo nemogoča. Poraja se tudi vprašanje o posledicah v primeru odpovedi sistema;
  - tveganje kadrov, ki vključuje dilemo, ali imamo na voljo dovolj sodelavcev in ali so ustrezno usposobljeni za izvedbo projekta. Pri večini projektov se izkaže, da izobraževanje ni bilo izvedeno v zadostni meri, zato je potrebno izobraževanju in usposabljanju posvetiti posebno skrb;
  - tveganje, ki vključuje (ne)zadostnost podpore in (ne)zaupanje vodstva glede realizacije projekta.

  V fazi zbiranja zahtev določimo do 80% primerov uporabe. Primeri uporabe tvorijo ogrodje informacijskega sistema.

- **Faza konstrukcije** vključuje načrtovanje izgradnje in samo izgradnjo. V tej fazi so primeri uporabe in diagrami razredov osnova za delo. Izgradnja je pretežno izvedena v okviru aktivnosti analiziranja, načrtovanja, izvedbe in testiranja (preizkušanja). Korake izgradnje ponavljamo do ustrezne ravni kakovosti, kar je uporabniško testirano in potrjeno. Ta način se uporablja zaradi zmanjšanja tveganja. Faza vključuje več ponavljanj, znotraj katerih postopno dograjujemo sistem. V vsaki ponovitvi se dodaja lastnost primerom uporabe, izvedena sta testiranje in integracija. Uporabniki sodelujejo pri testiranju vsake ponovitve in lahko vplivajo še na zadnje korekcije izdelanih lastnosti. Ob izdelani programski opremi pripravimo tudi ustrezno tehnično in uporabniško dokumentacijo, ki vključuje tudi UML diagrame. Dopolnimo tudi vse nedokončane modele iz prejšnjih faz. Na koncu faze je potrebno oceniti, ali je programska oprema primerna za vsakodnevno uporabo in predajo uporabnikom.

- **Faza prevzema** vključuje predajo izdelka v uporabnikovo okolje. Zadnja faza je faza prehoda. V tej fazi izvedemo zadnja testiranja, merimo zmogljivosti sistema in izobrazimo uporabnike. Po zagonu se lahko pokažejo težave, ki jih v fazi testiranja nismo uspeli identificirati. V kolikor so odstopanja večja, lahko odpravo problema posredujemo v naslednjo ponovitev izdelave sistema. Manjše popravke izvedemo kar znotraj faze prevzema. Tako dopolnimo model implementacije in testiranja ter zabeležimo pridobljene izkušnje za izboljšanje procesa v naslednjih ponovitvah. Razvoj programske opreme na opisan način omogoča stalno dopolnjevanje in spreminjanje lastnosti glede na zahteve poslovnega okolja in uporabniške zahteve. Ključnega pomena je postopnost in iterativnost ter možna povezanost posameznih

modelov. To omogoča enostavno odkrivanje potrebnih sprememb v modelih, ki so glede na spremembe v modelu primerov uporabe bližji implementaciji.

Drugo dimenzijo (razsežnost) poenotenega procesa predstavljajo statične komponente (sestavine), ki so vključene v posamezne faze procesa in vsebujejo aktivnosti:

- **zajemanje zahtev** - definiranje lastnosti sistema preko pridobivanja uporabniških zahtev,
- **analiziranje** - natančnejše opredeljevanje in dograjevanje ter strukturiranje lastnosti,
- **načrtovanje** – realizacija (uresničitev) lastnosti v okviru definiranja logične arhitekture sistema,
- **implementacija** - izgradnja programske kode,
- **testiranje** - izvajanje integracijskega preverjanja implementacije sistema.

### 2.3.2 Procesni model

Pri izgradnji informacijskih sistemov je potrebno zagotoviti čim višjo kakovost izgradnje informacijskih sistemov. To dosežemo tako, da znotraj organizacije (podjetje, projektna skupina,..) natančno določimo naloge in odgovornosti pri izgradnji informacijskega sistema. Čim natančnejši je popis nalog, ki jih je potrebno izvesti in popis odgovornosti, tem bolj je razvoj informacijskega sistema predvidljiv in nadzorovan.

Dodeljevanje odgovornosti znotraj združbe, ki se ukvarja z razvojem informacijskih sistemov imenujemo procesni model. Procesni modeli so pomembni, ker zagotovijo osnovo za pridobitev odgovorov na pet bistvenih vprašanj [8]:

- Načrtovanje – Kaj bomo naredili, da dosežemo zadane cilje?
- Pooblastila – Kako lahko vplivamo na dogajanje z namenom priti tja, kamor smo namenjeni?
- Napovedi – Kam bomo prišli oziroma kam gremo?
- Ocenitev – Kje v procesu smo in zakaj?
- Sledljivost – Kako smo dosegli učinek – npr., kako je posamezen del kode povezan z modeli v načrtovanju in analizi ter nazaj do zahtev?

Procesni model je torej skupek pravil, strategij, aktivnosti, metod opravil in korakov [8].

Pomembno pri razvoju IS je izbrati pravi (idealni) procesni model. Dobro bi bilo imeti splošni procesni model, ki bi ustrezal vsem projektom, vendar je to le utopija kljub prizadevanjem informatikov.

Za podjetja je najbolj pomembno, da že pri prvem projektu izberejo čim boljši procesni model. Še bolj pomembno pa je vzpostaviti mehanizme, da se procesni model stalno izboljšuje na podlagi izkušenj pri prejšnjih projektih.

Glede na raznolikost projektov lahko projekte združujemo v različne tipe projektov. Za vsak tip pa lahko imamo procesni model.

Objektni pristop združuje naslednje pomembne strategije [8]:

- iterativni razvoj,
- inkrementalni razvoj,
- prototipiranje,
- ponovno uporabo.

**Iterativni razvoj** predvideva uporabo večjega števila iteracij in je zelo uporabljena metoda doseganja končnih ciljev pri razvoju IS. V praksi pomeni, da se k istemu problemu vračamo večkrat, dokler ga ne rešimo.

Slabosti iterativnega razvoja:

- Težje načrtovanje poteka razvoja.
- Ni znano število iteracij (omejevanje ne reši problema).
- Večkratno vračanje k istemu problemu (čas rešitve je daljši).
- Stvari naredimo narobe, preden jih naredimo pravilno.
- Stvari naredimo slabo, preden ji naredimo dobro.

Prednosti iterativnega razvoja

- Izognemo se lahko sekvenčnemu delu, kjer nam to ustreza.
- Navidezni hitrejši napredek, kar povečuje motivacijo razvijalcev.
- Vračanje k problemu (končni rešitvi), ko je razpoloženje boljše.
- Hitrejši razvoj pri več podobnih problemih, saj delno rešitev hitro uporabimo na več problemov.

O **inkrementalnem razvoju** govorimo takrat, ko rešitev izdelujemo po delčkih, dokler ne sestavimo celote. Takšen razvoj zahteva razbitje celote na dele, ki jih lahko rešujemo sočasno in/ali izmenično. Inkrementalni razvoj predvideva delitev sistema na posamezne dele, medtem ko iterativni razvoj pomeni dopolnjevanje delnih rešitev.

**Prototipiranje** uporabljamo predvsem z namenom, da pomagamo uporabnikom pri postavljanju zahtev, prav tako pa zmanjšamo posledice napačnih odločitev pri načrtovanju sistema, saj lahko napake odkrijemo v fazi razvoja delnih rešitev. V začetni fazi pripravimo prototip končnega izdelka, ki ima končne lastnosti izdelka le nakazane ne pa tudi izdelane. Prototipiranje se uporabi tudi takrat, ko se uporabnik ne more odločiti med več možnostmi.

**Ponovna uporaba** je vse pomembnejši pojem pri izgradnji modernih sistemov, pomeni pa, da dobre rešitve ponovno uporabimo. Ponovno lahko uporabimo cele module ali pa le manjše delčke programske kode. Pomembno je to, da so ponovno uporabljivi deli v času razvoja prilagojeni k temu, da jih bomo v bodoče ponovno uporabili v drugih sistemih. Za ponovno uporabljive dele si moramo zgraditi sistem dokumentacije (listin) in hranjenja, ki ga imenujemo skladišče (repository).

### 2.3.3 Vloge (delovni programi) pri objektnem razvoju

Pri razvoju informacijskih sistemov sodeluje več oseb z različnimi znanji. Glede na svoje znanje in odgovornosti lahko definiramo seznam vlog[1] (delovnih programov), ki se pojavijo v času trajanja projekta (preglednica 2.1). Pomembno je to, da seznam vlog ne predstavlja dejanskih oseb, saj lahko ena oseba predstavlja več vlog, lahko pa je tudi obratno, da obstaja več oseb, ki opravljajo isto vlogo.

| **Vloga (delovni program)** | **Opis vloge (delovnega programa) v objektnem procesu** |
|---|---|
| Uporabnik, strokovnjak problemskega področja | Oseba, ki s svojim znanjem v fazi zajema zahtev podaja uporabniške zahteve, v fazi prevzema razvojno skupino obvešča o odpovedih ali nepravilnem delovanju sistema. |
| Snovalec primerov uporabe | Oseba, ki s sodelovanjem uporabnika razvije model primerov uporabe, prepoznava akterje (udeležence) in oblikuje prvo verzijo (inačico) pojmovnika. |
| Arhitekt | Oseba, ki razvije arhitekturo ciljnega sistema. Prevzema odgovornost implementacije arhitekture. |
| Načrtovalec uporabniškega vmesnika | Oseba, odgovorna za izdelavo prototipov uporabniškega vmesnika. |
| Načrtovalec primerov uporabe | Oseba, ki je odgovorna za popis in opis posameznega primera uporabe. |
| Inženir komponent | Oseba, ki definira in vzdržuje odgovornosti, atribute, relacije in neuporabne zahteve posameznega razreda v vseh fazah razvoja. |
| Projektni vodja | Oseba, ki je odgovorna za razvoj projektnega načrta, dodeljevanje vlog sodelujočim na projektu in vodenje projekta. |
| Sistemski integrator | Oseba, ki je odgovorna za integracijo komponent, ki so v pristojnosti posameznega inženirja komponent. |
| Načrtovalec | Oseba, odgovorna za skladnost proizvodov aktivnosti objektnega načrtovanja in modela analize. |
| Programer | Oseba, ki implementira model načrtovanja. Proizvod je programska koda. |
| Tester | Oseba, ki v sodelovanju s sistemskem integratorjem in inženirjem testiranja izvede teste integracije (povezanosti) sistema. |
| Pregledovalec | Oseba, ki izvaja preglede rezultatov posameznih aktivnosti. |
| Analitik | Oseba, ki identificira uporabne in neuporabne zahteve. |

**Preglednica 2.1: Opis vlog (delovnih programov) pri objektnem razvoju IS [8]**

[1] Pravilneje od vloge je delovni program. Po Miheliču [11] je potrebno kljub navidezni podobnosti loči vlogo posameznega člana od delovnega programa, ki ga je dolžan opraviti, saj ima ne tako nepomembno število članov v združbe tudi druge pristojnosti.

### 2.3.4 Aktivnosti (dejavnosti) objektnega razvoja informacijskih sistemov

#### 2.3.4.1 Zajem zahtev

Namen je preko primerov uporabe opredeliti lastnosti bodočega sistema. Zahtevano lastnost opredelimo, ko opišemo načine uporabe sistema in rezultate, ki jih sistem nudi uporabnikom. Pomembno je identificirati vloge, ki komunicirajo s sistemom, s čimer so opredeljeni akterji sistema. Primere uporabe zberemo v model s pomočjo diagramov primerov uporabe. Poleg primerov uporabe, ki so glavno orodje za zajemanje zahtev, si pomagamo še z diagrami za opisovanje obnašanja sistema (npr. z diagrami zaporedja). Poleg tega lahko naredimo tudi nekatere prototipe uporabniških vmesnikov, ki pomagajo k podrobnemu razumevanju načina interakcije uporabnika in sistema. Za razumevanje problemske domene lahko modelu primerov uporabe dodamo tekstovni opis z opisom akterjev in njihove vloge ter izvajanje primerov uporabe. Za razumevanje lahko dodamo tudi definicijo uporabljenih izrazov. Primerom uporabe opredelimo pomembnost in jih razvrstimo v vrstni red za nadaljnji razvoj.

#### 2.3.4.2 Analiziranje

Analiziranje predstavlja pogled na način realizacije zahtevane lastnosti, kjer je potrebno podrobno analizirati uporabniške zahteve, opredeljene v predhodnem koraku. Model analize je prikazan zgolj na konceptualnem nivoju, podrobnosti implementacije niso prikazane. Razlikuje se od modela primerov uporabe, saj je opisan v formalnem jeziku. Namenjenem je razvijalcu sistema in prikazuje notranji pogled na sistem, ki je sestavljen iz razredov in paketov. Prikazuje način izvedbe lastnosti znotraj sistema, ki je osnova za načrtovanje in opredeljuje način realizacije primerov uporabe s strani sistema. Opredelimo razrede in njihove odgovornosti, ki jih zagotavljajo razredne operacije in vmesniki, ter paketi, ki bodo zagotavljali posamezne lastnosti. Opredelijo se tudi povezave med razredi ter njihovo združevanje v pakete. Na miselnem nivoju preizkusimo delovanje primerov uporabe skozi tako opredeljeni model. Opredelimo tudi povezavo med primeri uporabe in elementi modela analize, ki omogočajo njihovo izvajanje. Za opisovanje modela analize uporabljamo predvsem razredni diagram in diagram sodelovanja. Odvisnosti med paketi, ki realizirajo (uresničijo) posamezne primere uporabe, prikažemo v razrednem diagramu na višjem nivoju abstrakcije.

#### 2.3.4.3 Načrtovanje

Znotraj aktivnosti načrtovanja opredelimo fizični model sistema, ki bo osnova za implementacijo. Opredelimo model načrta in model porazdelitve. V modelu načrta podrobno opredelimo razrede, povezave med njimi in dinamično obnašanje sistema. Dokončno opredelimo atribute razredov, opredelimo vidnost (javna, zasebna, zaščitena) in opišemo metode. Metode lahko opišemo v naravnem jeziku ali psevdokodi ter prepustimo programiranje naslednjemu koraku, lahko pa jih že programiramo v izbranem programskem jeziku. Povezave med objekti dokončno opredelimo glede številnosti, smeri navigacije in imenovanja vlog glede na predvideno okolje aplikacije. Prav tako je treba upoštevati predvideno tehnologijo shranjevanja podatkov, ko načrtujemo razrede, ki izvajajo zapisovanje podatkov v okviru transakcije. Pakete preoblikujemo v podsisteme. Treba je zagotoviti čim

manjšo odvisnost med podsistemi, opredeliti ustrezne vmesnike in zagotoviti, da podsistem ponuja pravilno realizacijo operacij, ki so opredeljene z njegovimi vmesniki. Dinamično obnašanje sistema podrobno opišemo z diagrami zaporedja, diagrami stanj in diagrami aktivnosti. Tako je model načrta podrobna predstavitev sistema z upoštevanimi lastnostmi okolja, v katerem bo uresničen. Model porazdelitve vsebuje vozlišča, njihove lastnosti in povezave ter začetno razporejanje razredov in podsistemov v vozlišča. Za opisovanje modela uporabimo diagram porazdelitve.

#### 2.3.4.4 Implementacija

Glavni rezultat te aktivnosti je izvorna koda, ki uresničuje izdelke predhodnih korakov in omogoča delovanje sistema. Implementacija vključuje posamezne programske komponente, ki implementirajo razrede in podsisteme, ki so bili opredeljeni v načrtovanju. Razrede implementiramo kot posamezne komponente, ki vsebujejo izvorno kodo. Implementacija poteka v več iteracijah in lahko tudi vzporedno, zato je pomembno načrtovati integracijo po vsaki iteraciji. V okviru iteracij se izdelajo obvladljivi deli programske kode, ki se sproti integrirajo z že razvitim delom sistema. V tem koraku izvedemo testiranje posameznih komponent, ki jih nato integriramo in namestimo na ustrezno strojno opremo. Celovito sistemsko preverjanje se izvede v koraku testiranja. V modelu uresničitve uporabljamo diagrama komponent in porazdelitve.

#### 2.3.4.5 Testiranje

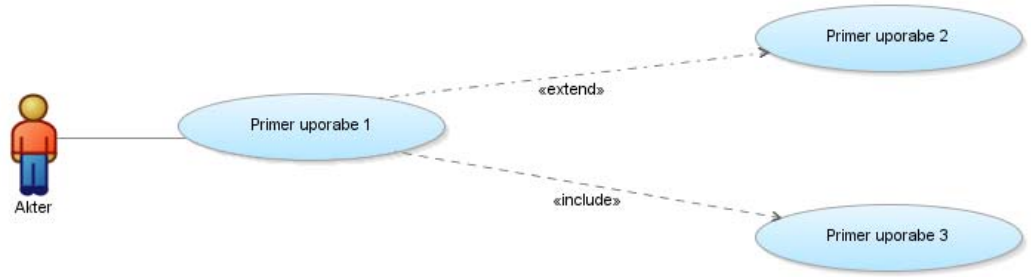
Namen testiranja je preizkus delovanja sistema tako glede izpolnjevanja zahtevanih lastnosti v smislu pravilnosti delovanja kot tudi izpolnjevanja neuporabnih zahtev npr. zmogljivosti. Testiranje na nivoju komponent je opravljeno v koraku implementacije, v koraku testiranja pa se izvaja, kot omenjeno, integracijsko in sistemsko testiranje. Za sistematičen način testiranja je pripravljen model testiranja, ki vsebuje testne primere, ki opredeljujejo vsebino testiranja, testne postopke, ki opredeljujejo način testiranja in testne komponente, ki omogočajo avtomatizacijo testnih postopkov [15]. Načrt testiranja mora vključevati urnik testiranja in osebe, ki bodo testiranje izvajale. Po izvedenem testiranju ocenimo uspešnost testiranja. V primeru uspešnega testa celotnega sistema je sistem pripravljen za predajo. Model testiranja dopolnjujemo in ga uporabimo ob vsaki novi ponovitvi znotraj življenjske dobe sistema. V nadaljevanju bosta strnjeno prikazani še dve metodologiji razvoja informacijskih sistemov, ki sta namenjeni predvsem razvoju sistemov delujočih v realnem času in temeljita na predpostavkah poenotenega procesa.

## 2.4 Jezik za objektno modeliranje

Jeziku za objektno modeliranje, ki je skupek diagramskih tehnik, imenujemo UML (*Unified Modeling Language*). V njem so združeni najboljši inženirski pristopi, ki so se izkazali za uspešne pri modeliranju velikih kompleksnih sistemov. Za grafično ponazoritev se v UML uporabljajo različni diagrami, ki so opisani v naslednjih poglavjih.

### 2.4.1 Diagrami primerov uporabe

Diagrami primerov uporabe (*use case diagrams*) predstavljajo komuniciranje med akterji (uporabniki, drugimi sistemi, navideznimi uporabniki) in računalniškim sistemom. Gradniki diagramov primerov uporabe so: primeri uporabe, akterji in relacije (povezave). Diagrami uporabe se uporabljajo za v fazi zajema uporabniških zahtev. Z njimi prikažemo delovanje bodočega sistema. Spodnja slika prikazuje osnovne gradnike diagramov primerov uporabe.




**Slika 2.2: Diagram primera uporabe**

Slika 2.2 prikazuje primer diagrama uporabe, ki vsebuje vse osnovne gradnike in je izdelan z orodjem JDevoloper10g.

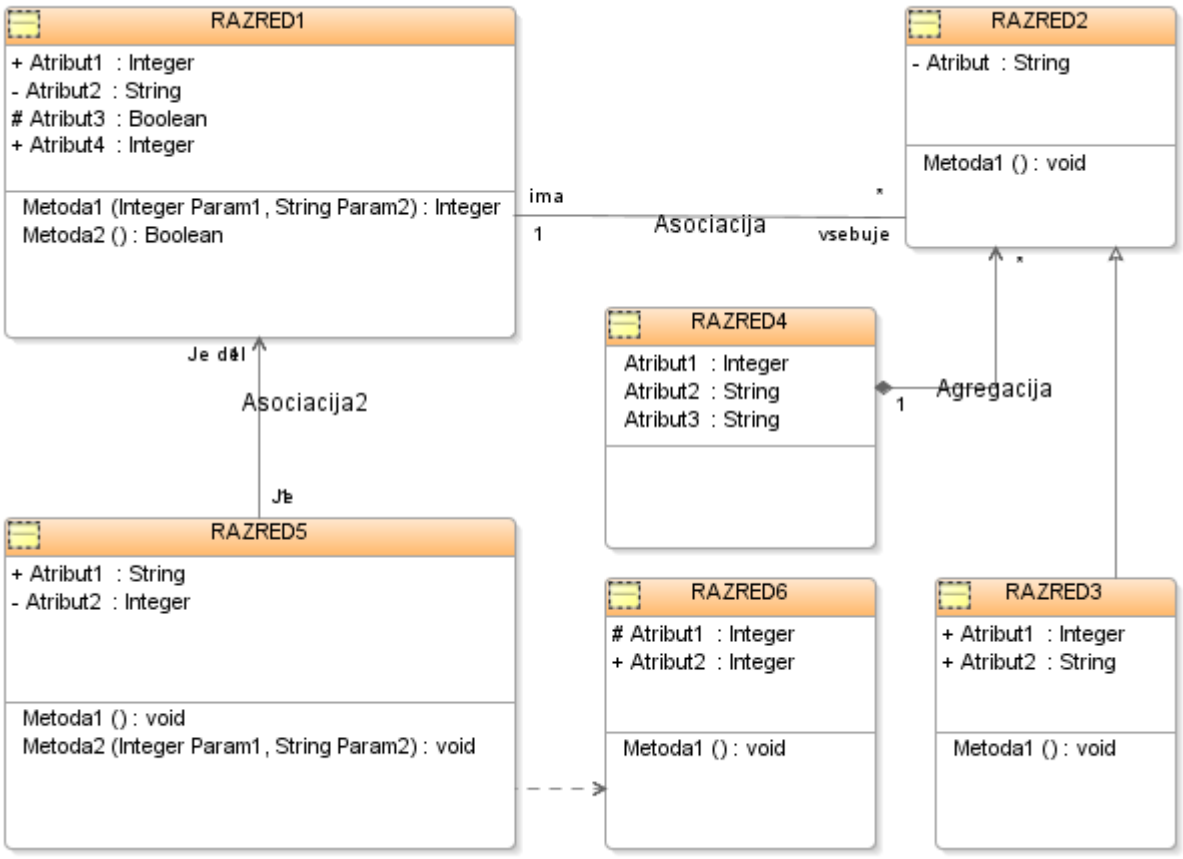
### 2.4.2 Razredni diagrami

Razredni diagrami prikazujejo statično strukturo modela sistema. Prikazuje razrede, njihovo strukturo, metode, atribute in relacije med razredi.

**Razred** predstavlja množico objektov, ki imajo podobno strukturo, obnašanje in relacije. Razred prikažemo s pravokotnikom, ki vsebuje ime razreda, seznam atributov in seznam metod. Razrede lahko združujemo v **pakete**. **Atributom** razreda določimo vidnost, ki je lahko javna (*public* +), zaščitena (*protected* #) ali pa zasebna (*private*), ter tip (število, beseda, datum, ….). Atributu lahko določimo tudi privzeto vrednost. Metodam določimo ime, vidnost (isto kot pri atributih), seznam parametrov (tudi tipi parametrov) in tip metode (možno je, da metoda ne vrne vrednosti).

Relacije v UML prikazujejo logične relacije med elementi in sicer:

- Asociacija,
- agregacija in kompozicija kot posebna primerka asociacije,
- relacija dedovanja,
- odvisnost med elementi.




**Slika 2.3: Razredni diagram**

Slika 2.3 prikazuje primer razrednega diagrama, ki vsebuje vse osnovne gradnike in je izdelan z orodjem JDevoloper10g.

### 2.4.3 Diagrami stanj

Obnašanje sistema opišemo z uporabo diagramov stanj in opisujejo vsa možna stanja objekta in način spreminjanja zaradi dogodkov, ki vplivajo nanj. Diagram stanj lahko izdelamo za primer uporabe, določen razred, ali pa kar za celoten sistem (oz del sistema).

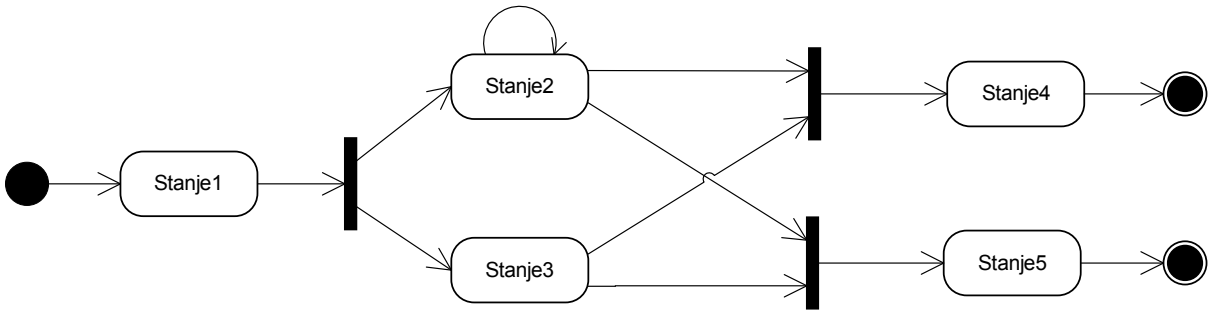
V času svojega obstoja lahko objekt zavzame različna stanja. V vsakem stanju čaka na dogodek, ki ga lahko prestavi v novo stanje. Pomembno pri razumevanju in izdelavi diagramov stanj je razlikovati med aktivnostmi in akcijami. Aktivnosti so procesi, ki trajajo več časa in so sestavljeni, lahko pa jih prekinemo z dogodki. Akcije so procesi, ki niso sestavljeni in se jih ne da prekiniti.

**Stanje** je položaj objekta v njegovi življenjski dobi. Možno je, da imamo v diagramu stanja z istimi imeni. Le ta predstavljajo isto stanje, kopija pa se uporablja za večjo preglednost diagrama. V simbolu stanja so opisane tudi notranje akcije in aktivnosti, ki jih objekt izvaja, ko se nahaja v določenem stanju.

Stanja lahko dodatno oplemenitimo s **Podstanji** (sočasnimi podstanji ali vzajemno izključevalnimi podstanji)**.** Na isti način lahko oplemenitimo tudi podstanja.

**Dogodek** je pojav, ki povzroči spremembo stanja, razdelimo pa jih na več tipov. Obstajata dva posebna dogodka: vhod (*entry*) in izhod (*exit*). Dogodek izhod se izvede, ko zapustimo stanje, pri vhodu pa nastopi dogodek vhod.

Povezavo med stanji ponazorimo s **prehodi**. **Enostavni prehod** je relacija med dvema stanjema, in ga ponazorimo s puščico iz izvornega stanja v ciljno stanje. Splošni prehod ima lahko več izvornih in ciljnih stanj. Takrat govorimo o **kompleksnem prehodu**, ki ga ponazorimo z debelo črto, ki ima lahko eno ali več puščic od stanja do črte in nazaj. Poznamo še **prehode v gnezdena stanja**, ki ponazarjajo prehod v sestavljeno stanje. **Notranji prehod** je prehod, ki ostane znotraj stanja in predstavlja nastop dogodka, ki ne spreminja stanja. Notranji prehod ni enak prehodu iz stanja in nazaj v isto stanje in ne proži vstopnih in izstopnih akcij in ne spreminja stanja.




**Slika 2.4: Diagram stanj**

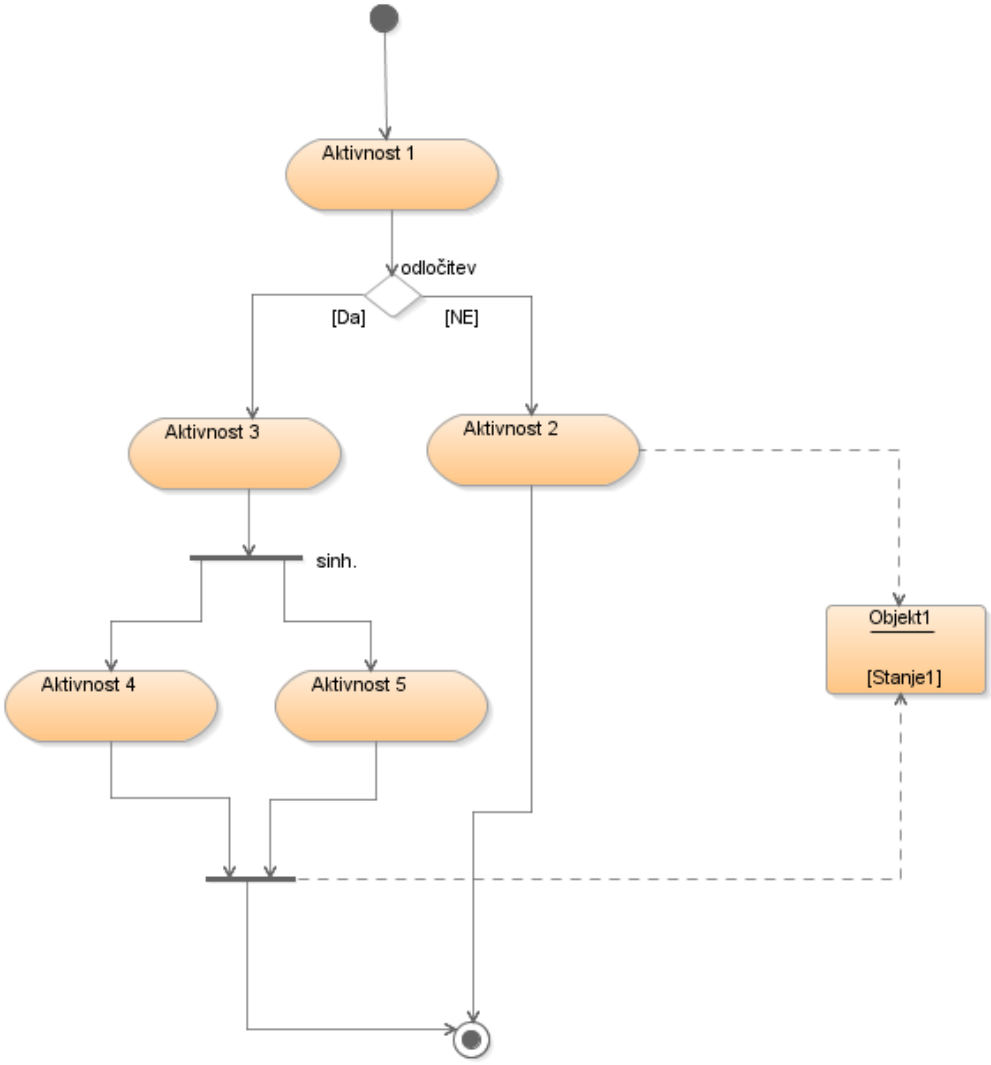
Slika 2.4 prikazuje primer diagrama stanj, ki vsebuje vse osnovne gradnike in je izdelan z orodjem Microsoft Visio.

### 2.4.4 Diagrami aktivnosti

Diagram aktivnosti je poseben primer diagrama stanj, kjer namesto stanj prikazujemo aktivnosti. Z diagramom aktivnosti prikazujmo korake v nekem algoritmu. Diagram se nanaša na razred, na operacijo, na primer uporabe ali na paket.

**Aktivnost** je stanje z notranjo aktivnostjo in vsaj enim prehodom med stanji. Če ima več izhodov, imamo opravka s pogoji. Ko uporabimo pogoje za različne možne prehode, govorimo o **odločitvah**. Kadar imamo opravka s sočasnimi procesi, uporabimo **sinhronizacijsko črto,** s katero prikažemo, da se neka aktivnost nadaljuje šele, ko se zaključijo vsi sočasni procesi. **Steze uporabimo,** da v diagramu aktivnosti ponazorimo, kateri razred, oseba ali oddelek je odgovoren za določene aktivnosti. Steze združujejo diagram aktivnosti z diagrami sodelovanja. V diagramih aktivnosti lahko prikažemo tudi

relacije med aktivnostmi in objekti, prav tako pa lahko prikažemo sprejemanje in pošiljanje signala.



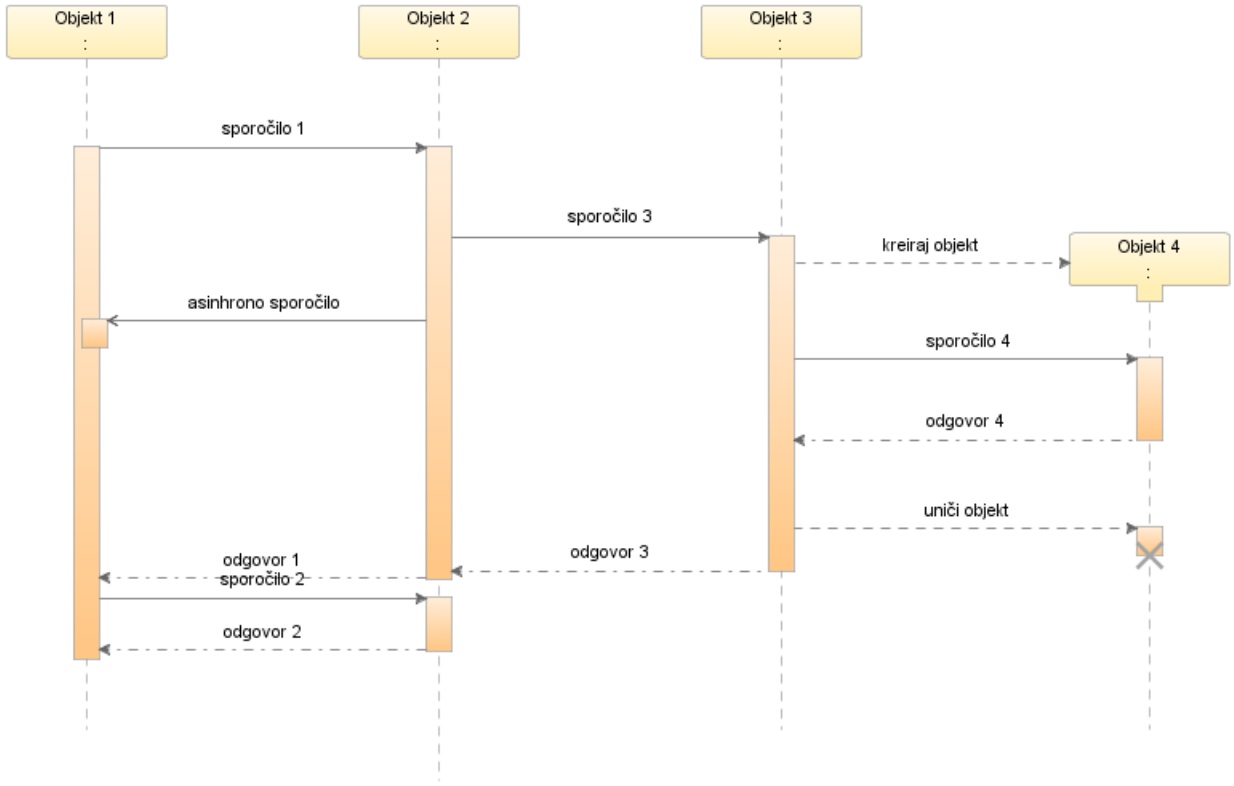

**Slika 2.5: Diagram aktivnosti**

Slika 2.5 prikazuje primer diagrama aktivnosti, ki vsebuje vse osnovne gradnike in je izdelan z orodjem JDevoloper10g.

### 2.4.5 Diagrami zaporedja

Diagrami interakcije so modeli, ki opisujejo sodelovanje skupine objektov pri določenem obnašanju. Zaporedje lahko opazujemo iz časovne (diagrami zaporedja) ali prostorske (diagrami sodelovanja) perspektive. Diagram zaporedja prikazuje, kako objekti sodelujejo in izmenjujejo sporočila v interakcijah na osnovni časovni črti. Vertikalna dimenzija prikazuje čas, medtem ko horizontala prikazuje različne objekte. Čas v diagramu teče navzdol.

Osnovna komponenta diagrama je **življenjska črta objekta** in predstavlja obstoj objekta v času. Na vrhu črte je simbol objekta. Življenjska črta se lahko začne oziroma konča (objekt obstaja v času, ki ga prikazuje diagram), lahko pa gre od vrha do dna diagrama (objekt obstaja tudi izven časa, ki ga prikazuje diagram). Če je objekt zbrisan, prikažemo le to z velikim **X**. Časovni odsek, v katerem objekt izvaja akcijo, imenujemo **aktivacija**. Akcija je lahko direktna ali preko metode. Na diagramu je potrebno prikazati tudi komunikacijo med objekti, kar storimo s **sporočili**.




**Slika 2.6: Diagram zaporedja**

Slika 2.6 prikazuje primer diagrama zaporedja, ki vsebuje vse osnovne gradnike in je izdelan z orodjem JDevoloper10g.

### 2.4.6 Diagrami sodelovanja

V nasprotju z diagrami zaporedja, diagram sodelovanja prikazuje asociacije med objekti. V diagramu ni časovne komponente, zato zaporedje sporočil in sočasne niti označimo z zaporednimi številkami.

Diagram prikazuje objekte, ki sodelujejo pri izvršitvi operacij. Operacija vsebuje argumente in lokalne spremenljivke. Stereotipe <<new>>, <<destroyed>> in <<transient>> uporabimo za med izvajanjem kreiran objekt, uničen objekt oziroma kreiran (ustvarjen) in uničen objekt.

### 2.4.7 Diagrami komponent in arhitekturni diagrami

Diagrami implementacije prikazujejo relacije med programskimi in strojnimi komponentami v sistemu. Z njim prikazujemo smeri premikov komponent in objektov v porazdeljenem sistemu. Vključujejo strukturo izvorne kode in strukturo izvajanja implementacije. Ločimo dva tima diagramov [8]:

- Diagrami komponent - prikazujejo strukturo kode.
- Arhitekturni diagrami- prikazujejo strukturo izvajanega sistema.

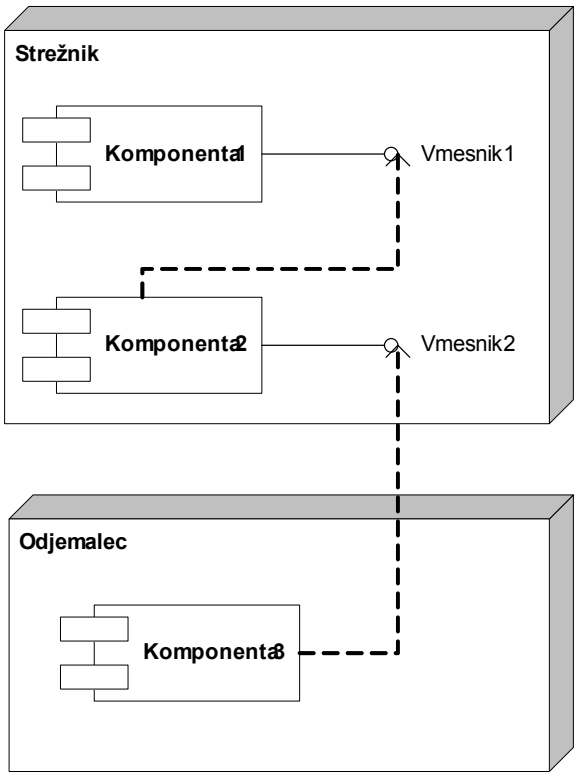
**Diagram komponent** prikazuje odvisnost med programskimi komponentami. Vključuje izvedljive komponente, izvorno in binarno kodo. Diagram je slika, kjer so vozlišča komponente, povezave pa relacije med njimi. **Komponenta** predstavlja del programske kode. Komponente so podobne razredom, saj imajo oboji imena, realizirajo lahko vmesnike, lahko so v medsebojnih odvisnostih, vendar pa lahko med komponentami in razredi najdemo nekaj pomembnih razlik [8]:

- Razredi predstavljajo logične abstrakcije, komponente pa fizične.
- Komponente predstavljajo fizično pakiranje logičnih razredov in so na drugem nivoju abstrakcije.
- Komponente imajo samo operacije, ki so dosegljive skozi vmesnike. Razredi direktno podpirajo operacije in atribute.

**Vmesnik** je zbirka operacij, ki se uporablja za zagotavljanje storitev razreda ali komponente. Vsi pomembni komponentni modeli uporabljajo vmesnike kot povezavo med komponentami. Z uporabo komponent razčlenimo fizično implementacijo tako, da definiramo vmesnik. Nato določimo komponente, ki te vmesnike realizirajo. Tak mehanizem omogoča izgradnjo sistema, ki je neodvisen od lokacij in je zamenljiv.

Z **arhitekturnim diagramom** predstavimo (fizično) arhitekturo sistema. Arhitekturni diagram prikaže konfiguracijo izvajalnega elementa in programskih komponent, procesov in objektov, ki se nahajajo v njem.

Vsako **vozlišče** v diagramu predstavlja enoto neke vrste in je izvedljiv objekt, ki uporablja pomnilnik in ima sposobnost procesiranja. Izvedljivi primerki, objekti in primerki komponent se nahajajo v vozlišču primerkov. Vozlišča so povezana s pomočjo asociacij z ostalimi vozlišči, ki prikazujejo komunikacijske poti med vozlišči. Primerke lahko lociramo znotraj drugih primerkov, tako da so objekti v procesih, le ti v komponentah, komponente pa so v vozliščih.




**Slika 2.7: Arhitekturni diagram kot primer diagram komponent**

Slika 2.7 prikazuje primer diagram komponent in sicer arhitekturni diagram. Diagram predstavlja komponente na različnih lokacijah (strežnik in odjemalec) ter dostop do njih preko vmesnikov. Diagram je izdelan z orodjem Microsoft Visio.

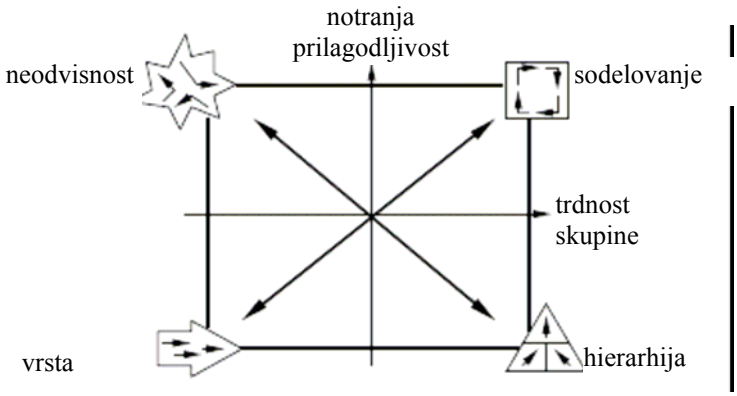
# 3 RAVNANJE S KADRI

## 3.1 Delo z ljudmi

### 3.1.1 Uvod

Vodenje projektov razvoja programske opreme vedno obsega veliko raznolikih organizacijskih vidikov, kot so oblikovane in polnjenje projektne skupine, določanje vlog (delovnih programov) članom skupine, način razvoja IS, ustreznost vodenja, medsebojno komuniciranje na delu, izobraževanje, sprejemanje novih tehnologij, kultura organizacije, … [3]

Temelji vsake organizacijske podobe pri razvoju programske opreme so osnova načela, ki so prikazani na sliki 3.1. Izkušnje dokazujejo, da štirje osnovni principi v kotih kvadrata na sliki 3.1 določajo izvor večine tipov organizacijskih kultur pri razvoju programske opreme. Iz tega sledi, da je za projektne vodje zelo pomembno, da popolno razumejo pomen in pomembnost principov. Principi se medsebojno izključujejo in vsi imajo tako prednosti kot tudi slabosti [3].




**Slika 3.1: Osnovna načela organizacije projektov razvoja programske opreme [3].**

Pomemben del uspešnosti projektov in projektnega vodenja je delo z ljudmi. Ljudi, ki delujejo znotraj podjetja, organiziramo na različne načine. Cilj je jasen. Procese v podjetju je treba organizirati tako, da bodo ob enakih učinkih vložki čim manjši. Zaposlene organiziramo v različne skupine, kjer je potrebno vzpostaviti nadzor na delom. Skupina mora biti povezana na znotraj in navzven, tu pa se pokaže sposobnost komuniciranja med ljudmi in skupinami. Posebnost povezave med skupinami predstavlja razmerje s podizvajalci, kjer je potrebno vzpostaviti povezavo s skupino izven podjetja.

### 3.1.2 Oblikovanje skupin

V življenju se v večini srečujemo s tako imenovanim skupinskim delom. Za nek dosežek ali opravljeno nalogo sta ponavadi potrebni vsaj dve osebi, ki morata za uspešno delo sodelovati. Kakor imamo opravka z več kot enim človekom, že lahko govorimo o skupini in če ti osebi opravljata še delo, lahko govorimo o skupinskem delu ali delu v skupini. V vsaki skupini pa je prisoten pojem delo z ljudmi, saj je potrebno delo opravit s sodelavcem, s katerim je potrebno komunicirati, se dogovarjati in si pomagati.

### 3.1.3 Splošno o skupinah

Skupina je socialna združba določenega števila posameznikov z istimi potrebami in interesi, ki jo zadovoljijo z uresničevanjem skupnih ciljev [16].

Iz teorije psihologije vemo, da so za delo v skupinah odločilni trije pogoji:
1. Interesi skupine morajo biti povezani – to je pogoj za močno kooperacijo in interakcijo.
2. Člani skupine si morajo prizadevati za dosego skupno odločenega ali izbranega cilja.
3. V skupini mora biti močna osebna (notranja) in zunanja (medosebna) motivacija.

Skupine, ki jih imamo, so zelo različne po svojem delovanju, nastanku in raznih drugih lastnostih.

Delitev na vrste skupin po različnih sodilih [16]:
- primarne in sekundarne,
- neformalne in formalne,
- delovne,
- male ali velike.

Vsekakor so zelo zanimive skupine, ki nastanejo spontano. Te skupine nastanejo zaradi istih interesov in so v določenih pogledih izredno uspešne. Zelo veliko takšnih skupin nastane iz skupnih interesnih dejavnosti (planinarjenje, kolesarjenje, slikarstvo). Značilnost teh skupin je ponavadi velika homogenost in tudi učinkovitost. Glavna značilnost teh skupin je ta, da delujejo v presledkih ali občasno in največkrat je to tudi glavni vzrok za homogenost. Zanimivo je ali se lahko (delovno) vzdušje takšnih skupin ustvari tudi v delovni skupni, ki deluje vsak dan po osem ali celo več ur?

Za učinkovitost skupine mora biti vsekakor prisotni sproščenost in velika mera zaupanja. Prevelika mera svoboščine ponavadi pomeni tudi razrahljanje skupine in celo njen razdor, katerega posledica je slaba učinkovitost in nezadovoljstvo članov skupine.

Delovanje skupine je odvisno od stopnje soudeležbe vseh tako v delu kot tudi v učinkih. Za primer navajam odločitveni problem, kjer nastopa nosilec odločitve in skupina oseb, ki pomaga pri izbiri prave variante. Lastnik odločitvenega procesa je odgovoren za odločitev, vendar pa se z oblikovanje skupine odgovornost prenese tudi na vse člane odločitvene skupine in to ne delno, ampak 100% na vsakega [24].

Primer kaže, da je potrebno v skupni imeti vodjo (mogoče jih je lahko tudi več), vendar pa je pomembno, da je v skupini vzpostavljen občutek pomembnosti vseh članov pri delu in tudi pri nagrajevanju.

### 3.1.4 Nadzor nad delom v skupinah - vodenje

Praktično si ne predstavljamo delovne skupine, kjer smo vsi enaki. V vsaki skupini je potrebno izpostaviti vodje. To so ljudje, ki od nadrejenih prejemajo določene naloge in so za izvedbo tudi odgovorni. Naloga vodje je, da skupina na bolj smotrn način izvede nalogo, ki ji je bila dodeljena, vendar pa je odgovornost izključno na strani vodje. Vodja lahko z delitvijo nalog odgovornosti delno prenese na člane skupine.

### 3.1.5 Načini vodenja

Obe skrajnosti vodenja predstavljata [16]:

- **demokratski slog vodenja**

Pri demokratskem vodenju je poudarek na dobrih razmerjih, dobrem sodelovanju in visoki učinkovitosti skupine. Vodja ima v takih skupinah položaj » prvega med enakimi«, kar pa ne pomeni, da v izjemnih primerih ne prevzame tudi nekaterih lastnosti avtokratskega vodenja.

- **avtokratski slog vodenja**

Pri tem slogu je osrednja in dominantna osebnost vodja skupine, ki želi vse glavne probleme rešiti sam na način, ki njemu najbolj ustreza. Od članov skupine zahteva poslušnost ter sprejemanje njegovih idej.

Oba načina sta skrajni obliki, v praksi pa poznamo veliko vmesnih izvedb.

Kako obdržati (dobre) delavce v podjetju?

V podjetjih za razvoj programske opreme je še posebej pomembno zagotoviti, da nam delavci ne odhajajo, saj je zelo težko pridobiti nove delavce, potem pa sledijo še uvajanje in morebitna izobraževanja, ki predstavljajo podjetju dodaten strošek. V osnovi lahko delimo dve skrajni možnosti:

- Podpis konkurenčne klavzule in pogodbe, ki delavcem preprečuje menjavo zaposlitve ali pa mora v tem primeru plačati podjetju primerno odškodnino.
- Zagotovitev takšnih pogojev dela in nagrajevanje, ki delavce ne vodi v razmišljanje po zamenjavi službe, če pa do tega pride, pa tekmeci ne morejo zagotoviti podobnih pogojev in nagrajevanja.

Prva možnost je sicer za podjetje lažja, vendar pa mora podjetje tudi v takšnem primeru nagraditi delavca (višja plača, sofinanciranje izobraževanja, nadomestilo v primeru uveljavljanja konkurenčne klavzule). Poleg tega lahko na delavce takšni podpisi pogodb vplivajo negativno, saj vzbujajo občutek prisile, vezanosti in nesvobode.

Druga možnost je za podjetje veliko boljša, vendar pa je potrebno priti do takšnega nivoja pogojev dela v podjetju, da ljudje niti ne čutijo potreb po odhodu. Najpomembnejši je sistem nagrajevanja. Kljub temu, da idealnega univerzalnega sistema za nagrajevanje ni, je potrebno

zagotoviti sistem, ki ne vzbuja pri delavcih občutka zapostavljenosti in manjvrednosti v primerjavi z drugimi. Primerno okolje je nemogoče vzpostaviti v trenutku, vendar pa je za to potreben čas in sodelovanje vseh delavcev. Kakšno je stanje pri delavcih, nam pokažejo analize, ki jih naredimo na podlagi vprašalnikov o zadovoljstvu pri delu. Primer takšnega vprašalnika nam predstavlja preglednica 3.1. Vprašanja se nanašajo na doživljanje zadovoljstva pri delu. Na vsako vprašanje odgovorite tako, da ocenite, koliko ste zadovoljni s posameznimi vidiki svojega dela.

| **Kako ste zadovoljni: (obkrožite ustrezno stopnjo)** | **zelo nezadovoljen** | **nezadovoljen** | **nekaj vmesnega** | **zadovoljen** | **zelo zadovoljen** |
|---|---|---|---|---|---|
| z vsebino in primernostjo dela | 1 | 2 | 3 | 4 | 5 |
| z ravnjo odgovornosti pri delu | 1 | 2 | 3 | 4 | 5 |
| s samostojnostjo pri opravljanju dela | 1 | 2 | 3 | 4 | 5 |
| z možnostmi za napredovanje | 1 | 2 | 3 | 4 | 5 |
| z možnostjo osebnega razvoja | 1 | 2 | 3 | 4 | 5 |
| s količino in pogostostjo priznanj in pohval za dobro opravljeno delo | 1 | 2 | 3 | 4 | 5 |
| z uspešnostjo pri delu (občutkom uspešnosti) | 1 | 2 | 3 | 4 | 5 |
| z neposrednim nadrejenim | 1 | 2 | 3 | 4 | 5 |
| s sodelavci | 1 | 2 | 3 | 4 | 5 |
| s plačo | 1 | 2 | 3 | 4 | 5 |
| s svojo poklicno kariero | 1 | 2 | 3 | 4 | 5 |
| z delovnimi razmerami | 1 | 2 | 3 | 4 | 5 |
| s svojim statusom v podjetju | 1 | 2 | 3 | 4 | 5 |

**Preglednica 3.1: Primer vprašalnika za zaposlene [27]**

Takšne vprašalnike je potrebno nato analizirati in na podlagi analize tudi pripraviti morebitne ukrepe. Ukrepe je potrebno nato izvajati in kasneje spremembe s ponovljenimi anketami meriti.

### 3.1.6 Nagrajevanje, izobraževanje

Pomembno vlogo v delovanju podjetij ima tudi sistem nagrajevanja delavcev. V to ni zajeta le plača, ampak vse dobrine, ki jih lahko podjetje nudi delavcu in izhajajo iz njegovega bolj ali manj uspešnega dela.

V večini podjetij plače in nagrade opredeljujejo pravilniki, ki so usklajeni z zakonom in kolektivnimi pogodbami. Sistem izračuna plače in nagrad mora biti razumljiv, saj mora delavec vedeti, zakaj je neko nagrado dobil in zakaj ne. To mu je lahko v naslednjem mesecu motivacija za doseganje boljših rezultatov dela in s tem tudi prednost za podjetje.

Ena od značilnosti nagrajevanja, ki zelo vpliva na motivacijo, je to ali obravnavamo plače kot tajne ali javne. Naši zakoni ne opredeljuje, da morajo biti plače javne. Višine plač v podjetju

mora na podlagi nekaterih dejavnikov (izobrazba, učinkovitost, pomembnost,…) usklajevati ravnateljstvo oz oddelek, ki je zadolžen za izvajanje plačne politike v podjetju. Pravičnega nagrajevanj za delo ni, vseeno pa mora podjetje strmeti k temu. V splošnem imamo dve skrajnosti plačnega sistema v podjetjih:

- Pregleden sistem nagrajevanja tako na podlagi izobrazbe, delovne uspešnosti, kot tudi kriterijev pomembnosti posameznega delavca za podjetje.
- Netransparenten način, ki temelji na pravilnikih, kjer so posamezni delavci z doseganjem ciljev (npr. starosti) v privilegiranem položaju.

V praksi je večina sistemov nekje vmes, vendar po mojem mnenju velja sledeče pravilo:

- Za sisteme, ki so bližje prvemu, so lahko plače in nagrade javne, saj lahko to na delavce vpliva motivacijsko. Delavci v praksi vidijo, da so ljudje, ki imajo večji učinek in so za podjetje pomembni, bolje nagrajeni. To jih motivira, da tudi sami dosežejo takšen nivo.
- Za sisteme, ki pa so bližje drugemu sistemu, pa je bolje, da so plače tajne, saj lahko delavcem javnost plač jemlje voljo do dela. Seveda ne more vsak delavec meriti uspešnost, vseeno pa veljajo nekateri občutki o tem, kdo naredi več, kdo je bolj učinkovit, in v takšnih primerih je bolje, da se ne ve, da nekdo zasluži veliko, čeprav njegov učinek (po mnenju prizadetih) ni najboljši.

Navedel bi primer, ko je nagrada delavcu pomenila motivacijo za delo v naprej, nato pa z informacijo (nagrade so pač javne), da je takšno nagrado dobil tudi drugi delavec, pomenila ravno nasprotno in vzbudila negodovanje in slabo voljo v podjetju. Delavcu so dodelili nagrado za dobro opravljeno delo. Delavec je bil zadovoljen in vse je bilo v redu, dokler ni do njega prišla informacija, da je isto nagrado prejel še njegov sodelavec, za katerega pa je bilo znano, da ima veliko reklamacij, v svojem delovnem času pa igra tudi računalniške igrice. Vodja, ki je delal predlog za nagrajevanje, je v tem primeru nepravično predlagal oba delavca. To je povzročilo negodovanje in slabo voljo, ki pa se v primeru, da bi bila nagrada tajna, ne bi dogodila.

Seveda tajnost plače ne sme biti razlog za nepravično dodeljevanje nagrad in plač, se pa napake vodij v sistemih, ki to dopuščajo, dogajajo, zato je morebiti bolje v takšnem primeru, da ne vemo za plače kolegov.

### 3.1.7 Povezanost (komuniciranje) v skupinah

Učinkovitost dela v skupinah je odvisna od medsebojne povezanosti skupine. Še pred nekaj časa je bilo pravilo, da so člani delovne skupine tudi fizično prisotni blizu drug drugega (npr. v sobi). Z novimi tehnologijami, ki skrbijo za povezljivost, pa to ni več ključnega pomena. Še posebej se to kaže pri razvoju programske opreme, saj je vse več primerov, ko se razvijalci sploh ne vidijo in ne poznajo razen preko elektronskih medijev, a vseeno je plod njihovega dela en sam izdelek.

Pri komuniciranju v skupinah je pomembno, da ves čas člani skupine vedo tisto, kar morajo vedeti, kako zagotoviti te informacije, pa je v glavnem lahko stvar tudi iznajdljivosti projektne skupine.

Viri informacij povezljivosti skupine pri razvoju programske opreme so:
- formalni sestanki,
- neformalni sestanki,
- elektronska pošta,
- video in glasovne povezave,
- bilteni,
- projektna dokumentacija (listine).

Preko teh virov projektna skupina komunicira. Seveda so glavni nosilci povezljivosti sestanki, brez katerih si izpeljavo projektov in nalog praktično ne moremo predstavljati. V tem primeru sem ločil tudi formalne in neformalne sestanke, saj bi v nadaljevanju rad predstavil razliko in smisel neformalnih sestankov, ki so ponavadi še učinkovitejši kot formalni. Postavki tri in štiri lahko povežemo kot delo na daljavo. Nosilca sta neodvisna od lokacije (razmestitve) članov projektne skupine, razen če preko e- pošte ne komuniciramo v smislu šale o Muju in Hasu, ko sta od sosedov dobila telefon (Pridi na balkon, da ti nekaj povem).

**Formalni sestanki** so glavno povezovalno telo vsake projektne skupine pri razvoju programske opreme. Sestanki so kazalec povezljivosti skupine in informator o napredovanju vsakega projekta. Vse spremembe projektnega načrta morajo biti naznanjene na sestankih projektne skupine. Pri razvoju programske opreme je potrebno izpostaviti tudi sestanke z naročniki, ki so povezovalno telo med projektno skupino in naročnikovimi željami.

Sestanki manjših podmnožic projektne skupine so tudi obvezni del, saj v večjih projektih vse informacije niso pomembne za vsakogar, zato velikokrat sestanki na določeno temo potekajo v ožjih zasedbah.

Najprej je potrebno sestanke sklicati. Seveda so izjema sestanki, ki se ponavljajo v naprej predvidenih časih in imajo v naprej določeno vsebino. Sklic sestanka mora biti pravočasen. Pri sklicu sestanka je potrebno dodati vsebino sestanka, kjer je vsebovano tudi morebitno gradivo, da se lahko udeleženci na sestanke učinkovito pripravijo. Vodenje sestanka je ponavadi domena tistega, ki je vodja za tematiko, ki je obravnavana na sestanku. Za sestanke celotne skupine je zadolžen vodja projekta. O vsakem sestanku se mora pisati zapisniki, ki mora biti vsem udeležencem dostopen v čim krajšem možnem času.

**Neformalne sestanke** ločimo od formalnih po naslednjih značilnostih:
- Neformalni sestanki nimajo sklicatelja sestanka in ne potekajo po nekem dnevnem redu.
- Potekajo na neformalnih mestih (športni tereni, konference, prireditve, predstavitve, gostilne, bari, okrepčevalnice).
- O njih ne vodijo evidenc in zapisnikov.

Takšni sestanki so lahko velikokrat dober vir informacij, do katerih pridemo spontano. Na takšnih sestankih je komuniciranje bolj sproščeno in velikokrat preide v konkreten in učinkovit dogovor med udeleženci takšnega sestanka.

Moje mnenje je, da so pri razvoju programske opreme takšni sestanki nujno potrebni, saj sodelujoči v projektni skupini, v katerem sodelujem, veliko dobrih zamisli izpostavijo ravno na takšnih sestankih.

**Elektronska pošta** je medij brez katerega si danes ne moremo več predstavljati sodelovanja projektnih timov. Preko pošte je dosežena povezljivost projektne skupin, ki lahko deluje tudi na oddaljenih krajih. Preko elektronske pošte si lahko izmenjujemo razne dokumente (listine), preko nje lahko potekajo tudi korespondenčni sestanki.

V novejšem času **video povezave** vse bolj dopolnjujejo **glasovne povezave** (telefon). To je medij, preko katerega lahko projektni tim izmenjuje informacije. Je idealen za delo na daljavo, saj ni potrebno, da se akterji dobivajo na skupnih mestih, ampak enostavno komunicirajo preko avdio video povezave. Preko te povezave je možna izvedba sestankov tudi na daljavo.

Večje projektne skupine težko komunicirajo med seboj in na sestankih, na katerih so prisotni vsi. Zagotovljen mora biti medij, kjer lahko vsak dobi informacijo in je jasno, s kom se lahko pogovori v primeru določenega problema. **Bilten** lahko izhaja bolj pogosto ali pa ne, odvisno od značilnosti projekta.

**Projektna dokumentacija** je srce vsakega projekta. Pomembno je, da je zbrana na enem mestu in dostopna vsem, ki jo potrebujejo. Zagotoviti je potrebno arhiviranje le te in kontrolo inačic (*version management*), saj je potrebno preprečiti, da sprotne verzije niso na skupnem mestu.

### 3.1.8 Podizvajalci in partnerji

Še bolj zapleteno od organizacije projektnega tima v podjetju je vodenje projekta, ki poteka v več podjetjih, znotraj katerega je delo organizirano po projektih. Poznamo dve vrsti povezljivosti:
- glavni izvajalec in podizvajalec,
- partnerski odnos (eden od partnerjev je glavni).

V takšnih primerih organizacije projekta imamo dve možnosti in sicer:
- Skupaj organiziramo projektno skupino, ki deluje enotno.
- V vsakem podjetju imamo projektno skupino, skupaj pa zagotovimo komuniciranje med njima.

Seveda je prva enostavnejša, saj jo obravnavamo kot novo projektno skupino, ki deluje neodvisno. V drugem primeru pa je potrebno vzpostaviti raven komuniciranja med projektnimi timi, kar pa ni ravno najlažje.

## 3.2 Izbira delavcev

### 3.2.1 Uvod

Izbira delavcev (*kadrov*) je za združbe eden od osnovnih odločitvenih problemov. Uspešni delavci so pogoj za uspešno delovanje podjetja in izbira le teh v večini primerov ne pomeni lahke in nepomembne odločitve. Še posebej pomembno je to pri uvajanju nove tehnologije. Uvajanje nove tehnologije ni dosti vredno, če niso za njeno uporabo vnaprej zagotovljeni ustrezno pripravljeni zaposleni [10].

Delovna sila je ena od osnovnih prvin poslovnega procesa [10]. Pod delovno silo si prestavljamo ljudi, ki s svojim delom in znanjem (s pomočjo drugih prvin) izvajajo opravke. Še vedno velja dejstvo, da so ljudje s svojimi lastnostmi najbolj pomemben proizvajalni tvorec v podjetjih.

Za skrb nad delavci je v podjetjih zadolžena kadrovska funkcija, ki skrbi za celotni spekter zadev okrog zaposlenih v podjetju. Ena od osnovnih je načrtovanje in izbira delavcev. Za opravljanje določenih nalog je potrebno izbrati primerne delavce. Izbira je le začetek delovanja v podjetju, zato so pomembne tudi naloga, ki skrbijo za delavce v času delovanja. Vse te naloge bi lahko tudi združili v pojem delo z ljudmi. Pod delo z ljudmi lahko povežemo naslednje naloge, ki jih opravlja kadrovska funkcija [10]:

- načrtovanje sestave in števila delavcev,
- pridobivanje delavcev,
- poklicno usmerjanje in izbiranje delavcev,
- spremljanje razvoja delavcev,
- izobraževanje delavcev,
- ocenjevanje delavcev,
- motiviranje in nagrajevanje delavcev,
- obveščanje delavcev,
- odpuščanje delavcev,
- varovanje delavcev pred poškodbami,
- reševanje socialnih vprašanj,
- vodenje evidence o delavcih.

Specifično pa velja, da je pri razvoju programske opreme, med prvinami poslovnega procesa, delovna sila oz. ljudje še bolj v ospredju. Zato moramo v podjetjih (ali delih podjetja), kjer se ukvarjajo z razvojem programske opreme, delavcem posvečati še večjo pozornost. Glavna značilnost trga razvoja programske opreme v Sloveniji (tudi svetu) je ta, da delovne sile te vrste primanjkuje. To le otežuje tako izbiro ljudi, kot tudi delo z njimi.

Carly Fiorina, direktorica Hewlett-Packarda, je izjavila sledeče o pomembnosti ljudi v podjetju: "Najboljša strategija, najboljši finančni načrt in največji prihodki na svetu so" zgolj prehodnega značaja, če niso "zasnovani na ljudeh".

Ko imamo v podjetju primerne ljudi in dobro skrbimo za njih, pa je potrebno, da je njihov izkoristek kar največji. Zagotoviti moramo njihovo dobro delovanje, to pa storimo tako, da organiziramo kakovostne projektne skupine, ki bodo skrbele za izvajanje konkretnih nalog.

### 3.2.2 Pomen izbire in teorija

Osnovni predpogoj za izbiro najprimernejših sodelavcev je usklajenost zahtev dela in delovnega mesta ter možnosti posameznika [27]**.** Če želite kar najbolje izkoristiti potenciale delavce na specifičnih delovnih mestih, morate zagotoviti, da bodo na ta mesta nameščeni posamezniki, ki bodo imeli:

- takšna **znanja, sposobnosti in zmožnosti,** da bodo **zmogli** dosegati zastavljene cilje in učinkovito izkoriščati zmogljivosti, ki jih organizacija zagotavlja na konkretnem delovnem mestu,
- takšno **motivacijo, ambicije in interese,** da bodo **hoteli** dosegati poglavitne cilje in vzdrževali ključne standarde, ki jih organizacija pričakuje na konkretnem delovnem mestu,
- takšne **osebnostne in vedenjske lastnosti,** da se bodo **učinkovito** prilagajali okolju, pritiskom, sodelavcem, strankam, režimu dela, načinu vodenja ter spremembam v poslovnih procesih.

Samo pri delavcih, ki ustrezajo tem pogojem, lahko računate na nadpovprečno učinkovitost, motiviranost in zadovoljstvo pri delu. S sistematičnim pristopom k ravnanju s sposobnostmi zaposlencev lahko z istimi ljudmi dosežete bistveno večje učinke in uresničite bistveno zahtevnejše cilje.

Izbira ljudi je najpomembnejša dejavnost v podjetju. S pravimi ljudmi, ki tvorno sodelujejo, bomo še tako zahteven projekt speljali do konce. Celostna izbira delavcev se sestoji iz treh delov:

- pridobivanje kandidatov,
- izbirni intervjuji,
- izbor ustreznega(ih) kandidatov (selekcija).

### 3.2.3 Pridobivanje kandidatov

Cilj pridobivanja kandidatov je pridobiti določeno število ustreznih (po nekaterih kriterijih) kandidatov, izmed katerih lahko izbiramo. Seveda je število kandidatov odvisno od več dejavnikov. Ne moramo si predstavljati, da bomo na razpis za ravnatelja računalniškega podjetja pridobili 50 kandidatov, izmed katerih bomo izbrali ustreznega. Prav lahko pa se zgodi, da bomo na delovno mesto programerja dobili več kot 10 prijav.

V osnovi si lahko postavimo naslednja vprašanja, ki nam dajo osnovne aktivnosti potrebne za pridobivanje kandidatov [27].

- Najprej moramo določiti, koga sploh potrebujemo?
- Kako pritegnemo največ pravih kandidatov?
- Katere informacije so nujne in katerih ne potrebujemo?
- Kako bomo čim bolj objektivno analizirali posameznikove lastnosti?

Tako ločimo naslednje aktivnosti:

- Analiziranje in opis delovnih mest.
- Oblikovanje objave ali razpisa.

- Zbiranje informacij s kadrovskim vprašalnikom.
- Psihološki testi.

### 3.2.4 Izbira

#### 3.2.4.1 Splošno

Izbira delavcev za razvoj programske opreme je posebej obsežno poglavje. Probleme izbire delavcev računalniških podjetjih sem opisal tudi v svoji seminarski nalogi pri predmetu Poslovne funkcije in odločanje. Problem nastane pri kalkulaciji ali se splača podjetju izobraziti ustrezne delavce ali pa jih poiskati na trgu in za njihovo delovno silo plačati ustrezno večjo ceno.

Podjetja za razvoj programske opreme morajo stalno skrbeti za ustrezno tehnološko podprtost svojih delavcev. Ena od možnosti je tudi zaposlovanje vedno novih delavcev z ustreznimi znanji. Druga možnost je izobraževanje obstoječih delavcev, kjer je problem začetne neizrabe delovne sile.

Izobraževanje je ena od možnosti pridobivanja delavcev z ustreznim znanji. V večini primerov je to draga možnost, saj so izobraževanja draga, poleg tega pa je v času odsotnosti potrebno upoštevati tudi okoličinske ali oportunitetne stroške (Koliko bi oseba v času odsotnosti naredila).

#### 3.2.4.2 Vrednotenje kandidatov

Vrednotenje ustreznosti delavcev za razvoj programske opreme je zelo težko. Največji dodatek predstavljajo priporočila in izkušnje. Veliko pomenijo tudi ustrezna potrdila o udeležbi na določenih izobraževanjih ali pa tudi potrdila o znanju (certifikati).

Problem, ki nastane, je predvsem vključevanje v skupino, saj je lahko neka oseba odličen programer ali analitik, pa se ne more ustrezno vključiti v skupino (zaradi osebnostnih lastnosti, ki niso dopolnjujoče z lastnostmi drugih), kljub temu, da je v prejšnji skupini (zaposlitvi) odlično deloval.

#### 3.2.4.3 Preizkusi

Pri izbiri ljudi imamo na voljo tudi ustrezne preizkuse. Že v času razgovora, se lahko z kandidati dogovorimo za preizkus znanja. To lahko naročimo pri podjetju za kadrovsko svetovanje (*head hunters*). Pri nas so na voljo podjetja Hill international, Kragelj & Kragelj,… .

Njihov test pri individualnem svetovanju kariere je tako imenovana psihometrična analiza, v kateri dobijo oceno kandidata, ki ga lahko posredujejo podjetjem.

Psihometrična analiza:

Test je sestavljen iz petih delov:

- Strukturirani osebnostni vprašalnik: ugotavljamo 13 osebnostnih lastnosti kandidata, ki so se izkazale kot ključne v podjetniškem okolju; na podlagi teh pa lahko ocenimo tudi prisotnost prodajnega talenta, sposobnosti vodenja, skupinske primernosti, volje do storilnosti in potrebe po socialni sprejetosti.
- Profil usposobljenosti: sestavljen je iz nalog testa inteligentnosti in ocenjuje zmogljivost kratkoročnega delovnega spomina ter sposobnosti verbalnega razumevanja, percepcije, miselne prožnosti, logike, analitike in sposobnost celostnega pregleda nad miselnim problemom.
- Koncentracija in učinkovitost pri delu: merimo natančnost in učinkovitost (hitrost) pri opravljanju rutinskih nalog.
- Poklicni interesi: ugotavljamo posameznikov interes glede na vsebino in vrsto dela (vodilen / podrejen položaj, fleksibilni / urejeni delovni pogoji, miselno / fizično delo, ustvarjalen / praktičen delovni slog, pomen zasebnega / poklicnega življenja) in glede na področje dela (ekonomija, tehnika, delo z ljudmi, ekologija, politika in javno življenje, kultura in umetnost).
- Test prodajnih sposobnosti: ocenjujemo prodajno sposobnost v fazah prodajnega postopka (prvi stik s kupcem, otvoritveni razgovor, predstavitev proizvoda, zaključna faza) in ugotavljamo, kateremu od štirih tipov prodajalca kandidat najbolj ustreza (antitip, "trdi" prodajalec, svetovalec, prodajalec).

**Psihometrično testiranje**

S psihološkimi testi, ki ustrezajo zelo visokim zahtevam objektivnosti in zanesljivosti, analiziramo kandidatove sposobnosti, izdelamo profil izraženosti približno 18 osebnostnih lastnosti, opredelimo 6 sklopov njegovih prevladujočih poklicnih interesov in ambicij, ugotovimo, katere od devetih skupinskih vlog pri njem prevladujejo... in zagotovimo še veliko drugih informacij, ki jih potrebujete [27].

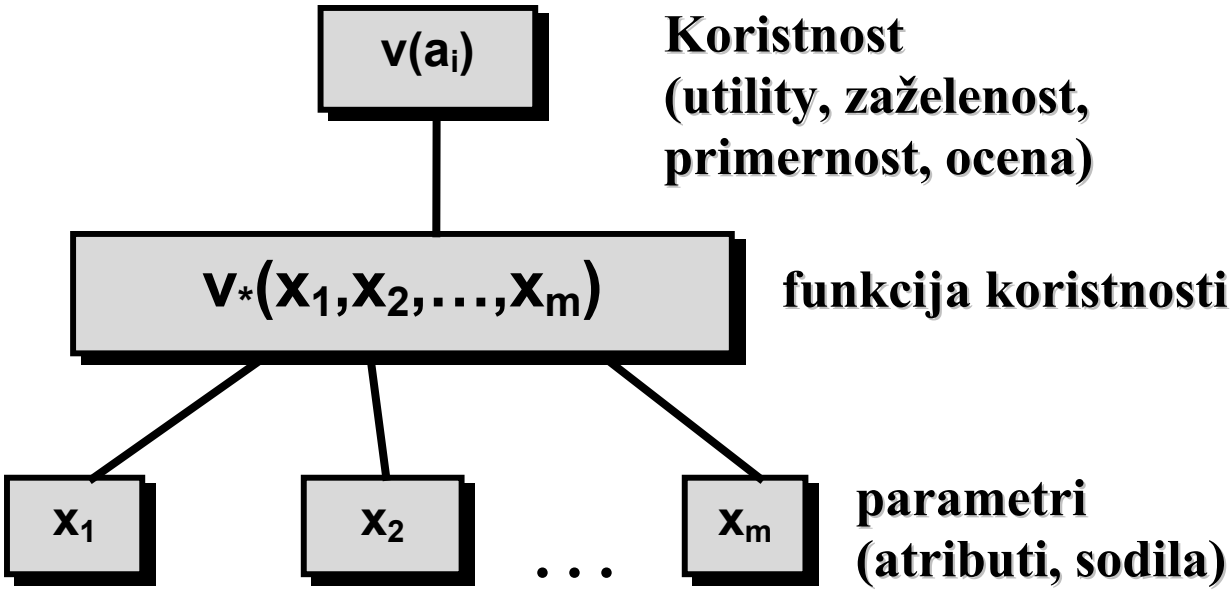
# 4 PRIMER IZBIRE DELAVCA

## 4.1 Opredelitev primera

### 4.1.1 Uvod

V podjetjih, v katerih se ukvarjamo z razvojem informacijskih sistemov, je izbira ustreznih ljudi zelo težavna naloga, saj nosilcev delovne sile s primernim znanjem in izkušnjami primanjkuje, če pa le ti so, so za podjetja zelo dragi, saj pričakujejo za svoje znanje primerno nagrado in ugodnosti. Druga možnost, ki ostane podjetjem, kot je RRC d.d., pa je zaposliti ljudi brez izkušenj s primernimi sposobnostmi in jih vključiti v izobraževanje na področju, kjer bodo delovali. Seveda tudi ta druga možnost za podjetja ni zastonj, saj je treba delavca poslati na draga izobraževanja, poleg tega pa si mora z delom, ki ga ponavadi ne moremo tržiti in katerega delo predstavlja v začetku le strošek, še pridobiti ustrezna znanja in izkušnje. Kje je meja? Ali se da oceniti znanje, ki ga nek kandidat ima in koliko bi podjetje stalo, da vzgoji delavca s podobno ravnjo znanja.

Izbira delavcev je zelo na široko raziskovano področje. Izbiro najboljše različice ali *variante* opravimo na osnovi večparametrskega odločanja, kjer različice opišemo z naborom vrednosti sodila ali *parametrov*, ki imajo vsak svoje zaloge vrednosti. Nato določimo še funkcijo koristnosti (slika 4.1), s katero vsako inačico ocenimo [24].




**Slika 4.1: Funkcija koristnosti [24]**

V preteklosti je bilo razvitih že kar nekaj odločitvenih modelov za izbiro delavcev. Razvit je udi večparametrski odločitveni model v DEX-u in prirejen celo za dlančnik PALM [2]. Večino teh odločitvenih modelov pa ne vključuje sodila, ki bi kandidata sodili po stroških, ki

bodo v povezavi s kandidatom nastali ob zaposlitvi in v nekem časovnem obdobju po zaposlitvi in so povezani z izobraževanjem ter pridobivanjem delovnih izkušenj kandidata.

### 4.1.2 Problematika zaposlovanja v RRC

Zanima me, ali v podjetju obstajajo računovodske informacije, iz katerih se da oblikovati informacijo za oceno posameznega delavca, na podlagi katere bo sprejeli odločitev. Za te informacije je zadolženo poslovodno računovodstvo, ki je usmerjeno k oblikovanju računovodskih in drugih informacij, ki jih uporabljajo pri poslovnih odločitvah [19]. Rad bi prikazal, da morajo v veliki meri na izbiro delavca vplivati tudi podatki o tem, koliko nas bo delavec stal ob zaposlitvi in v času delovanja, prav tako pa je pomembna tudi informacija o tem, koliko lahko pridobimo z delom tega istega delavca.

Osnovna ideja pri mojem razmišljanju se mi je utrnila, ko sem spremljal politiko zaposlovanja novih ljudi v podjetju RRC. Pridobivanje novih delavcev v podjetju RRC spremljam že od julija 2001 pa do danes. V zadnjih dve letih sem kot vodja enote in projekta pri izbiri tudi sodeloval. Novi delavci naj bi v podjetje prinesli svežino s poznavanjem novih orodij in postopkov izgradnje informacijskih sistemov. V grobem lahko rečem, da smo zaposlili dva tipa novih ljudi:

- Delavca s konkretnimi izkušnjami na področju za katerega se zaposluje (točno določena znanja iz poznavanja konkretnih tehnologij razvoja IS).
- Delavca, ki ni imel izkušenj z razvijem IS, torej ni bil predhodno zaposlen, prav tako pa kakšnih konkretnih znanja s področja razvoja IS ni pridobil tekom šolanja.

Prva razmišljanja in ugotovitve sem v marcu 2003 opisal v seminarski nalogi pri predmetu Poslovne funkcije v okviru magistrskega študija. Takrat je bilo stanje takšno, da smo v podjetju zaposlili pet novih delavcev, nihče pa ni imel niti dneva delovnih izkušenj (niso bili predhodno zaposleni). Prav tako niso imeli nobenih potrdil (certifikatov) o posebnih znanjih na področjih, za katere smo jih zaposlovali. Seveda so se različno vklopili v delovanje podjetja, nekateri hitreje, nekateri počasneje. Po kasnejših pogovorih z nekaterimi odgovornimi sem prišel do ugotovitve, da so takrat imeli na izbiro tudi delavce z izkušnjami in posebnimi znanji, vendar pa se za njih niso odločili zaradi previsokih želja glede plače in nagrad.

Kasneje se je to spremenilo, saj smo v obdobju do junija 2006 zaposlili še 7 delavcev. Med njimi so bili tudi takšni, ki so imeli konkretna znanja (potrdila in izkušnje) na področjih, za katere smo jih zaposlovali.

Naslednja stvar, ki sem jo ugotovil, je bila ta, da so zaposlenci, s povečanjem svojih znanj in izkušenj ter povečanjem pomembnosti, za svoje delo zahtevali vedno višje plače.

Na prvi pogled se mi je zdelo, da je podjetje s takšnim početjem le izgublja in bi bilo smotrneje zaposliti enega ali dva delavca z izkušnjami ter se sprijazniti z večjimi stroški v začetku, s tem pa bi podjetje prej doseglo boljši izkoristek delovne sile, saj bi se lahko učili od izkušenih delavcev in bi prej ter lažje prišli do zadovoljivega znanja in izkušenj. Zato sem svoja predvidevanja poskušal utemeljiti tudi na ta način, da sem razvil odločitveni model.

Ob teh podatkih in ugotovitvah sem si postavil vprašanje, kako bi lahko določenega delavca ocenil glede na razliko v stroških, ki jih je podjetje imelo z njim, in učinkom, ki ga podjetje lahko dobi od njega. Problem mora dobiti še časovno razsežnost, ki jo je lažje upoštevati kot eno od osnov za ocenjevanje.

### 4.1.3 Postavitev dejanskega primera okvirov

Zaradi velike širine problema sem si postavil vprašanje in določil robne pogoje (okvire), saj vseh možnih vplivov ni mogoče zajeti v primerjanje. Dobro poznavanje razpisanega prostega delovnega mesta je osnovni pogoj za odločitev, kakšen ekspertni ali informacijski sistem bo odločal pri izbiri novih delavcev. Zato je za posamezno delovno mesto[2] potrebno dobro poznavanje [12]:

- zahtev sistemizacije delovnega mesta,
- analize dela,
- analize delavca,
- vedenjske sistemizacije.

Sistemizacija delovnega mesta pomeni oblikovanje nalog, ki jih je treba opravljati na posameznem delovnem mestu, proučitev okoliščin, ki bodo delo spremljale, ter določitev časa opravljanja posamezne naloge [12].

V letu 2002 je bila v podjetju sprejeta strategija tehnološkega razvoja podjetja RRC. Eden od sklepov odločitvene skupine je bil tudi, da v podjetju uvedemo novo tehnologijo, ki bi jo radi osvojili v petih letih. Del sklepa je bil tudi, da zaposlimo osebo, ki bo orala ledino in sčasoma k svoji skupini pritegnila še druge delavce. V obdobju najmanj 5 let hočemo novo področje obvladovati do te mere, da bo podjetje sposobno izdelati in dobaviti sistem v novem orodju, za kar bi po sistemizaciji delovnih mest podjetja RRC zadoščala razporeditev na delovno mesto 4.b - Analitik. V mojem primeru se bom omejil na tri različice začetka in sicer:

1. **Različica 4.b**: V začetku zaposliti ustreznega delavca za delovno mesto 4.b Analitik (po aktu o sistemizaciji delovnih mest podjetja RRC d.d.), in mu čim prej dodeliti naloge za osnovanje skupine. Oseba mora imeti izkušnje in reference točno na področju, za katerega hočemo uvajati novo tehnologijo. Tako bomo lahko njegovo znanje tržili takoj po zaposlitvi. Za takšnega delavca pričakujemo, da ga bomo izobraževali z manj sredstvi.
2. **Različica 6.c**: Zaposlimo delavca, ki ustreza delovnemu mestu 6.c (po aktu o sistemizaciji delovnih mest podjetja RRC d.d.). Oseba ima sicer zahtevana znanja, nima pa referenčnih znanj točno na želenem področju. Z ustreznim izobraževanjem in pridobljenimi dodatnimi izkušnjami pa bo takšen delavec v določenem obdobju uspel sestaviti skupino za obvladovanje nove tehnologije.
3. **Različica 6.a:** Tretja različica je zaposliti delavca na delovno mesto 6.a (po aktu o sistemizaciji delovnih mest podjetja RRC d.d.) takoj po končanem diplomskem

[2] Po Mihelčiču [10] je delovno mesto opredeljeno kot najmanjša organizacijska enota v združbi, na katerih zaposlenci opravljajo opravke in s tem opravljajo delovne naloge. Delovno mesto, opredeljeno v listinah RRC d.d, bi bilo zato pravilneje poimenovati delovno področje, ki je opredeljeno z opravki, ki jih je zaposleni sposoben izvajati in za katerega izvajanje se je dogovoril ob nastopu zaposlitve [11].

študiju ustrezne smeri. Takšnega delavca bo potrebno pošiljati na izobraževanja, poleg tega pa bo njegov mrtvi tek v času, ko ga še ne moremo tržiti večji.

| | Delovno mesto | Pogoji | | Značilna dela in naloge |
|---|---|---|---|---|
| | | Izkušnje | Odlike | |
| 4b | Analitik – snovalec II - specialist | 5 let | - sistemska analiza<br>- razvojna orodja in razvojna metodologija<br>- programiranje<br>- tuj jezik (angleščina)<br>- samostojnost in inventivnost<br>- poznavanje IT in spleta ali *interneta*<br>- andragoške sposobnosti<br>- gradnja povezanih ali *integriranih* informacijskih sistemov<br>- poznavanje projektnega vodenja | - analiza, modeliranje in zasnova sistemov<br>- gradnja (programiranje) sistemov<br>- vodenje projektov, tudi *internetnih*<br>- odgovornost za upoštevanje standardnih postopkov<br>- predlogi inovacij pri izdelavi izdelkov<br>- svetovanje in šolanje uporabnikov<br>- uporabniška dokumentacija |
| 6c | Programer II - specialist | 3 leta | - poznavanje razvojnih orodij<br>- programiranje, tudi za *internet*<br>dobro poznavanje *interneta*<br>- tuj jezik (angleščina)<br>- samostojnost in inventivnost<br>- poznavanje preizkusnih metod<br>- razvojna metodologija | - načrtovanje in programiranje uporabniških rešitev ali *aplikacij*<br>- izdelava razvojne in uporabniške dokumentacije<br>- po potrebi izobraževanje uporabnikov<br>načrtovanje in programiranje uporabniških rešitev ali *aplikacij* za delo na internetu<br>- preizkušanje |
| 6a | Programer II | Po pripravništvu | - poznavanje razvojnih orodij<br>- programiranje<br>- poznavanje *interneta*<br>- tuj jezik (angleščina)<br>- delo v projektni skupini | načrtovanje in programiranje uporabniških rešitev ali *aplikacij*<br>- izdelava razvojne in uporabniške dokumentacije<br>- po potrebi izobraževanje uporabnikov<br>načrtovanje in programiranje uporabniških rešitev ali *aplikacij* za delo na *internetu*<br>- preizkušanje |

**Preglednica 4.1: Pogoji in naloge delavcev glede na delovno mesto (vir: Akt o sistemizaciji delovnih mest podjetja RRC d.d. )**

Za opredeljene tri različice sem potegnil vzporednico z opredelitvijo delovnih mest po Aktu o sistemizaciji delovnih mest podjetja RRC d.d in sicer: je prva različica povezana z delovnim mestom Analitik – snovalec II (specialist), druga z delovnim mestom Programer II – specialist, in tretja z delovnim mestom Programer II. Pogoje ter značilna dela in naloge posameznega delovnega mesta predstavlja preglednica 4.1.

Primer je treba postaviti še v časovno obdobje, saj le tako dobimo smiselno primerjavo. Glede na izkušnje, potrebne za napredovanje po delovnih mestih, sem ocenjevano obdobje postavil na 8 let, saj je osnovna predpostavka (želja podjetja), da v petih letih ustvarimo skupino. Tri leta pa sem dodal še zaradi tega, da je v primerjavo dodano še obdobje delovanja skupine. V obdobju osmih let je potrebno postavil tudi kontrolne točke, ki se pokrivajo z leti za napredovanje po razrednih lestvicah vsakega delavca. Kontrolne točke so postavljene na 2., 3. in 5. let.

## 4.2 Predstavitev sodil

### 4.2.1 Uvod

Pri ocenjevanju primernosti delavca lahko na končno oceno vplivajo različna sodila. Sodila izbora so praviloma: vrsta izobrazbe, raven strokovne usposobljenosti ter delovne izkušnje, torej možno delovno področje – pripravljenost za izvajanje različnih opravkov, osebnostne lastnosti in različne veščine oziroma določena znanja [10]. Pri postavitvi modela sem za izračun ocene različice upošteval tista sodila, ki so povezana s stroški oziroma prihodki in se jih da izraziti v denarni enoti. V primerjavah so vsa števila v slovenskih tolarjih (v nadaljevanju SIT). V naslednjem poglavju so opredeljena vsa sodila in obrazci za izračun končne ocene različice.

### 4.2.2 Sodila, ki so upoštevana pri izračunu končne ocene delavca

Prva skupina sodil predstavlja za podjetje stroške, ki jih imamo z delavcem. V oceni niso upoštevani stroški, ki prav tako nastanejo v povezavi z določenim delavcem, vendar pa so za vsakega delavca enaki. To so stroški, povezani z delovnimi sredstvi (računalnik, miza, stol, prostor,…)

**ZAČETNI STROŠKI**

Stroške sem v prvem delu razdelil na tiste, ki nastanejo ob zaposlitvi (začetni stroški). To so stroški, ki jih lahko ima podjetje z delavcem pred in ob zaposlitvi. Stroški, ki nastanejo pred ali ob zaposlitvi, nastanejo zaradi:

1) štipendiranja v letih izobraževanja,
2) poravnave obveznosti delavca do podjetja, v katerem je bil sedaj zaposlen in pridejo iz naslova štipendij ali plačila izobraževanj,
3) plačila obveznosti delavca do podjetja, v katerem je bil zaposlen in je z njim sklenil določene dogovore (pogodba o sofinanciranju študija, ...).

Tako lahko opredelimo štiri sodila in sicer:
$STR_{štip}$ – stroški pod točko 1, $STR_{obvizob}$ – stroški pod točko 2 in , $STR_{obvdog}$ – stroški pod točko 3 in $STR_{zac}$ – celotni začetni stroški, ki jih izračunamo:

$$STR_{zac} = STR_{štip} + STR_{obvizob} + STR_{obvdog}$$

Sodila, ki predstavljajo začetne stroške, so neodvisna od obdobja ocenjevanja.

**Stroški, ki nastanejo v primerjalnem obdobju**

To so stroški, ki nastajajo v času delovanja delavca in so vezani na obdobje, katerega opazujemo. Te stroške delimo na dva dela:

a) Stroški z naslova nagrajevanja delavca (plača).

Stroške nagrajevanja sem za primerjanje opredelil glede na prakso in Pravilnik o osnovah in merilih za delitev sredstev za plače RRC. Vsako delovno mesto je ocenjeno z ustreznim številom točk. Mesečna kosmata plača delavca pa predstavlja zmnožek točk in vrednosti točke, ki jo določi ravnateljstvo podjetja. Dobimo sledeča nova sodila: $PLAČA_{tocke}$, ki predstavlja število točk delavca in $VREDNOST_{točke}$, ki predstavlja vrednost točke. Skupaj dobimo $STR_{plača}$, ki se izračuna na takole:

$$STR_{plača} = PLAČA_{tocke} * VREDNOST_{točke} * 12*OBD_{let}$$

$OBD_{let}$ predstavlja število let v primerjalnem obdobju in se bo pojavilo tudi še pri izračunih vrednosti ostalih sodil.

b) Stroški, ki nastanejo zaradi izobraževanja delavca (plačila seminarjev).

Izračun stroškov izobraževanj $STR_{izob}$ sem razdelil na dve sodili in sicer $Št_{izob}$, ki predstavlja število dni izobraževanj v enem letu in $CENA_{izobdan}$, ki predstavlja ceno izobraževalnega dne. Celotne stroške, ki nastanejo zaradi izobraževanj, izračunamo tako:

$$STR_{izob} = Št_{izob} * CENA_{izobdan} * OBD_{let}$$

Tako lahko opredelimo $STR_{obd}$, ki predstavljajo vse stroške v primerjanem obdobju kot vsoto $STR_{plača}$ in $STR_{izob}$

$$STR_{obd} = STR_{plača} + STR_{izob}$$

**VREDNOST DELAVCA**

Druga skupina sodil predstavlja v denarni enoti izraženo oceno, koliko je določen delavec vreden za podjetje. V storitveni dejavnosti velja, da namesto poslovnega učinka prodajamo delavca, ki opravi storitev. V večini primerov delavce tržimo na delovno uro, katere višina je odvisna od znanj delavca in vrste del, ki jih opravlja.
Vrednost delavca lahko delimo na:

- Neposredno vrednost (cena, ki jo lahko za delo delavca iztržimo v nekem časovnem obdobju in je merljiva).
- Posredno vrednost (posredno merljive vrednote, s katero podjetje ustvarja dodano vrednost: ugled, možnost prijave na razpise, reference,…).

Zaradi težko izračunljive posredne vrednosti v izračunu upoštevam le neposredno vrednost.

Neposredno vrednost ($VRE_{nepos}$) izračunamo kot zmnožek $ŠTUR_{mesec}$, ki pomeni število ur, ki jih opravi delavec na mesec in jih uspemo zaračunati ter $CENA_{ure}$, ki pomeni ceno ure, ki jo uspemo zaračunati za opravljeno delo delavca:

$$VRE_{nepos} = ŠTUR_{mesec} * CENA_{ure} * 12*OBD_{let}$$

Ker sodilo $ŠTUR_{mesec}$ pomeni koliko opravljenih ur smo uspeli prodati, lahko s spreminjanjem vrednosti sodila primerjamo dva delavca z različno učinkovitostjo.

### 4.2.3 Celotna ocena delavca

Celotno oceno delavca dobimo kot seštevek začetnih stroškov in stroškov v primerjalnem obdobju, ki jih odštejemo od neposredne vrednosti delavca;

$$\mathbf{OCENA = VRE_{nepos} - STR_{zač} - STR_{obd}}$$

Celotna ocena predstavlja cenovno izraženo razliko med nekaterimi stroški in nekaterimi prihodki, ki jih podjetje ima/pridobi z določenim delavcem. Ocena ni primerljiva z dodano vrednostjo, ki jo podjetje ustvari z delavcem, saj v stroških niso zajeti stroški, ki so povezani z osnovnimi sredstvi in delovnimi predmeti. Ocena lahko služi le za primerjavo med dvema ali več primeri.

## 4.3 Pridobivanje informacij, vrednotenje sodil in izračun ocene kandidatov

### 4.3.1 Pridobivanje informacij in vrednotenje sodil

Za čim pravilnejšo oceno delavca je potrebno čim bolj stvarno oceniti sodila, ki so našteta v prejšnjem poglavju. Spodaj navajam, kako sem dobil vrednost določenega sodila ali pa pridobil informacije za oceno te vrednosti. Pri vsakem sodilu je navedena tudi vrednost za posamezno vrsto delavca. Pri pridobivanju informacij sem naletel na dva problema:

- Trije primeri delavcev, ki sem jih postavil kot omejitev, v začetku niso stvarni in so v tem trenutku tudi kandidati za zaposlitev v podjetju, zato sem nekaj ocen le predpostavil.
- Pridobivanje informacij o nagrajevanju delavcev in o višini izobraževanj za delavce so tajni in niso dostopni. V računovodstvu sem dobil le splošne informacije o nagrajevanju in podatke, ki veljajo zame. Zato sem v takih primerih poskušal potegniti vzporednico med mojimi osebnimi podatki in ocenjevanimi vrstami delavcev.

- $STR_{štip\ in}\ STR_{obvizob}$

Podatek o stroških štipendiranja sem pridobil v računovodstvu podjetja RRC, kjer so mi izračunali, koliko bi bil osebno dolžan podjetju RRC, če bi prekinil delovno razmerje s

podjetjem, ker so me štipendirali v času mojega rednega študija na fakulteti. Moja obveznost do podjetja je, da moram v podjetju službovati toliko časa, kot sem dobival štipendijo, v nasprotnem primeru pa moram vrniti štipendijo s pripadajočimi obrestmi.
Z vrednostjo, ki sem jo dobil, sem ocenil sodilo $STR_{obvizob}$ pri delavcu 6c, ki bi ga RRC lahko zaposlil po drugi različici. Vrednost sodila prikazuje preglednica 4.2.

- $STR_{obvdog}$

Podatek o stroških obveznosti do delodajalca sem dobil na podlagi podatkov, posredovanih iz računovodstva o mojem dolgu podjetju, če bi sam prekinil delovno razmerje, ki nastane na podlagi pogodbe o financiranju podiplomskega študija na Fakulteti za računalništvo in informatiko.

Z vrednostjo, ki sem jo dobil, sem ocenil sodilo $STR_{obvdog}$ pri delavcu 4b, ki bi ga RRC lahko zaposlil po prvi različici. Vrednost sodila prikazuje preglednica 4.2.

- $PLAČA_{točke}$

Podatki o plačah delavcev so tajni, zato nisem mogel dobiti natančnih podatkov o višini točk posameznega delavca. Za osnovo sem vzel Pravilnik o osnovah in merilih za delitev sredstev za plače RRC [23], kjer je opredeljena višina točk za posamezno delovno mesto. Točkam po pravilniku sem poskusil dodati še povprečno vrednost dodatkov na delovno uspešnost za posamezno delovno mesto, vendar teh podatkov nisem mogel dobiti. Ta dodatek sem upošteval pri vrednosti točke, vendar so tako vsa delovna mesta enako pridobila. Vrednost sodila prikazuje preglednica 4.2.

- $VREDNOST_{točke}$

Vrednost točke sem izračunal s svoje plačilne liste za januar 2006, katero sem pomnožil s faktorjem 1,15, saj je povprečen dodatek (po podatkih kadrovske službe) na delovno uspešnost v podjetju 0,15. Vrednost sodila prikazuje preglednica 4.2.

- $Št_{izob}$ in $CENA_{izobdan}$

Kadrovska služba v podjetju RRC vodi natančno evidenco načrtovanih in izvedenih izobraževanj za vsakega delavca. Tako sem za lahko hitro pridobil informacije o številu izobraževanj in ceni izobraževanj, razdeljenih tudi po delovnih mestih.

V vrednotenju sodila $Št_{izob}$ je upoštevano tudi dejstvo, da je treba delavca na delovnih mestih 6a in 6c intenzivneje izobraževati, saj smo predpostavili, da bosta oba vodila skupino za razvoj v novem orodju. Cena izobraževalnega dne po podatkih RRC raste z zahtevnostjo delovnega mesta, saj so osnovnejša izobraževanja cenejša. Vrednost sodil $Št_{izob}$ in $CENA_{izobdan}$ prikazuje preglednica 4.2.

- $ŠTUR_{mesec}$

Podatke, na podlagi katerih sem ocenil število zaračunanih ur, sem pridobil iz uporabniške rešitve ali *aplikacije* Mesečna poročila podjetja RRC, kjer so natančno zapisane vse opravljene in zaračunane ure za vsakega delavca. Izračunal sem povprečno vrednost zaračunanih ur za posamezno delovno mesto. Pri delovnih mestih 6a in 6c sem odvzel nekaj ur, ker se za njiju pričakuje intenzivnejše izobraževanje in uvajanje, v tem času pa ne moreta opravljati nalog, ki jih zaračunamo. Sodilo lahko glede na pričakovano učinkovitost delavca ustrezno popravimo, saj se učinkovitost programerjev (oziroma vseh delavcev na razvoju IS )

lahko razlikuje med seboj celo v razmerju 1 : 10 [16]. Vrednost sodila predstavlja preglednica 4.2.

- $CENA_{ure}$

Podatek o ceni ur je povzet po uradnem ceniku podjetja RRC s tem, da sem potegnil vzporednice vrste dela z delovnimi mesti. Ocena je bila opravljena na podlagi pogovora z vodjem prodajnega oddelka, za natančnejše podatke o povprečni zaračunani uri glede na delovno mesto, pa bi bilo potrebno vložiti več dela v računovodstvu, na kar pa ravnatelj ni pristal. Informacijo tem je vsekakor enkratna informacija, za katero bi v računovodstvu potrebovali kar nekaj vloženega dela in časa. Po Turku [19] pri poslovodnem računovodstvu od računovodje ne pričakujemo standardnih informacij, ampak enkratne informacije, prilagojene posamezni odločitvi. Po tej opredelitvi lahko informacijo o ceni ure štejemo v informacijo poslovodnega računovodstva. Vrednost sodila predstavlja preglednica 4.2.

### 4.3.2 Izračun ocene vseh treh različic

Že v poglavju 4.2.2 sem postavil omejitev glede obdobja primerjanja posameznih delovnih mest. Ker pa me ne zanima le končna vrednost ocene (glej opredelitev ocene v poglavju 4.2.3), sem ocene primerjal tudi ob zaposlitvi, po dveh, treh in petih letih.

V izračunu je upoštevano tudi to, da se po doseganju sodil za napredovanje na višje delovno mesto spremenijo vrednosti sodil. Tako se spremenijo sodila po dveh letih, ko delavec delovnega mesta 6c napreduje na delovno mesto 4b, po treh letih delavec delovnega mesta 6a napreduje na delovno mesto 6c in po petih letih, ko delavec delovnega mesta 6a napreduje na delovno mesto 4b. Pri upoštevanju sprememb sodil sem naredil dve različici. Pri prvi nisem upošteval povečanje plače ob napredovanju na višje delovno mesto, medtem ko sem pri drugi različici upošteval, da se z večanjem znanja, odgovornosti in uspešnosti povečajo tudi želje in zahteve po nagradi delavca.

Izračun ocene (glej 4.2.3) delavcev brez upoštevanja povečanja plače je prikazan v prilogi 1. Izračun ocene (glej 4.2.3) delavcev, brez upoštevanja povečanja plače, je prikazan v prilogi 2. Pri izračunu so bile upoštevane vrednosti sodil prikazane v preglednici 4.2. V prilogi 1 in 2 so predstavljene vrednosti ocene po posameznih obdobjih. Skupno pa so izračunane vrednosti predstavljene v preglednicah 4.3 in 4.4.

| **Delovno mesto** | 4b - Analitik | 6c – programer II specialist | 6a - programer II |
|---|---|---|---|
| **$STR_{obvizob}$** | 0 | 3.850.500 | 0 |
| **$STR_{obvdog}$** | 5.750.000 | 0 | 0 |
| **$PLAČA_{točke}$** | 3.078 | 2.113 | 1.256 |
| **$VREDNOST_{točke}$** | 184 | 184 | 184 |
| **$Št_{izob}$** | 7 | 11 | 14 |
| **$CENA_{izobdan}$** | 51.200 | 46.100 | 34.720 |
| **$ŠTUR_{mesec}$** | 118 | 98 | 77 |
| **$CENA_{ure}$** | 13.120 | 10.210 | 8.960 |

**Preglednica 4.2: Vrednost sodil za izračun ocene za posamezno delovno mesto**

| delovno mesto\različica | 4b | 6c | 6a |
|---|---|---|---|
| ob zaposlitvi | -5.750.000 | -3.850.500 | 0 |
| po dveh letih | 17.096.592 | 9.818.212 | 10.039.424 |
| po treh letih | 28.519.888 | 23.372.228 | 15.059.136 |
| po petih letih | 51.366.480 | 50.480.260 | 32.512.360 |
| **po osmih letih** | **85.636.368** | **92.739.736** | **79.800.712** |

**Preglednica 4.3: Ocene (glej 4.2.3) delavcev brez upoštevanja povečanja plače po obdobjih**

| delovno mesto\različica | 4b | 6c | 6a |
|---|---|---|---|
| ob zaposlitvi | -5.750.000 | -3.850.500 | 0 |
| po dveh letih | 17.096.592 | 9.818.212 | 10.039.424 |
| po treh letih | 28.519.888 | 21.241.508 | 14.869.248 |
| po petih letih | 51.366.480 | 44.088.100 | 29.800.936 |
| **po osmih letih** | **85.636.368** | **80.684.956** | **66.397.792** |

**Preglednica 4.4: Ocene (glej 4.2.3) delavcev z upoštevanjem povečanja plače po obdobjih**

## 4.4 Sprejem odločitve na podlagi dobljenih ocen

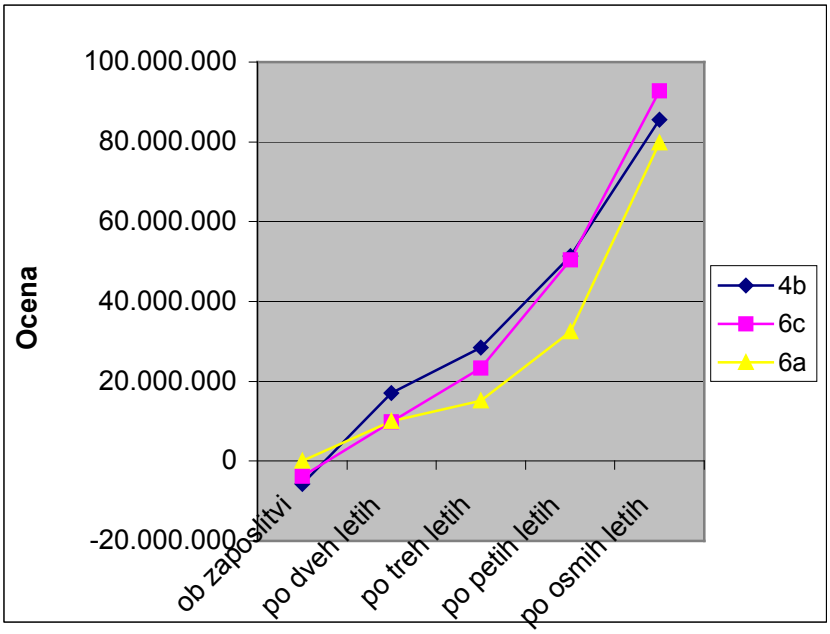
### 4.4.1 Analiza dobljenih izidov

Model za izračun ocene delavca je razvit v orodju MS Excel. Program za delo s preglednicami omogoča primerjavo vrednostno izraženih sodil in omogoča tudi slikovno ali grafično predstavitev izidov. Možna je tudi Kaj–če analiza (*If - then*), pri kateri lahko opazujemo končne ocene, če se spremeni vrednost enega ali pa več sodil, možno pa je tudi raziskati vpliv sprememb na izid [24]. Pomanjkljivost MS Excel-a je v nezmožnosti obdelave opisnih podatkov. Pri modelih z več kriteriji zagotovo pridemo tudi do mehkih podatkov, to je nepopolnih, netočnih, verjetnostnih, včasih pa tudi manjkajočih podatkov, ki pa jih je možno izraziti le opisno. Kljub tej pomanjkljivosti pa je MS Excel-a dobrodošel pripomoček [12].

V primeru, če ne upoštevamo povečanja plače, potem pridemo do naslednjih ugotovitev (glej vrednosti preglednice 4.3 in sliko 4.2). Kratkoročno je najboljša različica, da zaposlimo delavca na delovno mesto 6a, vendar pa je ta različica slaba, saj je že po obdobju dveh let ocena delavca delovnega mesta 6c enaka, v primeru, da ga zaposlimo na delovno mesto 4b, pa je ocena že boljša kot različica 6a. Če med seboj primerjamo še različici 4b in 6c je po obdobju osmih let boljša različica 6c, saj v zadnjem triletju pri obeh različicah dobivamo iste izide, medtem ko so vložki pri različici 6c manjši (manjša plača).

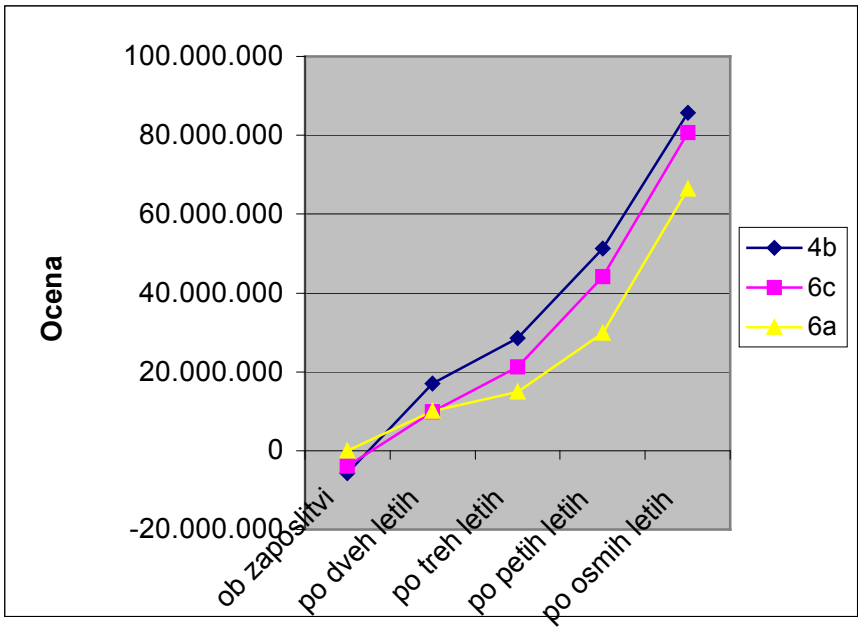
Po mojem mnenju je veliko stvarnejša različica, da bomo v podjetju morali poviševati plačo skladno z napredovanjem delavca. Zato več pozornosti namenjam analizi podatkov preglednice 4.4 in slike 4.3. Tudi v tem primeru se izkaže zaposlitev na delovno mesto 6a le kratkoročno dobra rešitev. Že po koncu prvih dveh let je ocena različice 4b (17.096.592) višja od različice 6a (10.039.424). Vrednost različice 6b (9.818.212) pa je le za malenkost manjša od vrednosti različice 6a. Že po koncu naslednjega obdobja pa se razlike še povečajo. Če natančneje primerjamo še konec obdobja petih let in na koncu osmih let vidimo, da se razlike

med inačicami ne razlikujejo več, saj sta tudi delavca 6c in 6b napredovala do delovnega mesta 4b.




**Slika 4.2: Prikaz gibanja ocene delavca po obdobjih (brez upoštevanja povečanja plače)**

Izidi kažejo, da je vzgoja lastnih delavcev dražja, kot če imamo možnost zaposliti proste strokovnjake. Delavce, ki so še brez izkušenj, je potrebno šolati, poleg tega pa podjetje stane tudi njihovo privajanje na projekt in nova orodja, medtem ko pa lahko že izšolane in izkušene delavce takoj uspešno prodamo in jih vključimo v delo, ki ga lahko naročnikom tudi zaračunamo.




**Slika 4.3: Prikaz gibanja ocene delavca po obdobjih (z upoštevanjem povečanja plače)**

### 4.4.2 Sprejem odločitve

Pri sprejemanju odločitve je potrebno v čim večji meri upoštevati tudi sodila, ki v tej analizi niso bila upoštevana. Predpostavka je bila, da rabimo delavca, ki bi vodil skupino razvijalcev v novih orodjih skladno s sprejeto strategijo tehnološkega razvoja podjetja RRC. Glede na ugotovitve, podane v prejšnji točki, je smotrneje zaposliti delavca na zahtevnejše delovno mesto in se zavedati višjih začetnih stroškov. Takega delavca bomo takoj vključili v vzpostavitev skupine in njegovo znanje takoj izkoristili v čim večji možni meri. Če zaposlimo delavca brez izkušenj in določenih znanj, bomo veliko sredstev porabili za njegovo uveljavitev in uvajanje, po določenem obdobju pa se lahko izkaže tudi to, da nima ustreznih odlik za vodenje skupine, za kar smo ga načrtovali. Ocene kažejo tudi to, da je smotrneje vložiti nekaj več truda, časa in denarja v izbiro že uveljavljenega delavca, je pa res, da so takšni delavci izredno cenjeni in jih na trgu delovne sile primanjkuje.

Pri novi zaposlitvi seveda ostaja veliko tveganje tudi zaradi drugih lastnosti delavca, kot so marljivost, urejenost, iznajdljivost, prilagodljivost... Tako lahko »kratek stik« pri izbiri novega sodelavca v katerikoli od drugih funkcij med ravnatelji druge funkcije in delavci v kadrovski službi privede do zaposlitve neprimernega delavca, ki s svojimi neprimernimi delovnimi navadami, bodisi uničuje vzdušje in učinkovitost v svoji organizacijski enoti bodisi razkraja vrednote združbe v celoti [10].

Večino od teh lastnosti lahko preverimo v preizkusnem obdobju, ki ga sklenemo z delavcem.

Omenil bi še to, da lahko podatke o delavcu (v primeru, da je že bil zaposlen) pridobimo tudi pri prejšnjih delodajalcih in s tem zmanjšamo možnost tveganja.

### 4.4.3 Zaključek

Model za oceno delavca glede na cenovno izražena sodila ob zaposlitvi in v času zaposlitve je prilagojen podjetju RRC. Ob možni členitvi kadrovskih strategij na akumulacijsko, uporabniško ali utilizacijsko ter pospeševalno strategijo, si kaže zastaviti vprašanje o povezavi med razvojno in kadrovsko strategijo [10]. Tudi v primeru podjetja RRC je tako. Postavljena je bila strategija tehnološkega razvoja, temu pa bi morala slediti tudi postavitev kadrovske strategije, pri kateri bi lahko kadrovska služba in ravnateljstvo združbe uporabila model za oceno delavca. Model je primeren predvsem za primerjavo dveh delavcev, ki jih v danem trenutku imamo.

Model nam da še en odgovor, ki bi ga morala kadrovska služba in vodstvo podjetja upoštevati pri politiki zaposlovanja delavcev na področju informatike. Dolgoročno smotrnejše je za podjetje zaposliti izkušene in uveljavljene delavce. Ker je res, da takšnih delavcev na trgu delovne sile ni na pretek ali pa jih sploh ni, bi moralo podjetje v primeru, da ima možnost zaposliti izkušenega delavca, le pristati tudi na morebitne višje pogoje glede nagrajevanja in začetnih stroškov.

Model za oceno delavca je razvit z  nekaterimi omejitvami, saj je za stvarnejše izide potrebno v model vključiti še dodatna sodila. Vsekakor je za mene osebno zanimiva takšna vrsta ocene ob zaposlitvi. Če bi bilo možno razviti model, ki bi nam že ob zaposlitvi dal stvarno oceno, kje in koliko lahko podjetje z zaposlitvijo pridobi, bi ga marsikatero podjetje z veseljem uporabljalo pri zaposlovanju novih delavcev. Izgradnja takšnega modela je dolgoročnejši postopek, saj je potrebno dobljene izide tudi preveriti in nato model v skladu z ugotovitvami popraviti.

# 5 PRIMER OBJEKTNEGA RAZVOJA INFORMACIJSKEGA SISTEMA

## 5.1 KPIS – kadrovsko plačni informacijski sistem

Ministrstvo za šolstvo in šport (v nadaljevanju MŠŠ) je naročnik aplikacije ŠKIS – šolski kadrovski informacijski sistem, ki zajema vnos podatkov o delavcih na osnovnošolskih, srednješolskih in višješolskih zavodih. Iz zajetih podatkov obračunavajo količnike za plače delavcev zaposlenih v teh zavodih.

V letu 2004 je stekel projekt prenove informacijskega sistema s prehodom iz arhitekture odjemalec strežnik (porazdeljeni podatki za vsak zavod in centralna baza (osrednja zbirka) vseh podatkov) na trinivojsko arhitekturo in dostop do podatkov centralne baze preko interneta (spletni brskalnik). Nov program se imenuje KPIS.

Ker so aktivnosti prehoda na trinivojsko arhitekturo sovpadla tudi s prehodi drugih aplikacij na splet in tudi s postavitvijo portala za dostop vsebin in aplikacij MŠŠ, je bil cilj MŠŠ poenoten vstopni center za vse in avtentikacija uporabnika le na enem mestu (*single sign on*). Zaradi varnosti je bila zahteva tudi po preverjanju identitete uporabnika s spletnimi potrdili.

Ker je bilo v to vpletenih več podjetij (izvajalcev), ki za MŠŠ izdelujejo in vzdržujejo aplikacije, je bilo potrebno razviti enoten sistem, pri arhitekturi katerega so sodelovali vsi vpleteni, nato pa je bilo pri vsaki aplikaciji potrebno zagotoviti priklop na sistem portala.

Podjetje RRC d.d. za naročnika vzdržuje program ŠKIS. Sistem je zasnovna kot distribuiran (porazdeljen), kjer ima vsak zavod (šola) lokalno shranjene podatke. Vsi podatki so shranjeni tudi v centralni podatkovni bazi. Lokalni in centralni podatki se osvežujejo na zahtevo.

V osnovi prenova zahteva postavitev nadzornega modula (aplikacijski in baza podatkov), ki bo omogočal hranjene in upravljanje podatkov in pravic o uporabnikih sistema ŠKIS, ki se po prehodu na internet preimenuje v KPIS. Razvoj in vpeljava nadzornega modula je tudi predmet te seminarske naloge.

POJMOVNIK:

MŠŠ – Ministrstvo za šolstvo in šport naročnik sistemov ŠKIS in KPIS
Portal MŠŠ - vstopna točka vseh uporabnikov sistema MŠŠ. Preko portala se bodo uporabniki enolično prepoznali s spletnimi potrdili.
ŠKIS - Šolski kadrovski informacijski sistem: Aplikacija ŠKIS je bila za MŠŠ razvita v letu 1997 za srednješolski sektor (podatki in plače zaposlenih v srednješolskih zavodih).
ŠKIS II - Šolski kadrovski informacijski sistem: V letu 2004 modificiran sistem ŠKIS, ki je bil prilagojen tudi za priključitev osnovnošolski zavodov v sistem.

KPIS - Kadrovsko plačni informacijski sistem, v okviru katerega je izdelan nov program, ki za hranjenje podatkov uporabljal obstoječo centralno bazo in bo razvit v trinivojski arhitekturi.

## 5.2 Opredelitev faz in aktivnosti

Projekt razvoja aplikacije KPIS smo razdelili tri module:
1. Razvoj modula za nadzor uporabnikov
2. Razvoj modula za prikazovanje podatkov o plačah
3. Razvoj modula za vnos podatkov za izračun plače

Vsak modul smo po objektni metodologiji razvoja aplikacij [8] razbili na faze (stopnje) in aktivnosti (dejavnosti),

**Faze in aktivnosti razvoja aplikacije KPIS:**
1. Razvoj modula za nadzor uporabnikov
**1.1. Zajem zahtev**
**1.2. Analiziranje**
**1.3. Načrtovanje**
**1.4. Implementacija**
**1.5. Testiranje**
2. Razvoj aplikacije KPIS (podatki o plačah)
2.1. Zajem zahtev
2.2. Analiziranje
2.3. Načrtovanje
2.4. Implementacija
2.5. Testiranje
3. Razvoj aplikacije KPIS (vnos podatkov za plače)
3.1. Zajem zahtev
3.2. Analiziranje
3.3. Načrtovanje
3.4. Implementacija
3.5. Testiranje

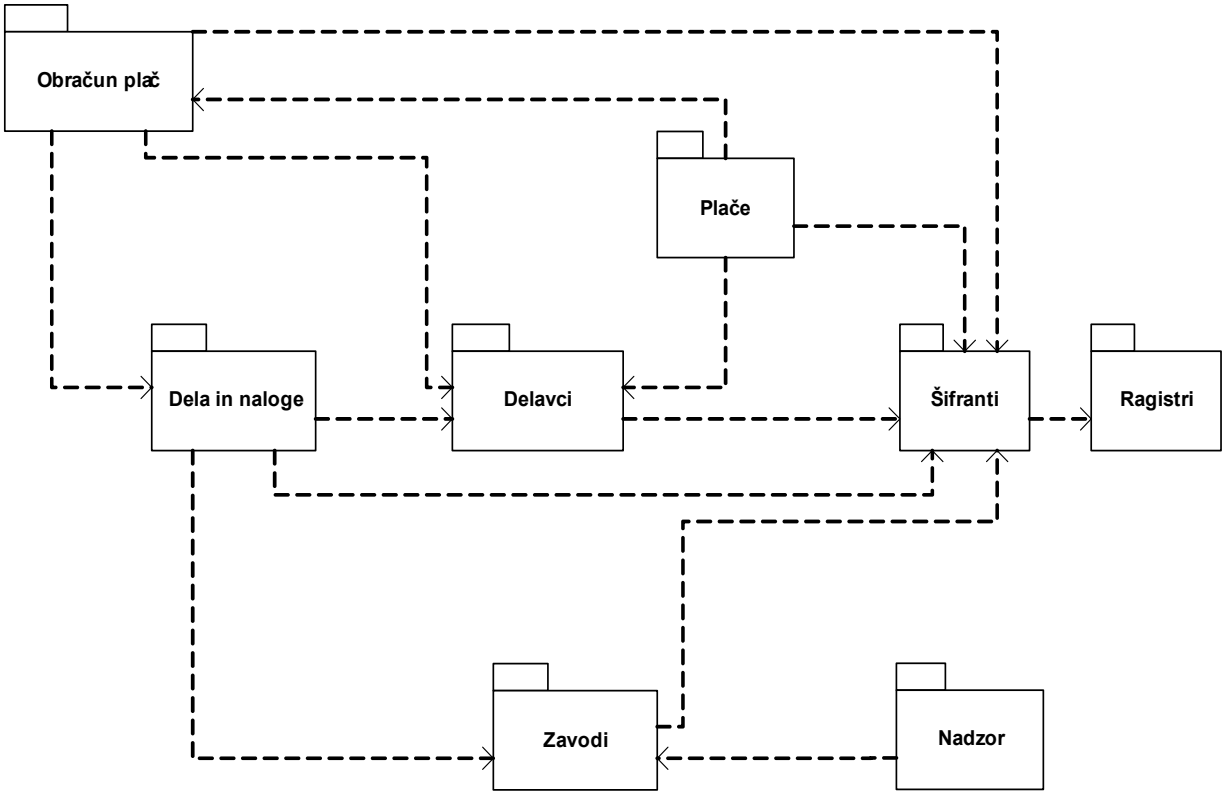
Modul za nadzor uporabnikov predstavlja dostop do spletne aplikacije, kjer je bila osnova tudi postavitev portala kot vstopne točke za vse aplikacije in vpeljavo certifikatov MSS CA za avtentikacijo uporabnikov. V modulu za nadzor je bilo potrebno zagotoviti varni dostop do spletnega dela aplikacije. Tukaj je bilo potrebno tudi razviti del hranjenja podatkov o uporabnikih (bazne tabele).

Drugi modul aplikacije KPIS zavzema prikazovanje podatkov o izračunanih plačah (podatki za izračun se vnašajo še s starim programom). Z modulom za vnos podatkov vnašamo podatke o delavcih (Matični podatki, pogodbe in dela ter naloge).

## 5.3 Opis sistema

Sistem KPIS je namenjen vnosu in obravnavi kadrovskih podatkov zaposlenih v osnovnošolskih in srednješolskih zavodih. Sistem služi za obračun financiranja zavodov s strani MŠŠ. Sistem je tudi kadrovska evidenca za vse zaposlene v zavodih.

Paketni diagram na sliki 5.1 prikazuje dela sistema KPIS. Predstavlja skupine razredov in njihove odvisnosti (povezave).




**Slika 5.1: Paketni diagram sistema KPIS**

**ŠIFRANTI:**
Predstavlja vse podatke, ki jih uporabljajo kot šifrante sistema. Del šifrantov vodimo v sistemu, del pa pridobimo iz registrov MŠŠ, kjer so centralno za vse aplikacije shranjeni nekateri šifranti (zavodi, izobraževalni programi,…)

**ZAVODI:**
Paket zavodov predstavlja podatke zavodov in podatke o oddelkih, ki se izvajajo na zavodih.

**DELAVCI:**
Najobsežnejši paket predstavlja delavce, njihove pogodbe.

**DELA IN NALOGE:**
Paket predstavlja razporeditve na delovne naloge. V grobem se ločijo naloge, na tiste, ki vsebujejo poučevanje (učitelji, laboranti,..) in tiste, ki ne (snažilke, računovodje, hišniki …), tu se paket navezuje na zavode.

**OBRAČUN PLAČ:**
Modul obsega zelo obsežno paketno obdelavo, ki iz vnesenih podatkov izračuna posamezne podatke za plačo delavca. Obdelava se lahko proži po različnih kriterijih (plačan ali testni obračun, poračunavanje, skupine delavcev, cel zavod,…). Podatki se prenesejo v paket Plače.

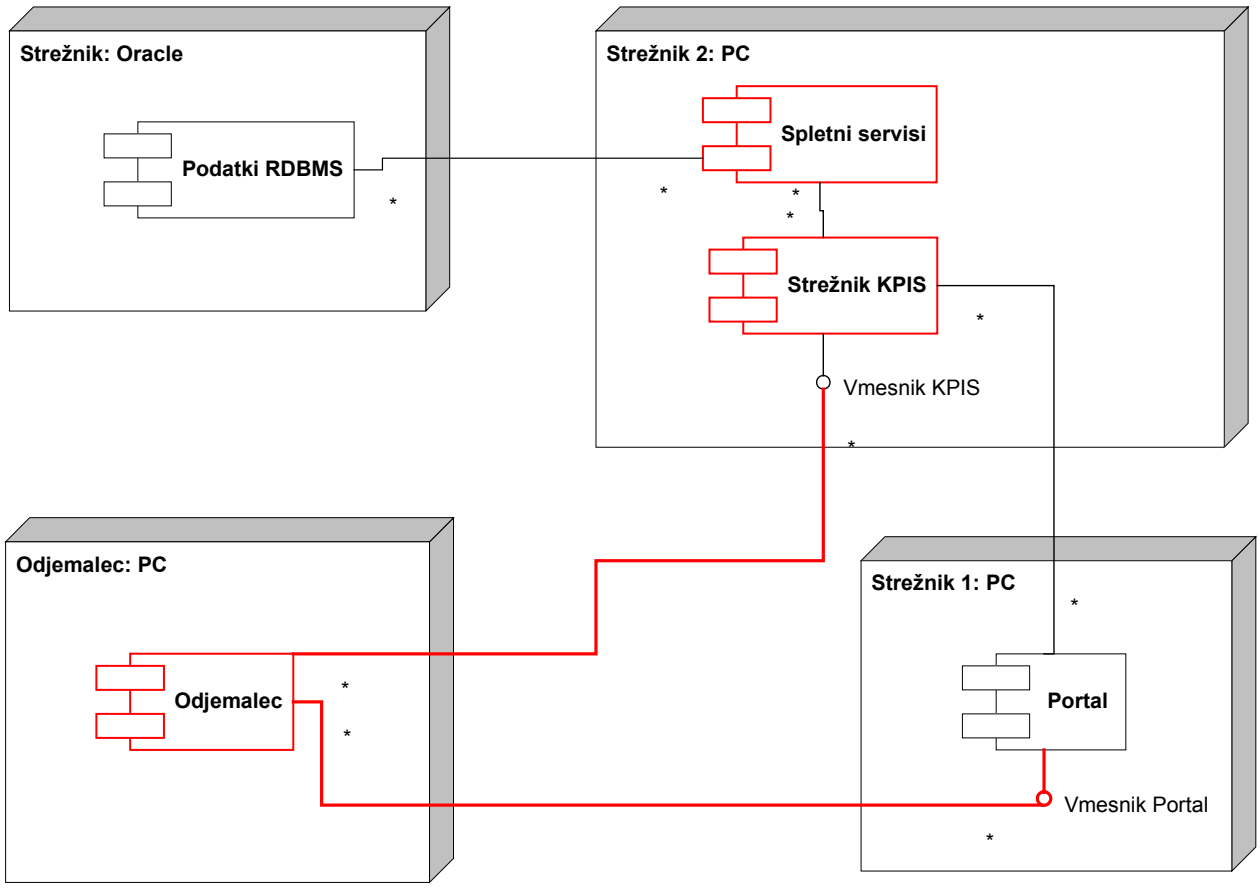
**PLAČE:**
Hranijo se podatki o plačah delavec, ki so razbiti na različne podskupine (količniki). Paket vsebuje tudi razne preglede nad podatki.

**NADZOR:**
Paket predstavlja modul za nadzor uporabnikov sistem. Tu se uporabnikom dodelijo različne pravice in vloge v sistemu. Nadzor vsebuje tudi nastavitve uporabnikov pri uporabi sistema KPIS.

Entitetni diagram v prilogi 3 predstavlja fizični model baze podatkov sistema KPIS – ŠKIS. Posamezne entitete so podrobneje opisane v dokumentu Opis podatkovne baze ŠKIS.

Arhitektura celotne rešitve na sliki 5.2 je štirinivojska, kjer je kot enotna vstopna točka dodan še portal, kjer se uporabnik prijavi s svojim certifikatom, od tam pa se glede na izbiro preusmeri na aplikacijo KPIS (eno od mnogih).

1. **NIVO**
Uporabnik do aplikacije dostopa le s spletnim brskalnikom. Na računalniku mora imeti naložene vse elemente zaščite (javni certifikat MŠŠ in svoj privatni certifikat).
2. **NIVO**
Spletni (*web*) nivo. Strežnik kjer vse zahteve uporabnika (brskalnika) obdela in primerno odgovori. Za primerno obdelane podatke komunicira z aplikacijskim strežnikom (web servisi).
3. **NIVO**
Nivo aplikacijskega strežnika, kjer poteka tudi del poslovne logike (poleg poslovne logike v bazi). Na aplikacijskem strežniku so nameščeni spletni servisi (*web servis*), ki komunicirajo s podatkovnim strežnikom preko shranjenih procedur (*stored procedure*).
4. **NIVO**
Na podatkovnem nivoju je nameščena podatkovna baza tipa Oracle 10g (*second release*). Poleg hranjenja podatkov je na tem nivoju implementirane tudi ogromno poslovne logike.




**Slika 5.2: Arhitekturni diagram sistema KPIS**

.

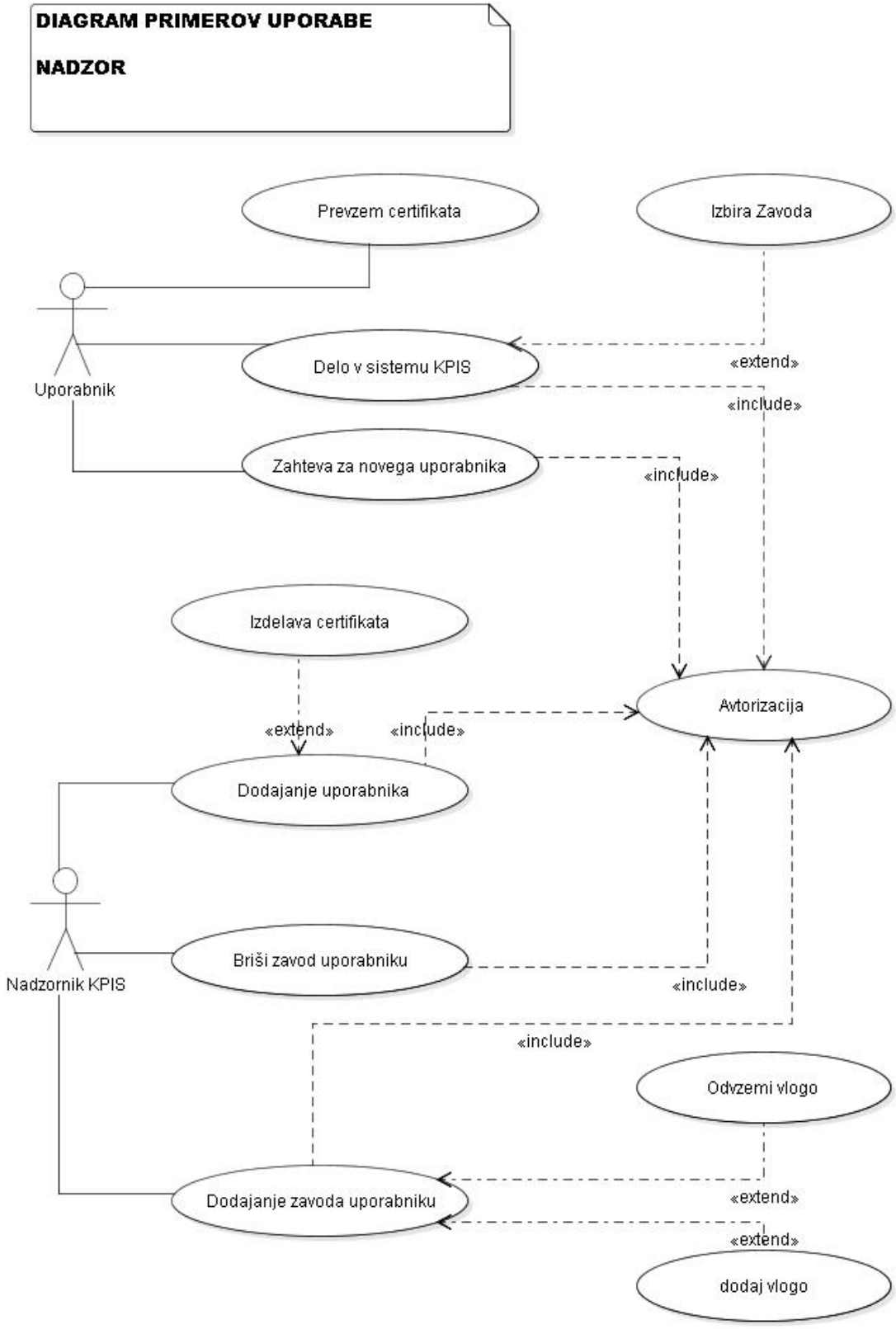
## 5.4 Izdelava modula za nadzor

### 5.4.1 Zajem zahtev

Osnova za uporabniške zahteve je bila vpeljava portala za enotni dostop do vsebin in vseh aplikacij MŠŠ. Vsi uporabniki se bodo z digitalnimi spletnimi potrdili svojo istovetnost na portalu. Portal pa bo glede na pravice uporabnika omogočal dostop do aplikacij; v našem primeru do aplikacije KPIS. To bo tehnično izvedljivo tako, da bo portal tvoril veljavni žeton, s katerim bo uporabnik vstopil v določeno aplikacijo. Uporabniška rešitev bo preverila veljavnost žetona in uporabnika. Nato bo glede na podatke o uporabniku le tega zavrnila, oziroma mu omogočila dostop do aplikacije KPIS. Naročnik je podal tudi zahteve glede uporabnikov in njihovih pravic v aplikaciji KPIS ter zahteve za vnos, pregled in vodenje uporabnikov.

#### 5.4.1.1 Diagram primerov uporabe

Na podlagi sestankov, ki smo jih imeli z uporabniki, smo definirali osnovne zahteve. Osnova zahtevam je bil enovit dostop do vseh aplikacij MŠŠ z enotno identifikacijo.




**Slika 5.3: Diagram primerov uporabe za modul nadzor**

#### 5.4.1.2 Zahteve uporabniškega vmesnika

**Splošne zahteve**

**Seznam**

Seznam prikazuje določene podatke. Izgled strani je v obliki preglednice, kjer so na vrhu napisana imena stolpcev, nato pa so prikazani podatki. Imena stolpcev in formati pa so opisani posebej. Vsak seznam je urejen po prvem stolpcu, uporabnik pa lahko s klikom na ustrezen stolpec urejenost poljubno spreminja (puščica prikazuje ali je urejeno naraščajoče ali padajoče).

Seznam ima na vrhu orodno vrstico, ki na levi vsebuje meni z Operacijami, ki so možne nad podatki, ki jih seznam prikazuje. V orodni vrstici so uporabniku na voljo tudi hitri gumbi za Akcije, ki se pogosto uporabljajo (levo tudi izpisi vsebine seznama). Podatki seznama naj bodo v primeru več kot 50 vrstic porazdeljeni po straneh.

Gumbi - Operacije (osnovne):

| Operacija | **Pogoji vidnosti** | **Preusmeritev** |
|---|---|---|
| Dodaj | Uporabnik ima pravico urejanja podatkov | Odpre vnosni obrazec za dodajanje novega podatka. |
| Izberi | Vedno | Odpre vnosni obrazec z izbranimi podatki za pregled oziroma dopolnjevanje- |
| Briši | Uporabnik ima pravico urejanja podatkov | Zbriše izbrani podatek. |

Na standardni strani s seznamom je možno tudi iskanje oziroma omejevanje seznama. Uporabnik izbere stolpec iz seznama in vpiše iskalni niz ter klikne gumb *Najdi*.

**Kriterij**

Kriterij uporabljamo za omejevanje podatkov seznama. Sestavljen je iz posameznih polj, kamor uporabnik vpiše ali izbere želene vrednosti ter izbiro potrdi s pritiskom na gumb *Prikaži podatke* oziroma *Pripravi izpise*.

**Obrazec Podrobnosti**

Vnosni obrazec omogoča prikazovanje, vnos, brisanje in popravljanje podatkov. Na desni strani so uporabniku na voljo gumbi z operacijami nad prikazanimi podatki.

Gumbi – Operacije :

| Operacija | **Pogoji vidnosti** | **Preusmeritev** |
|---|---|---|
| Zapri | Vedno | Zapre se vnosni obrazec – nazaj na seznam. |
| Shrani | Uporabnik ima pravico urejanja podatkov | Shrani podatke obrazca. |
| Shrani in zapri | Uporabnik ima pravico urejanja podatkov | Shrani podatke obrazca in zapre vnosni obrazec. |

| | | |
|---|---|---|
| Shrani in Nov | Uporabnik ima pravico urejanja podatkov | Shrani podatke obrazca in odpre obrazec. za vnos novih podatkov. |
| Nov | Uporabnik ima pravico urejanja podatkov | Odpre obrazec za vnos novih podatkov. |

Če želi uporabnik zapustiti obrazec in so podatki spremenjeni, a niso shranjeni, se le tega opozori z opozorilom. Želite preklicati spremenjene podatke (Da/Ne). V primeru klika na Da se željna akcija izvede do konca, v nasprotnem primeru pa se mu omogoči shranjevanje (nazaj na obrazec).

## Seznam uporabnikov

TIP: Kriterij, Seznam

Stran se uporablja za vnos, pregled in ažuriranje (posprotenje) podatkov o uporabnikih. Seznam prikazuje uporabnike v sistemu KPIS. Uporabnik ima na voljo tri prednastavljene kriterije (Vsi aktivni uporabniki, Neaktivni uporabniki, uporabniki brez vlog) in pa še omejitev seznama po priimku ali imenu uporabnika.

Podatki seznama:

| Podatek | **Opis** | **Format** |
|---|---|---|
| Priimek | Priimke uporabnika | Samo prikaz |
| Ime | Ime uporabnika | Samo prikaz |
| EMŠO | EMŠO | Samo prikaz |
| E-pošta | E-pošta | Samo prikaz |
| Zavod | Zavod | Samo prikaz |

Gumbi – Operacije:

| Operacija | **Opis** | **Preusmeritev** |
|---|---|---|
| Uredi uporabnika | Odpre se obrazec z pregled oziroma dopolnitev podatkov izbranega uporabnika. | Podrobnosti uporabnika |
| Dodaj uporabnika | Odpre se prazen obrazec za vnos podatkov uporabnika | Podrobnosti uporabnika |
| Dodaj uporabniku šolo | Odpre obrazec za vnos pravic za delo z novo šolo. | Podrobnosti uporabnika |
| Briši uporabniku šolo | Briše uporabniku šolo | Isti seznam osvežen |
| Aktiviraj | Aktivira uporabnika | Isti seznam osvežen |
| Deaktiviraj | Deaktivira uporabnika | Isti seznam osvežen |

## Podrobnosti uporabnika

TIP: Obrazec

Obrazec je namenjen pregledu in urejanju podatkov uporabnika.

Podatki obrazca:

| Podatek | **Opis** | **Format** |
|---|---|---|
| Ime | Ime uporabnika | Vnos |
| Priimek | Priimek uporabnika | Vnos |
| EMŠO | EMŠO | Vnos |
| E-pošta | E-pošta | Vnos |
| Zavod – Naziv | Naziv zavoda | Spustni seznam |
| Zavod - Oznaka | Oznaka zavoda | Spustni seznam |
| Vloge | Vloge | ☑ ☑ Izbira |

Gumbi – Operacije:

| Operacija | **Pogoji vidnosti** | **Preusmeritev** |
|---|---|---|
| Shrani | Vedno | Shrani vnesene podatke. |
| Zapri | Vedno | Zapre obrazec brez shranitve. |
| Shrani in zapri | Vedno | Zapre obrazec s shranitvijo vnesenih podatkov. |

## Dodaj uporabnike KPIS

TIP: Seznam

Določeni uporabniki (Ravnatelji) lahko predlagajo (zahtevajo) svoje delavce za uporabnike programa KPIS. V seznamu se prikazujejo vsi delavci na zavodu. Uporabnik pa lahko določi tiste, ki jim želi dodeliti dostop do aplikacije KPIS. Delavcem, ki že imajo pravice z delom v KPIS, se zahteva onemogoči.

Podatki seznama:

| Podatek | **Opis** | **Format** |
|---|---|---|
| Priimek in Ime | Priimek in Ime | Samo prikaz |
| EMŠO | EMŠO | Samo prikaz |
| Dodaj | Dodaj uporabnika | ☑ ☑ Izbira |

Gumbi – Operacije:

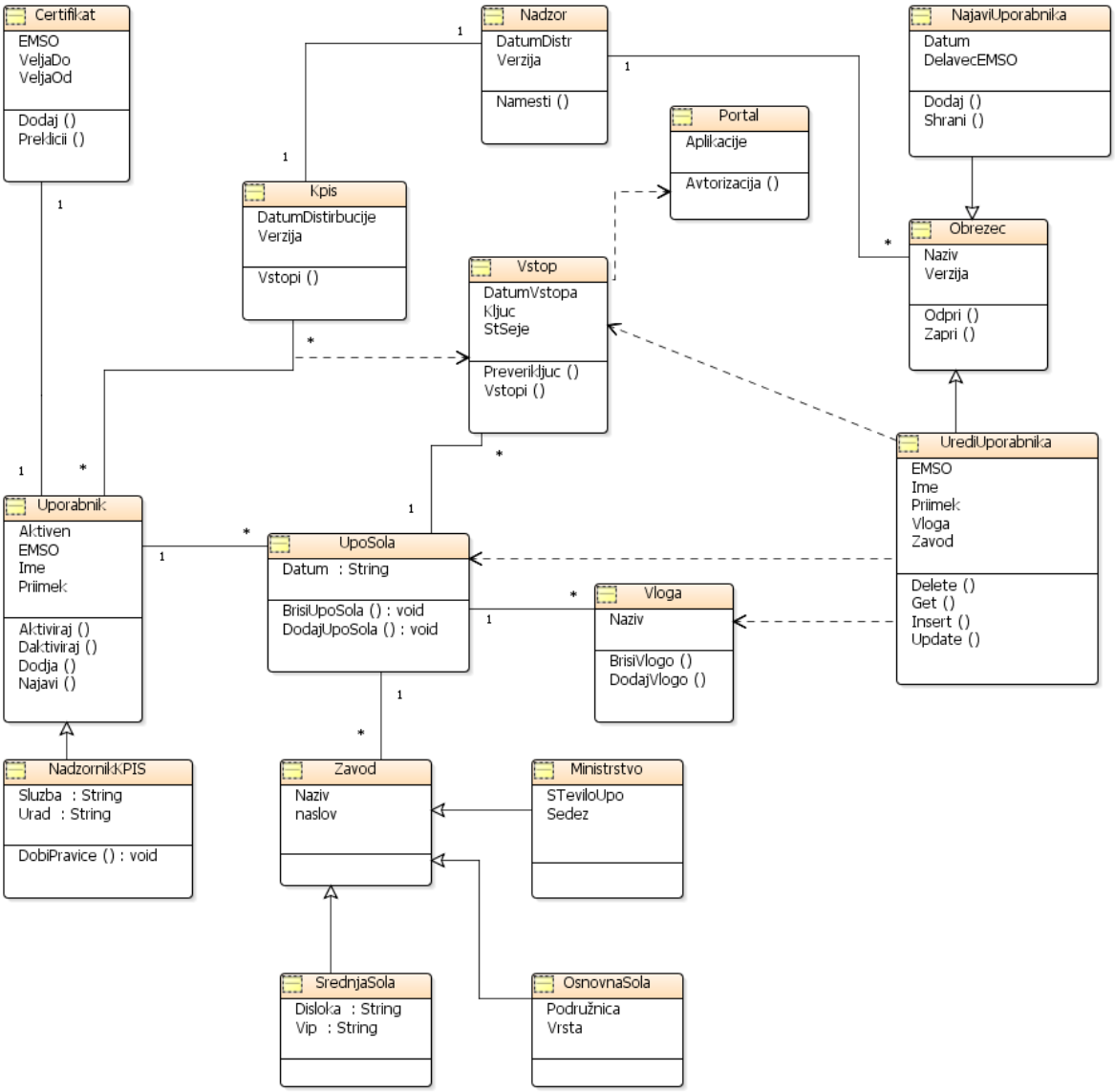
| Operacija | **Pogoji vidnosti** | **Preusmeritev** |
|---|---|---|
| Potrdi | Vedno | Potrdi izbiro in ponovno napolni seznam. |

Omejitve: Možnost samo tistim uporabnikom, ki imajo dodeljeno vlogo KPIS administrator.

### 5.4.2 Analiziranje

Zajemu zahtev je sledila analiza. Lotili so se izdelave konceptualnega razrednega modela in analize primerov uporabe.

#### 5.4.2.1 Izdelava konceptualnega razrednega diagrama




**Slika 5.4: Konceptualni razredni diagram sistema KPIS – modul NADZOR**

Razred **UPORABNIK**:
- Razred UPORABNIK predstavlja akterja sistema KPIS.
- Atributi razreda UPORABNIK opisujejo osnovne lastnosti uporabnika (ime, priimek, EMŠO, E-pošta,..).
- Metode služijo zato, da se uporabnik usposobi za uporabo v sistemu KPIS.

Razred **NADZORNIK KPIS**:
- Razred NADZORNIK KPIS predstavlja podrazred razreda UPORABNIK. Nadzornik je uporabnik, ki pa ima še nekatere dodatne značilnosti.
- Atributi razreda dodatno opisujejo uporabnika.

Razred **CERTIFIKAT**:
- Razred CERTIFIKAT predstavlja digitalno spletno potrdilo, ki ga uporabniki morajo pridobiti za avtentikacijo pri vstopu v sistem KPIS.
- Atributi razreda opisujejo osnovne lastnosti.
- Metodi dodaj in prekliči služijo za ravnanjem s certifikatom.

Razred **KPIS**:
- Razred KPIS predstavlja cel sistem

Razred **VSTOP**:
- Razred VSTOP predstavlja posamezni vstop v sistem. Zaradi občutljivih podatkov je potrebno zagotoviti popolno kontrolo nad tem, kdo podatke vidi ali jih celo spreminja. Razred VSTOP opisuje posamezni vstop v sistem.
- Atributi razreda natančno opisujejo posamezni vstop v sistem.
- Metode služijo za preverjanje ključa in izvedbo vstopa.

Razred **NADZOR**:
- Razred NADZOR predstavlja modul v sistemu KPIS. Modul Nadzor predstavlja tudi enoto konfiguracij v sistemu.
- Atributi razreda opisujejo enoto konfiguracije v sistemu.
- Metode so povezava modula s sistemom KPIS (meni aplikacije).

Razred **OBRAZEC**:
- Razred OBRAZEC predstavlja zaslonsko masko grafičnega uporabniškega vmesnika.
- Atributi predstavljajo skupne podatke obrazca.
- Metode služijo za interakcijo obrazca z uporabnikom, ki jih imajo vsi obrazci.

Razred **NAJAVIUPORABNIKA**:
- Razred NAJAVIUPORABNIKA je podrazred razreda OBRAZEC. Predstavlja zaslonsko masko za vnos zahteve, za izdelavo certifikata določenemu delavcu in se ga vključi kot uporabnika v sistem KPIS.
- Atributi razreda opisujejo podatke zaslonske maske osnovne lastnosti uporabnika.
- Metode služijo za interakcijo obrazca z uporabnikom in so specifične temu obrazcu.

Razred **UREDIUPORABNIKA**:
- Razred UREDIUPORABNIKA je podrazred razreda OBRAZEC. Predstavlja zaslonsko masko za vnos zavodov, katerega podatke uporablja uporabnik in kakšne vloge ima nad temi podatki.
- Atributi razreda opisujejo podatke zaslonske maske osnovne lastnosti uporabnika.
- Metode služijo za interakcijo obrazca z uporabnikom in so specifične temu obrazcu.

Razred **UPOSOLA**:
- Razred UPOSOLA v sistemu predstavlja povezavo med podatki uporabnika in zavoda. Razred pove da lahko uporabnik upravlja s podati določenega zavoda.
- Atributi razreda opisujejo osnovne lastnosti o tem, kateri uporabnik in zavod sta povezana.

- Metode služijo zato, da se vzpostavi ali umakne povezava uporabnik zavod.

Razred **VLOGA**:

- Razred VLOGA predstavlja pravico uporabnika nad določenimi podatki.
- Atributi razreda opisujejo posamezno vlogo.
- Metode služijo zato, da se določena vloga uporabniku doda ali odvzame.

Razred **PORTAL**:

- Razred PORTAL predstavlja zunanji razred in opisuje vstopno točko za vse aplikacije.
- Atributi razreda opisujejo osnovne lastnosti razreda.
- Med metodami je prikazana le tista, ki služi za povezavo v sistem KPIS in sicer je to avtorizacija, ki vrne veljaven žeton za vstop.

Razred **ZAVOD**:

- Razred ZAVOD je zunanji razred sistema KPIS in predstavlja podatke zavodov v registrih MŠŠ.

Razred **SREDNJASOLA**:

- Razred SREDNJASOLA je podrazred razreda ZAVOD in predstavlja srednješolski zavod.

Razred **OSNOVNASOLA**:

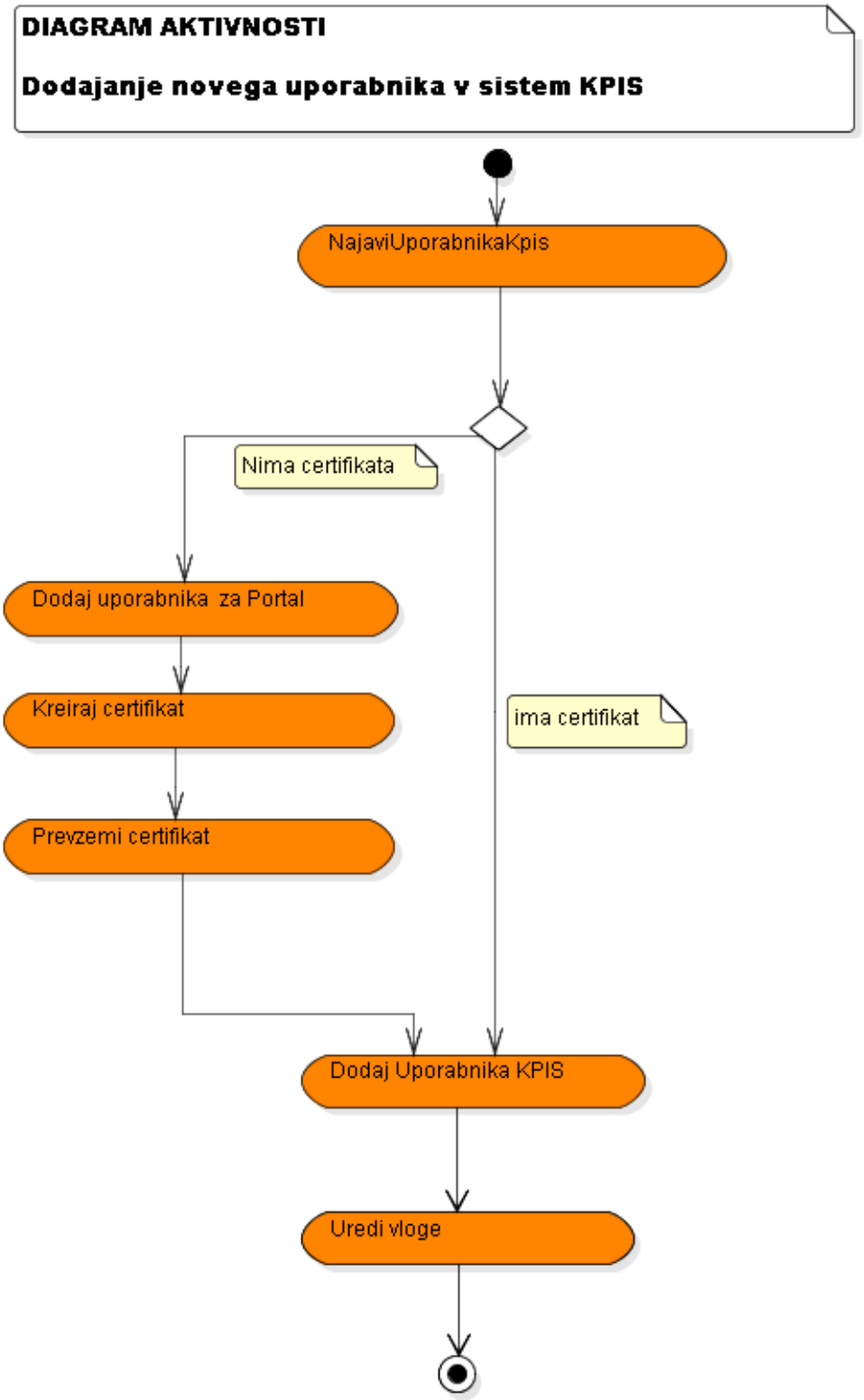
- Razred OSNOVNASOLA je podrazred razreda ZAVOD in predstavlja osnovnošolski zavod ali glasbeno šolo.

Razred **MINISTRSTVO**:

- Razred MINISTRSTVO je podrazred razreda ZAVOD in se uporablja zato, da se identificirajo tudi uporabniki ministrstva.

#### 5.4.2.2 Analiziranje primerov uporabe

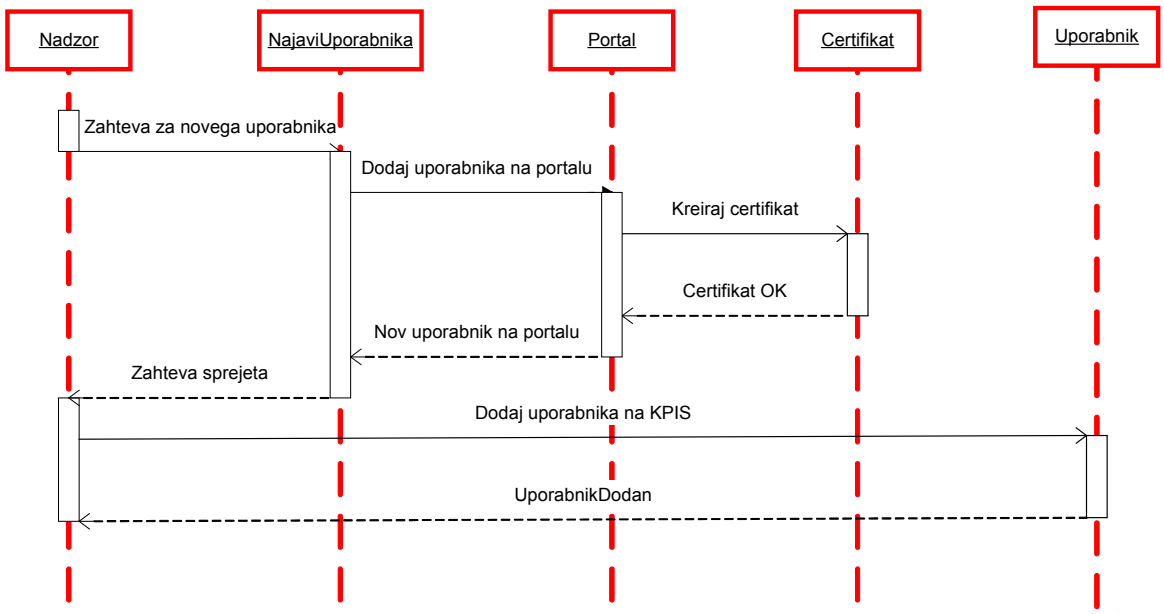
Tukaj smo podrobneje analizirali v fazi zajema zahtev izdelani diagram primerov uporabe. Podrobneje smo opisali vsak primer uporabe in določili aktivnosti posameznega primera uporabe. Na sliki 5.5. navajam podrobneje opisan primer uporabe Dodaj uporabnika.



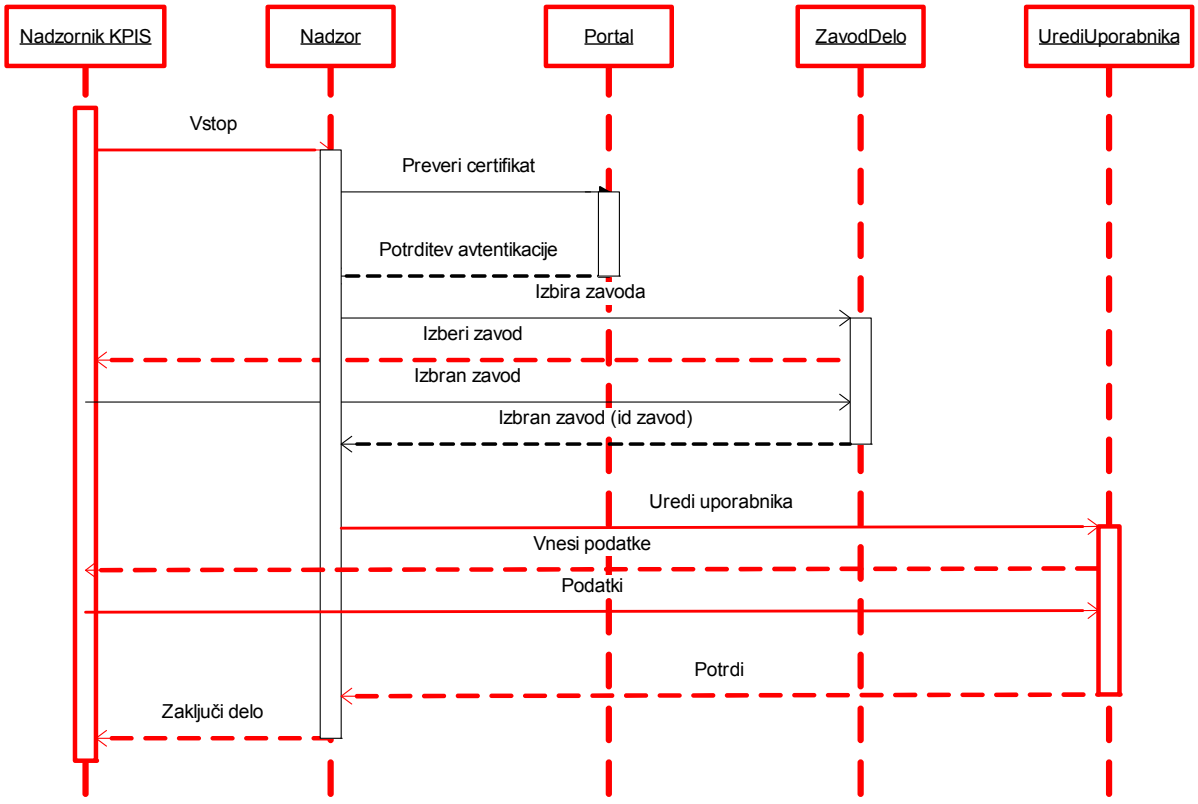

**Slika 5.5: Prikaz aktivnosti DODAJ UPORABNIKA**

#### 5.4.2.3 Diagrami zaporedja

Diagrama zaporedja podrobneje opisujeta dva primera uporabe iz diagrama primerov uporabe na sliki 5.3 in sicer Dodaj uporabnika (slika 5.6) in Dodajanje zavoda in vlog uporabniku (slika 5.7).




**Slika 5.6: Diagram zaporedja dodajanja uporabnika v sistem KPIS**




**Slika 5.7: Diagram zaporedja dodajanja zavoda in vlog uporabniku**

### 5.4.3 Načrtovanje modula

Po analizi je sledilo načrtovanje aplikacije. Na podlagi konceptualnega razrednega diagrama smo izdelali razredni diagram, ki je bil osnova za izdelavo danih objektov in shranjenih procedur (*stored procedure*). Izdelali smo tudi komponentni diagram, ki je osnova za izdelavo komponent tankega odjemalca.

#### 5.4.3.1 Razredni diagram

Razredni diagram na sliki 5.8. prikazuje le razrede, ki so del modula za nadzor. Na sliki niso prikazani razredi drugih modulov. Za vsak razred smo podrobneje določili atribute in operacije.

Razred **UPORABNIK**:
- Razred UPORABNIK je osnovni razred modula Nadzor.
- Atributi razreda UPORABNIK opisujejo osnovne lastnosti uporabnika (ime, priimek, EMSO, E-posta,..).
- Metode služijo za vnos podatkov uporabnika in urejanje podatkov uporabnika.

Razred **VLOGA**:
- Razred VLOGA predstavlja pravice (vloge) uporabnikov.
- Atributi razreda Vloga so: Ime – naziv vloge, Prioriteta (prednostni vrstni red) – število, ki posamezne vloge ureja od najpomembnejše (največ pravic) do najmanj pomembne (najmanj pravic).
- Metode služijo za dodajanje in odvzemanje vlog.

Razred **SOLA**:
- Razred SOLA predstavlja zavod (osnovna šola, srednja šola ali visoka strokovna šola).
- Prikazani so le atributi, ki so pomembni za modul Nadzor.
- Metode razreda Sola niso prikazani, ker za modul Nadzor niso pomembni.

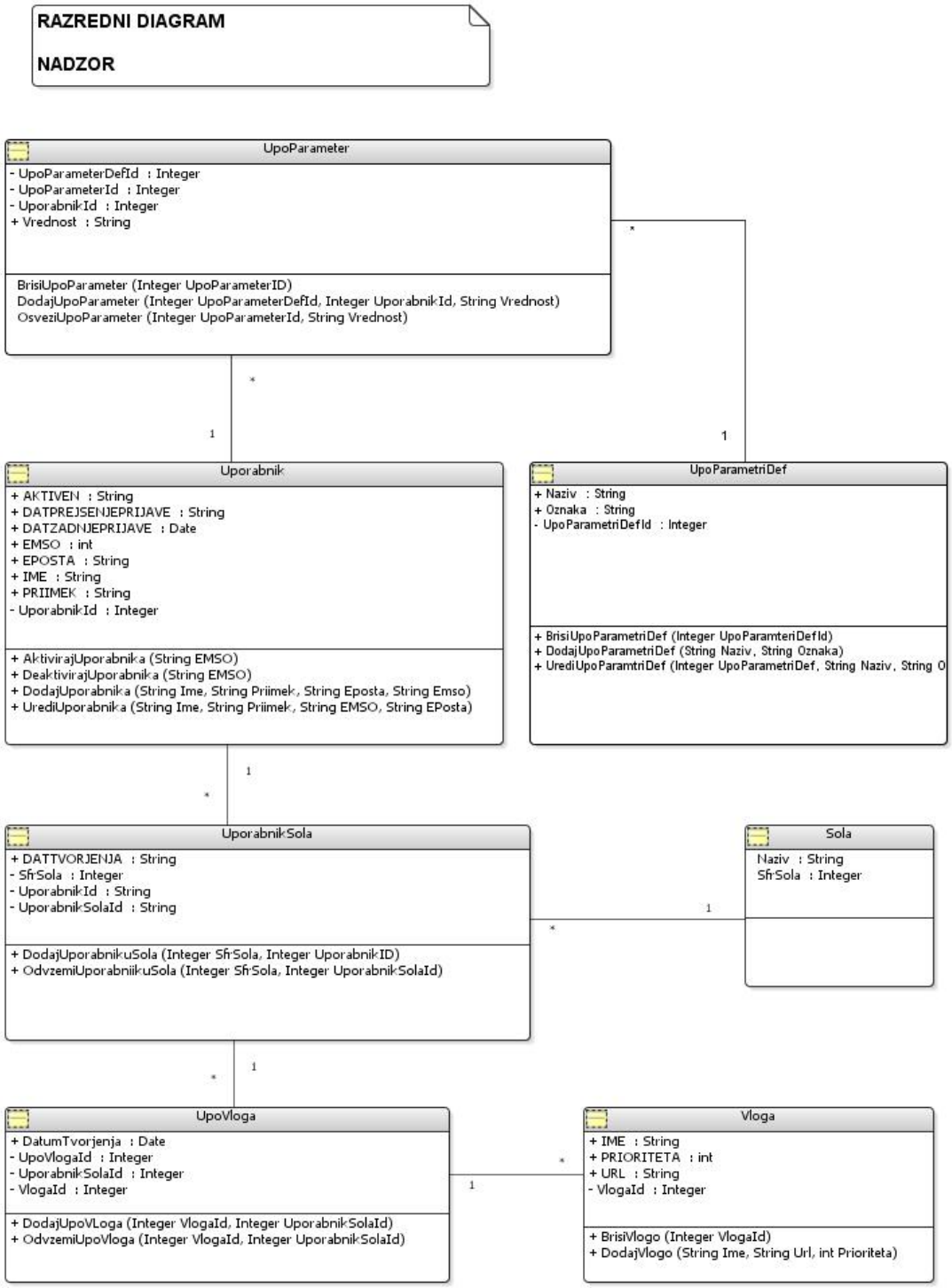
Razred **UPOPARAMETRIDEF**:
- Razred UPOPARAMETRIDEF predstavlja vrsto parametra, ki se vodi za vsakega uporabnika. Preko parametrov shranjujemo različna stanja oziroma nastavitev uporabnika v sistemu.
- Atributa razreda UPOPARAMETRIDEF sta Naziv in Oznaka .
- Metode služijo za dodajanje in odvzemanje parametrov.

Razred **UPOPARAMETER**:
- Razred UPOPARAMETER predstavlja vrednost določenega parametra za uporabnika.
- Atribut razreda UPOPARAMETER je Vrednost, kjer je shranjena trenutna vrednost parametra.
- Metode služijo za dodajanje, odvzemanje in urejanje vrednosti parametra uporabnika.

Razred **UPORABNIKSOLA**:
- Razred UPORABNIKSOLA predstavlja pravico, da lahko uporabnik upravlja s podati določenega zavoda.
- Atribut razreda UPORABNIKSOLA vsebuje le datum tvorjenja pravice.
- Metode služijo za dodajanje in odvzemanje uporabniku pravice do določene šole.

Razred **UPOVLOGA**:

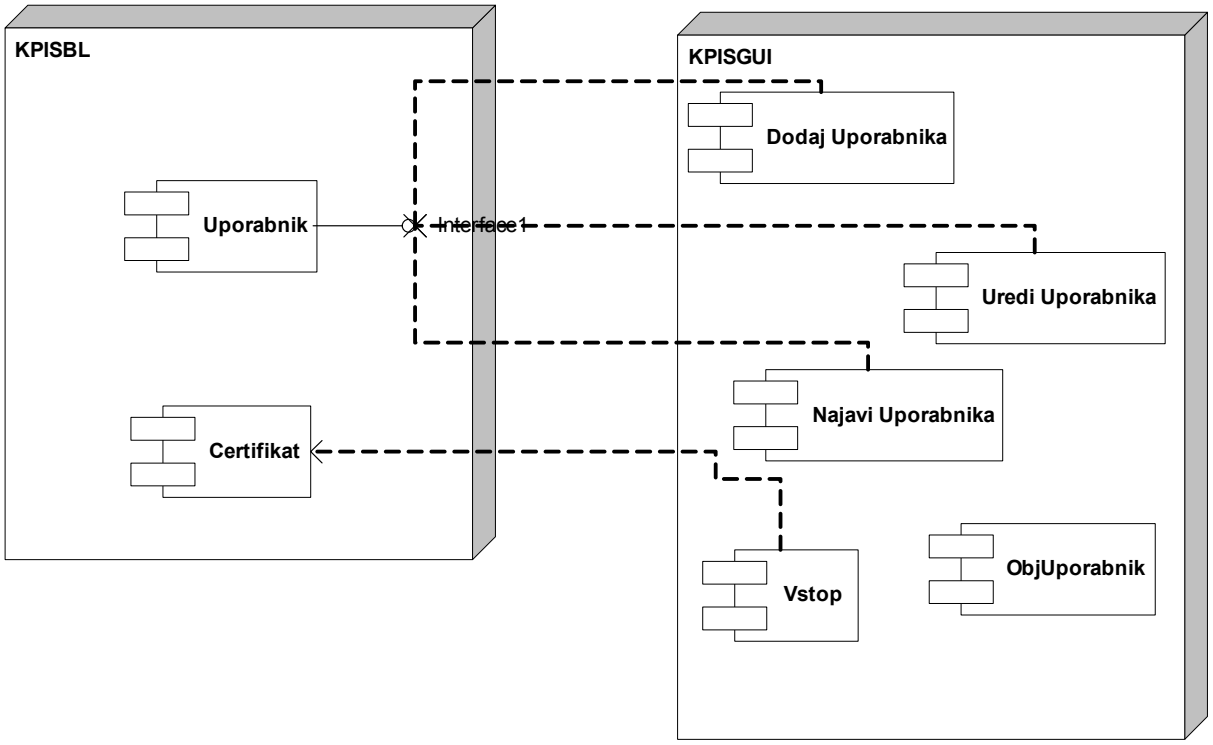
- Razred UPOVLOGA predstavlja, katere operacije lahko uporabnik izvaja s podatki šole.
- Atribut razreda UPOVLOGA vsebuje le datum tvorjenja pravice.
- Metode služijo za dodajanje in odvzemanje uporabniku pravice do določene šole.




**Slika 5.8: Razredni diagram za modul nadzor, ki je podlaga za bazne objekte in shranjene procedure (postopke) za interakcijo (součinkovanje) s tankim odjemalcem.**

#### 5.4.3.2 Komponentni diagram

Komponentni diagram predstavlja gradnike uporabniškega vmesnika (tankega odjemalca). V osnovi ločim dva nivoja. KPISGUI predstavlja predstavitveni nivo aplikacije, medtem ko KPISBL predstavlja poslovni in podatkovno raven, ki sta v tem primeru združena. Komponenti Uporabnik in Certifikat predstavljata dostop do podatkov uporabnika in njegovega certifikata. Komponente na strani KPISGUI pa predstavljajo grafično podobo obrazca (Dodaj Uporabnika, Uredi Uporabnika, Odjavi porabnika) oziroma postopke (Vstop)




**Slika 5.9: Komponentni diagram predstavlja gradnike uporabniškega vmesnika**

### 5.4.4 Implementacija in testiranje

Implementacija modula KPIS je potekala na dveh nivojih. Na podlagi razrednega diagrama je bilo potrebno izdelati bazne objekte in shranjene procedure, ki so služile interakciji odjemalca s podatki. Tanki odjemalec je bil implementiran na ogrodju .NET z orodjem Visual Studio .net 2005. Za povezavo s podatki je uporabljen Microsoft OracleClient for. NET.

Modul za nadzor je bil v test predan 1.9.2005. Nadzorni modul aplikacije KPIS je bil tudi testni modul za dodeljevanje in preverjanje delovanja portala in kontroliranega dostopa do aplikacij. Najprej je bil omogočen dostop ravnateljem 20 šol. Po 14 dnevnem obdobju (20 testnih uporabnikov je brez problema prišlo do testnih vsebin aplikacije KPIS) so na ministrstvu začeli dodeljevati certifikate še ostalim ravnateljem šol in drugim delavcem

(računovodje). Na osnovi predlogov administratorja smo naredili še nekaj dopolnitev modula, ki je bil v delovanje predan 1.11.2005, ko je bila nameščena tudi prva verzija aplikacije KPIS.

Dostop do osnovnih strani portala je mogoč na naslovu: http://portal.mss.edus.si. Dostop do zaprtih vsebin portala in aplikacij je mogoč le uporabnikom z ustreznimi spletnimi certifikati in ustreznimi pravicami.

Test aplikacije je potekal na dveh nivojih. Prvi je pri izvajalcu na razvojnem strežniku Borg. Drugi test pa je potekal pri naročniku na njegovem testnem strežniku DravaII.

Testiranja (preizkus) so potekala po ustaljenem postopku podjetja RRC za razvoj novih programov. Najprej so bili izdelani načrti testiranja. Za aplikacijo se najprej opredelijo vse lastnosti, za vsako lastnost se napiše testni postopek. Skupina testnih postopkov predstavlja načrt testiranja. Na podlagi načrta testnega načrta skupina za testiranje izvede test. V primeru odkritih napak se program vrne v dopolnitve in nato v ponovni test. Primer testnega načrta predstavlja priloga 4, kjer je opisan postopek testiranja osnovnih lastnosti programa KPIS. Na vsakem obrazcu (zaslonu) programa je potrebno testirati nekatere stvari, ki so skupne vsem obrazcem (slovnična pravilnost, navodila za uporabo, poimenovanje, opozorila uporabnikom o napaki, odzivnost oz. hitrost, ..).

# 6 Vodenje projekta KPIS

## 6.1 Uvod

Projekt KPIS je eden od večjih projektov v podjetju RRC. Projekt je vzpostavljen v okviru Enote za javni sektor. Projekt je voden v skladu z pravilniki in uveljavljeno prakso razvoja in vzdrževanja informacijskih sistemov v podjetju RRC.

## 6.2 Proces razvoja in vzdrževanja IS v RRC

Zahteve naročnika so danes pretežno nadzorovane z uporabo celotnega obvladovanja kakovosti (*Total Quality Management – TQM*). TQM je sistem, ki nenehno izboljšuje in združuje različne organizacijske elemente v načrtovanje, razvoj in implementacijo. Usmerjen je naročniku (kupcu) in mu zagotavlja stroškovno učinkovite proizvode (storitve) in primerno uporabnost. V organizaciji skrajšuje čase izdelave, odpravlja ozka grla, kar zviša kakovost proizvoda (storitve) in izboljša organizacijsko moralo [7].

Osnovi dokument (listina) obvladovanja kakovosti v podjetju RRC je Poslovnik kakovosti [22]. V njem je opisana politika kakovosti podjetja, organizacija podjetja, tipi projektov, procesi in sistem kakovosti. Pravilnik opredeljuje zunanje in notranje projekte:

- Zunanji projekti se izvajajo za zunanje naročnike in so oblikovani z namenom, da ustvarijo prihodek.
- Notranji projekti so projekti, ki jih družba izvaja za lastne potrebe ter so nujni za njeno normalno delovanje in razvoj v strokovnem in poslovnem smislu. Izvajajo se v enoti za kompetence ter v treh enotah oddelka za podporo poslovanju.

Glavni procesi[3] v podjetju so [22]:

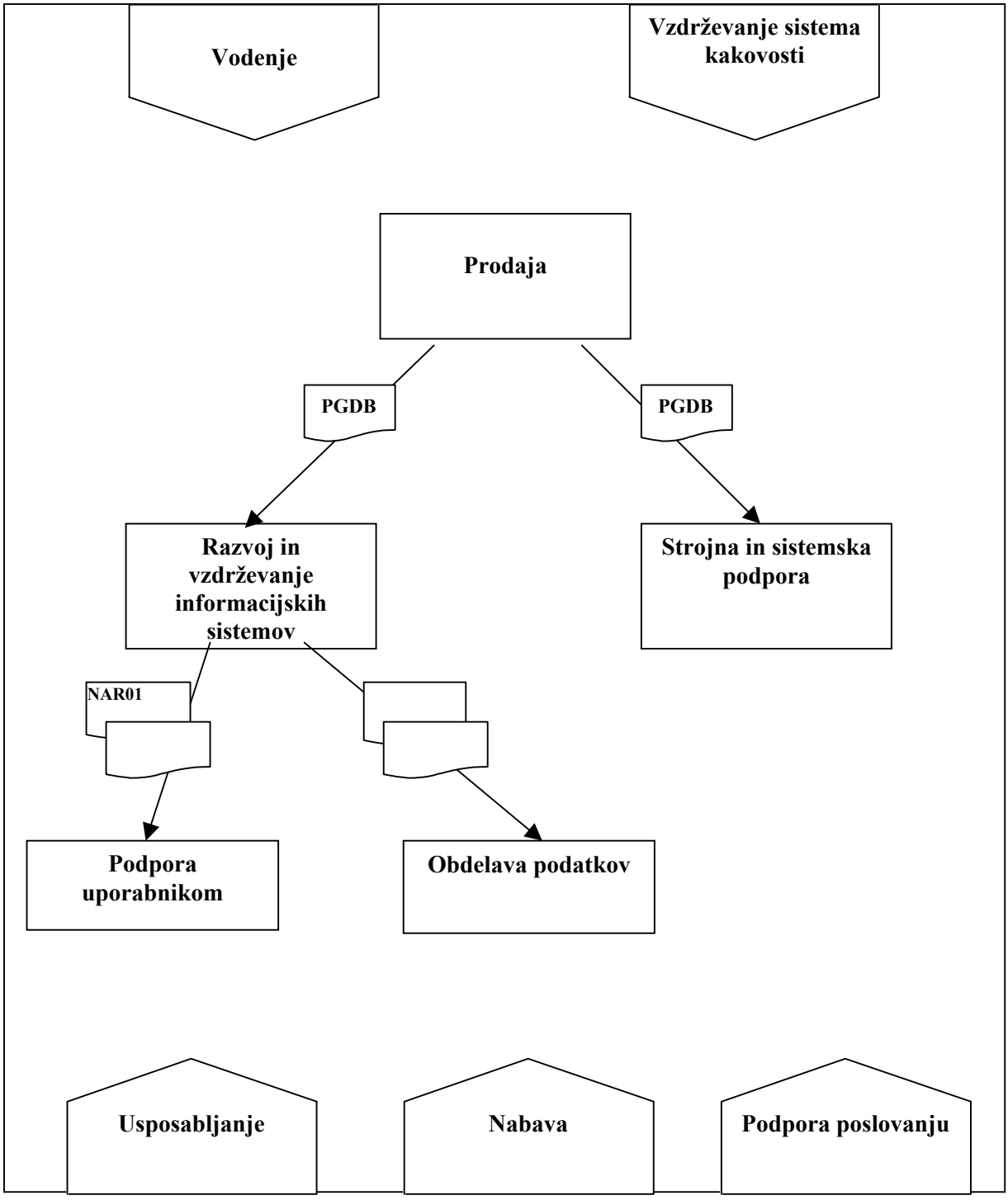
- vodenje,
- prodaja,
- razvoj in vzdrževanje informacijskih sistemov,
- strojna in sistemska podpora,
- vzdrževanje sistema kakovosti.

Poleg glavnih procesov potekajo v podjetju tudi podporni procesi [22]:

- usposabljanje,
- nabava,
- podpora poslovanju.

**Procesa razvoja in vzdrževanja IS** in **strojne in sistemske podpore** sta neposredno odvisna od procesa prodaje. Medsebojno sta neodvisna. Procesa sta razdeljena na posamezne projekte na osnovi sklenjenih pogodb z večjimi odjemalci. Povezave med procesi prikazuje slika 6.1.

[3] Poimenovanje procesa ni v skladu s terminologijo po Mihelčiču [10], ki opredeljuje delovni proces kot vsako (smotrno) človekovo dejavnost, katere namen je ustvarjanje opredmetenih in neopredmetenih dobrin in ga členimo na poslovni ali izvedbeni in organizacijski ali upravljalno – ravnateljevalni proces.




**Slika 6.1: Grafični prikaz povezav med osnovnimi in podpornimi procesi [22]**

Eden od dokumentov sistema kakovosti je tudi Razvojni model, ki opisuje različne modele razvojnega procesa, kateri temelje na različnih razvojnih metodologijah. Namen tega standardnega postopka je podati pregled in značilnosti razvojnih modelov s poudarkom na tistih modelih, ki jih uporabljamo v RRC.

Prva zelo pomembna ugotovitev v dokumentu je ta, da opisani modeli razvojnega procesa niso nekaj statičnega, ampak jih razvijajo glede na potrebe in glede na razvoj metodološkega inženirstva.

Po poslovniku kakovosti v RRC uporabljamo naslednje modele razvojnega procesa:

1. **Konvencionalni model, ki pokriva celotni življenjski razvojni cikel določenega projekta.** Zanj se odločimo, če nas je naročnik izbral le za ta projekt in ne želi vlagati v sodobna razvojna orodja in tem orodjem ustrezne strojne in sistemske programske platforme. Je pa tudi izhodišče za vzdrževanje in PO, ki jo je RRC razvil po tem modelu.

Njegove razvojne faze (stopnje) so:
- priprave na projekt,
- analiziranje zahtev,
- izdelava načrta arhitekture,
- gradnja sistema in testiranje,
- izdelava uporabniške dokumentacije,
- vpeljava in prevzem.

2. **Model informacijskega inženirstva oz. njegova inačica CASE Method**, ki so jo razvili pri združbi ORACLE, uporabljamo v RRC takrat, ko naročnik pričakuje, da najprej izdelamo strateški razvojni načrt in v tem okviru visokonivojski podatkovni in funkcijski model za celotno združbo ali za njen pomembni del. Visokonivojski podatkovni model je dobra osnova za povezovanje sistemov, ki so sicer lahko ločeno zgrajeni. Pri gradnji modelov velja načelo, da je izhodišče modeliranja dobro opredeljen poslovni (upravni) sistem s svojimi dolgoročnimi cilji in problemi. Pri tem je treba upoštevati tudi trenutno stanje in bodoči razvoj informacijske tehnologije.

Razvojne faze modela informacijskega inženirstva obsegajo:
- strateško načrtovanje
- analizo zahtev
- sistemsko zasnovo
- gradnjo (programiranje)
- izdelavo uporabniške dokumentacije (ta faza poteka vzporedno s samo gradnjo)
- vpeljava in prevzem.

3. **Hitri (ekspresni) razvoj aplikacij (Rapid Application Development - RAD) [9]** je zlasti učinkovit v okviru informacijskega inženirstva, kjer je bil okvirni podatkovni (entitetni) model že razvit. Model je možno uporabiti tudi povsem samostojno za celotni življenjski razvojni cikel PO. Model ima običajno 4 razvojne faze in sicer analiziranje zahtev, izdelava načrta arhitekture, gradnja prototipa in vpeljava. Običajno je življenjski cikel iterativen (vsaka večja dopolnitev gre skozi vse naštete razvojne faze).

4. **Model vzdrževanja** uporabljamo v RRC, kadar prevzamemo vzdrževanje in dopolnjevanje programskega izdelka, ki je bil praviloma razvit v RRC.

- **Vzdrževanje aplikacij, ki so zgrajene z orodji CASE**

Ker je vse znanje o aplikaciji zbrano v skladišču (repository), poteka tudi vzdrževanje aplikacije preko skladišča. Zato moramo vedeti, da v tem primeru spreminjamo zasnovo aplikacije, ker želimo, da opravijo vsa kodiranja generatorji kode.

- **Vzdrževanje lastnih proizvodov**

Ne glede na uporabljeni razvojni model je proces vzdrževanja enak. Naročnik in RRC oblikujeta skupno telo, ki sprejema (odklanja) zahteve po dopolnitvah. Uresničitev sprejetih dopolnil spremlja skupno telo po načelih projektnega vodenja. V odvisnosti od vrste vzdrževanja (odprava programskih napak, manjša ali večja dopolnila) je opredeljen izvedbeni projekt.

Med modeli razvojnega procesa v RRC trenutno še ni formalno evidentiran objektni proces razvoja, za katerega pa na podlagi izkušenj razvoja aplikacije KPIS pripravljamo dokumentacijo.

## 6.3 Projekt MŠŠ (ŠKIS-KPIS)

Projekt ministrstva za šolstvo in šport v podjetju poteka že od leta 1997. V letu 2004 je bil podpisan sporazum o projektu KPIS za obdobje 2004 – 2006. V poglavju 5 je projekt opisan vsebinsko, prav tako pa je del poglavja tudi opis objektnega pristopa pri razvoju modula aplikacije KPIS. Namen tega poglavja pa je prikazati sistem dela na projektu, kadrovanje na projektu.

### 6.3.1 Potek dela na projektu

Po pogodbi z naročnikom sta tako s strani naročnika kot tudi s strani izvajalca določena skrbnika pogodbe. Skrbnik pogodbe s strani RRC je vodja projekta. Predmet pogodbe je vzdrževanje in produkcijski tek obstoječe aplikacije ŠKIS in izdelava nove aplikacije KPIS. Po pogodbi so predvidene okvirne projektne naloge, letni obseg (v urah) in vrednost opravljenih del. Skrbnika pogodbe morata zagotavljati odpiranje in zaključevanje naročil v okviru pogodbe.

V okviru projektnih nalog na podlagi zahtev naročnika sproti odpiramo naročila. Naročilo lahko opredelimo kot zaključeno nalogo v okviru projekta. Naročila so lahko manj obsežna ali obsežnejša.

Primeri manj obsežnih naročil:

- Izdelava izpisa.
- Dopolnitev modula za vnos podatkov delavca.
- Dodatno spremljanje podatka na razporeditvah

Primeri obsežnejših naročil:

- Izdelava modula za vnos podatkov o delovnih nalogah.
- Dopolnitev obračuna plač (obračunavanje pomočnikov ravnateljev na nov način).
- Izpis preglednic 1, 2 in 1A (plače delavcev).

Preglednica 6.1 prikazuje življenjski cikel posameznega naročila. Za vsako stopnjo so opisane naloge, ki jih je potrebno izvesti.

| **Faza (stopnja)** | **Aktivnosti (dejavnosti)** |
|---|---|
| Specifikacija naročila | Na osnovi naročnikovih zahtev okvirno opredelimo naročilo. Izvajalec pripravi predlog naročila z okvirno oceno obsega in roka izvedbe. Vse to zapišemo v dokument Specifikacija naročila (SNxxyyzzz.doc). |
| Potrditev naročila | Na osnovi predloga naročila izvajalec naročilo potrdi, kar pomeni, da začnemo z izvedbo. |
| Izvedba naročila | Izvajalec poskrbi za izvedbo naročila. V primeru, da zahteve naročnika še niso podrobno zapisne, je potrebno izvesti še analizo. Ko so naročnikove zahteve znane, se začne z izdelavo. Pri izvedbi izvedemo vse aktivnosti za izdelavo informacijskih sistemov (načrtovanje, programiranje, testiranje, priprava dokumentacije in namestitev) |
| Zaključek naročila | Ko je delo končano, se dopolni dokument SNxxyyzzz.doc z delom o zaključku naloge (poročilo). Zaključeno naročilo je osnova za izdelavo in izdajo računa. |

**Preglednica 6.1: Opis faz (stopenj) v življenjskem ciklu izvedbe naročila.**

Osnovni dokument naročila je SN – Specifikacija naročila. Primer zaključenega naročila predstavlja priloga 5. SN se dopolnjuje tekom izvedbe naročila. Preden se naročilo formalno preda v izvedbo, je potrebno s strani izvajalca (RRC) opredeliti predvidene naloge, iz katerih izhaja tudi predviden obseg naročila in rok izvedbe. Ko se naročila preda v izvedbo, se podrobnejše popišejo uporabniške zahteve (v SN se popiše le osnovne zahteve, podrobnosti pa v ločen dokument). Po izvedbi naročila se izpolnijo še podatki poročila, naročnik pa s podpisi potrdi zaključek naročila (prevzem).

Naročilo obsega naslednje elemente:

- razlogi za odprtje naročila,
- opis rešitve,
- vrsto naročila,
- elemente konfiguracije,
- seznam opravil oziroma lastnosti,
- uporabniške zahteve,
- oceno obsega del in roka izvedbe,
- poročilo - zaključek.

### 6.3.2 Načrt dela in razporeditev dela

Med dogovorom z naročnikom o vzpostavitvi projekta (javni razpis) smo ugotovili, da bomo težko izpeljali projekt z obstoječimi kadri. Zato smo takoj objavili razpis za novega delavca na projektu. Po podpisu pogodbe pa smo naredili načrt izvedbe projekta, kjer smo definirali ure po posameznih obdobjih (letno in mesečno).

Za potrebe projekta smo v začetku pristopili k pridobitvi dodatnih delavcev. Na podlagi izkušenj smo se odločili za preverjenega delavca s predhodnimi izkušnjami na področju, za katerega smo rabili nove moči.

Obseg del v obdobju treh let je bil ocenjen na 75 človek/mesec in sicer vsako leto 25/človek leto. Projekt prenove smo razdelili na tri module, katerih izdelava se je bolj ali manj pokrivala s koledarskimi leti :

- modul za nadzor uporabnikov – izdelava v letu 2004,
- modul za prikazovanje podatkov o plačah – izdelava v letu 2005,
- modul za vnos podatkov za izračun plače – izdelava v letu 2006.

Za vsako leto smo naredili podrobnejši načrt razporeditve delavcev glede na predvidene aktivnosti, ki smo jih načrtovali skupaj z naročnikom. Časovni načrt na projektu v letu 2006 je prikazan v prilogi 6.

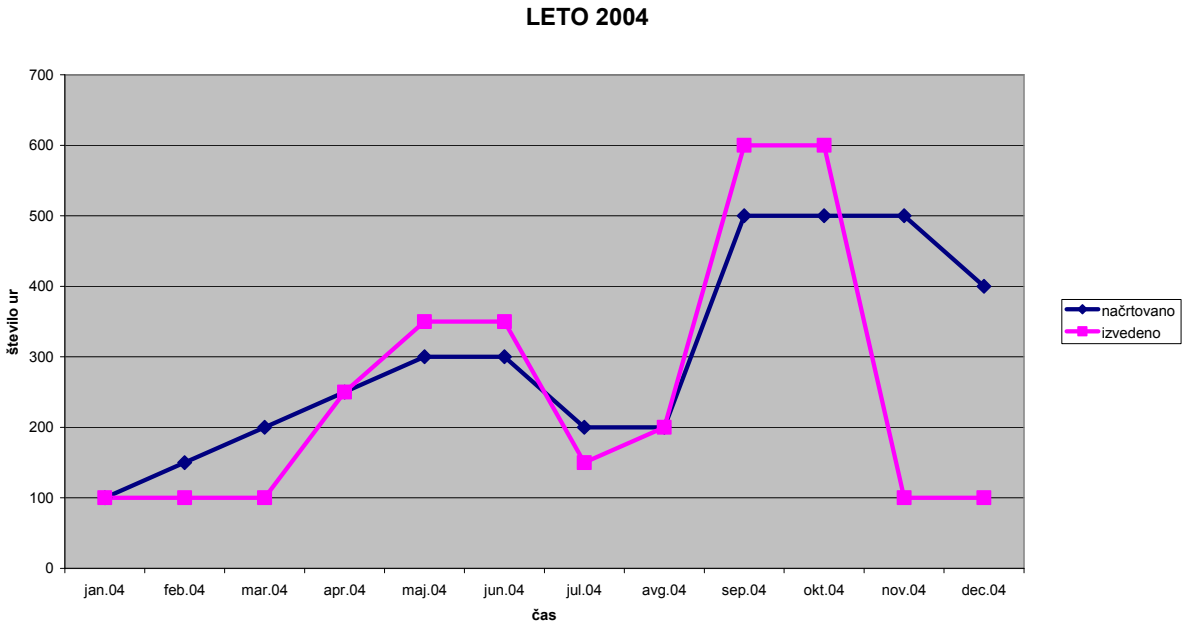
| Zaposleni | Skupaj | MORS | DURS.NET | DURS_SW | MP | MŠŠ | OSTALI_SW | RRC-KOMERC | RRC-REZIJA | RRC-TECH |
|---|---|---|---|---|---|---|---|---|---|---|
| Pregel Aleš | 100% | | | | | 91% | | 3% | 6% | |
| Berce Marko | 91% | | 20% | | | 71% | | | | |
| Jakhel Žiga | 115% | | | | | | 65% | 50% | | |
| Joras Uroš | 113% | | 15% | | | 82% | | | | 16% |
| Podgoršek Peter | 110% | | 10% | | | 40% | 60% | | | |
| Seratlić Radomir | 107% | 20% | | 5% | 76% | | 3% | 3% | | |
| Degen Tomaž | 84% | | | | 84% | | | | | |
| Maksimovič Dejan | 62% | | | | 62% | | | | | |
| Kašnik Andrej | 60% | | 20% | | | | 20% | 20% | | |
| SKUPAJ ENOTA: | 4.176h | 99h | 322h | 25h | 1.101h | 1.409h | 734h | 377h | 30h | 79h |

**Preglednica 6.2: Načrtovana zasedenost delavcev OKTOBER - DECEMBER 2005**

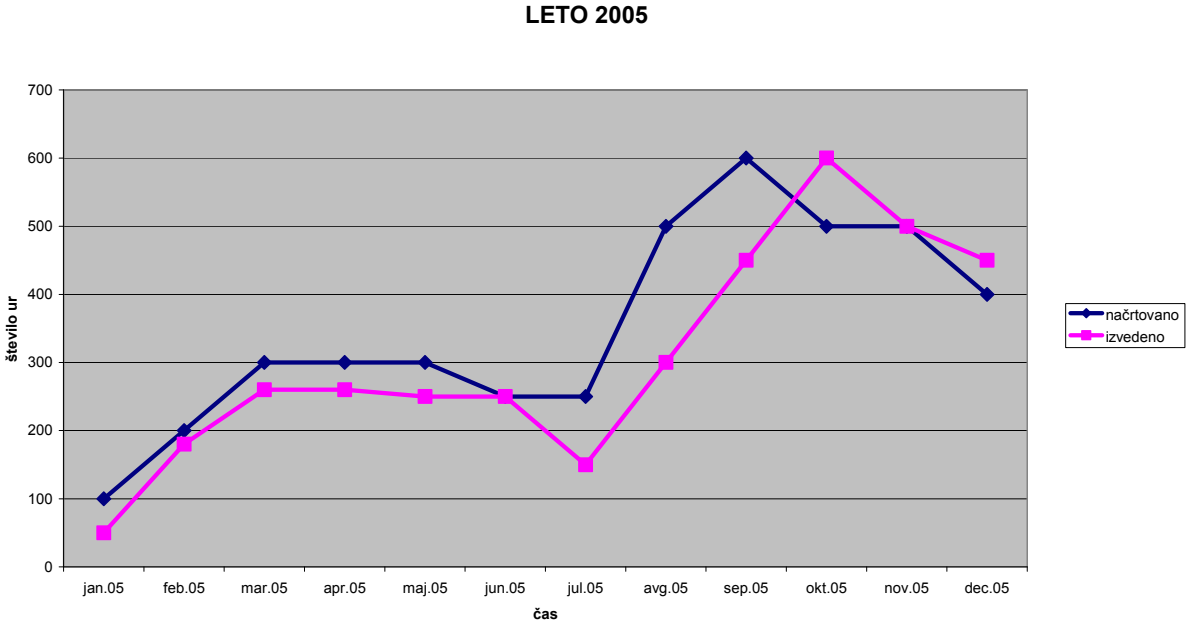
Za načrtovanje zasedenosti delavcev za vsako leto in vsako tromesečje pripravimo načrtovano zasedenost delavcev. Preglednica 6.2 prikazuje načrtovano zasedenost vseh delavcev v Enoti za javni trg za obdobje oktober 2005 do december 2005. Iz preglednice 6.3 je razvidno, da smo na projektu MŠŠ načrtovali 1.400 ur. Odstotek zasedenosti preko 100%

pomeni, da je bilo pri teh delavcih načrtovano opravljanje nadurnega dela. Zasedenost pod 100% pa je opravičena zaradi tega, ker delavci niso bili prisotni vse tri mesece (novi delavci). Iz slike 6.3 je v primerjavi s preglednico 6.2 razvidno, da smo v zadnjem tromesečju leta 2005 opravili celo 150 ur več od načrtovanih 1.400 ur.

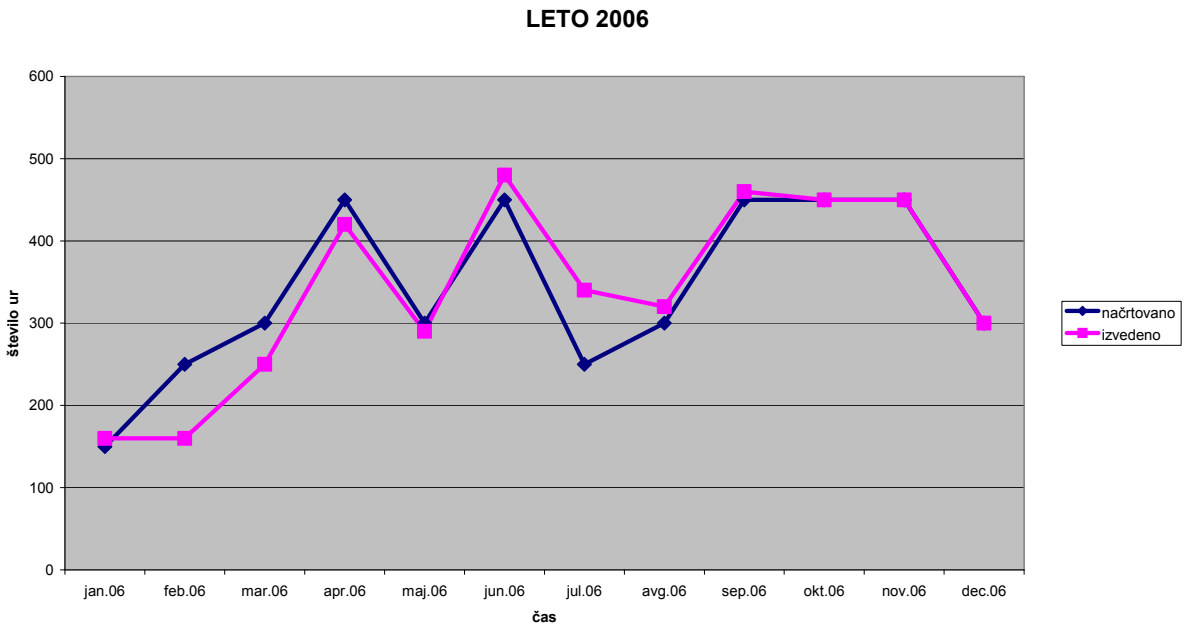
Na slikah 6.2, 6.3 in 6.4 je razvidna razlika med načrtovanimi in opravljenimi urami v posameznih mesecih v obdobju trajanja projekta.




**Slika 6.2: Razmerje načrtovanih in izvedenih ur na projektu v letu 2004**




**Slika 6.3: Razmerje načrtovanih in izvedenih ur na projektu v letu 2005**




**Slika 6.4: Razmerje načrtovanih in izvedenih ur na projektu v letu 2006**

| **Oseba** | **Oseba 1** | **Oseba 2** | **Oseba 3** | **Oseba 4** | **Oseba 5** | **Oseba 6** |
|---|---|---|---|---|---|---|
| **Vloga (Delovni program)** | Aleš | Uroš | Marko | Janez | Peter | Roman (MŠŠ) |
| Uporabnik, strokovnjak problemskega področja | | | | | | X |
| Snovalec primerov uporabe | X | | | | | |
| Arhitekt | X | X | | | | |
| Načrtovalec uporabniškega vmesnika | X | X | | | | |
| Načrtovalec primerov uporabe | X | | | | | |
| Inženir komponent | | X | | | | |
| Projektni vodja | X | | | | | |
| Sistemski integrator | X | X | X | | | |
| Načrtovalec | X | X | | | | |
| Programer | | X | X | X | X | X |
| Tester | | X | X | X | | |
| Pregledovalec | X | | | | | |
| Analitik | X | | | | | |

**Preglednica 6.3: Matrika vlog (delovnih programov) pri objektnem razvoju IS in članov projektne skupine**

Za vse načrte je bilo potrebno upoštevati tudi vrsto dela, ki ga posamezen delavec opravi. Vloge (delovne programe), definirane po objektni metodologiji razvoja informacijskih sistemov, smo med člane projektne skupine razdelil kot prikazuje preglednica 6.3. Kot je razvidno iz preglednice, imajo določeni delavci več vlog. Prav tako pa nekatere vloge opravlja več delavcev.

V okviru projekta je bilo potrebno opraviti več različnih delovnih nalog. Delovne naloge opravljajo delavci glede na njihov vloge:

- Projektni vodja
    - izpolnjevanje pogodbe
    - odpiranje naročil
    - zaključevanje naročil
    - načrtovanje zasedenosti delavcev
    - priprava poročil (vmesna in končno)
- Analitik, Snovalec primerov uporabe, Načrtovalec primerov uporabe
    - sestanki z uporabniki
    - priprava uporabniških specifikacij
    - izdelava diagramov primerov uporabe
- Arhitekt
    - načrtovanje arhitekture
- Inženir komponent
    - načrtovanje komponent
- Načrtovalec uporabniškega vmesnika
    - načrtovanje vmesnikov
- Načrtovalec
    - načrtovanje podatkovne baze
- Programer
    - izdelava podatkovnih struktur (Oracle 10g)
    - izdelava procedur za dostop do podatkov (poslovna logika) (Oracle PL-SQL)
    - izdelava grafičnega uporabniškega internetnega vmesnika (. NET arhitektura)
- Tester
    - priprava testnih načrtov
    - izvedba testiranja
- Sistemski integrator
    - namestitev na razvojne strežnike
    - namestitev na testne strežnike pri naročniku
    - namestitev na produkcijske strežnike pri naročniku

### 6.3.3 Merjenje uspešnosti projekta

Podjetje mora biti v svojem delovanju uspešno, to pa lahko uspe le, če v okviru podjetja delujejo uspešni projekti. Prvi pogoj je ta, da projekt že ob vzpostavitvi pravilno ovrednotimo, ter se hitro, kakovostno in predvsem pravilno odločimo glede izvajanja projekta. Ta začetni korak sicer zveni preprosto, dejansko pa zahteva globoko razumevanje tako ravnateljevanja projekta (obseg projekta, trajanje, vrednost projekta, razpoložljiva sredstva in oprema, prisotno znanje, voljnost in razpoložljivost ljudi, pričakovana kakovost projektnih rešitev, kompleksnost projektov) kot samega vrednostnega in posledično tudi odločitvenega procesa [17].

Informacije za odločitev pridobivamo iz
- trdih in
- mehkih dejavnikov.

Trdi so tisti, ki jih lahko brez večjih težav merimo, kvantificiramo, ovrednotimo (npr. različni poslovni izkazi, kot so bilance stanja, izkazi poslovnega uspeha in denarnih tokov, DuPont analiza itn.). Mehki dejavniki so vsebinski podatki, ki jim moramo najprej določiti ustrezna sodila za medsebojno primerjanje, zalogo vrednosti za vsako sodilo ter izdelati drevo sodil in podati funkcijo koristnosti, ki opredeljuje uteževanje in ustrezno medsebojno povezovanje sodil. Takšni mehki dejavniki so lahko npr. ugled podjetja, vrednost blagovne znamke, kakovost organizacije združbe, analiza ergonomije, motivacija zaposlencev, usmerjenost združbe itn [17].

Oba tipa dejavnikov, mehki in trdi, se lahko med seboj dopolnjujeta (komplementarnost), nikakor pa ne smeta biti drug drugemu nadomestek. Notranje in zunanje informacije je treba med seboj pravilno primerjati in povezovati,

Vse te dejavnike in še kakšne druge je potrebno meriti tudi v času trajanja projekta. V Poslovniku kakovosti podjetja RRC je opredeljeno, da se za vsak projekt vzpostavijo metrike, s katerimi merimo uspešnost projekta.

Metrika oziroma meritev (*measurement*) je izražanje značilnosti proizvoda ali procesa v kvantitativni ali kvalitativni obliki (npr. v vrsticah kode, ceni projekta ali delovnih urah). Metrika je lahko objektivna ali subjektivna glede na to, ali so podatki rezultat štetja ali subjektivne ocene znotraj podane merske lestvice (po projektu AMI - Application of Metrics in Industry Ltd.) [22].

Namen uporabe metrik je:
- Merjenje uspešnosti projekta.
- Vzpostavitev zahtev glede kakovosti procesa in izdelka, izraženih v merljivi obliki.
- Primerjavo različnih projektov preko metričnih izidov.
- Natančnejše ocenitve napora in stroškov pri bodočih projektih.
- Vrednotenje novih razvojnih okolij.

Pobudo za uvajanje metrike lahko po Poslovniku kakovosti podjetja RRC dajo: člani projektne skupine, vodja projekta, vodja kakovosti, vodstvo RRC ali naročnik. Z definiranjem

metrike se ukvarjata pobudnik in vodja kakovosti, ki je zadolžen za uvajanje metrike. Za posamezni projekt izbiro definirane metrike potrdi vodja projekta in/ali vodstvo RRC. Vsaka metrika je definirana v obrazcu **METRdef**. Vsebina le-tega je opredeljena v naslednjih točkah:

- **Ime metrike** - Metriko imenujemo, ji dodelimo okrajšavo in mersko enoto ter določimo tip).
- **Vplivnost** - Ocenimo vplivnost na projekt (1-zelo velika, 2-velika, 3-srednja, 4-majhna).
- **Definicija metrike** - Opišemo, katere atribute možnih entitet lahko merimo ter, če je metrika **izpeljana, opišemo računski postopek.**
- **Cilji** metrike - Podamo namen uporabe metrike in morebitne omejitve.
- **Postopek analize** - Opišemo, kako metriko uporabimo, pogoje za njeno implementacijo, pričakovane oz. ciljne vrednosti rezultatov, metode analize rezultatov - orodja, postopke preverjanja, zapišemo tudi implicitne lastnosti okolja ali metod ter možne interpretacije rezultatov.
- **Odgovornosti** - Navedemo osebe, ki imajo vpogled v rezultate, pripravljajo poročila in analizirajo podatke.
- **Usposabljanje** - Podamo predlog za dodatno usposabljanje za uporabo metrike.
- **Osebe, ki so pripravile in pregledale definicijo** - Navedemo tudi odgovorne osebe, ki so pripravile in pregledale oz. odobrile definirano metriko.

Merjenja se izvajajo na zaključku časovnih obdobij. Vsekakor je to zaključek projekta, praksa pa je, da vse metrike izvajamo še ob zaključku poslovnega (koledarskega) leta. Za projekt MŠŠ izvajamo merjenje nekaterih metrik tudi med letom. Katere so te metrike, je opisano v definicijah metrik.

Primeri metrik na projektu MŠŠ:

- Razmerje prihodkov in odhodkov na projektu.
- Razmerje vhodnih (opravljenih ) in izhodnih (plačanih) ur na projektu.
- Razmerje režijskih ur (vodenje projekta, izobraževanje, uvajanje na projekt,… ) na projektu glede na vse opravljene ure na projektu .
- Število reklamacij (tudi podrobnejša analiza).
- Število ur opravljenih za reševanje reklamacij glede na opravljene ur za razvoj.
- Število ur opravljenih za reklamacije glede na ure za testiranje .
- Zadovoljstvo naročnika (intervju, vprašalnik).
- Zadovoljstvo uporabnikov (anketa).
- Tehnološki napredek - uvajanje novih tehnologij.

# 7 SKLEP

Šest let je preteklo, ko sem diplomiral na fakulteti za računalništvo. Do danes sem na področju informatike deloval v podjetju RRC in študiral na podiplomskem študiju. Z gotovostjo lahko rečem, da se moje delovanje v podjetju pokriva z mojim izobraževanjem na fakulteti za računalništvo in informatiko. Razvoj informacijskih sistemov je moje glavno delo. Programiranju se je že zgodaj pridružilo načrtovanje IS, nato dogovarjanje z naročnikom in tudi vodenje manjšega projekta, ki je z leti prerasel v projekt zelo pomemben za podjetje. Vzporedno je potekal tudi moj podiplomski študij. Praktično pri vseh predmetih sem se naučil kaj novega, kar sem takoj uporabil pri svojem delu, povratno pa sem v svojem delu črpal teme za seminarske naloge in končno tudi magistrsko nalogo.

Pri svojem delu sem spoznal to, da se je pri delu najpomembneje odločati. Dobro je, da se odločamo pravilno oz. čim bolje. Dobro pa je tudi, da se iz nepravilnih odločitev čim več naučimo. Praktično bi lahko samo za področje, ki ga opisujem v magistrski nalogi, naštel toliko odločitev, da bi presegalo sam obseg naloge. Nekatere odločitve so čisto osnovne in nanje ponavadi sploh nismo pozorni. Na primer: Ali bo gumb na vmesniku levo ali desno? Kakšne barve bo ozadje? Kateri delavci bodo skupaj v sobi? Katere računalnike bomo kupovali? Nekatere odločitev pa so zelo pomembne: Katerega delavca bomo zaposlili? Za katero podatkovno bazo se bomo odločili? Kako bomo organizirali razvoj? Kateri bo model razvojnega procesa? Kakšna je ocena obsega naročila? Pri pomembnejših odločitvah je dobro odločitev zastaviti temeljiteje in evidentirati vsa sodila, jih glede na pomembnost ustrezno ovrednotiti, nato pa različice ustrezno oceniti in se na podlagi tega odločiti. Le tako smo lahko zanesljivi, da smo naredili največ kar je možno. Pri tem si lahko pomagamo tudi z različnimi računalniškimi programi. Potrebno se je zavedati, da je računalnik le pomočnik pri reševanju večparametrskih problemov in ne more prevzeti funkcije odločevalca. Potrebno je sodelovanje med odločevalcem in računalnikom, da bo rezultat izbire med kandidati primeren [12].

Izbira kadrov je le eden od odločitvenih problemov. Moje mnenje, ki sem ga z upoštevanjem robnih pogojev tudi dokazal, je te, da je bolje zaposliti bolj izkušene kadre, še posebej če imamo projekte, na katerih jih bomo maksimalno zaposlili. Prav tako je iz odločitvenega modela razvidno, da izobraževanje lastnih kadrov niti ni tako poceni, kot zgleda na prvi pogled. Pri odločitvi je potrebno upoštevati še podatek, ki ga pri razvoju modela nisem upošteval, da se vsi začetniki niti ne razvijejo glede na pričakovanja.

Objektni razvoj IS je prinesel veliko novosti, veliko pa je ostalo tudi isto. Koncept prinaša veliko pozitivnega, če to znaš izkoristiti, kar pa ponavadi lahko dosežeš potem, ko pridobiš izkušnje. Objektni razvoj je prinesel dobro osnovo za izboljšanje procesa razvoja IS, vendar pa je potrebno temu prirediti večino postopkov.

Pri vodenju projekta sem naletel na kar nekaj težav, ki bi jih morali odpraviti. Kot vodja bi rad vzpostavil mehanizem, da bi vsak delavec delo opravil tako, kot da bi bil zanj 100%-no odgovoren. Tako bi zmanjšali število ur, ki smo jih izgubili zaradi slabo izdelanih delov programa, ko smo le te predali testni skupini v testiranje. Prav tako smo odstopali od načrtovanih razporeditev ur na projektu. Problem je bil v tem, da smo imeli proste zmogljivosti, nismo pa imeli izdelanih zahtev od naročnika. Obratno pa je bilo, da v

določenih obdobjih nismo imeli delavcev, da bi lahko zadostili željam naročnika. Na srečo smo z obojestranskim prizadevanjem uspeli odstopanja proti koncu projekta izravnati.

# PRILOGE

- PRILOGA 1: Izpis izračuna skupne Ocene delavcev brez upoštevanja povečevanja plače.
- PRILOGA 2: Izpis izračuna skupne Ocene delavcev z upoštevanjem povečevanja plače.
- PRILOGA 3: Podatkovni diagram sistema KPIS
- PRILOGA 4: Primer testnega načrta
- PRILOGA 5: Primer Specifikacije naročila SN1306005.doc
- PRILOGA 6: Časovni načrt projekta MŠŠ za leto 2006

# LITERATURA

## AVTORSKA LITERATURA

[1] ARLOW, Jim, NEUSTADT, Ila: UML and the Unified Process. Boston: Addison-Wesley, 2002, 395 str.

[2] BOHANEC, Marko, RAJKOVIČ, Vladislav: Večparametrski odločitveni modeli, Organizacija, 28(7), 1995, strani 427-438.

[3] CONSTANTINE, Lary: Work organization: paradigms for project management and organization, Communications of the ACM, Oktober 1993 36 (10), strani 35-43.

[4] DIMITROV, Evgeni, SCHMIETENDORF, Andreas, DUMHE, Reiner: UML-based performance engineering possibilities and techniques, IEEE software, januar 2002 19(1), strani 74-83.

[5] ERIKSON, Hans-Erik, PENKER, Magnus: Business Modeling with UML, Wiley&Sons, New York, 2000, 449 strani.

[6] KOŠIR, Aleš: Projektno vodenje razvoja programske opreme, Kakovost, oktober 2005 (3), strani 23-28.

[7] KERZNER, Harold: Advanced Project Management - Best Practicies on Implementation, John Willey and Sons, Haboken, 2004, strani 864

[8] KRISPER, Marjan, RUPNIK, Rok: Enotna metodologija razvoja informacijskih sistemov – Objektni razvoj IS, Center Vlade RS za informatiko, Ljubljana, 2000, 299 strani.

[9] MARTIN, James: Rapid application development, Mc Millan Publishing Comp., New York, 1991, 788 strani.

[10] MIHELČIČ Miran: Poslovne funkcije, Fakulteta za računalništvo in informatiko, Ljubljana, 2004, 363 strani.

[11] MIHELČIČ Miran: Temelji organizacijske teorije, Fakulteta za organizacijske vede Kranj, Kranj, 1993, 382 strani.

[12] PENCA, Marija: Uporaba programskega paketa Excel in ekspertnega sistema DEX v procesu izbire kadrov v javni upravi, Geodetski vestnik, junij 2002 (1&2), strani 106-118.

[13] RAFFAI, Maria: Managing software development, Senet Project Management review, marec 2006, strani 19-28.

[14] RUMBAUGH, James: Object Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, 1991, 500 strani.

[15] RUMBAUGH, James, JACOBSON, Ivar, BOOCH, Grady: The Unified Modeling Language Reference Manual, Addison-Wesley, Reading, 1999, 550 str.

[16] SOLINA, Franc: Projektno vodenje razvoja programske opreme, Fakulteta za računalništvo in informatiko, Ljubljana, 1997, 212 strani.

[17] STEMBERGER, Mark: Vrednotenje projektov, Projektna mreža Slovenije, december 2004 (4), strani 12-20

[18] ŠALI, Borut: Osnove psihologije, Dopisna delavska univerza, Ljubljana, 1973, 227 strani.

[19] TURK Ivan, KAVČIČ Slavka, KOKOTEC-NOVAK, Majda: Poslovodno računovodstvo, Zveza računovodij, finančnikov in revizorjev Slovenije, Ljubljana, 2003, 856 strani.

**OSTALA LITERATURA:**

[20] Akt o sistemizaciji delovnih mest, RRC računalniške storitve d.d., velja od 27.6.2005

[21] Opis podatkovne baze ŠKIS, 2006

[22] Poslovnik kakovosti podjetja RRC Računalniške storitve d.d, velja od 21.10.2005

[23] Pravilnik o osnovah in merilih za delitev sredstev za plače, RRC računalniške storitve d.d. , velja od 15.7.2005

[24] RAJKOVIČ, Vladislav: Teorija odločanja II, prosojnice predavanj, 2002.

[25] Spletna stran podjetja Hill international (www.hill-int.si), 2006

[26] Spletna stran podjetja Kragelj & Kragelj (www.kadrovanje.com), 2006

[27] Spletna stran podjetja RRC Računalniške storitve d.d (www.rrc.si), 2006

[28] Unified Modeling Language Specification, marec 2003, verzija 1.5, (http://www.omg.org/docs/formal/03-03-01.pdf)

[29] Uporabniške zahteve aplikacije KPIS, 2006

# IZJAVA

Izjavljam, da sem magistrsko delo izdelal samostojno pod vodstvom mentorja prof. dr. Franca Soline. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.